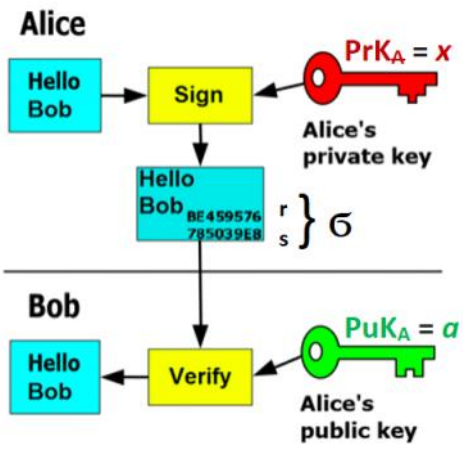
Subliminal Channel - Steganography
using ElGamal Signature

>> p = 268 435 019; % 2^28-1 --> >> int64(2^28-1) % ans = **268 435 455**
>> g=2; % >> dec2bin(2^28-1)
% ans = 1111 1111 1111 1111 1111 1111 1111
>> p=268435019;
>> g=2;

ElGamal Signature: **Sign(PrK,h)=σ=(r,s)**



1.***Signature generation on*** **h**

- Generate at random **k** <-- randi(2^28-1)
- Compute **k^{-1} mod (p-1): k^{-1} mod (p — 1)** exists if **gcd(k, p — 1) = 1**, then **k** and **p-1** are relatively prime.
  k^{-1} can be found using either Extended Euclidean algorithm or Euler theorem

>> k_m1=mulinv(k,p-1)	% k^{-1}mod (p-1) computation.

- Compute **$r=g^k$ mod p**
- Compute **$s=(h-x*r)*k^{-1}$ mod (p-1)** --> **$h=x*r+s*k$ mod (p-1),**
  Signature **σ=(r,s)**
  For subliminal channel creation s^{-1} mod (p-1) must exists, such that **$s*s^{-1}= 1$ mod (p-1).**

Subliminal Channel - Pasąmonės Kanalas: Bob was put in jail.
Paslėptas Kanalas

A: B:

>> z=randi(p-1)
x = 100497451 → **x** = 100497451
>> a=mod_exp(g,z,p)
a = 91968695 → **a** = 91968695

Let **M** is a message to be signed which can be arbitrary chosen.
To sign **M** the **h**-value of **M** must be computed: **h=H(M)**.
The secret message to be sent is a number **n=k** satisfying the following conditions:
1. Number **k<p**.
2. gcd(**k**, **p**-1)=1.

1.***Signature generation on*** **h** : k ← randi(p-1) & k=n.

- Compute **k^{-1} mod (p-1): k^{-1} mod (p — 1)** exists if **gcd(k, p — 1) = 1**, **k** and **p-1** are relatively prime

k^{-1} can be found using either Extended Euclidean algorithmtor Euler theorem

```
>> k_m1=mulinv(k,p-1)   % k^-1 mod (p-1) computation.
```

- Compute **r=g^k mod p**
- Compute **s=(h-z*r)*k^{-1} mod (p-1)** --> **h=z*r+s*k mod (p-1),**

Signature σ=**(r,s)**

For subliminal channel creation s^{-1} mod (p-1) must exists, such that **s*s^{-1}= 1 mod (p-1).**

$gcd(s, p-1) = 1$

A: $M, \sigma = (r, s)$ → B: $h = H(M)$

2.**Signature on h verification**

A signature (**r,s**) on message **h** is verified as follows.

1. 1<**r**<**p-1** and 1<**s**<**p-1.**
2. **V1=$a^r r^s$ mod p, V2=g^h mod p** and **V1=V2.**

The verifier accepts a signature if all conditions are satisfied and rejects it otherwise.

2.**Secret message n=k recovery from the equation**

s=(h-z*r)*k^{-1} mod (p-1) /* $k \mod (p-1)$

s must satisfy the condition that s^{-1} mod (p-1) exists, such that **s*s^{-1}= 1 mod (p-1).**

$k * s = h - x * r \mod (p-1) \mid \cdot s^{-1}$ % $gcd(s, p-1) = 1$

$k = (h - x * r) * s^{-1} \mod (p-1)$; $k = n$.

Subliminal Channel - Steganography
Using Schnorr Signature

```
>> x = int64(randi(p))     % gcd(x,p-1)=1  -->  Exsist x^-1 mod (p-1)
>> a = mod_exp(g,x,p)
```

```
>> x=int64(randi(p))
x = 69555360
>> gcd(x,p-1)
ans = 2
>> x=int64(randi(p))
x = 164448909
>> gcd(x,p-1)
ans = 1
```

M - masking message to be sign.

h = k - secret message to be sent.

$r = g^k \mod p$

$h = H(M \| r)$

$s = h + x * k \mod (p-1)$

$M, \sigma = (r, s)$ → B: $h = H(M \| r)$

$V1 = a^s \mod p$

$h = H(M \| r)$

$s = h + x * k \mod (p-1)$

$\rightarrow$ B: $h = H(M \| r)$

$V1 = g^s \mod p$

$V2 = g^h \cdot a^k \mod p$

$V1 \stackrel{?}{=} V2$

$s - h = x * k \ / * x^{-1} \mod (p-1)$

$k = (s-h) * x^{-1} \mod (p-1) = n$

Corrected version

M - masking message to be sign.

$n = k$ - secret message to be sent.

$r = g^k \mod p$

$h = H(M \| r)$

$s = k + x \cdot h \mod (p-1)$

$M, \sigma = (r, s) \rightarrow$ B: $h = H(M \| r)$

$V1 = g^s \mod p$

$V2 = g^{k+xh} = g^k \cdot (g^x)^h = r \cdot a^h \mod p$

$V1 \stackrel{?}{=} V2$

k=n recovery

k = s - x*h mod (**p**-1)

Bit Commitment using H-function

B: Should I must sell my bitcoins?

A: Don't hurry, I know the price for next month.

B: Then tell me please.

A: I'll tell you next month, but if you want to know immediatly give me 3 BTC.

B: How it is to know me that you are not cheating?

A: We can use Bit Commitment scheme.

Bit Commitment using HMAC

BTC-NM

$h = H(\text{BTC-NM} \| \text{rand})$

$\xrightarrow{h}$ B:

After 1 month

1 BTC = 27 000 $

Send me your guess

BTC-NM || rand →

h' = H(BTC-NM || rand)

h ≟ h'

A: k = lsb256(z)
symmetric key

B: k = lsb256(z)

HMAC(k, BTC-NM || rand) = h —h→

after 1 month

Send me your guess

Enc(k, BTC-NM || rand) = c —c→

Dec(k, c) = BTC-NM || rand

h' = HMAC(k, BTC-NM || rand)

If h = h' ⇒ A guessed BTC-NM.

Bit Commitment using RSA modification

Public Parameter PP = (p) p - may be strong prime

A: $K_A = (e_A, d_A)$ B: $K_B = (e_B, d_B)$ % private keys

$e_A \cdot d_A = 1 \bmod (p-1)$ $e_B \cdot d_B = 1 \bmod (p-1)$

$e_A \leftarrow randi : gcd(e_A, p-1) = 1 \Rightarrow \exists! \; d_A = e_A^{-1} \bmod (p-1)$

```
>> dA = mulinv(eA, p-1)
>> mod(eA * dA, p-1) = 1
```

M - message : Bitcoin price next month

A: Encrypts M with encryption function Enc( ) and sends ciphertext c_1 to B. $M < p$.

$Enc(e_A, M) = M^{e_A} \bmod p = c_1$

$Enc(e_A, M) = M^{e_A} \bmod p = c_1$

c_1 →

← c_2

$B: Enc(e_B, c_1) = c_1^{e_B} \bmod p = c_2$

After 1 month

$A: c_3 = c_2^{d_A} \bmod p$

c_3 →

$B: c_4 = c_3^{d_B} = \dots$

$= (c_2^{d_A})^{d_B} = ((c_1^{e_B})^{d_A})^{d_B} =$

$= (((M^{e_A})^{e_B})^{d_A})^{d_B} \bmod p =$

$= M^{(e_A d_A)(e_B d_B) \bmod (p-1)} \bmod p =$

$e_A d_A \bmod (p-1) = 1 \;\&\; e_B d_B \bmod (p-1) = 1$

$= M^{1 \cdot 1} \bmod p \overset{M<p}{=\!=} M$

Proxy signature

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.
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 highapplicability 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 s 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 Shigeo asks a proxy signer Peggy to carry out signing instead of him, 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 s that an original signer has, so that she can create the same signature he 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 creatt!d by the original signer.
(**Partial delegation**)
In the partial delegation, a new secret u is created from s, 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.
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 Peggy 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 Peggy.
   Or the warrant is just composed of only a message declaring Peggy is designated as a proxy signer.
   Given the warrant, Peggy 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 v 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 Peggy in a secure way.
   Given the warrant, Peggy 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.

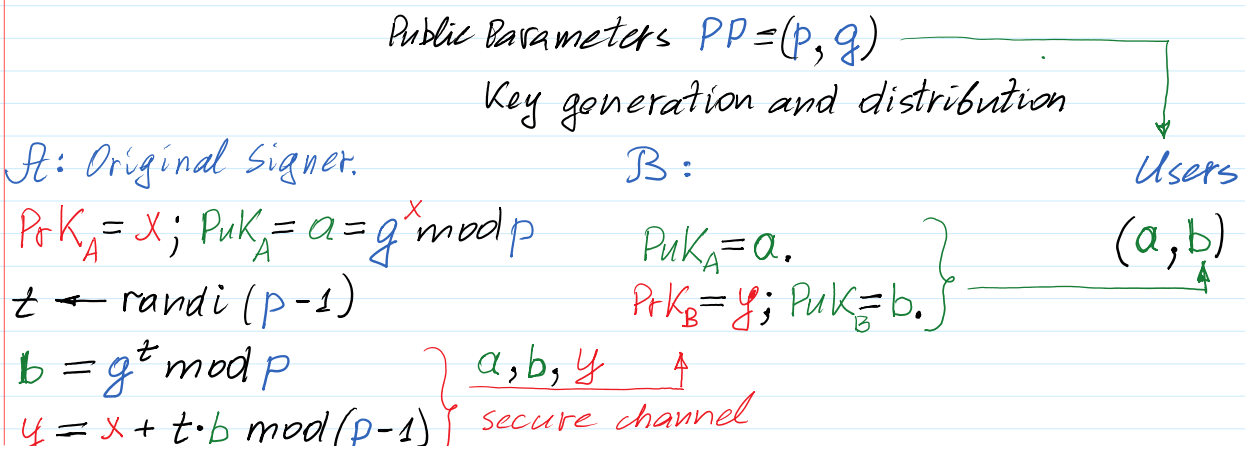
The former and the latter approaches correspond to two classes of proxies in [Neu93], called a delegate proxy and a bearer proxy, respectively.

Conditions of proxy signatures(partial delegation

(i) (**Unforgeability**) Besides an original signer only a designated signer, called a proxy signer, can create a valid proxy signature for the original signer.
(ii) (**Proxy signer's deviation**) A proxy signer cannot create a valid proxy signature not detected as his signature.
(iii) (**Secret-keys' dependence**) A new secret a is computed from a secret of an original signer.
(iv) (**Verifiability**) From proxy signatures a 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.
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%3Dinf o%3AD_WAwVLFjzwJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D0%26hl%3Den>



$b = g^x \bmod p$

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

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

Verification identity:

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

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

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

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

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