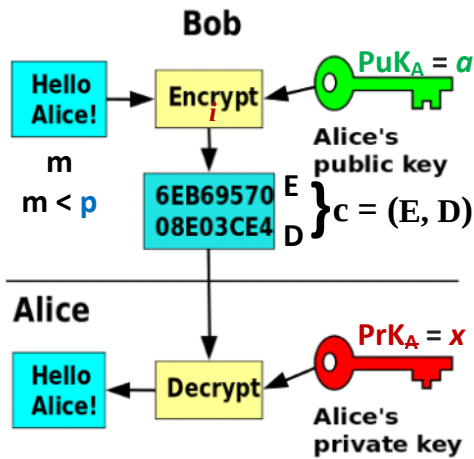


El-Gamal Encryption-Decryption

Additively-multiplicative homomorphic El-Gamal encryption

Public parameters: $PP = (p, g)$; $p=268435019$; $g=2$;



$$B: i \leftarrow \text{randi}(p-1)$$

$$E = m \cdot a^i \bmod p$$

$$D = g^i \bmod p$$

$$c = (E, D) \rightarrow A$$

A: is able to decrypt
 $c = (E, D)$ using her $PrK_A = x$.

1. $D^{-x} \bmod (p-1) \bmod p$
2. $E \cdot D^{-x} \bmod p = m$

Confidentiality

Actors:

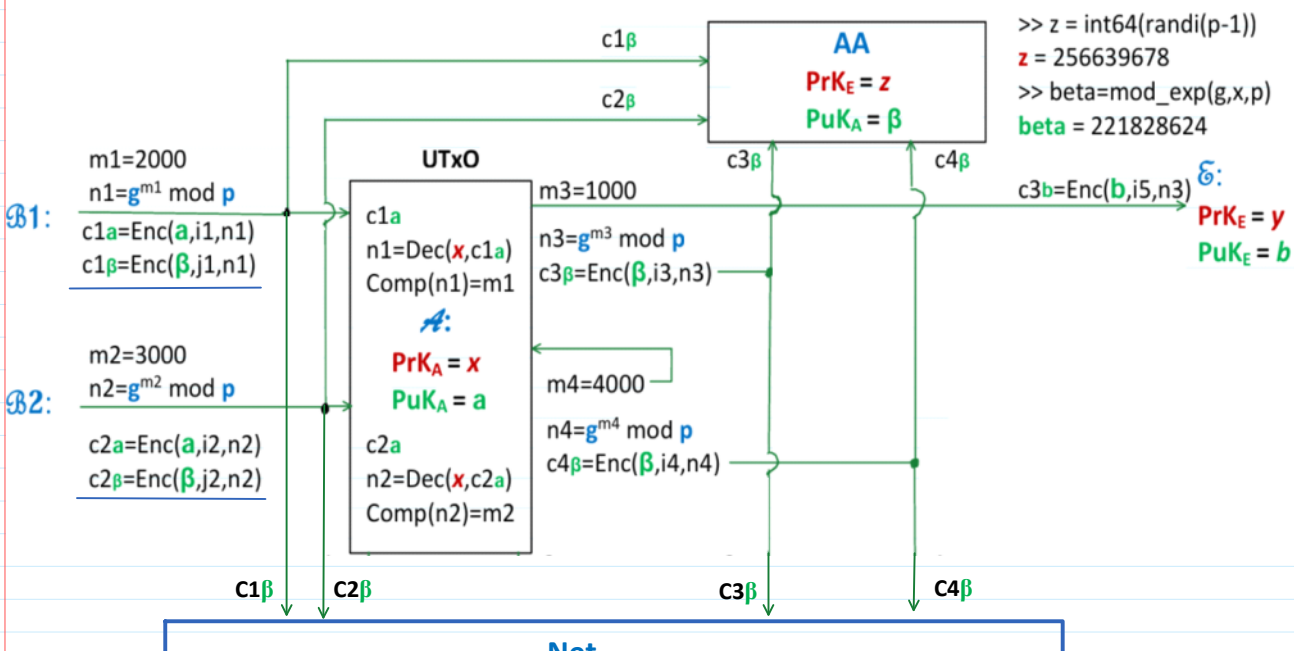
B1 - Bob 1; B2 - Bob2;

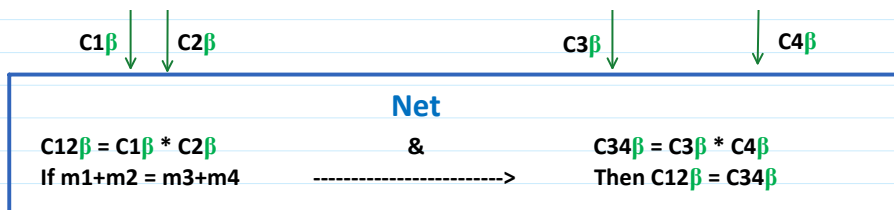
A - Alice;

AA - Audit Authority;

E - Emma.

Confidential Verifiable Transactions





Alice declares her encrypted incomes and expenses to **AA**.

Encryption is performed using **AA** public key β .

Alice incomes are encrypted by **C1β, C2β** and expenses by **C3β, C4β**.

Encryption is performed with the following random parameters satisfying relation:

$$j1 + j2 \text{ mod}(p-1) = i3 + i4 \text{ mod}(p-1)$$

Then

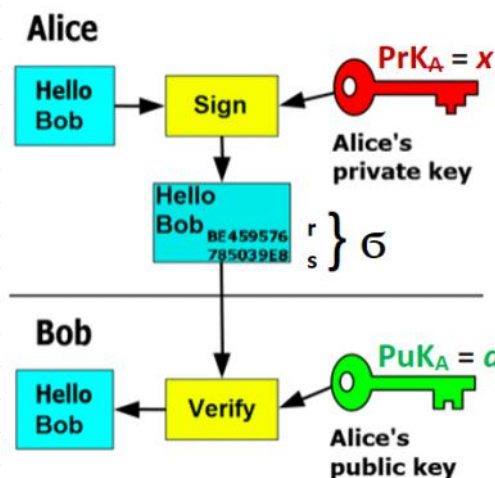
$$C12\beta = C34\beta$$

Asymmetric Signing - Verification

$$\text{Sign}(\text{PrK}_A, m) = \sigma = (r, s)$$

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

Proxy signature



In proxy signature, signature is formed with proxy signers private key.

Verification is performed with original signer public key and public key generated by original signer and presented to proxy signer.

Proxy signature allows a designated person, called a proxy signer, to sign on behalf of an original signer.

Classification of the proxy signatures is shown from the point of view of the degree of delegation, and conditions of a proposed proxy signature for partial delegation are clarified. The proposed proxy signature scheme is based on the discrete logarithm problem.

Compared to the consecutive execution of the ordinary digital signature schemes, it has a direct form.

Verifier needs a public key of a original signer and public key generated by original signer and presented to proxy signer in the verification stage.

Moreover, it requires less amount of computational work than the consecutive execution of the signature schemes.

Due to this efficiency together with the delegation property, an organization, e.g. a software company, can very efficiently create many signatures of its own by delegating its signing operations to multiple employees-

subsidiaries.

Another attractive feature of the proposed schemes is their high applicability to other ordinary signature schemes based on the discrete logarithm problem.

For instance, designated confirmer proxy signatures can be constructed.

Furthermore, using a proposed on-line proxy updating protocol, the original signer can revoke proxies of dishonest proxy signers.

Suppose a software company digitally signs all of its programs under its private key $\text{PrK}=\mathbf{x}$ in order to certify the correctness of their content.

Attached digital signatures offer a functionality of identifying the creator of the programs and, more importantly, of detecting any kind of alteration to the programs.

The most conceivable threat is the infection with computer viruses.

The president of the company knows that the fraction of programs infected with viruses before put on the market is not negligible. **Zero Day Attack**.

She does not want to give $\text{PrK}=\mathbf{x}$ to programmers.

Her first idea is to ask all programmers to submit their programs.

After checking the content of the programs, she signs them under $\text{PrK}=\mathbf{x}$ by herself. But this idea is not good enough since she cannot deal with enormous amount of programs.

Thus a new method should be sought where the president gives each programmer a private key value which is distinct from $\text{PrK}=\mathbf{x}$, but a signature created from which shows an agreement of the company.

With the use of such a signature, the company can simultaneously produce a lot of products accompanied by its signatures, which are generated by multiple employees, and the signing operation is performed in an efficient way.

2 Classification and conditions

It is assumed that a signer Alice asks a proxy signer Bob to carry out signing instead of her, and a verifier Veronica checks the validity of created signatures.

We consider two types of delegation: **full delegation**, and **partial delegation**.

Full delegation

In the full delegation, a proxy signer is given the same private key $\text{PrK}=\mathbf{x}$ that an original signer has, so that she can create the same signature she creates.

Obviously, when the proxy signer deliberately signs a document unfavorable for the original signer, her mischievous action is not detected because the signature created by the proxy signer is **indistinguishable** from the signatures created by the original signer.

Partial delegation

In the partial delegation, a new private key $\mathbf{y}$ is created from $\text{PrK}=\mathbf{x}$, which follows the modification of a verification equation, and $\mathbf{y}$ is given to a proxy signer in a **secure way**. The created signature is checked by the modified equation, but not by the original equation. That implies a signature created by the proxy signer is distinguishable from a signature created by the original signer, and the original signer, who has found a signed document with the content unfavorable for him, can distinguish his ordinary signature from a proxy signature for partial delegation.

More notably, a proxy signature for each proxy is distinct since it is created with different private keys $\mathbf{y}$.

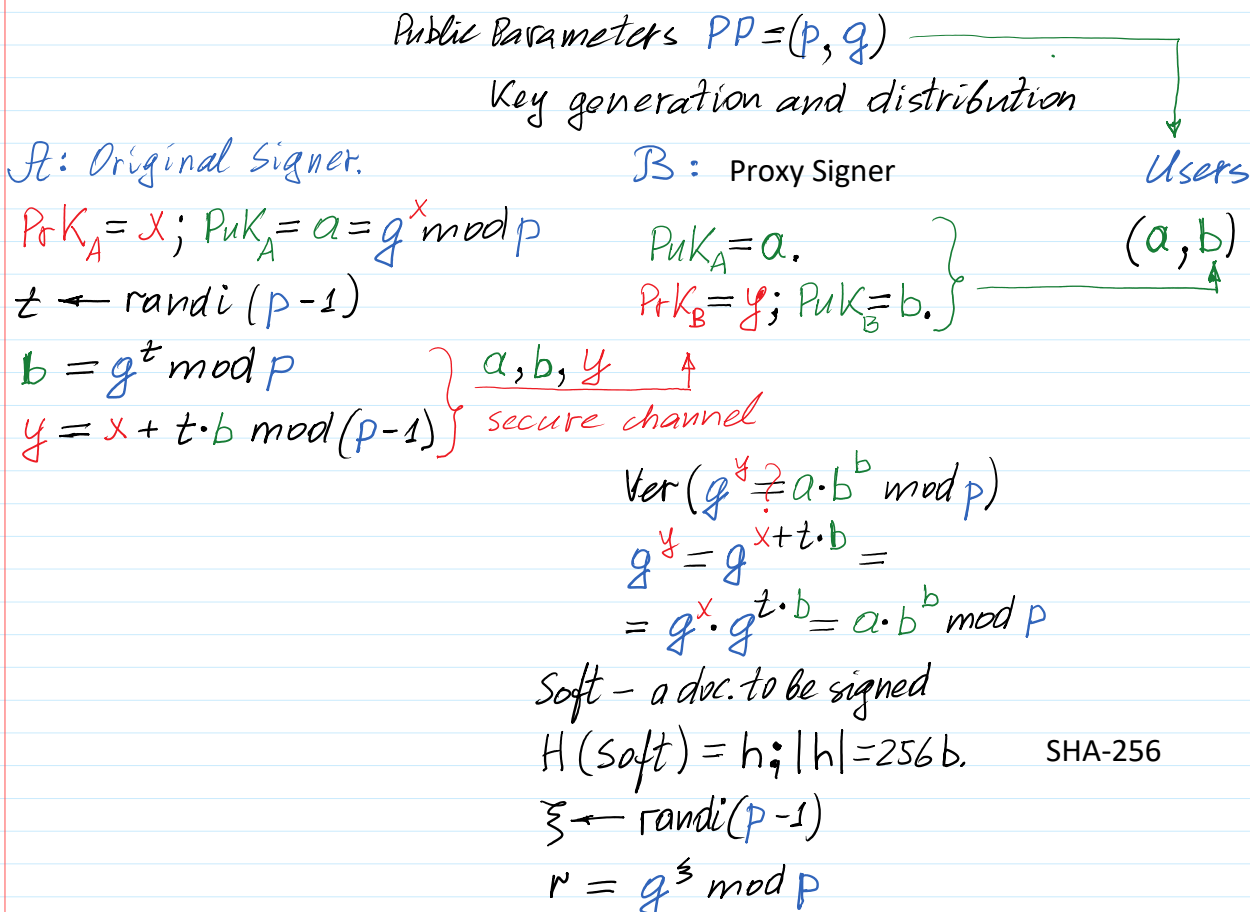
This property corresponds to a fact that seals with different rings or marks leave different images on a sheet.

Properties of proxy signatures with Partial delegation:

- (1) **Unforgeability.** Besides an original signer only a designated signer, called a proxy signer, can create a valid proxy signature for the original signer.
- (2) **Proxy signer's deviation.** A proxy signer cannot create a valid proxy signature not detected as his signature.
- (3) **Private-keys' dependence.** A new private key is computed from a private key of an original signer.
- (4) **Verifiability.** From proxy signatures a verifier can be convinced of the original signer's agreement on the signed message.
- (5) **Identifiability.** An original signer can determine from a proxy signature the **identity** of the corresponding proxy signer.
- (6) **Undeniability.** Once a proxy signer creates a valid proxy signature for an original signer, it is not disavowed (neišsīžadētas) even by the proxy signer. The (6) condition simply means the proxy signer cannot take back what she claimed.

Mambo, Masahiro, Keisuke Usuda, and Eiji Okamoto. "Proxy signatures: Delegation of the power to sign messages." *IEICE transactions on fundamentals of electronics, communications and computer sciences* 79.9 (1996): 1338-1354.

From <https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Masahiro+Mambo%2C+Keisuke+Usuda&btnG=#d=gs_cit&u=%2Fscholar%3Fq%3Dinfo%3AD_WAwVLFjzWJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26hl%3Den>



$$s = \xi + \gamma \cdot h \bmod (p-1)$$

$$\tilde{\sigma} = (r, s) \quad a, b, \tilde{\sigma}, \text{Soft}$$

Verification identity:

$$g^s = r \cdot (a \cdot b^b)^h \bmod p$$

$$1. \text{Ver}(a, b) \neq T$$

$$2. H(\text{Soft}) = h$$

$$3. \text{Ver}(\tilde{\sigma}, h, a, b) \neq T$$

$$\begin{aligned} g^s &= g^{\xi + \gamma \cdot h} = g^{\xi} \cdot g^{\gamma \cdot h} = r \cdot (g^{\gamma})^h = r \cdot (g^{x+t \cdot b})^h = \\ &= r \cdot (g^{x \cdot h + t \cdot b \cdot h}) = r \cdot (g^x)^h \cdot (g^t)^{b \cdot h} = r \cdot a^h \cdot (b^b)^h = \\ &= r \cdot (a \cdot b^b)^h \bmod p. \end{aligned}$$

```
p = int64(268435019);
>> g=2;
> x=int64(randi(p-1))
x = 128831375
>> a=mod_exp(g,x,p)
a = 99834208
```

```
>> t=int64(randi(p-1))
t = 54296150
>> b=mod_exp(g,t,p)
b = 267224695
>> y=mod(x+t*b,p-1)
y = 94866523
```

```
>> g_y=mod_exp(g,y,p)
g_y = 81395743
>> V1=g_y
V1 = 81395743
>> b_b=mod_exp(b,b,p)
b_b = 17947996
>> V2=mod(a*b_b,p)
V2 = 81395743
```

```
>> ksy=int64(randi(p-1))
ksy = 5716357
>> r=mod_exp(g,ksy,p)
r = 118257748
>> s=mod(ksy+y*h,p-1)
s = 521536
```

```
>> g_s=mod_exp(g,s,p)
g_s = 115985652
>> V1=g_s
V1 = 115985652
>> b_b=mod_exp(b,b,p)
b_b = 17947996
>> ab_b=mod(a*b_b,p)
ab_b = 81395743
>> ab_b_h=mod_exp(ab_b,h,p)
ab_b_h = 112511772
>> rab_b_h=mod(r*ab_b_h,p)
rab_b_h = 115985652
>> v2=rab_b_h
v2 = 115985652
```