Aplikasi penggunaan Hash pada file rahasia bidang tender proyek

Muhammad Alfandavi Aryo Utomo - 13519211
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13519211@std.stei.itb.ac.id

***Abstract*— Kerahasiaan tender proyek merupakan hal yang sangat penting bagi perusahaan, karena dari data tersebut bisa dipastikan kemenangan perusahaan tersebut dari saingannya dalam aspek penawaran. Demi menjamin keamanan dan keaslian data tersebut peran kriptografi sangatlah diperlukan, berupa penguncian file, hashing, modifikasi binary. Alternatif ini merupakan langkah pertama dalam melakukan pengamanan data lebih lanjut nantinya.**

Keywords—tender proyek; kriptografi; hash

I. PENDAHULUAN

Indonesia merupakan negara berkembang dengan tingkat pembangunan yang terbilang cukup tinggi, terlihat dari rencana, anggaran pembangunan di penjuru negeri yang sedang gencar-gencarnya dilakukan terutama saat masa kepemimpinan bapak Presiden Ir.Joko Widodo. Peningkatan jumlah pembangunan ini diiringi dengan meningkatnya jumlah pemain baru di bidang penyedia jasa pembangunan (atau yang bisa kita sebut perusahaan sektor infrastruktur), karena semakin meningkatnya permintaan hal ini mendorong penawaran meningkat juga untuk mengimbanginya. Proses pemilihan siapa yang mengambil proyek ditentukan dengan diadakannya tender proyek berupa bid penawaran dan permintaan antara penyedia proyek beserta kumpulan perusahaan yang berminat untuk mengambil proyek. Data penawaran dari masing-masing perusahaan akan dikirimkan kepada pihak penyedia proyek, kemudian akan dipilih yang memiliki prospek dan beberapa pertimbangan lainnya, data tersebut biasanya berisi biaya, waktu, kualitas, dan lain-lain. Data penawaran tersebut haruslah diamankan sehingga pihak saingan proyek tidak bisa mengetahui penawaran perusahaan lainnya, sehingga haruslah terisolasi dan terjaga keamanannya.

Data tersebut bisa saja disadap atau dimodifikasi apabila perusahaan tidak menjaga keamanannya dengan baik, disadap membuat perusahaan lain bisa menawarkan harga yang lebih baik dari perusahaan itu, misalnya penawaran 1 triliun rupiah kemudian penyadap menawarkan 999 miliar 999 juta rupiah, meskipun hanya berbeda 1 juta rupiah tapi dampaknya bisa mempengaruhi kemenangannya. Kemudian apabila dimodifikasi misalnya penawaran 1 triliun rupiah dengan spesifikasi tertentu kemudian penyadap memodifikasi biayanya menjadi 1,1 triliun rupiah dan spesifikasi yang lebih buruk, hal ini juga bisa mempengaruhi kemenangannya.

Oleh karena itu keamanan berupa kriptografi hadir dalam menyelesaikan ataupun mencegah kejadian yang tidak diharapkan seperti diatas, fungsi hashing, self destruct file, sandi file bisa digunakan dalam prosesnya.

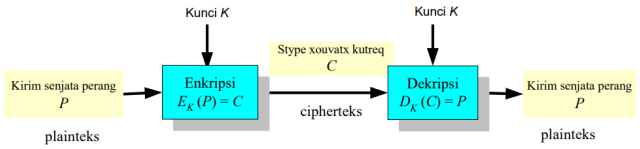
Namun tidak bisa dipungkiri meskipun penawaran yang diberikan terbilang menarik dan dominan, bisa saja tidak dimenangkan atas permainan dengan dasar rasa tidak enak atau balas budi akan jaringan yang dimiliki oleh penyedia proyek.

II. Dasar Teori

A. Kriptografi

Kriptografi merupakan keahlian untuk menyembunyikan pesan dan menerjemahkannya kembali sehingga terjalin komunikasi aman antara penerima pengirim pesan. Teknik kriptografi diterapkan pada berbagai aspek kehidupan manusia, seperti ATM, koneksi telefon, password wifi rumah, dan lain-lain. Kriptografi diterapkan dari zaman dahulu (klasik), yaitu saat julius caesar membuat algoritma penyembunyian pesan dari lawannya (*caesar cipher*), ilmuwan muslim seperti al kindi, vigenere, dan terus berkembang hingga menjadi kriptografi modern yang memiliki kompleksitas jauh lebih sulit dibandingkan kriptografi klasik.

Beberapa layanan dari kriptografi diantaranya kerahasiaan data, integritas data, autentikasi, non-repundansi. Enkripsi diartikan sebagai proses menyandikan plainteks menjadi cipherteks, sebaliknya Dekripsi diartikan sebagai proses mengembalikan cipherteks menjadi plainteks semula.



Kriptanalisis merupakan sebuah ilmu/seni memecahkan cipherteks menjadi plainteks tanpa mengetahui kunci yang digunakan. Dimulai sejak abad ke-9 masehi oleh ilmuwan muslim al-kindi dalam bukunya 'Risalah fi Istikhraj al-Mu'amma dengan analisis frekuensi.

B. Fungsi *Hash*

Fungsi *hash* atau prosesnya yang disebut *hashing* merupakan sebuah metode untuk menciptakan pesan berukuran tetap (*fixed*) dari file berukuran sembarang, sehingga setiap file bisa digunakan dalam prosesnya. Proses ini *irreversible* yang berarti tidak bisa dikembalikan dan hanya bisa satu arah saja.

Beberapa sifat dari fungsi hash yaitu:

1. *Collision resistance* : sangat sulit dan hampir tidak mungkin untuk menemukan hash dengan nilai hash yang sama, sehingga H(a) = H(b)
2. *Preimage resistance* : sangat sulit untuk menemukan inputan fungsi hash sehingga H(a) = y dengan y sudah diperkirakan/diincar sebelumnya
3. *Second Preimage resistance* : untuk input a dan output y = H(a), sangat sulit untuk mencari input kedua b sehingga H(b) = y

Karena beberapa sifat hash diatas, fungsi hash memiliki beberapa aplikasi sebagai berikut :

1. Menjaga integritas pesan, karena setiap perubahan sekecil mungkin (meskipun hanya 1 bit) bisa membawa hasil perubahan secara signifikan pada pesan.
2. Menghemat waktu pengiriman, karena dibandingkan mengirim file ulang secara terus menerus untuk *stay update*, hash bisa dilakukan untuk membandingkannya dan dipertimbangkan kalau sama tidak perlu dikirim ulang filenya.
3. Menormalkan panjang data yang beraneka ragam, untuk menyimpan data dengan panjang fixed (misalnya data *password* dan kredensial lainnya .

Fungsi hash memiliki beberapa varian, tetapi penulis hanya akan membahas yang berkaitan saja dengan implementasi yang dilakukan, yaitu md5 dan sha.

MD5 hash, diciptakan oleh Ron Rivest, menerima pesan dengan panjang sembarang dan mengubahnya menjadi *message digest* dengan panjang 128bit, memiliki 4 tahapan diantaranya :

1. *Padding bits*

   Dilakukan penambahan bit pengganjalnya (*padding bits*) sehingga panjang bit nya kongruen dengan 448 (mod 512), pengganjalnya diawali 1 dan dilanjut 0 sampai selesai (10000…).

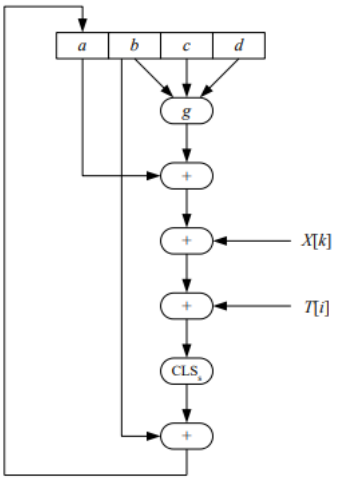
2. Penambahan Nilai Panjang Pesan

   Pesan yang telah diberi *padding bits* ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula, apabila K lebih besar dari 2^64 maka diambil K mod 2^64.

3. Inisialisasi Penyangga MD

   Membuat 4 *buffer* dengan ukuran masing-masing 32 bit

4. Pengolahan Pesan dalam Blok 512 Bit



Gambar operasi dasar MD5, sumber https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Fungsi-hash-MD5.pdf

$a \leftarrow b + CLSs\ (a + g(b, c, d) + X[k] + T[i])$

a, b, c, d = empat buah peubah penyangga 32-bit A, B, C, D

g = salah satu fungsi F, G, H, I

CLSs = circular left shift sebanyak s bit

X[k] = kelompok 32-bit ke-k dari blok 512 bit message ke-q. Nilai k = 0 sampai 15.

T[i] = elemen Tabel T ke-i (32 bit)

+ = operasi penjumlahan dalam modulo 232

Nama	Notasi	$g(b, c, d)$
f_F	$F(b, c, d)$	$(b \wedge c) \vee (\neg b \wedge d)$
f_G	$G(b, c, d)$	$(b \wedge d) \vee (c \wedge \neg d)$
f_H	$H(b, c, d)$	$b \oplus c \oplus d$
f_I	$I(b, c, d)$	$c \oplus (b \wedge \neg d)$

Gambar fungsi dasar MD5, sumber https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Fungsi-hash-MD5.pdf

SHA256, merupakan lanjutan dari SHA-2, prosesnya sebagai berikut

1. Membuat 8 buffer dengan panjang masing-masing 32 bit
2. Membuat *array of round constants*
3. Padding bits
4. Pengolahan Pesan dalam Blok 512 Bit
5. Dilakukan pemrosesan pesan dengan metode yang mirip dengan MD5 hash

Berikut dilampirkan pseudocode untuk sha-256 ini

```
Note 1: All variables are 32 bit unsigned integers and addition is calculated modulo 232
Note 2: For each round, there is one round constant k[i] and one entry in the message schedule array w[i], 0 ≤ i ≤
63
Note 3: The compression function uses 8 working variables, a through h
Note 4: Big-endian convention is used when expressing the constants in this pseudocode,
    and when parsing message block data from bytes to words, for example,
    the first word of the input message "abc" after padding is 0x61626380

Initialize hash values:
(first 32 bits of the fractional parts of the square roots of the first 8 primes 2..19):
h0 := 0x6a09e667
h1 := 0xbb67ae85
h2 := 0x3c6ef372
h3 := 0xa54ff53a
h4 := 0x510e527f
h5 := 0x9b05688c
h6 := 0x1f83d9ab
h7 := 0x5be0cd19

Initialize array of round constants:
(first 32 bits of the fractional parts of the cube roots of the first 64 primes 2..311):
k[0..63] :=
   0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
   0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
   0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
   0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
   0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
   0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
   0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
   0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2

Pre-processing (Padding):
begin with the original message of length L bits
append a single '1' bit
append K '0' bits, where K is the minimum number >= 0 such that L + 1 + K + 64 is a multiple of 512
append L as a 64-bit big-endian integer, making the total post-processed length a multiple of 512 bits
such that the bits in the message are L 1 00..<K 0's>..00 <L as 64 bit integer> = k*512 total bits

Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    create a 64-entry message schedule array w[0..63] of 32-bit words
    (The initial values in w[0..63] don't matter, so many implementations zero them here)
    copy chunk into first 16 words w[0..15] of the message schedule array

    Extend the first 16 words into the remaining 48 words w[16..63] of the message schedule array:
    for i from 16 to 63
        s0 := (w[i-15] rightrotate  7) xor (w[i-15] rightrotate 18) xor (w[i-15] rightshift  3)
        s1 := (w[i-2] rightrotate 17) xor (w[i-2] rightrotate 19) xor (w[i-2] rightshift 10)
        w[i] := w[i-16] + s0 + w[i-7] + s1

    Initialize working variables to current hash value:
    a := h0
    b := h1
    c := h2
    d := h3
    e := h4
    f := h5
    g := h6
    h := h7

    Compression function main loop:
    for i from 0 to 63
        S1 := (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)
        ch := (e and f) xor ((not e) and g)
        temp1 := h + S1 + ch + k[i] + w[i]
        S0 := (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)
        maj := (a and b) xor (a and c) xor (b and c)
        temp2 := S0 + maj

        h := g
        g := f
        f := e
        e := d + temp1
        d := c
        c := b
        b := a
        a := temp1 + temp2

    Add the compressed chunk to the current hash value:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
    h4 := h4 + e
    h5 := h5 + f
    h6 := h6 + g
    h7 := h7 + h

Produce the final hash value (big-endian):
digest := hash := h0 append h1 append h2 append h3 append h4 append h5 append h6 append h7
```

III. RANCANGAN SOLUSI

File tender proyek perlu diamankan untuk mencegah serangan yang mungkin terjadi, oleh karena itu penulis membuat rancangan solusi dengan skema sebagai berikut:

1. User membuka file dengan password tertentu, password bisa ditentukan oleh pemilik aplikasi sehingga hanya orang dalam saja yang mengetahuinya, ini merupakan layer pertama.
2. Apabila user telah login dan salah memasukkan password sebanyak 1-2 kali, dianggap wajar karena kemungkinan salah ketik, namun ketika 3 kali akan dilakukan skema corrupt data oleh aplikasi demi mencegah upaya bruteforce.
3. Skema corrupt dipilih karena cenderung lebih menyiksa penyerang, dan lebih melelahkan karena harus melakukan download ulang file, dan melewati layer-layer yang telah ada. Pertimbangan lainnya dibanding skema delete file, karena delete file bisa

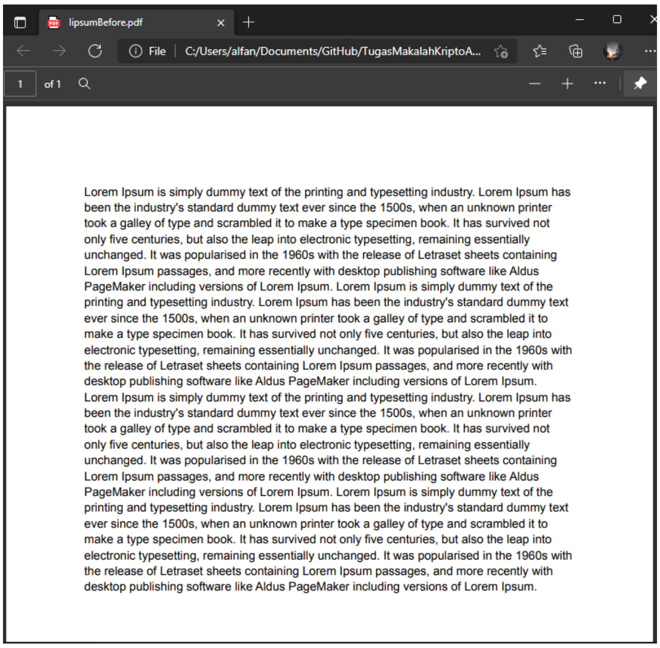
dilakukan recovery melalui trash can(recycle bin) dan juga kalau corrupt file biasanya penyerang tidak menyadari bahwa filenya telah corrupt meskipun telah melakukan brute force, alhasil meskipun telah menemukan kata sandinya file yang dipilih terlanjur kosong dan corrupt sehingga penyerang akan merasa kebingungan sendiri.

4. Upaya pencocokan hash (fitur kedua) bisa dilakukan sebagai upaya pencocokan atau pendeteksi file proyek tersebut, apakah sudah pernah dimodifikasi atau belum, skemanya dimulai dengan menyimpan hash file tersebut di dalam penyimpanan ataupun dikirim ke pihak penerima, kemudian dilakukan pencocokan dengan hash file yang baru diterima atas nama pengirim, apabila berbeda perlu dilakukan konfirmasi atas perbedaan tersebut, jika pihak penerima mengonfirmasikan tidak terdapat pada file maka bisa dipastikan telah dilakukan modifikasi secara tidak bertanggung jawab oleh pihak penyerang sehingga file ditolak dan file baru akan dikirim ulang.

Algoritma yang dipilih adalah sha256 karena mudah digunakan dan sudah terdapat librarynya untuk python, dan keamanannya cenderung lebih aman dibandingkan algoritma lain.

IV. Implementasi

Untuk implementasi dilakukan simulasi enkripsi dengan file dummy berisi library dummy text (lorem ipsum) dengan panjang 1 page format pdf, berikut isi filenya



Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

```python
import os
import hashlib
import webbrowser

def deleteFile(path):
    os.remove(path)

def openFile(path):
    webbrowser.open_new(path)

def sha256sum(filename):
    h  = hashlib.sha256()
    b  = bytearray(128*1024)
    mv = memoryview(b)
    with open(filename, 'rb', buffering=0) as f:
        for n in iter(lambda : f.readinto(mv), 0):
            h.update(mv[:n])
    return h.hexdigest()

def convertPdfToBinInsideTxt(pathpdf,pathtxt):
    file = open(pathpdf, 'wb')
    for line in open(pathtxt, 'rb').readlines():
        file.write(line)
    file.close()

def convertBinToPdfInsideTxt(pathpdf,pathtxt):
    file = open(pathtxt, 'wb')
    for line in open(pathpdf, 'rb').readlines():
        file.write(line)
    file.close()

def corrupt(pathpdf):
    f = open("temporary.txt", "x")
    file = open(pathpdf, 'wb')
    for line in open("temporary.txt", 'rb').readlines():
        file.write(line)

    file = open("temporary.txt", 'wb')
    for line in open(pathpdf, 'rb').readlines():
        file.write(line)
    file.write(b"corruptingpayload")
    f.close()
    file.close()
    os.remove("temporary.txt")

def main_interface():
    print("====================")
    print("DAVI INC FILE MANAGER")
    print("====================")
    print("1. Open File")
    print("2. Sha256 Check")
    print("3. Close File Manager")
    options = int(input("input : "))
    return options

def main():
    option = main_interface()
    while(option != 3):
        if (option == 1):
            secret_password = "indo1234"
            attempt_flag = False
            filename = str(input("input filename : "))
            for i in range (1,4):
                passwordInput = str(input("input password : "))
                if (passwordInput == secret_password):
                    print("Correct password")
                    print("File hash : "+sha256sum(filename))
                    attempt_flag = True
                    openFile(filename)
                    inp = str(input("press enter to continue : "))
                    break
                else:
                    print("Wrong password pls try again")
            if (attempt_flag == False):
                print("attempt failed exceed 3")
                corrupt(filename)
            break
        if (option == 2):
            source = str(input("source filename : "))
            target = str(input("target filename : "))
            if (sha256sum(source) == sha256sum(target)):
                print("Source and Target are exactly same !")
            else:
                print("Source and Target seems different")
            inp = str(input("press enter to continue : "))
            break
    os.system('cls')
    main()

if __name__ == "__main__":
```

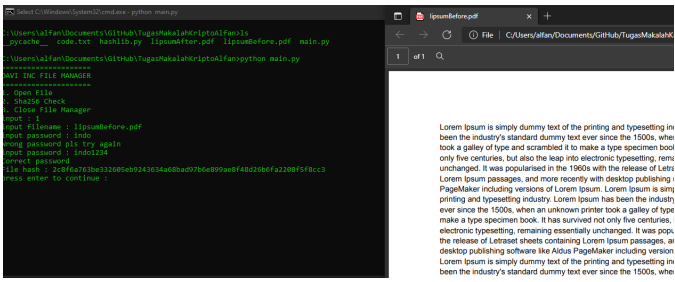
Kode program ditulis dengan python dengan library hashlib

Penjelasan kode program sebagai berikut :

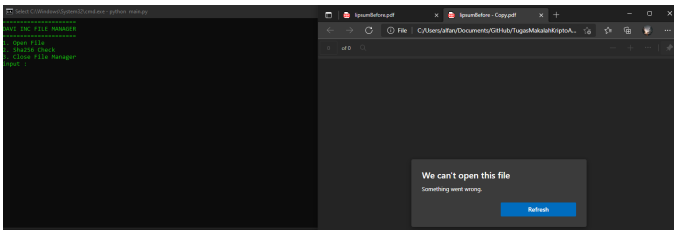
1. import os sebagai command pada cmd
2. import hashlib sebagai library sha256
3. import webbrowser sebagai open file pdf pada webbrowser yang digunakan default
4. fungsi deleteFile untuk melakukan deleteFile pada path file yang diinginkan
5. fungsi openFile untuk menampilkan file pada path file pada web browser default
6. fungsi sha256sum untuk melakukan hashing pada file yang diinginkan, dimulai dengan memanggil library hash, kemudian inisialisasi byte array dan membuat memory view atasnya. Kemudian dilakukan pembacaan file, sepanjang pembacaan dilakukan fungsi lambda dengan read into dan update setiap kali iterasinya, dan melakukan return hex digest sebagai hasil hash
7. fungsi convertPdfToBinInsideTxt untuk melakukan convert dari file pdf menjadi binary yang isinya disimpan didalam txt
8. fungsi convertBinToPdfInsideTxt untuk melakukan convert dari kumpulan binary didalam txt menjadi suatu file pdf utuh
9. fungsi corrupt untuk melakukan skema corrupt / perusakan file yang telah memasukkan password salah sebanyak 3 kali, diawali dengan mengubah pdf menjadi binary kemudian mengubahnya kembali ke bentuk pdf namun dilakukan insert bentuk binary dari kalimat “corruptingpayload” sehingga file otomatis corrupt karena struktur yang berantakan, kemudian file txt yang menjadi media konversi dihapus untuk membersihkan jejak
10. fungsi main untuk program utamanya dengan gui sederhana di command line, diawali dengan input option 1,2,3.
    a. Untuk option 1 akan dilakukan looping sebanyak 3 kali attempt input password, apabila benar akan melakukan set flag attempt_flag menjadi True, membuka file di webbrowser, dan menghentikan loop dengan break. Apabila password salah sebanyak 3 kali, maka flag akan tetap False dan dilakukan skema corrupt untuk merusak file.
    b. Untuk option 2 akan dilakukan pencocokan hash antara kedua file, diminta input nama file source dan targetnya, kemudian dilakukan pencocokan hasil hash keduanya, serta ditampilkan hasilnya.
    c. Untuk option 3 akan dilakukan penutupan program sebagai exit.

Berikut berbagai percobaan yang telah dilakukan terhadap program

1. Test case input password 2 kali salah dan 1 benar

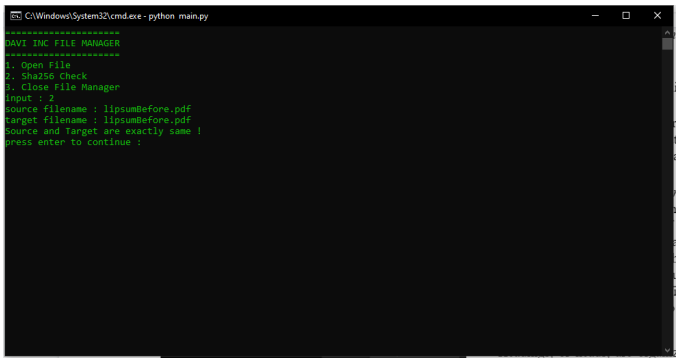


2. Test case input password salah 3 kali

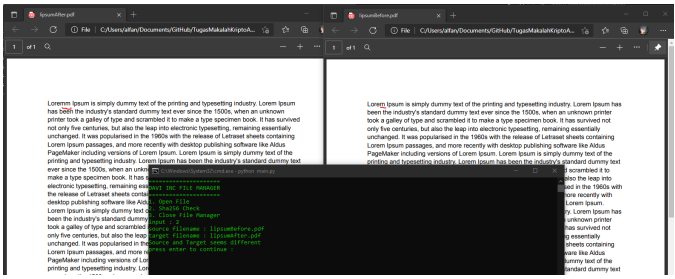


pada testcase ini program dilakukan input nama file lipsumBefore - Copy.pdf kemudian input password indo sebanyak 3 kali, program otomatis melakukan corrupt pada file dan membersihkan layar command line untuk mengecoh penyerang sehingga menganggap program refresh secara otomatis

3. Test case sha check file yang sama



4. Test case sha check pada file yang telah dimodifikasi sebelumnya



perbedaan terletak pada kata pertama yaitu lorem pada file awal dan menjadi loremm pada file kedua.

Analisis saya atas metode ini, cukup baik sebagai gambaran umum terhadap kriptografi tingkat lanjut yang bisa diaplikasikan, proses enkripsi file bisa ditingkatkan lagi, kemudian attempt password dan pola password bisa diperbaiki, pengecekan sha256 hash bisa di improve dengan tinggal memilih file di file manager atau drag and drop.

Sha256 bisa digunakan sebagai standar hash secara umum karena hasilnya sangat random.

V. Kesimpulan

Pengamanan file tender proyek menggunakan mekanisme yang telah dijelaskan merupakan langkah pertama dalam mencegah penyerang melakukan modifikasi, penyadapan file penting kita, banyak metode lain yang bisa diaplikasikan dan diterapkan dalam skema di dunia nyata, oleh karena itu penulis selalu belajar lebih lanjut dalam bidang ini dan senantiasa mengembangkan kemampuan.

Meskipun kita membuat keamanan tingkat yang hampir mustahil untuk ditembus, namun tetap ada saja kemungkinan atau celah untuk dimanfaatkan para penyerang dalam mengambil file kita. Oleh karena itu kita haruslah senantiasa mengembangkan dalam misi untuk mengurangi risiko dan peluangnya.

Video Link

Berikut merupakan link video demo saya untuk tugas makalah kali ini, silahkan menggunakan akun std untuk mengaksesnya
https://drive.google.com/file/d/1C9Lg4HEO4lU-7BjvTMcpr6iTBfF2rOWQ/view?usp=sharing

VI. Ucapan Terima Kasih

Terima kasih kepada Tuhan yang Maha Esa, yang oleh karena berkat dan rahmat-Nya makalah pengganti UAS ini dapat diselesaikan. Terima kasih juga diucapkan kepada Dr. Ir. Rinaldi Munir, M.T. sebagai dosen pengajar karena sudah memberikan ilmu yang sangat bermanfaat terkait kriptografi pada semester ini. Terakhir, terima kasih juga disampaikan kepada semua teman dan keluarga yang membantu dalam menyelesaikan makalah ini.

Referensi

[1] “Kriptografi”- https://id.wikipedia.org/wiki/Kriptografi (diakses 20 Desember 2021)

[2] “Pengantar kriptografi” - https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2021-2022/Pengantar-Kriptografi-(2021).pdf (diakses 20 Desember 2021)

[3] “Fungsi Hash “ - https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Fungsi-hash-2020.pdf (diakses 20 Desember 2021)

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.

Malang, 20 Desember 2021

Muhammad Alfandavi Aryo Utomo 13519211