II4031 Kriptografi dan Koding

Stream Cipher

Oleh: Rinaldi Munir

Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

Kriptografi Modern

- Kriptografi modern adalah era kriptografi setelah penemuan komputer digital.
- Perkembangan teknologi komputer digital membuat ilmu kriptografi berkembang dengan pesat.
- Komputer digital merepresentasikan data dan informasi dalam biner (0/1).
- Algoritma kriptografi modern beroperasi dalam mode bit atau *byte* (bandingkan dengan algoritma kriptografi klasik beroperasi dalam mode karakter)

  → kunci, plainteks, cipherteks, diproses dalam rangkaian bit/byte

  → operasi **xor** paling banyak digunakan di dalam algoritmanya

- Meskipun disebut kriptografi modern, namun *cipher* modern tetap menggunakan dua teknik dasar dasar di dalam kriptografi klasik: **teknik substitusi** dan **teknik transposisi**,

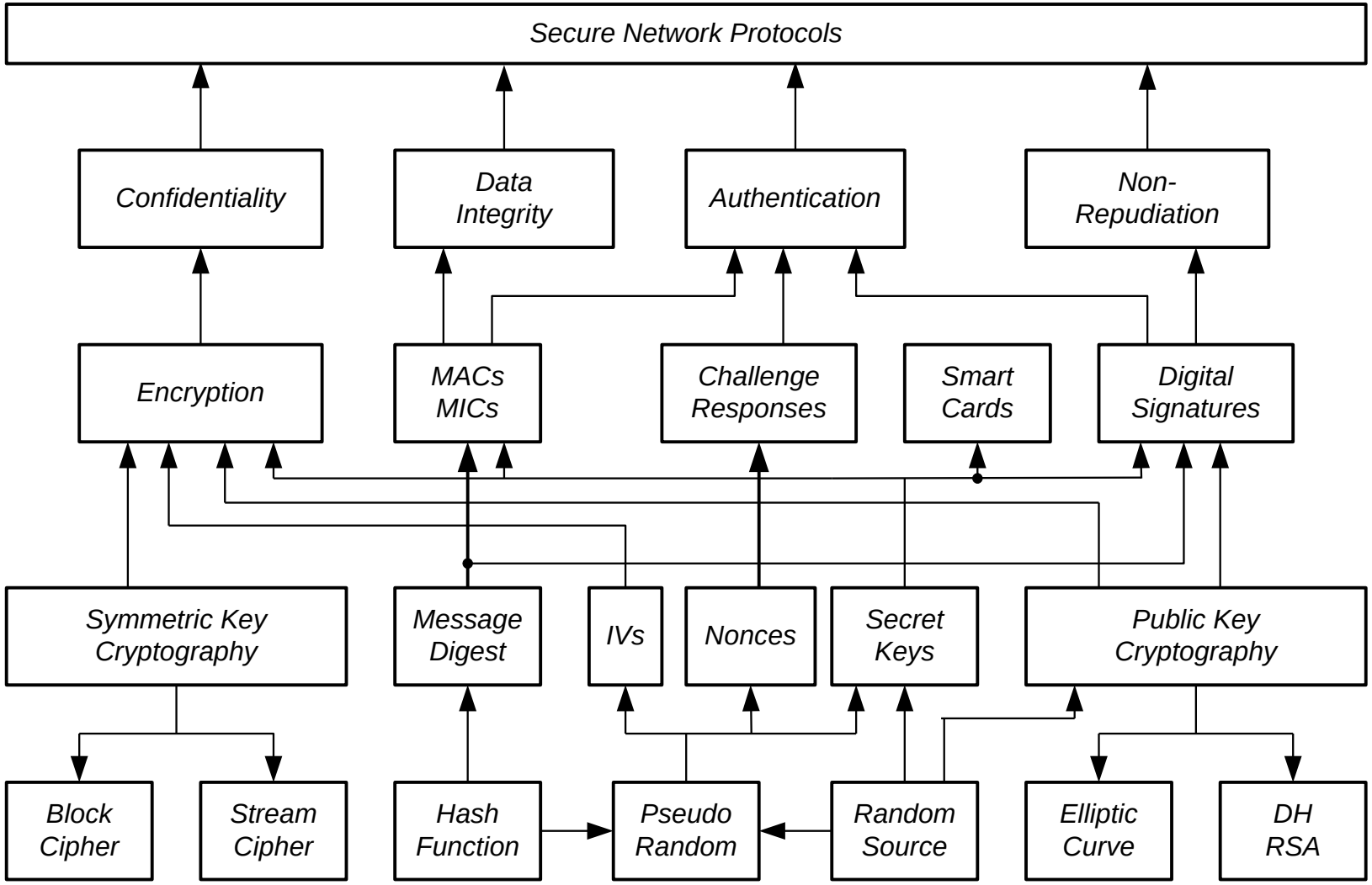
- tetapi operasinya dibuat lebih kompleks, tidak sesederhana cipher klasik.

  Tujuannya: agar *cipher* modern lebih sulit dikriptanalisis

- Selain kedua teknik dasar tersebut, juga digunakan teknik lain seperti rotasi, kompresi, ekspansi, penjumlahan modulo, dan lain-lain.

- Kriptografi modern melahirkan konsep-konsep baru seperti algoritma kriptografi kunci-publik, fungsi *hash*, protokol kriptografi, tanda-tangan digital, pembangkit bilangan acak, skema pembagian kunci, dsb.

Diagram Blok Kriptografi Modern



Bit, Byte, dan Kode Heksadesimal

- Pesan di dalam *cipher* modern dienkripsi bit-per-bit atau byte-per-byte, atau dalam kelompok bit (byte).

  1 byte = 8 bit

- Pada beberapa algoritma kriptografi, pesan direpresentasikan dalam kode heksadesimal (Hex).

  1 kode hex = 4 bit

0000 = 0	0001 = 1	0010 = 2	0011 = 3
0100 = 4	0101 = 5	0110 = 6	0111 = 7
1000 = 8	1001 = 9	1010 = A	1011 = B
1100 = C	1101 = D	1110 = E	1111 = F

- Contoh: Pesan 100111010110 dalam kode Hex dengan cara membagi pesan menjadi blok 4-bit:

  1001 1101 0110 = 9D6

Operasi *XOR*

- Di dalam *cipher* alir maupun cipher blok, operasi XOR adalah operasi yang paling sering digunakan
- Notasi: $\oplus$
- Operasi:

$0 \oplus 0 = 0$ $\quad\quad$ $0 \oplus 1 = 1$

$1 \oplus 0 = 1$ $\quad\quad$ $1 \oplus 1 = 0$

- Operasi XOR = penjumlahan dalam modulus 2:

$0 \oplus 0 = 0 \Leftrightarrow 0 + 0 \pmod 2 = 0$

$0 \oplus 1 = 1 \Leftrightarrow 0 + 1 \pmod 2 = 1$

$1 \oplus 0 = 1 \Leftrightarrow 1 + 0 \pmod 2 = 1$

$1 \oplus 1 = 0 \Leftrightarrow 1 + 1 \pmod 2 = 0$

- Sifat-sifat operasi XOR:

  (i) $a \oplus a = 0$

  (ii) $a \oplus b = b \oplus a$

  (iii) $a \oplus (b \oplus c) = (a \oplus b) \oplus c$

Contoh:

(i) $1 \oplus 1 = 0$

(ii) $1 \oplus 0 = 0 \oplus 1 = 1$

(iii) $1 \oplus (0 \oplus 1) = (1 \oplus 0) \oplus 1 = 0$

Operasi XOR *Bitwise*

- Jika dua rangkaian dioperasikan dengan *XOR*, maka operasinya dilakukan dengan meng-*XOR*-kan setiap bit yang berkoresponden dari kedua rangkaian bit tersebut.

  Contoh: 10011 ⊕ 11001 = 01010
  yang dalam hal ini, hasilnya diperoleh sebagai berikut:

$$
\begin{array}{ccccccccccc}
1 & & 0 & & 0 & & 1 & & 1 & & \\
1 & & 1 & & 0 & & 0 & & 1 & \oplus & \\
\hline
1 \oplus 1 & & 0 \oplus 1 & & 0 \oplus 0 & & 1 \oplus 0 & & 1 \oplus 1 & & \\
0 & & 1 & & 0 & & 1 & & 0 & &
\end{array}
$$

Cipher Sederhana dengan operasi XOR

- Sama seperti *Vigenere Cipher*, tetapi dalam mode bit
- Setiap bit plainteks di-*XOR*-kan dengan setiap bit kunci.

Enkripsi: $C = P \oplus K$

Dekripsi: $P = C \oplus K$

Contoh:	plainteks	01100101		(karakter ‘e’)
	kunci	00110101	⊕	(karakter ‘5’)
	cipherteks	01010000		(karakter ‘P’)
	kunci	00110101	⊕	(karakter ‘5’)
	plainteks	01100101		(karakter ‘e’)

- Jika panjang bit-bit kunci lebih pendek daripada panjang bit-bit pesan, maka bit-bit kunci diulang penggunaannya secara periodik (seperti halnya pada Vigenere Cipher)

- Contoh:

  Plainteks : 10010010101110101010001110001

  Kunci : 11011011011011011011011011011

  Cipherteks: 01001001110101110001010101010

- Jika plainteks dan kunci dalam bentuk karakter (1 karakter = 1 byte), maka di dalam komputer karakter dikonversi ke dalam biner. Proses XOR dilakukan secara bitwise XOR

Contoh:

Planteks: `BOHONG`

Kunci: `ABCABC`

Dalam bentuk biner:

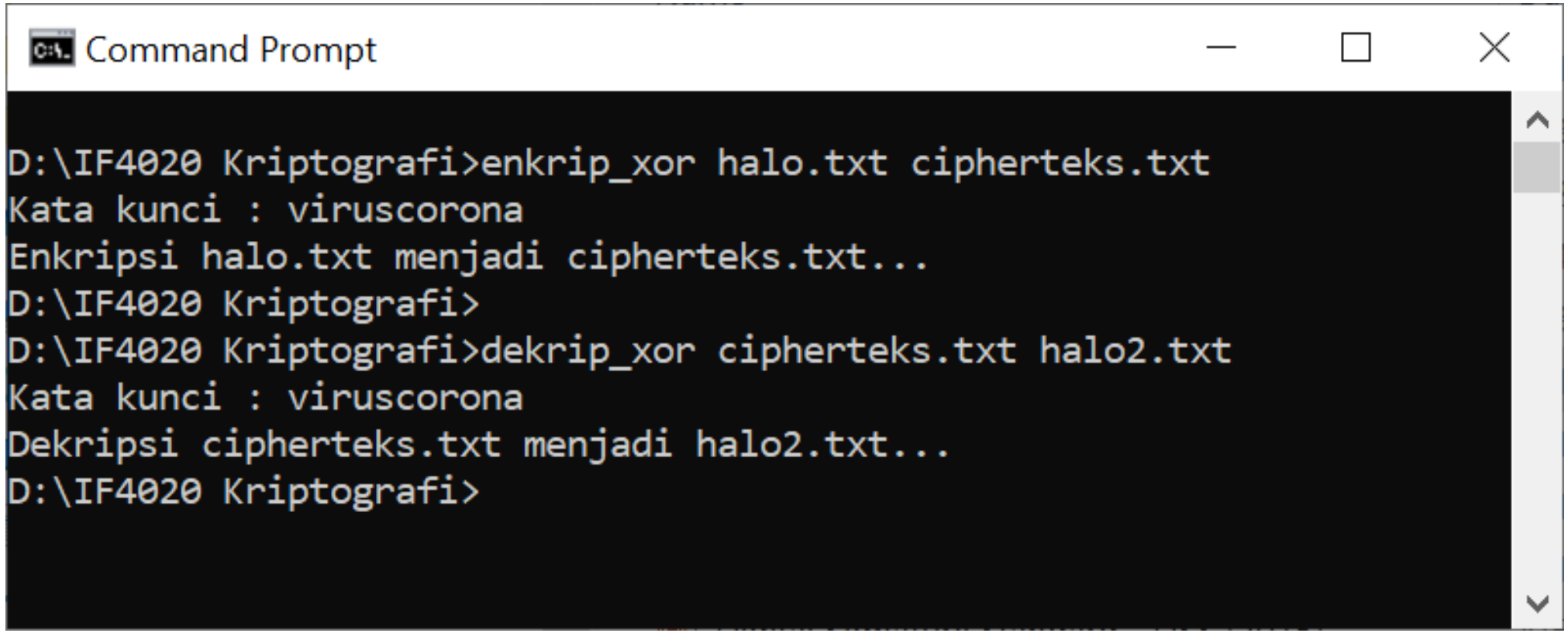
Plainteks	`01000010 01001111 01001000 01001111 01001110 01000111`
Kunci:	`01000001 01000010 01000011 01000001 01000010 01000010`
Cipherteks:	`00000011 00001101 00001011 00001110 00001100 00000101`

```
// Enkripsi sembarang berkas dengan
// algoritma XOR sederhana.
#include <iostream>
#include <string.h>
#include <fstream>
#include <stdlib.h>
using namespace std;

main(int argc, char *argv[])
{
 FILE *Fin, *Fout;
 char p, c;
 string K;
 int i;

 Fin = fopen(argv[1], "rb");
 if (Fin == NULL) {
   cout << "Berkas " << argv[1] <<"
tidak ada" << endl;
   exit(0);
 }

 Fout = fopen(argv[2], "wb");

 cout << "Kata kunci : "; cin >> K;
 cout <<"Enkripsi " << argv[1] << "
menjadi " << argv[2] << "...";
 i = 0;
 while (!feof(Fin)) {
   p = getc(Fin);
   c = p ^ K[i];  // operasi XOR
   putc(c, Fout);
   i = (i + 1) % K.length();
 }
 fclose(Fin);
 fclose(Fout);
}
```

(a) enkrip_xor.cpp

```
// Dekripsi sembarang berkas dengan
// algoritma XOR sederhana.
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <fstream>
using namespace std;

main(int argc, char *argv[])
{
 FILE *Fin, *Fout;
 char p, c;
 string K;
 int i;

 Fin = fopen(argv[1], "rb");
 if (Fin == NULL){
   cout << "Berkas " << argv[1] <<"
tidak ada" << endl;
   exit(0);
 }

 Fout = fopen(argv[2], "wb");

 cout << "Kata kunci : "; cin >> K;
 cout <<"Dekripsi " << argv[1] << "
menjadi " << argv[2] << "...";
 i = 0;
 while (!feof(Fin)) {
   c = getc(Fin);
   p = c ^ K[i];  // operasi XOR
   putc(p, Fout);
   i = (i + 1) % K.length();
 }
 fclose(Fin);
 fclose(Fout);
}
```

(b) dekrip_xor.cpp



- Cipher sederhana dengan XOR tidak aman, karena mudah dikriptanalisis dengan metode yang sama seperti metode Kasiski

Hasil *running* program cipher XOR sederhana:

Plainteks	Cipherteks *)
Pada wisuda sarjana baru, ternyata ada seorang wisudawan yang paling muda. Umurnya baru 21 tahun. Ini berarti dia masuk ITB pada umur 17 tahun. Zaman sekarang banyak sarjana masih berusia muda belia.	7 S S H IS A o S G H H KS= b EAYA FA. E S A G(:'y N - GPYE @ES2 E H b A H A S K

*) Beberapa karakter ASCII *unprintable*, sehingga tidak dapat dicetak

Kategori *cipher* berbasis bit

1. *Cipher* Alir (*Stream Cipher*)

   - beroperasi pada bit tunggal atau *byte* tunggal

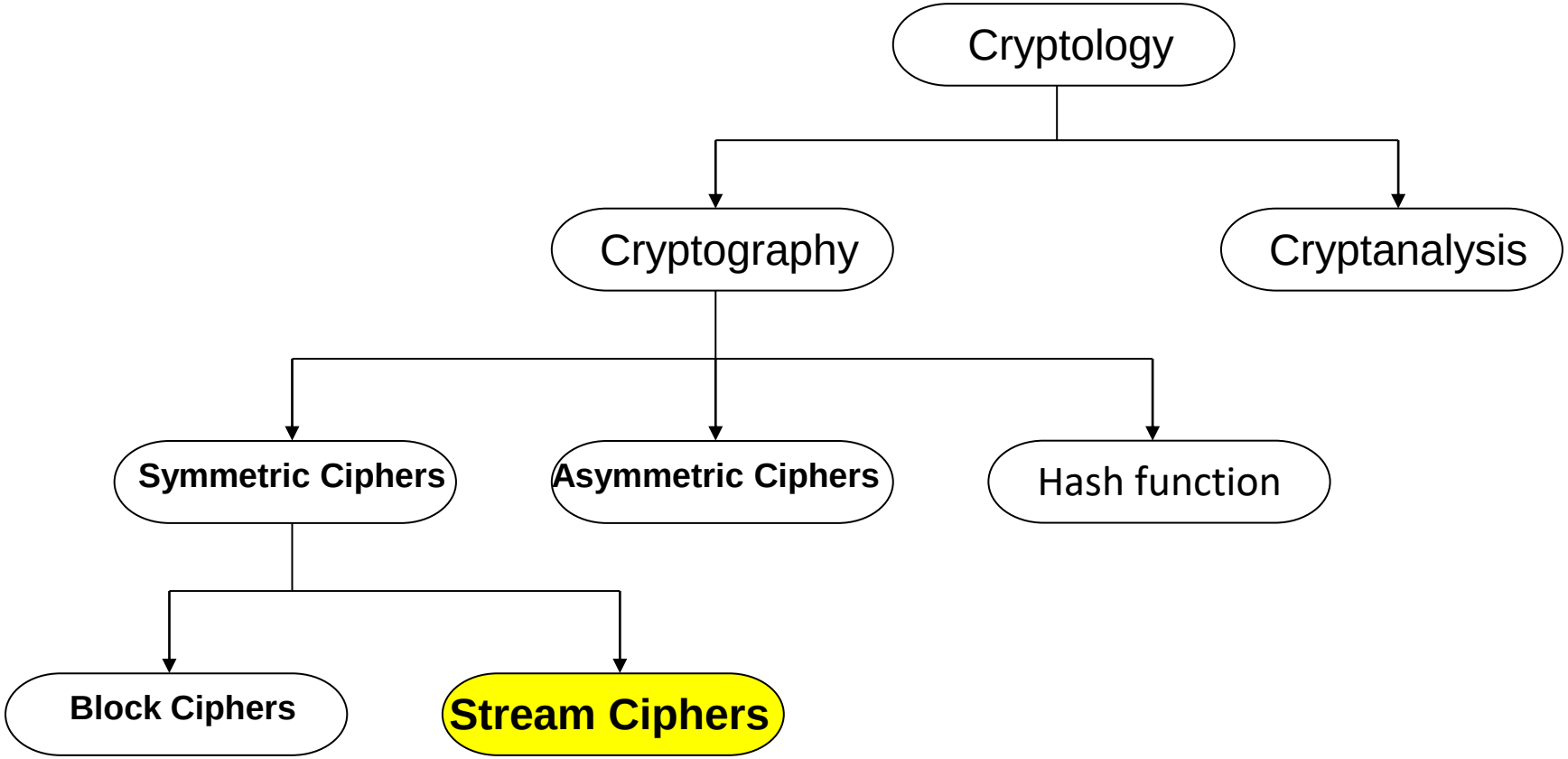
   - enkripsi/dekripsi pesan secara bit per bit atau *byte* per *byte*

2. *Cipher* Blok (*Block Cipher*)

   - beroperasi pada blok bit atau blok *byte*

     (contoh: 64-bit/blok = 8 karakter/blok)

   - enkripsi/dekripsi pesan secara blok per blok bit atau blok per blok byte


Cryptology
Cryptography
Cryptanalysis
Symmetric Ciphers
Asymmetric Ciphers
Hash function
Block Ciphers
Stream Ciphers

Cipher Alir (*Stream Cipher*)

- Mengenkripsi plainteks menjadi chiperteks setiap bit per bit dengan bit-bit kunci atau atau *byte* per *byte* (1 *byte* setiap kali transformasi).

- Diperkenalkan oleh Vernam melalui algoritmanya, **Vernam *Cipher***.

- Vernam *cipher* diadopsi dari *one-time pad cipher*, yang dalam hal ini huruf diganti dengan bit (0 atau 1).

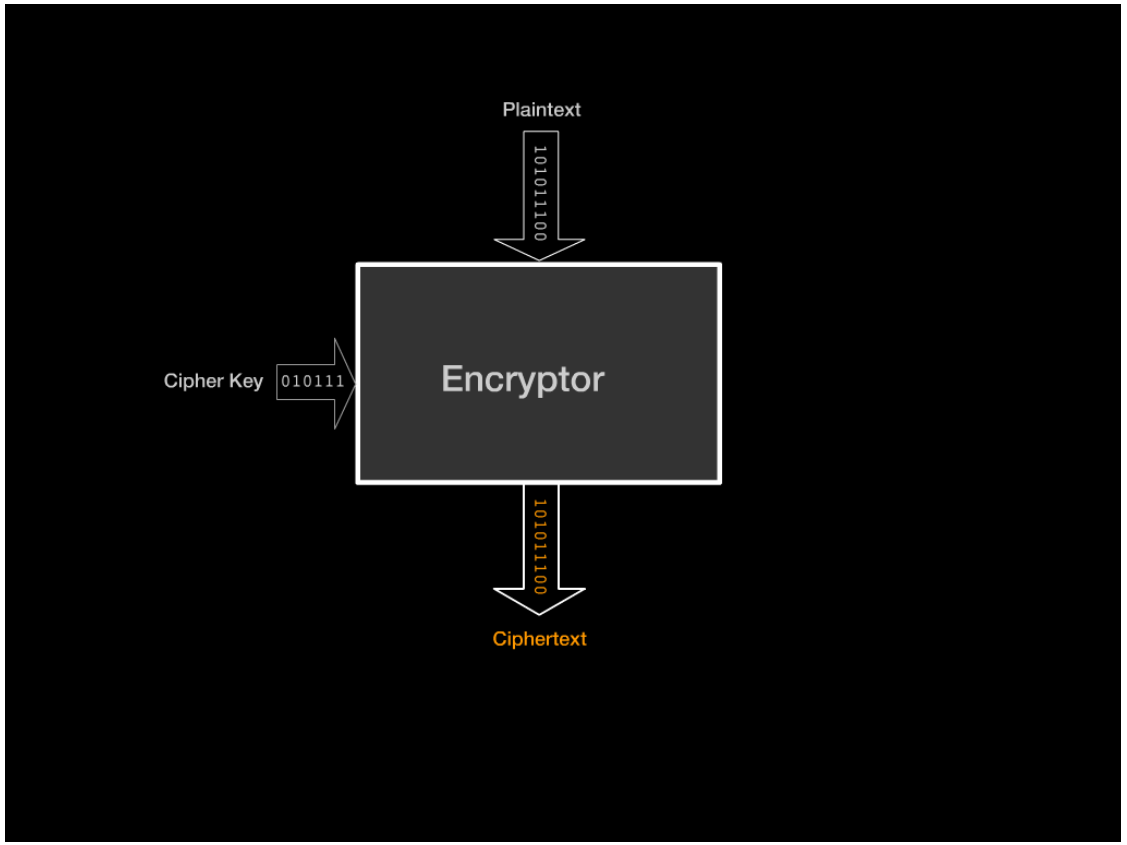
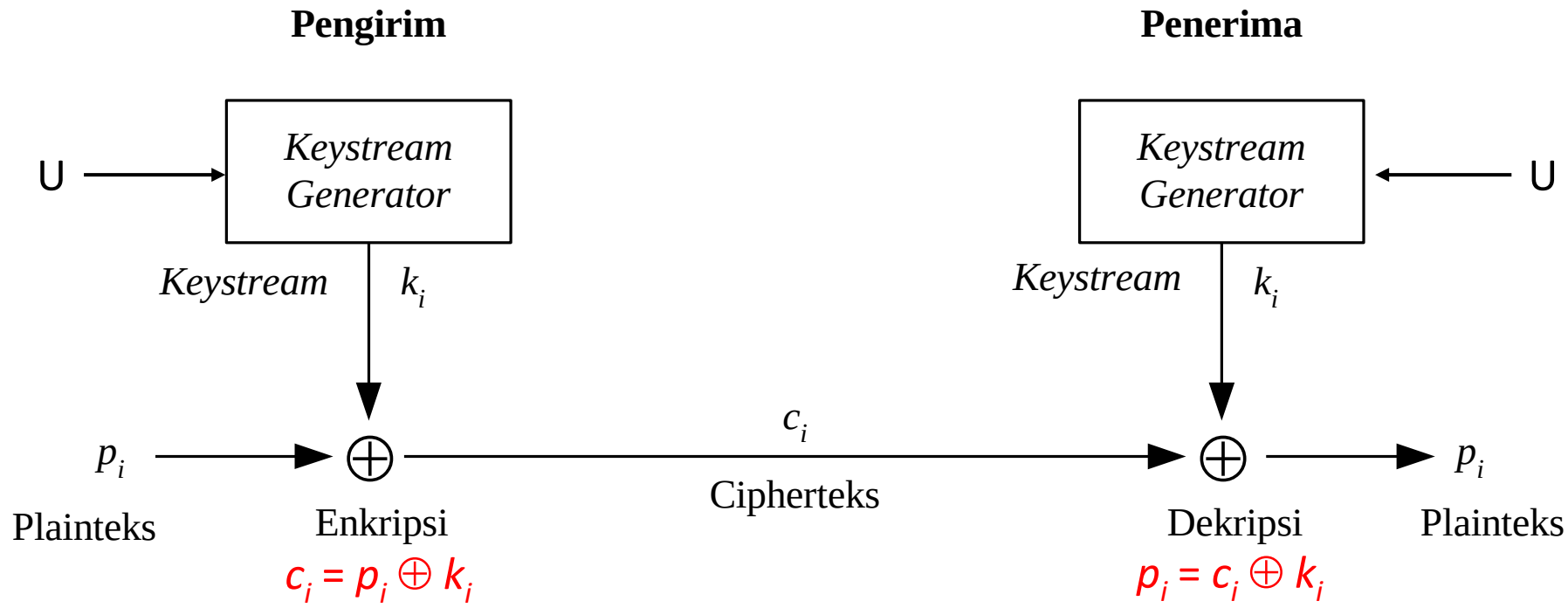


Diagram *cipher* alir



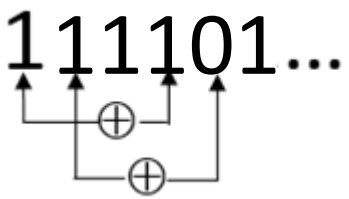
- Bit-bit aliran kunci untuk enkripsi/dekripsi disebut *keystream. Keystream* dibangkitkan oleh *keystream generator.*
- Enkripsi dan dekripsi dengan *cipher* alir komputasinya sederhana dan cepat
- Misalkan p_1, p_2, … adalah bit-bit plainteks, c_1, c_2, … adalah bit-bit plainteks, k_1, k_2, … adalah bit-bit kunci-alir
- **Enkripsi**: $c_i = p_i \oplus k_i$
- Dekripsi: $p_i = c_i \oplus k_i$

- Contoh: Misalkan *U* = 1111

  Contoh sebuah algoritma sederhana memperoleh *keystream* adalah sebagai berikut:

  XOR-kan bit ke-1 dengan bit ke-4 dari empat bit sebelumnya:

  1111010110010000

  dan akan berulang setiap 15 bit.

- Secara umum, jika panjang *U* adalah *n* bit, maka bit-bit kunci tidak akan berulang sampai $2^n - 1$ bit.

- Contoh:

Plainteks:	1100101010100110001	
Keystream:	1000110000101001101 ⊕	Enkripsi
Cipherteks:	0100011010001111100	
Keystream:	1000110000101001101 ⊕	Dekripsi
Plainteks:	1100101010100110001	

Aplikasi *Cipher* Alir

- *Cipher* alir cocok untuk mengenkripsi aliran data yang terus menerus melalui saluran komunikasi, misalnya:
  1. Mengenkripsi data pada saluran yang menghubungkan antara dua buah komputer.
  2. Mengenkripsi suara pada jaringan telepon *mobile* GSM.

- Alasan: jika bit cipherteks yang diterima mengandung kesalahan, maka hal ini hanya menghasilkan satu bit kesalahan pada waktu dekripsi, karena tiap bit plainteks ditentukan hanya oleh satu bit cipherteks.

- Selain itu, alasannya karena *cipher* alir itu komputasinya cepat (dibutuhkan untuk *real-time processing*)

Beberapa *cipher* alir dan tahun pembuatan

- RC4 (1987)
- A5 (1989)
- Rabbit (2003)
- Trivium (2004)
- SEAL (1997)
- Salsa20 (2004)
- Scream (2002)
- SOBER-128 (2003)
- WAKE (2003)
- Turing (2000-2003)
- Scream (2002)
- VEST (2005)
- SNOW (2003)
- CryptMT (2005)
- FISH (1993)
- Grain (2004)
- Isaac (1995)
- MICKEY (2004)

Stream cipher	Creation date	Speed (cycles per byte)	(bits)			Attack	
			Effective key-length	Initialization vector	Internal state	Best known	Computational complexity
A5/1	1989	?	54 or 64 (in 2G)	22 (in 2G)	64	Active KPA OR KPA time–memory tradeoff	~ 2 seconds OR $2^{39.91}$
A5/2	1989	?	54	114	64?	Active	4.6 milliseconds
Achterbahn-128/80	2006	1 (hardware)	80/128	80/128	297/351	Brute force for frame lengths $L \leq 2^{44}$. Correlation attack for $L \geq 2^{48}$.	2^{80} resp. 2^{128} for $L \leq 2^{44}$.
CryptMT	2005	?	Variable	up to 19968	19968	— (2008)	— (2008)
Crypto-1	Pre-1994	?	48	16	48	Active KPA (2008)	40 ms OR 2^{48} (2008)[2]
E0 (cipher)	Pre-1999	?	Variable (usually 128)	4	132	KPA (2005)	2^{38} (2005)[3]
FISH	1993	?	Variable	?	?	Known-plaintext attack	2^{11}
Grain	Pre-2004	?	80	64	160	Key derivation	2^{43}
HC-256	Pre-2004	4 (W_{P4})	256	256	65536	?	?
ISAAC	1996	2.375 ($W_{64\text{-bit}}$) – 4.6875 ($W_{32\text{-bit}}$)	8–8288 (usually 40–256)	—	8288	(2006) First-round weak-internal-state-derivation	4.67×10^{1240} (2001)
MICKEY	Pre-2004	?	80	Variable (0 to 80)	200	Differential Fault Attack (2013)	$2^{32.5}$ (2013)[4]
MUGI	1998–2002	?	128	128	1216	— (2002)	~ 2^{82}

PANAMA	1998	2	256	128?	1216?	Hash collisions (2001)	2^{82}
Phelix	Pre-2004	up to 8 (W_{x86})	256 + a 128-bit nonce	128?	?	Differential (2006)	2^{37}
Pike	1994	?	Variable	?	?	— (2004)	— (2004)
Py	Pre-2004	2.6	8–2048? (usually 40–256?)	64	8320	Cryptanalytic theory (2006)	2^{75}
Rabbit	2003-Feb	3.7(W_{P3}) – 9.7(W_{ARM7})	128	64	512	— (2006)	— (2006)
RC4	1987	7 W_{P5}[5]	8–2048 (usually 40–256)	RC4 does not take an IV. If one desires an IV, it must be mixed into the key somehow.	2064	Shamir initial-bytes key-derivation OR KPA	2^{13} OR 2^{33}
Salsa20	Pre-2004	4.24 (W_{G4}) – 11.84 (W_{P4})	256	a 64-bit nonce + a 64-bit stream position	512	Probabilistic neutral bits method	2^{251} for 8 rounds (2007)
Scream	2002	4–5 (W_{soft})	128 + a 128-bit nonce	32?	64-bit round function	?	?
SEAL	1997	?	?	32?	?	?	?
SNOW	Pre-2003	?	128 or 256	32	?	?	?
SOBER-128	2003	?	up to 128	?	?	Message forge	2^{-6}
SOSEMANUK	Pre-2004	?	128	128	?	?	?
Trivium	Pre-2004	4 (W_{x86}) – 8 (W_{LG})	80	80	288	Brute force attack (2006)	2^{135}
Turing	2000–2003	5.5 (W_{x86})	?	160	?	?	?
VEST	2005	42 (W_{ASIC}) – 64 (W_{FPGA})	Variable (usually 80–256)	Variable (usually 80–256)	256–800	— (2006)	— (2006)
WAKE	1993	?	?	?	8192	CPA & CCA	Vulnerable

(Sumber: https://en.wikipedia.org/wiki/Stream_cipher

RC4

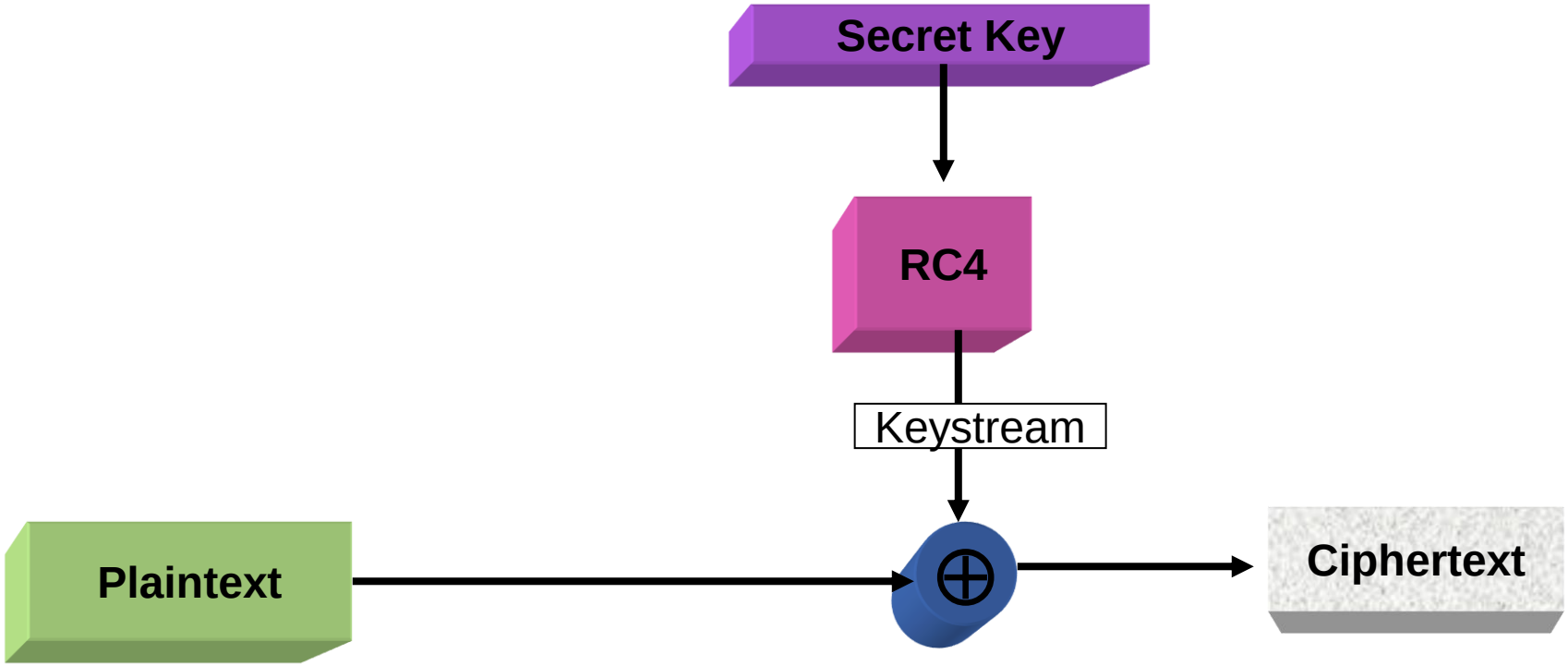
- *Cipher* alir yang paling populer
- Dibuat oleh Ronald Rivest (1987) dari Laboratorium *RSA*
- *RC* adalah singkatan dari *Ron's Code*. Versi lain mengatakan *Rivest Cipher* .
- Digunakan di dalam sistem keamanan seperti:

  - protokol *SSL* (*Secure Socket Layer*) dan TLS (*Transport Layer Security*)

  - Standard IEEE 802.11 *wireless* LAN: WEP (*Wired Equivalent Privacy*)

  - WPA (*Wi-fi Protocol Access*) untuk nirkabel

Diagram Blok RC4:



- Kunci rahasia (*secret key*) memiliki panjang maksimal 256 karakter (1 karakter = 1*byte*). Jika panjang kunci kurang dari 256 *byte* maka karakter-karakter kunci diulang secara periodik.
- RC4 menghasilkan luaran berupa kunci-alir (*keystream*) dengan panjang tak terbatas

- *RC4* membangkitkan kunci-alir (*keystream*) dalam satuan byte setiap kalinya, yang kemudian di-XOR-kan dengan byte plainteks

- Jadi, *RC4* memproses data dalam satuan *byte*, bukan dalam bit.

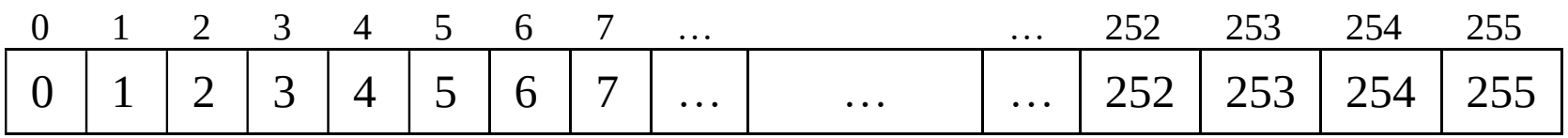
- Untuk membangkitkan kunci-alir, *cipher* menggunakan status internal yang terdiri dari:
  - Permutasi angka 0 sampai 255 di dalam larik S_0, S_1, …, S_{255}. Permutasi merupakan fungsi dari kunci rahasia *K* dengan panjang variabel.
  - Dua buah pencacah indeks, *i* dan *j*

- RC4 terdiri dari dua sub-proses:
  1. *Key-Scheduling Algorithm* (*KSA*)
  2. *Pseudo-random generation algorithm* (*PRGA*).

Key-Scheduling Algorithm (*KSA*)

1. Inisialisasi larik S: $S_0 = 0$, $S_1 = 1$, …, $S_{255} = 255$

```
for i ← 0 to 255 do
  S[i] ← i
end
```



3. Lakukan pengacakan (permutasi) nilai-nilai di dalam larik *S* berdasarkan kunci rahasia K:

```
j ← 0
for i ← 0 to 255 do
  j ← (j + S[i] + K[i  mod Length(K)]) mod 256
  swap(S[i], S[j]) {* Pertukarkan S[i] & S[j] *}
end
```

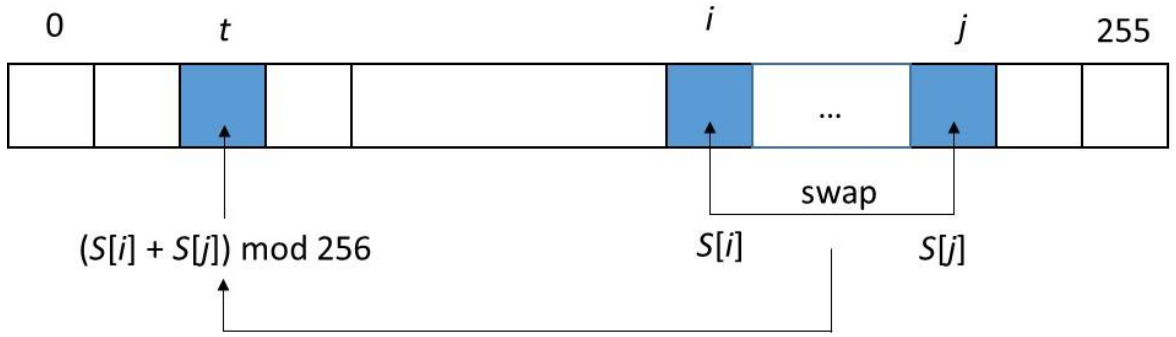
- Permutasi ini menyebabkan elemen-elemen di dalam larik *S* teracak.

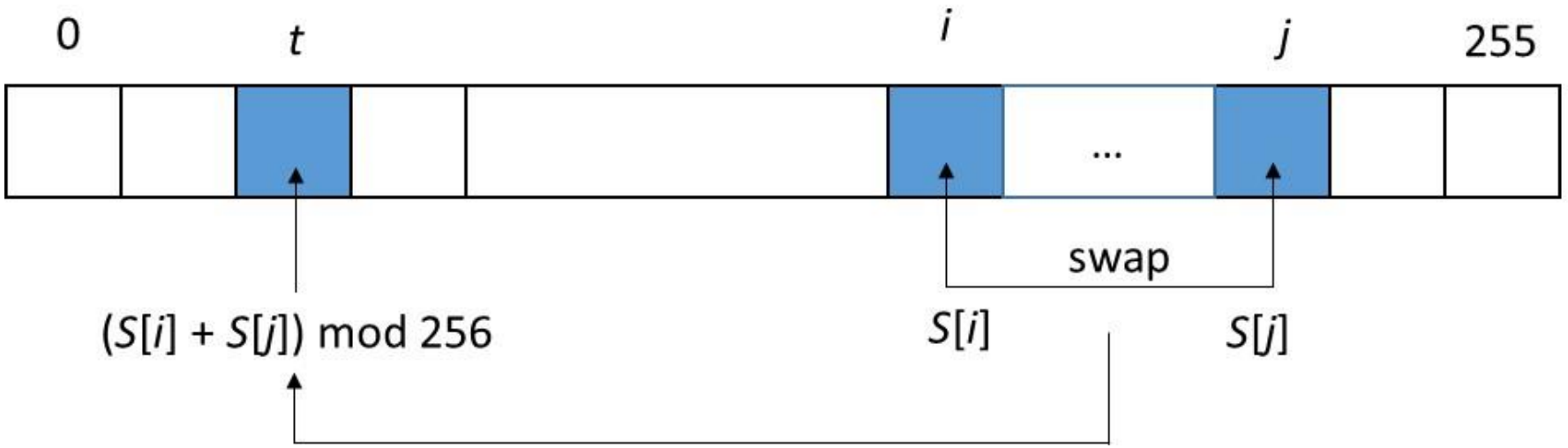
- `K[i mod Length(K)]` menyatakan karakter-karakter kunci diulang secara periodik jika panjangnya kurang dari 256

Pseudo-random generation algorithm (*PRGA*)

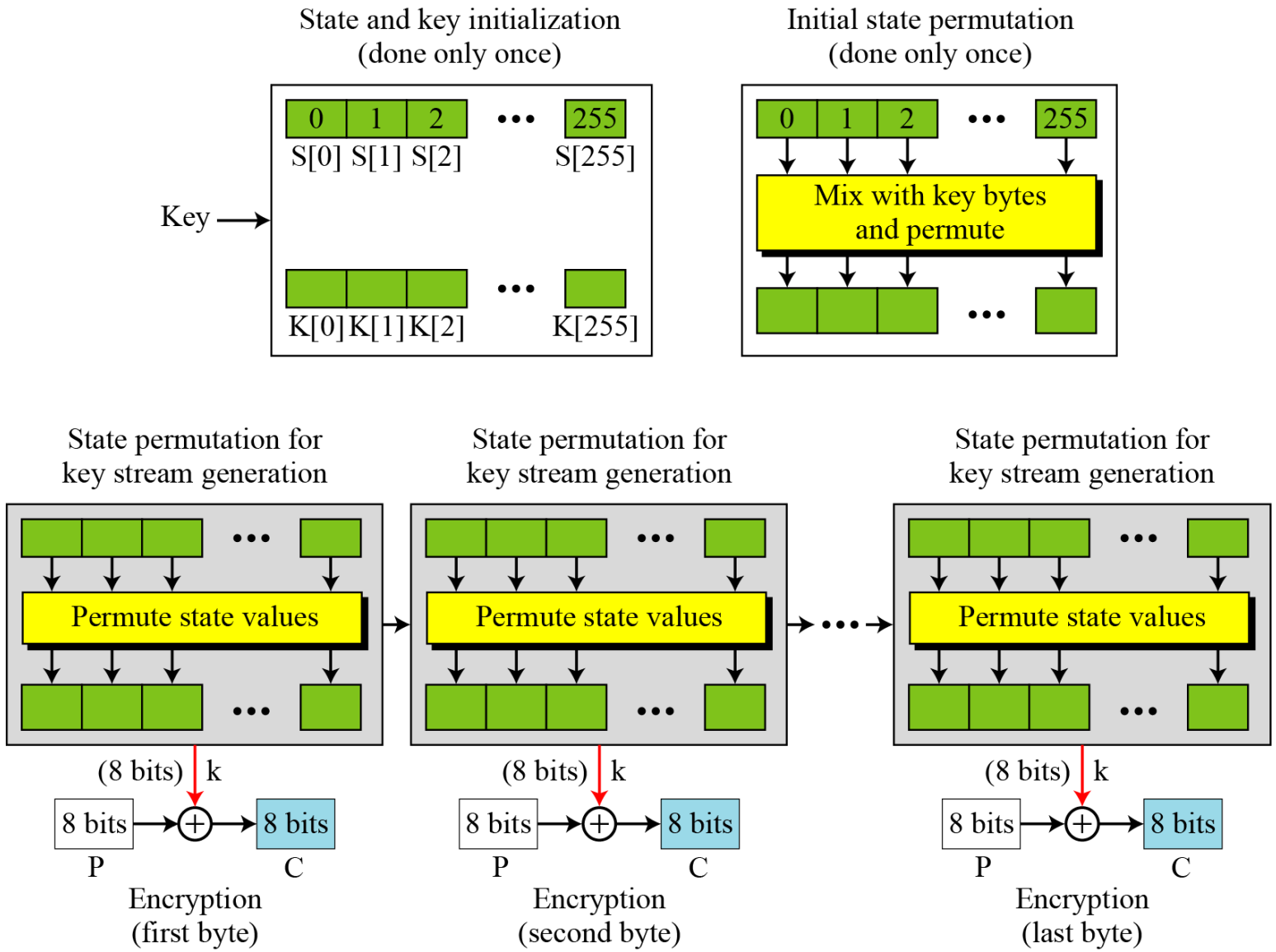
- *PRGA* membangkitkan kunci-alir (*keystream*) dengan cara mengambil nilai *S*[*i*] dan *S*[*j*], mempertukarkannya, lalu menjumlahkan keduanya dalam modulus 256.
- Kunci alir tersebut kemudian di-XOR-kan dengan sebuah karakter plainteks.

```
i ← 0
j ← 0
for idx ← 0 to Length(P) - 1 do
    i ← (i + 1) mod 256
    j ← (j + S[i]) mod 256
    swap(S[i], S[j])    {* Pertukarkan nilai S[i] dan S[j] *}
    t ← (S[i] + S[j]) mod 256
    u ← S[t]            (* keystream *)
    c ← u ⊕ P[idx]      { enkripsi }
end
```




0
t
i
j
255
...
swap
(S[i] + S[j]) mod 256
S[i]
S[j]

- Operasi RC4 secara keseluruhan:



Keamanan RC4

- RC4 memiliki kelemahan, yaitu *byte-byte* pada *keystream* awal pembangkitan memiliki korelasi yang tinggi dengan beberapa *byte* awal kunci

- Oleh karena itu direkomendasikan membuang 256 sampai 512 *byte keystream* awal
-
- RC4 adalah *cipher* alir, maka ia tidak kuat terhadap serangan seperti *flip-bit attack* maupun serangan-serangan *stream attack* lainnya.

- Saat ini keamanan RC4 sudah berhasil dipecahkan dalam hitungan jam atau hari.

- Pada bulan Februari 2015, RC4 dilarang penggunaannya di dalam *Transport Layer Security* (TLS) seperti disebutkan di dalam RFC 7465 (https://tools.ietf.org/html/rfc7465).

- Beberapa varian RC4 telah dibuat untuk mengatasi kelemahannya, yaitu Spritz, RC4A, VMPC, dan RC4+.

Kode Program RC4 (dalam Bahasa C++)

- Enkripsi

```
// Enkripsi sembarang berkas dengan algoritma RC4.
#include <iostream>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
using namespace std;

main(int argc, char *argv[])
{
 FILE *Fin, *Fout;
 char p, c, u;
 string K;
 int S[256];
 int i, j, t;
```

```
Fin = fopen(argv[1], "rb");
 if (Fin == NULL) {
   cout << "Berkas " << argv[1] <<" tidak ada" << endl;
   exit(0);
 }

 Fout = fopen(argv[2], "wb");

 cout << "Kata kunci : "; cin >> K;
 cout <<"Enkripsi " << argv[1] << " menjadi " << argv[2] << "...";

 for (i = 0; i<256; i++) {
  S[i] = i;
 }
```

```
j = 0;
for (i=0; i<256; i++) {
  j = (j + S[i] + K[i % K.length()]) % 256;
  swap(S[i], S[j]); // Pertukarkan nilai S[i] dan S[j]
}

i = 0;   j = 0;
count = 0;
while (!feof(Fin)) {
  p = fgetc(Fin);
  count = count + 1;
  i = (i + 1) % 256;
  j = (j + S[i]) % 256;
  swap(S[i], S[j]);    // Pertukarkan nilai S[i] dan S[j]
  t = (S[i] + S[j]) % 256;
  u = S[t];   // keystream
  c = u ^ p;
  fputc(c, Fout);
}
fclose(Fin);  fclose(Fout);
}
```

- Dekripsi

```
//Dekripsi sembarang berkas dengan algoritma RC4
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
using namespace std;

main(int argc, char *argv[])
{
 FILE *Fin, *Fout;
 char p, c, u;
 string K;
 int S[256];
 int i, j, t;
```

```
Fin = fopen(argv[1], "rb");
 if (Fin == NULL) {
   cout << "Berkas " << argv[1] <<" tidak ada" << endl;
   exit(0);
 }

 Fout = fopen(argv[2], "wb");

 cout << "Kata kunci : "; cin >> K;
 cout <<"Dekripsi " << argv[1] << " menjadi " << argv[2] << "...";

 for (i = 0; i<256; i++) {
  S[i] = i;
 }
```

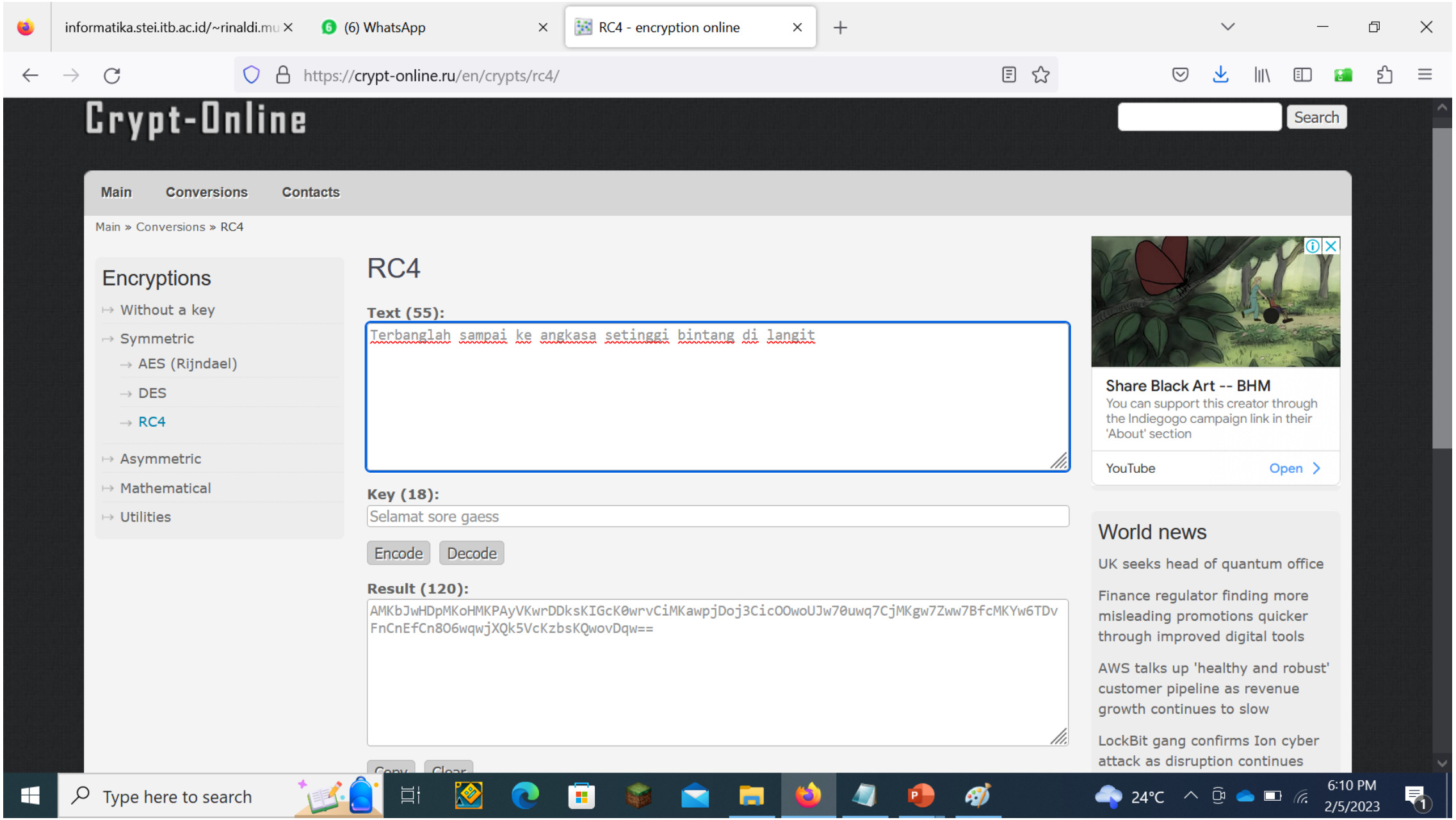
```
j = 0;
 for (i=0; i<256; i++) {
    j = (j + S[i] + K[i % K.length()]) % 256;
    swap(S[i], S[j]); // Pertukarkan nilai S[i] dan S[j]
 }

 i = 0;   j = 0;
 count = 0;
 while (!feof(Fin)) {
    c = fgetc(Fin);
    i = (i + 1) % 256;
    j = (j + S[i]) % 256;
    swap(S[i], S[j]);    // Pertukarkan nilai S[i] dan S[j]
    t = (S[i] + S[j]) % 256;
    u = S[t];   // keystream
    p = u ^ c;
    fputc(p, Fout);
 }
 fclose(Fin);  fclose(Fout);
}
```

Demo RC4 Online

1. https://crypt-online.ru/en/crypts/rc4/ (untuk teks saja)
2. https://www.lddgo.net/en/encrypt/rc4 (untuk teks saja)
3. http://rc4.online-domain-tools.com/ (untuk teks dan file)
4. https://tomeko.net/online_tools/rc4.php?lang=en (RC4 offline tool)

https://crypt-online.ru/en/crypts/rc4/

Enkripsi



Dekripsi

informatika.stei.itb.ac.id/~rinaldi.mu | (6) WhatsApp | RC4 - encryption online

https://crypt-online.ru/en/crypts/rc4/

Crypt-Online

Search

Main | Conversions | Contacts

Main » Conversions » RC4

Encryptions

- Without a key
- Symmetric
  - AES (Rijndael)
  - DES
  - RC4
- Asymmetric
- Mathematical
- Utilities

RC4

Text (120):

AMKbJwHDpMKoHMKPAyVKwrDDksKIGcK0wrvCiMKawpjDoj3CicOOwoUJw70uwq7CjMKgw7Zww7BfcMKYw6TDv
FnCnEfCn8O6wqwjXQk5VcKzbsKQwovDqw==

Key (18):

Selamat sore gaess

Encode | Decode

Result (55):

Terbanglah sampai ke angkasa setinggi bintang di langit

Share Black Art -- BHM

You can support this creator through the Indiegogo campaign link in their 'About' section

YouTube Open

World news

UK seeks head of quantum office

Finance regulator finding more misleading promotions quicker through improved digital tools

AWS talks up 'healthy and robust' customer pipeline as revenue growth continues to slow

LockBit gang confirms Ion cyber attack as disruption continues

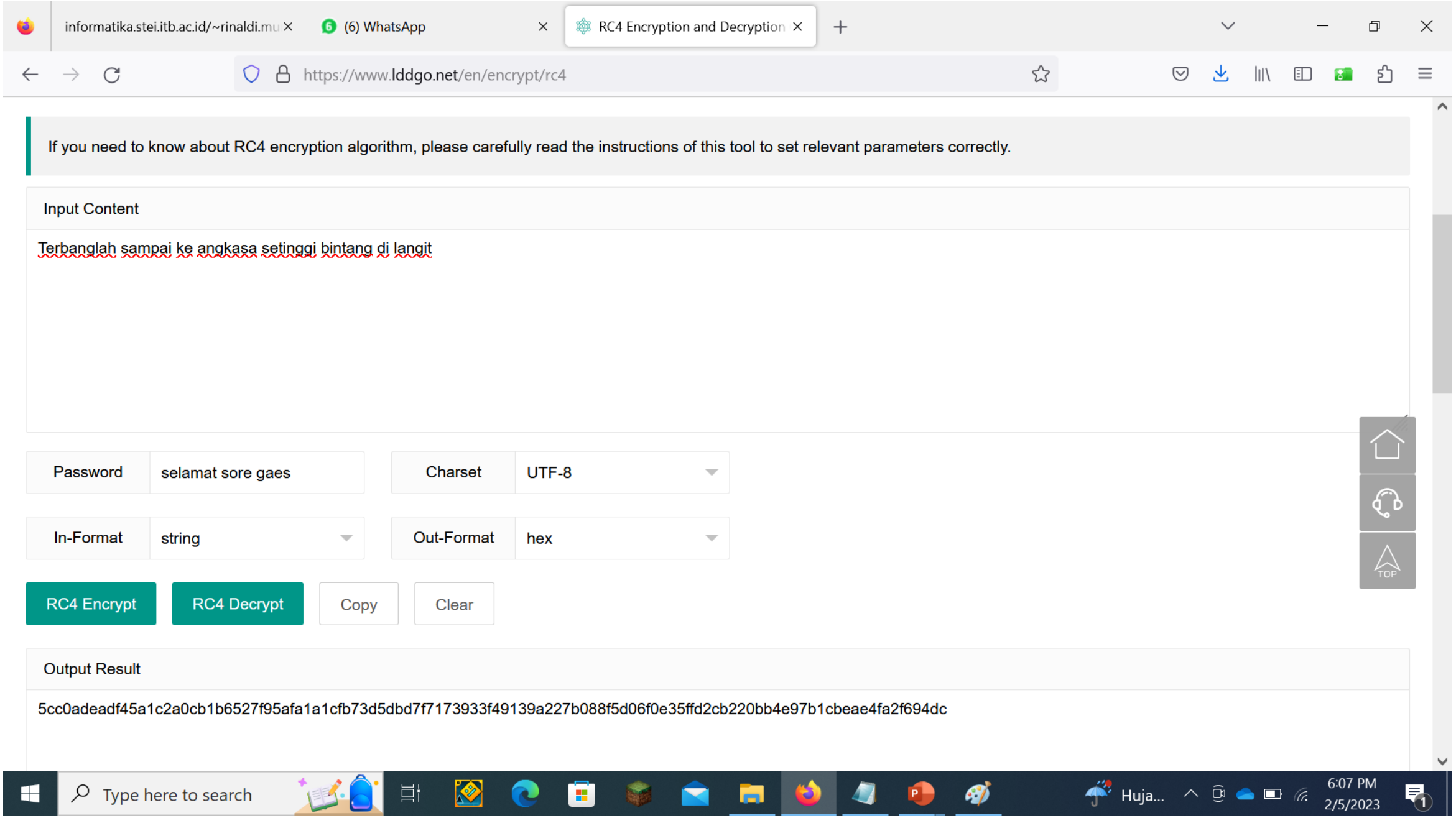
Type here to search

Senja

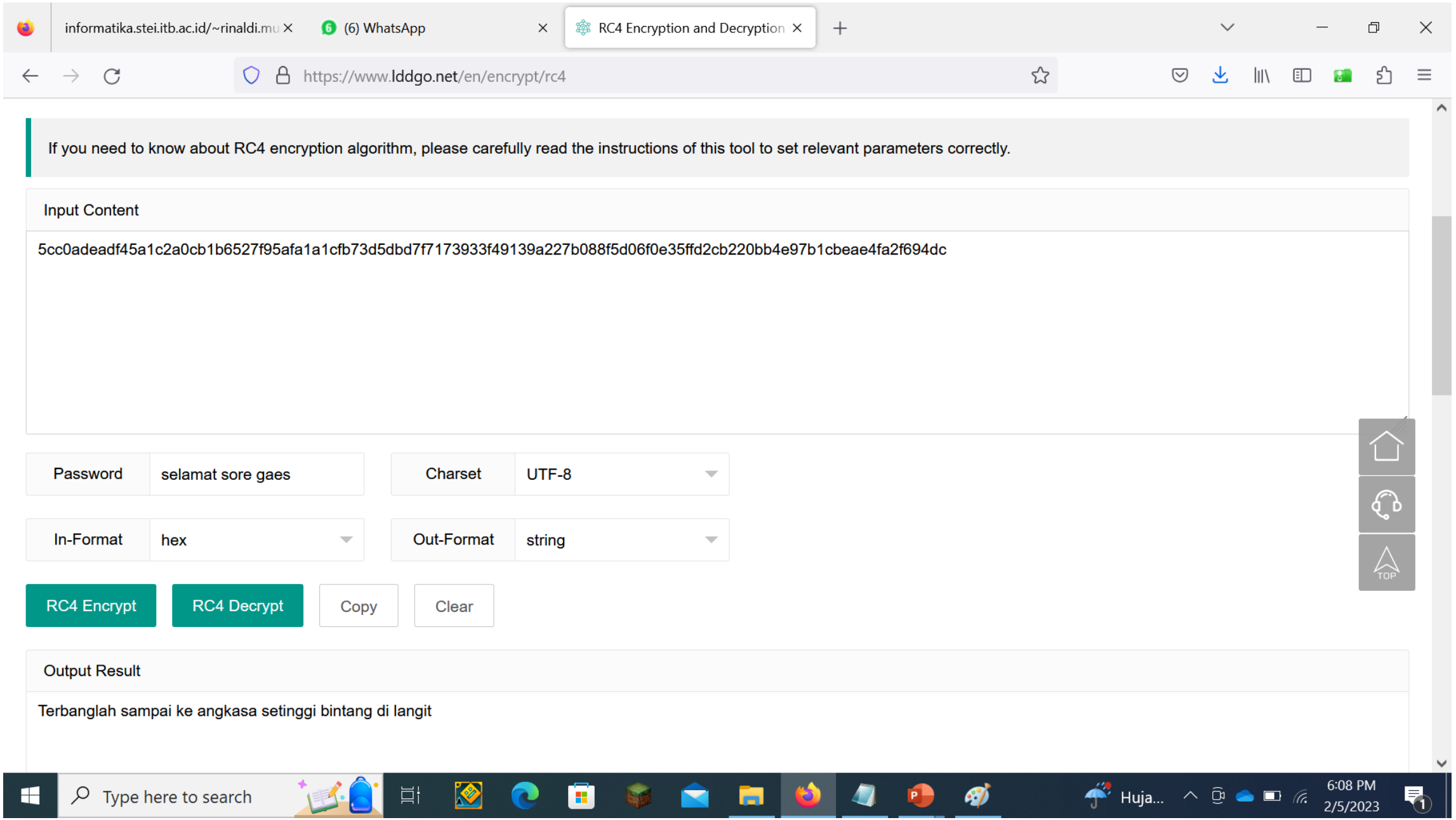
6:12 PM
2/5/2023

https://www.lddgo.net/en/encrypt/rc4

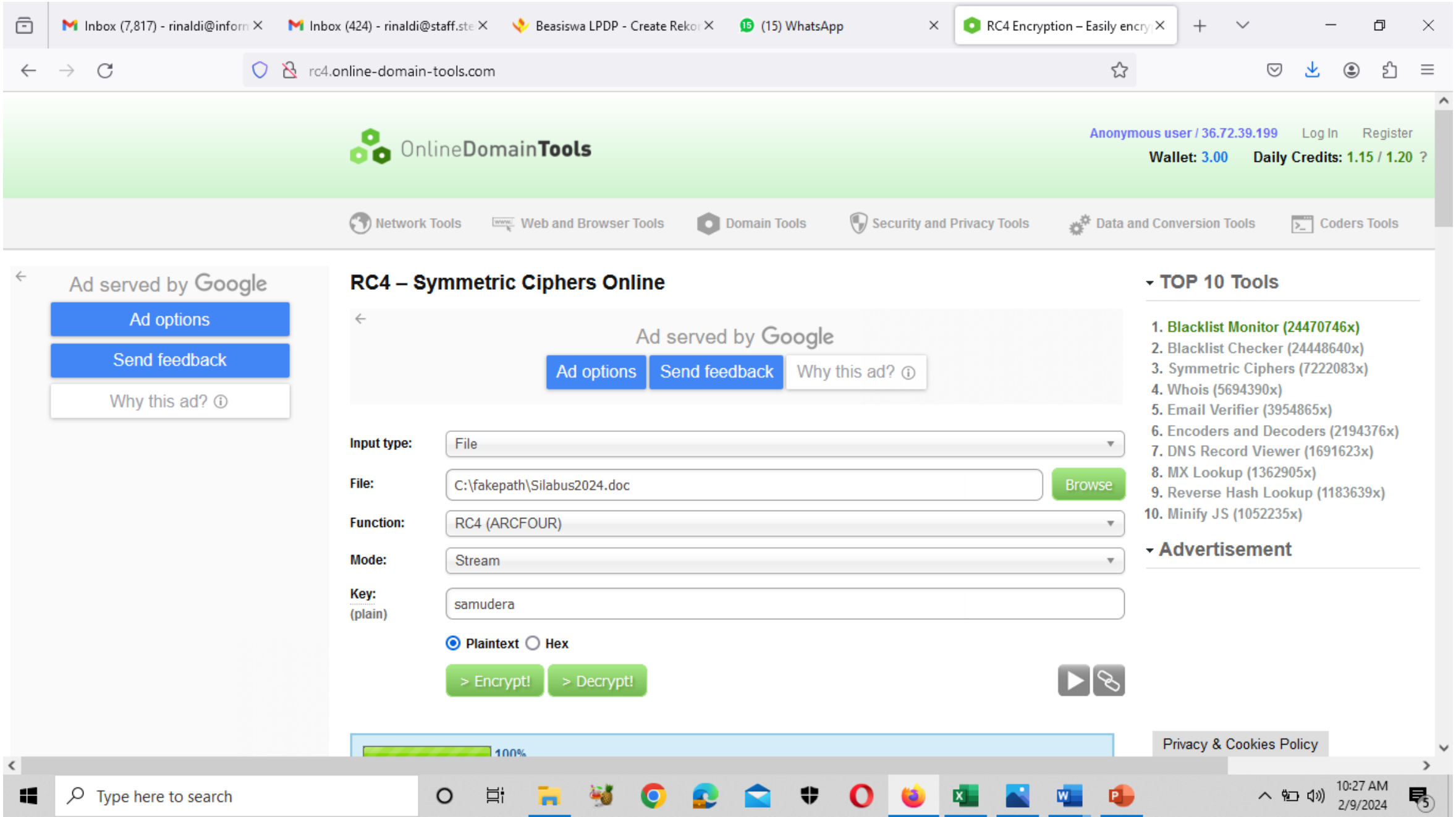
Enkripsi



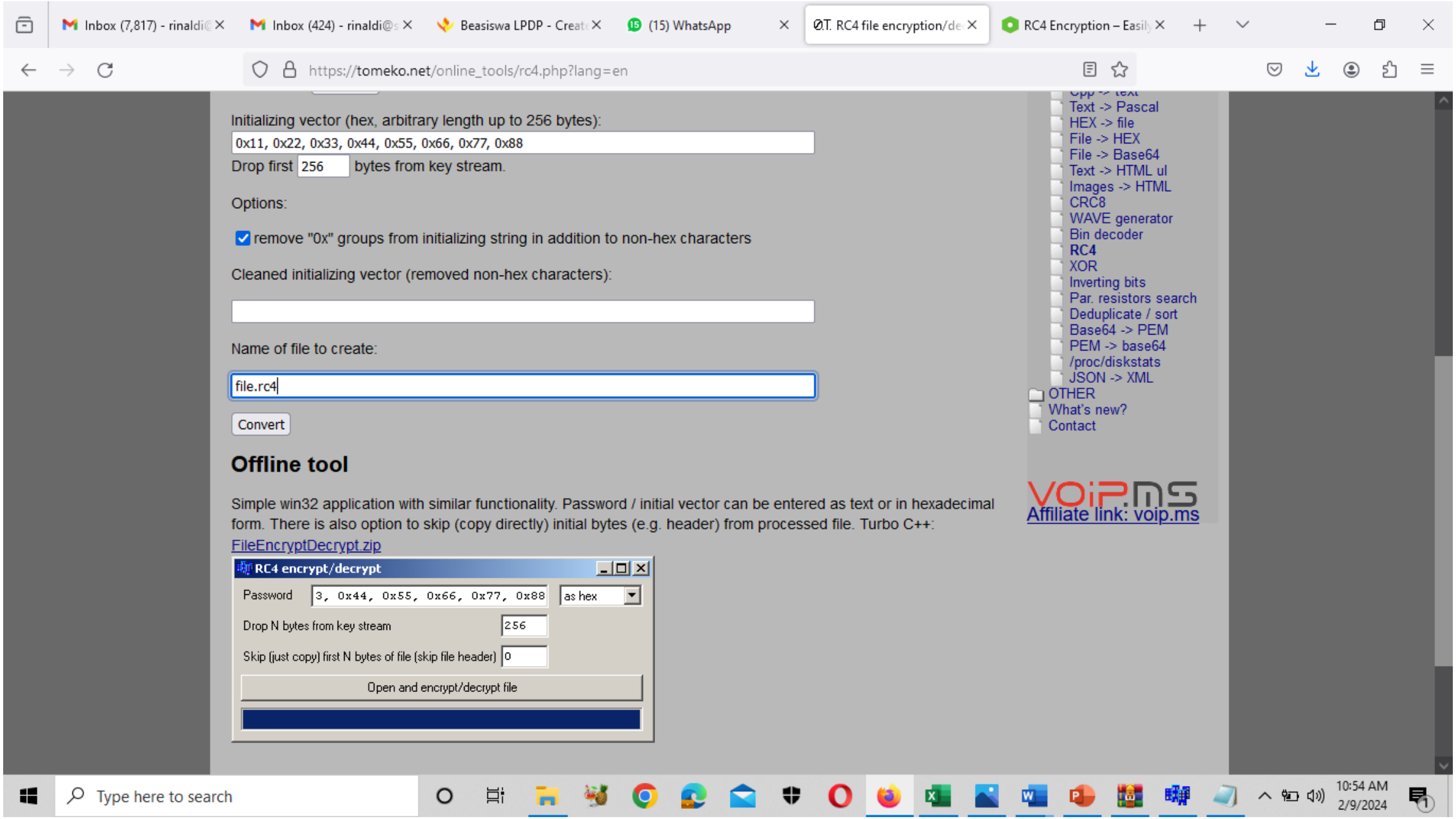
Dekripsi


informatika.stei.itb.ac.id/~rinaldi.mu
(6) WhatsApp
RC4 Encryption and Decryption
https://www.lddgo.net/en/encrypt/rc4
If you need to know about RC4 encryption algorithm, please carefully read the instructions of this tool to set relevant parameters correctly.
Input Content
5cc0adeadf45a1c2a0cb1b6527f95afa1a1cfb73d5dbd7f7173933f49139a227b088f5d06f0e35ffd2cb220bb4e97b1cbeae4fa2f694dc
Password
selamat sore gaes
Charset
UTF-8
In-Format
hex
Out-Format
string
RC4 Encrypt
RC4 Decrypt
Copy
Clear
Output Result
Terbanglah sampai ke angkasa setinggi bintang di langit
TOP
Type here to search
Huja...
6:08 PM
2/5/2023

http://rc4.online-domain-tools.com/



https://tomeko.net/online_tools/rc4.php?lang=en



Enkripsi citra dengan RC4

- RC4 cocok digunakan untuk mengenkripsi file citra (*image*) tanpa merusak *header* filenya,
- karena citra terdiri atas sejumlah *pixel*, setiap *pixel* berukuran 1 *byte* (*grayscale image*) atau 3 byte (*color image*).

- Ingatlah bahwa RC4 membangkitkan *keystream* berupa rangkaian *byte*

- Dengan mengenkripsi setiap *byte pixel* dengan setiap *byte keystream*, *pixel-pixel* citra terenkripsi.

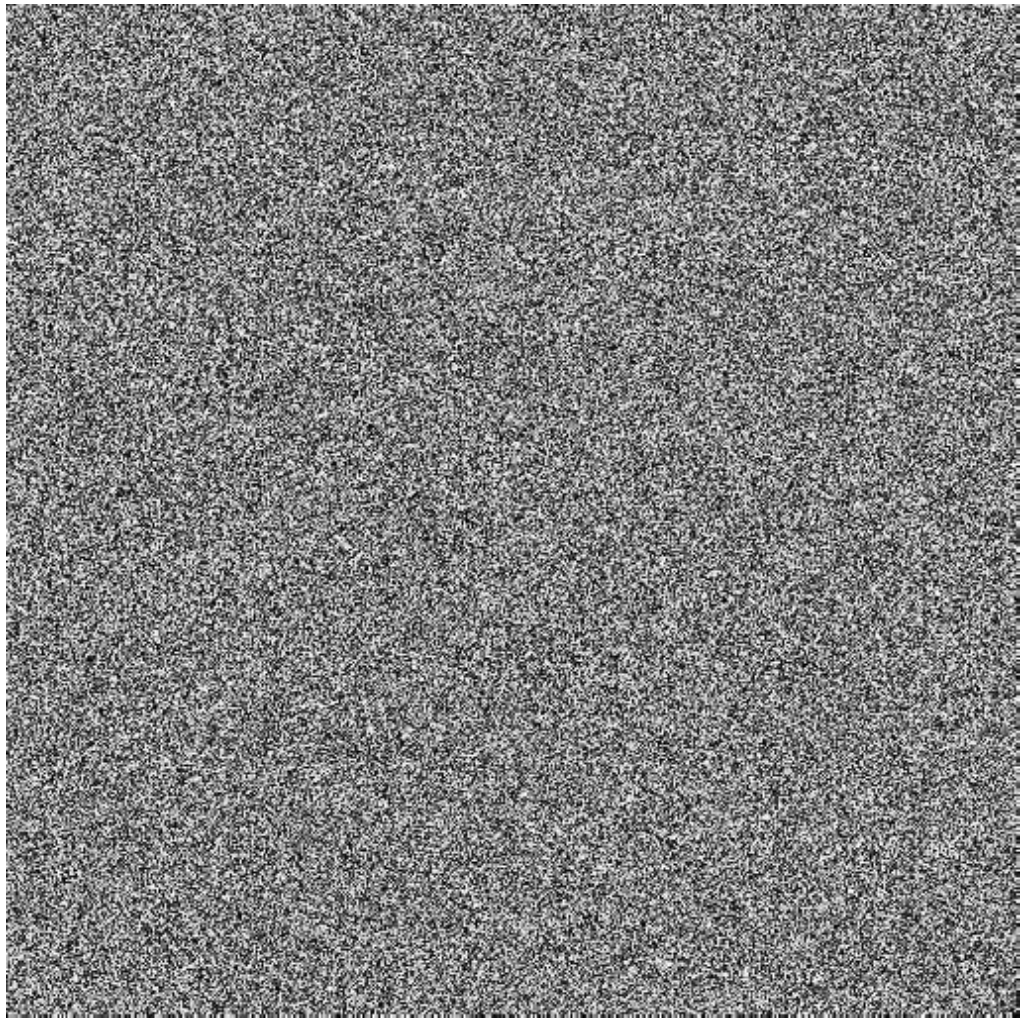
```
% Program enkripsi citra dengan RC4
% Dibuat oleh: Rinaldi Munir
clc;
clear all;
Image = imread('boat.bmp');
imshow(Image);
K = input('Ketikkan kunci (integer, contoh 123456): ', 's');
[m, n] = size(Image);
for i = 1 : 256
    S(i) = i;
end

j = 1;
for i = 1 : 256
      j = mod(j + S(i) + str2num(K(mod(i,  length(K)) + 1)),256) + 1;
      temp = S(i);
      S(i) = S(j);
      S(j) = temp;
end
```

```
i = 1;
j = 1;
for p = 1: m
    for q = 1 : n
        i = mod (i + 1, 256) + 1;
        j = mod(j + S(i), 256) + 1;
        temp = S(i);
        S(i) = S(j);
        S(j) = temp;
        t = mod(S(i) + S(j), 256) + 1;
        u = S(t);
        Image(p,q) = bitxor(uint8(u), Image(p, q));
     end
end
figure, imshow(Image);
figure, imhist(Image);
imwrite(Image, 'image_enc.bmp');
```

```
% Program dekripsi citra dengan RC4
% Dibuat oleh: Rinaldi Munir
clc;
clear all;
Image = imread('image_enc.bmp');
K = input('Ketikkan kunci (integer, contoh 123456): ', 's');
[m, n] = size(Image);
for i = 1 : 256
    S(i) = i;
end

j = 1;
for i = 1 : 256
      j = mod(j + S(i) + str2num(K(mod(i,  length(K)) + 1)),256)
+ 1;
      temp = S(i);
      S(i) = S(j);
      S(j) = temp;
end
```

```
i = 1;
j = 1;
for p = 1: m
    for q = 1 : n
        i = mod (i + 1, 256) + 1;
        j = mod(j + S(i), 256) + 1;
        temp = S(i);
        S(i) = S(j);
        S(j) = temp;
        t = mod(S(i) + S(j), 256) + 1;
        u = S(t);
        Image(p,q) = bitxor(uint8(u), Image(p, q));
     end
end
imshow(Image);
figure, imhist(Image);
imwrite(Image, 'original.bmp');
```

Plain-image



Encrypted image