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] Damien Vergnaud[1]

[1]École normale supérieure

[2]Université du Luxembourg

ACNS 2010

Outline

Outline

RSA Signatures

- Encrypting with textbook RSA:

$$c = m^e \bmod N$$

  is a bad idea (e.g. homomorphic properties, deterministic encryption).

- Therefore, encapsulate m using a padding scheme μ:

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

RSA Signatures

- Encrypting with textbook RSA:

$$c = m^e \bmod N$$

  is a bad idea (e.g. homomorphic properties, deterministic encryption).
- Therefore, encapsulate m using a padding scheme μ:

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

Padding schemes

- Two kinds of padding schemes:
  1. Ad-hoc paddings, e.g. PKCS#1 v1.5. Designed to prevent specific attacks. Often exhibit other weaknesses.
  2. Provably secure paddings, e.g. OAEP. Proven to be secure under well-defined assumptions.
- Although potentially less secure, ad-hoc paddings remain in widespread use in real-world applications. Re-evaluating them periodically is thus necessary.

Padding schemes

- Two kinds of padding schemes:
  1. Ad-hoc paddings, e.g. PKCS#1 v1.5. Designed to prevent specific attacks. Often exhibit other weaknesses.
  2. Provably secure paddings, e.g. OAEP. Proven to be secure under well-defined assumptions.
- Although potentially less secure, ad-hoc paddings remain in widespread use in real-world applications. Re-evaluating them periodically is thus necessary.

Padding schemes

- Two kinds of padding schemes:
  1. Ad-hoc paddings, e.g. PKCS#1 v1.5. Designed to prevent specific attacks. Often exhibit other weaknesses.
  2. Provably secure paddings, e.g. OAEP. Proven to be secure under well-defined assumptions.
- Although potentially less secure, ad-hoc paddings remain in widespread use in real-world applications. Re-evaluating them periodically is thus necessary.

Padding schemes

- Two kinds of padding schemes:
  1. Ad-hoc paddings, e.g. PKCS#1 v1.5. Designed to prevent specific attacks. Often exhibit other weaknesses.
  2. Provably secure paddings, e.g. OAEP. Proven to be secure under well-defined assumptions.
- Although potentially less secure, ad-hoc paddings remain in widespread use in real-world applications. Re-evaluating them periodically is thus necessary.

Outline

PKCS#1 v1.5

- The PKCS#1 v1.5 standard defines an ad-hoc padding scheme with a randomizer of 64 bits or more.
- Let k be the size of N in bytes. The padding has the following form:

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

with 2 leading fixed bytes, a string r of $k - |m| - 3$ non-zero random nonzero bytes, and a zero byte indicating the end of the randomizer, followed by message.

- Thus, the byte size $|m|$ of m must be at most $k - 11$.

PKCS#1 v1.5

- The PKCS#1 v1.5 standard defines an ad-hoc padding scheme with a randomizer of 64 bits or more.
- Let k be the size of N in bytes. The padding has the following form:

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

with 2 leading fixed bytes, a string r of $k - |m| - 3 \geq 8$ random nonzero bytes, and a zero byte indicating the end of the randomizer, followed by m itself.

- Thus, the byte size $|m|$ of m must be at most $k - 11$.

PKCS#1 v1.5

- The PKCS#1 v1.5 standard defines an ad-hoc padding scheme with a randomizer of 64 bits or more.
- Let k be the size of N in bytes. The padding has the following form:

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

with 2 leading fixed bytes, a string r of $k - |m| - 3 \geq 8$ random nonzero bytes, and a zero byte indicating the end of the randomizer, followed by m itself.

- Thus, the byte size $|m|$ of m must be at most $k - 11$.

PKCS#1 v1.5

- The PKCS#1 v1.5 standard defines an ad-hoc padding scheme with a randomizer of 64 bits or more.
- Let k be the size of N in bytes. The padding has the following form:

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

with 2 leading fixed bytes, a string r of $k - |m| - 3 \geq 8$ random nonzero bytes, and a zero byte indicating the end of the randomizer, followed by m itself.

- Thus, the byte size $|m|$ of m must be at most $k - 11$.

PKCS#1 v1.5

- The PKCS#1 v1.5 standard defines an ad-hoc padding scheme with a randomizer of 64 bits or more.
- Let k be the size of N in bytes. The padding has the following form:

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

with 2 leading fixed bytes, a string r of $k - |m| - 3 \geq 8$ random nonzero bytes, and a zero byte indicating the end of the randomizer, followed by m itself.

- Thus, the byte size $|m|$ of m must be at most $k - 11$.

PKCS#1 v1.5

- The PKCS#1 v1.5 standard defines an ad-hoc padding scheme with a randomizer of 64 bits or more.
- Let k be the size of N in bytes. The padding has the following form:

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

  with 2 leading fixed bytes, a string r of $k - |m| - 3 \geq 8$ random nonzero bytes, and a zero byte indicating the end of the randomizer, followed by m itself.
- Thus, the byte size $|m|$ of m must be at most $k - 11$.

Previous work

- In 1998, Bleichenbacher proposed an attack on PKCS#1 v1.5, recovering a plaintext with around 2^{21} queries to a validity-checking oracle: PKCS#1 v1.5 is not ℓ-OW-VCA-secure for large ℓ.

  Since many SSL implementations at the time behaved as validity-checking oracles, SSL session keys could be recovered by active adversaries in practice.

  Since then, this bug has been patched, but PKCS#1 v1.5 is still the default encryption algorithm for SSL/TLS.

  - In 2006, Coron, Naccache, Joye and Paillier improved on Bleichenbacher's attack, paving the way for chosen-plaintext attacks on PKCS#1 v1.5, implying in particular that PKCS#1 v1.5 is not IND-CPA-secure for small e or large ℓ.

Previous work

- In 1998, Bleichenbacher proposed an attack on PKCS#1 v1.5, recovering a plaintext with around 2^{21} queries to a validity-checking oracle: PKCS#1 v1.5 is not ℓ-OW-VCA-secure for large ℓ.

  Since many SSL implementations at the time behaved as validity-checking oracles, SSL session keys could be recovered by active adversaries in practice.

  Since then, this bug has been patched, but PKCS#1 v1.5 is still the default encryption algorithm for SSL/TLS.

- In 2000, Coron, Naccache, Joye and Paillier introduced chosen-plaintext attacks on PKCS#1 v1.5, implying in particular that PKCS#1 v1.5 is not IND-CPA-secure for small e or large $|m|$.

Previous work

- In 1998, Bleichenbacher proposed an attack on PKCS#1 v1.5, recovering a plaintext with around 2^{21} queries to a validity-checking oracle: PKCS#1 v1.5 is not ℓ-OW-VCA-secure for large ℓ.

  Since many SSL implementations at the time behaved as validity-checking oracles, SSL session keys could be recovered by active adversaries in practice.

  Since then, this bug has been patched, but PKCS#1 v1.5 is still the default encryption algorithm for SSL/TLS.

- In 2000, Coron, Naccache, Joye and Paillier introduced chosen-plaintext attacks on PKCS#1 v1.5, implying in particular that PKCS#1 v1.5 is not IND-CPA-secure for small e or large $|m|$.

Previous work

- In 1998, Bleichenbacher proposed an attack on PKCS#1 v1.5, recovering a plaintext with around 2^{21} queries to a validity-checking oracle: PKCS#1 v1.5 is not ℓ-OW-VCA-secure for large ℓ.

  Since many SSL implementations at the time behaved as validity-checking oracles, SSL session keys could be recovered by active adversaries in practice.

  Since then, this bug has been patched, but PKCS#1 v1.5 is still the default encryption algorithm for SSL/TLS.
- In 2000, Coron, Naccache, Joye and Paillier introduced chosen-plaintext attacks on PKCS#1 v1.5, implying in particular that PKCS#1 v1.5 is not IND-CPA-secure for small e or large $|m|$.

Our contribution

Our contribution to the study of the security of PKCS#1 v1.5 encryption is twofold:

1. Single-user setting (theoretical results). PKCS#1 v1.5 is:
   - not 1-IND-VCA-secure
   - not NM-CPA-secure
   - not 2-OW-CCA-secure for large $|m|$
   - OW-CPA-secure for large $|m|$ is RSA is hard (loose reduction).
2. Multi-user setting (more concrete results).
   - Coppersmith-based plaintext recovery attack on broadcast PKCS#1 v1.5 encryption:
   - for 1024-bit modulus, recovers a 128-bit message encrypted for 2^4 recipients (heuristic polynomial time).

Our contribution

Our contribution to the study of the security of PKCS#1 v1.5 encryption is twofold:

1. Single-user setting (theoretical results). PKCS#1 v1.5 is:
   - not 1-IND-VCA-secure
   - not NM-CPA-secure
   - not 2-OW-CCA-secure for large $|m|$
   - OW-CPA-secure for large $|m|$ is RSA is hard (loose reduction).
2. Multi-user setting (more concrete results).
   - Coppersmith-based plaintext recovery attack on broadcast PKCS#1 v1.5 encryption:
   - for 1024-bit modulus, recovers a 128-bit message encrypted for 2^24 recipients (heuristic polynomial time).

Our contribution

Our contribution to the study of the security of PKCS#1 v1.5 encryption is twofold:

1. Single-user setting (theoretical results). PKCS#1 v1.5 is:
   - not 1-IND-VCA-secure
   - not NM-CPA-secure
   - not 2-OW-CCA-secure for large $|m|$
   - OW-CPA-secure for large $|m|$ is RSA is hard (loose reduction).
2. Multi-user setting (more concrete results).
   - Coppersmith-based plaintext recovery attack on broadcast PKCS#1 v1.5 encryption:
   - for 1024-bit modulus, recovers a 136-bit message encrypted for 2^4 recipients (heuristic polynomial time).

Our contribution

Our contribution to the study of the security of PKCS#1 v1.5 encryption is twofold:

1. Single-user setting (theoretical results). PKCS#1 v1.5 is:
   - not 1-IND-VCA-secure
   - not NM-CPA-secure
   - not 2-OW-CCA-secure for large $|m|$
   - OW-CPA-secure for large $|m|$ is RSA is hard (loose reduction).
2. Multi-user setting (more concrete results).
   - Coppersmith-based plaintext recovery attack on broadcast PKCS#1 v1.5 encryption.
   - for 1024-bit modulus, recovers a 916-bit message encrypted for 64 recipients (heuristic polynomial time).

Our contribution

Our contribution to the study of the security of PKCS#1 v1.5 encryption is twofold:

1. Single-user setting (theoretical results). PKCS#1 v1.5 is:
   - not 1-IND-VCA-secure
   - not NM-CPA-secure
   - not 2-OW-CCA-secure for large $|m|$
   - OW-CPA-secure for large $|m|$ is RSA is hard (loose reduction).
2. Multi-user setting (more concrete results).
   - Coppersmith-based plaintext-recovery attack on broadcast PKCS#1 v1.5 encryption
   - for 1024-bit modulus, recovers an 128-bit message encrypted for 2^4 recipients (heuristic polynomial time).

Our contribution

Our contribution to the study of the security of PKCS#1 v1.5 encryption is twofold:

1. Single-user setting (theoretical results). PKCS#1 v1.5 is:
   - not 1-IND-VCA-secure
   - not NM-CPA-secure
   - not 2-OW-CCA-secure for large $|m|$
   - OW-CPA-secure for large $|m|$ is RSA is hard (loose reduction).
2. Multi-user setting (more concrete results).
   - Coppersmith-based plaintext-recovery attack on broadcast PKCS#1 v1.5 encryption
   - for 1024-bit moduli, recovers a 936-bit message encrypted for ≥ 4 recipients (heuristic polynomial time).

Our contribution

Our contribution to the study of the security of PKCS#1 v1.5 encryption is twofold:

1. Single-user setting (theoretical results). PKCS#1 v1.5 is:
   - not 1-IND-VCA-secure
   - not NM-CPA-secure
   - not 2-OW-CCA-secure for large $|m|$
   - OW-CPA-secure for large $|m|$ is RSA is hard (loose reduction).
2. Multi-user setting (more concrete results).
   - Coppersmith-based plaintext-recovery attack on broadcast PKCS#1 v1.5 encryption
   - for 1024-bit moduli, recovers a 936-bit message encrypted for ≥ 4 recipients (heuristic polynomial time).

Our contribution

Our contribution to the study of the security of PKCS#1 v1.5 encryption is twofold:

1. Single-user setting (theoretical results). PKCS#1 v1.5 is:
   - not 1-IND-VCA-secure
   - not NM-CPA-secure
   - not 2-OW-CCA-secure for large $|m|$
   - OW-CPA-secure for large $|m|$ is RSA is hard (loose reduction).
2. Multi-user setting (more concrete results).
   - Coppersmith-based plaintext-recovery attack on broadcast PKCS#1 v1.5 encryption
   - for 1024-bit moduli, recovers a 936-bit message encrypted for ≥ 4 recipients (heuristic polynomial time).

Outline

The main idea

Our analysis in the single-user setting is mostly based on the following observation.

- Consider a message m with $Z > 2$ trailing zero bits.

  $$\mu(m, r) = 0002_{16} \| r \| 00_{16} \| \cdots \| 00_2$$

  Easy to write down $\mu(m, r) \cdot 2^{-Z}$.
  With good probability, $\mu(m, r) \cdot (1 - 2^{-Z})$ is still a valid padding of some m'.
- Conversely, if the last Z bits of m are not all zero, $\mu(m, r) \cdot 2^{-Z}$ is more or less "random".
  With overwhelming probability, $\mu(m, r) \cdot (1 - 2^{-Z})$ is not a valid padding.

Thus, if c is a ciphertext corresponding to m, $c' = c \cdot (1 - 2^{-Z})^e \bmod N$ is a valid ciphertext roughly when m ends in Z zeroes.

The main idea

Our analysis in the single-user setting is mostly based on the following observation.

- Consider a message m with $Z > 2$ trailing zero bits.

$$\mu(m, r) = 0002_{16} \| r \| 00_{16} \| \cdots \| 00_2$$

  Easy to write down $\mu(m, r) \cdot 2^{-Z}$.
  With good probability, $\mu(m, r) \cdot (1 - 2^{-Z})$ is still a valid padding of some m'.
- Conversely, if the last Z bits of m are not all zero, $\mu(m, r) \cdot 2^{-Z}$ is more or less "random".
  With overwhelming probability, $\mu(m, r) \cdot (1 - 2^{-Z})$ is not a valid padding.

Thus, if c is a ciphertext corresponding to m, $c' = c \cdot (1 - 2^{-Z})^e \bmod N$ is a valid ciphertext roughly when m ends in Z zeroes.

Outline

Indistinguishability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{m_0, m_1}$	chooses m_0, m_1
$b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$	$\xrightarrow{c}$	
	$\xleftarrow{b'}$	guesses b'

Encryption is IND-ATK-secure if for all attackers in attack model ATK, $2\Pr[b = b'] - 1$ is negligible.

In our case, ATK = VCA = access to a validity-checking oracle.

Indistinguishability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{m_0, m_1}$	chooses m_0, m_1
$b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$	$\xrightarrow{c}$	
	$\xleftarrow{b'}$	guesses b'

Encryption is IND-ATK-secure if for all attackers in attack model ATK, $2\Pr[b = b'] - 1$ is negligible.

In our case, ATK = VCA = access to a validity-checking oracle.

Indistinguishability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{m_0, m_1}$	chooses m_0, m_1
$b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$	$\xrightarrow{c}$	
	$\xleftarrow{b'}$	guesses b'

Encryption is IND-ATK-secure if for all attackers in attack model ATK, $2\Pr[b = b'] - 1$ is negligible.

In our case, ATK = VCA = access to a validity-checking oracle.

Indistinguishability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{m_0, m_1}$	chooses m_0, m_1
$b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$	$\xrightarrow{c}$	
	$\xleftarrow{b'}$	guesses b'

Encryption is IND-ATK-secure if for all attackers in attack model ATK, $2\Pr[b = b'] - 1$ is negligible.

In our case, ATK = VCA = access to a validity-checking oracle.

Indistinguishability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{m_0, m_1}$	chooses m_0, m_1
$b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$	$\xrightarrow{c}$	
	$\xleftarrow{b'}$	guesses b'

Encryption is IND-ATK-secure if for all attackers in attack model ATK, $2\Pr[b = b'] - 1$ is negligible.

In our case, ATK = VCA = access to a validity-checking oracle.

Indistinguishability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{m_0, m_1}$	chooses m_0, m_1
$b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$	$\xrightarrow{c}$	
	$\xleftarrow{b'}$	guesses b'

Encryption is IND-ATK-secure if for all attackers in attack model ATK, $2\Pr[b = b'] - 1$ is negligible.

In our case, ATK = VCA = access to a validity-checking oracle.

Indistinguishability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{m_0, m_1}$	chooses m_0, m_1
$b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$	$\xrightarrow{c}$	
	$\xleftarrow{b'}$	guesses b'

Encryption is IND-ATK-secure if for all attackers in attack model ATK, $2\Pr[b = b'] - 1$ is negligible.

In our case, ATK = VCA = access to a validity-checking oracle.

Breaking IND-VCA

Our attacker, for any fixed message length of at least one byte is defined as follows:

1. Choose $m_0 = 00 \cdots 00_{16}$, and m_1 any message with a nonzero trailing nibble.
2. Upon receiving the challenge ciphertext c, compute $c' = c \cdot (1 - 2^{-4})^e$, and query the oracle on c'. If c' is valid, set $b' = 0$. Otherwise, set $b' = 1$.

This breaks IND-VCA with a single query, because:

- by a counting argument, if $b = 1$, c' can never be valid;
- by carefully writing the substraction, if $b = 0$, c' is valid with probability > 0.47.

Breaking IND-VCA

Our attacker, for any fixed message length of at least one byte is defined as follows:

1. Choose $m_0 = 00 \cdots 00_{16}$, and m_1 any message with a nonzero trailing nibble.
2. Upon receiving the challenge ciphertext c, compute $c' = c \cdot (1 - 2^{-4})^e$, and query the oracle on c'. If c' is valid, set $b' = 0$. Otherwise, set $b' = 1$.

This breaks IND-VCA with a single query, because:

- by a counting argument, if $b = 1$, c' can never be valid;
- by carefully writing the substraction, if $b = 0$, c' is valid with probability > 0.47.

Breaking IND-VCA

Our attacker, for any fixed message length of at least one byte is defined as follows:

1. Choose $m_0 = 00 \cdots 00_{16}$, and m_1 any message with a nonzero trailing nibble.
2. Upon receiving the challenge ciphertext c, compute $c' = c \cdot (1 - 2^{-4})^e$, and query the oracle on c'. If c' is valid, set $b' = 0$. Otherwise, set $b' = 1$.

This breaks IND-VCA with a single query, because:

- by a counting argument, if $b = 1$, c' can never be valid;
- by carefully writing the substraction, if $b = 0$, c' is valid with probability > 0.47.

Breaking IND-VCA

Our attacker, for any fixed message length of at least one byte is defined as follows:

1. Choose $m_0 = 00 \cdots 00_{16}$, and m_1 any message with a nonzero trailing nibble.
2. Upon receiving the challenge ciphertext c, compute $c' = c \cdot (1 - 2^{-4})^e$, and query the oracle on c'. If c' is valid, set $b' = 0$. Otherwise, set $b' = 1$.

This breaks IND-VCA with a single query, because:

- by a counting argument, if $b = 1$, c' can never be valid;
- by carefully writing the substraction, if $b = 0$, c' is valid with probability > 0.47.

Outline

Non-malleability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{\mathcal{R}, \mathcal{M}}$	relation $\mathcal{R}$, sampling $\mathcal{M}$
$m \leftarrow \mathcal{M};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$	$\xrightarrow{c}$	
	$\xleftarrow{c'}$	chooses $c' \neq c$

Encryption is NM-ATK-secure if for all attackers in attack model ATK, $\Pr[\mathcal{R}(m, m')] - \Pr[m_0 \leftarrow \mathcal{M}; \mathcal{R}(m_0, m')]$ is negligible.

In our case, ATK = CPA = no oracle.

Non-malleability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{\mathcal{R}, \mathcal{M}}$	relation $\mathcal{R}$, sampling $\mathcal{M}$
$m \leftarrow \mathcal{M};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$	$\xrightarrow{c}$	
	$\xleftarrow{c'}$	chooses $c' \neq c$

Encryption is NM-ATK-secure if for all attackers in attack model ATK, $\Pr[\mathcal{R}(m, m')] - \Pr[m_0 \leftarrow \mathcal{M}; \mathcal{R}(m_0, m')]$ is negligible.

In our case, ATK = CPA = no oracle.

Non-malleability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{\mathcal{R}, \mathcal{M}}$	relation $\mathcal{R}$, sampling $\mathcal{M}$
$m \leftarrow \mathcal{M};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$	$\xrightarrow{c}$	
	$\xleftarrow{c'}$	chooses $c' \neq c$

Encryption is NM-ATK-secure if for all attackers in attack model ATK, $\Pr[\mathcal{R}(m, m')] - \Pr[m_0 \leftarrow \mathcal{M}; \mathcal{R}(m_0, m')]$ is negligible.

In our case, ATK = CPA = no oracle.

Non-malleability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{\mathcal{R},\mathcal{M}}$	relation $\mathcal{R}$, sampling $\mathcal{M}$
$m \leftarrow \mathcal{M};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$	$\xrightarrow{c}$	
	$\xleftarrow{c'}$	chooses $c' \neq c$

Encryption is NM-ATK-secure if for all attackers in attack model ATK, $\Pr[\mathcal{R}(m, m')] - \Pr[m_0 \leftarrow \mathcal{M}; \mathcal{R}(m_0, m')]$ is negligible.

In our case, ATK = CPA = no oracle.

Non-malleability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{\mathcal{R},\mathcal{M}}$	relation $\mathcal{R}$, sampling $\mathcal{M}$
$m \leftarrow \mathcal{M};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$	$\xrightarrow{c}$	
	$\xleftarrow{c'}$	chooses $c' \neq c$

Encryption is NM-ATK-secure if for all attackers in attack model ATK, $\Pr[\mathcal{R}(m, m')] - \Pr[m_0 \leftarrow \mathcal{M}; \mathcal{R}(m_0, m')]$ is negligible.

In our case, ATK = CPA = no oracle.

Non-malleability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{\mathcal{R},\mathcal{M}}$	relation $\mathcal{R}$, sampling $\mathcal{M}$
$m \leftarrow \mathcal{M};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$	$\xrightarrow{c}$	
	$\xleftarrow{c'}$	chooses $c' \neq c$

Encryption is NM-ATK-secure if for all attackers in attack model ATK, $\Pr[\mathcal{R}(m, m')] - \Pr[m_0 \leftarrow \mathcal{M}; \mathcal{R}(m_0, m')]$ is negligible.

In our case, ATK = CPA = no oracle.

Non-malleability

Challenger		Attacker
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{\mathcal{R},\mathcal{M}}$	relation $\mathcal{R}$, sampling $\mathcal{M}$
$m \leftarrow \mathcal{M};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$	$\xrightarrow{c}$	
	$\xleftarrow{c'}$	chooses $c' \neq c$

Encryption is NM-ATK-secure if for all attackers in attack model ATK, $\Pr[\mathcal{R}(m, m')] - \Pr[m_0 \leftarrow \mathcal{M}; \mathcal{R}(m_0, m')]$ is negligible.

In our case, ATK = CPA = no oracle.

Breaking NM-CPA (I)

Our attacker, for any fixed message length of greater than half the byte-size of the modulus, is defined as follows:

1. $\mathcal{M}$ is the uniform distribution on messages of the form:

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

   $\mathcal{R}(m_1, m_2)$ holds iff the first M bits of m_1, m_2 coincide.

2. Upon receiving the challenge ciphertext c, compute $c' = c \cdot (1 - 2^{-Z})^e$.

Breaking NM-CPA (I)

Our attacker, for any fixed message length of greater than half the byte-size of the modulus, is defined as follows:

1. $\mathcal{M}$ is the uniform distribution on messages of the form:

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

   $\mathcal{R}(m_1, m_2)$ holds iff the first M bits of m_1, m_2 coincide.
2. Upon receiving the challenge ciphertext c, compute $c' = c \cdot (1 - 2^{-Z})^e$.

Breaking NM-CPA (II)

This adversary breaks NM-CPA, because:

- c' is always a valid ciphertext, associated with a message $m' \neq m$ with $\mathcal{R}(m, m')$: $\Pr[\mathcal{R}(m, m')] = 1$. Indeed, $\mu(m) \cdot (1 - 2^{-Z})$ is:

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

- clearly, on average for $m_0 \leftarrow \mathcal{M}$, $\Pr[\mathcal{R}(m_0, m')] = 2^{-M} \leq 1/2$.

In practice, the advantage is overwhelmingly close to 1.

Breaking NM-CPA (II)

This adversary breaks NM-CPA, because:

- c' is always a valid ciphertext, associated with a message $m' \neq m$ with $\mathcal{R}(m, m')$: $\Pr[\mathcal{R}(m, m')] = 1$. Indeed, $\mu(m) \cdot (1 - 2^{-Z})$ is:

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

- clearly, on average for $m_0 \leftarrow \mathcal{M}$, $\Pr[\mathcal{R}(m_0, m')] = 2^{-M} \leq 1/2$.

In practice, the advantage is overwhelmingly close to 1.

Outline

Investigating one-wayness

When $|m|$ is large, say $|m| = k - 11$, one can reduce the OW-CPA-security of PKCS#1 v1.5 encryption to the RSA problem.

Idea: suppose we are given an RSA challenge $y = x^e \bmod N$.

- Set $c = y \cdot r^e$ for a random r. With probability $(255/256)^8 \cdot 2^{-24}$, c is a valid PKCS#1 v1.5 ciphertext and we can decrypt.
- However, decryption doesn't reveal the randomizer.
- But the randomizer can be recovered using the method of [CJNP00] if we have *two* valid ciphertexts c_1, c_2. This is enough to compute the e-th root of y and solve RSA.

Hence, for large $|m|$, PKCS#1 v1.5 is OW-CPA-secure if the RSA problem is hard (loose reduction, though).

Investigating one-wayness

When $|m|$ is large, say $|m| = k - 11$, one can reduce the OW-CPA-security of PKCS#1 v1.5 encryption to the RSA problem.

Idea: suppose we are given an RSA challenge $y = x^e \bmod N$.

- Set $c = y \cdot r^e$ for a random r. With probability $(255/256)^8 \cdot 2^{-24}$, c is a valid PKCS#1 v1.5 ciphertext and we can decrypt.
- However, decryption doesn't reveal the randomizer.
- But the randomizer can be recovered using the method of [CJNP00] if we have *two* valid ciphertexts c_1, c_2. This is enough to compute the e-th root of y and solve RSA.

Hence, for large $|m|$, PKCS#1 v1.5 is OW-CPA-secure if the RSA problem is hard (loose reduction, though).

Investigating one-wayness

When $|m|$ is large, say $|m| = k - 11$, one can reduce the OW-CPA-security of PKCS#1 v1.5 encryption to the RSA problem.

Idea: suppose we are given an RSA challenge $y = x^e \bmod N$.

- Set $c = y \cdot r^e$ for a random r. With probability $(255/256)^8 \cdot 2^{-24}$, c is a valid PKCS#1 v1.5 ciphertext and we can decrypt.
- However, decryption doesn't reveal the randomizer.
- But the randomizer can be recovered using the method of [CJNP00] if we have *two* valid ciphertexts c_1, c_2. This is enough to compute the e-th root of y and solve RSA.

Hence, for large $|m|$, PKCS#1 v1.5 is OW-CPA-secure if the RSA problem is hard (loose reduction, though).

Outline

Broadcast RSA

Broadcast RSA encryption: multi-user protocol in which a sender encrypts the same message m to multiple recipients, each with its own RSA public key.

Public moduli N_i differ, but the public exponent e is usually shared.

Broadcast RSA has specific vulnerabilities:

- When textbook RSA is used, an attacker can apply the CRT to the $c_i = m^e \bmod N_i$ and deduce $c = m^e \bmod N_1 \cdots N_\ell$. If $\ell \geq e$, $c = m^e$ in $\mathbb{Z}$ and taking e-th roots recovers m.
- More generally, Håstad proved in 1988 that if public constant paddings are used:

$$c_i = (\omega_i + m)^e \bmod N_i \qquad (\omega_i \text{ public})$$

m can still be recovered when $\ell > e$.

Broadcast RSA

Broadcast RSA encryption: multi-user protocol in which a sender encrypts the same message m to multiple recipients, each with its own RSA public key.

Public moduli N_i differ, but the public exponent e is usually shared.

Broadcast RSA has specific vulnerabilities:

- When textbook RSA is used, an attacker can apply the CRT to the $c_i = m^e \bmod N_i$ and deduce $c = m^e \bmod N_1 \cdots N_\ell$. If $\ell \geq e$, $c = m^e$ in $\mathbb{Z}$ and taking e-th roots recovers m.
- More generally, Håstad proved in 1988 that if public constant paddings are used:

$$c_i = (\omega_i + m)^e \bmod N_i \qquad (\omega_i \text{ public})$$

m can still be recovered when $\ell > e$.

Broadcast RSA

Broadcast RSA encryption: multi-user protocol in which a sender encrypts the same message m to multiple recipients, each with its own RSA public key.

Public moduli N_i differ, but the public exponent e is usually shared.

Broadcast RSA has specific vulnerabilities:

- When textbook RSA is used, an attacker can apply the CRT to the $c_i = m^e \bmod N_i$ and deduce $c = m^e \bmod N_1 \cdots N_\ell$. If $\ell \geq e$, $c = m^e$ in $\mathbb{Z}$ and taking e-th roots recovers m.
- More generally, Håstad proved in 1988 that if public constant paddings are used:

$$c_i = (\omega_i + m)^e \bmod N_i \qquad (\omega_i \text{ public})$$

m can still be recovered when $\ell > e$.

Broadcast PKCS#1 v1.5

In broadcast PKCS#1 v1.5 encryption, the ciphertexts are of the form:

$$c_i = \mu(m, r_i)^e = (m + Ar_i + B)^e \mod N_i$$

with $A = 2^{8|m|+8}$ and $B = 2^{8k-7}$.

The randomizers r_i are not public, so Håstad's attack, or even its later generalizations, do not apply. Indeed, random padding was the prescribed countermeasure to Håstad's attack.

Broadcast PKCS#1 v1.5

In broadcast PKCS#1 v1.5 encryption, the ciphertexts are of the form:

$$c_i = \mu(m, r_i)^e = (m + Ar_i + B)^e \mod N_i$$

with $A = 2^{8|m|+8}$ and $B = 2^{8k-7}$.

The randomizers r_i are not public, so Håstad's attack, or even its later generalizations, do not apply. Indeed, random padding was the prescribed countermeasure to Håstad's attack.

Outline

Attacking broadcast PKCS#1 v1.5

Nevertheless, we can attack as follows. Using CRT, all ℓ equations can be rewritten as a single equation mod $N = N_1 \cdots N_\ell$:

$$\sum u_i c_i = \sum u_i (m + A r_i + B)^e \mod N$$

for explicit constants u_i.

Thus, $(m, r_1, \ldots, r_\ell)$ is a small root mod N of the multivariate polynomial

$$f(x, y_1, \ldots, y_\ell) = \sum u_i \left(c_i - (m + A r_i + B)^e\right)$$

Finding small modular roots of multivariate polynomials can be attempted using heuristic generalizations of Coppersmith's lattice-based techniques for computing small roots.

Attacking broadcast PKCS#1 v1.5

Nevertheless, we can attack as follows. Using CRT, all ℓ equations can be rewritten as a single equation mod $N = N_1 \cdots N_\ell$:

$$\sum u_i c_i = \sum u_i (m + A r_i + B)^e \mod N$$

for explicit constants u_i.

Thus, $(m, r_1, \ldots, r_\ell)$ is a small root mod N of the multivariate polynomial

$$f(x, y_1, \ldots, y_\ell) = \sum u_i \left(c_i - (m + A r_i + B)^e\right)$$

Finding small modular roots of multivariate polynomials can be attempted using heuristic generalizations of Coppersmith's lattice-based techniques for computing small roots.

Attacking broadcast PKCS#1 v1.5

Nevertheless, we can attack as follows. Using CRT, all ℓ equations can be rewritten as a single equation mod $N = N_1 \cdots N_\ell$:

$$\sum u_i c_i = \sum u_i (m + A r_i + B)^e \mod N$$

for explicit constants u_i.

Thus, $(m, r_1, \ldots, r_\ell)$ is a small root mod N of the multivariate polynomial

$$f(x, y_1, \ldots, y_\ell) = \sum u_i \left(c_i - (m + A r_i + B)^e\right)$$

Finding small modular roots of multivariate polynomials can be attempted using heuristic generalizations of Coppersmith's lattice-based techniques for computing small roots.

Theory and practice

To determine when the attack applies, we use the method of Jochemsz and May to construct the relevant lattices, and obtain bounds on the size of parameters to get short enough vectors.

Asymptotically, we find that the attack applies, and runs in heuristic polynomial time in e, k, $|m|$ (but exponential time in ℓ) when:

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

For 1024-bit moduli and 936-bit m, this gives $\ell \geq 4$, so in theory at least, the attack applies to very realistic settings.

However, the lattice sizes involved can be very large (much more than 1000), with big coefficients. Such realistic settings are currently out of computational reach.

Theory and practice

To determine when the attack applies, we use the method of Jochemsz and May to construct the relevant lattices, and obtain bounds on the size of parameters to get short enough vectors.

Asymptotically, we find that the attack applies, and runs in heuristic polynomial time in e, k, $|m|$ (but exponential time in ℓ) when:

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

For 1024-bit moduli and 936-bit m, this gives $\ell \geq 4$, so in theory at least, the attack applies to very realistic settings.

However, the lattice sizes involved can be very large (much more than 1000), with big coefficients. Such realistic settings are currently out of computational reach.

Theory and practice

To determine when the attack applies, we use the method of Jochemsz and May to construct the relevant lattices, and obtain bounds on the size of parameters to get short enough vectors.

Asymptotically, we find that the attack applies, and runs in heuristic polynomial time in e, k, $|m|$ (but exponential time in ℓ) when:

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

For 1024-bit moduli and 936-bit m, this gives $\ell \geq 4$, so in theory at least, the attack applies to very realistic settings.

However, the lattice sizes involved can be very large (much more than 1000), with big coefficients. Such realistic settings are currently out of computational reach.

Theory and practice

To determine when the attack applies, we use the method of Jochemsz and May to construct the relevant lattices, and obtain bounds on the size of parameters to get short enough vectors.

Asymptotically, we find that the attack applies, and runs in heuristic polynomial time in e, k, $|m|$ (but exponential time in ℓ) when:

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

For 1024-bit moduli and 936-bit m, this gives $\ell \geq 4$, so in theory at least, the attack applies to very realistic settings.

However, the lattice sizes involved can be very large (much more than 1000), with big coefficients. Such realistic settings are currently out of computational reach.

Conclusion

Before: PKCS#1 v1.5 was somewhat broken.

UBK-CCA	$\Longleftarrow$	UBK-VCA $\Longleftarrow$ UBK-CPA = Factoring
$\Downarrow$		$\Downarrow$ $\Downarrow$
OW-CCA	$\Longleftarrow$	~~OW-VCA~~ $\Longleftarrow$ OW-CPA
		large ℓ
$\Downarrow$		$\Downarrow$ $\Downarrow$
IND-CCA	$\Longleftarrow$	IND-VCA $\Longleftarrow$ ~~IND-CPA~~
		small e
$\Downarrow$		$\Downarrow$ $\Downarrow$
NM-CCA	$\Longleftarrow$	NM-VCA $\Longleftarrow$ NM-CPA

Conclusion

After: PKCS#1 v1.5 is somewhat more broken.

UBK-CCA	$\Longleftarrow$	UBK-VCA	$\Longleftarrow$	UBK-CPA $=$ Factoring
$\Downarrow$		$\Downarrow$		$\Downarrow$
~~OW-CCA~~	$\Longleftarrow$	~~OW-VCA~~	$\Longleftarrow$	OW-CPA $=$ RSA
$\ell = 2$		large ℓ		~~OW-CPA~~ multi-user
$\Downarrow$		$\Downarrow$		$\Downarrow$
~~IND-CCA~~	$\Longleftarrow$	~~IND-VCA~~	$\Longleftarrow$	~~IND-CPA~~
		$\ell = 1$		small e
$\Downarrow$		$\Downarrow$		$\Downarrow$
~~NM-CCA~~	$\Longleftarrow$	~~NM-VCA~~	$\Longleftarrow$	~~NM-CPA~~
				any e

Bottom line: isn't it about time we used OAEP?

Conclusion

After: PKCS#1 v1.5 is somewhat more broken.

UBK-CCA	$\Longleftarrow$	UBK-VCA $\Longleftarrow$ UBK-CPA $=$ Factoring
$\Downarrow$		$\Downarrow$
~~OW-CCA~~	$\Longleftarrow$	~~OW-VCA~~ $\Longleftarrow$ OW-CPA $=$ RSA
$\ell = 2$		large ℓ ~~OW-CPA~~ multi-user
$\Downarrow$		$\Downarrow$
~~IND-CCA~~	$\Longleftarrow$	~~IND-VCA~~ $\Longleftarrow$ ~~IND-CPA~~
		$\ell = 1$ small e
$\Downarrow$		$\Downarrow$
~~NM-CCA~~	$\Longleftarrow$	~~NM-VCA~~ $\Longleftarrow$ ~~NM-CPA~~
		any e

Bottom line: isn't it about time we used OAEP?

Thank you!