

Subliminal Channel - Steganography using ElGamal Signature

Bob was put in jail.

Subliminal Channel - Pasąmonės Kanalas Paslėptas Kanalas

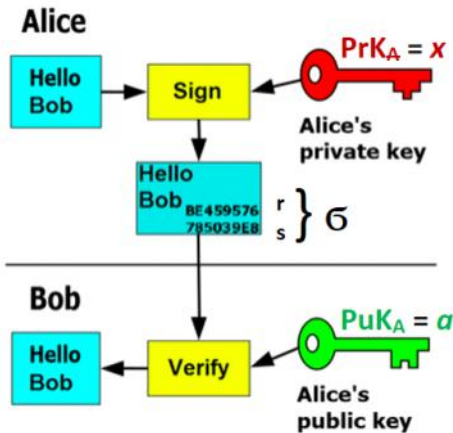
Grafity pictures

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

ElGamal Signature: $\text{Sign}(\text{PrK}, h) = \sigma = (r, s)$

A :

B :



```
>> z=int64(randi(p-1))
z = 100497451
>> c=mod_exp(g,z,p)
c = 91968695
```

z = 100497451
c = 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 k satisfying the following conditions:

1. Number $k < p$.
2. $\gcd(k, p-1)=1$.

1. Signature generation on h : Alice generates secret number $k < p$, $\gcd(k, p-1) = 1$, k and $p-1$ are relatively prime. The secret message Alice intends to send to Bob is encoded by k .

- Compute $k^{-1} \bmod (p-1)$: $k^{-1} \bmod (p-1)$ exists if $\gcd(k, p-1) = 1$, k and $p-1$ are relatively prime. k^{-1} can be found using either Extended Euclidean algorithm or Euler theorem

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

- Compute $r = g^k \bmod p$
- Compute $s = (h - z * r) * k^{-1} \bmod (p-1) \rightarrow k = (h - z * r) * s^{-1} \bmod (p-1)$.

Signature $\sigma = (r, s)$

For subliminal channel creation $s^{-1} \bmod (p-1)$ must exists, such that $s * s^{-1} = 1 \bmod (p-1)$: $\gcd(s, p-1)=1$.

>> s_m1=mulinv(s,p-1) % $s^{-1} \bmod (p-1)$ computation.

```
>> p = int64(268435019)    >> M='Bob I wait you'    >> s=int64(126312337)    >> s_m1=int64(212931827)
p = 268435019              M = Bob I wait you          s = 126312337          s_m1 = 212931827
>> g=2;                    >> h=hd28(M)              >> gcd(s,p-1)           >> mod(s*s_m1,p-1)
>> z=int64(randi(p-1))     h = 240745341          ans = 1               ans = 1
z = 7688965
>> c=mod_exp(g,z,p)
c = 217126641
```

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

$$A: \underbrace{M, \sigma = (r, s)}_C \rightarrow B: h = H(M)$$

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. $V1 = g^h \bmod p$, $V2 = c^r r^s \bmod p$, and $V1 = V2$.

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

2. Secret message k recovery from the equation

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

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

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

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

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

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

>> s_m1=mulinv(s,p-1)

>> k=mod(h-z*r,p-1)

```
k=int64(206415161)
```

```
gcd(k,p-1)
```

```
ans=1
```

```
r=mod_exp(g,k,p)
```

```
r = 113543563
```

Subliminal Channel - Steganography Using Schnorr Signature

M - masking message to be signed.

k - secret message to be sent.

$$N = g^k \bmod p$$

$$h = H(M || N)$$

$$S = k + z * h \bmod (p-1)$$

$\gcd(z, p-1) = 1$, then $z^{-1} \bmod (p-1)$ exists.

$$\left\{ \begin{array}{l} M, \sigma = (r, s) \end{array} \right\} \rightarrow B: h = H(M || r)$$

$$\left\{ \begin{array}{l} V1 = g^s \bmod p \\ V2 = r * c^h \bmod p \end{array} \right.$$

$$S = K + Z * h \bmod (P-1)$$

$$\begin{cases} V1 = g^s \bmod P \\ V2 = r^s * c^h \bmod P \end{cases} \rightarrow V1 \stackrel{?}{=} V2$$

$$K = (S - Z * h) \bmod (P-1)$$

```
>> p = int64(268435019)
p = 268435019
>> g=2;
>> z=int64(randi(p-1))
z = 7688965
>> c=mod_exp(g,z,p)
c = 217126641
```

```
>> M='Bob I wait you impatiently'
M = Bob I wait you
>> h=hd28(M)
h = 240745341
```

```
>> k=int64(randi(p-1))
k = 21398217
>> r=mod_exp(g,k,p)
r = 92029919
>> s=mod(k+z*h,p-1)
s = 34241676
```

```
>> V1=mod_exp(g,s,p)
V1 = 63032245
>> c_h=mod_exp(c,h,p)
c_h = 62405921
>> V2=mod(r*c_h,p)
V2 = 63032245
```

```
>> kk=mod(s-z*h,p-1)
kk = 21398217
```

Bit Commitment using H-function

B: Should I must sell my bitcoins?

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

B: Then tell me please.

A: I'll tell you next month, but if you want to know immediately 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

BTC-NM (Next Month)

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

$h \rightarrow$ B:

After 1 month

1BTC = 27 000 \$

Send me your guess

$\leftarrow$ BTC-NM $\parallel$ rand

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

$$h \stackrel{?}{=} h'$$

Bit Commitment using Hashed Message Authenticated Code - HMAC

A: $k = \text{lsb } 256(z)$
symmetric key

B: $k = \text{lsb } 256(z)$

$\text{HMAC}(k, \text{BTC-NM} \parallel \text{rand}) = h$ $\xrightarrow{h}$ Send me your guess

$\text{Enc}(k, \text{BTC-NM} \parallel \text{rand}) = c$ $\xrightarrow{c}$ Dec(k, c) = BTC-NM $\parallel$ rand
Send me Hashed data $\xleftarrow{\text{after 1 month}}$

$h' = \text{HMAC}(k, \text{BTC-NM} \parallel \text{rand})$

If $h = h' \Rightarrow$ A guessed
BTC-NM.

Bit Commitment using RSA cryptosystem approach modification

rsa key generation: 2048 bits arithmetics; we will use 28 bits arithm.

- Two primes p, q are generated at random. $|q| = 1024$ bits $|p| = 1024$ bits
- RSA module $n = p \cdot q$ is computed & $\phi(n) = (p-1) \cdot (q-1) = \phi$.
- Random RSA exponent e : $\gcd(e, \phi) = 1$ is computed.
According to RSA standard $e = 2^{16} + 1$.
- The inverse element to $e \bmod \phi$ is computed:
 $d = e^{-1} \bmod \phi \Rightarrow d e \bmod \phi = 1$.
- $\text{PrK} = d$; $\text{PuK} = (n, e)$.

```
>> e=2^16+1
e = 65537
>> isprime(e)
ans = 1
```

We use $|n| = 28$ bits $|p| = |q| = 14$ bits

RSA textbook encryption

m - message: $m < n \sim 2^{2048}$; $|m| < 2048$ bits

$m \bmod n = m$

B: $\xleftarrow{\text{PuK}_A}$

$c = \text{Enc}(\text{PuK}_A, m) = m^e \bmod n$ $\xrightarrow{c}$ A: $\text{PuK}_A = (n, e)$; $\text{PrK}_A = d$.

```
>> m=int64(111222333)
m = 111222333
>> c=mod_exp(m,e,n)
c = 51722206
```

$\text{Dec}(\text{PrK}_A, c) = m =$

$= c^d \bmod n =$
 $= (m^e)^d \bmod n = m^{ed} \bmod n =$
 $= m^{ed \bmod \phi} \bmod n = m^1 \bmod n = m \bmod n = m$

```
>> mm=mod_exp(c,d,n)
mm = 111222333
```

RSA textbook encryption is not randomised - is not probabilistic.

Key Generation

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

$\mathcal{A}: K_A = (e_A, d_A)$

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

$\mathcal{B}: K_B = (e_B, d_B)$ % private keys

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

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

$$d_A = e_A^{-1} \bmod p$$

$$d_B = e_B^{-1} \bmod p$$

```
>> p=int64(268435019)
```

```
p = 268435019
```

```
>> eA=int64(randi(p-1))
```

```
eA = 113108923
```

```
>> gcd(eA,p-1)
```

```
ans = 1
```

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

```
dA = 114348687
```

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

```
ans = 1
```

```
>> eB=int64(randi(p-1))
```

```
eB = 175845008
```

```
>> gcd(eB,p-1)
```

```
ans = 2
```

```
>> eB=int64(randi(p-1))
```

```
eB = 213299919
```

```
>> gcd(eB,p-1)
```

```
ans = 1
```

```
>> dB=mulinv(eB,p-1)
```

```
dB = 182812595
```

```
>> mod(eB*dB,p-1)
```

```
ans = 1
```

M - message : Bitcoin price next month

$\mathcal{A}$: Encrypts M with encryption function Enc and sends ciphertext c_1 to $\mathcal{B}$. $M < p$.

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

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

$$\xleftarrow{c_2}$$

After 1 month

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

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

$$\begin{aligned}
 &= (c_2^{d_A})^{d_B} = ((c_1^{e_B})^{d_A})^{d_B} = \\
 &= \left((M^{e_A})^{e_B} \right)^{d_A} \mod p = \\
 &= M^{(e_A d_A)(e_B d_B) \mod (p-1)} \mod p =
 \end{aligned}$$

$$e_A d_A \mod (p-1) = 1 \quad \& \quad e_B d_B \mod (p-1) = 1$$

$$= M^{1 \cdot 1} \mod p \stackrel{M < p}{=} M$$