DDK WIKI

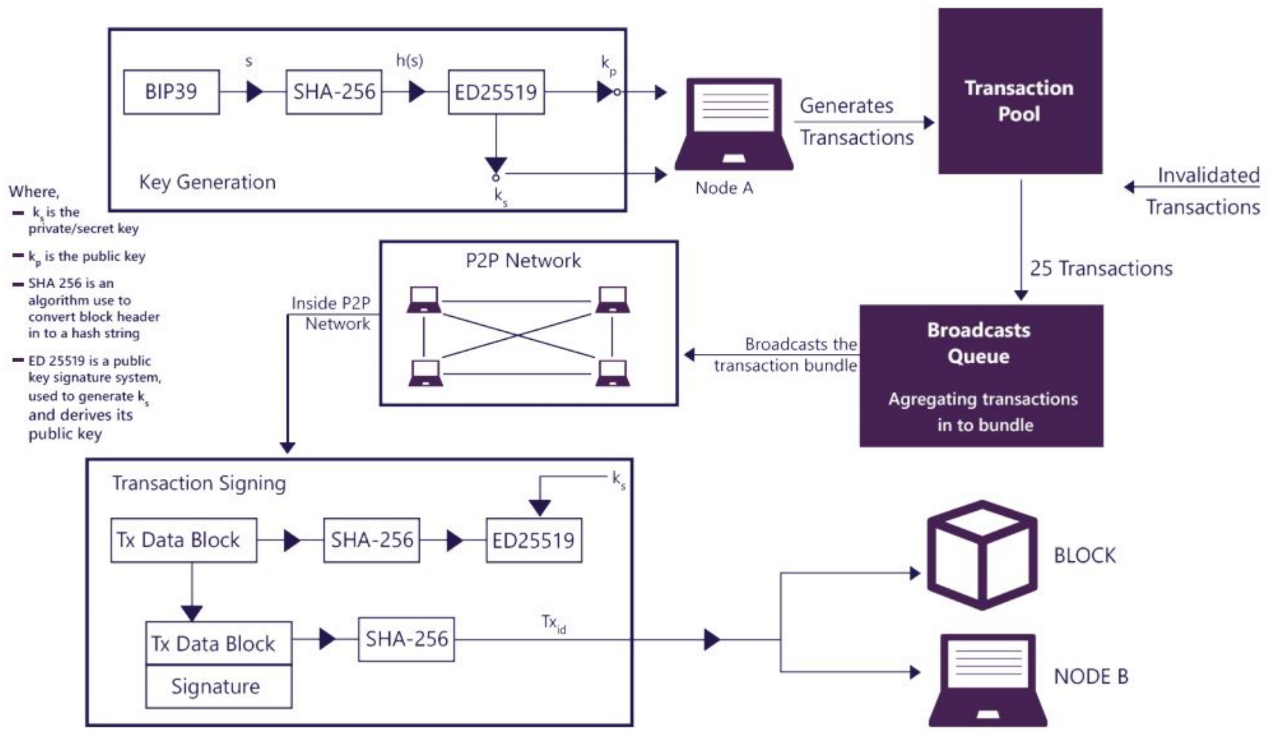

Fig 3.0

Blocks

A blockchain consists of a series of blocks that are linked together in an orderly sequence hence the name 'Blockchain'. A block is a structure of data that consists of,

1. A list of transactions
2. A blockheader – metadata that contains a specific set of information about the block.

The Block Header

The blockheader consists of the following data information:

1. A 64 bit Id of the block
2. A 32 bit integer identifying the version of the block
3. A 32 bit epoch timestamp of when the block was created
4. The 64 bit ID of the previous block
5. A 32 bit integer corresponding to the number of transactions processed in the block
6. A 64 bit integer corresponding to the block height
7. A 64 bit integer corresponding to the total amount of DDK transferred
8. A 64 bit integer corresponding to the total amount of fees associated with the bloc
9. A 32 bit integer corresponding to payload length
10. The 256 bit hash of the payload
11. The 256 bit public key of the delegate who generated the block.

12. The 512 bit signature of the block by the delegate

This is a structure of a normal block-header contains:

- Id of a block
- Timestamp
- Version
- Height of block
- Previous BlockID
- Number of transactions
- Length of payload
- Amount of DDK transferred
- Amount of fee
- Payload hash
- Delegate’s Public Key
- Block Signature

Block Generation

In DDK, block generation happens through the DPoS consensus mechanism. Where every block is generated in 10 seconds on the network, this happens through a delegate that has been given the permission to generate blocks by the election process held between stakeholders and the coin-holders.All active delegates (301 in DDK) are eligible to forge new block. These active delegates are elected based on vote count the users vote to a delegate. Who gets more vote will be on top in the ranking. In case, when vote count same for two delegates, the highest weighted delegate will be on top in the ranking. Block generation in DDK requires 51% of peers to maintain broadhash consensus.

P2P Communication

Peers communication is an essential component within the DDK network. The peering mechanisms provide the required architecture to facilitate network consensus, block propagation and transaction propagation. The DPoS mechanism further helps in the efficiency of communication between peers.

System Headers

System headers are used to identify full nodes and peers on the network. These headers are included in all messages sent between peers.

Block Propagation

In DDK network, blocks are made in a decentralized fashion and must be sent to all nodes found on the network in order to establish consensus. When a block is generated, it is broadcast to peers which broadcast that block to other peers. Without block propagation, the system would grind to a halt and the blockchain would cease to be functional.

Broadcast Queue

Transactions must move from one node to all other nodes in order to be included in blocks. The broadcast queue works by grabbing up to 25 transactions from the transactions pool and aggregating them into a bundle. This bundle is then broadcast to the network on an interval, which is currently specified as every 5 seconds. In addition to broadcasting the object, the bundle is given a relay limit to prevent over broadcasting of data. In the current implementation the relay limit is set as 2, which means every bundle will be broadcasted once from the originating node, and twice more from receiving nodes.

Transaction Pool

Transaction pool serves the DDK network for three purposes,

1) Reducing UXTO, the transaction pool is a solution for preserving unconfirmed transactions that have overflowed into the next block. Each block may only include up to 25 transactions and the transaction pool allows for up to 5,000 transactions to remain queued for the next block(s). The transaction pool could be thought of as a memory pool keeping transactions ready until they are signed into a block.

2) The second usage of the transaction pool is to provide a mechanism for propagating transactions. When a node prepares a transaction bundle, that node draws up to 25 transactions from the pool and performs validation on those transactions. These transactions are then broadcast to other nodes in a bundled JSON object.

3) The final use for the transaction pool is to house transactions with pending signatures. These transactions with pending signatures following the same model as unconfirmed transactions. This way multi-signature transactions that are isolated as incomplete within the transaction pool. Like unconfirmed transactions, these transactions will expire out of the pool based on the lifetime specified when the transaction is first generated.

In order to keep the transaction pool tidy, all transactions are assigned a time to live. This time to live is defined as 10800 seconds, or 1080 blocks.

Security

In order to secure the system, DDK will implement the same hashing mechanism of Edward Digital Signature Algorithm (EdDSA) which is found in LISK systems, rather than using the conventional ECDSA hashing mechanism that is found in legacy crypto-currencies such as Bitcoin.

Key-pair

A key-pair consists of,

- a private key
- a public key

Private key: is a piece of information known only to the owner of the key.

Public key: is derived from the private key and can be used to validate that the private key belongs to the owner, but not provide access to the owners private key.
Elliptic curve cryptography is used to generate cryptographically secure key pairs.

With this private key, the user is able to sign transactions into a transaction object and broadcast that object to the network. The public key is included as part of the transaction which will be used by the receiving nodes for verifying the validity of the signature. This provides an efficient mechanism for securing the network as Ks is known only to the user while Kp is can validate that the signature is valid.

How key-pair is generated

The process used to generate the key pair operates under the following assumptions:

1. When a user creates an account, a BIP39 mnemonics (the passphrase) is generated for the user.
2. This passphrase is hashed using the SHA-256 hash function into a 256 bits string.
3. This hash is subsequently used as a seed in ed25519 to generate the private key ks and derives its public key kp.

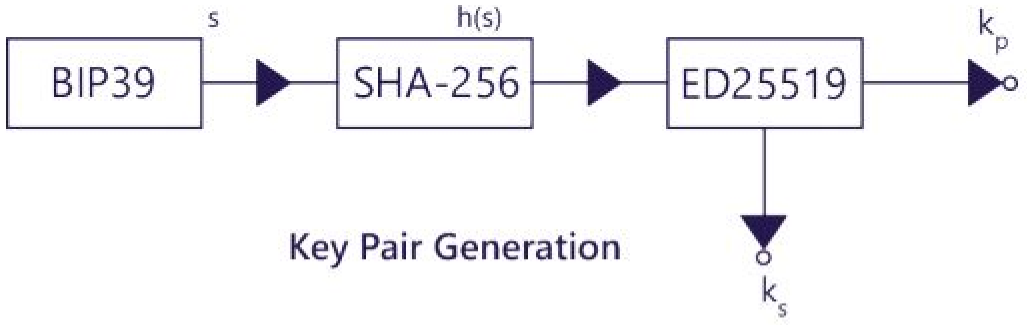

Figure 3.1: Key pair generation

Transactions

Transaction Signing

Every transaction, regardless of the type, must be signed by the sender prior to being accepted by the network. The process of signing the transaction is identical for every transaction. First, a data block representing the transaction must be generated. Each data block contains a specific set of standardized information. Additional information contained in the data block will differ depending on the type of the transaction.

The following fields must be present in all types of transactions:

- A 8 bit integer identifying the type of the transaction
- A 32 bit epoch timestamp of when the transaction has been created
- The 256 bit public key of the issuer of the transaction
- A 64 bit integer representing the amount of DDK to be transferred

The other fields will be added to this schema depending on the transaction type. Once the data block has been generated, it is hashed using the SHA-256 algorithm, and this hash is signed using the key pair of the issuer. If the issuer has enabled a second pass phrase, the first signature is appended at the end of the data block, and the process is repeated, generating a second signature. The same concept applies to multisignature accounts. The transaction-Id is generated from the data block. In order to compute the transactionId the system takes the data block with the completed signature information and hashes this block using SHA-256 and the first 8 bytes of the hash are reversed and which is then used as the transactionId.

A signed transaction uses the following flow:

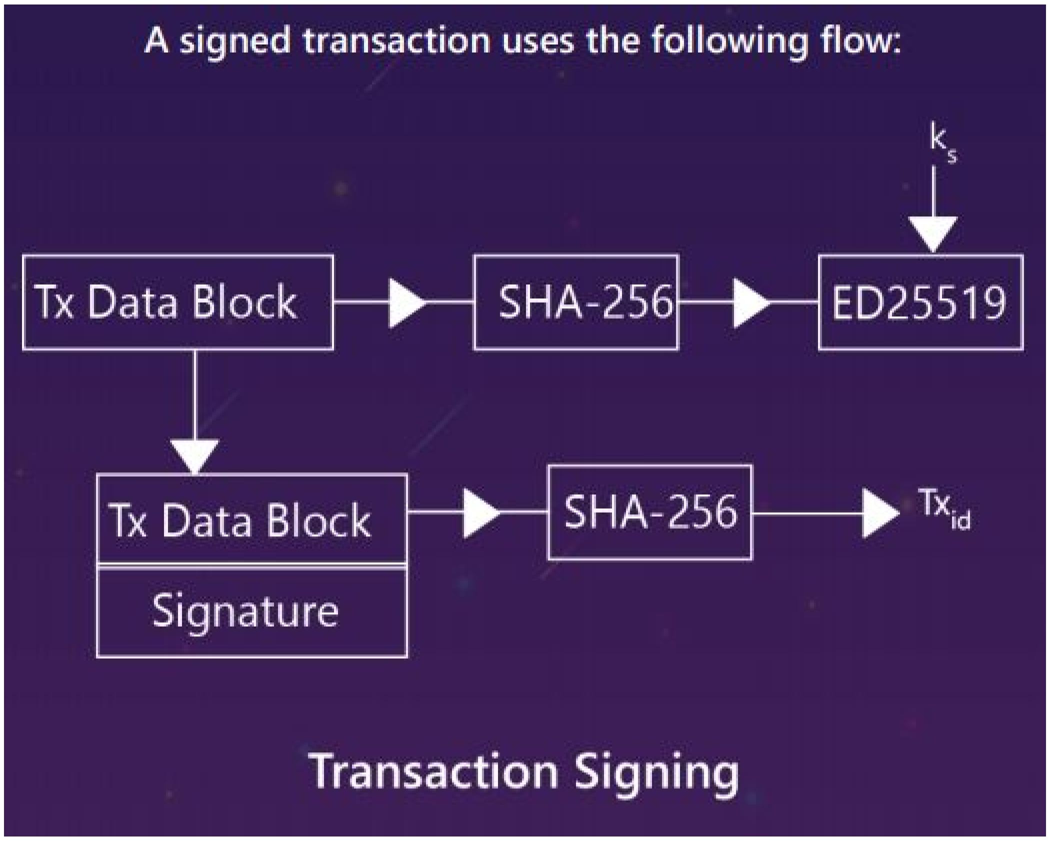

Figure 3.2: Transaction signing

With a second signature the flow looks like this:

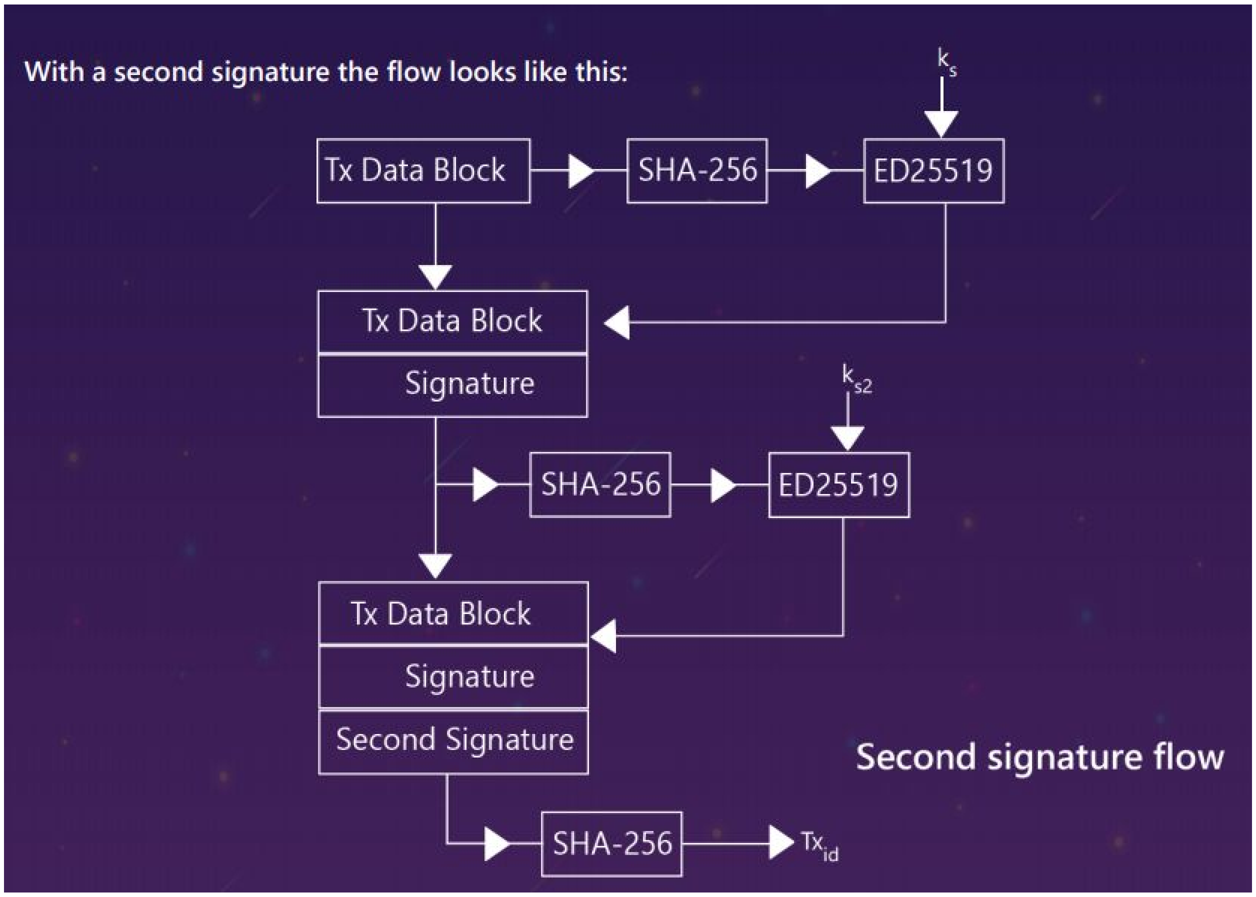

Figure 3.3: Second signature flow

[Please confirm how many transaction types are supported in DDK Blockchain]

Balance Transfer Transaction

A balance transfer transaction (type 0) is a transfer of DDK from one account to another account. In order to issue a balance transfer transaction, the following fields must be specified:

- The account Id of the recipient
- The amount of DDK to transfer
- The secret of the account

Once those three fields have been specified, the system will begin building the transaction object. First, the public key of the sender is computed using their secret. Following this, the data block is created using the process outlined in Transaction Signing and the additional field, recipientId, is added the to object. The resulting data block is 53 bytes.
This block is then signed. The final step of the transaction generation process is to compute the fee of the transaction. In the current system, the fee for SEND transaction is dynamic. Fee structure for SEND transaction is:

1. If balance <= 100DDK
   Then Fee: 0.01% Of Balance
2. If 100DDK< balance <= 1000DDK
   Then Fee: 0.005% Of Balance
3. balance > 1000DDK
   Then Fee: 0.001% Of Balance.

Once all these steps have been completed and the transaction validated, the transaction may be broadcast on the network. While the transaction may be present on the network, it will remain in unconfirmed status until it has been processed by a delegate. When the transaction is broadcast to the network and it is sent as a JSON object using the HTTP API.

A transaction object has:

- Timestamp
- Sender Public Key (256 bit)
- Recipient ID
- Amount

Second Signature Registration Transaction

A second signature registration transaction (type 1) is used to register a second passphrase on the blockchain. In order to issue this type of transaction the following fields must be specified:

- Secret: The secret of the account.
- Second Secret: The desired second pass phrase.

Once those two fields have been specified the system will begin building the transactions. The second public key is generated from the second pass phrase, and the system builds the data block of 85 bytes.

This data block is then signed using the user's secret ks, and the signature is appended to the object. Following this the computation of the fee is performed for the transaction type. In the existing implementation the fee for a signature transaction is fixed at 0.01 DDK.

The second signature registration transaction uses:

- Timestamp
- Sender Public Key (256 bit)
- Recipient ID
- Amount
- Sender Second Public Key (256 bit)

The final size of the transaction, with the signature, is 149 bytes.

Delegate Registration Transaction

A delegate transaction (type 2) is used to register the account as a delegate. In order to issue a delegate registration transaction the following fields are required:

- secret: The secret of the account.
- username: The delegate’s username.

The length of the username must be at least one character and at most twenty characters. Once those fields have been specified, the system can then compute the account's public key, and begin building the transaction’s data block with a maximum 73 bytes.
This data block is then signed using the account's secret, and the signature is appended to the transaction object. The system then computes the fee of the transaction. In the present implementation, the fee for a signature transaction is fixed and costs 10 DDK.

The delegate registration transaction utilises:

- Timestamp
- Sender Public Key (256 bit)
- Recipient ID
- Amount
- Delegate Username (160 bit)

The final maximum size of the transaction, with the signature, is 137 bytes, and with a second signature, 201 bytes.

Vote Transaction

A vote transaction (type 3) is a transaction used to vote for delegates. In order to issue a vote transaction, the following two fields are required:

- Secret: the secret of the account.
- Delegates: An array of public keys of delegates with '+' or '-' as prefix that indicate upvote or downvote respectively.

A vote is prepended with a ’+’ for adding stake to the delegate’s public key, and a ’-’ is prepended to the delegate’s public key if the account wants to remove the vote for the delegate. The maximum number of vote applications in one transaction is 3. Once those fields have been specified, the system can then compute the account's public key, and start building the transaction’s data block with a maximum 248 bytes.

This data block is then signed using the account's secret, and the signature is appended to the transaction object. The system will then compute the fee of the transaction. In the current implementation, the fee for a vote transaction is fixed at 0.01% of the total staked amount.

The voting transaction uses:

- Timestamp
- Sender Public Key (256 bit)
- Recipient ID
- Amount
- Delegate Votes (2145 bytes)

The final maximum size of the transaction, with the signature is 312 bytes, and with the second signature is 376 bytes.

Multisignature Registration Transaction

A multisignature registration transaction (type 4) is a transaction used to add a multisignature to an account. The following fields are needed in order to issue a multisignature registration transaction:

- **Secret:** The secret of the account the multisignature will be applied to.
- **Keys-group:** The array of keys to add or remove from the multisignature account.
- **Min:** The minimum amount of signature required to validate a transaction. (Minimum of 2)
- **Lifetime:** The time to wait for enough signatures before removing the transaction.

Each public key in the keys group is prepended with a '+' then the public key to be added to the multisignature account. The minimum amount of signature required to validate a transaction must be at least 2 and at most 16. The minimum amount of keys in the keys-group is two. The lifetime is specified in hours and a must be at least 1 hour and at most 24 hours. Once those fields have been specified, the system will then compute the account's public key, and start building the transaction's data block. The size of the data block depends on the amount of keys added to the multisignature registration transaction. Each key is 65 bytes due to the addition of the modifier.

This data block is then signed using the user's secret, and the signature is appended to the transaction object. The system will then compute the fee of the transaction. In the present implementation the fee for a multisignature registration transaction is 0.01 DDK per key in the keys-group. Note that the key of the account issuing the transaction is implicitly added in the multisignature.

The multisignature registration transaction utilises:

- Timestamp
- Sender Public Key (256 bit)
- Recipient ID
- Amount
- Multisignature Keys (65 bytes each)

The final size of a transaction with two keys in the keys group is 249 bytes, and 313 bytes if the account has a second pass phrase enabled.

Consensus mechanism DPoS

DDK uses **Delegated Proof of Stake, DPoS** as the consensus system of the chain, the delegates are the nodes that have the ability to generate blocks. The delegates are selected by rigorous voting between the stakeholders. The weight of the vote depends on the amount of DDK the stakeholder possess. A stakeholder can vote for a delegate using a vote transaction.

Consensus is a required aspect of any blockchain system. It serves a vital purpose for the system where there are many nodes and all nodes must agree on the integrity of the data. All nodes participating must agree on what transactional data is legitimate in order to move the blockchain forward.

Delegates

A delegate is a type of nodes that has been registered by the transaction delegate registration, These nodes have a special purpose within DDK as they are allowed to generate blocks for the system provided that the delegate has been allocated enough stake by other users of the system. Any node may become a delegate, but only if the pre-conditions of the required stake are allowed to generate blocks.