Implementasi Algoritma ElGamal untuk Pembuatan Web Token

I Kadek Yuda Budipratama Giri / 13516115
Informatika
Sekolah Tinggi Elektronika dan Informatika
Institut Teknologi Bandung, Jl. Ganesha No. 10 Bandung 40132, Indonesia
13516115@std.stei.itb.ac.id

***Abstract*—Untuk menghindari adanya pengaksesan *web service* secara illegal dan penyalahgunaan data yang diakses tanpa perijinan, dibutuhkan suatu penanda agar *request* dapat dikenali sebagai *request* yang boleh mengakses *web service*. Pada makalah ini, akan dibahas penggunaan *token* yang dibuat dengan menggunakan algoritma ElGamal.**

Keywords—Web Token, ElGamal, Autentikasi, Web Service

I. PENDAHULUAN

Teknologi *web* merupakan teknologi yang banyak digunakan oleh manusia pada masa kini. Penggunanya pun semakin bertambah tiap hari. Oleh sebab itu, keamanan menjadi hal penting untuk sebuah *web service*, karena pada masa ini, *web service* sudah menjadi kebutuhan banyak orang. Apabila suatu *web service* tidak dapat melindungi penggunanya dari kejahatan seperti penyalahgunaan data, pencurian data, dan serangan *privacy* lainnya, maka masyarakat tidak akan menggunakan *web service* tersebut.

Untuk melindungi suatu *web service*, diperlukan adanya autentikasi dan otorisasi pengguna. Autentikasi adalah proses untuk membuktikan bahwa seorang pengguna adalah benar pengguna yang terdaftar pada *web service* tersebut, sehingga pengguna berhak mengakses *web service* tersebut [1]. Pengguna harus divalidasi bersarkan data yang dapat dibuktikan keasliannya, seperti kombinasi antara *e-mail*, *password*, *username*, dan sebagainya. Tujuannya adalah untuk mencegah pengguna yang tidak terdaftar sebagai pengguna *web service* yang dibolehkan untuk menggunakan *web service* tersebut. Selain itu, autentikasi dibutuhkan agar siapapun yang ingin menggunakan *web service* untuk mendaftar pada *web service* yang dituju terlebih dahulu.

Otorisasi pengguna adalah proses untuk memberikan akses kepada pengguna yang telah melakukan autentikasi untuk mengakses *web service* [1]. Otorisasi dilakukan untuk membatasai penyebaran data untuk pengguna tertentu yang diinginkan. Selain itu, otorisasi juga mencegah terjadinya kebocoran data kepada orang yang tidak berhak

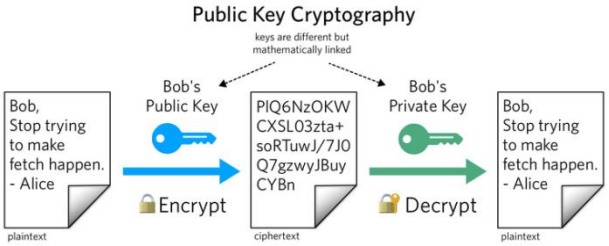
Untuk menjalankan sistem otorisasi dan autentikasi, terdapat dua metode yang paling umum digunakan, yaitu *session* dan *token*. *Token* adalah sebuah *string* yang digunakan sebagai penanda agar pengguna dapat mengakses suatu *web service* sebagai pengguna yang terdaftar. Di dalam token, terdapat informasi mengenai pengguna beserta dengan *digital signature*. *Digital signature* ini berfungsi untuk memastikan bahwa token yang dibuat adalah asli dan berasal dari sistem *web service*, sehingga pengguna tidak dapat sembarang mengubah informasi pada token atau membuat token sendiri untuk masuk pada sistem.

Pada makalah ini, akan dibahas mengenai autentikasi dan otorisasi sebuah *web service* dengan menggunakan *token* yang dibuat dengan menggunakan algoritma ElGamal untuk membuat *digital signature* pada *token*.

II. DASAR TEORI

A. Public Key Cryptography

Public Key Cryptography atau *Asymmetric Cryptography* adalah kriptografi yang menggunakan dua buah kunci, yaitu kunci publik (public key) dan kunci privat (private key) [2]. Public key dapat dibagikan dan digunakan untuk melakukan enkripsi pesan yang akan dikirim atau untuk melakukan verifikasi *signature*, sementara private key digunakan untuk melakukan dekripsi pada pesan yang diterima atau membuat *digital signature*. Dengan adanya mekanisme ini, tidak dibutuhkan pengiriman kunci rahasia kepada pihak yang menerima pesan, karena dekripsi dapat dilakukan dengan kunci privat yang dimiliki oleh penerima pesan. Contoh algoritma yang menggunakan kriptografi public key adalah algoritma RSA dan El Gammal.



Gambar 1. Ilustrasi Public Key Cryptography

B. Skema Tanda Tangan Algoritma ElGamal

Algoritma ElGamal adalah sebuah algoritma *public key crtptography* yang dikembangkan oleh Taher Elgamal di tahun 1985. Algoritma ini menggunakan sistem yang digunakan Diffie-Hellman Key Exchange untuk menukarkan kunci. Algoritma ini dipakai dalam enkripsi dan pembuatan tanda tangan *(signature*).

1) Pembangkitan Kunci

Untuk menghasilkan kunci untuk melakukan enkripsi maupun tanda tangan, pertama-tama dipilih sebuah bilangan p, di mana nilai p harus prima. Lalu, dipilih sebuah bilangan g yang nilainya $1 \leq g < 1 - p$ dan x yang nilainya $1 \leq x < p - 2$. Terakhir, dihitung nilai dari y sehingga dipenuhi

$$y \equiv g^x mod\ p$$

Pasangan kunci publik yang dapat dibagikan adalah (p, g, y), sementara pasangan kunci privat adalah (p,g,x) [5].

2) Pembuatan Tanda Tangan

Misal m adalah pesan yang ingin kita cantumkan sebagai tanda tangan, maka untuk melakukan pembuatan tanda tangan, kita membutuhkan pasangan kunci privat (p,g,x). Skema untuk membuat tanda tangan digital adalah sebagai berikut [5].

1. Pesan m di-*hash* terlebih dahulu, sehingga dihasilkan H(m).
2. Nilai k dipilih, sehingga $GCD(k, p-1) = 1$. Dengan kata lain, nilai k merupakan *primitive root* dari p.
3. Nilai r dihitung dengan formula berikut

$$r \equiv a^k mod\ p$$

4. Nilai s dihitung dengan formula berikut

$$s \equiv k^{-1}(H(m) - zr)$$

5. Tanda tangan yang dihasilkan adalah pasangan dari (r,s).

3) Verifikasi Tanda Tangan

Misalkan *H(m)* adalah pesan yang telah dihash, maka skema untuk melakukan verifikasi tanda tangan adalah sebagai berikut [5].

1. Nilai *v1* dihitung dengan formula berikut

$$v1 \equiv y^r r^s$$

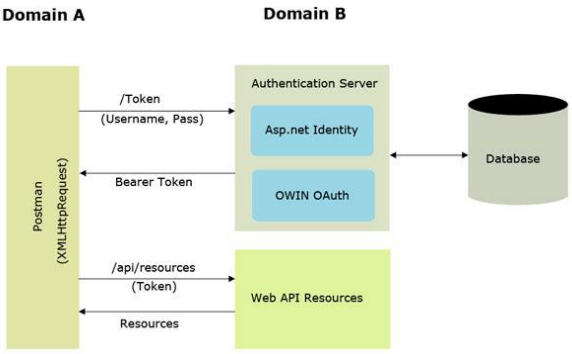
2. Nilai *v2* dihitung dengan formula berikut

$$v2 \equiv g^{H(m)}$$

3. Apabila *v1* dan *v2* tidak sama, maka tanda tangan tersbeut tidak valid untuk pesan yang yang ada.

C. Token-based Authorization

Token-based Authorization adalah sebuah sistem yang memungkinkan autorisasi dengan menggunakan *token* sebagai pengenal request yang masuk. *Token* disimpan oleh *client* untuk nantinya diverifikasi oleh *server* ketika masuk sebuah request [3].



Gambar 2. Ilustrasi Token-based Authorization [4]

Keunggulan dari *token-based authorization* adalah *server* tidak perlu menyimpan state dari pengguna, karena *state* dari pengguna sudah disimpan di bagian *client*, sehingga beban penyimpanan *server* menjadi berkurang. *Token-based Authorization* cocok digunakan untuk *server* dengan skala besar, karena beban penyimpanan server menjadi lebih kecil.

III. IMPLEMENTASI

A. Struktur Token

Struktur dari token yang dibuat terdiri atas dua bagian, yaitu bagian *Payload* dan *Signature*. Kedua bagian ini dipisahkan dengan tanda titik (“.”). Contoh hasil keluaran token adalah sebagai berikut.

```
yJzYW1wbGUiOiJIZWxsbyB3b3JsZCEiLCJpYXQi
OjE1NTc0NjMzMDIsImV4cCI6MTU1NzQ2NDIwMn0
=.zLvbk92q4Ky82o3OheChtdCPx4TEqdaVxa3Wi
eCrqtWK4KOu4KW64KWpwqDVmMSq3oPFuNCh2Zbb
vsiu1bXgpI7Ek8Wr3JzgqIvDiNCs4Kab3J3gop7
DiMWaw7LUi9+1xY7gob7XoOCjns2c4KG4zqTUiu
Cti9aJzqDgp5LVjHPgpKDNm+CuhN+X3ZXSo8e4z
o/aiNqv2IvgrI3CgOCgmOCkusOW3obOgeCpuMW4
3Y3gp6HEgQ==
```

Payload berisi data apa saja yang akan dimasukkan ke dalam token. Data ini bisa dibuat sendiri isinya, tetapi terdapat dua komponen yang selalu ada pada token, yaitu “iat”, yang berisi waktu token dibuat, dan “exp”, yang berisi wakktu token kedarluwasa. *Payload* nantinya juga akan menjadi komponen *message* dalam pembuatan *signature*. *Payload* didefinisikan dalam bentuk JSON (*JavaScript Object Notation*), tetapi dituliskan menjadi *string* dengan melakukan *encoding* JSON ke Base64. Contoh dari *payload* yang valid adalah sebagai berikut

```
{
    sample: 'Hello world!',
    iat: 1557463302,
    exp: 1557464202
}
```

Signature berisi *signature* dari data *payload* yang dikirim. Mekanisme pembuatan *signature* menggunakan skema tanda tangan algoritma ElGamal. Pada algoritma ini, digunakan fungsi *hash* SHA1 untuk melakukan *hash* pada pesan yang dimasukkan pada *signature*. Tujuannya adalah agar data tidak dapat diambil dari *signature*. Selain itu, penggunaan *hash* akan menghemat ukuran dari *signature* yang dihasilkan.

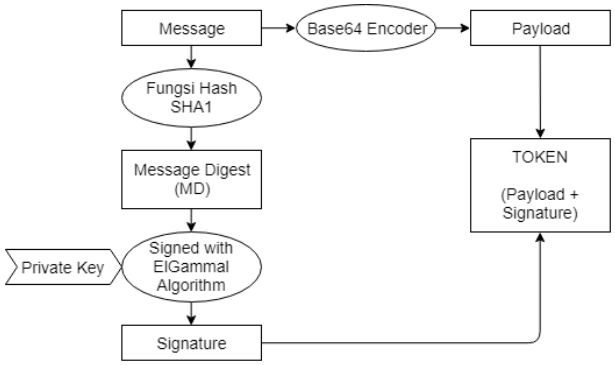
Kunci publik dan kunci privat disimpan di dalam *database* pengguna. Hal ini dilakukan agar setiap token yang masuk dapat diidentifikasi penggunanya.

B. Pembuatan Token

Pembuatan token dilakukan dengan cara memperoleh kunci publik dan kunci privat terlebih dahulu. Kunci publik tersebut diperoleh dengan cara mengakses *database* dari pengguna yang baru akan masuk atau dengan mengambil kunci publik yang ikut disimpan bersama token dalam bentuk *cookies*. Sementara itu, kunci privat hanya dapat diambil dengan cara mengakses *database server*. Dengan demikian, kunci privat tidak akan dapat terbaca oleh pengguna.

Setelah kunci didapat, pesan akan dimasukkan ke dalam fungsi untuk diproses menjadi token. Bagian *payload* diisi dengan pesan berbentuk JSON yang pertama dibuat menjadi *string* terlebih dahulu, lalu di-*encode* menjadi Base64. Sementara itu, bagian *signature* dibuat dengan memasukkan

string dari pesan berbentuk JSON ke dalam fungsi hash SHA1 menjadi *Message Digest* (MD). Setelah itu, MD akan masuk ke fungsi penandatanganan dengan algoritma ElGamal. Setelah itu, *payload* dan *signature* disatukan menjadi satu *string* dan dikembalikan sebagai token.

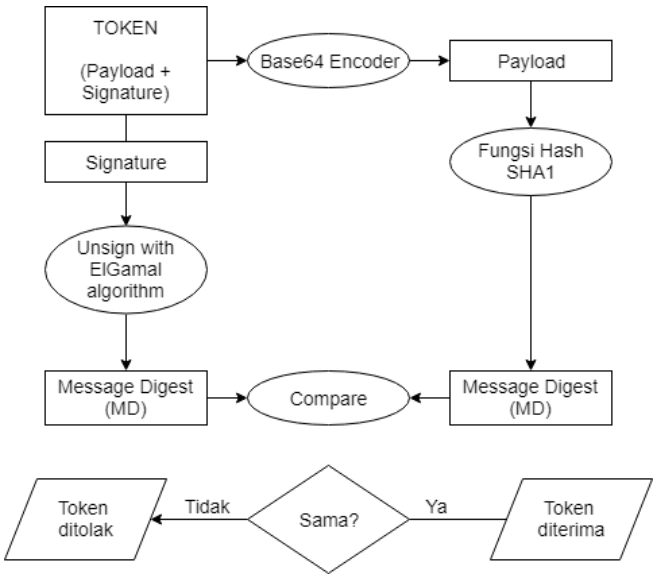


Gambar 3. Skema Pembuatan Token ElGamal

C. *Verifikasi Token*

Untuk melakukan verifikasi *token*, pertama-tama *token* dan kunci publik diambil terlebih dahulu dari *request* yang masuk. Setelah itu, *token* dipisah menjadi *payload* dan *signature*. *Payload* akan di-*decode* menjadi *string* biasa, sementara *signature* akan diproses untuk dinyatakan apakah *token* yang dating valid atau tidak.

Untuk memeriksa keaslian *token*, fungsi verifikasi akan mengambil *payload* yang telah di-*decode*. Setelah itu, *payload* akan dimasukkan ke dalam fungsi *hash* SHA1 untuk dijadikan *Message Digest* (MD). *Signature* yang diperoleh dari *token* akan dihitung nilai *v1* sesuai pada algoritma verifikasi tanda tangan ElGamal. Sementara itu, MD dari *payload* akan dihitung nilai *v2* sesuai pada algoritma verifikasi tanda tangan ElGamal. Apabila nilai *v1* dan *v2* sama, maka *token* dinyatakan valid. Jika tidak, token akan dianggap tidak valid dan ditolak.



Gambar 4. Skema Verifikasi Token

IV. Hasil Pengujian Dan Analisis

A. *Pengujian Performa Pembuatan dan Verifikasi Token*

Pembuatan *Token ElGamal* akan diujikan performanya berdasarkan panjang *token* yang dihasilkan dan waktu eksekusi pembuatan *token*. Untuk melakukan pengujian, akan dibuat sebuah pesan JSON yang berisi sebagai berikut.

Pesan

```
{
    name: 'Kadek Yuda',
    email: 'yudaikadek22@gmail.com',
    role: 'Admin'
}
```

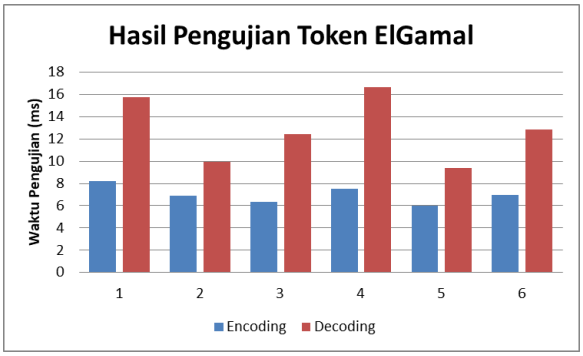
Hasil dari pengujian ini menghasilkan *token* sebagai berikut.

ElGamal Token

```
eyJuYW1lIjoiS2FkZWsgWXVkYSIsImVtYWlsIjo
ieXVkYWlrYWRlazIyQGdtYWlsLmNvbSIsInJvbG
UiOiJBZG1pbiIsImlhdCI6MTU1NzQ2NzE0MiwiZ
XhwIjoxNTU3NDY4MDQyfQ==.4KSa4KK524DLu8S
z4KiFzYrgqLTgo5/ZosaT2InIkOCqhNqDxpzche
Civ9OM4KKSfeCqr8Sy3LHgpKPJpNOb4KqJ4KKIy
L/XpcmExpvdj+CmnNyHyKvSvOCnm86UxrLgqLbg
qLvgopvgoI/frtaD4KC6T9K+24PCk8Ku0bzHism
EbuCotMyq4KaZzL7gqrPGotOL4KO8OOCnpeCksM
WO4KSY3LvIt+Cmp9ql4KecJ9+WVOCnk9iJ
```

Setelah token dibuat, dapat dilihat bahwa token yang dihasilkan sangat besar. Hal ini dikarenakan algoritma ElGamal menghasilkan *signature* dua kali lebih besar dibandingkan dengan ukuran pesan yang masuk.

Sementara itu, bila dibandingkan waktunya, waktu untuk pembuatan token lebih cepat dibandingkan dengan waktu untuk memverifikasi token. Hal ini disebabkan panjang dari *signature* yang lebih panjang daripada *payload* yang diberikan, sehingga waktu verifikasi menjadi lebih lama.

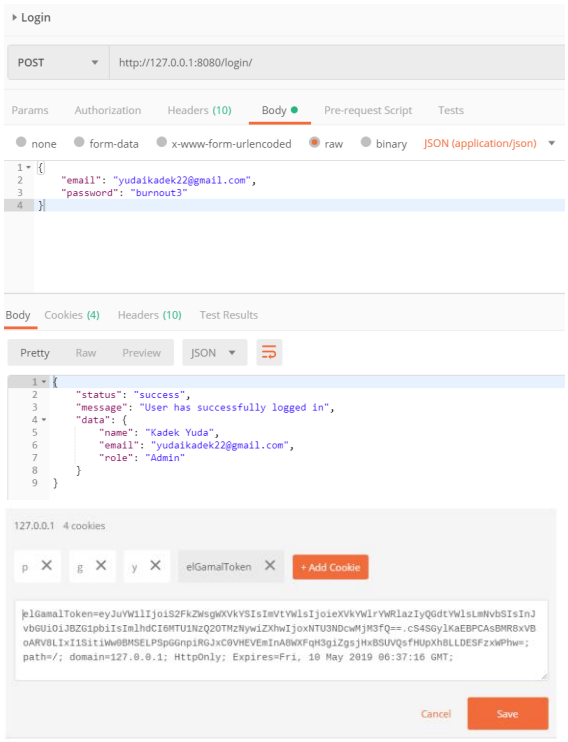


Gambar 5. Grafik Hasil Pengujian Token ElGamal

B. *Pengujian token pada Web Service*

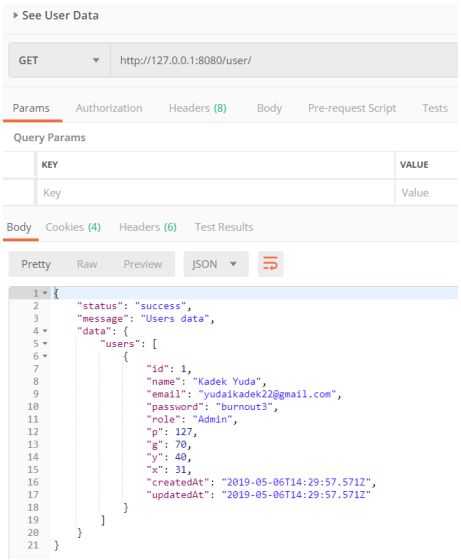
Untuk mengetahui apakah *token* dapat digunakan pada sebuah *web service*, token akan digunakan dalam melakukan autentikasi pengguna. Token akan dibuat pada saat pengguna login dan akan kedarluwasa dalam waktu 15 menit. Isi dari *payload* adalah data dari pengguna itu sendiri. Selain itu, pengujian akan dilakukan dengan mengakses

suatu halaman API *web server* dengan token yang benar dan dengan token yang salah.



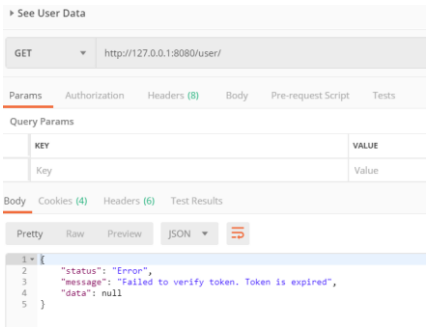
Gambar 6. Ilustrasi Login dan Mendapat Token

Setelah mendapatkan *token*, pengguna dapat mengakses halaman ini, karena pengguna sudah memiliki *token*.



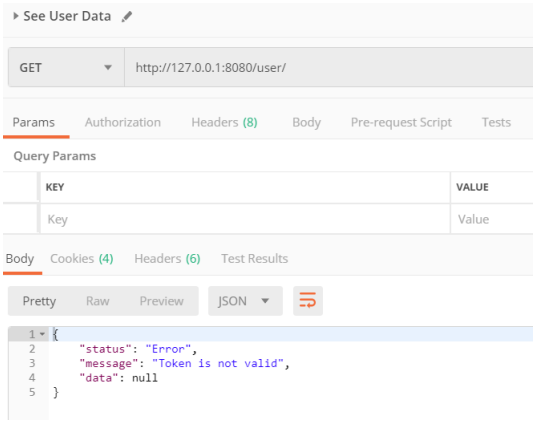
Gambar 7. Pengguna berhasil mengakses data dengan token yang telah diperoleh dari login.

Apabila pengguna sudah masuk terlalu lama, maka *token* akan kedarluwasa dan memberikan pesan kesalahan sebagai berikut.



Gambar 8. Token expired

Apabila kita mengganti nilai pada *payload* token, maka token akan menjadi tidak valid dan menghasilkan pesan seperti ini.



Gambar 9. Token Tidak Valid

Berdasarkan percobaan di atas, *token* berhasil berperan sebagaimana seharusnya, karena *token* dapat mengenali pengguna berdasarkan tokennya dan pengguna tidak dapat mengubah *token* untuk masuk secara illegal.

V. KESIMPULAN

Algoritma ElGamal dapat diimplementasikan ke dalam sebuah *token*. Akan tetapi, *token* yang diimplementasikan pada makalah ini masih memiliki kekurangan, yaitu ukurannya yang terlalu besar.

ACKNOWLEDGMENT

Pertama, penulis ingin mengucap syukur kepada Tuhan karena atas bantuan, rahmat, dan berkat dari Tuhan, makalah ini tidak mungkin dapat selesai. Penulis juga ingin berterima kasih kepada Dr. Ir. Rinaldi Munir sebagai Dosen Mata Kuliah IF4020 Kriptografi.

REFERENCES

[1] "Perbedaan antara otentikasi (authentication) dan otorisasi (authorization)", Internet: https://www.globhy.com/read-blog/33_perbedaan-antara-otentikasi-authentication-dan-otorisasi-authorization.html [10 Mei 2019]

[2] "Public Key Cryptography", Internet: https://www.ibm.com/support/knowledgecenter/SSYKE2_8.0.0/com.ibm.java.security.component.80.doc/security-component/jsse2Docs/publickeycryptography.html [10 Mei 2019]

[3] "Session vs Token Based Authentication – Sherry Hsu – Medium", Internet: https://medium.com/@sherryhsu/session-vs-token-based-authentication-11a6c5ac45e4 [10 Mei 2019]

[4] "Token Based Authentication using Postman as Client and Web API 2 as Server", Internet:

https://www.codeproject.com/Articles/1090252/Token-Based-Authentication-using-Postman-as-Client [10 Mei 2019]

[5] "Digital Signatures: ElGamal Signature Scheme and Digital Signature Algorithm (and Birthday Attacks)" Internet: https://www.commonlounge.com/discussion/35a1c2baa00b447f9275e8f71b02ef29 [10 Mei 2019]

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, 10 Mei 2019

TTD

I Kadek Yuda Budipratama Giri
13516115