Implementasi Algoritma HMAC dan SHA3 pada JWT untuk Autentikasi Web

Bonaventura Bagas Sukarno (18219017)
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): bonaventurabagas.bb@gmail.com

***Abstrak*—Internet yang terus berkembang turut berperan dalam mengembangkan industri digital. Industri digital ini tentunya tidak lepas dari pengembangan web untuk kebutuhan industri, seperti berbelanja, media hiburan, hingga pendidikan. Begitu luasnya pemanfaatan web mengharuskan adanya peningkatan keamanan web. Salah satu bentuk pengamanan web dapat dilakukan melalui pendekatan autentikasi berbasis token. Autentikasi ini pada dasarnya akan membangkitkan suatu token unik apabila pengguna dapat memasukkan credential yang valid. Salah satu format token yang banyak digunakan salah satunya adalah JSON Web Token (JWT) dengan default algoritma HMAC-SHA2. Dilain sisi pemanfaatan algoritma SHA3 masih belum banyak dimanfaatkan dalam pengamanan digital. Oleh karena itu, peneliti mengembangkan implementasi algoritma HMAC-SHA3 untuk pembangkitan JWT. Dari pengujian dapat disimpulkan bahwa algoritma HMAC-SHA3 dapat digunakan sebagai alternatif dalam pembangkitan dan verifikasi JWT.**

Kata Kunci—SHA3, JWT, Autentikasi Web, Web Token

I. PENDAHULUAN

Pada masa kini, pengembangan dan penggunaan *web* semakin meningkat. Berbagai kebutuhan mulai dari profil perusahaan, media belajar, hingga berbelanja dapat dilayani melalui *web* perusahaan terkait. Hal ini juga didukung oleh data dari laporan We Are Social yang menunjukkan peningkatan pengguna internet di Indonesia dalam kurun waktu 2018 – 2022[1]. Maka, tak dapat dipungkiri industri dari *web*, baik penyedia dan penggunanya, akan terus meningkat seiring dengan peningkatan pengguna internet.

Dengan semakin meningkatnya pengguna dan industri dari layanan berbasis *web* ini, maka risiko terkait tingkat keamanannya juga semakin rawan. Terdapat pihak – pihak yang akan mencari kelemahan dari suatu *web* dan memanfaatkan kelemahan tersebut untuk keuntungan pribadi. Oleh karena itu, seiring dengan perkembangan industri dan teknologi layanan berbasis *web* ini, harus diiringi dengan peningkatan mekanisme keamanan *web* untuk mengurangi dampak dari *cyber attack*.

Salah satu mekanisme yang digunakan dalam *web* untuk melindungi datanya adalah melalui proses autentikasi dan autorisasi. Autentikasi adalah proses untuk memverifikasi pengguna yang masuk ke dalam *web* berdasarkan identitas yang diberikan. Teknologi autentikasi yang banyak diterapkan adalah *login username password*, biometrik, dan lainnya. Sedangkan autorisasi adalah pengecekan wewenang atau hak akses yang dimiliki oleh pengguna. Dalam hal keamanan *web*, proses autentikasi memegang peran penting dalam mencegah pengguna yang tidak sah untuk mengakses data dalam *web*. Pada beberapa kasus *data breach* dapat terjadi akibat dari mekanisme autentikasi pengguna yang tidak aman, seperti pada kasus bocornya data pengguna Yahoo! dalam kurun waktu 2012 hingga 2016[2].

Autentikasi *web* berbasis token merupakan salah satu mekanisme autentikasi yang dapat diterapkan dalam pencegahan akses bagi pengguna yang tidak sah. Metode ini mengharuskan pengguna untuk memasukkan *credential* terkait sebagai dasar untuk membuat token (semacam kode rahasia pengguna). Token ini nantinya digunakan sebagai bukti untuk mengakses *web* yang terproteksi.

Berdasarkan masalah dan penjelasan di atas, akan dikembangkan implementasi mekanisme autentikasi berbasis *web* dalam format *JSON Web Token* menggunakan algoritma SHA-3.

II. METODE

Dalam mengembangkan algoritma SHA-3 pada format *JSON Web Token*, digunakan beberapa metode sebagai berikut.

A. Studi Literatur

Pada langkah awal, akan dilakukan studi literatur berkaitan dengan topik makalah, yaitu algoritma SHA-3 dan *JSON Web Token*. Dari bagian ini, akan dihasilkan mekanisme, struktur, dan informasi dari topik terkait sebagai dasar dalam pengembangan lebih lanjut.

B. Implementasi

Berdasarkan rancangan yang sudah dirumuskan, akan diimplementasi sebuah program untuk membuat JSON Web Token dengan algoritma SHA-3. Selain itu, juga dijelaskan lingkungan implementasi, seperti bahasa pemrogram, *library*, dsb. yang digunakan untuk mengembangkan program tersebut.

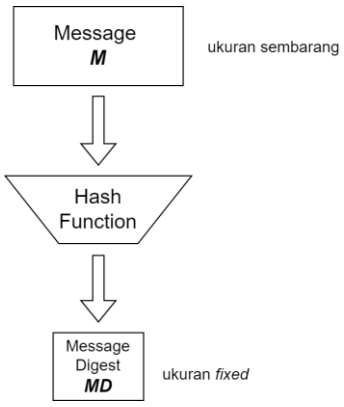
C. *Testing*

Metode terakhir yaitu pengujian dari program yang sudah dikembangkan. Pengujian dilakukan berdasarkan beberapa *test case*, serta akan dievaluasi hasilnya.

III. STUDI LITERATUR

A. *Hash Function*

Fungsi *hash* dalam kriptografi adalah algoritma matematis yang digunakan untuk mengkompresi data dengan ukuran sembarang (*message*) menjadi suatu *array* dengan ukuran statis (biasa disebut dengan *hash value* atau *message digest*) [3].



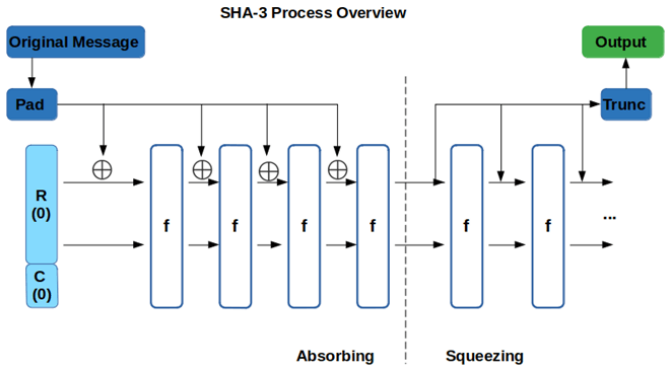
Gambar 1. Mekanisme Fungsi *Hash*

Fungsi ini bersifat *irreversible*, yaitu nilai *hash* tidak dapat dikembalikan ke bentuk awalnya atau *reverse* (sering juga disebut sebagai *one-way function*). Secara ideal, setiap *message* akan menghasilkan *message digest* yang berbeda sehingga berbagai jenis fungsi *hash* banyak digunakan pada implementasi keamanan kriptografi modern.

Fungsi *hash* ini dimanfaatkan untuk berbagai keperluan, seperti memverifikasi integritas dokumen atau *file*, membuat tanda tangan digital, verifikasi *password*, dan data *checksum*. Dalam mengimplementasikan fungsi *hash*, dapat digunakan salah satu jenis fungsi *hash* yang tersedia, yaitu MD-5, SHA-1, SHA-2, SHA-3, RIPEMD, dan lainnya. Sayangnya, tidak semua fungsi *hash* tersebut aman untuk digunakan karena sudah ditemukan kolisinya, seperti MD5 dan RIPEMD. Kolisi merupakan kondisi ketika terdapat dua *message* berbeda yang memiliki nilai *hash* yang sama. Hal ini tentunya sangat tingkat keaman informasi ketika diimplementasi.

B. *SHA3*

SHA-3 (*Secure Hash Algorithm* 3) atau Keccak adalah salah satu bentuk dari fungsi *hash* yang merupakan komplementar dari fungsi SHA-1 dan SHA-2. Algoritma ini didesain oleh Guido Bertoni, Joan Daemen, Michaël Peeters, dan Gilles Van Assche yang kemudian dirilis oleh NIST (*National Institute of Standards and Technology*) pada tanggal 5 Agusuts 2015.



Gambar 2. Proses *Hash* pada SHA-3
Sumber: [5]

SHA-3 bekerja dengan konstruksi *sponge* yang panjang *message digest* (*output length*) dapat disesuaikan. Proses *hash* pada suatu *message* diawali dengan proses *padding message* menyesuaikan dengan *output length* yang diinginkan. Proses selanjutnya adalah membagi *message* yang sudah di-*padding* menjadi berukuran r-bit [4]. Contoh panduan *output length* standar dan nilai r dapat dilihat pada tabel berikut.

TABEL I. STANDAR TIPE SHA-3

Tipe	*Output Length*	*Rate* (r)	*Capacity* (c)
SHA3-224	224	1152	448
SHA3-256	256	1088	512
SHA3-384	384	832	768
SHA3-512	512	576	1024

Sumber: [6]

Perlu diperhatikan bahwa penjumlahan nilai r+c harus sama dengan 1600 bit. *State* awal dari blok berukuran r+c bit akan diset menjadi 0 yang dilanjutkan pada dua fase utama yaitu fase penyerapan (*absorbing*) dan fase pemerasan (*squeezing*) [6]. Fase penyerapan (*absorbing*) bekerja dengan memasukkan setiap masukan blok *message* yang kemudian dikelola melalui fungsi permutasi. Pada fase pemerasan, dihasilkan *message digest* yang berukuran sesuai dengan *output length* yang ditentukan di awal [7].

C. *HMAC*

HMAC atau *Hash-based Message Authentication Codes* adalah salah satu tipe dari MAC (*Message Authentication Code*) yang memanfaatkan algoritma *hash*, seperti MD5 dan SHA serta sebuah *secret key*. Dengan menggunakan HMAC, fungsi autentikasi dan integritas juga dapat dicapai menggukan *secret key*, berbeda dengan pendekatan tanda tangan digital menggunakan kriptografi kunci publik [8].

D. *Autentikasi Berbasis Token*

Autentikasi berbasis token adalah suatu tipe mekanisme autentikasi yang digunakan dalam *web* dengan membangkitkan suatu *access token* yang unik, apabila identitas pengguna dapat diverifikasi. Setiap *access token* memiliki durasi aktif, selama dalam durasi tersebut pengguna dapat mengakses *protected*

web dengan meng-*attach* token tersebut tanpa harus melakukan *login* kembali [9].

Penggunaan token dalam metode autentikasi memberikan beberapa keuntungan. Keuntungan yang pertama adalah token bersifat *stateless* sehingga tidak ada informasi pengguna yang disimpan. Selain itu, token memiliki durasi aktif. Hal ini berdampak ketika pengguna melakukan *log out*, maka token akan dihancurkan untuk mengurasi risiko *cyberattack*. Keuntungan yang terakhir yaitu token dibangkitkan secara unik menggunakan algoritma kriptografi yang aman.

Secara umum, menurut [10] terdapat lima proses utama dalam autentikasi berbasis token, yaitu:

- *Request*
  Proses *login* oleh pengguna dengan memasukkan *credential*, sekaligus mengirimkan *request* akses ke server.
- *Verification*
  Server melakukan verifikasi terhadap *credential* pengguna. Apabila pengguna valid, maka dilanjutkan ke proses selanjutnya.
- *Token submission*
  Server membangkitkan *authentication token* untuk pengguna yang dapat digunakan dalam durasi waktu tertentu.
- *Storage*
  Token dikirimkan kembali kepada pengguna yang kemudian disimpan untuk digunakan untuk mengakses *web* pada waktu berikutnya.
- *Expiration*
  Token akan tetap aktif sampai pengguna *logout* atau melewati durasi.

E. JSON Web Token (JWT)

JSON Web Token (JWT) adalah *open standard* (RFC 7519) yang menjelaskan mekanisme transmisi informasi yang ringkas antara dua pihak menggunakan JSON *object*. Informasi ini dapat diverifikasi karena ditandatangani secara digital menggunakan *secret key* (algoritma HMAC) atau *public/private key* (algoritma RSA atau ECDSA) [11]. Selain itu, JWT dapat ditransmisikan melalui *query string*, *header*, dan *body* dari POST *request* karena ukurannya yang ringkas.

JWT terdiri dari tiga bagian yang dipisahkan oleh titik (.), yaitu *header*, *payload*, dan *signature*. Oleh karena itu, format JWT secara umum tampak seperti berikut.

```
xxxxx.yyyyyy.zzzzzz
```

Setiap bagian tersebut di-*encode* kedalam format base64. Sedangkan membangkitkan token dalam format JWT, diperlukan input dari tiap bagian yang ditunjukkan pada tabel 2.

TABEL II. CONTOH INPUT JWT

HEADER

{
"alg": "HS256",
"typ": "JWT"
}

PAYLOAD

{
"sub": "1234567890",
"name": "John Doe",
"admin": true
}

SIGNATURE
HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)

Jika disimpulkan, untuk membangkitkan token, digunakan formula seperti pada gambar 3.

$$\textbf{Token} = f(Base64Encode) \sum_{n=\alpha,\beta}^{\infty} (header.payload.signature)$$

Gambar 3. Formula Pembangkitan JWT
Sumber: [12]

Bagian *header* biasanya berisi informasi mengenai tipe dari token dan algoritma yang digunakan untuk *signing*. Kemudian bagian *payload* terdiri dari data pengguna secara umum dan informasi tambahan lainnya. Sedangkan pada bagian *signature* berisi tanda tangan menggunakan *encoded header* (base64) dan *encoded payload* (base64), dan *secret key* menggunakan algoritma yang dispesifikasikan pada *header*. Berikut merupakan contoh final JWT.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.cThIIoDvwdueQB468K5xDc5633seEFoqwxjF_xSJyQQ
```

Gambar 4. Contoh JWT
Sumber: Dokumentasi penulis

IV. IMPLEMENTASI

A. Lingkungan Implementasi

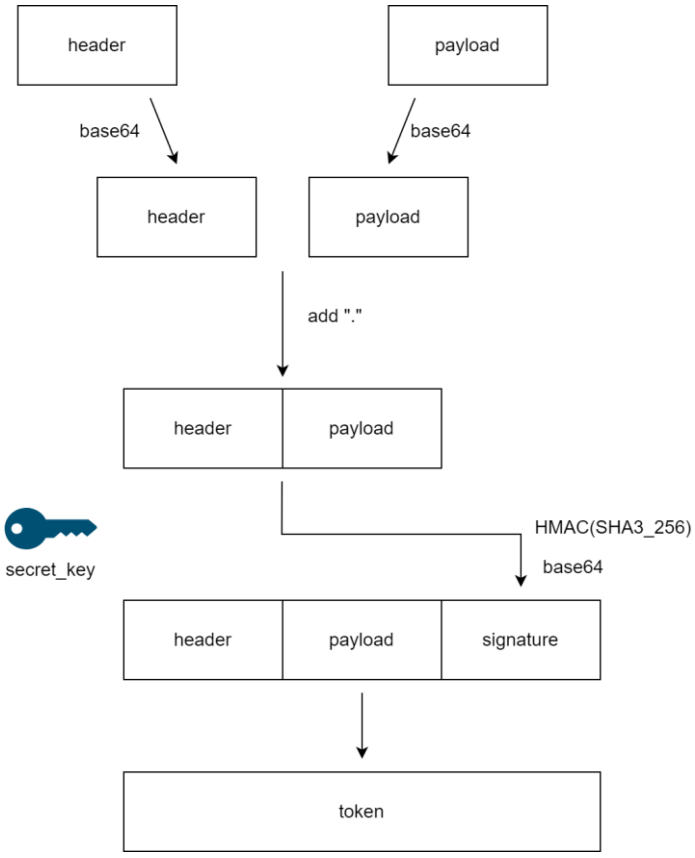
Implementasi pembangkitan dan verifikasi *JSON Web Token* dengan algoritma HMAC dan SHA3 dalam program menggunakan bahasa Python. Hal ini disebabkan oleh ketersediaan *library* pada Python yang cukup lengkap, mudah, dan cepat untuk mengembangkan program ini. Selain itu, algoritma SHA3 yang digunakan menerapkan *output length* sepanjang 256 bit (SHA3-256).

Berikut ini *library* yang digunakan dalam pengembangan program pembangkitan dan verifikasi JWT.

- hashlib
  Library untuk melakukan *hash* menggunakan algoritma SHA3 256 bit (SHA3_256).
- hmac
  Library yang berkaitan dengan fungsi HMAC.
- base64
  Library yang digunakan untuk mengonversi type data lain ke base64 dan sebaliknya.
- json
  Library yang digunakan untuk mengonversi tipe json ke tipe data lain dan sebaliknya.

B. *Pembangkitan Token*

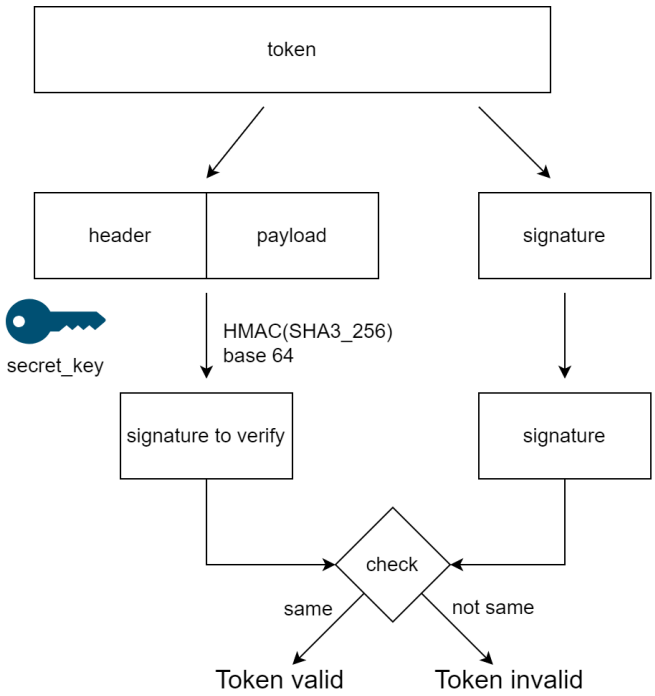
Langkah pembangkitan token dilakukan sesuai dengan diagram pada gambar 5.



Gambar 5. Langkah Pembangkitan Token
Sumber: Dokumentasi Penulis

C. *Verifikasi Token*

Langkah verifikasi token dilakukan sesuai dengan diagram pada gambar 6.



Gambar 6. Langkah Verifikasi Token
Sumber: Dokumentasi Penulis

V. TESTING DAN PEMBAHASAN

Pengujian terhadap program di atas dilakukan terhadap beberapa *testcase*. *Testing* dilakukan pada fungsi verifikasi token untuk mempermudah proses pengujian dan menyamakan input pada fungsi pembangkitan token.

Berikut merupakan input *header*, *payload*, *secret key* pada *pengujian*.

```
{
     "alg": "HMAC-SHA3_256",
     "typ": "JWT"
}
```

Gambar 7. Input *Header*
Sumber: Dokumentasi Penulis

```
{
     "nim": "18219017",
     "name": "Bonaventura Bagas",
     "mk": "Kriptografi dan Koding"
}
```

Gambar 8. Input *Payload*
Sumber: Dokumentasi Penulis

```
SECRET_KEY =
"7bf804c2360ccfa1c9fd38cc70830679d6e13fb3f
```

```
98501c1a707b3962c8c1239df1f63d2e033b76b873
f4972da57ad5f9fbf2daf1896b982a4b8e4aba5e03
8aa"
```

Gambar 9. Input *Secret Key*
Sumber: Dokumentasi Penulis

Berikut merupakan daftar kasus uji.

TABEL III. TEST CASE

Kasus Uji	Deskripsi	Hasil yang Diharapkan	Hasil Uji
Test Case-1	Token hasil fungsi pembangkitan token dimasukkan pada fungsi verifikasi token.	True	Berhasil
Test Case-2	Token hasil fungsi pembangkitan diubah, lalu dimasukkan pada fungsi verifikasi token	False	Berhasil

A. *Test Case-1*

Pengujian pada *testcase* ini merupakan pengujian apabila seluruh input valid. Setelah dilakukan pengujian, didapatkan bahwa keluaran dari fungsi verifikasi token adalah *True* yang menandakan pengujian berhasil.

```
Token:  eyJhbGciOiAiSE1BQy1TSEEzXzI1NiIsICJ0eXAiOiAiSldU
In0.eyJuaW0iOiAiMTgyMTkwMTciLCAibmFtZSI6ICJCb25hdmVudHVy
YSBCYWdhcyIsICJtayI6ICJLcmlwdG9ncmFmaSBkYW4gS29kaW5nIn0.
2ydFEtUavIr7ZEcTOFiCaU91OVfzF5_g3Ke9AdQ70nk
Hasil Verifikasi:  True
```

Gambar 10. Hasil *Test Case-1*
Sumber: Dokumentasi Penulis

B. *Test Case-2*

Pengujian pada *testcase* ini merupakan pengujian apabila seluruh terjadi modifikasi pada token yang diinput. Setelah dilakukan pengujian, didapatkan bahwa keluaran dari fungsi verifikasi token adalah *False* yang menandakan pengujian berhasil.

```
Token:  eyJhbGciOiAiSE1BQy1TSEEzXzI1NiIsICJ0eXAiOiAiSldU
In0.eyJuaW0iOiAiMTgyMTkwMTciLCAibmFtZSI6ICJCb25hdmVudHVy
YSBCYWdhcyIsICJtayI6ICJLcmlwdG9ncmFmaSBkYW4gS29kaW5nIn0.
2ydFEtUavIr7ZEcTOFiCaU91OVfzF5_g3Ke9AdQ70n
Hasil Verifikasi:  False
```

Gambar 11. Hasil *Test Case-2*
Sumber: Dokumentasi Penulis

VI. KESIMPULAN

Dari hasil implementasi dan *testing* yang dilakukan, dapat disimpulkan bahwa penggunaan algoritma HMAC-SHA3 dalam pembangkitan dan verifikasi *JSON Web Token* dapat dijadikan sebagai alternatif dalam penggunaan JWT untuk autentikasi *web*. Selain itu, tingkat keamanan yang diberikan oleh SHA-3 juga berada pada tingkatan yang berbeda karena menggunakan metode konstruksi *sponge* sehingga memerlukan metode kriptanalisis baru.

APPENDIKS

Kode Program

```
from base64 import urlsafe_b64encode,
urlsafe_b64decode
import hashlib
import hmac
import json

def base64UrlEncode(data):
  return
urlsafe_b64encode(data).rstrip(b'=').decode()

def base64UrlDecode(base64Url):
  base64Url = base64Url.encode()
  padding = b'=' * (4 - (len(base64Url) % 4))

  return urlsafe_b64decode(base64Url + padding)

class jsonwebtoken:
  header = {
  "alg": "HMAC-SHA3_256",
  "typ": "JWT"
  }

  def sign(payload, key, header=header,
algorithm=hashlib.sha3_256):
    encoded_header =
base64UrlEncode(json.dumps(header).encode())
    encoded_payload =
base64UrlEncode(json.dumps(payload).encode())
    signing_input =
encoded_header+"."+encoded_payload

    byte_signature = hmac.digest(
      key=key.encode(),
      msg=signing_input.encode(),
      digest=algorithm
    )
    encoded_signature =
base64UrlEncode(byte_signature)

    token =
encoded_header+"."+encoded_payload+"."+encoded_signa
ture
    return token


  def verify(token, key,
algorithm=hashlib.sha3_256):
    encoded_header = token.split(".")[0]
    encoded_payload = token.split(".")[1]
    encoded_signature = token.split(".")[2]
    signing_input =
encoded_header+"."+encoded_payload

    byte_verify_signature = hmac.digest(
      key=key.encode(),
      msg=(signing_input).encode(),
      digest=algorithm
    )

    encoded_verify_signature =
base64UrlEncode(byte_verify_signature)
    return
encoded_signature==encoded_verify_signature
```

DAFTAR PUSTAKA

[1] C. M. Annur, "Ada 204,7 Juta Pengguna Internet di indonesia Awal 2022: Databoks," *Databoks Pusat Data Ekonomi dan Bisnis Indonesia*, 23-Mar-2022. [Online]. Available: https://databoks.katadata.co.id/datapublish/2022/03/23/ada-2047-juta-pengguna-internet-di-indonesia-awal-2022. [Accessed: 23-May-2022].

[2] G. D. Maayan, "5 user authentication methods that can prevent the next breach," *ID R&D*, 06-Mar-2022. [Online]. Available: https://www.idrnd.ai/5-authentication-methods-that-can-prevent-the-next-breach/. [Accessed: 23-May-2022].

[3] R. Munir, "Fungsi Hash," [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2021-2022/17%20-%20Fungsi-hash-2021.pdf. [Accessed: 23-May-2022].

[4] R. Munir, "SHA-3 (Keccak)," [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2021-2022/20%20-%20SHA-3-2020 [Accessed: 23-May-2022].

[5] "Hash algorithm comparison: MD5, SHA-1, SHA-2 & sha-3," Code Signing Store, 23-Mar-2022. [Online]. Available: https://codesigningstore.com/hash-algorithm-comparison. [Accessed: 23-May-2022].

[6] A. Anand, "Breaking down : SHA-3 algorithm," *Medium*, 13-Jan-2020. [Online]. Available: https://infosecwriteups.com/breaking-down-sha-3-algorithm-70fe25e125b6. [Accessed: 23-May-2022].

[7] C. Paar and J. Pelzl, *SHA-3 and The Hash Function Keccak*, pp. 1–17.

[8] "HMAC (Hash-based Message Authentication Codes) Definition," *Okta*. [Online]. Available: https://www.okta.com/identity-101/hmac/. [Accessed: 23-May-2022].

[9] "What is token-based authentication?," *Okta*. [Online]. Available: https://www.okta.com/identity-101/what-is-token-based-authentication/. [Accessed: 23-May-2022].

[10] "What is an authentication token?," *Fortinet*. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/authentication-token [Accessed: 23-May-2022].

[11] auth0.com, "JSON web tokens introduction," *JSON Web Token Introduction*. [Online]. Available: https://jwt.io/introduction. [Accessed: 23-May-2022].

[12] R. Gunawan and A. Rahmatulloh, "JSON Web Token (JWT) untuk Authentication pada Interoperabilitas Arsitektur berbasis RESTful Web Service," *JEPIN (Jurnal Edukasi dan Peneilitan Informatika)*, vol. 5, no. 1, Apr. 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, 25 Mei 2022

Bonaventura Bagas Sukarno
18219017