Implementasi Tanda Tangan Digital untuk Pengamanan *Web Service*

Muhammad Nurdin Husen (13517112)[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung 40132, Indonesia
[1]13517112@std.stei.itb.ac.id

***Abstrak*—Perkembangan internet sudah semakin pesat, tak terkecuali pada penggunaan *web service*. Mayoritas pengguna internet merasa khawatir akan serangan keamanan terhadap informasi pribadi milik mereka. Salah satu serangan yang marak terjadi adalah kebocoran data dan pengaksesan data oleh pihak yang tidak berwenang. Oleh karenanya, perlu ada teknologi untuk melakukan pengamanan informasi pribadi tersebut. Salah satu caranya adalah dengan menerapkan tanda tangan digital berupa *token* untuk melakukan otentikasi terhadap *web service*. Pembuatan *token* tersebut dapat menggunakan algoritma RSA dan fungsi *hash* RIPEMD-160. Pada makalah ini, dibahas mengenai implementasi kedua algoritma tersebut dalam pembuatan tanda tangan digital berupa *token* untuk mengamankan *web service*.**

Keywords—Tanda tangan digital, Token, RIPEMD, RSA, Web Service

I. PENDAHULUAN

Perkembangan internet sudah semakin pesat. Hal tersebut terlihat dari semakin meningkatnya jumlah pengguna internet di dunia. Di tahun 2020, jumlah pengguna internet mencapai 4.54 milyar orang atau mencapai 59% penduduk di dunia [1]. Sekitar 64% pengguna internet di dunia mengkhawatirkan keamanan informasi personal akibat internet [1]. Salah satu teknologi untuk melakukan pengelolaan data dan informasi adalah layanan web atau *web service*. Mengingat hal tersebut, sudah seharusnyalah perkembangan internet diiringi dengan perkembangan teknologi pengamanan *web service* untuk menjamin kemanan data dan informasi.

Salah satu serangan terhadap *web service* yang marak terjadi adalah kebocoran data dan pengaksesan data oleh pihak yang tidak seharusnya. Serangan tersebut terjadi akibat adanya kerentanan atau kelemahan dalam aspek keamanan informasi, yaitu otentikasi dan otorisasi, dari *web service* tersebut. Aspek otentikasi berkaitan dengan pemeriksaan identitas seseorang. Terdapat tiga metode pemeriksaan identitas, yaitu metode *what you know*, metode *what you have*, dan metode *what you are*. Metode *what you know* akan mengenali identitas seseorang berdasarkan apa yang seseorang ketahui, seperti kata sandi dan *personal identification number* (PIN). Metode *what you have* akan mengenali identitas seseorang berdasarkan apa yang dimiliki oleh seseorang, seperti *smartcard* dan kartu tanda penduduk (KTP). Metode *what you are* akan mengenali identitas seseorang berdasarkan sesuatu yang melekat pada seseorang, seperti sidik jari dan wajah. Aspek otorisasi berkaitan dengan apa yang boleh dilakukan seseorang terhadap suatu sumber daya. Salah satu metode dalam otorisasi adalah dengan menggunakan *access control list.* Terdapat tiga jenis *access control list*, yaitu mandatori, diskresioner, dan berbasis peran. Dengan terjaminnya aspek otentikasi dan otorisasi suatu *web service* akan mencegah terjadinya kebocoran data atau pembacaan data oleh pihak yang tidak berwenang.

Hal yang dapat diterapkan untuk menangani permasalahan otentikasi dan otorisasi adalah dengan menggunakan tanda tangan digital. Pada tanda tangan digital, akan terdapat sebuah data berisikan identitas pengguna yang akan mengakses *web service* tersebut (*payload*) dan juga terdapat sebuah *signature* atau tanda tangan pengguna tersebut. Jika tanda tangan tersebut abash, pengguna dapat meneruskan untuk mengakses *web service* tersebut. Jika tidak, pengguna tidak akan bisa mengakses *web service* tersebut. Salah satu cara penandatangan adalah dengan menggunakan kriptografi kunci-publik dan fungsi *hash* [2]. Salah satu algoritma kriptografi kunci-publik adalah RSA dan algoritma fungsi *hash* adalah RIPEMD-160. Pengamanan *web service* menggunakan tanda tangan digital diimplementasikan menjadi sebuah *string* yang mengandung *payload* sekaligus *signature* yang biasanya dikenal dengan istilah *token*.

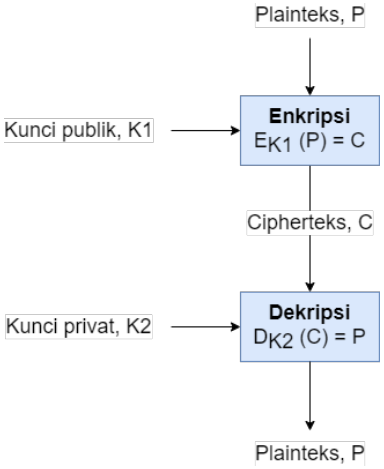
Pada makalah ini, akan dibahas pengamanan *web service* menggunakan tanda tangan digital berupa sebuah *token* dengan memanfaatkan algoritma RSA dan RIPEMD-160.

II. DASAR TEORI

A. Kriptografi Kunci-Publik

Kriptografi kunci-publik pertama kali diperkenalkan pada tahun 1976-an. Sebelumnya, hanya dikenali sistem kriptografi kunci-simetri. Pada kriptografi kunci-simetri proses enkripsi dan dekripsi menggunakan kunci yang sama. Permasalahan muncul ketika seseorang harus mengirimkan kunci tersebut kepada penerimanya agar pesan dapat tersampaikan. Untuk menghindari serangan terhadap pesan, kunci harus dikirimkan melalui saluran yang benar-benar aman. Akan tetapi, saluran tersebut umumnya sangat mahal dan lambat. Untuk mengatasi hal tersebut, mulai dikembangkan sistem kriptogarfi kunci-publik [3].

Secara umum, ide dasar kriptografi kunci-publik adalah dengan menggunakan pasangan kunci, kunci privat dan kunci public, untuk melakukan pengamanan pesan. Secara umum, cara kerja kriptografi kunci-publik diilustrasikan pada gambar 1.



Gambar 1. Gambaran Umum Kriptografi Kuci-Publik

Kunci publik milik penerima digunakan pengirim untuk mengenkripsi pesan yang akan dikirimkan. Kunci publik ini bersifat umum atau tidak rahasia. Berbeda halnya dengan kunci publik, kunci privat bersifat rahasia. Kunci privat digunakan penerima untuk mendekripsi pesan yang dikirimkan oleh pengirim.

Dibandingkan dengan kriptografi kunci-simetri, kriptografi kunci-publik memiliki beberapa kelebihan. Tidak diperlukan pengiriman kunci yang digunakan untuk enkripsi pesan sehingga risiko serangan akibat kebocoran kunci akan berkurang. Selain itu juga, tidak diperlukan saluran yang aman untuk pengiriman kunci tersebut.

Terdapat beberapa algoritma kriptografi kunci-publik untuk melakukan enkripsi/dekripsi pesan, diantaranya adalah algoritma RSA, algoritma Rabin, algoritma ElGamal, dan algoritma *elliptic curve cryptography* (ECC).

B. *Algoritma RSA*

Algoritma RSA ditemukan pada tahun 1976 oleh tiga orang peneliti dari *Massachusetts Institute of Technology* (MIT), yaitu Ronal Rivest, Adi Shamir, dan Len Adleman. Algoritma RSA diberi nama sesuai dengan penemunya, Rivest-Shamir-Adleman. Kemanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan bulat yang besar menjadi faktor-faktor prima [4].

Terdapat dua bagian penting dalam algoritma RSA, yaitu proses pembangkitan kunci dan proses enkripsi/dekripsi pesan. Berikut adalah algoritma pembangkitan kunci RSA [4]:

1. Pilih dua buah bilangan prima sembarang, p dan q. Kedua bilangan ini bersifat rahasia. Semakin besar nilai p dan q akan semakin sulit dipecahkan.
2. Hitung sebuah bilangan n yang merupakan hasil perkalian p dan q. Bilangan ini tidak rahasia.
3. Hitung nilai z yang merupakan *totient euler* dari n atau $z = (p - 1)(q - 1)$. Niali z bersifat rahasia.
4. Pilih sebuah bilangan e yang relatif prima terhadap z sebagai kunci publik. Bilangan ini tidak rahasia.
5. Hitung d sebagai kunci privat dengan $ed \equiv 1 \pmod z$. Niali d bersifat rahasia.

Algoritma di atas akan menghasilkan sebuah pasangan kunci privat dan kunci publik. Kunci privat yang dihasilkan adalah pasangan nilai d dan n. Sedangkan kunci publik adalah pasangan nilai e dan n. Proses enkripsi pada algoritma RSA adalah sebagai berikut [4]:

1. Nyatakan pesan menjadi blok-blok *plaintext* : $m_1, m_2, m_3, \ldots$ dengan ketentuan bahwa $0 \leq m_i \leq n - 1$.
2. Hitung blok *ciphertext* c_i untuk blok *plaintext* m_i menggunakan kunci publik e dengan persamaan berikut

$$c_i = (m_i)^e \bmod n$$

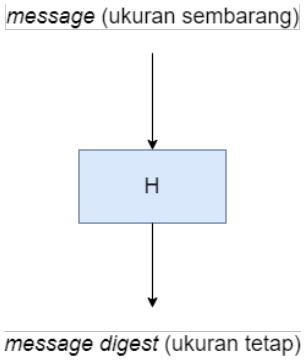
Adapun proses dekripsi pada algoritma RSA adalah sebagai berikut [4]:

1. Misalkan blok-blok *ciphertext* adalah c_i atau $c1, c_2, c_3, \ldots$
2. Hitung blok *plaintext* m_i dari blok *ciphertext* c_i menggunakan kunci privat d dengan persamaan berikut

$$m_i = (c_i)^d \bmod n$$

C. *Fungsi Hash*

Fungsi *hash* merupakan sebuah fungsi yang mengkompresi pesan berukuran sembarang. Luaran dari fungsi *hash* merupakan sebuah *string* berukuran tertentu dan tetap yang biasa disebut pesan ringkas (*message digest*) atau nilai *hash* (*hash value*). Hasil dari fungsi *hash* bersifat *irreversible*, artinya *message digest* tidak dapat dikembalikan menjadi pesan semula [5]. Gambaran umum fungsi *hash* diilustrasikan pada gambar 2.



Gambar 2. Ilustrasi Fungsi *Hash*

Terdapat tiga sifat fungsi *hash*. Sifat pertama adalah *collision resistance*, artinya sangat sulit menemukan dua buah pesan yang menghasilkan *message digest* yang sama. Sifat kedua adalah *preimage resistance*, artinya sulit menemukan pesan untuk sembarang *message digest.* Sifat lainnya adalah

second preimage resistance, artinya sulit menemukan pesan kedua yang memiliki *message digest* yang sama dengan suatu pesan [5].

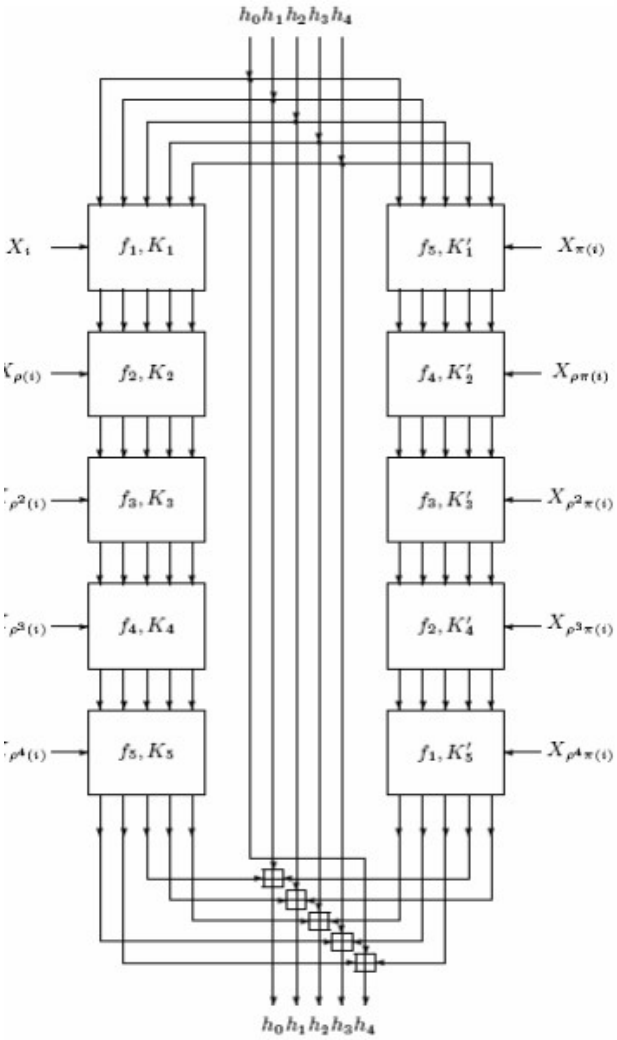
Terdapat beberapa fungsi *hash*. Diantaranya adalah MD2/4/5, RIPEMD, SHA-1/2/3, WHIRLPOOL, dan BLAKE.

D. Algoritma RIPEMD-160

Fungsi *hash* yang paling popular digunakan adalah MD4 yang dikemukakan oleh Ron Rivest pada tahun 1990. Namun, algoritma MD4 telah ditemukan koalisi. Pada tahun 1991, Ron Rivest mengajukan fungsi *hash* yang baru, bernama MD5 [6]. Fungsi *hash* ini menjadi algoritma yang banyak digunakan. Akan tetapi, juga sudah ditemukan kolisinya. Untuk menjawab analisa fungsi *hash* MD4 dan MD5 tersebut, konsorsium RIPE (RACE *Integrity Primitives Evaluation*) mengajukan sebuah fungsi *hash* baru yang bernama RIPEMD (RIPE *message digest*). Algoritma RIPEMD dikembangkan dari algoritma *hash* MD4 varian 256-bit. Fitur utama dari algoritma RIPEMD adalah adanya dua rantai komputasi yang berbeda, independen, dan paralel. Hasil kedua rantai komputasi tersebut akan digabungkan pada akhir prosesnya. Salah satu varian algoritma RIPEMD adalah RIPEMD-160 yang memiliki luaran berukuran 160-bit [6].

RIPEMD-160 akan melakukan modifikasi pesan masukkan sebelum diproses. Modifikasi dilakukan jika pesan memiliki panjang yang bukan kelipatan 512 bit. Modifikasi dilakukan dengan menambahkan *padding* pada pesan masukkan dengan menambah satu bit 1 dan diikuti dengan kumpulan bit 0. Bit-bit terakhir sebesar 64 bit dari *padding bits* menyatakan panjang pesan sebenarnya yang dinyatakan dalam bit. Setelah dilakukan modifikasi, pesan tersebut akan dibagi menjadi blok-blok yang masing-masing berukuran 512 bit. Setiap blok akan dibagi menjadi 16 *string* dengan ukuran masing-masing subblock adalah 4 *byte*. Setiap *string* berukuran 4 *byte* tersebut diubah menjadi *word* 32-bit dengan metode *little-endian*.

Nilai *hash* hasil RIPEMD-160 direpresentasikan dalam 5 *word* 32 bit yang akan digabung menjadi *string* 160 bit dengan metode *little-endian*. Pada *state* awal, terdapat lima *variable* yang diinisialisasi dengan suatu nilai tertentu (*initial vector*). Kelima nilai tersebut (dalam heksadesimal) adalah *67452301*, *EFCDAB89*, *98BADCFE*, *10325476*, dan *C3D2E1F0*. Kelima *variable* tersebut dinamakan *chaining variable*. Bagian utama algoritma RIPEMD adalah fungsi kompresi. Fungsi kompresi menerima masukkan berupa blok *string* beserta *chaining variable* untuk menghitung *state* baru berdasarkan *state* sebelumnya [6]. Secara umum, diagram fungsi kompresi diilustrasikan pada gambar 3.



Gambar 3. Diagram Fungsi Kompresi RIPEMD-160 [6]

Fungsi kompresi terdiri dari 5 ronde yang setiap rondenya terdapat 2 proses yang berjalan secara paralel dan independen satu sama lain. Dalam setiap proses, terjadi 16 langkah akibat fungsi kompresi yang menerima masukkan blok 512 bit yang dibagi menjadi 16 subblok seukuran 16 bit. Dalam satu langkah, hanya 1 subblok yang akan diproses. Total langkah yang terjadi pada fungsi kompresi ini ada sebanyak 160 langkah [6].

E. Tanda Tangan Digital

Seperti halnya pada dokumen cetak, tanda tangan digunakan untuk otentikasi dokumen tersebut. Tanda tangan digukanan sebagai otentikasi dokumen karena sifatnya sebagai bukti otentik yang tidak dapat dilupakan, tidak dapat dipindah untuk digunakan ulang, dan tidak dapat disangkal. Fungsi tanda tangan seperti itu juga diterapkan pada dokumen elektronik yang dikenal dengan nama tanda tangan digital. Perbedaan tanda tangan dokumen kertas dengan digital terletak pada isi tanda tangan tersebut. Pada dokumen kertas,

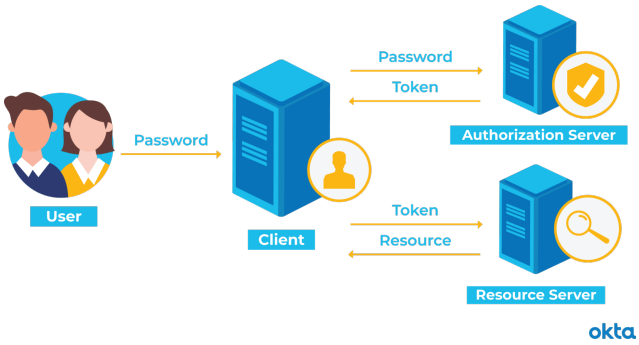
tanda tangan untuk setiap dokumen akan selalu sama. Untuk tanda tangan digital akan berbeda untuk setiap dokumen. Tanda tangan digital adalah nilai kriptografis yang bergantung pada isi pesan dan kunci yang digunakan [2].

Salah satu cara penandatanganan digital adalah dengan menggunakan kriptografi kunci-publik dan fungsi *hash*. Penandatanganan dengan cara seperti itu dapat memberikan kerahasiaan pesan dan otentikasi pesan tersebut. Pada beberapa kasus, seringkali kerahasiaan pesan tidak dibutuhkan. Akan tetapi, hal tersebut menjadi penting pada kasus penangamanan *web service*.

Secara umum, proses penandatanganan oleh pengirim akan dilakukan menjadi beberapa tahapan. Pesan yang akan dikirimkan akan dilakukan *hash* untuk mendapatkan *message digest* yang kemudian akan dienkripsi dengan menggunakan kunci privat milik pengirim. Hasil enkripsi itulah yang menjadi *signature*. Pada tahapan akhir, *signature* digabungkan dengan pesan yang akan dikirimkan. Proses verifikasi oleh penerima dilakukan dengan memisahkan *signature* dengan pesan. Kemudian *signature* akan didekripsi menggunakan kunci publik pengirim dan hasilnya akan dibandingkan dengan *message digest* dari pesan yang diterima. Jika keduanya sama, tandatangan tersebut absah.

F. Token

Salah satu protokol untuk memverifikasi identitas pengguna adalah dengan menggunakan *token*. Pengguna akan menerima sebuah *token* setelah melakukan otentikasi. Kegunaan *token* tersebut adalah sebagai tiket bagi pengguna untuk dapat berinteraksi dengan suatu layanan pada *web server*. Dengan adanya *token*, pengguna tidak harus melakukan otentikasi secara berulang-ulang ketika akan melakukan interaksi dengan *web server*. Pengguna hanya perlu menggunakan *token* yang didapat dari satu kali proses otentikasi. Secara umum, protocol otentikasi berbasis token dapat dilihat pada gambar 4.



Gambar 4. Protokol Otentikasi dengan *Token* [7]

III. Pengaman Web Service Menggunakan Token

A. Implementasi Token

Implementasi *token* adalah berupa *string* berukuran tertentu dan tetap. Bagian utama *token* terdiri dari 2 hal, yaitu *signature* dan *payload*. Kedua bagian ini digabungkan menjadi sebuah *string* dengan tanda titik sebagai pemisahnya. Berikut ini merupakan contoh *token* tersebut.

```
KY+AUKfzR21QWpcjKyzcNq2hqlb5mZtZhU2HTscXIM
BtL36M08P/pORtyGHoB1Iy3kOIt4zaXXbGLA3N2mZz
Ykm4pddxGWb5Qpn6xHgwOMEUiwhgrY5YGsqObAn3yD
HHnSp2jrxymXWwK2rbOi+PT/d/8Q5zzjLjiC2sgsvy
Jzo=.eyJlbWFpbCI6ImVtYWlsQGNvbnRvaC5jb20iL
CJuYW1lIjoiSm9obiBEb2UiLCJyb2xlIjoidXNlciJ
9
```

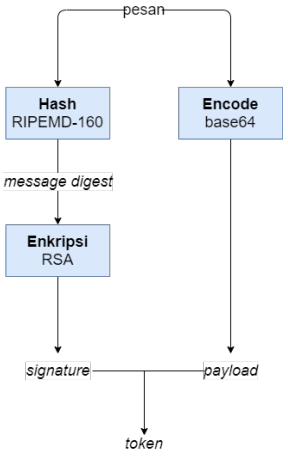
Bagian *signature* berisikan tanda tangan digital dari *payload* yang diberikan. Bagian ini merupakan hasil enkripsi *message digest* dari *payload* menggunakan kunci *private* penandatangan. Bagian *payload* berisikan data-data berkaitan dengan pengguna yang diotentikasi atau berisikan data-data lainnya yang ingin dikirimkan melalui *token*. Data-data tersebut direpresentasikan menjadi sebuah JSON (*javascript object nation*). Berikut adalah contoh *payload* yang dikirimkan

```
{
   email: email@contoh.com,
   name: John Doe,
   role : user
}
```

B. Pembangkitan Token

Pembangkitan *token* dilakukan dengan menandatangani *payload* dengan algoritma RSA dan fungsi *hash* RIPEMD-160. Setiap *resource server* atau *web server* memiliki dua buah kunci yang merupakan pasangan nilai, yaitu kunci publik dan kunci privat. Kedua kunci ini dibutuhkan saat membangkitkan dan verifikasi *token*.

Proses pembangkitan token diawali dengan membentuk data-data yang akan dikirimkan melalui token. Setelah itu, data-data tersebut akan di-*encode* menjadi base64 sehingga didapatkan sebuah *payload*. Data-data tersebut juga akan didapatkan *message digest*-nya menggunakan RIPEMD-160 dan dienkripsi dengan algoritma RSA menggunakan kunci privat *web server*. Secara umum, proses pembangkitkan *token* diilustrasikan pada gambar 5.

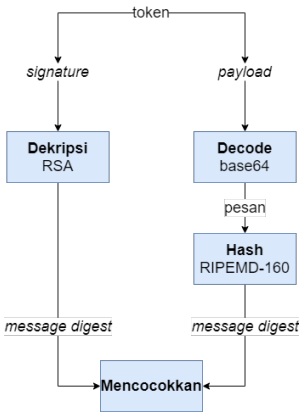


Gambar 5. Proses pembangkitan *token*

C. *Verifikasi token*

Verifikasi *token* dilakukan dengan memverifikasi tandatangan *payload* dengan algoritma RSA dan fungsi *hash* RIPEMD-160. Setiap *resource server* atau *web server* memiliki dua buah kunci yang merupakan pasangan nilai, yaitu kunci publik dan kunci privat. Kedua kunci ini dibutuhkan saat membangkitkan dan verifikasi *token.*

Proses verifikasi token diawali dengan memisahkan bagian *signature* dan *payload* dari *token* terlebih dahulu. Kemudian *payload* akan di-*decode* dengan base64 yang hasilnya akan di-*hash* dengan RIPEMD-160. Sementara itu, *signature* akan didekripsi dengan algoritma RSA dengan menggunakan kunci publik milik *web server*. Secara umum, proses verivikasi *token* digambarkan pada gambar 6.



Gambar 6. Proses Verifikasi *Token*

IV. PENGUJIAN

A. *Pendaftaran pengguna*

Pengujian pertama dilakukan dengan mendaftarkan pengguna baru (*register*) ke dalam *web service*. Pendaftaran dilakukan dengan mengisikan beberapa informasi berikut

```
{
      name: 'John Doe',
      email: 'email@contoh.com',
      password: 'admin123'
      role: 'pengguna'
}
```

Didapatkan sebuah umpan balik dari *web service* sebagai berikut:

```
{
      "status": 200,
      "data": {
          "email": "email@contoh.com",
          "role": "user",
          "name": "John Doe"
      }
}
```

Setelah didapatkan respon seperti tertera di atas, data pengguna tersebut sudah tersimpan di dalam *web service*.

B. *Pembangkitan Token*

Untuk melakukan pembangkitan *token*, pengguna perlu melakukan otentikasi dengan *login* menggunakan *email* dan *password* yang sesuai pada *web service*. Pengujian dilakukan dengan mengirimkan data berikut sebagai proses otentikasi

```
{
      email: 'email@contoh.com',
      password: 'admin123'
}
```

Data-data yang dikirimkan di atas sesuai dengan data yang tersimpan pada *web service* sehingga memberikan respon sebagai berikut

```
{
    "status": 200,
    "data": {
        "id": 1,
        "email": "email@contoh.com",
        "token":
"c8sh5Kax2Fr5YyKed3BmuG5Ryq/gC2XHEwDKcpou
fmEUOc8RG2qwCLUNEkjklw8sOd6/5VUoXFp8bleid
mnZcKvfLphVZOER2Y38h5sKBbppFIErrcZS3M4Y3X
Wo+Zk3h38wDg3sIWBBAdY0Xf6oFmHmyZOG/ryyTOB
pG67zyrY=.eyJlbWFpbCI6ImVtYWlsQGNvbnRvaC5
jb20iLCJuYW1lIjoiSm9obiBEb2UiLCJyb2xlIjoi
dXNlciJ9"
    }
}
```

Jika proses otentikasi gagal, didapatkan respon berikut

```
{
      "status": 400,
      "message": "CREDENTIALS_ERROR"
}
```

C. *Verifikasi Token*

Untuk melakukan verifikasi *token*, pengguna perlu melakukan *request* ke *end-point* pada *web service* yang bersifat privat. Proses verifikasi *token* dilakukan sebagai bentuk *middleware* sebelum melakukan eksekusi terhadap *request*. Saat melakukan *request*, pengguna perlu mengirimkan *headers* bernama *token* yang berisikan *token* yang *valid*. Pengujian dilakukan dengan beberapa skenario:

1. Menggunakan *token* yang *valid*

Pengujian dilakukan dengan mengirimkan *request* untuk mendapatkan informasi pribadi pengguna yang bersangkutan. Pengujian dilakukan dengan mengirimkan *request* dan terdapat sebuah *headers* bernama *token* dengan nilai berupa *token* yang *valid* sebagai berikut

```
c8sh5Kax2Fr5YyKed3BmuG5Ryq/gC2XHEwDKcpoufm
EUOc8RG2qwCLUNEkjklw8sOd6/5VUoXFp8bleidmnZ
cKvfLphVZOER2Y38h5sKBbppFIErrcZS3M4Y3XWo+Z
k3h38wDg3sIWBBAdY0Xf6oFmHmyZOG/ryyTOBpG67z
yrY=.eyJlbWFpbCI6ImVtYWlsQGNvbnRvaC5jb20iL
CJuYW1lIjoiSm9obiBEb2UiLCJyb2xlIjoidXNlciJ
9
```

Didapatkan respon dari *web service* sebagai berikut

```
{
      "status": 200,
      "data": {
          "id": 1,
          "email": "email@contoh.com",
          "phone_number": "08123456789",
          "address": "Jln. xyz",
          "bio": "Lorem Ipsum",
          "birth_date": "1999-12-31",
          "father_name": "Bob",
          "mother_name": "Alice",
          "name": "John Doe",
      }
}
```

2. Menggunakan *token* yang tidak *valid*

Pengujian dilakukan dengan mengirimkan *request* untuk mendapatkan informasi pribadi pengguna yang bersangkutan. Pengujian dilakukan dengan mengirimkan *request* dan terdapat sebuah *headers* bernama *token* dengan nilai berupa *token* yang telah diubah sebagai berikut

```
d8sh5Kax2Fr5YyKed3BmuG5Ryq/gC2XHEwDKcpoufm
EUOc8RG2qwCLUNEkjklw8sOd6/5VUoXFp8bleidmnZ
cKvfLphVZOER2Y38h5sKBbppFIErrcZS3M4Y3XWo+Z
k3h38wDg3sIWBBAdY0Xf6oFmHmyZOG/ryyTOBpG67z
yrY=.eyJlbWFpbCI6ImVtYWlsQGNvbnRvaC5jb20iL
CJuYW1lIjoiSm9obiBEb2UiLCJyb2xlIjoidXNlciJ
9
```

Didapatkan respon dari *web service* sebagai berikut

```
{
      "status": 400,
      "message": "CREDENTIALS_ERROR"
}
```

KESIMPULAN

Pengamanan *web service* dapat dilakukan dengan menggunakan *token.* Pembuatan dan verifikasi *token* merupakan implementasi tanda tangan digital dengan menggunakan algoritma RSA dan RIPEMD-160. Pada percobaan didapatkan hasil bahwa untuk pengguna yang terotentikasi akan memiliki sebuah *token.* Sebuah *token* yang *valid* dapat digunakan pengguna untuk melakukan *request* pada *private end-point*. Sedangkan *token* yang tidak *valid,* tidak dapat digunakan untuk melakukan *request* pada *private end-point*. Hal tersebut dapat mengurangi risiko data bocor dan pengaksesan data pribadi oleh orang yang tidak berwenang.

DAFTAR PUSTAKA

[1] https://wearesocial.com/digital-2020

[2] R. Munir, "Tanda-tangan Digital", [online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Tanda-tangan-digital-2020.pdf

[3] R. Munir, "Kriptogarfi Kunci-Publik" [online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/Kriptografi-Kunci-Publik-2021.pdf

[4] R. Munir, "Algoritma RSA", [online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Algoritma-RSA-2020.pdf

[5] R. Munir, "Fungsi Hash", [online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/Fungsi-hash-2021.pdf.

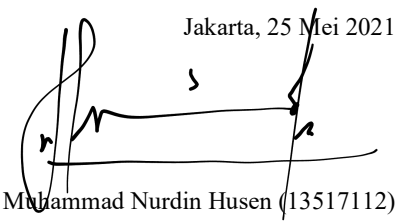
[6] L. Roland, "Algoritma RIPEMD", [online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2006-2007/Makalah2/Makalah-071.pdf

[7] Okta Inc., "What Is Token-Based Authentication?" https://www.okta.com/identity-101/what-is-token-basedauthentication/ (accessed Dec. 21, 2020).

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.

Jakarta, 25 Mei 2021



Muhammad Nurdin Husen (13517112)