RECAP: RSA CRYPTOSYSTEM

X: $P_X(M) = M^e \bmod n$; $S_X(M) = M^d \bmod n$ with $n = p \cdot q$: p, q "large" primes; $\gcd(e, \varphi(n)) = 1$; $d = e^{-1} \bmod \varphi(n)$

- If FACTORING or DISCRETE LOGARITHM are "easy" $\Rightarrow$ RSA is "easily" cracked
- Factoring n is equivalent to computing $\varphi(n)$
- RSA may be cracked in other ways!

ATTACKS TO RSA

The RSA cryptosystem is not immune to attacks (cryptoanalysis) usually targeting weaknesses and naive choices of the parameters - 5 ATTACKS -

(1) When choosing p and q one should make sure that they are not "too close"

Assume that $n = p \cdot q$, where p, q are very close primes ($|p - q| = k$, with small k)

W.l.o.g., let $p > q$. Then

$$q = p - k \text{ and } n = p(p-k),$$

therefore it holds

$$p^2 - kp - n = 0$$

Thus p is one solution to the above 2nd degree equation

$$x^2 - kx - n = 0$$

(the other is $-q$, prove it)

Since we don't know k we can proceed as follows

for $t = 1, 2, \ldots$

* solve $x^2 - tx - n = 0$ *

if * one solution p is integer positive *

then return p

The running time of this procedure is $O(k) = O(p-q)$

If this number is small, we can easily factor n

In practice: we generate p, q with ≥ 1024 bits randomly. With high probability |p-q| is very high (otherwise most bits of their binary representation should be the same – unlikely)

---

(2) REUSING PRIME FACTORS

Primes generated using Pseudo-Random Number Generators: deterministic generation (given initial seed). The PRNG could generate the same prime p for $n_1 = p \cdot q_1$ and $n_2 = p \cdot q_2$

But then $p = \gcd(n_1, n_2)$

(n_1 and n_2 are available in the respective public keys)

These are accidents caused by a sloppy use of RSA (must have really happened)

(3) CHOICE OF PUBLIC EXPONENT e

e should be chosen "small" (to achieve low complexity for public encoding) but NOT TOO SMALL!

Suppose that $\geq e$ participants choose the same value of e.

Suppose that e participants receive the same encrypted message:

$$c_1 = (M^e \bmod n_1),\ c_2 = (M^e \bmod n_2),\ \dots,\ c_e = (M^e \bmod n_e)$$

Two cases

1) (Unlikely)

$$\exists\, i,j \mid \gcd(n_i, n_j) > 1$$

Then I can factor n_i, n_j and decrypt M

(= p_i or = q_i)

2) (Likely)

$$\gcd(n_i, n_j) = 1 \quad i \neq j$$

In this case, let $n = \prod_{j=1}^{e} n_j$

Use the CRT to compute

$$M^e \bmod n = f^{-1}(c_1, c_2, \dots, c_e)$$

(observe that $(M^e \bmod n) \bmod n_j = c_j$)

But

$M \in \mathbb{Z}_{n_1}$, $M \in \mathbb{Z}_{n_e}$

$M < n_1, \dots, M < n_e \Rightarrow$

$M^e < n_1 \cdot \dots \cdot n_e = n$

$\Rightarrow M^e < n \Rightarrow M^e \bmod n = M^e$

M^e is not a modular exponentiation!

I can compute $M = x = \sqrt[e]{y}$ $(y = M^e)$ in polynomial time $\log(M^e)$

(binary search on $\{1, \dots, M\}$: find $x : x^e = y$ EXERCISE)

4 REUSING THE SAME MODULE

To save time on prime generation, use the same module n - e.g. together with different public/private exponents

Assume that we find two public keys

$$P_x = \langle e_1, n \rangle \quad P_y = \langle e_2, n \rangle$$

with $\gcd(e_1, e_2) = 1$

(likely if we pick e_i's as "small" primes)

then

$$\exists x, y : 1 = e_1 x + e_2 y$$

Since $e_1, e_2 > 1 : (x < 0) \overset{XOR}{\vee} (y < 0)$

Let $x < 0$. Assume that X, Y receive the same message:

$$c_1 = M^{e_1} \bmod n, \quad c_2 = M^{e_2} \bmod n$$

TWO CASES:

1) (Unlikely) (Why?)

$c_1 \notin \mathbb{Z}_n^*$ ⇒

$(\gcd(c_1, n) = p) \vee (\gcd(c_1, n) = q)$

→ factor n and obtain S_x, S_y

2) (Likely)

$c_1 \in \mathbb{Z}_n^*$

We have

$$M = M^1 = M^{e_1 x + e_2 y} = (M^{e_1 x + e_2 y}) \bmod n$$

$$= \left((M^{e_1} \bmod n)^x \, (M^{e_2} \bmod n)^y \right) \bmod n \quad (\text{Lo})$$

$$= (c_1^x \cdot c_2^y) \bmod n$$

Then $(c_1^x \cdot c_2^y) \bmod n = M_2$

$$= \left((c_1^{-1})^{-x} \cdot (c_2)^y \right) \bmod n$$

It is sufficient to compute

$z = c_1^{-1} \bmod n$ using $EE(n, c_1)$

then compute

$M_1 = 2^{-x} \bmod n$ using MOD-EXP$(a, -x)$ (since $-x > 0$), and

$M_2 = c_2^{y} \bmod n$ using MOD-EXP(c_2, y)

to obtain $M = M_1 M_2 \bmod n$!

MORALE: It is dangerous to send the same message M to many users! Also, no recycling of $?_B$!

WORKAROUND: RANDOM PADDING (add junk bits to make all messages different)

5 ATTACKS VIA AUTHENTICATION

- Assume that Charlie has intercepted $y = M^e \bmod n$ sent to Bob ($?_B = (e, n)$)
- Assume that Bob is not too careful in the use of his digital signature

- Charlie will generate a random $r \in \mathbb{Z}_n$. Clearly $r \in \mathbb{Z}_n^*$ with very high probability!

If $p, q \in O(\sqrt{n})$:

$|\mathbb{Z}_n - \mathbb{Z}_n^*| = |\{k_p, k_q : k \in \mathbb{N}\}|$

$\approx (p+q) = \Theta(\sqrt{n})$

$\Rightarrow Pr(r \notin \mathbb{Z}_n^*) = \frac{\Theta(\sqrt{n})}{n} =$

$= \Theta\left(\frac{1}{\sqrt{n}}\right)$!

$\frac{1}{\sqrt{n}} \approx \frac{1}{2^{1024}}$! (if we choose 1024-bit primes)

- Charlie sends $(r^e M^e) \bmod n$

$= (rM)^e \bmod n$ to Bob

asking for Bob's signature

(Charlie may claim that rM

is a petition, etc...)

If Bob signs, he generates

$$S_B((rM)^e \bmod n) =$$
$$= (rM)^{ed} \bmod n = rM \bmod n$$

NOTE Premultiplying by r, Charlie makes sure that Bob does not recognize plaintext message M!

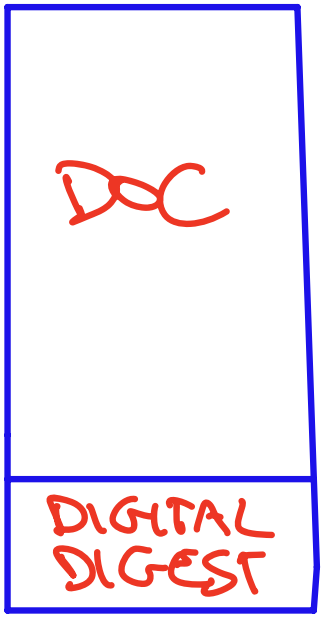
- Charlie receives $rM \bmod n$ and obtains M as

$$M = (r^{-1}(rM)) \bmod n$$

(r^{-1} exists since $\gcd(r, n) = 1$ and can be computed efficiently)

MORALE: Make sure that you understand what you sign!

WORKAROUND To avoid signature-based scams, the signature protocol has been strengthened:



DIGITAL-DIGEST = h(DOC)

h: ONE-WAY HASH FUNCTION (difficult to invert)

It is computationally hard to generate a document while DIGITAL DIGEST is a given string X!

The user only signs the digital digest

- Alice is not able to create a DOC with digital digest = $(rH)^e \bmod n$!

- Digital Digest also used for efficiency reasons:

$$|\text{DIGITAL-DIGEST}| << |\text{DOC}|$$

important, since applying the $S_K(\cdot)$ has cubic complexity.

PRIMALITY

Development of efficient tests to check whether $p > 1$ is prime.

Two variants:

1. (Simpler) Allows to check the primality of a randomly chosen number p (but the test itself is deterministic). The probabilistic analysis of the test is based on the probability space Ω induced by the random input
   → Useful for selecting p, q in RSA (random primes).

The analysis will prove that on a random input n the test fails with probability p_n:

$$\lim_{n \to +\infty} p_n = 0$$

(not correct)

2. (More complex) Provide a randomized algorithm (using RANDOM(·)) that for any fixed n, tests if n is prime. The Miller-Rabin algorithm can be made correct with probability arbitrarily close to 1 (by increasing its running time):

$$\forall s : \text{Prob}(MR(n,s) \text{ fails}) < 2^{-s}$$

$$T_{MR}(|\langle n \rangle|, s) = O(s \cdot |\langle n \rangle|^3)$$

(s=30 suffices to obtain a failure probability $< 1/10^9$)