Hacash: A Cryptocurrency System for Large-Scale Payments and Real-Time Settlement

An anonymous author released the paper in 2018

Abstract

This paper introduces a method for real-time offset settlement using ordered multi-signature transactions through channel chains, which can infinitely expand the transaction volume per second. This method employs punitive measures to deter dishonest parties, ensuring fund security and real-time availability in cryptocurrency issuance, payments, settlements, and an open financial system. It incorporates composite signature addresses, hierarchical equity control addresses, multi-party signed transaction structures, diverse category composite payment contracts, and asset change protocols, meeting the vast majority of modern financial, corporate, and individual payment needs. Through a combination of proof-of-work, historical witnessing, and fork voting-based preemptive bookkeeping rewards distribution, it effectively prevents double-spending, safeguards against computational attacks, and mitigates entrusted trust risks. Additionally, it implements incentive mechanisms such as new currency issuance based on market supply and demand adjustments, public account and channel fees, channel interest, Bitcoin one-way transfers, and block diamond minting. These mechanisms ensure the long-term effective operation of the entire currency system without the need to trust any institution.

The fundamental principle of the channel chain settlement network is as follows: Two accounts lock a certain amount of funds each, forming a payment channel. They can privately sign multiple payments without broadcasting confirmations to the entire network. Only the final balance distribution needs to be submitted to the main network after the last transaction to retrieve the correct funds owned by each party. This greatly increases the number of transactions per second in the system. If one party ends the channel, their funds will be locked for a period. If the other party provides evidence of updated balance distribution to the main network during this time and proves that the former is being dishonest, the disclosing party will seize all funds from the dishonest party, forcing both parties to remain honest. By connecting multiple payment channels and sequentially having all involved parties sign from the recipient's end until the payer's final signature, all parties will receive and expend funds simultaneously, ensuring complete payments, real-time availability, and fund security. The underlying technical principles are similar to addressing and transmitting data on the internet. Channels can collect minor fees to incentivize the provision of stable services.

Table of Contents

1. Preface

1.1 Crisis

We have arrived at a pivotal juncture in the transformation of the monetary system.

Throughout history, the advancement of civilization and economic progress has always been accompanied by a fundamental premise: cultural and financial freedom. Regardless of whether it's capitalism or socialism, in developed countries or developing nations, the recurring economic crises have led to an increasing demand for stricter regulation of the financial system. At the core, different modern socio-economic systems share a common characteristic: mandatory sovereign credit currency and a fractional reserve banking system. Fundamentally, this is a form of deception, the root of financial instability, and the primary cause of unjust distribution of wealth, aside from violent plunder. This inevitably gives rise to economic intervention policies that either exacerbate or overcompensate, resembling the cyclic dynastic changes that repeatedly harm the gradual growth of economies and people's living standards.

An environment of financial instability, unpredictability, human intervention, and manipulation allows large capital to effortlessly gain excessive returns without the necessity of entering the real economy. When the returns on capital, particularly colossal capital, exceed or significantly surpass the average returns of long-term overall economic development, the gap between the rich and the poor accelerates unavoidably. This continues until societal contradictions intensify, leading to economic crises and turmoil. If a generation comes of age right at the beginning of a decade-long, or even twenty-year-long, depression, their destiny is destined to be arduous and unproductive. To mitigate the severe intergenerational injustices caused by these cyclical crises (which could perhaps result in even more severe social unrest, revolutions, or even wars), resorting to economic intervention and stimulus policies becomes a painful expedient. However, facing unfavorable population and debt trends, it is utterly impossible to address long-term structural crises with short-term stimuli; it's essentially deceiving oneself.

Providing financial assistance to banks and large corporations and injecting monetary liquidity fundamentally perpetuates the act of taking from the poor to give to the rich. The costs and losses are socialized, making the entire population pay the price for the greed and recklessness of a small minority. What we require is a fundamental solution that avoids or significantly reduces financial crises and economic collapses, thereby safeguarding the interests of financially vulnerable groups from being manipulated and exploited.

During China's Northern Song Dynasty, due to the government's imposition of iron currency in the Sichuan region, an unwieldy situation arose. In response, an early form of paper currency called "jiaozhi" emerged among the populace. This seemed to be a step forward in currency and finance. The ruling authority at that time realized it was a perfect tool for taxation and promptly monopolized the right to issue paper currency. This eliminated the need for extensive resources to measure land, count households, collect money, and maintain records. All that was required was to activate the printing press gently, silently extracting wealth from the populace. However, the subsequent cycles of hyperinflation and economic collapse shattered the dynasty's grand plans. Eventually, during the early years of the Ming Dynasty, people completely lost faith in any form of paper currency, returning to the era of precious metal currency or even resorting to direct trade using divided and weighed silver. This seemingly regressive state of currency affairs, however, granted substantial financial and economic freedom to the populace, leading to an uncommon period of financial stability in Chinese history. This persisted until the later stages of the Qing Dynasty.

Currency emerges from the market and ultimately serves the market. Its form itself doesn't possess hierarchy, and its essence lacks an absolute good or bad; it's a matter of whether it is more adaptable and efficient. We require a currency market characterized by free competition and survival of the fittest, along with an open, transparent monetary and financial environment that avoids fraud, preventing institutionalized exploitation and redistributive wealth plundering.

1.2 The Future of Currency

Gold is inherently suited to serve as a form of currency on Earth, yet it is not being extensively used for large-scale transactions today. The primary reasons for this are as follows: 1. Its bulkiness makes it inconvenient to carry; 2. It cannot be infinitely divided for micro transactions; 3. Prolonged circulation results in wear and tear; 4. The growing sophistication of counterfeiting techniques. Throughout European history, gold was utilized as a reserve-backed currency for circulation. This method seemed to combine the advantages of gold while sidestepping its flaws as a means of daily payment. However, no individual or organization can resist the temptation of creating money out of thin air. Despite resolute commitments, paper currency eventually experiences excessive issuance. Similar to historical instances where other metals were mixed intentionally with gold, silver, or copper coins to cause devaluation, paper currency faces the same fate. In extreme cases, the drastic reduction in issuance costs can lead to severe hyperinflation and economic collapse. Such scenarios have recurred repeatedly.

Two controversial issues that hinder gold from serving as a universal means of payment are as follows: 1. Uneven distribution of reserves; countries with gold mines essentially levy a global coinage tax; 2. It is unable to rapidly adjust supply during fluctuations in economic growth rate and scale, potentially causing contraction and monetary scarcity. Debates, particularly regarding the second point, are abundant in the works of various economic schools such as Austrian economics, modern monetary theory, and Keynesian economics. Some argue that credit expansion is the root cause of economic crises and a flood-like danger, while others firmly believe that a currency devoid of "elasticity" will stifle the economic development as a whole.

In history, currency has undergone the following stages (without strict categorization, order of appearance, or inherent superiority):

(1) Universal value objects (grains, cloth, cattle, cigarettes, etc.)

(2) Rare stable-state objects (gold, silver, copper, gemstones, shells, etc.)

(3) Trust-based accounting certificates (deposit certificates, debt certificates, anonymous checks, vouchers, paper gold, etc.)

(4) Sovereign credit symbols (fiat currency)

Among these stages, the first and second have gradually fallen out of use due to their inability to meet the requirements of modern commercial transactions. The third stage seems to lead to credit inflation and the economic fragility and collapse caused by banker fraud. Consequently, people began looking to the establishment of central bank systems by governments, which led to the fourth stage with results as evident as we see them.

We believe that the future of currency, the fifth stage, will be characterized by an open electronic network system primarily based on "recognized rules" as the core and "individual credit" as branches and supplements. This system will continue to improve and reveal the optimal system through the process of free competition. These "recognized rules" are not subject to anyone's will, nor are they controlled by dominant interest groups, similar to the impossibility of creating gold out of thin air. Fair rules enable people and markets worldwide to participate effectively, resulting in a substantial increase in market size and efficiency. Currency goes beyond being a mere unit of equivalence or accounting; it is a signal and an information system for the functioning of the economy. While we can tolerate and rectify partial credit failures, we cannot afford the consequences of a complete credit collapse.

In 2008, the emergence of Bitcoin and the underlying blockchain technology, along with a decade of effective operation, provided us with a direction.

1.3 What Do We Need

The core value of blockchain does not lie in decentralization, permanent data storage, or evading regulation. Instead, it rests in the concept of trustlessness, which means mitigating structural trust risks within a system.

The emergence of Bitcoin wasn't intended to enhance local, temporary payment efficiency or reduce some momentary transaction costs; it was meant to avoid or mitigate the consequences of misconduct by enterprises and institutions we trust. Its vision encompasses complete anonymity and decentralized peer-to-peer trust. The future of blockchain, however, is about transparency and distributed self-audit trust, achieved by leveraging inexpensive computational resources to lower overall transaction costs and enhance negotiation efficiency within society. Currency serves as the cornerstone of the entire economic world; even slight improvements in this critical area can lead to orders of magnitude improvements in economic scale and human well-being.

Just as the internet fulfilled humanity's fundamental need for information communication, blockchain aims to address a problem that has plagued humans for hundreds of thousands of years: fraud. Historically, we've paid a substantial cost, consumed countless resources, and impeded numerous transactions in response to this issue. Therefore, we cannot introduce any possibility of fraud into the future monetary system, even with divine assurance.

We require a form of currency and an accompanying financial system characterized by "hard constraints" and minimized trust—a currency system that no one can easily debase and that doesn't demand excessive trust costs. Such a system could better uphold the stable development of the economic system, fully satisfying the needs of modern commercial transactions, corporate accounting, and financial clearing.

2. Basic Principles

2.1 Technical Theories

Currency needs relative scarcity to fulfill its function as a value signal. Nobody would use water, soil, or leaves as a medium of exchange. However, in the world of electronic information, almost everything can be replicated at almost no cost. Therefore, a currency based on binary data has a fatal inherent flaw: unlimited quantity and indistinguishability between genuine and counterfeit units. If we still rely on an online issuing authority or an electronic mint to ensure the authenticity and quantity limit of currency units, historical experience shows that nobody can resist the temptation of creating money out of thin air. Eventually, what everyone holds in their hands will become worthless.

One straightforward solution is to assign a unique integer number to each electronic currency unit and announce an upper limit for these numbers (or have this limit automatically increase by a fixed amount each year). Then, use electronic signatures to indicate the owners of each currency unit (ignoring the issue of who initially owns these units). During a payment, the current owner signs the recipient's public key with their private key, indicating that the currency unit has been transferred and preserving all historical payment records. This resolves the issues of counterfeiting and unlimited quantity of electronic currency.

However, the proposed solution above has several obvious shortcomings:

(1) It cannot be subdivided for small payments or change.

(2) The same currency unit can be paid to different people at the same time (double spending).

Issue 1 can be temporarily alleviated by using sufficiently small denominations (with a sufficiently large upper limit). However, the double-spending problem is more complicated. One approach is to have a central public database to save and prove every payment record. Nevertheless, this still relies on the honesty and integrity of institutional authorities, which opens the possibility of deception. Another approach is broadcasting, where everyone has their own copy of an account book that lists the payment history of everyone. When receiving currency, individuals check the ownership of the currency unit in their own records to avoid double spending.

While the solution of everyone having their own account book seems to solve the double-spending issue, it still has significant flaws:

(1) Maintaining an account book to continuously record all payments incurs substantial

costs. Individuals who do not receive currency regularly would find it uneconomical to maintain a full account book. Ultimately, people would still rely on larger institutions similar to banks to provide account books.

(2) When some payments do not reach all record-keepers due to technical issues with broadcasting (such as underwater cable disconnections), discrepancies will arise in individuals' account records. Over time, numerous conflicting versions will emerge, causing a breakdown in the payment system.

(3) If each person expends substantial effort to maintain an account book solely for checking the validity when receiving money, the currency system becomes inefficient. Additionally, the issue of free-riding could result in no one maintaining the account books.

Certainly, there is another critical question: Who should initially own these electronic currency?

2.2 Principle of Self-Interest

In the long run, any cooperation system that relies solely on the expectation of the other party's honesty, integrity, and adherence to commitments, without other incentives and checks, will become unsustainable.

Societal cooperation systems that consistently operate effectively are all based on correctly addressing human nature's greed and self-interest, and harnessing these qualities to maintain the system's self-sustainability. Bitcoin is on the right path and has grown stronger over its ten-year development. By creatively combining ledger recording and currency issuance, it provides ample motivation for everyone to maintain the ledger. It uses a proof-of-work competition for recording rights to ensure the uniformity of everyone's ledgers, cleverly addressing the deficiencies mentioned earlier. Of course, Bitcoin does not use a system of encoding each coin, but employs the Unspent Transaction Output (UTXO) method to support microtransactions and change. For more detailed theories about Bitcoin, please refer to Satoshi Nakamoto's paper.

2.3 Shortcomings

There's no such thing as a free lunch. A currency system based on open competition for accounting still suffers from two major criticisms:

(1) The method of competing through hash algorithm calculations for a certain period to gain the right to record account entries and receive newly minted currency rewards consumes a substantial amount of hardware and energy.

(2) The need to broadcast every transaction to all ledger recorders and await confirmation results in an overall low transaction efficiency (approximately 7 transactions per second), and leads to high transaction fees, making it fundamentally inadequate for modern commerce.

Strictly speaking, we do not consider the first point to be a genuine drawback. Advocates of paper currency won't tell you this: the "circulation cost" of currency should be as low as possible, but the "production cost" does not need to be zero to be optimal. Aside from short-term speculation and irrational prosperity, in a non-compulsory field of free competition, a balance between cost and benefit will always be found. The extensive consumption of energy and resources is not because we are crazy, but because doing so is overall profitable, and the costs expended will ultimately be compensated in other ways. Just as we do not extract all the oil and gold from the Earth, as doing so would be uneconomical.

The idea that a currency system should consume as little energy as possible is quite naive and primitive. Ancient people also had a similar question and bias: merchants simply transport goods from one place to another without creating anything, so why do they earn so much money?

Energy consumption returns currency to its commodity attributes, thereby introducing a "hard constraint" based on market supply and demand. This avoids the moral risks of entrusted trust in central institutions. The trade-off between consuming energy (or any form of resources) and avoiding victimization is as follows:

(1) Enjoy free and convenient benefits without having to pay any costs during normal times, but the central institutions providing the service may be breached or manipulated, or they may harm the interests of everyone in secret.

(2) Incur certain costs (fees or effort) during normal times to avoid the risk of someone or an organization deceiving you.

If someone claims to achieve both of the above (completely free and absolutely secure), they either fail to recognize the nature of the problem or have malicious intent.

One way to temporarily alleviate the second drawback is to scale up, which involves increasing the block size limit or reducing the block generation interval. However, this does not fundamentally solve the problem. The block size has a theoretical limit; if it is too large, it will extend the time for downloading, transmitting, and synchronizing. This further limits the upper limit of the block interval. This is a process of escalating demands; new transactions will always fill the expanded space, and the global transaction volume far exceeds the theoretical scaling limit constrained by hardware systems. On the other hand, when the data volume grows to a certain extent, regular personal computers will be unable to store the complete ledger, eventually eliminating the majority of participants, and data will converge towards well-funded large institutions.

Another approach that some proponents are attempting is to drastically reduce the total number of ledger recorders (e.g., 101 or 21), impose higher performance requirements on their tools, and provide them with additional currency as an incentive. This approach seems promising, but as mentioned earlier, it has inevitable drawbacks, and its essence is not significantly different from historical branch systems of minting facilities.

Some individuals also hope to achieve a significant increase in overall system throughput by tolerating temporary data inconsistencies (using a directed acyclic graph). However, this approach can only be applied in scenarios where stringent data authenticity requirements are not necessary or immediate verification is not needed. In a small-scale instant payment system, it is entirely unreliable.

Perhaps Satoshi Nakamoto did not consider this a troublesome matter, or perhaps it was deliberately overlooked due to a lack of clear thinking at the time. In any case, he did not specify a solution for accommodating massive transaction volumes in his paper.

3. Channel Chain Settlement Network

3.1 Fundamental Assumptions

Drawing from the way the world operates, a physically strong adult male could easily seize belongings from women and children on the street, but most people don't do so because they understand that the presence of law enforcement and a legal system would exact a higher cost, rendering such malicious behavior not worthwhile. This illustrates that even with the presence of risks, as long as a stringent punishment mechanism is in place, rational self-interested behavior will maintain the overall effectiveness of the system.

We propose that the public ledger should not include records of all transaction payments. Instead, it should function as a rigorously accurate, effective, and tamper-resistant arbitration system and final clearing system. This ensures that private payment activities conducted offline cannot be fraudulent. Otherwise, severe punishments are imposed. In an indefinitely repeatable game of equivalent transactions, individuals tend to uphold honesty for long-term cooperation when there is a risk of dishonesty being penalized. This system possesses tremendous scalability, capable of dynamically increasing transaction volume as needed and theoretically without an upper limit. Thus, it is fully equipped to meet the demands of modern business development.

3.2 Main Principles

We need to establish a system for real-time, no-loss microtransactions that can be used for purchasing in online or physical stores. If intermediaries provide financial services, everyone must collectively ensure the accuracy of transactions. The key is that the settlement of the fund chain should be real-time and synchronous, rather than asynchronous, to avoid serious centralization and fund security issues. Entrusting our cryptocurrency to any wallet, exchange, or payment institution for safekeeping is no different in essence from historical "gold certificates," which will ultimately lead us down the path of history: introducing a fractional reserve banking system into the world of cryptocurrency.

Firstly, we need to create a series of bi-directional settlement channels. These channels involve customers and financial channel service providers (referred to as nodes), as well as merchants and nodes. When necessary, customers and merchants can directly establish settlement channels. When initiating a payment, the merchant or the node serving the merchant queries the route of the fund channel chain, establishes a TCP connection across the entire chain, and retrieves information such as all channel IDs, the hash of the previous transaction, transaction sequence numbers, and balance confirmations from all nodes. Using this information, a complete transaction is constructed and sent to all parties involved. Starting from the merchant, each party signs the payment transaction and sends the signature to all other parties in sequence until all customer signatures are received. At this point, the merchant signs the transaction confirmation message, and then all TCP connections are closed sequentially, starting from the end of the chain. In the end, all relevant parties simultaneously receive (and disburse) the funds, and the payment transaction is completed. The critical aspect of this technology is that no intermediary can withhold funds, and transfers are "atomically" from one end of the channel chain to the other. The entire payment transaction is akin to a "database transaction," where either all parties succeed or all parties fail—no erroneous "intermediate state" where one side transfers funds but the other side hasn't received funds, ensuring that everyone is free from the risk of losing money. Each channel can levy a small service fee to incentivize nodes to provide stable services.

3.3 Technical Implementation

The following section describes all the technical details and the data state after each step is completed.

(1) Creation of Joint Settlement Channels

Start by creating a multi-signature transaction:

```
{
    // Open a settlement channel, with fund1 and fund2 being funding parties
    // Once the channel is open, balances of both parties are deducted until the channel is closed
    // Lockup period for one-sided channel closure (block count)
    lock: 2016, // approximately one week
    // Custom channel ID for settlement channel, randomly generated
    channelId: 232353253456,
    fund1: {
        address: '1313Rta8Ce99H7N5iKbGq7xp13BbAdQHmD', // funding address
        amount: 1234, // funding amount
    },
    fund2: {
        address: '19aqbMhiK6F2s53gNp2ghoT4EezFFPpXuM',
        amount: 1234,
    },
}
```

Both parties sign the above transaction and broadcast it to the main network for confirmation. After confirmation, unlimited and frequent bi-directional payments can be conducted off-chain without broadcasting to the main network.

(2) Off-Chain Settlement

For each payment, both parties sign the settlement information and exchange signature results. The transaction structure resembles:

```
{
    // Off-chain periodic balance allocation confirmation
    // If partial balance allocation is submitted on-chain, the address will be locked for the specified time
    channelId: 232353253456,
    prevTrsHash: Buffer.alloc(32), // Hash of the previous confirmed channel transaction (initial transaction's hash for the first time)
    autoincrement: 123135, // Auto-incremented channel transaction sequence number
    diffConfirm: {
        address: '1313Rta8Ce99H7N5iKbGq7xp13BbAdQHmD',
        amount: 1234,
    }
}
```

Signing a settlement channel for each customer and merchant would be cumbersome and lock up excessive funds. It's envisioned that specific nodes offering channel connection services can form a settlement network. Merchants and customers would only need to sign settlement channels with a few nodes, which would then allow them to easily transact with everyone else via this network, similar to how you access the Internet through a broadband provider without individually connecting to each website.

(3) Channel Chain Routing

Assuming customer A has a channel with node C, merchant D has a channel with node B, and nodes C and B have a channel together. When customer A wants to pay merchant D, the funds will flow from A to D through the channel chain wa. The route is determined through inquiries between nodes or third-party routing queries (similar to DNS servers for domain names), finding a possible shortest (or lowest fee) path and forming a bi-directional TCP connection route:

A <=> C <=> B <=> D

(4) Constructing Chained Payment Transactions

As described, customer A's payment funds flow through nodes C and B before reaching merchant D, requiring three settlement channels. Merchant D's service node B, by either querying or proactive broadcasting, makes sure all parties involved receive information about each channel's ID, transaction sequence number, fees, and balance confirmations. With this information, merchant D constructs a chained channel payment transaction, structured like:

```
{
    // Channel Transfer Transaction
    amount: 1234, // Payment amount
    // Pathway channels
    channels: [
        {
            /**** Channel 1: (A => Node C) ****/
            // Custom ID of the settlement channel
            channelId: 232353253456,
            // Previous transaction hash confirmed by both parties
            prevTransactionHash: Buffer.alloc(32),
            // Channel transaction number, auto-increment
            autoincrement: 123135,
            // Channel fee, can be zero or negative
            fee: 12,
            // Channel balance confirmation at the completion of this
transaction
            diffConfirm: {
                address: '19aqbMhiK6F2s53gNp2ghoT4EezFFPpXuM',
                amount: 1234, // Amount
            },
        },
        {
            /**** Channel 2: (Node C => Node B) ****/
            /* ...omitted... */
        },
        {
            /**** Channel 3: (Node B => Merchant D) ****/
            /* ...omitted... */
        }
    ],
}

// The above data is for illustration purposes only; the number of ele
ments in the `channels` field will be three, with the formats of the n
ext two channels being the same and are omitted for brevity.
```

The above transaction data will be broadcasted to all participating parties. Each channel will only take a single element from the channels array as a settlement voucher. Nodes C and B, because they each have two channels, can simultaneously receive and expend funds, maintaining a balanced state of payment and receipt. Moreover, no one will suffer losses due to the failure or disconnection of other nodes.

(5) Sequential Signing

All four participating parties have received and confirmed this transaction. In case of any information, fund, or technical errors, any party can disconnect the TCP connection, thereby closing the entire channel chain and terminating the payment process. At this point, the signing phase begins:

1. Merchant D signs the transaction using their private key and sends the signature to Node B.
2. Node B receives D's signature, validates it, then forwards it to Node C. Node B also signs the transaction and sends the result to both Node A and Merchant D.
3. Next is Node A, who follows the same process as Node B, forwarding D and B's signatures. Node A signs the transaction and sends it.
4. At this stage, all parties, including Customer A, have received signatures from D, B, and A. The entire channel is in a state of waiting for Customer A's signature.

The reason for beginning the signing process with Merchant D and proceeding in reverse order according to the direction of fund flow is that the receipt signature depends on the payment signature to take effect. Nodes must first confirm that the other party has signed the acknowledgment of receiving funds before they can sign the payment. This process is a chain reaction. Both the customer and the merchant need to ensure that all nodes in the channel chain have signed the transaction. Only then can the funds be credited in real-time after the payment. Otherwise, there is a risk that the customer has signed the payment but the merchant hasn't received the funds promptly.

(6) Receipt of Payment

At this point, the entire transaction and channel status depend on the signature of Customer A. Once A signs, everything takes effect simultaneously:

1. Customer A receives the transaction data and all signatures from Merchant D, Node B, and Node C. A verifies and confirms their accuracy.

2. A signs the transaction and sends the signature result to Node C.
3. Node C receives the signature, forwards it to Node B, and Node B forwards it to Merchant D. This completes the settlement channel.
4. Merchant D signs a message using their private key to confirm that they have received the payment and that it was successful. Merchant D sends this confirmation to Node B and then disconnects the TCP connection.
5. Node B forwards the payment success message to Node C, and then disconnects the connection. Node C, in turn, forwards it to Customer A and disconnects the connection.
6. Customer A receives the receipt from Merchant D, and all connections are disconnected. The entire channel chain settlement is completed, and the payment is successful.

(7) Settlement of Each Channel and Transaction Fees

In each individual channel within the channel chain, both parties have received a complete transaction, including records of how the funds moved throughout the entire chain. Each party only needs to settle within the channels array matching their own. The settlement amount is calculated as Customer A's payment amount minus the accumulated transaction fees from previous channels.

Each channel can charge a small transaction fee to cover the cost of providing the fund transfer service, similar to earning interest on loans. The transaction fee typically depends on two factors: 1. The size of the funds being transferred. 2. The hardware and network service costs. Transaction fees can be set to zero or even negative. Channels offering negative transaction fees would receive less money than they send out after signing a transaction, essentially functioning as a subsidy to attract customers and expand their market.

(8) Error Handling

We must ensure that funds arrive synchronously in real-time, and no one suffers losses due to possible errors. If any errors or issues occur during the payment process, the entire channel chain payment will be terminated, and all connections will be severed:

1. Technical failures that cause any party to disconnect before receiving the final payment receipt will terminate the channel chain.
2. Signature verification failure will terminate the process.
3. Incorrect payment amounts or fees will also lead to termination.
4. Errors in transaction sequence numbers, previous settlement hashes, or balance confirmations will result in termination.
5. Exceeding the timeout for signing will also trigger termination.

Nodes or merchants can customize a timeout period, such as 3 seconds. If they do not receive subsequent signatures or receipts within this time frame, they will disconnect the TCP connection, terminating the entire channel chain. This is done to avoid incomplete transactions that could result in one party losing funds.

Through the data exchange process described above, we complete the entire payment process, ensuring that all parties receive their funds in real-time and securely. The small transaction fees collected by intermediary nodes serve as an incentive for them to provide stable services.

3.4 Channel Closure

A channel will facilitate numerous payments over a period of time, with multiple bidirectional transactions taking place. If both parties agree on the final balance allocation, they can sign a transaction to close the channel and retrieve their respective balances. This transaction is broadcast to the main network for confirmation:

```
{
    // Mutual confirmation of balances, closing the settlement channel
    // Balance allocation takes effect immediately, with no lock-up pe
riod
    // Channel ID
    channelId: 232353253456,
    // Confirmation of balance allocation; diffConfirm represents the
difference in balances
    diffConfirm: {
        address: '19aqbMhiK6F2s53gNp2ghoT4EezFFPpXuM',
        amount: 1234, // Amount
    },
}
```

Once this transaction is confirmed on the main network, the funds locked in the settlement channel are immediately returned to both parties.

3.5 Arbitration Protection

Since settlement channels lock funds in a mutual agreement, if one party loses their private key, it will prevent the other party from unlocking and withdrawing the funds within the channel. Additionally, situations might arise where one party maliciously refuses to sign off on closing the channel or other issues that temporarily hinder cooperation. To address these concerns, the ability to unilaterally terminate a channel is necessary.

The method involves broadcasting the most recent channel chain payment transaction (or reconciliation transaction, which includes multisignature elements) to the main network for confirmation. Through this transaction, which explicitly states the balance, the passive terminating party will immediately receive their funds. Simultaneously, the initiating party (the active terminator) will also unlock their funds, but at the cost of the account being locked for a predefined period (specified by the 'lock' field, for example, a week). During this lock period, the balance cannot be withdrawn. This serves as a punishment mechanism to deter unilateral and arbitrary channel terminations.

If one party submits a transaction to the main network that is not the most recent balance allocation but is advantageous for themselves and attempts to unilaterally terminate the channel to seize the other party's funds, as described earlier, the submitting party's account will be locked, preventing them from withdrawing the balance. In this case, the other party can submit the most recent balance confirmation to the main network (determined by the transaction's auto-incremented sequence number), and once confirmed, they will immediately take control of all the funds belonging to the submitter, including the entire channel's balance and the locked balance. By enforcing severe penalties for dishonest behavior and providing incentives for proof (a game of equal importance for latecomers), both parties are encouraged to maintain honesty.

Considering that one party's contribution within a channel used for expenses or wage settlements could be zero or that the final party has paid out all the funds, the cost of malicious actions is the fees required to submit channel transactions to the main network. However, once successfully seized, there will be significant gains. Some accounts may choose to act maliciously under such circumstances. To mitigate this, a mechanism can be implemented to lock a certain amount of funds as compensation for malicious behavior across multiple settlement channels.

When one honest party unilaterally terminates the channel, the cost is having their account locked for the agreed-upon time. Since the other party cannot provide (does not exist) updated balance allocations, neither party loses funds. Through an infinite number of repeated symmetric games, participants tend to choose honesty and cooperation.

3. 6 Balancing Payments

Imagine the following scenario: a merchant initially signs a receipt channel with a service node to accept payments. At this point, the merchant's contribution to this channel should be zero, as customers make payments through the settlement network. After some time, the merchant signs a payment channel with the node, used for purchases or wage payments, where the node's channel contribution should be zero.

To facilitate accounting reports, both of these channels are set up as one-way payment channels, one solely for receiving payments, and the other solely for making payments. However, these channels have a maximum limit on locked funds, and to improve fund utilization, the amount of funds within them shouldn't be too large. After some time, all the funds within the channel may be transferred to the other party's account, making further transactions impossible. In this case, the other party must continuously add funds to the channel or close it and reopen a settlement channel with a larger limit. This process can be very inefficient and lock up an increasing amount of funds, making the entire system unsustainable in the long run.

This problem can also occur when personal spending channels are separated from wage payment channels.

To address this issue, a channel offset settlement method is used, employing the same underlying technical principles as channel chain payment transactions. In this offset transaction, there are only two channels involved, one for receiving and one for paying. Both parties, in this case, the merchant and the node (or an individual and a node, or even node-to-node), serve as both payers and payees. The structure of the transaction is similar to the following:

```
{
    amount: 1234, // Payment amount
    // Channels involved
    channels: [
        {
            // Custom ID of the settlement channel
            channelId: 1111,
            /**
             * Merchant's receiving channel, transferring funds to the
node
             */
        },
        {
            // Custom ID of the settlement channel
            channelId: 2222,
            /**
             * Merchant's payment channel, transferring funds back fro
m the node received in the receiving channel
             */
        },
    ],
}
```

The result of this transaction is that the balance of the merchant's receiving channel is transferred into the payment channel. This is an atomic operation, and neither party risks losing funds.

All participants in the settlement network can regularly perform offset settlements without the need for frequent interactions with the main network or locking up excessive funds. This keeps the entire settlement network in a state of high utilization, requiring only a small amount of funds to support a large volume of payment transactions.

3.7 Decentralization

Economies of scale and information opacity make it impossible to completely eliminate the existence of financial and data intermediaries, but admission controls and monopolies can turn cart drivers into highwaymen. Overcentralization can lead to severe single-point failures, taxation effects, and trust crises.

In cases like Bitcoin's Lightning Network, if most transactions are concentrated in a few intermediaries for fund transfers, they become banks. If one of these nodes experiences a failure, it will instantly render many transactions impossible. The funds stuck in these channels will explosively attempt to unlock on the main network, causing significant congestion and skyrocketing fees. Some smaller channels might not even have enough funds to cover withdrawal fees.

We should strive to avoid excessive centralization, and settlement channel networks have two features to mitigate this issue:

(1) Immediate Settlement of Funds

Once a payer signs a transaction, all participants receive their funds simultaneously, ensuring they are both credited and debited in real time. This mechanism prevents any issues in the event of technical failures or other unforeseen circumstances that may cause a node to go offline, as all transactions before that point have already been confirmed.

(2) Channel Payment Locking Period

Each channel can support only one transaction at a time, and concurrent payments are not possible. This bottleneck in the transaction volume per second on individual channels guarantees fund security, simplifies reconciliation, avoids transaction congestion, and deters central hub nodes.

The principle behind the fourth point is that, from the moment a TCP connection is established until the payer finally signs the transaction, the relevant channel remains locked (no other transactions can be processed during this time, and the lock time may be just a few tens of milliseconds). This makes it unprofitable to lock large amounts of funds to meet most payment demands in a hub channel, ultimately resulting in a situation where numerous small channels offer fully decentralized services. This, in turn, prevents single-point failures and centralization crises.

The decentralization of channels and the reduction in the amount of each individual channel also have another advantage: making malicious actions of fund appropriation less attractive due to the small potential gains.

3.8 Fast Channels

While channel locking provides strict security, it comes at the cost of reduced transaction throughput. Considering that a single node may have different business branches and different nodes may establish long-term trustworthy relationships, for certain microtransactions (e.g., buying a cup of coffee), there's no need for real-time reconciliation between nodes.

We can adopt a delayed reconciliation approach (e.g., once per hour) to significantly increase transaction throughput between designated nodes. From a technical standpoint, this shifts from serially verifying locked transactions to concurrent mode. In other words, not every payment needs confirmation of the final distribution of funds by both parties; funds can be allowed to pass through first, and then reconciliation can take place later. This mode can increase the transaction rate between designated channels from around 10 transactions per second to over 2000 transactions per second (depending on device performance).

The data structure for this looks something like:

```
{
    // Reconciliation type (1. Real-time reconciliation 2. Delayed rec
onciliation)
    type: 2,
    // Direction of funds
    side: 2, // foud1 => foud2
    // 8 type Settlement channel ID
    channelId: 232353253456,
    // 8 type Channel transaction serial number, incremented automatic
ally
    autoincrement: 123135,
    // Channel fee
    fee: {
        unit: 8,
        amount: 1234,
    },
}
```

Nodes that support the fast channel mode only need to periodically compare their channel transaction serial number lists and corresponding customer payment signature lists to determine if related transactions were successful. They can then calculate the correct balances and sign the reconciliation.

For different branches of the same node, there are no security risks with fast channel mode. Nodes, customers, or merchants will not lose funds because the delayed reconciliation is limited to within the node, while the external balances strictly maintain equilibrium.

For different nodes in close cooperation, it relies on business reputation and expectations of an infinitely repeated cooperative game. Due to concurrent payments, one party may have a negative actual balance for a period of time, and withdrawals on the main network cannot occur before reconciliation is signed. Risks can be limited to an acceptable range by restricting micro-payment amounts and increasing the reconciliation frequency.

The risks associated with the widespread use of fast channel mode among merchants, customers, and nodes will be discussed in Chapter 8.

3.9 Fund Calculation

Let's assume that on average, a payment passes through three channels connected by two intermediate nodes. We can calculate:

*T: Total Locking Time = N: Number of Channels * S: Data Steps * (t: TCP Transmission Time + c: Verification Calculation Time)*

Substituting data: 3 * 3 * (20ms + 15ms) gives 315ms, which means that on average, a channel chain can support three transactions per second (at worst). If 100 units of funds are allocated to the channel chain network, with all of them being peer-to-peer contributions, then the daily transaction volume would be:

*(100: Total channel amount / (3: Number of channels * 2: Bilateral peer-to-peer contributions)) * 3: Transactions per second * 60 * 60 * 24 = 4,320,000 units*

If there is a one-way fund flow with balanced accounting, the best-case scenario is that the fund flow will double: 8,640,000 units. This means that the upper limit of fund utilization is 86,400,000 times per day, 2,600,000 times per month, and over 30,000,000 times per year. This implies that we only need to lock in 0.0000116% of the funds to support a daily payment volume equivalent to the total issuance.

Assuming a transfer fee rate of one in a hundred million, the total daily fee amount would be 0.864 units, and the net annual return rate, without considering compounding, would be approximately 315%.

4. Transactions

4.1 Basic Data Structure

In order to ensure the efficiency of the system, the design of transaction data structures should be as simple and compact as possible and should be easily understandable by both humans and machines (the importance of human-readable financial rules will be discussed on Principles of Technical Design).

Overall, it can be divided into the following three levels:

Blocks >> Transactions >> Actions

Here's an example in JSON format:

```
{
  version: 0, // Block version number
  height: 0, // Block height
  timestamp: 0, // Block timestamp

  prevHash: Buffer.alloc(32), // Hash of the previous block
  mrklRoot: Buffer.alloc(32), // Merkle tree root of all transactions

  /* other extend field ... */

  transactions: [ // All transactions included in the block
    {
      type: 1, // Transaction type
      timestamp: 12313423, // Transaction timestamp
      address: "xxxxxxxxxxxxxxxxxx", // Default primary address of the
transaction (fee spending address)
      fee: { // Transaction confirmation fee
        unit: 248, // Fee unit
        amount: 1234, // Fee amount
      },
      actions: [
      // Specific asset objects or actions of the transaction
        {
          kind: 1, // Asset or action type (1 represents a transfer)
          bill: { // Total quantity for the transfer
            dist: 2, // Precision space
            amount: new Buffer(), // Amount
            unit: 248, // Unit
          },
          address: "oooooooooooooooo", // Sending address
        }
      ],
      signs: [ // Signatures
        {
          publicKey: Buffer.alloc(32), // Public key
          signature: Buffer.alloc(64), // Signature value
        }
      ],
      multisigns: [ // Multisignatures
        {
          publicKeyScript: Buffer.alloc(32, 96), // Public key script
          signatureScript: Buffer.alloc(64, 192), // Signature result
script
        }
      ],
    },
  ],
}
```

As you can see, a transaction is roughly divided into three parts: actions, signs, and multisigns. Why not use more flexible and "advanced" transaction structures like smart contracts will be discussed in Chapter 9.

4.2 Multisignature Addresses

Single-signature addresses carry the risk of key loss or theft and cannot meet the needs of jointly custodied funds. We need a functionality that allows two or three private keys, with different permission configurations, to manage funds. For example:

1. A and B share a deposit, requiring both to provide signatures to withdraw funds.
2. A joint account for a married couple, allowing either person to spend from the joint account.
3. Exchanges, online wallets, and offline private keys requiring at least two parties to withdraw funds, to mitigate the risks of theft and loss.

The transaction structure's multisigns feature supports multisignature addresses formed by two or three private keys, creating a multisignature address. These addresses can have different permission configurations such as 1/2, 2/2, 1/3, 2/3, 3/3, and so on. Multisignature addresses can support up to 200 private keys managing a single multisignature address.

Multisignature addresses do not have a single secret private key; instead, they have multiple public keys combined into a single piece of data. From this data, a public private key is calculated. Each transaction requires providing this combined public key as the base script and a list of combined signature data for verification.

4.3 Hierarchical Equity Control Accounts

Multisignature addresses have effectively addressed the security and simplicity of co-custody requirements for private key loss or theft. However, in the face of complex business structures, mainly dealing with beneficiary and voting rights, we need an account system that supports modern corporate equity structures.

These accounts must meet the following characteristics:

1. They can be jointly managed by several private keys to avoid security issues.
2. They can change (add, delete, modify) management private keys while the address remains fixed.
3. A changeable voting effectiveness ratio.
4. Support for different rights for the same equity.
5. Preventing fund loss in extreme cases.

(1) Construction

Due to the above characteristics, we need to save and manage accounts on the mainnet. Initiating a equity account transaction looks like:

```
{
    // Valid voting ratio (in basis points, out of 10000) required for
account operation (must be equal to or greater than this value)
    validRightsRatio: 6666,
    // Composition list
    forms: [ // Up to 200 entries
        // Voting rights and beneficiary rights can be non-proportiona
l (i.e., different rights for the same stock)
        {
            address: '19aqbMhiK6F2s53gNp2ghoT4EezFFPpXuM',
            // 4 bytes, 0~4294967295, equity count
            rights: 1,
            // 4 bytes, 0~4294967295, voting rights
            votes: 3,
        },
        {
            // Members can be composite addresses
            address: '29aqbMhiK6F2s53gNp2ghoT4EezFFPpXuM',
            rights: 3,
            votes: 5,
        },
        {
            // Members can also be other equity accounts
            address: '39aqbMhiK6F2s53gNp2ghoT4EezFFPpXuM',
            rights: 2,
            votes: 3,
        },
    ]
}
```

After the mainnet confirms this transaction, it will generate a private key and public key based on the data provided in forms and the transaction timestamp. This will create a new equity account.

An equity account's members can be regular addresses, multisignature addresses, or other equity accounts, with corresponding voting and beneficiary rights. The mainnet will store all address control structures in a database, and each address will have a control tree.

(2) Verification

To check if a transaction initiated by an equity account is valid, you need to read the control tree from the database and verify if the member signatures provide enough votes. In cases where enough votes are provided, subsequent signature checks can be ignored.

An equity account may be controlled by multiple other equity accounts, and those top-level equity accounts may include members of higher-level equity accounts. This is similar to multi-level investment relationships between companies in the real world. As a result, a transaction involving a large equity account may require verifying hundreds of signatures, which can consume significant data space and slow down transaction confirmation. Charging transaction fees based on the size of transaction data space helps prevent frequent transactions for extremely large equity accounts. In practice, routine payments can be authorized by a dedicated financial account, while equity account signature transactions are reserved for low-frequency transactions, such as significant industry investments.

(3) Management

After registration on the mainnet, equity accounts can add, delete, modify members, and change ownership requirements. Changing the membership and rights requires members with sufficient votes to sign a change transaction. Similarly, the effective vote ratio can also be changed.

Members and rights can be changed arbitrarily, while the equity account's address remains unchanged.

(4) Minimum Account Balance

Equity control addresses occupy significant data space and verification time, making them valuable resources. To avoid waste, in addition to allowing transaction validators to charge regular transaction confirmation fees, you also need to lock an amount equal to the transaction fee within the account as a maintenance deposit when registering and adding (excluding changing and reducing) members to the account.

Before registering an account on the mainnet, you need to send a certain amount of funds to the equity account to ensure that the minimum account balance is sufficient.

(5) Deregistration

To prevent the expansion and waste of state database space, equity accounts support deregistration (deleting the member list and control tree). The minimum account balance will be refunded to the specified other account upon address deregistration.

(6) Fund Security

Considering the possibility of collective private key loss due to force majeure, leading to an inability to gather enough signatures to meet the required vote count, and thus the inability to perform any transaction operations, the organization's funds may be permanently lost. In extreme cases, we need the ability to safely extract funds.

The design allows any higher-level member address in the equity account's control tree to initiate a fund protection mode, with the condition that their account must lock an amount equal to 1% of the equity account's funds, with a lock-up period of six months. If another member address initiates a release within six months or if the relevant private key of the equity account is retrieved and a transaction is initiated, the fund protection mode will automatically exit.

After six months, the account initiating the protection mode can transfer the funds inside the equity account to their own account, thus avoiding the permanent loss of funds in extreme cases.

4.4 Multi-Signature Transactions

Consider a scenario of equity investment: a company (referred to as A) receives an investment of ten thousand units of capital from an investor (referred to as B) and offers 20% equity. In this situation, either party, A or B, acting alone, could be at risk of fraud. If A first transfers ownership, B may not invest; if B transfers funds first, A might renege on transferring ownership.

We need a transaction that can simultaneously perform both investment and equity distribution, where if one operation fails, the other automatically fails as well. This requires both parties to jointly sign a transaction:

```
{
    "type": 1, // Transaction type
    "timestamp": 23423442, // Transaction initiation time
    // Transaction assets and actions
    "actions": [
        {
            "kind": 6, // Transfer from 'from' to 'to'
            "from": "xxxxxxxxxxxx", // Sender's address
            "to": "oooooooooooo", // Receiver's address
            "bill": { // Transfer amount
                "dict": 1,
                "amount": 10000, // Amount
                "unit": 248, // Unit
            }
        },
        {
            "kind": 7, // Add new equity management member
            "forms": [
                {
                    "address": "19aqbMhiK6F2s53gNp2ghoT4EezFFPpXuM",
                    "rights": 3, // Beneficiary rights
                    "votes": 3, // Number of votes
                },
            ]
        }
    ]
}
```

This transaction contains two actions, and when both parties sign it, it will simultaneously complete the transfer of funds and the allocation of equity.

However, the above transaction still has a vulnerability: if the company broadcasts a transaction during the period between signing the above transaction and it being confirmed but has not yet taken effect, which issues new shares one hundred times to the original shareholders, then when this transaction is successfully confirmed, the new investor's stake has been diluted to insignificance. To prevent this, we need to include a conditional action:

```
{
    "kind": 9, // Indicates that the transaction is effective only when the voting rights and beneficiary rights of the corresponding address are above a certain percentage.
    "address": "xxxxxxxxxxxxx", // Source address
    "targetAddress": "ooooooooooooo", // Target address
    "rightPercent": 20, // Percentage of beneficiary rights
    "votePercent": 18 // Percentage of voting rights
}
```

This conditional action is placed third, indicating that after completing the investment and equity distribution operations described above, the final equity ratio must not be less than 20%.

4.5 Payment Categories

To meet the needs of modern financial payments, we should support multiple payment methods based on both UTXO and balances. Additionally, we should consider the possibility of payment service providers playing a role in providing various payment types, such as:

1. Self-pay to the counterparty.
2. Request the counterparty to pay to oneself (requiring the counterparty's signature for the transaction).
3. Have party A pay party B while oneself only covers the transaction fee (requires party A's signature).
4. Let the funds specified in the inputs be paid to the outputs (requires the signatures

of all input owners).

5. Let all the funds contained in the inputs be paid to the outputs (requires the signatures of all input owners).

For equity control accounts, there will be some special asset change operations, including:

1. Distributing a specified amount to all members according to their beneficiary rights (equity dividends).
2. Distributing a specified amount to the shareholders based on their voting rights (management incentives).

More payment methods and details about data structures will be provided in the appendix.

4.6 Signature Stripping

Due to the presence of complex payment methods and equity control systems, a single transaction may contain a large number of signatures. In fact, in a block, signatures might constitute over half of the data. To save space and facilitate fast synchronization of data by other transaction validators, signature data must be designed to be strippable.

From a technical perspective, the data and order within the signature list (including multi-signatures and composite signatures) are not included in the final transaction hash value. This separation allows the core transaction data and signature data to be stored or transmitted separately.

Stripping signatures also enables participants in a transaction to independently and concurrently sign the transaction, which is beneficial for making independent decisions in business transactions.

4.7 Transaction Fees

Considering the need for large-scale commercial payments, some service providers may act as intermediaries for customers, covering the transaction confirmation fees. There might also be some payment services that bundle or mix payments. In such scenarios, it's important to separate the fee payer's signature from those of the regular transaction participants.

Technically, a transaction should include only one fee payment method, with the fee field included in the original transaction data signed by the fee payer. Other participants' signatures should not contain information related to the fee; they only sign their own transaction actions.

Since the transaction fees on the main network are constantly fluctuating, the fee payer can adjust the fee at any time and re-sign the transaction to achieve a more economically suitable transaction confirmation time. This eliminates the need for multiple re-signatures by each participant.

For transaction validators (miners), the unique hash of the transaction does not include the fee field, allowing them to filter out duplicate submissions of the same transaction and achieve the effect of dynamic fee bidding.

4.8 Field Formats

A mature transaction payment system should be adaptable to long-term future needs, with a representation for amounts that can preserve almost infinite precision while saving space as much as possible. For example:

```
{
    "bill": {
        "dict": 1, // Indicates the positive or negative value of fund
s and the size of the amount space
        "amount": Buffer.alloc(), // Amount quantity
        "unit": 248 // Unit, represents the number of zeros following
the amount
    }
}
```

However, for data like fees, there might not be as high a precision requirement:

```
{
    "fee": {
        "amount": 1234, // Amount
        "unit": 240 // Unit, decimal
    }
}
```

Detailed field formats and explanations will be provided in the appendix.

5. Incentives

The reason an economic system can operate smoothly over the long term is quite simple:

1. It rewards those who contribute creativity and improve efficiency.

2. It avoids rewarding those who don't put in any effort.

All economic advantages stem from rule-based advantages, fundamentally rooted in institutional advantages.

Some school of thought in monetary theory argues that money is neutral, a constant that can be substituted away. Based on this absurd assumption, they neglect to address one of the most important questions in the real world: who should own money initially?

Money is either a commodity itself or a representative of commodities. Just as money that cannot be exchanged for goods cannot function, there are no commodities without property rights. Property rights systems and market economies are two sides of the same coin; without clear property rights, genuine markets cannot exist. Justice and efficiency are two sides of the same coin as well; without justice, there is no efficiency.

The fairness of property rights is the foundation of all economic efficiency. While cryptocurrencies may not achieve absolute fairness, they can significantly promote such fairness.

5.1 Competitive Bookkeeping Rewards

Maintaining the correctness and uniformity of the public ledger is the most crucial task in a cryptocurrency system, and it deserves adequate rewards. Bitcoin ingeniously combines the competition for accounting with currency issuance, making it the engine that ensures the system's efficient operation.

We establish a system where the ledger is updated every 5 minutes, generating a block containing a list of new transactions. This block is created by continuously attempting a specific hash algorithm until data satisfying the difficulty requirements is found. This process is known as Proof of Work (PoW), and the first transaction within the new block creates a certain amount of new currency, rewarding the ledger validator (referred to as a miner) who first computes the target data. Other validators receive the new block, check the correctness of the transaction amounts and signatures within it, and then begin calculating anew, attempting to find the next block's hash data that meets the difficulty requirements and thereby claim the reward. Everyone automatically adjusts the target difficulty value based on the number of blocks generated within a certain time period to ensure that the ledger is updated roughly every 5 minutes, even as computing power fluctuates.

The number of newly created coins undergoes adjustments annually in the first phase, following the Fibonacci sequence, increasing from 1 to 8. In the second phase, adjustments occur every ten years, decreasing from 8 to 1. In the third phase, it stabilizes permanently at 1 unit of output. A total of 22 million coins will be produced in 66 years. Detailed information about the currency issuance algorithm will be provided in Chapter 6.

5.2 Public Ledger Fees

While incentivizing the competition to generate new blocks, it's also essential to encourage blocks to contain as many valid transactions as possible. Otherwise, the ledger system would run idle, wasting resources. Each transaction must include a certain amount of fees, which miners, responsible for recording transactions and creating blocks, receive. The fee amount is determined by a dynamic bidding mechanism.

Furthermore, because all transactions need to be verified by all miners, the recording capacity and space on the public ledger are always limited. Beyond the price offered, we can't accurately assess the urgency and value of transactions. Paying a higher fee ensures priority processing and is a relatively more efficient way to distinguish between transactions.

5.3 Channel Service Fees

In the chapter 3, we elaborated on the channel chain settlement network, envisioning the emergence of nodes dedicated to providing fund transfer and payment services. The service fees charged by these nodes depend on market competition, hardware network costs, and channel funding expenses.

5.4 Channel Interest

Similar to telephone lines and the internet, for this system to reach its maximum potential, we need a comprehensive network composed of a sufficient number of channels. It's important to incentivize participants to lock their surplus funds into channels to provide settlement services.

Since opening a channel also requires paying confirmation fees, we have set up a mechanism to generate a small amount of new currency proportionally as a reward for the parties locking their funds in the channels. This helps offset their fee expenses. By using data formats with limited precision (omitting fractional amounts), we ensure that larger channels receive fewer rewards compared to smaller ones, which encourages the creation of more channels. Due to confirmation fees and limits on real purchase payment amounts, the scale of channel funds will remain within an appropriate range, neither too large nor too small.

We calculate interest from the successful locking of a channel, using the total funds locked by both parties as the base. Every 10,000 blocks (approximately 34 days), we compound interest at a rate of one-thousandth (0.1%) of the total channel funds. This results in an annual interest rate of approximately 1.056%. When a channel is closed, new currency is generated and paid out proportionally based on the average funds held by both parties when the channel was opened and closed.

5.5 Block Diamond

An ideal currency that exists only in theory: one with zero transaction costs and a total supply that adjusts in real-time with the growth and consumption of society's overall wealth. It's akin to an infinite reserve of virtual gold, where as productivity increases, more currency is minted into circulation, and as productivity decreases, production automatically reduces due to rising mining costs. This system aims to avoid the economic harm caused by dramatic currency fluctuations resulting in inflation or deflation. The challenge lies in the harsh reality that we can never truly achieve this theoretical perfection.

Both the bookkeeping rewards and channel interest production quantities are fixed and do not change with variations in productivity or market conditions. Hence, we need a currency growth mechanism that can adapt to fluctuations in population and technological cycles. This mechanism should adjust production based on market competition, producing more new coins as computational power increases. Importantly, it should have mining difficulty that only increases but never decreases. This ensures that new coin production immediately decreases or stops when computational power drops due to market reasons.

Block Diamond is defined as a string of data satisfying specific formatting criteria, generated from a compressed calculation of a 32-bit hash value. Each block can contain at most one Block Diamond (or none, depending on computational power). The production algorithm is as follows:

hash256((genesis_block_hash ||prev_diamond_block_hash) + belong_user_public_key + nonce_number) ==> length_16_string

Specifically, it involves taking the concatenation of the genesis block hash or the previous block hash containing a diamond, the public key of the target owner, and a random nonce number, and performing a hash operation to obtain a 64-character string, similar to:

35534631f31dfcf12200cdbad65c66ffb9d3fbd3ac985aa8a401bc4c3616bab3

The result obtained in the previous step undergoes a special compression operation where every 4 bits are mapped to characters from the list 0WTYUIAHXVMEKBSZN, resulting in a 16-character string such as:

0NMSAK0ZYNSNBAZM, 00000000IXVKHNHZ, or 0000000000UKNWTH

When the result satisfies at least the first ten characters as "0" and contains no trailing zeros, a Block Diamond is produced. Based on the results above, we designate the literal identifier for this Block Diamond as "UKNWTH," and each literal identifier is unique. At this point, the Block Diamond is included in a block and broadcasted. All Block Diamond producers cease their previous calculations and begin recalculating the next Block Diamond's literal value using the hash of the new block. If multiple Block Diamond is produced within the same block interval, miners decide which one to include in the block, possibly favoring the one with the highest transaction fee.

The total number of Block Diamonds is capped at around 17 million. Each time one is mined, the overall mining difficulty exponentially increases, approaching infinity as the number of diamonds mined increases.

Block Diamond represents a high-dimensional heterogeneous form of currency, capable of achieving dynamic adjustments in currency supply. Their value is determined by mining costs and market recognition.

5.6 Data Services

In addition to the internal reward mechanisms described above, we also need various data-related services provided by specialized data computation businesses. They will charge service fees for offerings such as:

1. Channel routing
2. Transaction mixing and bundling
3. Transaction confirmation querying
4. Malicious channel termination monitoring
5. Cryptographic private key escrow
6. Credit auditing

7. Data security auditing

These service providers may also create specialized hardware devices, such as transaction signing machines, for use in the channel chain settlement network.

6 Currency

The most powerful technologies are always dedicated to addressing the most urgent needs and solving the most serious problems. Generalized distributed public ledger technology, if it cannot first reform the world's monetary system to reduce the exploitation, oppression, and exploitation of financially vulnerable groups, then it's even more challenging to consider other issues.

6.1 Total Supply and Growth

In the long term (in a static equilibrium, not for short-term speculation), if an asset encourages hoarding, its rate of appreciation must be higher than the society's average profit rate. Otherwise, people will use it for investment. For an asset with a constant total supply, its expected rate of appreciation would be equal to the social productivity. To avoid suppressing investments and consumption, we need to introduce an expectation of inflation even if the actual monetary issuance does not exceed the real economic growth rate.

The disruptive power of a deflation trap only occurs when people are forced to use a single currency and have taken on a massive amount of debt due to inflation. Inflation also forces the poor to invest, even though they lack information advantages and risk diversification, leading to more exploitation.

Unlike paper currency or gold, once a cryptocurrency is lost, it cannot be recovered. If the total supply remains constant, it will lead to excessive hoarding and speculative bubbles in the long term, diverting economic activity towards "stock competition" instead of creating value in new areas, damaging its core function as money. To avoid these problems, we need to introduce an expectation of inflation, even if the actual monetary issuance does not exceed the real economic growth rate and does not actually depreciate.

Because it's theoretically impossible to precisely adjust the supply of currency to match changes in the scale of economic transactions (the idea of issuing 1 unit of currency for every 1 unit of goods produced in society is an idealistic illusion), the more feasible approach is to provide an observable expectation of currency issuance. This allows everyone to estimate the growth in currency over a certain period, combined with market price indices and purchasing power indices. With these estimates, people can make informed decisions on consumption, production, and sales activities after accounting adjustments.

For example, in a scenario where the purchasing power of currency is continually increasing, as long as the rate of appreciation is stable and predictable, businesses can adjust wage amounts downward proportionally over time instead of keeping them constant. Workers can accept this approach because their contract terms are influenced by the limitations imposed by the specific asset price changes. They also understand that the actual goods they can purchase haven't decreased; it's just that long-term inflation has conditioned people to expect wages to only increase.

(1) Bookkeeping Rewards

We use the Fibonacci sequence to determine the issuance of currency rewards for blocks. In the first stage, the issuance quantity is adjusted approximately every 100,000 blocks, which is roughly 0.95 years, and the rewards gradually increase. In the second stage, the issuance quantity is adjusted approximately every 1,000,000 blocks, which is about 9.5 years, and the rewards gradually decrease. In the third stage, it finally stabilizes at a constant reward of 1 unit per block.

The issuance sequence looks like this: 1, 1, 2, 3, 5, 8, 8 (ten years), 5 (ten years), 3 (ten years), 2 (ten years), 1 (ten years), 1 (ten years), 1, 1, 1, 1, 1, ...

So, the total supply for the first 66 years is 22,000,000 units: (1 + 1 + 2 + 3 + 5 + 8 + 8*10 + 5*10 + 3*10 + 2*10 + 1*10 + 1*10) * 100,000 = 22,000,000

Afterward, the annual issuance rate is approximately 0.4785% ((1*100,000)/0.95/22,000,000), and it decreases year by year (about 0.3289% after 100 years, approximately 0.2506% after 200 years, and roughly 0.1462% after 500 years). This rate approaches zero infinitely over time.

(2) Channel Interest

To promote the widespread use of the channel chain settlement network, we have set up a settlement cycle of approximately 10,000 blocks, equivalent to about 34 days. Funds locked within the channels are rewarded with compound interest, and the single-time reward rate is 0.1%. Assuming that channel funds account for half of the total, this results in an annual issuance rate of approximately:

((1+(0.001)) ^ (365/(5000/288)) - 1) / 2 ≈ 0.0053 (0.53%)

When combined with bookkeeping rewards, the estimated total annual issuance rate during stable periods is roughly between 1% and 1.5%, and it infinitely converges to around 0.53% to 1%. As a reference, the world's average GDP growth rate from 1960 to 2012 was approximately 2% to 3%.

(3) Diamond Mining

The literal value of a diamond consists of 16 characters from the set

WTYUIAHXVMEKBSZN. The final hash value, with the last 6 characters being letters, is considered a valid literal value. The total number of possible diamonds is:

16^6 = 16,777,216

We have established that it takes approximately 25 minutes to mine one diamond every 5 blocks, without considering sharp increases in difficulty. To mine all the diamonds, it would take at least:

$$16{,}777{,}216 * 5 * 5 / (60 * 24 * 365) \approx 800 \text{ years}$$

This means that a maximum of about 58 diamonds can be mined each day, resulting in an annual maximum production of approximately 21,000 diamonds.

The difficulty of mining adjusts every 3,277 diamonds (3,277 = 16^6 / 256 / 20). When the first 20 bits of a 32-bit hash value are all zeros, the mining difficulty reaches its maximum, and all the diamonds will be mined at that point. However, due to the nature of hash calculations, the mining difficulty will double each time, effectively ensuring that not all diamonds can be mined. Depending on the level of computational power, there will be a point of equilibrium (e.g., several million diamonds) where mining a new diamond will require the majority of the network's computational power. This will make the marginal yield of diamond mining increasingly smaller, while the marginal cost continues to rise, thus ensuring the scarcity of diamonds in the market.

Block diamond accumulates as surplus production and serve as a wealth repository within the economic system.

6.2 One-Way Transfer Compatible with Bitcoin

In the digital world, where "everything can be copied," Bitcoin managed to address two seemingly conflicting issues that its predecessors like B-money, Hashcash, and others couldn't resolve: double-spending and decentralized issuance management. Prior cryptocurrencies either introduced a "ledger center" or a "coin minting authority" to prevent double-spending, or they allowed decentralized issuance and management but had to tolerate the existence of double-spending. These two seemingly irreconcilable issues were among the primary reasons previous cryptographic systems didn't receive widespread attention. Bitcoin ingeniously unified the maintenance of the ledger and the issuance of new currency through the concept of the blockchain, achieving scarcity in the digital realm. This innovation made Bitcoin the first digital commodity with "intrinsic value" broadly recognized.

We acknowledge Bitcoin's immense revolutionary value but recognize its imperfections, particularly in its "currency" aspect. For example, its fixed total supply of 21 million coins and halving of production every four years position it more as a digital commodity with intrinsic value than as a daily payment and settlement currency. Our mission is not to technologically overthrow or replace Bitcoin but to leverage the blockchain and proof-of-work issuance technologies pioneered by Bitcoin. We aim to expand, enhance, and perfect the system's "monetary" metrics, based on the "commodity money theory." These improvements will help create a comprehensive hierarchical monetary system in conjunction with Bitcoin and facilitate the widespread use of cryptocurrencies in personal payments and commercial settlements through a real-time settlement network.

Proposed solutions like "Bitcoin hard fork" or "Bitcoin Layer 2" can temporarily increase the system's transactions per second but fail to fundamentally improve its "monetary" deficiencies. We believe that "monetariness" is an inherent and unalterable characteristic because money fundamentally represents a "value exchange contract" that can be executed in the future, based on expectations of scarcity. This contract, when initially established, is in equilibrium. However, if this contract can be easily altered in the future, it implies that at least one party will incur unforeseen losses. The expectation of irredeemable losses will significantly impede the fulfillment of its monetary function, rendering a non-marketized monetary system a failure. It is not technological constraints like "block size" that hinder Bitcoin from becoming a daily payment and settlement currency but its fundamental "monetary" shortcomings.

We acknowledge Bitcoin's tremendous revolutionary value and base our improvement plan on it. In practice, whether Bitcoin is stored in exchange accounts or issued as Bitcoin-pegged tokens through collateralization on Ethereum, these "Bitcoin vouchers" are recognized as having the same value as the asset they represent. In reality, for cryptocurrencies like Bitcoin, which lack physical form, the recognition of value is not related to their location or presentation but to their "proof of scarcity." We employ the concept of "irreversible one-way transfer" as a systemic improvement.

The fundamental technical principle is as follows: In the new system, we use the same private key-address account generation algorithm as Bitcoin. We send Bitcoin from a specific address on the Bitcoin mainnet to a technically generated "black hole address" in integer amounts. Simultaneously, within the new system, a corresponding "transferred Bitcoin" is generated and sent to the original payer's account, proving the scarcity (total supply) of the Bitcoin transfer through technical means. This process is irreversible, and since Bitcoin is not sent to an account that anyone can access, there is no trust delegation risk involved.

However, because the new system initially cannot gain the same level of attention as the original Bitcoin mainnet, the first few Bitcoins transferred to the "black hole address" bear significant value risk. If the new system fails to gain widespread recognition, these initially transferred Bitcoins could be considered lost or destroyed. Nevertheless, assuming that more Bitcoins are transferred, and more people recognize the value of the new system, this risk diminishes, eventually becoming negligible. More importantly, as more "transferred Bitcoins" accumulate in the new system, they will provide higher added value to the system, greatly enhancing its applications in payment settlements and open finance.

For these two reasons, we design a mechanism to issue new currency concurrently with Bitcoin transfers. This mechanism serves as both a risk mitigation and a reward for the increased value in the new system. These newly issued coins are sent to the accounts that transferred Bitcoin to the "black hole address." Initially, the first few Bitcoins transferred receive more new coins, but this issuance gradually decreases until it eventually stabilizes at one new coin being issued for each Bitcoin transferred. To mitigate market volatility risk and suppress short-term speculative behavior, the new coins issued in the initial stages of Bitcoin transfers will be locked and released linearly on a weekly basis (the transferred Bitcoins themselves will not be locked). The lockup period for the first new coin issued for transferred Bitcoin is approximately 20 years, followed by roughly 5 years for the second and third coins, and approximately 2.5 years for the fourth, fifth, sixth, and seventh coins. This pattern continues until a sufficient amount of Bitcoin has been transferred, and the issuance of new coins becomes small enough to have minimal impact on market volatility, at which point the lockup on further issuance is lifted. Specific issuance amounts and lockup periods are detailed in the appendix.

In conclusion, the new system incorporates three distinct levels of heterogeneous digital currency:

1. block diamonds with an absolute finite total supply, indivisibility, unique identification, and ever-increasing mining difficulty.
2. Transferred Bitcoins with a finite total supply and divisibility.
3. New currency with an infinite total supply and infinite divisibility. These three forms of currency are generated from three sources:
   a. PoW Mining
   b. Transferred Bitcoin issuance
   c. Channel chain settlement network interest

6.3 Units and Symbols

For cryptocurrencies to truly find applications in the realm of business payments, as opposed to becoming collectibles like gold, they must meet the criteria of facilitating large-scale, secure, real-time transaction settlements, stable incremental production, and infinite divisibility.

Infinite divisibility ensures that regardless of the scale of economic development, microtransactions can always be conducted. Digital cryptocurrencies should completely avoid the transaction cost issues caused by the physical form of traditional currencies.

We employ a special data structure similar to scientific notation to store monetary amounts:

```
{
    "bill": {
        // 1 byte, 0~255, Unit (indicating the number of trailing zero
s)
        "unit": 248,
        // 1 byte, indicates the space occupied and sign, 1~127 for po
sitive, 128~255 for negative
        "dist": 1,
        // 1~127 bytes, transfer amount quantity
        "amount": Buffer.alloc()
    }
}
```

Where unit represents the decimal unit, for example, {amount: 1, unit: 4} signifies 1000, and {amount: 137, unit: 8} denotes 13700000000.

We set unit=248 to represent one currency Mei and use 100 million as the base. Thus, 1 Mei = 100 million Zhu, and we establish five units accordingly:

1. unit:248 represents 1 Mei = 10^8 Zhu
2. unit:240 represents 1 Zhu = 10^8 Shuo
3. unit:232 represents 1 Shuo = 10^8 Ai
4. unit:224 represents 1 Ai = 10^8 miao
5. unit:216 represents 1 miao

In everyday accounting, 273.58 Zhu can be recorded as さ273.58:240, while 1 Mei can be noted as さ1:248.

The introduction of the unit:248 implies that we can divide one unit into 10^248 parts, observable in the context of the universe where the order of magnitude of atomic quantities is approximately 10^80.

6.4 Prohibition of Artificial Monetary Policy

Currency should not be used to regulate the economy; this is both lazy and too dangerous.

The modern business and economic ecosystem has become increasingly complex, evolving from sparse grasslands to the complexity of the Amazon rainforest. Attempting to regulate the economy with monetary policy is akin to controlling the growth of a rainforest solely by adjusting precipitation. Grasslands thrive with water, but the formation of a rainforest requires numerous conditions and time—things are not as simple as they seem. A healthy economic ecosystem can only grow and evolve under suitable conditions, not through precise design. Today's "monetary policy" has become an institutionalized system of exploitation and wealth redistribution.

People tend to be overly confident in themselves, yet they struggle to comprehend that emergent patterns in certain complex systems cannot be entirely deconstructed and modeled. The emergence of cryptocurrencies is not meant to replace fiat currencies but to create entirely new financial rules and business models in a fresh context. Forcing the chronic issues of traditional monetary and financial systems onto the cryptocurrency framework is misguided. Financial and economic rules have long been manipulated by powerful interest groups. We need to fight for the financially disadvantaged, ensuring that the fruits of hard work are not wantonly defrauded. It's crucial to note that the ultimate beneficiaries of extensive and prolonged monetary policy interventions will inevitably be those closest to money and power. This leads to an inevitable widening of the wealth gap, with the poor sinking deeper into poverty, making it impossible for them to escape the trap through their efforts alone. At this point, people may anticipate a stronger government to enforce redistributive measures, and societal, cultural, and economic production may plummet or regress by decades, ultimately leading to a catastrophic human tragedy.

The idea of allowing certain individuals or organizations with significant influence to vote and deliberate on changing the core value parameters of the monetary system, such as modifying the algorithm, quantity, or speed of currency issuance, is incredibly foolish. The key to the future monetary system lies in providing everyone with an unmanipulable and stable expectation. If certain core value parameters are set unreasonably or are not adaptable, let better ones replace them.

7 Privacy

It seems that some people cannot understand that we operate transparently but still prefer not to be known. The most significant issue is that without privacy, the "fungibility" of currency will be greatly affected. As a result, the same unit of cryptocurrency may have different market prices forced upon it due to its transaction history, thereby reducing the efficiency of the entire monetary system. Merchants who steal consumer data will analyze past purchasing behavior to offer a price for each individual that is just acceptable, greatly harming our interests—especially when the offered product is a monopoly. The consequences can be easily imagined.

7.1 Anonymity

In a public ledger, anonymity is essentially a pseudonym and cannot always be maintained. Because at certain points, we are either voluntarily or involuntarily required to disclose our identities, it becomes possible to trace the entire transaction chain and expose all privacy. Anonymous addresses are just the foundation; we need additional measures to sever the direct connection between the payer and payee accounts, thereby avoiding tracking.

7.2 Payment Mixing

In typical transfer transactions, payments and receipts correspond one-to-one; it is one person initiating payment to another, and this is publicly broadcasted, making it easy to infer the inevitable connection between the two.

A feasible solution is to have a group of people collectively initiate transfers of the same amount to another group of people. This makes it impossible to precisely match the payee with the payer. The more people involved in the transfer, the better the privacy protection. This is known as fixed-amount payment mixing. The transaction structure might look like this:

```
{
    // Fixed-amount payment mixing
    "kind": 6,
    "fee": { // Service fee charged separately for each address, can be zero or even negative
        "amount": 1234,
        "unit": 248,
    },
    "bill": { // Unified transfer amount
        "dist": 1,
        "amount": Buffer.alloc(),
        "unit": 248,
    },
    "addressCount": 100, // Number of addresses participating in mixing
    "inputAddresses": ["1313Rta8Ce99H7N5iKbGq7xp13BbAdQHmD", "..."], // Multiple paying addresses
    "outputAddresses": ["19aqbMhiK6F2s53gNp2ghoT4EezFFPpXuM", "..."], // Multiple receiving addresses
}
```

This mixing method does not rely on complex technologies like ring signatures, making it simple and practical. It has the following advantages:

1. Reduces the size of transaction data, improving the throughput of the mainnet, and saving fees.
2. Some individuals with strong privacy requirements might attract enough mixing participants through zero fees or even subsidies, creating a win-win situation.
3. No need for the payee to participate in the signature (offline receipt).

However, there are also some disadvantages:

1. Transfer amounts are highly standardized, making it difficult for typical commodity purchases.
2. To accommodate different transfer volumes, it needs to be divided into multiple integer gradients for mixing, making it harder to gather a sufficient number of

participants.

3. There is still a possibility of being traced, although the probability can be reduced to an extremely low level through multiple rounds of mixing.

7.3 Forward Deferred Payment

Payment mixing blurs the direct connection between the sender and receiver of funds in "space" (while also disrupting the order of payment and receipt). To provide more assurance, we need to sever the direction of funds over "time." The principle is to utilize an intermediary who immediately transfers the funds but defers their receipt, making it impossible to know the final recipient of the transaction for a certain period.

Assuming the payer is A, the intermediary is B, and the ultimate recipient is C, with a transfer amount of 100 units, the basic steps are as follows:

1. B creates a transaction (trs1) to transfer 100 units to C and sends it to A.
2. Upon receiving trs1, A creates a special transaction (trs2) transferring 101 units (including a 1-unit fee) to a hashed temporary address (addrx). Trs2 specifies that B can only claim the 101 units from addrx after a certain period when trs1 becomes effective. A shows trs2 to B to ensure the safety of the funds.
3. A signs trs2 and broadcasts it to the main network for confirmation.
4. B signs trs1 and broadcasts it to the main network for confirmation. C receives the 100 units and can use them immediately.
5. After a certain time (e.g., 6 months), B initiates another transaction (trs3) to claim the 101 units from the temporary hashed address addrx, completing the entire transaction.

The above method has an implicit security risk due to the sequential order of sending and receiving funds. A sends funds to a hashed address, but B may lose the private key and be unable to sign the trs1 transaction, resulting in A losing the 101 units.

It's important to note that B has no deliberate incentive not to sign because completing the entire transaction earns B the corresponding fee. Even if B's account balance is insufficient, they can borrow from friends and return it immediately after retrieving the funds and the fee. Two situations might trigger a loss:

1. B loses the private key or is unable to sign due to uncontrollable reasons.
2. B maliciously wants A to incur a loss.

To mitigate this security risk, A can include a condition when constructing trs2: if B hasn't claimed the funds after a timeout (e.g., one year), the 101 units can be returned to A's account. The risk is then transferred to B: B needs to claim the funds within a 6-month window, or there's a potential loss of 100 units. Since B earns a fee, this risk can be offset from the earnings.

The trs1 is a regular transfer transaction, and the trs2 transaction data structure is as follows:

```
{
    // Sending funds to a one-time encrypted address
    "kind": 9,
    "bill": { // Amount
        "dist": 1,
        "amount": Buffer.alloc(),
        "unit": 248,
    },
    // 32-bit hash used as the encrypted address = sha3-256(previous transaction hash + transaction confirmation block + specified receiving address)
    "hashaddr": Buffer.alloc(32),
    // Timeout for retrieval, indicating that the funds can be retrieved by the user if unsuccessful after one year
    "overback": 105120, // Block count, set to zero for permanent validity
}
```

The trs3 transaction, to withdraw funds from the hashed address, has the following data structure:

```
{
    // Withdraw funds from a one-time encrypted address
    "kind": 10,
    // Encrypted address from which funds are withdrawn
    "hashaddr": Buffer.alloc(32),
    // Must be an existing transaction hash (precondition for withdraw
ing funds)
    "existTransaction": Buffer.alloc(32), // Hash from the previous tr
ansaction (as mentioned in trs1)
    // Number of confirmed blocks for the previous transaction
    "confirm": 50000, // Approximately six months
    // Specify the receiving address as the fee payment address, and t
he hash calculated with existTransaction and confirm must be equal to
hashaddr, proving the authority to withdraw
}
```

Except for the risk of locking up funds, forward deferred payment will not cause anyone tolose funds. Under conditions of sufficient trust (e.g., real-time signing of trs1 and trs2), trs2 can be set to be permanently valid. This allows the intermediary B to broadcast the trs3 transaction only when needed, which may be a long time (e.g., 5 years), thus ensuring A's privacy safety to a greater extent.

7.4 Encrypted Settlement Network

While payment mixing can reduce the probability of tracking, transactions are still public. If an enterprise or organization possesses a significant amount of identity information corresponding to account addresses and conducts big data analysis, we are still at risk.

An encrypted channel chain settlement network, in addition to being able to expand transaction throughput as needed, can protect our privacy to some extent since almost all payment data propagates off-chain and is not public. Conventional payment channel nodes might be required to provide identity information for all connected customers and could potentially leak every transaction's consumption data.

One feasible solution is to encrypt channel transactions using the public keys of payment participants (including the payer, payee, and intermediate nodes), thus avoiding interception by unrelated parties and preventing information leakage. However, the drawback is that this solution still relies on the security measures and confidentiality strength of the nodes.

7.5 Channel Reversal

Strictly speaking, channel reversal only hides the actual amount of funds owned by both nodes within the channel in each specific channel chain payment, but it does not technically guarantee it. Malicious actors can still calculate the respective funds owned by scanning the channels registered on the mainnet.

However, in a fast channel, since real-time confirmation of fund amounts is not necessary, it is impossible to know the funds of both parties in the channel unless all transactions in the channel within a settlement period are obtained.

While we aim to protect consumer privacy and essential business secrets, we do not provide an absolute anti-audit feature because the latter would consume several times the data space and computational resources compared to the former and could easily become a protection umbrella for malicious activities such as ransomware and mining trojans. We will discuss this in the design principles in Chapter 9.

8 Risks and Precautions

In this chapter, we will discuss the system's dependency conditions, potential risks, and arrangements for corresponding defense measures, highlighting key considerations.

8.1 Channel Chain Delayed Signature Attack

Channel chains, especially when using fast channels, significantly enhance the overall transaction throughput of the system. In a more mature stage, almost all routine product payments and transfers are conducted through the settlement network.

To ensure the security of instant payments from strangers and real-time receipt, a lock is generally maintained from channel establishment until the final payment receipt is signed. This lock period may vary from 100 milliseconds to a few seconds depending on network conditions. During this time, as the channel is in an exclusive state, it cannot process other transactions. Consequently, competitors of service nodes or malicious disruptors can initiate a massive number of very small payments targeting a specific channel. They deliberately delay payment signatures, reaching the lock timeout each time. As a result, the attacked channel becomes clogged with a large number of small payments for an extended period (similar to a DDoS attack on the internet), making it unable to process other normal business payments.

The solution is for each node to record temporary cache data, which keeps track of the payer's address, cumulative payment amount, and total lock time occupied. The ratio of cumulative payment amount to total lock time gives a score, representing the channel utilization score per unit of time. If this score is unusually low or significantly abnormal, it can restrict the payment frequency for the corresponding address or even refuse service, thereby preventing such attacks.

8.2 Low-Cost Channel Fraud

The normal operation of the channel chain settlement network relies on a severe punishment threat: any malicious behavior will result in the seizure of all account funds. Considering that at certain times, one side of the channel may have very low or zero actual fund balance, there is an incentive for that side to broadcast the favorable balance to the mainnet. If the other party does not rigorously monitor this, they will lose funds after some time. The cost for the malicious party is only the mainnet confirmation fee.

To address this trust issue, it is challenging to rely solely on technical means. Instead, certain insurance and penalty mechanisms can be employed:

1. Agree to reserve a certain amount of risk deposit for each account, which can serve as insurance for multiple channels simultaneously. If any of the channels defaults, the risk deposit will be seized. At this point, all other channels will be closed to prevent further disruptive actions.
2. Commercial service nodes can mutually disclose their channel lists and identity information. If one party breaches the agreement, cooperation on all channels can be terminated, and the malicious behavior can be publicly exposed, forming a strict penalty.

By limiting the risk and losses of channel defaults within a certain range, we can prevent the formation of a systemic threat to the entire settlement network.

8.3 Channel Credit Currency Creation and Default

The methods to significantly increase transaction throughput across orders of magnitude essentially boil down to two types:

1. Data and service centralization.

2. Allowing temporary local data inconsistencies.

The principle of fast channels in the channel chain falls into the latter category, postponing reconciliation time and reducing the number of reconciliations.

This will lead to a situation where the expenditure on a specific channel exceeds the initially locked amount at a certain time, and its available balance is actually negative. The channel's capacity includes the locked funds and the debt from one party to the other. At this point, credit currency is created, similar to the partial reserve system in banks that leads to credit expansion. The entire system magically creates a large amount of new currency!

We should pay great attention to the systemic risks of the payment network at this point. When certain channel parties have huge debts and cannot continue repayment, it will lead to a chain reaction of defaults. At this point, financial crises and economic collapses, in the traditional sense, occur within the cryptocurrency system.

However, it is speculated that in a cryptocurrency system without a central bank (ultimate lender), participants may not excessively borrow (or may not allow the other party to borrow too much if they clearly cannot repay). This is because everyone ultimately has to foot the bill themselves, and there is no way to transfer and socialize losses through so-called quantitative easing policies or financial assistance, forcing a large number of unrelated ordinary people, especially the poor, to bear the burden.

Another aspect that needs careful consideration and prevention is centralized cryptocurrency exchanges. If we delegate our cryptocurrencies to them for long-term management in the usual way, they will become banks. We need to establish a wide-ranging channel chain settlement network to replace the role that banks play in traditional economies. Importantly, it needs to be ensured that no one can steal our money, either directly or indirectly. If entry-restrictive banks and a partial reserve system emerge in the cryptocurrency system, this would be the greatest irony for everyone.

8.4. Centralization of Hash Power, 51% Attacks, and Guerrilla Mining

Bitcoin sought to avoid excessive centralization of power by abandoning IP address-based consensus algorithms and opting for a competition based on CPU computations. However, the unforeseen dominance of Application-Specific Integrated

Circuits (ASICs) has led to a serious centralization of hash power. In an open-access, freely competitive field, the gradual concentration of resources and personnel seems unavoidable due to the economic advantages of scale, resulting in lower costs and increased competitiveness. The concern is not so much about the centralization of hash power itself but the predatory, fraudulent, and destructive actions that may arise from it. Moreover, the fear is not about monopoly per se but rather about restricted access.

As the cryptocurrency space matures, specialization becomes more apparent, and almost all mining hash power can be temporarily rented. This guerrilla mining behavior can cause significant hash power fluctuations for smaller-market-cap coins, putting them at a severe risk of 51% attacks. This situation hampers the development of newer and potentially superior currencies, leaving the larger, older currencies with mediocre features to dominate the market.

In theory, 51% attacks are primarily a concern for centralized exchanges rather than individual users because of their substantial fund operations, making them attractive targets for attackers willing to take risks. If everyone uses channel chain networks to form exchanges, this type of attack would lose its target.

In essence, this problem cannot be completely solved, only mitigated. Several methods can be effective:

1. Inventing New Mining Algorithms: This aims to avoid or delay the appearance of specialized hardware that leads to the centralization of hash power.
2. Requiring a Stake for Mining Recognition: Requiring a certain amount of funds to be staked to be recognized as a miner makes miners stakeholders, reducing malicious incentives.
3. User Voting for Fork Proposals: Introducing a mechanism where "real users" vote

for proposed forks by "honest" miners to choose the "correct" fork.

(1) X16RS Hash Algorithm

X16RS is an upgraded version of the X16R algorithm, utilizing a base principle of randomly combining 16 different hashing algorithms to resist ASIC dominance. X16RS improves by randomizing the hash algorithm at each step, making it more challenging for FPGA designs to run efficiently.

(2) Historical Witness Path Selection

The 51% attack is a significant obstacle to the widespread use and adoption of cryptocurrencies. For smaller, newly created cryptocurrencies with less hash power, the potential risk of a 51% attack can stifle the development of more effectively designed new currencies, allowing those with a first-mover advantage to dominate the market.

Proof-of-Work (PoW) mining algorithms solve the core problem in a shared currency system: determining who owns the next batch of newly issued currency. In other words, the competitive mechanism of PoW determines the future.

The principle of a 51% attack is that a miner (or a very small number of miners) secretly applies hash power greater than the combined power of all other miners to calculate a longer chain without anyone knowing. After a period (a few blocks later), they suddenly broadcast the longer chain to the entire network, forcing everyone to abandon the recognized chain and switch to the attacker's fork, allowing the attacker to retract already confirmed transactions (double spending).

Essentially, a 51% attack is a form of rewriting history.

The only difference between an attacker and honest miners is whether they immediately broadcast the mined block to everyone. If no one conceals a newly mined block, there is no attack. The key question is how to force everyone to broadcast blocks promptly through incentives or penalty mechanisms, or design a mechanism to make the concealed block unacceptable.

To achieve this, a historical witness mechanism algorithm is introduced. It involves accounts in the network with the most currency voluntarily signing hash signatures for broadcasted blocks (those with the most currency have the most incentive to maintain system security) and adds up the balances of all participating witnesses to calculate a "witness value" written into the publicly broadcast chain. Honest miners will do their best to broadcast their blocks to these witnesses, obtaining their witness endorsements to avoid their mined blocks being rolled back, wasting hashing resources. The "witness value" represents the degree of broadcast of a block (in other words, "the higher the witness value, the more thorough the broadcast"). When an attacker conceals a secretly mined chain and broadcasts it to the network after a period, all miners will compare the total "witness value" of the two forks and choose to continue mining the fork with the larger "witness value." Since the essential feature of a 51% attack is to conceal blocks and keep them unknown, the "witness value" of the attack's fork cannot surpass the fork that has been thoroughly broadcast, making the attack unsuccessful.

In summary, this introduces a "witness value" to force all mining participants to broadcast newly mined blocks promptly, avoiding the secretive withholding of mining chains. It efficiently addresses the 51% hash power attack problem with minimal system overhead. Essentially, PoW determines the future, while PoS (Proof-of-Stake) determines history.

At this point, for an attacker to succeed, they theoretically must simultaneously possess more than 50% of the hash power and more than 50% of the monetary funds. If such a miner exists, their rational behavior would be to maintain the overall security of the system, as attacking it would ultimately harm their own interests. In practice, a widely-known and used cryptocurrency will not have a single node with more than 50% of both hash power and funds, making an anattack improbable.

However, there are still two flaws in this scheme:

1. Witnesses do not receive any monetary rewards outside the interest community.
2. Witnesses need to be online in real-time, potentially posing security risks for attacks and theft.

The first point is a deliberate design because witnesses and mining pools can sign private contracts for efficient pricing, eliminating the need to consume consensus resources in public agreements.

The second point's risk mitigation requires the improvement and widespread adoption of commercial signature hardware devices. As block hash signatures are fixed and unchanging structured data, hardware devices can handle this situation well and are resistant to external attacks, ensuring real-time online status while keeping private keys secure.

In the block header, a uint16 (two bytes) is used to store the "witness value":

```
type Block_v1 struct {
    // Version   fields.VarInt1
    Height           fields.VarInt5
    Timestamp        fields.VarInt5
    PrevHash         fields.Bytes32
    MrklRoot         fields.Bytes32
    TransactionCount fields.VarInt4
    // meta
    Nonce        fields.VarInt4 // Mining random value
    Difficulty   fields.VarInt4 // Target difficulty value
    WitnessStage fields.VarInt2 // Witness quantity level (represents
2^X funding levels)
    // body
    Transactions []typesblock.Transaction
}
```

(3) Fork Selection by Vote

In theory, this is not a technical guarantee but a deterrent. The results of the vote are not mandatory, and the power to choose which fork to follow remains in the hands of all miners. There is still an assumed premise here: the majority of miners and users are honest and willing to collaborate to maintain the normal operation of the system.

It is assumed that users who lock their funds in channels for an extended period are the most genuinely effective and relevant users in the system. They are given voting rights proportional to their funds in the channel. When the system is under a 51% double-spending attack, channels established within the last 10,000 blocks (approximately 35 days) have the eligibility to vote. They broadcast a voting transaction to all honest miners:

```
////////////////  Preventing 51% Attack Voting  ////////////////
{
    // Used for voting to elect the blockchain, urging miners to switc
h to a universally recognized chain.
    // The number of votes is based on channel-locked funds, and conve
rsion is initiated when a certain number of votes (determined by miner
s) is reached.
    // This is to rectify forks caused by a 51% attack, activated only
during critical moments.
    // Only channels with valid transactions within the last 10,000 bl
ocks (35 days) are eligible to vote.
    kind: 19,
    // The blocks' hash that the blockchain must include, usually the
starting block of a fork.
    // It must be a hash that already exists in the history of this bl
ockchain.
    targetHash: Buffer.alloc(32),
    // List of channel IDs participating in the vote
    channelIds: [
        232353,
        3847658374,
        874568376455,
    ],
}
```

Honest miners receive several such transactions and accumulate the total funds as votes. When the votes reach a negotiated threshold, all miners switch to the honest chain for continued mining.

Attackers consider that even if they invest a significant amount of hash power and successfully execute a 51% attack, there is a risk of other honest miners and users collectively voting to nullify the fork caused by the attack. This means that the attacker would essentially be the only one recognizing their own fork, unable to seize any economic benefits. Fork voting is a form of equity deterrence, akin to damaging one's own nuclear weapons. It is crucial to always be vigilant about risk monitoring and have voting preparations in place to make potential disruptors find no opportunity, but the use of this mechanism should not be taken lightly.

The methods outlined above can to some extent reduce the risks of hash power centralization, guerrilla mining, and 51% attacks.

8.5 Extreme Price Volatility

Cryptocurrency, fundamentally, should serve as a commodity and will inevitably face relative price fluctuations, whether in comparison to fiat currencies or a basket of goods. However, as a currency with functions of a settlement unit and store of value, it needs to ensure stability in its relative price.

The reasons behind extreme price fluctuations in cryptocurrencies can be broadly categorized as:

1. Speculative fervor for something new
2. Currency issuance algorithms prone to hoarding and speculation
3. Large concentrations of currency in a few hands leading to price manipulatio
4. Technical or mechanism errors causing confidence collapse

Points 2 and 3 can be mitigated through more reasonable currency issuance algorithms. When designing the currency issuance mechanisms for three stages, we thoroughly considered two objectives:

1. Disperse the currency among a large number of individual users as much as possible.

2. Ensure that the currency's growth aligns with economic principles to avoid excessive hoarding.

We have already explained the impracticality of artificially controlling the issuance and letting certain institutional leaders have discretionary power to maintain currency price stability. Beyond this, the only option is to anticipate the growth of its market value to a sufficiently large scale, capable of absorbing and offsetting localized conflict expectations and price fluctuations. This can also be achieved by utilizing hedging in the futures market. Like any other commodity, the risk of price fluctuations cannot be completely eliminated; it relies on sophisticated and developed financial markets for hedging, albeit with additional accounting costs.

All currencies are imperfect, but some can work more effectively than others.

9 Principles of Technical Design

(1) Simplicity and Intuitiveness

Financial system software cannot bear the losses caused by software vulnerabilities. Especially in an open and shared system, no one is responsible for your losses. The risk of potential vulnerabilities in "smart contracts" is very high, making it impractical for large-scale use in financial transactions. Another significant issue with "smart contracts" is that ordinary people cannot understand the actions in the contract code and always need the assistance of professional coding experts. This greatly limits its application scenarios, raising the threshold to a level where ordinary people cannot use it.

For an open and shared financial software system, we need a standardized set of instructions that are readable by humans. This allows users with no technical background to easily understand the details of protocol contracts, without any potential vulnerabilities. The degree of human understandability and intuitiveness is crucial.

(2) Compact Data and Efficient Execution

It is necessary to balance the relationship between the generality and efficiency of the protocol, even considering saving every byte of space and the time consumption of each instruction. The elegance in the design of program modules should yield to efficiency in the core critical parts.

Elegantly designed computer systems like Lisp and operating systems where everything is an object, such as SmallTalk, both faced failure. History chose the economically driven principles of compromise and "dirty implementation" found in the C language and UNIX systems.

(3) Controllable Scale of Public Ledger Data

Elevating the block space size and block frequency to a level unsupported by a typical device would lead to centralization of ledger accounting power in practice, jeopardizing the overall system's security.

We need a controllable data growth plan and a scale of controllable transaction processing resources to ensure the decentralization of ledger processing and recording. The technical indicators of the main network ledger do not need to be eternally fixed, but should roughly be limited to a range that a typical household's mid-range mainstream computer can handle.

(4) Signature Stripping and Data Compression

After a sufficient amount of time (e.g., one year), blocks effectively cannot be rolled back, and historical transactions become an irrefutable existence. At this point, we should support the stripping of signatures, which occupy a significant portion of block data, and compressing transaction data for storage. This supports updating and querying the ledger on devices with lower hardware performance or storage space.

For longer historical data, for ordinary ledger nodes, a snapshot of all "state data" can be taken at a specific moment each month or year using a data consistency algorithm. It can be written into the main chain and recognized by everyone. Newly added general ledger nodes can synchronize later blocks from a snapshot taken at some point in the middle, abandoning the tracing and verification of all transaction history since time immemorial. This significantly reduces the load and speeds up the availability time.

10 Conclusion

In summary, we have proposed a system for the issuance, circulation, and value storage of cryptocurrencies that can be used for large-scale real-time settlement of payments.

We first discussed the fundamental principles and technical details of the channel chains settlement network. We believe that a global public ledger with incentives, acting as the ultimate arbitration and clearing guarantee, can support the smooth conduct of massive payments privately, saving a significant amount in transaction fees and trust costs, given the existence of strict penalty mechanisms for default. The system is characterized by a highly market-oriented issuance mechanism for new currency that aligns well with economic principles, as well as strict guarantees of fund security and real-time settlement. Moreover, it does not rely on any central authority. We have fully considered division of labor and control of rights in a mature business environment, designing a rich set of transaction categories and technically sound protocols with simplicity, intuitiveness, and no latent vulnerabilities.

We also discussed currency issuance rules, Bitcoin-compatible integration solutions, the importance of privacy, protection for financially vulnerable groups, and potential risks and precautions.

This framework encompasses the overall rules and incentive measures needed for a fair, efficient, and trust-delegated cryptocurrency issuance, circulation, value storage system, and a large-scale payment settlement system.

Appendix

Appendix 1 Example and Annotations of Block Data Structure Definitions

```
type Block_v1 struct {
    // Version    fields.VarInt1
    Height           fields.VarInt5
    Timestamp        fields.VarInt5
    PrevHash         fields.Bytes32
    MrklRoot         fields.Bytes32
    TransactionCount fields.VarInt4
    // meta
    Nonce        fields.VarInt4 // Mining random value
    Difficulty   fields.VarInt4 // Target difficulty value
    WitnessStage fields.VarInt2 // Witness quantity level (represents 
2^X funding levels)
    // body
    Transactions []typesblock.Transaction
}
```

Appendix 2 Partial Algorithm Code

(1) X16RS Hash Algorithm

```
function X16RS_HASH(prevhash_buf, stuff_buf) {
    function SHA3_256(a) { return crypto.randomBytes(32) } // Assume
    var hashfuncs = [ // Assume
        function Blake(a) { return crypto.randomBytes(32) },
        function BMW(a) { return crypto.randomBytes(32) },
        function Groestl(a) { return crypto.randomBytes(32) },
        function Jh(a) { return crypto.randomBytes(32) },
        function Keccak(a) { return crypto.randomBytes(32) },
        function Skein(a) { return crypto.randomBytes(32) },
        function Luffa(a) { return crypto.randomBytes(32) },
        function Cubehash(a) { return crypto.randomBytes(32) },
        function Shavite(a) { return crypto.randomBytes(32) },
        function Simd(a) { return crypto.randomBytes(32) },
        function Echo(a) { return crypto.randomBytes(32) },
        function Hamsi(a) { return crypto.randomBytes(32) },
        function Fugue(a) { return crypto.randomBytes(32) },
        function Shabal(a) { return crypto.randomBytes(32) },
        function Whirlpool(a) { return crypto.randomBytes(32) },
        function SHA512(a) { return crypto.randomBytes(32) },
    ];

    var hashloopnum = hashfuncs.length;
    var stephashs = [];

    for (var i = 0; i < hashloopnum; i++) {
        var funcidx = prevhash_buf.readUInt8(31) % hashloopnum;
        prevhash_buf = stuff_buf = hashfuncs[funcidx](stuff_buf);
        stephashs.push(stuff_buf);
    }

    stuff_buf = Buffer.concat(stephashs, hashloopnum * 32);
    return SHA3_256(stuff_buf);
}
```

(2) Block Reward New Currency Quantity

```
function calcBlockCoinBaseReward(block_height)
{
    var rwdns = [1,1,2,3,5,8,8,5,3,2,1,1] // length must uneven number
    , frix = parseInt(rwdns.length / 2)
    , pos = parseInt(block_height / (10000*10)) // almost 1 year
    // console.log(frix, pos)
    if(pos < frix){
        return rwdns[pos]
    }else if(pos < frix+((frix+1)*10)){
        return rwdns[frix + parseInt((pos-frix)/10)]
    }else{
        return rwdns[rwdns.length-1]
    }
}
```

(3) One-way transfer of Bitcoin for issuing new currency and the lock-up period

Rank	BTC Transfer Amount	Total BTC Transferred	HAC Reward Amount	Total HAC Rewarded	Lockup Period(Weeks)	HAC(Weekly) Release Amount
1	1	1	1,048,576	1,048,576	1,024	1,024
2	2	3	524,288	2,097,152	512	1,024
3	4	7	262,144	3,145,728	256	1,024
4	8	15	131,072	4,194,304	128	1,024
5	16	31	65,536	5,242,880	64	1,024
6	32	63	32,768	6,291,456	32	1,024
7	64	127	16,384	7,340,032	16	1,024
8	128	255	8,192	8,388,608	8	1,024
9	256	511	4,096	9,437,184	4	1,024
10	512	1,023	2,048	10,485,760	2	1,024
11	1,024	2,047	1,024	11,534,336	1	1,024
12	2,048	4,095	512	12,582,912	0	512
13	4,096	8,191	256	13,631,488	0	256
14	8,192	16,383	128	14,680,064	0	128
15	16,384	32,767	64	15,728,640	0	64
16	32,768	65,535	32	16,777,216	0	32
17	65,536	131,071	16	17,825,792	0	16
18	131,072	262,143	8	18,874,368	0	8
19	262,144	524,287	4	19,922,944	0	4
20	524,288	1,048,575	2	20,971,520	0	2
21	1,048,576	2,097,151	1	22,020,096	0	1
...	...	...	1	...	0	1

(4) The Block Diamond hash algorithm and determination rules

```
function hash17diamond( buffer ){
    // console.log(str.length)
    if (buffer.length !== 32){
        throw new Error("buffer must be hash256")
    }
    let stuff = '0WTYUIAHXVMEKBSZN'
    , total = 16
    , hhlen = stuff.length
    let diamond = []
    , fv = 0
    for(let step=0;step<total;step++)
    {
        let i = step * 2
        , n1 = buffer[i]
        , n2 = buffer[i+1]
        fv = (fv + n1 + n2) % hhlen
        diamond.push( stuff.charAt(fv) )
    }
    return diamond.join('')
}

function checkDiamond(stuff) {
    let chars = '0WTYUIAHXVMEKBSZN'
    if(stuff.length == 16 && stuff.startsWith('0000000000')){
        var sarys = stuff.substr(10).split('')
        , first = true
        // console.log(sarys)
        while(true){
            var l = sarys.shift()
            , idx = chars.indexOf(l)
            if(!l){
                return first ? false : true
            }
            if(idx==-1){
                return false
            }else if(idx==0){
                if(first){
                    continue
                }else{
                    return false
                }
            }else{
                first = false
            }
        }
    }else{
        return false
    }
}
```

References

[1] Adam Back, "Hashcash - A Denial of Service Counter-Measure", http://www.hashcash.org/papers/hashcash.pdf, 2002.

[2] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", https://bitcoin.org/bitcoin.pdf, 2008.

[3] Ludwig von Mises, "Theory of Money and Credit", https://mises.org/sites/default/files/The%20Theory%20of%20Money%20and%20Credit_3.pdf, 1912.

[4] Friedrich August von Hayek, "Individualism and Economic Order", http://www.library.fa.ru/files/Hayek-Individualism.pdf, 1948.

[5] J. Huerta de Soto, "Money, Bank Credit and Economic Cycles", https://mises.org/sites/default/files/Money_Bank_Credit_and_Economic_Cycles_De%20Soto.pdf, 1997.

[6] Joseph Poon, Thaddeus Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments", https://lightning.network/lightning-network-paper.pdf, 2016.