Implementasi Message Authentication Code (MAC) untuk Menjaga Integritas Data dalam DBMS

Nathania Calista Djunaedi - 13521139
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521139@std.stei.itb.ac.id

***Abstract*—Di Tengah pesatnya perkembangan teknologi di dunia, jumlah data yang perlu disimpan semakin meningkat. Hampir seluruh perusahaan menggunakan sistem manajemen basis data (DBMS) untuk menyimpan dan mengelola data. Meskipun DBMS dapat mengatasi permasalahan besarnya jumlah data, integritas dan keamanan dari data yang tersimpan masih menjadi sebuah tantangan. Message Authentication Code (MAC) adalah sebuah algoritma kriptografi yang dapat membantu permasalahan ini. MAC dapat digunakan untuk mendeteksi apakah data sudah diubah atau belum dan dengan demikian dapat menjamin integritas data.**

Keywords—cryptography; DBMS; MAC; SHA-256; authentication

I. PENDAHULUAN

Di era digital saat ini, hampir seluruh manusia menggunakan perangkat lunak dalam kehidupan sehari – hari mereka. Hampir seluruh perangkat lunak pasti menyimpan data – data pengguna, baik data pribadi pengguna, data aktivias pengguna, ataupun data – data lainnya. Hal ini menyebabkan jumlah data yang disimpan meningkat seiring dengan meningkatnya jumlah pengguna perangkat lunak.

Data menjadi sebuah aset yang sangat penting bagi Perusahaan maupun bagi pengguna. Dengan menyimpan data – data pengguna, Perusahaan dapat membuat pengalaman pengguna (*user* experience) menjadi lebih personal. Sedangkan, dari sisi pengguna, data privas mereka tidak boleh tersebar karena dapat disalahgunakan oleh pihak – pihak dengan kepentingan tertentu. Seiring dengan meningkatnya jumlah data, serangan terhadap data juga semakin gencar terjadi.

Sistem Manejemen Basis Data (DBMS) sering digunakan untuk menyimpan dan mengelola data. DBMS menjadi salah satu alternatif untuk menyimpan data yang sangat besar karena menyediakan cara – cara yang memudahkan pengguna untuk mengakses atau memodifkasi data mereka. Selain itu, DBMS juga mengenalkan sebuah mekanisme transaksi yang membantu dalam menjaga kekonsistenan data ketika diakses oleh bnyak orang.

Walaupun ada DBMS yang dapat digunakan untuk mengelola data dalam jumlah besar, integritas dan keamanan data masih belum terjamin. Oleh karena itu, untuk menjaga integritas dan keamanan data dalam DBMS, dapat digunakan *Message Authentication Code (MAC).* MAC adalah algoritma kriptografi yang digunakan untuk memvalidasi keaslian dan integritas pesan. Dengan MAC, perubahan tidak sah pada data dapat terdeteksi, sehingga membantu dalam memastikan bahwa data yang disimpan dan diambil dari DBMS tetap utuh dan tidak dimanipulasi.

II. TEORI DASAR

A. Sistem Manajemen Basis Data (DBMS)

Sistem Manajemen Basis Data (DBMS) adalah perangkat lunak yang digunakan untuk mengeolal dan mengatur data dalam suatu basis data. Basis data adalah Kumpulan data yang terstruktur, tersimpan, dan dapat diakses dengan mudah. DBMS memungkinkan pengguna untuk melakukan berbagai operasi terkait data, seperti penyimpanan, pengambilan, pembaruan, dan penghapusan data.

Terdapat 3 jenis DBMS yang sering digunakan, yaitu *relational database management system* (RDBMS), *document database management system* (DoDBMS), dan *columnar database management system* (CDBMS). Pada RDBMS, data disimpan dalam table yang terdiri dari baris dan kolom. Setiap table memiliki kunci utama yang mengidentifikasi secara unik setiap baris dalam tabel. DoDBMS menyimpan data dalam bentuk dokumen, seperti JSON atau BSON. CDBMS menyimpan data dalam kolom, sehinga sangat efisien untuk mengambil data agregat dan biasanya digunakan dalam menganalisis data.

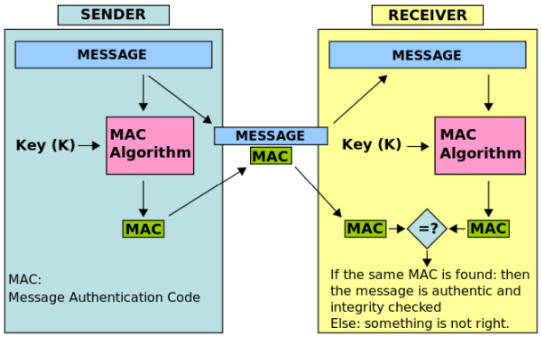
DBMS memiliki beberapa komponen, yaitu *storage engine, query language, query processor, optimization engine, metadata catalog, log manager, reporting and monitoring tool,* serta *data utilities.*

B. Message Authentication Code (MAC)

Message authentication code (MAC) adalah sebuah kode kecil berukuran tetap (*fixed*) yang dihasilkan dari pesan dan kunci. Kode kecil ini nanti akan digunakan untuk mengotentikasi pengirim dan memeriksa integritas pesan. MAC memastikan bahwa pesan yang diterima berasal dari pengirim yang benar dan tidak mengalami perubahan apapun.

Setelah mendapatkan nilai MAC, MAC akan dilekatkan kepada pesan dan dianggap sebagai *signature* dari pesan tersebut. Algoritma MAC adalah fungsi banyak ke satu (*many-to-one function*) yang memungkinkan banyak pesan memiliki MAC yang sama. Namun, pada implementasi di dunia nyata, sangat sulit menemukan dua pesan berbeda dengan nilai MAC yang sama.

Setelah menambahkan MAC ke dalam pesan, pesan akan dikirimkana ke penerima. Penerima kemudian akan menghitung nilai MAC dari pesan tersebut dengan menggunakan alagoritma MAC Yng sama. Jika hasil perhitungan penerima sama dengan nilai MAC pada pesan yang dikirimkan, maka pesan tidak mengalami perubahan apapun. Namun, jika nilai MAC hasil perhitungan penerima berbeda dengan nilai MAC pada pesan, maka pesan dapat dipastikan sudah pernah diubah oleh pihak eksternal.



Secara natural, konsep MAC hanya menekankan pada otentikasi pesan aja, sedangkan kerahasiaan pesan tidak ditekankan. Namun, jika ingin menjaga kerahasiaan pesan juga, maka konsep MAC dapat digabungkan dengan enkripsi pesan. Kedua konsep ini menggunakan 2 kunci yang terpisah. Nilai MAC dapat dihitung setelah atau sebelum pesan di enkripsi. Dengan menggabungkan kedua konsep tersebut, kerahasiaan pesan juga dapat dijamin.

Algoritma MAC yang diugunakan harus memenuhi syarat – syarat berikut ini :

1. Diberikan sebuah pesan dan MAC, maka sangat sukar menemukan pesan lain yang memiliki MAC yang sama
2. MAC dari berbagai pesna seharusnya terdistribusi secara *uniform*
3. MAC seharusnya bergantung rata pada semua bit – bit pesan

Berdasarkan syarat – syarat di atas, MAC dapat dihasilkan dengan dua cara, yaitu :

1. **Algoritma MAC berbasis *block cipher***

   Pada algoritma ini, MAC dibangkitkan dari *block cipher* dengan mode CBC atau CFB. Nilai *hash* yang digunakan adalah hasil enkripsi blok terakhir.

2. **Algoritma MAC berbasis fungsi *hash* satu arah yang sudah ada**

   Contoh fungsi *hash* yang paling lazim digunakan untuk MAC adalah MD5 atau SHA. Pada algoritma ini, pesan M digabung dengan kunci K, lalu dihitung nilai *hash* dari hasil penggabungan tersebut dengan menggunakan fungsi *hash* seperti MD5 atau SHA.

C. Secure Hash Algorithm 256-bit (SHA-256)

Secure hash algorithm (SHA) adalah fungsi hash kriptografi yang diterbitkan oleh Institut Standar dan Teknologi Nasional (NIST) sebagai Standar Pemrosesan Informasi Federal (FIPS) AS. Algoritma ini dirancang untuk menghasilkan nilai hash yang unik dan aman untuk setiap bagian data yang diproses. Nilai hash yang dihasilkan oleh SHA bersifat unik untuk setiap input data, sehingga tidak mungkin dua input yang berbeda menghasilkan nilai hash yang sama. Keunikan ini menjadikan SHA sangat penting dalam memastikan integritas dan keamanan data dalam berbagai aplikasi, termasuk tanda tangan digital, sertifikat digital, dan berbagai protokol keamanan lainnya.

SHA-256, sesuai dengan namanya, adalah varian dari algoritma SHA yang selalu menghasilkan nilai hash berukuran tetap sebesar 256 bit dari input data dengan ukuran yang bervariasi. Terlepas dari panjang atau kompleksitas input data, output dari SHA-256 akan selalu berupa string 256 bit yang unik. Ini berarti bahwa bahkan perubahan kecil pada input data akan menghasilkan nilai hash yang sangat berbeda, sebuah fenomena yang dikenal sebagai efek avalanche. Dengan sifat-sifat ini, SHA-256 memainkan peran kunci dalam berbagai sistem keamanan modern, seperti blockchain, pengamanan data, dan autentikasi pesan. Algoritma ini memastikan bahwa data yang di-hash tidak dapat dipalsukan atau dimodifikasi tanpa terdeteksi, sehingga memberikan lapisan keamanan tambahan untuk informasi yang sensitif.

D. Hash-Based Message Authentication Code

Hash-Based Message Authentication Code, atau biasa disingkat dengan HMAC, adalah salah satu algoritma MAC berbasis fungsi *hash* satu arah. Biasanya, fungsi *hash* yang digunakan adalah MD5, SHA-1, SHA-256, atau SHA-512.

HMAC merupakan kriptografi simteris karena menggunakan kunci yang sama antara penerima dan pengirim. Kunci yang digunakan akan disepakati oleh penerima dan pengirim di awal – awal komunikasi atau sebelum mengirim pesan. Selain menyepakati kunci yang akan digunakan, penerima dan pengirim juga harus menyepakati algoritma kriptografi yang akan digunakan dalam proses ini.

Secara singkat, rumus HMAC dapat dilihat pada gambar di bawah ini

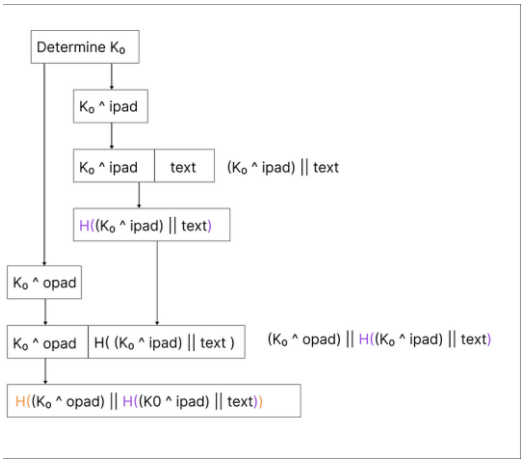
$$\mathrm{HMAC}(K,m) = \mathrm{H}\left(\left(K' \oplus opad\right) \parallel \mathrm{H}\left(\left(K' \oplus ipad\right) \parallel m\right)\right)$$

$$K' = \begin{cases} \mathrm{H}(K) & K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

K adalah kunci privat yang sudah diketahui dan disepakati oleh penerima den pengirim. H adalah fungsi *hash* yang akan digunakan, m adalah pesan yang akan diautentikasi, *opad* adalah 0x5C5C5C…5C dan *ipad* adalah 0X363636..36. Panjang *ipad* dan *opad* harus sama.

Langkah – Langkah yang dilakukan untuk mengimplementasikan HMAC adalah sebagai berikut :

1. Jika kunci (K) memiliki panjang yang sama dengan *block size* (B), lanjut ke Langkah sekian
2. Jika panjang K < B, tambahkan angka 0 sebanyak (B-K) untuk mendapatkan *string byte* K0, yang panjangnya sama dengan B.
3. Jika panjang K > B, *hash* K untuk mendapatkan L *string byte*. Setelah itu, *append* dengan angka 0 sebanyak (L-B) untuk mendapatkan *string byte* K0 yang panjangnya sama dengan B.
4. Lakukan XOR antara K0 dengan *ipad* untuk menghasilkan *string byte* sepanjang B
5. *Append string* ‘text’ ke dalam hasil dari Langkah 4
6. *Hash* hasil dari Langkah ke 5 dengan fungsi *hash* yang sudah ditentukan
7. Lakukan XOR antara K0 dengan *opad*
8. *Append string* dari Langkah ke 6 dengan *string* dari langkah ke 7
9. *Hash* hasil dari langkah ke 8 dengan *hash function* yang sudah ditentukan
10. Ambil *leftmost byte* sebanyak t dari hasil langkah ke 9



III. PENERAPAN ALGORITMA

Dalam makalah ini, penulis akan mengimplementasikan algoritma HMAC untuk menjaga integritas data di dalam DBMS dengan menggunakan bahasa pemrograman *Python*.

A. *Rancangan Solusi*

Program dibuat dengan menggunakan bahasa pemrograman *Python* dan DBMS berupa PostgreSQL. Program akan dibuat dengan CLI (*Command-Line interface*). Program ditulis dengan spesifikasi kebutuhan fungsional sebagai berikut :

- Program dapat menerima pesan dari pengguna dalam bentuk *string*
- Program dapat menghitung nilai MAC dari masukan yang diberikan dengan menggunakna fungsi *hash* seperti SHA
- Program akan menyimpan nilai MAC ke dalam DBMS
- Ketika ada pihak yang ingin mengambil data, program dapat mencocokan MAC yang dihitung dengan MAC yang disimpan di dalam *database*

Flow program adalah sebagai berikut :

1. Program meminta pengguna memasukkan pesan yang mau dihitung nilai MAC-nya
2. Program menghitung nilai MAC dari pesan yang diberikan oleh pengguna
3. Program menyimpan pesan yang sudah ditambahkan dengan nilai MAC nya di dalam *database*
4. Pengguna dapat memilih *option* untuk melihat data yang sudah disimpan di dalam DBMS
5. Ketika pengguna mengirim permintaan untuk membaca data di dalam DBMS, program menghitung nilai MAC dan membandingkannya dengan nilai MAC yang disimpan di dalam *database*
6. Jika nilai MAC nya berbeda, program akan menampilkan pesan *error* yang menandakan bahwa data di dalam DBMS sudah diubah oleh pihak eksternal.

B. *Implementasi Algoritma*

Karena makalah ini menitikberatkan pada HMAC, dan bukan kepada *hash function* yang digunakan, penulis menggunakan *library* dari *Python* yang menyediakan fungsi *hash* sha256.

Sebelum mengimplementasikan algoritma HMAC, penulis membuat beberapa *utilities function* yang digunakan untuk membuat koneksi ke *database.*

```
def connect_db():
    """Connect to Database"""
    conn = psycopg2.connect(
        dbname = "hmac",
        user="postgres",
        password="postgres",
        host="localhost",
        port="5432"
    )
    return conn

def create_table():
    """Create messages table"""
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS messages (
        id SERIAL PRIMARY KEY,
        message TEXT NOT NULL,
        mac TEXT NOT NULL
        )
    """)
    conn.commit()
    cursor.close()
    conn.close()
```

Dua fungsi di atas berguna untuk menghubungkan program ke *database* dan juga membuat tabel bernama "messages" yang memiliki beberapa atribut seperti id, *message*, dan *hmac*.

```
def hash_function(data):
    return hashlib.sha256(data).digest()
```

Fungsi di atas adalah *hash function* yang menggunakan *library sha256* yang dimiliki oleh Python. Fungsi ini menerima parameter berupa *message* atau *data* yang ingin di *hash* dan mengembalikannya.

```
def hmac_custom(key, message):
    if len(key) > BLOCK_SIZE:
        key = hash_function(key)
    if len(key) < BLOCK_SIZE:
        key = key.ljust(BLOCK_SIZE, b'\0')

    K0_ipad = bytes((x ^ IPAD) for x in key)

    inner_hash_input = K0_ipad + message.encode()

    inner_hash_result = hash_function(inner_hash_input)

    K0_opad = bytes((x ^ OPAD) for x in key)

    outer_hash_input = K0_opad + inner_hash_result

    hmac_result = hash_function(outer_hash_input)
    return hmac_result[:OUTPUT_SIZE]
```

Fungsi di atas adalah implementasi dari algoritma HMAC yang dijelaskan pada bagian dasar teori. Fungsi ini menerima 2 parameter, yaitu data yang ingin di *hash* dan juga *key* yang sudah disepakati untuk digunakan. Setelah itu, fungsi ini akan mengecek apakah panjang kunci sama dengan *block size* atau tidak. *Block size* di sini bergantung pada *hash function* yang digunakan.

Setelah itu, program akan melakukan operasi XOR antara *key* dengan *ipad (inner padding). Message* yang ingin di *hash* akan ditambahkan (*append*) ke dalam hasil dari operasi XOR tersebut. Langkah selanjutnya adalah menambahkan hasil *append* tadi ke dalam *opad (outside padding).* Hasil dari langkah sebelumnya perlu di *hashing* kembali untuk mendapatkan jawaban final.

```
def store_message(message, key):
    hmac_value = hmac_custom(key.encode(), message)
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute("INSERT INTO messages (message, mac) VALUES (%s, %s)", (message, hmac_value.hex()))
    conn.commit()
    cursor.close()
    conn.close()
```

Fungsi di atas adalah fungsi yang digunakan untuk menyimpan pesan ke dalam DBMS. Dapat dilihat bahwa yang disimpan di dalam DBMS hanya pesan (*message*) dan juga *hmac value (mac).*

```
def verify_message(message, key):
    hmac_value = hmac_custom(key.encode(), message).hex()
    print("ini hmac_value", hmac_value)
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute("SELECT mac FROM messages WHERE message = %s", (message,))
    row = cursor.fetchone()
    cursor.close()
    conn.close()
    print("Ini row", row)
```

```
        if row is None:
            print("No such message in the database.")
        elif hmac_value == row[0]:
            print("Data is valid and has not been altered.")
        else:
            print("Data integrity check failed. Data may have been altered.")
```

Fungsi di atas adalah algoritma yang digunakan untuk memverifikasi apakah pesan mengalami perubahan atau tidak. Hal itu dapat dideteksi dengan membandingkan nilai MAC yang disimpan dengan MAC yang didapatkan dari hasil perhitungan. Jika hasilnya berbeda dengan yang disimpan di dalam database, maka data di dalam database bisa saja sudah mengalami perubahan. Namun, kalua mislanya MAC *value* nya sama, maka data masih adapt dijamin integritasnya.

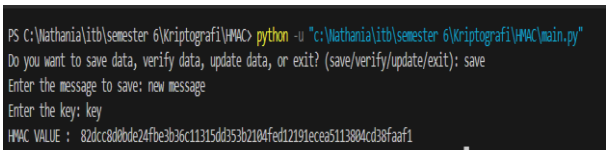
```
def update_message(id, new_message, key):
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute("SELECT mac FROM messages WHERE id = %s", (id,))
    row = cursor.fetchone()

    if row is None:
        print("Old message does not exist in the database.")
    else:
        new_hmac_value = hmac_custom(key.encode(), new_message).hex()

        cursor.execute("UPDATE messages SET message = %s, mac = %s WHERE id = %s", (new_message, new_hmac_value, id))
        conn.commit()
        print("Message updated successfully.")
    cursor.close()
    conn.close()
```

Fungsi di atas adalah fungsi yang mendemonstrasikan bagaimana kalua ada yang memperbaharui pesan secara illegal. Dapat dilihat, ketika pesan berubah, maka nilai HMAC nya juga akan berubah dan akan diperbaharui di DBMS. Sehingga, ketika ingin di verifikasi, akan terdeteksi bahwa terdapat perbedaan antara pesan yang dikirim dengan pesan yang diterima.

C. Hasil Pengujian

Pengujian dilakukan dengan cara menjalankan program dan memasukkan sebuah pesan dan sebuaah kunci yang ingin digunakan. Setelah itu, pesan akan disimpan di dalam DBMS beserta dengan nilai MAC yang didapatkan.

- Pengujian Penyimpanan Data
  Untuk menyimpan data, masukkan perintah seperti di bawah ini.

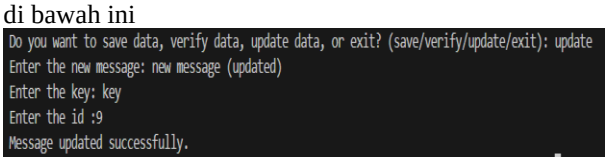


Untuk memastikan bahwa data tersimpan dengan baik, kita dapat melihat isi data di dalam DBMS.

Dari gambar di atas, dapat dilihat bahwa tabel *messages* sudah berisikan data *plain message* dan nilai MAC yang didapatkan dari perhitungan.

- Pengujian Memperbaharui Data

Untuk memperbaharui data, masukkan perintah seperti di bawah ini

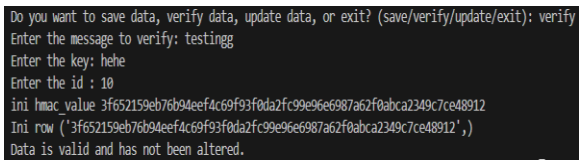


Ketika proses ini selesai, maka value dari MAC nya juga akan berubah. Inilah yang menyebabkan nilai MAC hasil perhitungan nanti akan berbeda dan digunakan untuk mendeteksi bahwa data sudah tidak valid lagi.

- Pengujian Verifikasi Integritas Data

Untuk menguji integritas data, masukkan perintah seperti di bawah ini.

Kasus di atas adalah contoh dimana integritas data gagal dipastikan karena *value* dari MAC yang dihitung berbeda dengan *value* MAC yang tersimpan di DBMS.



Gambar di atas menunjukkan kasus dimana verifikasi berhasil dan integritas data ditemukan masih terjaga karena nilai HMAC nya yang tidak berubah sama sekali.

IV. KESIMPULAN DAN SARAN

Berdasarkan implementasi yang sudah dilakukan, penulis mengambil kesimpulan bahwa algoritma HMAC merupakan salah satu algoritma yang dapat digunakan untuk memastikan bahwa data yang diambil masih valid. Namun, salah satu kelemahan dari HMAC adalah *message* yang dikirimkan tidak dalam bentuk yang sudah terenkripsi. Selain itu salah satu kelemahan dari algoritma ini adalah perlu kesepakatan mengenai kunci dan fungsi yang akan digunakan antara pengirim dan penerima

Penulis memberikan beberapa saran untuk penerapan algoritma HMAC dalam menjaga integritas data, yaitu, melakukan *hashing* terlebih dahulu terhadap *message*, sehingga kerahasiaan pesan juga tetap dapat terjaga. Selain itu, hal lain yang dapat dilakukan adalah tidak menggunakan kunci simetris, tapi dapat mencoba menggunakan kunci asimetris, sehingga pertukaran kunci tidak dilakukan melalui *channel* yang tidak aman.

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi resolution TIFF or EPS file with all fonts embedded) because this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord "Format" pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.

UCAPAN TERIMA KASIH

Penulis ingin memanjatkan puji dan Syukur kepada Tuhan yang Maha Esa atas berkat dan rahmatnya sehingga makalah ini yang berjudul "Implementasi Algoritma Message Authenticaion Code (MAC) untuk Menjaga Integritas Data dalam DBMS" dapat diselesaikan dengan baik dan lancar. Penulis juga mengucapkan terima kasih kepada bapak Rinaldi Munir sebagai pengampu mata kuliah IF4020 Kriptografi tahun akademik 2023/2024 semester genap yang sudah membagikan ilmunya kepada seluruh mahasiswa Teknik Informatika ITB angkaran 2020 dan 2021. Ucapan terima kasih tak lupa penulis ucapkan kepada orang tua dan keluarga yang selalu mendukung penulis, dan teman – teman terdekat penulis yang memberikan dukungan baik dari segi mental maupun materi.

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/28-MAC-2024.pdf

[2] https://medium.com/@short_sparrow/how-hmac-works-step-by-step-explanation-with-examples-f4aff5efb40e

[3] https://www.geeksforgeeks.org/what-is-hmachash-based-message-authentication-code/

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

Nathania Calista Djunaedi (13521139)