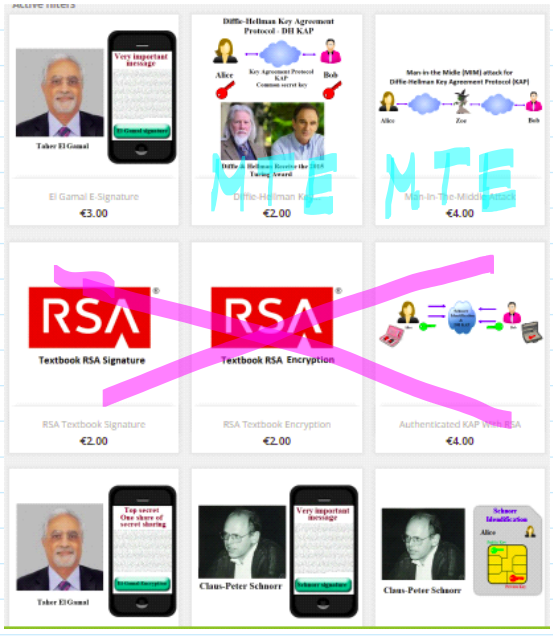
Public Key Cryptosystems - PKCS: Asymmetric Encryption - Decryption --> ElGamal Enc-Dec

Course work: blockchain, cryptocurrencies
smart contracts, ICO, STO, NFT
IoT → 4-th Industrial Revolution

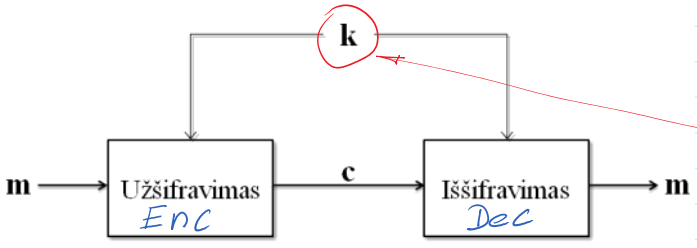
} Oral report
Text ≤ 15 p.
Presentation
~ 12 slides

Till the 16 week.

Midterm exam should be in 8-th of November.
It will be arranged during the execises lecture.
https://imimsociety.net/en/14-cryptography
Problems required to solve: DH-KAP, MiM Attack.
Please buy only one problem at time.



Symmetric encryption



Agreed using Diffie-Hellman Key Agreement Protocol - KAP

Symmetric ciphers: Block Ciphers and Stream Ciphers
↓ BC ↓ SC
for files encryption data stream encryption

Symmetric encryption allows to encrypt finite length of bits (bytes). The restriction to the length depends on the storage and computation power capacities.

The restriction is also made on the number of encryption using the same secret key k, e.g. the number of encrypted bits should not increase 2^{64} bits. If this quantity is over, then the new key k should be agreed.

$AES(k, F) : |k| \in \{128, 192, 256\}$

$|F| = 1\ GB$ $\quad 2^{30} = 1073741824$ $\quad \Downarrow$ SC

If we have a set of files $\{F_1, F_2, \ldots, F_N\}$ // $N = 1\,000\,000$

$AES(k, F_1) = C_1$
$AES(k, F_2) = C_2$
$\ldots$
$AES(k, F_N) = C_N$

This encryption is secure.
It has no Perfect Security property.

>> AES128() ⟶ Upgrade Octave 5.1.0.0 to 6.3.0

http://crypto.fmf.ktu.lt/xdownload/

- octave-6.3.0-w64-installer.exe
- Octave_Stud_2021.10 Updated.7z

C:\Octave\Octave-6.3.0\~Eli.m

addinv	2019.10.04 23:05	M File	1 KB
AES128	2021.10.22 20:33	M File	8 KB
bin2hex	2019.10.27 14:18	M File	1 KB
h24	2021.10.14 12:33	M File	1 KB
h26	2020.03.22 17:08	M File	3 KB
H26d	2020.05.24 19:37	M File	3 KB
h28	2020.05.04 16:58	M File	3 KB
H28d	2020.05.24 17:54	M File	3 KB
hd24	2021.10.14 12:33	M File	1 KB

>> h = h24('Hello')

>> hd = hd24('Hello')

>> hex2dec(h)

```
% AES128()
% in - text/ciphertext maximum 16 symbols or shorter
% key - shared secret key in hexadecimal number of length=32 symbols (128 bits)
% hexadecimal key can be obtained using function:
% >> key=dec2hex(decimal_number,32)
% >> keyd=12345;
% >> key=dec2hex(keyd,32)
% key = 00000000000000000000000000003039
```

```
% >> 3*16^3+3*16+9
% ans = 12345
% >> keyd=1234567
% keyd = 1234567
% >> key=dec2hex(keyd, 32)
% key = 0000000000000000000000000012D687
% >> 1*16^5+2*16^4+13*16^3+6*16^2+8*16+7
% ans = 1234567
%
% Nr - number of rounds (e.g. Nr = 10)
% EnDec - letter determining either encription or decryption
%
% Example:
% >> key = '0000000000000000000000000012D687';
% >> in = 'Laba diena';
% >> Nr = 10;
% >> C = AES128(in,key,Nr,'e')
% 7742110fdf5c568250f2673e7bb28ae0 % ciphertext in hex format
% C = wB??ß\V?P?g>{²?    % ciphertext in ASCII format
% >> D = AES128(C,key,Nr,'d')
% D = Laba diena
%
```

Public Key Cryptosystems - PKCS: Asymmetric Encryption - Decryption

ElGamal Cryptosystem

Public Parameters (PP) generation: 24 bits arithmetics.

1.Strong prime number **p** generation.

2.Find a generator **g** in $\mathbf{Z}_p^*$= {1, 2, 3, …, **p**-1} operations are performed mod **p**.

Prime number ***p*** is Strong prime if $p=2q+1$, where ***q*** is prime.

Then ***g*** is a generator of $\mathbf{Z}_p^*$={1, 2, 3, …, ***p***-1} iff $g^q \neq 1 \bmod p$ and $g^2 \neq 1 \bmod p$.

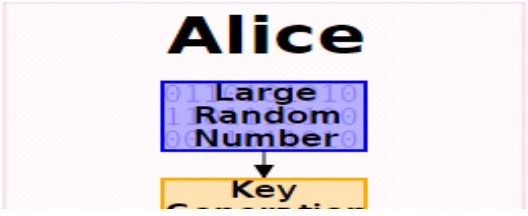
3.Declare **Public Parameters** to the network PP = (**p**, **g**);

```
>> p=genstrongprime(24)
p = 15728303
>> q=(p-1)/2
q =  7864151
>> isprime(p)
ans = 1
>> isprime(q)
ans = 1
>> pb=dec2bin(p)
pb = 1110 1111 1111 1110 1010 1111
>> ph=dec2hex(p)
ph = EFFEAF
```

```
p=int64(15728302)

g =  5
>> mod_exp(g,q,p)
ans = 15728302
>> mod_exp(g,2,p)
ans = 25
```

PP = (**p**=**15728303**, **g**=**5**);

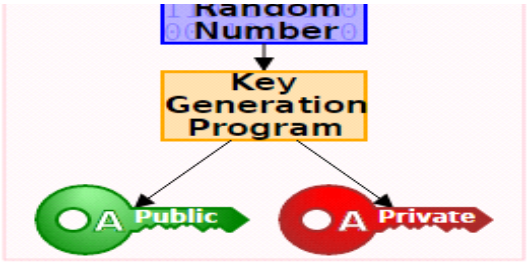


2.***Key generation***

- Randomly choose a private key ***X*** with

  $1 < x < p - 2$.

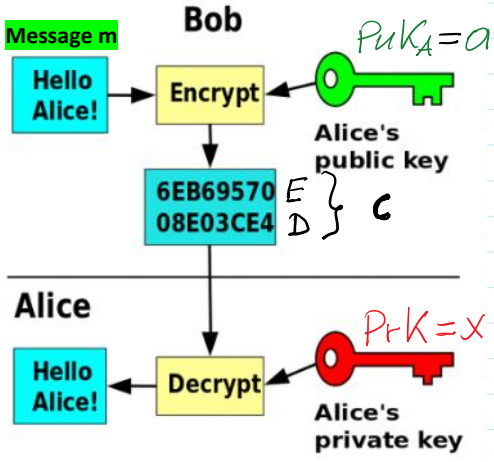
- Compute $a = g^x \bmod p$



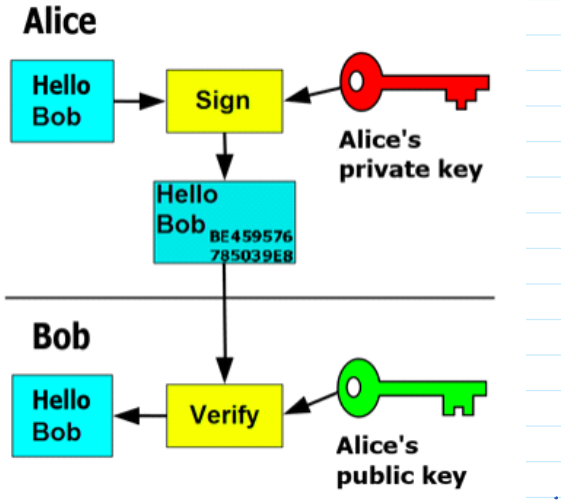
$1 < x < p - 2$.

- Compute $a = g^x \bmod p$.
- The public key is $PuK = a$.
- The private key is $PrK = x$.

Asymmetric Encryption - Decryption
c=Enc(PuK_A, m)
m=Dec(PrK_A, c)



Asymmetric Signing - Verification
S=Sig(PrK_A, m)
V=Ver(PuK_A, S, m), V∈{True, False}≡{1, 0}



El-Gamal Encryption

Let message ***m*** needs to be encrypted, then it must be encoded to the decimal number, less than **p**, to get one-to-one decryption when computations are performed mod **p**.
E.g. ***m*** = 111222, then **0** < ***m*** < **p**=**15728303**.

Let $p = 11$, then the computation mod p of numbers 3, 14, 25, ... yields the same result: $3 \bmod 11 = 14 \bmod 11 = 25 \bmod 11 = 3$.
If we have $0 < m < 11$, then $m \bmod 11 = m$.

PP = (**p**=**15728303**, **g**=**5**);

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

B: $t \leftarrow randi(\mathbb{Z}_p^*)$
$E = m \cdot a^t \bmod p$
$c = (E, D)$ → A: is able to decrypt

$E = m \cdot a^t \bmod p$
$D = g^t \bmod p$
$\Big\} C = (E, D)$ → A : is able to decrypt $C = (E, D)$ using her $PrK_A = X$.

1. $D^{-X} \bmod p = D\_mx$
2. $E \cdot D^{-X} \bmod p = m$

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

Correctness

$$Enc_{PuK_A}(m, t) = C = (E, D) = (E = m \cdot a^t \bmod p,\ D = g^t \bmod p)$$

$$Dec_{PrK_A}(C) = E \cdot D^{-X} \bmod p = m\, a^t \cdot (g^t)^{-X} \bmod p =$$
$$= m \underbrace{(g^X)^t}_{a} \cdot g^{-tX} = m \cdot g^{Xt} \cdot g^{-tX} = m \cdot g^{Xt - tX} \bmod p = m \cdot g^0 \bmod p =$$
$$= m \cdot 1 \bmod p = m \bmod p = m$$
since $m < p$

If $m > p$ → $m \bmod p \neq m$; $27 \bmod 5 = 2 \neq 27$.
If $m < p$ → $m \bmod p = m$; $19 \bmod 31 = 19$.
Decryption is correct if $m < p$.

ElGamal encryption is probabilistic: encryption of the same message m two times yields the different cyphertexts c_1 and c_2.

1-st encryption:
$t_1 \leftarrow randi(\mathcal{Z}_p^*)$
$E_1 = m \cdot a^{t_1} \bmod p$
$D_1 = g^{t_1} \bmod p$
$\Big\} C_1 = (E_1, D_1)$

2-nd encryption
$t_2 \leftarrow randi(\mathcal{Z}_p^*)$
$E_2 = m \cdot a^{t_2} \bmod p$
$D_2 = g^{t_2} \bmod p$
$\Big\} C_2 = (E_2, D_2)$

$r_1 \neq r_2$
$c_1 \neq c_2$

It is needed to provide encryption security requirements!

Security considerations. Total break of cryptosystem (CS) is

to find a $PrK = X$.

The data available to compromise CS are the following

$PP = (p, g)$; $PuK = a$; $\{(m_1, c_1), (m_2, c_2), \ldots (m_n, c_1)\}$

$a = g^X \bmod p$; $\Rightarrow$ to find X it is required to find a discrete logarithm of a:

$\log_g(a) = \log_g(g^X \bmod p) = X \log_g(g) \bmod p = X \cdot 1 \bmod p = X$.

since $X < p$.

For sufficiently large p to find X is infeasible.

$p \sim 2^{2048} \rightarrow |p| = 2048$ bits length.

Security of ElGamal CS relies on the Discrete Logarithm Assumption – DLA : finding x is infeasible when $PP = (p, g)$ and $PuK = a$ are given.

Till this place

Function of grows