Penerapan Fungsi Hash SHA-3 untuk Verifikasi Integritas Data Urutan DNA pada Platform Sequencing Cloud

Rici Trisna Putra - 13522026
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13522026@std.stei.itb.ac.id, ricitrisnap@gmail.com

***Abstract*—Makalah ini membahas implementasi sistem verifikasi integritas data hasil sekuensing DNA dalam format FASTQ yang disimpan di platform cloud simulasi. Sistem menggunakan fungsi hash kriptografis SHA-3 (*Keccak*) untuk menghasilkan hash dari data DNA, yang kemudian disimpan secara terpisah di basis data PostgreSQL. Arsitektur three-tier (API server, basis data hash, dan penyimpanan objek) dirancang untuk memastikan keaslian dan keutuhan data, mendeteksi perubahan sekecil apa pun pada data DNA, sehingga cocok untuk aplikasi sensitif seperti diagnosa medis dan penelitian penyakit.**

***Keywords*—FASTQ; SHA-3; DNA; Cloud Storage;**

I. Introduction

Perkembangan teknologi sekuensing DNA telah memungkinkan revolusi di bidang genomik yang memungkinkan kita untuk melakukan proses sekuensing ini dalam tempo yang singkat dan dengan akurasi sekuensing yang sangat tinggi. Data sekuensing bervolume besar yang biasa disimpan dalam format FASTQ atau FASTA ini dapat mencapai ukuran terabyte untuk proyek sekuensing genom yang lengkap. Data hasil sekuensing DNA ini kemudian harus disimpan di suatu penyimpanan yang dapat menampung data dengan volume yang sangat besar.

Seiring dengan perkembangan teknologi komputasi awan (*cloud*) dan adopsinya di berbagai domain seperti bioinformatika, platform *cloud* seperti Amazon Web Service (AWS) Genomics dan Google Cloud Life Sciences menjadi suatu alternatif penyimpanan dan pemroses data genomik hasil sekuensing seperti data DNA.

Namun hal ini dapat menimbulkan masalah karena terdapat tahap migrasi data yang bervolume besar dan sangat sensitif apabila terjadi kesalahan. Hal ini dikarenakan sifat DNA yang bahkan perubahan satu nukleotidanya saja dapat mengakibatkan konsekuensi yang sangat besar pada tahap pemrosesan selanjutnya seperti untuk melakukan diagnosa medis dan penelitian suatu penyakit tertentu. Oleh karena itu diperlukan suatu mekanisme yang dapat memastikan bahwa data hasil sekuensing DNA yang dipindah ke platform *cloud* tidak mengalami perubahan dan dapat diverifikasi keaslian dan keutuhannya.

II. Dasar Teori

A. Kriptografi Fungsi Hash

Fungsi *hash* adalah fungsi yang dapat melakukan kompresi terhadap suatu pesan M yang memiliki ukuran sembarang menjadi suatu string h yang memiliki ukuran tetap (*fixed*). Luaran dari fungsi *hash* ini sering disebut sebagai *message-digest* atau *hash value*.

Suatu fungsi harus memiliki beberapa

karakteristik yang harus terpenuhi untuk dapat dikatakan sebagai sebuah fungsi *hash* yang proper, yakni sebagai berikut:

1. Irreversible

Fungsi *hash* harus bersifat *irreversible* atau tidak dapat dikembalikan menjadi bentuk pesan semula. Hal ini lah yang membedakan fungsi *hash* dengan fungsi enkripsi yang bekerja dua arah dengan fungsi dekripsi dan sebuah kunci k untuk dapat mengembalikan pesan semula dari pesan cipher yang telah dikonstruksi.

2. Deterministik

Untuk suatu input yang sama maka output yang dihasilkan harus sama pula meskipun dijalankan berkali-kali.

3. *Avalanche Effect*

Perubahan kecil pada input harus berdampak besar dan drastis pada output *hash* sehingga terlihat seperti string acak.

4. *Collision Resistance*

Untuk suatu input a dan b maka fungsi *hash* H harus dibuat sedemikian rupa sehingga mencari H(a) = H(b) menjadi sangat sulit.

5. *Preimage Resistance*

Untuk suatu input y maka fungsi *hash* harus dibuat sedemikan rupa agar untuk menemukan input a yang bersifat H(a) = y sangat sulit.

6. *Second Preimage Resistance*

Untuk suatu input a dan output y = H(a) harus dibuat sulit untuk menemukan input b sehingga output fungsi *hash* dengan b juga menghasilkan y, y = H(b)

Fungsi *hash* memiliki beberapa kegunaan penting di dunia kriptografi praktis seperti menjaga integritas pesan (yang dicoba diuji di makalah ini), penghematan waktu verifikasi suatu *file*, dan untuk menormalkan panjang data beraneka ragam yang harus disimpan dengan panjang tertentu yang sudah ditentukan dan *fixed.*

Kolisi (*collision*) adalah suatu kondisi yang harus dihindari untuk suatu fungsi *hash* karena apabila terjadi kolisi maka terdapat dua string sembarang yang berbeda yang memiliki nilai *hash* yang sama. Hal ini menunjukkan bahwa fungsi *hash* tidak aman secara kriptografis.

B. SHA-3 (Secure Hash Algorithm 3)

SHA-3 merupakan standar hash terbaru dan diumumkan oleh NIST pada tahun 2015. SHA-3 ini dikembangkan melalui sebuah kompetisi publik terbuka selama lima tahun lamanya yakni dari 2007 hingga 2012. Algoritma pemenang dari semua kandidat yang mengikuti kompetisi adalah algoritma Keccak yang dikembangkan oleh Guido Bertoni, Joan Daemen, Michaël Peeters, dan Gilles Van Assche.

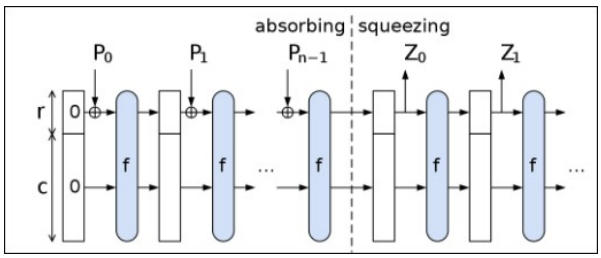
Algoritma *Keccak* ini menggunakan sebuah konstruksi baru yang berbeda dari semua algoritma yang menjadi kandidat kompetisi yakni kontruksi spons (*sponge construction*) yang merupakan fungsi yang bersifat non-kompresi dan memiliki dua fase yakni:

1. Fase Penyerapan (*Absorbing Phase*)

Pada fase ini data input akan dipecah pecah menjadi blok dan untuk setiap blok r-bit akan di XOR-kan dengan state S kemudian hasilnya dimasukkan ke dalam fungsi permutasi f untuk menghasilkan *state* terbaru. Tahap ini akan dilakukan hingga semua blok telah di proses

2. Fase Pemerasan (*Squeezing Phase*)

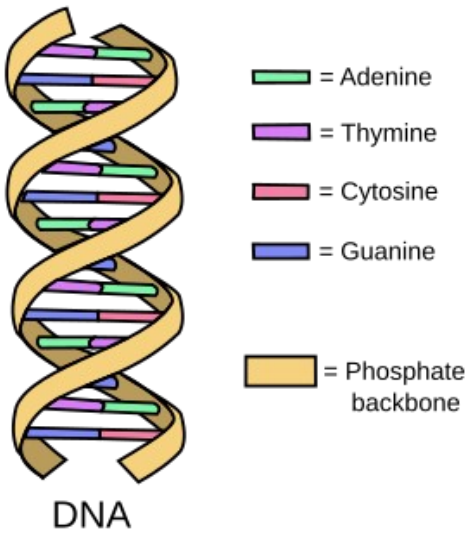
Digest akan disimpan dalam Z kemudian selagi panjang Z belum sama dengan target panjang *hash* maka r-bit pertama dari state S akan disambung dengan Z dan dilakukan fungsi permutasi f untuk menghasilkan state baru. Hal ini dilakukan hingga panjang *digest* menjadi d.



Gambar 1. Proses *Sponge Construction* algoritma *Keccak*

C. DNA

Deoxyribonucleic acid atau yang biasa disingkat DNA merupakan suatu asam nukleat yang terdiri dari susunan rantai nukleotida. DNA berbentuk helix ganda melingkar seperti terlihat pada gambar di bawah. Bentuk tersebut mengakibatkan DNA memiliki dua rantai polinukleotida yang saling terhubung oleh ikatan hidrogen dari basa nitrogen yang ada di tiap nukleotidanya yang melekat pada suatu kerangka fosfat. Setiap nukleotida yang ada di DNA terdiri dari dari salah satu jenis basa nitrogen yakni Adenin(A), Timina(T), Sitosin(C), atau Guanin(G). Rantai Polinukleotida yang ada pada DNA saling terhubung karena basa nitrogen Adenine berikatan hidrogen dengan Thymine sedangkan basa nitrogen Cytosine berikatan hidrogen dengan Guanine. Oleh sebab itu dua rantai polinukleotida pada DNA dikatakan sebagai saling komplemen (Complementary). Tentunya informasi dari salah satu rantai DNA cukup untuk membuat rantai lainnya sehingga dua rantai ini bisa dibilang membawa informasi yang sama (dalam bentuk komplemen)



Gambar 2.2 Struktur DNA
Sumber :
https://id.wikipedia.org/wiki/Berkas:DNA_simple2.svg

D. Pengurutan DNA (DNA *Sequencing*)

Merupakan proses atau teknik tertentu yang digunakan untuk menentukan urutan basa DNA yang ada pada molekul DNA tertentu. Hasil urutan yang dihasilkan oleh teknik ini sering disebut sebagai sekuens DNA yang merupakan informasi mendasar akan gen dan genom apa saja yang menyusun suatu molekul DNA dari sebuah atau sekumpulan organisme tertentu. Sekuens DNA ini sangat penting untuk dapat menentukan bagaimana pertumbuhan makhluk hidup melalui gen dan genom yang ada di DNA, seperti apa makhluk hidup tersebut berfungsi secara umum dari pembentukan hingga makhluk tersebut mati. Pengurutan DNA tentunya banyak digunakan pada bidang seperti Bioteknologi, Biologi, Kedokteran, Forensik dan Antropologi.

Teknologi pengurutan DNA dapat dikategorikan ke dalam tiga generasi tergantung pada teknik mendasar yang digunakan. Pengurutan DNA pada kategori generasi pertama melakukan pengurutan dengan melakukan amplifikasi (duplikasi) pada template DNA dan kemudian dilakukan proses *gel electrophoresis*. Adapun teknik yang masuk ke dalam kategori ini adalah Maxam-Gilbert dan Sanger. Untuk generasi kedua ditandai dengan metodenya yang dapat melakukan pengurutan secara cepat seperti metode Roche 454, Illumina, Solid, dan Ion Torrent. Metode generasi ketiga tidak hanya unggu dalam aspek kecepatan, tapi juga dalam biaya dan jumlah error dalam pengurutan. Metode ini berbeda dari dua generasi sebelumnya karena bisa melakukan pengurutan tanpa harus melakukan amplifikasi dan bersifat *real time* tanpa harus memfragmentasi DNA. Metode dalam kategori ini contohnya Pacific Bioscience dan Oxford Nanopore Technology.

E. Format FASTQ

Merupakan format representasi digital dari sekuens DNA yang memiliki *read identifier* dan indikasi kualitas bacaan per basa

nukleotida selain memiliki sekuens basa nukleotidanya. Format FASTQ merepresentasikan semua informasinya dalam bentuk data teks. Berikut merupakan definisi dan detail teknis bagian yang ada di format FASTQ:

- *Identifier* dan informasi lain yang merupakan data teks yang di terminasi oleh *white space*.
- Basa Nukleotida yang merupakan isi basa nukleotida yang ditulis dengan standar basa (ATCG) dan dapat bervariasi pada panjang
- *Separator* yang biasanya berupa tanda (+)
- Kualitas ditulis dengan beberapa opsi seperti *decimal encoding* atau Phred-33 ASCII atau Phred-64 ASCII dengan panjang sama persis dengan jumlah basa Nukleotida. Hal ini dikarenakan kualitas ini mengindikasikan kualitas hasil baca tiap basa nukleotida pada hasil bacaan sekuens DNA.

Pada umumnya format yang umum adalah sebagai berikut

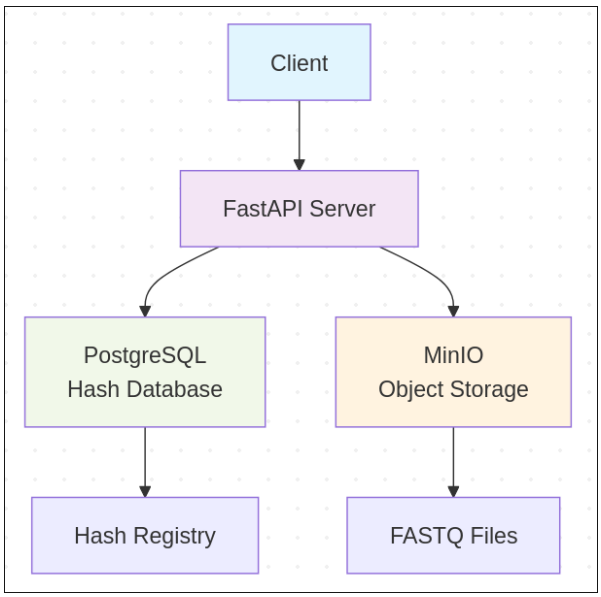
```
@<identifier and expected information>
<sequence>
+<identifier and other information OR empty string>
<quality>
```

Format FASTQ memiliki keunggulan karena menyimpan informasi kualitas baca pada tiap basa nukleotidanya dibandingkan format seperti FASTA yang hanya menyimpan basa nukleotida saja. Format FASTQ juga hanya bisa dihasilkan oleh teknik pengurutan DNA pada generasi kedua atau lebih karena membutuhkan hasil kualitas bacaan yang tidak dapat dihasilkan oleh generasi pertama. Format ini digunakan pada tugas kali ini karena merupakan format yang digunakan untuk menyimpan data SRA yang ada pada website NCBI.

III. Implementasi dan Pembahasan

3.1. Arsitektur Sistem

Sistem verifikasi integritas data DNA sekuens dikembangkan dengan arsitektur *three-tier* yang terdiri dari *client-server* dan penyimpanan terpisah. Arsitektur ini dirancang untuk memastikan keamanan dengan memisahkan penyimpanan *hash* dari penyimpanan file. Gambar dibawah ini menunjukkan detail diagram arsitektur sistem.



Sistem implementasi terdiri dari tiga komponen utama yakni:

1. **API Server (FastAPI)**:
Menyediakan endpoint RESTful untuk operasi *upload* dan verifikasi
2. **Hash Database (PostgreSQL)**:
Menyimpan *hash* SHA-3 secara terpisah untuk keamanan
3. **Object Storage (MinIO)**:
Menyimpan file FASTQ tanpa *hash* dalam metadata

3.2. Implementasi Fungsi Hash SHA-3

Implementasi algoritma SHA-3 dilakukan melalui kelas `SHA3DNAHasher` yang menyediakan empat variasi ukuran hash: 224, 256, 384, dan 512 bit dengan defaultnya adalah 256 bit karena alasan keseimbangan komputasi dan keamanan. Sistem *hashing*

yang diimplementasikan mendukung dua metode *hashing* yang berbeda yakni *whole-file hashing* yang melakukan *hash* langsung pada file FASTQ dan menghasilkan satu nilai *hash*. Metode ini diimplementasikan untuk melakukan verifikasi cepat dan efisien pada dataset yang berukuran besar. Metode lain yang diimplementasikan untuk *hashing* adalah *hash* per *read* yang menghasilkan satu *hash* untuk tiap *read* unit yang ada di FASTQ format. Metode ini diimplementasikan untuk memberikan granularitas lebih tinggi sehingga memastikan integritas file dengan lebih baik dan dapat melakukan deteksi *error* apabila ditemukan ada perubahan pada file original.

3.3. Sistem Penyimpanan Hash dengan PostgreSQL

Merupakan sistem penyimpanan hasil *hash* sebagai *ground truth* hasil *hash* yang valid untuk sekuens DNA dan sistem basis datanya dirancang untuk mencegah manipulasi data. Tabel utama `file_registry` menyimpan metadata file dan hash SHA-3. Setiap entri dalam tabel ini memiliki UUID sebagai *primary key*, memberikan identifikasi unik yang tidak dapat ditebak. Skema database juga mencakup tabel `read_hashes` untuk penyimpanan hash per-*read* dan tabel `verification_logs` yang berfungsi sebagai audit trail untuk semua operasi verifikasi.

3.4. Sistem Simulasi Penyimpanan Objek S3 dengan MinIO

MinIO diimplementasikan sebagai *object storage* dari file sekuens DNA dalam format FASTQ untuk mensimulasikan *cloud storage* S3 AWS yang memang MinIO sudah sepenuhnya kompatibel. sistem ini secara sengaja tidak menyertakan hash dalam metadata MinIO. Sebagai gantinya, metadata hanya berisi informasi non-kritis seperti `file_id` dan `original_filename`, sementara hash disimpan secara terpisah di PostgreSQL. Desain ini dipilih untuk diimplementasikan karena secara efektif memisahkan *data at rest* dari *verification data*, menciptakan tembok keamanan tambahan ke dalam sistem.

3.5. REST API Design dan Autentikasi

API yang digunakan pada sistem diimplementasikan secara sederhana menggunakan FastAPI framework yang bekerja dengan bahasa Python Desain endpoint mengikuti prinsip RESTful dengan struktur *resource-oriented* URL. Endpoint utama terdiri dari `/secure/upload` untuk upload file, `/secure/verify/{file_id}` untuk verifikasi integritas, dan `/secure/search` untuk pencarian berdasarkan metadata. Setiap endpoint mengembalikan response dalam format JSON yang konsisten dengan field `status`, `data`, dan `error` jika dapat dilakukan. Sistem autentikasi mengimplementasikan JSON Web Tokens (JWT) dengan mekanisme login melalui endpoint `/auth/login`. Setelah verifikasi kredensial, server mengeluarkan *access token* yang bisa digunakan untuk *request* yang akan dilakukan ke depannya.

3.6. Containerization dengan Docker

Sistem dikemas sepenuhnya menggunakan Docker *container* untuk memastikan konsistensi lingkungan antara pengembangan, pengetesan, dan *deployment*. Docker Compose digunakan untuk *orchestration multi-container* dengan konfigurasi yang mendefinisikan ketergantungan dan jaringan antar servis. Setiap servis dilengkapi dengan *health check* yang memonitor status service dan otomatis melakukan *restart* pada saat terjadi *failure*

3.7. Test

Berikut merupakan skenario uji terhadap sistem yang telah dibahas dalam

3.7.1 Proses Registrasi

```
[nekora@archlinux dna-hash-verifier]$ curl -s -X POST
http://localhost:8000/auth/register   -H "Content-Type
: application/json"   -d '{"username": "kripto", "pass
word": "kriptojaya"}' | python -m json.tool
{
    "message": "User registered successfully",
    "username": "kripto"
}
```

Berhasil didaftarkan seorang pengguna dengan username “kripto” dan password “kriptojaya”.

3.7.2 Proses Login

```
[nekora@archlinux dna-hash-verifier]$ curl -s -X POST
http://localhost:8000/auth/login   -H "Content-Type: a
pplication/json"   -d '{"username": "kripto", "passwor
d": "kriptojaya"}' | python -m json.tool
{
    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV
CJ9.eyJzdWIiOiJrcmlwdG8iLCJyb2xlIjoidXNlciIsImVtYWlsIj
oia3JpcHRvQGRuYS5sb2NhbCIsImV4cCI6MTc2NjY3MDM0M30.eurM
2o5sN2YMAQh7BOHJy45crvjTZvnU9Ns3CaW5cNI",
    "token_type": "bearer",
    "expires_in": 1800,
    "user": {
        "username": "kripto",
        "email": "kripto@dna.local",
        "role": "user"
    }
}
```

User dengan username “kripto” dan password “kriptojaya” berhasil login ke dalam sistem dengan informasi akun pribadinya

3.7.3 Informasi Akun

```
[nekora@archlinux dna-hash-verifier]$ curl -s -H "Auth
orization: Bearer $TOKEN" http://localhost:8000/auth/m
e | python -m json.tool
{
    "username": "kripto",
    "email": "kripto@dna.local",
    "role": "user"
}
```

Informasi akun yang memiliki Token ditampilkan.

3.7.4 Upload File FASTQ

```
[nekora@archlinux dna-hash-verifier]$ curl -s -X POST
http://localhost:8000/secure/upload \
  -H "Authorization: Bearer $TOKEN" \
  -F "file=@test.fastq" \
  -F "store_per_read=false" | python -m json.tool
{
    "status": "success",
    "file_id": "10bb0041-5e5a-4ab4-bc81-71d5273ef186",
    "verification_token": "v1:71GZDMUg0pnbc4WdxUjhFp13
+PvtuSeJ",
    "file_hash": "6c888e10e984e44f39dfbe9ea0087545f6c7
7735bb4b857c95d1318cafddc473",
    "file_size": 74304886,
    "storage": {
        "bucket": "dna-sequences",
        "key": "11890284-3fe6-4bf9-a5e6-6b618082f310/t
est.fastq"
    },
    "read_hashes_count": 0,
    "message": "File uploaded securely with hash store
d in PostgreSQL",
    "uploaded_by": "kripto"
}
```

Hasil upload file test.fastq ke cloud storage dengan pencatatan hash ke basis data berhasil

3.7.5 Verify File FASTQ

3.7.5.1 No Tampering

```
[nekora@archlinux dna-hash-verifier]$ curl -s -H "Auth
orization: Bearer $TOKEN" \
  "http://localhost:8000/secure/verify/10bb0041-5e5a-4
ab4-bc81-71d5273ef186" | python -m json.tool
{
    "file_id": "10bb0041-5e5a-4ab4-bc81-71d5273ef186",
    "stored_hash": "6c888e10e984e44f39dfbe9ea0087545f6
c77735bb4b857c95d1318cafddc473",
    "computed_hash": "6c888e10e984e44f39dfbe9ea0087545
f6c77735bb4b857c95d1318cafddc473",
    "is_valid": true,
    "verification_time": "2025-12-25T13:25:56.848405",
    "log_id": 2,
    "file_info": {
        "filename": "test.fastq",
        "size": 74304886,
        "uploaded_at": "2025-12-25T13:23:07.343182"
    },
    "read_hashes_count": 0,
    "verification_method": "postgresql_secure_storage"
,
    "verified_by": "kripto"
}
```

Bisa dilihat bahwa hasil yang asli dapat terverifikasi dengan jelas.

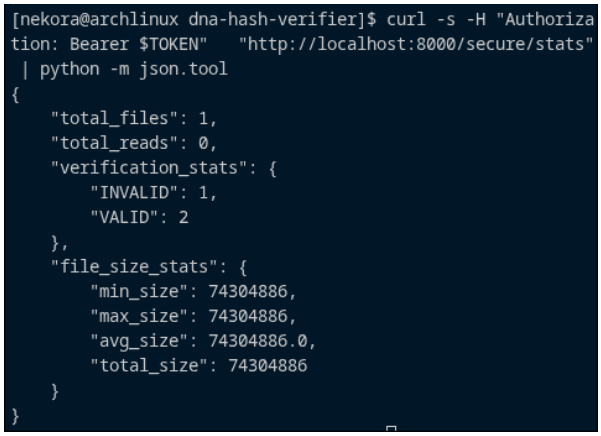
3.7.5.2 After Tampering

Bisa dilihat bahwa hasil yang sudah di tamper di bagian cloud untuk mensimulasikan kegagalan transfer atau serangan oleh *attacker* dapat terverifikasi dengan jelas bahwa statusnya tidak valid (hash dengan yang ada di basis data tidak sama).

3.7.6 Search

Hasil pencarian informasi file apa saja yang ada pada basis data dan bagaimana detailnya (*ground truth*)

3.7.7 Database Statistics



Hasil laporan statistik dari basis data.

3.7.8 File Verification History

Untuk keperluan audit, dapat dilihat berapa kali verifikasi telah dilakukan terhadap suatu file dan apa saja hasil yang didapat. Dari hasil di atas dapat disimpulkan file telah cek verifikasinya sebanyak 3 kali dengan 2 kali bersifat valid namun pada verifikasi terakhir file sudah tidak valid sehingga ada indikasi perubahan pada file di cloud storage.

IV. KESIMPULAN

Penggunaan ilmu kriptografi di berbagai bidang ternyata sangat penting untuk menjamin keamanan data. SHA-3 yang merupakan salah satu fungsi *hash* standar memiliki potensi yang baik dan mirip dengan SHA-256 dalam bidang penerapan *hashing* di domain bioinformatika.

V. UCAPAN TERIMA KASIH

Puji Syukur kepada Allah SWT yang telah memberikan Rahmat sehingga penulis dapat menyelesaikan makalah ini dengan tepat waktu yang tentunya keberhasilan ini tidak lepas dari kontribusi berbagai pihak.

Pertama-tama penulis mengucapkan terimakasih yang sebesar-besarnya kepada dosen pengampu Mata Kuliah IF4020 Kriptografi bapak Dr. Ir. Rinaldi Munir, M.T. Karena atas bimbingan beliau selama ini saya berhasil menyelesaikan mata kuliah ini dengan baik. Tentunya arahan dan ajaran beliau sangat memberikan dampak bagi diri saya untuk bisa lebih baik kedepannya.

Selain itu penulis juga berterima kasih kepada kedua orang tua penulis yang telah mendidik dan membesarkan penulis hingga sampai saat ini.

Terakhir penulis sadar bahwa terdapat berbagai kesalahan pada tulisan ini sehingga apabila terdapat kritik dan saran maka penulis sangat mengharapkan untuk dapat disampaikan.

REFERENCES

[1] K. Eren, N. Taktakoğlu, and I. Pirim, “DNA Sequencing Methods: From Past to Present,” *Eurasian J. Med.*, vol. 54, no. Suppl 1, pp. S47–S56, Dec. 2022, doi: 10.5152/eurasianjmed.2022.22280. Diakses 24 Desember 2025

[2] https://www.genome.gov/genetics-glossary/Deoxyribonucleic-Acid Diakses 25 Desember 2025.

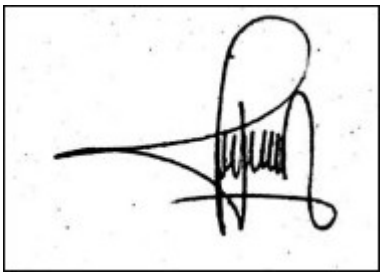
[3] https://www.ncbi.nlm.nih.gov/sra Diakses 25 Desember 2025.

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2025-2026/26-Fungsi-hash-2025.pdf Diakses 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



Rici Trisna Putra
13522026