RSA – Part II:

Recall: RSA is an asymmetric cryptsystem.

- public key (e, n), private key d, $n = p \cdot q$ (prime numbers)
- Encryption: $m \in \mathbb{Z}$, $m < n$: $c \equiv m^e \bmod n$
- Decryption: $m' \equiv c^d \bmod n$.

Problems of RSA:

1. Relation between messages

Assume e is small and messages m_1 and m_2 satisfy

$$m_2 = a m_1 + b,$$

where $a, b \in \mathbb{Z}$ are known, but m_1, m_2 are unknown.

Take $e = 3$ and consider

$$c_1 \equiv m_1^3 \bmod n \qquad c_2 \equiv (a m_1 + b)^3 \bmod n$$

Clearly c_2 satisfies

$$\circledast \quad c_2 = \underbrace{a^3 m_1^3}_{= c_1 a^3} + 3a^2 m_1^2 b + 3 a m_1 b^2 + b^3 \quad \triangleq \text{known}$$

We get m_1 from this, by solving the quadratic equation. But these are hard to solve modulo composite numbers $n = p \cdot q$.
Let's try a more intricate approach:

Put $A = b(c_2 + 2a^3 c_1 - b^3) \overset{\circledast}{=} b(a^3 m_1^3 + 3a^2 m_1^2 b + 3 a m_1 b^2 + b^3 + 2 a^3 \underbrace{c_1}_{= m_1^3} - b^3)$

$$= 3ab m_1 (a^2 m_1^2 + a m_1 b + b^2)$$

$$B = a(c_2 - a^3 c_1 + 2b^3) = \ldots = 3ab(a^2 m_1^2 + a m_1 b + b^2)$$

Thus $A/B \equiv m_1 \bmod n$.

If e is larger the expressions for A and B get more complicated but we can still get m_1.

General situation:

Given: $c_1 \equiv m^e \bmod n$ and $c_2 \equiv (f(m))^e \bmod n$

for some polynomial f. Then, $z^e - c_1$ and $(f(z))^e - c_2$ have both the root $z = m$. Hence, both polynomial have $(z-m)$ as a common factor, which we get by computing

$$\gcd\left((z^e - c_1), ((f(z))^e - c_2)\right)$$

over $\mathbb{Z}/n$.

2. Multiplicativity:

Let (e,n) a public RSA key. If two messages are encrypt by (e,n) we have

(i) $c_1 \equiv m_1^e \bmod n$ (ii) $c_2 \equiv m_2^e \bmod n$.

Together we obtain

$$\boxed{\begin{aligned} c_1 \cdot c_2 &\equiv (m_1 \cdot m_2)^e \bmod n \\ &\equiv m_1^e \cdot m_2^e \bmod n \end{aligned}}$$

This property is called homomorphic encryption.

We get $c = c_1 c_2$ the ciphertext of $m = m_1 \cdot m_2$ without knowing the plaintexts

Remark:

Homomorphic encryption is highly desirable, e.g. for computing in the cloud on encrypted data. But then we want to multiply and add, not only multiply.

~~For our security definition~~

Special attack:

Assume the attacker can ask for the decryption of any ciphertext other than the target one. (The target one is called challenge)

Target: decrypt $c \equiv m^e \bmod n$.

Ask for the decription of e.g. $c' = 2^e \cdot c$. This gives

$$m' \equiv (2^e \cdot c)^d \equiv 2^{ed} \cdot c^d \equiv 2 \cdot c^d \equiv 2 \cdot m \bmod n$$

$$\Rightarrow \frac{1}{2} m' \equiv m \bmod n.$$

Remark: Similarly, ask for the signature on $2^e \cdot m'$ to compute a signature on m.

3. Identical messages:

(similar to what we have seen with ECB)

Identical messages lead to identical keys ⇒ dictionary attacs are possible.

Summary:

All attacks (1-3) use that we have algebraic operations on the ciphertexts and that all are valid.

Solution:

Work around these problems by using padding and randomization.

- Randomization ⇒ freshness of the encryption
- Padding: several tweaks are detected because they destroy a fixed padding.
  ⇒ the will be invalid ciphertexts, which will be rejected.

Recall on security notion:

Chosen-plaintext-attack (CPA):

Gets access to an oracle that decrypts ciphertexts which are the result of encryptions.
In the public key setting this does not make much sense because we can encrypt ourselves (big difference to symmetric crypto settings)

Chosen-Ciphertext-attackI (CCAI):

[The challenge for an attacker is a fixed ciphertext c he wants to decrypt.]
The attacker can ask for arbitrary decriptions of ciphertexts before he gets the challenge and not afterwards.

Usual setting:

Attacker chooses 2 messages m_0 and m_1 and challanger gives back the encryption of m_b, for $b \in \{0,1\}$ randomly chosen.

Then the attacker makes a guess g and wins if $g=b$.
If guessing is random $\Rightarrow$ 50% success chance.
An attack strategy is successful $\Leftrightarrow$ wins with probability $>50\%$.

CCAII:
Same as above + attacker can ask for decryption of $c' \neq$ challenge (CCAI) even after receiving the challenge ciphertext.

Remark: Schoolbook RSA does not satisfy CPA, CCAI nor CCAII security.

In case of CCAI one could just encrypt m_0 and m_1 and check which one matches.

How should RSA be used in practice?

Optimal Asymmetric Encryption padding (OAEP):

- Let (e,n), d be RSA-keys
- Set $\ell := \lfloor \log_2 n \rfloor + 1$ and fix integers k_0 and k_1 with $k_0+k_1<\ell$. (Parameters for padding and randomness)
- Restrict messages to at most $\ell-k_0-k_1$ bits. That is
$$m = \sum_{i=0}^{\ell-k_0-k_1-1} m_i \cdot 2^i \longrightarrow m = (m_{\ell-k_0-k_1-1} \dots m_0).$$
- We need two hash functions
$$G: \{0,1\}^{k_0} \longrightarrow \{0,1\}^{\ell-k_0} \quad \text{and} \quad H: \{0,1\}^{\ell-k_0} \longrightarrow \{0,1\}^{k_0}$$

These can be derived from general hash functions and restricting their range.

Preparing m before sending it through RSA:

Encoding: Pick r a random k_0-bit string

$$m' = (m \,\|\, \underbrace{0\,0 \dots 0}_{k_1 \text{ copies}}) \oplus G(r) \qquad \text{length } \ell-k_0$$

$$r' = r \oplus H(m') \qquad \text{length } k_0$$

The encoding of m is

$$M = m' \| r' \qquad \text{length } l$$

Now apply RSA to encrypt M. Note each encryption uses fresh randomness r to avoid dictionary attacks.

Decoding: Split M: $M = m' \| r'$

Get r back: $r = H(m') \oplus r'$

Get m: $(m \| 0 \dots 0) = m' \oplus G(r)$

Check if m is properly formatted; i.e. if rightmost k_1 bits are all zero. If not discard the message, else return m.

Remark:

- Even if two messages are linearly related this relation does not hold for the resulting values. The attacks described before using homomorphic properties usually do not work either.