

BAB II

KAJIAN LITERATUR

2.1 *Crowdfunding*

Crowdfunding merupakan sumber pendanaan alternatif yang terus berkembang untuk mendanai ide – ide pemilik usaha nirlaba. Prosesnya dicirikan oleh keberhasilan interaksi antara organisasi yang memfasilitasi kegiatan atau yang sering disebut sebagai *platform*, dengan berbagai *fundraiser* untuk memperoleh pendanaan ide dan usaha dari *funder* yang tertarik untuk melakukan investasi, meminjamkan, maupun berdonasi [10]. Alasan *crowdfunding* diminati dan dapat berkembang pesat dikarenakan munculnya internet. Sebelumnya, para wirausahawan akan mencari keluarga dan temannya untuk mendanai ide dan usaha yang akan dikembangkan, hanya saja pendanaan yang didapatkan sebatas dari orang terdekatnya dan sulit untuk menerima pendanaan dari orang lain. Dengan adanya internet, orang – orang dapat berkumpul pada satu tempat secara *online* untuk membantu pendanaan ide dan usaha [11]. Perkembangan media sosial juga menjadi faktor pendorong munculnya *crowdfunding* di zaman sekarang. Proyek akan lebih mudah didanai karena pengikut di media sosial turut membantu dalam memberikan pendanaan dan juga pengiklanan [11].

Dalam istilah awam, *crowdfunding* merupakan praktik meminta pendanaan untuk kepentingan bisnis maupun proyek dari sekelompok orang yang tersebar di berbagai lokasi. Kata *crowdfunding* berasal dari kata *crowdsourcing*, yang menunjukkan praktik pengumpulan sekelompok orang untuk memperoleh ide, pendapat, dan solusi untuk mengembangkan operasi perusahaan. *Crowdsourcing* memanfaatkan segala aspek individual berupa aset, sumber daya, informasi, dan pengalaman yang dimiliki. Meski demikian, tujuan utama *crowdfunding* adalah untuk memperoleh bantuan finansial, tidak seperti *crowdsourcing* yang tujuannya terbatas hanya untuk memperoleh ide [11].

2.1.1 Tipe *Crowdfunding*

Dalam praktiknya, ada dua tipe *crowdfunding* yang digunakan dalam *crowdfunding platform* [12], yaitu:

- *All or nothing* (AON), *fundraiser* akan menentukan target pendanaan yang jika target pendanaan tersebut tidak tercapai, maka *fundraiser* tidak akan memperoleh dana yang

telah dikumpulkan dan *funder* tidak akan menerima imbalan apapun, dana yang sudah dikontribusikan akan dikembalikan ke tiap *funder*.

- *Keep it all* atau *take it all*, kebalikan dari tipe AON, *fundraiser* dapat menerima seluruh dana yang sudah dikumpulkan pada *campaign* meskipun target pendanaannya tidak tercapai. Meski demikian, biaya operasional yang harus dibayar kepada pihak *platform* umumnya lebih besar dibandingkan tipe AON.

2.1.2 Model *Crowdfunding*

Model pada *crowdfunding* yang sering digunakan ada 4 yaitu [13] :

1. *Equity Based*

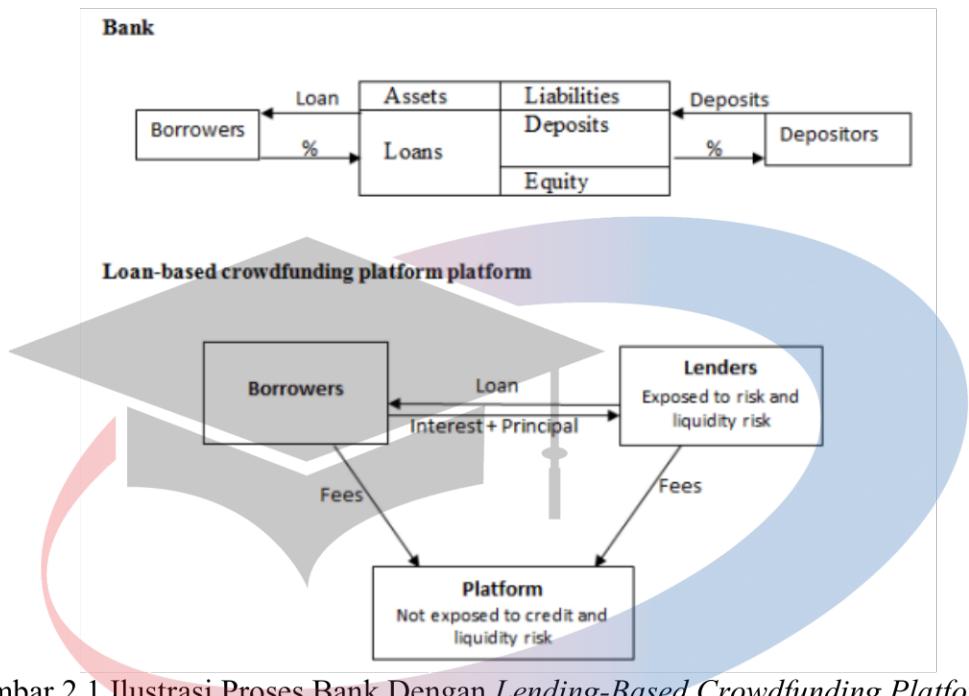
Equity-based merupakan model *crowdfunding* dimana *funder* yang berkontribusi terhadap sebuah *campaign* akan diberikan ekuitas berupa pembagian hak kepemilikan atau pembagian pendapatan di masa depan jika target *campaign* tercapai. Pada model ini, hasil timbal balik yang akan diterima oleh *funder* tergantung pada kesuksesan bisnis di masa mendatang. Oleh sebab itu, *equity-based crowdfunding* memiliki potensi untuk memberikan keuntungan yang besar kepada pihak *funder*. Meski demikian, pihak *funder* tidak terlepas dari kemungkinan menerima kerugian yang besar jika ada hambatan atau permasalahan yang dihadapi oleh bisnis tersebut [13]. AON merupakan tipe *crowdfunding* yang digunakan pada model ini. Keuntungan menggunakan tipe ini adalah agar dapat melindungi dana *funder* dan mendukung sebuah *campaign* untuk mengatur target pendanaan yang realistik namun cukup untuk membiayai target proyek yang sudah ditampilkan kepada *funder* [13].

Contoh *crowdfunding platform* model *equity-based* yang akan dibahas adalah Seedinvest dan Wefunder. Seedinvest menyediakan aplikasi dengan sistem finansial yang inklusif dan terbuka. Dengan fitur *auto invest* yang dimilikinya, *funder* lebih mudah untuk melakukan diversifikasi. Untuk melakukan investasi terhadap *campaign* yang ada di aplikasi ini, maka minimal jumlah kontribusi yang perlu dikeluarkan oleh *funder* adalah \$200. Saat ini, Seedinvest sudah mempunyai 620.000 *funder*. Untuk Wefunder, *platform* ini menyediakan robot asisten yang dapat membantu *auto invest funder* pada berbagai bidang *campaign* seperti obat – obatan, edukasi, inovasi, konstruksi, dan sebagainya. *Platform* ini telah mendapatkan pendapatan sebesar \$300.000.000 per tahun [14].

2. *Lending Based*

Lending-based crowdfunding platform mewakilkan perantara pendanaan terbaru dengan cara menghubungkan *lender* dan *borrower* melalui *platform* internet. Ketika *platform* ini muncul pertama kali di UK dan US, *platform* ini dinamakan sebagai *peer-to-*

peer (P2P) consumer atau *business lending platform* [15]. *Lender* akan menerima hasil persentase bunga pinjaman jika proyek yang diberikan pinjaman tersebut berhasil. Berbeda dengan bank tradisional, *platform* ini tidak melakukan penyaringan *campaign* dan memberi kebebasan kepada para *lender* untuk memilih [12]. Perbedaan proses antara bank dengan *lending-based crowdfunding platform* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Ilustrasi Proses Bank Dengan *Lending-Based Crowdfunding Platform* [15]

Contoh *crowdfunding platform* model *lending-based* yang akan dibahas adalah Ant Check Later dan Zopa. Ant Check Later (ACL) adalah pelayanan dari Alipay yang dimiliki oleh Alibaba di Cina. ACL bekerjasama dengan Taobao, Amazon, dan situs *marketplace* lainnya. Ketika berbelanja di situs yang telah bekerjasama dengan ACL dan melakukan pembayaran menggunakan ACL, maka *borrower* harus melunasi utang tersebut beserta bunganya dalam kurun waktu yang telah ditentukan [16].

Zopa merupakan perusahaan yang didirikan pada tahun 2005 dan sejak dari tahun tersebut, Zopa telah menyetujui 3,2 miliar pinjaman. Angka bunga pinjaman berkisar dari 2.9% - 34.9%, dimana bunganya lebih rendah dibandingkan bank. Pinjaman dapat disetujui ketika *borrower* ingin meminjam dengan jumlah sekian dalam waktu sekian, dan *lender* mau meminjamkan uang tersebut dengan bunga yang sudah ditetapkan olehnya [16].

3. Reward Based

Berbeda dengan model yang telah disebutkan sebelumnya, *reward-based* tidak memberikan imbalan berupa nilai moneter. Pada model ini, *funder* akan menerima imbalan berupa produk nyata, tetapi bukan berupa keuntungan finansial untuk kontribusi mereka [12]. Selain produk yang akan mereka terima, *funder* juga akan menerima kompensasi

abstrak seperti surat terima kasih [17]. *Keep it all* dan *all or nothing*, merupakan tipe yang diterapkan pada model ini [18].

Imbalan yang dipakai pada model *reward-based* jatuh kedalam tiga kategori, yaitu, *pre-order*, pelayanan, dan penghargaan. Biasanya, kategori *pre-order* lebih banyak digunakan oleh *fundraiser* karena hanya perlu memberikan akses awal produk kepada *funder*. Keuntungan lainnya, *fundraiser* dapat mendapatkan *feedback* dari *funder* mengenai akses awal produk sehingga *fundraiser* dapat membuat produk akhir yang memenuhi keinginan *funder*. Pelayanan dan penghargaan merupakan bagian dari imbalan abstrak. Contoh imbalan berupa pelayanan seperti pertunjukan privat, pelatihan, pelayanan komersial gratis, dan sebagainya. Untuk imbalan penghargaan, biasanya dalam bentuk surat terima kasih, maupun berupa penulisan nama kontributor pada website dan produk [17].

Contoh *crowdfunding platform* model *reward-based* yang akan dibahas adalah Kickstarter dan Indiegogo. Kickstarter berdiri pada tahun 2009, dan mulai dari tahun tersebut, *platform* ini telah membantu pendanaan lebih dari 10.000 proyek pembuatan perangkat keras, dan *funders* juga telah menyumbangkan sebesar \$3.000.000.000 [18]. Genre *campaign* jatuh ke kategori – kategori yang sudah dibuat pihak *platform*, yaitu, seni, komik, dansa, desain, *fashion*, film, makanan, permainan, musik, fotografi, penerbitan, teknologi, dan teater. *Fundraiser* biasanya memberikan banyak variasi imbalan, sehingga *funder* dapat melakukan sumbangan kecil seperti \$5 sampai sumbangan besar yang dapat mencapai ribuan dolar [12].

Ketika Kickstarter sering dijadikan topik penelitian, *crowdfunding platform* terbesar kedua, Indiegogo, jarang mendapatkan perhatian [19]. Meski demikian, Indiegogo hanya mengutip *fee* sebesar 4% dari total pendapatan *campaign* dibandingkan Kickstarter yang *fee*-nya mencapai 5%, dan *campaign* di Indiegogo lebih mudah diberi persetujuan [20]. Perbedaan pada aturan dan prosedur dari seluruh *platform* dapat mempengaruhi siapa yang akan mendapat keuntungan terbesar. Seperti Kickstarter, Indiegogo menggunakan model *reward-based*. Perbedaan terletak pada tipenya, dimana *platform* menerapkan tipe *keep it all* dan *all or nothing*. *Campaign* yang tidak mencapai target pendanaan tetap dapat menerima dana tersebut, sehingga tipe yang diterapkan oleh Indiegogo dapat dikatakan sebagai *flexible funding* [19].

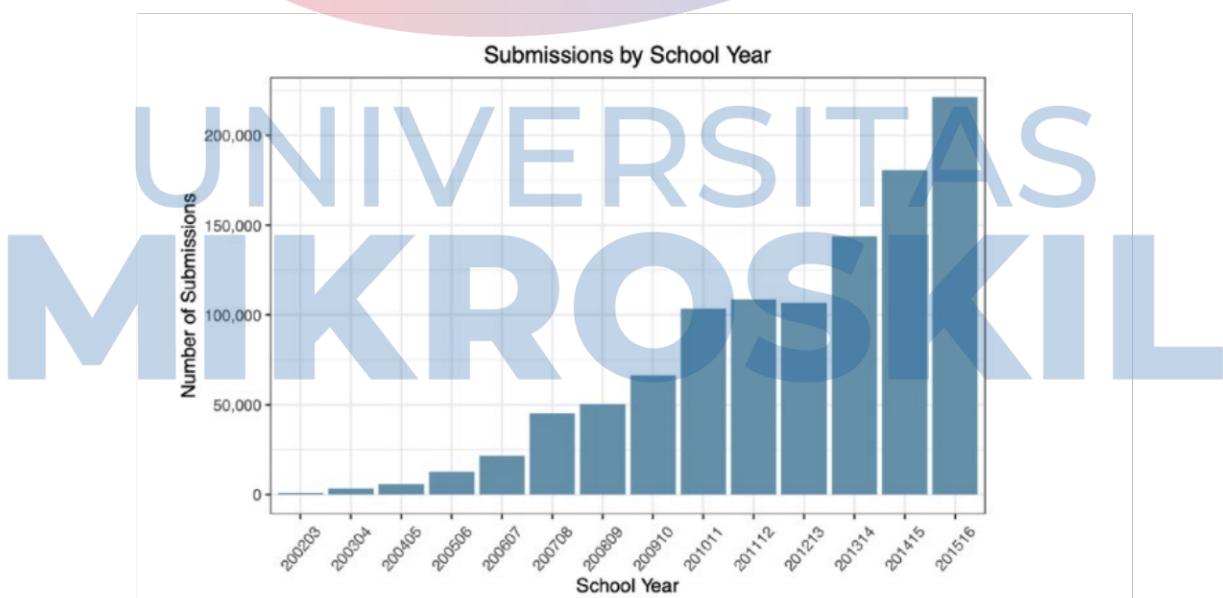
4. *Donation Based*

Internet terus berkembang dan mempermudah kehidupan manusia, praktik donasi juga diadaptasikan ke teknologi ini sehingga membuat *fundraising* tradisional hampir tidak digunakan lagi. *Donation-based* sebagai inovasi filantropis terbaru, mulai mendapatkan

popularitasnya dengan semakin banyaknya proyek yang dibuat dan sukses terealisasikan [21]. Selama dekade ini, perhatian peneliti mulai tertuju ke *donation-based crowdfunding*. Hal ini dikarenakan *donation-based* dapat menjadi sebuah sumber finansial yang menjanjikan bagi organisasi nirlaba. Meskipun hanya memiliki persentase 13-20% keberhasilan pendanaan dari keseluruhan *campaign crowdfunding*, proyek *crowdfunding* nirlaba memiliki angka rata – rata keberhasilan yang lebih tinggi dibandingkan proyek *crowdfunding* untuk laba [22].

Memiliki kesamaan seperti *reward-based*, keberhasilan *donation-based campaign* tergantung preferensi *funder* terhadap karakteristik *campaign*. *Donation-based campaign* mengandalkan kontribusi yang didasari dengan penuh ikhlas demi kepentingan umum. *Fundraiser* tidak menawarkan imbalan moneter selain dari penghargaan kontribusi di dalam suatu komunitas [12].

Contoh *crowdfunding platform* model *donation-based* yang akan dibahas adalah DonorsChoose dan GoFundMe. DonorsChoose berdiri pada tahun 2000 oleh Charles Best, seorang guru sekolah menengah atas di New York City. *Platform* ini sebenarnya diciptakan hanya untuk proyek yang dibuat oleh para guru sekolah publik di New York. Karena kepopulerannya, *platform* ini telah dibuka untuk 50 wilayah di Amerika [23]. Peningkatan penggunaan *platform* DonorsChoose dapat dilihat pada Gambar 2.2.



Gambar 2.2 Perbandingan Pengajuan Ke Donorschoose Berdasarkan Tahun [23]

Agar dapat mengajukan proyek ke DonorsChoose, seorang guru harus bekerja waktu penuh di sebuah sekolah di Amerika Serikat. Proses pengajuan membutuhkan para guru untuk membuat sebuah daftar yang berisikan barang apa saja yang dibutuhkan kelas tersebut. DonorsChoose akan menilai pengajuan tersebut apakah sudah memenuhi segala persyaratan

yang sudah ada atau belum. Setelah melalui seleksi, pengajuan tersebut akan ditampilkan di *platform* agar *funder* dapat melakukan donasi. Menggunakan tipe *all or nothing*, ketika *campaign* tidak mencapai target pendanaan selama 4 bulan, maka dana yang sudah terkumpulkan akan dikembalikan kepada pihak *funder* [23].

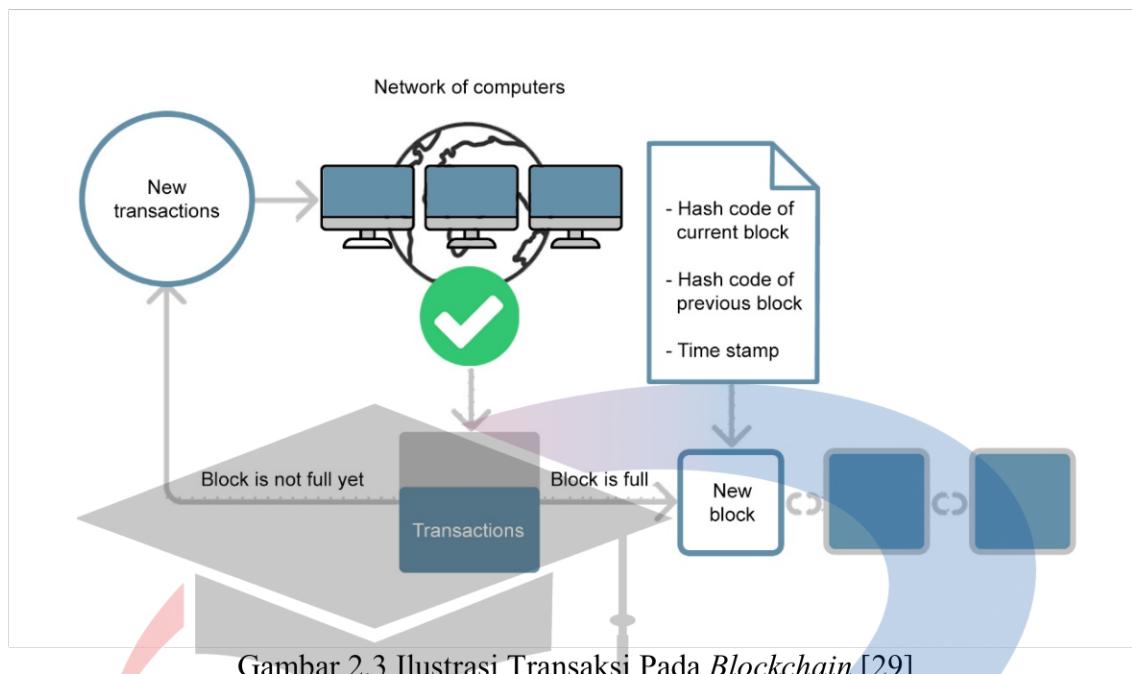
GoFundMe berdiri pada tahun 2010. *Platform* ini mengambil 5% dari donasi pada *campaign* untuk tujuan administratif. Pada tahun 2015, *Campaign* GoFundMe telah mengumpulkan lebih dari \$5.000.000.000 [24]. *Platform* ini memperbolehkan *fundraiser* untuk meminta sumbangan untuk berbagai hal, dari peristiwa kehidupan seperti pernikahan sampai peristiwa kemalangan seperti penyakit dan kecelakaan [25]. Selain digunakan sebagai *donation-based crowdfunding platform*, GoFundMe juga menyediakan kategori untuk *reward-based*, meski demikian, *donation-based* masih menjadi fokus utama dari GoFundMe. Cara GoFundMe melakukan rekomendasi *campaign* kepada calon *funder* adalah dengan cara melihat kategori, area geografi, dan pendanaan untuk pengikut di Facebook [12].

2.2 Blockchain

Sejak ditemukannya Bitcoin oleh Satoshi Nakamoto di tahun 2008, teknologi *blockchain* mulai berkembang pesat dan telah menarik perhatian industri maupun akademik [26]. Teknologi ini dapat diibaratkan sebagai buku besar yang bersifat publik yang seluruh transaksinya dicatat pada rantai *block*. Rantai ini akan terus berkembang seiring dengan penambahan *block* baru ke rantai. Karakteristik utama dari teknologi ini adalah desentralisasi, tangguh, anonimitas, dan kemampuan audit [27]. Karena kelebihannya yang berupa desentralisasi, kekal, aman, dan transparan, *blockchain* menjadi salah satu teknologi generasi selanjutnya untuk sistem *internet-based*, seperti pelayanan publik, *internet of things* (IOT), sistem reputasi, dan pelayanan sekuritas [26].

Dalam *crowdfunding* yang masih tersentralisasi, informasi yang disimpan dalam sebuah kertas seperti kontrak dan daftar pemegang saham atau informasi lainnya, hanya dapat diakses oleh beberapa orang. Kelebihan yang disediakan oleh *blockchain* dapat membuat sistem *crowdfunding* bebas dari kecurangan, lebih transparan, dan aman. *Blockchain* sangat membantu mengurangi pekerjaan yang melelahkan seperti penandatanganan, pendaftaran, otorisasi, dan sertifikasi. Fitur ini dapat membantu *crowdfunding platform* semakin luas digunakan dikarenakan hak dari *funder* jauh lebih aman dan dapat diakses kapanpun dan dimanapun selama ada internet. Kepercayaan *funder* dan *fundraiser* juga meningkat dikarenakan transparansi yang disediakan oleh *blockchain*.

Kredibilitas *crowdfunding platform* menjadi kunci keberhasilan *campaign* dan *platform* [28]. Proses transaksi *blockchain* yang aman dapat dilihat pada Gambar 2.3.



Gambar 2.3 Ilustrasi Transaksi Pada *Blockchain* [29]

Blockchain menyimpan *block* baru secara kronologis dan linier. Tiap *block* ditandai dengan *hash* dan memiliki rekaman waktu serta *hash* dari *block* sebelumnya. Jika ada yang mengubah data *block* sebelumnya, maka akan terjadi perubahan pada *hash* dan jejak waktu pada *block* saat ini. Ini membuat *blockchain* tidak dapat diubah dan mencegah para *hacker* mengubah isi *blockchain*. Serangan hanya akan berhasil jika 51% salinan *blockchain*-nya menyerupai dengan versi yang dimiliki *hacker*. Tetapi ini mustahil untuk dilakukan, dikarenakan *hacker* harus memanipulasi lebih dari setengah salinan *blockchain* sekaligus termasuk mengatur *hash* dan jejak waktu *block* yang terpengaruh [29].

Pada masa sekarang, terdapat 2 tipe *blockchain*, yaitu [30]:

1. Sebuah *blockchain* dikatakan *public* jika partisipan dapat membaca dan menggunakananya untuk menjalankan transaksi, tetapi juga jika semua dapat berpartisipasi dalam proses pembuatan konsensus. Tidak ada pencatatan pusat, dan juga tidak ada pihak ketiga. Dalam sistem ini, *nodes* dalam *network* memvalidasi pilihan yang didiskusikan dan diinisiasi oleh para *developer* dengan membuat keputusan apakah mau mengintegrasikan modifikasi yang sudah diusulkan. Operasi ini berlandaskan pada "*cryptoeconomics*", kombinasi insentif ekonomi dan mekanisme verifikasi menggunakan kriptografi. Berdasarkan komunitas, atau alternatifnya, pendekatan pada ekonomi, sistem ini telah memberikan demonstrasi tentang kekuatannya dan ketangguhannya. Segala *public blockchain* dapat bekerja dengan *coin* atau *token*.

2. Di sisi lainnya, sebuah *blockchain* dikatakan *private* jika proses konsensus hanya dapat dicapai oleh jumlah partisipan yang terbatas. Akses penulisan hanya diberi oleh organisasi dan akses pembacaan dapat dibuat menjadi *public* maupun dibatasi. Pada kasus ini, proses konsensus dikontrol oleh *nodes* yang sudah terseleksi. *Private blockchain* tidak menggunakan mekanisme yang berdasarkan kriptografi. Untuk kasus *private blockchain*, tidak ada yang namanya *mining*, *proof of work*, dan *remuneration*. Ini yang membedakan kedua tipe teknologi transmisi dan penyimpanan.

2.2.1 Komponen *Blockchain*

Blockchain dapat terlihat rumit, tetapi, teknologi ini dapat disederhanakan dengan memeriksa tiap komponennya. Pada tingkat tinggi, *blockchain* memanfaatkan mekanisme ilmu komputer ternama dan kriptografi primitif (*cryptographic hash functions*, *digital signatures*, *asymmetric-key cryptography*). Berikut penjelasan mengenai komponen inti *blockchain* [31] :

1. *Cryptographic Hash Functions*

Hashing merupakan mekanisme yang melibatkan konversi input sepanjang apapun menjadi sebuah keluaran *string of text* dengan ukuran tetap menggunakan fungsi matematika. Data pada web berurusan dengan informasi penting dan membawa berbagai jenis informasi sensitif, oleh sebab itu dibutuhkan perencanaan keamanan yang matang. Jika yang tidak berkepentingan dapat mengakses data bisnis atau sensitif, maka terjadilah kebocoran privasi. Data yang telah terenkripsi oleh *hashing* membuat data menjadi sulit untuk diakses oleh *hacker*. *Hash* mengompres berbagai tipe pesan seperti gambar, video, dan lain sebagainya menjadi pesan terenkripsi dengan ukuran tetap yang kondisi tiap *hash*-nya harus berbeda antara satu dengan lainnya [32]. Terdapat berbagai jenis algoritma *hash* dan perbandingannya dapat dilihat pada Tabel 2.1.

Tabel 2.1 Perbandingan Algoritma *Hash* [33]

Parameter	MD5	SHA-1	SHA-2	SHA-3	Whirlpool
<i>Block size (bits)</i>	512	512	512, 1024	1600-2*bits	512
<i>Digest size (bits)</i>	128	160	160, 224, 256, 384, 512	160, 224, 256, 384, 512	512
<i>Word size</i>	32	32	32, 64	64	8

<i>(bits)</i>					
<i>Rounds</i>	4	80	80	24	10
<i>Collision found</i>	Yes	<i>Theoretical attack</i>	<i>None</i>	<i>None</i>	Yes
<i>Operations</i>	AND, OR, XOR, ROT	AND, OR, XOR, ROT	AND, OR, XOR, ROT, SHR	AND, OR, XOR, ROT, SHR	AND, OR, XOR, NOT

Secara tidak resmi, fungsi *hash* diekspetasikan harus memiliki properti utama sebagai berikut [34]:

- *Preimage*

Fungsi hash harus fungsi searah. Misalnya, ada pesan digest y , tidak mungkin dapat diperoleh nilai masukan dari pesan y tersebut dimana $H(m) = y$.

- *Second-Preimage*

Diberikan sebuah pesan m dan pesan *digest* $H(m)$, tidak mungkin dapat diperoleh nilai masukan yang berbeda untuk mencari nilai *digest* yang sama dimana $H(m) = H(m')$. Fungsi searah memiliki sifat *preimage resistance* dan *second-image resistance*. Hanya saja, dimungkinkan untuk membuat fungsi yang *second-image resistance*, tetapi tidak *preimage resistance*. Karena perbedaan ini, maka *preimage resistance* dan *second-image resistance* dipertimbangkan untuk memiliki artian berbeda.

- *Collision*

Tidak mungkin dapat ditemukan 2 masukan yang menghasilkan pesan *digest* yang sama dimana $m \neq m'$ untuk setiap $H(m) = H(m')$. Tabrakan tercipta dikarenakan prinsip *pigeon-hole*, tetapi untuk fungsi *hash* secara komputasi tidak mungkin dapat ditemukan. *Collision resistance* mengindikasikan *second-image resistance*, hanya saja memiliki artian yang lebih kuat.

- *Function Behavior*

Selalu ada properti implisit yang datang dari fungsi matematika apapun. Properti fungsi ini membutuhkan untuk setiap pesan m apapun, hasil *digest* $H(m)$ selalu deterministik. Jika m tidak diubah, maka akan selalu menghasilkan $H(m)$ yang sama.

Cryptographic hash functions yang sering diimplementasikan pada *blockchain* adalah *Secure Hash Algorithm* (SHA) dengan ukuran output 256-bit (SHA-256). Banyak komputer yang mendukung algoritma ini sehingga cepat untuk dikomputasikan. SHA-256

memiliki keluaran 32 bits, umumnya ditampilkan sebagai *string* 64 karakter heksadesimal [31]. Contoh *input* dan *output*-nya dari algoritma *hash* SHA-256 dapat dilihat pada Tabel 2.2.

Tabel 2.2 Contoh Teks Input Dan Nilai *Digest*-nya [31]

Input	SHA-256 Digest Value
1	0x6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d 49c01e52ddb7875b4b
2	0xd4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f 90da3a666eec13ab35
Hello, World!	0xdfdf6021bb2bd5b0af676290809ec3a53191dd81c7f70a 4b28688a362182986f

2. *Transactions*

Transaksi pada sistem *blockchain* memiliki kemiripan dengan *online transaction processing* (OLTP) yang bergerak berdasarkan kumpulan data. Aplikasi *blockchain* tradisional seperti Bitcoin mempunyai transaksi yang mewakilkan sebuah pertukaran uang antara dua entitas atau *user*. Setiap transaksi yang valid direkam dalam sebuah *block* yang dapat menyimpan dua atau lebih transaksi. Ketetapan dapat dicapai dengan memanfaatkan properti kriptografi yang kuat seperti *hashing* [35]. Sebuah transaksi *cryptocurrency* umumnya membutuhkan informasi berikut ini, tetapi dapat juga lebih [31]:

- *Inputs*

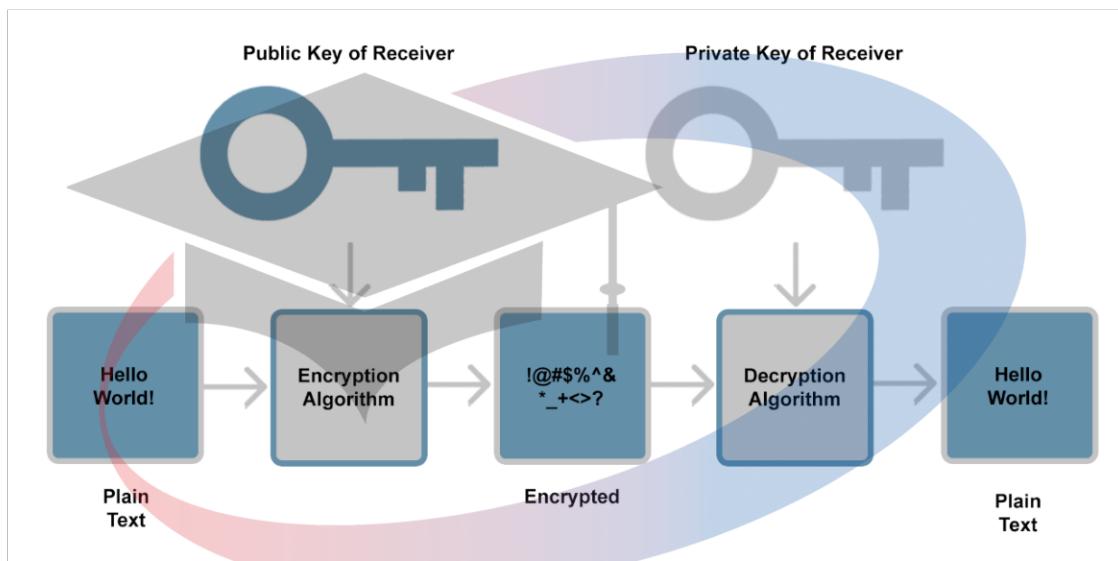
Masukan biasanya berupa daftar dari aset digital yang akan ditransfer. Transaksi mereferensikan sumber dari aset digital tersebut. Sumber tersebut dapat berasal dari transaksi sebelumnya dimana diberikan kepada *sender*, atau untuk kasus aset digital baru, pada peristiwa pembentukannya. Dikarenakan masukan dari transaksi berdasarkan peristiwa masa lampau, aset digital tidak dapat berubah. Untuk kasus *cryptocurrencies*, nilai tidak dapat ditambahkan atau dibuang dari aset digital yang sudah ada.

- *Outputs*

Keluaran biasanya berupa akun yang akan menjadi *recipients* dari aset digital tersebut beserta jumlah aset digital yang akan diterima. Tiap keluaran menspesifikasikan jumlah aset digital yang akan ditransfer kepada pemilik baru, *identifier* pemilik baru, dan kondisi - kondisi yang harus dipenuhi untuk menggunakan aset tersebut. Jika aset digital yang diberikan ternyata berlebih, maka aset lebih harus dikembalikan kepada *sender*.

3. Asymmetric-Key Cryptography

Asymmetric-key cryptography dikenal sebagai kriptografi *public key*. Dalam teknik ini, *sender* menggunakan *public key* milik *receiver* untuk enkripsi dan *receiver* menggunakan *private key*-nya untuk dekripsi pesannya. Konsep *self-certification* tidak digunakan lagi, melainkan *digital signatures* yang dipergunakan sebagai alat untuk mensertifikasikan kuncinya. Metode ini lebih mudah dan memberikan autentikasi yang lebih baik serta privasi yang terjaga [36, 37]. Ilustrasi proses penggunaan *asymmetric-key cryptography* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Ilustrasi *Asymmetric-Key Cryptography* [36]

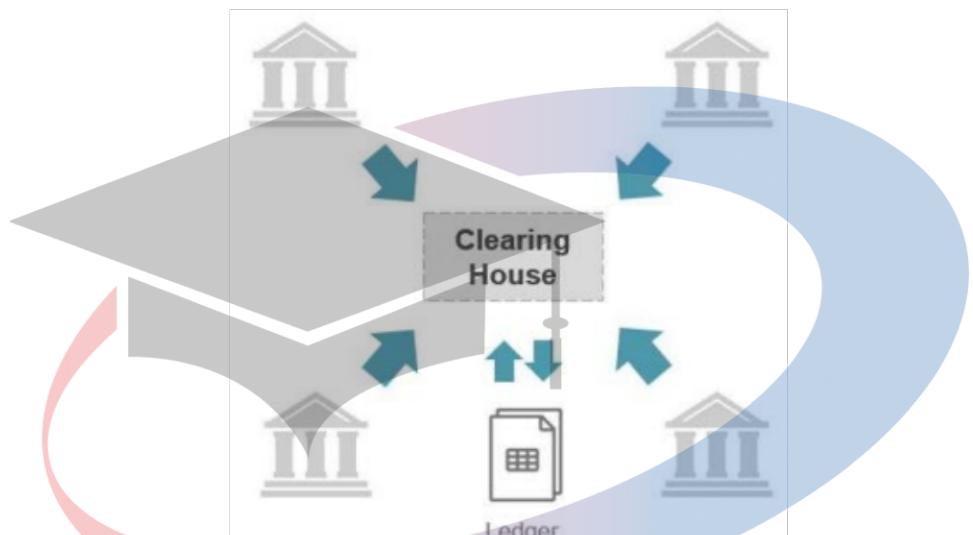
4. Address

Address digunakan oleh beberapa jaringan *blockchain* sebagai titik awal dan titik akhir suatu transaksi. Isi dari *address* yaitu sebuah *string* alfanumerik pendek yang karakternya diambil dari *public key user* pada *blockchain* menggunakan *hashing*. Salah satu metode untuk menghasilkan *address* adalah dengan membuat *public key*, lalu dilakukan *hashing* yang nantinya diubah lagi menjadi teks. Ada berbagai cara untuk mencari *address* berdasarkan *blockchain* yang digunakan. Jaringan *permissionless blockchain* mampu membuat akun yang anonim dan *blockchain* mampu menghasilkan pasangan *asymmetric-key* sebanyak mungkin, begitu juga *addresses* yang diinginkan. Hal ini memungkinkan akun untuk memiliki tingkat anonim yang beragam [31].

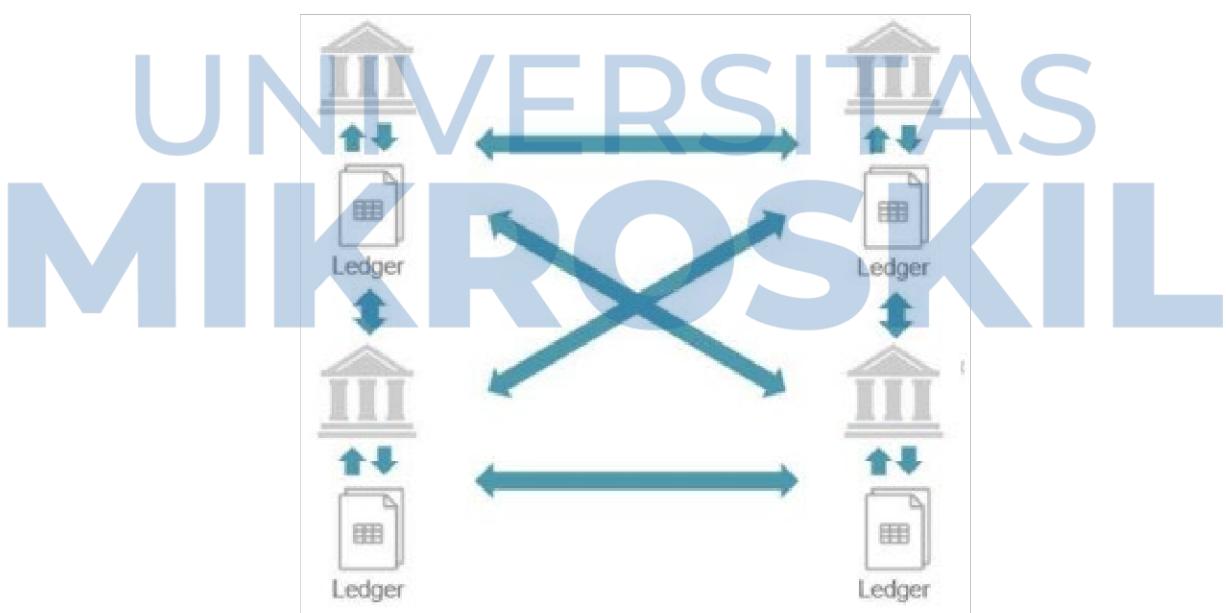
5. Ledgers

Kumpulan dari transaksi disebut sebagai dengan *ledger*. Sebelum memasuki era modern, semua transaksi dicatat menggunakan pena dan kertas. Era sekarang, transaksi sudah dapat dicatat dan disimpan secara digital. *Centralized ledger* dan *distributed ledger* adalah jenis dari *ledger* yang digunakan [31]. Banyak perusahaan yang masih menggunakan

centralized ledger untuk mencatat transaksi. Sistem *ledger* ini mencatat semua transaksi dimana transaksi tersebut hanya diatur oleh satu pihak atau hanya berpusat pada satu titik. Selain itu *centralized ledger* tidak memiliki batasan sehingga pengguna dapat dengan bebas mengubah data pada *ledger*. Hal ini dapat dimanfaatkan untuk membuat transaksi dengan pernyataan yang salah sehingga dapat memungkinkan terjadinya kecurangan. *Distributed ledger* dibuat untuk mengatasi persoalan tersebut [38]. Perbedaan proses pencatatan pada *centralized ledger* dengan *distributed ledger* dapat dilihat pada Gambar 2.5 dan Gambar 2.6.



Gambar 2.5 Ilustrasi Proses Pencatatan *Centralized Ledger* [38]



Gambar 2.6 Ilustrasi Proses Pencatatan *Distributed Ledger* [38]

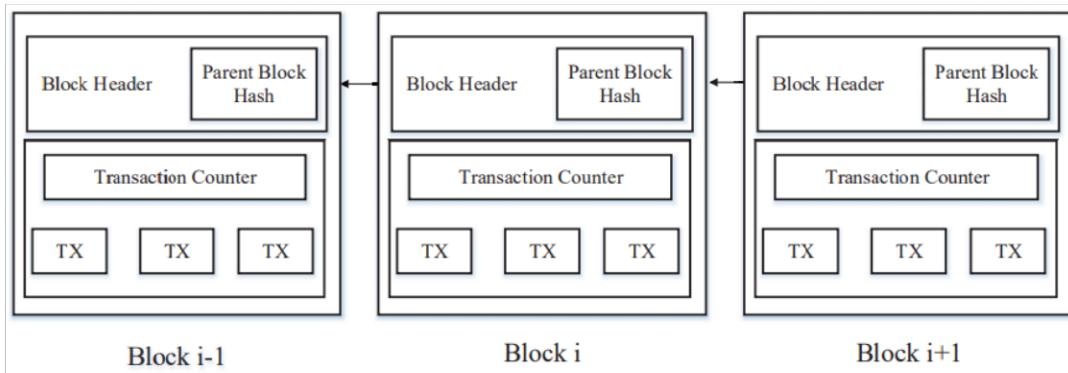
Distributed ledger dapat diakses semua orang dan transaksi yang didalamnya akan sama dengan apa yang dilihat oleh lainnya. Data *ledger* ini tersebar dan tidak dipegang oleh satu entitas. Transaksi yang dicatat semua disinkronisasikan untuk menjaga ketangguhan, integritas, dan ketersediaannya. Tidak seperti *centralized ledger*, jika ada kesalahan yang membuat sistem berhenti untuk bekerja, data yang tersebar tadinya aman dan dapat digunakan sebagai cadangan. Hal ini mampu mengurangi terjadinya kesalahan dan informasi yang diberikan dapat dilihat langsung secara *real-time* [38]. *Blockchain* menggunakan sistem *distributed ledger* sehingga transaksi dapat diakses oleh siapa saja dan dimana saja, dan ini yang memunculkan sifat desentralisasi pada *blockchain* [39]. Untuk perbandingan *distributed ledger* dengan *centralized ledger* dapat dilihat pada Tabel 2.3.

Tabel 2.3 Perbandingan *Distributed Ledger* dengan *Centralized Ledger* [38]

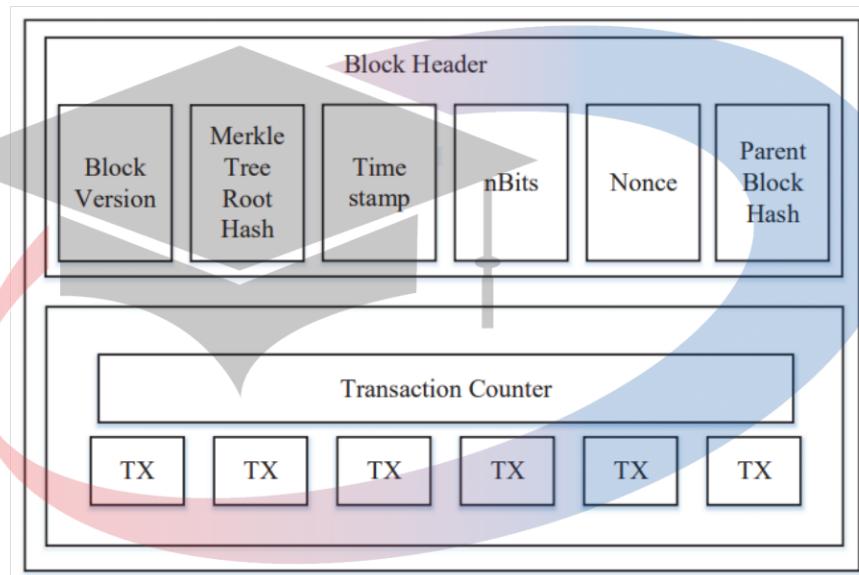
<i>Distributed Ledger</i>	<i>Centralized Ledger</i>
Adanya konsensus pada data	Membutuhkan penyesuaian internal dan eksternal
Transaksi tidak dapat diubah	Tidak ada batasan perubahan
Tersebar	<i>Single point of failure</i>
Terdesentralisasi	Tersentralisasi
<i>Peer-to-peer</i>	Menggunakan <i>gateway</i> dan perantara yang harusnya tidak diperlukan
Verifikasi menggunakan kriptografi	Kriptografi hanya diimplementasi jika diinginkan
Autentikasi dan otorisasi kriptografi	Aksi dilandaskan dari pihak lainnya
Ketangguhan dan ketersediaan meningkat berdasarkan jumlah <i>node</i>	Harus membuat cadangan secara manual

6. *Block* Dan *Chaining Blocks*

Segala bentuk informasi pada *blockchain* disimpan pada *block*. *Block* terhubung dengan *block* lainnya dengan menggunakan nilai *hash* pada *block* sebelumnya yang disebut sebagai *parent block* [40]. Jika *block* sebelumnya diubah, maka akan diterima sebuah *hash* yang berbeda. Karena perubahan ini, maka nilai *hash* pada *block* seterusnya akan berganti. Hal ini memberikan kemudahan dalam membuang *block* yang sudah tidak murni atau diubah [31]. Gambaran *chaining blocks* dapat dilihat pada Gambar 2.7, dan untuk struktur *block*-nya dapat dilihat pada Gambar 2.8.



Gambar 2.7 Ilustrasi *Chaining Blocks* [27]



Gambar 2.8 Bentuk Dan Struktur *Block* [27]

Block awal dari setiap *blockchain* memiliki istilah *genesis block*. *Genesis block* adalah *block* spesial bermotor nol, dan di-hard-coded di seluruh aplikasi *blockchain* [35]. Sebuah *block* memiliki *header* dan *body*. *Header* terdiri atas *block version*, *parent block hash*, *Merkle tree root*, *timestamp*, *nBits*, dan *nonce* [40]. Penjelasan atribut ini dapat dilihat pada Tabel 2.4.

Tabel 2.4 Atribut Pada *Block Header* [40]

Atribut <i>Header</i>	Definisi
<i>Block version</i>	Mengindikasikan aturan validasi <i>block</i> apakah yang harus diikuti
<i>Previous block hash</i>	Nilai <i>hash</i> yang dimiliki <i>block</i> sebelumnya
<i>Merkle tree root</i>	Nilai <i>hash</i> dari semua transaksi yang ada pada <i>block</i>

<i>Timestamps</i>	Jejak waktu saat ini menggunakan detik dimulai dari 1970-01-01T00:00 UTC
<i>nBits</i>	Nilai target pada nilai <i>hash</i>
<i>Nonce</i>	Sebuah <i>field</i> 4-bits, dimana biasanya dimulai dari 0 dan bertambah seiring kalkulasi <i>hash</i>

Block body terdiri atas *transaction counter* dan *transactions*. Ukuran *block* dan ukuran tiap transaksi berpengaruh terhadap banyaknya transaksi yang dapat disimpan dalam sebuah *block*. Agar dapat mengecek autentikasi setiap transaksi, maka *blockchain* menggunakan *asymmetric-key cryptography* untuk melakukan validasi. *Digital signature* dipakai jika tidak ada faktor kepercayaan pada sebuah *environment* [40].

2.2.2 Konsensus

Agar transaksi *blockchain* dianggap berhasil, *block* dari transaksi itu harus ditambahkan ke rantai *block* yang telah dikenal oleh seluruh *node* dalam jaringan. Sebuah *node* akan mem-*broadcast* *block* ke node lain agar *block* ini juga ditambahkan ke rantai *block* dari setiap *node* yang ada. Hal ini dapat mengakibatkan kebingungan ketika setiap *node* mem-*broadcast* *block* yang ditemukannya yang juga ditemukan oleh *node* lain. Untuk menghindari hal tersebut, digunakanlah algoritma konsensus untuk membuat kesepakatan antara semua *node* mengenai *node* mana yang diizinkan untuk menambahkan *block* dan *block* mana yang akan ditambahkan [41].

Algoritma konsensus juga memungkinkan *user* yang tidak saling mengenal maupun saling percaya satu sama lain untuk bekerja sama. Ketika *user* menggunakan jaringan *blockchain*, mereka telah setuju terhadap keadaan awal sistem yang disimpan dalam *genesis block*. *Genesis block* merupakan *block* pertama yang ada dalam setiap jaringan *blockchain* sehingga setiap *block* baru wajib ditambahkan ke jaringan setelahnya berdasarkan algoritma konsensus yang disepakati. Terlepas dari algoritma konsensus yang digunakan, setiap *block* harus bersifat valid dan dapat divalidasi secara independen oleh setiap *user* jaringan *blockchain* [31].

Berikut properti yang ada pada algoritma konsensus [31]:

1. Keadaan awal sistem yang disepakati (*genesis block*)
2. *User* setuju dengan algoritma konsensus yang menentukan *block* mana yang akan ditambahkan ke sistem

3. Setiap *block* berkaitan dengan *block* sebelumnya karena menyimpan *header hash block* sebelumnya kecuali *genesis block* yang merupakan *block* pertama
4. *User* dapat memverifikasi setiap *block* secara independen

Terdapat beberapa jenis algoritma konsensus yaitu:

1. *Proof Of Work*

Pada algoritma *Proof of Work* (PoW), *node* yang dapat menambahkan *block* adalah yang pertama kali menyelesaikan *puzzle* komputasional. *Puzzle* tersebut didesain sedemikian rupa sehingga sulit untuk diselesaikan namun mudah untuk mengecek apakah solusinya valid. Hal ini membuat *full node* lain dapat dengan mudah memvalidasi setiap *block* yang diajukan dan menolak *block* yang tidak menyelesaikan *puzzle*. Metode *puzzle* umumnya mengharuskan nilai *hash* dari *block header* kurang dari nilai yang ditentukan. *Publishing node* akan melakukan perubahan pada *block header* mereka untuk mencari nilai *hash* yang memenuhi kriteria *puzzle*. Persyaratan nilai *puzzle* dapat dimodifikasi dari waktu ke waktu untuk menyesuaikan tingkat kesulitan *puzzle* yang mengatur seberapa sering *node* ditambahkan ke jaringan [31].

Sebagai contoh, terdapat sebuah *puzzle* yang menggunakan algoritma SHA-256, maka komputer harus mencari nilai *hash* yang memenuhi target berikut:

SHA256("blockchain" + Nonce) = Hash Digest starting with "000000"

Pada contoh ini, teks "blockchain" ditambahkan dengan nilai *nonce* untuk menghitung nilai *hash*. Hasil keluarannya berupa:

SHA256("blockchain0") =
0xbd4824d8ee63fc82392a6441444166d22ed84eaa6dab11d4923075975ac
ab938 (not solved)

SHA256("blockchain1") =

0xdb0b9c1cb5e9c680dff7482f1a8efad0e786f41b6b89a758fb26d9e223
e0a10 (not solved)

...

SHA256("blockchain10730895") =
0x000000ca1415e0bec568f6f605fcc83d18cac7a4e6c219a957c10c6879d
67587 (solved)

Untuk menyelesaikan *puzzle* tersebut, dibutuhkan 10.730.896 percobaan yang selesai dalam 54 detik. Pada contoh berikutnya, masalah *puzzle* tetap sama namun kesulitan *puzzle*

dingkatkan dengan menambahkan awalan digit nol (*leading zero*) yang tadinya berjumlah enam digit menjadi tujuh digit nol. Untuk menyelesaiannya, dibutuhkan 934.224.175 percobaan yang selesai dalam 1 jam 18 menit 12 detik.

```
SHA256("blockchain934224174") =  
0x0000000e2ae7e4240df80692b7e586ea7a977eacbd031819d0e603257ed  
b3a81
```

Umumnya, *publishing nodes* pada algoritma PoW akan saling bekerja sama dalam kelompok untuk menyelesaikan *puzzle* dan berbagi insentif yang didapatkan. Hal ini dapat dilakukan karena pekerjaan dapat didistribusikan kepada dua *node* atau lebih dalam satu kelompok untuk saling berbagi beban kerja dan insentif. Dibawah ini adalah contoh beban pekerjaan yang dipisahkan menjadi empat bagian, setiap *node* mengambil nilai *nonce* dari rentang yang diberikan untuk melakukan percobaan:

```
Node 1: check nonce 0000000000 to 0536870911  
Node 2: check nonce 0536870912 to 1073741823  
Node 3: check nonce 1073741824 to 1610612735  
Node 4: check nonce 1610612736 to 2147483647
```

Berikut solusi *puzzle* yang pertama kali diperoleh:

```
SHA256("blockchain1700876653") =  
0x00000003ba55d20c9cbd1b6fb34dd81c3553360ed918d07acf16dc9e75  
7c7f1
```

Meski nilai *nonce* yang digunakan berbeda dari contoh sebelumnya, ia tetap dapat menyelesaikan *puzzle*. Dibutuhkan 90.263.918 percobaan yang diselesaikan dalam 10 menit 14 detik untuk memperoleh keseluruhan solusi yang ada. Dengan berbagi pekerjaan, hasil yang diperoleh akan lebih baik dan insentif yang didapatkan akan lebih konsisten [31].

2. *Proof Of Stake*

Algoritma *Proof of Stake* (PoS) didasarkan pada konsep jika semakin banyak *stake* yang diinvestasikan oleh *node*, maka semakin besar keinginan mereka agar sistem dapat berhasil. *Stake* umumnya merupakan besaran *cryptocurrency* yang diinvestasikan *node* ke jaringan blockchain yang dapat dilakukan dengan menguncinya dengan transaksi khusus, atau mengirimkannya ke *address* tertentu, atau dapat pula dengan menahannya dalam *wallet* khusus. *Cryptocurrency* yang telah diinvestasikan umumnya tidak dapat digunakan lagi. Jaringan *blockchain* PoS menggunakan besaran investasi *node* untuk menentukan *node* mana yang akan menjadi *publishing node*. Oleh karena itu, ada kaitan antara besaran investasi yang dilakukan *node* dengan pemilihan *publishing node* baru kedepannya [31].

Terdapat empat metode pemilihan *publishing node* yang digunakan oleh jaringan *blockchain PoS* yaitu *random choice*, *multi-round voting*, *coin aging systems* dan *delegate systems*. Meski demikian, *node* dengan *stake* terbanyak tetap memiliki kemungkinan lebih besar untuk menjadi *publishing node*. Berikut penjelasan dari keempat metode tersebut [31]:

- *Random Choice*

Pada metode ini, jaringan *blockchain* akan memantau semua *node* dan memilih berdasarkan rasio *stake* yang dimiliki. Jika keseluruhan rasio yang dimiliki *node* adalah 42%, maka kemungkinan terpilihnya adalah 42% dan *node* yang memiliki rasio 1% memiliki kemungkinan 1% untuk terpilih.

- *Multi-Round Voting*

Pada metode ini, sistem pemilihan yang dilakukan lebih rumit. Jaringan *blockchain* akan memilih beberapa *node* yang memiliki *stake* untuk mengajukan *blocks* yang kemudian akan divoting oleh *node* lain yang memiliki *stake* juga. Proses pemilihan ini mungkin dapat dijalankan beberapa kali sebelum akhirnya ada *block* yang dipilih. Metode ini memungkinkan *node* yang memiliki *stake* untuk memiliki hak suara dalam setiap pemilihan *block* baru.

- *Coin Aging Systems*

Pada metode ini, *stake* yang diinvestasikan oleh *node* memiliki umur yang akan menjadi faktor penentu pemilihan *publising node*. Setelah *node* terpilih, umur *stake* akan direset kembali dan *stake* tidak dapat digunakan lagi hingga batas waktu tertentu. Metode ini memungkinkan *node* dengan *stake* yang lebih banyak untuk menambahkan lebih banyak *block*, namun tidak mendominasi sistem karena adanya *cooldown timer*. *Stake* dengan umur dan jumlah yang lebih besar akan meningkatkan kesempatan terpilihnya sebuah *node*. Untuk mencegah *node* melakukan penimbunan *stake*, umumnya terdapat umur maksimum yang dijadikan sebagai batasan.

- *Delegate Systems*

Pada metode ini, pemilihan *publishing node* dilakukan dengan voting oleh *node* yang ada pada jaringan. Besarnya efek *vote* dari *node* ditentukan oleh besaran *stake* yang dimiliki, semakin besar *stake*-nya, semakin besar pula efek *vote*-nya. *Node* yang mendapatkan *vote* terbanyak akan menjadi *publishing node*. Setelahnya, *node* lain masih dapat melakukan voting untuk menurunkan *publishing node* karena voting dilakukan secara terus menerus dan bersifat kompetitif. Oleh karena itu, *publishing node* diberikan insentif sebagai imbalan atas hasil kerjanya dan agar tidak melakukan kesalahan. Selain itu, *node* juga akan melakukan voting untuk memilih delegasi yang akan berpartisipasi

dalam memberikan ide perubahan dan perbaikan yang kemudian akan divoting kembali oleh *node* pada jaringan *blockchain*.

3. Round Robin

Algoritma *Round Robin* umumnya digunakan pada jaringan *permissioned blockchain*, atau dapat disebut juga dengan jaringan yang bertipe *private*. Pada algoritma ini, setiap *node* bergiliran menjadi *publishing node*. Untuk menghindari situasi dimana *publishing node* tidak dapat bekerja ketika gilirannya, sistem memberikan batas waktu untuk mengaktifkan *node* lain yang tersedia sebagai *publishing node*. Oleh karena itu, *publishing node* yang tidak dapat bekerja pada saat gilirannya tidak akan menghambat pembuatan *block* baru. Model ini menjamin tidak ada satu *node* yang dapat membuat sebagian besar *block* dalam suatu jaringan *blockchain* [31].

4. Proof Of Identity/Proof Of Authority

Algoritma *Proof of Identity* atau yang juga disebut dengan *Proof of Authority* hanya digunakan pada jaringan *permissioned blockchain* karena mengandalkan kepercayaan parsial dari *publishing nodes* melalui identitas mereka di dunia nyata. *Publishing nodes* harus memiliki identitas yang sah dan dapat diverifikasi dalam jaringan *blockchain*. Konsep algoritma ini adalah *node* menginvestasikan identitasnya sebagai *stake* untuk dapat menjadi *publishing node*. Reputasi *publishing node* dipengaruhi oleh *user* jaringan *blockchain*, reputasi akan naik jika *publishing node* melakukan hal yang disetujui oleh *user* dan akan menurun jika melakukan hal yang tidak disetujui oleh *user*. Semakin tinggi reputasinya, semakin besar kemungkinannya untuk dapat membuat *block* dan begitu juga sebaliknya [31].

5. Proof Of Elapsed Time

Pada algoritma *Proof of Elapsed Time* (PoET), setiap *publishing node* melakukan *request* waktu tunggu (*wait time*) dari sumber waktu pada perangkat keras dalam sistem komputer. *Publishing nodes* akan menerima respon dari sistem berupa waktu acak yang kemudian dijadikan *idle time*-nya. Setelah *publishing node* aktif kembali, *block* baru akan ditambahkan ke jaringan *blockchain* olehnya dan *publishing node* lain yang sedang *idle* akan berhenti menunggu dan seluruh proses dimulai kembali dari awal [31].

6. Proof Of History

Algoritma *Proof of History* (PoH) merupakan komputasi yang menyediakan cara untuk memverifikasi *passage of time* dari dua peristiwa secara kriptografis. Dengan menggunakan fungsi yang diamankan secara kriptografis, *output* tidak dapat diprediksi dari inputan sehingga algoritma harus selesai dieksekusi dahulu untuk mendapatkan *output*. Fungsi akan dijalankan pada *single core* dengan menggunakan *output* sebelumnya sebagai

inputan pengeksekusian saat ini, hasil dan jumlah pemanggilan yang ada saat ini akan dicatat secara berkala. *Output* ini nantinya dapat diverifikasi oleh komputer eksternal dengan melakukan pengecekan setiap segmen pada *core* yang terpisah [42].

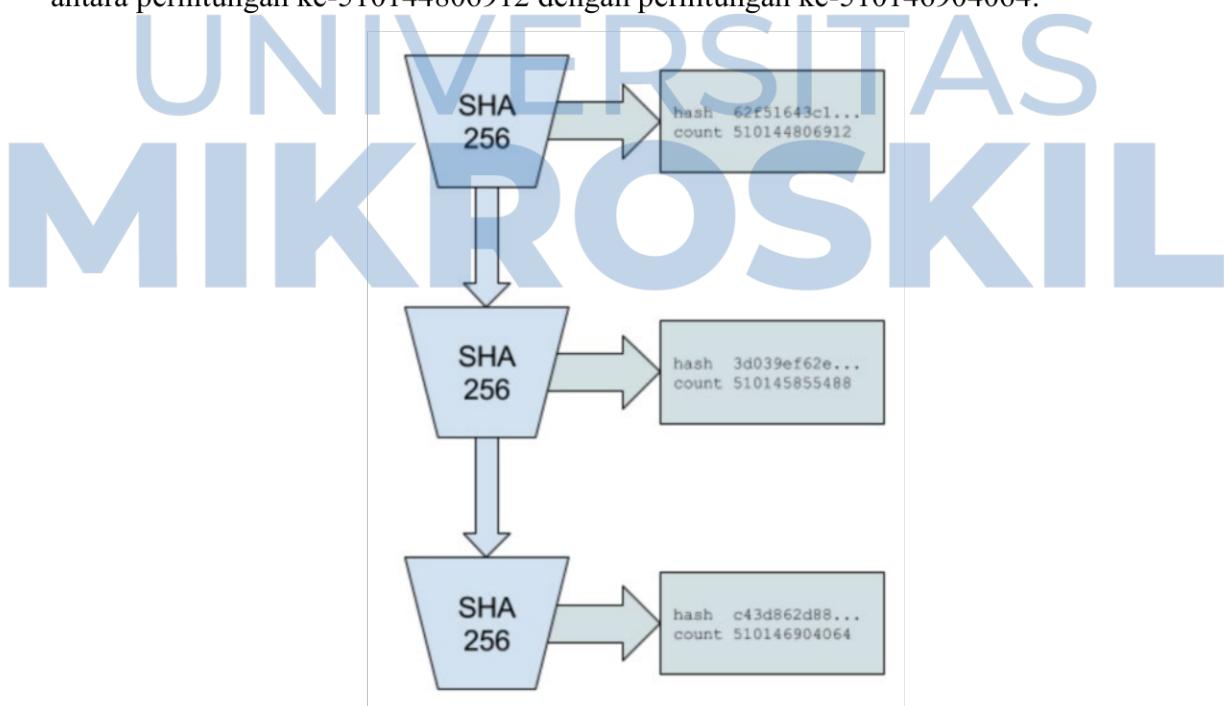
Operasi dimulai dengan fungsi yang mengambil nilai acak sebagai inputan, lalu *output*-nya akan dijadikan sebagai inputan untuk operasi selanjutnya. Berikut contoh operasi algoritma PoH yang dapat dilihat pada Tabel 2.5.

Tabel 2.5 PoH Sequence [42]

Index	Operation	Output Hash
1	sha256("any random starting value")	hash1
2	sha256(hash1)	hash2
3	sha256(hash2)	hash3

Selama fungsi *hash* yang terpilih tahan benturan (*collision resistant*), kumpulan *hash* ini hanya dapat dikomputasikan pada *single thread*. Ini sejalan dengan ketentuan bahwa *output index* ketiga tidak dapat diprediksi jika tidak menjalankan algoritma dari *index* pertama. Berdasarkan struktur data tersebut, dapat dipastikan bahwa ada waktu nyata (*real time*) yang telah berlalu antara *index* pertama dengan *index* ketiga.

Pada contoh berikut yaitu pada Gambar 2.9, *hash* 652f51643c1 dihasilkan pada perhitungan ke-510144806912 dan *hash* c43d862d88 dihasilkan pada perhitungan ke-510146904064. Pada algoritma PoH, dapat dipastikan bahwa ada waktu nyata yang berlalu antara perhitungan ke-510144806912 dengan perhitungan ke-510146904064.



Gambar 2.9 PoH Sequence [42]

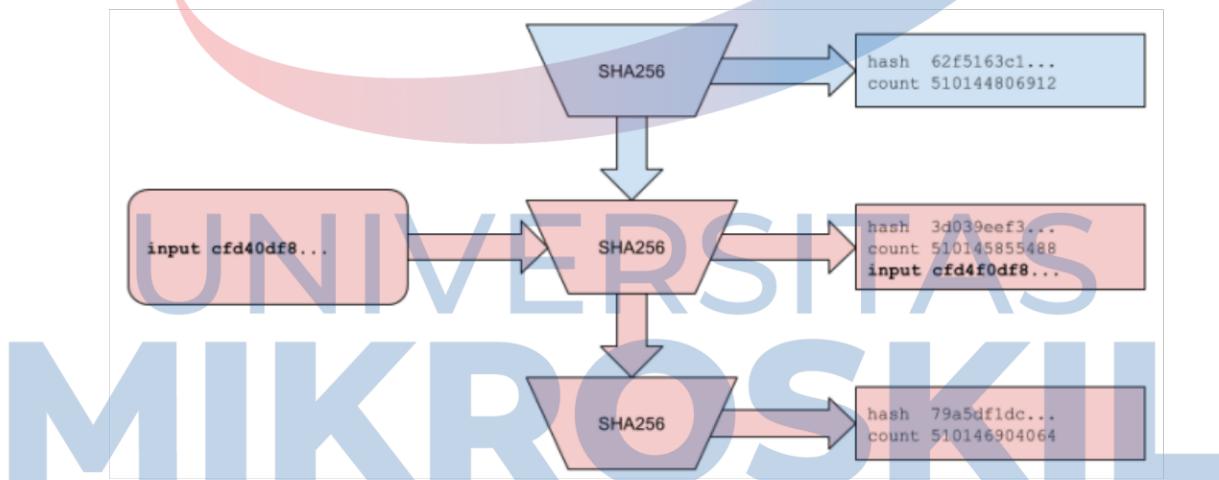
Hash juga dapat digunakan untuk menyimpan data yang dibuat sebelum nilai *hash* tertentu dihasilkan dengan menggunakan fungsi gabung (*combine*) untuk menggabungkan data tersebut dengan *hash* pada *index* saat ini. Contoh penambahan data pada PoH dapat dilihat pada Tabel 2.6.

Tabel 2.6 PoH Sequence Penambahan Data [42]

Index	Operation	Output Hash
1	sha256("any random starting value")	hash1
200	sha256(hash199)	hash200
300	sha256(hash299)	hash300
336	sha256(append(hash335, photograph_sha256))	hash336

Nilai *hash336* merupakan *output* dari komputasi dari *hash335* dengan data *photograph*.

Pada contoh berikut, data *cfdf40df8...* ditambahkan ke PoH ketika perhitungan ke-510145855488 dengan kondisi nilai hashnya *3d039eef3*. Proses ini mempengaruhi nilai *hash* yang akan dihasilkan kedepannya, adapun perubahan yang dimaksud ditandai dengan perbedaan warna pada Gambar 2.10.



Gambar 2.10 PoH Sequence Penambahan Data [42]

Proses verifikasi PoH dapat dilakukan pada komputer *multicore* agar waktu verifikasi lebih singkat dibandingkan waktu yang dibutuhkan untuk menghasilkan *hash*. Berikut contoh verifikasi secara *multicore* yang dapat dilihat pada Tabel 2.7 dan Tabel 2.8.

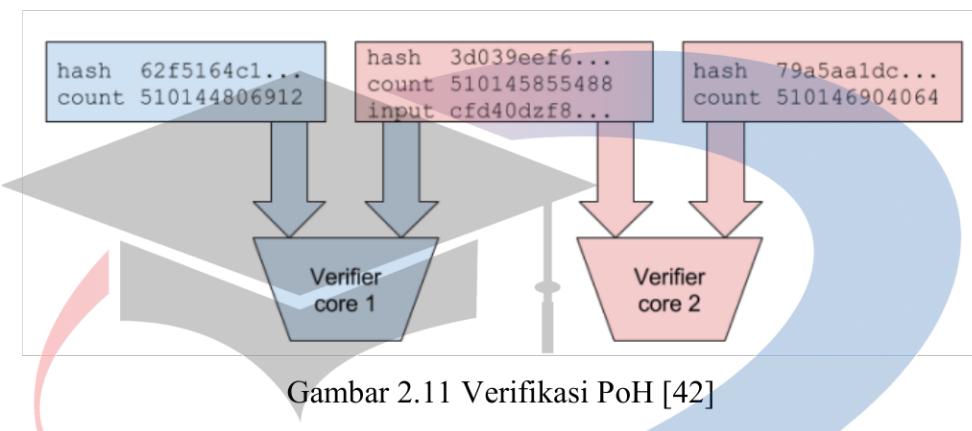
Tabel 2.7 Verifikasi PoH Core 1 [42]

Index	Data	Output Hash
200	sha256(hash199)	hash200
300	sha256(hash299)	hash300

Tabel 2.8 Verifikasi PoH Core 2 [42]

Index	Data	Output Hash
300	sha256(hash299)	hash300
400	sha256(hash399)	hash400

Verifier dapat memisahkan kumpulan *hash* menjadi beberapa bagian sesuai *core* yang ada dan dijalankan secara paralel untuk memastikan setiap nilainya adalah benar seperti pada Gambar 2.11.



Gambar 2.11 Verifikasi PoH [42]

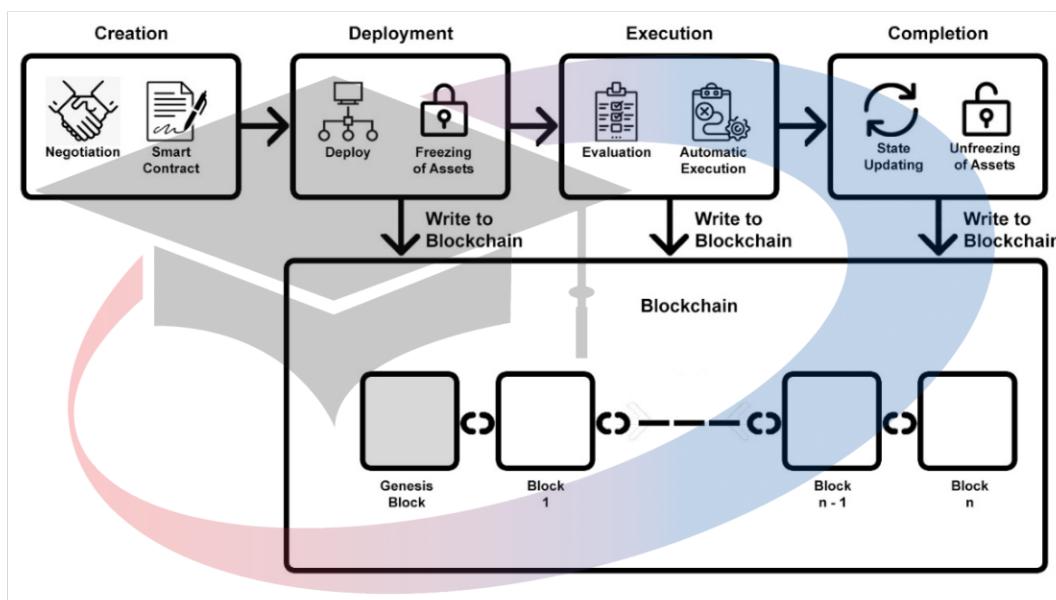
2.3 Smart Contract

Smart contract adalah istilah yang diperkenalkan oleh ilmuwan bernama Nick Szabo pada tahun 1994. Szabo mendefinisikan kata “smart” dalam kontrak ketika membandingkannya dengan kontrak tertulis tradisional (*paper-based contracts*) karena kemampuannya untuk dapat dieksekusi secara otomatis. Dalam *blockchain*, *smart contract* merupakan sebuah program yang dapat dikonfigurasi dan dieksekusi secara otomatis tanpa pihak ketiga [43]. Sebuah *smart contract* berisi kumpulan tahapan yang dapat dieksekusi, nilai (*value*), *address* dan status (*state*). Karakteristik dari *smart contract* sebagai berikut [44]:

1. *Smart contract* merupakan kode yang dapat dibaca oleh mesin dan dijalankan pada *blockchain platform*.
2. *Smart contract* merupakan bagian dari program aplikasi.
3. *Smart contract* merupakan program yang dijalankan berdasarkan peristiwa (*event driven program*).
4. *Smart contract* bersifat autonom yang berarti tidak perlu dimonitor setelah dibuat.
5. *Smart contract* bersifat terdistribusi.

Smart contract dapat dilacak, tidak dapat diubah dan hanya akan dieksekusi jika ada penambahan *block* dalam *blockchain* karena adanya transaksi baru. Istilah untuk tarif yang

perlu dibayarkan sebelum sebuah *smart contract* dieksekusi adalah *transaction fee* atau *gas fee*. *Fee* ini akan ditambahkan sebagai tahapan yang perlu dijalankan oleh *smart contract* selain dari tahapan yang sudah diprogramkan sebelumnya. Semakin rumit *smart contract* yang dirancang, maka semakin besar *fee* yang perlu dibayarkan untuk setiap pengeksekusianya. Ketika dieksekusi, *smart contract* akan menjalankan semua tahapan yang sudah diprogramkan dan disimpan dalam *blockchain platform* sehingga memiliki keunggulan dari aspek keamanan, permanen dan kekal yang disediakan *blockchain* [43]. Gambaran proses kerja dari *smart contract* dapat dilihat pada Gambar 2.12.



Gambar 2.12 Proses Kerja *Smart Contract*

2.4 Cryptocurrency

Cryptocurrency adalah mata uang digital yang memanfaatkan teknologi *blockchain* dalam pengembangannya. Memiliki fungsi yang sama layaknya uang pada umumnya membuat *cryptocurrency* menjadi mata uang yang digunakan pada transaksi *blockchain*.

Konsep *cryptocurrency* terdiri dari [45]:

1. Digital

Cryptocurrency tidak memiliki bentuk fisik, hanya berada di dalam komputer dalam bentuk digital atau virtual.

2. Decentralized

Umumnya *cryptocurrency* tidak memanfaatkan hanya satu server sentral dalam penyalurannya, namun ratusan hingga ribuan server atau komputer.

3. Peer Peer

Interaksi dari transaksi *cryptocurrency* hanya melibatkan individu yang berkepentingan dalam transaksi itu, tidak ada pihak ketiga sebagai perantara.

4. Username

Siapapun dapat menggunakan *cryptocurrency* tanpa perlu melampirkan data pribadinya sebagai syarat penggunaan.

5. Without Trust

User *cryptocurrency* ingin memiliki kebebasan penuh atas kontrol data pribadinya termasuk uang yang dimiliki sehingga tidak ada pihak ketiga yang terlibat dalam transaksi. Kedua user yang terlibat dalam transaksi hanya berkomitmen melalui *blockchain* tanpa perlu saling mengenal ataupun saling percaya satu sama lain.

6. Encrypted

User akan diberikan kode spesial layaknya kunci untuk melindungi data pribadi mereka agar sulit diretas oleh user lain karena memanfaatkan kriptografi. Hal ini berarti seluruh data pribadi user akan dienkripsi.

7. Global

Cryptocurrency dapat digunakan secara global karena tidak ada batasan pengiriman antar negara seperti mata uang negara yang dipakai atau yang disebut dengan istilah *fiat* dalam *blockchain*.

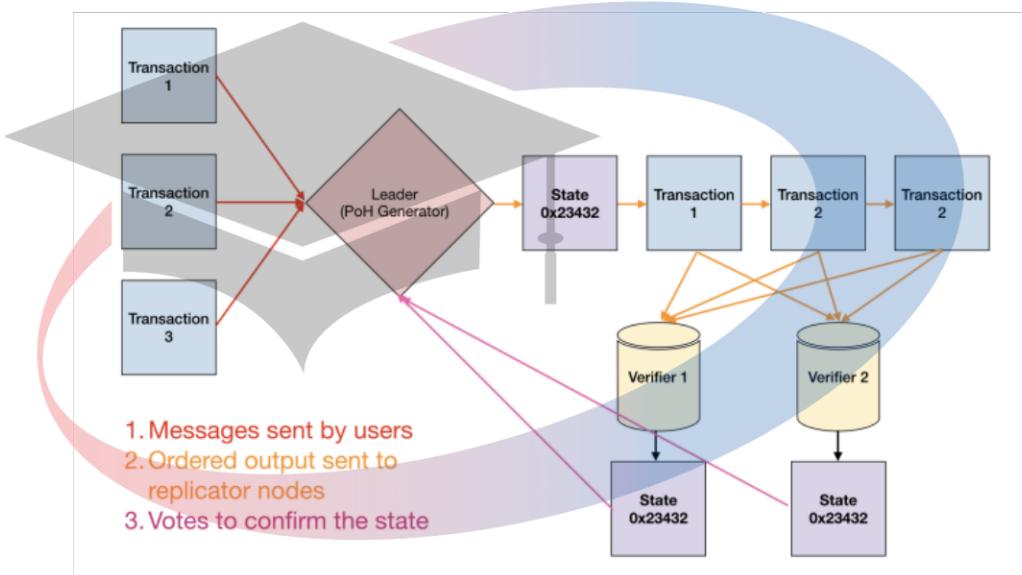
Pada dasarnya, yang disimpan dari transaksi *cryptocurrency* adalah riwayat dari apa yang terjadi ketika transaksi berjalan di jaringan, termasuk nominal uang yang sedang dikirimkan. Setelah transaksi berhasil, data transaksi akan disimpan secara permanen dalam *blocks* sehingga data tidak dapat dimanipulasi, dimodifikasi ataupun diretas. Kumpulan *blocks* dari transaksi yang berhasil akan membentuk suatu rantai panjang yang saling berhubungan [45].

2.5 Solana

Solana adalah *blockchain platform* bertipe publik dengan menggunakan algoritma konsensus *Proof of History* (PoH) yang diperkenalkan pada April 2018 oleh Anatoly Yakovenko. SOL yang merupakan singkatan dari nama Solana adalah *cryptocurrency token* yang digunakan untuk seluruh keperluan operasional dalam Solana [46]. *Source code* dari Solana terbuka untuk publik (*open source*) sehingga *developers* dimungkinkan untuk mengembangkan aplikasinya sendiri [47]. *Smart Contract* dalam Solana adalah sebuah

program otomatis yang digunakan untuk menyimpan syarat kontrak dan dapat dikonfigurasi dengan berbagai bahasa pemrograman seperti C, C++, dan Rust [46].

Transaksi dalam Solana melibatkan beberapa komponen yaitu *Leader*, *State* dan *Verifier*. *Leader* akan menerima pesan dari *user* dalam bentuk transaksi untuk diproses dalam *Random Access Memory* (RAM) komputernya agar dapat memaksimalkan kinerja *node* lain dalam sistem. Hasil pemrosesan ini berupa *signature* dari *state* yang dikirimkan ke *verifiers*. *Verifiers* akan mengeksekusi transaksi yang sama dan menghasilkan *signature* dari *state* sebagai konfirmasi selesainya pengeksekusian transaksi [42]. Proses transaksi pada Solana dapat dilihat pada Gambar 2.13.



Gambar 2.13 Aliran Transaksi Pada Solana [42]

Pada subbab ini, akan dijelaskan apa saja komponen dan arsitektur dari Solana.

2.5.1 Komponen Solana

Berikut merupakan penjelasan dari komponen Solana yang terdiri dari [43,48]:

1. Leader

Leader adalah *Proof of History* (PoH) *generator* yang dipilih *validator* menggunakan pemilihan berbasis *Proof of Stake* (PoS) untuk memproses transaksi pengguna agar unik dan terstruktur di dalam sistem. Setiap sejumlah transaksi selesai diproses, *signature* dari *state* yang merupakan hasil dari pemrosesan transaksi akan dihasilkan oleh *leader* dan ditandatangi dengan identitas *leader*-nya sendiri.

2. State

Tabel *hash* yang diindeks dengan *address* pengguna, setiap kolomnya berisi *address* lengkap dari *user* dan memori yang dibutuhkan untuk komputasi yang akan dilakukan.

3. *Verifier*

Verifier merupakan *node* dimana *leader* mengirimkan hasil pemrosesan transaksinya. Untuk menjamin kebenaran dari hasil kerja *leader*, *verifier* melakukan pengecekan secara *asynchronous*. Jika mayoritas *verifier* menemukan kesalahan pada hasil yang disampaikan *leader*, maka ia akan kehilangan perannya.

4. *Validator*

Validator adalah partisipan dalam bentuk *virtual node* yang bertugas untuk membuat *blocks* baru dan memvalidasi transaksi yang dimasukkan ke sistem.

2.5.2 Arsitektur Solana

Berikut merupakan penjelasan dari arsitektur Solana yang terdiri dari [42]:

1. *Network*

Leader diharapkan dapat menerima *user packets*, mengurutkannya dan memprosesnya menjadi transaksi dalam bentuk PoH yang kemudian dikirimkan ke *verifiers*. Dalam koneksi jaringan 1GB per detik, maksimal transaksi yang dapat dilakukan adalah 710.000 transaksi per detik.

2. *Computational*

Setiap transaksi akan melewati fase verifikasi. Proses tersebut tidak akan menggunakan memori diluar dari yang telah digunakan dalam transaksi itu meski dapat dijalankan secara *asynchronous*. Oleh karena itu, ada batasan kemampuan berdasarkan ketersediaan *cores* yang dimiliki oleh sistem, semakin sedikit *cores*, semakin sedikit transaksi yang dapat diproses dan begitu juga sebaliknya.

3. *Memory*

Secara teoritis, jika setiap akun mengimplementasi *hashtable* dengan 32 *byte entries*, maka penyimpanan dengan ukuran 640GB dapat memuat 10 miliar akun. Dan jika terdapat 2 pembacaan dan 2 penulisan per transaksi, maka memori dapat menangani 2,75 juta transaksi per detiknya. Pengukuran tersebut dilakukan dengan Amazon Web Service.

2.5.3 Keamanan Solana

Solana memiliki tujuan untuk menyediakan *blockchain* yang dapat mendukung mata uangnya (*token*) dan dirancang untuk dapat digunakan dan diakses secara publik, juga sepenuhnya terdesentralisasi. Untuk memperoleh kepercayaan, ada beberapa faktor

keamanan yang harus diperhatikan. Berikut hasil temuan dari perusahaan audit keamanan Kudelski Security [49]:

1. *Theft*

Tidak ditemukannya indikasi pencurian aset, *token*, atau mata uang lainnya dapat terjadi melalui implementasi teknologi *blockchain* yang sudah mereka terapkan menggunakan inovasinya sendiri yakni konsensus *Proof of History*. Rancangannya sudah kokoh dan dapat dipercaya dari kedua perspektif, baik dari keamanan aset dan juga keamanan buku besar akuntansi.

2. *Asset Vulnerability*

Tidak ditemukannya indikasi aset dalam bentuk *token*, insentif, mata uang, *smart contracts*, atau data didalam *block* yang rentan terhadap inkonsistensi rancangan. Demikian juga, tidak ditemukannya perilaku fungsional yang dapat membahayakan aspek integritas pada aset.

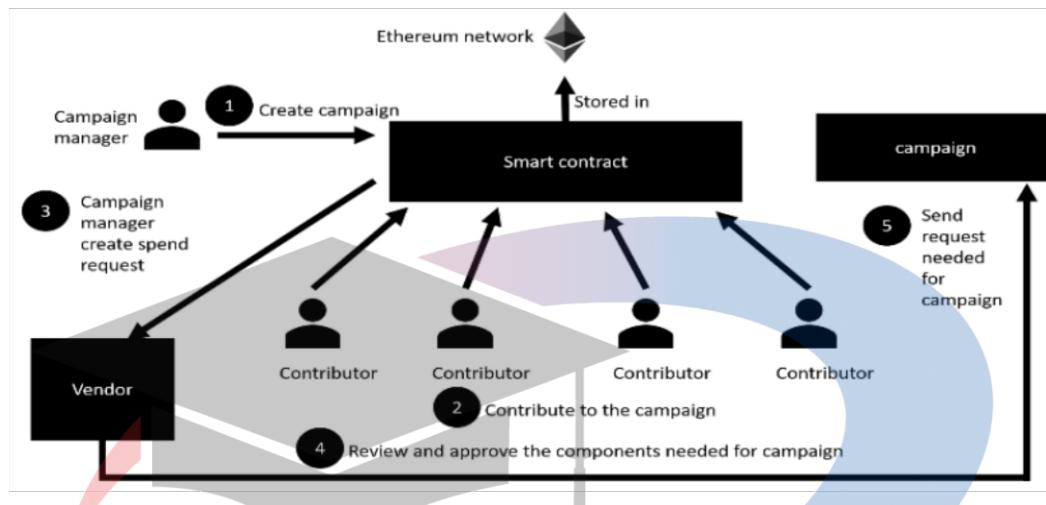
Ada hal penting yang harus dicatat bahwa struktur transaksi dalam Solana menspesifikasikan daftar *public keys* dan juga *signatures* untuk *key* tersebut, sehingga ada daftar berurut dari instruksi yang akan dioperasikan untuk setiap tahap yang berhubungan dengan *account keys*. Ini meningkatkan kecepatan *output*, tetapi karena hal ini juga pengguna dengan niat yang buruk dapat memasukkan *account* sesuka hatinya. Disinilah tugas program atau *smart contract* untuk melindungi informasi dan data dari *account* yang berbahaya. Oleh sebab itu, *smart contract* yang dibuat harus dapat memvalidasi dan melindungi *account* yang terlibat dalam transaksi tersebut [50].

2.6 Blockchain-Based Crowdfunding Platform

Blockchain-based crowdfunding platform adalah *crowdfunding platform* yang memanfaatkan *blockchain* dalam kebutuhan operasionalnya untuk menunjang keamanan, efisiensi dan kemudahan dalam bertransaksi. *Smart contract* milik *blockchain* dapat melindungi *fundraiser* dan *funder* dengan menciptakan *platform* yang sehat berdasarkan transparansi data. Ketika *fundraiser* dan *funder* menandatangi kontrak, mereka telah sepakat terhadap ketentuan dan syarat dari kontrak yang tidak dapat diubah oleh siapapun. Oleh karena itu, pendekatan ini menjadi solusi alternatif untuk menangani masalah kepercayaan *funder* terhadap *fundraiser* ataupun *platform* karena *smart contract* menjamin tidak akan ada kecurangan atau kesalahan yang terjadi pada *platform* yang dapat mengakibatkan kerugian pada kedua pihak. Penggunaan *blockchain* juga menciptakan sistem yang terdesentralisasi

pada layanan keuangan sehingga dapat melindungi dana dari *funder* tanpa melibatkan pihak ketiga [5].

Gambar 2.14 merupakan contoh proses kerja *blockchain-based crowdfunding platform* dari penelitian terdahulu yang menggunakan Ethereum sebagai *blockchain platform*. Ethereum merupakan salah satu *blockchain platform* bertipe publik [1].



Gambar 2.14 Proses Kerja *Blockchain-Based Crowdfunding Platform* Terdahulu [1]

1. *Fundraiser (campaign manager)* akan membuat *campaign* dengan mengisi informasi detail dari *campaign* yang akan dibuat untuk dilihat oleh *funder (contributor)*.
2. *Funder* akan melakukan kontribusi jika tertarik pada *campaign* yang ada.
3. *Fundraiser* kemudian melakukan permintaan pencairan dana yang berisi informasi rencana penggunaan dana.
4. *Funder* akan mendapatkan notifikasi bahwa ada permintaan pencairan dana yang dilakukan dan melakukan voting.
5. Jika mayoritas *funder* setuju, maka *smart contract* akan mengirimkan dana yang telah dikumpulkan ke *platform (vendor)* yang kemudian akan dikirimkan ke *fundraiser*.