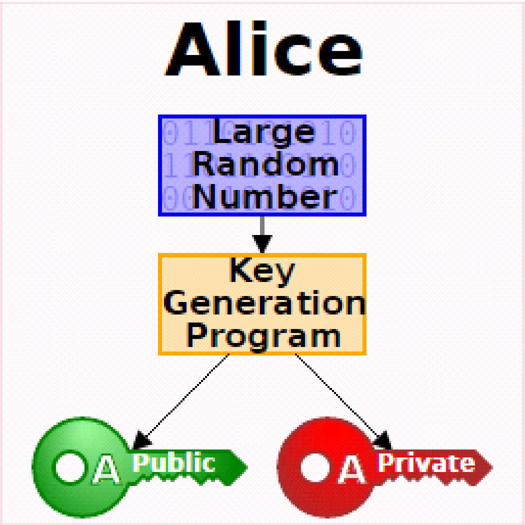
The list of Course Work topics are presented in my Google drive:

Asymmetric - Public Key Cryptography

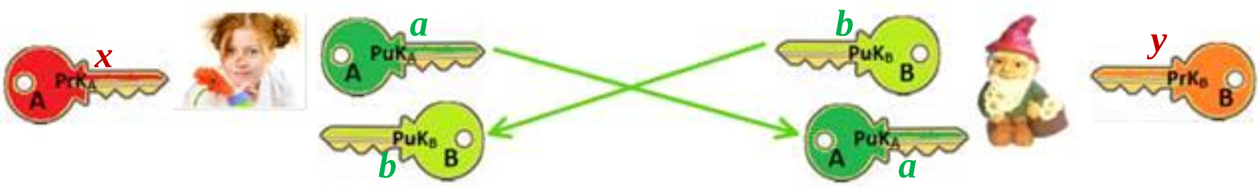


Public Parameters PP = (p, g)
p - strong prime number of 2048 bit length: $p \sim 2^{2048}$;
We will use $p \sim 2^{28}$, i.e. of 28 bit length: $p \sim 2^{28}$.
g - generator in $Z_p^* = \{1, 2, 3, \ldots, p\text{-}1\}$

PrK = x **<-- randi ==> PuK** = $a = g^x$ **mod** p
In general, **PrK** and **PuK** are related by function F:
PuK = F(**PrK**)
F is one-way function - OWF
Having **PuK** it is infeasible to find
PrK = F^{-1}(**PuK**)

$F(x)=a$ is OWF, if:
1. It easy to compute a, when F and x are given.
2. It is infeasible compute x when F and a are given.

Threaths of insecure PrK generation

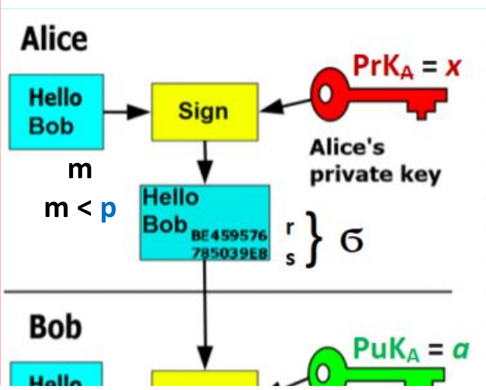


Message m < p

Asymmetric Signing - Verification

Sign(PrK$_A$, m) = σ = (r, s)

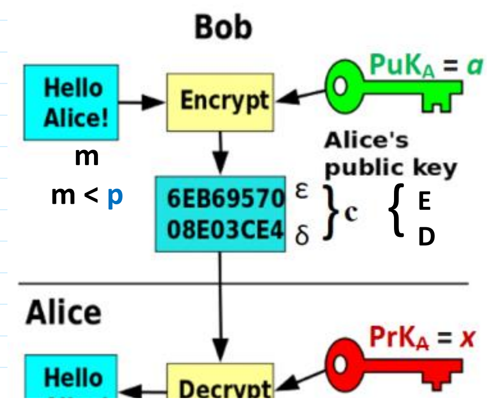
V=Ver(PuK$_A$, m, σ), V$\in$\{True, False\} $\equiv$ \{1, 0\}

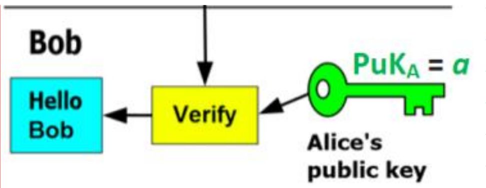


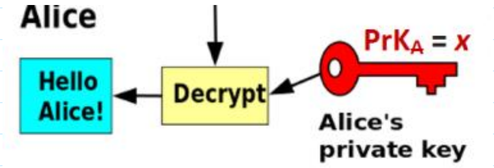
Asymmetric Encryption - Decryption

c=Enc(PuK$_A$, m)

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 Z_p*= {1, 2, 3, ..., **p**-1} using condition.
Strong prime ***p***=2***q***+1, where ***q*** is 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

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

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***.

$A: \xrightarrow{PuK_A = a}$ B: is able to encrypt m to A: $m < p$

B: $i \leftarrow randi(Z_p^*)$

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

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

$p-1 \,/\, (p-1) = 1$, $p-1 - (p-1) = 0$

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

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

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

$D \quad \bmod (p-1) = D \qquad \bmod (p-1)$

$>> D\_mx = mod\_exp(D, p-1-x, p-1)$

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

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

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

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

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 $m < p$

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 $m < p$.

ASCII: 8bits per char.
$\frac{2048}{8} = 256$ char.

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

1-st encryption:

$i_1 \leftarrow randi(\mathbb{Z}_p^*)$

$E_1 = m \cdot a^{i_1} \bmod p$
$D_1 = g^{i_1} \bmod p$
$\Big\} c_1 = (E_1, D_1)$

2-nd encryption

$i_2 \leftarrow randi(\mathbb{Z}_p^*)$

$E_2 = m \cdot a^{i_2} \bmod p$
$D_2 = g^{i_2} \bmod p$
$\Big\} c_2 = (E_2, D_2)$

$i_1 \neq i_2$

$c_1 \neq c_2$

Enigma

Necessity of probabilistic encryption.
Encrypting the same message with textbook RSA always yields the same ciphertext, and so we actually obtain that any deterministic scheme must be insecure for multiple encryptions.

Authenticated Key Agreement Protocol using ElGamal Encryption and Signature.
Hybrid encryption for a large files combining asymmetric and symmetric encryption method.

Hybrid encryption. Let ***M*** be a large finite length file, e.g. of gigabytes length.
Then to encrypt this file using asymmetric encryption is extremely ineffective since we must split it into millions of parts having 2048 bit length and encrypt every part separately.
The solution can be found by using **asymmetric encryption** together with **symmetric encryption**, say AES-128.
It is named as **hybrid encryption method**.
For this purpose the **Key Agreement Protocol** (**KAP**) using **asymmetric encryption** for the same symmetric secret key ***k*** agreement must be realized and encryption of ***M*** realized by **symmetric encryption** method, say AES-128.

AKAP: Asym.Enc & Digital Sign.

How to encrypt large data file M: Hybrid enc-dec method.

1. Parties must agree on common symmetric secret key k.
   for symmetric block cipher, e.g. AES-128, 192, 256 bits.

A: $PrK_A = x$; $PuK_A = a$.
$PuK_B = b$.

1) $k \leftarrow randi(2^{128})$
$i_k \leftarrow randi(2^{128})$

Io $Enc(PuK_B = b, i_k, k) = c = (E, D)$

2) M - large file to be encrypted
$E_k(M) = AES_k(M) = G$

3) Signs ciphertext G
3.1) $h = H(G)$
3.2) $Sign(PrK_A = x, h) = \sigma = (r, s)$

$\xrightarrow{c,\ G}$ σ, PuK_A, $Cert_A$

B: $PrK_B = y$; $PuK_B = b$.
$PuK_A = a$.

1.1. Verify if PuK_A and $Cert_A$ are valid?
1.2. Verify if σ on $h = H(G)$ is valid?
$h' = H(G)$
$Ver(PuK_A, \sigma, h') = True$
2. $Dec(PrK_B, c) = k$
3. $D_k(G) = AES_k(G) = M$.

A was using so called encrypt-and-sign (E-&-S) paradigm.
(E-&-S) paradigm is recomended to prevent so called Choosen Ciphertext Attacks - CCA: it is most strong attack but most complex in realization.

Homomorphic property of ElGamal encryption

Let we have 2 messages m_1, m_2 to be encrypted

$i_1 \leftarrow randi(\mathbb{Z}_p^*)$ $\quad$ $i_2 \leftarrow randi(\mathbb{Z}_p^*)$ $\quad$ m_2 - how many electrocars Bob would like to have.

$E_1 = m_1 \cdot a^{i_1} \bmod p$ $\quad$ $E_2 = m_2 \cdot a^{i_2} \bmod p$

$D_1 = g^{i_1} \bmod p$ $\quad$ $D_2 = g^{i_2} \bmod p$

$c_1 = (E_1, D_1)$ $\quad$ $c_2 = (E_2, D_2)$

Let we intend to encrypt product $m_1 \cdot m_2 \bmod p = m < p$ of corresponding plaintexts m_1 and m_2 using random param $i = (i_1 + i_2) \bmod (p-1)$.

$$Enc(a, (i_1+i_2) \bmod (p-1), m_1 \cdot m_2 \bmod p) = c_{12} = (E_{12}, D_{12})$$

$$E_{12} = m_1 \cdot m_2 \cdot a^{i_1+i_2 \bmod (p-1)} \bmod p = (\underbrace{m_1 \cdot a^{i_1} \bmod p}_{E_1} \cdot \underbrace{m_2 \cdot a^{i_2} \bmod p}_{E_2}) \bmod p$$

$$E_{12} = E_1 \cdot E_2 \bmod p$$

$$D_{12} = g^{i_1+i_2} \bmod p = (\underbrace{g^{i_1} \bmod p}_{D_1} \cdot \underbrace{g^{i_2} \bmod p}_{D_2}) \bmod p$$

$$D_{12} = D_1 \cdot D_2 \bmod p$$

$$Enc(a, (i_1+i_2) \bmod (p-1), m_1 \cdot m_2 \bmod p) = c_{12} =$$
$$= (E_{12}, D_{12}) =$$
$$= (E_1 \cdot E_2 \bmod p, D_1 \cdot D_2 \bmod p) = c_1 \cdot c_2$$

Multiplicative isomorphism

Encryption funktion of production $m_1 \cdot m_2$ of two plaintexts m_1 and m_2 maps to ciphertext $c_1 \cdot c_2 = c$ of two ciphertexts c_1 and c_2, when $c_1 = Enc(a, i_1, m_1)$ and $c_2 = Enc(a, i_2, m_2)$.

$$Enc(m_1 \cdot m_2) = Enc(m_1) \cdot Enc(m_2) = c_1 \cdot c_2$$

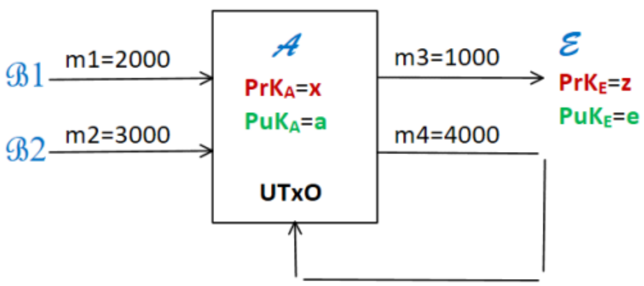
Additively Multiplicative Isomorphism

$Enc(m_1 + m_2) = c = c_1 \cdot c_2$ ← Pascal Paillier encryption.

Application in eVoting and Blockchain systems.

One special case of ElGamal encryption

is instead of m_1, m_2 encryption
to encrypt messages $n_1 = g^{m_1}$, $n_2 = g^{m_2}$; $n_3 = g^{m_3}$, $n_4 = g^{m_4}$;



How to provide anonymity of transaction amounts and to verify the **balance**: **m1+m2 = m3+m4** ?

n1= g^{m1} mod **p** | **n3**= g^{m3} mod **p**

n2= g^{m2} mod **p** | **n4**= g^{m4} mod **p**

If **m1+m2 = m3+m4**,
Then **n1*n2 = n3*n4**.

$c_1 \cdot c_2 = c_3 \cdot c_4$

Till this place

$Enc(a, i_1 + i_2, n_1 \cdot n_2) = Enc(a, i_1, n_1) \cdot Enc(a, i_2, n_2)$

$$E_{12} = E_1 \cdot E_2 \bmod p = n_1 a^{i_1} \bmod p \cdot n_2 a^{i_2} \bmod p =$$
$$= g^{m_1} a^{i_1} \bmod p \cdot g^{m_2} a^{i_2} \bmod p =$$
$$= g^{m_1 + m_2} \cdot a^{i_1 + i_2} \bmod p.$$

Let $m_1 + m_2 = m_3 + m_4$

$$\begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ n_1 & n_2 & n_3 & n_4 \\ \downarrow i_1 & \downarrow i_2 & \downarrow i_3 & \downarrow i_4 \\ c_1 & c_2 & c_3 & c_4 \end{array}$$

If $m_1 + m_2 = m_3 + m_4 \mod (p-1) \Rightarrow c_1 \cdot c_2 = c_3 \cdot c_4$;

Homomorphic encryption: cloud computation with encrypted data.

Paillier encryption scheme is additively-multiplicative homomorphic and has a potentially nice applications in blockchain, public procurement, auctions, gamblings and etc.

$Enc(Puk, m_1+m_2) = c_1 \bullet c_2$.

Blockchain and IPFS application to data storage.

IPFS, web3.