Problems with Schoolbook RSA III

Tanja Lange

Eindhoven University of Technology

2WF80: Introduction to Cryptology

RSA encryption is homomorphic

An encryption system is homomorphic if there exist operations $\circ$ on the ciphertext space and $\triangle$ on the message space so that

$$\mathrm{Enc}_k(m_1) \circ \mathrm{Enc}_k(m_2) = \mathrm{Enc}_k(m_1 \triangle m_2).$$

RSA encryption is homomorphic

An encryption system is homomorphic if there exist operations $\circ$ on the ciphertext space and $\triangle$ on the message space so that

$$\mathrm{Enc}_k(m_1) \circ \mathrm{Enc}_k(m_2) = \mathrm{Enc}_k(m_1 \triangle m_2).$$

For RSA we have

$$c_1 \cdot c_2 \equiv m_1^e \cdot m_2^e \equiv (m_1 \cdot m_2)^e \bmod n,$$

so RSA is homomorphic with $\circ = \triangle$ being multiplication modulo n.

RSA encryption is homomorphic

An encryption system is homomorphic if there exist operations $\circ$ on the ciphertext space and $\triangle$ on the message space so that

$$\mathrm{Enc}_k(m_1) \circ \mathrm{Enc}_k(m_2) = \mathrm{Enc}_k(m_1 \triangle m_2).$$

For RSA we have

$$c_1 \cdot c_2 \equiv m_1^e \cdot m_2^e \equiv (m_1 \cdot m_2)^e \bmod n,$$

so RSA is homomorphic with $\circ = \triangle$ being multiplication modulo n.

Homomorphic properties can be desired, so this is not strictly a problem, but it's important to be aware of them.

RSA signatures are not homomorphic because they use $h(m)$.

Security requirements

Attacker goals

- Recover m from $\text{Enc}_{\text{pk}}(m)$,
  i.e. break one-wayness (OW).

Attacker abilities

- Chosen ciphertext attack (CCA I / II)
  Attacker can ask for decryptions of ciphertexts of his choice.
  For II the attacker can continue asking for decryptions after receiving a challenge ciphertext.

Security requirements

Attacker goals

- Recover m from $\text{Enc}_{\text{pk}}(m)$,
  i.e. break one-wayness (OW).

Attacker abilities

- Chosen ciphertext attack (CCA I / II)
  Attacker can ask for decryptions of ciphertexts of his choice.
  For II the attacker can continue asking for decryptions after receiving a challenge ciphertext.

Homomorphic systems cannot be OW-CCA II secure:

Security requirements

Attacker goals

- Recover m from $\mathrm{Enc}_{\mathrm{pk}}(m)$,
  i.e. break one-wayness (OW).

Attacker abilities

- Chosen ciphertext attack (CCA I / II)
  Attacker can ask for decryptions of ciphertexts of his choice.
  For II the attacker can continue asking for decryptions after receiving a challenge ciphertext.

Homomorphic systems cannot be OW-CCA II secure:
Pick random message r compute $c_r = \mathrm{Enc}_{\mathrm{pk}}(r)$ and submit

$$c \neq c' = c_r \circ c = \mathrm{Enc}_{\mathrm{pk}}(r) \circ \mathrm{Enc}_{\mathrm{pk}}(m) = \mathrm{Enc}_{\mathrm{pk}}(r \triangle m)$$

for decryption.

Security requirements

Attacker goals

- Recover m from $\mathrm{Enc}_{\mathrm{pk}}(m)$,
  i.e. break one-wayness (OW).

Attacker abilities

- Chosen ciphertext attack (CCA I / II)
  Attacker can ask for decryptions of ciphertexts of his choice.
  For II the attacker can continue asking for decryptions after receiving a challenge ciphertext.

Homomorphic systems cannot be OW-CCA II secure:
Pick random message r compute $c_r = \mathrm{Enc}_{\mathrm{pk}}(r)$ and submit

$$c \neq c' = c_r \circ c = \mathrm{Enc}_{\mathrm{pk}}(r) \circ \mathrm{Enc}_{\mathrm{pk}}(m) = \mathrm{Enc}_{\mathrm{pk}}(r \triangle m)$$

for decryption. From $r \triangle m$ recover m.

The fine print: This requires $\triangle$ to be an operation so that m can be recovered from $r \triangle m$ and r. Note that the attacker has no restrictions in choosing r other than $c' \neq c$.

What if Sign(m) $\equiv m^d \bmod n$?

Attacker goals

- Produce forgeries on any message m.
  i.e., break universal unforgeability (UU).
- Create some forgery (no control over the message),
  i.e., break existential unforgeability (EU).

What if $\text{Sign}(m) \equiv m^d \bmod n$?

Attacker goals

- Produce forgeries on any message m.
  i.e., break universal unforgeability (UU).
- Create some forgery (no control over the message),
  i.e., break existential unforgeability (EU).

Attacker abilities

- Known message attack (KMA)
  Attacker knows some $(m, \text{Sign}(m))$ pairs.
- Chosen message attack (CMA)
  Attacker can request signatures $(m, \text{Sign}(m))$
  on messages m of his choice.

Attack on EU-KMA,, given $(m_1, \text{Sign}(m_1))$

What if $\text{Sign}(m) \equiv m^d \bmod n$?

Attacker goals

- Produce forgeries on any message m.
  i.e., break universal unforgeability (UU).
- Create some forgery (no control over the message),
  i.e., break existential unforgeability (EU).

Attacker abilities

- Known message attack (KMA)
  Attacker knows some $(m, \text{Sign}(m))$ pairs.
- Chosen message attack (CMA)
  Attacker can request signatures $(m, \text{Sign}(m))$
  on messages m of his choice.

Attack on EU-KMA,, given $(m_1, \text{Sign}(m_1))$
Compute $m_2 \equiv m_1^2 \bmod n$ and $s_2 \equiv (\text{Sign}(m_1))^2 \bmod n$.
Then (m_2, s_2) is a valid signature on a new message m_2.

What if $\text{Sign}(m) \equiv m^d \bmod n$?

Attacker goals

- Produce forgeries on any message m.
  i.e., break universal unforgeability (UU).
- Create some forgery (no control over the message),
  i.e., break existential unforgeability (EU).

Attacker abilities

- Known message attack (KMA)
  Attacker knows some $(m, \text{Sign}(m))$ pairs.
- Chosen message attack (CMA)
  Attacker can request signatures $(m, \text{Sign}(m))$
  on messages m of his choice.

Attack on EU-KMA,, given $(m_1, \text{Sign}(m_1))$
Compute $m_2 \equiv m_1^2 \bmod n$ and $s_2 \equiv (\text{Sign}(m_1))^2 \bmod n$.
Then (m_2, s_2) is a valid signature on a new message m_2.

Attack on UU-CMA:
To eventually construct a signature on m, compute $m' \equiv m \cdot 2^e \bmod n$
and request a signature on m'.

What if Sign(m) $\equiv m^d$ mod n?

Attacker goals

- Produce forgeries on any message m.
  i.e., break universal unforgeability (UU).
- Create some forgery (no control over the message),
  i.e., break existential unforgeability (EU).

Attacker abilities

- Known message attack (KMA)
  Attacker knows some $(m, \text{Sign}(m))$ pairs.
- Chosen message attack (CMA)
  Attacker can request signatures $(m, \text{Sign}(m))$
  on messages m of his choice.

Attack on EU-KMA,, given $(m_1, \text{Sign}(m_1))$
Compute $m_2 \equiv m_1^2 \bmod n$ and $s_2 \equiv (\text{Sign}(m_1))^2 \bmod n$.
Then (m_2, s_2) is a valid signature on a new message m_2.

Attack on UU-CMA:
To eventually construct a signature on m, compute $m' \equiv m \cdot 2^e \bmod n$
and request a signature on m'.
Upon receipt of $(m', \text{Sign}(m')) = (m', (m \cdot 2^e)^d) = (m', m^d \cdot 2)$,
present $(m, (\text{Sign}(m')/2 \bmod n))$ as valid signature on m.

Back to RSA encryption

Attacker goals

- Learn any information about plaintext (semantic security).
  Equivalent to breaking Indistinguishability (IND),
  i.e., learning which of two attacker-chosen messages m_0, m_1 was encrypted in $c = \text{Enc}_{\text{pk}}(m_i)$ (beyond 50% chance of guessing.)

Attacker abilities

- Chosen plaintext attack (CPA)
  Attacker gets encryption of plaintexts of his choice.

Schoolbook RSA is not IND-CPA secure:
Attacker chooses two random messages m_0, m_1.
Challenger picks $b \in \{0, 1\}$ at random and sends back $c = \text{Enc}(m_b)$..

Back to RSA encryption

Attacker goals

- Learn any information about plaintext (semantic security).
  Equivalent to breaking Indistinguishability (IND),
  i.e., learning which of two attacker-chosen messages m_0, m_1 was encrypted in $c = \text{Enc}_{\text{pk}}(m_i)$ (beyond 50% chance of guessing.)

Attacker abilities

- Chosen plaintext attack (CPA)
  Attacker gets encryption of plaintexts of his choice.

Schoolbook RSA is not IND-CPA secure:
Attacker chooses two random messages m_0, m_1.
Challenger picks $b \in \{0, 1\}$ at random and sends back $c = \text{Enc}(m_b)$..

Schoolbook RSA is deterministic!
The attacker can just compute $m_0^e \bmod n$ and $m_1^e \bmod n$ and check which one matches c.

Not IND-CPA secure implies not IND-CCA secure.

RSA PKCS#1 v1.5

All the following numbers are written in hexadecimal, i.e. 0 means 0000.

PKCS#1 v1.5 randomizes and pads message m to

$$\text{pad}(m) = 00\,02\,r\,00\,m,$$

where r is a randomly chosen, with the condition that r does not include 00. The length of r is at least 8 bytes and is chosen so that pad(m) has the same length as the modulus n.

RSA PKCS#1 v1.5

All the following numbers are written in hexadecimal, i.e. 0 means 0000.

PKCS#1 v1.5 randomizes and pads message m to

$$\text{pad}(m) = 00\,02\,r\,00\,m,$$

where r is a randomly chosen, with the condition that r does not include 00. The length of r is at least 8 bytes and is chosen so that pad(m) has the same length as the modulus n.

Decoding must check for 00 02 as the start of pad(m), else output failure.
Then find 00 (scanning from the left) and thus m.
If no 00 is found, the decoder outputs failure.

RSA PKCS#1 v1.5

All the following numbers are written in hexadecimal, i.e. 0 means 0000.

PKCS#1 v1.5 randomizes and pads message m to

$$\mathrm{pad}(m) = 00\,02\,r\,00\,m,$$

where r is a randomly chosen, with the condition that r does not include 00. The length of r is at least 8 bytes and is chosen so that $\mathrm{pad}(m)$ has the same length as the modulus n.

Decoding must check for 00 02 as the start of $\mathrm{pad}(m)$, else output failure.
Then find 00 (scanning from the left) and thus m.
If no 00 is found, the decoder outputs failure.

1998 Bleichenbacher noticed that the failure messages can be used for an attack. Let $c \equiv (\mathrm{pad}(m))^e \bmod n$ and $\ell = \lfloor \log_2 n \rfloor + 1$.

Send $s^e \cdot c \bmod n$ for some s.

RSA PKCS#1 v1.5

All the following numbers are written in hexadecimal, i.e. 0 means 0000.

PKCS#1 v1.5 randomizes and pads message m to

$$\text{pad}(m) = 00\,02\,r\,00\,m,$$

where r is a randomly chosen, with the condition that r does not include 00. The length of r is at least 8 bytes and is chosen so that pad(m) has the same length as the modulus n.

Decoding must check for 00 02 as the start of pad(m), else output failure.
Then find 00 (scanning from the left) and thus m.
If no 00 is found, the decoder outputs failure.

1998 Bleichenbacher noticed that the failure messages can be used for an attack. Let $c \equiv (\text{pad}(m))^e \bmod n$ and $\ell = \lfloor \log_2 n \rfloor + 1$.

Send $s^e \cdot c \bmod n$ for some s.
If there is no decoding failure then $s \cdot \text{pad}(m)$ starts with 00 02, i.e.,

$$s \cdot \text{pad}(m) - k \cdot n \in [2 \cdot 2^{\ell-16}, 3 \cdot 2^{\ell-16}].$$

RSA PKCS#1 v1.5

All the following numbers are written in hexadecimal, i.e. 0 means 0000.

PKCS#1 v1.5 randomizes and pads message m to

$$\text{pad}(m) = 00\,02\,r\,00\,m,$$

where r is a randomly chosen, with the condition that r does not include 00. The length of r is at least 8 bytes and is chosen so that $\text{pad}(m)$ has the same length as the modulus n.

Decoding must check for 00 02 as the start of $\text{pad}(m)$, else output failure.
Then find 00 (scanning from the left) and thus m.
If no 00 is found, the decoder outputs failure.

1998 Bleichenbacher noticed that the failure messages can be used for an attack. Let $c \equiv (\text{pad}(m))^e \bmod n$ and $\ell = \lfloor \log_2 n \rfloor + 1$.

Send $s^e \cdot c \bmod n$ for some s. know s so "know" k
If there is no decoding failure then $s \cdot \text{pad}(m)$ starts with 00 02, i.e.,

$$s \cdot \text{pad}(m) - k \cdot n \in [2 \cdot 2^{\ell-16}, 3 \cdot 2^{\ell-16}].$$

Build up many relations and recover m.

Lessons learned

- Must use RSA with randomized padding!
- PKCS#1 v1.5 is a negative example which is broken using Bleichenbacher's attack, see https://robotattack.org/ for a recent attack in practice.
- RSA-OAEP is a better padding scheme.
- The hash function is essential in RSA signatures.