Penjagaan Integritas Dokumen Struk Belanja Digital menggunakan HMAC

Hengky Surya Angkasa - 13518048
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): hengkysurya88@gmail.com

***Abstract*—Saat ini, transaksi atau belanja digital sangat mudah dilakukan akibat tren teknologi digital. Masyarakat dapat memenuhi kebutuhannya melalui belanja dari gawai yang dimiliki. Potensi keamanan muncul dari sisi dokumen struk belanja digital. Pesan dalam dokumen dapat diubah sehingga diperlukan penjagaan. Kode MAC dapat menjadi kode yang menunjukkan integritas dokumen. Metode HMAC menghasilkan kode MAC dari *message digest* fungsi *hash* dengan input kunci rahasia yang di-*concat* dengan pesan. Dari hasil eksperimen, kode MAC berhasil dibangkitkan dan dimasukkan pada dokumen. Selain itu, kode MAC dapat digunakan pada proses verifikasi integritas dokumen.**

Keywords—Dokumen struk belanja digital, integritas pesan, kode MAC,* metode *HMAC

I. LATAR BELAKANG

Pada masa ini, masyarakat sudah secara masif menggunakan teknologi digital sebagai sarana untuk memenuhi kebutuhan, baik kebutuhan primer, sekunder, atau tersier. Hal tersebut diikuti oleh *trend* toko digital. Toko digital sudah menjamur pada dewasa ini, baik berbasis aplikasi *smartphone* atau berbasis *web* salah satunya *marketplace*. Masyarakat dengan mudah dapat memenuhi kebutuhan melalui gawai yang dimiliki.

Kemudahan yang ada diikuti oleh peluang ancaman. Salah satunya, ancaman keaslian atau integritas dokumen transaksi digital. Setiap transaksi yang berhasil dilakukan pengguna akan menghasilkan suatu dokumen struk belanja dalam bentuk digital. Dokumen tersebut perlu dijaga integritasnya karena sebagai bukti apabila terjadi proses retur barang, dsb. Pihak toko juga tidak akan dirugikan apabila ada dokumen yang dipalsukan.

Kriptografi adalah ilmu yang mempelajari teknik matematika yang berhubungan dengan keamanan informasi, salah satunya integritas pesan. Teknik kriptografi membuat pesan yang telah diubah dapat diketahui penerima pesan.

Salah satu metode penjagaan integritas dokumen yang ada adalah *Message Authentication Code* (MAC). MAC adalah kode yang dibangkitkan melalui fungsi *hash* dengan memanfaatkan kunci rahasia dalam pembangkitannya. Kode MAC dilekatkan pada dokumen dan diverifikasi dengan kode yang dibangkitkan penerima dokumen (verifikator).

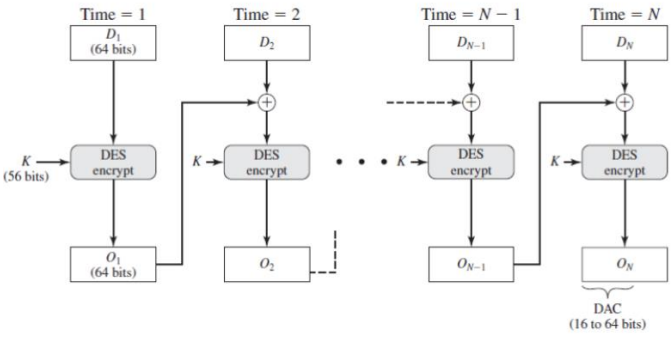
II. DASAR TEORI

A. *Message Authentication Code (MAC)*

Message Authentication Code (MAC) [1] adalah kode yang dibangkitkan oleh fungsi *hash*, namun menggunakan kunci rahasia dalam proses pembangkitannya. Cara kerja MAC adalah (1) pengirim membangkitkan kode MAC dari pesan dan kunci rahasia menggunakan algoritma MAC. Pesan kemudian di-*embed* kedalam dokumen sebelum dikirim ke penerima. (2) Penerima kemudian membangkitkan kode MAC dari pesan yang diterima dan kunci rahasia yang ada menggunakan algoritma MAC. Kode MAC yang dibangkitkan kemudian dicocokan dengan kode MAC yang terdapat pada pesan. Dokumen masih terjaga integritasnya apabila kode MAC sama.

Fungsi MAC adalah menjaga integritas isi dokumen terhadap adanya perubahan pihak lain. Kelebihan MAC dibandingkan fungsi *hash* dalam menjaga integritas dokumen adalah terdapat kunci rahasia untuk pembangkitan *message digest*. Pihak lain tidak dapat mengelabui verfikator dengan cara mengganti pesan dan *message digest* pada dokumen.

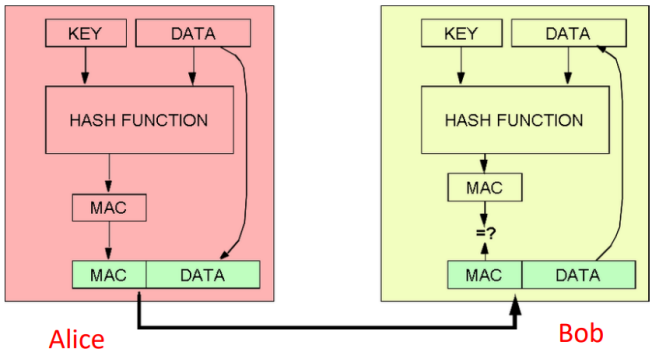
Terdapat dua jenis algoritma MAC yang digunakan. Jenis pertama adalah algoritma MAC berbasis *block cipher*. Kode MAC merupakan hasil enkripsi dengan kunci rahasia pada blok terakhir. Contohnya adalah *Data Authentication Algorithm* (DAA), yang merupakan algoritma MAC berbasis DES dan CBC (Gambar 1).



Gambar 1. Algoritma MAC berbasis DES [1]

Jenis kedua adalah algoritma MAC berbasis fungsi *hash* satu-arah (HMAC). Kode MAC didapatkan dari *message digest* fungsi *hash* dengan *input* kunci rahasia yang di-*concat* dengan

pesan (Gambar 2). Fungsi *hash* yang dipakai misalnya MD5 dan SHA.



Gambar 2. Algoritma HMAC [1]

B. SHA-3

Keccak adalah fungsi *hash* yang menjadi pemenang kompetisi oleh *National Institute of Standards and Technology* (NIST) dan menjadi SHA-3 yang dikenal saat ini. SHA-3 merupakan generasi baru dari algoritma *hash* SHA yang menjadi komplementer algoritma sebelumnya, yaitu SHA-1 dan SHA-2. Konsep desain *keccak* berbeda dengan para finalis lainnya. *Keccak* menggunakan prinsip *sponge construction*. Desain *keccak* berbeda dengan desain lainnya yang bergantung pada *compression function*. *Keccak* menggunakan fungsi non-kompresi untuk menyerap dan kemudian 'memeras' *digest*.

SHA-3 [2] memiliki dua fase, *absorbing* dan *squeezing*. Namun, perlu dilakukan proses *padding* pada pesan terlebih dahulu. Proses *padding* adalah sebagai berikut

1. Panjang digest adalah d bit
2. Pertama, pesan M ditambah dengan bit-bit pengganjal (*padding*) menjadi *string* P sehingga habis dibagi dengan r atau n = length(P)/r
3. Selanjutnya, P dipotong menjadi blok-blok Pi berukuran r-bit
4. Kemudian, b-bit (b = r+c) dari peubah status (*state*) S diinisialisasi menjadi nol dan konstruksi spons dimulai

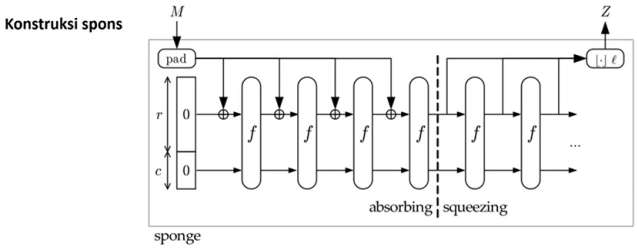
Hasil *padding* digunakan pada fase *absorbing*. Fase *absorbing* berjalan seperti berikut

1. Setiap blok masukan Pi berukuran r-bit, XOR-kan dengan r-bit pertama dari *state* S
2. Hasilnya dimasukkan ke dalam fungsi permutasi f untuk menghasilkan *state* baru S

Setelah semua blok diproses pada fase *absorbing*, *state* S masuk ke dalam fase *squeezing*.

1. Message digest akan disimpan di dalam Z
2. Inisialisasi Z dengan string kosong (null string)
3. Selagi panjang Z belum sama dengan d, r-bit pertama dari state S disambungkan (append) ke Z
4. Jika panjang Z masih belum sama dengan d, masukkan ke dalam fungsi permutasi f menghasilkan state baru S

Proses lengkapnya diilustrasikan pada Gambar 3.



Gambar 3. Arsitektur Algoritma SHA-3 [2]

Salah satu jenis SHA-3 (*Keccak*) adalah SHA-3 dengan *output bits* sebesar 256 bit. Contoh *message digest* yang dihasilkan SHA-3 versi 256 bit adalah sebagai berikut

Tabel 1. Contoh hasil fungsi *hash* SHA-3 256

Pesan	***Message digest***
Halo	51e5ddea5dcdaa85bc8623b 8ac163bed5c3b00e9e2ceaec a78f7f3cd8016d836
Halu	beccbf92062e8427947bfed8 1a546d4ccebba76d3f002bc2 54e19a6d3359d144

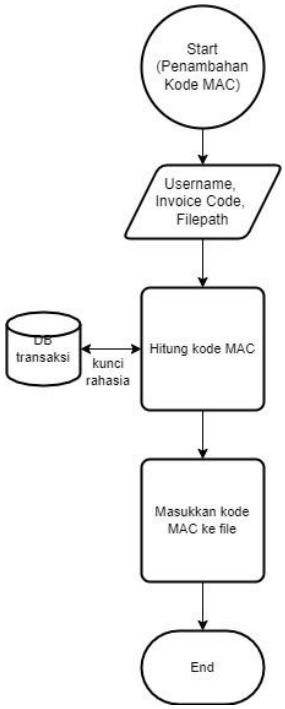
Dari Tabel 1 diatas, terlihat bahwa perbedaan satu kata (bit) pada pesan menhasilkan *message digest* yang sangat berbeda. Hal ini menjadi keuntungan pada pengamanan pesan.

III. RANCANGAN SOLUSI

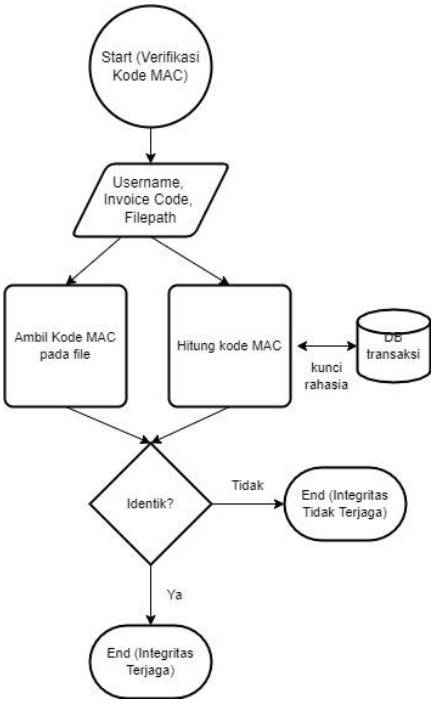
Pada bagian ini, dijelaskan rancangan solusi dari permasalahan yang ada.

A. Skema Umum

Secara umum, solusi dibagi menjadi dua bagian, yaitu bagian penambahan kode MAC pada dokumen (Gambar 4) dan bagian verifikasi dokumen (Gambar 5). Bagian penambahan kode MAC pada dokumen adalah proses membangkitkan kode MAC berdasarkan isi dokumen transaksi digital dan kunci rahasia. Kemudian, kode MAC di-*embed* pada dokumen. Dokumen yang telah memiliki kode MAC dapat diberikan kepada *customer*. Bagian verifikasi dokumen adalah proses melakukan verifikasi integritas dokumen transaksi digital yang masuk ke sistem. Verifikasi dilakukan dengan membandingkan kode MAC pada dokumen dengan kode MAC hasil kalkulasi isi dokumen yang diterima dan kunci rahasia. Bagian ini menunjukkan apakah integritas terjaga atau tidak.



Gambar 4. Skema Umum Penambahan Kode MAC



Gambar 5. Skema Umum Proses Verifkasi

B. Skema Penambahan Kode MAC pada Dokumen

Proses penambahan kode MAC adalah sebagai berikut

- Sistem membaca dokumen dalam bentuk *bytes*
- Sistem melakukan *query* untuk mendapatkan *private key* yang ada pada basis data sesuai *username* dan *invoice code*
- Sistem melakukan *concat* antara *private key* (dalam bentuk *bytes*) dan *bytes* dokumen
- Sistem membangkitkan kode MAC memakai fungsi hash SHA3-256 dengan hasil *concat* sebelumnya sebagai *input bytes*
- Sistem menyelipkan kode MAC (dalam bentuk *hexadecimal*) pada akhir *file* dokumen struk digital

C. Skema Verifikasi Dokumen

Proses verifikasi dokumen adalah sebagai berikut

- Sistem membaca metadata pada dokumen dan mengambil kode MAC (terletak setelah *end of file*)
- Sistem mengambil *bytes* pesan pada dokumen (*bytes* yang ada sebelum *end of file*)
- Sistem melakukan *query* untuk mendapatkan *private key* yang ada pada basis data sesuai *username* dan *invoice code*
- Sistem melakukan *concat* antara *private key* (dalam bentuk *bytes*) dan *bytes* pesan pada dokumen
- Sistem membangkitkan kode MAC memakai fungsi hash SHA3-256 dengan hasil *concat* sebelumnya sebagai *input bytes*
- Kode MAC yang dibangkitkan kemudian dicek dengan kode MAC yang diambil dari dokumen
- Jika sama, sistem menghasilkan pesan “Integritas terjaga!”, dan jika berbeda menghasilkan pesan "Integritas tidak terjaga!"

IV. IMPLEMENTASI

Solusi aplikasi dikembangkan menggunakan bahasa pemograman Python 3.9, dengan basis data SQLite. Aplikasi merupakan *terminal-based* dan pengguna dalam menjalankan proses pembuatan kode MAC dan verifikasi dokumen pada *terminal* atau *command prompt*.

A. Lingkungan Implementasi

Lingkungan (*environment*) yang digunakan untuk implementasi adalah

1. Sistem Operasi Windows 8.1 64-bit dengan RAM 8GB dan CPU 4-core
2. Bahasa pemograman Python 3.9.6 64-bit
3. Basis data yang digunakan adalah SQLite (dengan DB-API sqlite3 bawaan Python)

B. Persiapan Lingkungan Implementasi

Pada bagian ini diasumsikan bahwa instalasi bahasa pemograman Python telah dilakukan. Selain itu, file basis data SQLite telah dibuat. Pembuatan file basis data dapat mengikuti tautan berikut [3].

1. Langkah pertama yang perlu dilakukan adalah membuat tabel transaksi yang menyimpan informasi

username (*string*), *invoice code* (*string*), dan *private key* (*int*) suatu transaksi. *Query* dalam pembuatan tabel pada basis data adalah sebagai berikut.

```
CREATE TABLE IF NOT EXISTS transaksi (
    id integer PRIMARY KEY AUTOINCREMENT,
    username text NOT NULL,
    invoice_code text NOT NULL,
    private_key integer NOT NULL
);
```

2. Langkah kedua yang perlu dilakukan adalah mengisi tabel yang telah dibuat dengan data. Untuk keperluan pengujian, dimasukkan dua data pada tabel. *Query* yang dilakukan adalah sebagai berikut.

```
INSERT INTO transaksi(username, invoice_code, private_key) VALUES('johndoe', 'INV/16-12-2021/101', 192788)

INSERT INTO transaksi(username, invoice_code, private_key) VALUES('michael', 'INV/17-12-2021/211', 7851245)
```

3. Langkah ketiga yang perlu dilakukan adalah mempersiapkan fungsi *hash* SHA3-256. Fungsi *hash* tersebut tersedia pada *library* bahasa Python. Namun, penulis menggunakan implementasi pribadi (tidak menggunakan *library*). Dokumentasi implementasi SHA3 (Keccak) terdapat pada tautan berikut.

C. *Hasil Implementasi*

Terdapat empat file kode program yang dihasilkan pada implementasi. File pertama adalah *embed_MAC.py* sebagai implementasi penambahan kode MAC pada dokumen. File kedua adalah *verify_MAC.py* sebagai implementasi verifikasi dokumen struk belanja. File ketiga adalah *sha3.py* sebagai implementasi fungsi *hash* SHA3-256. File keempat adalah *db_conn.py* sebagai penyedia koneksi ke basis data untuk melakukan *query*.

Berikut merupakan cara pemakaian program, baik pada saat pembuatan kode MAC pada dokumen dan saat verfikasi dokumen.

```
python embed_MAC.py <username> <invoice_code> <input_doc_path>
```

```
python verify_MAC.py <username> <invoice_code> < input_doc_path>
```

Username dan *invoice code* wajib terdapat pada basis data, jika tidak sistem akan mengeluarkan peringatan "Input pada terminal tidak valid!". *Input_doc_path* merupakan lokasi dokumen yang akan ditambahkan kode MAC atau diverifikasi. Pada tahap *embed* kode MAC, dokumen lama langsung digantikan dokumen baru yang telah memiliki kode MAC.

V. EKSPERIMEN

Eksperimen atau pengujian yang dilakukan adalah sebagai berikut

1. Menambahkan kode MAC pada dokumen struk belanja digital
2. Memverifikasi dokumen struk belanja digital dengan kasus integritas terjaga
3. Memverifikasi dokumen struk belanja digital dengan kasus integritas tidak terjaga

Data pengguna yang digunakan sesuai dengan implementasi pada bab sebelumnya. Contoh dokumen struk belanja menggunakan dokumen PDF, isi dapat dilihat pada Gambar 6 dan Gambar 7.

STRUK BELANJA DIGITAL

Username: johndoe

Invoice code: INV/16-12-2021/101

Waktu: 16 Desember 2021 13:12:11

Nama Produk	Harga Satuan	Jumlah Beli	Total Harga
Kopi XYZ	Rp 2000	75	Rp 150000
The ABC	Rp 4000	25	Rp 100000
		Sub Total	**Rp 250000**
		Jasa Kirim	**Rp 10000**
		Total	**Rp 260000**

Gambar 6. Dokumen Struk Belanja Digital 1 (Contoh Struk Belanja.pdf)

STRUK BELANJA DIGITAL

Username: michael

Invoice code: INV/17-12-2021/211

Waktu: 17 Desember 2021 19:30:29

Nama Produk	Harga Satuan	Jumlah Beli	Total Harga
Samsang RAM 8GB DDR4	Rp 600000	1	Rp 600000
VD SSD SATA3 256 GB	Rp 550000	1	Rp 550000
		Sub Total	**Rp 1.150.000**
		Jasa Kirim	**Rp 50.000**
		Total	**Rp 1.200.000**

Gambar 7. Dokumen Struk Belanja Digital 2 (Contoh Struk Belanja Hardware.pdf)

A. *Eksperimen Penambahkan Kode MAC pada Dokumen Struk Belanja Digital*

Penambahan kode MAC dijalankan menggunakan *command* pada *terminal* atau *command prompt*, seperti pada Gambar 8. Input parameter yang digunakan seperti pada Tabel 2.

```
python embed_MAC.py "johndoe" "INV/16-12-2021/101" "Contoh Struk Belanja.pdf"
```

Gambar 8. *Command* Proses Penambahan Kode MAC

Tabel 2. Input proses penambahan Kode MAC

Username	johndoe
Invoice Code	INV/16-12-2021/101
Filepath	Contoh Struk Belanja.pdf

Setelah dijalankan kode MAC ditambahkan akhir file sebagai kumpulan *bytes*. *Output* kode MAC juga ditampilkan pada terminal, seperti pada Gambar 9.

Gambar 9. Hasil Penambahan Kode MAC pada File Contoh Struk Belanja.pdf

Kode MAC:
c7f8ebf270f4630664713b9501c0e2ccf85922fef6ed98587636fc42e16cc387

Eksperimen dilanjutkan dengan dokumen struk digital lain, yaitu Contoh Struk Belanja Hardware.pdf. *Command* yang digunakan masih sama dengan input seperti pada tabel 3.

Tabel 3. Input proses penambahan Kode MAC

Username	michael
Invoice Code	INV/17-12-2021/211
Filepath	Contoh Struk Belanja Hardware.pdf

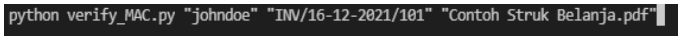
Setelah kode dijalankan, kode MAC ditambahkan pada akhir file dalam bentuk *bytes*. *Output* yang dihasilkan seperti pada gambar 10.

Gambar 10. Hasil Penambahan Kode MAC pada File Contoh Struk Belanja Hardware.pdf

Kode MAC:
c33c767495994ff1b0e9cfb73e63697dbabc76238a999ae0ff27486948eb604b

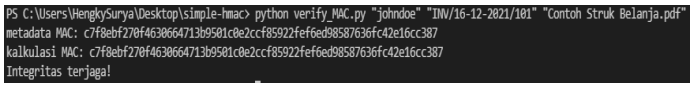
B. Eksperimen Verifikasi Dokumen Struk Belanja Digital dengan Integritas Terjaga

Verifikasi dilakukan pada salah satu dokumen hasil eksperimen sebelumnya (Contoh Struk Belanja.pdf). Verifikasi dijalankan menggunakan *command*, seperti pada Gambar 11.



Gambar 11. *Command* Proses Verifikasi Dokumen (integritas terjaga)

Setelah dijalankan, keluaran program adalah kode MAC dokumen (metadata), kode MAC hasil kalkulasi, dan informasi integritas. Gambar 12 menunjukkan *output* program. Tabel 4 menunjukkan perbandingan MAC pada dokumen dan MAC hasil kalkukasi tahap verifikasi.



Gambar 12. Hasil verifikasi dokumen (integritas terjaga)

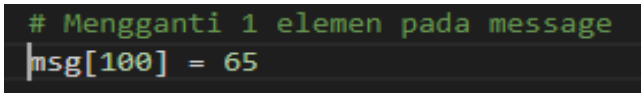
Tabel 4. Perbandingan kode MAC pada kasus integritas terjaga

Metadata MAC	**Kalkulasi MAC**
c7f8ebf270f46306647 13b9501c0e2ccf85922 fef6ed98587636fc42e1 6cc387	c7f8ebf270f46306647 13b9501c0e2ccf85922 fef6ed98587636fc42e1 6cc387

C. Eksperimen Verifikasi Dokumen Struk Belanja Digital dengan Integritas Tidak Terjaga

Pada eksperimen ini satu bit pada pesan diubah nilainya menjadi sembarang nilai. Dengan demikian, dapat dilihat apakah jika satu bit berubah akan berdampak pada proses verifikasi.

Dokumen yang dipakai untuk verifikasi adalah salah satu dokumen hasil eksperimen A (Contoh Struk Belanja.pdf). *Command* untuk menjalankan proses verifikasi sama seperti eksperimen B. Hal yang membedakan adalah terdapat penambahan satu baris kode pada program *verify_MAC.py*, yaitu mengubah nilai suatu bit, seperti yang terlihat pada Gambar 13. Misalnya, bit ke-100 pada *message* diubah nilainya menjadi 65 (nilai harus antara 0-255).



Gambar 13. Kode Program untuk Mengubah Pesan

Setelah *command* dijalankan, program menghasilkan *output* yang menyatakan bahwa "Integritas tidak terjaga!" (Gambar 14). Kode MAC yang terdapat pada dokumen tidak sama dengan hasil kalkulasi tahap verifikasi, seperti yang terlihat pada Tabel 5.

Gambar 14. Hasil Verifikasi Dokumen (integritas tidak terjaga)

Tabel 5. Perbandingan Kode MAC pada kasus integritas tidak terjaga

Metadata MAC	**Kalkulasi MAC**
c7f8ebf270f46306647 13b9501c0e2ccf85922 fef6ed98587636fc42e1 6cc387	3f649c1611d525342ec 8ed198fc28aca620dc9e 3bb89757d66be815334 ebf184

VI. ANALISIS HASIL EKSPERIMEN

Pada eksperimen pertama, yaitu penambahan kode MAC pada dokumen belanja digital, terlihat bahwa kode MAC berhasil dibangkitkan dari pesan dokumen dan kunci rahasia. Kode MAC berhasil disimpan pada akhir file "Contoh Struk Digital.pdf" dan "Contoh Struk Belanja Hardware.pdf". Hasil dokumen yang telah diberikan kode MAC tersebut dapat diberikan kepada pengguna.

Pada eksperimen kedua, yaitu verifikasi dokumen belanja digital yang dihasilkan eksperimen sebelumnya. Pengecekan dilakukan pada kode MAC dokumen *input* "Contoh Struk Digital.pdf". Kode MAC dokumen dibandingkan dengan kode MAC hasil perhitungan. Kode MAC dihitung berdasarkan pesan pada dokumen dan kunci rahasia. Hasilnya, kode MAC identik sehingga proses verifikasi berhasil. Artinya, jika dokumen struk belanja digital tidak diubah, maka kode MAC memiliki nilai yang sama dengan hasil kalkulasi.

Pada eksperimen ketiga, yaitu verifikasi dokumen belanja digital yang telah diubah pesannya. Pesan diubah dengan mengganti nilai satu bit secara acak. Pada eksperimen, bit pesan ke-100 diubah nilainya menjadi 65. Sama seperti eksperimen sebelumnya, kode MAC pada dokumen dibandingkan dengan kode MAC hasil perhitungan. Hasilnya, kode MAC pada dokumen tidak sama dengan kode MAC hasil kalkulasi. Hal tersebut disebabkan *bits* pesan yang tidak sama. Fungsi *hash* menghasilkan *message digest* yang berbeda walaupun pesan hanya berbeda 1 bit saja. Proses verifikasi gagal ditandai dengan pesan "Integritas tidak terjaga!".

Dari ketiga eksperimen, penggunaan HMAC dapat menjadi salah satu teknik menjaga integritas pesan. Kode MAC dapat dimasukkan pada dokumen dan proses verifikasi dapat dilakukan dari kode tersebut.

VII. KESIMPULAN

Message Authentication Code (MAC) adalah kode yang dibangkitkan oleh fungsi *hash*, namun menggunakan kunci rahasia dalam proses pembangkitannya. Salah satu algoritma MAC adalah HMAC. Kode MAC didapatkan dari *message digest* fungsi *hash* dengan input kunci rahasia yang di-*concat* dengan pesan. Kode MAC yang dihasilkan dapat digunakan untuk menjaga integritas dokumen. Kode MAC yang dibangkitkan dimasukkan pada akhir dokumen. Untuk melihat apakah integritas terjaga, kode MAC yang ada pada dokumen dibandingkan dengan kode MAC hasil perhitungan antara isi pesan dokumen saat ini dan kunci rahasia.

PRANALA KODE PROGRAM

https://github.com/hengkysuryaa/simple-hmac

UCAPAN TERIMA KASIH

Pertama-tama penulis mengucapkan puji dan syukur ke hadirat Tuhan Yang Maha Esa karena rahmat-Nya dan pertolongan-Nya penulis dapat menyelesaikan tugas makalah IF4020 Kriptografi ini, dan juga tugas-tugas lain pada mata kuliah ini.

Penulis juga mengucapkan terima kasih kepada dosen Pak Rinaldi Munir. Terima kasih atas ilmu yang diajar dikelas selama ini sehingga pengetahuan penulis dapat meningkat.

REFERENCES

[1] Munir, Rinaldi. 2021. Slide Kuliah IF4020 Kriptografi: *Message Authentication Code* (MAC) https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/MAC-2020.pdf Diakses 16 Desember 2021

[2] Munir, Rinaldi. 2021. Slide Kuliah IF4020 Kriptografi: *SHA-3* (Keccak) https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/SHA-3-2020 Diakses 16 Desember 2021

[3] SQLite Python Tutorial, https://www.sqlitetutorial.net/sqlite-python Diakses 16 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.

Sleman, 20 Desember 2021

Hengky Surya Angkasa
NIM 13518048