Implementasi Fungsi *Hash* dan Enkripsi Simetris Sebagai Teknik Autentikasi dan Pengecekan Integritas Pesan

Go Dillon Audris – 13521062
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13521062@std.stei.itb.ac.id

***Abstrak*—Kriptografi sebagai ilmu atau seni menjaga keamanan pesan dapat menyediakan empat buah layanan. Empat layanan tersebut adalah kerahasiaan pesan (*confidentiality*), keaslian pesan (*integrity*), keaslian pengirim dan penerima pesan (*authentication*), serta anti-penyangkalan (*non-repudiation*). Keaslian pesan atau integritas data merupakan properti dimana pesan tidak diubah dan memiliki konten yang sama dengan yang ditulis pengirim ketika sampai kepada penerima. Sementara itu, autentikasi merupakan properti di mana hanya pengirim dan penerima pesan yang dapat mengetahui isi pesan dan tidak ada pihak lain yang dapat berpura-pura mengirimkan pesan kepada penerima. Data integritas dan autentikasi merupakan dua properti penting yang harus dipenuhi. MAC atau *Message Authentication Code* merupakan salah satu cara untuk menyediakan layanan data integritas dan autentikasi dengan melekatkan kode kecil berukuran tetap pada pesan. Namun, selain MAC, terdapat cara lain untuk menyediakan layanan integritas data dan autentikasi, yaitu dengan memanfaatkan fungsi *hash* dan enkripsi simetri.**

***Kata kunci*—integritas data, autentikasi, *MAC*, fungsi *hash*, enkripsi simetri**

I. PENDAHULUAN

Komunikasi merupakan aspek yang telah menjadi kebutuhan penting sejak zaman dahulu. Komunikasi dilakukan oleh manusia untuk menjalin hubungan, bertukar informasi, membangun komunitas, dan berbagai hal lainnya. Media komunikasi menjadi bagian penting dalam cara manusia berkomunikasi. Awalnya manusia berkomunikasi hanya dengan cara lisan atau tatap mata. Selanjutnya manusia mulai mengenal cara menulis dan membuat tulisan. Dari sana, media komunikasi semakin beragam, mulai dari burung merpati, kentongan, surat pos, telepon, dan lain sebagainya. Pada era digital sekarang, sudah sangat banyak media komunikasi yang dapat digunakan oleh manusia.

Era digital membawa kemudahan dalam melakukan komunikasi. Teknologi seperti telepon, *video-call*, *e-mail*, media sosial, dan lain-lain berkembang secara cepat. Selain memudahkan komunikasi, teknologi tersebut juga mempercepat dan mempermudah akses terhadap informasi. Orang-orang yang jauh dari kita dapat dengan mudah mendapatkan informasi tertentu serta mengetahui hal baru yang terjadi di dunia. Data pribadi kita pun juga dengan mudah mengalir dalam dunia maya. Tak hanya mengalir lewat media komunikasi, namun juga lewat aplikasi yang membutuhkan data pribadi kita untuk beroperasi.

Kemudahan akses informasi dan data pribadi membawa tantangan sendiri dalam era digital karena merupakan celah baru untuk berbagai ancaman keamanan siber. Berbagai kejahatan seperti penyadapan, peretasan, dan serangan lainnya dapat dilakukan untuk membocorkan informasi. Informasi dan data pribadi kemudian dapat dicuri oleh penjahat siber, yang kemudian dapat dimanipulasi, disalahgunakan, hingga diperjualbelikan kepada oknum yang membutuhkan. Kebocoran informasi dapat terjadi karena beberapa faktor, namun terutama karena kurang kuat dan amannya suatu sistem.

Kriptografi hadir untuk menyelesaikan permasalahan ini. Kriptografi merupakan sebuah ilmu atau seni untuk menjaga keamanan pesan sekaligus informasi dan pesan yang terkandung di dalamnya. Kriptografi telah menjadi kebutuhan fundamental di era sekarang untuk melindung informasi serta privasi, dan membangun kepercayaan digital. Dalam prinsipnya, kriptografi mampu untuk memberikan empat layanan. Kerahasiaan pesan atau *confidentiality*, di mana isi dari pesan tidak dapat diketahui oleh pihak-pihak yang tidak berwenang. Selanjutnya adalah keaslian pesan atau *integrity*, di mana isi pesan dipastikan tidak berubah dan memiliki konten yang sama dengan yang ditulis pengirim ketika sampai kepada penerima. Keaslian pengirim dan penerima atau *authentication*, yang merupakan kepercayaan bahwa pesan dikirim oleh pengirim asli dan bukan oleh pihak lain yang menyerupai. Autentikasi juga membuktikan bahwa hanya pengirim dan penerima yang dapat mengakses isi dari pesan. Layanan terakhir adalah anti-penyangkalan atau *non-repudiation*, di mana pengirim pesan tidak dapat menyangkal bahwa dia merupakan orang yang mengirim suatu pesan kepada penerima.

Dua jenis layanan yang telah dijelaskan, yaitu integritas data dan autentikasi dapat diberikan oleh suatu algoritma yang disebut sebagai MAC atau *Message Authentication Code*. MAC merupakan kode kecil berukuran tetap yang diletakkan pada akhir pesan. Kode kecil ini dibangkitkan dengan memanfaatkan isi pesan serta suatu kunci. Penerima pesan dapat menggunakan kode kecil ini untuk mengotentikasi pengirim dan memeriksa integritas pesan. Dalam implementasinya, MAC dapat dihasilkan dengan dua cara, yaitu algoritma berbasis *block-cipher* atau algoritma berbasis fungsi *hash* satu arah yang ada.

Selain dengan memanfaatkan MAC, terdapat beberapa teknik lain yang dapat digunakan yang juga memberikan layanan autentikasi dan integritas data. Teknik tersebut diantaranya adalah dengan memanfaatkan suatu nilai rahasia yang disepakati, menggunakan enkripsi dengan kriptografi kunci publik, ataupun memanfaatkan fungsi *hash* dan enkripsi simetri. Makalah ini akan membahas mengenai teknik pemanfaatan fungsi *hash* dan enkripsi simetri untuk menyediakan layanan autentikasi dan integritas data.

II. LANDASAN TEORI

A. Enkripsi Simetri

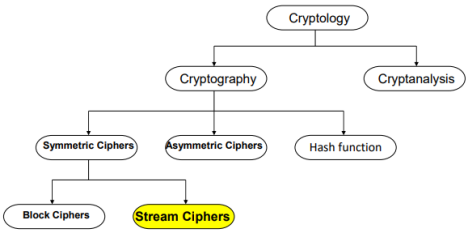
Enkripsi simetri atau kriptografi simetri atau kriptografi kunci tunggal adalah kriptografi yang menggunakan kunci yang sama dalam proses enkripsi dan dekripsinya. Kunci ini harus dijaga kerahasiaannya dan dibagikan secara aman di antara pengirim dan penerima. Kunci dapat berupa angka, kata-kata, maupun sekumpulan karakter acak.

Desain dari enkripsi simetri menyebabkan enkripsi ini memiliki beberapa kelebihan dan kekurangan. Kelebihannya antara lain [1]:

- Algoritma pada enkripsi simetri dirancang sehingga membutuhkan waktu yang singkat untuk melakukan enkripsi atau dekripsi.
- Ukuran atau panjang kunci relatif pendek.
- Algoritma pada enkripsi simetri dapat dimanfaatkan untuk membangkitkan bilangan acak (*random number generator*).
- Algoritma pada enkripsi simetri dapat disusun untuk menghasilkan algoritma yang lebih kuat.
- Layanan autentikasi dapat langsung dilakukan ketika cipherteks diterima. Hal ini karena kunci hanya diketahui oleh pengirim dan penerima saja.

Adapun kekurangan dari enkripsi simetri adalah [1]:

- Kunci harus dikirim antara kedua pihak melalui saluran yang aman untuk menjaga kerahasiaan kunci.
- Kunci harus sering diubah dalam setiap sesi komunikasi untuk mencegah kunci dapat diketahui lewat serangan seperti *known-plaintext attack* atau *known-ciphertext attack*.

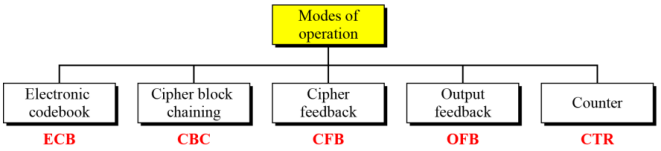


Gambar 2.1. Bidang kriptografi
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/10-Kripto-modern-2024.pdf)

Enkripsi simetri dapat dibedakan menjadi dua macam *cipher*, yaitu *cipher* alir dan *cipher* blok. *Cipher* alir atau *stream cipher* merupakan *cipher* yang beroperasi pada bit secara individual. Enkripsi dan dekripsi pesan pada *stream cipher* dilakukan bit per bit dengan operasi XOR. Sementara itu, *cipher* blok atau *block cipher* merupakan *cipher* yang beroperasi pada blok-blok atau sekumpulan bit. Ukuran blok dapat berupa 32 bit, 64 bit, 128 bit, dan seterusnya. Enkripsi dan dekripsi pesan pada *block cipher* dilakukan untuk setiap blok bit [2].

B. Block Cipher (Cipher Blok)

Block cipher merupakan jenis enkripsi simetri yang melakukan pemrosesan enkripsi atau dekripsi dalam sebuah blok bit. Ukuran blok bit dapat bervariasi, namun umumnya terdiri atas 64 bit, 128 bit, dan 256 bit. Pada *block cipher*, proses enkripsi dilakukan pada setiap blok plainteks dengan memanfaatkan bit-bit kunci, dan menghasilkan cipherteks dengan panjang yang sama dengan plainteks [3].



Gambar 2.2. Mode operasi *block cipher*
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/12-Block-Cipher-Bagian1-2024.pdf)

Block cipher memiliki beberapa jenis mode operasi. Mode operasi berkaitan dengan cara blok pesan dioperasikan sebelum diproses dengan fungsi enkripsi atau fungsi dekripsi. Mode operasi ini ditambahkan untuk memperkuat keamanan dan kerahasiaan pesan. Ada lima jenis mode operasi *block cipher* [3]:

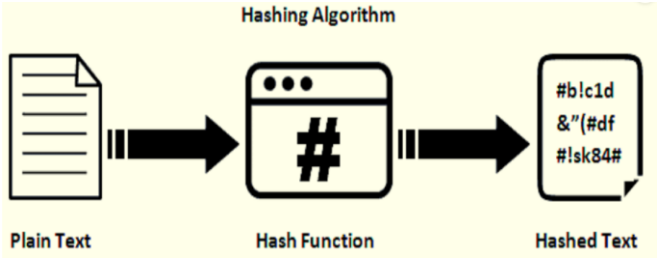
1. *Electronic Code Book* (ECB):
   Pada mode operasi ECB, setiap blok plainteks akan dienkripsi secara individual dan independen menjadi block cipherteks. Mode operasi ECB memiliki kelebihan karena sederhana dan efisien. Selain itu, kesalahan pada suatu blok plainteks hanya memengaruhi blok cipherteks yang bersangkutan. Namun, mode ini tidak aman digunakan pada data berulang karena pola plainteks akan terlihat pada cipherteks. Pihak lain juga dapat memanipulasi cipherteks untuk memanipulasi penerima pesan.
2. *Cipher Block Chaining* (CBC):
   Pada mode operasi CBC, blok cipherteks tidak hanya bergantung pada blok plainteks bersangkutan, namun juga pada hasil enkripsi dari blok cipherteks sebelumnya. Hasil enkripsi blok sebelumnya akan diumpan-balikkan ke dalam proses enkripsi blok selanjutnya. Hal ini menciptakan ketergantungan atau dependensi antar blok. Dengan memanfaatkan mode operasi CBC, keamanan dan kerahasiaan pesan lebih baik karena pola plainteks dapat disembunyikan. Kekurangan dari mode operasi ini adalah kesalahan bit pada suatu blok plainteks akan menyebabkan kesalahan pada blok cipherteks yang bersangkutan dan merambat ke blok cipherteks setelahnya.
3. *Cipher Feedback* (CFB):
   Mode operasi CFB muncul untuk mengatasi kekurangan mode operasi CBC ketika digunakan pada pesan yang berukuran kurang dari satu blok. Pada mode operasi CFB, pesan dienkripsi dalam ukuran yang lebih kecil dibanding ukuran blok. CFB memanfaatkan suatu *shift register*, dimana bagian dari cipherteks akan menjadi bagian dari *shift register* untuk proses enkripsi selanjutnya. Mode operasi CFB memiliki kelemahan yang sama dengan mode CBC.
4. *Output Feedback* (OFB):
   Mode operasi OFB memiliki karakteristik yang serupa dengan mode operasi CFB. Pada CFB, *shift register* akan ditambahkan dengan bagian dari cipherteks. Sedangkan, pada mode OFB, *shift register* akan ditambahkan dengan bagian dari hasil enkripsi *shift register* dengan fungsi enkripsi.
5. *Counter Mode* (CTR):
   Pada mode operasi CTR, tidak dilakukan *chaining* seperti pada CBC. Pada mode ini, terdapat sebuah *counter* yang merupakan sebuah nilai berupa blok bit dengan ukuran sama dengan blok plainteks. Nilai *counter* akan di-*increment* dalam setiap proses enkripsi. Kelebihan mode operasi ini adalah sederhana dan efisien. Selain itu, mode operasi CTR hanya memerlukan fungsi enkripsi saja.

Kekurangan mode operasi ini adalah nilai awal dari *counter* yang tidak boleh dapat dipredikasi dengan mudah.

Terdapat beberapa konsep yang harus diperhatikan ketika mendesain dan membuat suatu *block cipher*. Di antaranya adalah prinsip *confusion-diffusion* Shanon, teknik substitusi dan permutasi, *iterated cipher* atau *cipher* berulang, pembangkitan kunci putaran, serta ukuran atau panjang blok dan kunci [4]. Terdapat beberapa *block cipher* yang dikembangkan dan telah digunakan secara luas, di antaranya adalah:

1. *Data Encryption Standard* (DES):
   Dikembangkan pada tahun 1970-an, DES merupakan *block cipher* pertama yang digunakan secara luas di dunia. DES melakukan enkripsi terhadap blok berukuran 64 bit dengan melakukan 16 kali putaran *enciphering*. Kunci DES berukuran 56 bit. Sekarang, DES sudah jarang digunakan karena ukuran kuncinya yang dianggap terlalu kecil dan sudah tidak aman.
2. *Advanced Encryption Standard* (AES):
   Block cipher yang banyak digunakan sekarang dan dikenal karena keamanan, efisiensi, dan keserbagunaannya. Didasari oleh algoritma Rijndael yang dikembangkan oleh Vincent Rijmen dan Joan Daemen. Algoritma Rijndael mendukung panjang kunci 128 hingga 256 bit untuk enkripsi berbagai ukuran blok bit.
3. *Blowfish*:
   Block cipher yang dikembangkan oleh Bruce Schneier pada tahun 1993. Bekerja untuk memproses blok berukuran 64 bit dan mendukung kunci dengan panjang 32 hingga 448 bit. *Blowfish* didesain untuk kecepatan pada prosesor 32-bit, dan terkenal karena efisiensi dalam melakukan *key scheduling*.

C. Fungsi Hash



Gambar 2.3. Ilustrasi cara kerja fungsi *hash*
(Sumber: https://www.opengrowth.com/blogs/what-is-a-hash-function-in-cryptography)

Fungsi *hash* merupakan sebuah fungsi yang mengkompresi pesan berukuran sembarang menjadi sebuah *string* berukuran tetap. *Output* dari fungsi *hash* disebut sebagai pesan-ringkas (*message-digest*) atau nilai *hash* (*hash value*) [5]. Fungsi *hash* memiliki beragam aplikasi dan dimanfaatkan untuk berbagai kebutuhan seperti tanda tangan digital, autentikasi, dan pembangkitan bilangan acak.

Fungsi *hash* yang baik harus memiliki sifat-sifat berikut [5]:

1. *One-way*:
   Fungsi *hash* harus berlaku satu arah saja, artinya pesan yang telah diubah ke dalam *message-digest*, tidak dapat diubah kembali menjadi pesan asli. Sifat ini menjaga keamanan dari pesan karena tidak memungkinkan pihak tidak berwenang untuk menerka pesan asli.
2. *Collision Resistance*:
   Fungsi *hash* yang baik harus juga tahan terhadap kolisi. Artinya, diberikan sebuah *message-digest*, akan sangat sulit untuk menemukan dua buah pesan berbeda yang memiliki *message-digest* tersebut.
3. *Preimage Resistance*:
   Untuk sembarang *message-digest*, akan sulit ditemukan pesan asli dari *message-digest* tersebut.
4. *Second Preimage Resistance*:
   Untuk sebuah sembarang *message-digest* dan pesan yang bersangkutan, akan sulit ditemukan pesan lain dengan *message-digest* yang serupa pula.

Fungsi *hash* memiliki banyak sekali peran karena sifat-sifat yang diberikannya, beberapa di antaranya adalah [5]:

1. Menjaga integritas data:
   Fungsi *hash* sangat peka dengan perubahan 1 bit dalam pesan. Ketika terjadi perubahan, maka *message-digest* yang dihasilkan oleh fungsi *hash* akan sangat berbeda. Hal ini membantu untuk melakukan pengecekan integritas data yang diterima.
2. Menghemat waktu pengiriman:
   Ukuran *message-digest* yang kecil dan tetap membuatnya lebih efisien untuk dikirimkan dalam keperluan atau kebutuhan tertentu dibandingkan mengirimkan pesan asli.
3. Menormalisasi panjang pesan bervariasi:
   Ukuran pesan yang bervariasi menjadikannya tantangan sendiri untuk keperluan penyimpanan. Adanya fungsi *hash* membuat panjang pesan dapat diubah menjadi berukuran tetap sehingga memudahkan mekanisme penyimpanan.

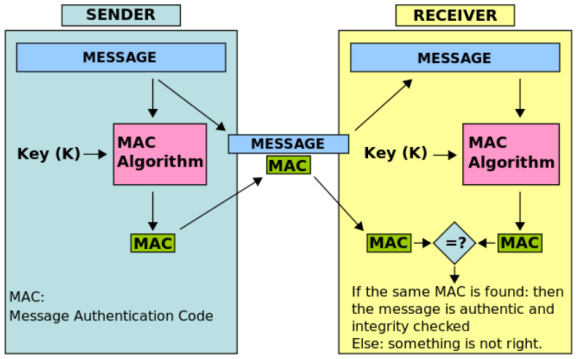
Terdapat beberapa macam fungsi *hash* yang digunakan secara luas pada berbagai ranah aplikasi:

1. MD5:
   MD5 merupakan fungsi *hash* satu arah yang dibuat oleh Ron Rivest. MD5 dikembangkan untuk memperbaiki fungsi MD4 yang telah diketahui kolisinya. Fungsi MD5 akan menerima pesan berukuran sembarang dan menghasilkan *message-digest* berukuran 128 bit. Dalam algoritma ini, secara garis besar, beberapa langkah yang harus dilakukan untuk membuat *message-digest* adalah:
   a. Melakukan penambahan bit-bit *padding*.
   b. Melakukan penambahan nilai panjang pesan semula.
   c. Menginisialisasi penyangga atau *buffer* MD.
   d. Melakukan pengolahan pesan dalam blok berukuran 512 bit.
2. SHA-256:
   SHA-256 atau *Secure Hash Algorithm* 256 merupakan fungsi *hash* yang banyak digunakan sekarang dan dikenal karena sifat *collision resistance*-nya. Fungsi SHA-256 menghasilkan *message-digest* berukuran 256 bit dari sembarang masukan pesan.

D. Message Authentication Code (MAC)

Message Authentication Code atau MAC merupakan teknologi yang menyediakan layanan autentikasi dan pengecekan integritas data dalam kriptografi. MAC adalah suatu kode kecil berukuran tetap yang dapat dilekatkan pada pesan asli untuk kemudian dikirimkan kepada penerima. Dengan memanfaatkan kode kecil ini, penerima dapat mengautentikasi pengirim pesan sekaligus memerika integritas pesan yang diterima [6].

Algoritma MAC berbeda dengan fungsi *hash* pada umumnya. Hal ini karena proses pembangkitan MAC memanfaatkan kunci unik untuk menghasilkan kode. Umumnya, kunci ini akan disepakati oleh pengirim dan penerima dan menjadi salah satu komponen penting dalam layanan autentikasi dan pengecekan integritas yang disediakan MAC.



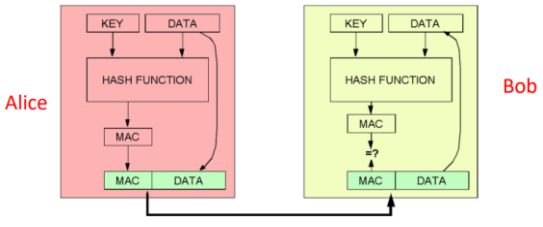
Gambar 2.4. Ilustrasi cara kerja algoritma MAC
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/28-MAC-2024.pdf)

Algoritma MAC memiliki banyak kesamaan dengan fungsi *hash*. Seperti fungsi *hash*, algoritma MAC memiliki sifat *irreversible*. Algoritma MAC juga memiliki potensi kolisi dimana lebih dari satu pesan memiliki kode MAC yang sama. Namun, kolisi sangat jarang terjadi pada algoritma MAC. Algoritma MAC yang baik harus memenuhi beberapa persyaratan berikut [6]:

1. Diberikan sebuah pesan dan kode MAC-nya, akan sangat sulit menemukan pesan lain dengan kode MAC yang sama (*Second Preimage Resistance*).
2. Kode MAC dari berbagai pesan harus terdistribusi secara *uniform* atau merata.
3. Pembangkitan kode MAC seharusnya bergantung kepada semua bit-bit pesan asli.

Terdapat 2 cara untuk menghasilkan kode MAC [6]:

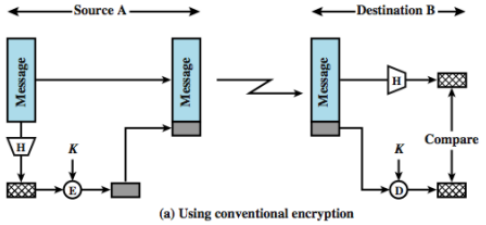
1. Algoritma berbasis *block-cipher*:
   Pada algoritma MAC berbasis *block-cipher*, kode MAC dibangkitkan dengan memanfaatkan *block-cipher* pada mode operasi CBC atau CFB. Pada algoritma ini, hasil enkripsi blok terakhir dari *block cipher* akan menjadi kode MAC yang dilekatkan pada pesan. Salah satu algoritma MAC berbasis *block-cipher* yang terkenal adalah *Data Authentication Algorithm* (DAA) yang berbasis DES-CBC. Pada DAA, panjang kode MAC akan sama dengan panjang blok cipherteks yaitu 64 bit. Kunci MAC yang digunakan juga merupakan kunci DES yang berukuran 56 bit.
2. Algoritma berbasis fungsi *hash* satu arah:
   Pada algoritma ini, fungsi *hash* seperti MD5 dan SHA dimanfaatkan untuk membangkitkan kode MAC. Pesan asli dan kunci rahasia awalnya akan digabungkan menjadi satu. Hasil penggabungan inilah yang kemudian menjadi masukan bagi fungsi *hash*. *Message-digest* yang dihasilkan oleh fungsi *hash* menjadi kode MAC yang dilekatkan pada pesan asli. Oleh karena itu, panjang kode MAC bergantung pada fungsi *hash* yang digunakan. Sebagai contoh, penggunaan fungsi SHA-1 akan menghasilkan kode MAC berukuran 160 bit.



Gambar 2.5. Ilustrasi cara kerja algoritma MAC berbasis fungsi *hash*
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/28-MAC-2024.pdf)

III. ANALISIS PERMASALAHAN

Penggunaan *Message Authentication Code* untuk memberikan layanan autentikasi dan pengecekan integritas pesan telah terbukti efektif. Namun sebenarnya, selain menggunakan MAC, terdapat teknik lain yang dapat digunakan untuk memberikan kedua jenis layanan tersebut. Beberapa di antaranya adalah menggunakan nilai rahasia yang telah disepakati, menggunakan kriptografi kunci publik, dan memanfaatkan fungsi *hash* dan enkripsi simetri.



Gambar 3.1. Penggunaan fungsi *hash* dan enkripsi simetri
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/28-MAC-2024.pdf)

Makalah ini bertujuan untuk mengimplementasikan teknik pemanfaatan fungsi *hash* dan enkripsi simetri untuk memberikan layanan autentikasi dan pengecekan integritas pesan selayaknya algoritma MAC. Oleh karena itu, berdasarkan skema pada gambar 3.1, berikut merupakan garis besar dari program yang akan diimplementasikan:

1. Praproses:
   Kunci rahasia yang akan digunakan oleh pengirim dan penerima sudah harus diketahui dan dibagikan secara aman.
2. Tahap pengiriman pesan:
   Pengirim akan melekatkan suatu kode pada pesan dengan memanfaatkan fungsi *hash* dan enkripsi simetri.
   - Program akan menerima kunci rahasia dan pesan asli dari pengirim.
   - Program akan menghitung nilai *message-digest* dari pesan dengan memanfaatkan suatu fungsi *hash*.
   - Nilai *message-digest* kemudian akan dienkripsi dengan suatu algoritma enkripsi simetri memanfaatkan kunci rahasia.
   - Hasil enkripsi *message-digest* kemudian dilekatkan pada pesan dan dikirimkan lewat saluran komunikasi.
3. Tahap penerimaan pesan:
   Penerima menerima pesan dari saluran komunikasi dan akan mengecek autentikasi serta integritas pesan.
   - Program menerima kunci rahasia dan pesan yang diterima.
   - Program memisahkan pesan asli dan enkripsi *message-digest* dari pesan yang diterima.
   - Program akan melakukan dekripsi terhadap enkripsi *message-digest* dengan algoritma enkripsi simetri memanfaatkan kunci rahasia.
   - Program akan melakukan *hashing* terhadap pesan asli memanfaatkan fungsi *hash* yang sama.
   - Nilai hasil dekripsi dibandingkan dengan nilai hasil fungsi *hash*. Jika sama, maka diketahui bahwa

pengirim pesan asli dan isi pesan tidak berubah.

IV. IMPLEMENTASI

Pada makalah ini, penulis membuat program dalam bahasa Python. Bahasa Python dipilih sebagai bahasa implementasi karena adanya pustaka yang memudahkan proses implementasi. Untuk kemudahan implementasi, akan dimanfaatkan pustaka fungsi *hash* SHA-256 dan pustaka algoritma AES dengan memanfaatkan Fernet. Berikut merupakan penjelasan lengkap implementasi program.

A. *Hash Function*

Fungsi *hash* diimplementasikan ke dalam kelas *Hash*. Kelas ini diciptakan untuk menghitung nilai *message-digest* dari suatu pesan. Spesifikasi fungsi *hash* yang digunakan diletakkan sebagai *inner-class* dari kelas *Hash*. Pada implementasi program, dimanfaatkan fungsi *hash* SHA-256.

```python
import hashlib

# Outer class Hash that will do hashing function
class Hash:

    # Inner class depending on the type of hash algorithm used (here is SHA256)
    class SHA256:
        # Hash method, change this according to the hash algorithm used
        def hash(self, message: str):
            sha256_hash = hashlib.sha256(message.encode())
            hashed_message = sha256_hash.digest()

            return hashed_message

    hash_algorithm: SHA256

    # Constructor method
    def __init__(self):
        self.hash_algorithm : Hash.SHA256 = Hash.SHA256()

    # Hash method to call hash function based on inner class
    def hash(self, message: str):
        return self.hash_algorithm.hash(message)
```

Gambar 4.1. Implementasi fungsi *hash*
(Sumber: Dokumen Pribadi)

B. *Symmetric Encryption*

Algoritma enkripsi simetri diimplementasikan ke dalam kelas *Encryption*. Kelas ini diciptakan untuk melakukan proses enkripsi dan dekripsi terhadap masukan dengan memanfaatkan suatu kunci tertentu. Spesifikasi algoritma enkripsi simetri yang digunakan diletakkan sebagai *inner-class* dari kelas *Encryption*. Pada implementasi program, dimanfaatkan algoritma AES dengan Fernet. Fernet menspesifikasikan penggunaan kunci sepanjang 32 *bytes*.

```python
from cryptography.fernet import Fernet
import base64

# Outer class Encryption that will do ciphering
class Encryption:

    # Inner class depending on the type of encryption algorithm used (here is AES using Fernet)
    class AES:
        cipher: Fernet

        def __init__(self, key: str):
            encoded_key = base64.urlsafe_b64encode(key.encode())
            self.cipher : Fernet = Fernet(encoded_key)

        # Encrypt method, change this according to the encryption algorithm used
        def encrypt(self, message: bytes):
            return self.cipher.encrypt(message)

        # Decrypt method, change this according to the encryption algorithm used
        def decrypt(self, message: bytes):
            return self.cipher.decrypt(message)

    key: str
    encryption_algorithm: AES

    # Constructor method
    def __init__(self, key: str):
        self.key : str = key
        self.encryption_algorithm : Encryption.AES = Encryption.AES(self.key)

    # Change key used for encryption
    def change_key(self, key: str):
        self.key = key
        self.encryption_algorithm = Encryption.AES(self.key)

    # Encrypt method
    def encrypt(self, message: bytes):
        return self.encryption_algorithm.encrypt(message)

    # Decrypt method
    def decrypt(self, message: bytes):
        return self.encryption_algorithm.decrypt(message)
```

Gambar 4.2. Implementasi algoritma enkripsi simetris
(Sumber: Dokumen Pribadi)

C. *App*

Kelas *App* diciptakan untuk menangani logika utama dari pemanfaatan fungsi *hash* dan enkripsi simetri untuk menyediakan layanan autentikasi dan pengecekan integritas pesan. Logika tersebut terdiri atas melekatkan kode hasil *hash* dan enkripsi pada pesan, serta melakukan autentikasi dan pengecekan integritas pada pesan yang diterima.

```python
# Embed a "code" to message before delivery using hash and symmetric encryption
def embed(self, message: str):
    raw_message = message

    # Hash the message first
    hashed_message = self.hash.hash(raw_message)

    # Encrypt the hash
    encrypted_hash_message = self.encryption.encrypt(hashed_message)

    # Encode encryption result using base64 and turn it to string
    encoded_code = base64.b64encode(encrypted_hash_message)
    encoded_code = encoded_code.decode()

    # Embed the code to the raw message, return it
    return raw_message + encoded_code
```

Gambar 4.3. Implementasi logika ketika pesan akan dikirim
(Sumber: Dokumen Pribadi)

```python
# Verify a message with code with a key. This will check for message authentication and integrity
def verify(self, message: str):
    message_with_code = message

    if len(message_with_code) <= self.CODE_LENGTH:
        raise Exception("No code found on message")

    try:
        raw_message = message_with_code[:(len(message_with_code)-self.CODE_LENGTH)]
        code = message_with_code[(len(message_with_code)-self.CODE_LENGTH):]

        # Decode the code from base64
        decoded_code = code.encode()
        decoded_code = base64.b64decode(decoded_code)

        # Decrypt the code using encryption algorithm
        decrypted_hash_message = self.encryption.decrypt(decoded_code)

        # Hash the raw message
        hashed_message = self.hash.hash(raw_message)

        # Compare decrypted hash with hashed message
        if decrypted_hash_message == hashed_message:
            return True
        else:
            return False
    except:
        # Catch all exception, either when decoding or decrypting
        # All this indicate message has been tampered or key is wrong
        return False
```

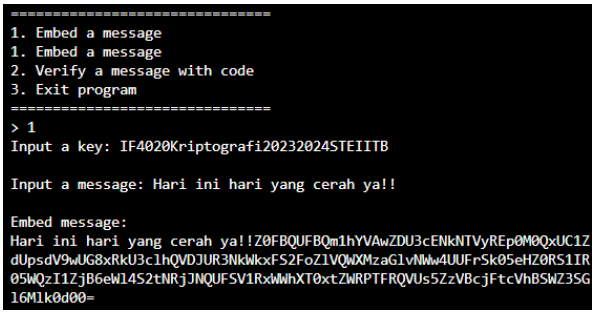
Gambar 4.4. Implementasi logika ketika pesan telah diterima
(Sumber: Dokumen Pribadi)

V. PENGUJIAN

Untuk melakukan pengujian terhadap program yang telah dibuat, akan dilakukan percobaan simulasi pengiriman pesan dan penerimaan pesan. Simulasi penerimaan pesan akan dicoba untuk berbagai kondisi, seperti pesan yang asli, pesan yang dimodifikasi, kunci rahasia yang salah, atau kode yang dimodifikasi.

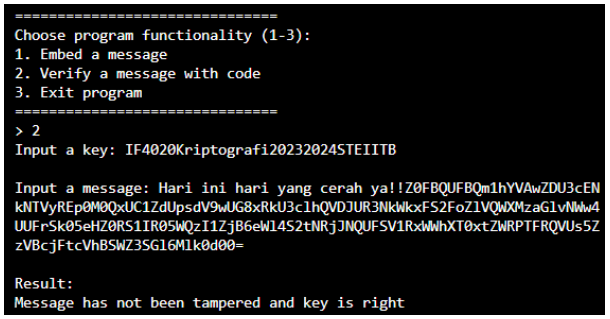
1. Praproses:
   Dimisalkan terjadi komunikasi antara pengirim dan penerima, dimana mereka menyepakati kunci rahasia sepanjang 32 *bytes* berupa “IF4020Kriptografi20232024 STEIITB”. Pengirim akan mengirim sebuah pesan yaitu “Hari ini hari yang cerah ya!!”

2. Tahap Pengiriman Pesan:
   Pengirim akan memasukkan pesan asli dan kunci rahasia ke dalam program, diperoleh pesan sebagai berikut

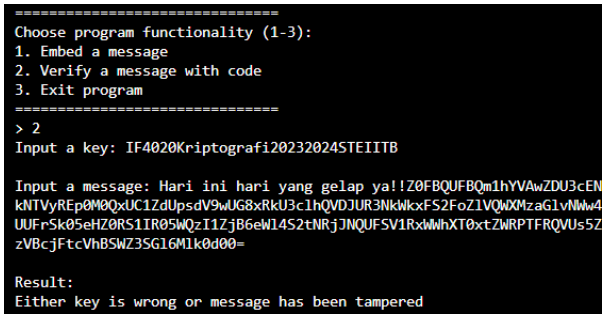


Gambar 5.1. Pengirim ingin mengirimkan sebuah pesan kepada penerima
(Sumber: Dokumen Pribadi)

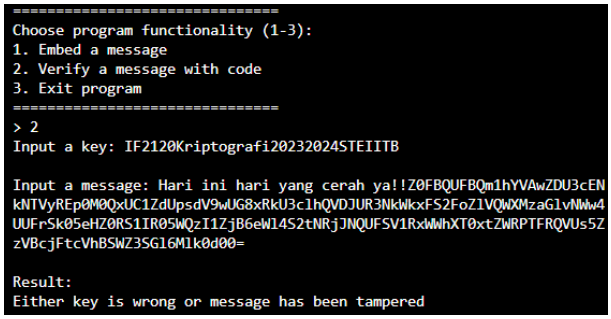
3. Tahap Penerimaan Pesan:
   Penerima pesan kemudian menerima pesan yang dikirim oleh pengirim. Ia akan memeriksa autentikasi dan integritas dari pesan.



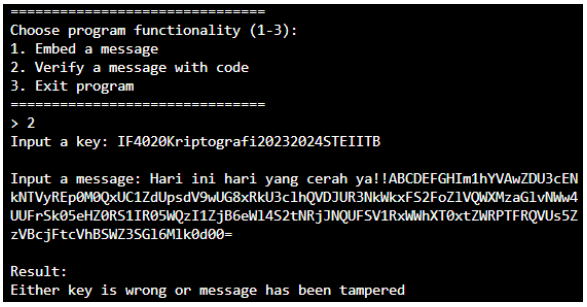
Gambar 5.2. Penerima menerima pesan yang benar (berasal dari pengirim asli dan belum berubah)
(Sumber: Dokumen Pribadi)



Gambar 5.3. Penerima menerima pesan yang salah (konten pesan telah diubah)
(Sumber: Dokumen Pribadi)



Gambar 5.4. Penerima menerima pesan yang benar namun tidak memiliki kunci yang sesuai
(Sumber: Dokumen Pribadi)



Gambar 5.5. Penerima menerima pesan dengan kode yang telah dimodifikasi
(Sumber: Dokumen Pribadi)

Dari hasil pengujian yang telah dilakukan, dapat dilihat bahwa teknik pemanfaatan fungsi *hash* dan enkripsi simetri dapat memberikan layanan autentikasi dan pengecekan integritas pesan. Pesan yang telah dimodifikasi dapat diketahui dan pengirim pesan juga dapat dipastikan keasliannya.

VI. KESIMPULAN

Kriptografi sebagai ilmu dan seni untuk menjaga keamanan pesan telah menjadi kebutuhan fundamental di era digital sekarang. Layanan *confidentiality*, *integrity*, *authentication*, dan *non-repudiation* yang disediakan oleh kriptografi menjadi krusial untuk mencegah terjadinya kejahatan. Layanan autentikasi dan integritas data mampu diberikan oleh algoritma MAC atau *Message Authentication Code*. Namun, selain MAC, terdapat teknik lain yang juga dapat melakukan hal tersebut. Salah satunya adalah dengan memanfaatkan fungsi *hash* dan enkripsi simetri. Teknik ini telah terbukti mampu memberikan hasil yang serupa dengan algoritma MAC dan dapat memberikan layanan autentikasi dan pengecekan integritas data yang baik.

VII. UCAPAN TERIMA KASIH

Puji Syukur kepada Tuhan Yang Maha Esa karena berkat dan kasih karunia-Nya penulis dapat menyelesaikan makalah ini dengan baik dan tanpa kendala yang berarti. Terima kasih juga penulis sampaikan kepada orang tua penulis yang telah mendukung penulis dalam perkuliahan dan dalam penulisan makalah ini. Tak lupa penulis mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengampu mata kuliah IF4020 Kriptografi 2023/2024 yang telah memberikan ilmunya kepada penulis sehingga makalah ini dapat ditulis dengan baik. Penulis berharap bahwa pembahasan pada makalah ini tidak berhenti dan dapat dilanjutkan demi kemajuan bangsa dan negara.

VIII. LAMPIRAN

Penulis berhasil melakukan implementasi lengkap dari penggunaan fungsi *hash* dan enkripsi simetri dalam memberikan layanan autentikasi dan pengecekan integritas pesan dan mengarsipkannya pada sebuah repositori publik yang dapat diakses pada laman Github berikut:

https://github.com/GoDillonAudris512/Hash-and-Symmetric-Encryption

REFERENSI

[1] Munir, Rinaldi. 2004. “Sistem Kriptografi Kunci-Publik”, https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/Sistem%20Kriptografi%20Kunci-Publik.pdf. Diakses pada 12 Juni 2024 pukul 14.01.

[2] Munir, Rinaldi. 2024. “Kriptografi Modern”, https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/10-Kripto-modern-2024.pdf. Diakses pada 12 Juni 2024 pukul 14.32.

[3] Munir, Rinaldi. 2024. “Block Cipher (Bagian 1)”, https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/12-Block-Cipher-Bagian1-2024.pdf. Diakses pada 12 Juni 2024 pukul 14.47.

[4] Munir, Rinaldi. 2024. “Block Cipher (Bagian 2)”, https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/13-Prancangan-block-cipher-2024.pdf. Diakses pada 12 Juni 2024 pukul 16.48.

[5] Munir, Rinaldi. 2024. “Fungsi Hash”, https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/24-Fungsi-hash-2024.pdf. Diakses pada 12 Juni 2024 pukul 17.23.

[6] Munir, Rinaldi. 2024. “MAC (Message Authentication Code)”, https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/28-MAC-2024.pdf. Diakses pada 12 Juni 2024 pukul 18.28.

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

Go Dillon Audris
13521062