Cryptography:
Information confidentiality, integrity, authenticity, person identification

1976 m.

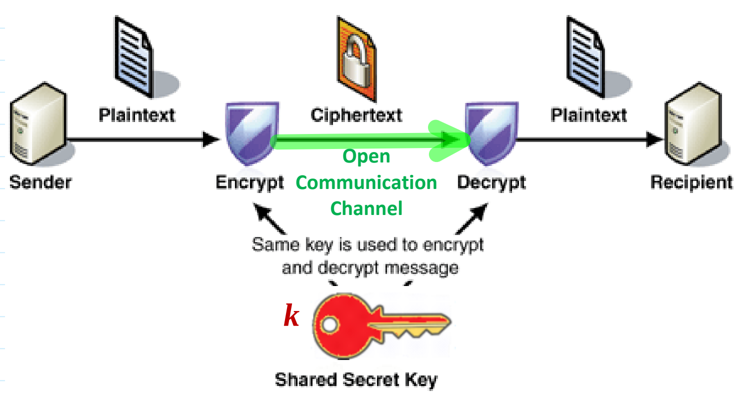
Symmetric cryptography -------------------- Asymmetric cryptography

Symmetric encryption:
- block ciphers
- stream ciphers

H-functions, Message digest
HMAC H-Message Authentication Code

Asymmetric encryption
E-signature - Public Key Infrastructure - PKI
E-money
E-voting
Digital Rights Management - DRM Marlin
Etc.

Symmetric - Secret Key Encryption - Decryption



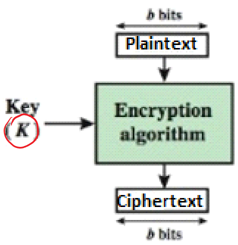
Imagine that number of users of cryptosystem is 100.

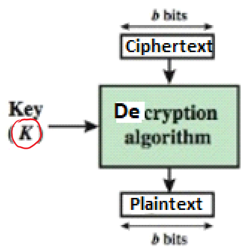
$$C_{100}^{2} = \frac{100*99}{2} = 4950$$

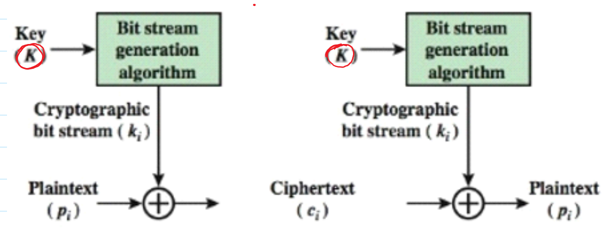
Symmetric ciphers

AES: **Block Ciphers** — **Stream Ciphers**

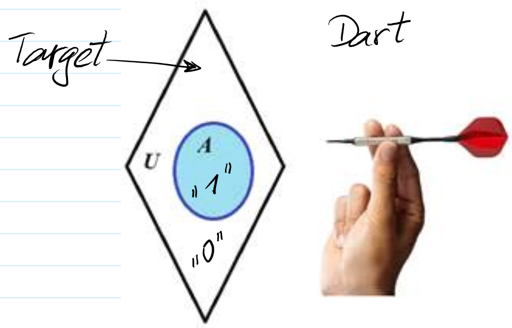
128, 196, 256







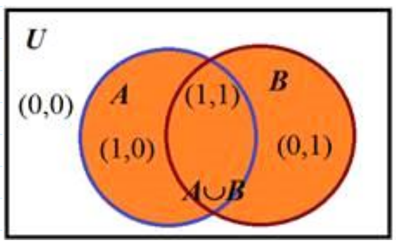
bitwise XOR

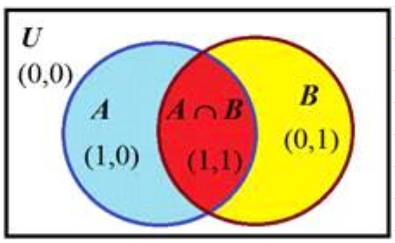


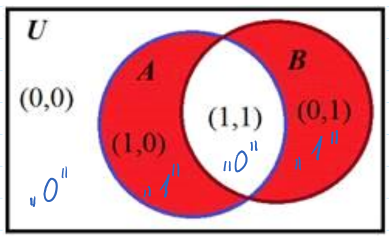
Vernam cipher (1917) - One Time Pad

Logical operations

$A \cup B$ $\qquad$ $A \cap B$ $\qquad$ $A \oplus B$







„0" No

„1" Yes

$m \in \{0,1\}$

A	B	A ⊕ B
0	0	0 ←
0	1	1
1	0	1
1	1	0 ←

$k \leftarrow rand\{0,1\}$; $k \in \{0,1\}$

$c = m \oplus k$ $\xrightarrow{\quad c \quad}$

m	k	m ⊕ k = c
0	0	0 ←
0	1	1 ←
1	0	1 ←
1	1	0 ←

If $Pr(k=0) = Pr(k=1) = 1/2$

Then $Pr(c=0) = Pr(c=1) = 1/2$

$\quad \rightarrow Pr(m=0) = Pr(m=1) = 1/2$

⊕ - is inverse to itself

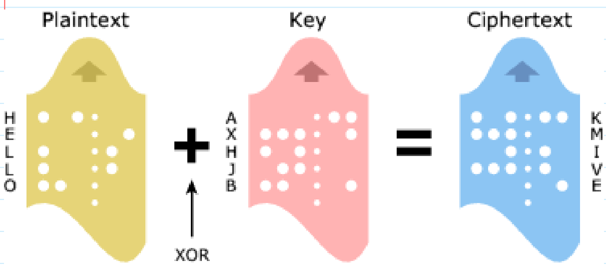
B: $c = m \oplus k - k = m$

$c \oplus k = m \oplus k \oplus k =$

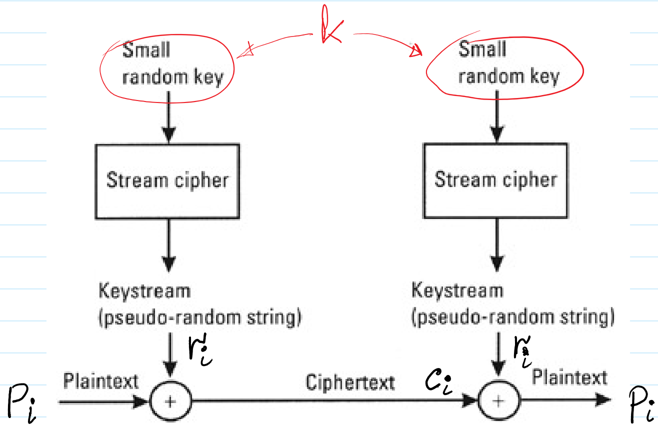
$= m \oplus 0 = m = 1$

$c \oplus k = m$

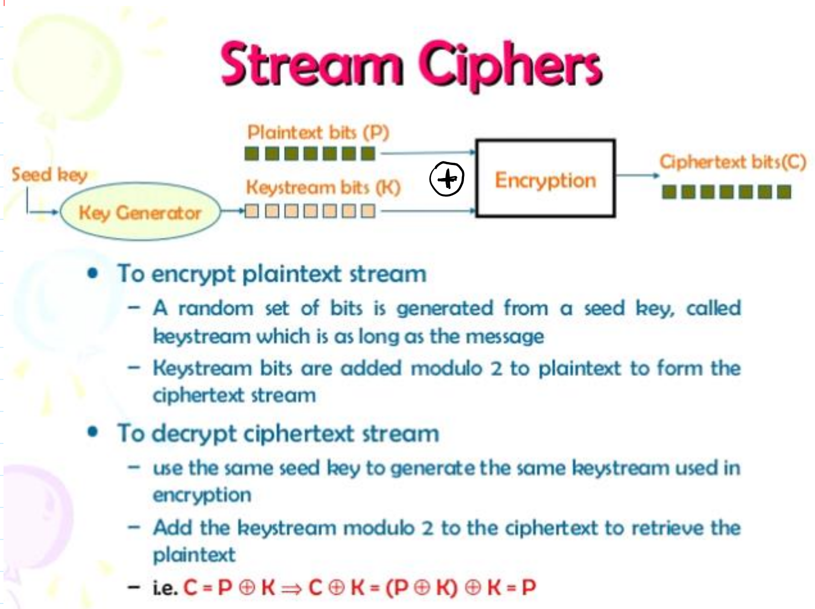
Stream Cipher - Vernam Cipher - One-Time Pad



bitwise



2^{64}



Pseudo Random Numbers Generators - PRNG FIPS - 142

$r_{i+1} = PRNG(r_i)$

$r_1 = PRNG(r_0)$; r_0 - initial value: $r_0 = k$.

$r_2 = PRNG(r_1)$

– – – – – – –

$r_n = PRNG(r_{n-1})$

Vernam cipher: Plaintext ***m*** --> Encryption $E_k(m)=c$ --> Ciphertext ***c***.

For general encryption and decryption bitwise XOR operation $\oplus$ is used for bitstrings.
Plaintext ***m*** and key ***K*** (bit stream equal *to* ***m*** bit stream)are transformed to binary form consisting of bitstrings.

Encryption: $c = m \oplus K$.

Decryption: $m = c \oplus K = m \oplus K \oplus K = m \oplus 0 = m$.

Security requirements.
1.Key K must be generated at random using (*Pseudo*)Random Number Generators - PRNG.
2.Key K bit length must be no less than plaintext bit length: $|k|>=|m|$.
3.Key K can be used only once.

Active adversary, Eavesdropping adversary.

Attention! If the same agreed secret key K is used twice in Vernam cipher for any two messages m_1 and m_2 encryption, then ***eavesdropping*** adversary can obtain an data d_b which is equal to bitwise XOR between m_1 and m_2. Let ciphertexts c_1 and c_2 are obtained by the following encryption with the same symmetric key K

$c_1 = m_1 \oplus K$,

$c_2 = m_2 \oplus K$,

where $\oplus$ is bitwise XOR operation.
Then eavesdropping adversary computes the following data d

$d_b = c_1 \oplus c_2 = m_1 \oplus K \oplus m_2 \oplus K = m_1 \oplus m_2 \oplus K \oplus K = m_1 \oplus m_2 \oplus 0 = 0 = m_1 \oplus m_2$.

It is reckoned as a ***crucial insecurity*** since cryptanalysis of data d_b is significantly facilitated and both m_1 and m_2 can be disclosed.
Moreover, if any message of two m_1 or m_2 are revealed by some circumstances, say message m_2, then the other message m_1 becomes clear to the adversary by computing

$d_b \oplus m_2 = m_1 \oplus m_2 \oplus m_2 = m_1 \oplus 0 = m_1$.

Never use the same secret key K twice in Vernam cipher!
The same secret key K can be used multiple times in standardized block ciphers (AES) and stream ciphers.

```
>> kAB = int64(195681379)
kAB = 195681379
>> k=kAB
k = 195681379
>> kb=dec2bin(k)
kb = 1011101010011101110001100011

>> m1=120000
m1 = 120000
>> m1b=dec2bin(m1)
m1b = 11101010011000000
>> m2=9
m2 = 9
>> m2b=dec2bin(m2)
m2b = 1001
```

```
>> c1b=binaryxor(m1b,kb)
c1b = 1011101010000000100010100011
>> c2b=binaryxor(m2b,kb)
c2b = 1011101010011101110001101010
>> c12b=binaryxor(c1b,c2b)
c2b = 1011101010011101110001101010

>> c12b=binaryxor(c1b,c2b)
c12b = 11101010011001001
>> m12b=binaryxor(m1b,m2b)
m12b = 11101010011001001
>> db=c12b
db = 11101010011001001
>> mm1b=binaryxor(db,m2b)
mm1b = 11101010011000000
>> mm1b=bin2dec(mm1b)
mm1b = 120000
```

Block cipher AES - 128, 192, 256 --> Encryption --> Decryption

Advanced Encryption Standard ~ 2000

Key length 128, 192, 256 bits: $|k| \in \{128\,b, 192\,b, 256\,b\}$

Data to be encripted : message m

| P_1 | P_2 | P_3 | ---- | P_n |

The length of any block B_i should be $|B_i| = 128$ bits

192 bits

256 bits

$|B_i| = |k| = 128$ bits

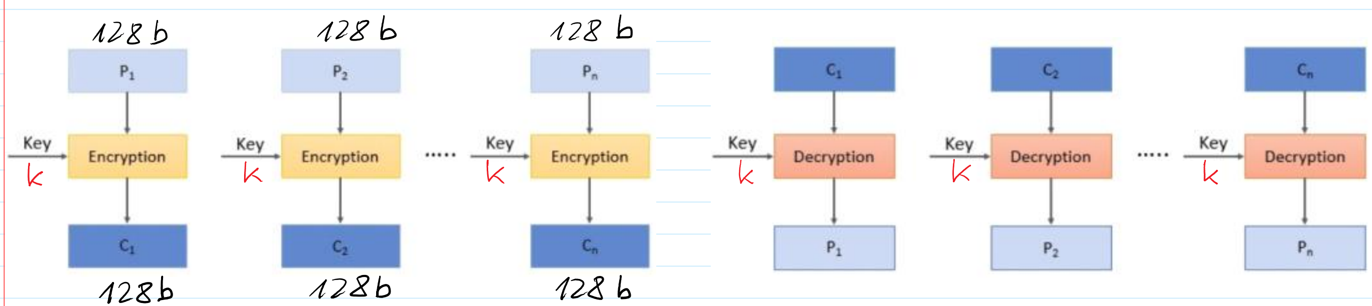
Block Cipher: Electronic Code Book -**ECB** mode of encryption

From <https://binaryterms.com/block-cipher.html>

1. Electronic Code Book (ECB) mode in AES-128

This is considered to be the easiest block cipher mode of operation. In electronic codebook mode (ECB) the plain text is divided into the blocks, each of 128-bit. Each block is encrypted one at a time to produce the cipher block. The same key is used to encrypt each block.

When the receiver receives the message i.e. ciphertext. This ciphertext is again divided into blocks, each of 128-bit and each block is decrypted independently one at a time to obtain the corresponding plain text block. Here also the same key is used to decrypt each block which was used to encrypt each block.

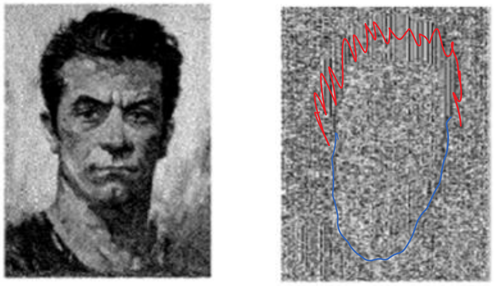


$EncAES(k, P_1) = C_1$

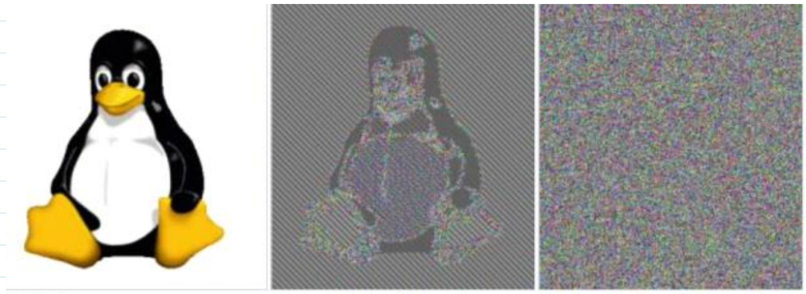
$EncAES(k, P_2) = C_2$

- - - - - - - - - -

$EncAES(k, P_n) = C_n$



(a) plaintext (b) plaintext encrypted in ECB mode using AES



Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

https://binaryterms.com/block-cipher.html

2. Cipher Block Chaining - CBC Mode

To overcome the limitation of ECB i.e. the repeating block in plain text produces the same ciphertext, a new technique was required which is Cipher Block Chaining (CBC) Mode. CBC confirms that even if the plain text has repeating blocks its encryption won't produce same cipher block.

To achieve totally different cipher blocks for two same plain text blocks **chaining** has been added to the block cipher. For this, the result obtained from the encryption of the first plain text block is fed to the encryption of the next plaintext box.

In this way, each ciphertext block obtained is dependent on its corresponding current plain text block input and all the previous plain text blocks. But during the encryption of first plain text block, no previous plain text block is available so a random block of text is generated called **Initialization vector**.

Now let's discuss the encryption steps of CBC

Step 1: The initialization vector and first plain text block are XORed and the result of XOR is then encrypted using the key to obtain the first ciphertext block.

$C_{01} = IV \oplus k$

Step 2: The first ciphertext block is fed to the encryption of the second plain text block. For the encryption of second plain text block, first ciphertext block and second plain text block is XORed and the result of XOR is encrypted using the same key in step 1 to obtain the second ciphertext block.

Similarly, the result of encryption of second plain text block i.e. the second ciphertext block is fed to the encryption of third plain text block to obtain third ciphertext block. And the process continues to obtain all the ciphertext blocks.

Decryption Steps:

$IV \oplus k = C_{10}$

CBC	
Cipher block chaining	
Encryption parallelizable:	No
Decryption parallelizable:	Yes

Step 1: The initialization vector is placed in the shift register. It is encrypted using the same key.

Keep a note that even in the **decryption process** the **encryption** algorithm is implemented instead of the decryption algorithm.

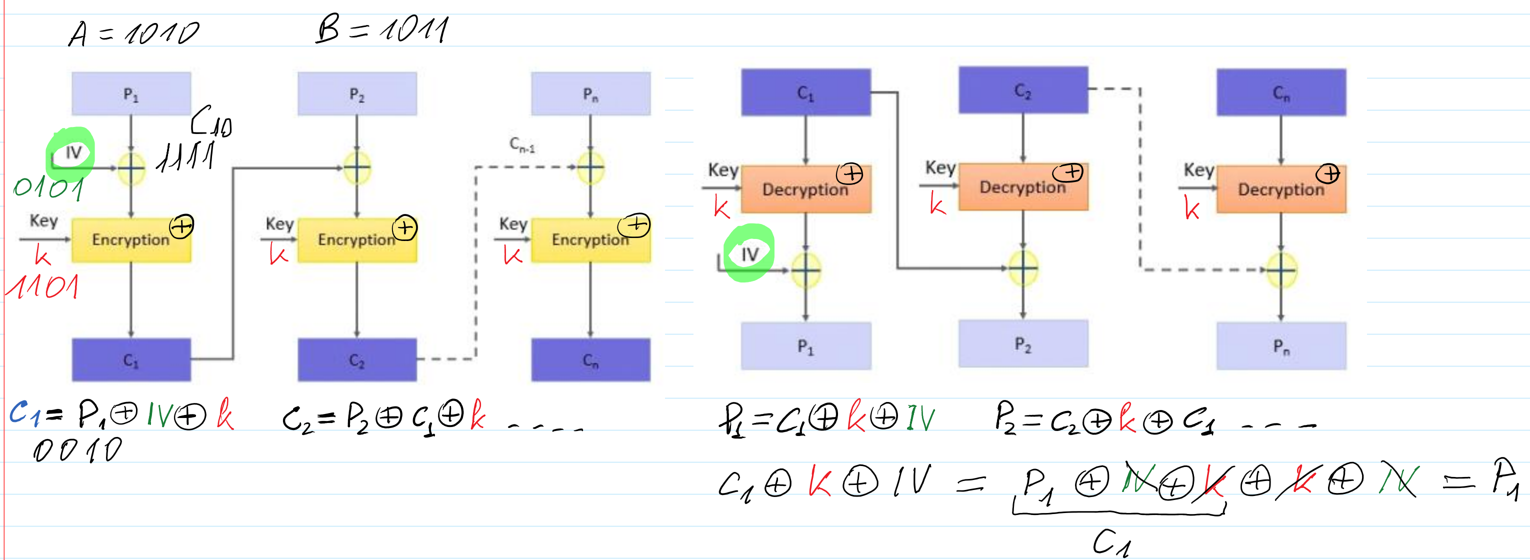
Then from the encrypted IV s bits are XORed with the s bits ciphertext C1 to retrieve s bits plain text P1.

parallelizable:	
Decryption parallelizable:	Yes
Random read access:	Yes

Then from the encrypted IV s bits are XORed with the s bits ciphertext C1 to retrieve s bits plain text P1.

Step 2: The IV in the shift register is left-shifted by s bits and the s bits C1 replaces the rightmost s bits of IV.

The process continues until all plain text fragments are retrieved.



https://binaryterms.com/block-cipher.html

5. Counter Mode - CTR

It is similar to OFB but there is no feedback mechanism in counter mode. Nothing is being fed from the previous step to the next step instead it uses a sequence of number which is termed as a **counter** which is input to the encryption function along with the key. After a plain text block is encrypted the counter value increments by 1.

Steps of encryption:

Step1: The counter value is encrypted using a key.

Step 2: The encrypted counter value is XORed with the plain text block to obtain a ciphertext block.

To encrypt the next subsequent plain text block the counter value is incremented by 1 and step 1 and 2 are repeated to obtain the corresponding ciphertext.

The process continues until all plain text block is encrypted.

CTR	
Counter	
Encryption parallelizable:	Yes
Decryption parallelizable:	Yes
Random read access:	Yes

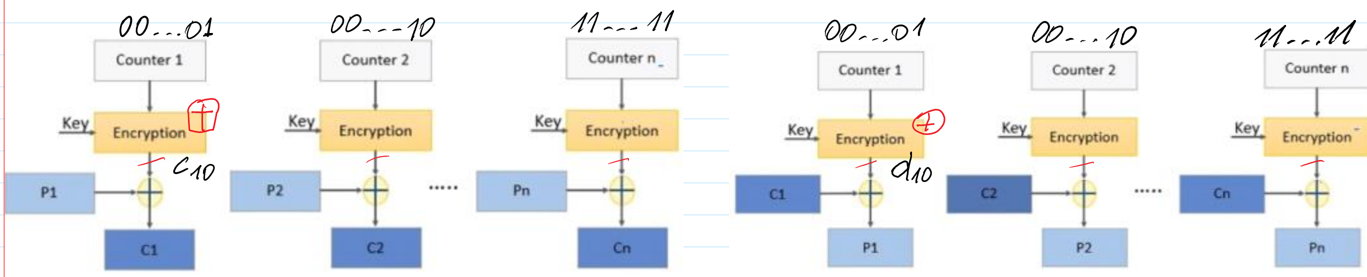
Steps for decryption:

Step1: The counter value is encrypted using a key.

Note: Encryption function is used in the decryption process. The same counter values are used for decryption as used while encryption.

Step 2: The encrypted counter value is XORed with the ciphertext block to obtain a plain text block.

obtain a plain text block.



$C_{10} = CTR_1 \oplus k$

$C_1 = C_{10} \oplus P_1$

$d_{10} = CTR_1 \oplus k$

$c_{10} = d_{10}$

$C_1 \oplus d_{10} = C_{10} \oplus P_1 \oplus CTR_1 \oplus k = \; =$

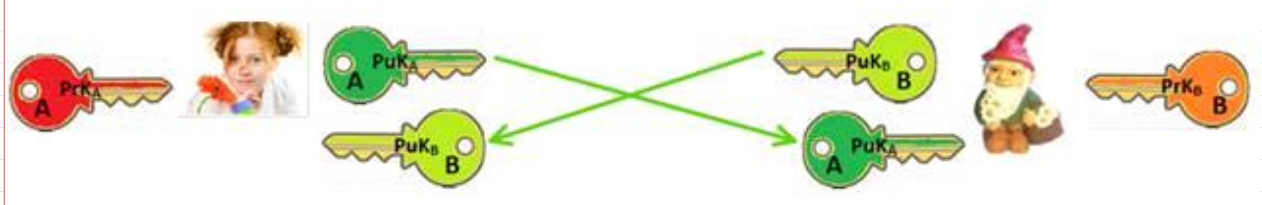
$= CTR_1 \oplus k \oplus P_1 \oplus CTR_1 \oplus k = P_1 .$

```
% 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 = 00000000000000000000000099828F0
%
% 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 = '00000000000000000000000099828F0';
% >> NR = 10;
% >> Ch = AES128(in,kh32,NR,'e')
% ASCII_e =   ?1  ~mV                  % ciphertext in ASCII format
% Ch = 0f9a2c08d191310fb27ed16d90f45686   % ciphertext in hexadecimal format
%
% Decryption example:
% >> Dh = AES128(Ch,kh32,NR,'d')
% Dh = 00000000000048656c6c6f7720426f62   % decrypted message in hex format
% D = Hello Bob                          % Decrypted message in ASCII format
%
function Out = AES128(in, key ,Nr, mode)
```

Encryption security depends of the number of rounds - NR
Test when NR=1
NR=10
And compare ciphertexts in hex format.

Till this place

Asymmetric cryptography main actors and their credentials.



A:

$u \leftarrow rand(\mathbb{Z}_p^*)$

$t_A = g^u \mod p$

$PP = (p, g)$

$t_A \rightarrow$

$\leftarrow t_B$

B:

$v \leftarrow rand(\mathbb{Z}_p^*)$

$t_A = g^v \mod p$

$t_A = g^u \bmod p$ $\xleftarrow{\ t_B\ }$ $t_A = g^v \bmod p$

$k_{AB} = (t_B)^u \bmod p$ $=$ k $=$ $k_{BA} = (t_A)^v \bmod p$

$k \rightarrow k_b$ binary format

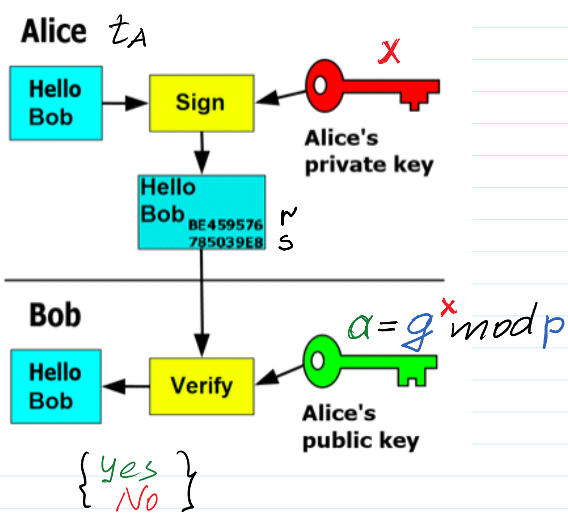
$M =$ 'A 25000 E --> B'

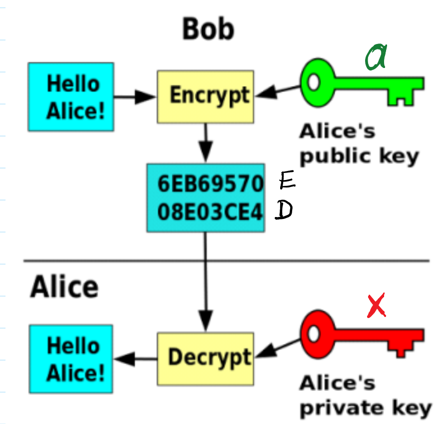
$M = 25000$

$M \rightarrow M_b$

$C_b = M_b \oplus k_b$ $\xrightarrow{\ C_b\ }$ $M_b = C_b \oplus k_B$

Authenticated KAP − (AKAP)



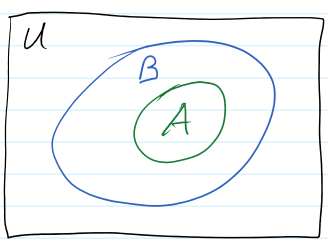


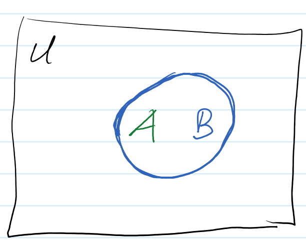
Till this place

The reader confusing implication and equivalence operations (functions) can accept the following proposition as valid:

if talker has a head and donkey has a head, then talker is a donkey.

To accept this proposition as valid means that thinker confuses notions of implication and equivalence. If reader is afraid to make such a mistake, we recommend to read about that in any external source.





Implication: If $A \xrightarrow{\text{then}} B$; Equivalence: If $A \xrightarrow{\text{then}} B$ & If $B \xrightarrow{\text{then}} A$

Implication: If $A \xrightarrow{\text{then}} B$; Equivalence: If $A \xrightarrow{\text{then}} B$ & If $B \xrightarrow{\text{then}} A$