Analisis End-to-End Encryption dengan Diffie-Hellman Key Exchange Menggunakan Prinsip Aljabar Boolean dan Teori Bilangan

Louis Caesa Kesuma - 13521069[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521069@std.stei.itb.ac.id

***Abstract*—Kriptografi adalah salah satu ilmu atau keahlian yang sangat sering digunakan di kehidupan sehari-hari, penerapannya dapat berupa perbincangan sederhana dengan orang lain hingga pengiriman informasi dari satu perangkat ke perangkat lainnya. Salah satu metode kriptografi yang sering kita gunakan pada saat ini adalah *End-To-End Encryption* (E2EE). E2EE tergolong aman karena tidak ada pihak ke-3 yang bisa menerima data yang dikirim dari satu perangkat ke perangkat lain. Cara agar E2EE bisa memfilter perangkat-perangkat yang bisa menenkripsi dan mendekripsi informasi adalah dengan cara menggunakan *key* yang ada di setiap perangkat. Salah satu contoh penerapan dari E2EE adalah dengan menggunakan metode *Diffie-Hellman key exchange*. E2EE dengan *Diffie-Hellman key exchange* dapat diterapkan dengan prinsip-prinsip pada aljabar boolean dan teori bilangan.**

***Keywords*—Kriptografi, *End-To-End Encryption*, *key*, Aljabar Boolean, Teori Bilangan.**

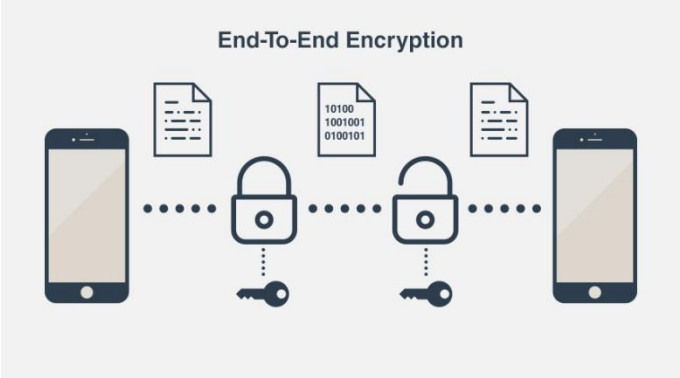
I. PENDAHULUAN

Kriptografi adalah salah satu ilmu/keahlian yang sangat sering kita gunakan sehari-hari. Walaupun kemungkinan besar masyarakat masih belum mengetahui tentang istilah kriptografi. Kriptografi dapat berupa perbincangan sederhana antara satu orang dengan orang yang lain, perbincangan tersebut mendorong pembicara untuk merancang kata-kata yang ingin dia sampaikan ke orang lain, dan pendengar untuk memahami kata-kata tersebut.

Salah satu penerapan kriptografi yang terkenal adalah *Enigma Machine* yang digunakan oleh Jerman pada Perang Dunia Ke-2 untuk mengirimkan informasi ke prajurit-prajurit mereka. *Enigma Machine* tersebut dulu dianggap Jerman adalah alat kriptografi yang sempurna dikarenakan kode-kode yang dihasilkan oleh mesin tersebut dapat berubah-ubah sehingga menyusahkan para musuh mereka untuk menerjemahkan kode-kode yang mereka gunakan. Akan tetapi, mesin tersebut kemudian dipecahkan kodenya oleh sekutu. Cara kerja mesin tersebut adalah dengan memanfaatkan rotor-rotor yang berputar seiring input dimasukkan ke mesin tersebut, sehingga untuk setiap input kode yang mengubah inputnya juga akan berbeda.

Sekarang kriptografi biasanya digunakan pada perangkat-perangkat digital kita untuk mengamankan informasi yang dikirim/diterima pengguna agar informasi tersebut tidak mudah dicuri oleh orang lain. Metode-metodenya sangatlah beragam, salah satunya adalah *End-To-End Encryption*.

Proses dari *End-To-End Encryption* berupa mengenkripsikan terlebih dahulu informasi yang ingin dikirm dan mendekripsikannya saat informasi tersebut diterima. Cara mengenkripsi dan mendekripsinya adalah dengan menggunakan *key* yang telah disetujui tiap pengguna. Biasanya tiap perangkat memiliki *key* yang berbeda-beda, dan *key* yang digunakan saat pengiriman informasi adalah *key* gabungan dari 2 buah perangkat.



Gambar 1.1. Ilustrasi sederhana End-To-End Encryption
(Sumber: https://pinngle.me/blog/why-use-end-to-end-encrypted-messaging-apps/)

Proses enkripsi dan dekripsi pada *End-To-End Encryption* dapat kita realisasikan dengan menggunakan operasi-operasi pada aljabar Boolean, contohnya seperti operasi gerbang XOR dan operasi gerbang AND.

Salah satu contoh metode penggabungan 2 buah *key* adalah dengan menggunakan *Diffie-Hellman key exchange*. *Diffie-Hellman key exchange* adalah salah satu metode untuk saling mengirim *key* secara *public* namun tidak membocorkan *key* yang sebenarnya. Proses pengiriman *key* tersebut biasanya dibantu oleh *gateway* yang disediakan perusahaan penyedia jasa pengiriman informasi, contohnya seperti *WhatsApp* yang mengunakan metode End-To-End Encryption.

II. LANDASAN TEORI

A. *Aljabar Boolean*

Aljabar boolean adalah salah satu bentuk aljabar yang berhubungan dengan bilangan biner yang dapat

merepresentasikan True atau False. Pengaplikasian aljabar boolean sangatlah beragam, contoh: *Integrated Circuit*, dan Jaringan saklar.

Operasi-operasi dasar aljabar boolean berupa:

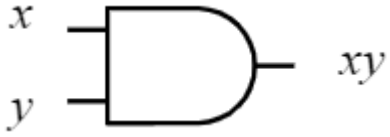
1. +
   i. 1 + 1 = 1
   ii. 1 + 0 = 1
   iii. 0 + 0 = 0
2. .
   i. 1 . 1 = 1
   ii. 1 . 0 = 0
   iii. 0 . 0 = 0

Hukum-hukum dasar aljabar boolean berupa:

1. Hukum Identitas
   i. a + 0 = a
   ii. a . 1 = a
2. Hukum Indempoten
   i. a + a = a
   ii. a . a = a
3. Hukum Komplemen
   i. a + a’ = 1
   ii. aa’ = 0
4. Hukum Dominasi
   i. a . 0 = 0
   ii. a + 1 = 1
5. Hukum Involusi
   i. (a’)’ = a
6. Hukum Penyerapan
   i. a + ab = a
   ii. a (a+b) = a
7. Hukum Komutatif
   i. a + b = b + a
   ii. a . b = b . a
8. Hukum Asosiatif
   i. a + (b + c) = (a + b) + c
   ii. a . (b . c) = (a . b) . c
9. Hukum Distributif
   i. a . (b + c) = a . b + a . c
   ii. a + (b . c) = (a + b) . (a + c)
10. Hukum De Morgan
    i. (a + b)’ = a’b’
    ii. (ab)’ = a’ + b’
11. Hukum 0/1
    i. 0’ = 1
    ii. 1’ = 0

Gerbang-gerbang logika sederhana yang tercipta dari hukum-hukum dan operasi tersebut berupa:

1. Gerbang AND

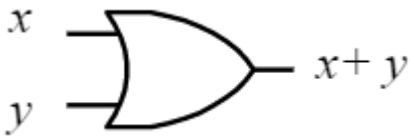


Gambar 2.1. Gerbang AND
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Aljabar-Boolean-(2020)-bagian1.pdf)

Input		Output
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

Tabel 2.1. Hasil gerbang logika AND

2. Gerbang OR



Gambar 2.2. Gerbang OR
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Aljabar-Boolean-(2020)-bagian1.pdf)

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	1

Tabel 2.2. Hasil gerbang logika OR

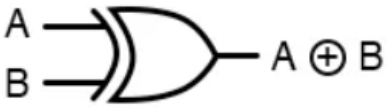
3. Gerbang NOT

Gambar 2.3. Gerbang NOT
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Aljabar-Boolean-(2020)-bagian1.pdf)

Input	Output
A	
0	1
1	0

Tabel 2.3. Hasil gerbang logika NOT

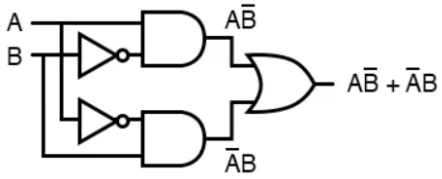
4. Gerbang XOR



Gambar 2.4. Gerbang XOR
(Sumber: https://www.allaboutcircuits.com/textbook/digital/chpt-7/the-exclusive-or-function-xor/)

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

Tabel 2.4. Hasil gerbang logika XOR



Gambar 2.5. Gerbang XOR dinyatakan dengan gerbang AND dan NOT
(Sumber: https://www.allaboutcircuits.com/textbook/digital/chpt-7/the-exclusive-or-function-xor/)

5. Gerbang NAND

Gambar 2.6. Gerbang NAND
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Aljabar-Boolean-(2020)-bagian1.pdf)

Input		Output
A	B	
0	0	1
0	1	1
1	0	1
1	1	0

Tabel 2.5. Hasil gerbang logika NAND

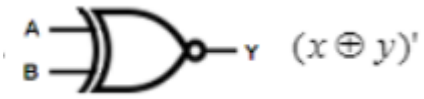
6. Gerbang NOR

Gambar 2.7. Gerbang NOR
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Aljabar-Boolean-(2020)-bagian1.pdf)

Input		Output
A	B	
0	0	1
0	1	0
1	0	0
1	1	0

Tabel 2.6. Hasil gerbang logika NOR

7. Gerbang XNOR



Gambar 2.8. Gerbang XNOR
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Aljabar-Boolean-(2020)-bagian1.pdf)

Input		Output
A	B	
0	0	1
0	1	0
1	0	0
1	1	1

Tabel 2.7. Hasil gerbang logika XOR

B. *Teori Bilangan*

Teori bilangan adalah cabang matematika murni yang bertujuan untuk mempelajari bilangan bulat (*integer*) atau fungsi bilangan bulat. Bilangan bulat (*integer*) adalah bilangan yang tidak mempunyai pecahan desimal.

Beberapa teorema di teori bilangan adalah:

1. Sifat Pembagian pada Bilangan Bulat

   Misalkan a habis membagi b, maka bisa dinyatakan dalam:

   $$a|b$$

   Contoh:
   - 4 habis membagi 12
     Bisa dinyatakan dalam:
     $$4|12$$

2. Teorema Euclidean

   Misalkan bilangan m dan n adalah bilangan bulat dimana $n > 0$. Jika m dibagi dengan n maka hasil pembagiannya adalah q $(quotient)$ dan sisanya adalah r $(remainder)$, sehingga dapat dinyatakan dalam:

   $$m = nq + r, \text{dimana} : \ 0 \le r < n$$

   Contoh:
   - 1987/97 = 20, sisa 47
     bisa dinyatakan dalam:
     $$1987 = 20 \cdot 97 + 47$$
   - -22/3 = -8, sisa 2
     bisa dinyatakan dalam:
     $$-22 = (-8) \cdot 3 + 2$$

3. Pembagi Bersama Terbesar (PBB)

   Misalkan a, dan b adalah bilangan bulat tidak nol. Jika ada bilangan bulat terbesar d sedemikian rupa sehingga $d|a$, dan $d|b$, maka kita bisa menyatakannya dalam:

   $$PBB(a, b) = d$$

   Contoh:
   - PBB(45, 36) = 9
     Karena faktor pembagi 45 adalah: 1, 3, 5, 9, 15, 45. Faktor pembagi 36 adalah: 1, 2, 3, 4, 9, 12, 18, 36. Sehingga bilangan yang sama dan terbesar dari faktor pembagi tersebut adalah 9, sehingga PBB(45, 36) adalah 9.

4. Algoritma Euclidean

   Misalkan m dan n adalah bilangan bulat tak negatif dengan $m \ge n$. Misalkan $r_0 = m$ dan $r_1 = n$. Lakukan secara berturut-turut pembagian untuk memperoleh:

   $$r_0 = r_1 q_1 + r_2, 0 \le r_2 < r_1$$
   $$r_1 = r_2 q_2 + r_3, 0 \le r_3 < r_2$$
   $$\vdots$$
   $$r_{n-2} = r_{n-1} q_{n-1} + r_n, 0 \le r_n < r_{n-1}$$
   $$r_{n-1} = r_n q_n + 0$$

Menurut teorema 2:

$$PBB(m,n) = PBB(r_0, r_1) = PBB(r_1, r_2) = \cdots$$
$$= PBB(r_{n-2}, r_{n-1})$$
$$= PBB(r_{n-1}, r_n) = r_n$$

Jadi, PBB dari m dan n adalah sisa terakhir yang tidak nol dari runtutan pembagian tersebut.

Contoh:

- $m = 80$ dan $n = 12$ dan dipenuhi syarat $m \geq n$

$$80 = 6 \cdot 12 + 8$$
$$12 = 1 \cdot 8 + 4$$
$$8 = 2 \cdot 4 + 0$$

Maka didapat PBB(80,12) adalah 4

5. Relatif Prima

Relatif prima adalah Ketika PBB dari kedua bilangan adalah 1.

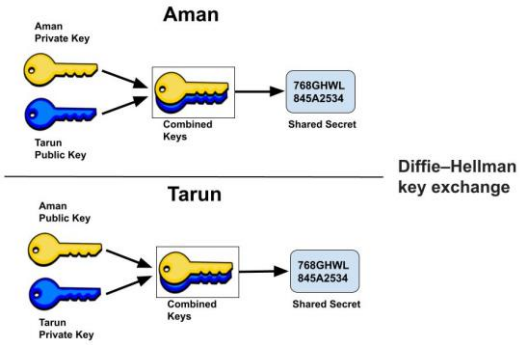
Contoh:

- 23 dan 5 relatif prima sebab PBB(23, 5) adalah 1

C. *End-To-End Encryption*

End-To-End Encryption adalah sistem komunikasi dimana tidak ada orang lain yang bisa mengakses pesan-pesan yang dikirim serta diterima kedua pihak yang sedang berkomunikasi. Bahkan perusahaan yang digunakan pihak-pihak tersebut untuk saling mengirim informasi tersebut tidak bisa mengakses informasi dari pengguna-penggunanya.

Perusahaan-perusahaan penyedia layanan pengiriman informasi yang menggunakan *End-To-End Encryption* biasanya tidak memiliki akses terhadap *key* yang dapat digunakan oleh mereka untuk mendekripsi informasi yang masuk ke perangkat user. Perusahaan-perusahaan tersebut menggunakan *key* setiap perangkat sehingga hanya perangkat dengan key tersebut yang bisa mendekripsi informasinya. *Key* (biasanya berupa *binary* 256-bit) setiap perangkat akan dibuat saat ada informasi yang dikirim/diterima perangkat tersebut. *Key* yang dibuat setiap perangkat ada 2 macam, yaitu: *public key*, dan *private key*. *Key-key* tersebut dibuat sedemikian rupa sehingga *private key* (penerima) jika digabungan dengan *public key* (pengirim) akan sama dengan *key* (pengirim) digabungan dengan *public key* (penerima). *Key* yang telah digabungkan tersebut adalah *key* yang akan digunakan oleh perangkat-perangkatnya untuk menenkripsikan dan mendekripsikan informasi. Proses penggabungan kedua key tersebut dinamakan *key exchange*.



Gambar 2.9. Analogi proses *key exchange*
(Sumber: https://cybermeteoroid.com/diffie-hellman-key-exchange-algorithm-how-it-works/)

Terdapat banyak metode *key exchange* yang bisa dipakai, salah satu yang mudah untuk diimplementasikan adalah dengan menggunakan *multiplicative group of integers modulo*. Metode ini menggunakan 4 *integer*, 2 *public*, dan 2 *private*. 2 *integer* yang *public* tersebut bukanlah public key yang setiap user melainkan merupakan angka yang akan digunakan untuk menghasilkan *public key* setiap perangkat. Misalkan 2 *public integer* tersebut adalah p dan q, dan 2 *private key* tersebut adalah x dan y. p merupakan angka prima dan q merupakan angka yang relatif prima dari q. Misalkan bahwa $p = 23$, $q = 5$, $x = 6$, dan $y = 15$. Maka *public key* dapat dibuat dengan cara:

$$publickey1 = q^x \ mod \ p$$
$$publickey2 = q^y \ mod \ p$$

Setelah *public key* tersebut dibuat, maka masing-masing perangkat akan menggabungkan *private key* mereka dengan *public key* untuk membuat *combined key* sesuai dengan gambar 6.5. *Combined key* dapat dibuat dengan cara:

$$combinedkey1 = publickey2^x \ mod \ p$$
$$combinedkey2 = publickey1^y \ mod \ p$$

Seharusnya $combinedkey1$ akan sama dengan $combinedkey2$ sehingga *combined key* tersebut dapat digunakan perangkat untuk memverifikasi perangkat mana yang berhak untuk mengirim/menerima informasi dengannya.

Karena perusahaan-perusahaan tersebut tidak memiliki akses ke *keys* dari user-user mereka, maka mereka hanya menjadi gateway penghubung untuk transfer informasi tiap usernya. Akan tetapi, akan salah juga jika dibilang bahwa perusahaan tersebut tidak mempunyai akses ke *public key* penggunanya karena untuk membuat *combined key* setiap perangkat harus bisa menerima *public key* dari perangkat yang mau menerima atau mengirim informasi.

III. Analisis End-To-End Encryption

Karena *key* yang digunakan di *End-To-End Encryption* berupa bit, prinsip-prinsip aljabar boolean dapat digunakan untuk menenkripsikan dan mendekripsikan informasi tersebut.

Pertama-tama masing-masing perangkat akan membuat 2 buah *key*, yaitu *public key* dan *private key*. *Public key* dibuat dengan cara men-*generate* 2 buah bilangan bulat yang dibuat sedemikian rupa sehingga keduanya relatif prima.

```python
# function untuk mengenerate key random
def generatePublicVal():
    # max value dari 256 bits
    max_value = math.pow(2, 256) - 1

    # random untuk keynya
    temp = random.randint(0, max_value)
    temp2 = random.randint(0, max_value)

    # generate key ke 2 yang relatif prima ke key pertama
    while euclidean(temp, temp2) != 1:
        temp2 = random.randint(0, max_value)

    return temp, temp2
```

Gambar 3.1. Implementasi sederhana pembuatan 2 bilangan bulat yang relatif prima
(Sumber: arsip pengguna)

```python
def euclidean(m, n):
    while n != 0:
        r = m % n
        m = n
        n = r

    return m
```

Gambar 3.2. Implementasi sederhana fungsi PBB dengan menggunakan teorema Euclidean
(Sumber: arsip pengguna)

```python
def generatePrivateKey():
    # max value dari 256 bits
    max_value = math.pow(2, 256) - 1

    # random untuk keynya
    temp = random.randint(0, max_value)
    temp2 = random.randint(0, max_value)

    return temp, temp2
```

Gambar 3.3. Implementasi sederhana pembuatan 2 private key
(Sumber: arsip pengguna)

Kemudian 2 bilangan bulat *public* dan 2 bilangan bulat private tersebut akan diproses sehingga membuat *combined key* yang nantinya dapat digunakan untuk mengenkripsi dan dekripsi informasi.

```python
def getCombinedKey(public1, public2, private1, private2):
    publickey1 = math.pow(public2, private1) % public1
    publickey2 = math.pow(public2, private2) % public1

    combinedKey1 = math.pow(publickey2, private1) % public1
    combinedKey2 = math.pow(publickey1, private2) % public1

    return int(combinedKey1), int(combinedKey2)
```

Gambar 3.4. Implementasi sederhana pembuatan combined key
(Sumber: arsip pengguna)

Informasi pertama-tama diubah dulu kedalam bentuk yang dapat memudahkan transfer data dari perangkat ke perangkat lain (misalkan bentuk informasi yang telah diubah menjadi *binary*). Karena *key* yang telah digabungkan itu sama untuk setiap user, maka gerbang XOR bisa digunakan untuk menenkripsikan serta mendekripsikan informasi tersebut.

```python
# function untuk mengconvert string ke dalam bentuk binary
def toBinary(a):
    temp = str(a)
    tempbin = []
    textAsBinary = ""

    for i in temp:
        tempbin.append(ord(i))

    for i in tempbin:
        textAsBinary += str(bin(i)[2:])

    return textAsBinary
```

Gambar 3.5. Implementasi fungsi untuk mengubah string/angka menjadi binary
(Sumber: arsip pengguna)

```
PS D:\VS code\Analisis-End-to-End> py xor.py
misalkan:
p: 23
q: 5

misalkan:
Private key 1: 10
Private key 2: 9

Combined key 1: 2
Combined key 2: 2

Information: Matdis
```

Gambar 3.6. Permisalan sederhana dari ke-4 value
(Sumber: arsip pengguna)

```
Data dalam binary:
Private key 1: 110001110000
Private key 2: 111001

Combined key 1: 110010
Combined key 2: 110010

Information: 1001101110000111101001100100110100111110011
```

Gambar 3.7. data yang didapat saat diconvert ke binary
(Sumber: arsip pengguna)

```
Informasi yang telah dienkripsi: 00000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000001001101110000
11110100110010011010011000001

Informasi yang telah didekripsi: 00000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000001001101110000
11110100110010011010011110011
```

Gambar 3.8. informasi yang dienkripsi dan didekripsi
(Sumber: arsip pengguna)

Pada contoh di gambar 3.6, gambar 3.7, gambar 3.8, dilakukan permisalan pada nilai p, q, x, dan y. Karena umumnya *key-key* tersebut berukuran 256 bits, maka nilai

terbesar dari *key* tersebut secara decimal adalah $256^2 - 1$. Tentunya dengan nilai sebesar itu maka program akan kesusahan untuk melakukan kalkulasi *key exchange*. Kemungkinan besar untuk menghemat waktu para perusahaan-perusahaan tersebut tidak merandom sampai nilai terbesar dari 256 bit, atau bisa dibilang perusahaan-perusahaan tersebut hanya merandom nilainya sampai *range* tertentu sehingga kalkulasi mereka bisa cepat.

Akan tetapi, kemungkinan besar juga cara yang digunakan perusahaan-perusahaan tersebut bukanlah *Diffie-Hellman key exchange* secara perusahaan-perusahaan tersebut sangatlah besar dan jika hanya mengacak nilai *key*-nya dalam range tertentu tidak akan mencukupi jumlah klien mereka setiap harinya. Kemungkinan besar perusahaan-perusahaan tersebut memiliki caranya masing-masing dalam men-*generate key*-nya sehingga *key* yang didapat masih berbentuk 256 bit dan nilainya lebih acak.

Contoh jika *key* dan informasinya lebih random (gambar 3.9) dibanding dengan contoh sebelumnya (gambar 3.8):

Misalkan bahwa:

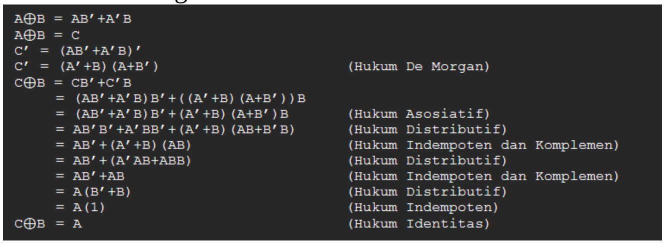
- *Combined key*:
  10110000001000001000100111010110110000010010101001010000100111000111001111001110001011100100000000010101101010011011010011111000010100000001000010100101111010000101101111001110010011100101111110011001000001011111111100011001101011011101111110011000110101011
- Informasi:
  10111110010100001001101101101101110110110010000101111011111011101110011101011010110101110111000101011011101101000011110010000111111101100111001110111011000100101010000010011000100011000011110010100101110100010010011100000100100001111110100011101001010010010

Maka hasil yang didapat:

- Informasi yang telah dienkripsi:
  000011100111000000001001010111011000110100000101100101011010000011011110101111011100000000101000101010001011000000111001101111110111101111001111011111101001111001101000110101111111101010110001001100111100110101001101100000011101001010100011011101110001100111001
- Informasi yang telah didekripsi:
  1011111001010000100110110110110111011011001000010111101111101110111001110101101011010111011100010101101110110100001111001000011111110110011100110111011000100101010000010011000100011000011110010100101110100010010011100000100100001111110100011101001010010010

Gambar 3.9. Pembuktian operasi XOR sebanyak 2 kali dengan nilai yang sama
(Sumber: arsip pengguna)

Bisa dilihat bahwa informasi tentunya berubah saat kita dekripsikan dengan *key* menggunakan operasi XOR, dan jika kita melakukan XOR lagi dengan *key* yang sama maka informasinya akan kembali seperti informasi yang awal. Hal ini bisa dibuktikan di gambar 3.10:



Gambar 3.10. Pembuktian operasi XOR sebanyak 2 kali dengan nilai yang sama
(Sumber: arsip pengguna)

Sehingga terbukti bahwa dengan melakukan operasi XOR sebanyak 2 kali ke data yang sama dapat mengembalikan data tersebut seperti semula. Sehingga operasi XOR ini dapat digunakan pada enkripsi dan dekripsi *End-To-End Encryption*.

V. KESIMPULAN

End-To-End Encryption adalah salah satu dari banyak macam kriptografi yang digunakan perangkat-perangkat sekarang untuk mengirim informasi secara aman. Jika dibandingkan dengan metode lain jelas bahwa *End-To-End Encryption* terkesan lebih aman karena penyedia layanannya tidak memiliki akses secara langsung terhadap informasi yang dikirim atau diterima pengguna-pengguna mereka. Tentunya terdapat metode lain untuk men-*generate key* yang akan digunakan agar *key*-nya lebih random dan lebih banyak variasinya. Namun untuk kebutuhan skala kecil, penggunaan metode *Diffie-Hellman key exchange* yang menggunakan *multiplicative group of integers modulo* sudah tergolong cukup.

VI. LAMPIRAN

Source code yang digunakan untuk melakukan operasi XOR: https://github.com/Ainzw0rth/Analisis-End-to-End.git

VII. UCAPAN TERIMA KASIH

Terima kasih kepada Tuhan Yang Maha Esa karena berkat rahmat dan kasih karunia-Nya penulis dapat menyelesaikan makalah kali ini dengan baik dan tanpa kendala yang terlalu berat. Tak lupa juga penulis ingin mengucapkan terima kasih

kepada keluarga yang telah memberikan dukungan sehingga penulis dapat menyelesaikan makalah ini. Terima kasih juga penulis sampaikan kepada dosen-dosen pengampuh mata kuliah IF2120 terutama kepada Dr. Nur Ulfa Maulidevi, S.T., M. Sc. selaku dosen pengampuh mata kuliah IF2120 untuk kelas 01 karena telah memberikan pengetahuan yang dapat digunakan penulis untuk menulis makalah ini. Penulis berharap bahwa makalah ini nanti kedepannya dapat digunakan sebagai referensi baik bagi para pelajar yang penasaran dengan ilmunya atau bahkan sebagai referensi penulis kedepannya.

REFERENSI

[1] Ralph C. Merkle. 1978. *Secure communications over insecure channels, Communications of the ACM*. https://dl.acm.org/doi/abs/10.1145/359460.359473. Diakses pada 10 Desember 2022.

[2] Munir, Rinaldi. 2020. "Aljabar Boolean (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Aljabar- Boolean-(2020)-bagian1.pdf , diakses 11 Desember 2022.

[3] Munir, Rinaldi. 2020. "Teori Bilangan (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian1.pdf , diakses 11 Desember 2022.

[4] Greenberg, Andy. 2014. "Hacker Lexicon: What Is End-to-End Encryption?". https://www.wired.com/2014/11/hacker-lexicon-end-to-end-encryption/. Diakses pada 10 Desember 2022.

[5] Nasser, Hussein. 2020, 14 April. How End-to-End encryption works?. *Youtube*. https://youtu.be/hwQbPgvEQyw.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Desember 2022

Louis Caesa Kesuma 13521069