

IF4020 Kriptografi

Kriptografi Modern



Oleh: Rinaldi Munir

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika

ITB

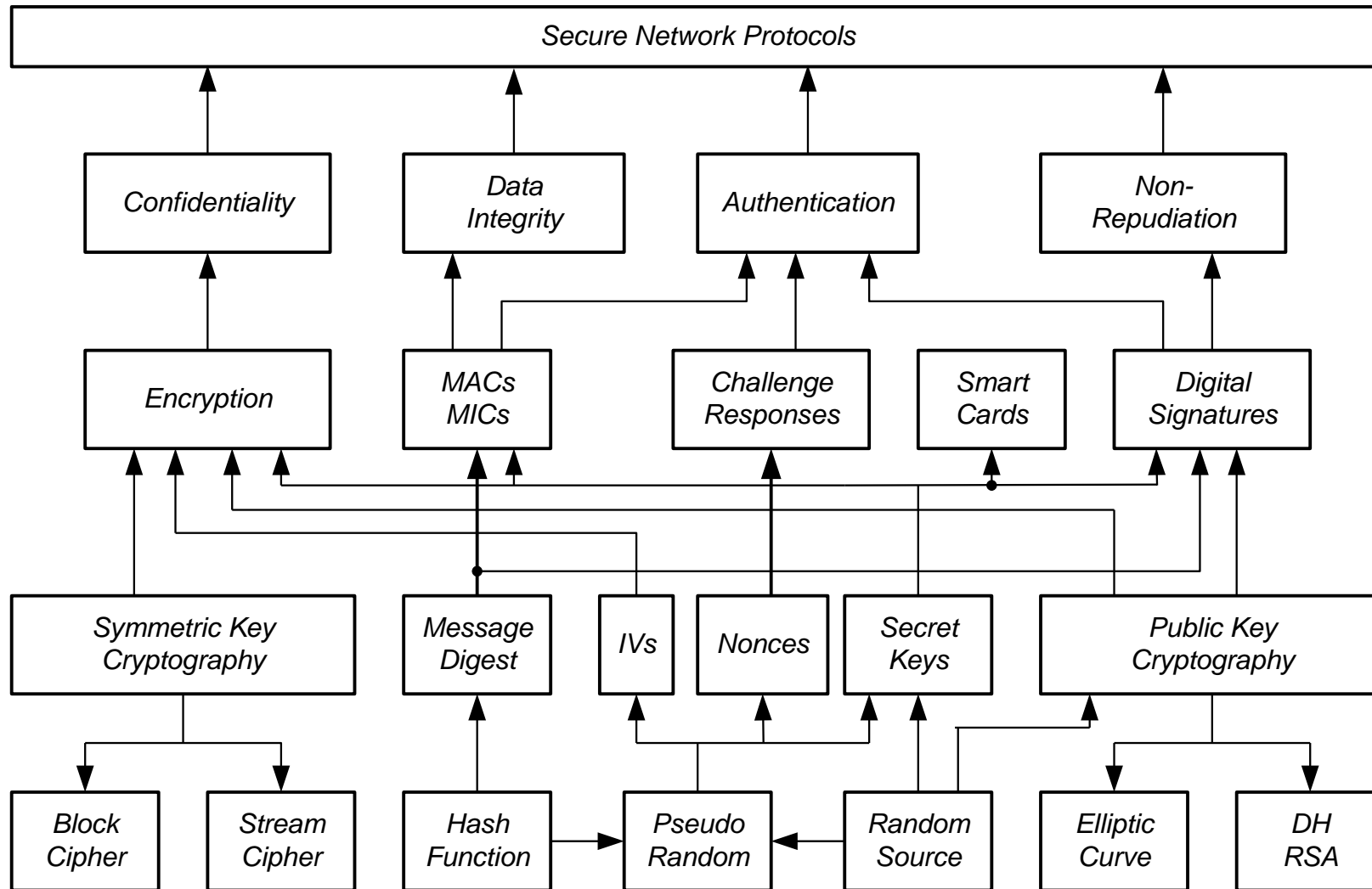
Rinaldi M., IF4020 Kriptografi

Pendahuluan

- Beroperasi dalam mode bit atau *byte* (algoritma kriptografi klasik beroperasi dalam mode karakter)
 - kunci, plainteks, cipherteks, diproses dalam rangkaian bit/byte
 - operasi bit **xor** paling banyak digunakan

- Tetap menggunakan teknik pada algoritma klasik: substitusi dan transposisi, tetapi lebih kompleks (Tujuan: sangat sulit dikriptanalisis)
- Perkembangan algoritma kriptografi modern didorong oleh penggunaan komputer digital untuk keamanan pesan.
- Komputer digital merepresentasikan data dalam biner.

Diagram Blok Kriptografi Modern



Rangkaian bit

- Pesan (dalam bentuk rangkaian bit) dipecah menjadi beberapa blok

- Contoh: Plainteks 100111010110

Bila dibagi menjadi blok 4-bit

1001 1101 0110

maka setiap blok menyatakan 0 sampai 15 :

9 13 6

Bila plainteks dibagi menjadi blok 3-bit:

100 111 010 110

maka setiap blok menyatakan 0 sampai 7 :

4 7 2 6

- *Padding bits*: bit-bit tambahan jika ukuran blok terakhir tidak mencukupi panjang blok

- Contoh: Plainteks 100111010110

Bila dibagi menjadi blok 5-bit:

10011 10101 00010

Padding bits mengakibatkan ukuran cipherteks sedikit lebih besar daripada ukuran plaintexts semula.

Representasi dalam Heksadesimal

- Pada beberapa algoritma kriptografi, pesan dinyatakan dalam kode Hex:

0000 = 0 0001 = 1 0010 = 2 0011 = 3

0100 = 4 0101 = 5 0110 = 6 0111 = 7

1000 = 8 1011 = 9 1010 = A 1011 = B

1100 = C 1101 = D 1110 = E 1111 = F

- Contoh: plainteks **100111010110** dibagi menjadi blok 4-bit:

1001 1101 0110

dalam notasi Hex adalah **9 D 6**

Operasi *XOR*

- Paling banyak digunakan di dalam *cipher* modern
- Notasi: \oplus
- Operasi:

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

$$1 \oplus 1 = 0$$

- Operasi XOR = penjumlahan modulo 2:

$$0 \oplus 0 = 0 \iff 0 + 0 \pmod{2} = 0$$

$$0 \oplus 1 = 1 \iff 0 + 1 \pmod{2} = 1$$

$$1 \oplus 0 = 1 \iff 1 + 0 \pmod{2} = 1$$

$$1 \oplus 1 = 0 \iff 1 + 1 \pmod{2} = 0$$

- Hukum-hukum yang terkait dengan operator XOR:

(i) $a \oplus a = 0$

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

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

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: $10011 \oplus 11001 = 01010$

yang dalam hal ini, hasilnya diperoleh sebagai berikut:

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

Cipher dengan 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	\oplus	(karakter '5')
<hr/>				
	cipherteks	01010000		(karakter 'P')
	kunci	00110101	\oplus	(karakter '5')
<hr/>				
	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

```

// 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;

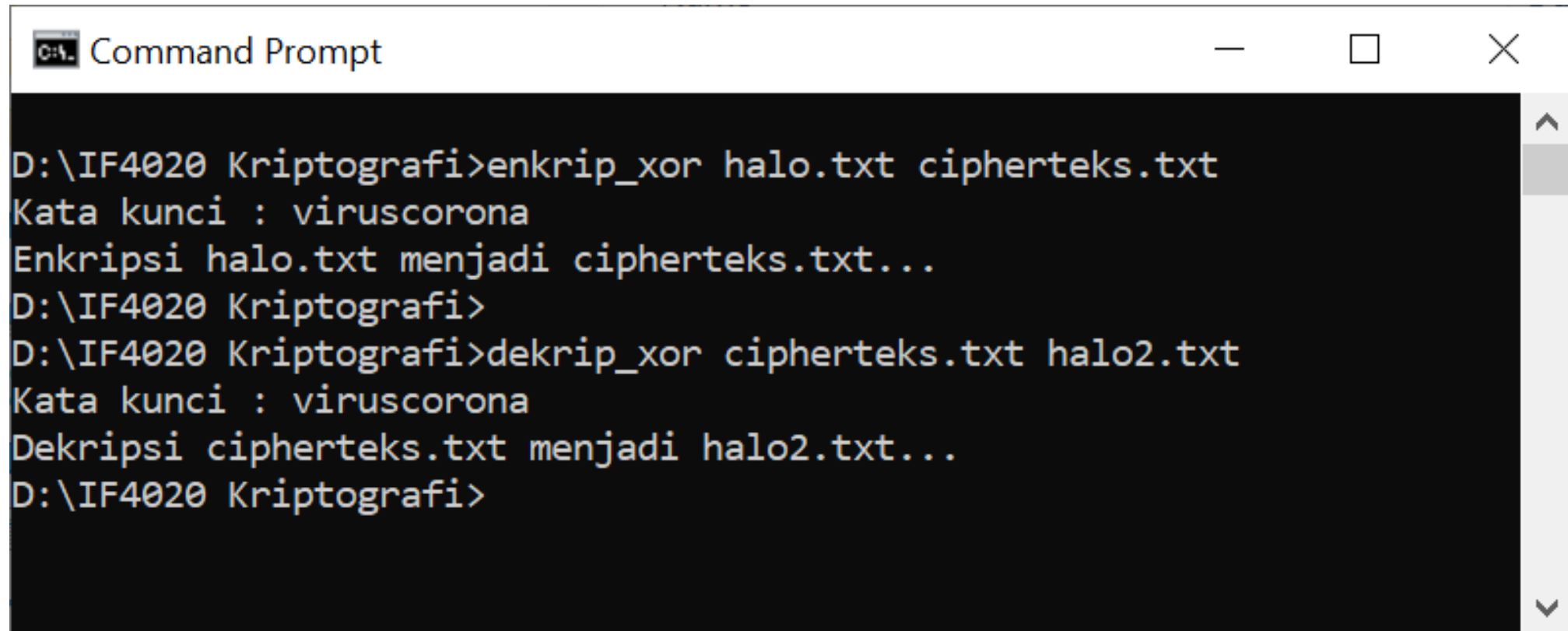
    Fin = fopen(argv[1], "rb");
    if (Fin == NULL){
        cout << "Berkas " << argv[1] << "
tidak ada" << endl;
        exit(0);
    }

    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



```
Command Prompt
D:\IF4020 Kriptografi>enkrip_xor halo.txt cipherteks.txt
Kata kunci : viruscorona
Enkripsi halo.txt menjadi cipherteks.txt...
D:\IF4020 Kriptografi>
D:\IF4020 Kriptografi>dekrip_xor cipherteks.txt halo2.txt
Kata kunci : viruscorona
Dekripsi cipherteks.txt menjadi halo2.txt...
D:\IF4020 Kriptografi>
```

<p>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.</p>	<pre> 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 </pre>
--	--

Sayangnya, algoritma *XOR* sederhana tidak aman karena ciphertekstanya mudah dipecahkan. Panjang kuncinya dapat ditemukan dengan Metode Kasiski.

Kategori *cipher* Berbasis Bit

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

- beroperasi pada bit tunggal
- enkripsi/dekripsi bit per bit

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

- beroperasi pada blok bit
(contoh: 64-bit/blok = 8 karakter/blok)
- enkripsi/dekripsi blok per blok