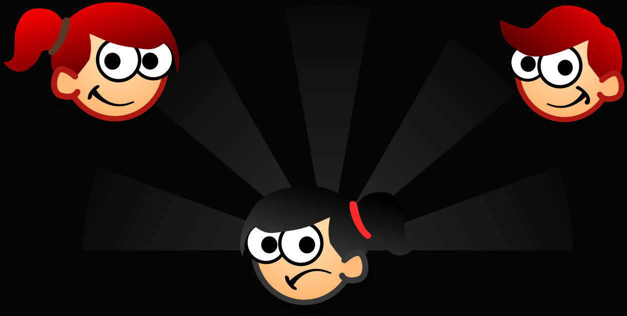
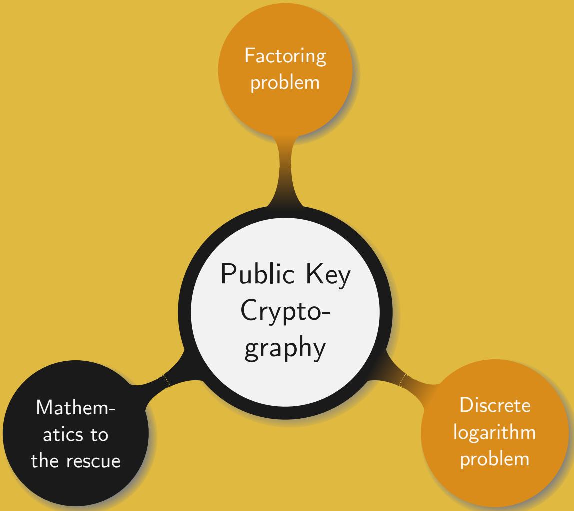


Introduction to Cryptography

3. Public Key Cryptography

Manuel – Summer 2022

Chapter organisation



Public Key Cryptography

Factoring problem

Discrete logarithm problem

Mathematics to the rescue

Reminder

Symmetric-key cryptosystem:

- Encryption depends on a secret key K
- Decryption depends on a secret key K' easily derived from K
- Knowing K or K' is the same
- The key must be securely shared beforehand

Public-key cryptosystem:

- Encryption depends on a public key K
- Decryption depends on a secret key K'
- Finding K' when knowing K is computationally infeasible
- No prior communication is required

One way functions

One way function: function easy to evaluate but hard to invert

Requirements for encrypting with a one-way function E:

- E must be injective
- Some secret that allows to invert E

Trapdoor one-way function: one-way function, easy to invert when knowing a *trapdoor*

Definition (Group)

A *group* is a pair $(G, \circ)$ consisting of a set G and a *group operation* $\circ \colon G \times G \to G$ that verifies the following properties:

i *Associativity:* $a \circ (b \circ c) = (a \circ b) \circ c$ for all $a, b, c \in G$

ii *Existence of a unit element:* there exists an element $e \in G$ such that $a \circ e = e \circ a = a$ for all $a \in G$

iii *Existence of inverse:* for every $a \in G$ there exists an element $a^{-1} \in G$ such that $a \circ a^{-1} = a^{-1} \circ a = e$

A group is called *abelian* if in addition to the above properties

iv *Commutativity:* $a \circ b = b \circ a$ for all $a, b \in G$.

Mathematical structures

Definition (Ring)

A *ring* is a triple $(R, +, \cdot\,)$ consisting of a set R and two *binary operations* $+, \cdot : R \times R \to R$ such that

1. $(R, +)$ is an abelian group
2. *Multiplicative unit:* there exists an element $1 \in R$ such that
$$a \cdot 1 = 1 \cdot a = a \qquad\qquad \text{for all } a \in R$$
3. *Associativity:* for any $a, b, c \in R$,
$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$
4. *Distributivity:* for any $a, b, c \in R$,
$$a \cdot (b + c) = (a \cdot b) + (a \cdot c), \quad (b + c) \cdot a = (b \cdot a) + (c \cdot a)$$

A ring is called *commutative* if in addition to the above properties

5. *Commutativity:* $a \cdot b = b \cdot a$ for all $a, b \in R$

Mathematical structures

Definition (Field)

Let $(F, +, \cdot)$ be a commutative ring with unit element of addition 0 and unit element of multiplication 1. Then F is a *field* if

i. $0 \neq 1$

ii. For every $a \in F \setminus \{0\}$ there exists an element a^{-1} such that
$$a \cdot a^{-1} = 1.$$

Remark. Another way of writing this definition is to say that $(F, +, \cdot)$ is a field if $(F, +)$ and $(F \setminus \{0\}, \cdot)$ are abelian groups and $0 \neq 1$, and $\cdot$ distributes over $+$.

Mathematical structures

Example. Let n be an integer, and $\mathbb{Z}/n\mathbb{Z}$ be the set of the integers modulo n

- $(\mathbb{Z}/n\mathbb{Z}, +)$ also denoted $(\mathbb{Z}_n, +)$ is a group
- $(\mathbb{Z}/n\mathbb{Z}, +, \cdot)$ is a ring
- If n is prime then $(\mathbb{Z}/n\mathbb{Z}, +, \cdot)$ is the field $\mathbb{F}_n$
- The invertible elements of $\mathbb{Z}/n\mathbb{Z}$, with respect to '$\cdot$', form a group denoted $\mathrm{U}(\mathbb{Z}/n\mathbb{Z})$ or sometimes $\mathbb{Z}_n^\times$ or $\mathbb{Z}_n^*$
- $(\mathbb{Z}/n\mathbb{Z}[X], +, \cdot)$ is the ring of the polynomials over $\mathbb{Z}/n\mathbb{Z}$
- If n is prime and the polynomial $P(X)$ is irreducible over $\mathbb{F}_n[X]$, then $(\mathbb{F}_n[X]/\langle P(X)\rangle, +, \cdot)$ is a field; this is $\mathbb{F}_{n^{\deg P(x)}}$

Order

Definitions

Let G be a group.

1. The *order* of G is its cardinality
2. The *order* of an element $g \in G$ is the smallest positive integer m such that $g^m = 1$
3. An element of order equal to the order of the group is called a *primitive element* or a *generator*
4. When $G = \mathbb{Z}/n\mathbb{Z}$, *Euler's totient function* $\varphi(n)$ counts the number of invertible elements, that is the number of elements k such that $\gcd(n, k) = 1$

Order

Example. The order of $\mathrm{U}(\mathbb{Z}/13\mathbb{Z}) = 12$ and 2 is a generator:

i	$2^i \bmod 13$	i	$2^i \bmod 13$	i	$2^i \bmod 13$	i	$2^i \bmod 13$
1	2	4	3	7	11	10	10
2	4	5	6	8	9	11	7
3	8	6	12	9	5	12	1

Remark. Let p be a prime and α be a generator of $G = \mathrm{U}(\mathbb{Z}/p\mathbb{Z})$. Then any element $\beta \in G$ can be written $\beta = \alpha^i$, $1 \leq i \leq p-1$. Noting $d = \gcd(i, p-1)$ we have

$$\beta^{\frac{p-1}{d}} = \left(\alpha^i\right)^{\frac{p-1}{d}} = \left(\alpha^{p-1}\right)^{\frac{i}{d}} = 1.$$

Suppose that the order of β divides $\frac{p-1}{d}$. Then $\operatorname{ord}(\beta) = \frac{p-1}{kd}$ for some $k > 1$ such that $kd \nmid i$, meaning that $\frac{p-1}{k} \cdot \frac{i}{d}$ is not a multiple of $p-1$.↯ Hence the order of β is $\frac{p-1}{d}$.

Revisiting the CRT

In theorem 2.17 we recalled that a system of congruences has a unique solution modulo the product of all the moduli of the system. In fact this result can be rephrased in term of group structure.

We first recall that an *isomorphism* is a bijection that preserves algebraic structures.

Theorem (Chinese Remainder theorem (CRT))

Let n be a positive integer with prime decomposition $n = \prod_i p_i^{e_i}$. Then there exists a ring *isomorphism* between $\mathbb{Z}/n\mathbb{Z}$ and $\prod_i \mathbb{Z}/p_i^{e_i}\mathbb{Z}$.

Order of the group of units

From the previous theorem (3.11)

$$\mathrm{U}(\mathbb{Z}/n\mathbb{Z}) \approx \mathrm{U}\left(\prod_i \mathbb{Z}/p_i^{e_i}\mathbb{Z}\right).$$

Noting that a non invertible element of $\mathbb{Z}/p_i^{e_i}\mathbb{Z}$ is of the form kp_i for some integer k, it cannot be coprime to n and as such is not invertible modulo n. Conversely an element that is not invertible mod n is a multiple of some p_i. Therefore

$$\mathrm{U}(\mathbb{Z}/n\mathbb{Z}) \approx \mathrm{U}\left(\prod_i \mathbb{Z}/p_i^{e_i}\mathbb{Z}\right) \approx \prod_i \mathrm{U}\left(\mathbb{Z}/p_i^{e_i}\mathbb{Z}\right).$$

Proposition

If m and n are two coprime integers then $\varphi(mn) = \varphi(m)\varphi(n)$. In particular if m and n are prime $\varphi(mn) = (m-1)(n-1)$.

Lagrange's theorem

Having a way to determine the order of $\mathrm{U}(\mathbb{Z}/n\mathbb{Z})$, we now focus on the order of its elements. We first recall a fundamental result from group theory.

Theorem (Lagrange's theorem)

Let G be a finite group and H be a subgroup of G. Then the order of H divides the order of G.

Noting that each element x of G generates a subgroup of order $\operatorname{ord}_G x$, it follows that the order of any element x of G divides the order of G.

Using Lagrange's theorem it is then possible to derive a result to quickly verify whether an invertible element modulo a prime p is a generator of $\mathrm{U}(\mathbb{Z}/p\mathbb{Z})$. But first we provide an example and then extend Fermat's little theorem (2.13).

Lagrange's theorem

Example. For $n = 5$, $\mathrm{U}(\mathbb{Z}/5\mathbb{Z}) = \{1, 2, 3, 4\}$ which is a group of order 4. Therefore each of those four elements generates a subgroup of $\mathrm{U}(\mathbb{Z}/5\mathbb{Z})$. Moreover these subgroups will have order 1, 2, or 4, since 4 is divisible by 1, 2, and 4.

In fact we have

$$\begin{aligned} \langle 1\rangle &= \{1\}\,, & \langle 2\rangle &= \{2, 4, 3, 1\}\,, \\ \langle 4\rangle &= \{4, 1\}\,, & \langle 3\rangle &= \{3, 4, 2, 1\}\,. \end{aligned}$$

That is, we have two groups of order 4 ($\langle 2\rangle$ and $\langle 3\rangle$), one group of order 2 ($\langle 4\rangle$), and one group of order 1 ($\langle 1\rangle$).

In particular note that the order of an element is equal to the order of the subgroup it generates.

Euler's theorem

Theorem (Euler's theorem)

Let a and n be two coprime integers. Then

$$a^{\varphi(n)} \equiv 1 \bmod n.$$

Proof. From the previous reasoning on Lagrange's theorem (3.13) there exists $k > 0$ such that $k|\varphi(n)$ and $a^k = 1$. Writing $\varphi(n) = kl$ for some integer l we have

$$a^{\varphi(n)} = a^{kl} \equiv \left(a^k\right)^l \equiv 1^l = 1 \bmod n.$$

□

Simple calculation

Example. Calculate $2^{639613} \bmod 5353$.

First we note that 5353 can be written as the product of two primes: 101 and 53. Therefore $\varphi(5353) = 100 \cdot 52 = 5200$.

Observing that $639613 \equiv 13 \bmod 5200$ we need to consider $2^{13} \bmod 5353$.

As $2^{13} = 8192$ we obtain $2^{639613} \equiv 2839 \bmod 5353$.

Remark. The previous discussion can be simply summarized as follows: when working modulo n, the exponent must be considered mod $\varphi(n)$.

Finding primitive elements

Theorem

Let $p > 2$ be a prime and $\alpha \in \mathrm{U}(\mathbb{Z}/p\mathbb{Z})$. Then α is a generator of $\mathrm{U}(\mathbb{Z}/p\mathbb{Z})$ if and only if for all primes q such that $q|(p-1)$, $\alpha^{(p-1)/q} \not\equiv 1 \bmod p$.

Proof.

($\Rightarrow$) Since α is a generator, for all $1 \leq i < p-1$, $\alpha^i \not\equiv 1 \bmod p$.

($\Leftarrow$) Suppose that α is invertible but does not generate $\mathrm{U}(\mathbb{Z}/p\mathbb{Z})$. Calling its order d, the fraction $(p-1)/d$ defines an integer larger than 1. This is true because $d|(p-1)$ (Lagrange's theorem (3.13)) and $d < (p-1)$. If q is a prime divisor of $(p-1)/d$, then d divides $\frac{p-1}{q}$. So $\alpha^{(p-1)/q} \equiv 1 \bmod p$.

□

Finding primitive elements

Corollary

Let α be a generator of $\mathrm{U}(\mathbb{Z}/p\mathbb{Z})$.

1. Let n be an integer. Then $\alpha^n \equiv 1 \bmod p$ if and only if $n \equiv 0 \bmod (p-1)$.
2. Let j and k be two integers. Then $\alpha^j \equiv \alpha^k \bmod p$ if and only if $j \equiv k \bmod (p-1)$.

Proof. (1) This is straightforward from the previous theorem (3.17).

(2) Without loss of generality assume $j \geq k$. First suppose that $\alpha^j \equiv \alpha^k \bmod p$. Dividing both sides by α^k yields $\alpha^{j-k} \equiv 1 \bmod p$. From (1) we have $j-k \equiv 0 \bmod (p-1)$.

Conversely if $j \equiv k \bmod (p-1)$ then $j-k \equiv 0 \bmod (p-1)$, and by (1) $\alpha^{j-k} \equiv 1 \bmod p$. Finally we multiply by α^k. □

Order and factorization

We now relate the order of the elements in $\mathrm{U}(\mathbb{Z}/n\mathbb{Z})$, where n is a composite integer, to factoring n.

Let x be an element of order r in $\mathrm{U}(\mathbb{Z}/n\mathbb{Z})$. By definition we have $x^r \equiv 1 \bmod n$, that is $n|(x^r - 1)$.

If the order r is even then $x^r - 1 = (x^{r/2} - 1)(x^{r/2} + 1)$. In this case both $\gcd(x^{r/2} - 1, n)$ and $\gcd(x^{r/2} + 1, n)$ are factors of n.

Conversely knowing the factorization of n gives $\varphi(n)$. Since the order of an element x in $\mathrm{U}(\mathbb{Z}/n\mathbb{Z})$ divides $\varphi(n)$ it suffices to write $\varphi(n) = \prod_i p_i$, where the p_i are the prime factors of $\varphi(n)$. Then calculate $x^{a/p_i} \bmod n$, with $a = \varphi(n)$. If $x^{\varphi(n)/p_k} \equiv 1 \bmod n$, for some k, then redefine a as a/p_k.

When all the p_i have been tested a defines the order of x. If none of the $x^{\varphi(n)/p_i} \bmod n$ is 1 then x is a generator, i.e. has order $\varphi(n)$.

Square roots modulo p

The previous discussion highlights the difficulty of determining the order of a random element of $\mathrm{U}(\mathbb{Z}/n\mathbb{Z})$, since it is equivalent to factoring n.

Another hard problem related to factorization was presented in chapter 2, namely the QR problem (2.12). In that chapter we studied the case where the primes are congruent to 3 modulo 4.

We now provide a more general result that gives a method to determine whether or not an element is a square modulo an arbitrary prime p.

Proposition

For p an odd prime and a such that $a \not\equiv 0 \bmod p$, $a^{\frac{p-1}{2}} \equiv \pm 1 \bmod p$. Moreover a is a square mod p if and only if $a^{\frac{p-1}{2}} \equiv 1 \bmod p$.

Square roots modulo p

Proof. Defining $y \equiv a^{\frac{p-1}{2}} \bmod p$ and applying Fermat's little theorem (2.13), we have $y^2 \equiv a^{p-1} \equiv 1 \bmod p$. Therefore we have

$$y^2 - 1 \equiv (y-1)(y+1) \equiv 0 \bmod p.$$

As p is prime all the elements but 0 are invertible, meaning that either $y \equiv 1 \bmod p$ or $y \equiv -1 \bmod p$.

If $a \equiv x^2 \bmod p$, then $a^{\frac{p-1}{2}} \equiv x^{p-1} \equiv 1 \bmod p$.

Conversely let g be a generator mod p and write $a \equiv g^j$ for some j. If $a^{\frac{p-1}{2}} \equiv 1 \bmod p$, then

$$g^{j\frac{p-1}{2}} \equiv a^{\frac{p-1}{2}} \equiv 1 \bmod p.$$

From corollary 3.18 we see that $j\frac{p-1}{2} \equiv 0 \bmod (p-1)$ implying that j must be even. Hence $a \equiv g^j \equiv g^{2k}$ and a is a square. □

Legendre symbol

Proposition 3.20 provides a simple way to computationally check if an element is a square modulo a prime. Since this criteria is difficult to use by hand we now introduce an alternative strategy.

Definition (Legendre symbol)

Given p be an odd prime and $a \not\equiv 0 \bmod p$, we define the *Legendre symbol* by

$$\left(\frac{a}{p}\right) = \begin{cases} +1 \text{ if a is a square mod } p \\ -1 \text{ if a is not a square mod } p \end{cases}$$

Legendre symbol

Proposition

Let p be an odd prime.

1. If $a \equiv b \bmod p$, then $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.
2. If $a \not\equiv 0 \bmod p$, then $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \bmod p$.
3. If $ab \not\equiv 0 \bmod p$, then $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$.
4. If $p \equiv 1 \bmod 4$ then -1 is a square mod p.

Proof.

1. The solutions to the congruence $x^2 \equiv a \bmod p$ and $x^2 \equiv b \bmod p$ are the same when $a \equiv b \bmod p$.
2. Combining the definition of Legendre symbol (3.22) with proposition 3.20 yields the result.

Legendre symbol

3. From (2), we have

$$\left(\frac{ab}{p}\right) = (ab)^{\frac{p-1}{2}} \equiv a^{\frac{p-1}{2}} b^{\frac{p-1}{2}} \equiv \left(\frac{a}{p}\right)\left(\frac{b}{p}\right) \bmod p.$$

   Both ends being congruent to ± 1 modulo the odd prime p they are equal.

4. Applying (2) with $a = -1$ we have

$$\left(\frac{-1}{p}\right) \equiv (-1)^{\frac{p-1}{2}} \bmod p.$$

   Again, since both ends are congruent to ± 1 modulo the odd prime p they are equal. Noting that when $p \equiv 1 \bmod 4$, $(p-1)/2$ is even gives the result.

□

Legendre symbol

Example. Is 12 a square mod 31?

Since $12 = 2^2 \cdot 3$ we write

$$\left(\frac{12}{31}\right) = \left(\frac{2}{31}\right)^2 \left(\frac{3}{31}\right).$$

Moreover

$$\left(\frac{3}{31}\right) \equiv 3^{15} \equiv -1 \bmod 31.$$

Hence 12 is not a square mod 31.

Extending Legendre symbol

In definition 3.22 the Legendre symbol is defined for primes. We would like to extend this definition to any odd integer n.

As a first attempt we define the symbol to be $+1$ if an integer a is a square and -1 otherwise.

Example. Is 6 a square mod 35?

Noting that $6 = 2 \cdot 3$ we need to consider whether 2 and 3 are squares mod 35. In fact neither of them is, since they are not squares mod 5.

Similarly 6 is not a square mod 7, and as such cannot be a square mod 35.

Consequently, none of 2, 3, and 6 is a square mod 35, implying the third property of proposition 3.23 to give $(-1) \cdot (-1) = -1$.↯

Jacobi symbol

To preserve the third property of the Legendre symbol (3.23) we define the Jacobi symbol as follows.

Definition (Jacobi symbol)

Given $n = \prod_i p_i^{e_i}$ an odd integer and a a non-zero integer coprime to n, we define the *Jacobi symbol* by

$$\left(\frac{a}{n}\right) = \prod_i \left(\frac{a}{p_i}\right)^{e_i},$$

where each of the $\left(\frac{a}{p_i}\right)$ is a Legendre symbol.

Jacobi symbol

Remark.

- When n is prime the Jacobi symbol reduces to the Legendre symbol
- Let $n = 135 = 3^3 \cdot 5$. Then
$$\left(\frac{2}{135}\right) = \left(\frac{2}{3}\right)^3 \left(\frac{2}{5}\right) = (-1)^3(-1) = 1.$$
However 2 is not a square mod 135 since it is not a square mod 5.
Hence a value of $+1$ for the Jacobi symbol does not imply that an integer is a square mod n.

Jacobi symbol

Proposition

Let n be an odd integer.

1. If $a \equiv b \bmod n$ and $\gcd(a,n) = 1$, then $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$.
2. If $\gcd(ab,n) = 1$ then $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$.
3. $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$.
4. $\left(\frac{2}{n}\right) = \begin{cases} +1 \text{ if } n \equiv 1 \text{ or } 7 \bmod 8 \\ -1 \text{ if } n \equiv 3 \text{ or } 5 \bmod 8 \end{cases}$
5. If m and n are odd coprime positive integers, then
$$\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right) & \text{if } m \equiv n \equiv 3 \bmod 4 \\ +\left(\frac{n}{m}\right) & \text{otherwise} \end{cases}$$

Jacobi symbol

Example. Calculate $\left(\frac{4567}{12345}\right)$.

$$
\begin{aligned}
\left(\frac{4567}{12345}\right) &= +\left(\frac{12345}{4567}\right) && \text{by (5), since } 12345 \equiv 1 \bmod 4 \\
&= +\left(\frac{3211}{4567}\right) && \text{by (1), since } 12345 \equiv 3211 \bmod 4567 \\
&= -\left(\frac{1356}{3211}\right) && \text{by (5) and (1)} \\
&= -\left(\frac{2}{3211}\right)^2\left(\frac{339}{3211}\right) && \text{by (2)} \\
&= +\left(\frac{3211}{339}\right) && \text{since } (\pm 1)^2 = 1 \text{ and by (5)} \\
&= +\left(\frac{2}{339}\right)^5\left(\frac{5}{339}\right) && \text{by (1) and since } 2^5 \cdot 5 = 160 \equiv 3211 \bmod 339 \\
&= -\left(\frac{2}{5}\right)^2 && \text{by (4), (5), and (2)} \\
&= -1
\end{aligned}
$$

Jacobi symbol

Remark.

- In proposition 3.29 the fifth point is called the *quadratic reciprocity law*. When m and n are primes it relates the question of m being a square mod n to the one of n being a square mod m.
- Let n be the product of two primes p and q and a be an integer. If $\left(\frac{a}{n}\right) = -1$, then a is not a square mod n. What can be concluded if $\left(\frac{a}{n}\right) = +1$?

  As $\left(\frac{a}{n}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{q}\right)$, either

  $$\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1 \text{ or } \left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = +1.$$

  In the first case a is not a square mod p and as such cannot be a square mod n, while in the second case a is a square.

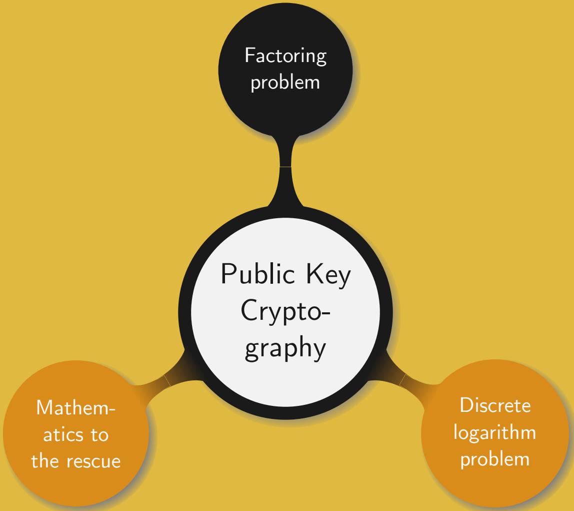
The Quadratic Residuosity Problem

From the previous remark (3.31) we see that if $\left(\frac{a}{n}\right) = +1$ then a can be either a square or a non-square. Deciding which one holds is known as the *Quadratic Residuosity Problem*, loosely introduced in problem 2.12.

Problem (Quadratic Residuosity (QR))

Let $n = pq$ be the product of two primes. Let y be an integer such that $\left(\frac{y}{n}\right) = 1$. Determining whether or not y is a square modulo n is called the *Quadratic Residuosity Problem*.

Chapter organisation



From mathematics to cryptography

From the previous mathematical discussions in chapters 1 and 3 we know that given two primes p and q, it is easy to compute their product n as well as $\varphi(n)$ (proposition 3.12).

Then if an integer e, coprime to $\varphi(n)$, is chosen, it suffices to run the extended Euclidean algorithm (1.32) in order to determine the integer d such that $ed \equiv 1 \bmod \varphi(n)$.

Therefore given e and n it is possible to compute $c \equiv m^e \bmod n$ for any integer m. Then computing $c^d \bmod n$ yields m since

$$c^d \equiv (m^e)^d \equiv m^{ed \bmod \varphi(n)} \equiv m \bmod n.$$

The goal is now to use this mathematical setup in order to build a trapdoor one-way function and design a public key cryptosystem.

Toward a new cryptosystem

Intuition:

- Generate p and q, then compute n and $\varphi(n)$
- Choose e coprime to $\varphi(n)$ and determine d
- Anybody can encrypt: n and e are public
- Only one person can decrypt: d is secret

Questions:

- How to effectively define the cryptosystem?
- Can the modular exponentiations to encrypt and decrypt be efficiently computed?
- How to efficiently generate p and q?
- How secure is it?

This cryptosystem, named RSA after its inventors Rivest, Shamir and Adleman, is the most popular public key cryptosystem.

The RSA cryptosystem

Initial setup:

- p, q two primes
- $n = pq$ and $\varphi(n)$
- e, d such that $ed \equiv 1 \bmod \varphi(n)$

- The n from Bob
- The e from Bob

Encryption:

- A message m
- Send $c \equiv m^e \bmod n$ to Bob

Decryption:

- Receive c from Alice or Eve
- Compute $m \equiv c^d \bmod n$

The RSA cryptosystem

Example.

1. Bob generates:
   - $p = 101$ and $q = 113$
   - $n = 11413$ and $\varphi(n) = 11200$
   - Select a random e and compute both its $gcd(e, \varphi(n))$ and $d = e^{-1}$ using the Extended Euclidian Algorithm (1.32). If the gcd is not one select a different e and retry. Let e be 3533, then $gcd(e, \varphi(n)) = 1$ and $d = 6597$.
2. Bob publishes n and e
3. Anybody can use those parameters to encrypt a message and send it to Bob. In particular Alice or Eve can encrypt 9726:

$$9726^{3533} \equiv 5761 \bmod 11413.$$

4. On receiving the message Bob computes

$$5761^{6597} \equiv 9726 \bmod 11413.$$

Modular exponentiation

The modular exponentiations required to encrypt and decrypt the message can be done efficiently in $\mathcal{O}((\log n)^2 \log d)$ bit operations, using the following algorithm.

Algorithm. (*Square and multiply*)

```
Input   : $m$ an integer, $d = (d_{k-1} \dots d_0)_2$ and $n$ two positive integers
Output: $x = m^d \bmod n$
power ← 1;
for $i \leftarrow k-1$ to 0 do
    $power \leftarrow (power \cdot power) \bmod n$;
    if $d_i = 1$ then $power \leftarrow (m \cdot power) \bmod n$;
end for
return power
```

Modular exponentiation

Example. Calculate 9726^{3533} mod 11413.

We run the previous algorithm with: $m = 9726$, n=11413 and $d = 3533 = (110111001101)_2$.

i	d_i	*power* mod 11413	i	d_i	*power* mod 11423
11	1	$1^2 \cdot 9726 \equiv 9726$	5	0	$7783^2 \equiv 6298$
10	1	$9726^2 \cdot 9726 \equiv 2659$	4	0	$6298^2 \equiv 4629$
9	0	$2659^2 \equiv 5634$	3	1	$4629^2 \cdot 9726 \equiv 10185$
8	1	$5634^2 \cdot 9726 \equiv 9167$	2	1	$10185^2 \cdot 9726 \equiv 105$
7	1	$9167^2 \cdot 9726 \equiv 4958$	1	0	$105^2 \equiv 11025$
6	1	$4958^2 \cdot 9726 \equiv 7783$	0	1	$11025^2 \cdot 9726 \equiv 5761$

Faster decryption

Two useful optimizations to the decryption can be applied. The first and most obvious consists in saving $d \bmod \varphi(n)$ such that it is not recomputed at each decryption.

The second idea consists in using the CRT (2.17, 3.11) to speed up the computation. Instead of storing $d \bmod \varphi(n)$ one can save $d \bmod (p-1)$ as well as $d \bmod (q-1)$, recover the "two sub-messages" in $\mathbb{Z}/p\mathbb{Z}$ and $\mathbb{Z}/q\mathbb{Z}$, and combine them over $\mathbb{Z}/n\mathbb{Z}$.

Example. Let $p = 11$, $q = 23$ and $e = 3$. Then $n = 253$, $\varphi(n) = 220$ and $d = 147$. To encrypt $m = 57$ we compute $c = 57^3 \equiv 250 \bmod 253$.

Instead of computing $m \equiv 250^{147} \bmod 253$ we do

$$\begin{cases} 250^{147 \bmod 10} \equiv & 8^7 \equiv & 2 \bmod 11 \\ 250^{147 \bmod 22} \equiv & 20^{15} \equiv & 11 \bmod 23. \end{cases}$$

Faster decryption

It now suffices to combine the results mod p and q into a single result mod n.

Bézout's identity gives $(-2) \cdot 11 + 1 \cdot 23 = 1$. Therefore 1_p is mapped into 23 mod 253 and 1_q into $-22 \equiv 231$ mod 253.

Hence,

$$\begin{aligned} (2, 11) &= 2 \cdot 1_p + 11 \cdot 1_q \\ &= 2 \ \cdot \ 23 + 11 \ \cdot \ 231 \bmod 253 \\ &\equiv 2587 \bmod 253 \\ &\equiv 57 \bmod 253. \end{aligned}$$

And the plaintext is recovered.

Generating primes

Which strategy to choose:

- Generate a random integer, pick the next prime
- Generate random integers until one of them is prime

Remark.

- The prime number theorem states that in the range 1-n approximately $n/\ln n$ integers are prime. As we will discuss later, the primes p and q are expected to be about 1024 bits long. Therefore the probability for a random integer between 1 and 2^{1024} to be prime is $1/\ln 2^{1024} \approx 1/710$.
- Although a deterministic polynomial time algorithm exists for primality testing (AKS), Monte Carlos algorithms, which are much faster solutions, are often used in practice.

The Solovay-Strassen primality test

From proposition 3.23 we know that if n is prime then for any $a \not\equiv 0 \bmod n$,

$$\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \bmod n. \tag{3.1}$$

Unfortunately there exist some integers a for which it is also true although n is not prime. It is therefore impossible to derive a deterministic algorithm from this proposition.

On the other hand, we can observe the following property. Let n be composite and $A = \left\{a \; / \; gcd(a, n) = 1 \text{ and } (3.1) \text{ holds}\right\}$. Since n is composite there exists an integer b such that $gcd(b, n) = 1$ and $\left(\frac{b}{n}\right) \not\equiv b^{(n-1)/2}$. For any $a \in A$ we have

$$(ab)^{\frac{n-1}{2}} = a^{\frac{n-1}{2}} b^{\frac{n-1}{2}} = \left(\frac{a}{n}\right) b^{\frac{n-1}{2}} \not\equiv \left(\frac{a}{n}\right) \left(\frac{b}{n}\right) \bmod n.$$

Hence, for any $a \in A$ there is an element coprime to n that does not belong to A. It is then possible to construct a Monte Carlo Algorithm that determines whether or not n is prime.

The Solovay-Strassen primality test

The following algorithm requires $\mathcal{O}(k(\log n)^3)$ operations to test the primality of n, k being the number of random elements generated for the test.

Algorithm. (*Solovay-Strassen*)

```
Input  : $n$ an integer, and $k$ the number of tests to run
Output: $n$ is composite or probably prime
for $i \leftarrow 1$ to $k$ do
    $a \leftarrow \mathrm{rand}(2, n-2)$;
    if $\gcd(a, n) \neq 1$ then return $n$ is composite;
    $x \leftarrow \left(\frac{a}{n}\right)$;
    $y \leftarrow a^{(n-1)/2} \bmod n$;
    if $x \not\equiv y \bmod n$ then return $n$ is composite;
end for
return $n$ is probably prime
```

The Miller-Rabin primality test

The Miller-Rabin test is a Monte Carlo Algorithm that determines whether or not an integer is prime.

Let $n \in \mathbb{N}$ be an odd integer. Then $n-1 = 2^s m$, where s is an integer and m is odd. The integer n passes the *Miller-Rabin test to base a* if either

$$a^m \equiv 1 \bmod n \qquad \text{or} \qquad a^{2^j m} \equiv -1 \bmod n$$

for some j with $0 \leq j \leq s-1$.

To see it, observe that if n is prime then $x^2 \equiv 1 \bmod n$ has only two solutions: $+1$ and -1. Moreover Fermat's little theorem (2.13) applies and $a^{n-1} \equiv 1 \bmod n$.

Therefore taking the square root of a^{n-1} yields 1 or -1. On -1 the second congruence holds. If this is 1 then the square root can be taken again until it is either -1 or only m is left. Hence one of the two congruences holds.

The Miller-Rabin primality test

Finally the contrapositive states that if neither of the congruences holds then n is composite.

Noticing the two following points we can now derive a probabilistic algorithm which returns whether an integer is composite or probably prime.

- If n is prime and $1 < a < n$, then n passes Miller's test to base a.
- If n is composite, then there are fewer than $n/4$ bases a with $1 < a < n$ such that n passes Miller's test to base a.

The Miller-Rabin primality test

The Monte Carlo algorithm now randomly selects k bases a and performs the Miller-Rabin test.

- If n fails the test for any of the bases used, the algorithm will return “true” (n is composite).
- If n passes each test, the answer is still unknown. Nevertheless, the algorithm will return “false” (n is probably prime).

The probability that n is composite and still passes the test each of the k times is

$$p_k = \frac{1}{4^k}.$$

For instance if $k = 30$ tests are performed, $p_k < 10^{-18}$. It is almost certain that a number that the algorithm returns as prime actually is prime.

The Miller-Rabin primality test

Algorithm. (*Miller-Rabin*)

```
Input  : $n$ an odd integer, and $k$ the number of tests to run
Output : $n$ is composite or probably prime
$m \leftarrow (n-1)/2$; $s \leftarrow 1$;
while $2|m$ do  $m \leftarrow m/2$; $s \leftarrow s+1$;
for $i \leftarrow 1$ to $k$ do
    $a \leftarrow \mathrm{rand}(2, n-2)$;
    if $\gcd(a, n) \neq 1$ then return $n$ is composite;
    $a \leftarrow a^m \bmod n$;
    if $a = \pm 1$ then continue;
    for $j \leftarrow 1$ to $s-1$ do
        $a \leftarrow a^2 \bmod n$;
        if $a \equiv 1 \bmod n$ then return $n$ is composite;
        if $a \equiv -1 \bmod n$ then b← 1 ; break;
    end for
    if $b$=$1$ then continue else return $n$ is composite;
end for
return $n$ is probably prime
```

Testing RSA security

The last question that remains to be answered is related to the security of RSA. The RSA cryptosystem can be viewed as having three secret parameters: p, q and d.

If n and $\varphi(n)$ are known, then p and q can be efficiently recovered.
Note that

$$n - \varphi(n) + 1 = pq - (p-1)(q-1) + 1 = p + q.$$

Since we know pq and $p+q$, p and q are the roots of the quadratic equation $X^2 - (n - \varphi(n) + 1)X + n$. Hence

$$p, q = \frac{n - \varphi(n) + 1 \pm \sqrt{(n - \varphi(n) + 1)^2 - 4n}}{2}.$$

Said otherwise, if $\varphi(n)$ can be computed then n can be factorised. Since factorizing n is believed to be hard there should be no way of efficiently compute $\varphi(n)$.

Testing RSA security

Suppose that both e and d are known. Then n can be efficiently factorised.

Since $de \equiv 1 \bmod \varphi(n)$, for any a coprime to n, $a^{de-1} \equiv 1 \bmod n$.

The idea is now to select some random a coprime to n, and apply the strategy described on slide 3.19. Note that if a and n are not coprime then $\gcd(a, n)$ is a factor of n. Therefore lets assume their gcd to be 1.

We start by writing $ed - 1$ as $2^s m$ and then define $b_0 = a^m$ and $b_{i+1} \equiv b_i^2 \bmod n$. As we expect to find the order of a we want $b_{i+1} \equiv 1 \bmod n$, while $b_i \not\equiv 1 \bmod n$. Moreover if $b_i \equiv -1 \bmod n$ then the factors are trivial. Therefore our aim is to find an a such that $b_i \not\equiv \pm 1$ and $b_{i+1} \equiv 1 \bmod n$. In this case, $\gcd(b_i - 1, n)$ and $\gcd(b_i + 1, n)$ are non-trivial factors of n.

Hence finding d should be hard since it allows to factorize n.

The RSA problem

From the previous discussion it appears that the RSA cryptosystem relies on the hardness of factoring large composite integers. But more precisely it is the hardness of determining $\varphi(n)$ when only n is known without its prime decomposition. The RSA problem can be formally stated as follows.

Problem (RSA problem)

Let n be a large integer and $e > 0$ be coprime to $\varphi(n)$. Given y in $\mathrm{U}(\mathbb{Z}/n\mathbb{Z})$, compute $y^{1/e} \bmod n$, i.e. find x such that $x^e \equiv y \bmod n$.

Although factoring $\varphi(n)$ or computing d solves the RSA problem there is no proof that no other way of solving it exists. Therefore it cannot be concluded that the RSA problem is as hard as factoring. Indeed it may be that the RSA problem can be solved in polynomial time even though the factoring problem cannot.

Complexity of a few factorization algorithms for n a k-bit integer:

Algorithm	Complexity
Trial division	$\mathcal{O}\left(2^{k/2}/k)\right)$
Pollard-ρ	$\mathcal{O}\left(\sqrt[4]{n}\right)$
ECM	$L_p\left[1/2, \sqrt{2}\right]$
GNFS	$L_n\left[1/3, \sqrt[3]{64/9}\right]$

The $L_n(\alpha, c)$ function is defined by

$$L_n(\alpha, c) = e^{\left(c+o(1)\right)\left((\ln n)^{\alpha}(\ln \ln n)^{1-\alpha}\right)}.$$

Factoring integers

Given a large random integer N, the probability for N to be divisible by 2 is 1/2; by 3, 1/3; by 5, 1/5 etc. One can deduce that about 88% of integers have a factor smaller than 100 and 92% a factor smaller than 1000.

Therefore, despite its exponential complexity, trial division is used in almost all factoring programs. All the small factors are first removed before more advanced strategies are employed to totally factorize N.

In practice, trial division is implemented through a large table containing all the primes, or alternatively the difference between two consecutive primes, up to 10 million. Then even for a 1000 digit long integer it only takes a few seconds to perform all the trial divisions and ensure that N is free of any small factor.

Another simple idea in order to remove small factors consists in computing $\gcd(n, P)$ where P corresponds to the product of all the prime numbers below a given bound B. Compared to trial division this strategy seems appealing since computing a gcd can be done in polynomial time. In practice, this method is much more efficient when considering primes below 1000 but it becomes extremely slow when checking prime factors of size around one million.

In the first case the product of all the primes is about 1,400 bits while in the second case it is approximately 1,500,000. Computing the gcd of an integer around 2^{3072} and P, then takes much longer.

This simple example highlights how practical cases can highly diverge from the theoretical asymptotic analysis.

Pollard's Rho Algorithm

We now introduce an example of a more sophisticated factoring scheme. It is asymptotically faster than trial factorization and can be used when small numbers have been eliminated as possible factors.

Let n be a composite integer with an unknown prime factor $p \leq \sqrt{n}$. Define the function

$$f : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}, \qquad f(x) = x^2 + 1 \bmod n$$

(other functions $f : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ can be used). We now recursively define a sequence (x_k) by

$$x_0 = 2, \qquad x_{k+1} = f(x_k), \qquad k \in \mathbb{N}.$$

Since there are exactly n elements in $\mathbb{Z}/n\mathbb{Z}$, the sequence must at some point produce a repeated value and enter a cycle.

Pollard's Rho Algorithm

We then **hope** that the cycle contains two or more elements with the same remainder modulo p, i.e., that we can find x_i and x_j, $i \neq j$, in the cycle such that

$$x_i \equiv x_j \bmod p.$$

If that is the case, then $x_i - x_j$ is divisible by p and $\gcd(x_i - x_j, n)$ gives a factor of n.

In summary, when testing all of the x_i and x_j of the cycle, this GCD can evaluate as follows:

$$\gcd(x_i - x_j, n) = \begin{cases} n & \text{if } x_i = x_j, \\ 1 & \text{if } x_i \not\equiv x_j \bmod p \text{ for all factors } p \text{ of } n, \\ t & \text{if } x_i \equiv x_j \bmod p, \text{ where } p \mid t \text{ and } t \mid n. \end{cases}$$

Pollard's Rho Algorithm

The algorithm now uses the following method to evaluate pairs x_i, x_j in the cycle: two sequences (x_k) and (y_k) are defined,

$$x_0 = 2, \quad x_{k+1} = f(x_k) \qquad \text{and} \qquad y_0 = 2, \quad y_{k+1} = f(f(y_k)).$$

The sequences (x_k) traverses the cycle normally, while the sequence (y_k) traverses the cycle in double steps. This is intended to be an efficient manner of generating "random" pairs (x_i, x_j). For each pair, $\gcd(x_i - x_j, n)$ is evaluated.

Example. Suppose that we want to factor the number $n = 8051$. We start with $x_0 = 2$ and set $x_{k+1} = x_k^2 + 1 \bmod 8051$. We obtain the sequence

$$(x_i) = (2,\ 5,\ 26,\ 677,\ \mathbf{7474},\ 2839,\ 871,\ 1848,\ 1481,\ 3490,\ 6989,$$
$$705,\ 5915,\ 5631,\ 3324,\ 3005,\ 4855,\ 5749,\ 1647,\ \mathbf{7474},\ 2839, \ldots$$

and we have found a cycle starting at $x_4 = 7474$.

Pollard's Rho Algorithm

In practice, we simply generate the sequences (x_k) and (y_k) and evaluate the GCDs:

x_k	2	5	26	677	7474	2839
y_k	2	26	7474	871	1481	6989
$\gcd(x_k - y_k, n)$	8051	1	1	97	1	83
x_k	871	1848	1481	3490	6989	705
y_k	5915	3324	4855	1647	2839	1848
$\gcd(x_k - y_k, n)$	97	1	1	97	83	1
x_k	5915	5631	3324	3005	4855	5749
y_k	3490	705	5631	3005	5749	7474
$\gcd(x_k - y_k, n)$	97	1	1	8051	1	1

Even before the cycle is entered by (x_k), a factor $p = 97$ of $n = 8051$ is found.

Pollard’s Rho Algorithm

Algorithm. (*Pollard-ρ – Factorization*)

Input: n, a composite integer, $f(x) = x^2 + 1 \bmod n$.

Output: d a non-trivial factor of n, or failure.

```
$a \leftarrow 2$; $b \leftarrow 2$;
repeat
    $a \leftarrow f(a)$; $b \leftarrow f(f(b))$;
    $d \leftarrow \gcd(a - b, n)$;
until $d \neq 1$;
if  $d = n$ then
    return failure
else
    return $d$
end if
```

Complexity of Pollard's Rho Algorithm

The Pollard Rho algorithm derives its name from the shape of the sequence (x_k). At some point in the sequence, $x_k \equiv x_{k+T} \bmod p$ for some $T > 0$ and the sequence can be represented as a cycle from that point onwards - this is the circle of the letter ρ. The sequence terms $x_0, x_1, \dots x_{k-1}$ then form the "tail" of ρ.

Remark.

- The role of f is to "randomly" select numbers in $\mathbb{Z}/n\mathbb{Z}$. Its precise form is not essential, but it should be a polynomial for
  $$f(f(x) \bmod n) \bmod n = f(f(x)) \bmod n$$
  when calculating the sequence (y_k).
- It is not guaranteed that the Pollard Rho algorithm actually will be successful - it could happen that all of the x_i in the cycle have distinct remainders modulo p. In that case, a different starting point x_0 should be chosen and the algorithm run once more.

Complexity of Pollard's Rho Algorithm

We now attempt to make a rough estimate of the average time complexity of the algorithm. Let us ignore the specifics and suppose that the algorithm simply selects random numbers $x_i, x_j \in \mathbb{Z}/n\mathbb{Z}$ for comparison of their remainders.

Suppose that any given number between 0 and n has an equal probability $1/p$ of having a remainder m modulo p, $0 \le m < p$:

$$P[x_k \bmod p = m] = \frac{1}{p} \qquad \text{for all } m = 0, \dots, p-1 \text{ and all } k \in \mathbb{N}.$$

Suppose that for any two x_i and x_j, $i \neq j$, these probabilities are independent. Then the probability that x_i and x_j have different remainders is

$$P[x_i \not\equiv x_j \bmod p] = \frac{p-1}{p},$$

Complexity of Pollard's Rho Algorithm

Suppose that $x_0, \ldots, x_{k-1}$ have distinct remainders modulo p, then the probability of x_k to have a remainder different from $x_0, \ldots, x_{k-1}$ is $\frac{p-k}{p}$.

Then (assuming the independence of the value of the remainder) the probability that a group of k numbers has distinct remainders mod p is

$$P_k := P[x_i \not\equiv x_j \bmod p,\ 0 \leq i < j \leq k]$$
$$= \prod_{l=0}^{k} \frac{p-l}{p} = \frac{p!}{(p-k-1)!p^k}.$$

It can be shown that $1 - P_k < 1/2$, if $k > 1.177\sqrt{p}$.

This indicates that the average-case complexity of Pollard's Rho algorithm should be

$$\mathcal{O}(\sqrt{p}) = \mathcal{O}(\sqrt[4]{n}).$$

Factoring the RSA modulus

In the RSA case n is known to be product of two large primes. Therefore the algorithm of choice is the GNFS. Then applying the strategy described on slide 1.51 we set

$$2^{128} = e^{\sqrt[3]{\frac{64}{9}}(\ln n)^{1/3}(\ln \ln n)^{2/3}},$$

which gives approximately $n = 7.65 \cdot 10^{763}$. In terms of bit length, n should be about 2500 bits long. In practice, this is rounded up to 3072 bits (2048+1024), and a bit length of 3072 is considered "secure" as it corresponds to 128 bits security.

As of 2014, the largest product of two large primes officially factorized occurred in the breaking of RSA-768 (a 768 bits RSA modulus):

1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202199786
469389956474942774063845925192557326303453731548268507917026122142913461670429214311602221240479
274737794080665351419597459856902143413

$=$

3347807169895689878604416984821269081770479498371376856891243138898288379387800228761471165253174
3087737814467999489

$\times$

3674604366679959042824463379962795263227915816434308764267603228381573966651127923337341714339681
0270092798736308917

Attacks on RSA

Strategy for short messages and small e:

- A message $m < n^{1/e}$
- The ciphertext is $c \equiv m^e \bmod n$
- The encryption does not require any modular reduction
- Over the integers c is also m^e
- Solve $c^{1/e}$ over the integers recovers m

Typical use: encrypt a 128 bits long secret key using RSA

Attacks on RSA

Strategy for short messages:

- A message m of less than about 10^{17}
- The ciphertext is $c = m^e \bmod n$
- Compute and store in a table $cx^{-e} \bmod n$, for all $1 \leq x \leq 10^9$
- Compute $y^e \bmod n$, for all $1 \leq y \leq 10^9$, and test for a collision
- If a collision is found then $c = (xy)^e \bmod n$
- If $m \leq 10^{17}$ it is likely that such x and y exist

Attacks on RSA

Strategy for small e:

- A message m sent to $i \geq e$ persons using the keys $\langle n_i, e \rangle$
- If $\gcd(n_k, n_j) \neq 1$ then one of the n_i can be factorized
- Otherwise set $n = \prod_i n_i$
- Use the CRT over all the ciphertext $c_i = m^e \bmod n_i$, to compute $c \equiv m^e \bmod n$
- As $m < \min_i(n_i)$ and $i \geq e$, then $m^e < n$
- Finally $m = c^{1/e}$ can be computed in $\mathbb{N}$

A more secure RSA

Current version of RSA:

- Referred to as “textbook RSA”
- Insecure but simple to understand

PKCS #1 v1.5:

- Randomly pad the message before encrypting
- Assumed to be CPA secure
- No proof based on the RSA problem
- Not CCA secure
- Still widely used for backward-compatibility reasons

Toward a secure RSA

Optimal Asymmetric Encryption Padding (RSA-OAEP):

- Due to Bellare and Rogaway
- Standardized as PKCS #1 v2
- Similar to feistel network in the construction
- Proved to be CCA secure

Notations:

- Concatenation of two bit strings a and b: $a \| b$
- Repetition of a bit b, d times: b^d
- Random oracle are informaly defined on slide 2.21

RSA-OAEP

Generate: set $p, q, n, e,$ and d, and choose two random oracles $G : \{0,1\}^{2l} \to \{0,1\}^{2l}$ and $H : \{0,1\}^{2l} \to \{0,1\}^{2l}$, for $l \in \mathbb{N}$

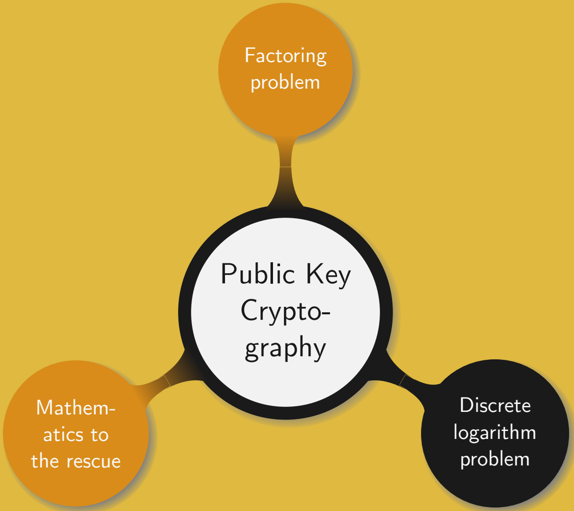
Encrypt: a message $m \in \{0,1\}^{l}$

1. Pick a random $r \in \{0,1\}^{2l}$
2. Set $m' = m \| 0^{l}$
3. Compute $m'' = \left(G(r) \oplus m'\right) \| \left(r \oplus H\left(G(r) \oplus m'\right)\right)$
4. Define the ciphertext as $c = (m'')^{e} \bmod n$.

Decrypt: a ciphertext c

1. Compute $m'' = c^{d} \bmod n$.
2. Parse m'' as $m_1 \| m_2$, with m_1, m_2 of size $2l$
3. Recover $r = H(m_1) \oplus m_2$
4. Recover $m' = m_1 \oplus G(r)$
5. If the l last bits are not 0^{l} output error ↯
6. Otherwise output m, the first l bits of m'

Chapter organisation


Factoring problem
Public Key Crypto- graphy
Mathem- atics to the rescue
Discrete logarithm problem

The Discrete Logarithm Problem

After investigating the RSA problem (3.51) we now turn our attention to another hard problem from number theory.

Problem (Discrete Logarithm Problem (DLP))

Let $\mathbb{F}_q$ be a finite field, with $q = p^n$, for a positive integer n. Given α a generator of G, a subgroup of $\mathbb{F}_q^*$, and $\beta \in G$, find x such that $\beta = \alpha^x$ in $\mathbb{F}_q$.

Note that x is unique only up to congruence mod $|G|$, therefore x is usually restricted to $0 \leq x < \mathrm{ord}_{\mathbb{F}_q^*}(\alpha)$.

Example. For $p = 13$ and $n = 1$ the field of concern is $\mathbb{Z}/13\mathbb{Z}$. The multiplicative group $\mathrm{U}(\mathbb{Z}/13\mathbb{Z})$ has order 12 and as such has a subgroup of order 6 (Lagrange's theorem (3.13)).

From example 3.10, 2 has order 12 and is a generator of $\mathrm{U}(\mathbb{Z}/13\mathbb{Z})$. Therefore 4 generates a subgroup of order 6, namely

$$G = \{4, 3, 12, 9, 10, 1\}.$$

Example of DLP in G: find x such that $4^x \equiv 9 \bmod 13$.
Clearly 4 is a solution, but also 10, 16, 22...However, restricting x to the range $0 - 6$ makes it unique.

Pollard's Rho Algorithm

In the previous section Pollard's Rho algorithm was investigated as a way to solve the factorization problem. Since Factorization and Discrete Logarithm have much in common Pollard's Rho algorithm can be adjusted to this new context. We now present its details.

Let α be a generator of a group G of prime order p. Any element of G can be written $\alpha^a\beta^b$ for some $a, b \in \mathbb{N}$ and $\beta \in G$.

Assuming two integers x and y such that $x \equiv y \bmod p$ can be found, then there exist a_1, b_1 such that $x \bmod p$ can be written $\alpha^{a_1}\beta^{b_1}$, and a_2, b_2 such that $\alpha^{a_2}\beta^{b_2} \equiv y \bmod p$.

Rewriting $x \equiv y \bmod p$ as $\alpha^{a_1}\beta^{b_1} \equiv \alpha^{a_2}\beta^{b_2} \bmod p$ yields

$$\beta^{b_1-b_2} \equiv \alpha^{a_2-a_1} \bmod p.$$

Taking the $\log_\alpha$ on both sides leads to

$$(b_1 - b_2)\log_\alpha \beta = a_2 - a_1 \bmod p.$$

Pollard's Rho Algorithm

As long as $p \nmid (b_1 - b_2)$ we get

$$\log_\alpha \beta = \frac{a_2 - a_1}{b_1 - b_2}.$$

Therefore the goal of Pollard's Rho algorithm is to find x and y with $x \equiv y \bmod p$. This is achieved by considering three partitions S_1, S_2 and S_3 of G of approximately the same size, based on an easily testable property, and defining three functions f, g and h.

$$f(x) = \begin{cases} \beta x & x \in S_1 \\ x^2 & x \in S_2 \\ \alpha x & x \in S_3 \end{cases}$$

$$g(a, x) = \begin{cases} a & x \in S_1 \\ 2a \bmod p & x \in S_2 \\ a + 1 \bmod p & x \in S_3 \end{cases} \qquad h(b, x) = \begin{cases} b + 1 \bmod p & x \in S_1 \\ 2b \bmod p & x \in S_2 \\ b & x \in S_3 \end{cases}$$

Pollard's Rho Algorithm

Starting with two elements $x = y = 1$, f is iteratively applied, once to x and twice to y. Since G is a cyclic group repeatedly applying f to x and y will result in a collision at some stage.

The function f, g and h are defined such as the progress of x and y appears "random", while y goes twice as fast as x. Then by the birthday paradox (4.11) a collision can be expected in time $\sqrt{p}$, since p is the order of the group G.

Remark. The cyclic group G is taken as generic and no further assumption is made. This means that Pollard's Rho method applies to any group G of prime order p.

Pollard's Rho Algorithm

Algorithm. (*Pollard-ρ – Discrete Logarithm*)

Input : α a generator of G, a group of prime order p and $\beta \in G$, f, g and h three functions.

Output: $\log_\alpha \beta$, or failure.

```
$a_1 \leftarrow 0$; $b_1 \leftarrow 0$; $x \leftarrow 1$; $a_2 \leftarrow 0$; $b_2 \leftarrow 0$; $y \leftarrow 1$;
repeat
    $a_1 \leftarrow g(a_1, x)$; $b_1 \leftarrow h(b_1, x)$;
    $x \leftarrow f(x)$;
    $a_2 \leftarrow g(g(a_2, y), f(y))$; $b_2 \leftarrow h(h(b_2, y), f(y))$;
    $y \leftarrow f(f(y))$;
until $x \equiv y \bmod p$;
$r \leftarrow b_1 - b_2$;
if $r \neq 0$ then return $r^{-1}(a_2 - a_1) \bmod p$;
else return failed;
```

Pollard's Rho Algorithm

Example. Let $\alpha = 2$ be a generator of G, the subgroup of order 191 of $\mathbb{Z}_{383}^*$. Let $\beta = 228$. Partition G into $S_1 = \{x \in G | x \equiv 1 \bmod 3\}$, $S_2 = \{x \in G | x \equiv 0 \bmod 3\}$ and $S_3 = \{x \in G | x \equiv 2 \bmod 3\}$.

x	228	279	92	184	205	14	28
a_1	0	0	0	1	1	1	2
b_1	1	2	4	4	5	6	6
y	279	184	14	256	304	121	144
a_2	0	1	1	2	3	6	12
b_2	2	4	6	7	8	18	38
x	256	152	304	372	121	12	**144**
a_1	2	2	3	3	6	6	12
b_1	7	8	8	9	18	19	38
y	235	72	14	256	304	121	**144**
a_2	48	48	96	97	98	5	10
b_2	152	154	118	119	120	51	104

Then compute $(38 - 104)^{-1}(10 - 12) \equiv 110 \bmod 191$. Hence in $\mathbb{Z}_{383}^*$ $\log_2(228) = 110$.

Polhig-Hellman Algorithm

Although slightly more advanced Polhig-Hellman Algorithm is interesting in the sense that it takes advantage of the structure of the prime p. In fact it was noticed that if $p-1$, the order of the multiplicative group of $\mathbb{Z}_p$, is featuring many small primes then the Discrete Logarithm Problem can be solved using the Chinese Remainder Theorem.

Let $p-1 = q_1^{e_1} q_2^{e_2} \dots q_r^{e_r}$, $r \in \mathbb{N}$. If $x = \log_\alpha \beta$ then it suffices to determine $x_i = x \bmod q_i^{e_i}$ for $1 \leq i \leq r$ and then use the Chinese Remainder Theorem in order to recover x. Therefore it only remains to compute all the x_i. This can be efficiently achieved at the cost of some mathematical technicalities.

More on the Discrete Logarithm Problem

Remark. A common mathematical strategy consists in transposing a difficult problem over a given structure into an easier one over a similar structure. Following this idea one could think of solving a hard Discrete Logarithm Problem into an isomorphic group and then map it back to the original group.

In particular since the Discrete Logarithm Problem is easy to solve in the additive group $\mathbb{Z}_n$, $n \in \mathbb{N}$, it is possible to map the multiplicative group of $\mathbb{Z}_p$ into the additive group $\mathbb{Z}_{p-1}$. Solving the problem in this simpler group and mapping back the solution to $\mathbb{Z}_p$ seems to be a very attractive solution.

The major problem with this approach is finding the map. In fact such a map would have to be built element by element, which would be time consuming. As a result this solution is not applicable in practice.

The DLP in cryptography

It is simple to see that the DLP is the inverse operation of the modular exponentiation, which can be very efficiently computed (3.38). However solving the DLP is not an easy task if the group is carefully chosen.

For instance as we will study in chapter 8, in groups having only a very basic algebraic structure the best algorithm available is the Pollard's Rho algorithm.

For more common groups over finite fields the best algorithms have a sub-exponential complexity similar to the one of the GNFS. Therefore in a cryptographic context, that is for the DLP to be intractable, the group is expected to have order larger than 2^{3072}.

We now present several cryptographic protocols based on the hardness of solving the DLP.

Diffie-Hellman key exchange

Alice and Bob publicly agree on some parameters:

G a group of order p

α a generator of G

Both Alice and Bob generate a random secret:

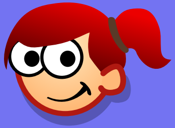
Choose a random element x in G

Choose a random element y in G

Alice and Bob send each other α^{secret}:

- x in G and α^y
- α^{yx}

- y in G and α^x
- α^{xy}



Diffie-Hellman problems

Clearly solving the DLP implies breaking the Diffie-Hellman key exchange protocol. However in order to determine α^{xy} it might not be necessary to solve the DLP, but only to solve the so called Computational Diffie-Hellman problem.

Problem (Diffie-Hellman problems)

Let G be a group of prime order p and α be a generator of G.

1. *Computational Diffie-Hellman (CDH):* given α^x and α^y, for some unknown integers x and y, compute α^{xy}.
2. *Decisional Diffie-Hellman (DDH):* given α^x and α^y, decide whether or not some $c \in G$ is equal to α^{xy}.

While solving the DLP implies solving the CDH problem, it is not known whether or not solving the CDH problem solves the DLP.

At present no method to solve CDH from DDH is known, and in fact in some groups DDH is efficiently solved while CDH remains hard.

Elgamal cryptosystem

Initial setup:

- G a group of prime order p
- α a generator of G
- x a secret integer

- G from Bob
- α from Bob
- $\beta \equiv \alpha^x \bmod p$ from Bob

Encryption:

- A message m
- $r \equiv \alpha^k \bmod p$ for some random integer k
- Compute $t \equiv \beta^k m \bmod p$
- Send $c = \langle r, t \rangle$ to Bob

Decryption:

- Receive c from Alice or Eve
- Compute $tr^{-x} \equiv \beta^k m \left(\alpha^k\right)^{-x} \equiv \left(\alpha^x\right)^k m\alpha^{-xk} \equiv m \bmod p$

Elgamal and CDH

Proposition

Solving the CDH problem is equivalent to breaking Elgamal.

Proof. Assume we can decrypt any Elgamal ciphertext and we are given α^a and α^b. First set β to α^a, i.e. a is the secret key, and r to α^b. Then choosing a non zero value for t we get

$$m \equiv tr^{-a} \equiv t\alpha^{-ab}.$$

Hence $tm^{-1} \equiv \alpha^{ab}$ mod p solves the CDH problem (3.82).

Conversely assume we can solve the CDH problem and are given the Elgamal ciphertext $\langle r, t\rangle$. Then we can determine α^{xk} mod p, and since m is $t\alpha^{-xk}$ we can recover the plaintext. □

CCA on Elgamal

Given an Elgamal ciphertext $c = \langle r, t \rangle$, the goal is to recover the plaintext corresponding to c by constructing another ciphertext $c' = \langle r', t' \rangle$ and getting it decrypted by a "decryption oracle".

Constructing a chosen ciphertext c'':

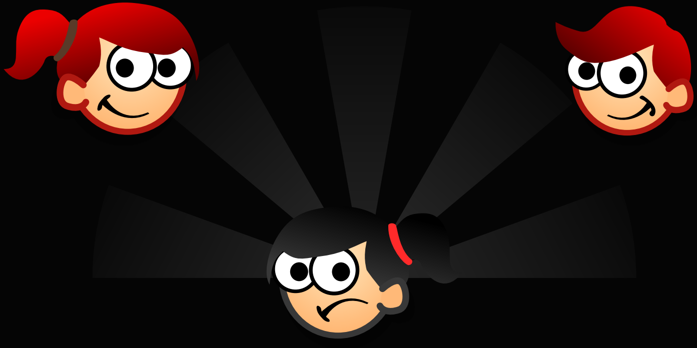
- By definition $c = \langle r, t \rangle = \langle \alpha^k, \beta^k m \rangle$
- For a message m', compute $c'' = \langle r\alpha^{k'}, t\beta^{k'} m' \rangle$
- Submitting c'' to the decryption oracle, m'' is returned

Recovering m from m'':

- In the group G, observe that c'':
  - $r\alpha^{k''} = \alpha^{k+k'} = \alpha^{k''}$
  - $t\beta^{k''} m' = \beta^k m \beta^{k'} m' = mm'\beta^{k+k'}$
- Then c'' is the ciphertext corresponding to $m'' = mm'$
- Finally compute $m = m'' m'^{-1}$

Key points

- What is the order of a group, of an element?
- How to efficiently perform primality testing?
- Which version of RSA is secure? Describe it
- When to use Diffie-Hellman key exchange protocol?
- Describe Elgamal



Thank you!