Secure Password Reset Token Menggunakan HMAC-SHA256

Muhammad Roihan 13522152
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522152@std.stei.itb.ac.id

***Abstract*—Mekanisme *password reset* merupakan fitur krusial dalam sistem autentikasi modern yang memungkinkan pemulihan akses akun, namun sering menjadi celah keamanan jika tidak dirancang dengan benar. Makalah ini membahas perancangan dan implementasi *Secure Password Reset* Token menggunakan algoritma HMAC-SHA256 untuk menjamin integritas dan autentisitas token tanpa perlu menyimpan token tersebut di dalam basis data. Solusi yang ditawarkan memanfaatkan struktur token yang terdiri dari versi, identitas pengguna, waktu kedaluwarsa, dan penanda versi password yang ditandatangani secara kriptografis. Melalui pendekatan ini, sistem mampu memverifikasi validitas token secara mandiri, mendeteksi modifikasi sekecil apa pun, serta mencegah serangan *replay attack* jika token bocor atau password telah diperbarui. Implementasi ini bertujuan meminimalkan risiko keamanan seperti pemalsuan token dengan tetap menjaga efisiensi dan kenyamanan pengguna.**

***Keywords*—HMAC, SHA-256, *Password Reset*, Replay Attack, Autentikasi.**

I. PENDAHULUAN

Password merupakan mekanisme perlindungan akun yang paling umum digunakan saat ini. Namun, sifat manusia yang cenderung mudah melupakan informasi menyebabkan hampir seluruh layanan web modern wajib menyediakan fitur *password reset*. Sayangnya, fitur pemulihan ini sering menjadi titik kritis dalam keamanan sistem. Jika dirancang dengan buruk, mekanisme ini dapat dieksploitasi oleh penyerang untuk melakukan pengambilalihan akun melalui pemalsuan atau manipulasi token.

Tantangan utama dalam merancang fitur ini adalah bagaimana memastikan bahwa tautan pemulihan yang dikirimkan benar-benar berasal dari server yang sah, belum dimodifikasi oleh pihak ketiga, dan hanya berlaku untuk jangka waktu tertentu. Pendekatan konvensional sering kali mengharuskan penyimpanan token acak di dalam basis data untuk divalidasi nanti. Namun, pendekatan ini menambah beban penyimpanan dan kompleksitas manajemen data.

Makalah ini mengusulkan solusi *Secure Password Reset Token* menggunakan algoritma *Keyed-Hashing for Message Authentication* (HMAC) berbasis SHA-256. Pendekatan ini memungkinkan pembuatan token yang memuat seluruh informasi yang dibutuhkan seperti identitas pengguna dan waktu kedaluwarsa, yang kemudian dikunci dengan tanda tangan kriptografis menggunakan kunci rahasia server. Dengan metode ini, integritas token dapat dijamin tanpa perlu menyimpannya di basis data server. Selain itu, desain token ini juga menyertakan mekanisme pencegahan *replay attack* dengan memvalidasi versi password pengguna, sehingga token lama tidak dapat digunakan kembali setelah password berhasil diubah.

II. LANDASAN TEORI

A. Fungsi Hash dan SHA-256

Fungsi hash kriptografi adalah algoritma yang menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan luaran (*message digest*) dengan ukuran tetap. Sifat utama dari fungsi hash yang aman meliputi:

1. *collision resistance*
   sangat sukar menemukan dua input a dan b sedemikian sehingga $H(a) = H(b)$
2. *preimage resistance*
   untuk sembarang output y, sukar menemukan input a sedemikian sehingga $H(a) = y$
3. *second preimage resistance*
   untuk input a dan output $y = H(a)$, sukar menemukan input kedua b sedemikian sehingga $H(b) = y$

SHA-256 (*Secure Hash Algorithm* 256-bit) adalah bagian dari keluarga SHA-2 yang dirancang oleh NSA dan distandardisasi oleh NIST. SHA-256 memproses pesan dalam blok berukuran 512 bit dan menghasilkan *message digest* sepanjang 256 bit. Berbeda dengan pendahulunya seperti MD5 atau SHA-1 yang kini dianggap memiliki kelemahan terhadap serangan kolisi, SHA-256 masih direkomendasikan untuk aplikasi keamanan modern karena belum ada serangan kolisi praktis yang ditemukan terhadapnya.

B. Message Authentication Code (MAC)

Message Authentication Code (MAC) adalah kode berukuran tetap yang dihasilkan dari pesan dan kunci rahasia untuk mengautentikasi pengirim dan memeriksa integritas pesan. Perbedaan mendasar antara MAC dan fungsi hash biasa adalah keberadaan kunci rahasia (K).

Pada fungsi hash biasa ($h = H(M)$), siapa pun dapat menghitung nilai hash dari sebuah pesan. Namun, pada MAC ($MAC = C_K(M)$)), hanya pihak yang memiliki kunci rahasia yang dapat menghitung dan memverifikasi kode tersebut. Hal ini memastikan bahwa jika nilai MAC pada pesan yang diterima cocok dengan hasil perhitungan penerima, maka pesan tersebut dijamin berasal dari pengirim yang sah dan belum diubah oleh pihak ketiga.

C. Hash-based Message Authentication Code

HMAC adalah mekanisme khusus untuk melakukan autentikasi pesan menggunakan fungsi hash kriptografi iteratif (seperti SHA-256) yang dikombinasikan dengan kunci rahasia. HMAC didefinisikan sebagai berikut:

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

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

Gambar 2.1 *Definisi HMAC*
Sumber : Rinaldi Munir – MAC-2025

Dimana:

- H adalah fungsi hash kriptografi (misalnya SHA-256)
- K adalah kunci rahasia.
- m adalah pesan yang akan diautentikasi.
- *opad* (*outer padding)* adalah byte 0x5c yang diulang sebanyak ukuran blok fungsi hash.
- *ipad* (*inner padding*) adalah byte 0x36 yang diulang sebanyak ukuran blok fungsi hash.

  Keunggulan HMAC adalah ia dapat menggunakan fungsi hash yang tersedia tanpa memodifikasi kodenya, serta menangani kunci dengan cara yang sederhana namun tetap mempertahankan kekuatan kriptografis dari fungsi hash yang mendasarinya.

III. Implementasi dan Pembahasan

Untuk menyelesaikan permasalahan keamanan pada mekanisme *password reset,* penulis merancang sebuah solusi di mana *password reset link* harus memenuhi beberapa kriteria utama, yaitu aman, tidak dapat dipalsukan, tidak dapat dimodifikasi, serta tidak dapat digunakan ulang (*replay attack)*. Solusi ini dicapai dengan menerapkan mekanisme token berbasis HMAC-SHA256.

A. Struktur Payload

Struktur *payload* yang diajukan pada penelitian ini ditunjukkan pada Gambar 3.1.

```
v1 | user_id | expiry | pwd_version | signature
```

Gambar 3.1 *Struktur Payload*
Sumber : Dokumen Penulis

Token yang digunakan merupakan sebuah string terenkripsi yang dikodekan menggunakan *Base64* URL-*safe*. Token ini dibangun dari beberapa komponen data penting yang digabungkan menjadi satu kesatuan, kemudian ditandatangani menggunakan algoritma HMAC-SHA256 untuk menjamin keaslian dan integritas data. Secara konseptual, struktur token yang diajukan terdiri dari lima bagian utama, yaitu:

1. Versi Token
   Bagian pertama berfungsi sebagai penanda versi token, misalnya v1. Penambahan versi token bertujuan untuk menjaga kompatibilitas sistem di masa depan apabila terjadi perubahan struktur token atau algoritma kriptografi yang digunakan.
2. Identitas Pengguna
   Bagian ini berisi identitas unik pengguna, seperti user_id. Identitas ini digunakan untuk mengaitkan token secara spesifik dengan satu akun pengguna sehingga token tidak dapat digunakan oleh akun lain.
3. Waktu Kedaluwarsa
   Komponen ini berisi waktu kedaluwarsa token. Dengan adanya batas waktu ini, token hanya valid dalam jangka waktu tertentu, sehingga dapat mencegah penyalahgunaan token yang telah bocor atau disimpan oleh pihak tidak berwenang.
4. Penanda Perubahan Password
   Bagian ini menyimpan informasi waktu terakhir password pengguna diperbarui. Nilai ini berfungsi untuk memastikan bahwa token akan otomatis menjadi tidak valid apabila pengguna telah mengganti password-nya, sehingga mekanisme *replay attack* dapat dicegah.
5. Tanda Tangan Kriptografis (*HMAC Signature*)
   Bagian terakhir merupakan tanda tangan kriptografis yang dihasilkan menggunakan algoritma HMAC-SHA256 dengan *secret key* yang hanya diketahui oleh server. Tanda tangan ini menjamin bahwa seluruh isi token tidak dapat dimodifikasi atau dipalsukan tanpa mengetahui *secret key* tersebut.

   Seluruh bagian di atas digabungkan dalam satu string dengan delimiter tertentu, kemudian dikodekan menggunakan *Base64 URL-safe* sebelum dikirimkan sebagai bagian dari *password reset token*. Dengan struktur ini, sistem dapat melakukan verifikasi token secara mandiri tanpa menyimpan token di database, sekaligus memastikan keamanan, keaslian, dan validitas token.

B. Pembuatan Token Password Reset

Proses pembuatan token *password reset* dilakukan dengan memanfaatkan algoritma HMAC-SHA256 untuk

menghasilkan token yang aman dan tidak dapat dimodifikasi. Proses ini diimplementasikan pada sisi server dan dijalankan ketika pengguna mengajukan permintaan *password reset*.

```
function generateResetToken(user) {
   const expiry = Math.floor(Date.now() / 1000) + 15 *
60; // 15 menit

   const payload = `v1|${user.id}|${expiry}|$
{user.passwordUpdatedAt}`;

   const signature = crypto
      .createHmac("sha256", SECRET)
      .update(payload)
      .digest("base64url");

   return Buffer
      .from(`${payload}|${signature}`)
      .toString("base64url");
}
```

Sumber : Dokumen Penulis

Berdasarkan kode yang diimplementasikan, langkah-langkah pembuatan token adalah sebagai berikut:

Pertama, sistem menentukan waktu kedaluwarsa token (e*xpiration time*) dengan menambahkan durasi tertentu, misalnya 15 menit, terhadap waktu saat ini. Nilai waktu ini disimpan dalam bentuk *Unix timestamp* untuk memudahkan proses validasi di kemudian hari.

Selanjutnya, sistem menyusun *payload* token, yaitu rangkaian data yang berisi versi token, identitas pengguna, waktu kedaluwarsa, serta penanda waktu terakhir perubahan password pengguna. Payload ini digabungkan dalam satu *string* menggunakan delimiter tertentu agar dapat diproses secara konsisten.

Setelah *payload* terbentuk, sistem menghasilkan tanda tangan kriptografis dengan cara menghitung nilai *Hash-based Message Authentication Code* (HMAC) menggunakan algoritma SHA-256. Proses ini memanfaatkan *secret key* yang hanya diketahui oleh server. Hasil HMAC ini berfungsi sebagai bukti keaslian dan integritas payload.

Tahap berikutnya adalah penggabungan *payload* dan tanda tangan menjadi satu string *token* utuh. Token ini kemudian dikodekan menggunakan *Base64* URL-*safe encoding* agar dapat dikirimkan melalui URL tanpa menimbulkan masalah karakter khusus.

Dengan mekanisme ini, token yang dihasilkan memiliki sifat unik, terikat pada pengguna tertentu, memiliki masa berlaku terbatas, serta tidak dapat dimodifikasi tanpa terdeteksi oleh sistem.

C. Proses Verifikasi Token Password Reset

Proses verifikasi token dilakukan ketika pengguna mengakses password reset link yang dikirimkan melalui email. Tujuan utama dari proses ini adalah memastikan bahwa token yang diterima asli, belum kedaluwarsa, belum digunakan ulang, dan tidak dimodifikasi.

```
function verifyResetToken(token, user) {
  const decoded = Buffer.from(token,
"base64url").toString("utf8");
  const [v, userId, expiry, pwdVersion, sig] =
decoded.split("|");

  const payload = `${v}|${userId}|${expiry}|$
{pwdVersion}`;

  const expectedSig = crypto
    .createHmac("sha256", SECRET)
    .update(payload)
    .digest("base64url");

  if (
    !crypto.timingSafeEqual(
      Buffer.from(sig),
      Buffer.from(expectedSig)
    )
  ) {
    throw new Error("Invalid signature");
  }

  if (v !== "v1") throw new Error("Invalid version");

  const now = Math.floor(Date.now() / 1000);
  if (now > Number(expiry)) throw new
Error("Expired");

  if (Number(pwdVersion) !==
user.passwordUpdatedAt)
    throw new Error("Replay detected");

  return true;
}
```

Sumber : Dokumen Penulis

Berdasarkan kode yang digunakan, proses verifikasi token dilakukan melalui beberapa tahap berikut:

Pertama, sistem melakukan dekode token dari format *Base64* URL-*safe* menjadi bentuk *string* aslinya. Setelah itu, sistem memisahkan *payload* dan tanda tangan kriptografis berdasarkan delimiter yang telah ditentukan.

Kedua, sistem melakukan rekonstruksi *payload* dan menghitung ulang tanda tangan HMAC menggunakan algoritma SHA-256 serta *secret key* yang sama seperti pada proses pembuatan token. Nilai tanda tangan hasil perhitungan ulang ini kemudian dibandingkan dengan tanda tangan yang terdapat di dalam token. Apabila hasil perbandingan tidak cocok, maka token dinyatakan tidak valid, yang mengindikasikan bahwa token telah dimodifikasi atau dipalsukan.

Ketiga, sistem melakukan verifikasi versi token dengan membaca nilai versi yang terdapat pada payload, misalnya v1. Sistem kemudian membandingkan nilai versi tersebut dengan daftar versi token yang masih didukung. Apabila versi token tidak dikenali atau sudah tidak didukung, maka token langsung dinyatakan tidak valid tanpa melanjutkan proses verifikasi lainnya.

Mekanisme ini memungkinkan sistem untuk tetap aman dan kompatibel apabila di masa depan terjadi perubahan struktur token atau algoritma kriptografi.

Keempat, sistem memeriksa waktu kedaluwarsa token dengan membandingkan nilai *expiration time* di dalam dengan waktu saat ini. Jika token telah melewati batas waktu yang ditentukan, maka token dianggap tidak valid dan permintaan *password reset* ditolak.

Kelima, sistem memverifikasi penanda perubahan password dengan membandingkan nilai *passwordUpdatedAt* di dalam token dengan data terbaru yang tersimpan di basis data. Jika nilai tersebut tidak sesuai, maka token dinyatakan tidak valid karena pengguna telah mengganti password setelah token dibuat. Mekanisme ini berfungsi untuk mencegah *replay attack.*

Apabila seluruh proses verifikasi berhasil, maka token dinyatakan valid dan sistem mengizinkan pengguna untuk melanjutkan ke tahap penggantian password. Dengan mekanisme verifikasi ini, sistem secara efektif mencegah pemalsuan token, manipulasi data, serta serangan *replay attack*.

D. Uji Kasus dan Analisis Keamanan

Untuk memastikan bahwa mekanisme token *password reset* yang diusulkan benar-benar memenuhi aspek keamanan yang diharapkan, dilakukan serangkaian uji kasus. Uji kasus ini bertujuan untuk memvalidasi bahwa token memiliki ketahanan terhadap pemalsuan, modifikasi data, penggunaan ulang, serta penyalahgunaan akibat masa berlaku yang telah berakhir.

Uji kasus pertama bertujuan untuk membuktikan bahwa token tidak dapat dipalsukan oleh pihak tidak berwenang. Pada pengujian ini, penyerang disimulasikan dengan cara mengubah satu karakter terakhir dari token yang sah tanpa mengetahui *secret key* yang digunakan dalam proses HMAC.

```
function testTokenCannotBeForged() {
  const user = { id: "user123", passwordUpdatedAt:
1696089600 };
  const token = generateResetToken(user);

  // Penyerang mencoba memalsukan token
  const forgedToken = token.replace(/.$/, "A"); //
Mengubah karakter terakhir

  try {
    verifyResetToken(forgedToken, user);
    console.error("Test gagal: Token palsu berhasil
diverifikasi.");
  } catch (error) {
    console.log("Test berhasil: Token palsu tidak dapat
diverifikasi.");
  }
}
```

Sumber : Dokumen Penulis

Berdasarkan potongan kode testTokenCannotBeForged(), token palsu dibuat dengan memodifikasi karakter terakhir token yang telah dihasilkan. Ketika token palsu tersebut diverifikasi oleh sistem, proses verifikasi menghasilkan kegagalan karena nilai tanda tangan kriptografis tidak lagi sesuai dengan payload yang ada.

```
Test berhasil: Token palsu tidak dapat diverifikasi.
```

***Gambar 3.2** Hasil Uji Kasus 1*
Sumber : Dokumen Penulis

Hasil pengujian menunjukkan bahwa token palsu tidak dapat diverifikasi, yang membuktikan bahwa penggunaan HMAC-SHA256 secara efektif mencegah pemalsuan token tanpa kepemilikan secret key.

Uji kasus kedua bertujuan untuk memastikan bahwa sistem mampu mendeteksi modifikasi terhadap isi *payload* token. Dalam pengujian ini, penyerang mencoba mengubah nilai waktu kedaluwarsa token dengan tujuan memperpanjang masa berlaku token secara tidak sah.

```
function testPayloadModificationDetected() {
  const user = { id: "user123", passwordUpdatedAt:
1696089600 };
  const token = generateResetToken(user);

  // Penyerang mencoba memodifikasi payload
(misalnya, memperpanjang waktu kedaluwarsa)
  const parts = Buffer.from(token,
"base64url").toString("utf8").split("|");
  parts[2] = (Math.floor(Date.now() / 1000) + 60 *
60).toString(); // Memperpanjang waktu kedaluwarsa
  const modifiedToken = Buffer.from(parts.join("|"),
"utf8").toString("base64url");

  try {
    verifyResetToken(modifiedToken, user);
    console.error("Test gagal: Token yang dimodifikasi
berhasil diverifikasi.");
  } catch (error) {
    console.log("Test berhasil: Modifikasi payload
terdeteksi.");
  }
}
```

Sumber : Dokumen Penulis

Pada fungsi testPayloadModificationDetected(), token yang sah didekode, kemudian nilai *expiration time* pada *payload* dimodifikasi. Token yang telah dimodifikasi tersebut kemudian dikodekan kembali dan dikirimkan ke proses verifikasi.

```
Test berhasil: Modifikasi payload terdeteksi.
```

***Gambar 3.3** Hasil Uji Kasus 2*
Sumber : Dokumen Penulis

Hasil pengujian menunjukkan bahwa token yang telah dimodifikasi gagal diverifikasi karena tanda tangan HMAC yang terdapat di dalam token tidak lagi sesuai dengan *payload* yang telah diubah. Hal ini membuktikan bahwa mekanisme tanda tangan kriptografis mampu menjamin integritas data, sehingga setiap perubahan terhadap payload dapat terdeteksi oleh sistem.

Uji kasus ketiga dilakukan untuk memastikan bahwa token tidak dapat digunakan setelah melewati waktu kedaluwarsa yang telah ditentukan. Pada pengujian ini, sistem menunggu hingga token melewati masa berlaku sebelum dilakukan proses verifikasi ulang.

```
function testTokenExpiry() {
  const user = { id: "user123", passwordUpdatedAt:
1696089600 };
  const token = generateResetToken(user);
  setTimeout(() => {
    try {
      verifyResetToken(token, user);
      console.error("Test gagal: Token kedaluwarsa
berhasil diverifikasi.");
    } catch (error) {
      console.log("Test berhasil: Token kedaluwarsa tidak
dapat diverifikasi.");
    }
  }, 20000);
}
```

Sumber : Dokumen Penulis

Berdasarkan fungsi testTokenExpiry(), token diverifikasi setelah melewati interval waktu tertentu. Sistem menolak token tersebut karena nilai waktu saat ini telah melampaui nilai *expiration time* yang tercantum di dalam payload token.

```
Test berhasil: Token kedaluwarsa tidak dapat diverifikasi.
```

Gambar 3.4 *Hasil Uji Kasus 3*
Sumber : Dokumen Penulis

Hasil pengujian ini membuktikan bahwa mekanisme pembatasan waktu berlaku token berjalan dengan baik dan efektif dalam mengurangi risiko penyalahgunaan token yang telah kadaluwarsa.

Uji kasus keempat bertujuan untuk memverifikasi bahwa token tidak dapat digunakan kembali setelah pengguna mengganti password-nya. Pada pengujian ini, token dibuat menggunakan nilai passwordUpdatedAt awal, kemudian nilai tersebut diubah untuk mensimulasikan proses penggantian password oleh pengguna.

```
function testPasswordUpdatedAt() {
  const user = { id: "user123", passwordUpdatedAt:
1696089600 };
  const token = generateResetToken(user);

  // Simulasi pengguna mengganti password
  user.passwordUpdatedAt = Math.floor(Date.now() /
1000);

  try {
    verifyResetToken(token, user);
    console.error("Test gagal: Token lama berhasil
diverifikasi setelah password diganti.");
  } catch (error) {
    console.log("Test berhasil: Token lama tidak valid
setelah password diganti.");
  }
}
```

Sumber : Dokumen Penulis

Pada fungsi testPasswordUpdatedAt(), token lama diverifikasi kembali setelah nilai passwordUpdatedAt pengguna diperbarui. Proses verifikasi menghasilkan kegagalan karena nilai passwordUpdatedAt pada token tidak lagi sesuai dengan kondisi akun pengguna saat ini.

```
Test berhasil: Token lama tidak valid setelah password diganti.
```

Gambar 3.5 *Hasil Uji Kasus 4*
Sumber : Dokumen Penulis

Hasil pengujian ini menunjukkan bahwa token lama secara otomatis menjadi tidak valid setelah terjadi perubahan password, sehingga mekanisme ini efektif dalam mencegah *replay attack* tanpa memerlukan penyimpanan status token di basis data.

Berdasarkan analisis di atas, mekanisme token password reset yang diusulkan mampu memberikan tingkat keamanan yang tinggi serta memenuhi kebutuhan sistem modern dalam mencegah pemalsuan, manipulasi data, dan serangan *replay attack*.

IV. KESIMPULAN

Berdasarkan implementasi dan analisis keamanan yang telah dilakukan, dapat disimpulkan bahwa penggunaan algoritma HMAC-SHA256 dalam mekanisme *password reset* mampu memberikan jaminan keamanan yang tinggi terhadap integritas dan autentisitas token.

Implementasi struktur token yang menggabungkan versi token, identitas pengguna, waktu kedaluwarsa, dan penanda perubahan password terbukti efektif dalam menangani berbagai ancaman keamanan modern. Penggunaan tanda tangan kriptografis (HMAC) dengan kunci rahasia memastikan bahwa token tidak dapat dipalsukan atau dimodifikasi oleh pihak yang tidak berwenang, karena perubahan sekecil apa pun pada payload akan merusak validitas tanda tangan.

Lebih lanjut, mekanisme ini berhasil mengatasi masalah serangan *replay attack* melalui integrasi data passwordUpdatedAt ke dalam payload token. Hal ini memastikan token otomatis menjadi tidak valid segera setelah pengguna melakukan penggantian password, meskipun waktu kedaluwarsa token tersebut belum habis.

Secara keseluruhan, solusi yang diajukan menawarkan keseimbangan antara keamanan dan efisiensi. Sistem dapat melakukan verifikasi secara mandiri tanpa membebani penyimpanan basis data, sekaligus tetap fleksibel terhadap pembaruan algoritma di masa depan melalui mekanisme *versioning*. Implementasi ini sangat direkomendasikan untuk sistem autentikasi modern yang memprioritaskan keamanan data pengguna tanpa mengorbankan performa sistem.

V. UCAPAN TERIMA KASIH

Dengan penuh rasa syukur, penulis ingin mengucapkan

puji dan syukur kepada Tuhan Yang Maha Esa atas segala rahmat, karunia, serta taufik dan hidayah-Nya, yang telah memandu dan memberikan keberkahan sehingga penulis berhasil menyelesaikan makalah berjudul "*Secure Password Reset* Token Menggunakan HMAC-SHA256". Makalah ini disusun sebagai bagian dari tugas mata kuliah Kriptografi IF4020.

Penulis juga mengucapkan terima kasih yang setinggi tingginya kepada Dr. Ir. Rinaldi, M.T., sebagai dosen pengajar dalam mata kuliah Kriptografi IF4020. Terima kasih atas dedikasi, bimbingan, dan pengajaran yang telah diberikan selama satu semester ini, memberikan kontribusi besar dalam pembentukan pemahaman kami sebagai mahasiswa. Penulis juga beterimakasih kepada seluruh pihak yang telah berkontribusi baik secara langsung maupun tidak langsung terhadap kelancaran penulisan makalah ini yang namanya tidak bisa disebutkan satu persatu. Penulis juga ingin menyampaikan permohonan maaf apabila terdapat kesalahan dalam penulisan makalah ini. Semoga makalah ini dapat bermanfaat dan memberikan sumbangan positif dalam pemahaman mata kuliah Kriptografi IF4020..

REFERENCES

[1] Rinaldi Munir, "Beberapa Fungsi Hash", https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2025-2026/27-Beberapa-fungsi-hash-2025.pdf diakses pada tanggal 25 Desember 2025.

[2] Rinaldi Munir, "MAC", https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2025-2026/28-MAC-2025.pdf diakses pada tanggal 25 Desember 2025

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, 25 Desember 2025

Muhammad Roihan 13522152