BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

Algoritma konsensus *Proof-of-Authority* (*PoA*) merupakan metode konsensus yang umum digunakan dalam jaringan *blockchain* privat dan berijin. Dalam mekanisme *PoA*, hanya *validator node* yang telah diverifikasi yang memiliki otoritas untuk memproduksi blok data baru dalam jaringan. *PoA* menawarkan beberapa keunggulan yaitu kecepatan transaksi tinggi karena tidak memerlukan proses penambangan seperti pada *Proof-of-Work* (PoW), konsumsi energi yang lebih rendah, toleransi kesalahan lebih baik untuk jaringan dengan entitas yang saling mengenal.

Salah satu implementasi *PoA* yang digunakan dalam penelitian ini adalah *Quorum Byzantine Fault Tolerance* (*QBFT*) yang diadopsi oleh *Hyperledger* Besu. *QBFT* menggabungkan prinsip *PoA* dengan mekanisme *Byzantine Fault Tolerance* (*BFT*) untuk memastikan keandalan konsensus meskipun sebagian *node* berperilaku tidak jujur. Secara umum, *QBFT* melibatkan tahapan:

1. Proposal blok oleh *Proposer*
2. Verifikasi (*Prepare*) oleh *validator*
3. Konfirmasi akhir (*Commit*) jika mayoritas dua pertiga node menyetujui

Teknologi *blockchain* ini mendukung integritas data terdistribusi dan meningkatkan efisiensi sinkronisasi data dalam sistem *cloud* terdistribusi.

2.1. Tinjauan Pustaka

Hasil penelitian Obadah Hammoud dan Ivan Tarkhanov (2020), menyarankan metode membagi catatan data menjadi blok untuk sinkronisasi data antara klien menggunakan teknologi *blockchain*. Metode sinkronisasi data *blockchain* dapat mengoptimalkan penggunaan kapasitas jaringan dan pemrosesan data di sisi klien. Hasil penelitannya menyimpulkan bahwa jika terdapat 1.000.000 entri data di *blockchain* yang perlu disinkronisasi, maka hanya 1.000 entri data yang akan diunduh pada proses pembaruan data yang sudah ada.

Liang, R., dkk. (2020), mengusulkan dan memperkenalkan implementasi sistem sinkronisasi data dengan teknik kontrak pintar *blockchain* untuk mencapai

verifikasi konsistensi dan keterlacakan proses sinkronisasi data antara server. Dalam penelitianya Liang, R., dkk. (2020), mengusulkan strategi sinkronisasi data menggunakan *Rsync* dan satu-ke-satu (*P2P*) untuk memilih mode sinkronisasi sesuai dengan ambang batas yang ditetapkan oleh sistem sinkronisasi data. Hasil eksperimennya menunjukkan bahwa sistem sinkronisasi data berbasis *blockchain* dapat secara efisien mengurangi *overhead* jaringan dan aman mencapai sinkronisasi data satu-ke-satu atau satu-ke-banyak di seluruh server.

Wang, Y., dkk. (2019), mengusulkan *ChainFileSynch* sebagai model sinkronisasi *folder* dan *file* penyimpanan *cloud* yang inovatif. Arsitektur *ChainFileSynch* berisi sinkronisasi file penyimpanan *cloud* yang cepat berdasarkan pemrosesan dan algoritma *blockchain*. Arsitektur *ChainFileSynch* menerapkan sinkronisasi penyimpanan *cloud* berdasarkan transaksi *file cloud*. *ChainFileSynch* memanfaatkan distribusi dan keterlacakan *blockchain*, mekanisme konsensus, dan sejenisnya. *ChainFileSynch* meningkatkan efisiensi sinkronisasi penyimpanan *cloud*, verifikasi sinkronisasi data, dan keandalan jaringan.

Kang, P., dkk. (2022), mengusulkan metode penyimpanan dan berbagi *file* pengetahuan menggunakan *blockchain NDN*. Metode tersebut menggunakan *NDN* untuk tanda tangan dan enkripsi konten file, guna memisahkan keamanan *file* dan proses transmisi. Metode tersebut menggunakan strategi penerusan dan perutean jalur balik *NDN* yang fleksibel dan menggabungkan jaringan penyimpanan pribadi *IPFS* untuk meningkatkan keamanan penyimpanan data terenkripsi. Metode tersebut mengambil keuntungan dari semua konsensus *node* yang berpartisipasi dan berbagi file di *blockchain* yang disinkronkan untuk memastikan keterlacakan.

Chunmiao, Li., dkk. (2019), mengusulkan arsitektur berbagi data medis yang membahas izin kontrol menggunakan kontrak pintar *blockchain*. Metode tersebut membagi data menjadi potongan-potongan halus untuk dibagikan dengan rekan-rekan yang berbeda kemudian mensinkronkan data lengkap dan potongan-potongan tersebut dengan transformasi dua arah. Data medis berada di basis data lokal setiap pengguna dan data terkait izin disimpan di kontrak pintar. Hasilnya hanya rekan yang memperoleh data bersama terbaru dapat melakukan operasi selanjutnya

menggunakan kontrak pintar. Proses bersama berbasis *blockchain* yang tidak dapat diubah dan memungkinkan pengguna untuk melacak riwayat pembaruan data.

Siddesh G. M., dkk. (2021), mengusulkan model sinkronisasi menggunakan *blockchain Ethereum* untuk memecahkan beberapa masalah rantai pasokan utama dalam mengelola *database* terdistribusi. Model tersebut menggabungkan teknik untuk memprediksi naik turunnya permintaan obat di pasar dengan menggunakan algoritma pembelajaran mesin seperti regresi linier dan *LSTM* (*Bidirectional Long Short-Term Memory*). Hasilnya menunjukkan bahwa saat menggunakan regresi linier, tren yang diprediksi tidak terlalu akurat dan tidak dapat melacak tren sebenarnya secara dekat. sedangkan *BLSTM* (*Bidirectional Long Short-Term Memory*) bekerja lebih baik dalam memprediksi tren data deret waktu.

Biswas, S., dkk. (2020), menyajikan sistem terpadu untuk migrasi sistem e-*health* konvensional independen ke satu ekosistem berbasis *blockchain*. Mereka membahas masalah perbedaan dalam struktur data untuk relasi *database* konvensional dan *database file blockchain*. Solusi tersebut menggambarkan proses konversi dan sinkronisasi informasi dalam sistem terpadu untuk data e-*health* skala besar. Hasil implementasi dan analisis menunjukkan penyimpanan data, kontrol akses, dan migrasi tanpa batas meningkat secara signifikan.

Susan Zucker dan Shouhong Wang (2009), meneliti dampak penerapan sinkronisasi data pada tiga perusahaan produk barang konsumen besar. Mereka mengungkapkan pentingnya penyelarasan internal seputar pembersihan dan akurasi data, serta peluang untuk peningkatan penyelarasan eksternal dari perspektif sistem. Hasil studinya menyimpulkan bahwa sinkronisasi data diperlukan untuk manajemen rantai pasokan di lingkungan e-*commerce B2B*. Mereka menemukan sinergi yang tercipta antara management item produk, sinkronisasi data, dan pemenang internal di ketiga perusahaan tersebut. Desain ulang alur kerja, peningkatan proses, dan pengembangan standar yang dikenakan pada perusahaan memberikan manfaat besar dari proses sinkronisasi data.

JinTao, Hao., dkk. (2018), mengusulkan model penyimpanan data berdasarkan *Inter-Planetary File System* (*IPFS*) dan *blockchain*. Tujuannya untuk menghindari pengguna jahat jika terjadi serangan pemalsuan data. *IPFS* digunakan untuk

menyimpan video, gambar, dan pemantauan data waktu nyata (*real-time*) yang dilaporkan dari sensor. *Blockchain* digunakan untuk menyimpan alamat *hash IPFS* dari data asalnya. Kemudian JinTao, Hao., dkk (2018), merancang sebuah mekanisme otentikasi berbasis *blockchain* yang dapat memverifikasi data dan memastikan data dengan keamanan yang efektif. Hasil eksperimen menunjukkan bahwa pendekatan yang diusulkan dapat mengungguli metode yang ada.

Wasseem N. Ibrahem Al-Obaydy, dkk. (2022), menggunakan algoritma *term frequency-inverse document frequency* (*IDF*) dan klasterisasi *K-means* untuk mengelompokkan dokumen teks artikel penelitian ke dalam kelompok-kelompok ekspresif yang mencakup bidang keilmuan yang sejenis. Hasil eksperimen menunjukkan bahwa teknik yang disarankan mengungguli algoritma *k-nearest neighbor* dalam hal akurasi dalam mengambil informasi.

Donghwa Kim, dkk. (2018), mengusulkan *multi-co-training* (*MCT*) untuk meningkatkan kinerja klasifikasi dokumen. Tujuan penelitianya untuk meningkatkan variasi *set* fitur untuk klasifikasi, mereka mengubah dokumen menggunakan tiga metode representasi dokumen *term frequency–inverse document frequency* (*TF–IDF*) berdasarkan skema *bag-of-words*, distribusi topik berdasarkan *Laten Dirichlet Allocation* (LDA), dan penyisipan dokumen berbasis jaringan saraf dikenal sebagai dokumen ke vektor (*Doc2Vec*). Hasil percobaan menunjukkan bahwa *MCT* yang diusulkan kuat terhadap perubahan parameter dan mengungguli metode *benchmark* dalam berbagai kondisi.

2.2. Dasar Teori

2.2.1. Sinkronisasi Data

Naveen Malhotra dan Anjali Chaudhary (2014) menyatakan bahwa sinkronisasi data merupakan proses membangun konsistensi antara data pada sumber dan target penyimpanan, serta melakukan harmonisasi data secara berkelanjutan. Aashima dan Anit Kaur (2014) mendefinisikan bahwa sinkronisasi data dalam ilmu komputer adalah proses menjaga konsistensi data dari sumber ke target penyimpanan, termasuk harmonisasi data secara kontinu dari waktu ke waktu. Pendapat D. Sethia, dkk. (2014), bahwa sinkronisasi data mengacu pada upaya

menjaga integritas data atau menjaga beberapa salinan dataset agar tetap koheren satu sama lain.

Model sinkronisasi data diklasifikasikan berdasarkan pertimbangan jenis data yang akan disinkronkan. Model sinkronisasi data tak-beraturan (*unordered data*) (juga dikenal sebagai masalah rekonsiliasi himpunan) dimodelkan sebagai upaya untuk menghitung perbedaan simetris (*symmetric difference*) antara dua set jarak jauh S_A dan S_B dari angka *b*-bit (Minsky, Y. dkk, 2003).

$$S_A \oplus S_B = (S_A - S_B) \cup (S_B - S_A) \quad (2.1)$$

Keterangan:

$S_A \oplus S_B$: Perbedaan simetris antara dua set jarak jauh S_A dan S_B

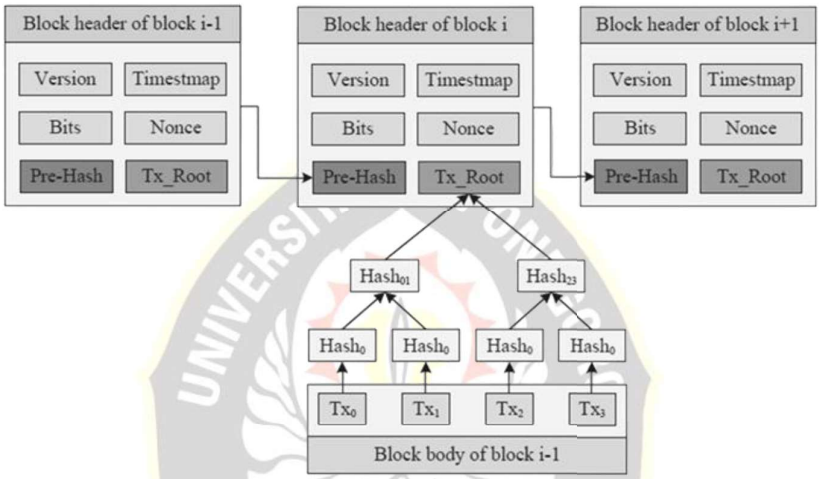
Model sinkronisasi data teratur (*Ordered data*) adalah model sinkronisasi melalui proses pengurangan jarak edit antara dua *string* jarak jauh σ_A dan σ_B, hingga jumlah pengeditan yang tetap (yaitu penyisipan karakter, penghapusan, atau modifikasi) telah mencapai jarak yang ideal dari nol. Model sinkronisasi data teratur banyak diterapkan di semua sinkronisasi berbasis sistem *file*.

2.2.2. *Blockchain*

Rantai blok (*blockchain*) atau semula dieja *block chain* adalah penyimpanan (*record*) yang disebut *block* yang terus berkembang yang terhubung dan diamankan menggunakan teknik kriptografi. Setiap blok memuat *hash* kriptografis dari blok sebelumnya, stempel waktu, dan data transaksi. *Blockchain* merupakan sebuah buku besar terdistribusi (*distributed ledger*) yang dapat mencatat transaksi antara dua pihak secara efisien dan dengan cara yang dapat diverifikasi dan permanen (Xianhua Wei1, dkk, 2020). *Blockchain* adalah sistem komputasi terdistribusi yang sangat penting untuk mengatasi masalah umum Bizantium (Lamport, dkk, 1982).

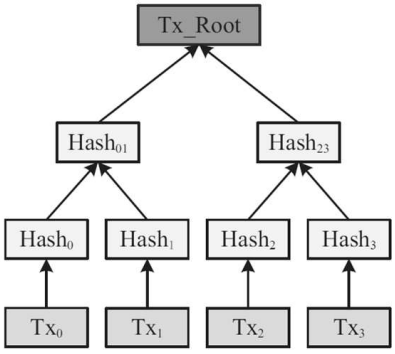
Blockchain adalah jaringan *peer-to-peer* tempat menyimpan buku besar terdistribusi yang berisi informasi tentang transaksi. Transaksi ini dikumpulkan dan disimpan dalam blok. Blok-blok ini terhubung satu sama lain seperti rantai sehingga sangat sulit mengubah blok, karena perubahan yang dibuat pada sebuah blok akan menyebabkan perubahan pada blok lain di rantai blok. *Blockchain* menggunakan

fungsi *hash* kriptografis yang membuat rantai blok tidak dapat diubah, hanya bisa ditambahkan, dan aman (Omer F, dkk, 2021). Gambar 2.1 adalah struktur lapisan data *blockchain*. *Header* blok berisi nomor versi, nomor acak, *hash* dari blok sebelumnya, *hash* akar pohon *Merkle*, stempel waktu, kesulitan bukti beban kerja saat ini, dll (Weichu Deng, dkk, 2022).



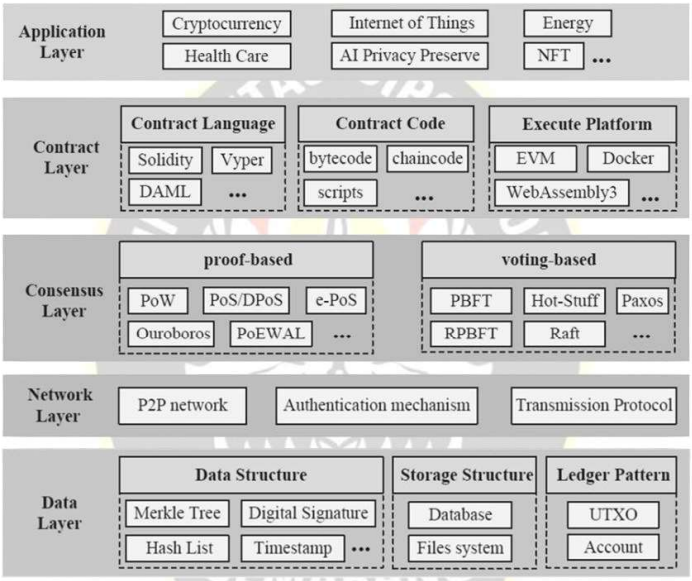
Gambar 2.1 Struktur *layer* data *Blockchain*

Blockchain menyimpan semua sejarah catatan transaksi. Transaksi di blok disimpan sebagai pohon *Merkle* untuk mempercepat transaksi. Pohon *Merkle* pada Gambar 2.2 menghubungkan *node* induk dan anak dengan penunjuk *hash*.



Gambar 2.2 Pohon *Merkle*

Weichu Deng, dkk. (2022) menjelaskan bahwa arsitektur *blockchain* terdiri atas lima lapisan, sebagaimana ditunjukkan pada Gambar 2.3. Lapisan konsensus merupakan komponen yang sangat unik dan mendasar dalam sistem *blockchain*. Du, dkk. (2017) menyatakan bahwa protokol konsensus dikembangkan untuk menjamin keberlanjutan sistem. Anderson (2019) menyebutkan bahwa arsitektur platform *blockchain* yang terdesentralisasi bersifat global, sehingga mampu mentoleransi kegagalan sistem atau serangan berskala lokal.

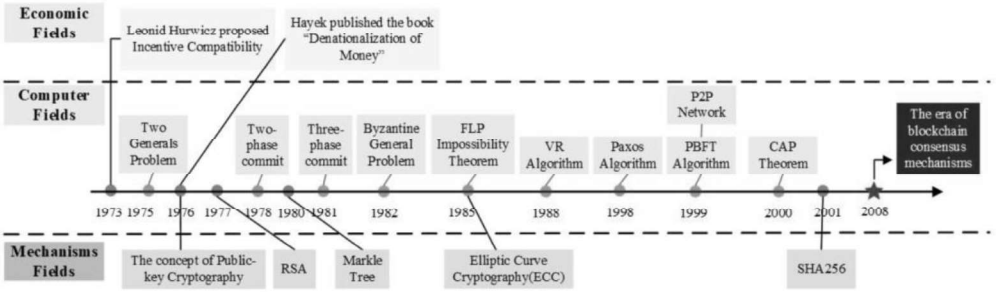


Gambar 2.3 Arsitektur teknologi *Blockchain*

2.2.3. Algoritma Konsensus *PoA Blockchain*

Konsensus adalah komponen dasar dari *blockchain* yang berkembang dari waktu ke waktu. Mekanisme konsensus *blockchain* mulai diperkenalkan sejak 2008 setelah munculnya alogritma hash kriptografi SHA256. Gambar 2.4 memperlihatkan peta jalan dari algoritma konsensus *blockchain*. Konsensus *blockchain* memungkinkan pemeliharaan dan pemutakhiran buku besar (*ledger*) dan menjamin keterpercayaan catatan yaitu keandalan, keaslian, dan keakurasian (Tasca, dkk, 2019). Lapisan konsensus mengimplementasikan algoritma konsensus yang mengatur dan mengoordinasikan sistem terdistribusi, sehingga

memungkinkan *blockchain* beroperasi dengan aman dan stabil (Weichu Deng, dkk, 2022).



Gambar 2.4 Dasar teori dari mekanisme konsensus *Blockchain*

PoA adalah keluarga algoritma konsensus *blockchain* berizin. *PoA* dikenal karena peningkatan kinerja sehubungan dengan algoritma *BFT* (*Byzantine Fault Tolerance*) yang khas dari hasil pertukaran pesan yang lebih ringan (De Angelis, dkk, 2017). Algoritma *PoA* bergantung pada sekumpulan *N node* tepercaya yang disebut otoritas. Setiap otoritas diidentifikasi dengan *id* yang unik dan sebagian besar dianggap jujur, yaitu minimal *N*/2+1. Otoritas menjalankan konsensus untuk memesan transaksi yang dikeluarkan oleh klien (De Angelis, dkk, 2017). Menurut Lamport, L. dkk, (2019), jika kita ingin mencapai konsistensi keputusan dalam mekanisme konsensus *BFT*, jika jumlah pengkhianat adalah *f*, maka jumlah jendral yang kita butuhkan minimal adalah 3*f*+1.

Algoritma konsensus *PoA* menetapkan *node* validasi yang dapat menghasilkan *block* baru. *Node* validasi menjaga jaringan *blockchain* dan *ledger* terdistribusi. Daftar *node* validasi disimpan dalam *registry blockchain*. Rumus berikut menentukan simpul pemimpin yang harus menghasilkan blok baru pada waktu tertentu.

$$leader = ((time - first) / step) \% nodes \quad (2.2)$$

Keterangan:

leader : Pemimpin *node*

time : Waktu sistem

first : Waktu pembuatan blok

step : Jumlah detik pembuatan blok

Dengan *leader* adalah pemimpin *node* saat ini, *time* adalah waktu sistem saat ini, *first* adalah waktu dimana blok dihasilkan pertama kali, *step* adalah jumlah detik dalam interval menghasilkan blok, dan *node* adalah jumlah *node* pada interval menghasilkan blok saat ini.

2.2.4. Teori *CAP*

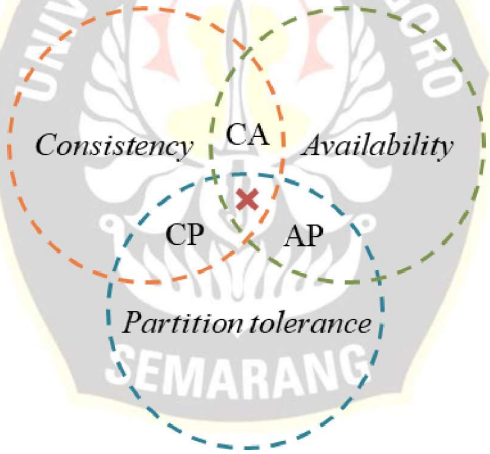
Teori CAP (*Consistency*, *Availability*, *Partition Tolerance*) dan prinsip ketidakmungkinan FLP (Fischer, Lynch, dan Paterson) menjelaskan bahwa terdapat batasan fundamental dalam perancangan sistem terdistribusi, termasuk sistem *blockchain*. Fox, dkk. (1999) menyatakan bahwa dalam kondisi terjadinya partisi jaringan, suatu sistem terdistribusi tidak dapat menjamin secara simultan ketiga aspek utama, yaitu konsistensi, ketersediaan, dan toleransi terhadap partisi. Implikasinya, ketika sistem mengalami gangguan komunikasi antar *node*, pengembang harus memilih dua dari tiga aspek tersebut dan mengorbankan salah satunya.

Toleransi terhadap partisi (*partition tolerance*) merupakan syarat mutlak dalam konteks *blockchain*, karena jaringan *blockchain* dirancang untuk tetap berfungsi meskipun terjadi gangguan konektivitas antar *node*. Oleh karena itu, pengembang *blockchain* harus memilih antara konsistensi (yakni data yang seragam di seluruh *node*) dan ketersediaan (yakni setiap permintaan selalu memperoleh respons). Pada *blockchain* publik seperti Bitcoin atau Ethereum, sistem lebih mengutamakan ketersediaan dan toleransi terhadap partisi, sehingga finalitas transaksi bersifat probabilistik. Sebaliknya, pada *blockchain* privat seperti Hyperledger Besu yang menggunakan algoritma konsensus PoA varian QBFT, sistem lebih mengedepankan konsistensi dan toleransi terhadap partisi, meskipun harus mengorbankan ketersediaan apabila jumlah *quorum validator* tidak tercapai.

Weichu Deng, dkk. (2022) menjelaskan bahwa keterbatasan yang dikemukakan dalam teori *CAP* merupakan pertimbangan penting dalam merancang algoritma konsensus pada sistem *blockchain*. Setiap algoritma konsensus harus

menyeimbangkan kebutuhan terhadap konsistensi, ketersediaan, dan toleransi terhadap partisi, yang tidak dapat dicapai secara bersamaan dalam kondisi terjadinya gangguan jaringan.

Gambar 2.5 menyajikan representasi visual dari Teori *CAP* dalam bentuk diagram Venn, yang terdiri atas tiga lingkaran: *Consistency* (C), *Availability* (A), dan *Partition Tolerance* (P). Titik perpotongan di tengah (diberi tanda silang) menunjukkan bahwa tidak ada sistem terdistribusi yang dapat menjamin ketiga aspek tersebut secara simultan. Oleh karena itu, pemahaman terhadap batasan teori *CAP* menjadi landasan penting dalam menentukan strategi sinkronisasi data dan pemilihan algoritma konsensus dalam pengembangan sistem *blockchain* terdistribusi.



Gambar 2.5 Teori *CAP*

2.2.5. *Algoritma Elliptic Curve Digital Signature* (ECDSA)

Algoritma *Elliptic Curve Digital Signature Algorithm* (*ECDSA*) merupakan algoritma kriptografi yang menggunakan sistem kunci publik (*asymmetric key*) dan berfungsi untuk menandatangani serta memverifikasi keaslian data digital. *ECDSA* merupakan varian dari *Digital Signature Algorithm* (*DSA*) yang diimplementasikan di atas konsep *Elliptic Curve Cryptography* (*ECC*). Keunggulan utama dari *ECDSA* terletak pada efisiensi ukuran kunci dan kecepatan proses komputasi, sehingga sangat cocok diterapkan dalam sistem dengan keterbatasan sumber daya.

Algoritma *Elliptic Curve Digital Signature Algorithm* (ECDSA) telah digunakan secara luas dalam berbagai aplikasi modern yang membutuhkan keamanan dan keaslian data digital. Salah satu penerapan utama *ECDSA* adalah pada sistem *blockchain*, seperti pada jaringan Bitcoin, Ethereum, serta Hyperledger Besu yang menggunakan algoritma konsensus *Quorum Byzantine Fault Tolerance* (QBFT). Dalam konteks ini, *ECDSA* digunakan untuk menandatangani transaksi dan blok data guna memastikan bahwa informasi yang dikirim atau disimpan bersumber dari entitas yang sah. Secara matematis, *ECDSA* bekerja berdasarkan prinsip kurva eliptik, yang dinyatakan dalam persamaan umum berikut:

$$y2=x3+ax+b \tag{2.3}$$

Persamaan ini didefinisikan di atas bidang hingga (*finite field*) Fp, di mana p adalah bilangan prima yang besar. Operasi titik pada kurva eliptik menghasilkan struktur grup aditif yang digunakan untuk membentuk sistem kriptografi yang aman.

Algoritma *ECDSA* menggunakan sepasang kunci kriptografi, yaitu kunci privat (*private key*) dan kunci publik (*public key*), yang saling berkaitan secara matematis namun tidak dapat dihitung secara langsung satu sama lain. Kunci privat, yang dilambangkan dengan d, merupakan bilangan bulat acak yang diambil dari rentang $[1,n-1]$, di mana n merupakan order dari titik dasar pada kurva eliptik yang digunakan. Kunci privat ini bersifat rahasia dan hanya diketahui oleh pemiliknya.

Kunci publik, yang dilambangkan dengan Q, merupakan titik pada kurva eliptik yang diperoleh dari hasil operasi perkalian skalar antara kunci privat d dengan titik dasar G pada kurva tersebut. Secara matematis, kunci publik dituliskan sebagai berikut:

$$Q=d\cdot G \tag{2.4}$$

Keterangan:

G : Generator point (titik dasar di kurva)

N : Order dari G

Titik G dikenal sebagai *generator point*, yaitu titik awal yang telah ditentukan pada kurva eliptik dan digunakan sebagai referensi untuk menghasilkan seluruh titik lainnya melalui operasi penggandaan. Hubungan antara d dan Q bersifat satu arah dan tidak dapat dibalik secara praktis, sehingga menjamin keamanan sistem tanda tangan digital pada *ECDSA*.

2.2.6. *Metode Klasterisasi K-Mean*

T. Vo-Van, dkk. (2020) menyatakan bahwa klasterisasi atau pengelompokan merupakan teknik untuk membagi populasi atau titik data ke dalam beberapa kelompok sedemikian rupa, sehingga titik data dalam satu kelompok memiliki kemiripan yang lebih tinggi satu sama lain dibandingkan dengan titik data dalam kelompok yang berbeda. Abiodun M. Ikotun, dkk. (2021) menjelaskan bahwa klasterisasi merupakan metode pembelajaran tanpa pengawasan (*unsupervised learning*) yang dilakukan dengan cara mengelompokkan item data ke dalam klaster yang memiliki kesamaan internal tinggi, namun berbeda secara signifikan dengan item data dari klaster lain. Analisis klaster, menurut T. Vo-Van, dkk. (2020), merupakan metode untuk menemukan struktur laten dalam suatu kumpulan data dengan mempartisi elemen-elemen serupa ke dalam kelompok yang sama, dan memisahkan elemen yang berbeda ke dalam kelompok lain.

Algoritma untuk pengelompokan data dibagi menjadi dua kategori besar yaitu algoritma pengelompokan hirarki dan algoritma pengelompokan partisi (Abiodun M. Ikotun dkk, 2021). Algoritma pengelompokan hirarki mempartisi objek data ke dalam kelompok dalam bentuk hirarki baik dalam pendekatan *bottom-up* (metode agglomeratif) atau pendekatan *top-down* (metode divisif). Algoritma pengelompokan partisi menghasilkan satu partisi dari kumpulan data awal sebagai ganti struktur pengelompokan dari dendrogram. klaster dibentuk dengan pendekatan heuristik sementara fungsi optimalisasi kriteria didefinisikan secara *global* pada semua objek kumpulan data atau secara lokal pada bagian dari objek data (Abiodun M. Ikotun, dkk, 2021). Fungsi optimalisasi kriteria pada sekumpulan objek data menggunakan pencarian kombinatorial dari semua nilai yang mungkin untuk mendapatkan nilai optimal. Konfigurasi algoritma pengelompokan partisi

memerlukan spesifikasi nilai *k* yang berbeda yang disediakan pada proses yang berbeda untuk menghasilkan klaster yang optimal. Langkah-langkah kegiatan pengelompokan pola yang khas sebagai berikut (Jain dan Dubes, 1988):

1. Representasi pola (opsional termasuk ekstraksi fitur dan/atau pilihan)
2. Definisi pola kedekatan ukuran yang sesuai dengan domain data
3. Pengelompokan atau Klasterisasi
4. Abstraksi data (jika diperlukan)
5. Penilaian output (jika diperlukan)

Gambar 2.6 menampilkan urutan tipikal dari tiga langkah pertama kegiatan pengelompokan, termasuk jalur umpan balik keluaran proses pengelompokan yang dapat mempengaruhi ekstraksi fitur dan komputasi kesamaan selanjutnya.

Gambar 2.6 Tahapan pengelompokan data

Pemilihan fitur adalah proses mengidentifikasi bagian yang paling efektif dari fitur asli untuk digunakan dalam pengelompokan. Representasi pola mengacu pada jumlah kelas, jumlah pola yang tersedia, dan jumlah, jenis, dan skala fitur yang tersedia untuk algoritma pengelompokan (A.K. JAIN, dkk, 1999). Ekstraksi fitur adalah penggunaan dari satu atau lebih transformasi dari fitur input untuk menghasilkan fitur baru yang menonjol. Salah satu atau kedua teknik ini dapat digunakan untuk mendapatkan kumpulan fitur yang sesuai untuk digunakan dalam pengelompokan. Kedekatan pola biasanya diukur oleh fungsi jarak yang didefinisikan berpasangan dari pola. Abstraksi data adalah proses penggalian ekstrasi yang sederhana dan ringkas dari kumpulan data. Pada konteks pengelompokan, abstraksi data yang khas adalah sebuah deskripsi ringkas dari setiap klaster (Diday dan Simon, 1976).

Algoritma *K-means* dikategorikan sebagai algoritma pengelompokan partisi (Abiodun M. Ikotun dkk, 2022). *K-means* mempartisi kumpulan data yang

diberikan ke dalam klaster termasuk menemukan kesalahan kuadrat minimum antara berbagai titik data dalam kumpulan data dan rata-rata klaster, kemudian menugaskan setiap titik data ke pusat klaster terdekat. Nilai rata-rata *mean* sebuah objek dalam klaster digunakan untuk menentukan pusat klaster (*centroid*). Langkah utama dalam algoritma *K-means* adalah penugasan *centroid* ke setiap klaster. Penugasan *centroid* untuk setiap klaster dilakukan dengan menerapkan fungsi kesamaan berbasis jarak *Euclidean* ke semua objek dalam klaster tersebut menggunakan persamaan berikut:

$$D(\boldsymbol{x}_i, \boldsymbol{\pi}_i) = \sqrt{\sum_{j=1}^{p} \left(x_{i,j} - \pi_{i,j}\right)^2} \qquad (2.5)$$

Keterangan:

$D(\boldsymbol{x}_i, \boldsymbol{\pi}_i)$: jarak antara vektor $\boldsymbol{x}$ dengan pusat klaster $\boldsymbol{\pi}$ pada kata ke-$\boldsymbol{i}$

$x_{i,j}$: nilai objek data *training* $\boldsymbol{x_i}$ pada variabel ke-$\boldsymbol{j}$

$\pi_{i,j}$: nilai objek data *testing* $\boldsymbol{\pi_i}$ pada variabel ke-$\boldsymbol{j}$

p : banyaknya variabel bebas

Algoritma *K-means* secara acak memilih sejumlah *k centroid* tertentu dari kumpulan data. Setiap titik data jarak dari semua *centroid* yang dipilih dievaluasi, dengan masing-masing ditugaskan ke *centroid* terdekat sebagai anggota gugus *centroid* tersebut. Titik pusat (*centroid*) dari masing-masing *k* klaster diperkirakan sebagai rata-rata dari dokumen pelatihan *d* yang dialokasikan ke klaster tersebut menggunakan rumus berikut:

$$C_k = \frac{1}{n_k} \sum d_i \qquad (2.6)$$

Keterangan:

n_k : jumlah data dalam klaster *k*

d_i : jumlah dari nilai jarak yang masuk dalam masing-masing klaster

Pusat klaster dievaluasi lagi pada setiap penugasan anggota baru ke sebuah klaster. Proses iteratif algoritma *K-means* ini dilakukan sampai keanggotaan klaster stabil.

2.2.7. Metode *Term frequency-Inverse Document Frequency* (*TF–IDF*)

TF-IDF adalah bentuk representasi dokumen yang paling mendasar (Robertson, S., 2004). *TF-IDF* didasarkan pada skema *bag-of-word* sehingga sebuah dokumen dapat diwakili oleh kumpulan kata-kata yang digunakan dalam dokumen tersebut. *TF–IDF* mengasumsikan bahwa jika sebuah kata penting untuk sebuah dokumen, maka kata tersebut seharusnya muncul berulang kali di dokumen tersebut dan jarang muncul di dokumen yang lain. *TF* dikaitkan dengan asumsi sebelumnya sedangkan *IDF* dikaitkan dengan asumsi terakhir. Parameter tf_{ij} didefinisikan sebagai berapa kali kata i muncul dalam dokumen j. Semakin besar nilainya, semakin penting kata tersebut. Pendekatan TF-IDF menyajikan teks dengan ruang tabel yang disetiap fitur dalam teks sesuai dengan satu kata. TF (*Term Frequency*) akan menghitung frekuensi kemunculan sebuah kata dan dibandingkan dengan jumlah seluruh kata yang ada di dalam dokumen, berikut persamaan yang digunakan untuk menghitung TF.

$$tf\ (i) = \frac{freq\ (t_1)}{\sum freq\ (t)} \qquad (2.7)$$

Keterangan:

tf(i) : nilai *Term Frequency* sebuah kata dalam sebuah dokumen

Freq (t_i) : frekuensi kemunculan sebuah kata dalam sebuah dokumen

$\sum freq\ (t)$: jumlah keseluruhan kata dalam dokumen

Parameter df_i adalah jumlah dokumen di mana kata *i* muncul setidaknya sekali; semakin besar nilainya, semakin umum kata tersebut.

$$tdf\ (i) = \log \frac{|D|}{|\{d:t_i \in d\}|} \qquad (2.8)$$

Keterangan:

idf(*i*) : nilai *Inverse Document Frequency* sebuah kata di seluruh isi dokumen.

|*D*| : jumlah seluruh dokumen.

|(*d*: *ti*∈ *d*}| : jumlah dokumen yang mengandung kata (*t*)

Jika kata *i* dapat dianggap penting untuk dokumen *j*, maka kata tersebut harus memiliki *TF* (tf_{ij}) yang besar dan *DF* (df_i) yang kecil. Dengan *IDF* mengambil logaritma rasio jumlah total dokumen dalam korpus terhadap frekuensi dokumen *word* ***i*** dengan *IDF smoothing* agar tidak terbagi nol.

$$\text{TF–IDF } ij = tf\,ij \times \log\left(\frac{\text{N}}{df\ i + 1}\right) \tag{2.9}$$

Keterangan:

tf ***ij*** : Jumlah kata ***i*** muncul pada dokumen *j*

df_i : Jumlah dokumen dengan kata ***i***

2.2.8. Metode *Elbow*

Metode *Elbow* digunakan untuk menentukan jumlah klaster terbaik dengan cara melihat persentase hasil perbandingan antara jumlah klaster yang membentuk siku pada suatu titik. Elbow merupakan metode visual untuk menguji konsistensi jumlah klaster optimal. Jumlah klaster optimal dapat diketahui dengan membandingkan perbedaan *Sum of Square Error* (*SSE*) tiap klaster yang membentuk sudut siku-siku paling ekstrim. Rumus untuk menghitung *Sum of Square Error* (SSE) sebagai berikut.

$$SSE = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in S_k} \left|\left|\boldsymbol{x}_i - \boldsymbol{C}_k\right|\right|^2 \tag{2.10}$$

Keterangan:

k : jumlah klaster

$\boldsymbol{x}_i$: data ke – *i*

C_k : pusat klaster (*centroid*)

Sum of Square Error (*SSE*) merupakan rumus yang digunakan untuk mengukur perbedaan antara data yang diperoleh dengan model perkiraan yang telah dilakukan sebelumnya. *SSE* sering digunakan sebagai acuan penelitian terkait dalam menentukan klaster yang optimal.

2.2.9. Komputasi Awan Terdistribusi

Komputasi awan merujuk pada cara komputasi yang berbeda melalui *Internet* sehingga secara dinamis sumber daya bersama yang diskalakan disediakan sebagai layanan untuk menghindari biaya penyediaan sumber daya yang berlebihan (Peter Mell and Timothy Grance, 2011). Komputasi awan menyusun jaringan besar layanan *virtual*, yaitu, sumber daya perangkat keras dan sumber daya perangkat lunak. Segala jenis layanan milik pusat data dan dikenal sebagai data *farm* (Zhang, J., dkk, 2016).

Komputasi awan dapat diklasifikasikan sebagai Infrastruktur sebagai Layanan (IaaS, *Infrastructure as a Service*), Platform sebagai Layanan (PaaS, *Platform as a Service*), atau Perangkat Lunak sebagai Layanan (SaaS, *Software as a Service*) tergantung pada bagaimana digunakan oleh pengguna akhir (Abdullah Saleh Al-Saleh, dkk, 2019). Komputasi awan juga dapat diklasifikasikan sebagai awan publik, pribadi, atau hibrida tergantung pada metode pengembanganya (Abdullah Saleh Al-Saleh, dkk, 2019). Komputasi awan terdistribusi memiliki kontrol *multi* sumber daya dan strategi komputasi yang lebih baik (Hu Sheng dan Xiaodong Qi., 2021).

2.2.10. *Distributed Hash Table* (DHT)

Distributed hash table (*DHT*) adalah struktur data terdistribusi yang digunakan untuk penyimpanan dan pencarian data secara efisien dalam jaringan peer-to-peer (P2P). *DHT* digunakan oleh sistem jaringan *peer-to-peer* untuk pencarian *query* yang efisien. *DHT* memungkinkan setiap *node* di jaringan menyimpan sebagian kecil data, tetapi tetap dapat menemukan data mana pun dengan cepat melalui algoritma pencarian terdistribusi. Sistem jaringan *P2P* mampu menangani *churn*, kegagalan *node*, kerumunan *flash*, dan menyeimbangkan beban efisien (Praveen

Khethavath dkk, 2013). DHT memetakan key (kunci unik) ke *value* (nilai/data) dan menyimpan pasangan *key-value* secara terdistribusi di antara *node* dalam jaringan.

Distributed Hash Table (DHT) merupakan metode dasar dalam sistem *peer-to-peer* (*P2P*) yang digunakan untuk pendistribusian dan pencarian data secara terdesentralisasi. *DHT* mengandalkan proses *hashing* dan perhitungan jarak logis untuk menentukan lokasi penyimpanan data secara efisien. Terdapat tiga tahapan utama dalam *DHT*, yaitu *Mapping* Data ke ID, Perhitungan Jarak (*Distance Function*), dan Penentuan *Node* Penyimpanan (*Node Target*).

Setiap data yang memiliki kunci unik atau *identifier* akan diubah menjadi *ID* numerik dengan menggunakan fungsi *hash* kriptografi. Proses ini dirumuskan sebagai berikut:

$$ID_{data} = Hash(Key) \tag{2.11}$$

Keterangan:

IDdata : Hasil *hash* dari *key*, merepresentasikan identitas data dalam *DHT*

Hash (*Key*) : Fungsi *hash* kriptografi untuk mengubah *key* menjadi bilangan numerik dalam ruang ID *DHT*

Key : Kunci atau nama unik dari data yang akan disimpan

Setelah ID data diperoleh, *DHT* menghitung jarak antara ID data dengan setiap *Node* ID dalam jaringan. Proses ini menggunakan operasi bitwise XOR untuk menentukan seberapa dekat *node* dengan data. Rumus perhitungan jaraknya adalah:

$$Distance(ID_a, ID_b) = ID_a \oplus ID_b \tag{2.12}$$

Keterangan:

$Distance(ID_a, ID_b)$: Nilai jarak logis antara ID *node* a dan ID data b

ID_a : *Node* ID milik *node* ke-a dalam jaringan *DHT*

ID_b : ID data hasil proses *hashing*

$\oplus$: Operasi XOR bitwise antara dua ID

Hasil perhitungan ini akan digunakan untuk memilih *node* dengan jarak terdekat ke data yang dicari. *DHT* akan memilih *node* penyimpanan (*Node* Target) berdasarkan hasil perhitungan jarak terkecil. *Node* dengan jarak terdekat dianggap paling cocok menyimpan data. Proses ini dirumuskan sebagai berikut:

$$Node_{target} = Min_{Node_i}(Distance(ID_{Node_i}, ID_{data})) \qquad (2.13)$$

Keterangan:

$Node_{target}$: *Node* yang dipilih sebagai penyimpanan data karena memiliki jarak terdekat ke ID data

Min_{Node_i} : Fungsi untuk mencari *node* i dengan jarak terkecil

$Distance(ID_{Node_i}, ID_{data})$: Jarak antara ID node i dengan ID data yang telah dihitung sebelumnya

ID_{Node_i} : ID milik setiap node i di dalam jaringan *DHT*

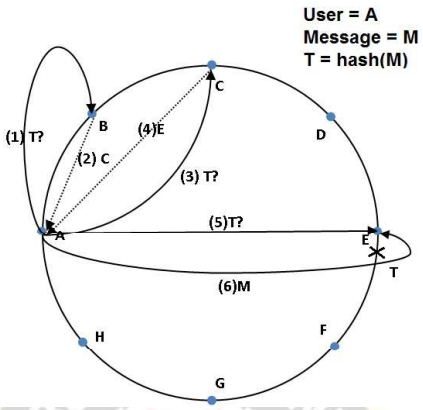
ID_{data} : ID hasil *hash* dari *key* data

IPFS menggunakan Kademlia *DHT* untuk mendistribusikan file dan *metadata* di jaringan *peer-to-peer*. Protokol *Kademlia* menyediakan pencarian dan penerbitan *query* yang efisien dengan total *N peer* dalam jaringan. Setiap *query* mengambil rata-rata *O*(*logN*) *hops* untuk menemukan *peer* yang diminta. Setiap *peer* yang masuk *Kademlia* memiliki ID *node* 160-*bit*. IDnya dibuat acak atau dihasilkan dari alamat IP. Setiap *peer* juga memiliki tabel *routing* yang berisi *subset* dari semua *peer* dalam jaringan (Maymounkov dkk, 2002).

Gambar 2.7 mengilustrasikan Langkah-langkah *routing* dalam tabel *hash* terdistribusi (*DHT*) sebagai berikut:

1. Pengguna *A* menghubungi *B* (*node* terdekat ke *T* di tabel *routing A*)
2. *B* menjawab dengan *C* (simpul terdekat ke *T* yang diketahui *B)*
3. *A* menghubungi *C*
4. *C* membalas dengan *E*

5. *A* menghubungi *E* dan menemukan bahwa *E* adalah simpul terdekat ke *T*
6. *A* mengirim pesan permintaan *M* ke *E*



Gambar 2.7 *Routing* dalam tabel *hash* terdistribusi (*DHT*)