YELLOW PAPER OF
FOREST PUBLIC BLOCKCHAIN

Table of Contents

1. Overview of Projects

1.1. Review of Existing Blockchain Projects

1.1.1. MimbleWimble remains to be improved compared with Bitcoin

The Downside of Bitcoin

In Bitcoin, every account corresponds to a wallet address. A full node needs to store UTXO (unspent transaction output) data signed under all addresses, thereby endorsing UTXO under each address. By the end of 2018, a full node had needed to download about 200GB of complete historical ledger data, in order to check all historical transaction data of Bitcoin, which greatly raised the threshold to become a full node of Bitcoin. And lowering the threshold of a full node is very crucial to guarantee the decentralization of Bitcoin. What is more, the size of historical ledger will grow over time. In the future, it will be difficult for ordinary PCs to support the operation of a full node of Bitcoin. Then is there a way to dispense with the download of all historical ledger data by a full node, yet still guarantee the legitimacy of UTXO set and the security of blockchain? Apart from to the over-large size of historical ledger data, the anonymity and privacy of Bitcoin are also worse than expected. In accounting, UTXO model is used to record the transfer of UTXO. Since the account address and transfer amount are open, it is possible to analyze the identity corresponding to the address, according to the transfer history, using complex data analysis technology. The privacy of user and the anonymity of transaction are threatened. How to ensure that the transaction is legal and trustworthy without recording the transaction address and amount?

MimbleWimble is a powerful technology that can solve the above problems with Bitcoin. A user called Tom elvis Jedusor (pseudonym) published a technical paper on MimbleWimble on #Bitcoin-wizards IRC channel on July 2016. This paper proposed an optimization method to encrypt Bitcoin transaction data with elliptic curve cryptography, which can enhance the privacy of transaction, while saving the storage space of blockchain ledger.

1.1.2. Brief Introduction to Grin and Beam

MimbleWimble was created to improve the scalability and privacy of Bitcoin. But from political and technical perspectives, MimbleWimble cannot be integrated into the current Bitcoin system, for it involves a large number of modifications and attribute transaction-offs. For this reason, two teams have begun to try to launch a stand-alone MimbleWimble blockchain system. A developer with the pseudonym of Ignotus Peverell proposed an open source project called Grin. Grin took a similar approach to Bitcoin and was completely funded by community donations. There was no ICO, pre-mining or mining tax. Another open source project was called Beam, which took a similar approach to Zcash, established a formal company and set up mining tax for the project team to fund development and reward the founders. Beam was led by an Israeli entrepreneur named Alexander Zaidelson.

Grin		Beam
Block Interval	1 min	1 min
Inflation Rate	60 Grin coins will be generated by each block	263 million coins in total
Mining Tax	N/A	20% of the mining income in the first five years

Consensus Algorithm	PoW:	Cuckoo
Cycle	PoW: Cuckoo Cycle	PoW: Equihash
Language	Rust	C++

A Comparison between Grin and Beam

The monetary policy of the Grin project is quite different from most blockchain projects in that there is no upper limit for the total number of Grin coins. Grin coins are released continuously at a rate of 60 Grin coins per block, 1 block per minute. This makes Grin a digital currency with persistent inflation. In the above two projects of MimbleWimble, both of them use a network routing proposal called Dandelion. Nodes send transactions through several hops, aggregate transactions randomly after receiving them, send them to miners and then package into blocks. This makes it more difficult for monitoring nodes in the network to figure out how the transactions happen.

1.1.3. DAG ledger technology and TrustNote public blockchain

The ledger data structure used by Bitcoin is blockchain-based. Each block records transactions that occur over a period of time and each block references the hash value of the last block, thus forming a blockchain. The blockchain technology represented by Bitcoin has been developed for 10 years. Although it has gradually become mature, the problem of insufficient concurrent processing capacity has been exposed, too. For example, the theoretical concurrent processing capacity of Bitcoin network is only 7TPS, the concurrent processing capacity of Grin network is 10TPS, and the concurrent processing capacity of Beam network is 17TPS. Such low concurrent processing capacities are far from satisfying the needs of global users. For example, the daily processing capacity of VISA is 2,000 TPS and the average daily peak is 4,000 TPS. It is obvious that the concurrent processing capacity of blockchain networks still seriously lags behind VISA.

Directed acyclic graph (DAG) ledger technology can solve the problem of insufficient concurrent processing capacity of a distributed ledger system, using a schematic ledger structure and asynchronous parallel accounting. First of all, data are recorded using a DAG structure so that each node can record data to different forks of the DAG in parallel. Secondly, DAG allows point-to-point cross-check in the wallet client. This kind of check is parallel. Suppose that there are 10,000 transactions occurring on the earth at the same time. All of these 10,000 transactions can be cross-checked according to the transaction relationship. If this number is bifurcating, different transactions will be recorded on different forks of different wallets on the earth. Thirdly, the accounting unit of DAG becomes more granulated. The accounting unit is not block, but transaction. Once a transaction occurs, it will be written in immediately, at a higher speed than the conventional system. In a conventional system, it needs to wait for writing with other transactions, until the whole block is finished. Fourthly, DAG dispenses with “double spend” detection in the process of accounting. All of the transactions will be recorded. The “double spend” detection is done after the trunk of DAG is identified. On the other

hand, DAG also has another characteristic. That is, it relies on the cross-check between neighboring nodes, the greater number of wallets in the entire network, the greater number of transactions, the greater variation amount will be supported. So the concurrent processing capacity of DAG ledger technology will increase with the increase of the number of nodes.

TrustNote public blockchain is a DAG public blockchain positioned in the token economy of the IOT era. Using high-speed asynchronous DAG ledger technology, this project implements a two-layer consensus system of TrustME+DAG. The TrustNote project has made great modifications and innovations in the implementation of DAG database, DAG consensus algorithm and smart contracts. Compared with Bitcoin, the transaction confirmation speed of TrustNote is very high. A transaction can be verified in 2~3 seconds. The transaction can be stable in 1~2 minutes. After the transaction is stable, it can be immediately used for the next transfer. The concurrent processing capacity of TrustNote is greater than 2000TPS and continues to grow with the expansion of network.

In conclusion, the token policy of TrustNote project is 1 billion in total, 50% for pre-mining and the remaining 50% to be mined within 30 years. The mining reward will decrease year by year. 10% of mining tax should be implemented for the mining reward, which will be reserved by the team to cover development expenditure.

1.1.4. Brief Introduction to Forest

Forest public blockchain is an open source blockchain platform specially developed for high-speed private payment. This platform integrates a large number of latest technologies in the field of blockchain technically. To be specific, it includes the abovementioned MimbleWimble privacy protection technology, high-speed asynchronous ledger technology based on DAG, anti-ASIC mining work algorithm based on Equihash and lightning payment network technology. The application scenarios of Forest public blockchain mainly include cross-border payment, cross-border e-commerce, offline retail and offline digital currency ATM, etc. with high privacy requirements.

In the design of the technical architecture of Forest public blockchain, we will adhere to the principles of concise, efficient and available to a vast majority of users. And we firmly believe that only a concise technical architecture can truly realize high throughput and high-speed transfer.

In this paper, we will briefly introduce the technical architecture, module composition and implementation of Forest public blockchain, so as to make readers understand the implementation scheme in the development of Forest public blockchain.

2. Design of the Technical Architecture of Forest

2.1. Key Features of Forest Public Blockchain

The design of the technical architecture of Forest public blockchain focuses on the realization of high-speed private transfer. To sum up, it has the following key features:

1. On the function of private transaction, it implements the latest MimbleWimble protocol;
2. The ledger database implements high-speed asynchronous parallel accounting using DAG;
3. It implements a PoW consensus algorithm under the architecture of DAG and realizes a perfect combination of PoW consensus and DAG ledger;
4. Based on MimbleWimble privacy protocol, it constructed a script function similar to Bitcoin, on which basis it implements a lightning network for private transactions.

The token policy of Forest public blockchain is:

1. The total number of tokens is constant. The upper limit is 2.1 billion;
2. . Each token can be correct to the eighth decimal place.
3. Forest tokens will be gradually mined out within 60 years, and the mining reward will decrease year by year;
4. There is no pre-mining or ICO for Forest tokens. All tokens must be obtained through mining;
5. Forest project team reserves 10% of the mining income, as the project fund, in order to cover project expenditure.

2.2. Systematic Architecture of Forest

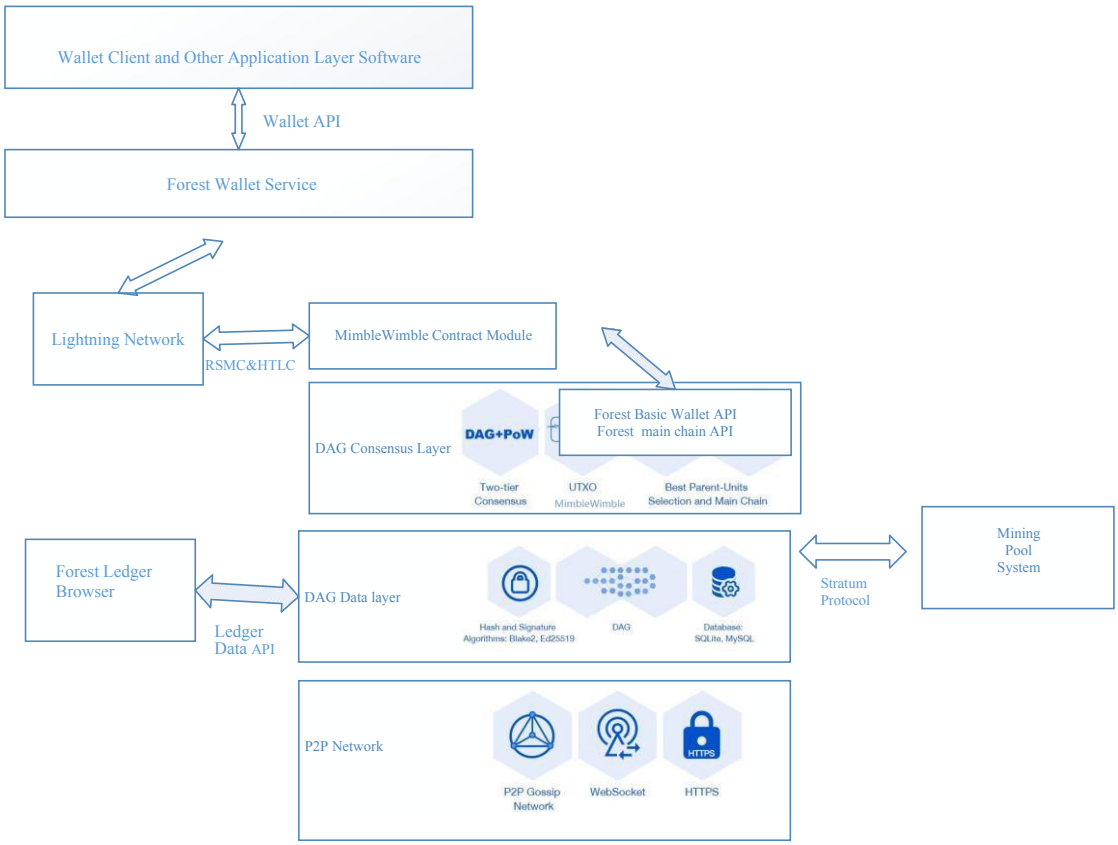
Forest public blockchain system adopts a layered modular architecture. The whole system is divided into network layer, data layer, consensus layer, protocol layer and application layer, etc. Among them:

• The network layer includes P2P network communication module and IP address hiding module, etc.;
• The data layer includes encryption module, MimbleWimble module, in-memory DAG data engine and DAG database module, etc. ;
• The consensus layer includes reference and confirmation consensus, PoW and GhostPlus trunk consensus and other modules;
• The protocol layer includes MimbleWimble conditional contract, master node API, wallet API, Stratum API, lightning network and other modules;
• The application layer includes wallet App, DAG browser, mining pool and mining program, etc.

In view of the tight deadline, given the open source culture in the field of blockchain, in the development of Forest public blockchain, we will make full use of the achievements of current open source blockchain projects and use as many software source codes that have already been verified as possible. In doing so, a lot of coding time can be saved and the developers in the open source community can be fully integrated and gathered, to lay a foundation for spontaneous contribution of the community and code maintenance in the later stage of project.

2.2.1. Architecture diagram of Forest

The architecture diagram of Forest is as follows:



Architecture Diagram of Forest

2.2.2. Description of main modules

1. P2P network communication module

The blockchain system is in the first place a distributed decentralized communication network. Therefore, for all blockchains, P2P network communication module is an underlying key module. Forest public blockchain is no exception. To implement a high-performing P2P network communication module is one of the core tasks of this project. After each node of Forest begins to run, it is necessary to start the P2P network communication module, communicate by the peer list and add yourself to the blockchain network. After nodes join in, they will first perform a handshake protocol, verify version compatibility and other problems and then maintain a long connection through PING/PONG heartbeat protocol. The interactive data of P2P module can be roughly divided into four types: peer-to-peer interactive messages, broadcasting of transaction data, loading or updating of ledger data, consensus and mining data.

2. Transaction pool management module

- On the implementation of Forest transaction pool, we plan to improve and perfect the transaction pool management module of Grin and integrate the transaction pool and the confirmation and reference module of DAG ledger as a whole.

3. DAG reference and confirmation consensus module

Forest adopts DAG ledger technology, so we specifically design a consensus system that is suitable for DAG ledgers. In this consensus system, the whole consensus process is divided into two stages: reference and confirmation consensus stage and work consensus stage (also known as trunk consensus stage). Reference and confirmation

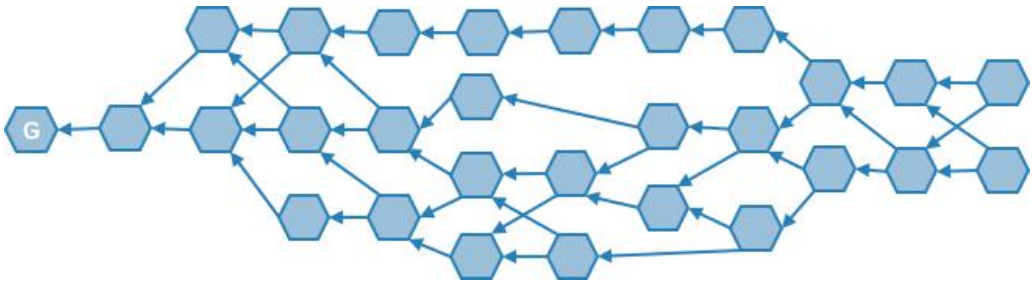
consensus is mainly used to account asynchronously at a high speed after receiving transaction data, while work consensus is mainly used to identify the trunk of DAG ledger while synchronizing ledger, sequence the whole DAG, check the legitimacy of transaction and avoid double spend. In this section, we will chiefly introduce the reference and confirmation consensus module.

The definition of directed acyclic graph

If there doesn't exist a path p=(e1,e2,…) that starts from v and ends in v, ei∈E for any vertex v∈V in a directed graph G=(V,E), G is called a DAG (directed acyclic graph).

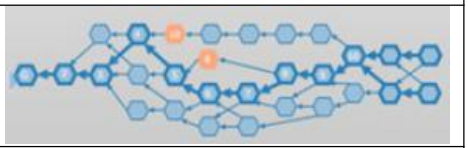
The structure of Forest ledger chain

The structure of Forest ledger chain is shown in the following figure. Every transaction is an accounting unit. Any unit can select one or several units as its parent unit. Each unit verifies and confirms its parent unit and incorporates all hashes that reference its parent unit into its own unit data, thus forming a hash chain. All units and reference and confirmation relationships constitute a fast-growing DAG. With this DAG ledger, if any node tries to modify the unit data, it will find that many subsequent units must be modified and the number will cumulate. There is no way to start. This DAG ledger ensures that Forest public blockchain is traceable, tamper-proof and irreversible.



Two-step consensus mechanism

The DAG of Forest public blockchain adopts a two-step consensus mechanism, which is not only able to achieve high concurrency, but also able to guarantee the security of final consensus.

Step 1: Consensus of DAG Graph	Step 2: Consensus of DAG Main Chain
	
Every transaction is a unit and verifies one or more transactions before it. All transactions are pushed forward rapidly, thus forming an ever-growing hash chain, ensuring that unit and unit links are tamper-proof and finally reaching a consensus of DAG graph.	Use PoW proof to generate PoW units and use GhostPlust algorithm to determine the main chain. The PoW units push forward the main chain and arrange a total order for all transactions according to the main chain and finally reach a consensus.

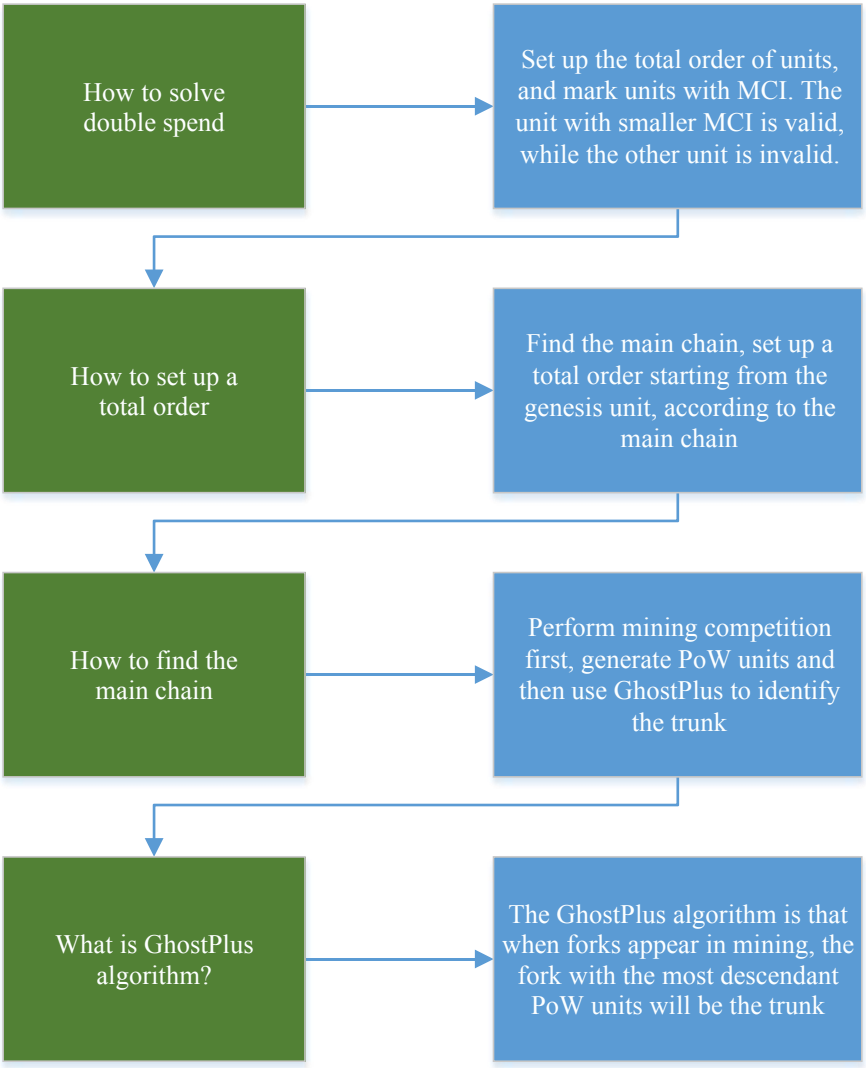
In the first step of consensus, Forest only roughly verifies a new unit, and doesn't judge on the double spend. The purpose for doing this is to guarantee concurrency and enhance the difficulty in tampering with historical units quickly, leaving no chances to tamper. In this case, this unit is unstable and needs to be confirmed in the second step of consensus. To be specific, in the first step, we mainly reach a consensus on unit information and link information. As these two types of information are not allowed to be tampered with in the subsequent ledger synchronization, the nodes will finally reach a network-wide consensus.

In the second step of consensus, according to the overall conditions of multiple units that are newly added, including the newly added PoW units, Forest will decide whether some units are stable. Stable units are in a confirmed state, that is, they are either valid or invalid. Since different nodes have a determined and consistent mining stability promotion algorithm, and conditions the algorithm relies on (the PoW mining trunk) are also determined and consistent, all nodes that honestly observe this consensus protocol will also finally reach a network-wide consensus.

By adopting this two-step consensus scheme, we postpone the execution of trunk confirmation algorithm with high computational difficulty, but solve the problem of transaction congestion in the existing blockchain, which is similar to the practice of changing some complex businesses from strong consistency to final consistency in order to improve concurrency in Internet business.

How to detect double spend

Forest public blockchain detects double spend using the following procedures:



How to detect double spend

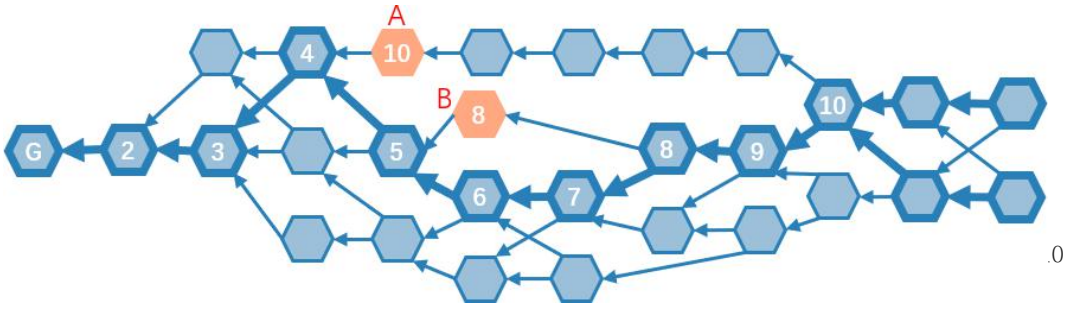
- Identify the PoW trunk of DAG using GhostPlus

Forest public blockchain identifies the trunk of DAG formed by PoW units using GhostPlus. GhostPlus can be described as the following rules:

"Starting from the genesis unit, select a unit on the next trunk from descendant PoW units of the last PoW unit of the trunk iteratively, and select a PoW unit with the most descendant PoW units as the trunk unit."

In Forest's DAG ledger, in most cases, PoW units are not adjacent to one another. There will be one or more ordinary transaction units in between. When selecting a trunk, GhostPlus will ignore ordinary transaction units, but still count the number of PoW units. This is because in DAG ledgers, only PoW units are generated by consuming a large amount of computation. After ignoring ordinary units, we will get a PoW-DAG similar to Block Tree in Ghost.

- Example of double spend detection:



To take an example, as shown in the above figure, units with bold frames are main chain units. While two orange units, A and B, are double spend units.
According to this principle, the MCI of Unit A is the MCI of the first main chain unit that contains it. The MCI of the first main chain is 10, so the MCI o Unit A is 10. Similarly, the MCI of Unit B is 8. From this, we can identify that Unit B is valid, while Unit A is invalid. To sum up the double spend detection principle from another perspective, units referenced and confirmed by mining transactions early take precedence over units referenced by mining transactions late. Being referenced and confirmed by mining transactions is an act of "credit enhancement" to a transaction unit.

4. In-memory DAG data engine

The purpose for Forest for using DAG is to tap the high performance potential of DAG ledger technology, while the management of in-memory DAG data structure is one of the key performance-sensitive modules. The management of in-memory DAG data structure is particularly important when the blockchain network has a high TPS (greater than 1,000). In the previous implementation and performance optimization of DAG chains, the Forest development team had tested and verified that the in-memory DAG data engine can significantly affect the TPS performance of public blockchains. For example, if the reading and writing of DAG data in memory were completely in series, then when a large number of transaction requests were received, Forest nodes needed to put all requests into a queue and cannot modify DAG data structure in parallel using multiple threads. If we are able to implement a new in-memory DAG engine that allows operating non-conflicting forks on multiple DAGs simultaneously, then we can improve the concurrency of DAGs. During development, the Forest public blockchain will implement a high-performing DAG data engine independently using the Rust language and lay a good foundation for Forest to reach a concurrency of 10000TPS.

5. DAG database

In order to guarantee high-speed storage of DAG ledgers, Forest also needs to implement and optimize a high-performing schematic database in order to record data on disks. But this feature is not well implemented in the current DAG chain project. In the current DAG public blockchain, the implementation of database includes the following solutions: the first solution is to store DAG data using SQL database. The second solution is to store DAG data using key-value NOSQL database. This solution enables the addition, deletion, modification, checking and query of schematic data structures using key-value database. Forest public blockchain will implement a database module using the second solution. Considering that both Grin and Monero use LMDB to store blockchain data, LMDB has many remarkable features. It is a database based on binary tree. The whole database is memory-mapped. In all data acquisitions, all of the data are directly returned from the mapped memory, so no malloc or memcpy occurs during data acquisitions. It doesn't require a page cache layer, so it is very efficient and memory-saving. Meanwhile, LMDB is completely in line with ACID in semantics (atomicity, consistency, isolation and durability). When the mapped memory is read-only, the integrity of database won't be spoiled by the stray pointer of application. So the Forest public blockchain will store block data using LMDB at the bottom layer. On the other hand, to guarantee efficient reading and writing of DAG data on LMDB, the Forest project will also implement a high-performing graph data management module on LMDB to ensure that the DAG data in memory can be efficiently stored in LMDB.

6. The PoW consensus module of trunk of DAG

When introducing the reference consensus module, we have already introduced that

DAG consensus system is divided into two parts, the "best parent unit" part that identifies the reference relationship between transactions and the "trunk stability part" that identifies the trunk of DAG. In public blockchains such as IOTA and Byteball, all of the trunk identification algorithms are realized by centralization or semi-centralization. Thus, these public blockchains lose the characteristic of decentralization trust. As a result, the credibility of digital currency generated by these public blockchains will decline. In the Forest public blockchain project, we will insist on using PoW algorithm as the most essential consensus algorithm and determining the mining of Forest tokens using PoW competitive mining. Meanwhile, the Coinbase transaction that mines out tokens will also become a GhostPlus transaction that identifies the trunk of DAG. We use GhostPlus to choose a fork with the most PoW transactions, thereby identifying a trunk advancing along the time shaft. The use of GhostPlus in Forest makes us fully return to Nakamoto's PoW algorithm and identify the winner per unit time, using the PoW competition algorithm. The winner will have the power to issue a Coinbase transaction as the candidate "trunk transaction". Once a Coinbase transaction is identified as a "trunk transaction," this Coinbase transaction will take effect and the winner will be rewarded a certain number of Forest tokens.

7. Token management module

With P2P network layer, ledger data layer and consensus layer, Forest will have the basic functions of distributed ledgers. But in order to make Forest into a digital currency that supports high-performing private transaction, we also need to determine a token specification for Forest digital currency, specify a way to represent the input and output of money and a way to represent a Coinbase transaction (a transaction in which new tokens are mined). At the same time, we need to specify the total number of tokens and the time interval and mining speed for generating all tokens.

Generation rate of mining transactions

The generation rate of mining transactions in Forest is theoretically determined, which is approximate to 1 mining transaction every 5 seconds (provisional).The specific rate shall be adjusted dynamically according to the network-wide computing power and mining difficulty.

Adjustment of mining reward

The mining reward of Forest should be adjusted once a year (provisional). It should be controlled in such a way that the Forest tokens will be mined out after 60 years.

8. The overall development of DAG main chain

During the development of Forest, in addition to the independent core modules stated above, we also need to develop a lot of auxiliary modules. These core modules and auxiliary modules must be constantly integrated and deployed as the test chain network of Forest at each stage. Also, the development of the above core modules must be done on an active and dynamic Forest test chain. Only in this way can the functions and performance required be truly developed and verified.

9. Mining program and fork upgrade tool

Similar to Bitcoin, Grin and other public blockchains, Forest will also implement an independent mining program. But considering the need to resist ASIC mining and increase the sales of a certain type of mining machine, during the development of Forest mining program, we will isolate the work algorithm during the mining process as a separate and easy-to-upgrade module. At the same time, to facilitate users' subsequent fork upgrade, as well as resistance to ASIC mining, currently Forest selects equihash as the mining algorithm.

10. Mining pool service

In the ecology of public blockchain, mining pool service can lower the mining difficulty of miners, enhance the engagement in mining and facilitate the decentralization of

public blockchain. Therefore, Forest also needs to develop a mining pool software package, as a sub-project of the Forest project. It is particularly noteworthy that different work algorithms have different poolability. Poolability refers to the ability to calculate the contribution of each node fairly, when the calculation of an algorithm is done by multiple nodes in a distributed manner. For example, some people question the "poolabilty" of Cuckoo Cycle algorithm currently used by Grin. In contrast, the poolability of SHA256D algorithm used by Bitcoin is much better, which is guaranteed by the mathematical characteristics of sha256 function. Hence, in the subsequent development of work algorithm, we will pay sufficient attention to the "poolability" and anti-ASIC mining ability of algorithm.

11. Command line wallet and wallet API

A public blockchain cannot be used by users unless it provides an interactive interface. Therefore, our Forest project will provide a wallet module as the user interface. This wallet module offers a set of command lines and a set of APIs to ordinary users and software programs respectively. First of all, the Forest wallet is implemented in the same way as Grin wallet. Butthe implementation of Grin wallet doesn't fully support BIP32, BIP39, BIP44 and other specifications, which makes it inconvenient for users to generate private keys and back up. If users operate improperly, security issues will be triggered. Later, the Forest project will improve and upgrade this defect, to make sure the users can enjoy sufficient convenience and security when generating and managing private keys. At the same time, if these specifications are supported, Forest can also well support a hardware wallet of digital currency.

12. Forest ledger data browser

In the Forest public blockchain project, since every transaction is encrypted with MimbleWimble, the ability to check, monitor, and analyze data on a browser will be greatly weakened. The ledger browser of the Forest project is mainly used to check the status and statistical data of network-wide mining, the display of the topology structure of DAG ledgers, the display of transaction delay, transaction fee, concurrency and other statistical data.

13. Conditional payment contract module

Forest is a public blockchain which protects privacy and encrypt transfer and transaction information using MimbleWimble. Therefore, Forest doesn't support the transaction scripts of Bitcoin. However, transaction scripts are required by many functions, such as multi-signature, cross-chain atomic swap, locked position transaction and lightning network. So how to implement these functions on the basis of MimbleWimble is a problem to solve. Our Forest project will use Andrew Poelstra's approach and implement the above functions using elliptic curve cryptography and Pedersen Commitment. The Forest project will encapsulate a Contract module and offer APIs for the upper module to call, so as to realize the four functions stated above.

14. Lightning payment module

The Forest public blockchain adopts MimbleWimble. The sender and the receiver need to interact in real time when a transaction is created, so the creation of a transaction in itself is an instant message based on P2P communication. This instant message is similar to the way a payment channel is implemented in the lightning network. On the other hand, the Forest public blockchain adopts high-performing asynchronous DAG ledger technology and greatly improve TPS. To achieve high concurrent processing of DAG data, all nodes in the Forest need to forward the above P2P instant message as much as possible quickly. This is very similar to the routing and forwarding between nodes in the lightning network. To sum up the above two points, it seems to be a better choice to implement a built-in lightning payment module in the Forest public blockchain. Of course, the lightning payment module of Forest is still a relatively

independent module. Compared with the lightning network of Bitcoin, the lightning payment of Forest can be implemented by reusing more modules of Forest. This won't prevent the independent implementation of the lightning network of Forest. If someone just wants to implement the lightning network of Forest, rather than implement the main chain, the technical architecture of Forest will support, too.

15. Wallet service module

To better support mobile wallet and Web wallet, the Bitcoin project has already implemented a multi-signature HD wallet technology. For details, see https://en.Bitcoin.it/wiki/Deterministic_wallet. This technology enables the cloud service provider to implement a Bitcoin wallet cloud service and makes it easier for users' mobile App and Web pages to achieve the wallet function. Under this architecture, users can achieve the wallet function without installing full nodes. At the same time, they can control the private keys of wallets, manage and back up the private keys on their own. The cloud service provider is unable to threaten the security of users' wallets. This technology is very mature on Bitcoin, but has not yet been implemented on Grin. Since the Forest public blockchain is built on Grin, in the future, Forest must develop and implement this function, to make it easier to develop wallet mobile App and Web App, thereby greatly increasing the usability of Forest and facilitating the rapid popularization of Forest.

3. List of Upper Software Modules of Forest

1. Main program of Forest

The core module of the Forest project, which includes the implementation of main chain and a built-in wallet interface.

2. Automatic deployment scripts of main chain

These automatic scripts can quickly call the main program of Forest using Docker and set up a test network for Forest public blockchain.

3. Mining machine program

After determining the work algorithm (Equihash or cuckoo cycle), we also need to develop and maintain a mining machine program that has already been used.

4. Mining pool service

To develop or fork an open source mining pool service software.

5. Forest wallet service

As mentioned above, a wallet cloud service can greatly simplify the development of wallet App.

6. Forest wallet App

To officially develop and maintain a standard Forest wallet App. In this way, users can easily use Forest on their mobile phone.

4. Development Environment

1. The desktop development environment for the Forest project is Ubuntu18.04 64bit and Docker and the development language is Rust 1.31. Key modules of the mining program are developed with C/C++, while the wallet App is developed with Object C, Java and HTML 5, etc.
2. The deployment, maintenance and test of the main chain shall be conducted on a cloud host equipped with 64bit Ubuntu18.04. The cloud host service providers are principally Aliyun and AWS.
3. The source codes of the project shall be managed with Github. During internal

development, a non-public source code repository dedicated to internal developers will be used. After each version becomes stable, we will push the source codes to an external code repository.

4. If main chain nodes of Forest need to be built into the ATM, we can compile Forest codes to ARM in a cross-platform manner. But so far, we cannot guarantee that the cross-platform compilation will succeed.

5. Description of Key Technologies

5.1.1. Combination of MimbleWimble transaction and DAG ledgers

According to the technical document of MimbleWimble, MimbleWimble transactions include the following content:

- A set of inputs, references and costs, a set of previous outputs
- A set of new outputs, including a value and a blinding factor (which is merely a new private key), multiplied on the curve and added together as r.G + v.H.
- The range proof shows that v is non-negative.
- Explicit transaction cost
- A signature, calculated using excess value (the sum of all outputs plus costs, minus inputs), by taking the excess value as the private key

In the implementation of the DAG chain of Forest project, each accounting unit stores a transaction data. Each unit contains the following information:

The data to be stored. A storage unit can contain a MimbleWimble transaction as stated above.

The signature of the storage unit.

The hash reference of one or more previous storage units (parent units).

5.1.2. Design and implementation of DAG consensus algorithm

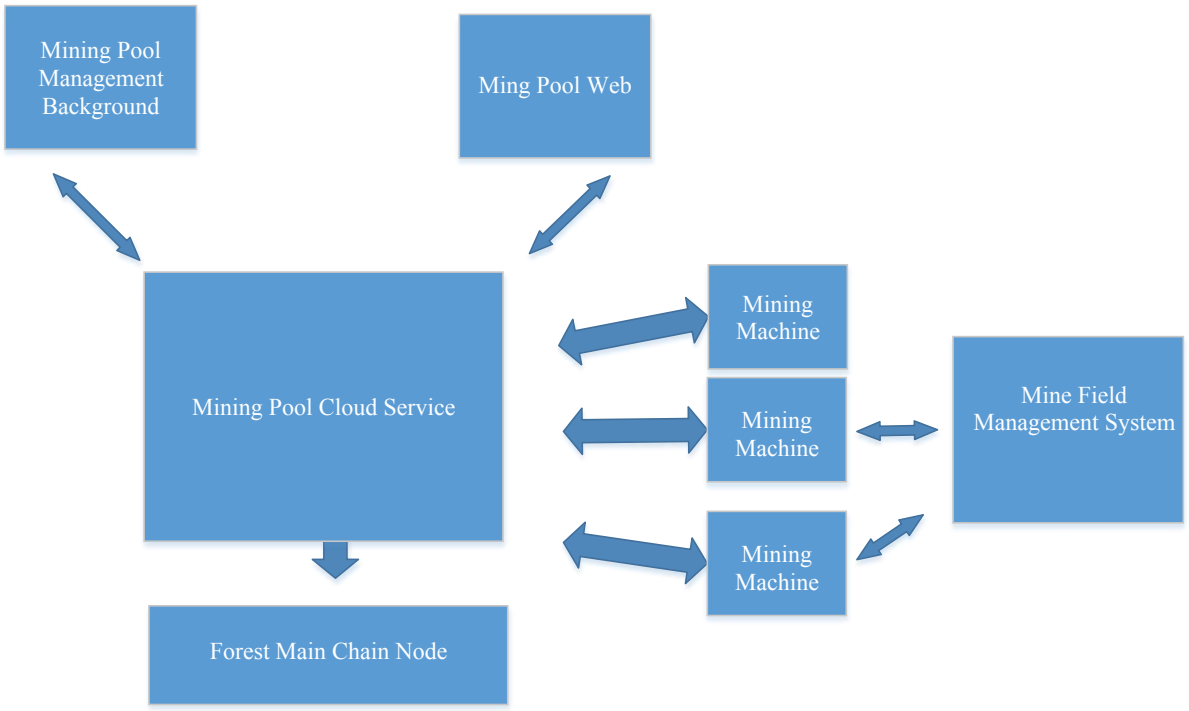
Since Forest public blockchain uses to store data, the Forest consensus algorithm needs to be further upgraded and improved on the basis of the consensus algorithm in DAG chain. The consensus system of DAG chain is divided into two parts. If we analyze with schematic data sorting, there two parts correspond to the side order stage and total order stage during the sorting of schematic data structure respectively. The side order stage doesn't detect the double spend, but keep accounts quickly. While the total order stage arranges a total order for all transactions, to identify illegal double spend transactions. In a nutshell, the key to DAG's arrangement of a total order is to identify a trunk of DAG. Once a trunk is identified, the transaction order of other forks in DAG can also be determined according to the units on the trunk. How to identify a trunk has become a key problem to be solved by every consensus algorithm of DAG public blockchain.

The DAG of the Forest public blockchain can turn into a similar graph to the BlockDAG of the abovementioned Ghost principle after simple conversion. When identifying a trunk, we first ignore ordinary transaction in the DAG. If there are only ordinary transactions between two mining transactions, we first assume that the two mining transactions are directly connected. At this moment, we get the same mining transaction graph as in the Ghost algorithm. We first identify a trunk of this simplified DAG, so as to get a unique trunk path. After that, we identify ordinary transactions between mining transactions on this trunk path as trunk transactions. We finally identify the trunk of the whole DAG, in order to arrange a total order for all transactions in the DAG. This algorithm, which ignores part of transactions first and then uses Ghost to

identify the trunk, is called GhostPlus.

References to consensus algorithms:
Ghost, https://eprint.iacr.org/2013/881.pdf
ByteBall, https://obyte.org/Byteball.pdf
IOTA Coordinator, https://docs.iota.org/docs/the-tangle/0.1/concepts/the-coordinator
Ghost DAG, https://eprint.iacr.org/2018/104.pdf
Conflux, https://arxiv.org/abs/1805.03870
SPECTRE, https://eprint.iacr.org/2016/1159.pdf
TrustME, https://github.com/trustnote/document

5.1.3. Framework of mining system



Framework of the Mining Pool

- Forest node

The mining pool system requires a Forest full node in order to participate in the PoW consensus of Forest blockchain network. The Forest node interacts with the mining pool service module using Stratum protocol and RPC protocol. Only in this way can the Forest node have the right to broadcast the mined blocks to the Forest network.

- Mining pool service

Mining pool service is a back-end cloud service module which interacts with the mining machine program, sends mining tasks to the mining machine and receives the mining results reported by the mining machine. When the mining is very difficult throughout the network, the mining pool service will break down tasks with high difficulty into tasks with low difficulty and send them to mining program. After receiving results reported by the mining machine, it will screen the results with low difficulty to make them satisfy the demands of high difficulty. The mining pool service also has a mining income calculation module, which calculates the contribution degree of each mining machine to the income of a certain block and then calculates the reward for each mining machine, according to its contribution.

- Mining pool Web foreground

The users can use the Web foreground to configure and monitor their own mining machines.

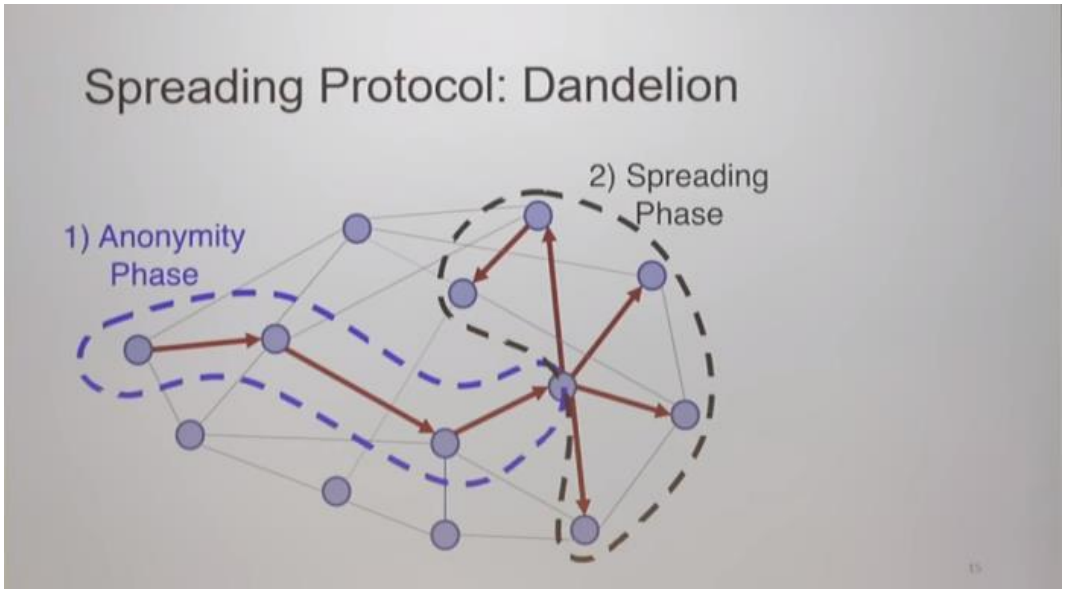
- Mining pool Web management background

The mine pool operators can use the management background interface to manage their mining pools.

- Mining machine program

The mining program is a program which calculates a certain workload by calling CPU, GPU or ASIC.

5.1.4. The hiding of IP address in the P2P protocol



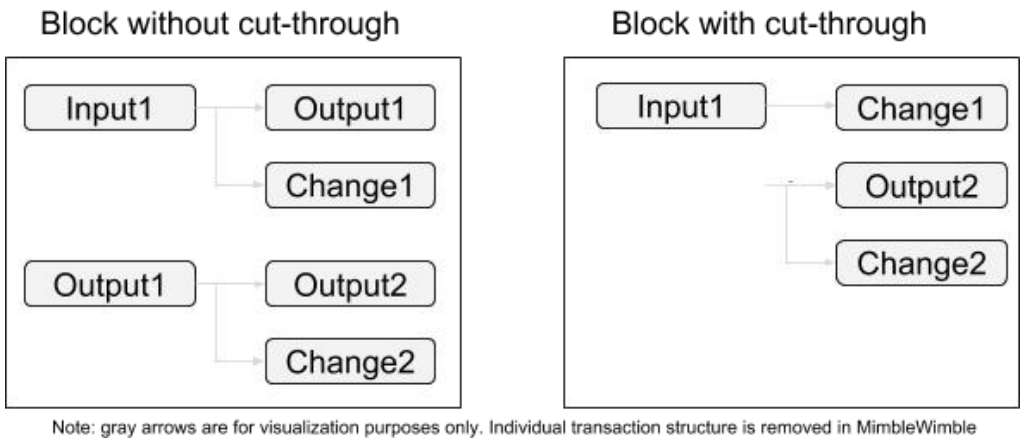
Transaction Data Spreading Protocol

When all nodes in the blockchain communicate with each other, a P2P protocol will be used to transmit transactions and block data. During the P2P communication, the IP addresses of nodes will be exposed. If some nodes in the blockchain monitor the IP addresses of nodes purposefully, associate the transaction data sent by nodes with their IP addresses, they will be able to figure out the identity of owners of some addresses on the blockchain. The Bitcoin community has recognized this problem and designed a technical solution to solve the privacy leakage problem of P2P layer. Tor and Dandelion are two technical solutions this problem, of which Dandelion is specifically designed for

this. For relevant technology, see the link below:
Dandelion++ protocol, https://arxiv.org/pdf/1805.11060.pdf
https://medium.com/@thecryptoconomy/dandelions-and-a-bright-future-for-Bitcoin-privacy-712dbc4b1ec5
At present, two public blockchains that implement MimbleWimble have realized the network routing proposal of Dandelion. In MimbleWimble, nodes send transactions through several hops, aggregate transactions randomly (after receiving them), send them to miners and then package into blocks. This makes it more difficult for monitoring nodes to figure out how the transactions happen. As Forest public blockchain must process transaction data at a high speed and the accounting unit is single transaction, the implementation scheme of Dandelion protocol probably need to be modified, in order to increase the speed of transaction jump.

5.1.5. Description about cut-through in Forest

The MimbleWimble protocol not only hides transactions with Pedersen Commitment, but also prunes transaction information by integrating the same transaction inputs and outputs within a block, which is called cut-through. The following figure is a schematic diagram of cut-through of transaction data within a block:



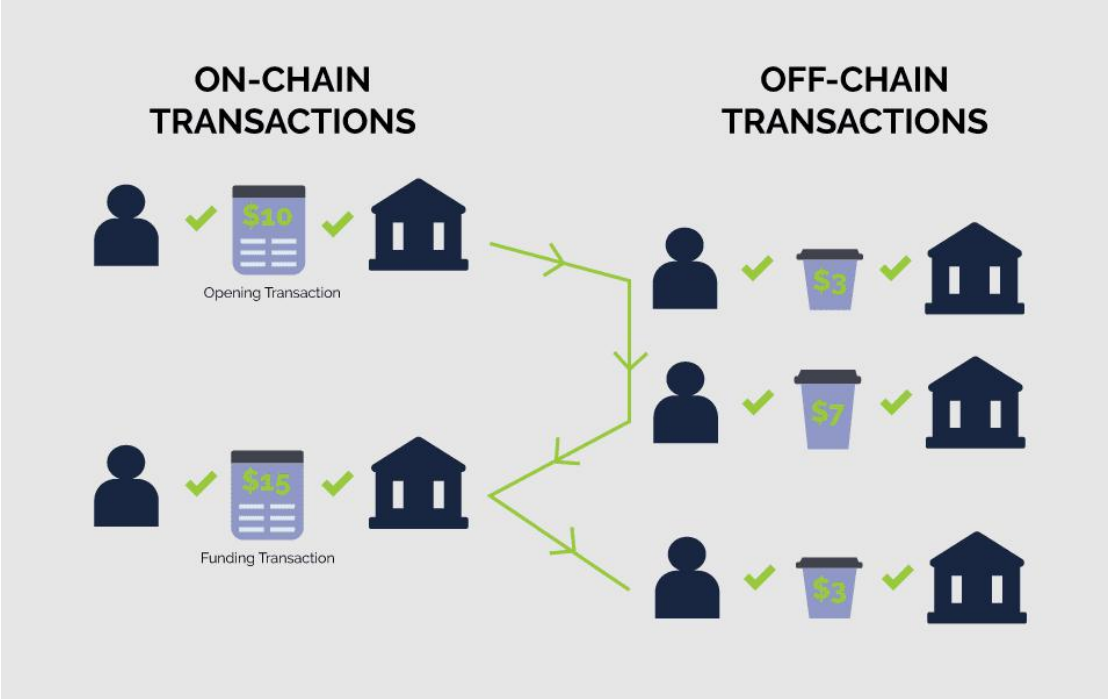
A Simple Schematic of Cut-through

As mentioned above, MimbleWimble merges all transactions within a block into a block-wide transaction and remove the structure and boundary between transactions. If a transaction uses a very “new” (unidentified) input, it is absolutely possible to remove the intermediate outputs, without affecting the chain validation. Out of technical consideration, please note that the transaction cut-through cannot eliminate all intermediate transactions. Every transaction will be saved in the transaction kernel permanently so that the blockchain data can be verified correctly. The transaction kernel proves the actual ownership of transaction inputs and allows using mathematical rules to verify transactions and the whole chain.
In the current technical architecture design of Forest public blockchain, we first use a mature accounting scheme in DAG chain and take a single transaction data as the accounting unit. Therefore, at present, the Forest public blockchain is unable to realize transaction cut-through within a block. It should be noted that this kind of design is reasonable, for wallet node wants to confirm the transaction as soon as possible at this moment, rather than have a cut-through later for the sake of transaction. In DAG quick accounting, to make other nodes confirm the transaction as soon as possible, the wallet is weighing between quick confirmation and cut-through, and the result of weighing is that transaction speed comes first.

5.1.6. Description about lightning network

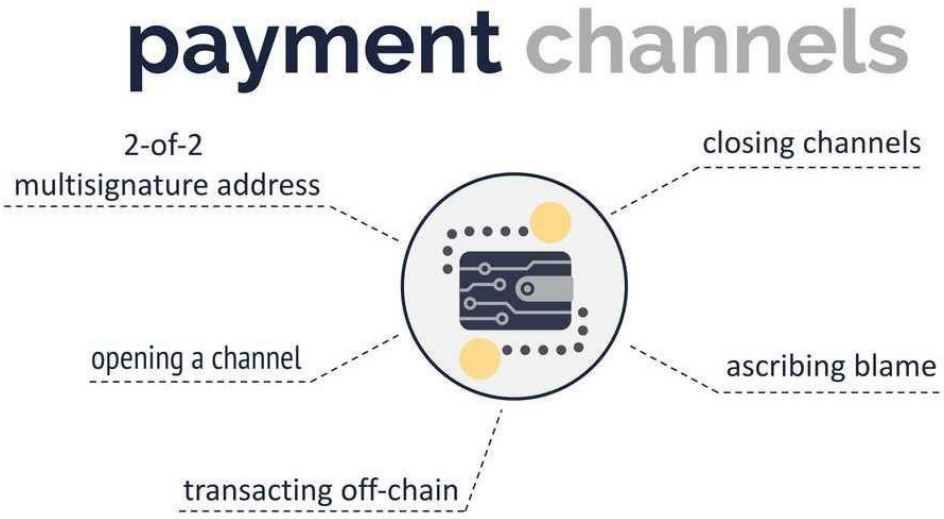
Overview of lightning network

Since the Bitcoin network cannot fully support concurrent processing, the lightning network needs to transfer on-chain transactions to off-chain to complete them.



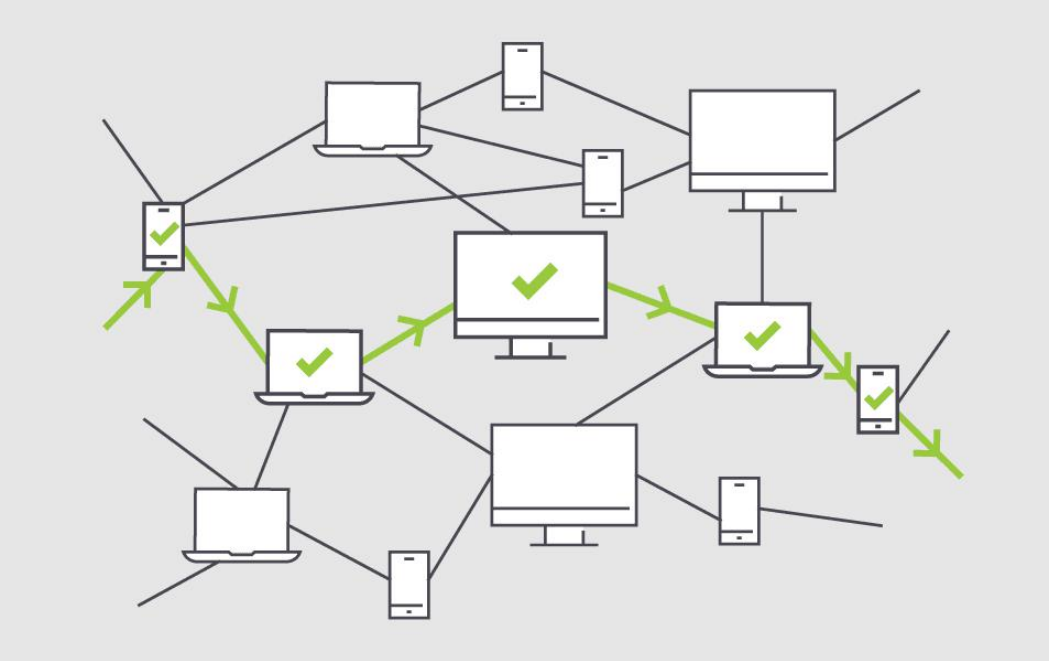
Transfer On-chain Transactions to Off-chain

The off-chain transaction and transfer are realized through payment channels. The key to payment channels is double-signature deposit and dishonesty penalty



A Schematic of Payment Channels

If only a point-to-point payment channel can be established, then the liquidity of lightning network is still insufficient. It is necessary to make the payment channels form a payment network. Thus, the lightning network will adopt hashed timelocks to realize the forwarding and routing of payment.



Lightning Payment Network

To sum up, two keys to lightning network are RSMC and HTLC. As long as we can support multi-signature and hashed timelocks, we will be able to support the lightning network.

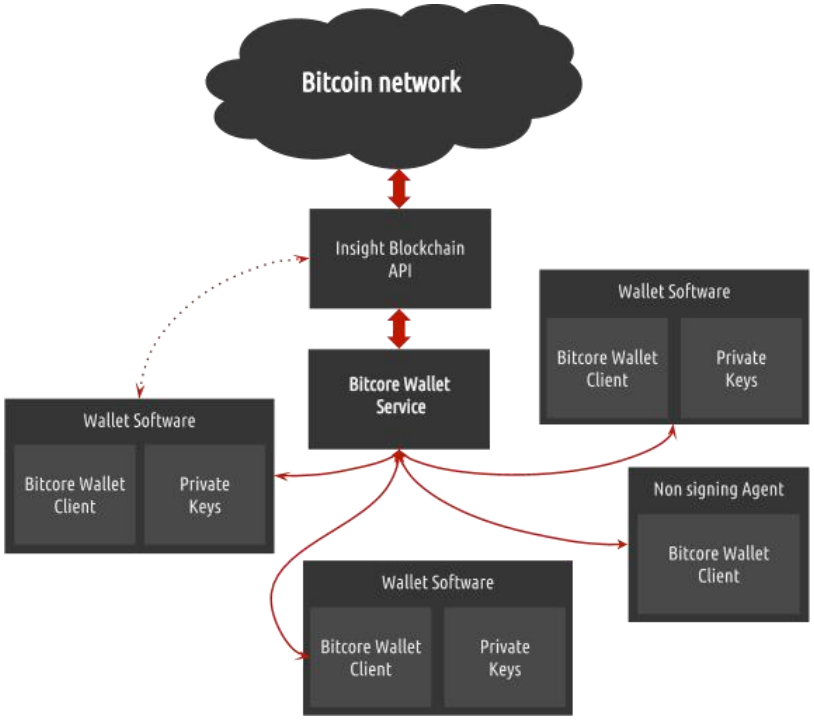
How does MimbleWimble realize contracts?

- Andrew Poelstra discussed how MimbleWimble realizes smart contracts using elliptic curve cryptography and Pedersen Commitment, in the absence of scripts
- For details about this technical solution, see the following link: https://www.reddit.com/r/Bitcoin/comments/ap3qt6/andrew_poelstra_scriptless_scripts_with/.
- The Grin project also has a underlying technical document describing how to realize the basic contract functions on the basis of basic functions of Grin: https://github.com/mimblewimble/grin/blob/master/doc/contracts.md
- The technical plan of the Beam project has already analyzed the technical feasibility of implementing lightning network on the basis of MimbleWimble. See the link below: https://medium.com/beam-mw/mimblewimble-lightning-network-e16 27538aca2
- It can be concluded from the description of materials related to Beam that as long as a MimbleWimble project possesses the above basic contract functions, it has the foundation to implement the lightning network.
- In the subsequent development of Forest public blockchain, we will track and draw lessons from the development progress of Lightning Network Daemon, Rust-lightning, Beam and other projects in order to carry out the development of Forest Lightning.

5.1.7. Wallet service module

Below is a Bitcoin wallet service architecture designed by the Bitpay company.

From the architecture, it can be seen that wallet service module is a cloud service module that falls in between Bitcoin node and Bitcoin wallet application. With a wallet service module, the wallet application software no longer needs to treat a lot of technical details related to the blockchain ledger data or store blockchain ledger data, but achieve the sending and receiving of transactions and manage the private keys of wallet.



An open source project of Bitcoin wallet service:
https://github.com/bitpay/bitcore/tree/master/packages/bitcor e-wallet-service

Forest public blockchain is developed for subsequent large-scale application scenarios. Therefore, we also need to develop and implement a similar wallet service module, to allow developers to develop and implement Forest wallet App quickly.

6. Implementation Path and Milestones

The development of the main chain of Forest will be implemented in the following three milestones:

1. The basic MimbleWimble main chain of Forest

The main work of this milestone is to integrate and improve Grin, determine a monetary policy of Forest, integrate and verify the mining system of Forest, improve the transaction pool module and prepare for the subsequent development of lower layers of DAG ledger. The specific tasks at this stage include:

1.1	Architecture design and development plan
1.2	The development of a MimbleWimble private main chain
1.3	The development of a token mining policy of Forest
1.4	The development of mining software and mining pool software
1.5	The development of a dynamic adjustment algorithm of mining difficulty
1.6	The development of an anti-ASIC mining algorithm
1.7	The development of a half-year fork upgrade system of mining algorithm

2. High-performing DAG main chain of Forest

The main work of this milestone is to implement lower layers of the DAG database, refine the P2P communication protocol, develop a PoW+DAG consensus algorithm, and develop the main chain of DAG. The specific tasks at this stage include:

2.1	The development of a P2P network layer of DAG chain
2.2	The development of an in-memory DAG data engine
2.3	The development of the lower layers of DAG database
2.4	The development of a reference consensus algorithm of DAG
2.5	The development of a PoW consensus algorithm of DAG
2.6	The development of a private main chain of DAG+MW
2.7	DAG ledger browser

3. Forest DAG+ lightning network

The main work of this milestone is to develop and implement a lightning network system that is compatible with MimbleWimble and DAG, based on the DAG chain developed in Milestone 2, and develop Forest wallet service, so as to facilitate the development of Forest wallet App. The specific tasks at this stage include:

3.1	The development of a conditional payment contract module
3.2	The development of a DAG lightning network
3.3	The development of Forest wallet service (only including the token function of public chain)

The above implementation plan of the Forest project in three milestones fully considers the operation needs of Forest community, the development sequence of all kinds of Forest modules, the building process of a Forest development team and the ability enhancement process of the developer community, so as to make the Forest public blockchain available as soon as possible, popularize it in the community and carry out mining as soon as possible. Also we will gradually improve the Forest public blockchain through constant iteration and upgrade.