# Topic 2.3.1 - Asymmetric Encryption

Eric Casanovas

Universitat d'Andorra

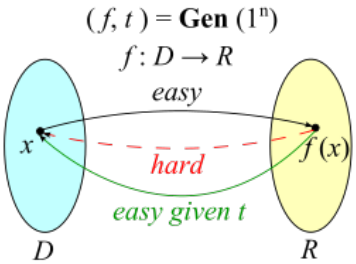
23rd February, 2022

## Basis

- The key used to encrypt and decrypt are **different**
- We have 2 keys, a secret key (**SK**) and a public key (**PK**)
- The SK must be known only by the owner of the pair of keys
- The PK can be sent to a 3rd person
- The keys are dependent on a "trapdoor function"
- It can be used for **Encrypt/Decrypt**, **Sign** and **Exchange Keys**

## Trapdoor functions I

- Function that is easy to perform one way, but has a **secret** that is required to perform the inverse calculation **efficiently**
- Are hash functions trapdoor functions?



## Trapdoor functions II

- Are hash functions trapdoor functions?
- No! Hash functions are One-way functions, they are not reversible
- Examples of trapdoor functions:
  - Integer (prime) factorization (RSA)
  - Discrete logarithm (We will see this trapdoor in Diffie-Hellman key exchange)

## Example trapdoor function (left to right)

- We have 2 prime numbers: 1093 and 1039
- Multiply this 2 numbers
- 1093 * 1039 = 1,135,627

## Example trapdoor function (right to left, no secret) I

- We have this number **N**: 1,135,627
- N is the product of 2 prime numbers
- How can we find these 2 prime numbers?

## Example trapdoor function (right to left, no secret) II

- How can we find these 2 prime numbers?
- You have to try all prime numbers from 1 to N/2 and check if N%i = 0 (i is a prime number)
- This is not efficient for large numbers

## Example trapdoor function (right to left, knowing secret)

- We have this number **N**: 1,135,627
- N is the product of 2 prime numbers
- We know that one of these primes is 1039
- How can we find the other prime number?
- 1,135,627 / 1039 = 1093

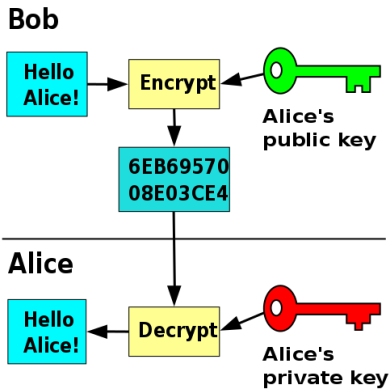
## Encrypting and Signing I

- Which are the steps to encrypt/decrypt a message?
- Which are the steps to sign/verify a message?

## Encrypting/Decripting I

- Bob wants to send a message to Alice:
  - Bob and Alice generate a key pair.
  - Alice sends her public key to Bob
  - Bob **encrypts** the message m using Alice's public key
  - Bob sends the ciphertext
  - Alice receives the ciphertext and **decrypts** the message using its private key

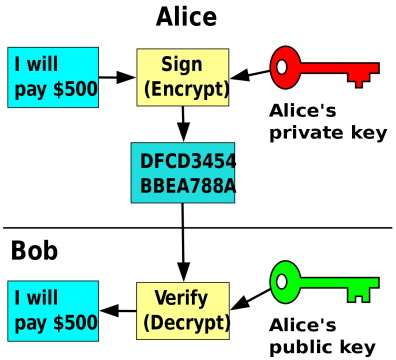
# Encrypting/Decripting II



## Sign/Verify I

- Bob wants to send a proof that its him to Alice:
  - Bob and Alice generate a key pair.
  - Bob **signs** the message using its own private key
  - Bob sends the ciphertext and its public key
  - Alice receives the ciphertext and **verifies** the message using Bob's public key

# Sign/Verify II



## Symmetric vs Asymmetric I

# Symmetric v/s Asymmetric

| Characteristic | Symmetric Key Cryptography | Asymmetric Key Cryptography |
|---|---|---|
| Key used for encryption / decryption | Same key is used for encryption and decryption | One key used for encryption and another, different key is used for decryption |
| Speed of encryption / decryption | Very fast | Slower |
| Size of resulting encrypted text | Usually same as or less than the original clear text size | More than the original clear text size |
| Key agreement / exchange | A big problem | No problem at all |
| Number of keys required as compared to the number of participants in the message exchange | Equals about the square of the number of participants, so scalability is an issue | Same as the number of participants, so scales up quite well |
| Usage | Mainly for encryption and decryption (confidentiality), cannot be used for digital signatures (integrity and non-repudiation checks) | Can be used for encryption and decryption (confidentiality) as well as for digital signatures (integrity and non-repudiation checks) |

## Symmetric vs Asymmetric II

- Symmetric key can be considered a little bit more secure than Asymmetric key
- Thinking about distributed networks:
  - If we have 5 entities to communicate in a secure way we need:
  - Asymmetric key: 2 keys per person (10 keys in total)
  - Symmetric key: (n-1)*n = 20 keys!!
  - Symmetric key doesn't scale well
- It is possible also to mix the 2 techniques, this is named hybrid cryptosystem
- Hybrid cryptosystems tries to improve speed and security at once

## Symmetric vs Asymmetric III

- Which technique should you use for?
  - Store data in a database
  - Sign a document
  - Online Chat
  - Blockchain
  - HTTPS
  - Sending a key

## Asymmetric key examples and use cases

| Algorithm | Trapdoor | Encrypt | Sign | Key exchange |
|---|---|---|---|---|
| RSA | Integer factorization | Yes | Yes | No |
| DSA | Discrete logarithm | No | Yes | No |
| Diffie Hellman | Discrete logarithm | No | No | Yes |
| Elliptic curves | Elliptic curve | Yes | Yes | Yes |

## Basis RSA I

- Asymmetric key encryption algorithm
- RSA comes from Rivest Shamir Adleman
- These 3 guys described the algorithm in 1977
- Can be used to **encrypt/decrypt** or **sign**
- Security relies on **large integer prime factorization**
- Keys are from 2,048 to 4,096 bit

## How RSA works?

4 steps:

1. Key generation
2. Key distribution
3. Encrypting/Signing
4. Decrypting/Verifying

## RSA - Key generation I

- Choose two distinct prime numbers p and q ($p = 3$, $q = 11$)
  - p and q should be kept secret
- Calculate n such that, $n = pq$ ($n = 33$)
  - n is a part of the public key
- Compute $\Phi(n)$, where $\Phi$ is the Euler's totient function, in sum $\Phi(n) = (p-1) * (q-1)$ ($\Phi(n) = 2 * 10 \implies \Phi(n) = 20$

## RSA - Key generation II

- Determine a d that, $de \equiv 1 \mod \Phi(n)$
  - Choose an e such that $1 < e < \Phi(n)$, and such that e and $\Phi(n)$ share no divisors other than 1 ($gcd(\Phi(n), e) = 1$). e and $\Phi(n)$ are coprimes (e = 7)
  - e is kept as the **public key**, so pk = (7, 33)
  - This can be calculated by the **euclidean algorithm**, one solution is d = 3
  - d is kept as the **private key**, so sk = (3, 33)

## RSA - Key distribution

Encryption:

- If bob wants to send a message to Alice
- Alice sends its own e and n, pk = (7, 33)
- Bob receives Alice's public key

Signing:

- If bob wants to sign a message to Alice
- Bob sends to Alice is own e and n, pk = (7, 33)
- Alice receives Alice's public key

**In sum, Bob and Alice exchange its public keys**

## RSA - Encryption/Decryption I

Encryption:

- Bob uses Alice public key to encrypt the message
- The encryption of m = 2 is $c = 2^7 \mod 33 = 128 \mod 33 = 29$
- Bob sends c = 29
- Alice receives c

## RSA - Encryption/Decryption II

Decryption:

- Alice receives c
- Alice uses its private key to decrypt the message
- The decryption process if c = 29, $m = 29^3 \mod 33 = 24389 \mod 33 = 2$
- The message is 2!!!

## RSA - Decryption/Verifying I

Signing:

- Bob uses it own private key to sign the message
- The signing of m = 2 is $s = 2^3 \mod 33 = 8 \mod 33 = 8$
- Bob sends s = 8
- Alice receives s

## RSA - Decryption/Verifying II

Verifying:

- Alice receives s
- Alice uses Bob's public key to verify the signature
- The verifying process if s = 8, $m = 8^7 \mod 33 = 2097152 \mod 33 = 2$
- The signature is 2!!!!

# RSA - In short I

**Key generation:**

| | |
|---|---|
| $Select\ \ p, q$ | $p, q\ \ both\ \ prime$ |
| $calculate\ \ n = p * q$ | |
| $calculate\ \ \phi(n) = (p-1) * (q-1)$ | |
| $select\ \ integer\ \ e$ | $gcd(\phi(n), e) = 1; 1 < e < \phi(n)$ |
| $calculate\ \ d$ | |
| $PublicKey$ | $KU = e, n$ |
| $PrivateKey$ | $KR = d, n$ |

## RSA - In short II

- The video: https://youtu.be/sBO3gH1uGzQ?t=40
- Or you can check the books
- Public params: e, n
- Private params: d, p, q, $\Phi(n)$
- Public key = (e, n)
- Private key = (d, n)
- Factoring n must be hard

Correctness:

- $c^d \mod n = (m^e)^d \mod n = m^{ed} \mod n = m$
- $m^{\Phi(n)} \mod n = 1$, Fermat's little theorem

## RSA - Security I

- Security resides in Integer factorization
- This scheme is deterministic (for an input we have the same output)
  - Remember modes of operation of symmetric crypto
- For this reason RSA is the base to build a secure asymmetric key encryption method
- Plain RSA has some issues that could be exploited (not explained in this course):
  - Multiplicative homomorphism
  - Blind signatures
  - etc...

## RSA - Security II

RSA improvements:

- Probabilistic RSA?
  - Add random padding before encrypting/signing
- Encryption/Decryption schemes:
  - RSAES-OAEP
  - RSAES-PKCS1-v1_5
- Sign/Verify schemes:
  - RSASSA-PSS
  - RSASSA-PKCS1-v1_5

## Basis Diffie-Hellman I

- Asymmetric key exchange algorithm
- RSA comes from its creators Whitfield Diffie and Martin Hellman
- It was described in 1976
- Can be used to **key exchange**
- Security relies on **discrete logarithm**

## How Diffie-hellman works? I

- Alice and Bob publicly agree to use a modulus p = 23 and base g = 5 (which is a primitive root modulo 23)
- Alice chooses a secret integer a = 4 and computes $A = g^a \mod p$ ($A = 5^4 \mod 23 = 4$)
- Alice send A to Bob
- Bob chooses a secret integer b = 3 and computes $B = g^b \mod p$ ($B = 5^3 \mod 23 = 10$)
- Bob send B to Alice

## How Diffie-hellman works? II

- Alice computes $s = Ba \mod p$ ($s = 10^4 \mod 23 = 18$)
- Bob computes $s = Ab \mod p$ ($s = 4^3 \mod 23 = 18$)
- Alice and Bob now share a secret (the number 18)
- Correctness:
  - $A^b \mod p = g^{ab} \mod p = g^{ba} \mod p = B^a mod p$
  - $((g^a \mod p)^b \mod p = (g^b \mod p)^a \mod p)$

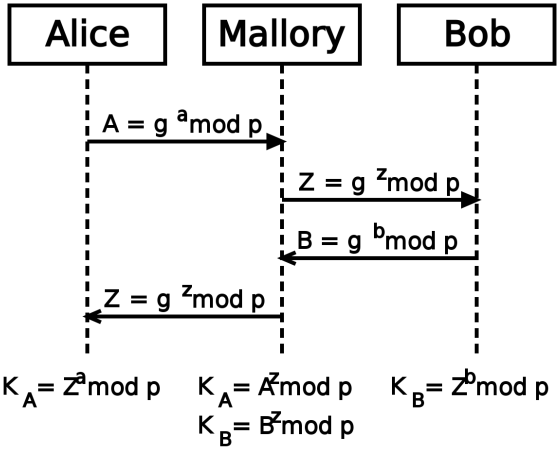
# How Diffie-hellman works? III

| Alice | | Bob | | Eve | |
|---|---|---|---|---|---|
| **Known** | **Unknown** | **Known** | **Unknown** | **Known** | **Unknown** |
| $p = 23$ | | $p = 23$ | | $p = 23$ | |
| $g = 5$ | | $g = 5$ | | $g = 5$ | |
| $\mathbf{a = 6}$ | $\mathbf{b}$ | $\mathbf{b = 15}$ | $\mathbf{a}$ | | $\mathbf{a}, \mathbf{b}$ |
| $A = 5^a \bmod 23$ | | $B = 5^b \bmod 23$ | | | |
| $A = 5^6 \bmod 23 = 8$ | | $B = 5^{15} \bmod 23 = 19$ | | | |
| $\mathbf{B = 19}$ | | $\mathbf{A = 8}$ | | $A = 8, B = 19$ | |
| $s = B^a \bmod 23$ | | $s = A^b \bmod 23$ | | | |
| $s = 19^6 \bmod 23 = 2$ | | $s = 8^{15} \bmod 23 = 2$ | | | $s$ |

## Diffie-Hellman security I

- Security resides in discrete logarithm (considered secure)
- Vulnerable to MITM attacks as we will see
- To avoid MITM attacks we need a 3rd party to certificate the public key
- The validator is a trusted entity called PKI (Public Key Infraestructure)

## Diffie-Hellman security II



## Diffie-Hellman security III

https://www.youtube.com/watch?v=M-0qt6tdHzk

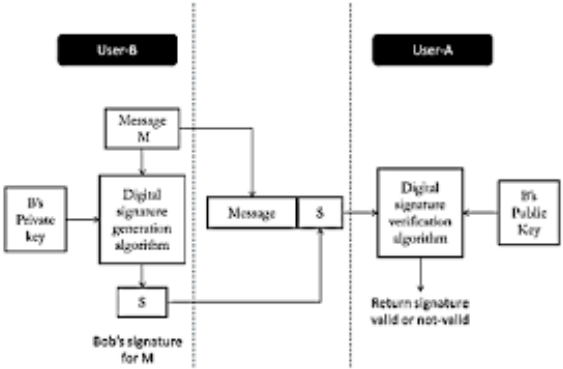
## Other asymmetric key algorithms I

- DSA
- ElGamal
- Hashed ElGamal
- Paillier
- Elliptic curve
  - ECDSA (Elliptic Curve Digital Signature Algorithm)
  - ECDH (Elliptic Curve Diffie Hellman)
  - ECIES (Elliptic Curve Integrated Encryption Scheme)

## DSA I

- Asymmetric key signing algorithm
- DSA comes from Digital Signature Algorithm
- These 3 guys described the algorithm in 1977
- Can be used to **sign/verify**
- Security relies in **discrete logarithm**

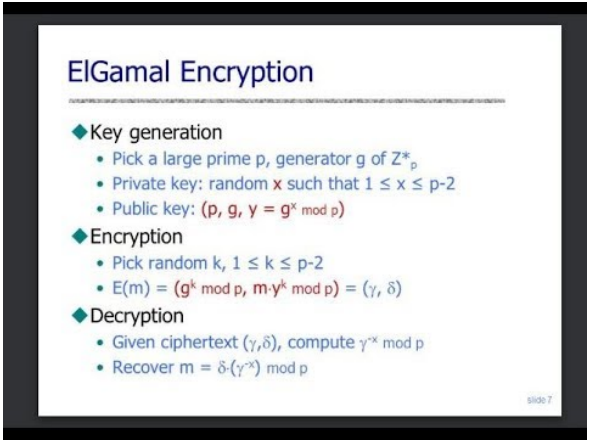
# DSA II


User-B
User-A
Message M
B's Private key
Digital signature generation algorithm
S
Bob's signature for M
Message
S
Digital signature verification algorithm
B's Public Key
Return signature valid or not-valid

## ElGamal I

- Asymmetric key encryption and signing algorithm
- It was described by Taher Elgamal in 1985
- Can be used to **sign/verify** and **encrypt/decrypt**
- Security relies in **discrete logarithm**

# ElGamal II


ElGamal Encryption
Key generation
Pick a large prime p, generator g of $Z^*_p$
Private key: random x such that $1 \le x \le p-2$
Public key: $(p, g, y = g^x \bmod p)$
Encryption
Pick random k, $1 \le k \le p-2$
$E(m) = (g^k \bmod p, m \cdot y^k \bmod p) = (\gamma, \delta)$
Decryption
Given ciphertext $(\gamma,\delta)$, compute $\gamma^{-x} \bmod p$
Recover $m = \delta \cdot (\gamma^{-x}) \bmod p$
slide 7

# ElGamal III

## ElGamal Signature Example

- use field GF(19) $q=19$ and $a=10$
- Alice computes her key:
  - A chooses $x_A=16$ & computes $y_A=10^{16} \bmod 19 = 4$
- Alice signs message with hash $m=14$ as $(3,4)$:
  - choosing random $K=5$ which has $\gcd(18,5)=1$
  - computing $S_1 = 10^5 \bmod 19 = 3$
  - finding $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$
  - computing $S_2 = 11(14-16.3) \bmod 18 = 4$
- any user B can verify the signature by computing
  - $V_1 = 10^{14} \bmod 19 = 16$
  - $V_2 = 4^3.3^4 = 5184 = 16 \bmod 19$
  - since $16 = 16$ signature is valid

## Paillier I

- Asymmetric key encryption algorithm
- It was described by Pascal Paillier in 1999
- Can be used to **encrypt/decrypt**
- Security relies in **Integer prime factorization**
- Interesting applications as online voting and electronic cash (not bitcoin)
- Interesting homomorphic encryption properties (as ElGamal and RSA)

## Paillier II

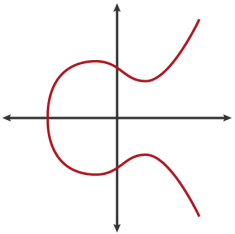
| **Key Generation** |
| --- |
| $k$, the bit length of prime $p$ |
| $n = p^2q$, the modulus |
| $g \in \mathbb{Z}/n\mathbb{Z}$ s.t. $p \mid ord_{p^2}(g)$ |
| $g_p = g \bmod p^2$ |
| Public-key: $(n, g, k)$, Secret key: $p, g_p$ |

| **Encryption of $m$** |
| --- |
| $m \in \{0, 1, ..., 2^{k-2}\}$, a message |
| $r \in \mathbb{Z}/n\mathbb{Z}$, a random integer |
| $c = g^{m+rn} \bmod n$, a ciphertext |

| **Decryption of $c$** |
| --- |
| $m = L(c^{p-1} \bmod p^2)L(g_p^{p-1} \bmod p^2)^{-1} \bmod p$ |

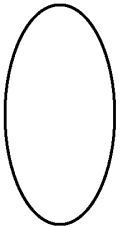
# Elliptic curves I

I DON'T UNDERSTAND WHY PEOPLE GET CONFUSED ... I DON'T LOOK ANY THING LIKE YOU!



ELLIPTIC CURVE

I THINK IT'S THE NAME! LET'S ASK JAVA AND JAVASCRIPT TO SEE HOW THEY DEAL WITH IT

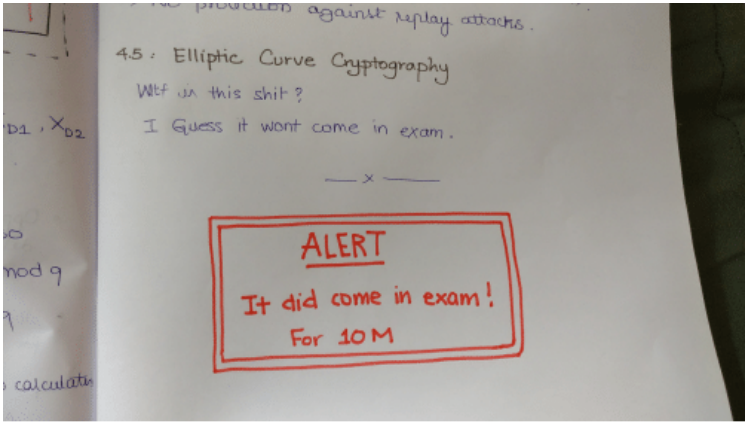


ELLIPSE

## Elliptic curves II

- According to wikipedia elliptic curves are an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields
- ECC allows smaller keys compared to non-EC cryptography with equivalent security
- All points in an eliptic curve must satisfy this equation: $y^2 = x^3 + ax + b$
- The main idea of elliptic curves is to replace the groups $Z_p$ in discrete logarithm problem
- Examples: ECIES, ECDSA, ECDH

# Elliptic curves III



## Attention I

ATTENTION!!

- This is an exam like exercise
- Any question about how the exam will be?

## Statement I

- I want to send a message to someone in this class
- I want that that person answers me
- The message that I want to send is 1GB in size
- The answer is 1kB in size
- Only me and the other person have to know the message and the answer
- How can you do this? Talk me about the technologies, methodologies or algorithms that you will use

## Statement II

- I have a ransomware
- Describe me how you will encrypt the victim's hard drive

# The END!