Principles of Public Key Cryptography

Instead of using single symmetric key shared in advance by the parties for realization of symmetric cryptography, asymmetric cryptography uses two ***mathematically*** related keys named as private key and public key we denote by **PrK** and **PuK** respectively.

PrK is a secret key owned ***personally*** by every user of cryptosystem and must be kept secretly. Due to the great importance of **PrK** secrecy for information security we labeled it in red color. **PuK** is a non-secret ***personal*** key and it is known for every user of cryptosystem and therefore we labeled it by green color. The loss of **PrK** causes a dramatic consequences comparable with those as losing password or pin code. This means that cryptographic identity of the user is lost. Then, for example, if user has no copy of **PrK** he get no access to his bank account. Moreover his cryptocurrencies are lost forever. If **PrK** is got into the wrong hands, e.g. into adversary hands, then it reveals a way to impersonate the user. Since user's **PuK** is known for everybody then adversary knows his key pair (**PrK**, **Puk**) and can forge his Digital Signature, decrypt messages, get access to the data available to the user (bank account or cryptocurrency account) and etc.

Let function relating key pair (**PrK**, **Puk**) be ***F***. Then in most cases of our study (if not declared opposite) this relation is expressed in the following way:

$$\mathbf{PuK}=\boldsymbol{F}(\mathbf{PrK}).$$

In open cryptography according to Kerchoff principle function ***F*** must be known to all users of cryptosystem while security is achieved by secrecy of cryptographic keys. To be more precise to compute **PuK** using function ***F*** it must be defined using some parameters named as public parameters we denote by **PP** and color in blue that should be defined at the first step of cryptosystem creation. Since we will start from the cryptosystems based on discrete exponent function then these public parameters are

$$\mathbf{PP} = (\boldsymbol{p}, \boldsymbol{g}).$$

Notice that relation represents very important cause and consequence relation we name as the direct relation: when given **PrK** we compute **PuK**.

Let us imagine that for given ***F*** we can find the inverse relation to compute **PrK** when **PuK** is given. Abstractly this relation can be represented by the inverse function $\boldsymbol{F}^{-1}$. Then

$$\mathbf{PrK}=\boldsymbol{F}^{-1}(\mathbf{PuK}).$$

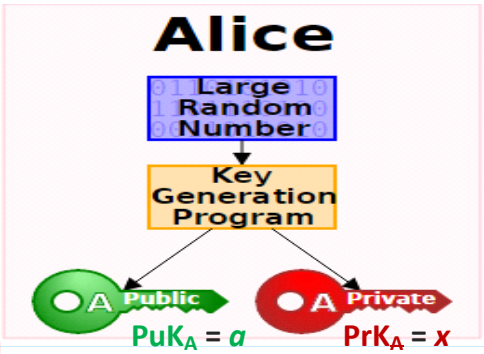
In this case the secrecy of **PrK** is lost with all negative consequences above. To avoid these undesirable consequences function ***F*** must be **one-way function** – OWF. In this case informally OWF is defined in the following way:

1. The computation of its direct value **PuK** when **PrK** and ***F*** in are given is effective.
2. The computation of its inverse value **PrK** when **PuK** and ***F*** are given is infeasible, meaning that to find $\boldsymbol{F}^{-1}$ is infeasible.

The one-wayness of ***F*** allow us to relate person with his/her **PrK** through the **PuK**. If ***F*** is 1-to-1, then the pair (**PrK**, **Puk**) is unique. So **PrK** could be reckoned as a unique secret parameter associated with certain person. This person can declare the possession or **PrK** by sharing his/her **PuK** as his public parameter related with **PrK** and and at the same time not revealing **PrK**.

So, every user in asymmetric cryptography possesses key pair (**PrK**, **PuK**). Therefore, cryptosystems based on asymmetric cryptography are named as **Public Key CryptoSystems** (PKCS).

We will consider the same two traditional (canonical) actors in our study, namely Alice and Bob. Everybody is having the corresponding key pair ($\mathbf{PrK_A}$, $\mathbf{PuK_A}$) and ($\mathbf{PrK_B}$, $\mathbf{PuK_B}$) and are exchanging with their public keys using open communication channel as indicated in figure below.



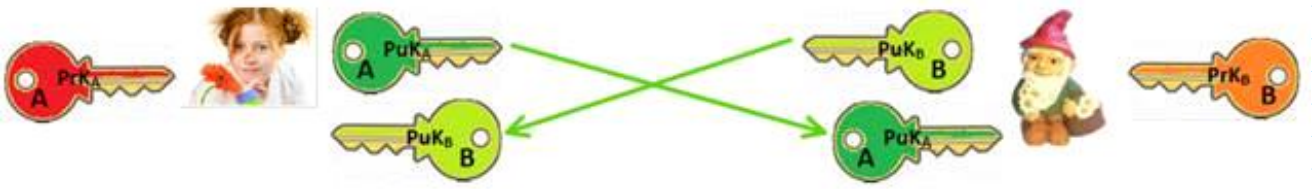
PP = (p, g).

Key generation

- Randomly choose a private key x with $1 < x < p - 1$.
- The private key is **PrK = x = randi(p-1)**
- Compute $a = g^x \bmod p$.
- The public key is **PuK = a = g^x mod p**.

1. Identification.

If person can prove that he/she knows **PrK** corresponding to his/her **PuK** without revealing any information about **PrK** then everybody can trust that he is communicating with person posessing (**PrK**, **Puk**) key pair. This kind of proof is named as ***Zero Knowledge Proof*** (ZKP) and plays a very important role in cryptography. It is very useful to realize identification, Digital Signatures and many other cryptographically secure protocols in internet. In many cryptographic protocols, especially in identification protocols **PrK** is named as **witness** and **PuK** as a **statement** for **PrK**. Every actor is having the corresponding key pair (**PrK$_A$**, **PuK$_A$**) and all **PuK** are exchanged between the users using open communication channel as indicated in figure below.
Let Bob is sure that **PuK$_A$** is of Alice and wants to tell Alice that he intends to send her his photo with chamomile flowers dedicated for Alice. But he wants to be sure that he is communicating only with Alice itself and with nobody else. He hopes that at first Alice will prove him that she knows her secret **PrK$_A$** using ZKP protocol. In general, this protocol is named as identification protocol, it is interactive and has 3 communications to exchange the following data named as ***commitment***, ***challenge*** and ***response***.



Registration phase: Bank generates **PrK$_A$** = **x** and **PuK$_A$** = **a** to Alice and hands over this data in smart card, or other crypto chip in Alice's smart phone, or in software for Smart ID.

Schnorr Id Scenario: **Alice** wants to prove **Bank** that she knows her Private Key - **PrK$_A$** = **x** which corresponds to her Public Key - **PuK$_A$**= a not revealing **PrK$_A$**: Zero Knowledge Proof - ZKP Protocol execution between **Alice** and **Bank** has time limit.
Alice's computation resources has a limit --> protocol must be computationally effective.
PrK$_A$=x is called a **witness** and corresponding **PuK$_A$**=a=g^x **mod** p is called a **statement**.
This protocol is initiated by Alice and has the following three communications.

P(x, a) - **Prover** - **Alice** **V**(a) - **Verifier** - **Bank**

C:\Users\Eligijus\Documents\REKLAMA ZKP_Pasaka

Schnorr Identification: Zero Knowledge Proof - ZKP $PP = (p, g)$.

Schnorr Id Scenario: **Alice** wants to prove **Bank** that she knows her Private Key - $PrK_A = x$ which corresponds to her Public Key - $PuK_A = a = g^x \bmod p$ not revealing $PrK_A = x$.

$\mathcal{A}$: ZKP of knowledge x:

$PrK_A = x$ = randi(p-1)

$PuK_A = a = g^x \bmod p$

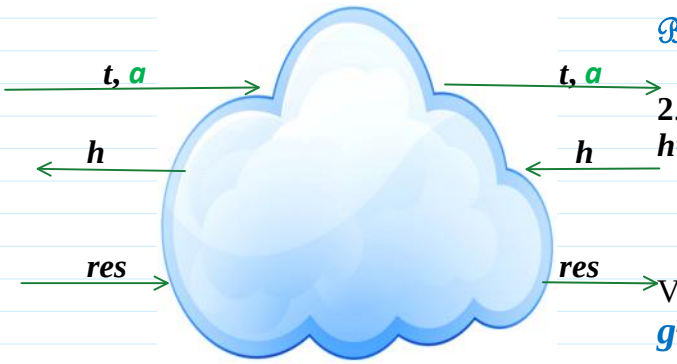
1.Computes **commitment** t for random number i:

i=randi(p-1)

$t=g^i \bmod p$

3.Computes **response** res:

$res=i+xh \bmod (p-1)$



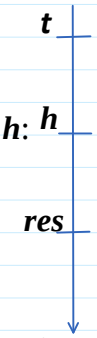
$\mathcal{B}$: $PuK_A = a$

2.Generates **challenge** h:

h=randi(p-1)

Verifies:

$g^{res}=ta^h \bmod p$



Correctness:

$g^{res} \bmod p = g^{i+xh \bmod (p-1)} \bmod p = g^i g^{xh} \bmod p = t(g^x)^h \bmod p = ta^h \bmod p$.

Non-Interactive Zero Knowledge Proof - NIZKP $PP = (p, g)$.

$\mathcal{A}$: NIZKP of knowledge x:

$PrK_A = x$ = randi(p-1)

$PuK_A = a = g^x \bmod p$

1. Computes r for random number i:

i=randi(p-1)

$r=g^i \bmod p$

2. Generates h:

h=randi(p-1)

3. Computes:

$s=i+xh \bmod (p-1)$

$PuK_A = a$

(r, s)

$\mathcal{B}$: $PuK_A = a$

Verifies:

$g^s=ra^h \bmod p$

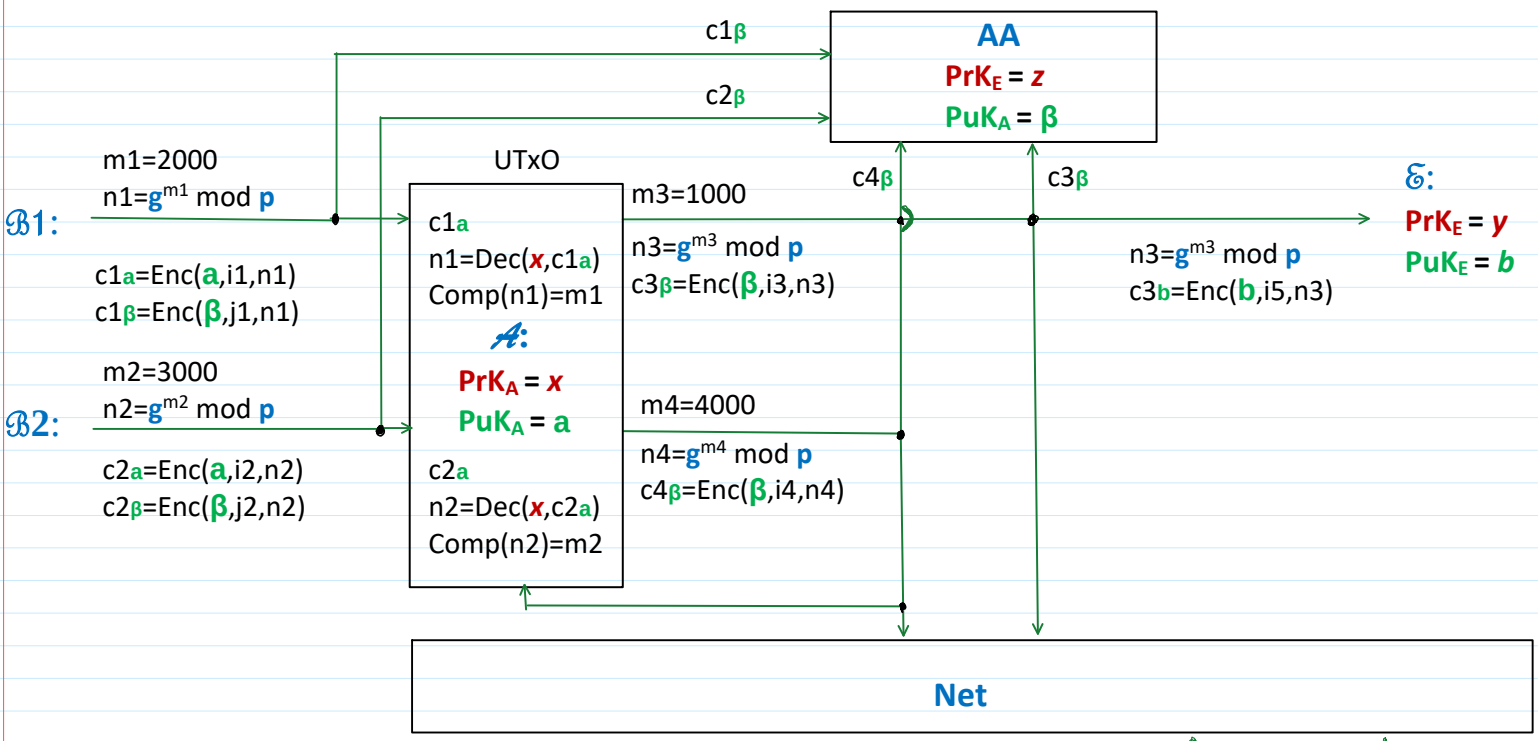
```
>> p= int64(268435019)
>> g=2;

>> x=int64(randi(p-1))
x = 89089011
>> a=mod_exp(g,x,p)
a = 221828624

>> i=int64(randi(p-1))
i = 228451192
>> r=mod_exp(g,i,p)
r = 33418907
h=int64(randi(p-1))
>>  s=mod((i+x*h),p-1)
s = 147250342
```

```
>> g_s=mod_exp(g,s,p)
g_s = 185672370
V1=g_s;
>> a_h=mod_exp(a,h,p)
a_h = 263774143
>>  V2=mod(r*a_h,p)
V2 = 185672370
```

β a b β a b



c_{12a} $c_{34\beta}$

It: $c_{12a} = c_{1a} \cdot c_{2a} = (E_{1a}, D_{1a}) \cdot (E_{2a}, D_{2a}) =$

$= (E_{1a} \cdot E_{2a} \bmod p,\ D_{1a} \cdot D_{2a} \bmod p).$

$c_{34\beta} = c_{3\beta} \cdot c_{4\beta} = (E_{3\beta}, D_{3\beta}) \cdot (E_{4\beta}, D_{4\beta}) =$

$= (E_{3\beta} \circ E_{4\beta} \bmod p,\ D_{3\beta} \cdot D_{4\beta} \bmod p).$

If $m_{12} = m_1 + m_2 \bmod (p-1) = m_{34} = m_3 + m_4 \bmod (p-1)$

$n_{12} = n_1 \cdot n_2 \bmod p \quad \Longrightarrow \quad n_{34} = n_3 \cdot n_4 \bmod p$

$c_{12a} \neq c_{34\beta}$

It: must prove that c_{12a} and $c_{34\beta}$ encrypts the same product

$$n = n_{12} = n_{34}$$

It is named as ciphertexts equivalency proof.

E_{1a} E_{2a} D_{1a} D_{2a}

$c_{12a} = (n_1 a^{i_1} \bmod p \cdot n_2 a^{i_2} \bmod p,\ g^{i_1} \bmod p \cdot g^{i_2} \bmod p)$

$= (n_1 \cdot n_2 a^{i_1+i_2} \bmod p,\ g^{i_1+i_2} \bmod p) =$

$$= (\underbrace{n_{12}\, g^{i_1+i_2} \bmod p}_{E_{12a}},\ \underbrace{g^{i_1+i_2} \bmod p}_{D_{12a}})$$

$$c_{34\beta} = (\overbrace{n_3\, \beta^{i_3} \bmod p}^{E_{3\beta}} \cdot \overbrace{n_4\, \beta^{i_4} \bmod p}^{E_{4\beta}},\ \overbrace{g^{i_3} \bmod p}^{D_{3\beta}} \cdot \overbrace{g^{i_4} \bmod p}^{D_{4\beta}}) = \dots$$

$$= (\underbrace{n_{34}\, \beta^{i_3+i_4} \bmod p}_{E_{34\beta}},\ \underbrace{g^{i_3+i_4} \bmod p}_{D_{34\beta}})$$

However, the scheme presented above is insufficient to realize a proof of ciphertext equivalency. We propose the modification of the existing NIZKP to realize two ciphertext equivalency proofs, namely $C_{\alpha,I}$ in (18), (19), and $C_{\beta,E}$ in (20), (21). Recall that $C_{\alpha,I}$ is a ciphertext of plaintext I encryption with Alice's PuK=α and $C_{\beta,E}$ is a ciphertext of plaintext E encryption with the AA's PuK=β. The statement St of our proposed NIZKP consists of the following:

$$St = \{(\varepsilon_{\alpha,I}, \delta_{\alpha,I}), (\varepsilon_{\beta,E}, \delta_{\beta,E}), \alpha, \beta\}. \quad (22)$$

The random integers $u \leftarrow randi(Z_q)$ and $v \leftarrow randi(Z_q)$ are generated by Alice, and the value $(-v) \bmod q$ is computed. The proof of ciphertext equivalence is computed using three computation steps:

1. The following commitments are computed:
$$t_1 = g^u \bmod p; \quad (23)$$
$$t_2 = g^v \bmod p; \quad (24)$$
$$t_3 = (\delta_{\alpha,I})^u \cdot \beta^{-v} \bmod p. \quad (25)$$
2. The following h-value is computed using the cryptographically secure h-function H:
$$h = H(\alpha \| \beta \| t_1 \| t_2 \| t_3 \|). \quad (26)$$
3. Alice, having her $PrK_A = x$ randomly generates the secret number l for E encryption and computes the following two values:
$$r = x \cdot h + u \bmod q; \quad (27)$$
$$s = l \cdot h + v \bmod q. \quad (28)$$
Then Alice declares the following set of data to the Net:
$$\{\alpha, \beta, t_1, t_2, t_3, r, s\} \rightarrow \text{Net}. \quad (29)$$
To verify the transaction's validity, the Net computes the h-value according to (26) and then verifies three identities:
$$g^r = \alpha^h \cdot t_1; \quad (30)$$
$$g^s = (\delta_{\beta,E})^h \cdot t_2; \quad (31)$$
$$(\varepsilon_{\beta,E})^h \cdot (\varepsilon_{\alpha,I})^{-h} \cdot (\delta_{\alpha,I})^r \cdot \beta^{-s} = t_3. \quad (32)$$