P-ISSN :2580-4316
E-ISSN :2654-8054

Komparasi Fungsi Hash Md5 Dan Sha256 Dalam Keamanan Gambar Dan Teks

Izhar Rahim[1], Nizirwan Anwar[2], Agung Mulyo Widodo[3], Kundang Karsono Juman[4]
Iwan Setiawan[5]

[1,2,3,4] Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Esa Unggul
[5] Program Studi Teknik Informatika Fakultas Engineering, Computer and Design, Universitas Nusa Putra

Email : rahimizhar9897@gmail.com[1;] nizirwan.anwar@esaunggul.ac.id[2], agung.mulyo@esaunggul.ac.id[3], kundang.karsono@esaunggul.ac.id[4], iwan.setiawan@nusaputra.ac.id[5]

ABSTRAK

Implementasi ilmu kriptografi menjadi semakin sering digunakan seiring dengan daruratnya keamanan data. Ilmu kriptografi yang di dalamnya terdapat fungsi hash mengubah data menjadi sebuah nilai representasi agar data yang dimaksud tidak dapat dengan mudah diretas dan disalahgunakan oleh orang lain. Banyak macam fungsi hash yang dapat digunakan untuk fungsi kriptografi di antaranya adalah MD5 dan SHA256. Dari hasil percobaan menunjukkan bahwa keduanya mampu mengubah data menjadi nilai representatif dan tidak mudah dibaca oleh orang lain. Meskipun begitu, fungsi hash SHA256 lebih baik dalam mengamankan data karena nilai representatif yang dihasilkan lebih rumit dibandingkan dengan MD5.

Kata kunci: Kriptografi, Hashing, Keamanan Data, MD5, SHA256

ABSTRACT

Along with the urgency of data security, the deployment of cryptography is becoming increasingly prevalent. The field of cryptography in which a hash function turns data into a value representation so that the data cannot be readily stolen and exploited by others; MD5 and SHA256 are two examples of hash functions that can be used for cryptographic purposes. The experimental results indicate that both can convert data into representative values and cannot be deciphered by others with minimal effort. Nonetheless, the SHA256 hash function is more effective at protecting data than MD5 since the resulting representative value is more complex.

Keywords: *Cryptography, Hashing, Data Security, MD5, SHA256*

1. Pendahuluan

Keamanan suatu data menjadi sangat penting saat sudah berkaitan dengan data yang sangat rahasia. Karena jika sampai dapat diretas atau diketahui oleh orang lain, data yang ada dapat disalahgunakan. Dalam meningkatkan suatu keamanan data, dalam dunia teknologi berkembang sebuah ilmu yakni kriptografi. Kriptografi adalah ilmu yang mempelajari bagaimana membuat suatu pesan yang dikirim pengirim dapat disampaikan kepada penerima dengan aman. Selain pengertian tersebut terdapat pula pengertian ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, intergritas data, serta autentikasi data.

Dalam kriptografi memuat ilmu lain yakni hash. Hash memiliki arti memenggal lalu menggabungkan. Adapun proses hash dikenal sebagai *hashing*. H*ashing* seringkali digunakan sebagai metode pencarian suatu data dan menyimpannya dalam memori[1][2]. Namun seiring berjalannya waktu, karena kemampuan *hashing* mengubah suatu data menjadi string yang merepresentasikan data tersebut, akhirnya dikembangkanlah *hashing* untuk kriptografi. Fungsi hash yang dikembangkan untuk kriptografi sudah cukup beragam. Beberapa di antaranya adalah SHA256 dan MD5. Kedua fungsi tersebut umum digunakan dalam kasus kriptografi untuk melakukan pengamanan data. Maka dari itu, masalah yang akan diangkat adalah bagaimana kedua fungsi hash tersebut mampu mengubah data inputan menjadi nilai hash yang menjadi representasi dari data agar tidak dapat dengan mudah diretas atau diketahui oleh orang lain.

2. Landasaran Teori

2.1 Hashing

Hashing merupakan sebuah metode yang digunakan untuk mentransformasi suatu string menjadi karakter yang nilainya merepresentasikan string aslinya dengan operasi aritmatika. Dari segi Bahasa, hashing berasal dari kata hash yang berarti memenggal lalu menggabungkan. Dalam penggunaannya, metode hashing dapat digunakan sebagai metode penyimapanan data dalam sebuah larik (array) agar penyimpanan, pencarian, penambahan, dan penghapusan data dapat dilakukan dengan cepat[3].

Dasar proses dalam metode hashing adalah dengan menghitung posisi *record* yang dicari di dalam array dan bukan membandingkan *record* denga nisi pada array. Hal ini yang membuat proses hashing dapat mempercepat dalam proses pencarian data. Fungsi yang mengembalikan nilai atau kunci disebuah fungsi *hash* (*hash function*) dan array yang digunakan disebut tabel hash (*hash table*).

Fungsi hash menyimpan nilai asli atau kunci pada alamat yang sama dengan nilai hashnya. Pada pencarian suatu nilai pada tabel hash, yang pertama dilakukan adalah menghitung nilai hash dari kunci atau nilai aslinya, kemudian membandingkan kunci atuau nilai asli dengan isi pada memori yang beralamat nomor hashnya. Dengan cara ini, pencarian suatu nilai dapat dilakukan dengan cepat tanpa harus memeriksa seluruh isi tabel satu per satu. Secara teori, kompleksitas waktu (T(n)) dari fungsi hash yang ideal adalah O(1). Untuk mencapai itu setiap record membutuhkan suatu kunci yang unik.

Fungsi Hash (dilambangkan dengan h(k)) bertugas untuk mengubah k (key) menjadi suatu nilai dalam interval [0....X], dimana "X" adalah jumlah maksimum dari record-record yang dapat ditampung dalam tabel. Jumlah maksimum ini bergantung pada ruang memori yang tersedia. Fungsi Hash yang ideal adalah mudah dihitung dan bersifat random, agar dapat menyebarkan semua key. Dengan key yang tersebar, berarti data dapat terdistribusi secara seragam bentrokan dapat dicegah. Sehingga kompleksitas waktu model Hash dapat mencapai O(1), di mana kompleksitas tersebut tidak ditemukan pada struktur model lain.

Selain digunakan pada penyimpanan data, fungsi hash juga digunakan pada algoritma enkripsi sidik jari digital (fingerprint) untuk mengautentifikasi pengirim dan penerima pesan. Sidik jari digital diperoleh dengan fungsi hash, kemudian nilai hash dan tanda pesan yang asli dikirim kepada penerima pesan. Dengan menggunakan fungsi hash yang sama dengan pengirim pesan, penerima pesan mentransformasikan pesan yang diterima. Nilai hash yang diperoleh oleh penerima pesan kemudian dibandingkan dengan nilai hash yang dikirim pengirim pesan. Kedua nilai hash harus sama, jika tidak, pasti ada masalah. Ada beberapa macam fungsi hash yang relatif sederhana yang dapat digunakan dalam penyimpanan database:

a. Pembagian Bersisa (division-remainde)

Metode division remainder atau bisa juga disebut modulo adalah membagi nilai kunci dengan sebuah bilangan pembagi yang telah ditentukan (biasanya menggunakan ukuran maksimal bucket pada tabel hash), kemudian menggunakan sisa dari pembagian sebagai alamat relatif. Jumlah lokasi memori yang tersedi dihitung, kemudian jumlah tersebut digunakan sebagai pembagi untuk membagi nilai yang asli dan menghasilkan sisa. Sisa tersebut adalah nilai hashnya. Secara umum, rumusnya h(k)= k mod m. Dalam hal ini m adalah jumlah lokasi memori yang tersedia pada array. Fungsi hash tersebut menempatkan record dengan kunci k pada suatu lokasi memori yang beralamat h(k).

1) Melipat (*folding*)

Metode ini membagi nilai asli ke dalam beberapa bagian, kemudian menambahkan nilai-nilai tersebut, dan mengambil beberapa angka terakhir sebagai nilai hashnya. Pada metode ini, nilai kunci dibagi menjadi beberapa bagian yang masing-masing memiliki jumlah digit yang sama (biasanya kecuali bagian awal atau bagian terakhir). Kemudian bagian-bagian ini nantinya dijumlahkan untuk mendapatkan alamat relatif bucket dari kunci yang bersangkutan. Metode yang digunakan sebelum dilakukan penjumlahan pada masing-masing bagian umumnya menggunakan 2 (dua) cara. Yang pertama, masing-masing bagian dapat langsung dijumlahkan. Yang kedua, pada melakukan pembalikan posisi digit pada bagianbagian tertentu berdasarkan aturan yang telah ditetapkan sesuai kebutuhan. Misalnya terdapat bagian dengan susunan digit 2345, maka dibalik susunannya menjadi 5432.

2) Transformasi Radiks (*radix transformation*)

Nilai dalam bentuk digital, basis angka atau radiks dapat diganti sehingga menghasilkan urutan angka-angka yang

P-ISSN :2580-4316
E-ISSN :2654-8054

berbeda. Contohnya nilai desimal (basis 10) bisa ditransformasikan kedalam heksadesimal (basis 16). Digit atas hasilnya bisa dibuang agar panjang nilai hash dapat seragam.

b. Pengaturan Ulang Digit (*Digit Rearrangement*)

Metode ini mengubah urutan digit dengan pola tertentu. Contohnya mengambil digit ke tiga sampai ke enam dari nilai aslinya, kemudian membalikan urutannya dan menggunakan digit yang terurut terbalik itu sebagai nilai hash.

Fungsi hash yang bekerja dengan baik untuk penyimpanan pada database belum tentu bekerja dengan baik untuk keperluan kriptografi atau pengecekan kesalahan. Ada beberapa fungsi hash terkenal yang digunakan untuk keperluan kriptografi. Diantaranya adalah fungsi hash message-diggest, contohnya MD2, MD4, dan MD5, digunakan untuk menghasilkan nilai hash dari tanda tangan digital yang disebut message-diggest. Ada pula Secure Hash Algorithm (SHA), sebuah algoritma standar yang menghasilkan message-diggest yang lebih besar (60- bit) dan serupa dengan MD4[4].

Hashing selalu merupakan fungsi satu arah. Fungsi hash yang ideal tidak bisa diperoleh dengan melakukan reverse engineering dengan menganalisa nilai hash. Fungsi hash yang ideal juga seharusnya tidak menghasilkan nilai hash yang sama dari beberapa nilai yang berbeda . Jika hal yang seperti ini terjadi, inilah yang disebut dengan bentrokan (collision). Kemungkinan terjadinya bentrokan tidak dapat dihindari seratus persen. Fungsi hash yang baik dapat meminimalkan kemungkinan terjadinya bentrokan.

2.2 Kriptografi

Kriptografi adalah ilmu yang mempelajari bagaimana membuat suatu pesan yang dikirim pengirim dapat disampaikan kepada penerima dengan aman.[5] Selain pengertian tersebut terdapat pula pengertian ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, intergritas data, serta autentikasi data. Terminology empat tujuan mendasar dari ilmu kriptografi ini yang juga merupakan aspek keamanan informasi yaitu :

a. Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka/mengupas informasi yang telah disandi.
b. Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.
c. Autentikasi, adalah berhubungan dengan identifikasi/pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.
d. Non-repudiasi, atau nirpenyangkalan adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman/terciptanya suatu informasi oleh yang mengirimkan/ membuat.

Kriptografi juga tak lepas dari prosesnya dalam memanfaatkan fungsi hash. Fungsi hash Kriptografis adalah fungsi hash yang memiliki beberapa sifat keamanan tambahan sehingga dapat dipakai untuk tujuan keamanan data. Umumnya digunakan untuk keperluan autentikasi dan integritas data. Fungsi hash adalah fungsi yang secara efisien mengubah string input dengan panjang berhingga menjadi string output dengan panjang tetap yang disebut nilai hash[6]. Berdasarkan sifatnya, hash kriptografi memiliki sifat sebagai berikut:

a. Tahan preimej (*Preimage resistant*): bila diketahui nilai hash h maka sulit (secara komputasi tidak layak) untuk mendapatkan m dimana h = hash(m).
b. Tahan preimej kedua (*Second preimage resistant*): bila diketahui input m1 maka sulit mencari input m2 (tidak sama dengan m1) yang menyebabkan hash(m1) = hash(m2).
c. Tahan tumbukan (*Collision-resistant*): sulit mencari dua input berbeda m1 dan m2 yang menyebabkan hash(m1) = hash(m2)

Saat pengirim pesan hendak mengirimkan pesannya, dia harus membuat sidik jari dari pesan yang akan dikirim untuk penerima pesan. Pesan (yang besarnya dapat bervariasi) yang akan di-hash disebut preimej, sedangkan outputnya yang memiliki ukurannya tetap, disebut nilai hash. Kemudian, melalui saluran komunikasi yang aman, dia mengirimkan sidik jarinya kepada penerima. Setelah penerima menerima pesan si pengirim –

tidak peduli lewat saluran komunikasi yang mana – penerima kemudian juga membuat sidik jari dari pesan yang telah diterimanya dari pengirim. Kemudian kedua sidik jari dibandingkan. Jika kedua sidik jari itu identik, maka penerima dapat yakin bahwa pesan itu utuh tidak diubah-ubah sejak dibuatkan sidik jari yang diterimanya. Jika pesan sudah diubah, tentunya akan menghasilkan nilai hash yang berbeda.

Fungsi hash untuk membuat sidik jari tersebut dapat diketahui oleh siapapun, tak terkecuali, sehingga siapapun dapat memeriksa keutuhan dokumen atau pesan tertentu. Tak ada algoritma rahasia dan umumnya tak ada pula kunci rahasia. Jaminan dari keamanan sidik jari berangkat dari kenyataan bahwa hampir tidak ada dua preimej yang memiliki nilai hash yang sama. Inilah yang disebut dengan sifat bebas bentrokan dari suatu fungsi hash yang baik. Selain itu, sangat sulit untuk membuat suatu preimej jika hanya diketahui hash-valuenya saja. Contoh algoritma fungsi hash satu arah adalah MD-5 dan SHA. Message authentication code (MAC) adalah salah satu variasi dari fungsi hash satu arah, hanya saja selain preimej, sebuah kunci rahasia juga menjadi input bagi fungsi MAC.

Selain itu, penerima pesan memang dapat merasa yakin bahwa sidik jari yang datang bersama pesan yang diterimanya memang berkorelasi. Namun belum tentu pesan yang diterimanya adalah pesan sebenarnya yang dikirim. Bisa saja pesan tersebut sudah diterima orang lain saat melalui saluran komunikasi yang tidak aman, yang kemudian mengganti isi pesan dan membuat sidik jari lain seolah-olah itu adalah pesan yang sebenarnya.. Untuk mencegah pemalsuan, pengirim membubuhkan tanda tangannya pada pesan tersebut. Dalam dunia elektronik, pengirim membubuhkan tanda tangan digitalnya pada pesan yang akan dikirimkan agar penerima dapat merasa yakin bahwa pesan itu memang yang sebenarnya. Sifat yang diinginkan dari tanda tangan digital diantaranya adalah:

1. Tanda tangan itu asli (otentik), tidak mudah ditulis/ditiru oleh orang lain. Pesan dan tanda tangan pesan tersebut juga dapat menjadi barang bukti, sehingga penandatangan tak bisa menyangkal bahwa dulu ia tidak pernah menandatanganinya.
2. Tanda tangan itu hanya sah untuk dokumen (pesan) itu saja. Tanda tangan itu tidak bisa dipindahkan dari suatu dokumen ke dokumen lainnya. Ini juga berarti bahwa jika dokumen itu diubah, maka tanda tangan digital dari pesan tersebut tidak lagi sah.
3. Tanda tangan itu dapat diperiksa dengan mudah.
4. Tanda tangan itu dapat diperiksa oleh pihak-pihak yang belum pernah bertemu dengan penandatangan.
5. Tanda tangan itu juga sah untuk kopi dari dokumen yang sama persis.

Meskipun ada banyak skenario, ada baiknya kita perhatikan salah satu skenario yang cukup umum dalam penggunaan tanda tangan digital. Tanda tangan digital memanfaatkan fungsi hash satu arah untuk menjamin bahwa tanda tangan itu hanya berlaku untuk dokumen yang bersangkutan saja. Bukan dokumen tersebut secara keseluruhan yang ditandatangani, namun biasanya yang ditandatangani adalah sidik jari dari dokumen itu beserta timestamp-nya dengan menggunakan kunci privat. Timestamp berguna untuk menentukan waktu pengesahan dokumen. Keabsahan tanda tangan digital itu dapat diperiksa oleh penerima. Pertama-tama pesan yang diterima dibuat sidik jarinya. Kemudian tanda tangan digital didekripsi untuk mendapatkan sidik jari yang asli. Bandingkan kedua sidik jari tersebut. Jika kedua sidik jari tersebut sama, maka dapat diyakini bahwa pesan tersebut ditandatangani pengirim yang sebenarnya.

Pengiriman pesan dengan tanda tangan digital dapat dilakukan jika pengirim pesan dan penerima pesan sudah saling mengenal dan mengetahui kunci untuk enkripsi dan dekripsi pesan. Jika pengirim pesan dan penerima pesan belum pernah mengenal atau tidak saling mengetahui kunci untuk enkripsi dan dekripsi, perlu adanya pihak ketiga. Pihak ketiga ini diasumsikan telah memiliki saluran komunikasi yang aman dengan pengirim pesan dan penerima pesan. Pengirim pesan menggunakan saluran ini untuk mengirim pesan kepada penerima. Skenario ini tetap membutuhkan kunci-kunci kriptografi lagi (baik itu kunci simetris ataupun kunci asimetris) untuk pengamanan saluran komunikasi antara pihak ketiga dengan pengirim atau penerima pesan. Masalah di atas dapat dipecahkan dengan penggunaan sertifikat digital. Pihak ketiga tidak lagi setiap saat menjadi penukar kunci, namun cukup menandatangani kunci publik milik setiap orang di jaringan tersebut. Sebenarnya dalam sertifikat tersebut tak hanya berisi kunci publik, namun dapat berisi pula informasi penting lainnya mengenai jati diri pemilik kunci publik, seperti misalnya nama, alamat, pekerjaan, jabatan, perusahaan dan bahkan hash dari suatu informasi rahasia. Semua orang mempercayai otoritas pihak ketiga dalam memberikan tanda tangan, sehingga orang-orang dalam jaringan itu merasa aman menggunakan kunci publik yang telah ditandatanganinya.

P-ISSN :2580-4316
E-ISSN :2654-8054

Jika seseorang berhasil mencuri sertifikat digital yang dipertukarkan, serta menggantinya dengan sertifikat digital milik dirinya sendiri, dapat segera terlihat bahwa sertifikat digital yang diterima bukan 'lawan bicara' yang semestinya.

Serangan terhadap sistem yang memiliki pengamanan dengan sertifikat digital sulit dilakukan. Secara teoritis keunggulan dari tanda tangan digital adalah kemampuan untuk melakukan proses otentikasi secara off-line. Pemeriksa cukup memiliki kunci publik dari OS utama untuk mengetahui sah-tidaknya kunci publik dari lawan bicaranya. Selain itu untuk meningkatkan keamanan, kunci publik OS utama bisa saja diintegrasikan dalam program aplikasi. Namun kenyataannya, karena ada kemungkinan sertifikat digital tersebut hilang, tercuri atau identitas pemilik sertifikat berubah (perubahan alamat surat elektronik atau nomor KTP misalnya), maka sertifikat digital perlu diperiksa keabsahannya dengan melihat daftar sertifikat terbatalkan (certificate revocation list) yang disimpan oleh OS.

3. Metodologi

A. Hash MD5

MD5 adalah salah satu fungsi hash yang digunakan untuk keperluan kriptografi. Dalam kriptografi, MD5 (Message-Digest algortihm 5) ialah fungsi hash kriptografik yang digunakan secara luas dengan nilai hash 128-bit[7]. Pada standard Internet (RFC 1321), MD5 telah dimanfaatkan secara bermacam-macam pada aplikasi keamanan, dan MD5 juga umum digunkan untuk melakukan pengujian integritas sebuah file. MD5 di desain oleh Ronald Rivest, salah satu pembuat algoritma RSA, pada tahun 1991 untuk menggantikan hash function sebelumnya, MD4. MD5 memproses pesan dengan panjang variabel menjadi output dengan panjang tetap 128 bit. MD5 adalah fungsi hash yang populer. Ia bekerja pada blok 512-bit, dan memproses setiap blok melalui 4 putaran, di mana setiap putaran secara bergantian memproses 16 sub-blok (masing-masing 32-bit). Pesan 512-bit dibagi menjadi 16 sub-blok sebelum diproses. Pada tahun 1996, sebuah kecacatan ditemukan dalam desainnya, walau bukan kelemahan fatal, pengguna kriptografi mulai menganjurkan menggunakan algoritma lain, seperti SHA-1. Pada tahun 2004, kecacatan-kecacatan yang lebih serius ditemukan menyebabkan penggunaan algoritma tersebut dalam tujuan untuk keamanan jadi makin dipertanyakan. Hash-hash MD5 sepanjang 128-bit (16-byte), yang dikenal juga sebagai ringkasan pesan, secara tipikal ditampilkan dalam bilangan heksadesimal 32-digit.

Sangat tidak aman untuk menyimpan kata sandi dalam teks biasa di database. Untuk meningkatkan keamanan kata sandi, algoritma MD5 dapat digunakan untuk meng-hash kata sandi asli dan nilai hash, alih-alih plaintext disimpan dalam database. Selama otentikasi, kata sandi input juga di-hash oleh MD5 dengan cara yang sama, dan nilai hash hasil dibandingkan dengan nilai hash dalam database untuk pengguna tertentu. Berikut ini merupakan contoh pesan ASCII sepanjang 43-byte sebagai masukan dan hash MD5 terkait:

```
MD5("The quick brown fox jumps over
the lazy dog") =
9e107d9d372bb6826bd81d3542a419d6
```

Bahkan perubahan yang kecil pada pesan akan (dengan probabilitas lebih) menghasilkan hash yang benar-benar berbeda, misalnya pada kata "dog", huruf d diganti menjadi c:

```
MD5("The quick brown fox jumps over
the lazy cog") =
1055d3e698d289f2af8663725127bd4b
```

B. Hash SHA

Kemajuan dalam kriptanalisis fungsi hash kriptografi cukup lambat hingga baru-baru ini, komunitas kriptografi dikejutkan dengan kemajuan kriptanalisis fungsi hash, seperti serangan terhadap MD5 untuk menemukan tabrakan (*collision*) dan serangan dengan strategi baru pada SHA- 0 dan serangan untuk menemukan multi-tabrakan. Namun, teknik ini tidak berlaku untuk SHA-256 karena jadwal pesan dan fungsi putarannya yang lebih kompleks.

Fungsi hash SHA merupakan fungsi hash lain yang sering digunakan untuk keperluan kriptografi. Jika pada MD5, nilai hash yang dihasilkan memiliki panjang 128-bit, pada SHA-1 nilai yang dihasilkan memiliki ukuran 160 bits. Apabila seseorang mencoba untuk memecahkan kode enkripsi pada fungsi hash SHA-1, maka membutuhkan jumlah percobaan sebanyak 2^{160} setara dengan 1.461.501.637.330.902.918.203.684.832.716.283.019.655.932.542.976. Selain SHA-1, adapula SHA-256 yang sering digunakan.

Lebih dari SHA-1, SHA256 memiliki ukuran 256 bits. SHA-256 adalah fungsi hash kriptografi yang diusulkan pada tahun 2000 sebagai generasi baru fungsi SHA dan diadopsi sebagai standar FIPS

pada tahun 2002. SHA-256 dibangun dari konstruksi MD (Merkle-Damg ard) dan mode Davis-Meyer. Fungsi kompresi SHA-256 memiliki 64 putaran, dua jenis fungsi non-linier, rotasi siklik, dan konstanta yang bergantung pada putaran. SHA256 lebih sering digunakan dibandingkan SHA-1 karena kemampuan enkripsinya yang lebih baik karena menghasilkan nilai hash yang lebih rumit untuk dipecahkan[4][8].. Untuk memecahkan nilai hash dengan fungsi SHA256, dibutuhkan jumlah percobaan sebanyak 2^{256} atau sekitar

```
115.792.089.237.316.195.423.570.985.00
8.687.907.853.269.984.665.640.564.039.
457.584.007.913.129.639.936
```

C. Librari Cryptography

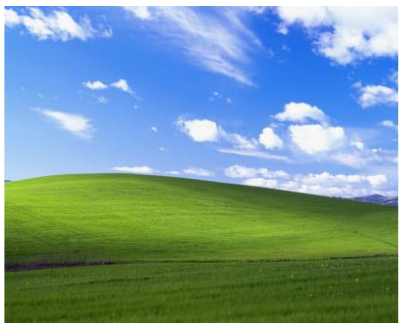
Librari Cryptography merupakan salah satu librari yang memuat berbagai fungsi yang dapat digunakan dalam proses kriptografi. Diciptakan Paul Kehrer dan dikembangkan secara *open-source*. Librari ini dapat digunakan dengan bahasa pemrograman python. Terdapat berbagai fungsi terkait kriptografi baik fungsi hash, *key generator*, dan fungsi-fungsi lain yang dapat membantu proses kriptografi[9].

4. Hasil dan Pembahasan

Dalam melakukan percobaan hashing menggunakan fungsi hash SHA256 dan MD5, ditentukan dua tipe file yaitu teks dan gambar. Adapun kedua contoh yang digunakan pada pengujian adalah sebagai berikut:

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Aliquam
hendrerit diam et justo suscipit,
vel.
```

Gambar 1 Teks yang digunakan untuk pengujian



Gambar 2 Image yang digunakan untuk pengujian

Percobaan dilakukan dengan menggunakan Bahasa pemrograman Python dan librari Cryptography. Maka dari itu, sebelum melakukan pengkodean untuk meng-*hash* kedua masukan tersebut, diperlukan pemanggilan library. Selain library Cryptography, dibutuhkan juga beberapa librari lain yang digunakan dalam pengkodean. Berikut kode python untuk pemanggilan librari-librari yang akan digunakan:

```
from cryptography.hazmat.primitives
import hashes
import os
import binascii
```

Fungsi yang memuat fungsi hash terdapat dalam fungsi hashes milik librari cryptography. Sedangkan untuk os merupakan fungsi yang digunakan untuk melakukan input file ke dalam mesin dan binascii merupakan decoder untuk menghasilkan nilai hex pada hash yang nantinya dihasilkan agar lebih dapat dibaca oleh manusia. Selanjutnya, dibuat suatu fungsi yang memuat proses hashing menggunakan fungsi hash sha256. Adapun pendeklarasian fungsinya dikodekan seperti di bawah ini:

```
def sha256(filename):
  hash_sha256 =
hashes.Hash(hashes.SHA256())
  with open(filename, "rb") as file:
    #read all file data
    file_data = file.read()
  hash_sha256.update(file_data)
  data = hash_sha256.finalize()
  hex =
binascii.b2a_hex(data).decode()

  return hex
```

Fungsi sha256 akan menerima parameter masukan filename yang memuat nama file yang akan diproses. Variabel hash_SHA256 digunakan untuk memanggil fungsi hases pada librari cryptography dan menggunakan fungsi hash SHA256. Selanjutnya file akan dibuka dan disimpan dalam variabel file_data. File yang sudah disimpan akan mulai di*hash* dan disimpan dalam variabel data, yang selanjutnya akan di*decode* ke bilangan hex agar lebih terbaca. Fungsi SHA256 akan melakukan pengembalian nilai hex agar nantinya dapat diperlihatkan nilai hex yang dihasilkan dari proses hash.

Selain fungsi hash256, dibuat pula fungsi hash md5 yang akan memuat proses hash menggunakan fungsi md5. Adapun kodenya dituliskan sebagai berikut:

P-ISSN :2580-4316
E-ISSN :2654-8054

```
def md5(filename):
  hash_md5 = hashes.Hash(hashes.MD5())
  with open(filename, "rb") as file:
    #read all file data
    file_data = file.read()
  hash_md5.update(file_data)
  data = hash_md5.finalize()
  hex =
binascii.b2a_hex(data).decode()

  return hex
```

Hampir sama dengan sebelumnya, fungsi md5 akan menerima parameter masukan filename yang memuat nama file yang akan diproses. Variabel hash_md5 digunakan untuk memanggil fungsi hases pada librari cryptography dan menggunakan fungsi hash md5. Selanjutnya file akan dibuka dan disimpan dalam variabel file_data. File yang sudah disimpan akan mulai di*hash* dan disimpan dalam variabel data, yang selanjutnya akan di*decode* ke bilangan hex agar lebih terbaca. Fungsi md5 akan melakukan pengembalian nilai hex agar nantinya dapat diperlihatkan nilai hex yang dihasilkan dari proses hash. Setelah kedua fungsi dideklarasi, maka langkah selanjutnya menginisialisasi fungsi yang telah dibuat lalu mencetak nilai hash sha256 dan md5. Hal tersebut dilakukan dengan kode berikut:

```
h_256 = sha256("window.jpg")
h_md5 = md5("window.jpg")

print("SHA256 Hash value: " + h_256)
print("MD5 Hash Value: " + h_md5)
```

Variabel h_256 menginisialisasi fungsi SHA256 dengan meminta parameter filename. Pada contoh, filename atau nama file yang digunakan adalah window.jpg. Lalu variabel h_md5 menginisialisasi fungsi h_md5 yang juga meminta parameter filename. Filename yang digunakan untuk parameter h_md5 sama dengan h_256 yakni window.jpg yang berupa file gambar. Untuk file teks yang akan dilakukan pengujian memiliki nama file tes.txt. Adapun keseluruhan kode dituliskan sebagai berikut:

```
from cryptography.hazmat.primitives
import hashes
import os
import binascii

def sha256(filename):
  hash_sha256 =
hashes.Hash(hashes.SHA256())
  with open(filename, "rb") as file:
    #read all file data
    file_data = file.read()
  hash_sha256.update(file_data)
  data = hash_sha256.finalize()
  hex =
binascii.b2a_hex(data).decode()

  return hex

def md5(filename):
  hash_md5 = hashes.Hash(hashes.MD5())
  with open(filename, "rb") as file:
    #read all file data
    file_data = file.read()
  hash_md5.update(file_data)
  data = hash_md5.finalize()
  hex =
binascii.b2a_hex(data).decode()

  return hex

h_256 = sha256("tes.txt")
h_md5 = md5("tes.txt")

print("SHA256 Hash value: " + h_256)
print("MD5 Hash Value: " + h_md5)
```

Dari tahapan tersebut, saat program dieksekusi maka akan menghasilkan nilai hash untuk SHA256 dan MD5 dari file yang dijadikan input. Adapun hasil nilai hash pada kedua contoh inputan gambar dan teks adalah sebagai berikut:

```
SHA256 Hash value: fb1d763b00fb89d9438df817b55f1a51acd445b3b5e5d41d116c9b20cc194d19
MD5 Hash Value: e72b3dc109311d97f55798a14a944222
```

Gambar 3 Nilai hash untuk file window.jpg

```
SHA256 Hash value: 9d2a8a79022ebe366c0727fe2fb1e19464c78585c66e76507ea84e320a66c87e
MD5 Hash Value: a93689eb27e07bec7945e2c308efade6
```

Gambar 4 Nilai hash untuk file tes.txt

5. Kesimpulan

Fungsi hash MD5 dan SHA256 dapat melakukan pengamanan dengan melakukan peubahan terhadap data inputan menjadi nilai yang merepresentasikan data tersebut. Hal ini dapat membuat file yang dimiliki lebih aman dan tidak mudah diretas oleh orang lain. Selain itu, dalam segi kekuatan keamanan, fungsi hash SHA256 dapat melakukan pengamanan lebih baik dibandingkan MD5. Dilihat dari hasil percobaan, nilai hash yang diproduksi oleh fungsi hash SHA256 lebih rumit dibandingkan dengan MD5, hal ini memungkinkan orang lain yang ingin mencuri data lebih kesulitan untuk memecahkan nilai hash SHA256. Namun dalam penggunaan pengamanan data yang tidak memiliki kerahasiaan tinggi, fungsi hash MD5 masih tetap dapat digunakan.

Daftar Pustaka

[1] J. Sinuraya, S. F. Rezky, and M. Tarigan, "Data Search Using Hash Join Query and Nested Join Query," *J. Phys. Conf. Ser.*, vol. 1361, no. 1, p. 12079, Nov. 2019, doi: 10.1088/1742-6596/1361/1/012079.

[2] P. Li, X. Zhu, X. Zhang, P. Ren, and L. Wang, "Hash Code Reconstruction for Fast Similarity Search," *IEEE Signal Process. Lett.*, vol. 26, no. 5, pp. 695–699, May 2019, doi: 10.1109/LSP.2019.2898772.

[3] S. W. Bray, *Implementing Cryptography using Python*. Indianapolis: John Wiley & Sons, Inc., 2020.

[4] S. J. Nielson and C. K. Monson, *Practical cryptography in python: Learning Correct Cryptography by Example*. 2019. doi: 10.1007/978-1-4842-4900-0.

[5] A.-C. Onwutalobi, "Overview of Cryptography," *SSRN Electron. J.*, 2011, doi: 10.2139/ssrn.2741776.

[6] A. Hashmi and R. Choubey, "Cryptographic Techniques in Information Security," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 3, pp. 854–859, 2018.

[7] M. Kioon, Z. Wang, and S. Das, "Security Analysis of MD5 Algorithm in Password Storage," *Appl. Mech. Mater.*, vol. 347–350, 2013, doi: 10.2991/isccca.2013.177.

[8] H. Yoshida and A. Biryukov, "Analysis of a SHA-256 Variant," 2005, pp. 245–260. doi: 10.1007/11693383_17.

[9] P. Kehrer, "Cryptography Library." https://cryptography.io/en/ (accessed May 27, 2022).