Information Security
(WBCS004-05)

Fatih Turkmen

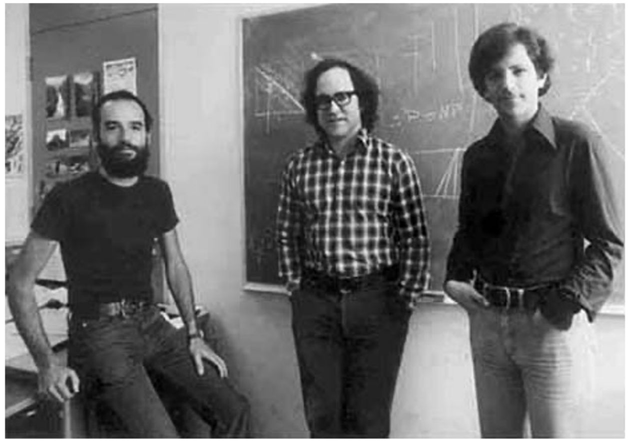
Some slides are borrowed from Dr. Frank B. Brokken

Today

- Topics of this lecture:
  - RSA
  - Applications of the Chinese Remainder Theorem
  - Diffie-Hellman key exchange
  - Elliptic Curve Cryptography

RSA

- Named after Shamir, Rivest, Adleman (left to right)
- Concept originally by Clifford Cocks (Government Communications Headquarters (GCHQ))
- Used in, e.g., PGP/GPG, SSL
- Its security depends on the difficulty of **factoring large numbers**.



RSA - Key Generation

1. Receiver *chooses* two large prime numbers **p** and **q**. Their product, **n**=*pq*, is half of the ***public key***.

2. Receiver *calculates* $\boxed{\phi(pq) = (p-1)(q-1)}$

and *chooses* a number **e** *rp* to **ϕ(pq)**. ***e*** will be the other half of the public key.

3. The receiver *calculates* the multiplicative inverse *d* of *e* modulo $\phi(n)$:

$$\boxed{de \equiv 1 \pmod{\phi(n)}}$$

d is the private key.

4. The receiver distributes both parts of the public key: *n* and *e*. *d* is kept secret.

Two communicating parties : **sender** and **receiver**

How do we use the keys?

relative prime (rp): only 1 is a common divisor

Linear Congruences

ax ≡ b (mod m)

RSA – Encryption/Decryption

Encryption

1. First, the sender converts his/her message into a number m (e.g., uses the ASCII alphabet)
2. The sender calculates

$$c \equiv m^e \pmod{n}$$

Decryption

1. The receiver computes

$$c^d \equiv m \pmod{n}$$

thus retrieving the original number m.

2. The receiver translates m back into letters, retrieving the original message.

Does RSA work?

We will mostly talking about groups, i.e., multiplication modulo N

DEFINITION 7.9 *A* group *is a set* $\mathbb{G}$ *along with a binary operation* $\circ$ *such that:*

(Closure) *For all* $g, h \in \mathbb{G}$, $g \circ h \in \mathbb{G}$.

(Existence of an Identity) *There exists an* identity $e \in \mathbb{G}$ *such that for all* $g \in \mathbb{G}$, $e \circ g = g = g \circ e$.

(Existence of Inverses) *For all* $g \in \mathbb{G}$ *there exists an element* $h \in \mathbb{G}$ *such that* $g \circ h = e = h \circ g$. *Such an* h *is called an* inverse *of* g.

(Associativity) *For all* $g_1, g_2, g_3 \in \mathbb{G}$, $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.

When $\mathbb{G}$ *has a finite number of elements, we say* $\mathbb{G}$ *is a* finite group *and let* $|\mathbb{G}|$ *denote the* order *of the group; that is, the number of elements in* $\mathbb{G}$. *A group* $\mathbb{G}$ *with operation* $\circ$ *is* abelian *if the following additional condition holds:*

(Commutativity) *For all* $g, h \in \mathbb{G}$, $g \circ h = h \circ g$.

When the binary operation is understood, we simply call the set $\mathbb{G}$ *a group.*

Borrowed from Katz&Lindell's book

Let's try to find : Z^*_{15}

$Z^*_{15}=\{1,2,4,7,8,11,13,14\}$

Does RSA work?

Can RSA correctly decrypt?

Show $\mathbf{m = c^d \ (mod\ N) = m^{ed}(mod\ N)}$

1. Consider Euler's *totient* function $\Phi(N)$:

the number of positive integers smaller than N that are *relative prime (rp)* to N

❗ $\Phi(6) = ?$ 2 {1,5}

We will mostly talking about groups, i.e., multiplication modulo N

- For $N = p * q$: (p, q: primes)

$\Phi(N) = (p-1) * (q-1)$

Does RSA work? (cont.)

2.1 Fermat's little theorem (FLT): for p, a prime number, and m an integer co-prime to p; the number $m^p - m$ is an integer multiple of *p*

$$\boxed{m^p = m \pmod{p} \rightarrow m^{p-1} = 1 \pmod{p}}$$

Can we generalize this?

2.2 Well-known (Euler's) theorem: Two numbers **m** and **n**

$$\boxed{\text{If } \underline{\mathbf{m\ rp\ n}}\text{, then } m^{\Phi(n)} = 1 \pmod{n}}$$

FLT is a special case: when n is prime and thus $\Phi(n) = n-1$

Example:

- 5 rp 6, $\Phi(6) = 2$.
- thus we have $5^2 = 1 \pmod{6}$

Proof? See [3]

Does RSA work? - Encryption

- Scenario: Use RSA to exchange a secret key *K*
- RSA steps:
  - To *encrypt* a <u>*key*</u> *K*, select K such that:
    (1) $K < N$, **(2)** *K rp N*, *and* **(3)** $K^e > N$

  - We use *K rp N* when selecting/retrieving *K*
  - K is *not* the message in this context but a **symmetric encryption key**

  - To encrypt, compute:

$$C = K^e \pmod N$$

Does RSA work? - Encryption

- Example (RSA Encryption):
  - To *encrypt* *K*:

$$\boxed{C = K^e \pmod{N}}$$

  - Numeric example:
    - p = 23, q = 29 so: **N = 667**, **Φ**(N) = 616
    - select, e.g., **e = 5**, we use K = 21

K < *N,* *21 < 667*
K rp N, *21 rp 667*
$K^e > N$ $21^5 > 667$

K = 21, so **C** = K^5 (mod N): 21^5 (mod 667)
= 4084101 (mod 667) = **60**.

Does RSA work? - Decryption

- RSA: decryption in steps[1]:
  - from:

$$C = K^e \pmod{N}$$

  - compute:

$$\mathbf{C}^d = (K^{\mathbf{e}})^d = K^{ed}$$

[1]: all computations are "mod N"

RSA Does RSA work? - Decryption

- RSA decryption in steps[1]:

$de \equiv 1 \pmod{\phi(n)}$

d is chosen s.t.

$$C^d = (K^e)^d = K^{ed}$$

Since $de = 1 + x\Phi(N)$:

x is some integer

$$C^d = K^{ed} = K^{1+x\Phi(N)}$$

[1]: all computations are "mod N"

Does RSA work? - Decryption

- RSA decryption in steps[1]:

We said in the previous slide:
($de = 1 + x\Phi(N)$)

$$C^d \quad = K^{ed} = K^{1+x\Phi(N)}$$

$$= K * K^{x\Phi(N)}$$

[1]: all computations are "mod N"

Does RSA work? - Decryption

- RSA decryption in steps[1]:

$(de = 1 + x\Phi(N))$

$$C^d = (K^e)^d = K^{ed} = K^{1+x\Phi(N)}$$

$$= K * K^{x\Phi(N)}$$

$$= K * \underbrace{K^{\Phi(N)} * \ldots * K^{\Phi(N)}}_{x \text{ times}}$$

[1]: all computations are "mod N"

Does RSA work? - Decryption

- RSA decryption in steps[1]:

$(de = 1 + x\Phi(N))$

$$C^d = (K^e)^d = {}^{ed} = K^{1+x\Phi(N)}$$

$$= K * K^{x\Phi(N)}$$

$$= K * \underbrace{K^{\Phi(N)} * \ldots * K^{\Phi(N)}}_{x \text{ times}} \; ?$$

$K^{\Phi(N)} = 1 \bmod N$, *so:* $= K$

Where is this coming from?

Euler's Theorem

Note: we selected K such that K rp N

[1]: all computations are %N

Does RSA work? - Example

- Back to the example:
  - d, computed from

    $ed = 1 \% \Phi(N)$

    $5d = 1 \% 616$

    $d = 493$
  - From $K = 21$ we computed $C = 60$ ($N = 667$).
  - K is computed as $C^{d} \% N =$ **21**.

Numeric example:

$p = 23$, $q = 29$ so: $\mathbf{N = 667}$, $\Phi(N) = 616$

select, e.g., **$e = 5$**, we use $K = 21$

next (hidden) slides show how to compute $60^{493} \% 667$ (i.e., large exponents modulo)

RSA – Key Generation and Security

Select e such that e *rp* $\Phi(N)$, then compute d and x:

$$de = 1 + x\,\Phi(N)$$

- Finding the private key requires solving Linear congruences ($ax \equiv b \pmod m$) which can be stated as **xa + ym = b**

Watch [6] for understanding LDE better!

- Enter *: Linear Diophantine equation (LDE):*

E.g, 8x + 6y = 2, solves for x = 1, y = -1 or x = -5, y = 7

$$\boxed{xa + ym = b}$$

  - Solvable for integral values if $d \mid b$ for some x and y
  - If a *rp* m we can solve:

$$xa + ym = 1$$

The Euclidian algorithm can be used to solve this

d = GCD (a,m)

if ***d* | *b*** *(d can be divided by b) then there is a solution to the Equation (**Bezout's Identity**)*

Refresher: Euclidian Algorithm

Problem: Find *gcd(a,b)*

1. Find repeatedly $\boldsymbol{a} = q\boldsymbol{b} + \boldsymbol{r}$, $0 \le r < |b|$

2. If r=0, stop and output b; gcd(a,b) = b

3. If $r \neq 0$, replace ($\boldsymbol{a}$,$\boldsymbol{b}$) by ($\boldsymbol{b}$,$\boldsymbol{r}$). Go to Step 1.

Special case if *a* **rp** *b* then gcd(a,b) = 1

Find gcd(210, 50)

$$\underset{a}{210} = \underset{q}{4} * \underset{b}{50} + \underset{r}{10} \qquad (1)$$

$r \neq 0$ then Find gcd(50, 10)

$$\underset{a}{50} = \underset{q}{5} * \underset{b}{10} + \underset{r}{0} \qquad (2)$$

r=0 then **gcd(210, 50)** = **10**

Solving LDEs [1] (Extended Euclidean)

$xa + ym = b$

- Use the Euclidean algorithm to compute gcd(a,m) = d (record all steps for substitution)
- Determine if d | b. If not, then there are no solutions.
- Reformat the equations from the Euclidean algorithm.
- Using substitution, go through the steps of the algorithm to find a solution to the equation.
- The initial solution to the equation xa + ym = b is the ordered pair $(\mathrm{xi}\,\frac{b}{\mathrm{d}}\;, yi\,\frac{b}{d})$
- Other solutions are $(\mathrm{x_i} + m\frac{b}{\gcd(a,b)}\;, y_i - m\frac{a}{\gcd(a,b)})$ for an integer m and “a” solution (x_i,y_i)

LDE Example

- Use the Euclidean algorithm to compute gcd(a,m) = d (record all steps for substitution)
- Determine if d | b. If not, then there are no solutions.
- Reformat the equations from the Euclidean algorithm.
- Using substitution, go through the steps of the algorithm to find a solution to the equation $ax_i + by_i = d$.
- The initial solution to the equation ax + by = n is the ordered pair $(\mathrm{xi}\frac{n}{\mathrm{d}}, yi\frac{n}{d})$
- Other solutions are $(\mathrm{xi} + m\frac{b}{\gcd(a,b)}, yi - m\frac{a}{\gcd(a,b)})$ for an integer m and "a" solution (x_i, y_i)

- Example:

Solve: 491x + 41y = 10

Approach:
(1) gcd(491,41) = ?
(2) gcd(491,41) | 10 ?

❗ What values of x and y make the equation work?

use the x,y factors, start applying Euclidean (***a*** = *q**b*** + ***r***):

491 = 11*41 + 40

LDE Example (cont.)

- Example:

Solve: $491x + 41y = 10$
Approach: how to find x,y:

$491 = 11*41 + 40$
then:

replace (***a***,***b***) by (***b***,***r***).

$41 = 1*40 + \mathbf{1}$

*GCD: if the next step results in 0, stop (40 = 40 * 1 + 0). This special case since GCD(491,41) =1!*

LDE Example (cont.)

- Example:

Solve: 491x + 41y = 10
Approach: how to find x,y:

491 = 11*41 + 40

Rewrite

$\underbrace{491 - 11*41}$ = 40

then:

41 - 1*40 = 1

so:

Substitute

41 - 1*$\overbrace{(491 - 11*41)}$ = 1

LDE Example (cont.)

- Example:

Solve: 491x + 41y = 10

Approach: how to find x,y:

$\underbrace{491 - 11*41}$ = 40

then:

41 - 1*40 = **1**

so:

41 - 1*(491 - 11*41) = 1

equals:

-1 *491 + 12*41 = 1

Solution:

x = -1, y = 12 (i.e., $\overbrace{\text{x = -10, y = 120}}$)

Watch [7] for a nice explanation

LDE Example (cont.)

Instead of textbook solution, i.e.

$$\left(x^* + m\frac{b}{gcd(a,b)},\ y^* - m\frac{a}{gcd(a,b)}\right)$$

- Finding more solutions:

Solve: 491x + 41y = 10

Solution: x = -1, y = 12

least common multiple

As a*b = gcd(a,b) * lcm(a,b) we can add *and* subtract *k*lcm.*

E.g.,

491*41 - 491*41 + -1*491 + 12*41 = 1

and so:

491*40 - 41*479 = 1

Alternative solution: x = 40, y = -479. (*10)

Further Interest

- RSA is homomorphic over multiplication

- *See this for an example: https://asecuritysite.com/encryption/hom_rsa*

Extra RSA explanation

https://youtu.be/wXB-V_Keiu8

Diffie-Hellman Key Exchange

Key exchange

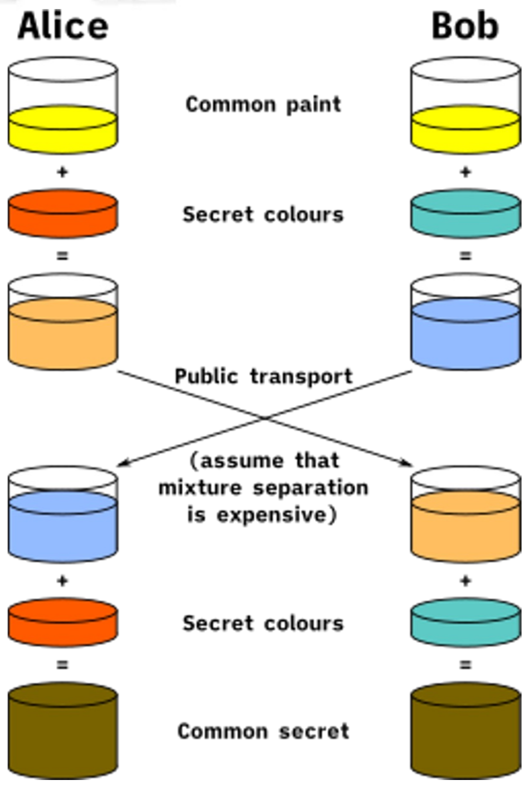
- *Sideline*: cast of characters:
  - *Alice* and *Bob* exchange confidential info.
  - *Eve* tries to *eavesdrop* and to intercept the confidential info.

- Its security depends on the difficulty of **discrete log problem**.

Diffie-Hellman Key Exchange

- Diffie-Hellman key exchange
  - Publish, or agree upon: p and g.
  - Now, the steps:
    - *Alice* chooses x and sends g^x *(mod p)* to *Bob;*
    - *Bob* chooses *y and* sends g^y *(mod p)* to *Alice.*
    - *Alice* computes $(g^y \pmod p)^x \pmod p = g^{xy} \pmod p$
    - *Bob* computes $(g^x \pmod p)^y \pmod p = g^{xy} \pmod p$

Shared secret: g^{xy} *(mod p)*



Diffie-Hellman Key Exchange

Discrete log problem

- to find k in $x = g^k$. Normally we compute $\log_g x$. E.g., $8 = 2^3$, and $\log_2 8 = 3$.
- Calculating k is *difficult* in the following case:
  - p is a prime, used in $x = g^k$ **(% p)**. (g is a **generator** for p)
    - e.g., find k if x = 23, p = 29, g = 5? ($23 = 5^k$ (% 29))

Enc: If k is given then calculating x is **easy**

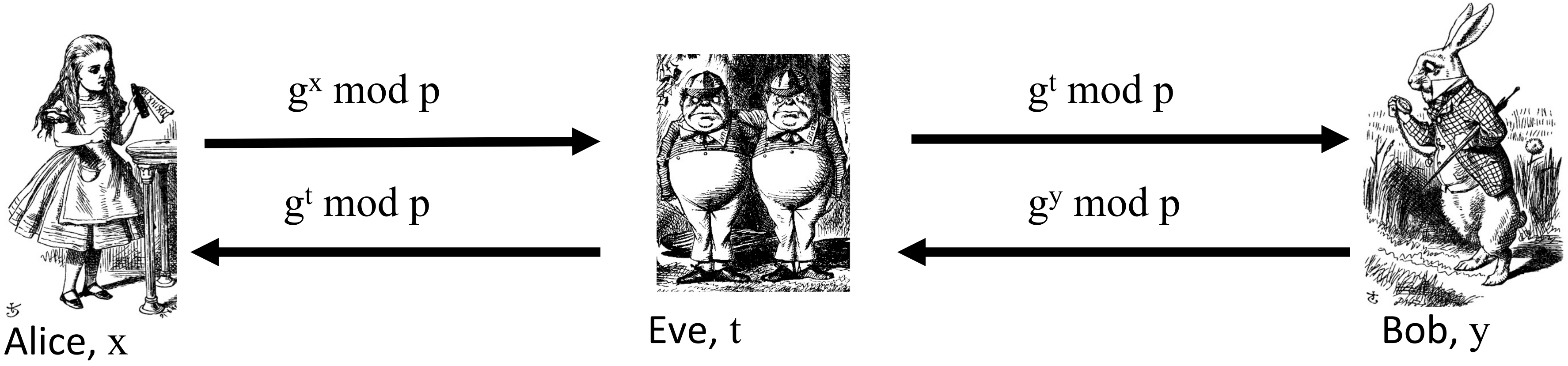
Security of the key: Given x, calculating k is **difficult** under modulo p

Diffie-Hellman Key Exchange

- *Eve* has seen $g^x \pmod p$ and $g^y \pmod p$ but she cannot retrieve $g^{xy} \pmod p$ since she has to find either x or y.
- But there are some pitfalls:
  - Man in the middle (see the next slide where g^x and g^y are replaced)
  - g is not a “proper” generator, but generates a small subgroup of values
  - p is too small

Diffie-Hellman

- Man-in-the-middle (MiM) attack



- Eve <u>establishes</u> a secret g^{xt} mod p with Alice
- Eve <u>establishes</u> secret g^{yt} mod p with Bob
- Alice and Bob don’t know Eve is MiM

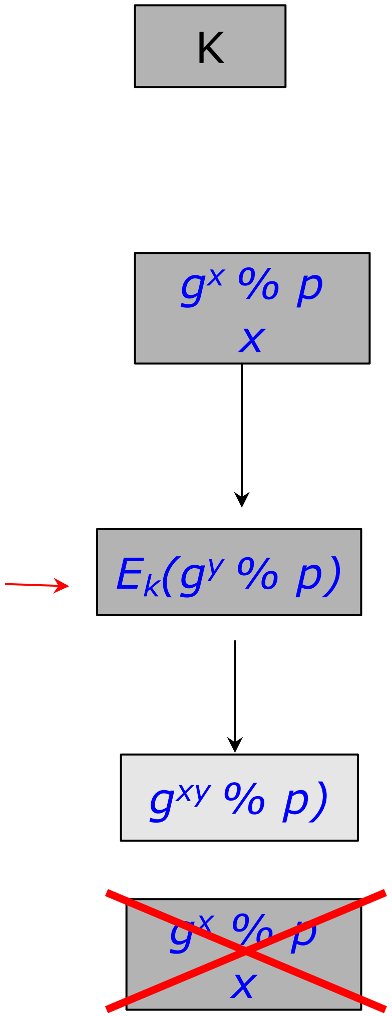
What happens?

Diffie-Hellman Key Exchange

- What is and how to find a *generator*?
  - What is a *generator*?
    - A *generator* g allows you to find values n satisfying for x in $\{1, 2, ..., p-1\}$: $x = g^n \pmod p$
  - How to find a *generator*?
    - Determine the *prime factors* of $p-1$ (e.g., q_i)
    - If for all q_i: $g^{(p-1)/q} \neq 1 \pmod p$ then g is a *generator* for p.

(Ephemeral) Diffie-Hellman

In what sense? ☺
See the comment in the slide

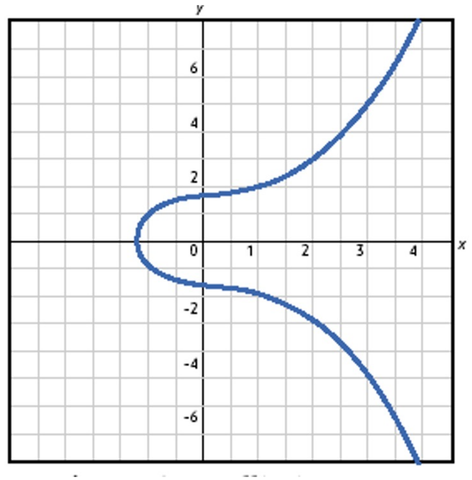


- Both parties ***share*** a long-lasting encryption key ***K***
- Both parties compute (but do not save) their x and $g^x \% p$
  - *This prevents replay-attacks (each connection has a new session key).*
- Both parties exchange their $E_K(g^x \% p)$
  - *This prevents the MiM attack*
- Both parties obtain the other party's $g^x \% p$ and compute $g^{xy} \% p$: a *session* key.
  - A compromised ***K*** doesn't yield the session key.

Elliptic Curve Cryptography

- Elliptic Curve[1] Cryptography (ECC)
  - RSA keys must be long or they can be factored;
  - Elliptic curves use another approach to encryption and can be used for public key cryptography as well.
  - ECC requires fewer bits to achieve the same level of security
  - Its foundation consists of elliptic curves of the form

$$y^2 = x^3 + a*x + b$$

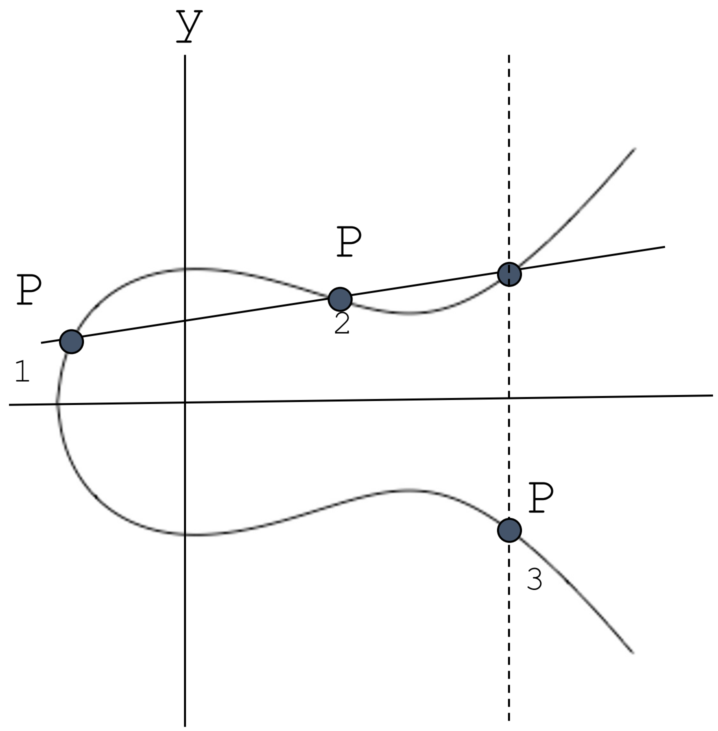


[1]: A.k.a. *elliptic equations*

Elliptic Curve Cryptography

- For cryptography only discrete (X, Y) points are used (i.e., modulo N). So, only x_i values for which y_i are integral values.
- Given P_1 and P_2, addition refers to finding point/s P_3 in the geometric sense
- Stamp provides an algorithm for computing point addition which is the only operation we need

$$(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$$



Point Addition Algorithm

(Point) Addition in arithmetic terms:

$P_1 + P_2 = P_3 \rightarrow (x_1,y_1) + (x_2,y_2) = (x_3,y_3)$

$x_3 = m^2 – x_1 - x_2 \quad (\text{mod } p)$

$y_3 = m (x_1 – x_3) – y_1 \quad (\text{mod } p)$

$$m = \begin{cases} (y_2-y_1) / (x_2-x_1) \quad (\text{mod } p) \quad \text{if } P_1 \neq P_2 \\ (3(x_1)^2 + a) / (2y_1) \quad (\text{mod } p) \quad \text{if } P_1 = P_2 \end{cases}$$

See also [4]

Infinity (∞) is the identity element!

Special case 1: If $m = \infty$ then $P_3 = \infty$

Special case 2: $\infty + P = P$ for all P

Point Addition Example

Assume ($a=2$ and $b=1$) and $N=5$

i.e. $y^2 = x^3 + 2*x + 1$

Two points: $P_1(1,3)$ and $P_2(3,2)$

❗ What is $P1 + P2$?

(0,4)

(Point) Addition in arithmetic terms:

$P_1 + P_2 = P_3 \rightarrow (x_1,y_1) + (x_2,y_2) = (x_3,y_3)$

$x_3 = m^2 - x_1 - x_2 \quad (\text{mod } p)$

$y_3 = m(x_1 - x_3) - y_1 \quad (\text{mod } p)$

$$m = \begin{cases} (y_2-y_1) / (x_2-x_1) \quad (\text{mod } p) & \text{if } P_1 \neq P_2 \\ (3(x_1)^2 + a) / (2y_1) \quad (\text{mod } p) & \text{if } P_1 = P_2 \end{cases}$$

Infinity (∞) is the identity element!

Special case 1: If $m = \infty$ then $P_3 = \infty$

Special case 2: $\infty + P = P$ for all P

(Scalar) Point Multiplication (double-and-add algorithm)

point_add: we know this already

For computing **dP** (which is $\underbrace{P + P + \ldots}_{d}$, point addition)

$d = d_0 + 2^1 d_1 + 2^2 d_2 + \ldots + 2^m d_m$ (d in binary form)

point_double: $2(2^x P)$

$N \leftarrow P$

$Q \leftarrow 0$

for i from 0 to m do

if $\boldsymbol{d_i = 1}$ *then*

$Q \leftarrow$ point_add(Q, N)

$N \leftarrow$ point_double(N)

return Q

Point Multiplication (Example)

$N \leftarrow P$

$Q \leftarrow 0$

for i from 0 to m do

if $d_i = 1$ then

$Q \leftarrow$ point_add(Q, N)

$N \leftarrow$ point_double(N)

return Q

Equation: a = -7, b = 10, calculate m = -1

15 * P(1,2)

00001111 ≡ $2^3 + 2^2 + 2^1 + 1$ ≡ $\mathbf{2^3P + 2^2P + 2^1P + 2^0P}$

- Take P. (1,2)
- *Double* it, so that we get 2^1P. (1,2) + (1,2) = (-1,-4) this is 2P
- *Add* 2P to P (in order to get the result of $\mathbf{2^1P + 2^0P}$). (-1,-4) + (1,2) = (1,6)
- *Double* 2P, so that we get 2^2P = 2 (2P). (-1,-4) + (-1,-4) ….
- *Add* it to previous result (so that we get $\mathbf{2^2P + 2^1P + 2^0P}$).
- *Double* 2^2P to get 2^3P.
- *Add* it to our result (so that we get $\mathbf{2^3P + 2^2P + 2^1P + 2^0P}$).

See [5] for more details

Note: if d_i is zero (0) don't perform any operation

DH with Elliptic Curve Cryptography

- DH Key Exchange by using ECC
  - The *public key* consists of four elements:

    - the *a* and *b* parameters of

    $$y^2 = x^3 + \boldsymbol{a}*x + \boldsymbol{b}$$

    - an initial point P1 = (x_1, y_1)
    - a prime *N*.

  - The *secret key* is a *multiplier m*.
  - These allow us to exchange a *shared secret*.

DH with Elliptic Curve Cryptography

- DH public key cryptography using ECC
  - The DH ECC starts with $y^2 = x^3 + a*x + b$ $(\%N)$
  - Select, e.g., a = 11, N = 167 (i.e., our curve and modulus), and an initial point P1, e.g., (2, 7)
    - From this: compute b = 19

(prime)

  - Make available (publicly):

$y^2 = x^3 + \mathbf{11}*x + \mathbf{19}$ $(\% \mathbf{167})$, and
(2,7)

a **b** **N**

P_1

DH with Elliptic Curve Cryptography

- DH public key cryptography using ECC
  - Make available (publicly):

    $y^2 = x^3 + 11*x + 19$ (% 167), and
    (2,7)

  - Procedure:
    - *Alice* selects *$m = 15$*, and sends $\overbrace{15 * (2,7)}^{\text{Scalar multiplication}} = \boxed{}$ to *Bob*;
    - *Bob* selects *$n = 22$*, and sends $22 * (2,7) = (9, 43)$ to *Alice;*

❗ Try this at home.

DH with Elliptic Curve Cryptography

- DH public key cryptography using ECC
  - Procedure, step 2:
    - *Alice* computes
      $P = m * (n * P1) = 15 * (9, 43) = (131, 140)$
      which is the *shared secret;*

    - *Bob* computes
      $P = n * (m * P1) = 22 * (102, 88) = (131,140)$
      and obtains the same shared secret.

    - *Alice* and *Bob* now use P as shared key.

RSA, ECC and the Future

- RSA requires longer key lengths
- Transition towards CDH/DHH-based on **ECC**

What did we learn today?

- Details of RSA
- Applications of the Chinese Remainder Theorem
- Diffie-Hellman key exchange to share a secret kay
- Elliptic Curve Cryptography

References

[1] https://brilliant.org/wiki/linear-diophantine-equations-one-equation/

[2] https://math.stackexchange.com/questions/87718/rsa-how-eulers-theorem-is-used

[3] https://brilliant.org/wiki/eulers-theorem/

[4] https://en.wikipedia.org/wiki/Elliptic_curve_point_multiplication#Point_addition

[5] https://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/

[6] https://www.youtube.com/watch?v=gMGmWSr8-Aw

[7] https://www.youtube.com/watch?v=FjliV5u2IVw

That’s all for today.