

Some number theoretical background for

RSA - cryptography, and Shor's algorithm

need: modular arithmetics

\triangleq arithmetics in $\mathbb{Z}_N = \{0, 1, \dots, N-1\}$

for $a, b \in \mathbb{Z}$:

$$a \equiv b \pmod{N}$$

iff $a = b + k \cdot N$ for

some $k \in \mathbb{Z}$

e.g.: $5 \equiv 0 \pmod{5}$

$$7 \equiv 2 \pmod{5}$$

$$-8 \equiv 2 \pmod{5}$$

$$143 \equiv 53 \pmod{5}$$
 etc.



sometimes (particularly in comp. science)

mod N as operation:

$$a \text{ mod } N := b \in \mathbb{Z}_N \text{ s.t. } a = b + kN$$

(= remainder of N divided by a)

→ " + ", " - ", " · " in \mathbb{Z}_N as usual, up
to " mod N " !

Note:

- $a + b = a \text{ mod } N + b \text{ mod } N \pmod{N}$
- $a \cdot b = (a \text{ mod } N) \cdot (b \text{ mod } N) \pmod{N}$

Modular Exponentiation

efficiently "by squaring":

$$b^a = b \sum_{e=0}^{u-1} a_e 2^e = \prod_{e=0}^u \left(b^{(2^e)} \right)^{a_e}$$

$\{0, 1\}$
↙
 \approx
↗

$b^2, b^4, b^8, b^{16} \dots \pmod{N}$ can
be determined recursively:

$$b \xrightarrow{\cdot b \pmod{N}} b^2 \xrightarrow{\cdot b^2 \pmod{N}} b^4 \xrightarrow{\cdot b^4 \pmod{N}} b^8 \text{ etc.}$$

→ $b^a \pmod{N}$ can be efficiently
computed in $\text{poly}(\log N)$ time!

Modular division in \mathbb{Z}_N :

Def.:

a and b co-prime: \Leftrightarrow a and b don't
have common divisors

$$\Leftrightarrow \gcd(a, b) = 1$$



greatest common divisor of
 a and b

Thm. (Bezout):

$$\gcd(a, b) \stackrel{!}{=} \min \left\{ h a + l b > 0, \quad h, l \in \mathbb{Z} \right\}$$

↑ proof e.g. in M. Schroeder, Number

Theory in Science and Communication (springer)

For a and b co-prime this means:

!

$$1 = \gcd(a, b) = ha + lb$$

$$\rightarrow \exists h : \quad ha = 1 \pmod{b}$$

$\Leftarrow a^{-1}$!

→ Thm.:

inverse a^{-1} of $a \bmod N$ exists

iff a and N are co-prime.

→ efficient computation of $a^{-1} \bmod N$
with extended Euclid's algorithm:

Euclid's algorithm repeatedly makes
use of

$$\text{gcd}(\alpha, \beta) = \text{gcd}(b, \alpha \bmod b),$$

as in the following example for

$$\text{gcd}(71, 31) :$$

$$71 = 2 \cdot 31 + 9 \quad (1)$$

$$31 = 3 \cdot 9 + 4 \quad (2)$$

$$9 = 2 \cdot 4 + 1 \quad (3)$$

$$4 = 4 \cdot 1 + 0 \quad (4)$$

$$\rightarrow \text{gcd}(71, 31) = 1 \quad \underline{\text{end}}$$

$$1 \stackrel{(3)}{=} 9 - 2 \cdot 4$$

$$\stackrel{(2)}{=} 9 - 2(31 - 3 \cdot 9) = 7 \cdot 9 - 2 \cdot 31$$

$$\stackrel{(1)}{=} 7(71 - 2 \cdot 31) - 2 \cdot 31$$

i.e.

!

$$1 = -16 \cdot 31 + 7 \cdot 71$$

$$= \cancel{-16} \cdot \cancel{31} + \cancel{7} \cdot \cancel{71}$$

hence $31^{-1} \doteq -16 \equiv 55 \pmod{71}$

Euler's φ -function

$\varphi(n) :=$ number of elements in

$$\{1, 2, 3, \dots, n-1, n\}$$

that are co-prime to n

e.g.:

$$\bullet \varphi(1) = 1$$

$$\bullet \varphi(10) = \# \{ \underset{1}{1}, \cancel{2}, \cancel{3}, \cancel{4}, \cancel{5}, \cancel{6}, \cancel{7}, \cancel{8}, \cancel{9}, \cancel{10} \} = 4$$

"number of elements in"

$$\bullet \varphi(11) = 10$$

for p prime : $\varphi(p) = p-1$

(since all $a < p$ or co-prime to prime p !)

Multiplication-theorem:

!

For a co-prime b : $\varphi(a \cdot b) = \varphi(a) \varphi(b)$

$$\rightarrow p, q \text{ prime} \rightarrow \varphi(p \cdot q) = \varphi(p) \varphi(q) \\ = (p-1)(q-1)$$

E.g.: $\varphi(15) = \varphi(3 \cdot 5) = 2 \cdot 4 = 8$

]

Euler's theorem:

For a and n co-prime:

$$a^{\varphi(n)} = 1 \pmod{n}$$

(for a proof see e.g. Schroeder's book)

for the special case that n is prime

also known as

Fermat's Little Thm.

for p prime: $a^{p-1} = 1 \pmod{p}$

Examples:

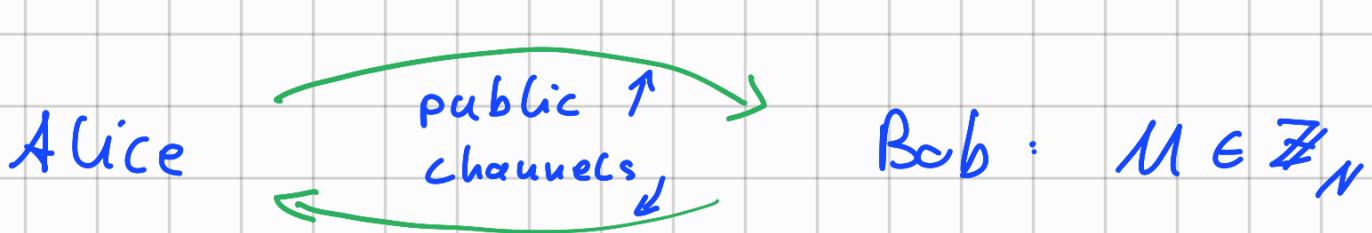
- $2^6 \equiv 1 \pmod{7}$ ✓
 $(64 = 1 + 9 \cdot 7 \therefore)$
- 7 divides 999 999 !

↑ Euler/Fermat: $10^{7-1} \equiv 1 \pmod{7}$
 $\Leftrightarrow 10^6 - 1 \equiv 0 \pmod{7}$ —

Euler's theorem is basis for

R S A - public-key-encryption

[[L Adleman
Shamir
Rivest (1977)]]



How can Bob transfer message M
secretly to Alice via public channels?
(= www)

RSA solve the problem as follows:

Alice:

- draws randomly two primes p and q
(of about 100 digits each)
- computes $n = p \cdot q$
and $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$
- draws e co-prime to $\varphi(n)$
- publishes "public key" (e, n)

Bob:

- encodes message m with to

$$E(m) = m^e \mod n$$

and sends $E(m)$ to Alice

Alice

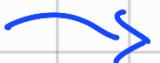
- receives E from Bob and decodes with

$$D(E) := E^d \bmod n$$

where $d = e^{-1} \bmod \varphi(u)$

Note: Computing prime factors p and q of n , a potential eavesdropper Eve can't compute $\varphi(u)$ as $\varphi(n=pq) = \varphi(p) \cdot \varphi(q) = (p-1)(q-1)$ and thus also can't compute d in order to decode E !

Why does Alice's decoding work?



- d as inverse of $e \pmod{\varphi(n)}$ satisfies
 $de = 1 + k\varphi(u)$;
- assume that M and n are co-prime
(e.g. when $M < p$ and $M < q$)

$$\begin{aligned} \rightarrow E(M)^d &= (M^e)^d = M^{1+k\varphi(u)} \\ &= M \underbrace{(M^{e(u)})}_\text{Euler}^k = M \pmod{n} \end{aligned}$$

✓

Ruvh.:

RSA - encryption is considered to
be safe as much as factoring
 n into p and q is seen to be
a hard problem! (as if seems)

to be the case for classical
computers)

Since Shor's algorithm running
on a quantum computer easily
finds factors p and q of N ,
quantum computing poses
a real threat to private
communication via WWW !

Actually, Shor's algorithm primarily
finds the order r of $b \bmod N$;

\hookrightarrow least integer r s.t.

$$b^r \equiv 1 \pmod{N}$$

this enables efficient integer factori-
zation, but also immediately \longrightarrow

RSA - decryptions by eavesdroper

Eve:

- determines order r of encrypted message $E \bmod n$ by use of Schnorr's algorithm
- computes \hat{d} as inverse of e mod r ($\rightarrow \hat{d}e = 1 + kr$)
- decodes by
$$\hat{D}(E) := E^{\hat{d}} \bmod n$$

this works because of the following

fact:

If e and $\varphi(n)$ co-prime,
then order r of $M^e \bmod n$

also order of $\underline{M} \bmod n$!

(*) true for e and $\varphi(n)$ of RSA-key (e, n) !

fher :

$$\begin{aligned}\tilde{D}(\underline{E(\underline{M})}) &= \tilde{D}(M^e) = (M^e)^{\hat{d}} \\ &= M^{\hat{d}e} = M^{1+hr} \\ &= M (M^r)^k = \underline{M} \quad (\text{mod } u) \\ &\quad // \\ &1 \quad (\text{mod } u)\end{aligned}$$

✓