Lecture 5

Cryptography 1: block ciphers

$whoami

Jarosław Jedynak

- CERT Polska / p4 ctf
- Malware researcher
- Dawniej programista
- Dekryptory do ransomware, łamanie krypto innych ludzi, testy bezpieczeństwa, CTF
- msm@tailcall.net

Adam \`adami\` Iwaniuk

- Sumo Logic Poland
- Dragon Sector member
- kontakt@adami.pl
- crypto, re, algo, web, low-level

Today: Block Ciphers

- Block ciphers: theory
- ECB mode & attacks
- CBC mode & attacks
- Feistel ciphers (...)
- (...)

XOR

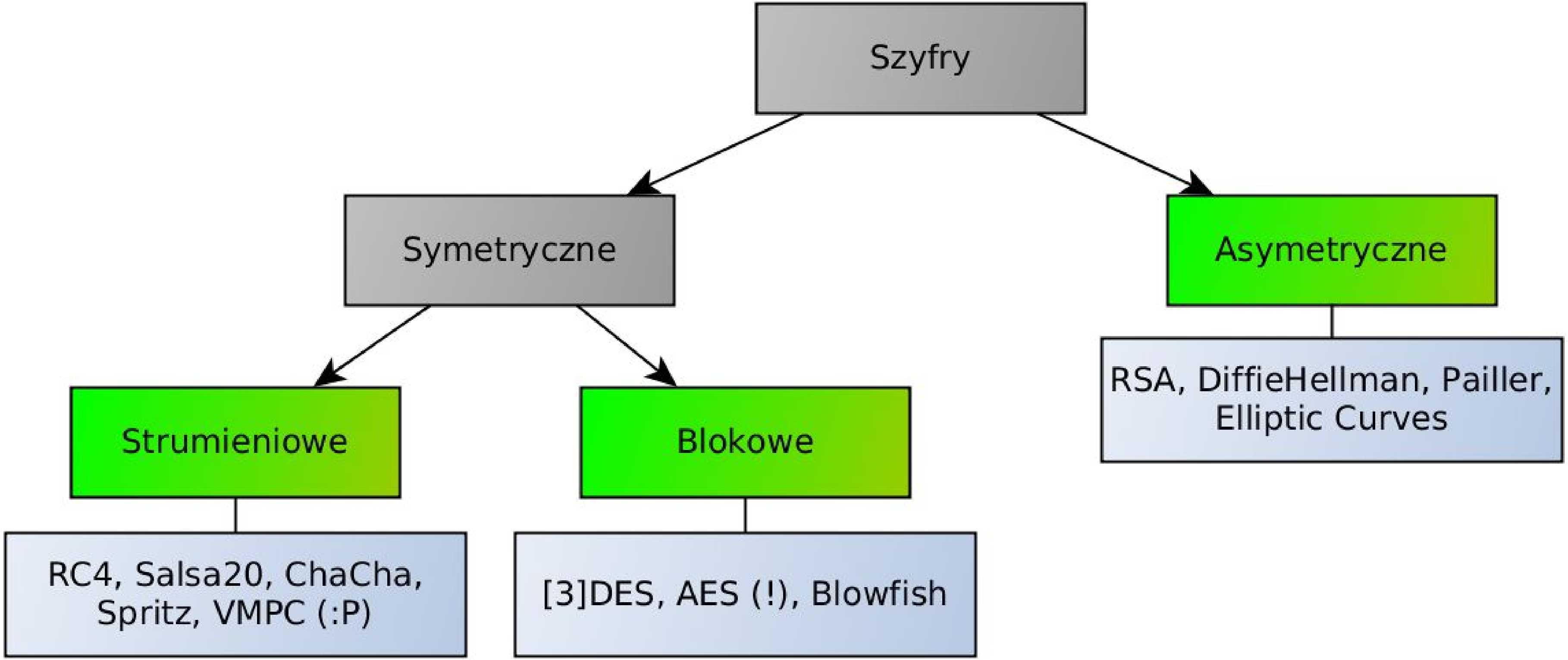
```
def xor(a, b):
    return ''.join(chr(ord(ac)^ord(bc)) for ac, bc in zip(a, b))

def xor(a, b):
    out = ''
    for i in range(min(len(a), len(b))):
        out += chr(ord(a[i]) ^ ord(b[i]))
    return out
```

Block Ciphers

Block ciphers: theory

- Block ciphers vs stream ciphers
- Block encryption functions
- More theory

Ciphers



Szyfry

Symetryczne

Asymetryczne

RSA, DiffieHellman, Pailler, Elliptic Curves

Strumieniowe

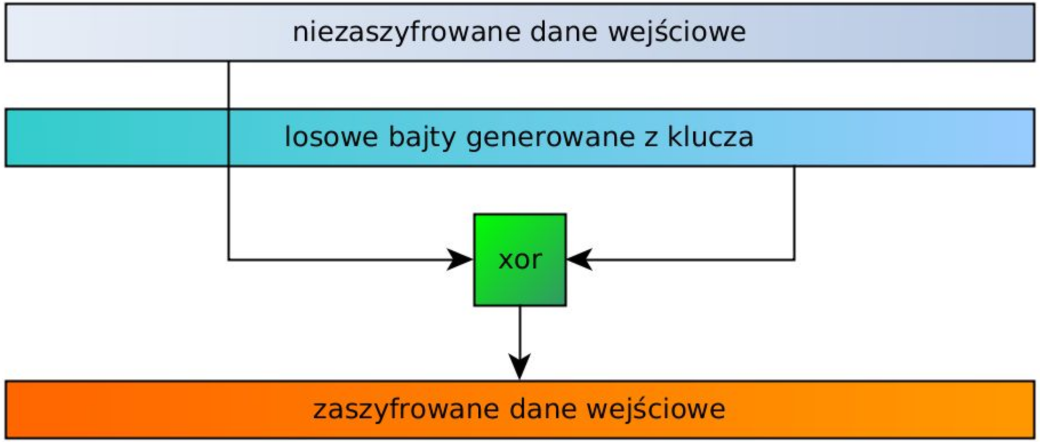
Blokowe

RC4, Salsa20, ChaCha, Spritz, VMPC (:P)

[3]DES, AES (!), Blowfish

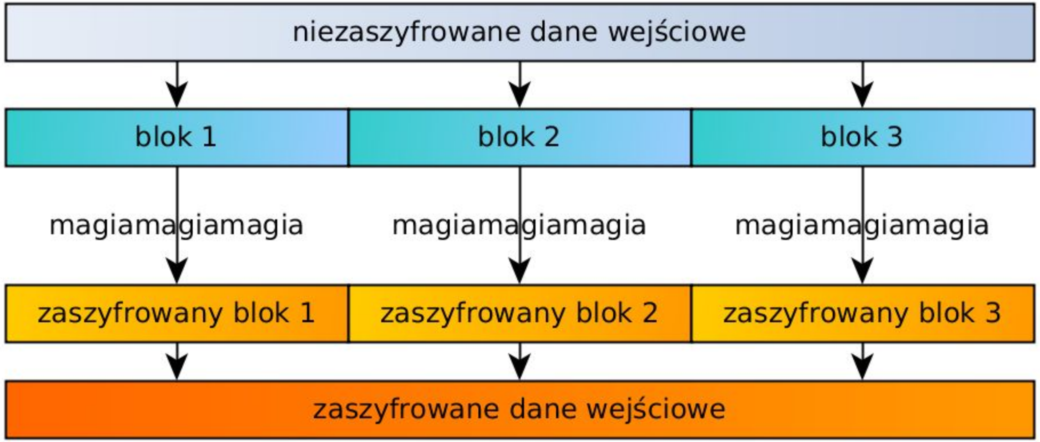
Stream cipher (simplified)

- Examples: RC4 (!), Salsa20, ChaCha, Spritz, VMPC (:P)
- "plaintext digits are encrypted with corresponding digits of the keystream, to give digits of ciphertext stream"


niezaszyfrowane dane wejściowe
losowe bajty generowane z klucza
xor
zaszyfrowane dane wejściowe

Block cipher (simplified)

- Examples: [3]DES (obsolete), AES (!), Blowfish
- "algorithm operating on fixed-length groups of bits (blocks), with transformation specified by symmetric key"



PKCS#7 padding scheme

- Plaintext length must be multiple of block length
- What to do when it isn't?
- Padding schemes
- PKCS#7 padding scheme

PKCS#7 Valid Padding

'A'	'B'	'C'					
41	42	43	05	05	05	05	05

'A'	'B'	'C'	'D'				
41	42	43	44	04	04	04	04

'A'	'B'	'C'	'D'	'E'			
41	42	43	44	45	03	03	03

'A'	'B'	'C'	'D'	'E'	'F'		
41	42	43	44	45	46	02	02

'A'	'B'	'C'	'D'	'E'	'F'	'G'	
41	42	43	44	45	46	47	01

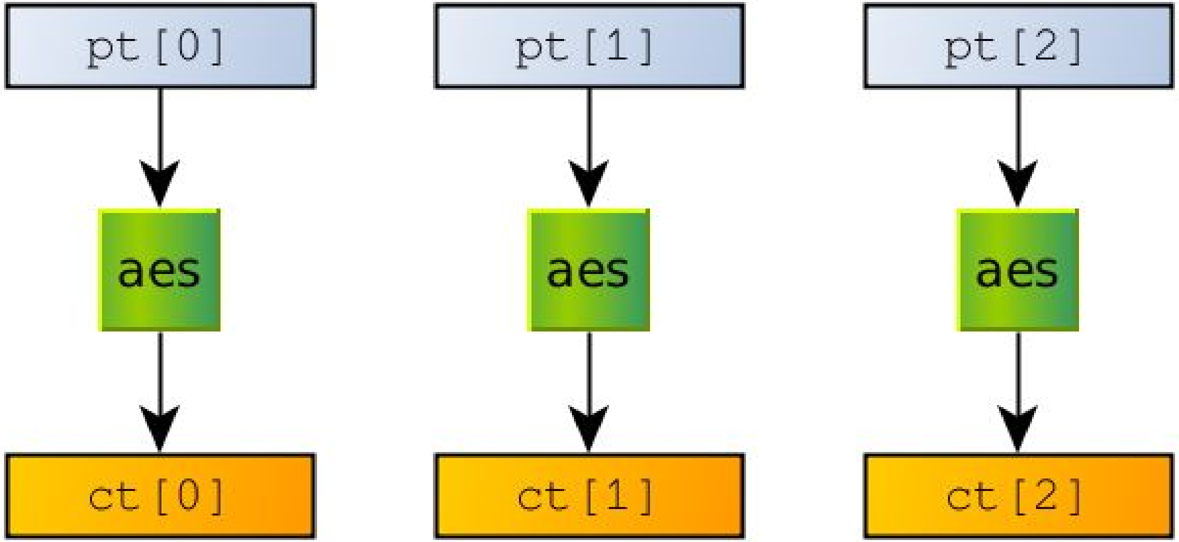
'A'	'B'	'C'	'D'	'E'	'F'	'G'	'H'								
41	42	43	44	45	46	47	48	08	08	08	08	08	08	08	08

Cipher modes: ECB, CBC

OFB mode... CTR mode...

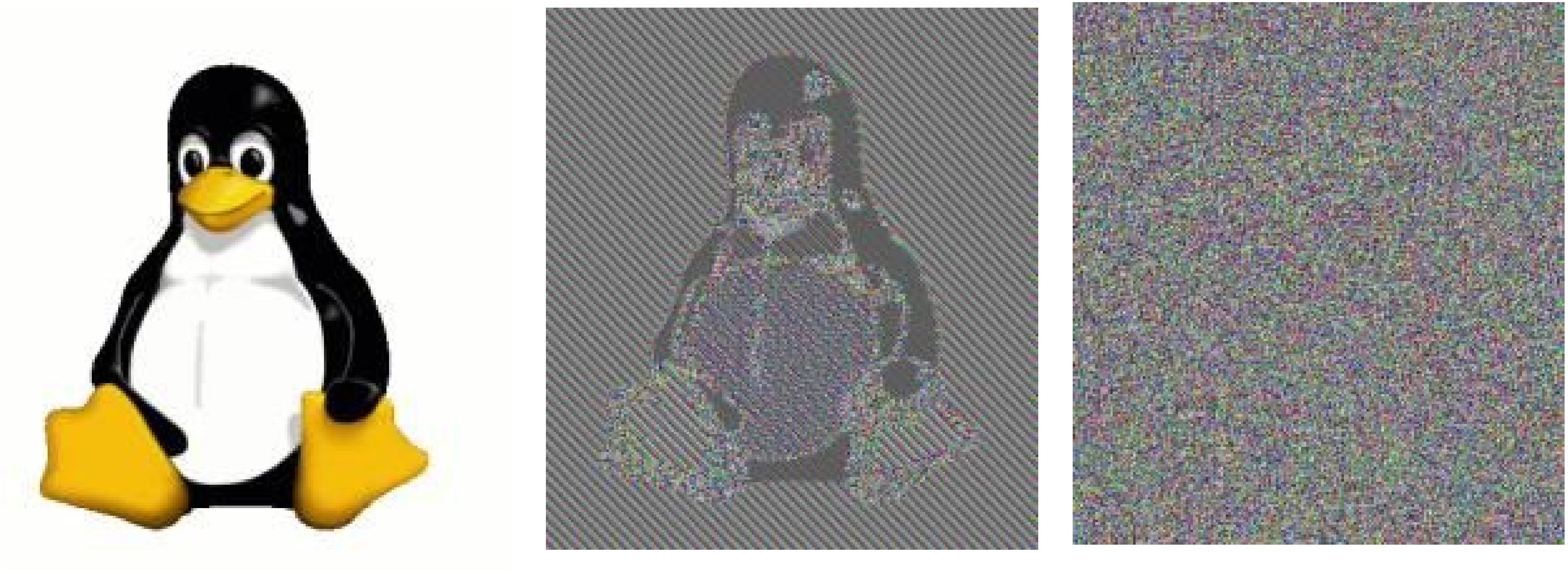
ECB Mode

Simplest encryption mode possible



ECB Mode

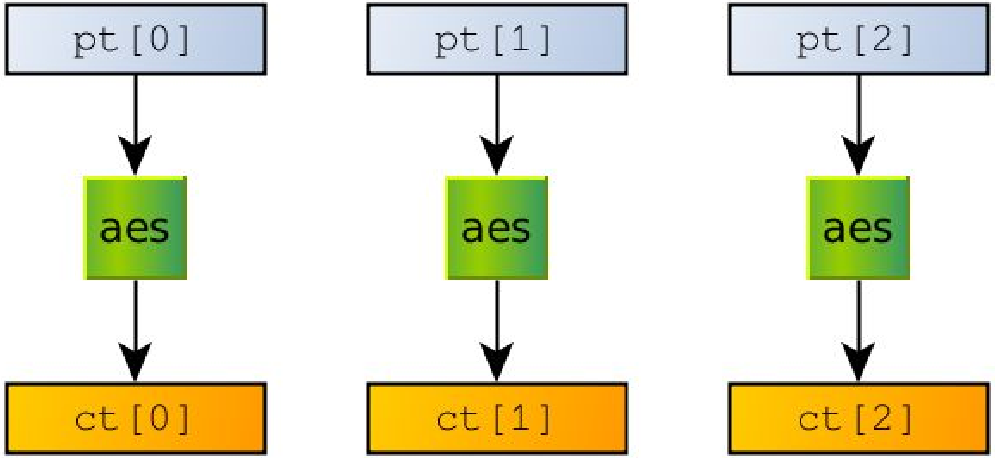
Obligatory penguin image



ECB mode attacks

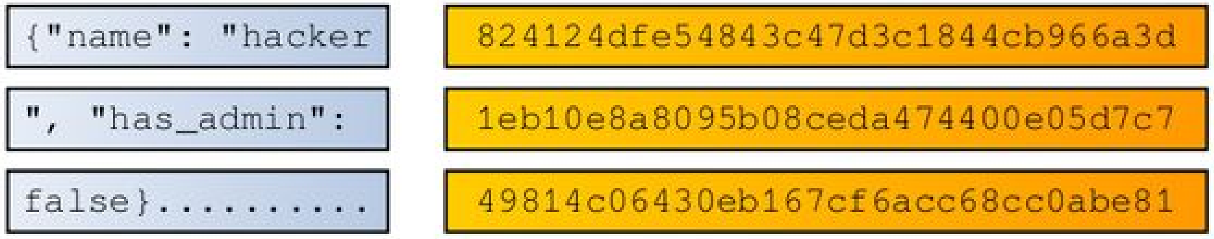
https://var.tailcall.net/ecb

ECB Mode Attack: Copy&Paste

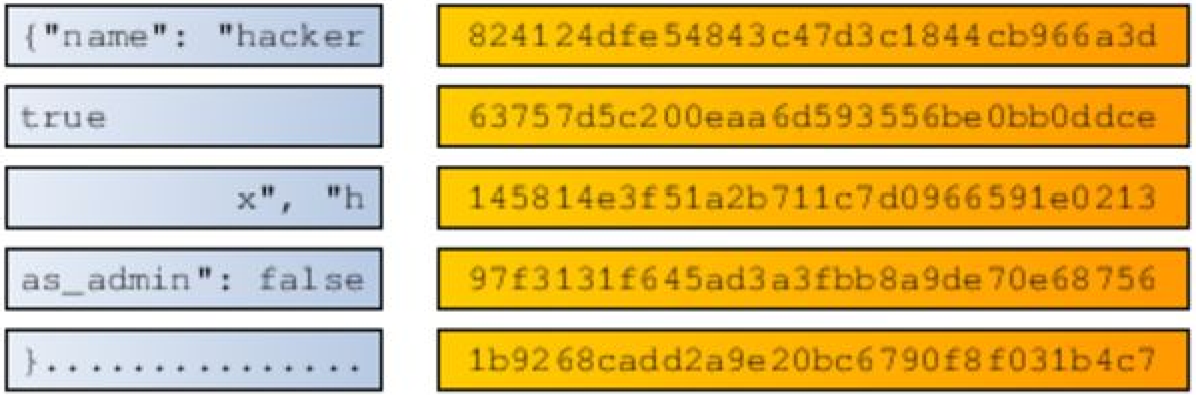


```
{'username': 'alamakota12345', 'is_admin': 'false'}
{'username': 'alamakota12345', 'is_admin': 'true'}
```

ECB Mode Attack: Copy&Paste

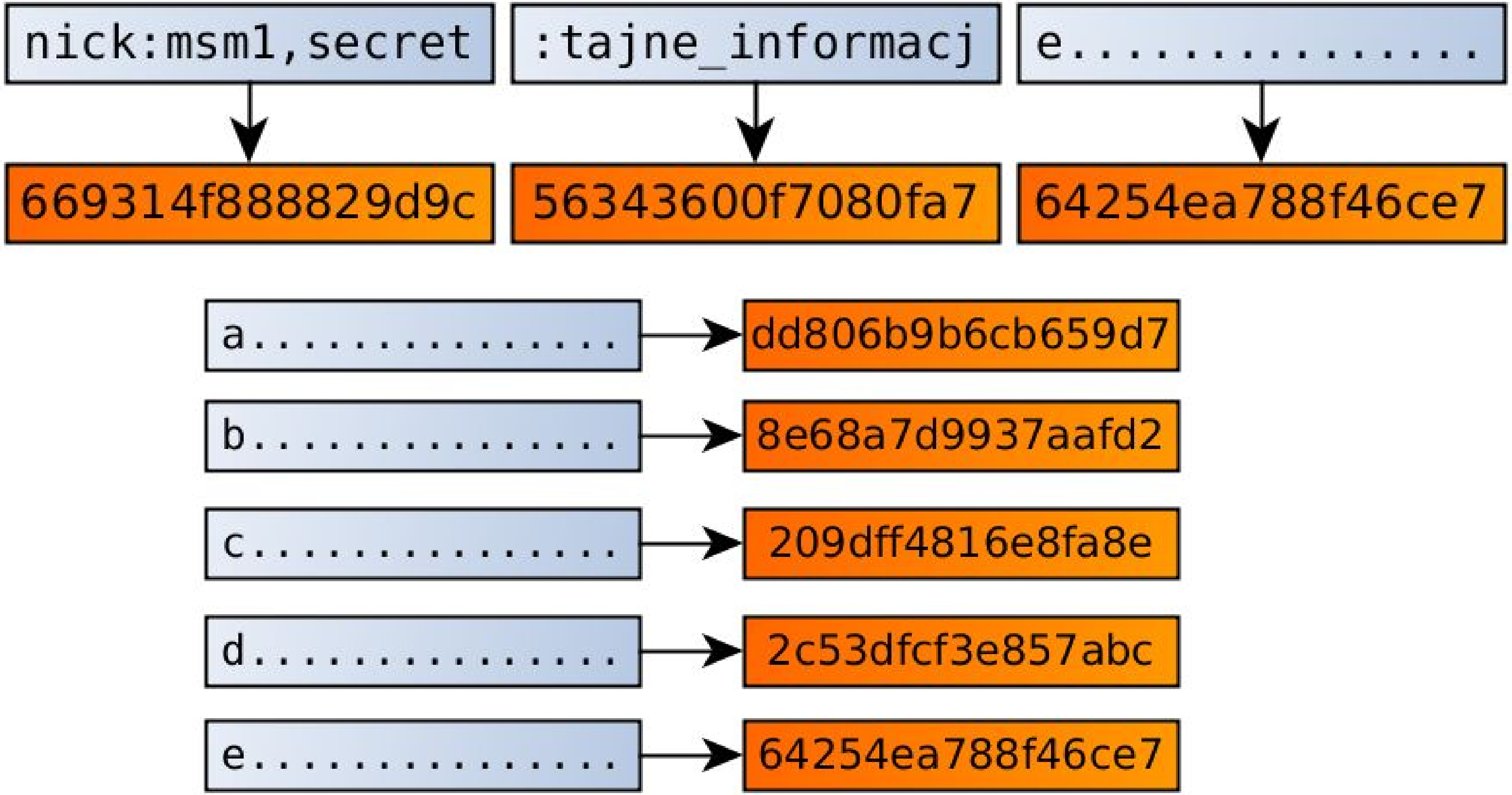


{"name": "hacker	824124dfe54843c47d3c1844cb966a3d
", "has_admin":	1eb10e8a8095b08ceda474400e05d7c7
false}..........	49814c06430eb167cf6acc68cc0abe81



{"name": "hacker	824124dfe54843c47d3c1844cb966a3d
true	63757d5c200eaa6d593556be0bb0ddce
x", "h	145814e3f51a2b711c7d0966591e0213
as_admin": false	97f3131f645ad3a3fbb8a9de70e68756
}...............	1b9268cadd2a9e20bc6790f8f031b4c7

ECB Mode Attack: Decryption



nick:msm1,secret

:tajne_informacj

e...............

669314f888829d9c

56343600f7080fa7

64254ea788f46ce7

a............... → dd806b9b6cb659d7

b............... → 8e68a7d9937aafd2

c............... → 209dff4816e8fa8e

d............... → 2c53dfcf3e857abc

e............... → 64254ea788f46ce7

"Encryption is not authentication"

- What does it mean?
- Why?
- What is authentication?

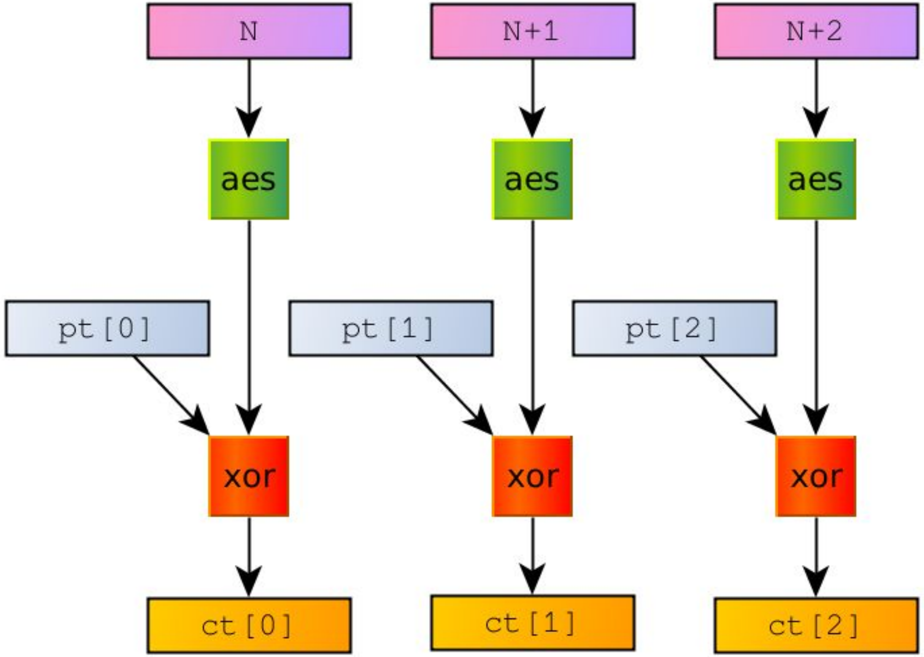
Off topic: so what is authentication?

- Hashes?
- Md5? Sha1? Sha256?
- Nope (why?)
- Message **Authentication** Codes
- HMAC construction

$$HMAC(K,m) = H\Big((K' \oplus opad) \mid\mid H\big((K' \oplus ipad) \mid\mid m\big)\Big)$$

CTR Mode

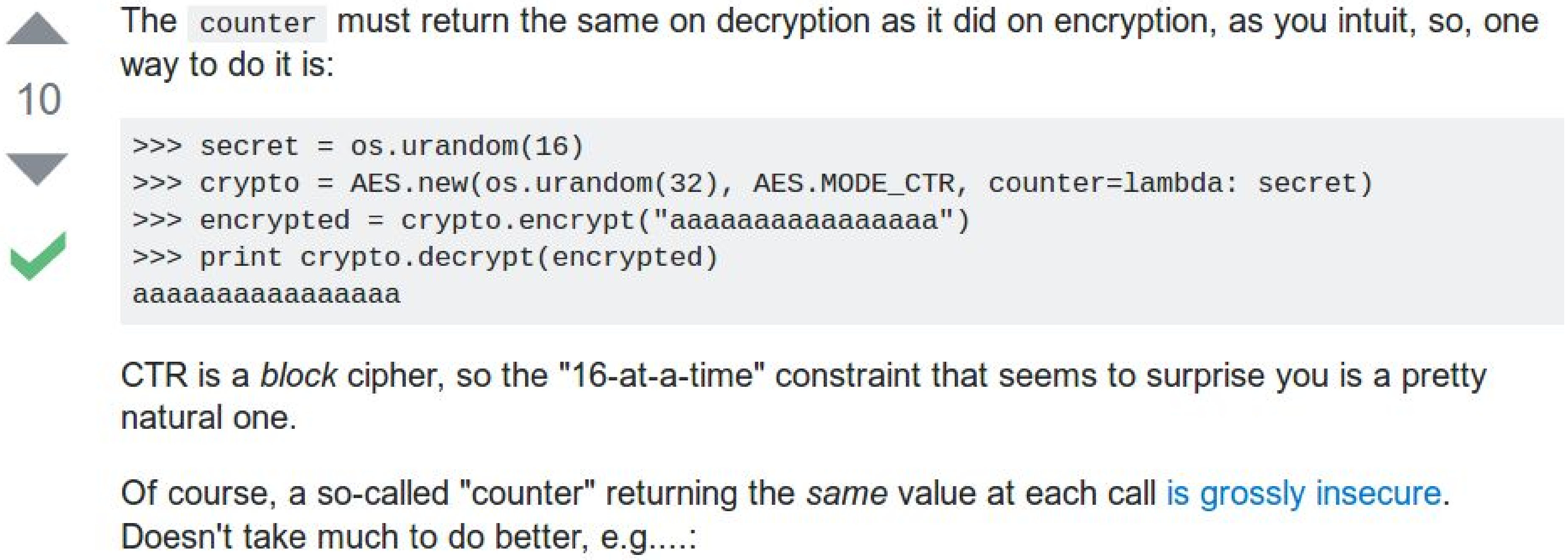
Counter Mode



CTR mode attack: ?

https://var.tailcall.net/ctr

CTR mode fails



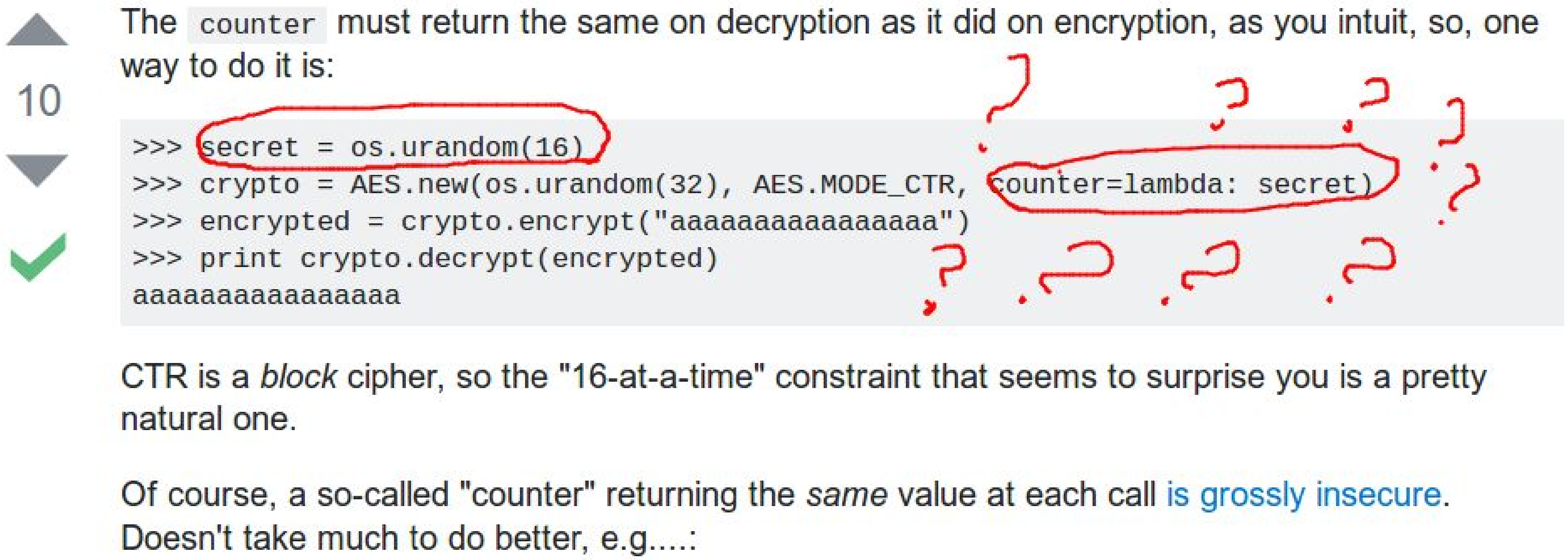
The `counter` must return the same on decryption as it did on encryption, as you intuit, so, one way to do it is:

```
>>> secret = os.urandom(16)
>>> crypto = AES.new(os.urandom(32), AES.MODE_CTR, counter=lambda: secret)
>>> encrypted = crypto.encrypt("aaaaaaaaaaaaaaaa")
>>> print crypto.decrypt(encrypted)
aaaaaaaaaaaaaaaa
```

CTR is a *block* cipher, so the "16-at-a-time" constraint that seems to surprise you is a pretty natural one.

Of course, a so-called "counter" returning the *same* value at each call is grossly insecure. Doesn't take much to do better, e.g....:

CTR mode fails



The `counter` must return the same on decryption as it did on encryption, as you intuit, so, one way to do it is:

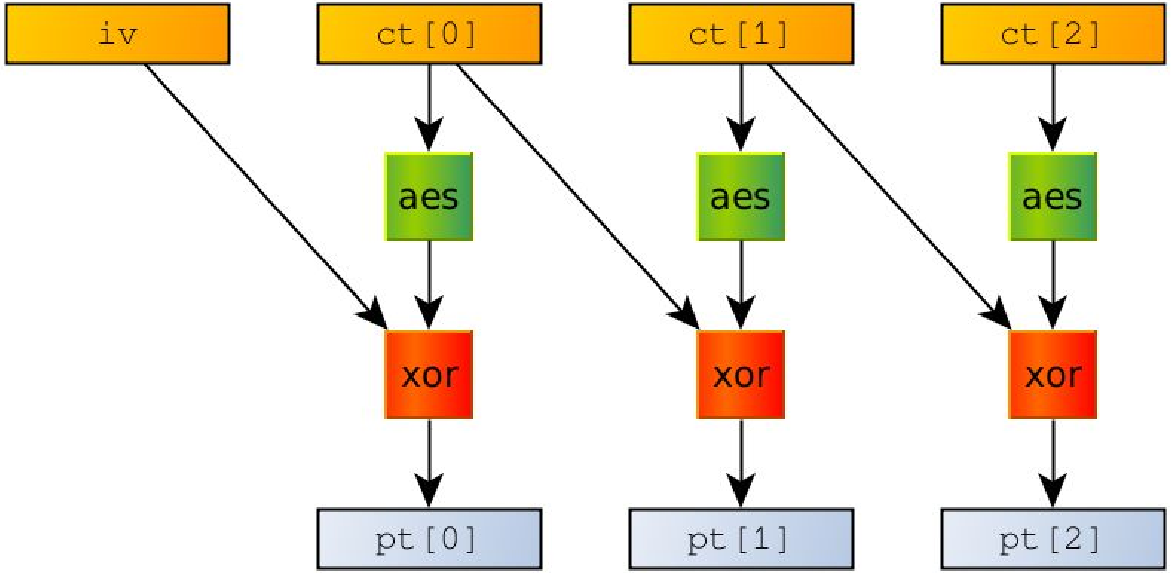
```
>>> secret = os.urandom(16)
>>> crypto = AES.new(os.urandom(32), AES.MODE_CTR, counter=lambda: secret)
>>> encrypted = crypto.encrypt("aaaaaaaaaaaaaaaa")
>>> print crypto.decrypt(encrypted)
aaaaaaaaaaaaaaaa
```

CTR is a *block* cipher, so the "16-at-a-time" constraint that seems to surprise you is a pretty natural one.

Of course, a so-called "counter" returning the *same* value at each call is grossly insecure. Doesn't take much to do better, e.g....:

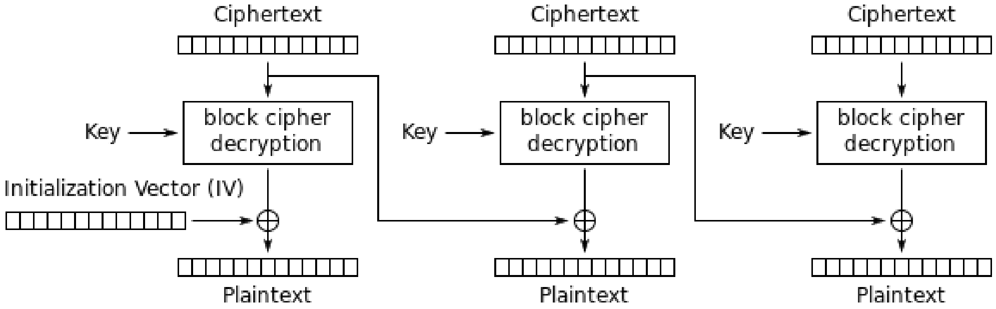
CBC Mode

Cipher Block Chaining



CBC mode attacks (byte flipping)

- Encryption is not authentication... again



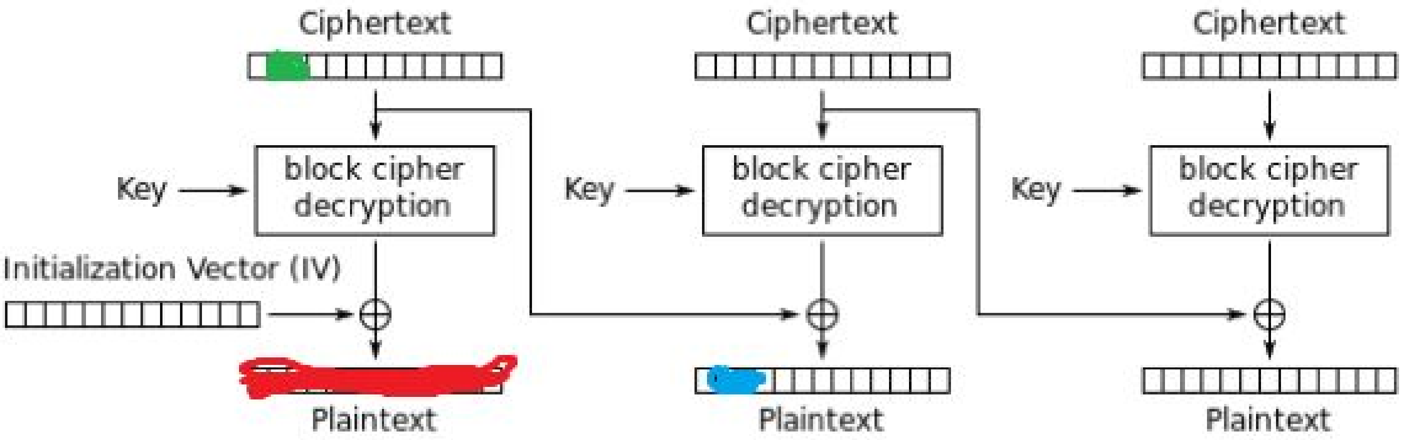
Cipher Block Chaining (CBC) mode decryption

- What if we can tamper with ciphertext?
  - What can we do with it?

CBC mode attack: ?

https://var.tailcall.net/cbc

CBC Mode Attack: Byte Flipping



Cipher Block Chaining (CBC) mode decryption

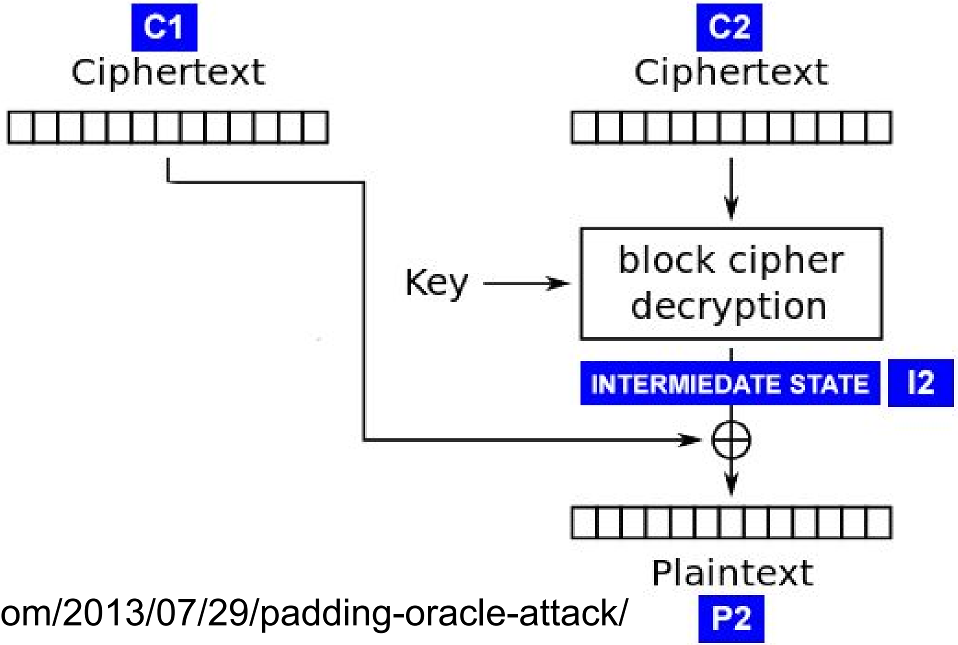
```
{'username': 'alamakota12345', 'anything': 'true'}
{'username': 'f(3&3€Nf#;c]!ó', 'isadmin':  'true'}
```

CBC Mode Attack: ?

```
def process_message(ciphertext):
    plaintext = decrypt_message(ciphertext)
    if plaintext == 'admin':
        return 'you are an admin'
    else:
        return 'you\'re not an admin'

def decrypt_message(ciphertext):
    if not padding_ok(ciphertext):
        raise new Exception('Invalid padding')
    return aes_decrypt(ciphertext)
```

Is something wrong with this code?

CBC Mode Attack: Padding oracle



http://robertheaton.com/2013/07/29/padding-oracle-attack/

I2 = C1 ^ P2
P2 = C1 ^ I2

P2[15] == 1?
I2[15] = ? C1[15] = ?

P2[14] == P2[15] == 1?
I2[14] = ? C1[14] = ?

Block ciphers: crypto building blocks

- Block ciphers => stream ciphers (CTR, OFB)
- Block ciphers => cryptographic hash function (1WCF)
- Block ciphers => CSPRNGs
- Block ciphers => PRP
- Block ciphers => MAC
- Block ciphers => AE (CCM, GCM, OCM...)

Block cipher design

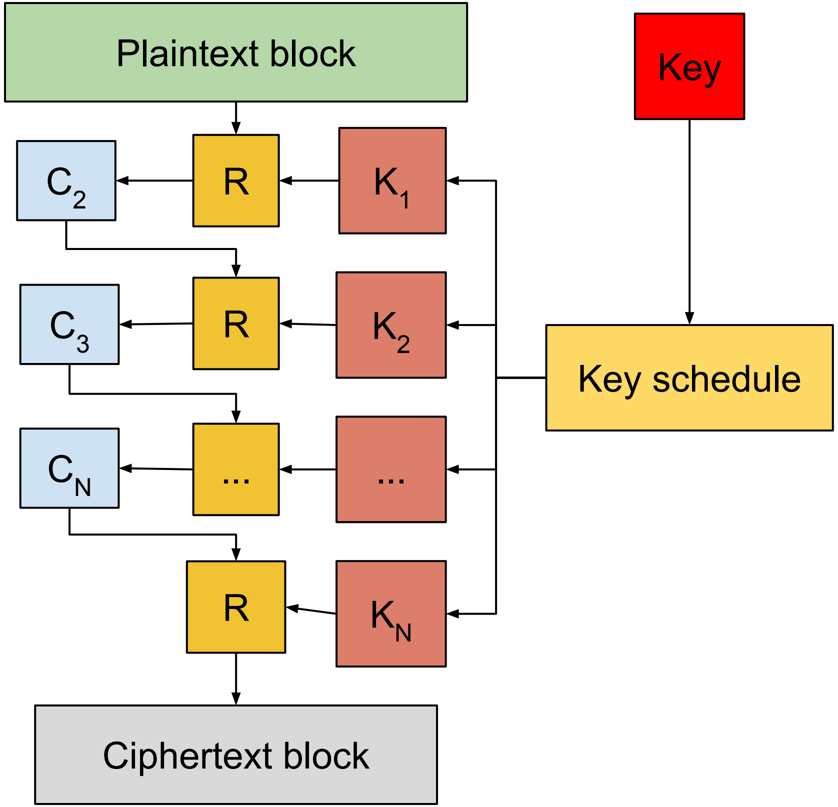
- Iterated block ciphers
- Feistel ciphers
- Substitution-permutation ciphers

Attacks

- Slide attack
- Square attack

Iterated block ciphers

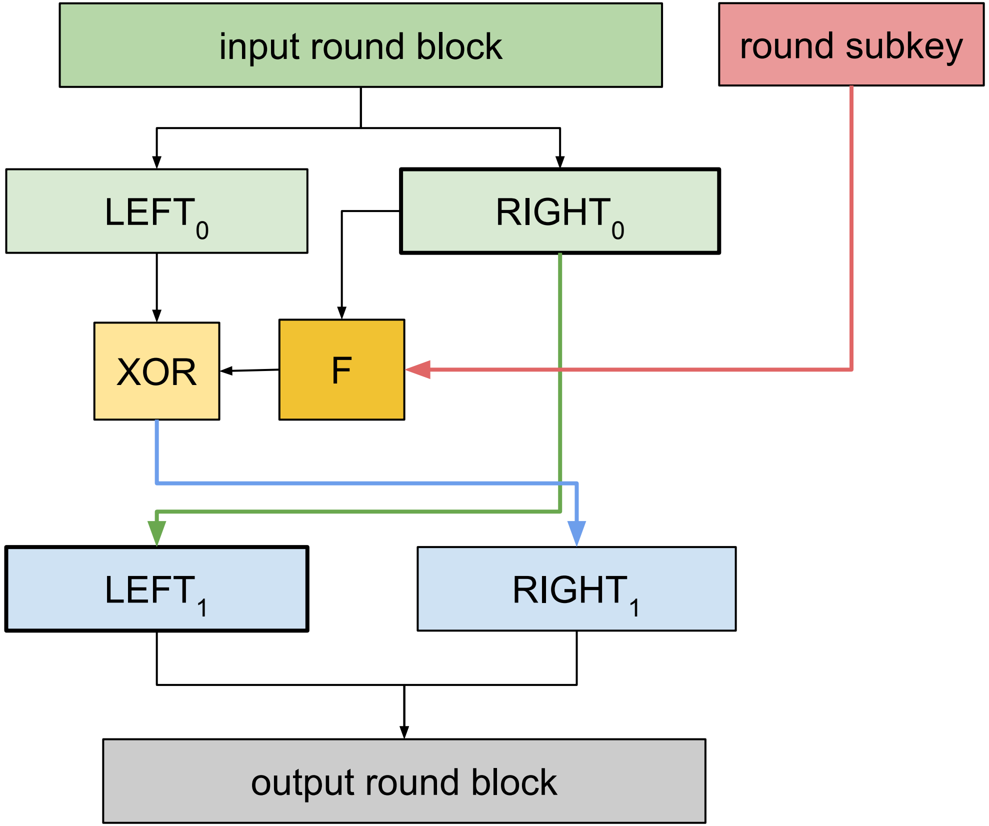
- Invertible round function: R
- Key schedule: K -> K_1,K_2,...,K_N
- C_{i+1}= R(K_i, C_i)
- P=C_1, C = C_{N+1}



Feistel Ciphers

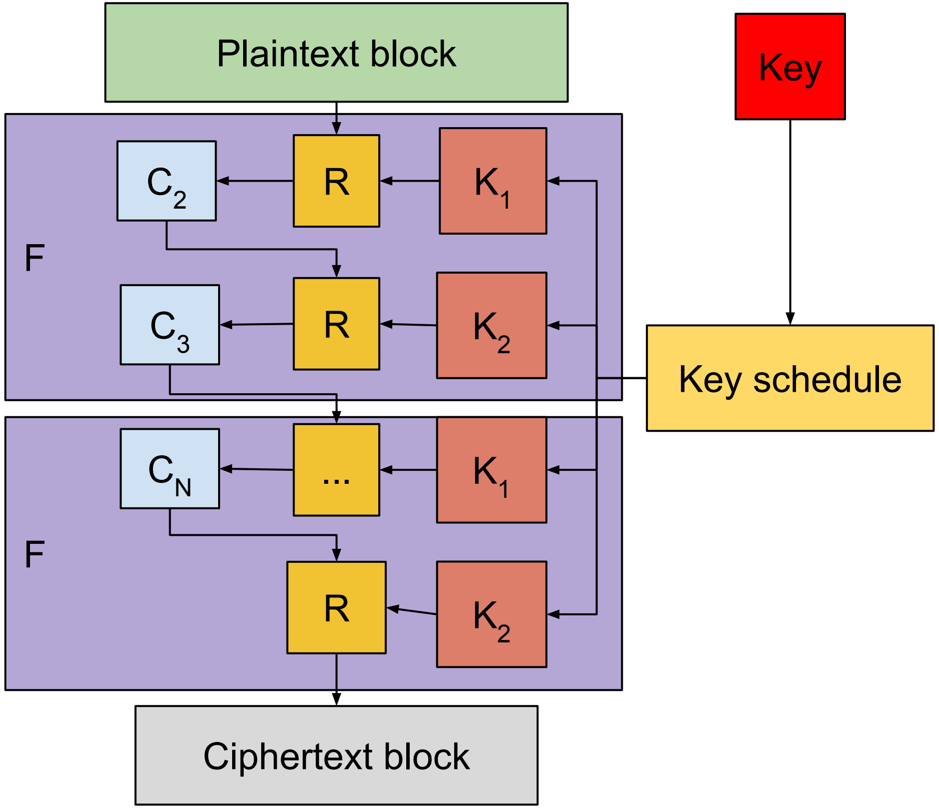
- Core function: F (not needed to be invertible)

- $LEFT_1 = RIGHT_0$

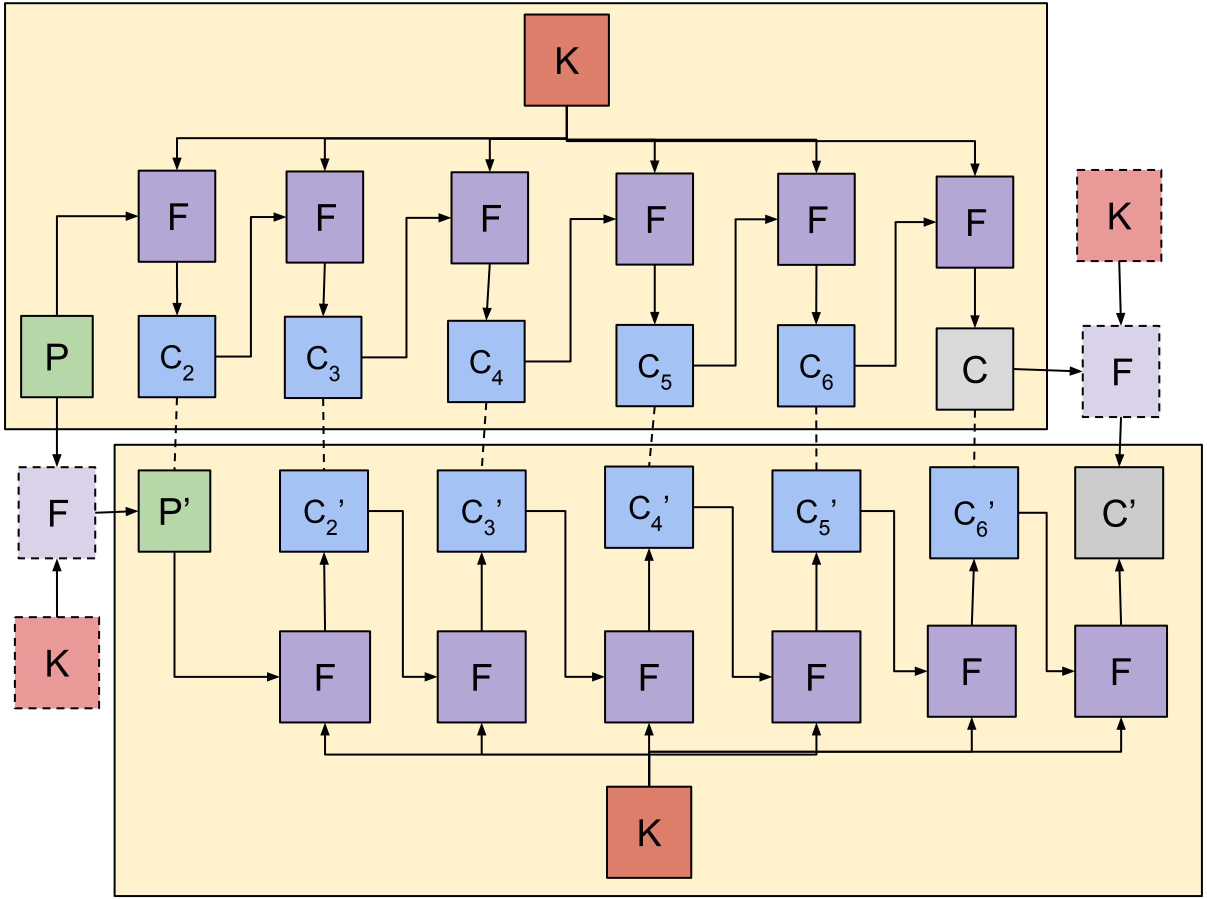


Slide attack

- Key schedule: periodic key
- Periodic part (F) vulnerable to known-plaintext attack
- N bit block: $2^{N/2}$ plaintext - ciphertext pairs

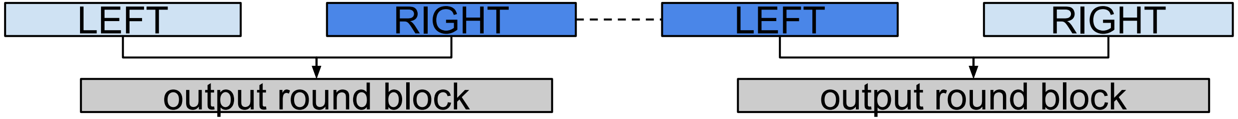


- (P, C),
  (P’,C’)
- P’ = F(K, P)
- C’ = F(K, P’)
- Time: 2^N !

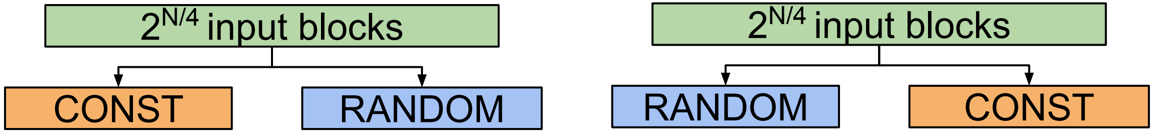


Slide attack on Feistel cipher

- Pair identification: RIGHT(C) = LEFT(C’)



- Chosen plaintext: $2^{N/4}$: $P_i = b_i|a$, $2^{N/4}$: $P_i’= a|b_i$



Bibliography

Joan Daemen and Vincent Rijmen, "AES Proposal: Rijndael"

Alex Biryukov and David Wagner, "Advanced Slide Attacks"

Bruce Schneier, "Applied Cryptography"