Implementasi Skema Pembangkitan dan Penukaran Voucer Daring Menggunakan Algoritma SHA-256

Reyvan Rizky Irsandy - 13518136
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 13518136@std.stei.itb.ac.id

***Abstract*—Seiring perkembangan zaman, voucher elektronik semakin banyak digunakan pada platform penyedia produk dan layanan daring. Voucer elektronik (*e-voucher*) merupakan suatu kode unik yang dapat ditukarkan menjadi kredit atau uang virtual yang dapat digunakan untuk membeli produk atau layanan pada platform tertentu. Namun, voucer elektronik sangat rentan terhadap pembobolan. Platform yang menggunakan sistem *e-voucher* tanpa konsiderasi keamanan yang matang memungkinkan adanya oknum yang berhasil menukarkan kode unik tanpa membeli voucer dengan cara melakukan *bruteforce*. Oleh karena itu, makalah ini dibuat untuk meningkatkan keamanan platform menggunakan algoritma SHA-256 pada sebuah skema pembangkitan dan penukaran voucer elektronik. Sehingga, hanya pengguna yang membeli voucerlah yang dapat menukarkan voucer tersebut.**

Keywords—pembangkitan voucer daring; penukaran voucer daring; algoritma SHA-256; fungsi hash; kriptografi

I. Pendahuluan

Dewasa ini, banyak platform penyedia layanan dan produk yang menyediakan *e-voucher* atau voucer elektronik dalam pembelian produk atau jasanya. Voucer elektronik ini merupakan kupon yang dapat ditukarkan oleh pembeli voucher untuk membeli produk atau jasa yang mereka inginkan.

Namun, di balik maraknya penggunaan voucer elektronik ini, terdapat beberapa permasalahan besar yang menyangkut keamanan data serta menerima voucer yang tidak seharusnya dimiliki olehnya. Penggunaan voucer elektronik yang cukup memasukkan kode unik voucer sangat rentan terhadap oknum yang melakukan *bruteforce* untuk mendapatkan *voucher* tanpa membeli. Tentu saja, terdapat beberapa platform yang telah menerapkan *rate limit* atau batas jumlah akses ke server platform. Namun, tidak semua platform menerapkan hal tersebut, yang menyebabkan kerentanan pembobolan sistem yang semakin besar.

Oleh karena itu, diperlukan sistem keamanan platform yang lebih kuat dalam pembelian dan penukaran voucer.

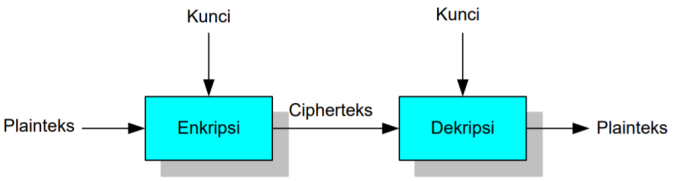
II. Landasan Teori

A. Kriptografi

Kriptografi berasal dari kata cryptós yang dalam bahasa Yunani berarti rahasia, dan gráphein yang dalam bahasa Yunani berarti menulis. Sehingga secara bahasa, kriptografi berarti tulisan rahasia. Secara umum, kriptografi berarti ilmu yang mempelajari teknik matematika yang berkaitan dengan keamanan informasi seperti kerahasiaan, integritas data, dan otentikasi. Pada bidang keamanan informasi, kriptografi merupakan kakas yang sangat penting.

Pada bidang keamanan informasi, terdapat empat hal utama yang harus diperhatikan. Keempat hal tersebut adalah terjaga kerahasiaannya (*data confidentiality*), terjaga keasliannya (*data integrity*), memastikan pengirim pesan adalah pengirim yang asli dan bukan penyamar (*authentication*), dan pengirim pesan tidak dapat menyangkal bahwa pengirim tersebut telah mengirim pesan (*non-repudiation*).

Secara sederhana, proses kriptografi diawali dengan adanya pesan asli yang akan dikirimkan. Pesan yang disebut plainteks ini dapat berbentuk teks, gambar, audio, video, dan lain-lain.. Kemudian, plainteks ini dienkripsi (*encryption*) dengan sebuah kunci sehingga menjadi pesan tersebut tidak dapat dibaca oleh pihak yang bukan penerima. Pesan yang telah disandikan atau dienkripsi ini disebut cipherteks. Cipherteks akan dikirim ke penerima pesan, dan dekripsi (*decryption*) menggunakan suatu kunci lagi, sehingga pesan asli dapat terlihat (kembali ke plainteks) dan dapat dipahami oleh penerima. Visualisasi dari proses kriptografi dapat dilihat pada Gambar 1.



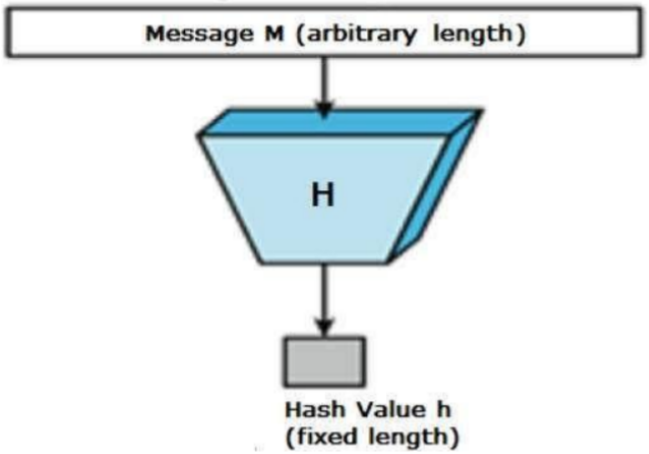
GAMBAR 1. Proses Kriptografi

B. *Fungsi Hash*

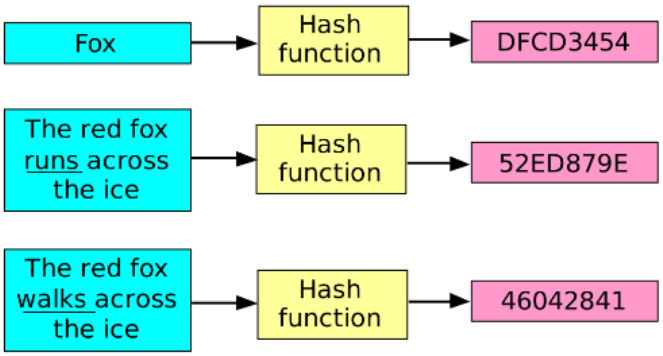
Fungsi *hash* adalah fungsi yang mengkompresi pesan (M) yang ukurannya bermacam-macam menjadi sebuah *string* (h) yang ukurannya pasti (*fixed*). Fungsi ini bersifat *irreversible*. Sehingga, sebuah pesan yang telah dilakukan *hashing* tidak dapat dikembalikan ke bentuk pesan semula. Fungsi ini memiliki persamaan yang dapat dilihat pada persamaan (1).

$$h = H(M) \quad (1)$$

Visualisasi dari fungsi *hash* dapat dilihat pada GAMBAR 2 dan GAMBAR 3.



GAMBAR 2. FUNGSI HASH 1



GAMBAR 3. FUNGSI HASH 2

Terdapat tiga keuntungan dari penggunaan fungsi *hash* yang bersifat satu arah. Keuntungan-keuntungan tersebut adalah sebagai berikut:

1) Menjaga integritas pesan. Integritas pesan dapat terjaga karena perubahan sedikit saja pada pesan asli dapat mengubah hasil *hash* secara signifikan. Hal ini sangat membantu memastikan apakah hasil *hash* telah dimodifikasi dengan membandingkan hasil *hash* yang baru dan yang lama.
2) Menghemat waktu pengiriman. Hal ini dapat mempermudah memastikan salinan pesan sesuai dengan yang asli. Waktu pengiriman dapat lebih singkat dengan cukup mengirim hasil *message digest*, tanpa perlu mengirimkan salinan pesan secara keseluruhan.
3) Menormalkan panjang data yang beraneka ragam. Panjang data dan ukuran *file* pesan tersebut dapat bermacam-macam. Fungsi *hash* dapat menyeragamkan panjang dari data.

Namun, terdapat kekurangan dari fungsi *hash*, yaitu kolisi (*collision*). Kolisi adalah suatu kondisi ketika dua atau lebih *string* memiliki hasil hash yang sama. Hal ini menyebabkan fungsi *hash* tidak aman. Saat ini, sudah terdapat beberapa algoritma *hash* yang ditemukan kolisinya.

Terdapat beberapa contoh algoritma fungsi *one-way hash* yang telah ada. Contoh-contoh tersebut dapat dilihat pada TABEL 1.

TABEL I. CONTOH FUNGSI HASH SATU ARAH

Algoritma Fungsi *Hash*	Ukuran Message Digest (h) dalam Bit	Ditemukan Kolisi
MD2	128	Ya
MD4	128	Hampir
MD5	128	Ya
SHA-0	160	Ya
SHA-1	160	Ditemukan kekurangan
SHA-256/SHA-224 (SHA-2)	256/224	Belum
SHA-512/SHA-384 (SHA-2)	512/384	Belum
SHA-3 (Keccak)	Sembarang	Belum

C. *Algoritma SHA-2*

SHA merupakan singkatan *Secure Hash Algorithm*. Algoritma ini merupakan salah satu dari fungsi *one-way hash* dan merupakan standar dari *hash* satu arah. Algoritma ini dibuat berdasarkan algoritma fungsi *hash* MD4 yang telah dibuat sebelumnya. Algoritma SHA menerima *input* dengan ukuran maksimum 2^{64}. Algoritma ini menghasilkan *message digest* dengan panjang 160 bit.

Terdapat beberapa varian dari SHA. Varian-varian tersebut adalah SHA-0, SHA-1, SHA-2 yang terdiri dari SHA-224, SHA-256, SHA-384, dan SHA-512. Pada makalah ini, akan digunakan algoritma SHA-256 yang merupakan bagian dari SHA-2.

Algoritma SHA-256 termasuk algoritma SHA-2. SHA-2 dibuat didasarkan pada algoritma SHA sebelumnya, SHA-1. Algoritma ini dibuat karena terdapat kelemahan dari SHA-1. Pada awalnya, hanya terdapat dua fungsi *hash* pada SHA-2, yaitu SHA-256 dan SHA-512. Namun, dibuat lagi versi lebih singkat dari kedua fungsi tersebut, yaitu SHA-224, SHA-384, SHA-512/224, dan SHA-512/256.

Terdapat beberapa perbedaan antara SHA-2 dengan SHA-1. Perbedaan utamanya adalah SHA-1 menghasilkan *message digest* berukuran 160 bit, sedangkan SHA-2

menghasilkan *message digest* berukuran 224 atau 256 bit. Selain itu, selain dapat memiliki ukuran blok 512 bit seperti SHA-1, SHA-2 juga dapat memiliki ukuran blok yang berukuran 1024 bit. Menggunakan *bruteforce* pada SHA-2 juga cenderung lebih sulit dibandingkan pada SHA-1, sehingga algoritma SHA-2 cenderung lebih aman.

D. Algoritma SHA-256

SHA-256 pertama kali dirilis pada tahun 2001. Algoritma ini memiliki ukuran *output* 256 bit. Algoritma ini juga memiliki ukuran blok 512 bit. Terdapat beberapa langkah dalam algoritma SHA-256. Langkah-langkah tersebut akan dijelaskan pada bagian ini.

1) Pre-Processing

Hal ini dilakukan dengan mengkoversi pesan ke bilangan biner. Kemudian, menambakan angka 1 di bit terakhir. Kemudian, menambahkan 0 sampai data merupakan kelipatan 512, kurang 64 bit. Terakhir, menambahkan 64 bit berupa bilangan bulat big-endian yang mewakili panjang input asli dalam bilangan biner.

2) Inisialisasi nilai *hash* (h)

Dibuat 8 nilai *hash* awal. Nilai ini merupakan 32 bit pertama dari hasil akar kuadrat 8 bilangan prima pertama (2-19).

3) Inisialisasi Konstanta Bulat (k)

Dibuat 64 nilai konstanta yang merupakan 32 bit pertama dari hasil akar kubik dari 64 bilangan prima pertama (20-311).

4) Potong *Loop*

Dilakukan pemotongan input setiap 512 bit. Masing-masing pemotongan tersebut akan mengubah nilai *hash* awal yang telah didefinisikan pada langkah nomor 2.

5) Buat Jadwal Pesan (w)

Hal ini dilakukan dengan menyalin data masukan dari langkah pertama menjadi sebuah *array*. KEmudian dibuat 48 kata lagi yang diinisialisasi dengan bilangan 0. Angka-angka tersebut kemudian dimodifikasi dengan algoritma khusus. Akhirnya, akan didapatkan 64 kata pada *message schedule* atau jadwal pesan.

6) Kompresi

Dibuat variabel inisialisasi a sampai h. Kompresi dilakukan menggunakan algoritma khusus yang mengubah nilai a sampai h.

7) Ubah Nilai Akhir

Dilakukan perubahan nilai *hash* pada hasil *loop* kompresi dengan menambahkan variabel yang sesuai. Setiap penambahan bernilai 2^{32}.

8) Gabungkan *Hash* Terakhir

Terakhir, dilakukan penggabungan dari seluruh nilai *hash* yang telah dilakukan pada tahap-tahap sebelumnya.

III. Solusi dan Implementasi

A. Deskripsi Solusi

Berdasarkan permasalahan keamanan dalam pembelian voucer elektronik yang telah dijelaskan secara rinci pada bagian Pendahuluan, dirancang solusi yang dapat menyelesaikan permasalahan tersebut.

Solusi yang diajukan adalah skema pembangkitan kode unik voucer yang ditujukan secara spesifik untuk pembeli. Sehingga, penukaran voucer hanya dapat dilakukan oleh pembeli voucer tersebut. Solusi ini tentu saja dapat meningkatkan keamanan sistem karena menghindari *bruteforce* oleh orang yang tidak berhak menerima voucer.

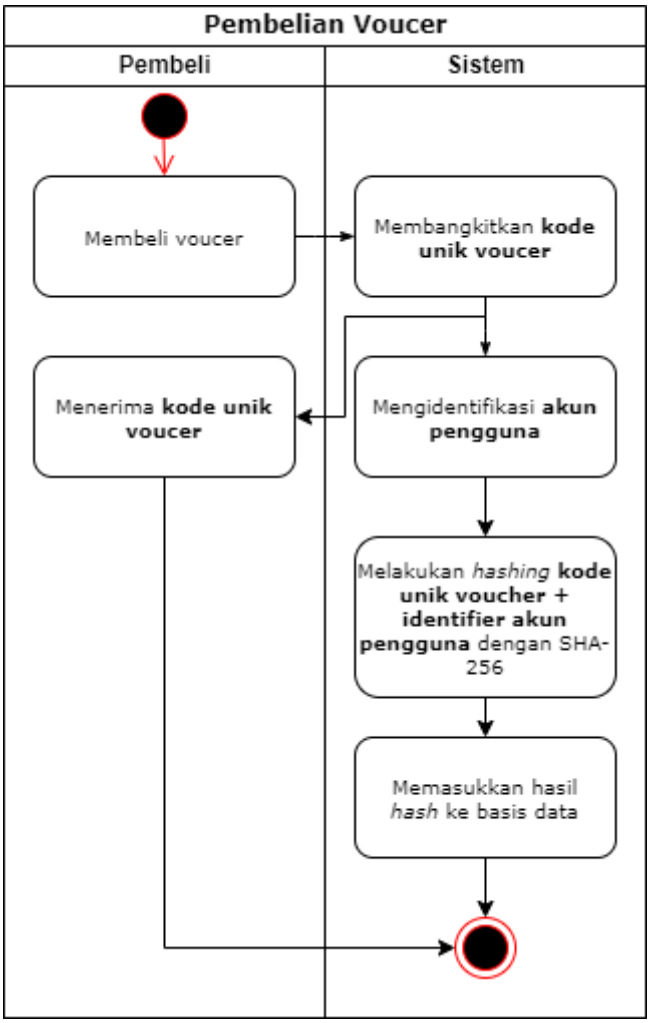
Pada pembelian voucer, tentu kasusnya tidak selalu pembeli membeli voucer untuk diri sendiri. Terdapat kasus lain ketika pembeli ingin memberikan voucer tersebut untuk dapat digunakan oleh orang lain. Namun, kasus pembelian voucer untuk orang lain tidak tercakup dalam makalah ini. Sehingga, cakupan solusi yang diajukan terbatas pada pembelian dan penukaran voucer yang dilakukan oleh akun pengguna yang sama.

Sistem ini menggunakan fungsi *hash* SHA-256 (ukuran 256 bit) untuk mengamankan voucer yang telah dibeli. Fungsi *hash* ini dipilih karena sampai saat ini, belum ditemukan kolisi pada fungsi SHA-256. Selain itu, ukuran 256 lebih aman di diproses lebih cepat dibandingkan dengan SHA

B. Rancangan Solusi

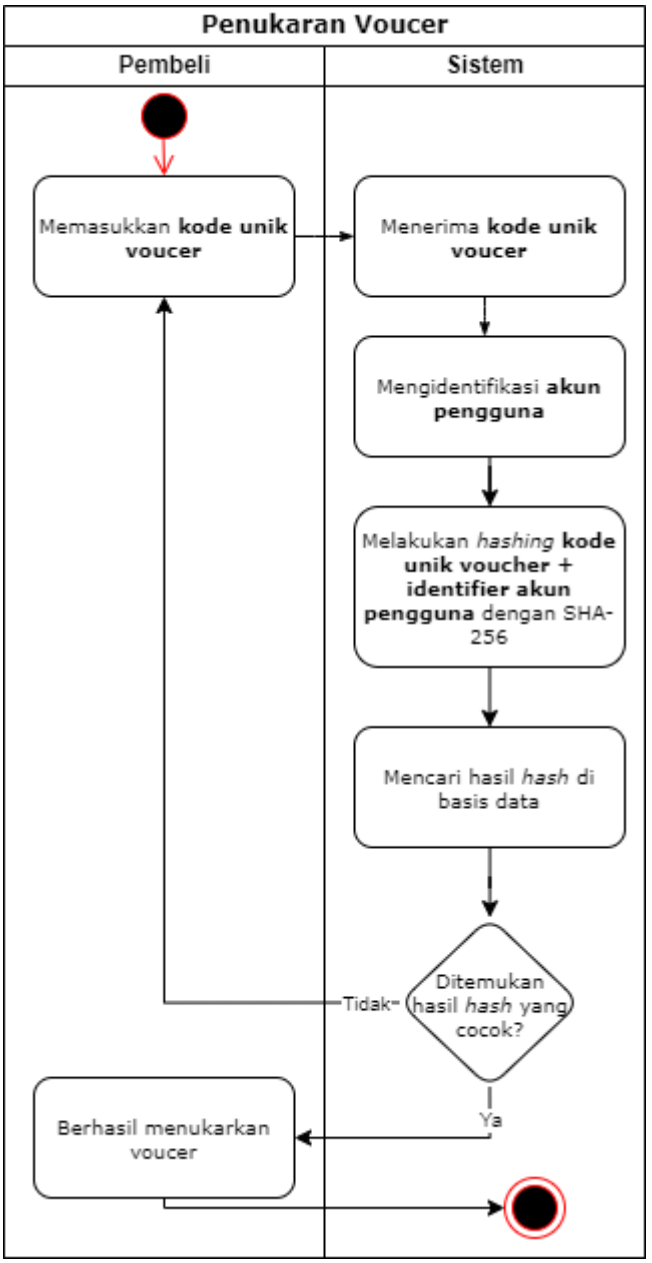
Solusi yang diajukan terdiri dari dua skema. Skema pertama adalah skema pembelian voucer, yaitu ketika pembeli membeli voucer dan mendapatkan kode unik pembelian. Skema yang kedua adalah skema penukaran voucher, yaitu ketika pembeli telah memiliki voucer dan akan ditukarkan dengan produk atau layanan.

Pada skema pembelian voucer, terdapat beberapa tahap yang terjadi. Pertama, pembeli dapat membeli voucer sesuai keinginannya dengan melakukan input kepada sistem. Kemudian, sistem akan membangkitkan sebuah kode voucer unik yang kemudian diberikan kepada pembeli. Kode tersebut dapat digunakan oleh pembeli untuk ditukarkan pada skema “Penukaran Voucer”. Kemudian, sistem akan mengidentifikasi akun dari pengguna yang melakukan pembelian akun tersebut. Kemudian, kode unik yang telah dibangkitkan pada tahap sebelumnya akan dilakukan *hashing* secara bersamaan dengan *identifier* akun pembeli.Terakhir, hasil *hash* tersebut akan dimasukkan ke dalam basis data. Visualisasi skema pembelian voucer dapat dilihat pada Gambar 4.



GAMBAR 4. SKEMA PEMBELIAN VOUCER

Pada skema pembelian voucer, skema dimulai ketika terdapat pembeli yang ingin melakukan penukaran voucher. Pembeli tersebut dapat memasukkan kode unik voucer yang telah didapatkan dari skema sebelumnya. Kode unik voucer akan dibangkitkan dengan *pseudorandom*. Kemudian, sistem melakukan identifikasi akun dari pengguna yang melakukan penukaran voucer. Berdasarkan kode unik yang diterima sistem serta hasil *identifier* dari akun yang melakukan penukaran, dilakukan *hashing* dari kedua hal tersebut menggunakan SHA-256. Kemudian, hasil dari *hash* tersebut akan dicocokkan dengan basis data. Jika ditemukan hasil *hash* yang sesuai, maka penukar voucer akan berhasil menukarkan voucernya. Jika tidak ada yang sesuai, maka penukar voucer diminta memasukkan ulang kode unik yang sesuai. Visualisasi skema penukaran voucer dapat dilihat pada GAMBAR 5.



GAMBAR 5. SKEMA PENUKARAN VOUCER

C. *Implementasi Solusi*

Pada implementasi skema pembangkitan dan penukaran voucer daring menggunakan algoritma SHA-256, digunakan bahasa pemrograman Python dan library Python Flask.

Bahasa pemrograman Python dipilih karena memiliki fleksibilitas tinggi untuk mengimplementasi suatu aplikasi. sedangkan *library* Flask digunakan karena aplikasi yang dibuat bersifat *web service*.

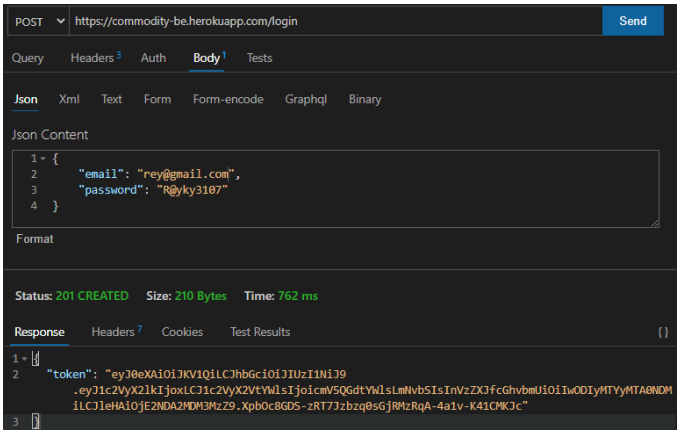
Pada proses implementasinya, implementasi dilakukan dan dijalankan secara lokal. Spesifikasi dari mesin yang digunakan untuk melakukan implementasi adalah sebagai berikut:

a) Processor: Intel Core i7 9750HF

b) RAM: 16GB
c) OS: Windows 10

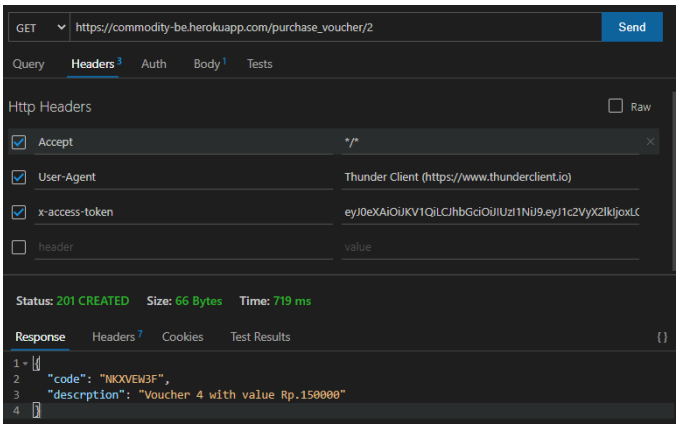
Terdapat 5 *endpoint* yang diimplementasikan menggunakan Python dan Flask, yaitu sebagai berikut

1. POST /login
   Proses transaksi memerlukan informasi pengguna. Sehingga, pengguna wajib login ke dalam sistem untuk melakukan transaksi voucer. Percobaan untuk *endpoint* ini dapat dilihat pada GAMBAR 6. Setiap pengguna akan mendapatkan *token* yang akan menjadi *identifier* pengguna tersebut dalam menggunakan aplikasi.



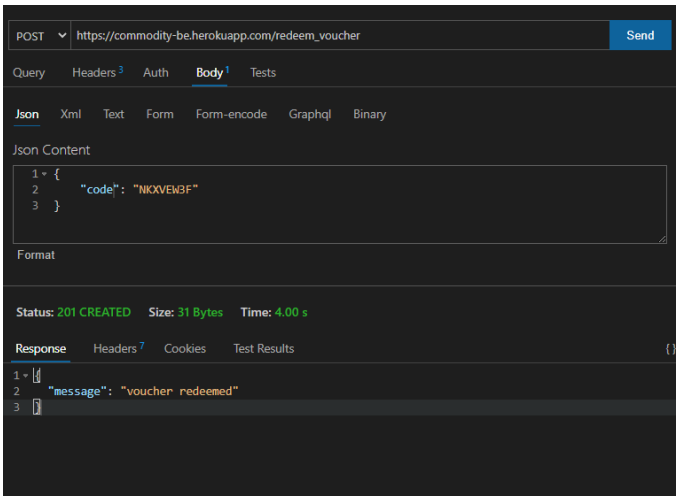
GAMBAR 6. HASIL IMPLEMENTASI LOGIN

2. GET /purchase_voucher/:voucher_id
   Endpoint ini digunakan untuk membeli voucer atau membangkitkan kode voucer. Setiap pengguna yang mengakses *endpoint* ini wajib melakukan *login* kedalam sistem terlebih dahulu karena informasi pengguna dibutuhkan untuk membeli voucer. Informasi pengguna yang terdapat pada token akan digunakan untuk membangkitkan *hash value* yang selanjutnya digunakan untuk penukaran voucer. *Endpoint* ini memiliki parameter berupa voucher id yang berisi informasi voucer yang dibeli oleh pengguna. Pada GAMBAR 7, dapat dilihat bahwa setiap pengguna yang memanggil *endpoint* ini akan mendapatkan informasi kode voucer dan deskripsi voucer yang dibeli. Detail implementasi mengenai *endpoint* ini telah dicantumkan pada rancangan solusi.



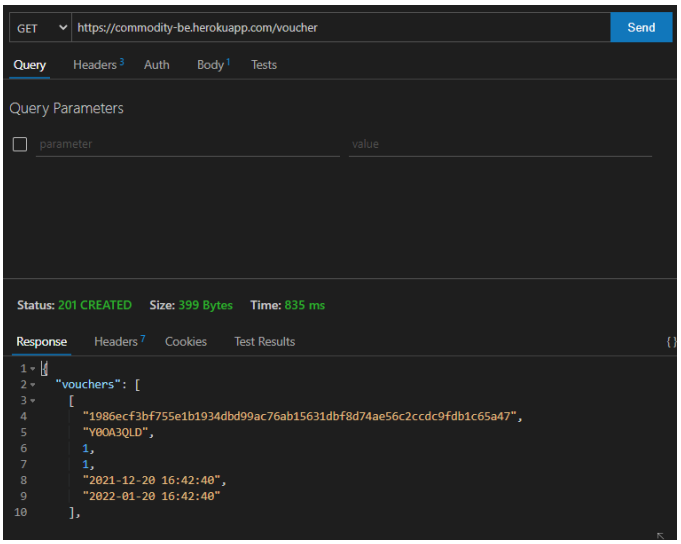
GAMBAR 7. HASIL IMPLEMENTASIPURCHASE VOUCHER

3. POST /redeem_voucher
   Endpoint ini digunakan untuk menukarkan voucer yang dibeli oleh pengguna, seperti *endpoint* pembelian voucer, pengguna wajib *login* ke dalam sistem. Informasi pengguna yang terdapat pada *token* yang terletak pada *request header* digunakan untuk memverifikasi apakah pengguna membeli voucher dengan kode yang dimasukkan. Akibatnya, apabila aplikasi tidak memiliki *rate-limit*, sistem ini akan menjadi proteksi tambahan.



GAMBAR 8. HASIL IMPLEMENTASI REDEEM VOUCHER

4. GET /voucher
   Endpoint ini digunakan untuk melihat daftar voucer yang terdaftar pada sistem. *Endpoint* ini digunakan hanya untuk pengujian.



GAMBAR 9. DAFTAR VOUCHER YANG DIDAPATKAN

5. PATCH /
   Endpoint ini digunakan untuk mendapatkan informasi detail pengguna yang *login* kedalam sistem. *Endpoint* ini digunakan hanya untuk pengujian yang bertujuan untuk melihat jumlah akhir *wallet* pengguna.

Web server aplikasi dapat diakses pada pranala berikut https://commodity-be.herokuapp.com/. Selain implementasi web service, berikut adalah skema database yang digunakan oleh aplikasi.

```
create table user (
  id integer primary key autoincrement,
  email text not null,
  firstName text not null,
  lastName text not null,
  phoneNumber text not null,
  wallet integer not null
);

create table user_password (
  id integer primary key,
  password text not null,
  foreign key (id)
    references user (id)
);

create table voucher (
  id char(64) not null primary key,
  code char(6) not null,
  isUsed boolean not null,
  voucher_id integer not null,
  createdAt timestamp not null,
  expiredAt timestamp not null,
  foreign key (voucher_id)
    references voucher_properties (id)
);

create table voucher_properties (
  id integer not null primary key autoincrement,
  value integer not null,
  description text
);
```

IV. PENGUJIAN HASIL IMPLEMENTASI

Berdasarkan hasil implementasi dari rancangan solusi yang telah dijelaskan pada bagian sebelumnya, dilakukan pengujian dari hasil implementasi tersebut. Pengujian dilakukan secara lokal dengan spesifikasi mesin sebagai berikut:

a) Processor: Intel Core i7 9750HF
b) RAM: 16GB
c) OS: Windows 10

Pengujian dilakukan dengan mensimulasikan beberapa kasus. Kasus-kasus tersebut akan dirinci pada bagian ini.

A. Pengujian Skema Pembelian Voucer

1) Pembangkitan voucer yang berbeda untuk setiap pengguna (hasil fungsi *hash* yang unik)
   - Masukan: Memanggil API dengan pranala sebagai berikut dengan metode GET https://commodity-be.herokuapp.com/purchase_voucher/2
   - Keluaran yang diharapkan: Kode unik diberikan kembali kepada pengguna dan voucher baru ditambahkan pada database dengan id merupakan hash value dari informasi pengguna dan kode unik voucher.
   - Keluaran hasil pengujian:

```
JSON response sebagai berikut
{
 "code": "LO2P20S4",
 "descrption": "Voucher 4 with value Rp.150000"
}

Pada endpoint /voucher terdapat voucher baru
dengan property seperti berikut
[
"32c4c2f5844ccde0822798eb8b60ee20d456acbdc
2f980cbe724e549cadd93cd",
    "LO2P20S4",
    0,
    2,
    "2021-12-20 12:21:07",
    "2022-01-20 12:21:07"
]
```

Setiap pengguna yang membeli voucer mendapatkan kode unik voucer, dan pada database ditambahkan

object voucer baru dengan id yang merupakan *hash value*.

B. Pengujian Skema Penukaran Voucer

1) Kasus menukarkan voucer yang telah dibeli
   - Masukan: Memanggil API penukaran voucher dengan *payload* kode unik yang didapatkan dari bagian A.1
   - Keluaran yang diharapkan: Voucher ditukarkan dan pengguna mendapat tambahan *wallet*
   - Keluaran hasil pengujian:

```
Informasi pengguna sebelum penukaran
[
  1,
  "rey@gmail.com",
  "Reyvan",
  "Rizky",
  "082216210443",
  460000
]

Response pemanggilan API penukaran
{
  "message": "voucher redeemed"
}

Informasi pengguna setelah penukaran
[
  1,
  "rey@gmail.com",
  "Reyvan",
  "Rizky",
  "082216210443",
  610000
]

Response pemanggilan API dengan kode yang
sudah ditukarkan
{
  "message": "Voucher not Found"
}
```

Wallet pengguna yang merupakan bagian terakhir dari informasi pengguna bertambah sebesar 150000. Artinya, voucer berhasil ditambahkan. Selain itu, pengguna tidak dapat menukarkan voucer lebih dari satu kali.

2) Kasus menukarkan voucer yang dibeli oleh pengguna lain
   - Masukan: Pengguna bernama Rey akan membeli voucer dan mendapat kode unik
   - Keluaran hasil pengujian:

```
pembangkitan voucer baru oleh pengguna A
{
  "code": "F5QIXN6G",
  "descrption": "Voucher 5 with value Rp.300000"
}

Penukaran voucer dengan kode yang dimiliki
pengguna A oleh pengguna B
{
  "message": "Voucher not Found"
}

Kondisi voucer pada database
[
"88093d430897e481250afb638437e4bdbd9030c19
7fec066a30c2644550307aa",
    "F5QIXN6G",
    0,
    3,
    "2021-12-20 12:31:24",
    "2022-01-20 12:31:24"
]
```

Voucer yang dibangkitkan oleh sebuah akun pengguna tidak dapat digunakan oleh pengguna lain. Pada keluaran hasil pengujian di atas, baris ke-2 dari kondisi voucer merupakan informasi bahwa voucer belum pernah digunakan (0 bernilai *false* dan 1 bernilai *true*).

V. Kesimpulan dan Saran

Keamanan transaksi pada dunia sosial sangatlah penting, terdapat beberapa kesimpulan dari makalah ini. Kesimpulan dan saran adalah sebagai berikut:

1) Berdasarkan permasalahan kemanan pada sistem pembelian voucer pada platform jual beli produk dan layanan, dibuatlah sistem yang memanfaatkan algoritma SHA-256 dalam pembangkitan dan penukaran voucer elektronik.
2) Skema solusi yang dibuat saat ini hanya memenuhi kasus ketika penukaran voucer dilakukan oleh pengguna yang melakukan pembelian.
3) Berdasarkan tiga kasus pengujian, sistem telah berhasil memenuhi seluruh kasus uji tersebut.

Saran yang dapat diberikan untuk pengembangan sistem selanjutnya adalah mengimplementasi kasus ketika pembeli voucer ingin membelikan voucer untuk pengguna lain, sehingga tidak terbatas hanya pembeli yang dapat menukarkan voucer.

Ucapan Terima Kasih

Ucapan puji syukur dipanjatkan kepada Allah SWT atas berkat, rahmat, taufik, serta hidayah-Nya, sehingga makalah dengan judul “Implementasi Skema Pembangkitan dan

Penukaran Voucer Daring Menggunakan Algoritma SHA-256" berhasil selesai.

Terima kasih yang sebesar-besarnya juga diberikan kepada Dr. Ir. Rinaldi Munir, M.T. sebagai dosen pengampu mata kuliah IF4020 Kriptografi yang telah dengan sabar mendidik dan membimbing mahasiswanya selama satu semester.

Referensi

[1] Munir, Rinaldi. 2020. Slide Kuliah IF4020 Kriptografi: Pengantar Kriptografi

[2] Munir, Rinaldi. 2020. Slide Kuliah IF4020 Kriptografi: Fungsi Hash

[3] Wikipedia. SHA-2.

[4] Wagner, Lane. 2020. "How SHA-256 Works Step-By-Step", https://qvault.io/cryptography/how-sha-2-works-step-by-step-sha-256/, diakses pada 20 Desember 2021 pukul 12.56.

[5] Landman, dkk. "Secure Hash Algorithms", https://brilliant.org/wiki/secure-hashing-algorithms/#sha-2, diakses pada 20 Desember 2021 pukul 12.56.

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, 20 Desember 2021

Reyvan Rizky Irsandy 13518136