# MODERN CRYPTOGRAPHY FOR INFORMATION SECURITY APPING2: [Level I]

cryptoing2@gmail.com

MATHS FOR CRYPTO

# Cryptography for security

## *Numbers* : Some *sets* of *numbers*

1. The set of *natural* numbers : $\mathbb{N} = \{0, 1, 2, 3, 4, \cdots\}$
2. The set of *integers* numbers : $\mathbb{Z} = \{\cdots, -2, -1, 0, 1, 2, 3, \cdots\cdots\} = \{0, \pm 1, \pm 2, \pm 3, \cdots\}$
3. The set of *rational* numbers : $\mathbb{Q} = \{\frac{a}{b} : a, b(\neq 0) \in \mathbb{Z}\}$
4. The set of *real* numbers : $\mathbb{R}$
5. The set of *complex* numbers : $\mathbb{C}$
6. And the most important cryptographic set of numbers : the set of all *residue classes* modulo a positive integer $n$ : $\mathbb{Z}/n\mathbb{Z} = \mathbb{Z}_n = \{0, 1, 2, 3, \cdots n-1\}$.

## Binary Operations

A binary operation $\circ$ on a set $S$ is a rule that assigns to each ordered pair $(a, b)$ of elements of S, *i.e.* : $(a, b) \in S \times S$, a unique element of $S$.
Exemples :

- Ordinary addition (symbol : +) : on the set $\mathbb{N}$ or $\mathbb{Z}$ or $\mathbb{Z}_n$ or on an elliptic curve
- Ordinary multiplication (symbol : $*$ or $\times$ or .) : on the set $\mathbb{N}$ or $\mathbb{Z}$ or $\mathbb{Z}_n$.




## Hard Problems for cryptography

- Some "mathematical" problems are hard
- It means : No efficient polynomial time algorithm exists
- NP-hard problems : SAT (Satisfiability problem) and TSP (Traveling Salesman Problem) are such classical hard problems (there are NP-Hard)
- Is it a real problem ? No, because (most of) modern cryptography relies on these hard problems
- We *need* proven hard problems to construct resistant cryptographic algorithms
- Hard means : NP-hard (in an ideal world) but unfortunately practically a lot of cryptographic algorithms relie on problems just *computationally difficult* to solve

# Hard Problems for cryptography



## Computational hardness assumption [?]

*In cryptography, a major goal is to create cryptographic primitives with provable security. In some cases, cryptographic protocols are found to have information theoretic security ; the one-time pad is a common example. However, information theoretic security cannot always be achieved ; in such cases, cryptographers fall back to computational security. Roughly speaking, this means that these systems are secure assuming that any adversaries are computationally limited, as all adversaries are in practice. Because hardness of a problem is difficult to prove, in practice certain problems are "assumed" to be difficult.*

# Problems for cryptography



## Two widely "difficult" (hard) problems used in cryptography [?, ?, ?]

1. Let $n \in \mathbb{N}^*$ a large number : find a "non trivial' factor of $n$ : this is the *factorization problem*. Example : try to factorize 1000000100000000000001800000090000000000000081
2. Let $g \in G$, $(G, *)$ is a large group generated by $g$, with $y \in G$, find the least integer $x$ such that $y = g^x$ ; this is the *discrete logarithm problem* (DLP). Example : $G = \mathbb{Z}_p^*$, p is a large prime.

For these problems, it seems *computationally difficult* to solve them if parameters are well chosen. [[?] : . . . *there are no proofs that integer factorization is computationally difficult* . . . ].

# Cryptography for security : remainder



## Widely used (not so hard) problems [?, ?, ?]

1. Find a large prime p (1024 bits, 2048 etc.)
2. Find a large group $G = \mathbb{Z}_p$ and g a generator of $G$ (or sometimes of a large subgroup $H$ of $G$).
3. Find a a square root modulo a prime $p$, *i.e.* solve $y^2 = a \bmod p$.
4. How to generate pseudo-random numbers with a good *quality of randomness* ?

**For these problems, we can solve them for any reasonable parameters : these easy problems are common tools of any industrial cryptographic library.**

# Cryptography for security : remainder

**Find a a square root modulo a prime $p$, *i.e.* solve $y^2 = a \bmod p$ : quite easy !**

For example :

- If $p \equiv 3 \bmod 4$
- Compute $Y = \pm a^{(p+1)/4}$
- $Y$ is a solution of $y^2 = a \bmod p$
- A very Quick Proof :

$$Y^2 = (\pm a^{(p+1)/4})^2 = a^{(p+1)/2} = (a^{p-1}a^2)^{1/2} = (a^2)^{1/2} = a$$

# Cryptographic Problems



## Some cryptographic important problems

- Very Easy : How to compute $\gcd(a, b)$ ? With the *Euclidean Algorithm*. Fast algorithms are fortunaltely known.
- Quite Easy How to prove that a number is a prime ? [Primality Tests, which are probabilistic algorithms, like Miller-Rabin are very efficient]
- Quite Easy : How to generate quickly a prime of a given length (example : a 1024 bits prime number) ?

## Fermat's little theorem [?, ?, ?]

It's states that if p is a prime number, then for any integer $a$, (coprime to $p$ or not), then $a^p - a$ will be evenly divisible by $p$ :

$$a^p \equiv a \bmod p$$

## A variant :

If $p$ is a prime and $a$ is an integer (coprime to $p$) , then $a^{p-1} - 1$ will be evenly divisible by $p$ ; i.e. :

$$a^{p-1} \equiv 1 \bmod p.$$

## The (famous) Euler totient function [?, ?]

We can generalize the Fermat little Theorem with the Euler totient function $\varphi$ : let $n = p_1^{e_1} \cdots p_k^{e_k}$ the prime decomposition of $n$ then the Euler totient $\varphi(n)$ is defined by :

$$\varphi(n) = (p_1 - 1)p_1^{e_1 - 1} \cdots (p_k - 1)p_k^{e_k - 1} = n\Pi_{p|n}(1 - 1/p)$$

Then the Euler theorem states that we have :

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

Fermat's little theorem and Euler generalization is the basis for the Fermat primality test (not good) and efficient variants (Miller-Rabin) and explains why RSA works!!



## The SOCAT *false prime*

SOCAT has used/published the following "prime" number in a new release :

```
Q=12118534197456004471544151826551243891225082214
25169539094603250961222565925994212326221918582222
62115484159870425610109169500160661013673910956593
39344189581242129110469501186733850735505575616261
61195247553538583177814124651159600562207154177438
1205194074173
```

https ://www.cryptologie.net/article/329/socat-new-dh-modulus/ : *Socat did not work in FIPS mode because 1024 instead of 512 bit DH prime is required. Thanks to Zhigang Wang for reporting and sending a patch.*

## The SOCAT *false prime* : $2^{Q-1} \neq 1 \bmod Q$

Unfortunately, this number is **composed** : $Q = p\,q\,r$ with :

```
p=326657
q=21582445153831
r=171892717523409047992503847678444629219867260
491682070057566150434865341181706721817919900033
227746212266197628521590427693714652830107583165
679727701674379592403204645779667235160021527049
0908956854553701586832290013338111712436495754503419
```

You can verify that $Q$ is not prime just with $2^{Q-1} \bmod Q \neq 1$.

## Three important algebraic structures

1. Groups
2. Rings
3. Fields

All (almots all) cryptographic algorithms are defined on a group, a ring or a field of numbers but always **finite**.

# Plan: Introduction

## Groups : (Idendity=Neutral element)

A group, denoted $< G, \circ >$, $(G, \circ)$ or simply $G$, is a nonempty set $G$ of elements together with a binary operation $\circ$, such that the following axioms are satisfied :

1. *Closure* : $\forall a \in G, \forall b \in G, a \circ b \in G$.
2. *Existence of Identity* : there is a unique element $e \in G$ called the identity, such that $\forall a \in G, a \circ e = e \circ a = a$.
3. *Associativity* : $\forall a \in G, \forall b \in G, \forall c \in G, a \circ (b \circ c) = (a \circ b) \circ c$
4. *Existence of Inverse* : $\forall a \in G, \exists$ a unique $b \in G$ such $a \circ b = b \circ a = e$. This $b$ is called the inverse of $a$ nad we will write : $b = a^{-1}$ or $b = -a$.
5. *Commutativity* : A group $(G, \circ)$ is called commutative if it satisfies the following axiom : $\forall a \in G, \forall b \in G, a \circ b = b \circ a$.



## Commutative Groups

Some example and counter-examples :

- $(\mathbb{Z}, +)$ is a commutative group
- $(\mathbb{Z}, *)$ is **not** a group (example : 2 has no inverse in $\mathbb{Z}$)
- $(\mathbb{Q}, +)$ is a commutative group
- $(\mathbb{Q}^*, *)$ is a commutative group with $\mathbb{Q}^* = \mathbb{Q} - \{0\}$
- $(\mathbb{Z}_p, *)$ is a **finite** group if and only if $p$ is prime.
- $(\mathbb{Z}_n, +)$ is always a **finite** commutative group.
- Remark : An identity element is sometimes called a *neutral* element
- Remark : A commutative group is sometimes called an *abelian* group

# Plan: Introduction

**[?]**

A ring is a set $R$ equipped with two binary operations + and $\cdot$ satisfying the following three sets of axioms, called the ring axioms :

1. (R,+) is an abelian group (addition), meaning that
   - $(a + b) + c = a + (b + c)$ ∀ a, b, c in R (+ is associative).
   - $a + b = b + a$ ∀ a, b in R (+ is commutative).
   - There is an element 0 in R such that $a + 0 = a$ ∀ a in R (0 is the additive identity).
   - For each a in R there exists $?a$ in R such that $a + (?a) = 0$ ($-a$ is the additive inverse of a).
2. (R,*) is a monoid under multiplication, meaning that :
   - $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ ∀ a, b, c in R ($\cdot$ is associative).
   - There is an element 1 in R such that $a \cdot 1 = a$ and $1 \cdot a = a$ ∀ a in R (1 is the multiplicative identity).

## Ring

And the third ring axiom :

3 Multiplication is distributive with respect to addition :

- $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ ∀ a, b, c in R (left distributivity).
- $(b + c) \cdot a = (b \cdot a) + (c \cdot a)$ ∀ a, b, c in R (right distributivity).



## Ring in a few words

Let $(R, +)$ be a commutative (abelian) group with a second law $*$, called multiplication, that has the following properties :

1. Closure : $\forall a, b \in R$ we have $a * b \in R$
2. Associativity : well, $*$ is associative
3. Identity/Neutral element : there exists an element $e$ , it is unique, so we can call it 1.
4. Distributivity : $\forall a, b, c \in R$ we have
   - $a * (b + c) = a * b + a * c$
   - $(a + b) * c = a * c + b * c$

## Ring in a nutshell

A ring $(R, +, *)$ is a a set $R$ with two binary operations denoted $+$ and $*$ (or $*$))such that the following axioms are satisfied :

1. $(R, +)$ is an abelian group
2. $*$ is associative
3. $*$ is distributive over $+$

The ring $R$ is commutative if $\forall a, b \in R : a * b = b * a$. An example : $(\mathbb{Z}, +, *)$ is a ring.

# Plan: Introduction

## Field [?]

Intuitively, a field is a set F

1. that is a commutative group with respect to two compatible operations, addition and multiplication (the latter excluding zero), with "compatible" being formalized by distributivity,
2. and the caveat that the additive and the multiplicative identities are distinct ($0 \neq 1$).

## Field [?]

Most common way to formalize this is by defining a field $F$ as a set together with two operations, usually denoted by $+$ and $\cdot$, such that the following axioms hold :

1. Closure of F under addition and multiplication
2. Associativity of addition and multiplication
3. Commutativity of addition and multiplication
4. Existence of additive and multiplicative identity elements
5. Existence of additive inverses and multiplicative inverses (only for non zero elements)
6. Distributivity of multiplication over addition

A set is a field if it is a commutative ring in which each non zero elements have a multiplicative inverse.

## *Groups, Rings and Fields* : Some example and counter-examples :

- $(\mathbb{Z}, +, *)$ is a ring (and not a Field) because $(\mathbb{Z}, *)$ is **not** a group (example : 2 has no inverse)
- $(\mathbb{Q}, +)$ and $(\mathbb{Q}^*, *)$ are commutative groups
- So $(\mathbb{Q}, +, *)$ is a Field
- and $(\mathbb{Q}^*, *)$ is the set of invertible elements for the multiplication operation
- $(\mathbb{Z}_n^*, *)$ is a **finite** group if and only if $n$ is prime
- So $(\mathbb{Z}_n, +, *)$ is a **finite** Field if and only if $n$ is prime
- $(\mathbb{R}, +, *)$ and $(\mathbb{C}, +, *)$ are Fields

## Examples

$(\mathbb{Z}_3, +, *)$ and $(\mathbb{Z}_{31}, +, *)$ are Fields but $(\mathbb{Z}_4, +, *)$, $(\mathbb{Z}_{15}, +, *)$ are not, but they are Rings.




## A cryptographic finite Ring

- Let $p$ be a prime number (or a power of a prime)
- Let $\mathbb{Z}_p[x]$ be the (infinite) set of polynomials with coefficients in $\mathbb{Z}_p$ (with indeterminate x)
- + : classical polynomial addition ( mod $p$ for the coefficients)
- * : classical polynomial multiplication ( mod $p$ for the coefficients)

## Then :

$(\mathbb{Z}_p[x], +, *)$ is an (infinite) **ring** But in cryptography we love only **finite** sets so . . .

## "The" cryptographic finite Field used in AES-128 [?]

- Let $p$ be a prime number (variant : power of a prime)
- Let $f(x)$ a polynomial of degree $n$
- Let $\mathbb{Z}_p[x]/f(x)$ be the set of $p^n$ polynomials with degree at most $n-1$
- $+$ and $*$ : classical polynomial addition and multiplication (but $\bmod p$ and $\bmod f(x)$)
- $(\mathbb{Z}_p[x], +, *)$ is a **field** if and only if $f(x)$ is an irreducible polynomial.

Application to AES-128 :

1. $\mathbb{F}_{2^8}$ is a field and $(x^8 + x^4 + x^3 + x + 1)$ irreducible
2. This field is isomorphic to $\mathbb{Z}_2[x]/f(x)$
3. (Why ? Because $f(x)$ is irreducible)
4. Then $(\mathbb{F}_{2^8}, +, *) \approx (\mathbb{Z}_2[x]/f(x), +, *)$ is a (finite) **field**.

## An example of a ring : ring of matrices

Let $R$ be a ring, finite or infinite. Let $\mathcal{M}_n(R)$ be the set of all matrices of size $n * n$ with elements in $R$. Then let us consider $(\mathcal{M}_n(R), +, *)$ then :

- If the ring $R$ is finite then $(\mathcal{M}_n(R), +, *)$ is also finite and it is a ring
- If $R$ is infinite then $(\mathcal{M}_n(R), +, *)$ is also infinite and it is a ring

A matrix ring is any collection of matrices over a ring $R$ that form a ring under matrix addition $(+)$ and matrix multiplication $(*)$ [**?**].

## Divisibility

Let $a$, $b$ be integers, we say that $a$ *divides* $b$ and we write $a|b$ if there exists an integer $c$ such that $b = a\,c$

Some properties of divisibility :

- $a|a$ ($a$ divides $a$) (reflexivity)
- If $a|b$ and $b|c$ then $a|c$ (transitivity)
- If $a|b$ and $b|c$ then $a|(b\,x + c\,y) \quad \forall x, y \in \mathbb{Z}$
- If $a|b$ and $b|a$ then $a = \pm b$.

## Definition (important)

Let $n > 0$ and $a, b \in \mathbb{Z}$ :

- $a$ is said to be congruent to $b$ if $n|(a - b)$
- . . . and we write $a \equiv b \pmod{n}$
- . . . or $a - b \equiv 0 \pmod{n}$
- $n$ is the modulus

## Theorem

*Let $n > 0$ be an integer. For any integer $a$, there exists a unique integer $b$ such that*

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

*with $0 \leq b < n$, we will write $b = a \pmod{n}$.*

## Examples

1. $123 \equiv 4 \pmod{17}$ because $123 = 17 * 7 + 4$
2. $123 \equiv 18 \pmod{35}$ because $123 = 3 * 35 + 18$
3. $123 \equiv -17 \pmod{35}$ because $123 = 4 * 35 - 17$

## Some properties

- $a \equiv a \pmod{n}$
- $a \equiv b \pmod{n}$ is equivalent to : $\exists k \in \mathbb{Z}, a = b + k.n$
- $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$
- $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ implies $a \equiv c \pmod{n}$

## Arithmetic of congruences

If $a \equiv a' \pmod{n}$ and $b \equiv b' \pmod{n}$ then

- $a + b \equiv a' + b' \pmod{n}$
- $a \cdot b \equiv a' \cdot b' \pmod{n}$

## A theorem about multiplicative inverse :

### Theorem

*Let $n, a$ with $a \in \mathbb{Z}_n$, then $a$ has a multiplicative inverse modulo $n$ iff $GCD(a, n) = 1$.*

**Proof (part $\Rightarrow$) :**

- if $A$ is a multiplicative inverse of $a$ modulo $n$ then $a \cdot A = 1 \bmod n$
- let $k$ such that $a \cdot A = 1 + k \cdot n$
- if $d|a$ and $d|n$ then $d|1$
- therefore $GCD(a, n) = 1$.

# Multiplicative inverse



## A theorem about multiplicative inverse :

**Proof (part $\Leftarrow$) :**

- if $GCD(a, n) = 1$
- then it exists $k$ such that $a|1 + k \cdot n$ (we admit this for the moment)
- then it exists $A$ such that $a \cdot A = 1 + k \cdot n$
- therefore $A$ is a multiplicative inverse of $a$ modulo $n$

## The classical Euclidean algorithm :

**Algorithm** 1 : **First version of the Euclidean algorithm :**

**Input** : two integers $a$ and $b$ with $a \geq b$ ;

**Output** : $GCD(a, b)$ ;

**Begin** :

**While** $b \neq 0$ **Do**

**Set** $r = a \bmod b$ ;

**Set** $a = b$ ;

**Set** $b = r$ ;

**End Of While**

**Return** $a$ ;

**End**.




## The classical Euclidean algorithm : python

```
# Euclide Algorithm for computing  pgcd
def pgcd(a,b) :
    while a%b != 0 :
        a, b = b, a%b
    return b
```

## Bézout's theorem (in France, with the accent please)

### Theorem

*Let $a$ and $b$ be two integers such that $GCD(a,b) = d$, then there exist two integers $u$ and $v$ such that $a \times u + b \times v = d$.*

A *partial reciprocal* of this theorem exist : let $a$ and $b$ be two integers, if there exist two integers $u$ and $v$ such that $a \times u + b \times v = d$, then $GCD(a,b)$ is a divisor of $d$.

## An important case of the Bézout theorem

The particular case $GCD(a, b) = 1$ gives the following theorem :

## Theorem

*Let $a$ et $b$ be two integers*
$\exists u \in \mathbb{Z}, \exists v \in \mathbb{Z}$ *such that* $a \times u + b \times v = 1$
$\Longleftrightarrow GCD(a, b) = 1.$

## Properties

So, if $x \in \mathbb{Z}_n$, we will have the following *equivalent* properties :

1. $\exists y \in \mathbb{Z}_n \; x\,y \equiv 1 \bmod n$
2. $\exists y, z \in \mathbb{Z} \; x \times y + n \times z = 1$
3. $\mathrm{GCD}(x, n) = 1$.

**When $b = n$ and $a \in \mathbb{Z}_n$, then :**

$u$ is the multiplicative inverse of $a$ modulo $n$ :

$$\begin{aligned} 1 &= u \times a + v \times n \equiv u \times a \bmod n \\ u &\equiv a^{-1} \bmod n \end{aligned}$$

So, the Bézout theorem proves that the multiplicative inverse of $a$ modulo $n$ can be computed (efficiently as we will see).
The Extended Euclidean algorithm computes the integers $u$, $v$ and $d$ such that :

$$d = \mathrm{GCD}(a, b) = u \times a + v \times b.$$

**The set of invertible elements of $\mathbb{Z}_n$ is defined by :**

$$\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n \,|\, (\exists\, y \in \mathbb{Z}_n)\; x * y = 1\}.$$

An element of $\mathbb{Z}_n^*$ is called an invertible element (or sometimes a unit) of $\mathbb{Z}_n$.
This set $\mathbb{Z}_n^*$ with the multiplication in $\mathbb{Z}_n$, i.e. $(\mathbb{Z}_n^*, *)$, is a commutative group.

# Euclidean algorithms : Extended



## The Extended Euclidean algorithm

Let $a > 0$ and $b \geq 0$ be two integers with $GCD(a, b) = d$, how can we compute $u$ and $v$ such that

$$d = a * u + b * v$$

This is done with the Extended Euclidean algorithm, which can be found in any good cryptographic library. And you need this to compute an RSA key.

## The Extended Euclidean algorithm

- we will define three sequences $r_i$, $u_i$ and $v_i$
- we begin with $r_0 = a$ and $r_1 = b$
- we define for $i \geq 2$ $r_i = q_i \cdot r_{i+1} + r_{i+2}$ ($q_0$ is well defined)
- we define $u_0 = 1$ and $v_0 = 0$ and
- we define $u_1 = 0$ and $v_1 = 1$
- for $i \geq 2$ we define $u_i = u_{i-2} - q_{i-2} \cdot u_{i-1}$ and
- we define $v_i = v_{i-2} - q_{i-2} \cdot v_{i-1}$ and

Then, there exists $k > 0$ such that $r_k = 0$, this gives

$$GCD(a, b) = r_{k-1} = a \cdot u_{k-1} + b \cdot v_{k-1}.$$

## The Extended Euclidean algorithm (continue)

**Short Proof** We always have $r_i = a * u_i + b * v_i$

- $r_0 = a = 1 * a + 0 * b$
- $r_1 = b = 0 * a + 1 * b$
- if $r_{i-2} = a \cdot u_{i-2} + b \cdot v_{i-2}$ and
- $r_{i-1} = a \cdot u_{i-1} + b \cdot v_{i-1}$

then :

$$\begin{cases} a \cdot u_i + b \cdot v_i & = a \cdot (u_{i-2} - q_{i-2} \cdot u_{i-1}) + \\ & \quad b \cdot (v_{i-2} - q_{i-2} \cdot v_{i-1}) \\ & = r_{i-2} - q_{i-2} \cdot r_{i-1} \\ & = r_i \end{cases}$$

## The Extended Euclidean algorithm (recursion version) :

FONCTION 1 : **Extended-Euclidean**$(a, b)$ **:**

**Input** : two integers $a$ and $b$ with $a \geq b$ ;

**Output** : $(d, u, v)$ such that $d = a * u + b * v$ ;

**Begin** :

**If** (b==0) then **Return** (a,1,0) ;

$(d', u', v')$ = Extended-Euclidean$(b, a \bmod b)$ ;

$(d, u, v) = (d, v', u' - \lfloor a/b \rfloor v')$ ;

**Return** $(d, u, v)$.

**End**.

# Euclidean algorithms : Extended



## The Extended Euclidean algorithm (recursion version) : python

```
import sys
sys.setrecursionlimit(1000000)  # long type, 32bit OS 4B, 64b

# return (g, x, y) a*x + b*y = gcd(x, y)
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, x, y = egcd(b % a, a)
        return (g, y - (b // a) * x, x)
```

## Divisions algorithm for integers (Euclidean division)

Let $a$, $b > 1$ be integers, there exists integers $q$ and $r$ such that

$$a = q\,b + r$$

- $q$ is the Euclidean quotient
- $r$ is the Euclidean remainder
- integers $q$ and $r$ are unique if we ask $0 \leq r < b$.

The remainder $r$ is denoted $a$ mod $b$ and the quotient $a\ div\ b$. Remark : in C we write $a\%b$ for $a$ mod $b$ and $a/b$ for $a\ div\ b$, with the declaration **int a,b ;**

## Greatest common divisors

Let $a$, $b$ be integers, a non-negative integer $d$ is called the *greatest common divisor* of integers $a$ and $b$ , denoted $d = \text{GCD}(a, b)$ if

1. $d$ is a common divisor of $a$ and $b$ (i.e. $d|a$ and $d|b$) ;
2. whenever $c|a$ and $c|b$ then $c|d$

$\text{GCD}(a, b)$ is the *largest* positive integer that divides both $a$ and $b$, with an exception : $\text{GCD}(0, 0) = 0$.

# Euclidean algorithms : LCM



## Least common multiple

Let $a$, $b$ be integers, a non-negative integer $m$ is called the *least common multiple* of integers $a$ and $b$ , denoted $m = \text{LCM}(a, b)$ if

1. $a|m$ and $b|m$ ;
2. whenever $a|c$ and $b|c$ then $m|c$

$\text{LCM}(a, b)$ is the *smallest* non-negative integer divisible by both $a$ and $b$.



## Some facts

Let $a$, $b$ be positive integers :

$$\mathrm{LCM}(a,b) = \frac{a \cdot b}{\mathrm{GCD}(a,b)}$$

or

$$\mathrm{GCD}(a,b) \cdot \mathrm{LCM}(a,b) = a \cdot b$$

# Coprimes and Primes



## Coprimes

Let $a$, $b$ be positive integers, $a$ and $b$ are said to be *coprime* or relatively prime if $\text{GCD}(a, b) = 1$. $a$ and $b$ have no common divisor except the number 1.

## Primes

Let $p$ be an integer, $p$ is said to be *prime* if its only positive divisors are 1 and $p$.
Otherwise, $p$ is said *composite*.

## Permutations [?, ?]

- A permutation of a set of distincts objects is an ordered arrangement of these objects.
- Permutations occur, in more or less prominent ways, in almost every area of cryptography.
- They often arise when different orderings on certain finite sets are considered, possibly only because one wants to ignore such orderings and needs to know how many configurations are thus identified.
- For similar reasons permutations arise in the study of sorting algorithms in computer science.
- Reminder : The number of permutations of n distinct objects is n factorial usually written as $n!$.
- An ordered arrangement of r elements of a set is called an r-permutation.

## Examples

Let $X$ be the set $\{a, b, c, d\}$. and let $\sigma$ be a permutation :

- We can define $\sigma$ by :
  $\sigma(a) = d, \sigma(b) = c, \sigma(c) = b, \sigma(d) = a$.
- We can define $\sigma^k$ with $k \geq 0 : \sigma^k(x) = \sigma(\sigma^{k-1}(x))$.
- For example : $\sigma^2$ is defined by :
  $\sigma^2(a) = a, \sigma^2(b) = b, \sigma^2(c) = c, \sigma^2(d) = d$.
- So $\sigma^2 = \sigma$

A permutation that exchanges only two elements is called a *transposition* . A permutation $\sigma$ that verifies $\sigma^2 = \sigma$ is called an *involution*.

## Why permutations in cryptography ?

- Because a lot of encryption algorithms (almost all) are permutations of the (finite) set of clear texts (i.e. the messages !).
- Which means that for such an algorithm $\mathcal{A}$, there exists an integer $m$ such that $\mathcal{A}^m = \mathcal{A}$, $m$ is the order of the algorithm
- A fixed point of a permutation $P$ on a finite set $\mathcal{X}$ is an element $x \in \mathcal{X}$ that verifies $P(x) = x$.
- For a finite set $\mathcal{X}$ and a permutation $P$, for any $x \in \mathcal{X}$ there exists an integer $k$ such that $a$ is a fixed point of $P^k : P^k(x) = x$.

# Elliptic curves



**[?, ?, ?]**

- Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of *elliptic curves* over **finite fields**. One of the main benefits in comparison with non-ECC cryptography (with plain Galois fields as a basis) is the same level of security provided by keys of **smaller size**.
- Elliptic curves are applicable for : encryption, digital signatures, pseudo-random generators and other tasks.
- They are also used in several integer factorization algorithms, such as Lenstra elliptic curve factorization.

# Elliptic curves

## [?, ?, ?]

- For current cryptographic purposes, an elliptic curve is a plane curve over a finite field (rather than the real numbers)
- It consists of (all) the points satisfying the equation $y^2 = x^3 + ax + b$ (this is called the Weirstrass form).
- The general form is : $y^2 + a_1xy + a_2y = x^3 + a_3x^2 + a_4x + a_5$.
- But we add a *distinguished* point at infinity, denoted $\mathcal{O}$ : (remember) we need a neutral/identity element to have a group
- The coordinates here are to be chosen from a fixed finite field of characteristic not equal to 2 or 3, or the curve equation will be somewhat more complicated.

## Cryptographic Elliptic modulo $p$

- Let $p$ be a prime
- We will call $\mathcal{E}_p(a,b)$ (elliptic curve modulo $p$) or $E(\mathbb{Z}_p)(a,b)$ the (finite) set of points $(x,y) \in (\mathbb{Z}_p, \mathbb{Z}_p)$ $(\bigcup \mathcal{O})$ satisfying the modular equation :
- $y^2 = x^3 + ax + b \bmod p$
- This last equation is called the *Weirstrass* form
- With $a, b \in \mathbb{F}_p, 4a^3 + 27b^2 \neq 0$ (the Discriminant) (to ensure a solution to the modular equation).
- If we take a finite Field $\mathbb{F}_q$ with $q = p^n$ we will write $E(\mathbb{F}_q)(a,b)$ or $E(\mathbb{F}_{p^n})(a,b)$.

# Elliptic Curve Group



## Does a modular Elliptic Curve generate a Group ?

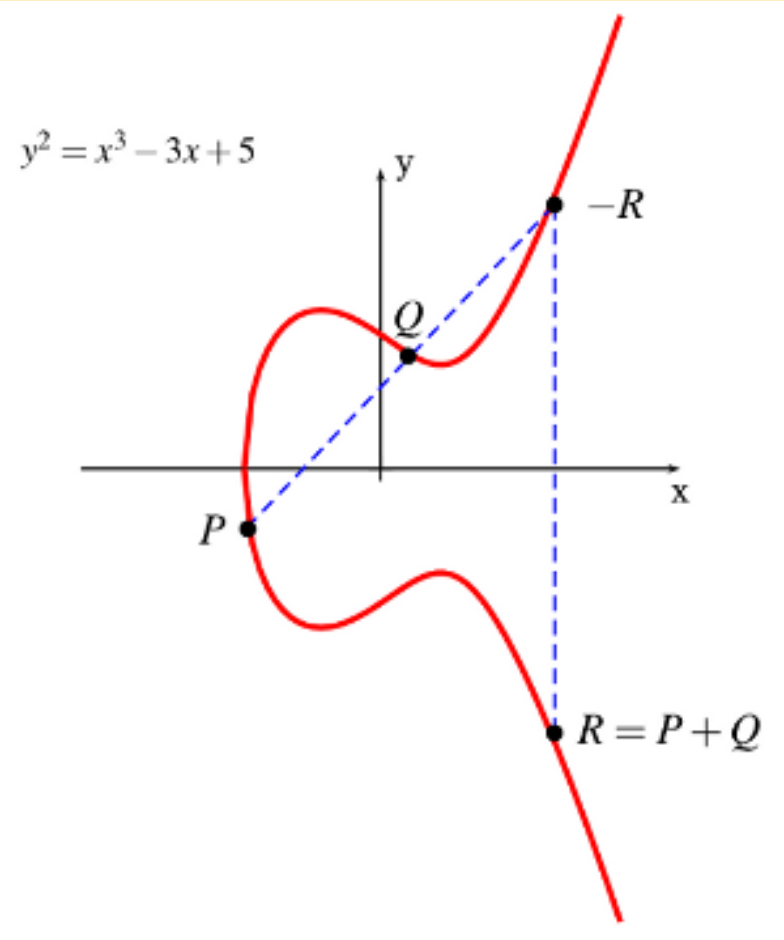
- Point addition ? $(X_R, Y_R) = (X_P, Y_P) + (X_Q, Y_Q)$
- Let s the slope of the line (P,Q))
  1. $s = (Y_P - Y_Q)(X_P - X_Q)^{-1} \bmod p$ if $X_P \neq X_Q$
  2. $s = (3X_P^2 + a)(2Y_P)^{-1} \bmod p$ in the case $X_P = X_Q$
- We now define :
  1. $X_R = s^2 - X_P - X_Q$
  2. $Y_R = s(X_P - X_R) - Y_P$
- If $Q = -P$ then $P + Q = \mathcal{O}$ (the point at infinity).
- "Exponentiation" ? $[k]P = P + P \cdots P$ (k times).
- Identity ? $\forall P$ : we have $P + \mathcal{O} = \mathcal{O} + P = P$

## Have we a Group ?

Yes : $(\mathcal{E}_p(a, b) \bigcup \mathcal{O}, +)$ is **a commutative group**.

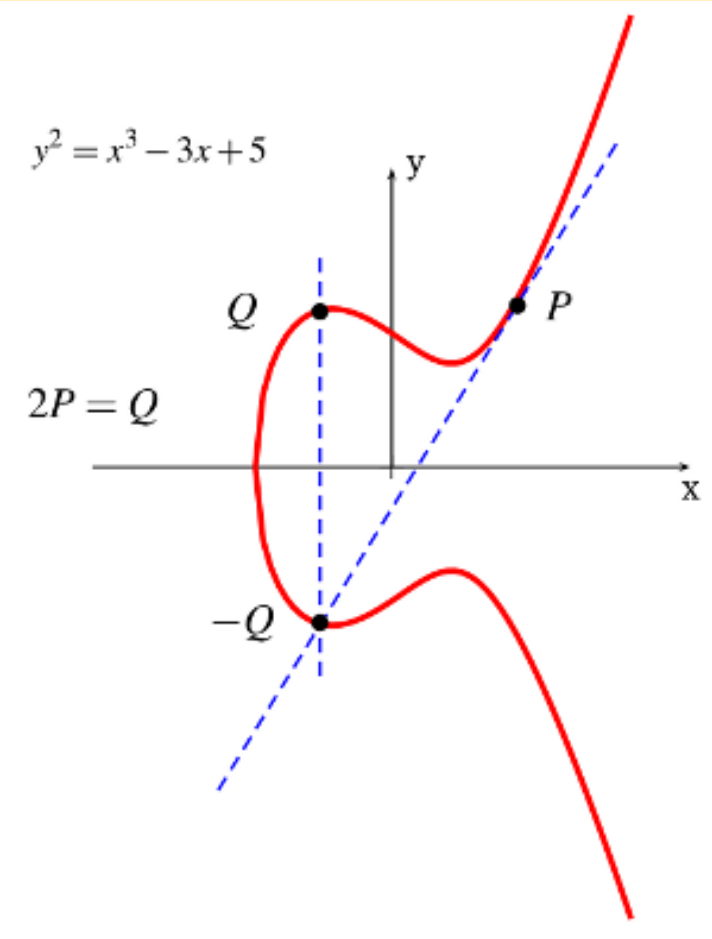
## Point Addition : graphical illustation [a]

a. http ://www.purplealienplanet.com/node/27



## Point Doubling : graphical illustation [a]

a. http ://www.purplealienplanet.com/node/27



# Some algorithms using EC



## How to use EC in cryptography ?

- ECDSA : Elliptic Curve Digital Signature Algorithm : the EC variant of DSA
- Elliptic Curve RSA : the EC variant of RSA
- ECDH : Elliptic Curve DH the EC variant of DH.
- *In fact quite all classical PKC algorithms have an EC variant* : signature, encryption, key exchange
- It is supposed that a EC key of 160-256 bits has the same level of security than a RSA/DH key of 1024-3072 bits.
- ECC is slowly killing classical PKC algorithms (RSA, DH, etc.) : well RSA & DH are still living but . . . ...
- ECM : Elliptic Curve Method of *integer factorisation* (Lenstra) : efficient to find small or small enough factors

# Curve25519 [?]



**Curve25519 was first released by Daniel J. Bernstein in 2005 [?]**

1. Curve25519 is an elliptic curve offering 128 bits of security
2. Designed for use with the elliptic curve Diffie-Hellman (ECDH) key agreement scheme
3. One of the fastest ECC curves
4. Not covered by any known patents, and it avoids problems with poor quality random-number-generators.

The curve used is defined by $y^2 = x^3 + 486662x^2 + x$, a Montgomery curve, over the prime field defined by the prime number $p = 2^{255} - 19$, and it uses the base point x = 9.

# Elliptic curves



**[?]**

Curve25519 was first released by Daniel J. Bernstein in 2005, but interest increased considerably after 2013 when it was discovered that the NSA had backdoored Dual_EC_DRBG. While not directly related, suspicious aspects of the NIST's Pxxx[a] curve constants led to concerns that the NSA had chosen values that gave them an advantage in factoring public keys,

*I no longer trust the constants. I believe the NSA has manipulated them through their relationships with industry.*

Bruce Schneier, prominent security researcher.

Since then, Curve25519 has become the de-facto alternative to P-256. In 2014, both OpenSSH and GPG default to Curve25519-based ECDH.

---

*a*. A set of curves proposed by NIST and calculated by NSA : see [**?**]

## Some "not safe" ECCs [?]

The NIST P-224 ECC : with $p = 2^{224} - 2^{96} + 1$ the curve is

$$y^2 = x^3 - 3x + a \bmod p$$

Where $a$ is :

18958286285566608000408668544493926415504680968679321075787234672564

## ECOH : Elliptic Curve Only Hash, ; not so secure

1. The ECOH-n algorithms family was a candidate for SHA-3.
2. n can be : 224, 256, 384, 512 (the length of the hash result)
3. Example : ECHO-224 uses the ECC B-233 (with a point G given) which is a NIST/NSA curve !
4. However, it was rejected in the beginning of the competition since a second pre-image attack was found.
5. Given n, ECOH divides the message M into n blocks $M_0, \ldots, M_{n-1}$. If the last block is incomplete, it is padded with single 1 and then appropriate number of 0.




## ECOH : Elliptic Curve Only Hash : not so secure [?]

1. Let furthermore $\mathcal{P}$ be a function that maps a message block and an integer to an elliptic curve point.
2. Then using the mapping $\mathcal{P}$, each block is transformed to an elliptic curve point $P_i$, and these points are added together with two more points : $X_1$ and $X_2$.
3. The first additional point $X_1$ contains the padding and depends only on the message length.
4. The second additional point $X_2$ depends on the message length and the XOR of all message blocks.
5. The result is truncated to get the hash H.

## ECOH : Elliptic Curve Only Hash : not so secure [?, ?]

1. $P_i = \mathcal{P}(M_i, i)$
2. $X_1 = \mathcal{P}'(n)$
3. $X_2 = \mathcal{P}^*(M_i, n)$
4. $Q = X_1 + X_2 + \sum_{i=0}^{n-1} P_i$
5. $H = f(Q)$.
6. $f(Q) := \lfloor x(Q + \lfloor x(Q)/2 \rfloor G)/2 \rfloor \bmod 2^n$ outputs the n-bit result.