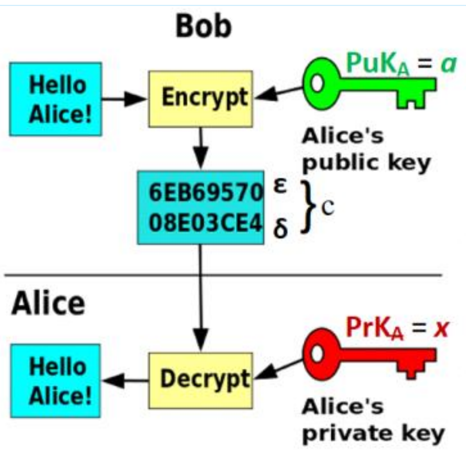
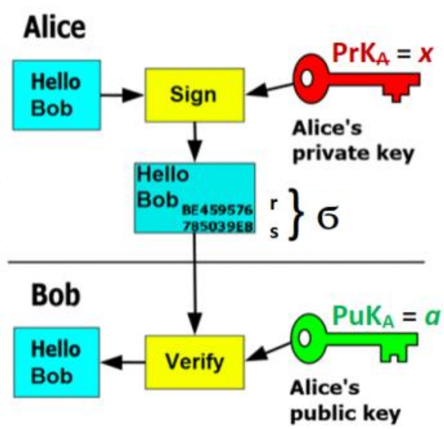
Course for repetition:
http://crypto.fmf.ktu.lt/telekonf/archyvas/B111%20Kriptologija/B111%202024-R/

Asymmetric Encryption - Decryption
c=Enc(PuK_A, m)
m=Dec(PrK_A, c)

Asymmetric Signing - Verification
Sign(PrK_A, m) = σ = (r, s)
V=Ver(PuK_A, m, s), V∈{True, False} ≡ {1, 0}

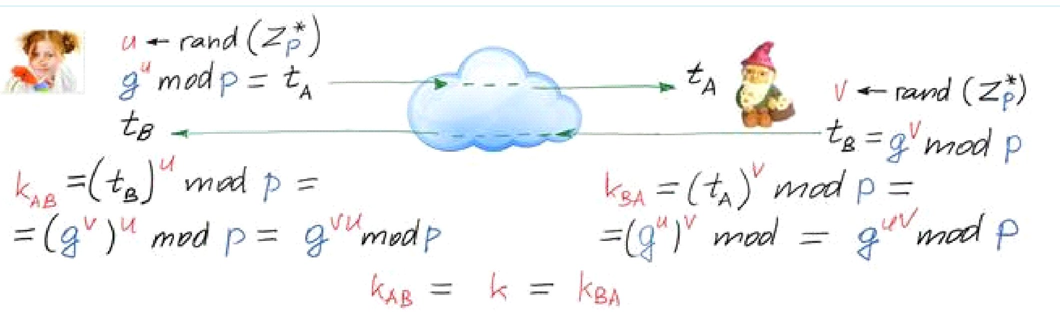




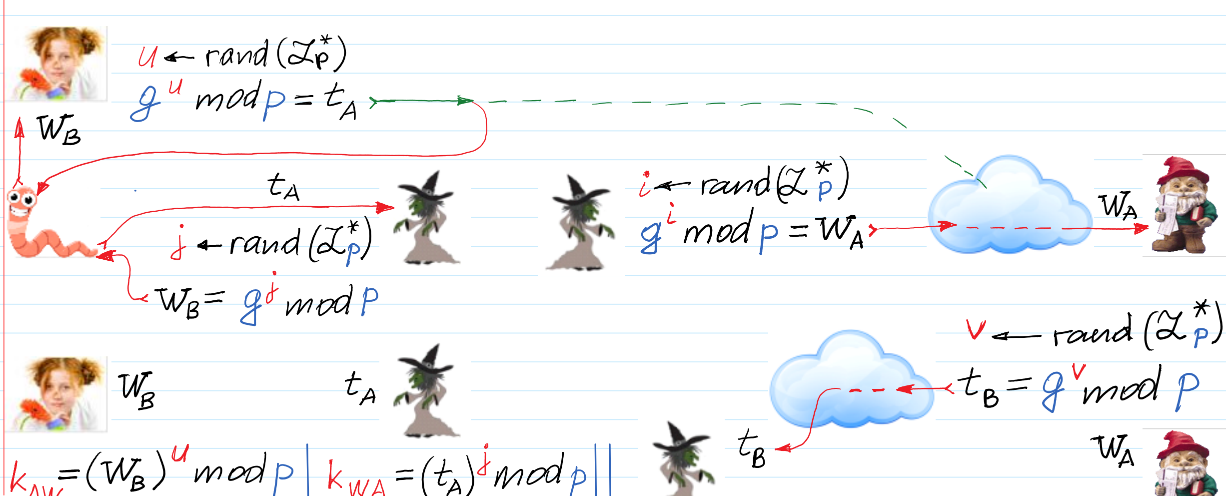
> Documents > 100 MOKYMAS
1 DLI v-4.pptx
2 MIMA.pptx

Diffie-Hellman Key Agreement Protocol (DH KAP)

Public Parameters PP=(p,g); $Z_p^* = \{1, 2, 3, \ldots, p-1\}$; *mod p.



Man in the Middle Attack - MiMA - Impersonation Attack



    t_B W_A 

$$k_{AW} = (W_B)^u \bmod p = (g^j)^u \bmod p = g^{ju} \bmod p$$

$$k_{WA} = (t_A)^j \bmod p = (g^u)^j \bmod p = g^{uj} \bmod p$$

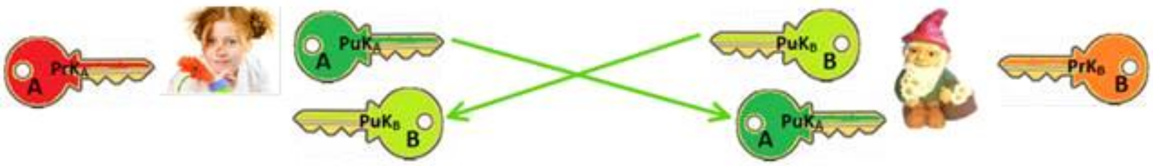
$$k_{AW} = k_1 = k_{WA}$$

$$k_{WB} = (t_B)^i \bmod p = (g^v)^i \bmod p = g^{vi} \bmod p$$

$$k_{BW} = (W_A)^v \bmod p = (g^i)^v \bmod p = g^{iv} \bmod p$$

$$k_{WB} = k_2 = k_{BW}$$

Encryption based KAP - EKAP



k=randi(***Z**ₚ******) → k=randi(Z_p^*)

c_k=Enc(**PuK_B**,***i***,***k***)

M-massage to be encrypted
with symmetric encryption method
e.g. AES128

C_k=E(***k***,***M***)=AES128(***k***,e,***M***)

c_k → C_k

k=Dec(**PrK_B**,c_k)

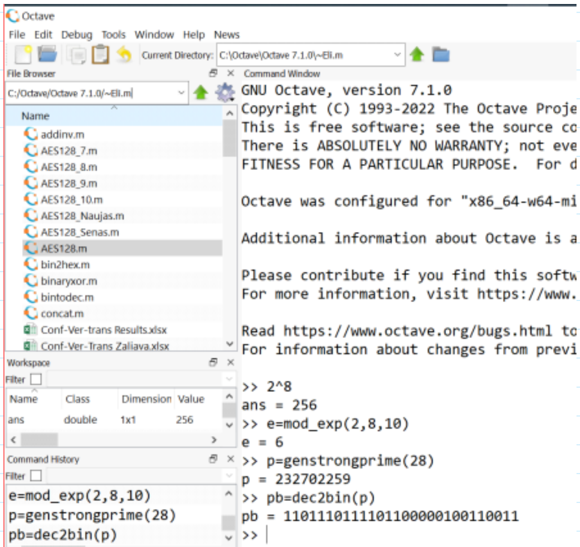
M=D(***k***,d,C_k)=AES128(***k***,d,C_k)

MiMA

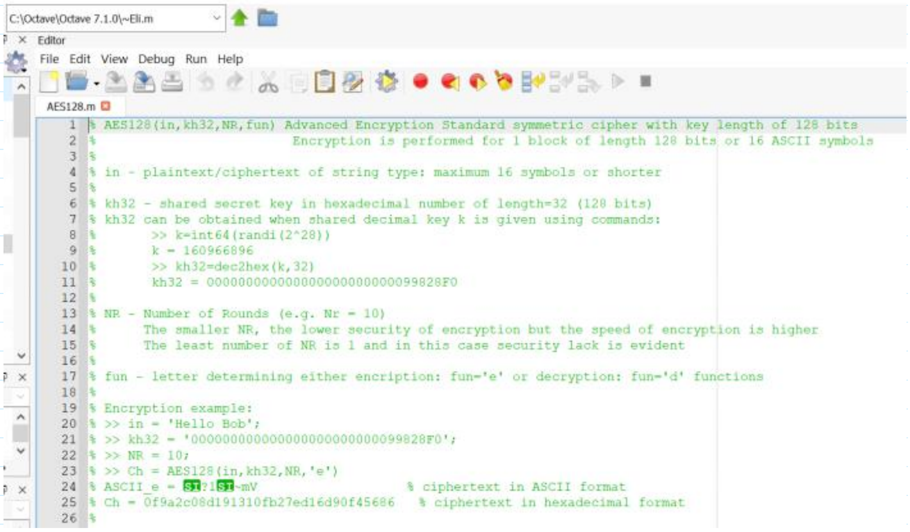
Imagine that W generated her $PrK_W = z$ and $PuK_W = e$.
W send a message to A writing the following message:
„Dear A I am B and I am sending you my PuK = e
for our further communications. Trully yours B."

http://crypto.fmf.ktu.lt/xdownload/

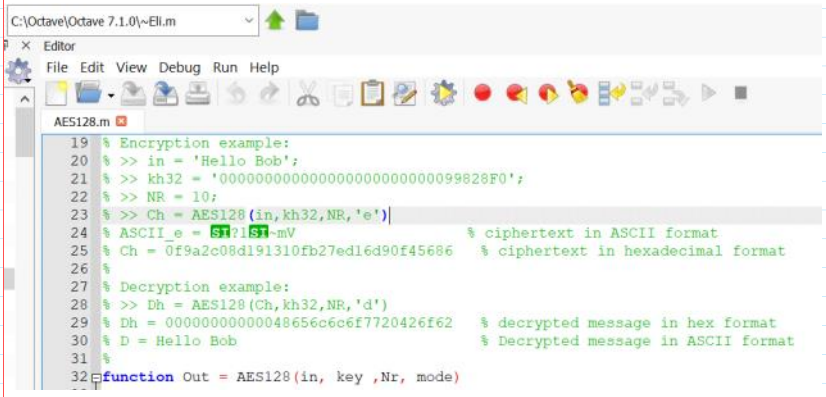
- octave-7.3.0-w64-installer.exe
- octave.Stud.7z



```
GNU Octave, version 7.1.0
Copyright (C) 1993-2022 The Octave Proje
This is free software; see the source co
There is ABSOLUTELY NO WARRANTY; not eve
FITNESS FOR A PARTICULAR PURPOSE.  For d

Octave was configured for "x86_64-w64-mi

Additional information about Octave is a

Please contribute if you find this softw
For more information, visit https://www.

Read https://www.octave.org/bugs.html to
For information about changes from previ

>> 2^8
ans = 256
>> e=mod_exp(2,8,10)
e = 6
>> p=genstrongprime(28)
p = 232702259
>> pb=dec2bin(p)
pb = 1101110111101100000100110011
>> 
```



```
% AES128(in,kh32,NR,fun) Advanced Encryption Standard symmetric cipher with key length of 128 bits
%                        Encryption is performed for 1 block of length 128 bits or 16 ASCII symbols
%
% in - plaintext/ciphertext of string type: maximum 16 symbols or shorter
%
% kh32 - shared secret key in hexadecimal number of length=32 (128 bits)
% kh32 can be obtained when shared decimal key k is given using commands:
%         >> k=int64(randi(2^28))
%         k = 160966896
%         >> kh32=dec2hex(k,32)
%         kh32 = 0000000000000000000000000099828F0
%
% NR - Number of Rounds (e.g. Nr = 10)
%      The smaller NR, the lower security of encryption but the speed of encryption is higher
%      The least number of NR is 1 and in this case security lack is evident
%
% fun - letter determining either encription: fun='e' or decryption: fun='d' functions
%
% Encryption example:
% >> in = 'Hello Bob';
% >> kh32 = '0000000000000000000000000099828F0';
% >> NR = 10;
% >> Ch = AES128(in,kh32,NR,'e')
% ASCII_e = SI?1SI~mV                    % ciphertext in ASCII format
% Ch = 0f9a2c08d191310fb27ed16d90f45686   % ciphertext in hexadecimal format
%
```



```
% Encryption example:
% >> in = 'Hello Bob';
% >> kh32 = '0000000000000000000000000099828F0';
% >> NR = 10;
% >> Ch = AES128(in,kh32,NR,'e')
% ASCII_e = SI?1SI~mV                    % ciphertext in ASCII format
% Ch = 0f9a2c08d191310fb27ed16d90f45686   % ciphertext in hexadecimal format
%
% Decryption example:
% >> Dh = AES128(Ch,kh32,NR,'d')
% Dh = 0000000000000048656c6c6f7720426f62   % decrypted message in hex format
% D = Hello Bob                            % Decrypted message in ASCII format
%
function Out = AES128(in, key ,Nr, mode)
```

Private and Public keys generation: $PrK = x$; $PuK = a$;

1) Generate strong prime number p.

>> p = genstrongprime(28) % generates 28 bit lenghts of p

2) Find a generator g in the set $Z_p^* = \{1, 2, 3, ..., p-1\}$

If p is strong prime, then p = 2•q+1, when q is also prime: 11=2•5+1; 23=2•11+1.

>> q = (p-1)/2

>> g = 2

>> mod_exp(g, q, p) ≠ 1 % I-st condition
% If it is equal to 1 → choose the other g
% If no, then verify:

>> mod_exp(g, 2, p) ≠ 1 % II-nd condition
% If it is equal to 1 → choose the other g.

3) Generate $PrK = x$ using random number generator function randi

>> x = int64 (randi (2^{28} - 1))

>> x=randi(2^28-1)
x = 1.0006e+08

3) Generate PrK = x using random number generator function randi

>> x = int64 (randi (2^{28} - 1))

4) compute PuK = a using DEF, i.e. function

>> a = mod_exp(g, x, p)

```
>> x=randi(2^28-1)
x = 1.9906e+08
>> x=int64(randi(2^28-1))
x = 256210849
```

PrK_A = *x* <-- randi ==> PuK_A = *a* = g^x mod p

PrK_B = *y* <-- randi ==> PuK_B = *b* = g^y mod p

For example:

```
>> k=randi(2^28-1)
k = 3.6754e+07
>> k=int64(randi(2^28-1))
k = 233360567
>> kb=dec2bin(k)
kb = 1101111010001100110010110111
>> maxb=dec2bin(2^28-1)
maxb = 1111111111111111111111111111
>> max=int64(2^28-1)
max = 268435455
```

```
>> p=genstrongprime(28)
p = 227317067
>> isprime(p)
ans = 1
>> q=(p-1)/2
q = 113658533
>> isprime(q)
ans = 1
>>
>> g=2
g = 2
>> mod_exp(g,q,p)
ans = 227317066
>> mod_exp(g,2,p)
ans = 4
```

```
>> x=int64(randi(2^28-1))
x = 125675623
>> a=mod_exp(g,x,p)
a = 67591766
```