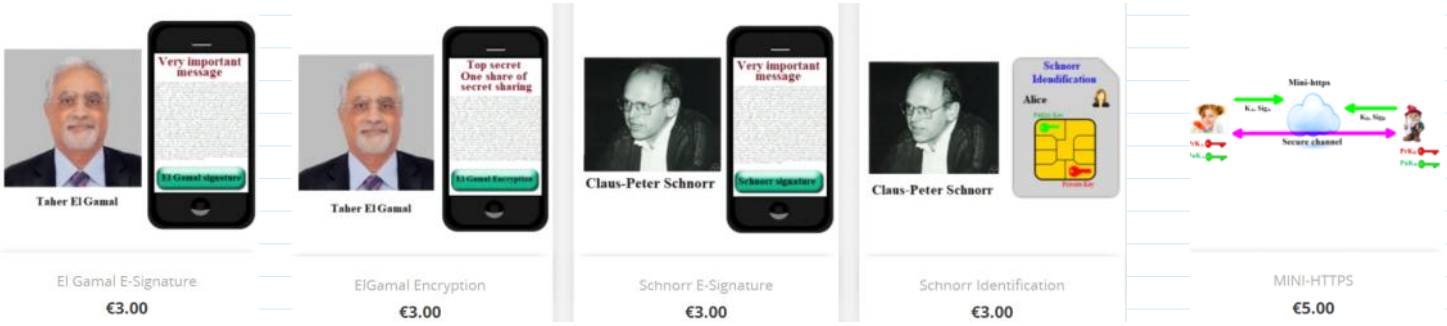
During the exam you must solve the following 4 problems.



Cryptography: Information confidentiality, integrity, authenticity, person identification

δ

Symmetric Cryptography ---------- Asymmetric Cryptography
Secret Key Cryptography — Public Key Cryptography

Symmetric encryption, e.g. AES-128
H-functions, Message digests
HMAC H-Message Authentication Code

Asymmetric encryption, ElGamal encryption
E-signature - Public Key Infrastructure - PKI
E-money, Blockchain
E-voting
Digital Rights Management - DRM (Marlin)
Etc.

Asymmetric Encryption-Decryption ElGamal Encryption-Decryption

Public Parameters generation $PP = (p, g)$.

```
>> 2^28-1
ans = 2.6844e+08
>> int64(2^28-1)
ans = 268435455
```

Asymmetric Signing - Verification

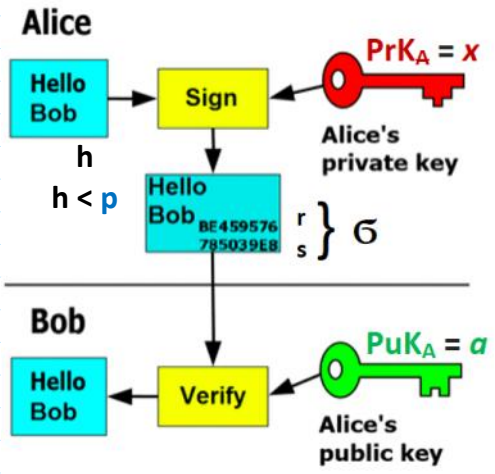
Sign(PrK_A, h) = σ = (r, s)

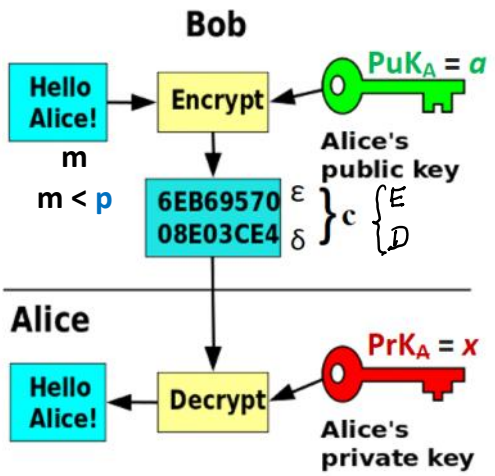
V=Ver(PuK_A, h, σ), V∈{True, False} ≡ {1, 0}

Asymmetric Encryption - Decryption

c = Enc(PuK_A, m) = (E, D)

m = Dec(PrK_A, c)





ElGamal Cryptosystem

1.Public Parameters generation **PP** = (***p***, ***g***).
Generate **strong prime** number ***p***: >> p=genstrongprime(28) % strong prime of 28 bit length
Find a generator **g** in the set Z_p*= {1, 2, 3, …, **p**-1}, where * and : operations are performed mod ***p***.
The search of generator **g** is performed using the following 2 conditions.
If we have strong prime ***p***=2***q***+1, where ***q*** is also prime, then ***g*** is a generator of Z_P* iff
$g^q \neq 1 \bmod p$ and $g^2 \neq 1 \bmod p$.
Declare **Public Parameters** to the network **PP** = (**p**, **g**); ***p***= **268435019**; ***g***=**2;**
2^28-1= 268,435,455

PrK = *x* <-- randi(Z_{p-1}) ==> PuK = *a* = g^x mod p

Asymmetric Encryption-Decryption: El-Gamal Encryption-Decryption

p=268435019; g=2;

Let message ***m*~** needs to be encrypted, then it must be encoded in decimal number ***m***: 1< ***m*** < ***p***.
E.g. ***m*** = 111222. Then ***m*** **mod** ***p*** = ***m***.

$\mathcal{I}_p^* = \{1, 2, 3, \ldots, p-1\}$; * and : operations are performed mod ***p***.

A: $PuK_A = a$ → B: is able to encrypt m to A: m < p

B: $i \leftarrow randi(\mathcal{I}_{p-1})$

Z_{p-1} = {0, 1, 2, 3, …, p-2}; *, :, +; - operations are performed mod (p-1).
Random number *i* is used in the exponent of Discrete Exponent Function (DEF) and therefore all operation in the exponet are performed mod (p-1).
It is an outcome of Little Fermat theorem.

Therefore the operations in the exponents are *, :, +; - are performed mod (p-1).

$E = m \cdot a^i \bmod p$
$D = g^i \bmod p$
$c = (E, D)$ →

A: is able to decrypt $c = (E, D)$ using her $PrK_A = x$.
1. $D^{-x \bmod (p-1)} \bmod p$
2. $E \cdot D^{-x} \bmod p = m$

$(-x) \bmod (p-1) = (0 - x) \bmod (p-1) =$
$= (p - 1 - x) \bmod (p-1)$

$(p-1) \bmod (p-1) = 0$ since

$(-x) \bmod (p-1) = (p-1-x)$

$D^{-x} \bmod (p-1) = D^{p-1-x} \bmod (p-1)$

```
>> D_mx = mod_exp(D, p-1-x, p)
```

```
> x=123
x = 123
>> p=127
p = 127
>> isprime(p)
ans = 1
>> mx=mod(-x,p-1)
mx = 3
>> mod(x+mx,p-1)
ans = 0
```

$p-1 \mid p-1$
$p-1 \quad 1$
0

D^{-x} mod p computation using Fermat theorem:
If p is prime, then for any integer a in Z_p^* holds $a^{p-1} = 1 \bmod p$.

$D^{p-1} = 1 \bmod p \quad / \cdot D^{-x \bmod (p-1)} \bmod p$
$D^{p-1} \cdot D^{-x} = 1 \cdot D^{-x} \bmod p \Rightarrow D^{p-1-x} = D^{-x} \bmod p$

$D^{-x} \bmod p = D^{p-1-x} \bmod p$

Correctness

$Enc(PuK_A = a, i, m) = c = (E, D) = (E = m \cdot a^i \bmod p;\ D = g^i \bmod p)$

$Dec(PrK_A = x, c) = E \cdot D^{-x} \bmod p = m \cdot a^i \cdot (g^i)^{-x} \bmod p =$
$= m \cdot (\underbrace{g^x}_{a})^i \cdot g^{-ix} = m \cdot g^{xi} \cdot g^{-ix} = m \cdot g^{xi - ix} \bmod p = m \cdot g^0 \bmod p =$
$= m \cdot 1 \bmod p = m \bmod p = m = 111222$
since $m < p$

$27 \mid 5$
$25 \quad 5$
2

If $m > p \rightarrow m \bmod p \neq m$; $27 \bmod 5 = 2 \neq 27$.
If $m < p \rightarrow m \bmod p = m$; $19 \bmod 31 = 19$.

Decryption is correct if $m < p$.

Let ***p*** is of order 2^{2048}, i.e. |***p***| = 2048 bits.
Let message ***m*** is encoded by ASCII, i.e. ***m*** is expressed by 8 bits per character.
For example ***m*** = 'Hello Bob'
Then **m** is represented by 9 characters consisting of 8*9 = 72 bits.
And hence ***m*** = 'Hello Bob' can be encrypted using ***p*** with more than 72 bit length.
And if we have ***p*** =~ 2^{2048} then we can encrypt the message consisting of 2048/8 = 256 characters.
As a consequence using standard length of prime ***p*** we can encrypt considerable very short messages.
How to encrypt a large messages having a Kbyte, Mbyte or even Gbyte messages.
The solution is to use symmetric encryption methods.

Next lecture:
Symmetric encryption, e.g. AES-128 - Advanced Encryption Standard having 128 symmetric secret key length.