Cryptography
Week #8:
RSA & Co.

Rogério Reis, ⟨rogerio.reis@fc.up.pt⟩
2023/2024
DCC FCUP

November, 10th 2023

The Public Key Cryptography (PKC) model

- Instead of one key per channel, each agent has two keys.
- A public key k_p.
- A secret (or private) key k_s.
- One needs $D_{k_s}(E_{k_p}(m)) = m$.
- $D_{k_p}(E_{k_p}(m)) \neq m$.
- Even better if $D_{k_p}(E_{k_s}(m)) = m$, but this is not necessary.

To achieve such model we will need some mathematical notions and results...

$$a \mid b \iff \exists k \in \mathbb{Z}, ak = b$$

1. $a \mid 1 \;\Rightarrow\; a = \pm 1$
2. $((a \mid b) \,\wedge (b \mid a)) \;\Rightarrow a = \pm b$
3. $(\forall b)(b \mid 0)$
4. $(b \mid g) \wedge (b \mid h) \Rightarrow ((\forall m, n \in \mathbb{Z})(b \mid (mg + nh)))$
5. $((a \mid (b + c)) \wedge (a \mid b)) \Rightarrow a \mid c$

$$\begin{aligned}
a \mid b &\quad\Rightarrow\quad b = ak \\
a \mid (b + c) &\quad\Rightarrow\quad (b + c) = ak' = (ak + c) \\
ak' = ak + c &\quad\Rightarrow\quad a(k' - k) = c \\
&\quad\Rightarrow\quad a \mid c
\end{aligned}$$

$\square$

Definition (prime number)

The integer $p > 1$ is called a **prime** number if its only positive divisors are **itself** and the **unity.**

Definition (greatest common divisor)

The greatest common divisor g of two integers a and b, $g = (a, b)$ if

$$g \mid a \ \wedge g \mid b \ \wedge ((\forall d)(d \mid a \wedge d \mid b) \Rightarrow d \mid g).$$

Definition (coprime integers)

Two positive integers, a and b, are coprime if $(a, b) = 1$.

Theorem

$g = (a, g)$ is the smallest positive linear combination of a and b.

Let $S = \{ax + by : x, y \in \mathbb{Z} \ \wedge \ ax + by > 0\}$. $S \neq \emptyset$ (as $a^2 + b^2 \in S$). Let $d = \min(S)$.

- Let d' be s.t. $d' \mid a \wedge d' \mid b$, thus

  $$d \ = \ ax + by \ = \ d'q_1x + d'q_2y \ = \ d'(q_1x + q_2y)$$

  thus, $d' \mid d$.
- $a = dq + r, \ 0 \leq r < d$, then

  $$r \ = \ a - dq \ = \ a - (ax + by)q \ = \ a(1 - xq) + b(-yq)$$

  i.e. r is linear combination of a and b.
  $r > 0 \implies r \in S$, but as $r < d$ that would be absurd as $d = \min(S)$. Thus.
  $0 \leq r \ \Rightarrow r = 0.$, and $d \mid a.$. With the same argument we show that $d \mid b$.

Thus, $d = (a, b)$. $\square$

Theorem

A integer p is prime if and only if

$$(\forall a, b \in \mathbb{Z} \setminus \{0\})(p \mid ab \implies p \mid a \vee p \mid b). \qquad (1)$$

($\Rightarrow$) Let p be a prime and $p \mid ab$. If $p \mid a$ the proof is done. If $p \nmid a$ then, as p has no divisors

$$(p, a) = 1$$

Thus

$$(\exists x, y)\ 1 = ax + py \ \Rightarrow\ b = bax + bpy \ \Rightarrow\ p \mid b.$$

($\Leftarrow$) Let p be s.t. (1) and $S = \{n \mid n > 1 \wedge n \mid p\}$. $S \neq \emptyset$ because $p \in S$. Let $m = \min(S)$,

$$m \mid p \ \wedge\ (\exists k)\ mk = p$$

as p satisfies(1), $p \mid m \vee p \mid k$. But

$$p \mid k \ \Rightarrow\ k \geq p \ \Rightarrow\ p = k \ \Rightarrow m = 1$$

(a contradiction!) Then

Theorem (fundamental theorem of arithmetic)

Every positive integer can be written in a unique way as a product of ascending primes.

Definition

Let $a, b \in \mathbb{Z}$, $q \in \mathbb{Z}$ and $r \in \mathbb{N}$ s.t. $0 \leq r < b \wedge a = bq + r$, then one writes

$$a \mod b = r \quad \text{or} \quad a \equiv r \pmod{b}.$$

$$a \equiv r \pmod{b} \iff b \mid (a - r)$$

Let $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$.
Observe that

1. $(w + x) \mod n = (x + w) \mod n$
2. $(w \times x) \mod n = (x \times w) \mod n$
3. $((w + x) + y) \mod n = (w + (x + y)) \mod n$
4. $((w \times x) \times y) \mod n = (w \times (x \times y)) \mod n$
5. $(w \times (x + y)) \mod n = ((w \times x) + (w \times y)) \mod n$
6. $(0 + w) \mod n = w \mod n$
7. $(1 \times w) \mod n = w \mod n$
8. $(\forall w \in \mathbb{Z}_n)(\exists z \in \mathbb{Z}_n)(w + z = 0 \mod n)$

Observe that an additive cancelation rule valid:

$$(a+b) \equiv (a+c) \pmod{n} \;\Rightarrow\; b \equiv c \pmod{n}$$

because $\forall a \in \mathbb{Z}_n \exists b \in \mathbb{Z}_n\ a+b \equiv 0 \pmod{n}$, but

$$((a \times b) \equiv (a \times c) \pmod{n} \;\Rightarrow\; b \equiv c \pmod{n}) \textbf{ if } (a,n)=1$$

if $(a,n) \neq 1$

$$\begin{array}{rrcl} f: & \mathbb{Z}_n & \longrightarrow & \mathbb{Z}_n \\ & z & \longmapsto & a \times z \pmod{n} \end{array}$$

then f **is not** surjective.Let $m=(a,n)$, thus $m \mid a \ \wedge\ m \mid n$, hence $((\exists x)(\exists y)\ a = xm \ \wedge n = ym)$. Then $f(0) = a \times 0 = 0$ and $f(y) = ay = xym = xn \equiv 0 \pmod{n}$.

Theorem (Fermat)

Let p be a prime and a s.t. $p \nmid a$, *then* $a^{p-1} \equiv 1 \pmod{p}$.

$$\begin{array}{rccl} f: & \mathbb{Z}_p^* & \longrightarrow & \mathbb{Z}_p^* \\ & z & \longmapsto & az \pmod{p} \end{array}$$

is surjective thus

$$\begin{aligned} \{a \mod p, 2a \mod p, \ldots, (p-1)a \mod p\} &= \{1, 2, \ldots, p-1\} \\ \prod_{i=1}^{p-1} ai \pmod{p} &= \prod_{i=1}^{p-1} i \\ a^{p-1}(p-1)! &\equiv (p-1)! \pmod{p} \end{aligned}$$

As $\big((p-1)!, p\big) = 1$ one can conclude $a^{p-1} \equiv 1 \pmod{p}$. $\square$

$$\phi(n) = \left|\{i \mid i < n \wedge (i,n) = 1\}\right|$$

If p is a prime then $\phi(p) = p - 1$.
If p and q are primes, then the set of elements n of $\mathbb{Z}_{pq}$ s.t. $(n, pq) \neq 1$ is $\{p, 2p, \ldots, (q-1)p, q, 2q, \ldots, (p-1)q\}$. Thus,

$$\begin{aligned}
\phi(pq) &= (pq - 1) - ((q-1) + (p-1)) \\
&= pq - (p+q) + 1 \\
&= (p-1)(q-1) \\
&= \phi(p)\phi(q).
\end{aligned}$$

Theorem (Euler)

$$(a, n) = 1 \implies a^{\phi(n)} \equiv 1 \pmod{n}.$$

$R = \{x \in \mathbb{N} : 0 < x < n \wedge (n, x) = 1\} = \{x_1, x_2, \ldots, x_{\phi(n)}\}$
$S = \{ax_1 \pmod{n}, ax_2 \pmod{n}, \ldots, ax_{\phi(n)}\}$
$(((a, n) = 1) \wedge ((x_i, n) = 1)) \Rightarrow ((ax_i, n) = 1)$, thus $S \subseteq R$.But
$((ax_i \mod n) = (ax_j \mod n)) \Rightarrow x_i = x_j$ thus $S = R$.

$$\begin{aligned} \prod_{i=1}^{\phi(n)} ax_i &\equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n} \\ a^{\phi(n)} \times \prod_{i=1}^{\phi(n)} x_i &\equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n} \\ a^{\phi(n)} &\equiv 1 \pmod{n} \end{aligned}$$

$\square$

Theorem (Corollary)

Let p and q be prime numbers, $n = pq$ and $0 < m < n$, then

$$m^{\phi(n)+1} \equiv m \pmod{n}.$$

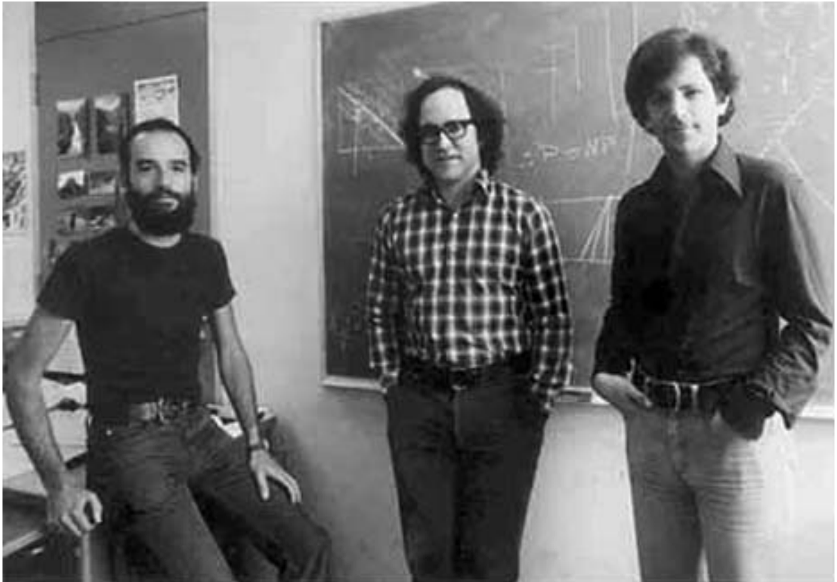
$(m, n) \neq 1 \iff p|m \vee q|m$. $p|m$ $(m = cp) \implies (q, m) = 1$, else,

$$p \mid m \wedge q \mid m \wedge m < pq$$

$$\begin{aligned}
m^{\phi(q)} &\equiv 1 \pmod{q} \\
(m^{\phi(q)})^{\phi(p)} &\equiv 1 \pmod{q} \\
m^{\phi(n)} &\equiv 1 \pmod{q} \\
(\exists k \in \mathbb{N})(m^{\phi(n)} &= 1 + kq) \\
m^{\phi(n)+1} &= m + kcpq = m + kcn \\
m^{\phi(n)+1} &\equiv m \pmod{n}
\end{aligned}$$

$\square$

RSA



RSA

Alice creates her pair of keys (public, private) using the following recipe:

1. Generates two big primes of comparable magnitude: p and q.
2. Defines $n = pq$.
3. Generates $e < \phi(n) = (p-1)(q-1)$ s.t. $(e, \phi(n)) = 1$.
4. Computes $d = e^{-1} \pmod{\phi(n)}$.

The public key is $\langle n, e \rangle$ and the private key is $\langle n, d \rangle$.
If Bob wants to send a message m to Alice, sends $m^e \pmod{n}$.
Alice deciphers the message computing:

$$\begin{aligned} \left(m^e \pmod{n}\right)^d \pmod{n} &= m^{ed} \pmod{n} \\ &= m^{k\phi(n)+1} \pmod{n} \\ &= m \end{aligned}$$

A toy example

1. Let $p = 7$ and $q = 17$.
2. Thus $n = pq = 119$.
3. $\phi(n) = (p-1)(q-1) = 96$.
4. Choose e s.t. $e < \phi(n)$ and $(\phi(n), e) = 1$. Let $e = 5$.
5. Compute $d = e^{-1} \pmod{\phi(n)}$. $d = 77$ $(77 \times 5 = 385 = 4 \times 96 + 1)$.
6. If $m = 19$, enciphered message will be
$$19^5 = 2476099 \equiv 66 \pmod{119}.$$
7. To decipher
$$\begin{aligned} 66^{77} &= 127316015002712725024996823827450919411351129158 \\ &\quad 643807873318778077633686286816610254398613549028 \\ &\quad 148573790434899358326117107662397756833529856 \\ &\equiv 19 \pmod{119}. \end{aligned}$$

RSA (and in general all PKC ciphers) is about 1000 times slower than normal symetric ciphers, this alone make them unusable to directly cipher texts.

There is, however, an even stronger reason. Because public key is public (duh!) it makes PKC vulnerable to a dictionary attack if the message comes from a relatively small set of admissible messages.

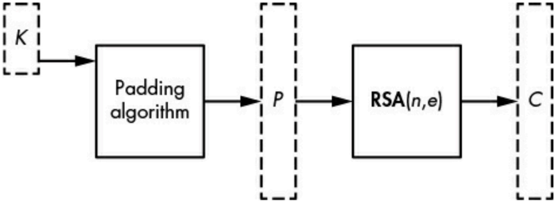
Moreover RSA is a multiplicative homomorfism, i.e.

$$\begin{aligned} E_k(x_1)E_k(x_2) &= (x_1^e \pmod n)(x_2^e \pmod n) = \\ &= (x_1 x_2)^e \pmod n = E_k(x_1 x_2) \end{aligned}$$

and this can get origin to some attacks in some contexts. We say that this weakness makes textbook RSA encryption *malleable*.

Strong RSA Encryption: OAEP

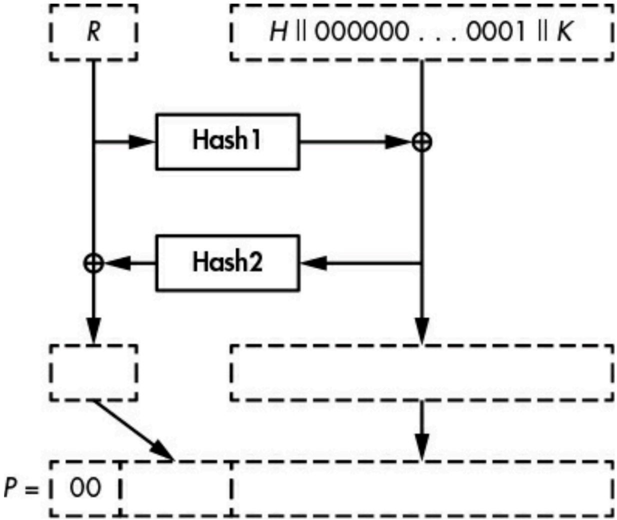
In order to make RSA ciphertexts nonmalleable, the ciphertext should consist of the message data and some additional data called *padding*.



Optimal Asymmetric Encryption Padding (OAEP)

OAEP's Security

OAEP uses a pseudorandom number generator (PRNG) to ensure the indistinguishability and nonmalleability of ciphertexts by making the encryption probabilistic. It has been proven secure as long as the RSA function and the PRNG are secure and, to a lesser extent, as long as the hash functions aren't too weak. You should use OAEP whenever you need to encrypt with RSA.


R
H || 000000 . . . 0001 || K
Hash1
Hash2
P = 00

Signing with RSA

To sign a message m an agent just need to compute

$$m^d \pmod{n}.$$

The verification is just a “deciphering” with the public key.

Breaking simple RSA signature

First, it is worthwhile to note that

$$\begin{aligned} 0^d \pmod{n} &= 0 \\ 1^d \pmod{n} &= 1 \\ (n-1)^d \pmod{n} &= (-1)^{d \bmod 2}, \end{aligned}$$

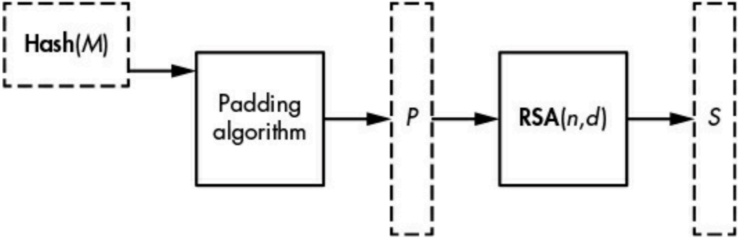
thus, disregarding the value of the private key, an attacker can forge signatures of 0, 1 and $(n-1)$.

More troublesome is the possibility of a *blinding attack*. If one finds a value r such that $r^e m \pmod{n}$ is a message that is plausible of being signed, then

$$s = (r^e m)^d = r m^d$$

and thus is simple to obtain m^d.

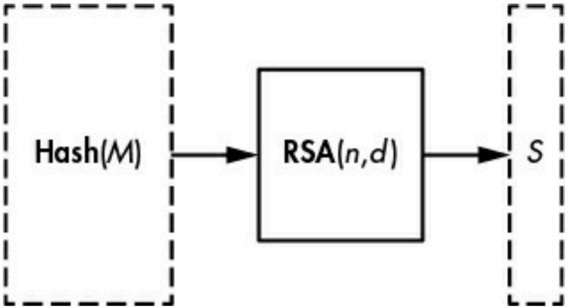
The PSS Signature Standard



1. Pick an r-byte random string r using the PRNG.
2. $m' = 0000000000000000||\text{Hash1}(m)||r$
3. Compute the h-byte string $h = \text{Hash1}(m')$.
4. Set $l = 00 \cdots 00||01||r$
5. Set $l = l \oplus Hash2(h)$
6. Convert $p = l||h||bc$ to a number, $x < n$, lower than n.
7. Given the value x just obtained, compute the RSA function $x^d \pmod{n}$ to obtain the signature.

Like OAEP, PSS is provably secure, standardised, and widely deployed. Also like OAEP, it looks needlessly complex and is prone to implementation errors and mishandled corner cases. But unlike RSA encryption, there’s a way to get around this extra complexity with a signature scheme that doesn’t even need a PRNG, thus reducing the risk of insecure RSA signatures caused by an insecure PRNG.

Full Domain Hash Signatures



It could not be simpler, but PSS has a better provable security.

These stronger theoretical guarantees are the main reason cryptographers prefer PSS over FDH, but most applications using PSS today could switch to FDH with no meaningful security loss. In some contexts, however, a viable reason to use PSS instead of FDH is that PSS's randomness protects it from some attacks on its implementation, such as fault attacks.

Fast exponentiation

How to compute a^{14}? $a \times a \times a \times \cdots \times a$ needs 13 operations.

a^2, a^4, a^8, only takes 3, and... $a^{14} = a^2 a^4 a^8$, only 6 operations.

To speedup decryption we need a little more wisdom...

The Chinese remainder theorem

Theorem (Chinese remainder theorem)

Let

$$m = \prod_{i=1}^{r} m_i$$

with $\quad (\forall i, j)(i \neq j \implies (m_i, m_j) = 1)$. *Then*

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ \vdots \\ x \equiv a_r \pmod{m_r} \end{cases}$$

has a solution for x, and all solutions y of the system are such

$$y \equiv x \pmod{m}.$$

First let us show that all solutions are congruent $\pmod{m}$. Let x' and x'' be solutions, make $x = x' - x''$. Thus $x \equiv 0 \pmod{m}$ because $(\forall i)(x \equiv 0 \mod m_i)$. Thus

$$x' \equiv x'' \mod m.$$

Let $m_i' = \frac{m}{m_i}$. Clearly $(m_i, m_i') = 1$, for all i. Thus

$$(\forall i)(\exists n_i)(m_i' n_i \equiv 1 \mod m_i).$$

Make

$$x = \sum_{i=1}^{r} a_i m_i' n_i.$$

For each $i \neq j$ $m_j \mid a_i m_i' n_i$, hence

$$(\forall i)\left(x = \sum_{i=1}^{r} a_i m_i' n_i \equiv a_i \mod m_i\right).$$

$\square$

Applying the CRT to RSA is quite simple, because there are only two factors for each n (namely p and q). Given a ciphertext y to decrypt, instead of computing $y^d \pmod{n}$, you use the CRT to compute $x_p = y^s \pmod{p}$, where $s = d \pmod{(p\breve{}1)}$ and $x_q = y^t \pmod{q}$,where $t = d \pmod{(q\breve{}1)}$. You now combine these two expressions and compute x to be the following:

$$x = x_p q(1/q \pmod{p}) + x_q p(1/p \pmod{q}) \pmod{n}.$$

This makes the computation 4 times faster.

Attacks!