Asymmetric Cryptography

DAT159 – Basic Cryptography

Tosin Daniel Oyetoyan

Symmetric vs. Asymmetric Cryptography

- Secure encrypted communication between two parties required that they first exchange keys by some secure physical channel (e.g. List transported by trusted courier)
- Symmetric cryptography
  - Key distribution problem (key exchange)
  - Key management problem
    - Scalability (Number of keys)
  - Alice or Bob can deny their actions (Non-repudiation)

Major breakthrough

- Symmetric cryptography was used for at least 4000 years
- 1976
  - Public Key Crypto was first presented by Martin Hellman, Ralph Merkle, and Whitfield Diffie at Stanford University in 1976.
  - Then named as Diffie–Hellman key exchange method, although built on the PhD work of Ralph Merkle
    - allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel
  - Declassified documents in 1997 by the British government revealed that James Ellie, Clifford Cocks and Graham Williamson from UK's GCHQ discovered/realized the public key crypto between 1970 and 1974
- 1977
  - RSA was officially published
    - named after (Rivest-Shamir-Adleman)
  - Public key crypto to secure data transmission

Public key cryptography

- Provides
  - Non-repudiation
  - Authentication
  - Confidentiality

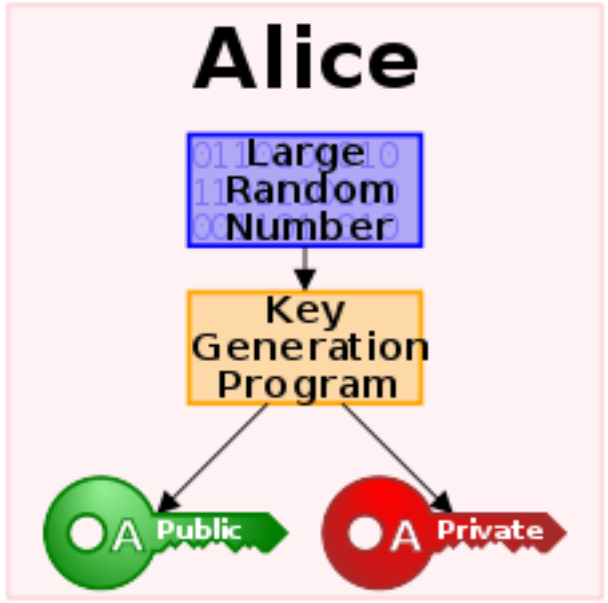
- ‘One-way’ function
- Trapdoor function
- Easy to compute (encrypt) but hard to invert (decrypt)
- Public/private keys are mathematically related

Challenges with asymmetric cryptography

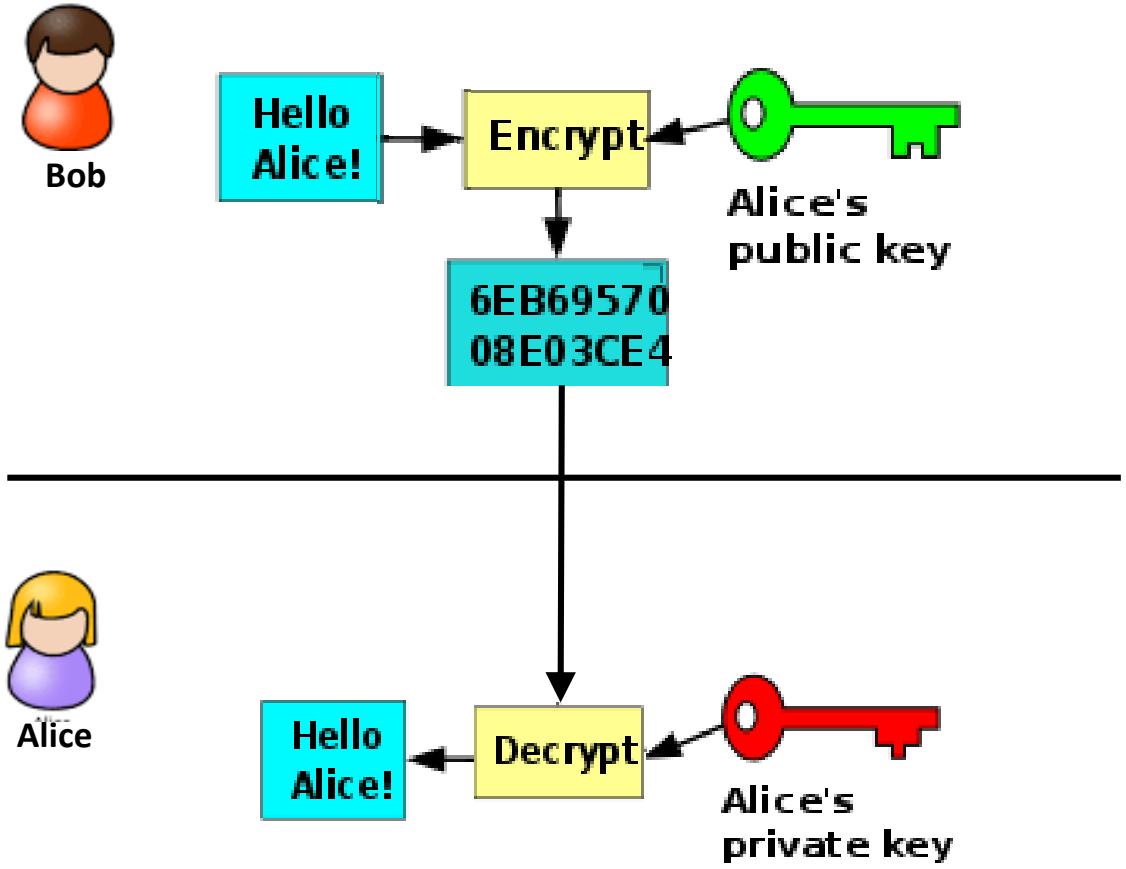
- Computationally expensive (100x – 1000x slower to symmetric)
- Distribution of new keys

Because asymmetric cryptography is slow, in practice, symmetric cryptography is used together with asymmetric cryptography (Hybrid cryptography) when encrypting large blocks of data
Asymmetric key is used to securely transport a jointly agreed symmetric key to be used for the actual encryption

Public/private key pairs



Public/Private Key pair

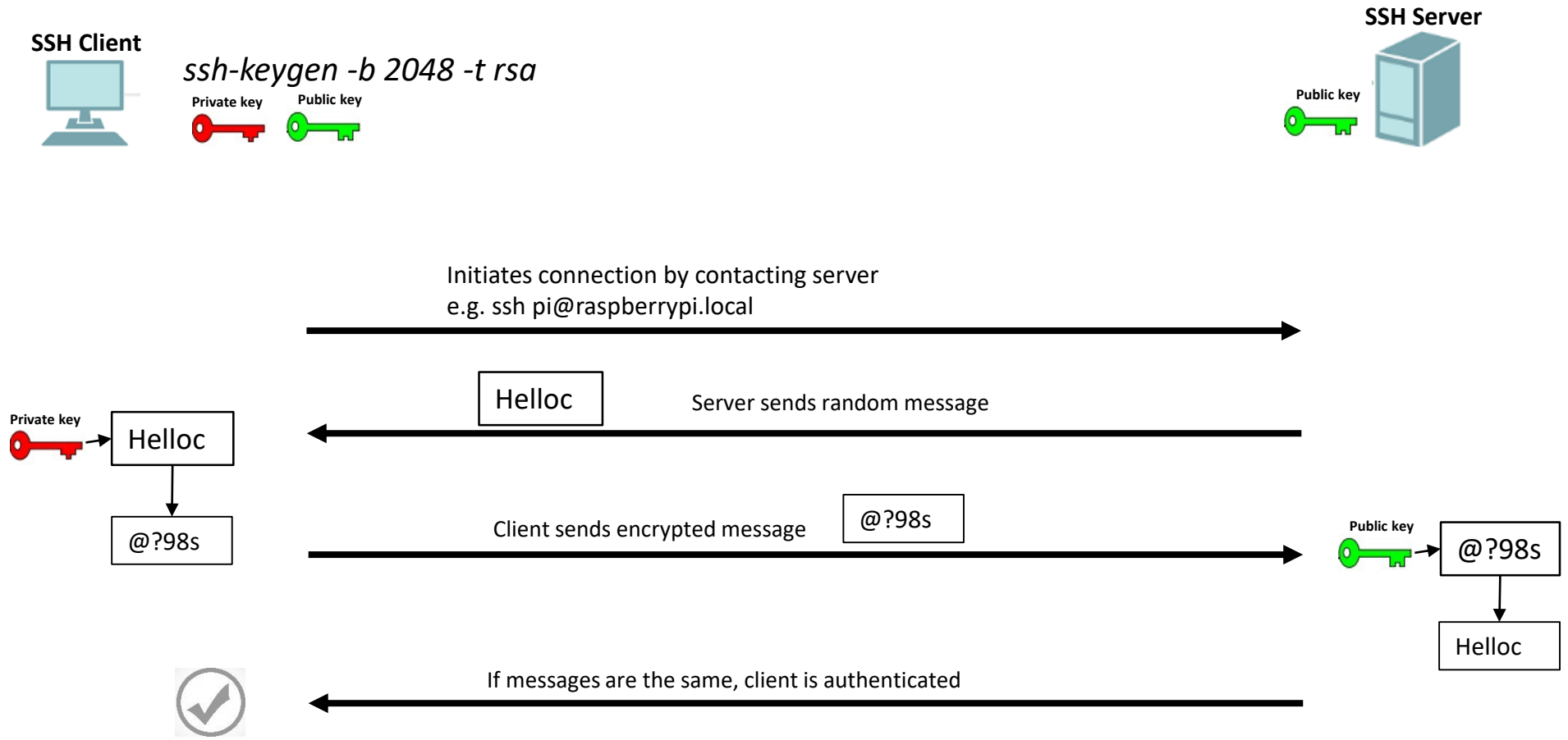


SSH protocol Illustrated

- Authentication
- Confidentiality
- Integrity

Secure Shell (SSH) – Authentication illustrated

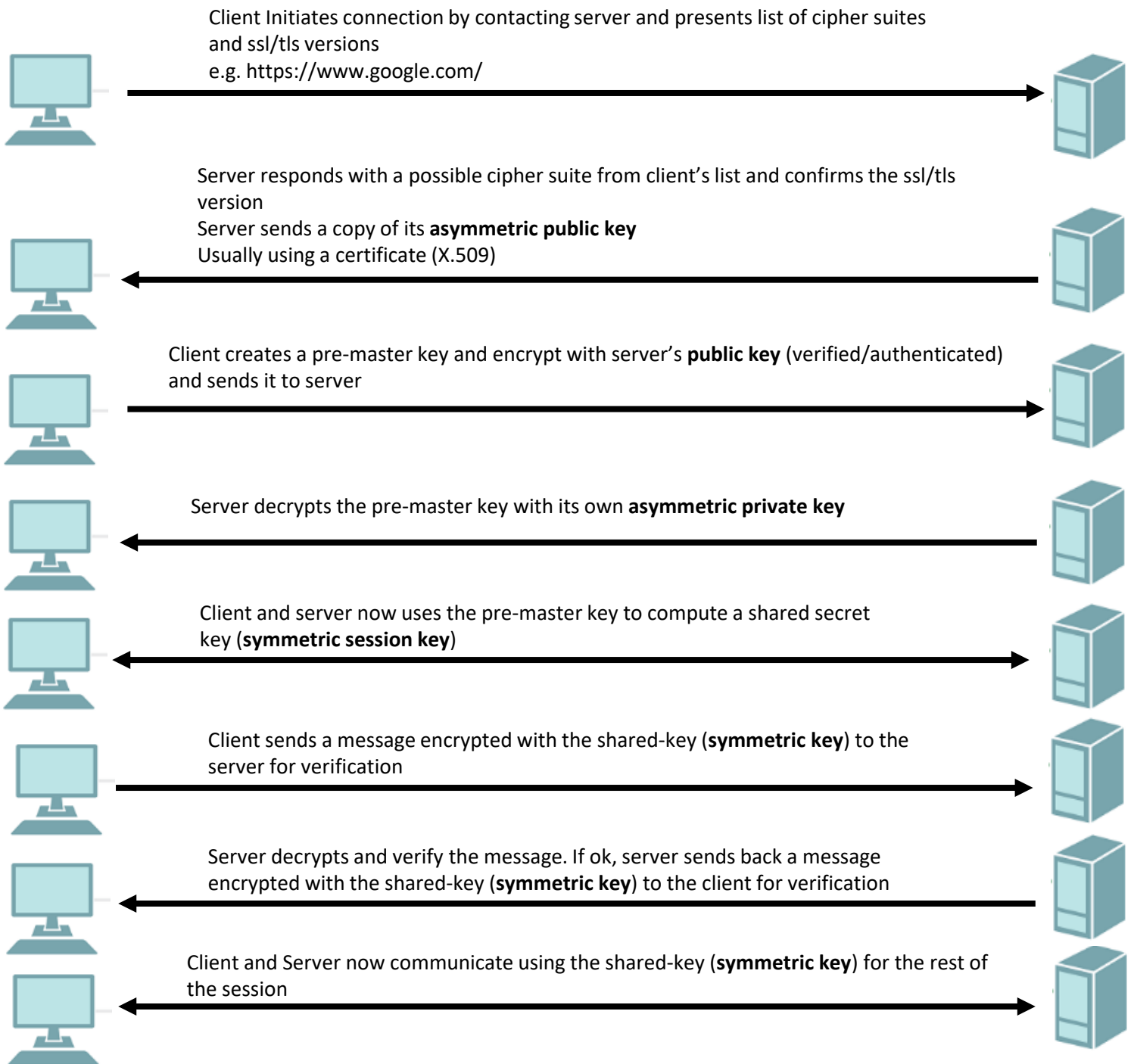
SSH-CONNECT
SSH-USERAUTH
SSH-TRANS



HTTPS (SSL/TLS protocol) Illustrated

Secure Socket Layer/Transport Layer Security

- Authentication
- Confidentiality


Client Initiates connection by contacting server and presents list of cipher suites and ssl/tls versions
e.g. https://www.google.com/
Server responds with a possible cipher suite from client's list and confirms the ssl/tls version
Server sends a copy of its **asymmetric public key**
Usually using a certificate (X.509)
Client creates a pre-master key and encrypt with server's **public key** (verified/authenticated) and sends it to server
Server decrypts the pre-master key with its own **asymmetric private key**
Client and server now uses the pre-master key to compute a shared secret key (**symmetric session key**)
Client sends a message encrypted with the shared-key (**symmetric key**) to the server for verification
Server decrypts and verify the message. If ok, server sends back a message encrypted with the shared-key (**symmetric key**) to the client for verification
Client and Server now communicate using the shared-key (**symmetric key**) for the rest of the session

Relevant Algorithms for asymmetric cryptography

- Asymmetric cryptography often rely on cryptographic algorithms based on mathematical problems that currently admit no efficient solution such as:
  - Integer-Factorization Schemes
    - Difficulty to factor large integers (e.g. RSA)
  - Discrete Logarithm Schemes
    - Based on discrete logarithm problem in finite fields (e.g. Diffie-Hellman key exchange, Elgamal encryption or the Digital Signature Algorithm)
  - Elliptic Curve (EC) Schemes
    - Generalization of the discrete logarithm algorithm (e.g. Elliptic Curve Diffie-Hellman key exchange (ECDH) and Elliptic Curve Digital Signature Algorithm (ECDSA)

Bits length and security

Algorithm family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

Key length	Security estimation
56 – 64 bits	short term: a few hours or days
112 – 128 bits	Long term: several decades in the absence of quantum computers
256 bits	Long term: several decades, even with quantum computers with currently known quantum computing algorithm

Source: Paar and Pelzl "Understanding Cryptography: A Textbook for Students and Practitioners

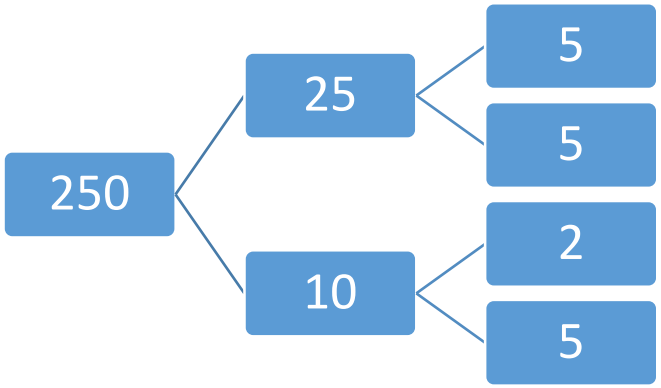
A RSA example

- The RSA algorithm is based on Euler's theorem
  - A Swiss Mathematician (Leonard Euler)
- We will now look at important components of RSA algorithms
- Suppose $\boldsymbol{n}$ - is a natural number
- $\boldsymbol{\Phi(n)}$ = Number of positive integers smaller than or equal to $\boldsymbol{n}$ with no common factor except $\mathbf{1}$ in common with $\boldsymbol{n}$
  - Search for all positive integers smaller than $\boldsymbol{n}$ and relatively prime to $\boldsymbol{n}$
  - Their gcd with $\boldsymbol{n}$ equals $\mathbf{1}$

Prime

prime number is an integer (a whole number) that has as its only factors 1 and itself (for example, 2, 17, 23, and 127 are prime)

- Unique prime factorization theorem
  - states that every integer greater than 1 either is a prime number itself or can be represented as the product of prime numbers



Factors $250 = 2 \times 5 \times 5 \times 5 = 2 \times 5^3$

Primes and Finding inverse

- Euler’s Phi Function
- Euler’s Theorem
- Euclidean Algorithm
- Extended Euclidean Algorithm

Euler's Phi Function

- Important to determine how many numbers in a given integer set $\boldsymbol{s}$ are prime to $\boldsymbol{s}$: $\boldsymbol{\Phi(s)}$

Example #1: if s = 5, associated set is {0,1,2,3,4}

gcd(0,5) = 5
gcd(1,5) = 1*
gcd(2,5) = 1*
gcd(3,5) = 1*
gcd(4,5) = 1*

$\boldsymbol{\Phi(5) = 4}$ $(4\ integers\ are\ coprime\ to\ 5)$

Example #2: if s = 6, associated set is {0,1,2,3,4,5}

gcd(0,6) = 6
gcd(1,6) = 1*
gcd(2,6) = 2
gcd(3,6) = 3
gcd(4,6) = 2
gcd(5,6) = 1*

$\boldsymbol{\Phi(6) = 2}$ $(2\ integers\ are\ coprime\ to\ 6)$

Let $\boldsymbol{s}$ have the following canonical factorization

$$s = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n}$$

Where the p_i are distinct prime numbers and e_i are positive integers, then:

$$\boldsymbol{\Phi(s) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1})}$$

Example: Let s = 250

$$s = 250 = 10 \cdot 25 = 2 \cdot 5 \cdot 5 \cdot 5 = 2.5^3 = p_1^{e_1} \cdot p_2^{e_2}$$

Given 2 distinct prime factors. Then:

$$\boldsymbol{\Phi(s) = \left(2^1 - 2^0\right) \cdot \left(5^3 - 5^2\right) = 2 \cdot 100 = 200}$$

There are 200 integers within the range {0,1,…,249} that are coprime to 250

Other properties of Euler's Phi Function

1. If p denotes a prime number, then $\boldsymbol{\Phi(p) = p - 1}$: because all the *p-1* integers *1,2,3,…,p-1* are relatively prime to $\boldsymbol{p}$

   e.g. $\boldsymbol{\Phi(5) = 4}$ $\boldsymbol{and}$ $\boldsymbol{\Phi(6) = 2}$

2. If p and q are two distinct primes, then:

   $\boldsymbol{\Phi(pq) = (p - 1)(q - 1)}$

Euler's Theorem

If ***a*** and ***n*** are two relatively prime positive integers (i.e. gcd(a,n) = 1):

$$a^{\Phi(n)}\ mod\ n \equiv 1$$

Euler's Theorem

We are interested in the case when ***n*** is the product of two distinct primes, ***p*** and ***q***

Since $\Phi(n) = (p-1)(q-1)$

$a^{\Phi(n)} mod\ n = a^{\Phi(p-1)(q-1)}\ mod\ pq \equiv 1$

Euler's Theorem: Example

Let a=5 and n=6

$5^{\Phi(6)} mod\ 6 = ?$

$\Phi(6) = \Phi(2 \times 3) = (2-1)(3-1) = 1 \times 2 = 2$

$5^2\ mod\ 6 = 25\ mod\ 6 = 1$

Further dependencies of RSA

- RSA is also based on the problem that it is extremely hard to factor very large numbers
- But it is easy to compute the greatest common divisor (gcd) using the Euclidean Algorithm

Euclidean Algorithm

- Euclidean Algorithm can be used to compute the ***gcd*** of two positive numbers ***a*** and ***b***

Example:
$a = 84 = 2 \times 2 \times 3 \times 7$
$b = 30 = 2 \times 3 \times 5$
$\gcd(30,84) = 2 \times 3 = 6$

For large numbers, manual factoring is often not possible
We therefore use the Euclidean Algorithm

This algorithm is based on a simple observation that:
$\gcd(a, b) = \gcd(a - b, b)$ *we assume* $a > b$ *and* $a, b \in \mathbb{Z}^+$

Euclidean Algorithm

$$\gcd(r_0, r_1) = \gcd(r_0 - r_1, r_1)$$

$r_0 = 84$
$r_1 = 30$
$r_0 - r_1 = 54 = 2 \times 3 \times 3$
$r_1 = 30 = 2 \times 3 \times 5$
$\gcd(30,54) = \gcd(30,84) = 2 \times 3 = 6$

By recursively applying the above,
we will eventually obtain

gcd (r_l, 0) = r_l

Example

1	gcd(84, 30)
2	gcd(54, 30)
3	gcd(30, 24)
4	gcd(24, 6)
5	gcd(18, 6)
6	gcd(12, 6)
7	gcd(6, 6)
8	gcd(6, 0)
	= 6

Euclidean Algorithm

- It follows that we can minimize the steps, if we use the modulus system

$$\gcd(r_0, r_1) = \gcd(r_1, r_0 \; mod \; r_1)$$

1	gcd(30, 84 mod 30)
	gcd(30, 24)
2	gcd(24, 30 mod 24)
	gcd(24, 6)
3	gcd(6, 24 mod 6)
	gcd(6, 0)
	= 6

Euclidean Algorithm
Input: $positive\ integers\ r_0\ and\ r_1\ with\ r_0 > r_1$
Output: $\gcd(r_0, r_1)$
Initialization: $i = 1$
Algorithm

1. *DO*
1.1 $i = i + 1$
1.2 $r_i = r_{i-2} \; mod \; r_{i-1}$
WHILE $r_i \neq 0$
2. *RETURN*
$\gcd(r_0, r_1) = r_{i-1}$

Exercise (3mins)

Solve:

$\gcd(r_0, r_1)$, where r_0= 95 and $r_1 = 35$

using: $\gcd(r_0, r_1) = \gcd(r_1, r_0 \ mod \ r_1)$

Solution

1	gcd(35, 95 mod 35)
	gcd(35, 25)
2	gcd(25, 35 mod 25)
	gcd(25, 10)
3	gcd(10, 25 mod 10)
	gcd(10, 5)
4	gcd(5, 10 mod 5)
	gcd(5, 0)
	= 5

Extended Euclidean Algorithm (EEA)

- Extension of Euclidean Algorithm
- Efficient to solve **Modular Inverse** problem which is important in public key cryptography

EEA extends the gcd computation by also computing a linear combination of the form:
$\gcd(r_0, r_1) = \mathrm{s}.r_0 + \mathrm{t}.r_1$
Thus, in each iteration of EA, the remainder $r_i = \mathrm{s}.r_0 + \mathrm{t}.r_1$

Recall that the inverse of two integers (r_0, r_1) only exist if:

$$\gcd(r_0, r_1) = 1$$

Assume we want to compute the inverse of $r_1 \; mod \; r_0$ (where $r_0 > r_1$)
Note that the inverse only exists if:

$$\gcd(r_0, r_1) = 1$$

If we apply EEA:

$$\mathrm{s}.r_0 + \mathrm{t}.r_1 = 1 = \gcd(r_0, r_1)$$

Reducing the equation modulo r_0, we obtain:

$$\mathrm{s}.r_0 + \mathrm{t}.r_1 = 1$$
$$\mathrm{s}.0 \; + \mathrm{t}.r_1 \equiv 1 \; mod \; r_0$$
$$r_1.t \equiv 1 \; mod \; r_0$$

This implies that $\boldsymbol{t}$ is the inverse of $\boldsymbol{r_1}$:

$$\boldsymbol{t = r_1^{-1} \; mod \; r_0}$$

Calculate $28^{-1}\ mod\ 75$

i.e. Find the inverse of 28 in modulo 75

$r_0 = 75$ and $r_1 = 28$

	$\boldsymbol{r_i = [s].r_0 + [t].r_1}$
$75 = 2\ \times\ 28 + \mathbf{19}$	$19 = [1].r_0 + [-2].r_1$
$28 = 1\ \times\ 19 + \mathbf{9}$	$9 = 28 - 1\ \times\ 19$ $9 = r_1 - 1\ \times ([1].r_0 + [-2].r_1)$ $9 = -r_0 + 3.r_1$
$19 = 2\ \times\ 9 + \mathbf{1}$	$1 = 19 - 2\ \times\ 9$ $1 = [1].r_0 + [-2].r_1 - 2 \times (-r_0 + 3.r_1)$ $= 3.r_0 - 8.r_1$
$9 = 9\ \times\ 1 + 0$	

Verify: $3.r_0 - 8.r_1 = 3 \times\ 75\ - 8\ \times\ 28 = 1$

$\boldsymbol{28^{-1}\ mod\ 75 = -8\ mod\ 75 = 67}$

Remember: -8 = 75-8 = 67 (mod 75)

Extended Euclidean Algorithm **(EEA)**

Input: *positive integers* r_0 *and* r_1 *with* $r_0 > r_1$

Output: $\gcd(r_0, r_1)$ *and* s, t *such that* $gcd(r_0, r_1) = \mathrm{s}.r_0 + \mathrm{t}.r_1$

Initialization:

$s_0 = 1 \quad t_0 = 0$

$s_1 = 0 \quad t_1 = 1$

$i = 1$

Algorithm

1. DO

1.1 $i = i + 1$

1.2 $r_i = r_{i-2} \; mod \; r_{i-1}$

1.3 $q_{i-1} = (r_{i-2} - r_i) \; / \; r_{i-1}$

1.4 $s_i = s_{i-2} - q_{i-1} \times s_{i-1}$

1.5 $t_i = t_{i-2} - q_{i-1} \times t_{i-1}$

WHILE $r_i \neq 0$

2. *RETURN*

$\gcd(r_0, r_1) = r_{i-1}$

$s = s_{i-1}$

$t = t_{i-1}$

Sum-up

- We can easily compute the gcd of two integers, and decide whether they are relatively prime.
- We can easily compute the inverse of an integer using EEA
- If $\boldsymbol{a}$ and $\boldsymbol{b}$ are coprime, then we can easily compute an integer $\boldsymbol{d}$ that satisfies:

  $\boldsymbol{b \cdot d \bmod a = 1}$.

  $\boldsymbol{b \cdot d = 1 \ (mod\ a)}$.

  It then means $\boldsymbol{d}$ is the multiplicative inverse of $\boldsymbol{b}$ in modulo system $\boldsymbol{a}$

The RSA Public Key Cryptography - protocol

- Most widely used public key cryptosystem
- Named after Ron **Rivest**, Adi **Shamir**, and Len **Adleman**

Process:

1. Key generation
2. Encryption and Decryption

RSA Key Generation

- Derive -
  - Public Key: $k_{pub} = (n, e)$
  - Private Key: $k_{pr} = (n, d)$

STEPS:

1. $Choose\ two\ large\ primes\ \boldsymbol{p}\ and\ \boldsymbol{q}$
2. $Compute\ \boldsymbol{n} = \boldsymbol{p} \cdot \boldsymbol{q}$
3. $Compute\ \boldsymbol{\Phi(n)} = (\boldsymbol{p} - \boldsymbol{1})(\boldsymbol{q} - \boldsymbol{1})$
4. $Select\ the\ public\ exponent\ \boldsymbol{e}\ \in \{\boldsymbol{1}, \boldsymbol{2}, \ldots, \boldsymbol{\Phi(n)} - \boldsymbol{1}\}\ such\ that\ \boldsymbol{gcd}\big(\boldsymbol{e}, \boldsymbol{\Phi(n)}\big) = \boldsymbol{1}$
5. $Compute\ the\ private\ key\ \boldsymbol{d}\ such\ that: \boldsymbol{d} \cdot \boldsymbol{e} \equiv \boldsymbol{1}\ \boldsymbol{mod}\ \boldsymbol{\Phi(n)}$

Step #4 is very important: Otherwise we cannot compute $\boldsymbol{d}\ \in \{\boldsymbol{1}, \boldsymbol{2}, \ldots, \boldsymbol{\Phi(n)} - \boldsymbol{1}\}$

RSA Encryption and Decryption

- RSA Encryption
  - Given the Public Key: $k_{pub} = (n, e)$ and the plaintext $\boldsymbol{x}$, the encryption function is:
    - $y = e_{k_{pub}}(x) \equiv x^e \ mod \ n$

- RSA Decryption
  - Given the Private Key: $k_{pr} = (n, d)$ and the ciphertext $\boldsymbol{y}$, the decryption function is:
    - $x = d_{k_{pr}}(y) \equiv y^d \ mod \ n$

Example

- Alice wants to send an encrypted message to Bob.
  - Bob computes his RSA key parameters (Public/Private keys).
  - Then Bob sends Alice his public key.
  - Alice encrypts the message, ***x*** and sends the ciphertext ***y*** to Bob.
  - Bob decrypts the ciphertext ***y*** using his private key.

Alice

Message x = 4

Bob

1. Choose p=3 and q=11
2. n=p.q=33
3. $\Phi(n)$=(3-1)(11-1)=20
4. Choose e=3
5. $d \equiv e^{-1} \equiv 7\ mod\ 20$

k_{pub}=(33,3) and k_{pr}=(33,7)

(n, e)

k_{pub}=(33,3)

$y = x^e \bmod n$

$= 4^3 \bmod 33 = 31$

y = 31

$x = y^d \bmod n$

$= 31^7 \bmod 33 = 4 = x$

Exercise – 5mins

- Assume you know that ***n=91*** and the public key ***e=5***
  - Use integer factorization to determine the private key ***d***.

Solution

Step 1. Factors: 91 = 7 * 13

Brute-force by choosing the factors in pairs

Step 2. Find $\boldsymbol{\Phi(n)}$ = (p-1)(q-1)

$\boldsymbol{\Phi(n)}$ = 6*12 = 72

Step 3. Is gcd(e, $\boldsymbol{\Phi(n)}$) = 1?

gcd(5, 72) = 1

Proceed if Step 3 succeeds

Step 4: $\boldsymbol{Compute\ the\ private\ key\ d\ such\ that: d \cdot e \equiv 1\ mod\ \Phi(n)}$

i.e. $\boldsymbol{d \equiv e^{-1}\ mod\ \Phi(n)}$

We now find $\mathbf{d = 5^{-1}\ mod\ 72}$

$\boldsymbol{d = 29}$ (We have found the private key)

Mathematical attacks on RSA

RSA factoring records since 1991

Decimal digits	Bit length	Date
100	330	April 1991
110	364	April 1992
120	397	June 1993
129	426	April 1994
140	463	Feb. 1999
155	512	Aug. 1999
200	663	May 2005
232	768	Dec. 2009

See: https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

Fast exponentiation: Square-and-Multiply method

- How do we deal with very large exponentiation
  - Computational problem if we need to compute e.g. 2^{1024} by multiplication method
- Works using two basic operations
  1. Square the current result
  2. Multiply the current result by the base element

2^{10} = 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2

2^{1024} = 2 * 2 * 2 *…

- Algorithm is based on scanning the bit of the exponent from the left bit (MSB) to the right (LSB)
- For each iteration (1st bit to the last bit),
  - the current result is squared
  - if the current bit has value 1, then the result is also multiplied by base number

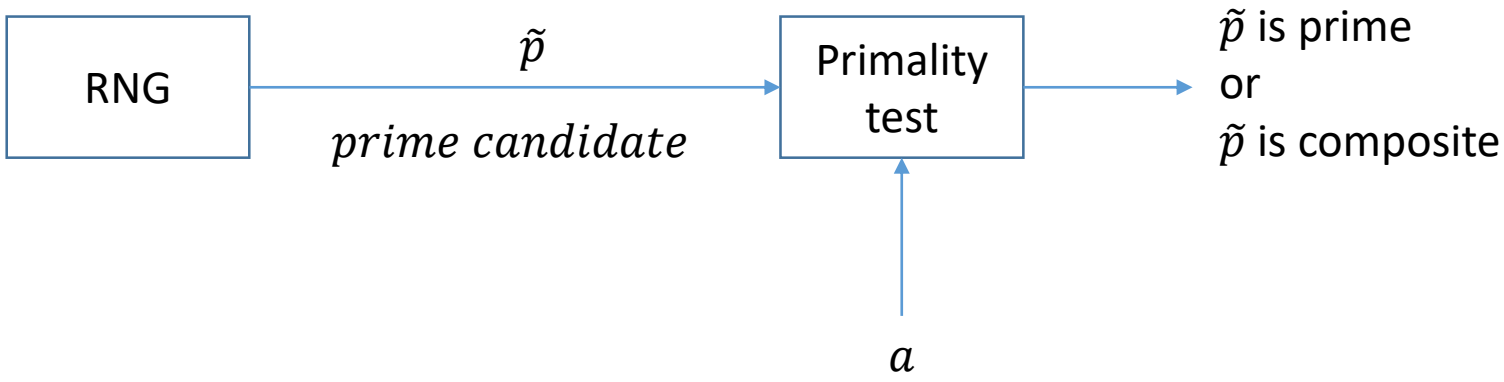
Example

$x^{26} = x^{11010_2} = x^{(h_4 h_3 h_2 h_1 h_0)_2}$

Step		Bit processed
0	$x = x^1$	$h_4 = 1$
1a	$(x^1)^2 = x^2 = x^{10_2}$	h_3: SQ
1b	$x^2 \cdot x = x^3 = x^{11_2}$	$h_3 = 1$: MUL
2a	$(x^3)^2 = x^6 = x^{110_2}$	h_2: SQ
2b		$h_2 = 0$: **no** MUL
3a	$(x^6)^2 = x^{12} = x^{1100_2}$	h_1: SQ
3b	$x^{12} \cdot x = x^{13} = x^{1101_2}$	$h_1 = 1$: MUL
4a	$(x^{13})^2 = x^{26} = x^{11010_2}$	h_0: SQ
4b		$h_0 = 0$: **no** MUL

Finding Large Primes

- Generating primes ***p*** and ***q***
- How common are primes?
- How fast can we check for primes?



How common are primes?

- We need to know whether the probability of a randomly picked prime p is sufficiently high
- Primes become less dense as the value increases
  - 2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,…
- $P(p\ is\ prime) \approx \frac{2}{\ln(p)}$
- Example: To have a 1024-bit modulus n, the primes p and q should have lengths of about 512 bits each (i.e. $p, q \approx 2^{512}$)
- $P(p\ is\ prime) \approx \frac{2}{\ln(2^{512})} = \frac{2}{512\ln(2)} \approx \frac{1}{177}$

Means we need to test approx. 177 random numbers before we find one that is prime

How fast can we check for primes?

Difficult problem!!!

Naive approach

```
boolean isPrime(integer n)
 if (n < 2) return false
 for(i=2 to n-1)
   if(n/i)
        return false
 return true
```

Time complexity = O(n)

e.g. Test 2^{64} for primality
$= 1.8*10^{19}$

Time complexity = $O(10^{19})$

How fast can we check for primes

Fermat's Little Theorem for Primality Testing

If $\boldsymbol{p}$ is a prime and $\boldsymbol{a}$ is any integer not divisible by $\boldsymbol{p}$ then:

$$a^p \equiv a \ (mod\ p)$$

Can further be stated in the form:

$$a^{p-1} \equiv 1 \ (mod\ p)$$

p is prime if $a^{p-1} \equiv 1 \ (mod\ p)$

Check:
p=7
$a \in \{2,3,4,5\}$
a=2
$a^{p-1}\ mod\ p = \ 2^6\ mod\ 7$
64 mod 7= 1

a=3
$a^{p-1}\ mod\ p = \ 3^6\ mod\ 7$
729 mod 7= 1
.
.
.

Algorithms based on Discrete Logarithm

- Diffie-Hellman Key Exchange
- Elgamal
- DSA

Discrete Logarithm

Consider y = 2^x mod 11 (11 is prime)

x=1, y=2	x=6, y=9
x=2, y=4	x=7, y=7
x=3, y=8	x=8, y=3
x=4, y=5	x=9, y=6
x=5, y=10	x=10, y=1

$3^1 = 3 = 3^0 \times 3 \equiv 1 \times 3 = 3 \equiv 3 \ (mod\ 7)$
$3^2 = 9 = 3^1 \times 3 \equiv 3 \times 3 = 3 \equiv 2 \ (mod\ 7)$
$3^3 = 27 = 3^2 \times 3 \equiv 2 \times 3 = 3 \equiv 6 \ (mod\ 7)$
$3^4 = 81 = 3^3 \times 3 \equiv 6 \times 3 = 3 \equiv 4 \ (mod\ 7)$
$3^5 = 243 = 3^4 \times 3 \equiv 4 \times 3 = 3 \equiv 5 \ (mod\ 7)$
$3^6 = 729 = 3^5 \times 3 \equiv 5 \times 3 = 3 \equiv 1 \ (mod\ 7)$
$3^7 = 2187 = 3^6 \times 3 \equiv 1 \times 3 = 3 \equiv 3 \ (mod\ 7)$

Each non-zero value in the set Z_{11}=

{0,1,2,3,4,5,6,7,8,9,10} is a value of x for some y.

In this case, 2 is a **primitive root** of the set

Z_{11}= {1,2,3,4,5,6,7,8,9,10}.

$The\ period\ of\ 2^x\ mod\ 11 = 10$

$The\ period\ of\ 3^x\ mod\ 7 = 6$

Sources: wikipedia
homepage.divms.uiowa.edu/~ghosh/1914F-6.pptx

Discrete Logarithm

A **primitive root** modulo a prime number **p** is an integer **b in Z_p**

such that **every nonzero element of Z_p is a power of b**

$Z_p = \{0, 1, 2, 3, 4, \ldots, p-1\}$

$y = b^e \bmod p$

where $b \in Z_p$

In other words, b is a generator of the multiplicative group of integers modulo p.

Discrete Logarithm

- Given $\mathbf{2^x \pmod{11} = y}$ in the set {1,2,3,4,5,6,7,8,9,10} , you can find a unique value of x for a given y.
- We will say that x is the **discrete logarithm** of y (mod 11) (to the base 2).

- It is easy to compute $2^x \pmod{11} = y$.
- But it is extremely difficult to solve $\mathbf{2^? \pmod{11} = y}$.
- This is the **discrete logarithm** problem. For example, how difficult is it to compute $\mathbf{g^? \bmod p = q}$ where p is a 100 digit prime number?
- This is the heart of DL-based cryptography.

Discrete Logarithm Problem (DLP)

- Discrete logarithm is an inverse operation in modulo arithmetic

$$An\ integer\ \boldsymbol{k}\ that\ solves\ the\ equation\ \boldsymbol{b^k = a}\ is\ termed\ a\ discrete\ logarithm$$

We can further write in logarithm form:

$$k = log_b a$$

- Considered to be computationally hard problem
  - That is, no efficient method is known for computing them in general

Diffie-Hellman Key Exchange

- Enables two parties to derive a common secret key by communicating over an insecure channel
- Solves the key distribution problem
- Used in many open/commercial crypto protocols
  - Secure Shell (SSH)
  - Transport Layer Security (TLS)
  - Internet Protocol Security (IPSec)
- Based on primitive root element and cyclic group
- Basic idea is that exponentiation in $\mathbb{Z}_p^*$, p prime is a one way function and exponentiation is commutative

$$\boldsymbol{k = (N^x)^y \equiv (N^y)^x \bmod p}$$

- The value $k = (N^x)^y \equiv (N^y)^x \bmod \mathrm{p}$ is the joint key to be used as the session key between the two parties

Alice

Bob

Agree and choose prime number, ***p*** and a base number ***N (primitive root)***

Choose a secret exponent, x :
$x \in \{2, \dots, p-2\}$

Choose a secret exponent,
$y : y \in \{2, \dots, p-2\}$

Computes A $\equiv N^x mod\ p$

Computes B $\equiv N^y mod\ p$

Alice sends ***A*** to Bob

Bob sends ***B*** to Alice

Computes $k_{A,B} \equiv B^x mod\ p$

Computes $k_{B,A} \equiv A^y mod\ p$

$k = k_{A,B} \equiv k_{B,A}$ ***(Surprise!!!)***

Both Alice and Bob now have a joint session key ***k***

Alice

Bob

Agree and choose prime number, ***p=29*** and a base number ***N=2***

Choose a secret exponent,
$\boldsymbol{x = 5\ [x : x \in \{2, \dots, 27\}]}$

Choose a secret exponent,
$\boldsymbol{y = 12[y : y \in \{2, \dots, 27\}]}$

Computes A $\equiv 2^5 \equiv 3\ mod\ 29$

Computes B $\equiv 2^{12} \equiv 7\ mod\ 29$

Alice sends ***A=3*** to Bob

Bob sends ***B=7*** to Alice

Computes $k_{A,B} = B^x = 7^5 \equiv 16\ mod\ 29$

Computes $k_{B,A} = A^y = 3^{12} \equiv 16\ mod\ 29$

$k = 16$

Both Alice and Bob now have a joint session key ***k=16***

Why do they get the same key?

- Proof (commutative property):
- Alice computes:
  - $B^x \equiv (N^y)^x \equiv N^{xy} mod\ p$
- Bob computes:
  - $A^y \equiv (N^x)^y \equiv N^{xy} mod\ p$

Question: Can someone compute k given that A, B, N,and p are all known?

Elgamal Encryption Protocol

Alice

Message x

Bob

1. Choose large prime ***p***
2. Choose primitive element $\alpha \in \mathbb{Z}_p^*$
   $or\ in\ a\ subgroup\ of\ \mathbb{Z}_p^*$
3. Choose $k_{pr} = d \in \{2,\ldots,p\text{-}2\}$
4. $k_{pub} \equiv \beta = \alpha^d\ mod\ p$

$k_{pub}=(p, \alpha, \beta)$

1. Choose $i \in \{2,\ldots,p\text{-}2\}$
2. Compute ephemeral key
   $k_E \equiv \alpha^i\ mod\ p$
3. Compute masking key
   $k_M \equiv \beta^i\ mod\ p$
4. $Encrypt\ message\ x$
   $y \equiv x.k_M\ mod\ p$

(k_E, y)

5. Compute masking key
   $k_M \equiv k_E^d\ mod\ p$
6. Decrypt y
   $x \equiv y.k_M^{-1}\ mod\ p$

Alice

Bob

Message x=26

Generate ***p=29*** *and* ***α=2***
Choose $k_{pr} = d = 12$
Compute $\beta = \alpha^d \equiv 7\ mod\ 29$

k_{pub}=(p, α, β)=(29,2,7)

Choose $i = 5$
Compute $k_E = \alpha^i \equiv 3\ mod\ 29$
Compute $k_M \equiv \beta^i \equiv 16\ mod\ 29$
$Encrypt\ y \equiv x.k_M \equiv 10\ mod\ 29$

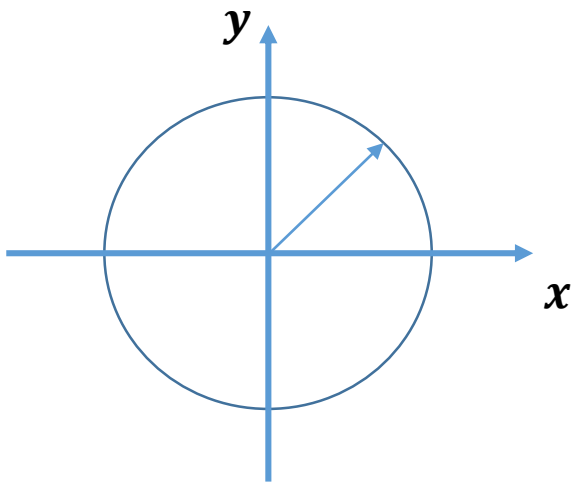
(k_E,y)=(3,10)

Compute $k_M = k_E^d \equiv 16\ mod\ 29$
Decrypt y
$x = y.k_M^{-1}\ 10.20 \equiv 26\ mod\ 29$

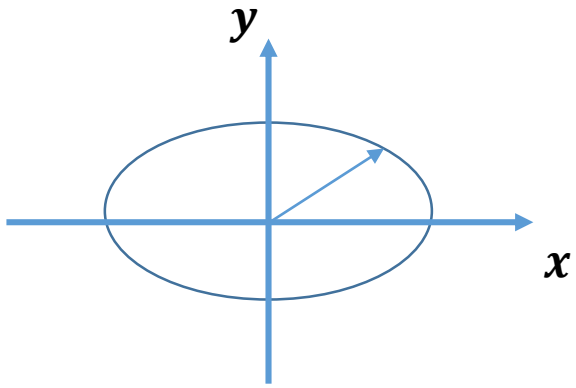
- In Elgamal protocol
  - Alice sends only one message to Bob as opposed to 2 in DHKE
  - Elgamal is a probabilistic encryption scheme
    - i.e. encrypting x_1 using the same public keys produce two different ciphertexts $y_1 \neq y_2$. Because i is chosen at random from {2,. . . ,p-2}

Elliptic Curve Cryptography

- Uses smaller key-size compared to RSA and Elgamal
- Based on generalized discrete logarithm problem
- Cyclic group
- One-way properties
- It is possible to realize DL-protocols such as DHKE using elliptic curves



$x^2 + y^2 = r^2 \; over \; \mathbb{R}$



$a.x^2 + b.y^2 = c \; over \; \mathbb{R}$

We obtain an Ellipse by introducing constants **a** and **b** to the circle equation

The promise of Elliptic curves

Algorithm family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

- We can therefore form certain types of curves from polynomial equations
- Curves are considered over a finite field
- F(p)
  - Arithmetic is performed modulo a prime ***p***.

The elliptic curve over $\mathbb{Z}_p, p > 3, is\ the\ set\ of\ all\ pairs\ (x, y) \in \mathbb{Z}_p$

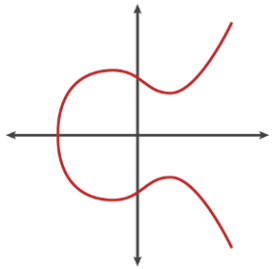
that fulfills

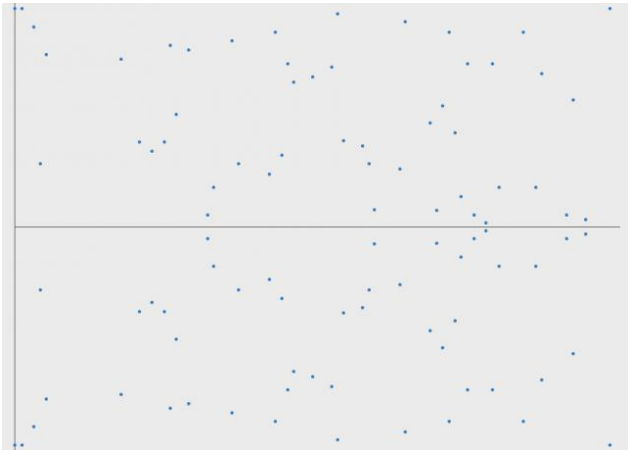
$$y^2 \equiv x^3 + a.x + b\ mod\ p$$

together with infinity point O, where

$$a, b \in \mathbb{Z}_p$$

with the condition that $4.a^3 + 27.b^2 \neq 0\ mod\ p$.





Discrete Logarithm Problem on ECC

- The security of ECC depends on the difficulty of Elliptic Curve Discrete Logarithm Problem (ECDLP)
- If P and Q are two points on an elliptic curve such that:
  - $kP = Q \; where \; k \; is \; a \; scalar$
  - Given P and Q, it is computationally infeasible to obtain k, if k is sufficiently large.
  - k is the discrete logarithm of Q to the base P
- The main operation involved in ECC is point multiplication
  - Multiplication of a scalar k with any point P on the curve to give Q on the curve

Finite Fields for EC

- Prime Field F_p
- Binary Field F_2^m

Elliptic Curves over prime field F_p

1. Let $F_p\ be\ a\ prime\ finite\ field$: p is an odd prime
2. Let $a, b \in F_p$ satisfy $4.a^3 + 27.b^2 \not\equiv 0\ (mod\ p)$
3. Then an elliptic curve $E(F_p)\ over\ F_p\ defined\ by\ the\ parameters$ $a, b \in F_p$ consists of the set of solutions or points $\mathrm{P} = (x, y)\ for\ x, y \in F_p$ to the equation:

$$E: y^2 \equiv x^3 + a.x + b\ (mod\ p)$$

   together with an extra point O called point at infinity.
4. The equation $y^2 \equiv x^3 + a.x + b\ (mod\ p)$ is called the defining equation of $E(F_p)$.
5. For a given point $\mathrm{P} = \left(x_p, y_p\right), x_p$ is the x-coordinate of P, and y_p is the y-cordinate of P
6. Number of points on $E(F_p)$ is denoted by $\#E(F_p)$
   - $p + 1 - 2\sqrt{p}\ \leq \#E(F_p) \leq p + 1 + 2\sqrt{p}.$ (Hasse Theorem)

Arithmetic Operations on prime Field F_p

1. **Adding a point at infinity to itself**:
   $O + O = O$.

2. **Adding a point at infinity to any other point:**
   $(x, y) + O = O + (x, y) = (x, y) \; \forall \; (x, y) \in E(F_p)$.

1. **Adding two points with the same x-coordinates when the points are either distinct or have y=0:**

   $(x, y) + (x, -y) = O \; \forall \; (x, y) \in E(F_p)$.

   i.e the negative of the point (x, y) is $-(x, y)$=$(x, -y)$

4. Adding two points with different x-coordinates

Let
$(x_1, y_1) \in E(F_p)$ and $(x_2, y_2) \in E(F_p)$ be two points such that $x_1 \neq x_2$. Then
$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where:

$$x_3 = s^2 - x_1 - x_2 \ (mod\ p)$$
$$y_3 = s \cdot (x_1 - x_3) - y_1 \ (mod\ p)$$

where s is the slope:

$$s = \frac{y_2 - y_1}{x_2 - x_1} \ (mod\ p)$$

5. Adding a point to itself (point doubling)

Let
$(x_1, y_1) \in E(F_p)$ be a point such that $y_1 \neq 0$. Then
$(x_1, y_1) + (x_1, y_1) = (x_3, y_3)$, where:

$$x_3 = s^2 - 2 \cdot x_1 \ (mod\ p)$$
$$y_3 = s \cdot (x_1 - x_3) - y_1 \ (mod\ p)$$

where s is computed as:

$$s = \frac{3 \cdot x_1^2 + a}{(2 \cdot y_1)} (mod\ p)$$

Example

Double the point P = (5,1) on curve ***E*** defined over a small field F_{17} $(p = 17), a = 2\ and\ b = 2$

$$\boldsymbol{E}: y^2 \equiv x^3 + 2x + 2\ (mod\ 17)$$

First we compute s:

$s = (2 \cdot 1)^{-1}(3 \cdot 5^2 + 2) = 2^{-1} \cdot 9 = 9 \cdot 9 = 13\ (mod\ 17)$

Then

$x_3 = 13^2 - 2 \cdot 5\ (mod\ 17) = 6$

$y_3 = 13(5 - 6) - 1\ (mod\ 17) = -14 = 3\ mod\ 17$

2P = (5,1) + (5,1) = (6,3) (mod 17)

Find all points on the curve $\boldsymbol{E}: y^2 \equiv x^3 + 2x + 2 \ (mod\ 17)$.

Start with the primitive element P = (5,1)

$$\#\boldsymbol{E} = ?$$

2P = (5,1) + (5,1) = (6,3) (mod 17)

3P = 2P+P = (10,6)

4P = 3P+P = (3,1)

5P = (9,16)

.

.

.

18P = (5, 16)

19P = O

20P = 19P + P = O + P = P

21P = 20P +P = 2P

.

.

.

(Cyclic group)

$$\#\boldsymbol{E} = \mathbf{19}$$

- The set of points on $E(F_p)$ forms a group under the addition rule
- The group is commutative meaning that:
  - $P_1 + P_2 = P_2 + P_1 \; \forall \; P_1, P_2 \in E(F_p)$

- ECC relies on scalar multiplication of elliptic curve points
  - Given an integer k and a point $P \in E(F_p)$, scalar multiplication is the process of adding P to itself k times
  - i.e. $k \times P \; or \; kP$

Point Multiplication

- A point P on the elliptic curve is multiplied with a scalar k to obtain point Q (kP = Q)
- The multiplication operation uses:
  - Point addition (J + K = L)
  - Point doubling (2J = L)

  - Ex. If k = 23, then kP = 23.P = 2(2(2(2P) + P) + P) + P. (uses both point addition and point doubling repeatedly)

Elliptic Curves over binary Field F_2^m

- Involves arithmetic of integer of length m bits
- Considered as binary polynomial of degree m-1
- The binary string $(a_{m-1} \ldots a_1\ a_0)$ can be expressed as polynomial $a_m x^{m-1} + a_{m-1} x^{m-2} + \ldots + a_1 x + a_0\ where\ a_i = 0\ or\ 1$
- Ex. The 4 bit number 1101_2 can be expressed by the polynomial
  - $x^3 + x^2 + 1$. where $x = 2$
- Coefficients of the polynomial can either be 0 or 1
- Uses irreducible polynomial(s)

Elliptic Curves over binary Field F_2^m

1. Let $F_2^m be\ a\ characteristic\ 2\ finite\ field$
2. Let $a, b \in F_2^m$ satisfy $b \neq 0\ in\ F_2^m$
3. Then an elliptic curve $E(F_2^m)\ over\ F_2^m\ defined\ by\ the\ parameters\ a, b \in F_2^m$ consists of the set of solutions or points $\mathrm{P} = (x, y)\ for\ x, y \in F_2^m$ to the equation:

$$y^2 + x \cdot y = x^3 + a.x^2 + b\ (in\ F_2^m)$$

   together with an extra point O called point at infinity.

4. Number of points on $E(F_2^m)$ is denoted by $\#E(F_2^m)$
   - $2^m + 1 - 2\sqrt{2^m} \leq \#E(F_p) \leq 2^m + 1 + 2\sqrt{2^m}$. (Hasse Theorem)

Polynomial Arithmetic Operations over binary Field F_2^m

We use the field F_2^4 on irreducible polynomial $f(x) = x^4 + x + 1$

m=4

1. Addition

Given two polynomials $A = x^3 + x^2 + 1$ and $B = x^2 + x$

A + B (mod 2) = $x^3 + 2x^2 + x + 1\ (mod\ 2) = x^3 + 0 \cdot x^2 + x + 1$ **(since 2 mod 2 = 0)**

$A + B = x^3 + x + 1 \pmod 2$

$A = 1101_2$

$B = 0110_2$

$A + B = 1011_2$

2. Subtraction

Given two polynomials $A = x^3 + x^2 + 1$ and $B = x^2 + x$

A - B (mod 2) = $x^3 - x + 1\ (mod\ 2) = x^3 + x + 1$ **(since -1 mod 2 = 1)**

$A - B = x^3 + x + 1 \pmod 2$

$A = 1101_2$

$B = 0110_2$

$A - B = 1011_2$

3. Multiplication

Given two polynomials $\mathrm{A} = x^3 + x^2 + 1$ and $\mathrm{B} = x^2 + x$

A * B = $x^5 + 2x^4 + x^3 + x^2 + x\ (mod\ 2) = \ x^5 + 0 \cdot x^4 + x^3 + x^2 + \ x$
$A * B = x^5 + x^3 + x^2 + \ x\ (\mathrm{mod}\ 2)$

Since ***m=4***, we have to reduce the result to polynomial of degree less than ***m***
i.e. $x^5 + x^3 + x^2 + \ x\ (\mathrm{mod}\ f(x))$
$= (x^4 + x + 1)x + x^5 + x^3 + x^2 + \ x$
$= 2x^5 + x^3 + 2x^2 + 2x$
$= x^3\ (mod\ 2)$

$A = 1101_2$
$B = 0110_2$
$A * B = 1000_2$

4. Division

Given two polynomials $\mathrm{A} = x^3 + x^2 + 1$ and $\mathrm{B} = x^2 + x$
The division:
$\frac{A}{B}\big(mod\ f(x)\big) = A * B^{-1}(mod\ f(x))$. Where $B^{-1}$is the multiplicative inverse of B and$f(x)$ is the irreducible polynomial

<u>EEA can be used to compute the multiplicative inverse</u>

By introducing auxilliary polynomials $s(x)$ and $\mathrm{t}(x)$:

$$s(x)f(x) + t(x)B = \gcd(f(x), B) = 1$$

If we reduce modulo f(x), we obtain:

$$s(x).0 + t(x)B \equiv 1\ mod\ f(x)$$
$$t(x) \equiv B^{-1}\ mod\ f(x)$$

Irreducible polynomials

- Analogous to modulus p in modular arithmetic
- Polynomial of degree ***m*** that cannot be expressed as the product of two polynomials of lesser degree.

F_2^m *should have* (*according to standard*) $m \in \{113, 131, 163, 193, 233, 239, 283, 409, 571\}$

Field	Reduction Polynomials
F_2^{113}	$f(x) = x^{113} + x^9 + 1$
F_2^{131}	$f(x) = x^{131} + x^8 + x^3 + x^2 + 1$
F_2^{163}	$f(x) = x^{163} + x^7 + x^6 + x^3 + 1$
F_2^{193}	$f(x) = x^{193} + x^{15} + 1$
F_2^{233}	$f(x) = x^{233} + x^{74} + 1$
F_2^{239}	$f(x) = x^{239} + x^{36} + 1$ or $f(x) = x^{239} + x^{158} + 1$
F_2^{283}	$f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$
F_2^{409}	$f(x) = x^{409} + x^{87} + 1$
F_2^{571}	$f(x) = x^{571} + x^{10} + x^5 + x^2 + 1$

Domain parameters for F_p

- Sextuple $T = \{p, a, b, G, n, h\}$:
  - p = prime number defined for the finite field F_p
  - a, b are parameters defining the curve $y^2 \equiv x^3 + a.x + b \ (mod\ p)$
  - G is the base point (x_G, y_G)
  - n is the order of the elliptic curve (scalar k is chosen between 0 and n-1)
  - h is the cofactor $h = \frac{\#E(F_p)}{n}$, where $\#E(F_p)$ is the number of points on the curve

Domain parameters for F_2^m

- Septuple $T = \{m, f(x), a, b, G, n, h\}$:
  - m = length of bits of element in the finite field
  - f(x) is the irreducible polynomial of degree m
  - a, b are parameters defining the curve $y^2 + x \cdot y = x^3 + a.x^2 + b$
  - G is the base point (x_G, y_G)
  - n is the order of the elliptic curve (scalar *k* is chosen between 0 and n-1)
  - h is the cofactor $h = \frac{\#E(F_2^m)}{n}$, where $\#E(F_2^m)$ is the number of points on the elliptic curve

Elliptic Curve Diffie-Hellman Key Exchange (ECDH)

- Domain parameters for ECDH
- Choose a prime p and the elliptic curve $\boldsymbol{E}$: $y^2 \equiv x^3 + a \cdot x + b \ (mod \ p)$.
- Choose a primitive element $\mathrm{P} = (x_p, y_p)$.

Alice

Bob

1. Choose $k_{prA} = a \in \{2,3,\ldots,\#E-1\}$
2. Compute
   $k_{pubA} = aP = A = (x_A, y_A)$

1. Choose $k_{prB} = b \in \{2,3,\ldots,\#E-1\}$
2. Compute
   $k_{pubB} = bP = B = (x_B, y_B)$

$k_{pubA} = A$ →

← $k_{pubB} = B$

3. Compute: $aB = T_{AB}$

3. Compute: $bA = T_{AB}$

Joint secret between Alice and Bob: $T_{AB} = (x_{AB}, y_{AB})$

Example:
Consider ECDH with the following domain parameters
prime $p = 17$ and the elliptic curve $\boldsymbol{E}: y^2 \equiv x^3 + 2x + 2\ (mod\ 17)$.
which forms a cyclic group of order $\#\mathrm{E} = 19$. The base point $\mathrm{P} = (5,1)$.

Alice

Bob

1. Choose $k_{prA} = a = 3$
2. Compute $k_{pubA} = 3P = A = (10,6)$

1. Choose $k_{prB} = b = 10$
2. Compute $k_{pubB} = 10P = B = (7,11)$

$k_{pubA} = A$ →

← $k_{pubB} = B$

3. Compute: $aB = T_{AB} = 3(7,11) = (13,10)$

3. Compute: $bA = T_{AB} = 10(10,6) = (13,10)$

Why do they get the same key?

- Proof (Associative property):
- Alice computes:
  - $aB = a(bP)$
- Bob computes:
  - $bA = b(aP)$

Elliptic curve Diffie-Hellman Problem

- Solve either of the discrete logarithm problems:
- $a = log_p A$
- $b = log_p B$

Bit length and security of Elliptic Curve Cryptography

$$[log_2 p] \in \{192, 224, 256, 384, 521\}$$

we can rewrite as:

$$p \in \{2^{192}, 2^{224}, 2^{256}, 2^{384}, 2^{521}\}$$

Quiz – 5mins

1. Asymmetric cryptography uses:
   a) Public and private keys
   b) A secret key known to two parties
   c) Public keys
2. The security level of 128-bit asymmetric key is the same as 128-bit symmetric key
   a) True
   b) False
3. The security of RSA is based on the difficulty of
   a) Solving the logarithm problem
   b) Factoring large prime numbers
   c) Solving quadratic problem
4. A 4-bit (1101_2) message in an Elliptic curve can be expressed in polynomial form as:
   a) x^4+x^2+1
   b) x^3+x^2+x+1
   c) x^3+x^2+1

Protocols using asymmetric cryptography

- S/MIME
- GPG, an implementation of OpenPGP
- Internet Key Exchange
- PGP
- ZRTP, a secure VoIP protocol
- Secure Socket Layer, now codified as the IETF standard Transport Layer Security (TLS)
- SILC
- SSH
- Bitcoin
- Off-the-Record Messaging