BAB 2
LANDASAN TEORI

2.1. Perbandingan dengan Penelitian Terdahulu

Adapun hasil penelitian terdahulu akan dijadikan sebagai teori pendukung dalam melakukan penelitian. Penelitian terdahulu yang dijadikan referensi pada penelitian ini dapat dilihat pada Tabel 2.1.

Tabel 2.1. Perbandingan dengan Penelitian Terdahulu

No	Peneliti	Judul	Persamaan	Perbedaan
1	Nizamuddin dkk., 2018.	*IPFS-Blockchain-Based Authenticity of Online Publications*	Input: - Jaringan *blockchain* yang digunakan adalah Ethereum. - Penyimpanan gambar yang digunakan adalah IPFS.	Output: - Studi kasus yang digunakan yaitu publikasi buku *online*. Hutomo: - Studi kasus yang digunakan yaitu penggalangan dana.
2	Saadat dkk., 2019.	*Blockchain Based Crowdfunding Systems in Malaysian Perspective*	Input: - Jaringan *blockchain* yang digunakan adalah Ethereum. - Studi kasus yang digunakan yaitu penggalangan dana.	Output: - Tampilan *client* divisualisasikan dengan aplikasi berbasis *website*. - Objek penelitian yaitu otomatisasi sistem penggalangan dana. Hutomo: - Tampilan *client* divisualisasikan dengan aplikasi berbasis *android*. - Objek penelitian yaitu *gas smart contract*.
3	Ainun Fajar, 2020.	*Quality of Service Ethereum Blockchain* berbasis IPFS untuk Validasi Ijazah Sekolah.	Input: - Jaringan *blockchain* yang digunakan adalah Ethereum. - Penyimpanan gambar yang digunakan adalah IPFS. - Objek penelitian *gas smart contract.*	Output: - Studi kasus yang digunakan yaitu Validasi Ijazah Sekolah. Hutomo: - Studi kasus yang digunakan yaitu penggalangan dana.

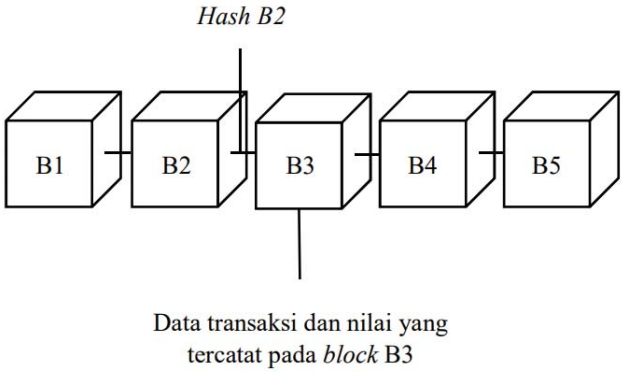
Tabel 2.1 (Lanjutan)

No	Peneliti	Judul	Persamaan	Perbedaan
4	Hoffman dkk., 2021.	*Bountychain: Toward Decentralizing a Bug Bounty Program with Blockchain and IPFS.*	Input: - Jaringan *blockchain* yang digunakan adalah Ethereum. - Penyimpanan gambar yang digunakan adalah IPFS.	Output: - Studi kasus yang digunakan yaitu Program *Bug Bounty*. Hutomo: - Studi kasus yang digunakan yaitu penggalangan dana.
5	Aprialim dkk., 2021.	Penerapan *Blockchain* Dengan Integrasi *Smart Contract* Pada Sistem *Crowdfunding*.	Input: - Jaringan *blockchain* yang digunakan adalah Ethereum.	Output: - Data penggalangan dana yaitu *title*, *fund goal*, *duration* dan *description*. - Objek penelitian yaitu *blockchain* untuk membuat aplikasi *decentralized*. Hutomo: - Data penggalangan dana yaitu gambar, judul, deskripsi, dana yang dibutuhkan dan waktu aktif penggalangan dana. - Objek penelitian yaitu *gas smart contract*.

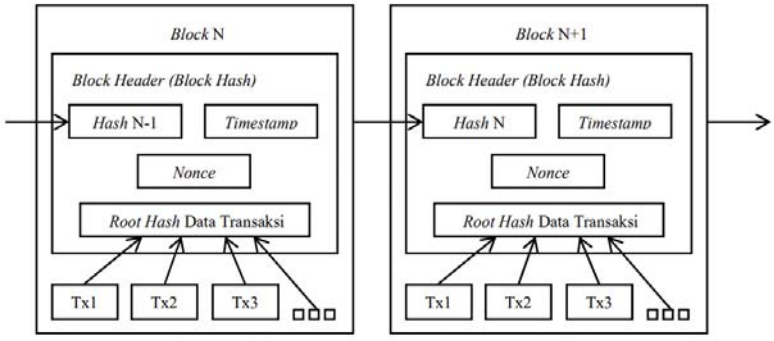
2.2. *Blockchain*

Teknologi *blockchain* awalnya dikembangkan dalam bentuk Bitcoin (Nakamoto, 2008), sebagai sistem pembayaran elektronik dengan arsitektur *peer-to-peer* yang bersifat terdesentralisasi tanpa adanya institusi finansial yang bertindak sebagai pengatur jalannya transaksi. *Blockchain* merupakan salah satu bentuk dari *Distributed Ledger Technology* (DLT), dimana teknologi ini bersifat terdesentralisasi dan memiliki protokol konsensus yang digunakan untuk mencapai kesepakatan bersama dalam proses pengelolaan basis data yang ada.

Pada *blockchain* bitcoin, transaksi yang akan dilakukan oleh suatu *node* menggunakan tanda pengenal atau *digital signature* yang dimiliki, *node* tersebut selanjutnya akan mengumumkan transaksi yang terjadi ke jaringan. *Node* lain kemudian akan menerima pengumuman-pengumuman transaksi yang terjadi dan menggabungkannya dengan membentuk *block* dengan mekanisme protokol *Proof of Work* (PoW). Mekanisme PoW akan membentuk suatu *block* baru yang terhubung dengan *block* terakhir pada rantai *block* menggunakan fungsi *hash SHA-256*. *Block* dibentuk dengan cara menghitung nilai *hash* yang dimilikinya. Nilai *hash* ini biasa disebut sebagai *block hash* atau *block header*. *Block hash* didapatkan melalui komputasi fungsi *hash* dari data transaksi yang tergabung dalam *block* dan beberapa data khusus seperti *timestamp*, *nonce* dan *block hash* dari *block* terakhir yang sebelumnya terbentuk pada rantai *block*. Hubungan antara suatu *block* terbentuk dari nilai *hash* terakhir dengan data masukan dalam pembentukan *block* 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*

Teknologi *blockchain* terus berkembang dan telah dikenal secara umum sebagai *framework* atau kerangka kerja dalam pengembangan sistem yang bersifat terdesentralisasi (Gao dkk., 2018). Setiap pengembangan sistem yang bersifat terdesentralisasi menerapkan konsep *blockchain* yang disesuaikan berdasarkan kebutuhan yang diperlukan. Beberapa kebutuhan ini meliputi jenis *protocol consensus* yang digunakan, implementasi konsep teknologi, mekanisme pencatatan data dan lain sebagainya.

Terdapat beberapa jenis *blockchain* yang dibedakan berdasarkan tiga aspek, yaitu aksesibilitas data, partisipasi *node* dan fungsionalitas (Shrivas, 2019). Jenis-jenis *blockchain* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Jenis Blockchain

Jenis	Aspek	Penjelasan
Public blockchain	Aksesibilitas data	Jenis *blockchain* ini memperbolehkan setiap individu atau kelompok untuk bergabung sebagai *node* dalam melakukan pembacaan dan pencatatan data (Lin & Liao, 2017).
Consortium blockchain	Aksesibilitas data	Jenis *blockchain* ini bersifat tertutup, tetapi individu tertentu yang tergabung dalam *consortium* dapat berpartisipasi sebagai *node* dalam melakukan pembacaan dan pencatatan data (Lin & Liao, 2017).
Private blockchain	Aksesibilitas data	Jenis *blockchain* ini bersifat tertutup dan hanya ada satu pihak atau *node* saja yang dapat melakukan pembacaan dan pencatatan data (Lin & Liao, 2017).
Permissionless blockchain	Partisipasi node	Jenis *blockchain* ini tidak memiliki protokol perizinan yang harus dipenuhi oleh suatu pihak untuk berpartisipasi sebagai *node*.

Tabel 2.3 (Lanjutan)

Jenis	Aspek	Penjelasan
Permissioned blockchain	Partisipasi *node*	Jenis *blockchain* ini memiliki protokol perizinan yang harus dipenuhi oleh suatu pihak untuk dapat berpartisipasi sebagai *node*.
Stateless blockchain	Fungsionalitas	Jenis *blockchain* ini hanya dapat menjalankan logika komputasi sederhana seperti pencatatan data transaksi (Hileman & Rauchs, 2017).
Stateful blockchain	Fungsionalitas	Jenis *blockchain* dapat menjalankan logika komputasi yang lebih kompleks daripada hanya sekedar komputasi pencatatan data. Contoh komputasi kompleks ini seperti pemrosesan *state* berdasarkan logika bisnis yang ada dalam suatu sistem (Hileman & Rauchs, 2017).

2.3. *Smart Contract*

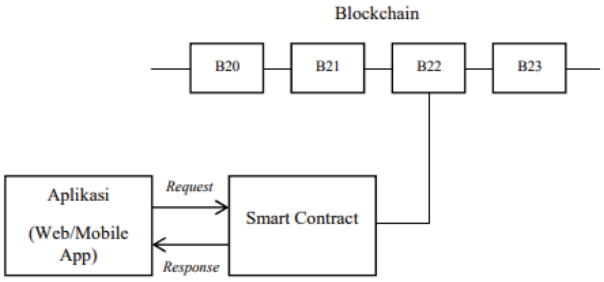
Kontrak pada dasarnya merupakan cara untuk membentuk kesepakatan persetujuan terhadap suatu hal (Szabo, 2018). Kontrak secara umum digunakan untuk keperluan dalam membangun protokol persetujuan terhadap suatu hubungan yang dibentuk oleh dua pihak individu/kelompok atau lebih. Kontrak akan dibentuk dengan bantuan supervisi dari pihak ketiga yang dianggap terpercaya. Supervisi ini sangat penting dimiliki untuk menghindari adanya manipulasi oleh salah satu pihak terhadap kontrak yang dibentuk.

Seiring dengan perkembangan teknologi, terbentuk suatu konsep kontrak baru yang dinamakan dengan *smart contract*. *Smart contract* pada dasarnya merupakan perangkat lunak yang berisi protokol kesepakatan dan hubungan antara dua pihak atau lebih yang dikelola menggunakan sistem terdesentralisasi. Pengawasan terhadap kesepakatan dan hubungan yang terbentuk akan dilakukan

oleh semua pihak yang tergabung dalam jaringan berdasarkan protokol konsensus sistem sehingga kebutuhan akan supervisi dari suatu pihak ketiga tidak diperlukan.

Smart contract merupakan sebuah program kecil yang ada di dalam *blockchain* yang sudah terprogram dan akan berjalan secara otomatis jika syarat kondisi yang sudah ditetapkan terpenuhi (Gatteschi dkk., 2018). Saat melakukan transaksi, setiap *node* yang terlibat dalam jaringan akan ikut mengeksekusi *smart contract*. Maka dari itu, setiap *node* di dalam *blockchain* harus menyetujui input, output, dan kondisi yang ada di dalam *smart contract* (Azzi dkk., 2019).

Smart contract memungkinan adanya pengembangan yang lebih pada teknologi *blockchain* karena keduanya dibangun menggunakan ekosistem yang sama, yaitu dengan jaringan terdesentralisasi. *Blockchain* yang pada awalnya hanya digunakan untuk melakukan proses komputasi sederhana, seperti pencatatan data transaksi, tetapi sekarang telah 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* (DApp). Arsitektur *decentralized application* dapat dilihat pada Gambar 2.3.



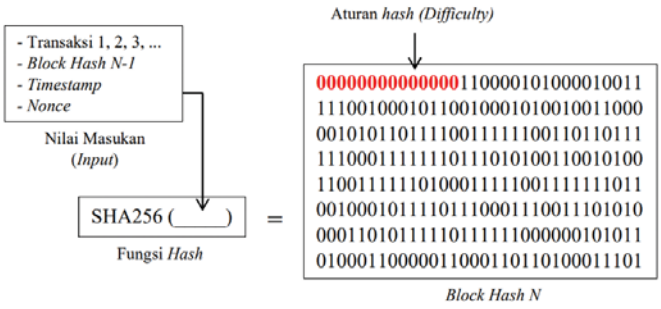
Gambar 2.3 Arsitektur Aplikasi *Decentralized*

Decentralized application bekerja dengan tiga komponen utama, yaitu aplikasi *client*, *smart contract* dan blockchain. Aplikasi *client* dapat berupa aplikasi *web* ataupun *mobile* yang memiliki kemampuan untuk berkomunikasi dengan *smart contract* melalui *Application Binary Interface* (ABI). *Smart contract* sendiri akan bekerja layaknya seperti aplikasi *server-side* (*back-end*). *Smart contract* tersimpan pada *blockchain* yang tersebar di setiap *node* yang tergabung dalam jaringan terdesentralisasi. *Smart contract* akan dieksekusi berdasarkan permintaan yang dikirim oleh aplikasi *client* dan hasilnya akan dikembalikan dalam bentuk respon data (Sayeed dkk., 2020).

2.4. *Proof of Work* (PoW)

Setiap data transaksi dalam *blockchain* tercatat menggunakan kriptografi dan melalui *protocol consensus* (Zheng dkk., 2017). *Protocol* ini digunakan dalam jaringan *blockchain* agar setiap *node* yang tergabung dalam jaringan menggunakan rantai *block* yang sama. Pada mekanisme *protocol consensus Proof of Work* (PoW), *node* membentuk suatu *block* yang akan dianggap *valid* dengan melakukan komputasi untuk mendapatkan nilai *hash* dari *block* berdasarkan aturan nilai yang harus dipenuhi (Zheng, dkk., 2017). Ketika *block* data telah terbentuk, *block* data kemudian akan dikirim ke *node* lain yang tergabung dalam jaringan untuk dilakukan validasi. Ketika *block* data telah tervalidasi, maka *node* lain akan menambahkan *block* baru tersebut ke dalam rantai *block* yang telah tersedia (Zheng, dkk., 2017).

Pada *protocol* PoW terdapat peraturan yang perlu dipenuhi dalam perhitungan *block hash* dari suatu *block* baru. Aturan ini biasa disebut *Difficulty*. *Difficulty* mengharuskan nilai dari suatu *block hash* memiliki beberapa angka nol pada *bit* awalnya. Dalam perhitungan nilai *block hash* yang sesuai berdasarkan *difficulty*, nilai masukan *nonce* merupakan kunci utama penyelesaiannya. *Nonce* adalah nilai masukan yang ditentukan oleh *node* dalam melakukan komputasi *block hash*. Penggunaan nilai masukan yang dinamis seperti *nonce* akan dibutuhkan untuk mendapatkan *block hash* yang sesuai berdasarkan *difficulty* yang ditetapkan. Nilai *nonce* 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 (Aprialim dkk., 2021). Gambaran komputasi *block hash* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Komputasi *Block Hash*
Sumber: (Aprialim dkk., 2021)

Fungsi *hash* kriptografi seperti *SHA-256* akan menghasilkan nilai *hash* yang sangat tidak terduga dan bersifat satu arah. Dengan kata lain, *hash* yang akan terbentuk tidak dapat dilakukan *decryption* untuk mendapatkan nilai masukan awal yang digunakan pada fungsinya. *Block hash* yang memenuhi kriteria *difficulty* yang ditetapkan hanya dapat dihasilkan dengan cara menjalankan fungsi *hash* secara berulang kali menggunakan *nonce* yang berbeda. *Node* akan saling berkompetisi dalam membentuk suatu *block* dengan cara mencari *nonce* yang sesuai sehingga *block hash* yang didapatkan dapat memenuhi kriteria *difficulty*. Proses ini akan membutuhkan komputasi yang besar. Dengan demikian, didapatkannya nilai *nonce* yang sesuai dapat menjadi bukti bahwa pembentukan *block* telah melalui proses komputasi yang besar. *Block* yang terbentuk dengan nilai *nonce* yang sesuai akan dianggap oleh jaringan sebagai *block* yang *valid* (Aprialim dkk., 2021).

Umumnya, berdasarkan protokol *blockchain* yang ditetapkan, ketika suatu *block* berhasil dibentuk oleh suatu *node*, *node* tersebut akan mendapatkan hadiah berupa *cryptocurrency*, tetapi karena proses pembentukan *block* ini membutuhkan komputasi yang besar, *node* memiliki pilihan untuk melakukannya atau tidak. *Node* yang melakukan pembentukan *block* dikenal dengan istilah *miner* (Aprialim dkk., 2021).

Ketika *miner* telah menemukan *nonce* yang sesuai, selanjutnya *block* yang dibentuk akan diumumkan ke seluruh *node* yang tergabung dalam jaringan. Seluruh *node* akan melakukan verifikasi terkait pembentukan *block* hanya dengan sekali menjalankan fungsi *hash* menggunakan nilai masukan yang telah didapatkan oleh *miner*. Kemudian apabila *block* yang terbentuk sesuai dengan protokol yang ditetapkan, maka *block* tersebut akan dimasukan ke dalam rantai *block* yang sudah ada (Aprialim dkk., 2021).

Fungsi *hash* kriptografi seperti SHA-256 akan menghasilkan nilai *hash* yang sangat tidak terduga dan bersifat satu arah. Dengan kata lain, *hash* yang akan terbentuk tidak dapat dilakukan *decryption* untuk mendapatkan nilai masukan awal yang digunakan pada fungsinya. *Block hash* yang memenuhi kriteria *difficulty* yang ditetapkan hanya dapat dihasilkan dengan cara menjalankan fungsi *hash* secara berulang kali menggunakan *nonce* yang berbeda. *Node* akan saling berkompetisi dalam membentuk suatu *block* dengan cara mencari *nonce* yang sesuai sehingga

block hash yang didapatkan dapat memenuhi kriteria *difficulty*. Proses ini akan membutuhkan komputasi yang besar. Dengan demikian, didapatkan nilai *nonce* yang sesuai sehingga dapat menjadi bukti bahwa pembentukan block telah melalui proses komputasi yang besar. *Block* yang terbentuk dengan nilai *nonce* yang sesuai akan dianggap oleh jaringan sebagai *block* yang *valid* (Aprialim dkk., 2021).

Umumnya, berdasarkan protokol *blockchain* yang ditetapkan, ketika suatu *block* berhasil dibentuk oleh suatu *node*, *node* tersebut akan mendapatkan hadiah berupa *cryptocurrency*, tetapi karena proses pembentukan *block* ini membutuhkan komputasi yang besar, *node* memiliki pilihan untuk melakukannya atau tidak. *Node* yang melakukan pembentukan *block* dikenal dengan istilah *miner* (Aprialim dkk., 2021). Ketika *miner* telah menemukan *nonce* yang sesuai, selanjutnya *block* yang dibentuk akan diumumkan ke seluruh node yang tergabung dalam jaringan. Seluruh *node* akan melakukan verifikasi terkait pembentukan *block* hanya dengan sekali menjalankan fungsi hash menggunakan nilai masukan yang telah didapatkan oleh *miner*. Kemudian apabila *block* yang terbentuk sesuai dengan protokol yang ditetapkan, maka *block* tersebut akan dimasukan ke dalam rantai *block* yang sudah ada (Aprialim dkk., 2021).

2.5. *Proof of Stake* (PoS)

Ethereum melakukan pergantian mekanisme *protocol consensus* yang semula *Proof-of-Work* (PoW) menjadi *Proof-of-Stake* (PoS) pada tahun 2022. Perubahan *protocol consensus* menjadi PoS ini diklaim dapat membuat jaringan lebih aman, hemat energi dan dapat mengatasi masalah penskalaan dari *protocol consensus* sebelumnya yaitu PoW (Smith, 2022).

Pada ethereum yang menggunakan *consensus* PoS, para *validator* melakukan staking modal dalam bentuk ETH yang disimpan kedalam *smart contract* yang terdapat pada ethereum. ETH yang di-*staking* ini digunakan sebagai jaminan jika *validator* melakukan kecurangan dan jarang melakukan validasi. *Validator* memiliki kewajiban untuk memeriksa *block* baru yang disebarkan ke jaringan, membuat *block* baru dan menyebarkan *block* ke seluruh jaringan (Smith, 2022).

2.6. Ethereum

Ethereum merupakan salah satu bentuk pengembangan dari teknologi *blockchain* seperti bitcoin. Ethereum dikembangkan untuk memungkinkan pengerjaan komputasi yang lebih kompleks pada *framework blockchain* daripada hanya sekedar komputasi pencatatan data transaksi. Sama halnya dengan bitcoin, ethereum pada dasarnya merupakan sistem pembayaran mata uang digital (*cryptocurrency*) yang terdesentralisasi. Perbedaan utama yang ada pada ethereum yaitu sistemnya dibangun dengan menggunakan bahasa pemrograman *turing-complete* (Buterin, 2014), sehingga memungkinkan pengerjaan komputasi yang lebih kompleks, seperti *smart contract*. Ethereum telah dikenal luas sebagai *framework* dalam mengembangkan *decentralized application*.

Blockchain ethereum pada dasarnya merupakan *state machine* berbasis transaksi. *State machine* sendiri mengacu pada proses pengelolaan suatu susunan input untuk mengubah *state* yang tersimpan. *State machine* pada ethereum disebut sebagai *Ethereum Virtual Machine* (EVM). Pengubahan suatu *state* dilakukan oleh suatu *node* dengan mengirim transaksi yang berisi input untuk melakukan proses pengubahan *state* (Kasireddy, 2017).

State pada ethereum merepresentasikan seluruh transaksi yang terjadi. Sama halnya dengan bitcoin, transaksi-transaksi ini tergabung pada suatu *block*, dimana setiap *block* ini saling terhubung dengan *block* yang telah terbentuk sebelumnya. *Block* akan dibentuk dengan menggunakan *protocol consensus* yang dinamakan GHOST (*Greedy Heaviest Observed Subtree*) (Kasireddy, 2017). *Protocol* GHOST pada dasarnya merupakan *protocol proof-of-work*, tetapi dengan beberapa penyempurnaan. Penyempurnaan ini berkaitan dengan permasalahan *state block*. *State block* yaitu *block* lain yang terbentuk secara bersamaan dengan *block* yang tervalidasi. Pada protokol GHOST, para penambang tetap akan menerima hadiah apabila berhasil membentuk sebuah *state block*. Hal ini dikarenakan pembentukan *block* pada jaringan ethereum tergolong lebih cepat jika dibandingkan dengan jaringan bitcoin.

2.5.1. *Account*

Account merupakan tanda pengenal atau identitas dari suatu entitas, seperti pengguna, *node*, ataupun *smart contract* yang tergabung dalam ethereum. *Account*

pada ethereum terbagi menjadi dua jenis, yaitu *externally owner account* dan *contract account* (Kasireddy, 2017). *External owner account* pada dasarnya merupakan akun yang dimiliki oleh pengguna ataupun *node*. Sementara *contract account* merupakan tanda pengenal dari suatu *smart contract* yang tercatat dalam ethereum. Pada dasarnya, hanya *externally owned account* yang dapat memulai mengirim pesan ke *account* lainnya dengan cara membuat dan menandatangani transaksi menggunakan *digital signature* berupa *private key*, sementara *contract account* hanya bisa melakukan transaksi sebagai bentuk respon apabila telah menerima suatu transaksi dari *account* lain.

Account merupakan objek yang membentuk *state* pada *ethereum*. Setiap *account* memiliki alamat (*address*) dan terdiri dari empat komponen *state* (Wood, 2014), berupa:

1. *Nonce* merupakan nilai yang merepresentasikan jumlah transaksi yang telah dilakukan oleh suatu *account*.
2. *Balance* merupakan jumlah saldo dalam satuan ETH yang dimiliki oleh suatu *account*.
3. *StorageRoot* merupakan *root hash* dari konten-konten yang dimiliki oleh suatu *account* yang tersimpan. Secara *default*, *storageRoot* memiliki nilai kosong.
4. *CodeHash* merupakan *hash* dari kode yang dimiliki oleh suatu *account* yang akan dieksekusi oleh EVM. Pada *contract account, codeHash* merupakan *hash* dari program *smart contract* yang dibentuk itu sendiri. Sementara pada *externally owned account*, *codeHash* merupakan *hash* dari *string* yang bernilai kosong.

2.5.2. *Transaction*

Pada dasarnya, transaksi merupakan suatu instruksi yang dihasilkan oleh *externally owned account*. Transaksi akan dilakukan dengan menggunakan *digital signature public* ataupun *private key* milik *externally owner account* yang akan melakukan transaksi. Setiap transaksi yang dilakukan akan tercatat ke dalam *blockchain* ethereum. Transaksi dalam ethereum terbagi menjadi dua jenis, yaitu *message call* dan *contract creation* (Kasireddy, 2017). *Message call* merupakan transaksi yang dilakukan dalam melakukan suatu perubahan *state*. *Contract*

creation merupakan transaksi yang dilakukan untuk ketika pembuatan *contract* baru.

Setiap proses komputasi yang dibutuhkan untuk memproses suatu transaksi dalam ethereum akan dieksekusi oleh *node miner*. *Miner* akan membutuhkan sebuah bayaran sebagai hadiah dalam mengeksekusi proses komputasi dari suatu transaksi. Biaya pemrosesan ini dibayar oleh *account* selaku pengirim transaksi berdasarkan unit khusus dalam *ethereum* yang dinamakan *gas* (Kasireddy, 2017). *Gas* pada dasarnya merupakan satuan unit yang digunakan untuk mengukur besar komputasi yang dibutuhkan dari suatu transaksi. Penentuan biaya transaksi akan diikuti dengan menggunakan dua variabel yaitu *gas price* dan *gas limit*. *Gas limit* merupakan jumlah maksimum *gas* yang akan dipakai dalam proses komputasi transaksi. Sedangkan *gas price* merupakan ETH yang pengirim transaksi bersedia bayar untuk penggunaan satu *gas* dalam proses komputasi transaksi.

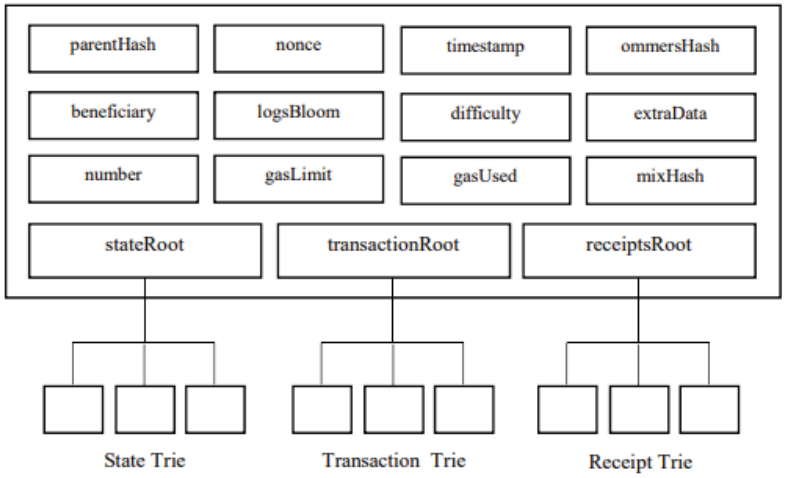
2.5.3. *Block*

Block pada ethereum terdiri dari tiga bagian, yaitu *block header*, informasi transaksi-transaksi yang tergabung dalam *block* dan kumpulan *block header* dari *ommer*. *Ommer* merupakan *stale block* yang terbentuk secara bersamaan dengan *block* yang tervalidasi (Kasireddy, 2017). Pada ethereum, *ommer* dipertimbangkan sebagai salah satu penunjang mekanisme *blockchain* agar dapat berjalan dengan baik. Hal tersebut dikarenakan ethereum hanya membutuhkan waktu 15 detik untuk membentuk suatu *block* dibanding *blockchain* lain seperti bitcoin yang memerlukan waktu hingga 10 menit. Semakin cepat waktu pembentukan *block* pada suatu *blockchain* maka semakin besar juga persaingan yang ada dalam proses pembentukan *block*.

Protokol *blockchain* pada umumnya hanya memberikan hadiah kepada *miner* yang melakukan pembentukan *block* yang tervalidasi. Pada protokol yang diterapkan ethereum, *miner* yang membentuk *ommer* juga akan mendapatkan hadiah, walaupun jumlahnya tidak sebanyak hadiah *block* tervalidasi. *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*. *Block header* berisi informasi-informasi yang terdiri dari (Wood, 2014):

1. *Parent hash*, merupakan *hash* dari *block header* milik *parent block*, yaitu *block* terakhir sebelum *block* terkait.
2. *Ommer hash,* merupakan *hash* dari daftar *ommer block* terkait.
3. *Beneficiary,* merupakan *address* dari *account* yang menerima biaya pembayaran dalam proses *mining block* terkait.
4. *State root,* merupakan *hash* dari *state* yang tersimpan pada *block* terkait. *Hash* ini terbentuk menggunakan struktur *Merkle Patricia Trie*.
5. *Transaction root,* merupakan *hash* dari transaksi yang tercatat pada *block* terkait. *Hash* ini terbentuk menggunakan struktur *Merkle Patricia Tr*ie.
6. *Receipts root,* merupakan *hash* dari riwayat transaksi yang tercatat pada *block* terkait. *Hash* ini terbentuk menggunakan struktur *Merkle Patricia Trie*.
7. *Logs bloom,* merupakan struktur data *bloom filter* yang terdiri atas catatan informasi.
8. *Difficulty,* merupakan tingkat *difficulty* atau kerumitan dari penyelesaian *block* terkait.
9. *Number,* merupakan nilai penjumlahan (*count*) dari *block* terkait.
10. *Gas limit,* merupakan batas maksimum *gas* yang dapat digunakan dalam pemrosesan komputasi transaksi pada *block* terkait.
11. *Gas used,* merupakan batas maksimum *gas* yang dapat digunakan dalam pemrosesan komputasi transaksi pada *block* terkait.
12. *Timestamp*, merupakan waktu *timestamp* yang berbasis *unix* dari pembentukan *block* terkait.
13. *Extra data,* merupakan data tambahan yang berhubungan dengan *block* terkait.
14. *Mix hash,* merupakan sebuah *hash* yang jika dikombinasikan dengan *nonce* maka akan berisi informasi bahwa *block* terkait telah dibentuk melalui proses komputasi yang cukup.
15. *Nonce,* merupakan sebuah *hash* yang jika dikombinasikan dengan *mix hash*, akan membuktikan bahwa *block* terkait telah dibentuk melalui proses komputasi yang cukup.

Ilustrasi *block header* yang terdapat pada suatu *block* ethereum dapat dilihat pada Gambar 5.



Gambar 2.5 *Block Header*

Sumber (Wood, 2014)

2.5.4. Eksekusi *Transaction*

Eksekusi transaksi dalam ethereum pada dasarnya mengacu pada proses transisi *state* yang ada (Wood, 2014). Sebelum transaksi dieksekusi terdapat beberapa syarat yang harus dipenuhi terlebih dahulu, yaitu:

1. Transaksi harus memiliki format yang sesuai. Format *Recursive Length Prefix* (RLP) merupakan format data yang diterima oleh *ethereum*.
2. *Signature* transaksi yang *valid*.
3. *Nonce* transaksi yang *valid*.
4. *Gas limit* transaksi yang harus memiliki jumlah yang lebih besar atau sama dengan jumlah *gas* yang terpakai dalam eksekusi komputasi transaksi.
5. Jumlah *ether* yang dimiliki oleh pengirim transaksi dapat menutupi biaya pemakaian *gas* yang terbentuk berdasarkan perhitungan *gas limit* dan *gas price*.

Apabila syarat-syarat yang diperlukan telah terpenuhi, selanjutnya eksekusi transaksi dapat dilakukan. Eksekusi transaksi secara umum akan dilakukan melalui tahap-tahap sebagai berikut:

1. Biaya transaksi akan diukur berdasarkan *gas limit* dan *gas price* yang ditentukan. *Ether* pengirim transaksi akan terpotong berdasarkan hasil

perhitungan biaya perhitungan biaya transaksi tersebut.

2. Memberikan jumlah *default gas* yang akan digunakan selama proses komputasi transaksi.
3. Komputasi-komputasi yang diperlukan akan dieksekusi pada *ethereum virtual machine* oleh *miner*.
4. Ketika komputasi yang diperlukan oleh transaksi telah terproses, maka biaya yang terbayar untuk sisa *gas* yang tidak terpakai akan dikembalikan ke pengirim transaksi. Kemudian biaya *gas* yang terpakai akan dikirim ke *miner*.
5. *State* baru dan *log* yang berisi catatan transaksi yang terjadi akan terbentuk

2.7. *InterPlanetary File System* (IPFS)

InterPlanetary File System atau dikenal sebagai IPFS merupakan protokol *hypermedia peer-to-peer* terdistribusi yang digunakan untuk menyimpan dan berbagi konten digital dalam sistem yang terdesentralisasi (Nizamuddin dkk., 2018). IPFS menggunakan *content based addressing* untuk melakukan identifikasi terhadap *file* yang dituju. Hal ini menyebabkan ketika ada perubahan data pada *file* maka akan menghasilkan hasil *hash* yang berbeda juga. IPFS sudah dilengkapi dengan metode *cryptocurrency* untuk *relending* data. Sebagian *user* menggunakan IPFS untuk membagikan *file* berukuran besar karena IPFS menggunakan *hosting local* yang dapat mengurangi kebutuhan *bandwidth* untuk berbagi *file* besar di internet (Nyaletey dkk., 2019). *File* yang sudah tersimpan di IPFS menghasilkan *hash file* yang kemudian disimpan pada jaringan ethereum. *Hash file* yang tersimpan pada ethereum dapat digunakan untuk mengakses *file* yang tersimpan pada IPFS (Rajalakshmi dkk., 2018).

2.8. *Crowdfunding*

Crowdfunding atau biasa disebut dengan *crowd financing*, *equity crowdfunding* atau *hyper funding* merupakan praktik penggalangan dana dari seseorang atau kelompok orang untuk mendukung sebuah proyek atau bisnis yang biasanya dilakukan melalui media internet (Mollick, 2014). Penggalangan dana yang dimaksud seperti penggalangan dana untuk penanggulangan bencana alam,

perusahaan *startup* bahkan untuk penelitian ilmiah. Dalam perkembangannya *crowdfunding* memiliki beberapa model, diantaranya:

1. *Donation-based crowdfunding,* kampanye model ini dirancang untuk kegiatan amal, dimana orang diminta untuk mendukung proyek *non-profit* seperti membantu korban bencana (Hossain & Oparaocha, 2017). Pada model ini donatur tidak mendapatkan keuntungan, tetapi mereka biasanya akan mendapatkan hadiah atau produk (Forbes & Schaefer, 2017).
2. *Rewards-based crowdfunding,* dalam kampanye jenis ini setiap individu yang melakukan donasi akan mendapatkan reward berupa hal-hal *non-financial*, seperti *voucher*, tiket, dan lain sebagainya.
3. *Equity-based crowdfunding* pada model ini orang yang mendukung sebuah bisnis akan mendapatkan *reward* berupa *equity* (persentase kepemilikan suatu perusahaan yang biasanya dalam bentuk saham).

Platform khusus *crowdfunding* saat ini telah banyak dikembangkan untuk memenuhi kebutuhan masyarakat dalam melakukan kegiatan penggalangan dana. Selain sebagai media publikasi, platform khusus ini menyediakan berbagai fitur untuk memudahkan masyarakat dalam membentuk *crowdfunding* secara efisien dan sesuai. Fitur yang disediakan dapat berupa verifikasi kebenaran dan keabsahan dari suatu *crowdfunding*, integrasi *payment gateway* untuk memudahkan kegiatan pemberian dana dan lain sebagainya (Aprialim dkk., 2021).