Implementasi Algoritma Brute Force untuk Mengeksploitasi Kelemahan Keamanan pada Metode Enkripsi AES-ECB

Ahmad Naufal Ramadan - 13522005
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13522005@std.stei.itb.ac.id

***Abstract*—Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berkaitan dengan keamanan informasi, termasuk kerahasiaan, integritas data, dan otentikasi. Sejak zaman kuno, seperti penggunaan sandi Caesar oleh bangsa Romawi, kriptografi telah berperan penting dalam melindungi informasi sensitif. Dengan perkembangan teknologi, algoritma enkripsi modern seperti AES dan RSA menjadi vital dalam menjaga keamanan data di era digital. Enkripsi modern seperti RSA dan AES tidak sepenuhnya kebal terhadap serangan brute force, terutama dengan kemajuan teknologi komputasi, termasuk komputasi kuantum. Makalah ini akan membahas kelemahan metode enkripsi AES-ECB yang rentan terhadap serangan brute force.**

Kata Kunci—Kriptografi, enkripsi, brute force, AES-ECB.

I. PENDAHULUAN

Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi. Kriptografi berperan penting dalam melindungi informasi sensitif. Penggunaan kriptografi sudah dilakukan sejak peradaban kuno dengan menggunakan sandi substitusi. Pada masa itu, kriptografi digunakan untuk menjaga kerahasiaan pesan yang dikirim antar pihak berwenang. Sebagai contoh, bangsa Romawi menggunakan sandi Caesar untuk mengamankan komunikasi militer mereka. Dengan perkembangan teknologi, pengembangan teknik enkripsi yang lebih canggih diperlukan untuk memperkuat perlindungan informasi berharga. Algoritma enkripsi modern seperti AES (Advanced Encryption Standard) dan RSA menjadi sangat penting dalam menjaga keamanan data di era digital ini.

Seiring dengan kemajuan teknologi, penggunaan kriptografi menjadi semakin signifikan. Dengan banyaknya data yang ditransmisikan melalui jaringan semakin meningkat, potensi intersepsi dan akses yang tidak sah semakin besar. Teknologi modern seperti perbankan online, komunikasi seluler, dan transaksi e-commerce sangat bergantung pada kriptografi untuk menjaga keamanan dan privasi pengguna. Algoritma enkripsi yang kompleks memastikan bahwa data yang dikirim melalui jaringan tidak dapat dibaca oleh pihak yang tidak berwenang. Namun, ancaman dari para penyerang siber terus berkembang, sehingga penting bagi para ahli kriptografi untuk terus mengembangkan dan memperbarui metode enkripsi mereka.

Pemecahan kriptografi dengan metode brute force merupakan teknik yang telah ada sejak lama. Metode ini sudah pasti dapat digunakan untuk mendekripsi pesan terenkripsi tanpa mengetahui kuncinya. Pada enkripsi kuno seperti sandi Caesar atau sandi substitusi lainnya, metode brute force dilakukan dengan mencoba setiap kemungkinan kunci hingga menemukan yang benar. Misalnya, dalam sandi Caesar yang hanya memiliki 25 kemungkinan kunci, brute force relatif mudah dilakukan. Namun, metode brute force tidak hanya terbatas pada enkripsi kuno. Selama Perang Dunia II, mesin Enigma yang digunakan oleh Nazi Jerman juga berhasil dipecahkan oleh para kriptanalisis Sekutu dengan bantuan metode brute force dan kriptanalisis.

Enkripsi modern seperti RSA dan AES, meskipun jauh lebih kompleks dan aman dibandingkan dengan metode kuno, juga tidak sepenuhnya kebal terhadap serangan brute force. RSA, misalnya, bergantung pada kesulitan pemfaktoran bilangan besar, namun dengan kekuatan komputasi yang cukup besar, pemecahan brute force masih mungkin dilakukan, meski memerlukan waktu yang sangat lama. Demikian pula, AES dengan kunci 128-bit atau 256-bit akan membutuhkan waktu yang luar biasa lama untuk dipecahkan melalui brute force dengan teknologi saat ini. Namun, perkembangan teknologi komputasi, khususnya komputasi kuantum, berpotensi mengurangi waktu yang dibutuhkan untuk brute force secara signifikan.

Dalam makalah ini, penulis akan membahas kelemahan metode enkripsi AES-ECB yang dapat dieksploitasi dengan mudah dengan menggunakan metode brute force. Penulis akan menjelaskan bagaimana cara kerja AES-ECB dan apa yang membuatnya rentan dengan pemecahan metode brute force. Dengan ini, diharapkan pembaca dapat mengetahui kelemahan metode enkripsi AES-ECB dan menghindari penggunaannya untuk menjaga rahasia penting.

II. DASAR TEORI

A. *Kriptografi*

Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi. Ilmu ini mencakup berbagai metode dan algoritma yang dirancang untuk melindungi informasi dari akses yang tidak sah. Sejak zaman kuno, teknik kriptografi seperti sandi Caesar dan substitusi telah digunakan untuk menjaga kerahasiaan pesan. Dengan perkembangan teknologi, teknik-teknik ini telah berevolusi menjadi algoritma yang lebih kompleks seperti AES (Advanced Encryption Standard) dan RSA, yang mampu memberikan tingkat keamanan yang lebih tinggi untuk melindungi data dalam era digital.

Kriptografi modern tidak hanya berfokus pada kerahasiaan, tetapi juga mencakup aspek lain seperti integritas data dan otentikasi. Integritas data memastikan bahwa informasi tidak diubah atau dirusak selama transmisi, sementara otentikasi memungkinkan verifikasi identitas pihak yang berkomunikasi. Metode seperti hashing dan tanda tangan digital digunakan untuk mencapai tujuan ini. Dalam konteks keamanan jaringan dan komunikasi, kriptografi memainkan peran penting dalam protokol seperti SSL/TLS yang digunakan untuk mengamankan koneksi internet.

B. *Kriptografi Simetris dan Asimetris*

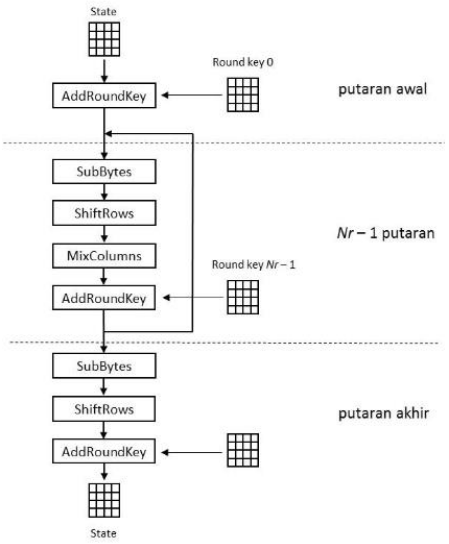
Kriptografi simetris adalah metode enkripsi di mana pengirim dan penerima menggunakan kunci yang sama untuk mengenkripsi dan mendekripsi pesan. Algoritma enkripsi simetris, seperti AES (Advanced Encryption Standard) dan DES (Data Encryption Standard), dikenal karena kecepatan dan efisiensinya dalam memproses data. Karena hanya satu kunci yang digunakan, menjaga kerahasiaan kunci ini sangat penting. Jika kunci tersebut jatuh ke tangan yang salah, semua data yang dilindungi oleh kunci tersebut dapat diakses oleh pihak yang tidak berwenang. Kriptografi simetris biasanya digunakan untuk mengenkripsi data dalam jumlah besar karena kinerjanya yang lebih cepat dibandingkan dengan metode asimetris.

Kriptografi asimetris adalah metode enkripsi yang menggunakan sepasang kunci, yaitu kunci publik dan kunci privat. Kunci publik digunakan untuk mengenkripsi data, sedangkan kunci privat digunakan untuk mendekripsi data. Contoh algoritma enkripsi asimetris adalah RSA (Rivest-Shamir-Adleman) dan ECC (Elliptic Curve Cryptography). Keunggulan utama kriptografi asimetris adalah kemampuan untuk mendistribusikan kunci publik secara bebas tanpa membahayakan keamanan data, karena hanya kunci privat yang dapat digunakan untuk mendekripsi pesan yang dienkripsi dengan kunci publik tersebut. Kriptografi asimetris sangat berguna untuk aplikasi seperti pertukaran kunci, tanda tangan digital, dan otentikasi, di mana keamanan komunikasi dan verifikasi identitas sangat penting. Namun, kriptografi asimetris umumnya lebih lambat dibandingkan dengan kriptografi simetris, sehingga sering digunakan dalam kombinasi dengan metode simetris untuk mencapai keseimbangan antara keamanan dan kinerja.

C. *Advance Encryption Standard (AES)*

Advanced Encryption Standard (AES) adalah algoritma enkripsi simetris yang paling banyak digunakan dalam berbagai aplikasi, mulai dari komunikasi online hingga penyimpanan data sensitif. AES menggunakan blok enkripsi dengan panjang tetap 128 bit dan mendukung tiga panjang kunci yang berbeda: 128, 192, dan 256 bit. Kelebihan AES terletak pada tingkat keamanan yang tinggi dan efisiensi dalam pemrosesan data, menjadikannya pilihan utama untuk mengamankan data di berbagai platform dan perangkat.

Proses enkripsi AES terdiri dari beberapa langkah yang diulang secara berurutan dalam putaran. Setiap putaran melibatkan serangkaian operasi yang dirancang untuk mengubah data masukan menjadi ciphertext yang tidak dapat dibaca tanpa kunci yang sesuai. Langkah-langkah ini termasuk SubBytes, yang menggantikan byte dalam blok dengan nilai dari tabel substitusi; ShiftRows, yang menggeser baris dalam blok; MixColumns, yang menggabungkan kolom dalam blok; dan AddRoundKey, yang menambahkan kunci putaran ke blok. Proses ini diulang sejumlah putaran tertentu tergantung pada panjang kunci yang digunakan, dengan 10 putaran untuk kunci 128-bit, 12 putaran untuk kunci 192-bit, dan 14 putaran untuk kunci 256-bit.



Gambar 1. Ilustrasi metode enkripsi AES, diambil dari [3]

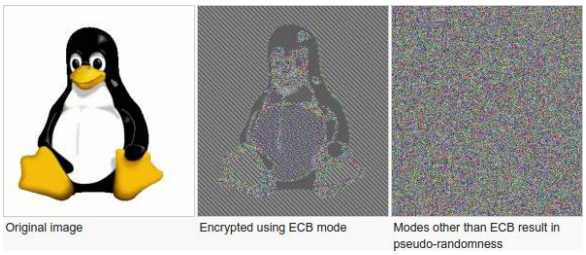
Salah satu keunggulan AES adalah ketahanannya terhadap serangan kriptanalisis yang umum, seperti serangan differential dan linear cryptanalysis. Selain itu, AES juga dapat diimplementasikan dengan efisien dalam perangkat keras dan perangkat lunak yang berbeda, memungkinkan penggunaannya dalam berbagai platform. Meskipun demikian, perkembangan dalam komputasi kuantum menimbulkan pertanyaan tentang keamanan AES di masa depan, karena komputer kuantum dapat dengan cepat memecahkan masalah

matematika yang rumit yang mendasari keamanan algoritma ini.

D. AES-ECB

Advanced Encryption Standard-Electronic Codebook (AES-ECB) adalah mode operasi sederhana dari algoritma enkripsi AES yang mengenkripsi setiap blok data secara terpisah dengan kunci yang sama. Meskipun sederhana dan mudah diimplementasikan, AES-ECB memiliki kelemahan yang signifikan dalam konteks keamanan. Kelemahan utama AES-ECB adalah bahwa blok-blok plaintext yang sama akan dienkripsi menjadi blok-blok ciphertext yang identik. Hal ini menyebabkan pola yang dapat dideteksi dalam ciphertext, yang dapat dimanfaatkan oleh penyerang untuk menganalisis dan mendapatkan informasi tentang data asli tanpa memerlukan kunci enkripsi.

Salah satu contoh konkret dari kelemahan AES-ECB adalah pada gambar. Jika sebuah gambar dienkripsi dengan AES-ECB, blok-blok yang mewakili warna yang sama atau pola yang berulang dalam gambar akan dienkripsi menjadi blok-blok ciphertext yang sama. Sebagai hasilnya, pola tersebut akan tetap terlihat dalam gambar yang dienkripsi meskipun pesan sebenarnya telah dienkripsi. Hal ini dapat dimanfaatkan oleh penyerang untuk menganalisis gambar dan mendapatkan informasi sensitif tentang isinya.



Gambar 2. Perbedaan enkripsi gambar menggunakan algoritma AES-ECB dan selain AES-ECB, diambil dari https://www.educative.io/answers/what-is-ecb

Untuk mengatasi kelemahan AES-ECB, disarankan untuk menggunakan mode operasi AES yang lebih aman, seperti Cipher Block Chaining (CBC) atau Counter (CTR). Mode-mode ini memperkenalkan tingkat kompleksitas tambahan dengan cara melibatkan blok-blok sebelumnya dalam proses enkripsi, sehingga setiap blok ciphertext tidak hanya tergantung pada blok plaintext yang sesuai, tetapi juga blok-blok sebelumnya. Dengan demikian, mode-mode ini menghilangkan pola yang dapat dideteksi dalam ciphertext dan meningkatkan keamanan enkripsi secara keseluruhan.

E. Brute Force

Algoritma brute force adalah pendekatan sederhana namun kuat untuk menemukan solusi dari masalah yang kompleks dengan mencoba semua kemungkinan secara berurutan. Meskipun metode ini dapat selalu menemukan solusi dalam suatu persoalan, namun untuk masalah dengan ruang pencarian yang sangat besar, brute force menjadi tidak praktis karena memerlukan waktu dan sumber daya komputasi yang sangat besar.

Keunggulan utama dari algoritma brute force adalah algoritma brute force sangat sederhana. Hal ini membuatnya dapat diterapkan pada berbagai jenis masalah tanpa perlu penyesuaian yang kompleks. Kelemahannya adalah keefektifan yang rendah, terutama dalam konteks masalah dengan ruang pencarian yang besar. Untuk mengatasi kelemahan ini, seringkali digunakan pendekatan lain yang lebih cerdas, seperti algoritma pencarian heuristik atau algoritma optimasi, yang dapat menemukan solusi dengan lebih efisien dengan memperhitungkan struktur masalah yang spesifik.

Dalam kriptografi, keamanan dari suatu sistem enkripsi seringkali diukur dari seberapa sulitnya untuk meretas kunci enkripsinya dengan menggunakan brute force. Semakin panjang kunci enkripsi dan semakin kompleks algoritma enkripsinya, semakin sulit bagi penyerang untuk memecahkan kunci tersebut dengan brute force. Oleh karena itu, pengembangan algoritma enkripsi yang kuat dan aman dari serangan brute force menjadi sangat penting dalam memastikan keamanan data dalam berbagai aplikasi dan sistem komputer.

F. Serangan Terhadap Kriptografi

Serangan terhadap kriptografi merupakan upaya yang dilakukan oleh pihak yang tidak berwenang, yang disebut kriptanalis, untuk mengungkapkan informasi rahasia yang dilindungi oleh enkripsi. Tujuan utama dari serangan ini adalah untuk mendapatkan akses ke pesan asli (plainteks) atau kunci enkripsi yang digunakan untuk melakukan enkripsi. Kriptanalis bisa saja merangkap sebagai penyadap (eavesdropper) yang hanya memantau komunikasi tanpa mengubahnya, atau sebagai pihak yang aktif terlibat dalam proses komunikasi untuk memanipulasi pesan.

Serangan terhadap kriptografi dapat diklasifikasikan berdasarkan keterlibatan penyerang dalam komunikasi pesan. Serangan pasif terjadi ketika penyerang hanya melakukan penyadapan data atau informasi yang dikirim antara pengirim dan penerima. Di sisi lain, serangan aktif melibatkan intervensi langsung penyerang dalam komunikasi. Penyerang dapat mengubah, menghapus, atau menyisipkan informasi dalam pesan yang dikirimkan.

Salah satu contoh serangan aktif yang cukup umum adalah serangan man-in-the-middle. Dalam serangan ini, penyerang menempati posisi antara pengirim dan penerima yang seharusnya berkomunikasi langsung. Dengan demikian, penyerang dapat memantau dan bahkan memanipulasi pesan yang dikirimkan antara kedua pihak tersebut. Serangan ini memungkinkan penyerang untuk membaca pesan rahasia, mengubahnya, atau bahkan menyisipkan pesan palsu tanpa pengetahuan kedua belah pihak yang sebenarnya berkomunikasi.

Serangan terhadap kriptografi juga dapat dibagi berdasarkan teknik yang digunakan oleh penyerang untuk mendapatkan informasi rahasia. Pertama, serangan Chipertext-only melibatkan penyerang yang hanya memiliki cipherteks tanpa akses ke plainteks asli. Teknik yang digunakan untuk serangan ini meliputi pencarian kunci secara eksaustif, menebak-nebak kunci, dan analisis frekuensi karakter dalam cipherteks. Selanjutnya adalah serangan Known-plaintext. Dalam serangan ini, penyerang memiliki pasangan plainteks dan cipherteks yang berkoresponden, sehingga dia dapat mendeduksi kunci enkripsi dengan membandingkan keduanya. Selanjutnya, serangan Chosen-plaintext memungkinkan penyerang untuk memilih plainteks tertentu untuk dienkripsi yang dapat membantu dalam menemukan kunci enkripsi. Lalu ada juga serangan Adaptive-chosen-plaintext. Serangan Adaptive-chosen-plaintext melibatkan penyerang yang memilih blok plainteks besar, kemudian mengamati hasil enkripsi untuk memilih blok plainteks yang lebih kecil, dan begitu seterusnya. Terakhir, serangan Chosen-ciphertext melibatkan penyerang yang memiliki akses ke cipherteks dan dapat mendeduksi kunci yang digunakan untuk enkripsi.

Sebuah algoritma kriptografi dianggap aman secara komputasi jika algoritma tersebut sangat kompleks secara matematis sehingga tidak mungkin dipecahkan secara analitis, biaya untuk memecahkan cipherteks melebihi nilai informasi yang terkandung di dalamnya, dan waktu yang diperlukan untuk memecahkan cipherteks melebihi batas waktu informasi harus dijaga kerahasiaannya.

III. Implementasi dan Pengujian

Dalam implementasi program, dibutuhkan sebuah service enkripsi AES-EBC yang memiliki celah dalam keamanan. Eksploitasi enkripsi AES-EBC membutuhkan sebuah service yang dapat mengenkripsi banyak plaintext dengan key yang sama. Service tersebut akan mengembalikan ciphertext yang berasal dari plaintext ditambah dengan rahasia yang ingin dipecahkan. Berikut adalah implementasi service yang diimplementasikan.

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from flask import Flask
import os

app = Flask(__name__)

KEY = os.urandom(16)
FLAG = "ini_pesan_rahasia"


@app.route("/encrypt/<plaintext>/")
def encrypt(plaintext):
  plaintext = bytes.fromhex(plaintext)
  padded = pad(plaintext + FLAG.encode(), 16)
  cipher = AES.new(KEY, AES.MODE_ECB)
  try:
    encrypted = cipher.encrypt(padded)
  except ValueError as e:
    return {"error": str(e)}

  return {"ciphertext": encrypted.hex()}


if __name__ == "__main__":
  app.run(debug=True)
```

Program ini merupakan aplikasi web sederhana yang menggunakan library `Crypto` untuk melakukan enkripsi AES dalam mode ECB. Aplikasi ini memiliki endpoint `/encrypt/<plaintext>/` yang dapat menerima plaintext dalam format hexadecimal. Pertama, program menginisialisasi kunci acak sepanjang 16 byte dan sebuah pesan rahasia. Ketika plaintext diterima, program melakukan padding pada plaintext dan menggabungkannya dengan pesan rahasia. Selanjutnya, program membuat objek AES dengan mode ECB dan kunci yang telah diinisialisasi sebelumnya. Selanjutnya, dilakukan enkripsi pada plaintext yang telah dipad, dan hasilnya dikembalikan dalam format hexadecimal.

Server mengembalikan aes_ebc(input + flag). Kita dapat memanipulasi input untuk mencari flag. EBC membagi input menjadi blok-blok dan mengenkripsi setiap blok dengan cara yang sama. Misalkan flag adalah crypto{secret_flag}. Jika input kita adalah "A" * 16, maka blok-bloknya akan menjadi

```
AAAAAAAAAAAAAAAA
crypto{secret_fl
ag}+padding
```

Untuk menebak huruf pertama dari flag, kita dapat membuat input kita "A" * 15 + tebakan + "A" * 15. Jika tebakan kita adalah "b", maka blok-bloknya akan menjadi

```
AAAAAAAAAAAAAAAb
AAAAAAAAAAAAAAAc
rypto{secret_fla
g}+padding
```

Kemudian, kita bisa membandingkan blok-blok keluaran dari dua input. Jika blok-blok keluaran dari dua input sama, maka tebakan benar, begitu juga sebaliknya. Kedua blok tersebut tidak sama, maka huruf pertama dari flag bukan "b". Jika kita menebak "c", maka blok-bloknya akan menjadi

```
AAAAAAAAAAAAAAAc
AAAAAAAAAAAAAAAc
rypto{secret_fla
g}
```

Blok-blok keluaran dari dua input tersebut akan cocok, sehingga kita tahu "c" adalah huruf pertama dari flag. Kemudian, kita bisa mencoba menebak huruf berikutnya. Kita tahu huruf pertama adalah "c" jadi untuk melihat apakah huruf

kedua adalah "a" kita akan membuat input kita "A" * 14 + "ca" + "A" * 14. Ini akan membuat blok-bloknya menjadi

```
AAAAAAAAAAAAAAca
AAAAAAAAAAAAAAcr
ypto{secret_flag
}
```

Sekali lagi, kita akan melihat bahwa kedua blok berbeda. Kita dapat melakukan iterasi melalui setiap karakter yang mungkin hingga ditemukan blok yang sama, kemudian beralih ke huruf ketiga dari flag dan seterusnya sampai flag berhasil ditemukan.

Dengan pola tersebut, akan sangat mudah untuk menemukan rahasia yang disimpan dari program. Dengan menggunakan pendekatan brute force dan scripting Python, rahasia tersebut dapat ditemukan dengan relatif mudah. Berikut adalah implementasi eksploitasi yang dapat dilakukan untuk menemukan flag.

```
import requests
from string import printable
from colorama import Fore, Style


def encrypt(plaintext):
  plain_hex = plaintext.encode().hex()
  url = "http://127.0.0.1:5000/encrypt/" + plain_hex
  r = requests.get(url)
  r_data = r.json()
  return r_data.get("ciphertext", None)


def pad_flag_guess(guess):
  padding = "A" * (16 - len(guess) % 16)
  padded_guess = padding + guess + padding

  return padded_guess


if __name__ == "__main__":
  flag = ""

  counter = 0
  while True:
    for l in printable:
      counter += 1
      flag_guess = flag + l
      padded_guess = pad_flag_guess(flag_guess)
      ciphertext = encrypt(padded_guess)
      guess_output_size = 2 * ((16 - len(flag_guess) % 16) + len(flag_guess))

      encrypted_guess = ciphertext[:guess_output_size]
      encrypted_flag = ciphertext[guess_output_size : guess_output_size * 2]

      print("Ciphertext:")
      for i in range(0, len(ciphertext), guess_output_size):
        cipher_block = ciphertext[i : i + guess_output_size]
        if cipher_block == encrypted_flag and cipher_block == encrypted_guess:
          for j in range(0, len(cipher_block), 32):
            print(Fore.YELLOW + cipher_block[j : j + 32])
        elif cipher_block == encrypted_flag:
          for j in range(0, len(cipher_block), 32):
            print(Fore.RED + cipher_block[j : j + 32])
        elif cipher_block == encrypted_guess:
          for j in range(0, len(cipher_block), 32):
            print(Fore.GREEN + cipher_block[j : j + 32])
        else:
            print(Fore.BLUE + cipher_block)
            print(Style.RESET_ALL, end="")

      print(f"Padded guess: {padded_guess}")
      print(f"Guessed character: {l}")
      if encrypted_guess == encrypted_flag:
        flag = flag_guess
      print(f"Flag: {flag}")
      print()
      break

    print()
    else:
      break

  print(f"Total requests: {counter}")
  print(f"Flag: {flag}")
```

Script ini memanfaatkan kelemahan pada metode enkripsi AES dalam mode ECB yang dapat dieksploitasi jika kita memiliki akses untuk mengenkripsi teks yang kita pilih sendiri.

Pertama, script ini menggunakan sebuah fungsi encrypt(plaintext) untuk mengirimkan teks yang ingin kita enkripsi ke server. Server akan mengembalikan teks yang sudah dienkripsi. Fungsi ini digunakan untuk mendapatkan hasil enkripsi dari teks yang kita pilih sendiri.

Selanjutnya, script memiliki fungsi pad_flag_guess(guess) yang digunakan untuk menambahkan karakter "A" ke teks yang ingin kita tebak sehingga sesuai dengan analisis yang telah dilakukan sebelumnya.

Dalam bagian utama script, script ini menggunakan perulangan while untuk menebak setiap karakter dari flag. Script akan mencoba semua karakter yang dapat dicetak (printable) satu per satu. Setelah menebak sebuah karakter, script akan mengenkripsi tebakan tersebut dan membandingkannya dengan hasil enkripsi dari teks yang sudah kita ketahui sejauh ini dan teks yang kita tebak sebelumnya. Jika hasil enkripsinya sama, maka karakter tersebut adalah bagian dari pesan rahasia. Proses ini diulangi untuk setiap karakter hingga seluruh pesan rahasia terungkap atau semua karakter yang dapat dicetak telah dicoba. Script akan mencetak pesan rahasia yang berhasil ditebak beserta jumlah total percobaan yang dilakukan untuk menebak pesan tersebut.

```
@ Ahmad Naufal Ramadan on ▸ C:/Projects/aes-ebc-bruteforce-attack
# python .\attack.py
Ciphertext:
87fc16af9fdb81cf1cfd9bdbac4badc0
02f561cc90a226c54e54d5653fa4e4c9
98d7649ef9047d291abefabbc3e01dfb
2bdc2d2cc0037f8535889beb50d2f1d9
Padded guess: AAAAAAAAAAAAAAA0AAAAAAAAAAAAAAA
Guessed character: 0

Ciphertext:
fe920d1478eaa50b883aaaddf9975849
02f561cc90a226c54e54d5653fa4e4c9
98d7649ef9047d291abefabbc3e01dfb
2bdc2d2cc0037f8535889beb50d2f1d9
Padded guess: AAAAAAAAAAAAAAA1AAAAAAAAAAAAAAA
Guessed character: 1

Ciphertext:
438edb780c8d1874095294d8808e0d65
02f561cc90a226c54e54d5653fa4e4c9
98d7649ef9047d291abefabbc3e01dfb
2bdc2d2cc0037f8535889beb50d2f1d9
Padded guess: AAAAAAAAAAAAAAA2AAAAAAAAAAAAAAA
Guessed character: 2
```

Gambar 3. Simulasi beberapa percobaan pertama, diambil dari sumber pribadi

```
Ciphertext:
02f561cc90a226c54e54d5653fa4e4c9
02f561cc90a226c54e54d5653fa4e4c9
98d7649ef9047d291abefabbc3e01dfb
2bdc2d2cc0037f8535889beb50d2f1d9
Padded guess: AAAAAAAAAAAAAAAiAAAAAAAAAAAAAAA
Guessed character: i
Flag: i

Ciphertext:
389e5e70dcd4c508f30378fdeb6b1314
a30c561f1703f02fe2fce5f63bf1078f
323a3860b3619c102ab72cc9cba1fd92
Padded guess: AAAAAAAAAAAAAAi0AAAAAAAAAAAAAAA
Guessed character: 0

Ciphertext:
087d38020d1551520617d28356a57817
a30c561f1703f02fe2fce5f63bf1078f
323a3860b3619c102ab72cc9cba1fd92
Padded guess: AAAAAAAAAAAAAAi1AAAAAAAAAAAAAAA
Guessed character: 1
```

Gambar 4. Karakter pertama flag yang berhasil ditemukan, diambil dari sumber pribadi

```
Ciphertext:
02f561cc90a226c54e54d5653fa4e4c9
98d7649ef9047d291abefabbc3e01dfb
02f561cc90a226c54e54d5653fa4e4c9
98d7649ef9047d291abefabbc3e01dfb
2bdc2d2cc0037f8535889beb50d2f1d9
Padded guess: AAAAAAAAAAAAAAAini_pesan_rahasiaAAAAAAAAAAAAAAA
Guessed character: a
Flag: ini_pesan_rahasia

Ciphertext:
a30c561f1703f02fe2fce5f63bf1078f
2c1b38d70e9b25ef2a4ec517d522871d
a30c561f1703f02fe2fce5f63bf1078f
323a3860b3619c102ab72cc9cba1fd92
Padded guess: AAAAAAAAAAAAAAini_pesan_rahasia0AAAAAAAAAAAAAA
Guessed character: 0

Ciphertext:
a30c561f1703f02fe2fce5f63bf1078f
e784778d4d1bcac9565c97e175845e60
a30c561f1703f02fe2fce5f63bf1078f
323a3860b3619c102ab72cc9cba1fd92
Padded guess: AAAAAAAAAAAAAAini_pesan_rahasia1AAAAAAAAAAAAAA
Guessed character: 1
```

Gambar 5. Karakter terakhir flag yang berhasil ditemukan, diambil dari sumber pribadi

```
Ciphertext:
a30c561f1703f02fe2fce5f63bf1078f
af146fa9d8439ee9ee4082d244468cb0
a30c561f1703f02fe2fce5f63bf1078f
323a3860b3619c102ab72cc9cba1fd92
Padded guess: AAAAAAAAAAAAAAini_pesan_rahasia
AAAAAAAAAAAAAA
Guessed character:


Ciphertext:
a30c561f1703f02fe2fce5f63bf1078f
49399d806df767178309eadbe3153d51
a30c561f1703f02fe2fce5f63bf1078f
323a3860b3619c102ab72cc9cba1fd92
Padded guess: AAAAAAAAAAAAAAini_pesan_rahasia
AAAAAAAAAAAAAA
Guessed character:


Total requests: 572
Flag: ini_pesan_rahasia
```

Gambar 6. Tampilan akhir program, diambil dari sumber pribadi

Pemberian warna pada hasil enkripsi digunakan untuk memudahkan pemahaman. Warna hijau adalah blok pertama yang ingin dibandingkan, yaitu blok yang berisi chosen-plaintext. Warna hijau adalah blok kedua yang ingin dibandingkan, yaitu blok yang berisi flag asli. Warna kuning menandakan bahwa kedua blok tersebut sama, yaitu ketika berhasil menebak karakter flag.

Algoritma brute force yang digunakan pada program akan membandingkan setiap karakter pada printable untuk semua karakter pada flag. Jika panjang flag direpresentasikan dengan n dan banyak karakter pembanding di printable direpresentasikan dengan m, maka kompleksitas algoritma dari eksploitasi ini adalah O(n*m). Kompleksitas algoritma ini dapat meningkat secara signifikan seiring meningkatnya panjang flag atau jumlah karakter pembanding. Dalam kasus-kasus di mana panjang flag sangat besar atau jumlah karakter pembanding sangat banyak, algoritma brute force seperti ini kurang praktis karena memerlukan waktu yang sangat lama untuk menemukan flag yang benar.

IV. Kesimpulan

Enkripsi AES dalam mode ECB memiliki celah keamanan yang dikenal karena sifatnya yang deterministik. Ini berarti bahwa jika plaintext yang sama dienkripsi dua kali dengan key yang sama, hasil enkripsinya akan sama. Pada implementasi tertentu, jika plaintext memiliki pola tertentu atau jika serangkaian plaintext yang berbeda dienkripsi dengan key yang sama, serangkaian blok ciphertext yang sama akan dihasilkan.

Eksploitasi enkripsi AES-EBC mungkin terjadi ketika terdapat service yang memungkinkan penggunaan key yang sama untuk mengenkripsi banyak plaintext. Dalam kasus ini, penyerang dapat menggunakan pendekatan brute force, yaitu mencoba semua kemungkinan kombinasi plaintext untuk menemukan pola yang sesuai dengan ciphertext yang diberikan. Dengan menggunakan scripting Python atau bahasa pemrograman lainnya, penyerang dapat secara otomatis memanipulasi input dan membandingkan blok-blok enkripsi dengan pola yang sudah diketahui, seperti pada contoh sebelumnya.

Meskipun algoritma brute force ini dapat berhasil dalam menemukan rahasia, namun keefektifannya tergantung pada kompleksitas rahasia dan kecepatan komputasi yang tersedia. Dalam kasus di mana panjang rahasia sangat besar atau jumlah karakter pembanding sangat banyak, algoritma ini mungkin tidak praktis karena memerlukan waktu yang sangat lama.

Dengan pemahaman akan kelemahan metode enkripsi AES-ECB, diharapkan pembaca dapat membuat keputusan yang lebih baik dalam menjaga keamanan informasi sensitif. Hindari menggunakan metode enkripsi ECB untuk data yang memerlukan tingkat keamanan tinggi, terutama jika plaintextnya dapat diprediksi atau memiliki pola tertentu. Sebagai gantinya, pertimbangkan penggunaan mode enkripsi yang lebih aman seperti CBC (Cipher Block Chaining) atau GCM (Galois/Counter Mode) yang menawarkan keamanan tambahan melalui pengacakan (IV - Initialization Vector) dan autentikasi. Dengan demikian, keamanan data yang dienkripsi dapat lebih terjaga dari serangan brute force dan manipulasi data yang tidak sah.

Ucapan Terima Kasih

Dengan rendah hati, penulis ingin mengucapkan terima kasih kepada semua pihak yang telah membantu dalam penyelesaian proyek ini. Pertama-tama, terima kasih kepada Allah SWT. atas segala berkat dan rahmat-Nya yang melimpah selama proses ini. Penulis juga ingin menyampaikan terima kasih kepada dosen pengampu mata kuliah Strategi Algoritma IF2211, yaitu Bapak Dr. Rinaldi Munir, yang telah memberikan ilmu yang dapat dipahami dengan baik oleh penulis. Penulis juga ingin menyampaikan terima kasih kepada teman-teman dan keluarga yang telah memberikan dukungan dan motivasi selama proses ini berlangsung. Semua kontribusi dan dukungan yang diberikan sangat berarti bagi penulis. Terima kasih atas kerjasama dan dedikasi yang telah ditunjukkan.

Referensi

[1] R. Munir, 'IF4020 Kriptografi · Semester II Tahun 2023/2024', https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/01-Pengantar-Kriptografi-(2024).pdf
Diakses pada 11 Juni 2024

[2] R. Munir, 'IF4020 Kriptografi · Semester II Tahun 2023/2024', https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/06-Serangan-pada-kriptografi-(2024).pdf
Diakses pada 11 Juni 2024

[3] R. Munir, 'IF4020 Kriptografi · Semester II Tahun 2023/2024', https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/15-Beberapa-block-cipher-bagian2-2024.pdf
Diakses pada 11 Juni 2024

[4] R. Munir, 'IF2211 Strategi Algoritma · Semester II Tahun 2023/2024', https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf
Diakses pada 11 Juni 2024

[5] Cryptohack, 'Symmetric Cryptography Course', https://cryptohack.org/courses/symmetric/course_details/
Diakses pada 11 Juni 2024

[6] Repository Github penulis, 'aes-ecb-bruteforce-attack', https://github.com/SandWithCheese/aes-ecb-bruteforce-attack

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, 12 Juni 2024

Ahmad Naufal Ramadan, 13522005