IMPLEMENTASI DAN PERBANDINGAN *BLOCKCHAIN* DENGAN ALGORITMA HASH *KECCAK* DAN *HASH SKEIN*

AHMAD ALI HUSAIN

D121201097

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

MAKASSAR

2024

IMPLEMENTASI DAN PERBANDINGAN *BLOCKCHAIN* DENGAN ALGORITMA HASH *KECCAK* DAN *HASH SKEIN*

AHMAD ALI HUSAIN

D121201097

Skripsi

Sebagai salah satu syarat untuk mencapai gelar sarjana

Program Studi Teknik Informatika

pada

PROGRAM STUDI TEKNIK INFORMATIKA

DEPARTEMEN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

MAKASSAR

2024

SKRIPSI

IMPLEMENTASI DAN PERBANDINGAN *BLOCKCHAIN* DENGAN ALGORITMA HASH *KECCAK* DAN *HASH SKEIN*

AHMAD ALI HUSAIN
D121 20 1097

Skripsi,

Telah dipertahankan di depan Panitia Ujian Sarjana Pada tanggal 18 September 2024
dan dinyatakan telah memenuhi syarat kelulusan
Pada

Program Studi Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknik
Universitas Hasanuddin
Makassar

Mengesahkan:
Pembimbing tugas akhir,

rudi

912 1 003

Mengetahui:
Ketua Program Studi,

Prof. Dr. Ir. Indrabayu, S.T., M.T., M.Bus.Sys., IPM., Asean.Eng.
NIP 19750716 200212 1 004

PERNYATAAN KEASLIAN SKRIPSI DAN PELIMPAHAN HAK CIPTA

Dengan ini saya menyatakan bahwa, skripsi berjudul "Implementasi dan Perbandingan *Blockchain* dengan Algoritma *Hash Keccak* dan Hash Skein" adalah benar karya saya dengan arahan dari Dr. Eng. Ady Wahyudi Paundu, S.T., M.T. Karya ilmiah ini belum diajukan dan tidak sedang diajukan dalam bentuk apa pun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka skripsi ini. Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini adalah karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut berdasarkan aturan yang berlaku.

Dengan ini saya melimpahkan hak cipta (hak ekonomis) dari karya tulis saya berupa skripsi ini kepada Universitas Hasanuddin.

Makassar, 18 September 2024

AHMAD ALI HUSAIN
D121 20 1097

Ucapan Terima Kasih

Bismillahirrahmanirahim, Alhamdulillahi rabbil'alamin puji syukur kehadirat Allah SWT atas karunia_Nya sehingga penulisan skripsi ini dapat diselesaikan. Shalawat dan salam tercurah pada junjungan kita Rasullullah Muhammad SAW.

Penelitian ini dapat berjalan dengan baik dan skripsi ini dapat terselesaikan atas bimbingan, arahan dan diskusi dari pembimbing. Untuk itu dengan segala kerendahan hati penulis menyampaikan penghargaan dan terima kasih yang setulus-tulusnya kepada Bapak Dr. Eng. Ady Wahyudi Paundu, S.T., M.T. atas segala waktu dan tenaganya. Juga kepada Dosen Penguji yang telah memberikan masukan dan saran. Selanjutnya kepada:

1. Rektor Universitas Hasanuddin yang telah memberikan fasilitas dan kemudahan-kemudahan selama mengikuti proses studi.
2. Dekan Fakultas Teknik Universitas Hasanuddin yang telah memberikan arahan dan motivasi selama penulis menempuh studi.
3. Ketua Departemen Teknik Informatika atas segala dukungan, arahan, motivasi dan kemudahan yang diberikan.
4. Bapak dan Ibu dosen serta staf pada Fakultas Teknik Unhas atas bantuan dan pelayanan yang diberikan.

Kepada yang kami cintai dan hormati, Ayahanda Dr. H. Tachrir, ST., MT. dan Ibunda Dr. Hj. Nini Hasriyani Aswad, ST., MT., serta adikku tersayang Muhammad Zaky. Terima kasih atas doa dan dukungannya dan juga kasih sayang yang selama ini telah dicurahkan untukku.

Rekan-rekan mahasiswa, Kanda Irfandi, Ravi, Akram, Adit, Iman, ın, teman-teman dari Rezolver angkatan 2020, dan dekat Alzena Jayanthi yang senantiasa membantu dan memberikan masukan dan dorongan hingga selesainya

Guna penyempurnaan skripsi ini penulis masih dengan tulus mengharapkan kritikan dan saran kepada semua pihak. Semoga skripsi ini dapat bermanfaat bagi pihak yang berkepentingan serta dapat menambah khasanah keilmuan. Amin.

Gowa, Juli 2024

Ahmad Ali Husain

ABSTRAK

Ahmad Ali Husain, **Implementasi dan perbandingan** *Block**chain*** **dengan algoritma hash *Keccak* dan *Hash Skein*** (Dibimbing Oleh Ady Wahyudi P.)

Penelitian ini bertujuan untuk mengimplementasikan dan membandingkan kinerja *Blockchain* menggunakan dua algoritma *Hash* yang berbeda, yaitu *Hash Keccak* dan *Hash Skein*. *Blockchain* adalah teknologi yang mengamankan transaksi digital melalui mekanisme desentralisasi dan kriptografi. Algoritma *Hash* berperan penting dalam memastikan integritas dan keamanan data dalam *Block*. Dalam penelitian ini, dilakukan implementasi *Blockchain* dengan masing-masing algoritma *Hash*, pada tulisan ini dilakukan pengujian terhadap ukuran kinerja yakni kecepatan *Hash*ing dan konsumsi daya. Hasil pengujian menunjukkan bahwa terdapat perbedaan signifikan antara kedua algoritma dalam hal efisiensi. Algoritma *Hash Skein* menunjukkan performa yang lebih cepat dan efisien dalam proses *Hash*ing, daripada Algoritma *Hash Keccak*. Temuan ini memberikan pandangan yang lebih mendalam tentang keunggulan masing-masing algoritma dalam konteks penerapan pada teknologi *Blockchain*, sehingga memberikan rekomendasi penggunaan *Hash* berdasarkan kebutuhan spesifik dari aplikasi *Blockchain* dan transaksi cryptocurrency yang akan dikembangkan dikemudian hari.

Kata Kunci: *Blockchain*, *Hash Keccak*, *Hash Skein*, Kecepatan

ABSTRACT

Ahmad Ali Husain, **Implementation and comparison of *Block**chain*** **using the *Keccak* hash and skein hash algorithms** (Supervised by Ady Wahyudi P.)

The objective of this study is to apply and evaluate the efficiency of *Blockchain* by employing two distinct *Hash* algorithms, specifically *Hash Keccak* and *Hash Skein*. *Blockchain* is a cryptographic sistem that ensures the security of digital transactions by utilizing decentralized procedures. *Hash* algorithms are crucial for maintaining the integrity and security of data within *Block*s. This study involved the deployment of *Blockchain* using several *Hash* algorithms. This article examines performance measurements, including the speed of *Hash*ing and power usage. The test results indicate notable disparities in efficiency between the two methods. The *Hash Skein* method demonstrates superior speed and efficiency in the *Hash*ing process compared to the *Hash Keccak* algorithm. These findings offer a comprehensive perspective on the benefits of each algorithm when applied to *Blockchain* technology. As a result, they provide recommendations for utilizing *Hash*es based on the specific requirements of future *Blockchain* applications and cryptocurrency transactions.

Keywords: *Blockchain*, *Keccak Hash*, *Skein Hash*, Speed

DAFTAR ISI

DAFTAR TABEL

Nomor urut Halaman

DAFTAR GAMBAR

Nomor urut Halaman

DAFTAR LAMPIRAN

Nomor urut Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Secara umum, transaksi *peer-to-peer* melibatkan dua pihak yang melakukan bisnis secara langsung satu sama lain, tanpa keterlibatan pihak ketiga. Dengan menggunakan teknologi *Blockchain* untuk mengamankan data transaksi melalui proses *Hash*ing *Block* individu yang berisi data tersebut, adalah mungkin untuk menegakkan kerahasiaan dan integritas informasi pembelian. Integrasi teknologi *Blockchain* ke dalam sistem perbankan kontemporer akan menguntungkan karena kapasitasnya untuk mengatur cryptocurrency, sehingga memfasilitasi perang melawan pencucian uang dan pendanaan terorisme pada skala nasional dan internasional (A, S, Ulhas, & Sharif, 2019). *Blockchain*, sistem basis data terdesentralisasi, mendistribusikan semua data yang terkandung di dalamnya secara adil di seluruh mitra yang terhubung ke sistem *Blockchain*. Teknologi *Blockchain* telah mendapatkan pengakuan yang signifikan sebagai kemajuan revolusioner dalam bidang khusus ini (Luhkito, Kusyanti, & Siregar, 2021). Adopsi teknologi ini tersebar luas di berbagai sektor, termasuk transaksi keuangan dan real estat, dan memiliki pengaruh signifikan pada industri keuangan (Mbaidin, Alomari, AlMubydeen, & Sbaee, 2024). Untuk memfasilitasi kontrol hari depan, dukungan real-time, dan penjadwalan generasi sumber daya energi terdistribusi, perdagangan energi prediktif berbasis *Blockchain* menggunakan transaksi *peer-to-peer* (Jamil, Iqbal, Imran, Ahmad, & Kim,

kerja *Hyperledger Fabric* mendukung sifat *Blockchain* lisasi. Dengan memfasilitasi transaksi peer-to-peer, ıng diusulkan memfasilitasi pembentukan pasar energi

lokal untuk komunitas energi warga (CEC) (Mousavi, Ghazizadeh, & Vahidinasab, 2023).

Blockchain, database terdistribusi yang saling berhubungan dan saling bergantung, memungkinkan penyimpanan dan pencatatan data dalam *Block*, yang merupakan kumpulan informasi. Langkah selanjutnya melibatkan pembuatan koneksi antara *Block* yang berisi data dan *Block* sebelumnya dengan memasukkan kode *Hash* dari *Block* sebelumnya ke dalam *Block* yang baru terbentuk. Buku besar yang terdistribusi dan transparan, *Blockchain* mampu menyimpan kode dan teks yang tidak dapat diubah. *Block* yang terdiri dari urutan data ditautkan ke *Block* sebelumnya di *Blockchain*, dan setiap perubahan atau revisi yang diterapkan di *Block* sebelumnya dimanifestasikan dalam *Block* saat ini atau berikutnya (Mohamed & Faisal, 2024). *Block* individu dalam *Blockchain* adalah kompilasi data yang ditambahkan ke buku besar melalui koneksi berurutan dengan *Block* lain, sehingga membentuk rantai *Block* yang saling berhubungan. Secara umum, *Blockchain* mengacu pada buku besar terdistribusi yang aman, tidak berubah, dan dapat ditukar. Ini terdiri dari buku besar transaksi yang tidak dapat diubah (Goyal, Ahmed, & Gopalani, 2022; Purwaningsih, Muslikh, Suhaeri, & Basrowi, 2024).

Banyak fungsi *Hash* ada untuk tujuan menghasilkan kode *Hash* yang membuat koneksi antara dua *Block*. Saat ini, algoritma SHA-2 tetap layak untuk memfasilitasi pengoperasian teknologi *Blockchain* pada *Bitcoin*; Namun demikian, pertimbangan masa depan harus mencakup keamanan dan skalabilitas. Sebagai finalis dalam kompetisi penentuan SHA-3, telah memantapkan diri mereka sebagai pesaing untuk *Hash* terbaru. Implementasi potensial mereka dalam *ain* memerlukan pertimbangan. Tidak diragukan lagi, lgoritma ini memiliki kelebihan dan kekurangannya

sendiri. Dengan membandingkan berbagai nilai *Hash* yang dihasilkan dan operasi pemrosesan, dimungkinkan untuk menentukan mana dari kedua algoritma ini yang lebih unggul dalam memproses data *Blockchain*.

Sementara SHA256 mungkin memerlukan lebih banyak waktu komputasi dalam hitungan detik, ia menemukan lebih sedikit *Block* baru dibandingkan dengan algoritma Scrypt, yang mampu menemukan lebih banyak *Block* (Mulyadi & Setianto, 2021).

Beranjak dari telah adanya algoritma *Hash* yang baru, dan masih sedikit penelitian terkait implementasinya pada *Blockchain* maka dari itu penulis mengusulkan judul "Implementasi dan Perbandingan *Blockchain* dengan Algoritma *Hash Keccak* dan *Hash Skein*". Penelitian ini diharapkan dapat menjadi landasan kepada pengembang *Blockchain* dalam memilih algoritma *Hash* yang ingin diterapkan pada *Blockchain* yang ingin dikembangkan.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah pada penelitian ini adalah:

1. Bagaimana membangun sistem *Blockchain* untuk membandingkan algoritma *Hash Keccak* dengan algoritma *Hash Skein*?
2. Bagaimana membandingkan performa algoritma *Hash Keccak* dengan Algoritma *Hash Skein* pada *Blockchain*?

1.3 Tujuan Penelitian

mahami proses membangun sistem *Blockchain* yang
ngkan kinerja algoritma *Hash Keccak* dengan algoritma
in, sehingga dapat memberikan pemahaman yang

mendalam tentang perbedaan dan keunggulan masing-masing algoritma dalam sistem *Blockchain*.

2 Untuk mengevaluasi dan membandingkan performa implementasi *Blockchain* dengan algoritma *Hash Keccak* dan *Hash Skein*, dengan tujuan untuk mengetahui mana yang lebih efisien pada sistem *Blockchain*.

1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat menjadi referensi bagi pengembang *Blockchain* untuk menggunakan algoritma yang diteliti dalam pengembangan *Blockchain*. Mengukur kinerja algoritma *Hash*ing berguna karena dapat diimplementasikan secara efisien dan memberikan kecepatan tinggi. Algoritma *Hash*ing yang efisien meningkatkan kinerja *Blockchain* secara keseluruhan dan memungkinkan proses mining yang lebih cepat.

1.5 Batasan Masalah

Batasan masalah yang ditentukan dalam sistem ini adalah:

1 Uji coba dilakukan dengan model eksperimen simulasi model yang akan menghasilkan prototipe.
2 *Blockchain* yang di maksud merupakan *private Blockchain*.
3 Visualisasi sistem ditampilkan melalui website.
4 Aspek performa yang diukur dalam penelitian ini adalah kecepatan transaksi untuk menemukan nilai *Hash* baru.

BAB II
TINJAUAN PUSTAKA

2.1 *Blockchain*

Blockchain dimulai pada tahun 2008 melalui *Bitcoin*, yaitu sistem pembayaran elektronik pada jaringan *peer-to-peer* yang bersifat terdesentralisasi tanpa adanya institusi finansial yang bertindak sebagai pengatur jalannya transaksi. *Blockchain* ini diterapkan untuk menghilangkan kebutuhan institusi finansial sebagai pihak ketiga dalam pengelolaan suatu proses transaksi.

Secara umum, *Blockchain* adalah DBMS (Database Management Sistem). Namun tidak hanya sekedar DBMS, *Blockchain* memberi perlindungan ekstra terhadap informasi yang ada di dalamnya dengan menambahkan beberapa bagian. Pertama adalah struktur data yang dikelompokkan ke dalam *Block-Block* dan mendapatkan pengamanan dengan menggunakan Merkle Tree. Melalui susunan Merkle Tree, informasi yang ada tidak mudah diubah. Selain itu data header ditambahkan ke dalam sistem *Blockchain*. Dimana header yang satu terkoneksi dengan 2 header lain yang berasal dari *Block* sebelum dan setelahnya. Manfaat dari koneksi antar header ini adalah untuk mempersulit pihak manapun melakukan modifikasi atas informasi header maupun informasi transaksi yang ada didalam *Block* (Wijaya dan Oscar, 2018). Aplikasi ini adalah untuk menawarkan sebuah lingkungan yang

arik bagi para pengguna untuk mengeksplorasi dasar-
lan keuntungan dari *Blockchain* dengan cara yang
.(Asmawi, Saifulbahri, & Arifin, 2024)

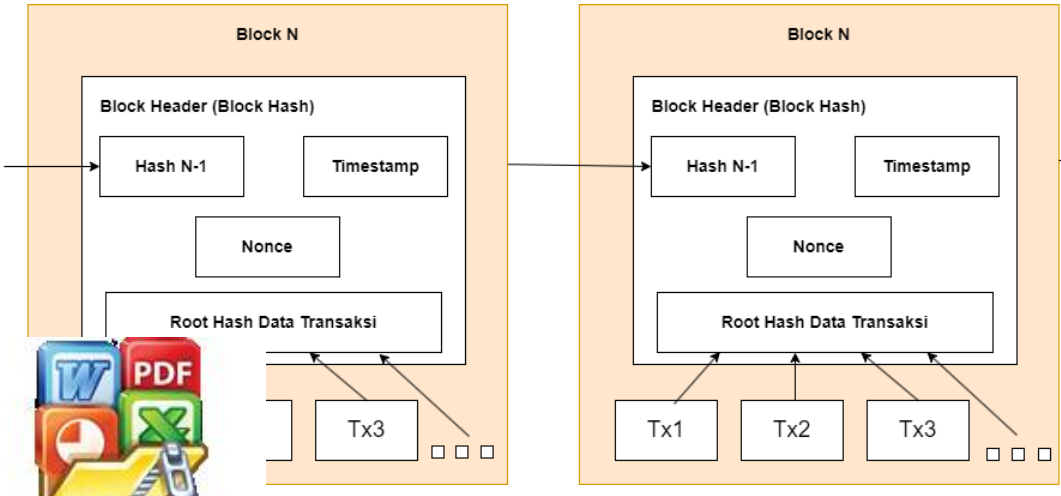
Pencatatan data pada teknologi *Blockchain* dilakukan melalui beberapa tahap, yaitu:

1. Input data: *Node* akan melakukan transaksi data.
2. Verifikasi data: *Node* yang lain dalam jaringan melakukan verifikasi terhadap data baru yang diterima.
3. *Hash*ing: Setelah *Block* baru telah diverifikasi, *Node* penerima akan melakukan pembentukan *Block* baru berdasarkan protokol konsensus yang diterapkan. Dalam konsensus *proof of work* pembentukan *Block* baru ini dikenal dengan istilah mining, yang dimana akan menghasilkan nilai *Hash* pada *Block*.
4. Penambahan *Block:* Setelah verifikasi dan *hashing* selesai, *Block* baru ditambahkan ke dalam jaringan *Blockchain* sebagai bagian dari sejarah transaksi.
5. Penyebaran data: Setelah *Block* baru ditambahkan, *Block* tersebut akan disebarluaskan ke semua *node* dalam jaringan agar informasi pada *Block* tersebut tersebar ke semua *node* yang ada pada jaringan *Blockchain*.

Pada *Blockchain* yang digunakan *Bitcoin*, setelah transaksi mata uang digital (cryptocurrency) dilakukan oleh suatu node melalui wallet, *Block* yang berisikan informasi mengenai transaksi tersebut akan diumumkan ke semua node yang ada pada jaringan. Node lain kemudian akan menerima *Block* yang diumumkan tersebut lalu akan menjalankan mekanisme protokol *Proof of Work*. Mekanisme *Proof of work* akan *Block* baru yang terhubung dengan *Block* sebelumnya menggunakan fungsi *Hash* kriptografi, *SHA-256*. *Block* cara menghitung nilai *Hash*-nya. Nilai *Hash* ini biasa *Block Hash* atau *Block* header. *Block Hash* akan

didapatkan melalui komputasi fungsi *Hash* yang dilakukan oleh miner, miner melakukan komputasi fungsi *Hash* dari nilai data transaksi yang tergabung dalam *Block* dan beberapa nilai khusus, seperti timestamp, nonce, ataupun *Block Hash* dari *Block* sebelumnya. Hubungan antara suatu *Block* dengan *Block* lainnya akan terbentuk dengan adanya penggunaan nilai *Block Hash* terakhir sebagai nilai masukan (input) dalam pembentukan nilai *Block Hash* baru. Ilustrasi *Blockchain* dapat dilihat pada Gambar 2.1 dan Gambar 2.2.

Gambar 2.1 Ilustrasi *Blockchain*



ambar 2.2 Detail *Block* pada *Blockchain*

Protokol konsensus seperti *Proof of work* merupakan mekanisme yang digunakan jaringan *Blockchain* untuk mencapai kesepakatan bersama dalam pembentukan *Block* data yang valid. Melalui konsensus, *Blockchain* tidak perlu mengandalkan pihak manapun dalam menjalankan sistemnya. Dalam implementasi *Blockchain* pertama di sistem *Bitcoin*, para miner adalah pihak yang memiliki hak untuk memberikan suara, dimana jumlah suara yang mereka miliki bergantung pada persentase mining power mereka terhadap total mining power di dalam jaringan.

Dalam protokol *Proof of Work*, ada aturan yang harus dipenuhi dalam perhitungan *Hash Block* untuk membuat *Block* baru. Aturan ini biasanya disebut sebagai *Difficulty*. *Difficulty* pada *Blockchain* adalah parameter yang digunakan untuk mengatur tingkat kesulitan dalam menambang (mining) *Block* baru dalam jaringan *Blockchain*. Ini ditentukan oleh beberapa faktor, salah satunya yaitu, mengharuskan nilai dari suatu *Block Hash* memiliki beberapa angka nol pada bit awalnya. Tujuan dari mengatur tingkat *Difficulty* adalah untuk memastikan bahwa *Block* baru dapat ditambang dan ditambahkan ke ledger *Blockchain* secara teratur dan stabil. Jika daya komputasi yang tersedia meningkat, maka tingkat kesulitan akan ditingkatkan sehingga waktu rata-rata untuk mining *Block* tetap konstan. Sebaliknya, jika daya komputasi menurun, tingkat kesulitan akan diturunkan. Ini memastikan bahwa jaringan dapat berfungsi secara efisien dan stabil.

Dalam perhitungan nilai *Block Hash* yang sesuai berdasarkan *Difficulty* nilai masukan *nounce* merupakan kunci utama dalam *Nounce* adalah nilai masukan yang ditentukan oleh lakukan komputasi *Block Hash*. Penggunaan nilai dinamis seperti *nounce* akan dibutuhkan untuk *ock Hash* yang sesuai berdasarkan *Difficulty* yang

ditetapkan. Nilai *nounce* dibutuhkan oleh node dalam komputasi *Block Hash* karena nilai masukan lain seperti data utama, timestamp, ataupun *Block Hash* dari *Block* sebelumnya memiliki nilai yang tetap. Jenis-jenis *Blockchain* dapat dilihat pada Tabel 2.1

Tabel 2.1 Jenis *Blockchain*

Jenis *Block**chain***	**Penjelasan**
Public Blockchain	*Public Blockchain* merupakan jaringan besar yang terdistribusi yang dijalankan melalui token dan dapat diakses melalui sebuah kode yang dikelola oleh komunitas yang bersangkutan (Rahardja dkk, 2021).
Permissioned Blockchain	*Permissioned Blockchain* mengacu pada jenis *Blockchain* dengan batasan pada keanggotaan dan prosedur pengendalian. Dalam *Blockchain* seperti Ripple, konfigurasi intrinsik menentukan peran peserta di mana beberapa anggota dapat mengakses, menulis informasi pada *Blockchain*, atau menyetujui penerimaan anggota baru (Liu, M dkk 2019).
Private Blockchain	*Blockchain* berbasis *private* merupakan tipe *Blockchain* yang tidak dipublikasikan kode sumbernya dan menjadi *proprietary* (hak intelektual). *Blockchain private* juga berarti jaringan *Blockchain* yang digunakan untuk kepentingan perusahaan secara spesifik, misalnya *Blockchain* yang dikembangkan

	untuk meningkatkan performa perdagangan komoditas.

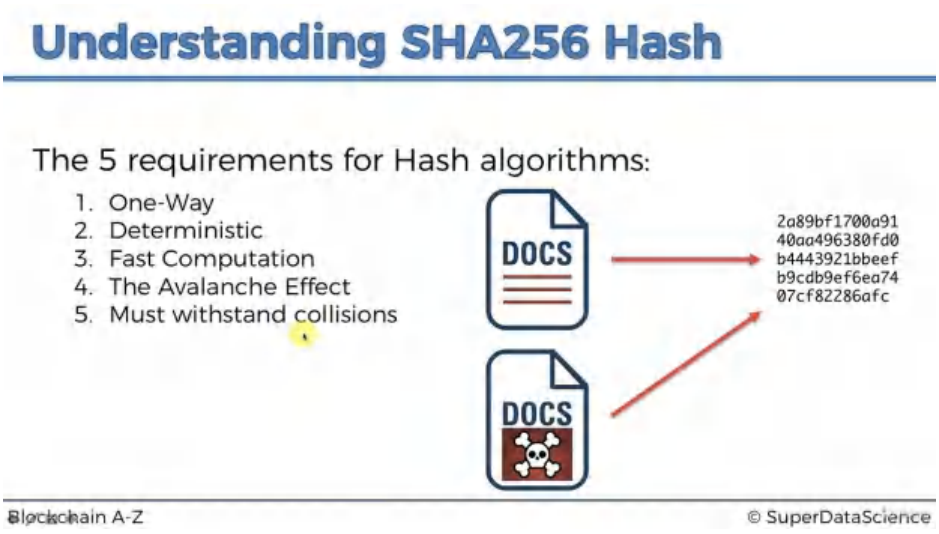
2.2. Algoritma Secure Hash Algorithm (SHA)

SHA merupakan algoritma teknik enkripsi yang dikembangkan pada tahun 1993 oleh National Insitute and technology (NIST) dan dipublikasi sebagai Federal Information Standars (FIPS 180). Pada tahun 1995 ditemukan kelamahan pada algoritma SHA-0 sehingga berbagai revisi dan perbaikan dilakukan untuk mengembangkan algoritma yang lebih baik. SHA-1 merupakan hasil perbaikan dari SHA-0 yang diterbitkan pada tahun 1995. Pada tahun 2001 SHA-2 diterbitkan setelah beberapa tahun SHA-1 diterbitkan, pada tahun 2006 ditemukan kelemahan pada SHA-1 yang rentan terhadap serangan brute attack sehingga NIST membuat pernyataan untuk mengadopsi penggunaan SHA-2 kepada Lembaga federal pada tahun 2010. SHA 256 adalah algoritma *Hash* aman yang banyak digunakan dengan ukuran digest 256 bits. Ini adalah bagian dari keluarga algoritma SHA-2, yang diterbitkan pada tahun 2001 sebagai penerus SHA-1 karena meningkatnya kerentanan terhadap serangan brute force. SHA 256 menyediakan nilai *Hash* ireversibel yang memastikan integritas data dan otentikasi pesan. Algoritma SHA 256 mengubah pesan input, yang bisa memiliki panjang berapa pun, menjadi nilai *Hash* 256-bit berukuran tetap. Ini dilakukan dengan menambahkan bit padding ke es pesan empuk di *Block* 512-bit, dan menerapkan pulasi bit, fungsi logis, dan penambahan modular untuk *Hash*.(Vanamala, Jampani, & Sk, 2024)

Fitur utama SHA 256 meliputi:

a) Panjang Pesan: Pesan cleartext harus kurang dari 264 bit (sekitar 14 MB).
b) Panjang Digest: Nilai *Hash* yang dihasilkan selalu 256 bit.
c) Irreversibilitas: SHA 256 menghasilkan nilai *Hash* unik untuk pesan yang berbeda, sehingga sulit untuk merekayasa balik pesan asli dari nilai *Hash*.
d) Resistensi tabrakan: Sangat tidak mungkin bahwa dua pesan yang berbeda akan menghasilkan nilai *Hash* yang sama, memastikan integritas data.
e) SHA 256 digunakan dalam berbagai aplikasi seperti:
    - Password *Hash*ing: Menyimpan versi terenkripsi dari password pengguna untuk melindungi dari akses yang tidak sah.
    - Tanda tangan digital: Memastikan keaslian dan non-penolakan dokumen elektronik.
    - Komunikasi aman: Memberikan kerahasiaan dan integritas data dalam komunikasi terenkripsi.

SHA 256 juga digunakan secara luas dalam teknologi *Blockchain*, khususnya di jaringan *Bitcoin*, karena sifat keamanannya yang kuat. Langkah-langkah komputasi yang tepat yang terlibat dalam algoritma SHA 256 sangat kompleks dan di luar cakupan respons ini. Namun, memahami struktur umum dan aplikasi harus memberikan dasar yang kuat untuk eksplorasi lebih lanjut. Perhatikan Gambar 2.3



Gambar 2.3 Konsep SHA256

Terdesentralisasi (*Decentralized*) Pengelolaan *Blockchain* dilakukan secara terdesentralisasi tanpa adanya satu pihak utama yang memiliki kontrol penuh terhadap prosesnya. Setiap node dalam sistem *Blockchain* dapat mengelola rantai data secara bersama-sama, sehingga hak akses yang dimiliki oleh setiap node sama terhadap rantai data yang terbentuk. Dengan cara ini, tidak ada pihak yang memiliki kekuasaan yang lebih besar daripada yang lainnya dalam sistem *Blockchain*. Jaringan *Peer-to-peer* (P2P) Dalam sistem *Blockchain*, setiap node akan berkomunikasi dengan yang lainnya melalui jaringan *peer-to-peer* yang didasarkan pada sifat terdesentralisasi *Blockchain*. Ketika sebuah node mencatat data dengan membentuk *Block* baru pada rantai *Block* yang ada, *Block* baru tersebut akan diumumkan ke node lain dalam jaringan. Seiring waktu, rantai *Block* yang terbentuk akan tersalin ke seluruh node dalam sistem

ni menyebabkan data yang tercatat pada rantai *Block* hilang atau terhapus karena tersebar di seluruh node. node yang tergabung dalam jaringan *Blockchain*,

semakin sulit untuk hilang atau terhapus. Karakteristik ini membuat data dalam sistem *Blockchain* menjadi lebih andal dan terjamin.

Protokol Konsensus Dalam sistem *Blockchain*, desentralisasi dapat terjadi karena adanya penerapan mekanisme protokol konsensus seperti *Proof of work* atau Proof of Stake, sehingga tidak ada satu pihak utama yang memiliki kontrol penuh terhadap proses yang ada. Ketika sebuah node dalam jaringan *Blockchain* melakukan pencatatan data dengan membentuk *Block* baru, proses ini harus mengikuti mekanisme yang diterapkan oleh protokol konsensus agar node lain dalam jaringan dapat menyetujui dan menganggap *Block* yang terbentuk sebagai *Block* yang valid. Dalam protokol *Proof of Work*, validitas suatu *Block* baru ditentukan melalui proses komputasi yang besar, dimana bukti komputasi ini ditunjukkan melalui pembentukan nilai nonce yang tepat sehingga syarat angka *Hash* pada konsensus terpenuhi. *Block* yang terbentuk melalui mekanisme ini saling terhubung satu sama lain, sehingga jika terjadi perubahan pada sebuah *Block*, hal ini akan mempengaruhi keabsahan *Block-Block* yang terletak di belakangnya. Oleh karena itu, penyerang yang ingin melakukan manipulasi data pada rantai *Block* harus melakukan proses yang panjang dan memakan waktu serta komputasi yang besar.

Protokol konsensus dalam *Blockchain* dapat dianggap sebagai cara untuk mencegah adanya manipulasi data yang dilakukan oleh penyerang yang tergabung sebagai node dalam jaringan. Setiap node akan mengandalkan rantai *Block* terpanjang sebagai dasar kepercayaannya. Rantai *Block* terpanjang dianggap sebagai yang valid karena dibentuk melalui proses komputasi terbesar dibandingkan dengan rantai *Block* lainnya. Node yang jujur akan berusaha untuk membangun *Block* terpanjang sehingga apabila terdapat node penyerang yang ingin memanipulasi rantai *Block*, ia harus membentuk rantai *Block* baru yang lebih panjang daripada rantai *Block* yang dibentuk oleh node jujur.

Namun, untuk melakukan hal tersebut, node penyerang harus memiliki kekuatan komputasi yang lebih besar daripada gabungan kekuatan komputasi node jujur. Semakin banyak node jujur yang tergabung dalam jaringan *Blockchain*, semakin sulit untuk terjadinya penyerangan. Mekanisme ini memberikan keandalan pada data yang tersimpan dalam sistem *Blockchain*. Aksesibilitas Publik Sistem *Blockchain* dioperasikan secara terdesentralisasi, di mana setiap node memiliki hak akses yang setara terhadap jaringan rantai *Block* data.

Block Block merupakan sekelompok transaksi yang dihubungkan satu sama lain dalam sebuah rantai dengan menggunakan *Hash Block* sebelumnya. Hal ini dimungkinkan karena *Hash* tersebut dihasilkan secara kriptografi dari data yang terdapat dalam *Block*. Adanya keterkaitan ini dapat mencegah terjadinya kecurangan, sebab apabila ada satu perubahan pada *Block*, maka seluruh *Block* yang ada setelahnya akan menjadi tidak sah karena *Hash*-nya akan berubah. Dan hal ini akan terdeteksi oleh semua orang yang menjalankan *Blockchain* (Wackerow, 2022). Informasi-informasi yang terdapat pada *Block* header dari suatu *Block* yaitu (Wood 2022):

- parent*Hash*: *Hash* dari *Block* header milik parent *Block*, yaitu *Block* terakhir sebelum *Block* terkait.
- ommer*Hash*: *Hash* dari daftar ommer *Block* terkait.
- Beneficiary: address dari account yang menerima biaya pembayaran dalam proses mining *Block* terkait.
- stateRoot: *Hash* dari state yang tersimpan pada *Block* terkait. *Hash* ini terbentuk menggunakan struktur Merkle Patricia Trie
- ıRoot: *Hash* dari transaksi yang tercatat pada *Block* *sh* ini terbentuk menggunakan struktur Merkle Patricia

- receiptsRoot: *Hash* dari resi transaksi yang tercatat pada *Block* terkait. *Hash* ini terbentuk menggunakan struktur Merkle Patricia Trie.
- logsBloom: struktur data bloom filter yang terdiri atas log/catatan informasi.
- difficulity: tingkat difficulity dari penyelesaian *Block* terkait.
- number: nilai penjumlahan (count) dari *Block* terkait.
- gasLimit: batas maksimum gas yang dapat digunakan dalam pemrosesan komputasi transaksi pada *Block* terkait.
- gasUsed: jumlah total gas yang digunakan pada pemrosesan komputasi transaksi dari *Block* terkait.
- timestamp: timestamp yang berbasis unix dari pembentukan *Block* terkait.
- mix*Hash*: sebuah *Hash*, yang jika dikombinasikan dengan nonce, akan membuktikan bahwa *Block* terkait telah dibentuk melalui proses komputasi yang cukup.
- nonce: sebuah *Hash*, yang jika dikombinasikan dengan mix*Hash*, akan membuktikan bahwa *Block* terkait telah dibentuk melalui proses komputasi yang cukup.

2.3 Algoritma Hash *Keccak*

Algoritma *Keccak* dibuat dan dirancang oleh Guido Breton, Joan Daemen, Michaël Peeters and Gilles Van Assche. Nama *Keccak* berasal dari Tari Kecak, tarian Bali. Pada tahun 2004-2005, beberapa algoritma *Hash* kriptografi berhasil diserang, dan serangan serius dipublikasikan terhadap

ujui NIST. Sebagai tanggapan, NIST mengadakan dua
ntuk menilai status algoritma *Hash* yang disetujui, dan
sukan publik mengenai kebijakan dan standar algoritma
a. Sebagai hasil dari lokakarya ini, NIST memutuskan

untuk mengembangkan algoritma *Hash* kriptografi baru untuk standardisasi melalui kompetisi publik. Algoritma *Hash* baru akan disebut sebagai SHA-3. NIST mengumumkan Kompetisi Algoritma *Hash* Kriptografi SHA-3 pada tanggal 2 November 2007, dan mengakhiri kompetisi pada tanggal 2 Oktober 2012, ketika mengumumkan *KECCAK* sebagai algoritma pemenang untuk distandarisasi sebagai SHA-3 baru. *Keccak* berbeda dari finalis SHA3 lainnya dalam hal menggunakan konstruksi spons (sponge construction), di mana setiap *Block* pesan masukan akan di-XOR-kan dengan *Block bitrate* dan state sebelumnya untuk kemudian dilewatkan ke dalam fungsi permutasi *Keccak*-f. (fase absorbing & sqeezing). Jika desain lainnya bergantung pada fungsi kompresi, *Keccak* menggunakan fungsi non-kompresi untuk menyerap dan kemudian mengompresi digest singkat (Ros & Sigurjonsson, 2016).

Algoritma *Keccak* menerima tiga parameter masukan, yaitu *bitrate* (r), capacity (c), dan difersity (d). Secara umum proses dari *Keccak* ini adalah (Paar & Pelzl, 2016):

1. Proses pesan masukan (P), yaitu menerapkan padding pada pesan masukan.
2. Pemecahan pesan masukan menjadi P0, P1,p2, …. Pi, dimana I = jumlah kelipatan panjang *bitrate* untuk panjang pesan masukan.
3. Absorbing pada semua pecahan pesan masukan.
4. *Squeezing* sebanyak j, dimana j = kelipatan panjang keluaran r/w untuk memenuhi panjang output yang diinginkan.

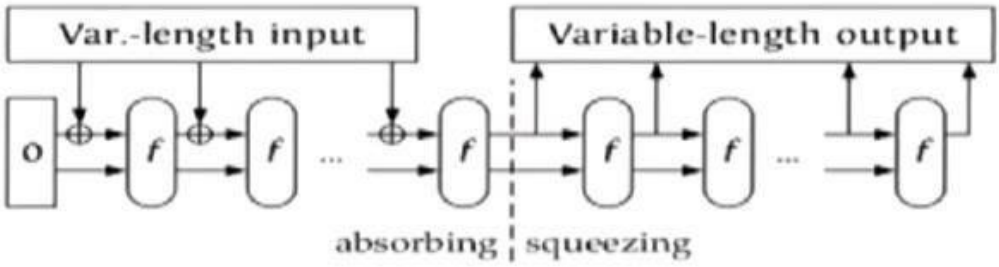
ıpakan konkatenasi dari keluaran *Squeezing* pada tentu. State pada *Keccak* merupakan serangkaian bit sebagai suatu array tiga dimensi dari bit-bit tersebut. a *Keccak* berbasis pada konstruksi spon. Konstruksi

spon merupakan konstruksi iterasi sederhana untuk membangun sebuah fungsi spon dengan variabel. Panjang input dan panjang output yang berubah-ubah bergantung pada panjang transformasi atau permutasi tetap yang beroperasi dalam sejumlah bit.

Konstruksi spon terdapat dua fase, yaitu fase absorbing dan fase *Squeezing*. (Paar & Pelzl, 2016)

1. Fase absorbing, merupakan fase dimana proses dilakukan terhadap semua pecahan dari input masukan dan di XOR-kan dengan bagian *bitrate* dari state kemudian dilewatkan kedalam fungsi f.
   - **Inisialisasi State:**
     - State internal *Keccak* dimulai dengan keadaan awal yang terdiri dari semua nol. State ini berukuran 1600 bit.
   - **Pembagian State:**
     - State dibagi menjadi dua bagian: "rate" (rrr) dan "capacity" (ccc).
     - rrr dan ccc memiliki ukuran sedemikian rupa sehingga r + c = 1600r + c = 1600r + c = 1600 bit. Misalnya, untuk SHA-3-256, r = 1088r = 1088-bit dan c = 512c = 512 bit.
   - **Proses Absorbing:**
     - Pesan input dibagi menjadi *Block-Block* yang masing-masing berukuran rrr bit.
     - Setiap *Block* pesan di-XOR dengan bagian rate dari state internal.
     - Setelah setiap XOR, seluruh state diproses menggunakan fungsi permutasi *Keccak*-f yang kompleks, yang melibatkan rangkaian operasi non-linier dan difusi.
   - **Proses Ulang:**
     - Langkah-langkah ini diulangi sampai seluruh pesan telah diproses.

2. Fase *Squeezing*, merupakan fase untuk mendapatkan hasil keluaran. Dalam fase ini dilakukan konkatenasi terhadap sejumlah bit tertentu dari hasil fungsi f sehingga jumlah bit konkatenasi tersebut sama dengan jumlah bit konkatenasi yang diinginkan. Kedua fase ini dapat dilihat pada gambar 2.4



Gambar 2.4 Fase Konstruksi spon Algoritma *Keccak*

- **Inisialisasi Output Hash:**
  - Setelah fase absorbing selesai, fase *Squeezing* dimulai untuk menghasilkan nilai *Hash* dari state internal.
- **Ekstraksi Hash:**
  - Nilai *Hash* diekstraksi dengan membaca rrr bit pertama dari state.
  - Jika *Hash* yang diinginkan lebih panjang dari rrr bit, maka fungsi permutasi *Keccak*-f diterapkan lagi pada state, dan proses ekstraksi dilanjutkan.
  - Proses ini diulang sampai panjang output *Hash* yang an tercapai.

Hash:

- Hasil dari fase *Squeezing* adalah nilai *Hash* akhir yang digunakan untuk keperluan kriptografi.

2.4 Ilustrasi Sederhana mengenai fase dalam *Keccak*

Misalkan kita memiliki pesan MMM yang panjangnya 2048-bit dan kita menggunakan *Keccak* dengan r = 1088r = 1088-bit dan c = 512c = 512 bit, maka:

1. **Absorbing:**
   - Bagi MMM menjadi *Block-Block* 1088 bit.
   - Misalkan M1M_1M1, M2M_2M2, ..., M2M_2M2 adalah *Block-Block* tersebut.
   - Lakukan XOR antara M1M_1M1 dengan 1088 bit pertama dari state, lalu permutasikan state.
   - Ulangi untuk M2M_2M2, M3M_3M3, sampai M2M_2M2 selesai.
2. **Squeezing:**
   - Baca 1088-bit pertama dari state untuk mendapatkan sebagian *Hash*.
   - Permutasikan state, lalu baca lagi 1088-bit pertama jika *Hash* yang dibutuhkan lebih panjang. Ulangi sampai panjang *Hash* yang diinginkan tercapai.

2.5. Algoritma *Hash Skein*

Skein diciptakan oleh Bruce Schneier , Niels Ferguson , Stefan Lucks , Doug Whiting, Mihir Bellare , Tadayoshi Kohno, Jon Callas dan Jesse

alah fungsi *Hash* kriptografi dan salah satu dari lima ngsi *Hash* NIST. Masuk sebagai kandidat standar SHA-1 dan SHA-2 , akhirnya kalah dari kandidat *Hash*

NIST *Keccak* . Nama *Skein* mengacu pada bagaimana fungsi *Skein* menjalin input, mirip dengan gulungan benang.

Skein didasarkan pada cipher *Block Threefish* yang dapat diubah dan dikompresi menggunakan mode rangkaian Unique *Block* Iteration (UBI), sebuah varian dari mode *Hash* Matyas Meyer Oseas , sambil memanfaatkan sistem argumen overhead rendah opsional untuk fleksibilitas. Algoritma *Skein* dan implementasi referensi diberikan ke domain publik Fungsi *Skein* adalah sebuah fungsi *Hash* kriptografi yang digunakan untuk menghasilkan nilai *Hash* dari data input. Nilai *Hash* adalah representasi digital dari data input yang memiliki ukuran tetap, dan digunakan untuk tujuan keamanan dan integritas data. *Skein* menggunakan konstruksi Merkle-Damgård, yang merupakan salah satu metode umum untuk merancang fungsi *Hash* kriptografi. Dalam konstruksi ini, data input dibagi menjadi *Block-Block* kecil yang diolah secara berurutan.

Skein menggunakan konsep putaran yang diiterasikan untuk mengolah *Block-Block* data input. Setiap putaran melibatkan serangkaian transformasi kriptografis yang kompleks untuk menghasilkan nilai *Hash*. *Skein* dirancang dengan filosofi konfigurasi yang kuat, yang memungkinkan pengguna untuk menyesuaikan parameter-parameter tertentu sesuai dengan kebutuhan aplikasi. Ini termasuk ukuran panjang *Hash*, jumlah putaran, dan penggunaan kunci.

Skein didesain untuk memberikan tingkat keamanan yang tinggi terhadap berbagai serangan kriptografi yang dikenal, sambil tetap mempertahankan tingkat efisiensi yang baik dalam pengolahan data. *Skein* telah diimplementasikan dalam berbagai aplikasi dan protokol

n keseimbangan yang baik antara keamanan, efisiensi,

salah satu algoritma *Hash* yang dikembangkan dalam untuk memilih SHA-3, yang merupakan pengganti

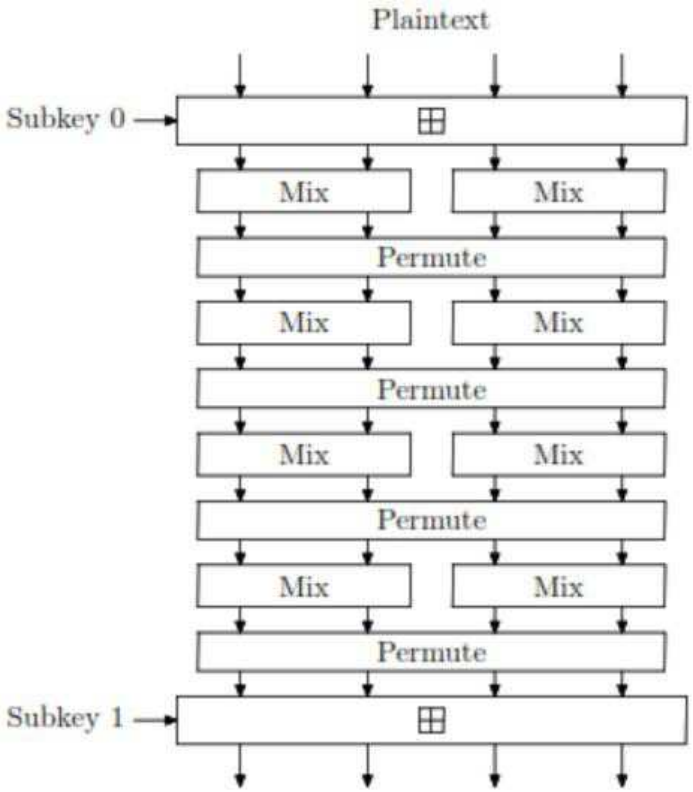
algoritma SHA-2. Tim yang merancang *Skein* terdiri dari ahli terkemuka seperti Bruce Schneier, Niels Ferguson, Stefan Lucks, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, dan Jesse Walker.

Pada tabel 2.2 menunjukan bahwa adanya perkembangan pengganti algoritma sebagai pengembangan awal dari MD5, kemudian terdapat pula ukuran dasar dan menghasilkan ukuran tertentu.

Gambar 2.5 menunjukkan model Empat dari 72 putaran sandi cipher *Block Threefish* – 256

Tabel 2.2 Susunan Pengganti MD5

Replace	With	State Size	Output Size
MD5	Skein-256-128	256	128
	Skein-512-128	512	128
SHA-1	Skein-256-160	256	160
	Skein-512-160	512	160
SHA-224	Skein-256-224	256	224
	Skein-512-224	512	224
SHA-256	Skein-256-256	256	256
	Skein-512-256	512	256
SHA-384	Skein-512-384	512	384
	Skein-1024-384	1024	384
SHA-512	Skein-512-512	512	512
	Skein-1024-512	1024	512



Gambar 2.5. Empat dari 72 putaran sandi cipher *Block Threefish* – 256

2.5.1 Komponen Utama dalam Skein

Mekanisme Threefish:

a. *Block* Cipher: *Skein* menggunakan cipher *Block* yang disebut *Threefish*, yang merupakan algoritma cipher yang sangat sederhana namun kuat, didesain untuk kinerja tinggi di perangkat keras dan perangkat lunak.

Structure: *Threefish* menggunakan struktur Feistel-like ungkinkan fleksibilitas dalam desain dan optimasi.

Block Cipher: Threefish menggunakan konsep block cipher, yang berarti memiliki parameter tambahan

yang disebut "tweak" yang dapat diubah untuk memberikan variasi dalam enkripsi tanpa mengubah kunci utama.

d. Key Schedule: Menggunakan skema penjadwalan kunci yang unik dan efisien untuk keamanan dan kinerja yang optimal.
e. *Skein Hash Function*:
f. Ubiquity: Didesain agar dapat diimplementasikan di berbagai platform, termasuk sistem tertanam dengan sumber daya terbatas hingga server dengan kinerja tinggi.
g. *Scalability*: Mendukung berbagai ukuran bit output, dari *Hash* 256-bit hingga 512-bit atau lebih, memberikan fleksibilitas yang besar bagi berbagai aplikasi keamanan.
h. *Unique Blok Iteration* (UBI): *Skein* menggunakan mekanisme iterasi *Block* unik yang memberikan kekuatan tambahan terhadap berbagai jenis serangan kriptografi. UBI terdiri dari beberapa komponen utama:
    - Chaining: Data dipecah menjadi blok-blok yang diproses satu per satu.
    - UBI Mode: Setiap blok diproses dengan operasi chaining yang melibatkan Threefish dan tweak yang unik untuk setiap blok.

2.5.2. Keamanan dan Efisiensi:

a. Analisis Keamanan: *Skein* dirancang untuk tahan terhadap berbagai serangan, termasuk serangan tabrakan (collision), pra-gambar (preimage), dan analisis diferensial.
b. Efisiensi: Algoritma ini dioptimalkan untuk kinerja tinggi dalam lingkungan, baik itu di perangkat keras maupun lunak, tanpa mengorbankan keamanan. (Malik, Aziz, &)13)

c. Implementasi dan Penggunaan *Skein* sangat serbaguna dan bisa digunakan dalam berbagai aplikasi kriptografi seperti: Enkripsi Data: Mengamankan data dalam proses penyimpanan dan transmisi.
d. Digital Signatures: Memastikan integritas dan autentikasi pesan.
e. Message Authentication Codes (MACs): Memastikan integritas pesan dan autentikasi pengirim.
f. Dengan desain yang mencakup aspek teoretis dan praktis, *Skein* menawarkan solusi yang kuat dan fleksibel untuk kebutuhan *Hash* kriptografi modern. Keunggulan ini menjadikan *Skein* salah satu kandidat kuat dalam kompetisi SHA-3 dan pilihan yang menarik untuk berbagai aplikasi kriptografi.