SKRIPSI

IMPLEMENTASI *BLOCKCHAIN* PADA SISTEM *E-VOTING* BERBASIS *MOBILE*

Disusun dan diajukan oleh:

IRFANDI KURNIAWAN ANWAR
D121171504

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
GOWA
2023

LEMBAR PENGESAHAN SKRIPSI

IMPLEMENTASI *BLOCKCHAIN* PADA SISTEM *E-VOTING* BERBASIS *MOBILE*

Disusun dan diajukan oleh

IRFANDI KURNIAWAN ANWAR
D121171504

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana Program Studi Teknik Informatika
Fakultas Teknik Universitas Hasanuddin
Pada tanggal 1 Agustus 2023
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping,
Dr. Eng. Ady Wahyudi Paundu S.T., M.T.	Adnan, S.T., M.T., Ph. D.
NIP. 197503132009121003	NIP. 197404262005011002

Ketua Program Studi,

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

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini ;

Nama	:	Irfandi Kurniawan Anwar
NIM	:	D121171504
Program Studi	:	Teknik Informatika
Jenjang	:	S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

IMPLEMENTASI *BLOCKCHAIN* PADA SISTEM *E-VOTING* BERBASIS *MOBILE*

Adalah karya tulisan saya sendiri dan bukan merupakan pengambilan alihan tulisan orang lain dan bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Semua informasi yang ditulis dalam skripsi yang berasal dari penulis lain telah diberi penghargaan, yakni dengan mengutip sumber dan tahun penerbitannya. Oleh karena itu semua tulisan dalam skripsi ini sepenuhnya menjadi tanggung jawab penulis. Apabila ada pihak manapun yang merasa ada kesamaan judul dan atau hasil temuan dalam skripsi ini, maka penulis siap untuk diklarifikasi dan mempertanggungjawabkan segala resiko.

Segala data dan informasi yang diperoleh selama proses pembuatan skripsi, yang akan dipublikasi oleh Penulis di masa depan harus mendapat persetujuan dari Dosen Pembimbing.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan isi skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Gowa, 1 Agustus 2023

Yang Menyatakan

Irfandi Kurniawan Anwar

ABSTRAK

IRFANDI KURNIAWAN ANWAR. *Implementasi Blockchain Pada Sistem E-Voting Berbasis Mobile* (dibimbing oleh Ady Wahyudi Paundu dan Adnan)

Sebagai negara demokrasi terbesar di dunia, Indonesia sering melakukan pemungutan suara. Namun, pada prakteknya terdapat beberapa permasalahan yang menonjol seperti, kekurangan logistik pemilihan, kotak suara yang tidak tersegel dan surat suara yang tertukar.

Untuk mengatasi beberapa permasalahan pemilu konvensional seperti kotak suara yang tidak tersegel dan surat suara yang tertukar dapat dilakukan dengan mengimplementasikan teknologi *blockchain* pada sistem *e-voting. Blockchain* merupakan sebuah buku besar bersama yang tidak dapat diubah, mempermudah pencatatan transaksi dan pelacakan aset dalam jaringan. Sistem *blockchain* menggunakan basis data terdistribusi, dimana setiap *block* menyimpan kode *hash* dari *block* sebelumnya, membentuk rantai yang terhubung dan disimpan pada setiap kode dalam jaringan *peer-to-peer*.

Salah satu cara untuk mengintegrasikan sistem *e-voting* dengan *blockchain* ialah dengan menggunakan *smart contract*. *Smart contract* adalah program komputer yang dirancang untuk mengeksekusi, mengelola, dan menegosiasikan kontrak secara otomatis di dalam jaringan *blockchain*. *Smart contract* memungkinkan pengguna untuk melakukan transaksi *peer-to-peer* tanpa melibatkan pihak ketiga dan dapat digunakan untuk berbagai aplikasi seperti *e-voting*.

Setelah dilakukan penelitian, mengintegrasikan sistem *e-voting* dan teknologi *blockchain* dapat dilakukan dengan menggunakan *smart contract*. Hasilnya sistem *e-voting* dapat lebih transparan dimana pemilih dan penyelenggara dapat memastikan kebenaran voting, dalam hal *dependability* sistem juga dapat diandalkan dikarenakan setiap suara yang masuk akan terekam secara permanen pada *ledger*, selanjutnya pada hal *anonymity* penyelenggara *e-voting* dapat memberikan tingkat anonimitas yang lebih tinggi bagi para pemilih. Akan tetapi perlu diperhatikan bahwa *smart contract* memiliki beberapa *vulnerability* yang dapat mengakibatkan sistem dapat diretas. Oleh karena itu, penting bagi peneliti untuk menguji *smart contract* sebelum digunakan.

Kata Kunci: *E-Voting, Blockchain, Smart Contract*

ABSTRACT

IRFANDI KURNIAWAN ANWAR. *Implementation of Blockchain in Mobile-Based E-voting System* (supervised by Ady Wahyudi Paundu and Adnan)

As the world's largest democracy, Indonesia often conducts election. However, in practice, there are several prominent issues such as a lack of election logistics, unsealed ballot boxes, and misplaced ballots.

To address several issues in conventional elections such as unsaled ballot boxes and misplaced ballot paper, *blockchain* technology can be implemented in *e-voting* systems. *Blockchain* is a shared, immutable ledger that simplifies transaction recording and asset tracking within a network. The *blockchain* system uses a distributed database, where each *block* stores the *hash* code of the previous *block*, forming a connected *chain* stored on each code in a *peer-to-peer* network.

One way to integrate *e-voting* systems with *blockchain* is by using *smart contracts. Smart contract* are computer programs designed to execute, manage, and automatically negotiate contract on the *blockchain* network. *Smart contract* enable users to conduct *peer-to-peer* transaction without involving third parties and can be used for various applications such as *e-voting*.

After conducting research, integrating *e-voting* system and blockchain technology can be achieved using smart contracts. The result is that the e-voting system can become more transparent, allowing voters and organizers to ensure the accuracy of the voting process. In terms of dependability, the system can be reliable as each vote is permanently recorded on the ledger. Furthermore, concerning anonymity, the *e-voting* organizers can provide a higher level of anonymity for the voters. However, it should be noted that *smart contracts* have certain vulnerabilities that could lead to system breaches. Therefore, it is important fo researchers to thoroughly test *smart contract* before implementation.

Keywords: *E-Voting, Blockchain, Smart Contract*

DAFTAR ISI

LEMBAR PENGESAHAN SKRIPSI ..... Error! Bookmark not defined.
PERNYATAAN KEASLIAN..... Error! Bookmark not defined.
ABSTRAK iv
ABSTRACT vv
DAFTAR ISI v
DAFTAR GAMBAR vi
DAFTAR TABEL vii
DAFTAR SINGKATAN DAN ARTI SIMBOL viii
DAFTAR LAMPIRAN ix
KATA PENGANTAR xii

BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan Penelitian/Perancangan 3
1.4 Manfaat Penelitian/Perancangan 3
1.5 Ruang Lingkup/Asumsi perancangan 3

BAB II TINJAUAN PUSTAKA 4
2.1 *E-Vote* 4
2.2 *Blockchain* 5
2.3 *Smart Contract* 12
2.4 Ethereum 18
2.4.1 *Account* 22
2.4.2 Transaksi 23
2.4.3 *Block* 24
2.4.4 Eksekusi Transaksi 26
2.4.5 Arsitektur *Blockchain* 26

BAB III. METODE PENELITIAN/PERANCANGAN 28
3.1 Lokasi Penelitian 28
3.2 Instrumen Penelitian 28
3.3 Diagram Alur Kerja 29
3.4 Gambaran Umum Sistem 30
3.4.1 *Activity Diagram Sistem* 31
3.4.2 Desain *Smart Contract* 35
3.4.3 Detail Perancangan Sistem 36
3.4.4 Data Flow Diagram Sistem 37
3.5 Skenario Analisis dan Pengujian 38
3.5.1 Pengujian Fungsionalitas Sistem 38
3.5.2 Analisis Pengaruh Penerapan *Blockchain* 39
3.5.3 Analisis *vulnerability smart contract* 39

BAB IV. HASIL DAN PEMBAHASAN 38
4.1 Pengujian Fungsionalitas Sistem 38
4.1.1 Pembentukan *E-Voting* 38

4.1.2 Pemberian Suara Pada *E-Voting* 38
4.1.3 Perhitungan Suara Pada *E-Voting* 39
4.2 Pengaruh Penerapan *Blockchain* Terhadap Data yang Dihasilkan 40
4.2.1 Pengaruh Penerapan *Blockchain* Pada Transparansi Sistem..... 40
4.2.2 Pengaruh Penerapan *Blockchain* Terhadap *Anonymity* Pemilih 41
4.2.3 Pengaruh Penerapan *Blockchain* Terhadapt *Dependability* Sistem 42
4.2.4 Pengaruh Penerapan *Blockchain* Terhadap *Eligibility* Sistem..... 44
4.3 Analisis *Vulnerability Smart Contract*..... 45
4.3.1 Mythril 45
4.3.2 Slither..... 50

BAB V. KESIMPULAN DAN SARAN..... 52
5.1 Kesimpulan 52
5.2 Saran..... 53

DAFTAR PUSTAKA..... 54

DAFTAR GAMBAR

Gambar 2.1 Ilustrasi *Blockchain* 6
Gambar 2.2 Detail *Block* pada *Blockchain* 7
Gambar 2.3 Komputasi *Block Hash* 8
Gambar 2.4 Arsitektur *Decentralized Application* 13
Gambar 2.5 Proses Pencatatan Data *Blockchain* 19
Gambar 2.6 *Block Header* 25
Gambar 3.1 Lokasi penelitian pada Laboratorium Ubquitous Computing & Networking Departemen Teknik Informatika, Universitas Hasanuddin 28
Gambar 3.2 Diagram Alur Kerja 29
Gambar 3.3 Gambaran Umum Sistem 30
Gambar 3.4 *Use Case Diagram* Sistem 31
Gambar 3.5 *Activity Diagram* Penyelenggaraan Pemilihan 32
Gambar 3.6 *Activity Diagram* Pemungutan Suara 33
Gambar 3.7 *Activity Diagram* Perhitungan Suara 35
Gambar 3.8 Pengembangan Aplikasi *Mobile* Sistem *E-Voting* 36
Gambar 3.9 Pengembangan *Smart Contract* Sistem *E-Voting* 37
Gambar 3.10 Data Flow Diagram Level 0 38
Gambar 3.11 Data Flow Diagram Level 1 38
Gambar 4.1 Bukti suara pemilih yang tersimpan dalam *ledger* 42
Gambar 4.2 Bukti suara pemilih memilih kandidat nomor index 1 43
Gambar 4.3 *Public key* pada *ledger* 44
Gambar 4.4 Pembentukan *block* baru yang ditandai dengan *hash* 45
Gambar 4.5 Kiri: *block* baru yang menunggu divalidasi oleh *node* yang ada pada jaringan. Kanan: *block* baru yang telah divalidasi oleh 3 *node* 46
Gambar 4.6 Pengujian menggunakan *tools* mythril 47
Gambar 4.7 *Analyzing on-chain contract* menggunakan *tools* mythril 48
Gambar 4.8 *Analyzing on-chain contract* menggunakan *tools* mythril 49
Gambar 4.9 atas: sebelum *modifier visibility* diganti. bawah: setelah *modifier visibility* diganti 51
Gambar 4.10 Hasil analisis Mythril pada *smart contract* yang telah diperbaiki . 52
Gambar 4.11 Pengujian *smart contract* menggunakan *tools* slither 53
Gambar 4.12 atas: sebelum variable ditambahkan *immutable*. bawah : setelah ditambahkan *immutable* pada variable *owner* 54
Gambar 4.13 Analisis *smart contract* setelah ditambahkan *immutable* pada variable *owner* 54

DAFTAR TABEL

Tabel 2.1 Jenis *Blockchain*..... 9
Tabel 2.2 Bidang Penerapan Teknologi *Blockchain*..... 10
Tabel 3.1 *Input* dan *Output* Penyelenggaraan Pemilihan..... 31
Tabel 3.2 *Input* dan *Output* Pemungutan Suara..... 32
Tabel 3.3 *Input* dan *Output* Perhitungan Suara..... 34
Tabel 3.4 *Variable* dan *Function* dari Contract *E-Voting*..... 34
Tabel 4.1 Hasil *Black Box Testing* Pembentukan *E-Voting*..... 39
Tabel 4.2 Hasil *Black Box Testing* Pemberian Suara pada Sistem *E-Voting*..... 40
Tabel 4.3 Hasil *Black Box Testing* Perhitungan Suara pada Sistem *E-Voting*..... 41
Tabel 4.4 *Vulnerability* yang ditemukan Mythril..... 48
Tabel 4.5 Rekomendasi Perbaikan dari *tools* Slither (*Severity Low*)..... 52

DAFTAR SINGKATAN DAN ARTI SIMBOL

Lambang/Singkatan	Arti dan Keterangan
API	*Application Programming Interface*
DBMS	Database Management System
ETH	Ethereum
EVM	*Ethereum Virtual Machine*
E-Voting	*Electronic Voting*
GHOST	*Greedy Heaviest Observed Subtree*
IDE	*Integrated Development Environment*
PoW	*Proof of Work*
SHA-256	*Secure Hash Algorithm* 256-*bit*
TOD	*Transaction Ordering Dependence*
RLP	*Recursive Length Prefix*
TPS	Tempat Pemungutan Suara

DAFTAR LAMPIRAN

Lampiran 1 57
Tampilan halaman untuk penyelenggara 57
Tampilan halaman *login* dan *voting* untuk pemilih 57
Lampiran 2 58
Kode *smartcontract* sebelum dianalisis *vulnerability* 58
Kode *smartcontract* setelah dilakukan analisis *vulnerability* 59
Lampiran 3 61
Kode untuk memanggil *function* pada *smartcontract* menggunakan library Web3.Swift 61
Kode untuk melakukan transaksi atau voting pada kandidat pertama 67
Kode untuk melakukan transaksi atau voting pada kandidat kedua 73
Lampiran 4 79
Kode untuk menampilkan dan menghubungkan *wallet* dengan sistem *e-voting* .. 79
Kode untuk menampilkan *function* yang telah dipanggil dari *blockchain* 80
Kode untuk menampilkan Etherscan (*ledger*) 82

KATA PENGANTAR

Puji syukur penulis hanturkan kepada Allah SWT, atas segala limpahan rahmat, taufiq, hidayah, serta inayah-nya sehingga penulis diberi kesempatan dalam menyelesaikan tugas akhir. Sholawat serta salam penulis hanturkan kepada junjungan Nabi Muhammad SAW sebagai suri tauladan, pelita dalam kegelapan hingga akhir zaman sehingga penulis dapat menyelesaikan penulisan skripsi yang berjudul **"Implementasi *Blockchain* pada *E-Voting* Berbasis *Mobile*"** guna memenuhi salah satu persyaratan dalam menyelesaikan jenjang Strata-1 di Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Penulis menyadari sepenuhnya bahwa skripsi ini masih jauh dari kesempurnaan karena menyadari segala keterbatasan yang ada. Dalam penulisan skripsi ini penulis menghadapi berbagai kendala dan masalah, namun itu tidak menurunkan semangat penulis dengan usaha yang maksimal dan kemampuan yang Allah berikan kepada penulis serta bantuan dan dukungan dari berbagai pihak, maka penulisan skripsi ini dapat selesai. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih kepada:

1. Allah SWT, zat yang maha segalanya sebagai penolong di setiap langkah hidup penulis.
2. Kedua orang tua penulis, Bapak Anwar Nuhung dan Ibu Maryuni yang selalu memberikan kasih sayang, nasehat, motivasi, dukungan, dan doa kepada penulis.
3. Bapak Dr. Eng. Ady Wahyudi Paundu, S.T., M.T. selaku pembimbing utama dan Bapak Dr. Adnan, S.T., M.T. selaku pembimbing pendamping yang senantiasa menyediakan waktu, tenaga, pikiran, dan perhatian yang luar biasa dalam mengarahkan penulis dalam penyusunan tugas akhir ini.
4. Bapak Robert, Bapak Zainuddin dan Ibu Yuanita serta segenap staf Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah membantu kelancaran penyelesaian tugas akhir penulis.

5. Segenap Dosen dan Staf Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah banyak membantu semasa perkuliahan hingga penyelesaian tugas akhir penulis.
6. Keluarga saya secara umum yang telah memberikan banyak sumbangsih baik secara materi maupun non-materi.
7. Anak Teknik 2017 yang menemani penulis dalam mengarungi suka dan duka dalam menjalani dunia perkuliahan dan organisasi di Fakultas Teknik Universitas Hasanuddin.
8. Saudara seperjuangan penulis RECOGN17ER yang telah menemani dan mendukung perjalanan penulis sekaligus tempat berbagi keluh kesah selama menjadi mahasiswa teknik di Departemen Informatika Fakultas Teknik Universitas Hasanuddin.
9. Kekasih tercinta penulis Ade Anggreani P yang telah membantu, menemani dan mendukung penulis sehingga dapat menyelesaikan tugas akhir ini.
10. Seluruh pihak yang tidak sempat disebutkan satu persatu yang telah banyak meluangkan tenaga, waktu, dan pikiran selama penyusunan tugas akhir ini.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan sistem digital di era ini semakin pesat, seiring pula dengan penggunaan peralatan elektronik dalam menyelesaikan sesuatu permasalahan yang ada di masyarakat. Dalam menyelesaikan permasalahan menggunakan sistem digital, diperlukan perancangan sistem yang tepat dan bisa memenuhi kebutuhan yang ada. Sebagai negara yang menjunjung tinggi demokrasi dalam menjalankan sistem pemerintahannya, pemungutan suara biasa digunakan dalam pengambilan keputusan seperti memutuskan orang yang akan menjadi pemimpin dalam sebuah organisasi, komunitas, maupun negara.

Sebagai salah satu negara demokrasi terbesar di dunia, masyarakat Indonesia tentu saja sering melakukan kegiatan pemungutan suara. Pada prakteknya, ada beberapa permasalahan yang menonjol pada pelaksanaan pemilu, khususnya pada pemilu serentak 2019, permasalahannya antara lain, penanganan logistik pemilu. Secara nasional, ada 10.520 TPS yang mengalami kekurangan logistik pemilu. Terjadi pula kasus kotak suara yang diterima KPPS tidak tersegel, yaitu terjadi di 6.474 TPS. Selain itu, terdapat pula kasus surat suara yang tertukar antar daerah pemilihan atau antar TPS. Berdasarkan data bawaslu, kasus ini terjadi di 3.411 TPS (Ardipandanto, 2019).

Salah satu cara mengatasi beberapa permasalahan pemilu konvensional seperti kotak suara yang tidak tersegel dan surat suara tertukar adalah dengan menerapkan *blockchain* pada sistem pemilu elektronik. Dalam istilah yang berbeda, *Blockchain* merupakan sebuah buku besar yang digunakan bersama dan tidak dapat diubah, yang mempermudah pencatatan transaksi dan pelacakan aset dalam sebuah jaringan. Sistem *Blockchain* adalah bentuk dari basis data terdistribusi yang berisi informasi yang disimpan dalam *block* data. Setiap *block* menyimpan kode *hash* dari *block* sebelumnya, yang memungkinkan *block-block* ini terhubung dan membentuk rantai yang disimpan pada setiap kode dalam jaringan *peer-to-peer*.

Sistem dan aplikasi yang akan dibangun ini diharapkan akan menjawab beberapa kebutuhan pemungutan suara dengan implementasi dari teknologi *blockchain*. Sistem tersebut akan menjawab *Transparancy,* yaitu data yang disimpan bersifat terbuka untuk publik sehingga meningkatkan keadilan dan kebenaran. *Anonymity,* yaitu hanya *voter* itu sendiri yang tahu informasi mengenai *vote* dan semua *ballot* yang terkumpul tidak ada hubungannya dengan voter. *Depandability,* yaitu setiap *vote* akan dihitung dan tidak dapat diganti, digandakan ataupun dihapus, serta mengeluarkan hasil yang dapat dipercaya. *Eligibility,* yaitu hanya *user* yang terverifikasi dan memiliki hak suara yang dapat membuat *ballot* dan melakukan *vote*. *Verifiability,* yaitu sistem bersifat terbuka untuk dapat diperiksa kebenarannya dari prosedur sistem hingga hasil yang dikeluarkan (Hu dkk, 2019).

Penelitian sebelumnya telah menunjukkan bahwa teknologi *blockchain* dapat digunakan dalam berbagai aplikasi, termasuk pada sistem *e-voting*. Namun, implementasi *blockchain* pada sistem *e-voting* berbasis *mobile* masih terbilang baru dan perlu dikaji lebih lanjut untuk mengetahui keefektifannya dalam meningkatkan keamanan dan integritas data dalam proses pemilihan.

Dalam skripsi ini, penulis akan mengkaji Implementasi *blockchain* pada sistem *e-voting* berbasis *mobile* dan menganalisa keefektifannya dalam meningkatkan keamanan dan integritas data dalam proses pemilihan. Diharapkan hasil penelitian ini dapat memberikan kontribusi yang signifikan dalam pengembangan sistem *e-voting* yang lebih aman dan terpercaya di masa depan.

1.2 Rumusan Masalah

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

1. Bagaimana membangun sistem *E-Voting* berbasis *mobile* dengan mengimplementasikan *blockchain*?
2. Bagaimana pengaruh penerapan *blockchain* pada *E-Voting* terhadap data yang dihasilkan oleh sistem?
3. Bagaimana pengujian *Smart Contract* yang digunakan pada sistem?

1.3 Tujuan Penelitian/Perancangan

Adapun tujuan dari penelitian ini meliputi :

1. Membangun sistem *E-Voting* berbasis *mobile* dengan menerapkan teknologi *blockchain*.
2. Mengetahui pengaruh penerapan *blockchain* terhadap data yang dihasilkan oleh sistem.
3. Menguji keamanan *Smart Contract* yang digunakan pada sistem.

1.4 Manfaat Penelitian/Perancangan

Penelitian ini diharapkan dapat bermanfaat bagi masyarakat pada saat melakukan pemungutan suara, sistem yang akan dibuat mencegah terjadinya manipulasi suara oleh pihak tertentu dan juga sebagai rujukan untuk sistem penyimpanan baru yang bersifat *decentraziled.*

1.5 Ruang Lingkup/Asumsi perancangan

Asumsi perancangan yang ditentukan dalam sistem ini adalah:

1. Uji coba dilakukan dengan eksperimen simulasi model yang akan menghasilkan *prototype.*
2. Pengimplementasian *blockchain* menggunakan Ethereum *Blockchain*.
3. Visualisasi sistem ditampilkan melalui *mobile*.
4. Hanya *penyelenggara* (*owner* dari *smart contract*) yang dapat mendaftarkan *wallet addres voter* untuk mendapatkan hak suara pada pemilihan.
5. Hanya *address* yang terdaftar yang mendapatkan hak suara pada pemilihan.

BAB II
TINJAUAN PUSTAKA

2.1 *E-Vote*

Electronic Voting (E-Vote) adalah sistem pemungutan suara yang menggunakan teknologi informasi dan perangkat elektronik sebagai sarana untuk melakukan kegiatan pemilihan. Teknologi *e-voting* muncul pertama kali di Amerika Serikat pada tahun 1889. Di tahun itu, Jacob H. Myers mematenkan mesin pemilihan umum pertama yang diberi nama *Lever Voting Machine*. Kemudian mesin tersebut disebut dengan *Myers Automatic Boots.* Mesin ini ditujukan untuk mencegah terjadinya penggelembungan suara, mempercepat proses perhitungan suara, dan mengurangi suara yang tidak sah (Habibi dan Achmad, 2018).

Salah satu keuntungan utama *e-voting* adalah kenyamanannya. Pemilih tidak lagi diharuskan melakukan perjalanan ke tempat pemungutan suara, yang bisa menjadi tantangan bagi orang tua, berkebutuhan khusus, atau tinggal di daerah terpencil. Sebalikanya, mereka dapat memberikan suaranya dari kenyamanan rumah mereka sendiri, menggunakan komputer, *smartphone*, atau perangkat elektronik lainnya. Ini tidak hanya menghemat waktu dan uang, tetapi juga membantu meningkatkan partisipasi pemilih.

Namun, *e-voting* juga memiliki sejumlah risiko keamanan. Sistem elektronik rentan terhadap serangan *cyber*, yang dapat mengakibatkan kerusakan atau manipulasi data. Ada juga risiko terhadap kerahasiaan suara, yang dapat terganggu jika sistem tidak diatur dengan benar. Oleh karena itu, penting untuk mengembangkan sistem *e-voting* yang aman dan dapat diandalkan, yang memenuhi standar keamanan yang ketat dan diuji secara teratur.

Untuk memastikan integritas *e-voting*, transparansi dan akuntabilitas harus dijaga. Jaminan bahwa suara dihitung dengan benar dan hasilnya diumumkan secara akurat sangat penting untuk membangun kepercayaan masyarakat dalam proses pemilihan. Oleh karena itu, regulasi dan pengawasan yang ketat diperlukan untuk memastikan integrasi pemilihan.

Dalam rangka untuk memanfaatkan teknologi *e-voting* secara efektif, pemangku kepentingan harus bekerja sama untuk mengatasi tantangan yang terkait dengan penggunaanya. Keamanan, integritas, dan aksesibilitas adalah faktor-faktor kunci yang harus dipertimbangkan dalam pengembangan sistem *e-voting* yang efektif. Dalam jangka Panjang, *e-voting* memiliki potensi untuk meningkatkan partisipasi pemilih dan meningkatkan demokrasi, asalkan diatur dengan baik dan aman.

2.2 *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 System). 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).

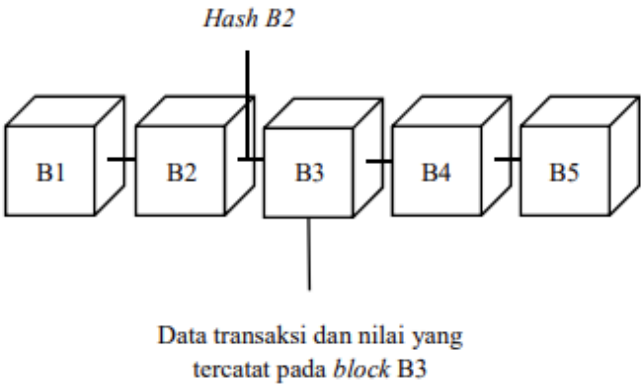
Pencatatan data pada teknologi *blockchain* dilakukan melalui beberapa tahap, yaitu :

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

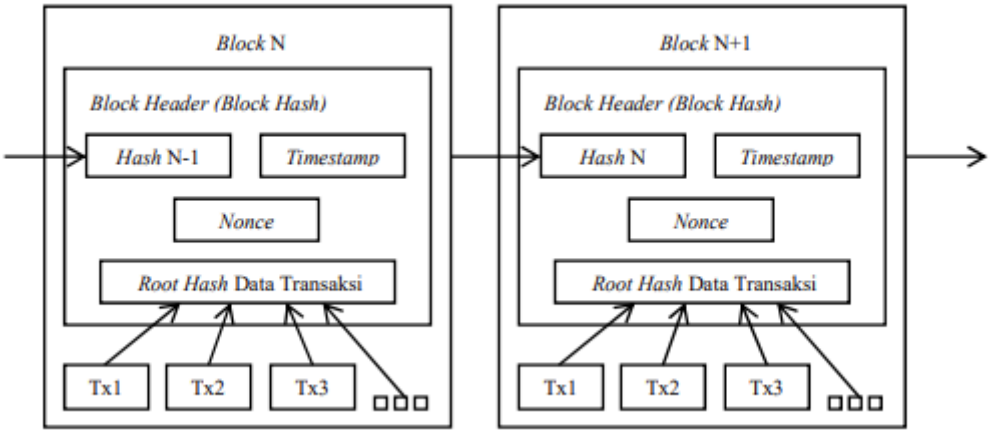
diterapkan. Dalam konsensus *Proof of Work* pembentukan *block* baru ini dikenal dengan istilah *mining,* yang akan menghasilkan nilai *hash* pada *block.*

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 membentuk suatu *block* baru yang terhubung dengan *block* sebelumnya pada rantai *block* menggunakan fungsi *hash* kriptografi, SHA-256. *Block* dibentuk dengan cara menghitung nilai *hash*-nya. Nilai *hash* ini biasa disebut sebagai *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*



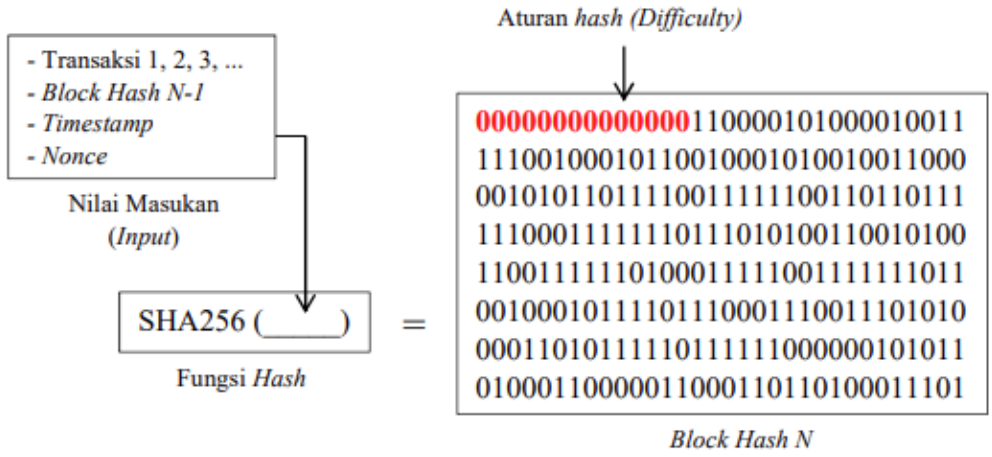
Gambar 2.2 Detail *Block* pada *Blockchain*

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. Data-data ditambahkan kedalam *block* secara demokratis yakni melalui pemungutan suara oleh para *node* yang ada pada jaringan. 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 penyelesaiannya. *Nounce* adalah nilai masukan yang ditentukan oleh *node* dalam melakukan komputasi *block hash*. Penggunaan nilai masukan yang dinamis seperti *nounce* akan dibutuhkan untuk mendapatkan *block 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.

Gambaran komputasi *block hash* dapat dilihat pada Gambar 2.3.



Gambar 2.3 Komputasi *Block Hash*

Fungsi *hash* kriptografi seperti SHA-256 akan menghasilkan nilai *hash* yang sangat acak dan bersifat satu arah. Artinya, nilai *hash* yang dibentuk tidak dapat didekripsi untuk mengetahui nilai masukan yang digunakan dalam proses

penghasilannya. *Block hash* yang memenuhi aturan *difficulty* yang ditentukan hanya bisa diperoleh dengan menjalankan fungsi *hash* berkali-kali dengan menggunakan nilai *nounce* yang berbeda. *Node-node* akan bersaing untuk membentuk *block* dengan mencari *nounce* yang sesuai sehingga *block hash* yang diperoleh memenuhi kriteria *difficulty*. Proses ini memerlukan komputasi yang besar. Oleh karena itu, penemuan *nounce* yang sesuai menjadi bukti bahwa pembuatan *block* membutuhkan komputasi yang besar. *Block* yang terbentuk dengan nilai *nounce* yang sesuai diakui oleh jaringan sebagai *block valid.*

Terdapat setidaknya 4 tipe *blockchain* yang sekarang ada di dunia, yakni *public, permissioned, private* dan *consortium.* Tipe *blockchain* tersebut ditentukan oleh pihak mana yang mengembangkan *blockchain* dan umumnya berpengaruh juga dalam cakupan pemanfaatan *blockchain* tersebut. Jenis-jenis *blockchain* dapat dilihat pada Tabel 2.1.

Table 2.1 Jenis *Blockchain*

Jenis	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

	komersial. (Wijaya dan Oscar, 2018:27).
Consortium Blockchain	*Consortium Blockchain* muncul sebagai konsep arsitektur yang menarik yang memanfaatkan efisiensi dan privasi transaksi *blockchain* pribadi, sambil memanfaatkan tata kelola terdesentralisasi dari *blockchain* publik (Dib, Omar dkk, 2018).

Keuntungan-keuntungan yang didapatkan dengan menerepkan *blockchain* pada suatu sistem antara lain sebagai berikut :

a. Sistem akan bersifat terdesentralisasi, dimana pengelolaan sistem dilakukan tanpa adanya otoritas yang terpusat.
b. Pengguna akan memiliki wewenang untuk berpartisipasi dalam melakukan pengelolaan data yang ada.
c. Data dapat tersedia secara konsisten, lengkap, dan terkini karena tersebar pada setiap *node* yang tergabung dalam jaringan sistem.
d. Dibangun tanpa satu pihak yang memiliki otoritas utama sehingga pengguna dapat memastikan bahwa setiap data akan diproses sesuai dengan mekanisme protokol yang ada.
e. Keandalan data dalam *blockchain* terjaga karena setiap data yang tercatat tidak dapat diubah atau dimanipulasi.
f. Mekanisme enkripsi dari protokol *blockchain* juga memastikan bahwa data yang tercatat terlindungi dengan baik.
g. Dikarenakan sistem *blockchain* dibangun dengan menggunakan jaringan *peer-to-peer* tanpa otoritas pusat, sistem memiliki kekebalan terhadap berbagai jenis serangan *cyber*.

Sejak pertama kali digunakan dalam *cryptocurrency*, minat telah meningkat saat industri dan akademisi mengevaluasi aplikasi dan operasi dari teknologi *blockchain* yang mendasar. Secara khusus, *blockchain* telah dianggap sebagai solusi yang layak untuk banyak kebutuhan aplikasi (Gao, William dan Wei, 2018). Penerapan-penerapan *blockchain* pada bidang tertentu dapat dilihat pada Tabel 2.2.

Tabel 2.2 Bidang Penerapan Teknologi *Blockchain*

Bidang Aplikasi	Penjelasan
Internet of Things (IoT)	Sistem Internet of Things (IoT) yang berskala besar dapat diimplementasikan secara terdesentralisasi melalui jaringan *peer-to-peer* menggunakan teknologi *blockchain*. Desentralisasi yang dihadirkan oleh protokol *blockchain* memberikan perlindungan pada sistem IoT dari serangan umum yang terjadi pada sistem terpusat.
Big Data	*Blockchain* dianggap sebagai solusi yang potensial dalam menangani masalah manajemen data dalam sistem *big data* seperti melindungi data pribadi dan property digital. *Blockchain* menawarkan keandalan dan keamanan yang lebih tinggi dalam proses pencatatan data, yang dapat memberikan manfaat pada sistem *bigdata*.
Cloud & Edge Computing	Mekanisme proteksi data adalah alasan utama mengapa *blockchain* digunakan dalam sistem *cloud* dan *edge computing*. Pencatatan data menggunakan *blockchain* tidak dapat dimanipulasi, sehingga integritas data dalam sistem *cloud/edge computing* dapat dipertahankan.
Manajemen Identitas	Dalam konteks ini, teknologi *blockchain* memberikan kemampuan untuk membentuk identitas menggunakan tanda tangan digital. Tanda tangan digital akan digunakan untuk memverifikasi keaslian suatu identitas.
Finansial & *Cryptocurrency*	Awalnya, pengembangan teknologi *blockchain* dilakukan di bidang finansial. Teknologi ini memiliki potensi untuk menciptakan sistem transaksi yang terdesentralisasi, sehingga kebutuhan untuk pihak ketiga sebagai pengelola transaksi dapat dieliminasi.
Smart contract & Otomatisasi	*Smart contract* merupakan suatu rangkaian kode yang digunakan untuk menjalankan suatu logika

	bisnis. Kode ini dapat diterapkan pada *blockchain* sehingga logika bisnis dapat dijalankan secara otomatis melalui jaringan terdesentralisasi.
Suplay Chain	*Blockchain* dapat mengatasi masalah kepercayaan pada sistem rantai pasok dengan memastikan integritas data. Hal ini penting karena rantai pasok melibatkan banyak entitas dan memerlukan kepercayaan di antara mereka.
Informasi Medis	Untuk menjaga kerahasiaan informasi medis yang bersifat pribadi dan sensitive, teknologi *blockchain* dapat diterapkan sebagai alat pengelola aksesibilitas data. Dengan cara ini, hanya pihak yang berwenang yang dapat melihat dan menyimpan informasi medis tersebut.
Komunikasi & Jaringan	Proteksi pada komunikasi jaringan dapat diberikan oleh *blockchain* dengan menggunakan digital signature untuk memverifikasi kebenaran identitas.

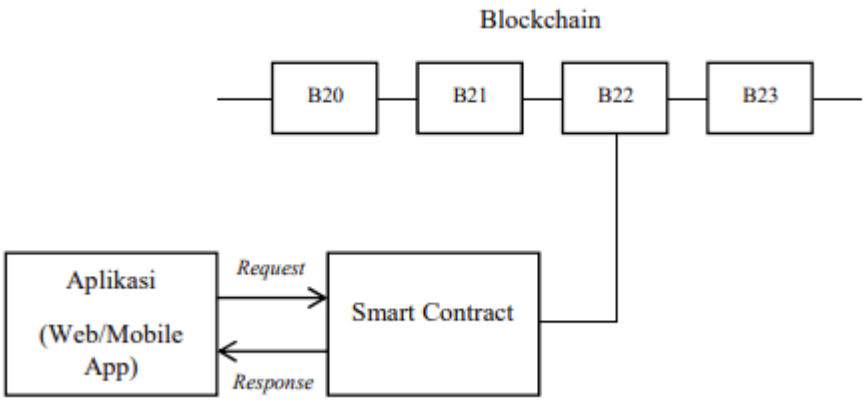
2.3 *Smart Contract*

Kontrak secara umum digunakan untuk membuat protokol persetujuan dalam hubungan antara dua atau lebih pihak individu/kelompok. Proses pembentukan kontrak melibatkan supervisi dari pihak ketiga yang di anggap dapat dipercaya untuk mencegah manipulasi oleh salah satu pihak yang terlibat. Supervisi ini sangat penting untuk memastikan integrasi kontrak. Seiring dengan perkembangan teknologi, terbentuk suatu konsep kontrak baru yang dinamakan dengan *smart contract.*

Smart Contract adalah program yang disimpan di *blockchain* yang berjalan ketika kondisi yang telah ditentukan terpenuhi. *Smart Contract* biasanya digunakan untuk mengotomatisasikan pelaksanaan perjanjian sehingga semua peserta dapat segera mengetahui hasilnya, tanpa keterlibatan perantara atau waktu. *Smart Contract* juga dapat mengotomatisasikan alur kerja, memicu tindakan selanjutnya saat kondisi terpenuhi.

Smart Contract memungkinkan adanya pengembangan yang lebih pada teknologi *blockchain* karena kedua teknologi pada dasarnya diterapkan pada ekosistem yang sama, yaitu dibangun dengan menggunakan jaringan terdesentralisasi. *Blockchain* yang pada awalnya hanya digunakan untuk melakukan proses komputasi sederhana, seperti pencatatan data transaksi, telah dapat dikembangkan untuk melakukan proses komputasi yang lebih kompleks dengan integrasi *smart contract* (Jani 2020).

Kombinasi dari teknologi *blockchain* dan *smart contract* ini dinamakan *decentralized application.* Pada *decentralized application*, *smart contract* digunakan untuk menyimpan logika bisnis (kode program) dan status terkait aplikasi (Antonopoulus dan Gavin, 2018). Arsitektur *decentralized application* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Arsitektur *Decentralized Application*

Decentralized Application terdiri dari tiga komponen utama, yaitu aplikasi klien, *smart contract*, dan *blockchain*. Aplikasi klien dapat berupa aplikasi *web* atau *mobile* yang dapat berkomunikasi dengan *smart contract* melalui API. *Smart Contract* berkerja sebagai *beck-end*, disimpan pada *blockchain* yang tersebar pada setiap *node* dalam jaringan terdesentralisasi. Eksekusi *smart contract* akan berlangsung berdasarkan permintaan dari aplikasi klien dan hasilnya akan diterima oleh klien dalam bentuk data respon (Sayeed, Marco-Gisbert and Caira 2020).

Smart Contract dapat digunakan untuk mengotomatisasikan pelaksanaan perjanjian sehingga semua peserta *e-voting* dapat mengetahui hasil tanpa

keterlibatan perantara. Secara umum, kelebihan yang diberikan *smart contract* antara lain sebagai berikut :

- Otomatisasi, *smart contract* ialah program yang berjalan secara otomatis. Ini memungkinkan pengguna untuk menghindari proses manual yang memakan waktu dan memungkinkan kontrak untuk dieksekusi secara instan saat kondisi yang ditentukan terpenuhi.
- Keamanan, *smart contract* dikodekan dan ditempatkan pada *blockchain*, yang membuatnya sulit untuk dimanipulasi atau diubah tanpa izin. Selain itu, kontrak tidak dapat dihapus atau diubah setelah dieksekusi, memberikan tingkat keamanan yang lebih tinggi daripada kontrak tradisional.
- Biaya rendah, *smart contract* dapat memotong biaya konvensional yang biasanya terkait dengan penggunaan pihak ketiga dalam kontrak tradisional. Pengguna hanya membayar biaya transaksi yang sangat rendah ketika menggunakan *smart contract*, yang lebih efisien dalam jangka panjang.
- Transparansi, semua transaksi yang dieksekusi melalui *smart contract* tercatat dalam *blockchain*, sehingga dapat diakses oleh semua pihak yang terlibat. Hal ini membuat *smart contract* lebih transparan dibandingkan dengan kontrak tradisional,
- Efisiensi, *smart contract* memungkinkan pelaksanaan kontrak secara otomatis dan dalam waktu yang sangat singkat, yang mengurangi biaya dan meningkatkan efisiensi.
- Penghapusan perantara, *smart contract* menghilangkan kebutuhan untuk memiliki perantara dalam transaksi, yang dapat mengurangi biaya dan waktu yang diperlukan untuk menyelesaikan transaksi.
- Fleksibilitas, *smart contract* dapat disesuaikan dengan berbagai jenis kontrak dan dapat digunakan dalam berbagai industri.

2.3.1 Daftar *Vulnerability Smart Contract*

Dalam bagian ini, dijelaskan tujuh *vulnerability* yang dapat menyebabkan dampak serius pada *smart contract*. Jika *vulnerability* berhasil dilaksanakan, maka *smart contract* dapat berperilaku dengan cara yang tidak diinginkan. Dalam hal

ini, pihak yang terkait dengan kesepakatan kontrak dapat mengalami kerugian yang besar. Oleh karena itu, penting bagi pihak-pihak terkait untuk mewaspadai kemungkinan adanya serangan dan mengambil tindakan pencegahan yang sesuai.

a. *Reentrancy*

Reentrancy dianggap sebagai salah satu teknik serangan paling berbahaya dalam *smart contract*. Teknik serangan ini dapat sepenuhnya menghancurkan kontrak atau mencuri informasi berharga. *Reentrancy* dapat terjadi ketika suatu fungsi memanggil kontrak lain melalui panggilan luar. Eksploitasi ini memungkinkan penyerang untuk menjalankan panggilan balik rekursif dari fungsi utama, sehingga terbentuk suatu *loop* yang tidak diinginkan dan diulang banyak kali. Sebagai contoh, ketika sebuah kontrak yang rentan mengandung fungsi mencabut (revoke), kontrak tersebut dapat memanggil fungsi mencabut secara illegal berkali-kali untuk menguras saldo yang tersedia dalam kontrak. Serangan *Reentrancy* pada satu fungsi dan serangan *Reentrancy* antar fungsi adalah dua jenis serangan yang berbeda yang dapat dimanfaatkan oleh penyerang. Eksploitasi ini memungkinkan penyerang untuk menggunakan panggilan eksternal untuk melakukan tugas yang diinginkan (Sayeed, Marco-Gisbert dan Caira 2020).

b. *Integer Overflow* dan *Underflow*

Vulnerability ini relative mudah diinisiasi dan terjadi pada transaksi yang menerima data masukan atau nilai yang tidak sah. *Overflow smart contract* terutama terjadi ketika lebih banyak nilai yang diberikan daripada nilai maksimum. Kontrak-kontrak ini biasanya ditulis dalam bahasa *solidity* yang dapat menangani angka hingga 256 bit, sehingga, penambahan dengan 1 akan menyebabkan *overflow*. *Underflow* pada *smart contract* terjadi dengan cara yang berlawanan dengan *overflow*. Namun, serangan *underflow* lebih mudah dilakukan karena mencapai token yang diperlukan untuk menyebabkan *overflow* seringkali sulit bagi penyerang (Sayeed, Marco-Gisbert dan Caira 2020).

c. *Timestamp Dependance*

Timestamp Dependance adalah *vulnerability* lain yang dapat dimanfaatkan oleh penambang yang tidak jujur. Untuk memperoleh keuntungan, penambang

dapat mengatur ulang *timestamp* selama beberapa detik. *Vulnerability timestamp dependency* terjadi karena pemahaman yang salah dalam menjaga waktu. Ini memungkinkan jaringan Ethereum terlepas dari waktu global yang disinkronkan. Sebagai contoh, *smart contract* menggunakan *timestamp* saat ini untuk menghasilkan angka acak guna menentukan hasil lotere. Karena *smart contract* memperbolehkan penambang menempatkan *timestamp* dalam 30 detik setelah validasi *block*, hal ini memberikan kesempatan lebih banyak bagi penambang untuk mengeksploitasi. Oleh karena itu, hasil dari pembangkit anga acak depat diubah untuk memperoleh keuntungan (Sayeed, Marco-Gisbert dan Caira 2020).

d. *Default Visibility*

Spesifier visibilitas dalam fungsi *solidity* mengontrol cara fungsi tersebut dipanggil. Spesifier visibilitas juga mengambil kontrol saat mengizinkan pengguna untuk memanggil fungsi external oleh kontrak turunan. Implementasi spesifier visibilitas yang tidak tepat dapat menyebabkan efek serius pada *smart contract*. Visibilitas default selalu diatur sebagai *public* untuk fungsi-fungsi, memungkinkan kontrak external untuk memanggil visibilitas ketika fungsi-fungsi tersebut tidak secara eksplisit menyebutkannya. *Vulnerability* ini muncul ketika pengembang lupa untuk mengatur spesifier visibilitas menjadi *private* (Sayeed, Marco-Gisbert dan Caira 2020).

e. *Short Address Attack*

Vulnerability ini terjadi karena kelemahan Ethereum Virtual Machine (EVM). EVM memperbolehkan argument yang tidak presisi dengan padding, yang memungkinkan lawan untuk mengirimkan alamat yang dirancang khusus dan mengakibatkan eksploitasi. Serangan alamat pendek mengikuti strategi serangan yang sama seperti *bug* injeksi SQL. Ketika *underflow* terdeteksi, EVM menyertakan angka nol di akhir alamat untuk memastikan bahwa itu terdiri dari tipe data 256-bit. Namun, penyerang dapat memanfaatkan kerentanan ini dengan menghilangkan angka nol terakhir dari alamat ether. Kerentanan ini adalah *bug* validasi input dan terutama terjadi dari sisi pengirim karena kode generasi transaksi yang lemah (Sayeed, Marco-Gisbert dan Caira 2020).

f. *Delegate Call*

Para pengembang kode cerdas memanfaatkan CALL dan DELEGATE-CALL untuk memodulasi kode yang telah ditulis. Opcode DELEGATE terdiri dari fungsi yang mirip dengan pesan CALL, namun, selain kode yang dieksekusi untuk memanggil kontrak. *msg.sender* dan *msg.value* tidak diubah. Atribut ini memungkinkan pada pengembang untuk menghasilkan kode yang dapat digunakan kembali, meningkatkan kemungkinan eksekusi kode secara tiba-tiba dengan menggunakan DELEGATE-CALL. Fitur DELEGATE-CALL menunjukkan bahwa memungkinkan untuk memperkenalkan kecacatan saat membangun perpustakaan khusus dan juga dapat menyebabkan kerentanan baru. Kerentanan DELEGATE-CALL dapat dihindari dengan mengamati celah pada kontrak *library* dan kontrak pemanggilan, dan juga mengembangkan *library* tanpa status (state-less) jika memungkinkan (Sayeed, Marco-Gisbert dan Caira 2020).

g. *Transaction Ordering Dependence (TOD)*

Transaction Ordering Dependence (TOD) adalah *vulnerability* yang dapat memungkinkan penambang korup untuk memiliki efek serius pada *smart contract*. *Vulnerability* ini adalah *bug* keamanan yang sangat umum pada *smart contract*, yang bergantung pada urutan eksekusi transaksi. Misalnya, *block* yang baru dibuat berisi dua transaksi yang menegakkan *smart contract* yang sama. Plot seperti itu tidak memberikan informasi yang cukup kepada pengguna untuk menentukan status kontrak atau kapan invokasi individu dimulai. Oleh karena itu, ketika *output* dari kedua transaksi bergantung pada urutan, kontrak menghasilkan kerentanan TOD.

Di *blockchain* Ethereum, penambang bertanggung jawab atas mengontrol urutan transaksi, dengan memprioritaskan transaksi dengan *gas* yang lebih tinggi. Oleh karena itu, setiap penambang yang menyelesaikan *block* dapat mempengaruhi urutan transaksi. Kemampuan bagi penambang potensial untuk mempengaruhi urutan transaksi untuk kegiatan illegal adalah hasil dari *Transaction Ordering Dependence* (TOD) (Sayeed, Marco-Gisbert dan Caira 2020).

2.4 Ethereum

Ethereum sering digambarkan sebagai “komputer dunia”. (Antonopoulus, Wood 2018) Ethereum merupakan salah satu bentuk pengembangan dari teknologi *blockchain* yang awalnya diterapkan pada Bitcoin. Platform ethereum memungkinkan *developer* untuk membangun *decentralized application* yang kuat dengan fungsi ekonomi bawaan. Selain memberikan ketersediaan tinggi, kemampuan audit, transparansi, dan netralitas. Hal ini juga mengurangi atau menghilangkan penyensoran dan mengurangi resiko rekanan tertentu. (Antonopoulus, Gavin 2018).

Dibandingkan dengan bitcoin, tujuan Ethereum tidak utama untuk menjadi jaringan pembayaran mata uang digital. Meskipun mata uang digital Ether sangat penting dan diperlukan untuk operasi Ethereum, Ether ditujukan sebagai mata uang pengguna untuk membayar pengguna platform Ethereum sebagai komputer dunia. Berbeda dengan bitcoin, yang memiliki bahasa pemrograman yang sangat terbatas, Ethereum dirancang untuk menjadi *blockchain* yang dapat diprogram dengan tujuan umum yang dijalankan mesin virtual dan mampu mengeksekusi kode dengan kompleksitas acak dan tidak terbatas, di mana bahasa skrip bitcoin secara sengaja dibatasi pada evaluasi sederhana *true/false* dari kondisi pengeluaran sedangkan bahasa yang digunakan Ethereum adalah *turing-complate*, artinya Ethereum dapat dengan mudah berfungsi sebagai komputer bersifat umum. (Antonopoulus, Gavin 2018).

Konsensus dalam jaringan Ethereum didasarkan pada protokol GHOST (*Greedy Heaviest Observed Subtree*) yang dimodifikasi. Ini dibuat untuk mengatasi masalah *block* mati dijaringan. *Block* mati dapat terjadi jika satu kelompok penambang yang digabungkan dalam *mining pool* memiliki lebih banyak daya komputasi daripada yang lain, yang berarti bahwa *block* dari *mining pool* pertama akan memberikan kontribusi lebih daripada yang lain, yang berarti bahwa *block* dari *mining pool* pertama akan memberikan kontribusi lebih banyak pada jaringan, sehingga menciptakan masalah sentralisasi. Protokol GHOST menyertakan *block* mati tersebut dalam perhitungan rantai terpanjang (Vujičić, Dijana dan Siniša, 2018).

Blockchain Ethereum mirip dengan *blockchain* Bitcoin. Perbedaan utamanya adalah bahwa *block* Ethereum tidak hanya berisi nomor *block*, *difficulity, nonce,* dan lain-lain. Tetapi juga daftar transaksi dan *state* terbaru. Untuk setiap transaksi dalam daftar transaksi, *state* baru dibuat dengan menerapkan *state* sebelumnya (Vujičić, Dijana dan Siniša, 2018).

Header *block* dalam *blockchain* Ethereum terdiri dari *hash* Keccak 256-bit dari header *block* induk, alamat penerima biaya penambangan, *hash* dari *root state*, transaksi dan tanda terima, *difficulity*, batas gas saat ini dari *block*, angka yang mewakili total gas yang digunakan dalam transaksi *block*, *timestamp*, *nonce*, dan beberapa *hash* tambahan untuk tujuan verifikasi (Vujičić, Dijana dan Siniša, 2018).

Setiap *node* dalam jaringan Ethereum berjalan di bawah EVM (*Ethereum Virtual Machine*) dan menjalankan instruksinya. *Smart contract* diterjemahkan ke dalam kode EVM dan kemudian dieksekusi oleh *node*. Salah satu bahasa pemrograman yang paling popular untuk menulis *smart contract* adalah *solidity* (Vujičić, Dijana dan Siniša, 2018).

Dalam dunia transaksi keuangan, teknologi *blockchain* telah terbukti efektif dalam memberikan tingkat keamanan yang lebih pasti. Sejauh ini, sudah banyak *cryptocurrency* yang beroperasi di seluruh dunia yang dibangun di atas teknologi *blockchain*, termasuk Ethereum. Dalam konteks sistem *e-voting*, Ethereum memiliki beberapa karakteristik *blockchain* yang dapat memastikan keamanan dan transparansi pada proses pemilihan kandidat, seperti yang dijelaskan berikut ini.

a. 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*.

b. Jaringan *Peer-to-Peer* (P2P)

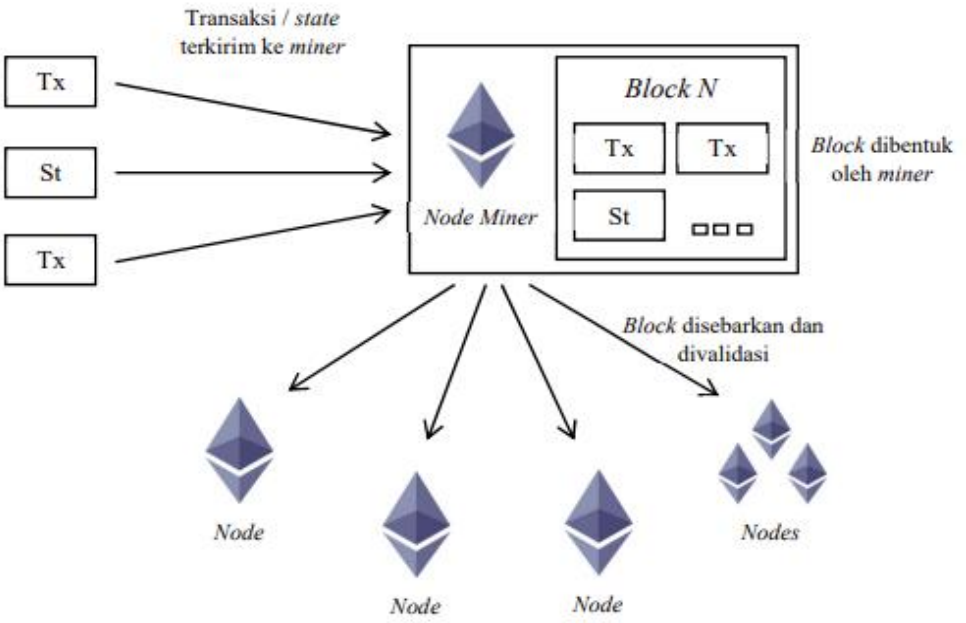
Dalam sistem *blockchain*, setiap *node* akan berkomunikasi dengan yang lainnya melalui jaringan *peer-to-peer* yang didasarkan pada sifat terdesentralisasi *blockchain*. Ketika sebuah *node* mencatat data dengan membentuk *block* baru pada rantai *block* yang ada, *block* baru tersebut akan diumumkan ke *node* lain dalam jaringan. Seiring waktu, rantai *block* yang terbentuk akan tersalin ke seluruh *node* dalam sistem *blockchain*. Hal ini menyebabkan data yang tercatat pada rantai *block* akan sangat sulit hilang atau terhapus karena tersebar di seluruh *node*. Semakin banyak *node* yang tergabung dalam jaringan *blockchain*, semakin sulit untuk hilang atau terhapus. Karakteristik ini membuat data dalam sistem *blockchain* menjadi lebih andal dan terjamin.

c. 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 sehinnga 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.

Gambaran umum proses pencatatan data dan struktur *block* data yang terbentuk dengan mekanisme protokol konsensus *Proof of Work* dapat dilihat pada Gambar 2.5.



Gambar 2.5 Proses Pencatatan Data *Blockchain*

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. Setiap *node* yang jujur akan berusaha untuk membangun *block* yang valid, 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*.

d. Aksesibilitas Publik

Sistem *blockchain* dioperasikan secara terdesentralisasi, di mana setiap *node* memiliki hak akses yang setara terhadap jaringan rantai *block* data. Secara umum,

blockchain bersifat terbuka dan dapat diakses oleh siapa saja yang ingin bergabung sebagai *node* untuk mengelola rantai *block* tersebut. Ketersediaan akses publik ini memberikan keutuhan dan transparansi pada setiap proses pencatatan data yang terjadi dalam sistem *blockchain*.

2.4.1 *Account*

Ethereum *state* terdiri dari *account*, dimana setiap *account* memiliki alamat 20 byte dan transisi status. Kumpulan *state* adalah pemetaan antara alamat dan status *account*. Ethereum mendukung dua jenis akun: *externally owned account* dan *contract account*. Sebuah *account* Ethereum terdiri dari empat bidang: *nonce*, saldo ether, *hash* kode kontrak, dan *root* penyimpanan (Vujičić, Dijana dan Siniša, 2018). Secara esensial, *externally owned account* merujuk pada *account* yang dimiliki oleh individu atau *node*, sedangkan *contract account* mengacu pada identitas suatu *smart contract* yang terdaftar di Ethereum. Hanya *externally owned account* yang dapat memulai pengiriman pesan ke *account* lain dengan cara membuat dan menandatangani transaksi menggunakan tanda tangan digital berupa *private key*. Sebaliknya, *contract account* hanya dapat melakukan transaksi sebagai respons terhadap transaksi yang diterima dari *account* lain.

Account merupakan obyek yang membentuk *state* pada Ethereum. Setiap *account* memiliki alamat (*address*) dan terdiri dari empat komponen *state* (Wood, 2022), berupa :

- *nonce*, merupakan nilai yang merepresentasikan jumlah transaksi yang telah dilakukan oleh suatu *account*.
- *balance*, merupakan jumlah *cryptocurrency* Ethereum (Ether) yang dimiliki oleh suatu *account*.
- *storageRoot*, merupakan *root hash* dari konten-konten milik suatu *account* yang tersimpan. Secara *default*, *storageRoot* memiliki nilai kosong.
- *codeHash*, merupakan *hash* dari kode yang dimiliki oleh suatu *account* yang akan dieksekusi oleh Ethereum Virtual Machine (EVM). Pada *contract account*, *codeHash* merupakan *hash* dari kode *smart contract* yang dibentuk. Sementara pada *externally owned account*, *codeHash* merupakan *hash* dari *string* yang bernilai kosong.

2.4.2 Transaksi

Secara prinsip, transaksi pada Ethereum adalah sebuah perintah yang dibuat oleh *externally owned account*. Transaksi tersebut dibuat dengan menggunakan tanda tangan digital yang berbasis *public/private key* milik *externally owned account.* Setiap transaksi yang dibuat akan dicatat ke dalam *blockchain* Ethereum.

Transaksi dalam Ethereum terbagi menjadi tiga jenis, yaitu *regular transaction*, *contract deployment transaction* dan *execution contract*. *Regular transaction* merupakan sebuah transaksi dari satu *account* ke *account* lainnya. *Contract deployment transaction* merupakan sebuah transaksi tanpa alamat "tujuan", di mana bidang data digunakan untuk kode kontrak. *Execution contract*, merupakan sebuah transaksi yang berinteraksi dengan *smart contract* yang telah diimplementasikan. Dalam hal ini, alamat "tujuan" adalah alamat *smart contract* (Pennella, 2023).

Gas merujuk pada unit yang mengukur jumlah upaya komputasional yang diperlukan untuk menjalankan operasi tertentu di jaringan Ethereum. Karena setiap transaksi Ethereum memerlukan sumber daya komputasional untuk dijalankan, tiap transaksi membutuhkan biaya. Gas merujuk pada biaya yang diperlukan agar berhasil menjalankan transaksi di Ethereum. Proses penentuan biaya transaksi diukur dengan menggunakan dua variabel, yakni *gasPrice* dan *gasLimit*. *GasPrice* merupakan jumlah Ether yang harus dibayarkan oleh pengirim transaksi untuk setiap unit *gas* yang digunakan dalam proses komputasi transaksi. Sementara itu, *gasLimit* adalah jumlah maksimum *gas* yang dapat digunakan dalam proses komputasi transaksi. Kedua variabel ini ditentukan oleh pengirim transaksi. Sebagai contoh, jika pengirim transaksi menentukan *gasLimit* sebesar 50000 dan *gasPrice* sebesar 20 Gwei (1 Ether = 1000000000 Gwei), maka biaya transaksi akan dihitung sebagai 50000 x 20 Gwei = 1000000 Gwei atau 0.001 Ether.

Sebuah transaksi yang dikirim meliputi informasi berikut :

- *recipient*, merupakan alamat yang menerima (jika *account* merupakan *externally owned account*, transaksi akan mentransfer nilai. Jika *account* merupakan *contract creation*, transaksi akan mengeksekusi *smart contract*).

- *signature,* tanda pengenal dari sang pengirim. Ini dihasilkan ketika *private key* pengirim menandatangani transaksi dan mengonfirmasi bahwa pengirim telah mengizinkan transaksi ini.
- *value,* merupakan jumlah ETH yang ditransfer dari pengirim ke penerima.
- *data*, *input* yang akan digunakan untuk mengubah suatu *state* pada Ethereum. Nilai ini hanya ada para transaksi *execution contract*.
- *gasLimit,* jumlah maksimum unit *gas* yang bisa dipakai dalam transaksi. *gasLimit* akan ditentukan oleh pengirim transaksi.
- *gasPrice,* jumlah Ether yang ditentukan oleh pengirim transaksi untuk membayar setiap unit *gas* yang digunakan dalam proses eksekusi transaksi.

2.4.3 *Block*

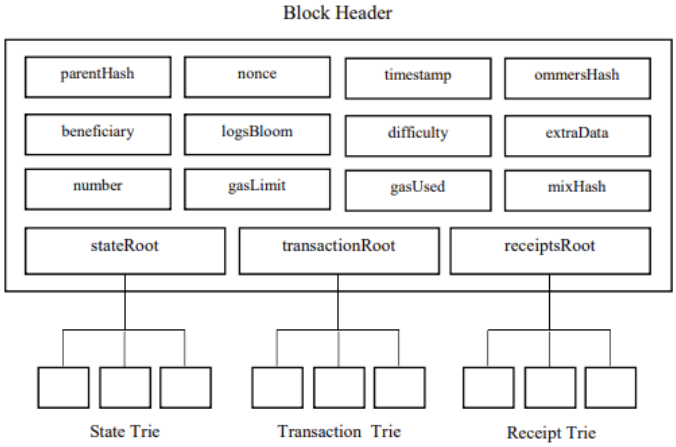
Block merupakan sekelompok transaksi yang dihubungkan satu sama lain dalam sebuah rantai dengan menggunakan *hash block* sebelumnya. Hal ini dimungkinkan karena *hash* tersebut dihasilkan secara kriptografi dari data yang terdapat dalam *block.* Adanya keterkaitan ini dapat mencegah terjadinya kecurangan, 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).

Block dalam Ethereum pada dasarnya mirip seperti *block* dalam Bitcoin, hanya saja berisi tambahan informasi khusus berdasarkan protokol yang ditetapkan. Setiap informasi yang ada akan tergabung dan membentuk *block header*. Informasi-informasi yang terdapat pada *block header* dari suatu *block* yaitu (Wood 2022) :

- *parentHash* : *hash* dari *block header* milik *parent block,* yaitu *block* terakhir sebelum *block* terkait.
- *ommerHash* : *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.

- *transactionRoot* : *hash* dari transaksi yang tercatat pada *block* terkait. *Hash* ini terbentuk menggunakan struktur Merkle Patricia Trie.
- *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.
- *mixHash* : 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 *mixHash*, akan membuktikan bahwa *block* terkait telah dibentuk melalui proses komputasi yang cukup.

Ilustrasi *block header* dari suatu *block* pada Ethereum dapat dilihat pada Gambar 2.6.



Gambar 2.6 *Block Header*

2.4.4 Eksekusi Transaksi

Eksekusi transaksi adalah proses yang menentukan transisi *state(t)* ke *state(t+1)* (Wood 2019). Dalam asumsi ini, setiap transaksi harus melewati serangkaian uji validitas intrinsik sebelum dapat dieksekusi. Uji validitas ini mencakup hal-hal berikut :

- Transaksi harus memiliki format yang sesuai. Format *Recursive Length Prefix* (RLP) merupakan format data yang diterima oleh Ethereum.
- *Signature* transaksi yang valid.
- *Nonce* transaksi yang valid.
- *gasLimit* transaksi harus memiliki jumlah yang lebih besar atau sama dengan jumlah *gas* yang terpakai dalam eksekusi komputasi transaksi.
- Jumlah Ether yang dimiliki oleh pengirim transaksi dapat menutupi biaya pemakaian *gas* yang terbentuk berdasarkan perhitungan *gasLimit* dan *gasPrice* yang ditentukan.

Setelah semua persyaratan terpenuhi, eksekusi transaksi dapat dimulai. Langkah-langkah yang umum dilakukan selama eksekusi transaksi adalah sebagai berikut :

1. Biaya transaksi berdasarkan *gasLimit* dan *gasPrice* yang ditetapkan. Ether yang dikirim oleh pengirim transaksi akan dikurangi berdasarkan biaya transaksi yang dihitung.
2. Jumlah gas yang akan digunakan selama proses komputasi transaksi diinisialisasi.
3. Komputasi yang dibutuhkan akan dieksekusi pada Ethereum Virtual Machine (EVM) oleh *miner*.
4. Setelah semua komputasi yang diperlukan oleh transaksi telah selesai, biaya *gas* yang tidak terpakai akan dikembalikan ke pengirim transaksi, sedangkan biaya *gas* yang telah terpakai akan diterima oleh *miner*.
5. *State* baru dan *log* yang berisi catatan transaksi yang terjadi akan dibuat.

2.4.5 Arsitektur *Blockchain*

Arsitektur *blockchain* Ethereum mirip dengan *blockchain* Bitcoin, tetapi memiliki perbedaan dalam hal informasi yang tersimpan pada *block* dan algoritma

yang digunakan dalam pembentukan *Proof of Work*. Selain transaksi, *block* Ethereum juga menyimpan *state* terbaru. Algoritma *Proof of Work* yang digunakan pada Ethereum disebut *Ethash*.

Algoritma *Ethash* secara formal didefinisikan pada persamaan 2.1.

$$(m, n) = PoW(HN, Hn, \mathrm{d}) \qquad (2.1)$$

Pada persamaan 2.1, ***m*** merupakan *mixHash*, ***n*** adalah *nonce*, ***HN*** diartikan sebagai *block header* dari *block* baru yang terbentuk (tanpa komponen *nonce* dan *mixHash*), ***Hn*** sebagai *nonce* dari *block header*, serta **d** yaitu DAG (*data set* besar) (Wood, 2022).

Pada dasarnya, algoritma *Ethash* akan mencari nilai *mixHash* dan *nonce* yang cocok dengan tingkat kesulitan yang ditetapkan dalam proses pembentukan *block*. Kedua nilai ini berfungsi sebagai bukti bahwa pembentukan *block* melalui proses komputasi yang intensif (*Proof of Work*).