BAB II
TINJAUAN PUSTAKA

2.1 KAJIAN PUSTAKA

Pada penelitian [2], yang dimana sistem *e-vote* diintegrasikan dengan *Hyperledger Fabric* untuk membangun arsitektur jaringan *permissioned blockchain*. Dari hasil penelitian tersebut sistem *e-vote* yang terintegrasi dengan *permissioned blockchain* menggunakan *Hyperledger Fabric* berhasil dibentuk. Penelitian tersebut menguji integritas data dengan cara menggunakan dua mekanisme, mekanisme pertama ialah jika *insider attack* melakukan perubahan pada Salinan data, maka *permissioned blockchain* akan mendeteksi perubahan tersebut dan tidak akan mengeksekusi transaksi. Mekanisme yang kedua ialah melakukan pengubahan data pada block disalah satu *peer*, pada saat *insider attack* melakukan hal tersebut maka sistem *chain* di mana *block* yang berubah akan putus, dan *chain* tersebut tidak akan diperbaharui lebih lanjut. Dari hasil pengujian tersebut dapat disimpulkan bahwa penggunaan *permissioned blockchain* pada *Hyperledger Fabric* dapat menjaga integritas data yang ada didalam jaringan tersebut.

Pada penelitian [3], memaparkan rancangan sistem *voting* menggunakan bahasa C++ pada *frontend* dan menggunakan bahasa pemrograman Node.JS pada *backend*. Perhitungan suara hasil *voting* dengan cara mengumpulkan blok dari jaringan *Blockchain* lalu direkonstruksi dengan menggunakan metode *Multi Party Computation*. Hasil implementasi *Blockchain* dengan menggabungkan dengan metode rancangan *Multi Party Computation* terbukti menjaga integritas hasil dari *voting* dikarenakan untuk mengubah suatu data harus dari persetujuan dari semua *node*.

Penelitian [6], memaparkan implementasi teknologi *blockchain* pada *website e-vote*, pada penelitian tersebut menggunakan bahasa pemrograman *python* dan menggunakan *framework Django* dalam pembuatan website. Hasil dari penelitian

tersebut ialah alur *voting* sama seperti *voting* konvensional yaitu masuk kedalam akun atau jaringan, kemudian *voter* memilih kandidat yang akan dipilih, kemudian *voter* mendapat kode yang nantinya akan digunakan untuk validasi data oleh admin, yang dimana kode tesebut dapat dicek oleh admin atau petugas untuk mengecek apakah ada duplikasi atau tidak.

Penelitian [7], memaparkan implementasi teknologi *blockchain* yang dimana studi kasusnya ialah *e-voting*, dan pemungutan suara dilakukan dengan menggunakan *website*. Penelitian tersebut menggunakan *Multichain* sebagai platform untuk *blockchain*. Hasil penelitian tersebut menyimpulkan bahwa dengan menggunakan teknologi *blockchain* dengan *Multichain tools* dapat membantu sistem *e-voting* dalam menyimpan data secara transparan, data pemilih tidak dapat diketahui identitasnya, dan data *voting* tidak dapat diubah, digandakan, atau dihapus.

Penelitian [8], melakukan implementasi jaringan *blockchain* menggunakan *Hyperledger Fabric* pada sistem *e-vote*. Replikasi *ledger* disimpan pada *CouchDB*. Hasil dari penelitian tersebut ialah *Hyperledger Fabric* memiliki masalah dalam manajemen konkurensi yang dimana entitas tidak dapat dimodifikasi dalam transaksi bersamaan, tetapi sistem *e-voting* yang telah dirancang dan diuji menggunakan *Hyperledger Fabric* layak untuk memenuhi persyaratan untuk skema *e-voting* seperti transparansi, konsistensi, dan ketahanan.

Table 2.1 Penelitian sebelumnya

No	Judul	Tahun	Isi	Perbandingan
1	Implementasi Permissioned Blockchain Berbasis Hyperledger Sebagai	2019	Penerapan teknologi *blockchain* pada sistem *e-vote* untuk menjaga integritas data pemungutan suara.	Perbedaan penelitian hanya sebatas penggunaan teknologi *blockchain*

No	Judul	Tahun	Isi	Perbandingan
	Penjamin Integritas Data Pada Sistem E-Vote			*Hyperledger fabric*, belum adanya *User interface* pada sistem.
2	Implementasi Teknologi *Blockchain* Dan *Multi Party Computation* Dalam Sistem *E-Vote*	2016	Menggunakan teknologi *blockchain* yang dikombinasikan dengan metode *Multi Party Computation* untuk menjaga integritas data pada pemungutan suara	Mengkombinasikan sistem *blockchain* dan *Multi Party Computation*.
3	Implementasi Teknologi *Blockchain* Pada *Website E-Vote* Menggunakan Bahasa Pemprograman *Phyton*	2021	Mengimplementasikan teknologi *blockchain* pada sistem *e-vote* dengan basis bahasa pemrograman *python*	Menggunakan basis bahasa pemrograman *python*, dan menggunakan *Django* untuk membangun sebuah *website e-vote*.
4	Implementasi *Blockchain*: Studi Kasus *e-Voting*	2019	Membangun aplikasi *e-vote* dengan menggunakan *Multichain* untuk membuat jaringan *blockchain*.	Menggunakan *multichain* dalam membangun jaringan *blockchain*.

No	Judul	Tahun	Isi	Perbandingan
5	*E-Voting System Using Hyperledger Fabric Blockchain and Smart Contracts*	2021	Memanfaatkan *Hyperledger Fabric* dan *smart contract* dalam membangun sistem *e-vote* menggunakan jaringan *blockchain*	Menggunakan *Angular* untuk *User Interface* berbasis website.

2.2 DASAR TEORI

2.2.1 E-vote

E-vote atau Electronic Vote adalah mekanisme pemungutan suara yang memanfaatkan sarana teknologi informasi atau perangkat elektronik pada saat penggunaaan hak suara. E-vote dilakukan untuk mempercepat tabulasi suara, mengurangi biaya, dan meningkatkan akurasi pada kegiatan pemilihan [1] [2] [3]. Di Indonesia beberapa daerah telah menerapkan *E-vote* meski diterapkan dalam lingkup Pilkades, daerah yang telah menerapkan *E-vote* yaitu Jembrana, Bali dan Boyolali, Jawa Tengah. Penerapan *E-vote* dapat mengurangi jumlah oknum potensial yang dapat melakukan kecurangan terhadap pemilihan suara karena dibutuhkan kemampuan dibidang ilmu computer sebagai dasar untuk memanipulasi data [3].

2.2.2 Blockchain

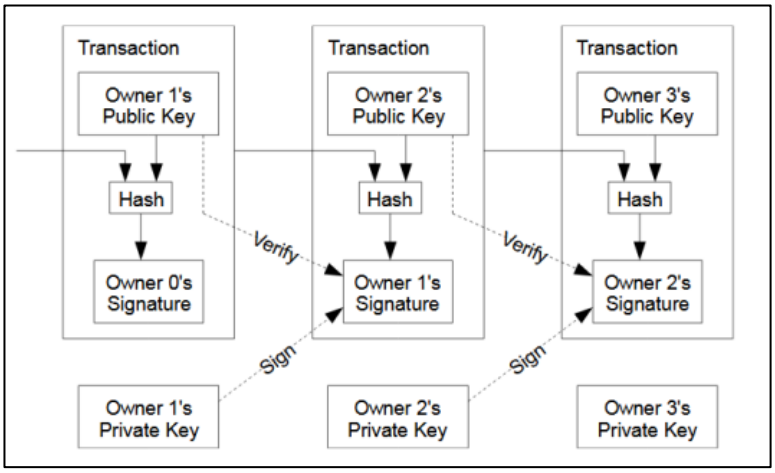
Blockchain adalah teknologi yang didasarkan pada teknologi desain arsitektur *cryptocurrency Bitcoin* yang ditemukan oleh Satoshi Nakamoto pada tahun 2008 dan terus berkembang baik dalam hal manajemen, privasi, keamanan, dan tata kelola data hingga diaplikasikan ke dalam beberapa bidang seperti kesehatan, pendidikan, bisnis, industry, manajemen rantai pasok dan keuangan [6] [9]. *Blockchain* merupakan salah satu solusi yang berkonsep seperti *database* berbentuk

jaringan terdistribusi yang mempertahankan data secara yang terus-menerus berkembang setelah data tersebut dikonfirmasi oleh setiap *node* yang berpartisipasi, yang dimana data tersebut berisi transaksi yang disimpan dalam sebuah blok data.

Setiap blok tersebut berisi penanda waktu (*timestamp*), ditandai dengan *hash* kriptografi dan tautan ke blok sebelumnya sehingga membentuk rantai blok (*chain*) [2] [6] [10] [11]. Blockchain memiliki beberapa jenis untuk diimplementasikan sesuai dengan kondisi yang dihrapakan, terdapat tiga jenis *Blockchain* yaitu [12]:

1. *Public Blockchain*. *Public Blockchain* merupakan jenis *blockchain* yang dapat diakses oleh siapapun. *Ledger* atau buku besar pada *blockchain* ini terbuka dan transparan sehingga semua orang dapat mengaksesnya. *Public blockchain* memiliki biaya operasi yang tinggi serta kecepatan trannsaksi yang lambat. *Bitcoin* dan *Ethereum* merupakan salah satu contoh penggunaan jenis *Blockchain* ini.
2. *Private Blockchain* atau *Permissioned Blockchain*. *Private Blockchain* merupakan jenis *blockchain* yang dipergunakan untuk pertukaran data secara pribadi pada sekelompok individua atau organisasi. Pengguna yang tidak dikenal akan tidak diberikan akses ke jaringan *blockchain* ini. Jenis *blockchain* ini digunakan pada *Hyperledger* dan *R3 Corda*.
3. *Consortium Blockchain*, merupakan jenis *blockchain* gabungan dari jenis *Public* dan *Private*, dimana tidak ada organisasi tunggal yang bertanggung jawab yang dapat mengontrol jaringan melainkan *node* yang telah ditentukan sebelumnya. *Hyperledger Fabric* merupakan salah satu yang menggunakan jenis *blockchain* ini.

Pada saat seseorang ingin memodifikasi data yang ada pada jaringan *Blockchain*, penyerang harus menelusuri data dari *node* awal dimana tempat memulai proses suatu jaringan *Blockchain*, kemudian mengulang proses tersebut disemua *node* yang terlibat pada sistem. Cara tersebut dilakukan agar meyakinkan sistem bahwa modifikasi data yang dilakukan seolah-olah benar. Hal tersebut yang membuat data pada *Blockchain* bersifat *tamper proof* dan memiliki integritas yang tinggi karena melakukan modifikasi pada semua *node* dianggap impraktikal [3].

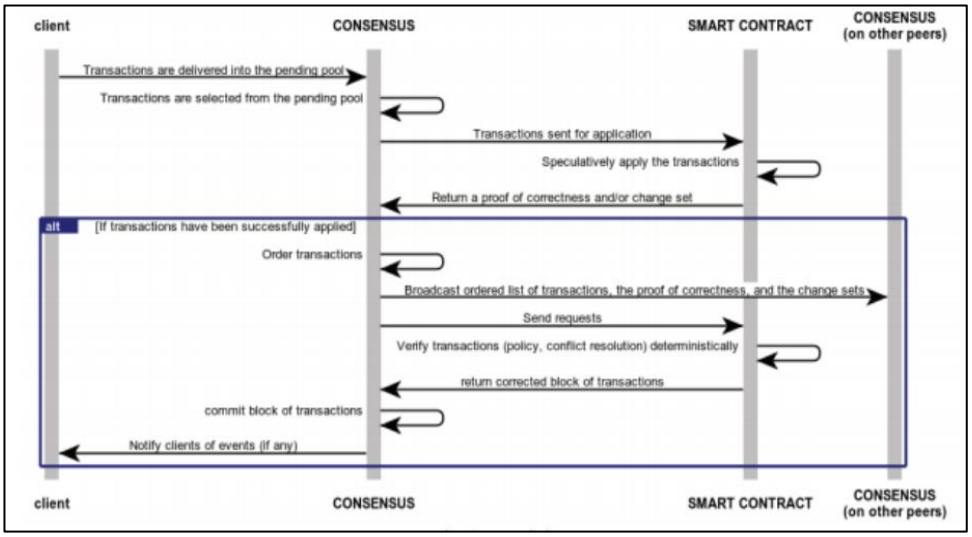


Gambar 2.1 Alur *Ledger* pada Bitcoin [5]

Mengenai cara kerja atau algoritma pada teknologi *Blockchain* dapat dilihat pada Gambar 2.1 dimana mekanisme konsensus atau mekanisme proses pemesanan transaksi yang terjamin membutuhkan *node* untuk menyelesaikan persamaan matematika yang kompleks. Setelah *node* menemukan nilai yang benar dari perhitungan matematika acak maka pada *ledger* atau buku besar akan ditambahkan sebuah blok baru [5].

2.2.3 *Hyperledger* dan *Hyperledger Fabric*

Hyperledger merupakan komunitas yang mengembangkan serangkaian *framework*, *tool*, dan *library* untuk pengembangan *enterprise-grade* blockchain yang bersifat *open-source* [5] [13].



Gambar 2.2 Cara Kerja *Hyperledger* [5]

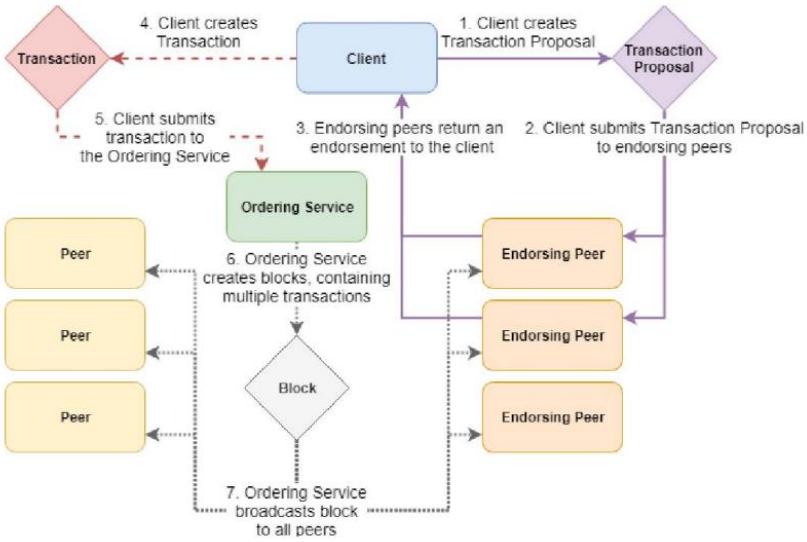
Mengenai cara kerja *Hyperledger* dijelaskan lebih lanjut pada Gambar 2.2. Cara kerangka kerja *blockchain Hyperledger* memenuhi konsensus yaitu dengan melakukan 2 aktivitas yaitu pemesanan transaksi dan validasi transaksi seperti yang merujuk pada *paper* mengenai "Studi Perbandingan Jaringan Blockchain sebagai Platform Sistem Rating" [5]. *Hyperledger Fabric* merupakan salah satu platform teknologi *blockchain* yang menggunakan sistem *permissioned blockchain* yang berjalan pada sekumpulan partisipan atau *node* yang telah diketahui dan telah diidentifikasi [2]. Salah satu *framework* atau kerangka kerja dari *Hyperledger* ialah *Hyperledger Fabric* yang merupakan salah satu *framework* untuk membangun infrastruktur sistem atau platform untuk membangun buku besar atau *ledger* pada *Private Blockchain* [5] [14].

Hyperledger Fabric merupakan salah satu pilihan *framework* yang lebih mudah dikembangkan daripada *bitcoin* dan *ethereum*, dan dapat mengatur akses pengguna pada sistem sehingga hanya peserta yang dalam suatu jaringan *blockchain* yang dapat melihat detail sensitif [14] [15]. *Hyperledger Fabric* memiliki beberapa fitur utama yaitu [13] :

1. *Permissioned Architecture*
2. Modular yang tinggi
3. Konsensus yang dapat dipasang

4. Model *Smart Contract* terbuka – fleksibilitas dalam penggunaan model yang diinginkan (model akun, model UTXO, data terstruktur, data tidak terstruktur, dll)
5. Memiliki latensi rendah untuk konfirmasi/finalitas
6. Fleksibel terhadap privasi data
7. Mendukung multi-bahasa pembuatan *Smart Contract* (*Go, Java, Javascript*)
8. Mendukung EVM (*Ethereum Virtual Machine*) dan bahasa pemrograman *solidity*
9. Dirancang untuk operasi berkelanjutan, termasuk peningkatan bergulir dan mendukung versi asimetris
10. Tata Kelola dan versi *Smart Contract*
11. Fleksibel dalam mencapai konsensus diseluruh organisasi yang diperlukan
12. *Queryable*

Hyperledger Fabric tidak terdapat dalam sistem *blockchain* tetapi merupakan sistem *blockchain* yang bersifat *private* dan *permissioned* [16]. *Hyperledger Fabric* dapat bersifat *consortium* jika terdapat dua atau lebih organisasi saling berinteraksi didalamnya [16]. Aliran transaksi dalam *Hyperledger Fabric* hanya dapat dimulai ketika dua atau lebih partisipan masuk kedalam suatu *channel*, yang kemudian membuat suatu *chaincode* didalam *channel* tersebut [14]. *Hyperledger Fabric* dalam pengembangannya ditulis dengan bahasa pemrograman umum tanpa bergantung pada *crypocurrency* asli apa pun, yang berbeda dengan kebanyakan *platform blockchain* lainnya yang dimana untuk menjalankan *smart contract* membutuhkan bahasa khusus domain dan mengandalkan *cryptocurrency* [5].



Gambar 2.3 Alur transaksi pada *Hyperledger Fabric* [17]

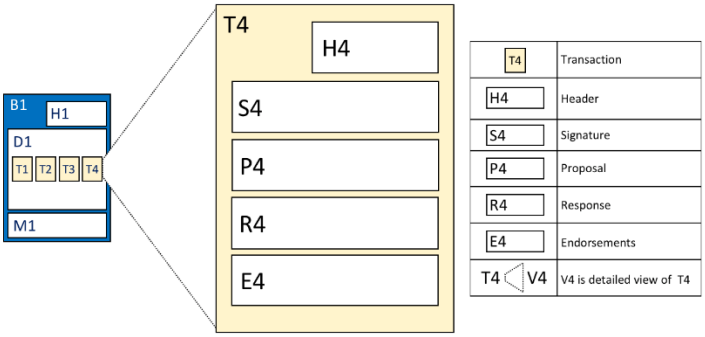
Pada *Hyperledger Fabric* alur transaksi dapat digambarkan pada Gambar 2.3, dimana alur transaksi dibagi menjadi tiga fase utama yaitu [18] [17]:

1. Fase Pengesahan atau *endorsement*, yaitu proses klien mengirimkan transaksi proposal yang ditandatangani ke *endorsing peers*. Tiap *endorsing peers* akan memverifikasi apakah klien tersebut berwenang untuk menjalankan transaksi, proposal transaksi telah terbentuk dengan baik, belum pernah diajukan sebelumnya (*replay-attack protection*), dan tanda tangan yang valid. Proses tersebut dilakukan secara parallel tanpa membutuhkan koordinasi terhadap *endorsing peers*. Setelah semua *endorsing peers* menandatangani pengesahan maka akan mengirimkannya kembali ke klien
2. Fase *ordering*, setelah klien melakukan *endorsement* atau pengesahan yang cukup sesuai dengan kebijakan pengesahan (*endorsement policy*), maka klien akan melakukan pengesahan transaksi dan meneruskan ke *Ordering Service Node* (*OSN*). Kemudian *OSN* melakukan transaksi secara global dengan membuat blok dan mengirim langsung ke *endorsing peers* atau melakukan *gossiping* ke *non-endoring peers*.
3. Fase validasi, pada saat semua *peer* menerima blok yang dikirimkan, maka setiap *peer* akan melakukan validasi transaksi pada blok yang dimana validasi tersebut

memastikan kebijakan pengesahan terpenuhi serta memastikan bahwa tidak ada perubahan status *ledger* pada saat transaksi dilakukan. Validasi tersebut dilakukan dengan setiap *peer* menjalankan *VSCC (Validation System Chaincode)* untuk memeriksa apakah kebijakan *endorsement* sudah terpenuhi dalam hal jumlah *endorsement* valid berasal dari *peer* yang diharapkan. Setelah itu, *peer* akan melakukan pengecekan *read-write conflict*. Transaksi pada blok akan ditandai dengan valid atau tidak valid. Transaksi dianggap valid jika versi setiap *key* yang ada pada *read set* pada blok data nilainya sama pada *current-state*. Setiap transaksi yang tidak valid akan dibuang dan efeknya pada status *blockchain* ialah akan di *rollback* seperti saat sebelum dilakukan transaksi. Pada akhirnya, transaksi yang dianggap valid akan ditambahkan ke *local ledger*.

Hyperledger Fabric memiliki tiga algoritma untuk menanganani proses transaksi pada *ordering service*, tiga algoritma tersebut ialah [19]:

- *RAFT*, merupakan algoritma layanan transaksi yang direkomendasikan. *RAFT* menggunakan layanan transaksi berjenis *crash fault tolerant* (*CFT*) berdasarkan implementasi protocol *RAFT* pada *etcd* yang mengadopsi model *leader and follower*. Dimana *leader node* dipilih dan keputusannya direplikasi oleh *follower* atau pengikut.
- *Kafka*, Merupakan algoritma layanan transaksi yang mirip dengan *RAFT*, serta mengimplementasikan *CFT*. *Kafka* telah dianggap usang atau *deprecated* pada versi dua atau yang lebih baru dikarenakan banyak pengguna tidak menginginkan biaya administrasai tambahan dalam mengelelola kluster kafka.
- *Solo*, Implementasi *solo* pada layanan transaksi hanya diperuntukkan untuk pengujian dan hanya terdiri dari satu layanan transaksi atau *ordering node*. *Solo* telah dianggap usang pada versi dua atau lebih baru dikarenakan *RAFT* memiliki *node* tunggal untuk fungsi yang sepadan.



Gambar 2.4 Struktur *Blockdata Hyperledger Fabric* [20]

Setelah transaksi telah dilakukan maka akan terbentuk sebuah blok baru yang dimana setiap riwayat transaksi akan disimpan pada *blockdata* [20]. Seperti pada Gambar 2.4 yang merupakan struktur dari *blockdata*. *Blockdata* terdiri beberapa bagian [20]:

1. *Header*, diilustrasikan oleh H4, yang berisi nama *chaincode* yang relevan dan versinya.
2. *Signature*, diilustrasikan oleh S4, berisi tanda tangan kriptografi, yang dibuat oleh *client application*. *Signature* merupakan pengesahan yang dilakukan oleh *peer* untuk memeriksa bahwa detail transaksi tidak dilakukan *data tamper*, karena memerlukan *private key* aplikasi untuk membuatnya.
3. *Proposal*, diilustrasikan oleh P4, berisi parameter *input* yang disediakan oleh aplikasi ke *smart contract* yang melakukan perubahan pada *ledger*.
4. *Response*, diilustrasikan oleh R4, yang berisi nilai sebelum dan sesudah pada *world-state* pada saat dilakukan transaksi. *Response* merupakan hasil dari *smart contract*, dan jika transaksi berrhasil divalidasi, maka transaksi tersebut akan diterapkan ke *ledger* untuk memperbarui *world-state*.
5. *Endorsement*, seperti yang ditunjukkan pada *E4*, yang merupakan daftar tanggapan transaksi yang ditandatangani oleh setiap *peer* organisasi.

Pada *Hyperledger Fabric*, pengguna dimungkinkan untuk mendefiniskan kebijakan untuk mengeksekusi *smart contract* [21]. Misalnya kebijakan *endorsement* mungkin seperti [21]:

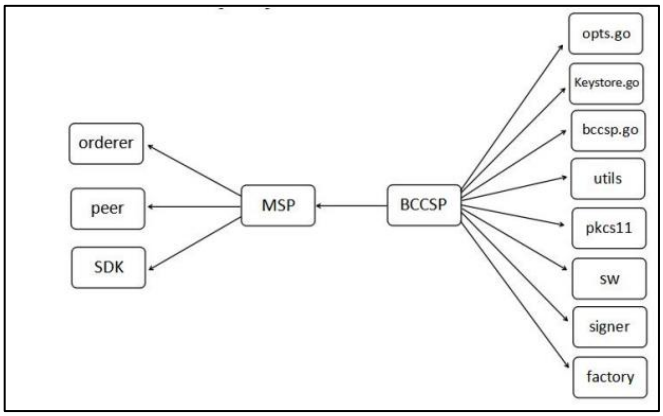
- Seluruh *peer* A, B, C, D pada *channel* harus menandatangani transaksi tipe A.

- Setidaknya dua dari A, B, C, D, E, F harus menandatangani atau melakukan *endorsement* transaksi tipe B.
- Mayoritas *peer* pada channel harus melakukan *endorse* transaksi tipe C.

Seluruh partisipan yang ingin bergabung pada jaringan *Hyperledger Fabric* harus memiliki identitas yang dikenali oleh jaringan *Hyperledger Fabric*. Untuk mengenali partisipan maka *Hyperledger Fabric* menyediakan *Membership Service Provider (MSP)*. Untuk melakukan transaksi pada jaringan *Hyperledger Fabric* , maka setiap partisipan harus [22]:

1. Memiliki identitas yang dikeluarkan oleh *Certificate Authority* (*CA*) yang dipercaya oleh jaringan tersebut.
2. Menjadi anggota organisasi yang diakui dan disetujui oleh anggota jaringan.
3. Menambahkan *MSP* ke konsorsium jaringan atau saluran (*channel*).
4. Disertakan dalam kebijakan jaringan pada *MSP*.

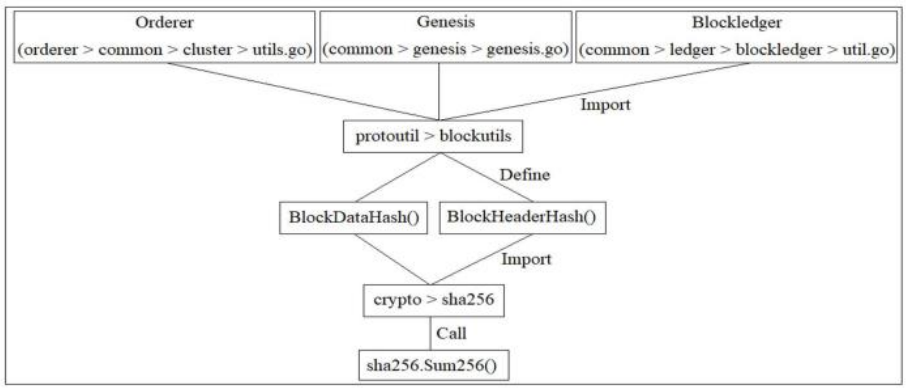
Pada *Hyperledger Fabric*, membutuhkan *BCCSP* (*Blockchain Crypto Service Provider*) untuk memenuhi algoritma kriptografi pada sistem. *BCCSP* tak hanya digunakan pada sistem, *BCCSP* juga digunakan pada beberapa modul *Hyperledger Fabric* seperti *MSP, cryptogen*, *Certificate Authority* (*CA*), dan *Fabric-sdk* [23].



Gambar 2.5 Struktur *BCCSP* pada *Hyperledger Fabric* [23]

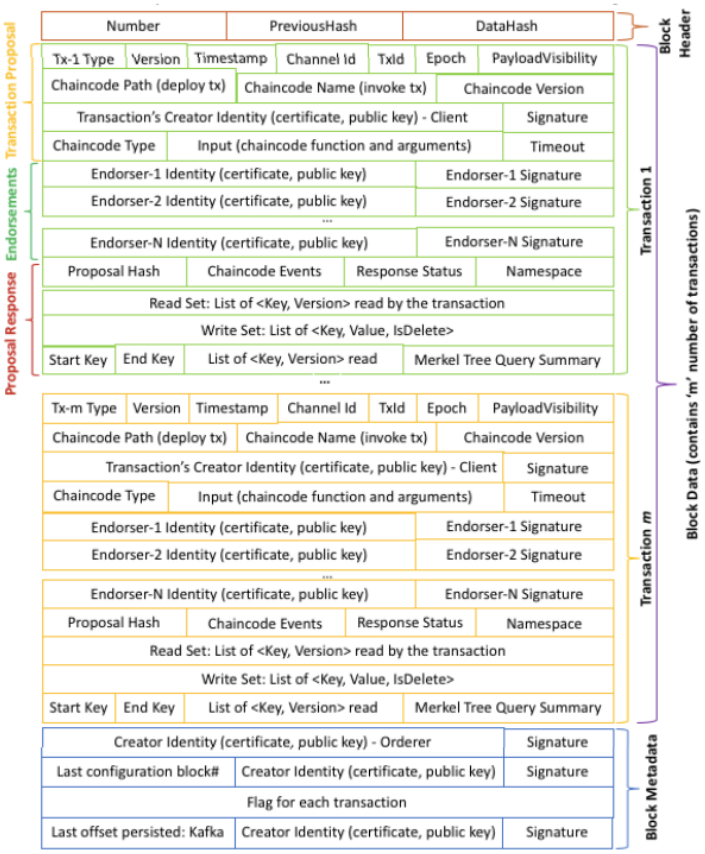
Pada Gambar 2.5 merupakan visualisasi penggunaan *BCCSP* pada *Hyperledger Fabric*. *BCCSP* menyediakan layanan terkait ke modul inti, klien, dan

SDK (*Software Development Kit*) melalui *MSP* (*Membership Service Provider*) [23].



Gambar 2.6 Struktur *Hashing* pada *Hyperledger Fabric* [21]

Seperti pada Gambar 2.6 yang merupakan struktur *hash* pada *Hyperledger Fabric*. *Hyperledger Fabric* menggunakan algoritma *Hash SHA256* sebagai algoritma bawaan untuk melakukan *hash* pada blok data dan blok *header* [21].



Gambar 2.7 Struktur blok pada *Hyperledger Fabric* [24]

Struktur *block* pada *Hyperledger* dapat digambarkan pada Gambar 2.7, yang dimana tiap *block* memiliki 3 bagian utama yaitu *Block header*, *block data*, dan *Block metadata*. Hash pada *block header* pada *Hyperledger Fabric* didapatkan dari kalkulasi hash dari *Number*, *PreviousHash*, dan *Datahash*, *Number* pada *block header* merupakan angka yang didapatkan dari blok keberapa blok tersebut *previoushash* merupakan *Blockheader* dari *block* sebelumnya, kemudian *Datahash* merupakan hasil *Hash* dari data yang ada pada blok tersebut. *Blockdata* merupakan bagian blok yang terdiri dari seluruh transaksi yang ada pada jaringan tersebut. Kemudian terdapat *Block Metadata* yang berisi informasi mengenai blok tersebut **[24]**.

2.2.4 *Smart Contract*

Smart contract merupakan sebuah program yang dapat dilakukan oleh setiap *peer* pada jaringan *blockchain* untuk melakukan sebuah transaksi [2]. *Smart*

contract terdiri atas tiga bagian yaitu struktur sumber daya (*resource*), akses kontrol (*control access*) dan logika transaksi. Struktur sumber daya pada *smart contract* merupakan templat untuk setiap obyek yang berpengaruh dalam suatu transaksi. Selanjutnya, akses kontrol akan mengatur siapa saja yang dapat mengakses suatu jaringan *blockchain*.. Logika transaksi adalah sebuah fungsi yang menunjukkan apa yang akan dilakukan pada suatu transaksi [2] [11]. *Smart contract* bekerja untuk mendukung pengelolaan siklus lengkap dari kontrak legal yang cerdas [10].

2.2.5 Algoritma *SHA-256*

Algoritma *SHA-256* merupakan salah satu algoritma hash dari jenis *SHA-2* yang menghasilkan *message digest* sepanjang 256 *bit* [25]. Algoritma fungsi *SHA* merupakan perkembangan dari *MD4* yang dibuat oleh Ronald L, Rivest dari *MIT*. *SHA-256* merupakan algoritma *hash* yang menggunakan enam kombinasi logika dasar seperti *AND, OR, XOR, shift-right, rotate-right* [26]. Untuk menghasilkan *message digest* sepanjang 256-bit, algoritma *SHA256* menggunakan sebuah *message schedule* yang dimana terdiri dari 64 elemen 32-bit *word*, delapan buah variabel 32-bit, dan variable penyimpan nilai *hash* sebanyak delapan buah 32-bit *word* [26].

2.2.6 *Android*

Android adalah sebuah sistem operasi untuk perangkat portable seperti *smartphone* dan komputer tablet yang berbasis *Linux*. *Android* merupakan salah satu platform yang terbuka (*open source*) bagi *programmer* untuk mengembangkan aplikasi pada berbagai perangkat yang menggunakan sistem operasi *Android*. *Google* merupakan pengembang dari sistem android yang membeli *Android* inc beserta teknologinya yang kemudian melanjutkan pengembangan dengan membentuk konsorsium *Open Handset Alliance* (*OHA*), yang terdiri dari 34 perusahaan perangkat keras, perangkat lunak, dan telekomunikasi, seperti HTC, Intel, Motorola, Qualcomm, T-Mobile dan Nvidia. Pada awal maret 2009 akhirnya Google berhasil merilis sistem operasi *Android* yang pertama yaitu *Android* versi 1.1 pada perangkat *Smartphone*. Untuk mendukung pengembangan sistem operasi

Android , Google merilis *Android Software Development Kit* (Android *SDK*) disetiap perilisaan sistem operasi Android [27].

2.2.7 *Docker* dan *Container*

Docker merupakan salah satu platform yang bersifat *open-source* bagi pengembang perangkat lunak dan pengelola sistem jaringan untuk membangun, mengirim, dan menjalankan aplikasi yang terdistribusi dengan tidak bergantung pada arsitektur mesin, bahasa pemrograman, dan sistem operasi [28] [29]. Pada *docker* terdapat teknologi virtualisasi untuk mengelola aplikasi server yang bernama *Container*. *Container* dapat memudahkan *system administrator* dalam membangun, mempersiapkan dan menjalankan aplikasi pada server. *Container* dapat membuat aplikasi dari bahasa pemrograman yang berbeda pada lapisan apapun [28] [29]. *Container* bekerja dengan mengisolasi *package* aplikasi, *library*, dan sistem operasi sehingga sangat ideal untuk mengeas layanan *microservices* [29].

2.2.8 *RESTful API*

REST (*Representational State Tranfer*) merupakan salah satu jenis dari *web service* yang memungkinkan *system request* dapat mengakses dan memanipulasi teks yang direpresentasikan dari sebuah *web service*. *RESTful API* merupakan *web service API* yang menggunakan *REST*. *RESTful API* tidak memiliki suatu standar yang resmi untuk notasinya karena *REST* merupakan sebuah arsitektur [30]. *REST* merupakan *web service* yang berfokus pada *resource* atau sumberr daya sistem, seperti bagaimana sumber daya yang dialamatkan dan ditransfer melalui oleh berbagi klien dan ditulis dalam bahasa pemrograman yang berbeda [30]. Setiap sumber daya pada *REST* diidentifikasi dengan *URI* (*Universal Resources Identifiers*) atau global ID, yang nantinya jika *client* mengakses *URI* tersebut maka *REST* akan merespon dengan menampilkan *resource* yang diminta oleh *client* [30].

2.2.9 UML (*Unified Modeling Languange*)

UML atau *Unified Modeling Languange* merupakan sebuah bahasa yang menjadi stanndar dalam industri untuk visualisasi, merancang, dan membuat dokumentasi sistem piranti lunak. UML mudah digunakan dalam membuat model untuk semua jenis piranti lunak. Dimana dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun [31].

2.2.10 *Use Case Diagram*

Use case diagram merupakan gambaran fungsionalitas yang diharapkan dari sebuah sistem. *Use case diagram* merupakan sebuah pekerjaan tertentu, misalnya melakukan registrasi akun, membuat daftar belanja, dan sebagainya. Pada *use case diagram* terdapat sebuah aktor yaitu entitas manusia yang berinteraksii dengan sistem untuk melakukan pekerjaan tertentu [32].

2.2.11 *Activity Diagram*

Activity Diagram merupakan diagram tipe khusus dari diagram *state* yang memperlihahtkkan alur dari suatu aktifitas ke aktifitas lainnya [31]. Oleh karena itu *activity diagram* tidak mennggambarkan *behaviour internal* sebuah sistem secara eksak, tetapi lebih menggambarkan proses aktivitas dari level atas secara umum [32]. *Activity Diagram* penting dalam pemodelan fungsi dalam suatu sistem dan memberi tekanan pada alur kendali antar objek pada sistem [31].