Homomorphic property of ElGamal encryption

Let we have 2 messages m_1, m_2 to be encrypted

$i_1 \leftarrow randi(\mathbb{Z}_p^*)$ $\quad$ $i_2 \leftarrow randi(\mathbb{Z}_p^*)$

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

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

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

m_2 - how many electrocars Bob would like to have.

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 Isamorphism

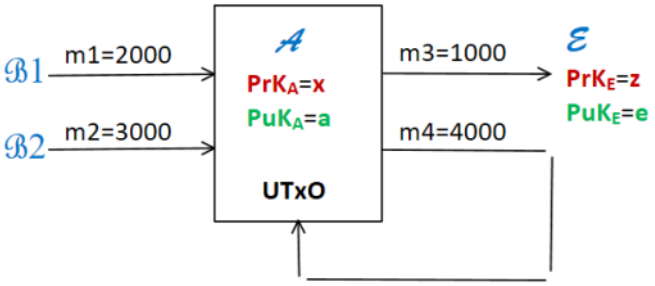
$Enc(m_1 + m_2) = c = c_1 \cdot c_2$ ⇐ Pascal Paillier encryption.

Application in eVoting and Blockchain systems.

One special case of ElGamal encryption

is instead of m_1, m_2 encryption

to encrypt messages $n_1 = g^{m_1}$, $n_2 = g^{m_2}$; $n_3 = g^{m_3}$, $n_4 = g^{m_4}$; mod p.



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**.

$c_1 \cdot c_2 = c_3 \cdot c_4$

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

$E_{12} = E_1 \cdot E_2 \ mod\ p = n_1 a^{i_1} mod\ p \cdot n_2 a^{i_2} mod\ p =$

$= g^{m_1} a^{i_1} mod\ p \cdot g^{m_2} a^{i_2} mod\ p =$

$= g^{m_1 + m_2} \cdot a^{i_1 + i_2} \ mod\ p.$

Let $m_1 + m_2 = m_3 + m_4$

n_1 n_2 n_3 n_4

i_1 i_2 i_3 i_4

c_1 c_2 c_3 c_4

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

A: c_1, c_2 → $Dec(x, c_1) = n_1 = g^{m_1} \ mod\ p$

$Dec(x, c_2) = n_2 = g^{m_2} \ mod\ p$

Sums $m_1, m_2 < 2^{30}$.

Sums $m_1, m_2 < 2^{30}$.

Why

```
>> p
p = 268435019
>> g
g = 2
>> x
x = 89089011
>> a
a = 221828624
>> m1=111222;
>> m2=2;
```

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

$E_1 = m_1 \cdot a^{i_1} \bmod p$

$D_1 = g^{i_1} \bmod p$

$c_1 = (E_1, D_1)$

```
>> n1=mod_exp(g,m1,p)
n1 = 54586077
>> i1=int64(randi(p-1))
i1 = 100411198
>> a_i1=mod_exp(a,i1,p)
a_i1 = 15179112
>> E1=mod(n1*a_i1,p)
E1 = 3807046
>> D1=mod_exp(g,i1,p)
D1 = 146933715
```

```
>> n2=mod_exp(g,m2,p)
n2 = 4
>> i2=int64(randi(p-1))
i2 = 123249696
>> a_i2=mod_exp(a,i2,p)
a_i2 = 27559877
>> E2=mod(n2*a_i2,p)
E2 = 110239508
>> D2=mod_exp(g,i2,p)
D2 = 172508970
```

```
>> E12=mod(E1*E2,p)
E12 = 37666
```

$Dec(X, c_{12}) = n_{12} = n_1 * n_2 \bmod p$

$54586077 * 4 \bmod p$

$n_{12} = g^{m_1 + m_2 \bmod (p-1)} \bmod p$

ProxySignature

The 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.

$a = g^x \bmod p$

Compared to the consecutive execution of the ordinary digital signature schemes, it has a direct form, and a verifier does not need a public key of a user other than the original 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.
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 secret s 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.
She does not want to give **PrK=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 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 secret value which is distinct from ***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 ***x*** instead of her, and a verifier **Veronica** checks the validity of created signatures.
There are different types of delegation, full delegation, partial delegation and delegation by warrant.

Full delegation

In the full delegation, a proxy signer is given the same secret that an original signer has, so that he can create the same signature she creates.
Obviously, when the proxy signer deliberately signs a document unfavorable for the original signer, his mischievous action is not detected because the signature created by the proxy signer is indistinguishable from the signatures creatted by the original signer.

Partial delegation

In the partial delegation, a new secret ***u*** is created from ***x***, which follows the modification of a verification equation, and ***u*** 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 from original signature.
This property corresponds to a fact that seals with different rings or marks leave different images on a sheet.
In this delegation, only the public key of the original signer is required for the verification.
As far as the author's knowledge, this type of delegation has not appeared in the literature.

Delegation by warrant

The last delegation is implemented by using a warrant, which certifies that Bob is exactly the signer to be entrusted.
Delegation by warrant is performed by the consecutive execution of signing of the public key signature scheme, and this type of delegation has appeared in the literature, e.g. [VAB91, Neu93].
There are two types of signature schemes for this approach.

1. In the first approach, a warrant is composed of a message part and an original signer's signature for a public key of Bob.
   Or the warrant is just composed of only a message declaring Bobis designated as a proxy signer.
   Given the warrant, Bob signs a document under her secret by an ordinary signature scheme, and a valid proxy signature consists of a created signature **S**, together with the warrant.
   It should be remarked that **S**, is not related to the original signer's public key ***a*** in this case.
2. In the second approach, a warrant is composed of a message part and an original signer's signature for a newly generated public key.
   The secret key compatible with this generated public key is given to Bob in a secure way.
   Given the warrant, Bob signs a document under the given secret by an appropriate signature scheme.
   A created message and its verification is similar to that in the first approach except that only the original signer's public key is required.

The former and the latter approaches correspond to two classes of proxies in [Neu93], called a delegate proxy and a bearer proxy, respectively.

Conditions of proxy signatures(partial delegation

(i) (**Unforgeability**) Besides an original signer Alice only a designated signer Bob, called a proxy signer,
can create a valid proxy signature for the original signer.
(ii) (**Proxy signer's deviation**) A proxy signer Bob cannot create a valid proxy signature not detected as his signature.
(iii) (**Secret-keys' dependence**) A new secret ***u*** is computed from a secret ***x*** of an original signer.
(iv) (**Verifiability**) From proxy signatures ***S*** verifier can be convinced of the original singer's agreement on the signed message either by a selfauthenticating form or by an interactive form.

(v) (***Distinguishability***) *Valid proxy signatures are distinguishable from valid self-signing signatures in polynomial time or size computation*.
Here, the self-signing signature means an ordinary signature created by the original signer.
(vi) (**Identifiability**) An original signer can determine from a proxy signature the identity of the corresponding proxy signer.
(vii) (**Undeniability**) Once a proxy signer creates a valid proxy signature for an original signer, it is not disavowed even by the proxy signer.

Proxy signer's unforgeability of other proxy signers' signatures is included in the second condition.
The seventh condition simply means the proxy signer cannot take back what she claimed, and it does not demand existence of a disavow protocol shown in (CA89, Cha90].

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.

Public Parameters $PP=(p,g)$

Key generation and distribution → Users

A: Original Signer.

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

$t \leftarrow randi(p-1)$

$b = g^t \bmod p$

$y = x + t \cdot b \bmod (p-1)$

B:

$PuK_A = a$.

$PrK_B = y$; $PuK_B = b$.

Users: (a, b)

a, b, y — secure channel

$Ver(g^y \stackrel{?}{=} a \cdot b^b \bmod p)$

$g^y = g^{x+t\cdot b} =$

$= g^x \cdot g^{t\cdot b} = a \cdot b^b \bmod p$

Soft – a doc. to be signed

$H(Soft) = h$; $|h| = 256$ b.

$\xi \leftarrow randi(p-1)$

$r = g^\xi \bmod p$

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

$\sigma = (r, s)$

a, b, σ, Soft

$Cert_A$, $Cert_B$

Emilia

Verification identity:

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

1. $Ver(a,b) \stackrel{?}{=} T$
2. $H(soft) = h$
3. $Ver(\sigma, h, a, b) \stackrel{?}{=} T$

$g^s = g^{\xi + y\cdot h} = g^\xi \cdot g^{y\cdot h} = r \cdot (g^y)^h = r \cdot (g^{x+t\cdot b})^h =$

$= r \cdot (g^{x\cdot h + t\cdot b\cdot h}) = r \cdot (g^x)^h \cdot (g^t)^{b\cdot h} = r \cdot a^h \cdot (b^b)^h =$

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