LIGHT · FAST · SWEET

TrustNote

Technical White Paper

TrustNote Foundation

March 2018

Overview

The TrustNote Development Team commissioned by the TrustNote Foundation is pleased to release Version V1.1.7 of the TrustNote Technical White Paper. This document describes TrustNote's background, technical characteristics, and user scenarios, it will be updated by the TrustNote Development Team from time to time to reflect evolving best practices and lessons learned during the implementation. For the latest information, technical white papers, software releases and technical support for developer communities, visit the official website at www.trustnote.org.

Contact Us

Business Enquiries: foundation@trustnote.org

Technical Support: community@trustnote.org

Copyright

Disclaimer

Technology continues to evolve and the blockchain technology in particular, continues to improve. Under the supervision of the TrustNote Foundation, the TrustNote Development Team is committed to keep improving the technology solution as well as updating the white paper whilst keeping the maximum supply and distribution policy of the TTT token unchanged. In addition, because of the existence of "private equity" scams targeting crypto-currency investors, TrustNote Foundation hereby informs organizations and individuals that participation in crypto-currency investments through unauthorized trading channels should always take precautionary measures against such risks. Neither TrustNote Foundation nor the TrustNote Development Team accept responsibility in any outcome resulting from investments via unauthorized trading channels.

Abstract

Today’s blockchain technologies face many challenges such as network congestion, high transaction fees, and long transaction confirmation delay, TrustNote seeks to address these problems by building the world-leading public Directed Acyclic Graph (DAG) ledger which is minable, capable of handling high concurrent transactions yet still maintain quick transaction confirmation. TrustNote is focused on creating an easy-to-use, decentralized, low-level digital token blockchain that leverages declarative Smart Contracts with enhanced expression capability, empowering users to create and publish digital tokens without having to write complex Smart Contracts. TrustNote has an extensible wallet that provides security and rich API interfaces for digital tokens, blockchain games and social networks, allowing new innovative ideas to run smoothly on the blockchain network, making user friendly blockchain applications accessible to everyone. TrustNote has an innovative two-tier consensus mechanism and the TrustME consensus algorithms. These algorithms work by periodically selecting a number of Super Nodes to act as attestation authorities and these receive mining rewards according to their validated attestations.

Table of Contents

1. BACKGROUND 1
2. WHAT IS TRUSTNOTE 2
   2.1 Key Features 2
   2.2 Directed Acyclic Graph 3
   2.3 Comparison 4
3. DATA STRUCTURES 5
   3.1 Unit 5
   3.2 Message Types 6
4. CONSENSUS 12
   4.1 Nodes 12
   4.2 Unit Inter-Reference 13
   4.3 Main Chain 14
   4.4 Transaction Confirmation 15
   4.5 Transaction Fees and Mining Reward 15
   4.6 TrustME-PoW 16
   4.7 TrustME-BA 17
      4.7.1 Design Goals 18
      4.7.2 Final Consensus and Tentative Consensus 18
      4.7.3 Lottery Algorithm 19
      4.7.4 Byzantine Agreement 20
5. SMART CONTRACT 20
6. ECOSYSTEM AND APPLICATIONS 25
7. ISSUANCE AND DISTRIBUTION 29
8. References 31

1. BACKGROUND

For nearly 10 years, since January 3, 2009, Bitcoin has been operating safely, a miracle in the history of computer network technology. The success of bitcoin unlocked a massive digital crypto-currency ecosystem full of imagination. As the original bitcoin developer, Satoshi Nakamoto creatively proposed the blockchain, a chained data structure based on hash functions, and succeeded in building a well-operated, decentralized peer to peer network which opened the new era of digital crypto-currencies. Blockchain technologies are evolving fast, driving change across many industries, sparking innovation and creativity.

Blockchain has provided a decentralized trust mechanism and has become a brand-new paradigm and key methodology in data protection and data value exchange. Now in its booming period, blockchain is constantly being integrated with various technologies, various scenarios are also being explored in terms of how to utilize the technical characteristics of blockchain, blockchain applications have been expanded from data tamper resistance and data value exchange to digital tokens and social-networking arenas. The growing number of blockchain user scenarios pose many challenges for blockchain technology, demanding stronger security, higher transaction concurrency, and shorter transaction acknowledgment delay.

In Bitcoin's blockchain, all the data blocks are aligned in one continuous chain, but due to the limitations on block size and consensus mechanism, the amount of concurrent transactions is limited and transaction confirmations are slow, which causes the rise of transaction fees and frequent trading congestions. To address these issues, the Bitcoin developer community has come up with solutions such as block size increases, segregated witness, and lightning networks, but none of them is perfect. Those solutions either just ease the problem, or sacrifice security or consistency, and none of them have reached full agreement within the community. The recent emergence of multiple bitcoin forks, has heated up the debate even further.

The structure of the ‘traditional’ blockchain is the bottleneck that hinders the technology from improving its concurrency. More efficient forms of distributed ledger technology are being sought and a solution which combines Directed Acyclic Graph (DAG) and blockchain (hereinafter referred to as "DAG-ledger") was proposed. The DAG-ledger has no concept of blocks, so there is no limit to the size of the blocks. In addition, DAG-ledger uses a new form of transaction verification referencing the old transaction for transaction confirmation. This allows minor temporary differences between the users’ ledgers, to achieve the goal of preventing transaction obstruction by weakening the consistency of the entire network in a short period. The larger the network is and the greater the transaction volumes are, the shorter the transaction confirmation delay is.

IOTA and Byteball both developed their own public DAG-ledgers in 2016 to accommodate high-frequency trading scenarios. However, the downside is that although DAG-ledger supports high-frequency trading, in the case of low-frequency trading, the old transaction cannot get enough new transactions to verify and reference, resulting in the old transaction not being confirmed in time, in extreme cases the transaction may never get confirmed. To address this problem, IOTA proposes a temporary centralized actor called coordinator, which is used to protect the network when the volume of transactions is low, however IOTA does not disclose the design details of such coordinator; Byteball introduces 12 witnesses, implementing transaction confirmation via witness attestation, although Byteball claims its users have the right to choose their own witness, but the transaction quoting rules make it very difficult for users to change witnesses if they choose to do so.

2. WHAT IS TRUSTNOTE

TrustNote is a minable public DAG-ledger with an innovative, two-tier consensus mechanism designed for new applications such as digital tokens issuance, blockchain games and social networks, its digital token is called "**TTT**". TrustNote's goal is to be Light, Fast, and Sweet. "Light" means TrustNote has a light architecture and intelligent contracting system that supports lightweight application extensions and micro wallets; "Fast" means TrustNote supports high concurrency transactions, enjoys fast transaction confirmation, and makes distributed application (DApp) development and deployment much easier; "Sweet" means it enables an ecosystem that allowing new innovative ideas to run smoothly on the ledger, making user friendly blockchain applications accessible to everyone.

2.1 Key Features

- Two-tier consensus mechanism, a minable public DAG-ledger
- Supports high concurrency transactions, benefits from fast transaction confirmation
- Supports advanced declarative Smart Contracts
- Token issuance system
- Hash Algorithm: BLAKE2
- Digital Signature Algorithm: EdDSA

- Multi-platform wallet, light wallet, micro wallet, support third-party extensions

2.2 Directed Acyclic Graph

A Directed Acyclic Graph (DAG), is a finite directed graph with no directed cycles. It consists of finitely many vertices and edges, with each edge directed from one vertex to another, such that there is no way to start at any vertex V and follow a consistently-directed sequence of edges that eventually loops back to V again. The use of the DAG data structure to store ledger data is gradually getting more developers' attention. Projects like IOTA and Byteball have successfully established stable public-ledgers using DAG, the feasibility of a DAG-ledger is proven.

In TrustNote terms, transactions are viewed as messages. Various types of messages are supported, multiple messages can be combined into a data block which is called a "**Unit**", a DAG is formed by inter-referenced Units. Since each Unit can reference any previous Unit or multiple previous Units, there is no need to spend more computing power and time for solving the consensus problem, nor need to wait for the completion of strong inter-node data synchronization, and because there is no need to assemble multiple Units into blocks, the performance of concurrent transactions is greatly improved and the confirmation delay are reduced to minimum.

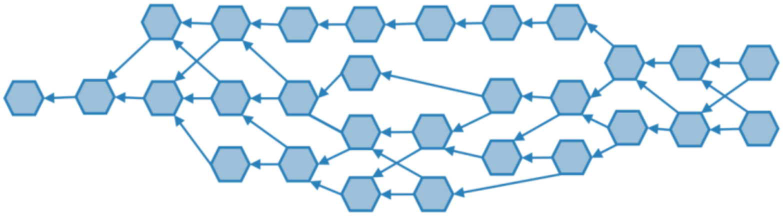


Figure 2-1 DAG Graph

TrustNote uses the following technique to solve the double-spending problem (double-spending is a problem unique to digital currency in which the same single digital token can be spent more than once). First, try to find a Main Chain (MC) starting from Genesis Unit on the DAG and assign indexes to the Units that lie on the MC, the Genesis Unit's index is 0, Genesis Unit's Child Unit's index is 1, and so on. Second, for those Units that do not lie on the MC, define their indexes equal to the first MC Unit references this Unit. Eventually, every transaction on the DAG has an index. If two transactions try to use the same output, we just need to compare the value of their indexes (Main Chain Index, MCI), the Unit with a smaller index is valid, the Unit with a larger index is invalid, thus solves the double-spending problem. For example, when double-spending occurs (as shown in figure 2-2), after the MCIs are assigned to each transaction, we can determine the transaction whose MCI is 11 is valid, the other transaction whose MCI is 15 is rejected.

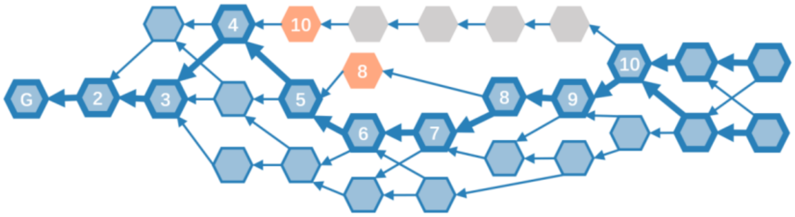


Figure 2-2 Main Chain (MC)

For security concerns, unlike Bitcoin's blockchain which is guaranteed by the massive computing power of the network, DAG based TrustNote relies on the fast advance of transactions and the uncertainty of the relationship between the transactions as the "**firewall**", which leaves the entire system looks too ruleless to be attacked. TrustNote benefits from a two-tier consensus mechanism and an innovative TrustME Consensus Algorithm. Those Super Nodes that participate in the TrustME consensus and contribute to the healthy expansion of DAG-ledger will get the mining reward.

2.3 Comparison

Standing on the shoulders of giants, absorbs the advantages of existing blockchain projects and addresses their major issues, a more prosperous TrustNote ecosystem become possible. A horizontal comparison of current well-known DAG-ledgers (IOTA and Byteball) with TrustNote is shown in Table 2-1 below.

Table 2-1 DAG-Ledger Comparison

	IOTA	Byteball	TrustNote
Token	IOTA	Byte	TTT
Market Value	US$ 14 Billion	US$ 0.4 Billion	--
Consensus Mechanism	PoW Cumulative Weight	12 Witnesses	Decentralized TrustME Consensus Mechanism
Smart Contract	N/A	Declarative Contract	Advanced Declarative Contract
Reward	N/A	Transaction Reference and Attestation	Transaction Reference and Mining
Nodes	Full Node Light Node	Full Node Light Node	Super Node Full Node Light Node Micro Node
Transaction Fee	No	Yes	Yes

	IOTA	Byteball	TrustNote
Double Spending	PoW Weight Comparison	Main Chain Sequencing	Main Chain Indexing
Low-frequency Trading	Centralized Coordinator	Weak Centralized Attestor	TrustME Attestor

3. DATA STRUCTURES

3.1 Unit

When a Node initiates a transaction or sends a message, it creates a new data block called a "**Unit**" and broadcasts the Unit to its peers. A Unit may contain multiple messages of various types, each Unit contains the following information:

- Header: The hash value of the current Unit's previous (parent) Unit.
- Messages: A Unit contains one or more messages, there are various types of message, and each message type has its own unique data structure.
- Signatures: A Unit contains one or more user signatures. A user can have multiple addresses, the addresses are generated with BIP-0044 algorithm.

Definition of each Unit's field is shown in table 3-1 below.

Table 3-1 Field Definition of Unit

Field Name	Definition	Remarks
version	TrustNote protocol version number	e.g. '1.0'
alt	Token identification	e.g. '1'
messages	Message array	See Table 2-2 for detail
authors	Author array	Address array of the Unit's authors
parent_units	Parent Unit's hash array	Hash value array of the Unit's parent Units

The messages field stores the actual data of the Unit, it is an array of one or more messages. TrustNote supports various types of messages which are distinguished by the app field.

Table 3-2 Field Definition of messages

Field Name	Definition	Remarks
app	Message type	e.g. "payment" or "text", see 2.2 for detail
payload_location	Location of the message body	"inline" indicates the message body is stored in the current message; "uri" indicates the message body is retrievable from a URL address; "none" indicates there is no message body
payload_hash	Hash value of the message body	

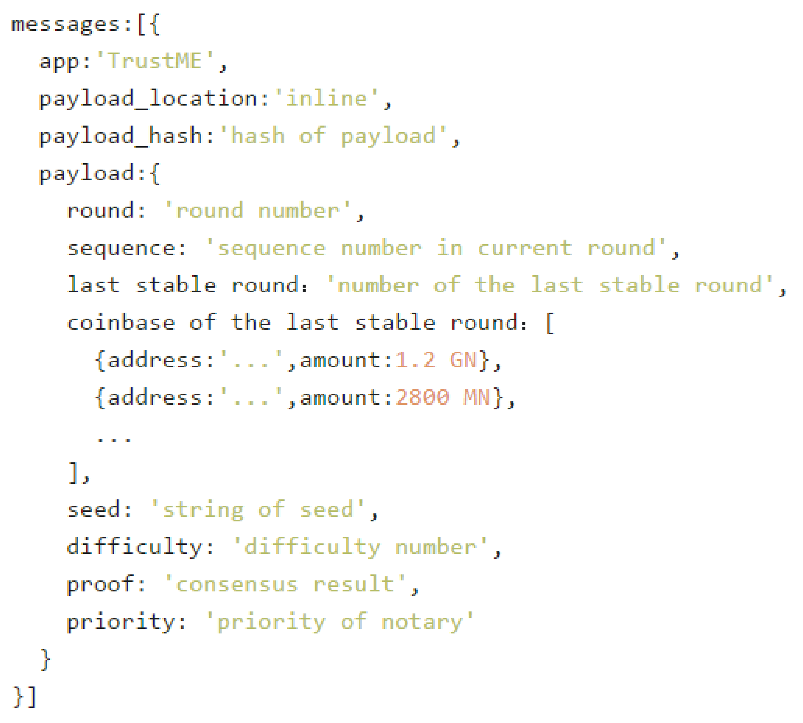
Field Name	Definition	Remarks
Payload	Message body	Various message types are used to kept different data format, e.g. a transaction message include inputs and outputs
payload_inputs	Message inputs array	include the hash value of the Unit who generated the message, message index, output index etc.
payload_outputs	Message outputs array	include addresses of recipients, amount of transaction etc.

3.2 Message Types

TrustNote supports various message types which can be further extended if needed, different types of messages are used to store different data formats interpretable by different parsing rules. Different types of TrustNote message are recognizable by the message’s app field.

- **Attestor (app = TrustME)**

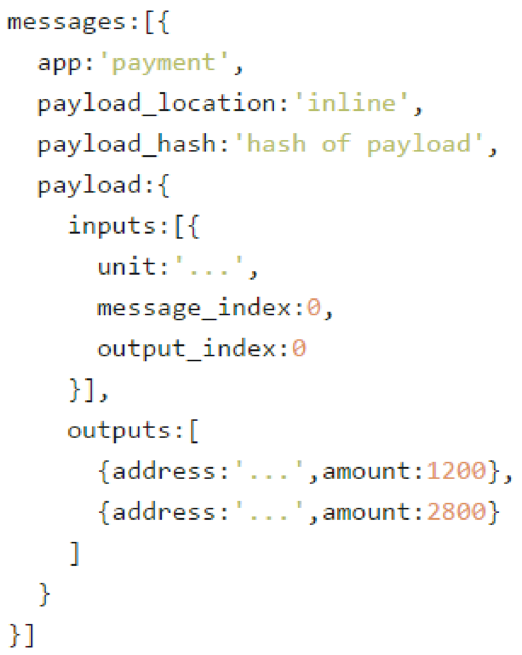
No Nodes other than the Attestor Nodes can generate “**Attestor**” messages, Attestor messages are used to store the attesting results the Node has. If the first message of a Unit is a Attestor message, then this Unit is a Attestor Unit. The contents of Attestor message include: Consensus Round, Attestor Unit Index, Latest Stable Consensus Round, Attestation Reward of Latest Stable Consensus Round, Seeds, Difficulty, Attesting Result, Attesting Priority, etc. For more information about the Attestor Unit and how to get attested, please refer to chapter 3: Consensus Mechanisms.



```
messages:[{
  app:'TrustME',
  payload_location:'inline',
  payload_hash:'hash of payload',
  payload:{
    round: 'round number',
    sequence: 'sequence number in current round',
    last stable round: 'number of the last stable round',
    coinbase of the last stable round: [
      {address:'...',amount:1.2 GN},
      {address:'...',amount:2800 MN},
      ...
    ],
    seed: 'string of seed',
    difficulty: 'difficulty number',
    proof: 'consensus result',
    priority: 'priority of notary'
  }
}]
```

Transaction (app = payment)

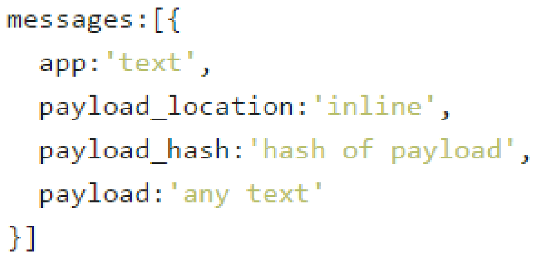
"**Transaction**" messages are used to hold tokens' transactional information. More than one inputs and outputs can be included in a transaction message. For user defined assets, it is necessary to specify the hash value of the Unit which defines the asset. A standard transaction message is as follows:



```
messages:[{
  app:'payment',
  payload_location:'inline',
  payload_hash:'hash of payload',
  payload:{
    inputs:[{
      unit:'...',
      message_index:0,
      output_index:0
    }],
    outputs:[
      {address:'...',amount:1200},
      {address:'...',amount:2800}
    ]
  }
}]
```

Text (app = text)

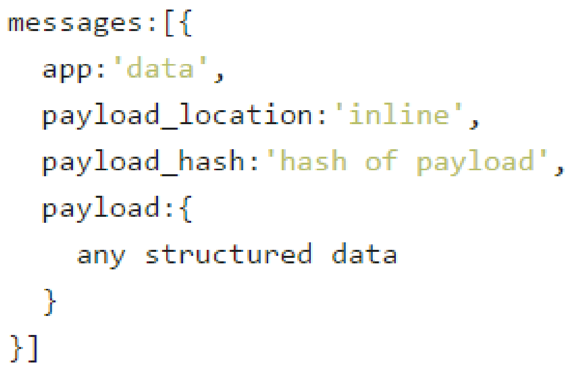
"**Text**" messages are used to hold arbitrary string data.



```
messages:[{
  app:'text',
  payload_location:'inline',
  payload_hash:'hash of payload',
  payload:'any text'
}]
```

Structured Data (app = data)

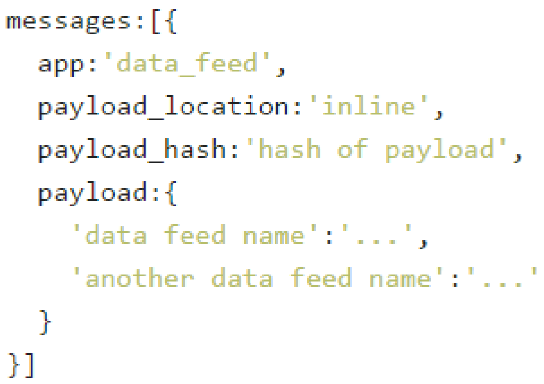
"**Structured Data**" messages are used to store arbitrary structured data.



```
messages:[{
  app:'data',
  payload_location:'inline',
  payload_hash:'hash of payload',
  payload:{
    any structured data
  }
}]
```

Data Feed (app = data_feed)

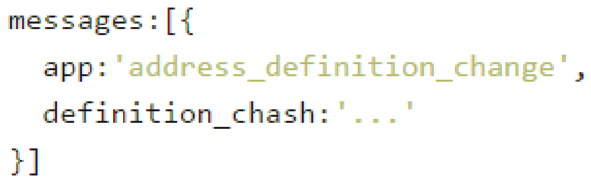
"**Data Feed**" messages are sent by trusted third-parties to trigger Smart Contract.



```
messages:[{
  app:'data_feed',
  payload_location:'inline',
  payload_hash:'hash of payload',
  payload:{
    'data feed name':'...',
    'another data feed name':'...'
  }
}]
```

Address Definition Change (app = address_definition_change)

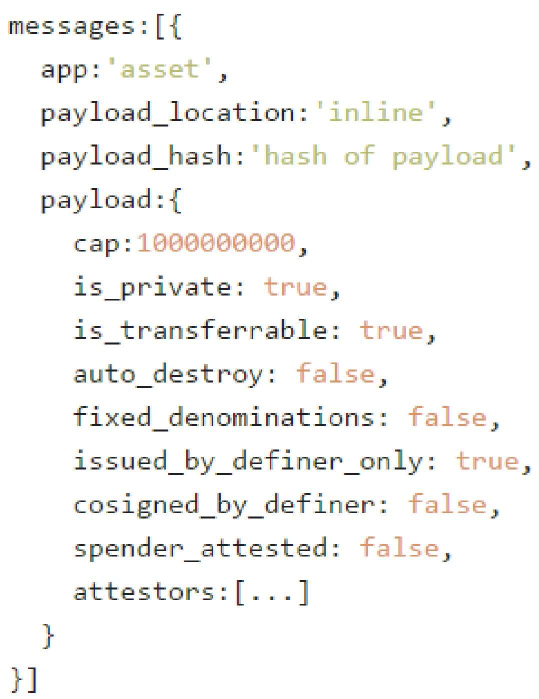
"**Address Definition Change**" messages are used to update the address definition while retain the old address.



```
messages:[{
  app:'address_definition_change',
  definition_chash:'...'
}]
```

Asset Definition (app = asset)

"**Asset Definition**" messages are used to define new digital assets.



```
messages:[{
  app:'asset',
  payload_location:'inline',
  payload_hash:'hash of payload',
  payload:{
    cap:1000000000,
    is_private: true,
    is_transferrable: true,
    auto_destroy: false,
    fixed_denominations: false,
    issued_by_definer_only: true,
    cosigned_by_definer: false,
    spender_attested: false,
    attestors:[...]
  }
}]
```

Definition of each field is shown in table 3-3 below.

Table 3-3 Field Definition of Asset Definition message

Field Name	Definition	Remarks
cap	Maximum amount of assets can be defined	
is_private	Indicates whether the exchange of assets is private or public	
is_transferrable	Indicates if the asset can be transferred between third parties while bypassing the asset definer	If set as “false”, the asset definer must be either the sole sender or the sole receiver of each transfer
auto_destroy	Indicates if the asset should be destroyed when it is sent to the definer	
fixed_denominations	Indicates if the asset can be sent in arbitrary integer amount, or in fixed denominations like traditional currency or coins, e.g. 1, 2, 5, 10, 20, etc.	
issued_by_definer_only	Indicates if the asset can only be defined by the definer himself	
cosigned_by_definer	Indicates if every transfer must be cosigned by all asset definers	Useful for regulated assets
spender_attested	Indicates if the spender of the asset must get attested before he spends. If he happens to receive the asset but it is not yet been attested, he must get attested by one of the listed attesters before spending the asset	Useful for regulated assets
attesters	The list of attesters’ addresses recognized by the asset definer (only when spender_attested is set as “true”)	The list can be later-on updated by the definer by

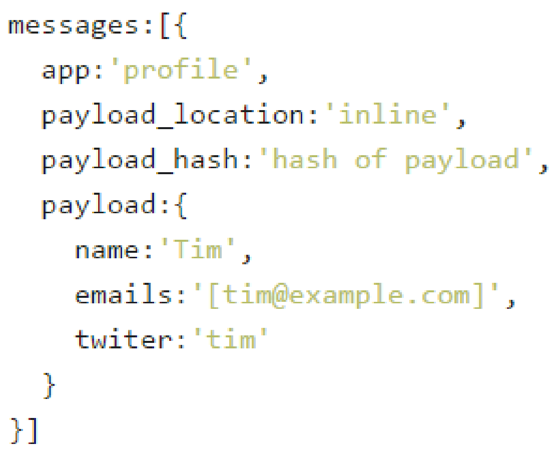
Field Name	Definition	Remarks
		sending an “asset_attestors” message
denominations	Lists every supported denominations and total amount of each denomination	Used for fixed_denominations assets only, not shown in the example
transfer_condition	Defines the condition when assets are allowed to be transferred. The syntax of this defination is the same as address, except that it cannot reference any attestation data, such as “sig”	Usually there are no restrictions except those already defined by other fields
issue_condition	Same as transfer_condition but for issue transactions only	

Asset Attestors (app = asset_attestors)

“**Asset Attestors**” messages are used by asset definers to update the attestors of the asset.

Profile (app = profile)

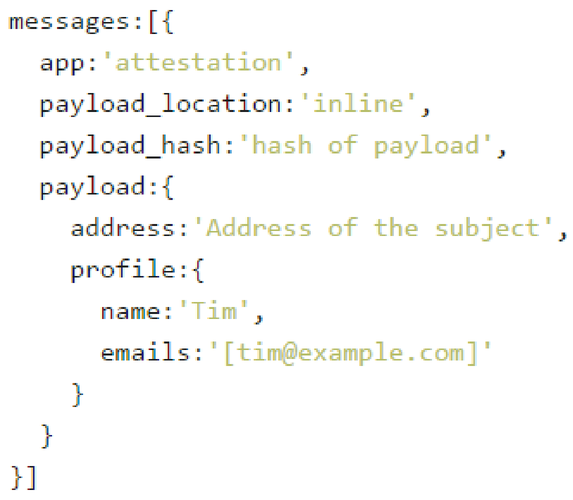
“**Profile**” messages are used to disclose a user’s personal profile information; the authenticity of a Profile message can be verified by an Attestation message from a trusted third-party.



```
messages:[{
  app:'profile',
  payload_location:'inline',
  payload_hash:'hash of payload',
  payload:{
    name:'Tim',
    emails:'[tim@example.com]',
    twiter:'tim'
  }
}]
```

Attestation (app = attestation)

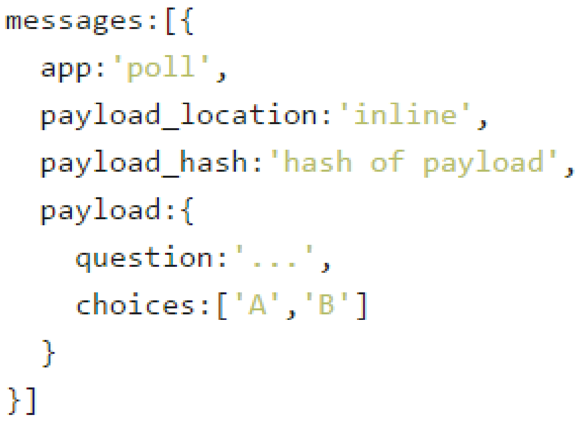
“**Attestation**” messages are used to publish personal information associated with a user, Attestation messages are always sent by trusted third-party and in some cases Attestation messages are used to implement user authentication.



```
messages:[{
  app:'attestation',
  payload_location:'inline',
  payload_hash:'hash of payload',
  payload:{
    address:'Address of the subject',
    profile:{
      name:'Tim',
      emails:'[tim@example.com]'
    }
  }
}]
```

Poll (app = poll)

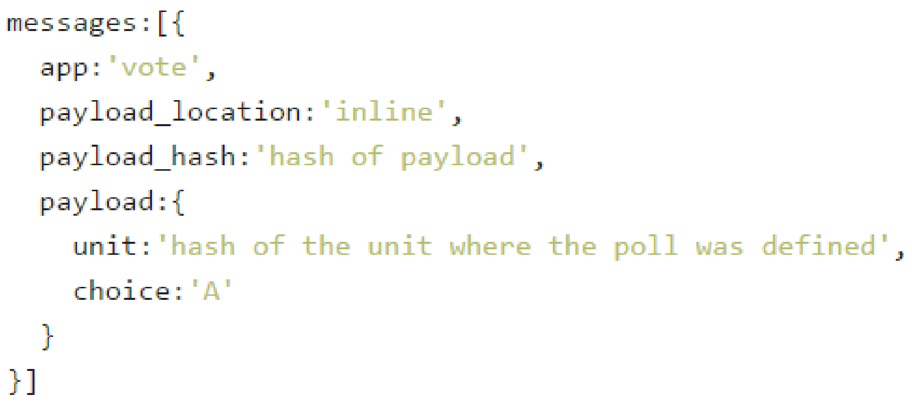
“**Poll**” messages are used to initiate a poll.



```
messages:[{
  app:'poll',
  payload_location:'inline',
  payload_hash:'hash of payload',
  payload:{
    question:'...',
    choices:['A','B']
  }
}]
```

Vote (app = vote)

“**Vote**” messages are used for initiating a vote.



```
messages:[{
  app:'vote',
  payload_location:'inline',
  payload_hash:'hash of payload',
  payload:{
    unit:'hash of the unit where the poll was defined',
    choice:'A'
  }
}]
```

4. CONSENSUS

TrustNote adopts a two-tier consensus mechanism comprising “base consensus” and “attested consensus”. The base consensus, also known as “DAG consensus”, requires new transaction Units to be sent out by Nodes to verify and reference previous transaction Units. The attested consensus, or “TrustME Consensus”, requires that the sequences of Non-Attestor Units be rigorously determined by Attestor Units generated from the Attestor Nodes. Such two-tier consensus mechanisms can improve transaction throughput and reduce transaction confirmation delay, thus effectively solving the problem of Excessive Bifurcation and double-spending.

For a more robust TrustNote ecosystem, two TrustME consensus schemes are developed. Initially, TrustNote uses a Proof of Work (PoW) based scheme called TrustME-PoW; in future, TrustNote will adopt a Byzantine Agreement (BA) based scheme called TrustME-BA. No matter which scheme is used, any Super Node participating in the consensus will receive a reward in the form of TTT if they are elected as Attestor Node.

Under the TrustME-PoW scheme, Super Nodes get attestation authority by proving their superior computing power; under the TrustME-BA scheme, a pseudo-random algorithm is used to select Attestor Nodes from Super Nodes. In both scenarios, Attestor Units issued by Attestor Nodes always comply with the unit inter-reference rules, and do not affect the existed references between other Units. Only after a Attestor Unit become a stable Unit in the Main Chain, it could finally justify that a Attestor Node has contributed to TrustNote positively, and thus receive the Attestation reward. In addition, both schemes encourage fair participation of all Nodes, TrustME consensus mechanisms are fairer, more trust worthy, and safer than those centralized and weak centralized schemes.

4.1 Nodes

TrustNote supports 4 variants of Nodes: Super Node, Full Node, Light Node and Micro Node. The comparison of these Nodes is shown in table 4-1 below.

Table 4-1 Comparison of Nodes

	Super Node	Full Node	Light Node	Micro Node
ledger	full ledger	full ledger	light ledger	N/A
transaction	√	√	√	commissioned
DAG consensus	√	√	indirect	×
TrustME-PoW	√	×	×	×

	Super Node	Full Node	Light Node	Micro Node
TrustME-BA	√	×	×	×
Hosting Micro Node	√	×	×	×
deployment	Mining Systems Cloud Host Server/Workstation PC	Cloud Host Server/Workstation PC	Smartphone Tablet PC	MCU Smart Card

TrustNote has a peer to peer network similar to bitcoin, each Node can select a random set of peer Nodes to propagate messages. To ensure the messages cannot be forged, each message is signed by the private key of its original sender, other Nodes must validate the signature before forwarding the message. To avoid message forwarding loop, one Node does not forward the same message twice.

To qualify as a TrustNote Super Node, the following conditions must be met:

- Resource: Has good internet bandwidth, large storage space and enough computing power, ideally with public IP address;
- TTT: Hold a certain amount of TTT in multiple consensus rounds;
- Credibility: Has a reputable history of always submitting valid Units.

A Super Node can participate in the TrustME consensus and become an Attestor Node. Attestor Nodes will get an Attestation Reward by sending Attestor Units, and earn attestation fees from attesting ordinary Units.

4.2 Unit Inter-Reference

Each Unit in TrustNote can reference multiple Units that have no Parent-Child relationship with each other, the new Unit will preferentially reference the Units with more Parents. When following a Parent Unit in its Child Unit's direction, we would see many forks if a Unit is referenced by many Child Units, and many Parent Units will merge into one if these Parents are referenced by one Child.

The purpose of referencing a Parent Unit is to establish an obscure order among the Units. Before referencing a Parent Unit, a Node needs to validate the Parent Unit, such as checking whether the signature is valid, whether the reference is legal, etc. TrustNote does not require strong synchronization between Nodes, different Nodes may see temporarily inconsistent DAGs. But this does not undermine the Parent-Child relations that has already been established among the Units, and it may only cause the Parent Node has many Childs. TrustNote supports high transaction throughput with low network latency simply because TrustNote does not enforce strong data synchronization among Nodes.

To minimize the number of garbage Units generated in the DAG, a transaction fee must be paid when any Node submits a new Unit. The transaction fee is divided and paid to 1) the Node(s) who generate newer Unit and reference this Unit as Parent, and 2) the Attestor Node who attested the Unit; if a Unit is referenced by multiple Child Units, the Node who sends the Child Unit with the smallest hash value will get the referencing fee. In addition, to qualifying the rewards, the Main Chain Index (MCI) of the Child Unit must equal or be slightly greater than its Parent's MCI, this restriction encourages Nodes to reference the most recent Parent Unit as quickly as possible and as much as possible so they can get more referencing fees, thus helping the DAG to get fast convergence and reduce the number of forks.

4.3 Main Chain

To choose a single chain along Child-Parent links within the DAG, and then relate all Units to this chain. All the Units will either lie directly on this chain, or be reachable from it by a relatively small number of hops along the edges of the graph, this single chain is called Main Chain (MC). If starts from another Childless Unit, we can build another MC. If we build 2 MCs from 2 different Childless Units follow the same rule, the 2 MCs will completely overlap with each other when and after the 2 MCs intersect at some point. The worst-case scenario is that 2 MCs intersect at the Genesis Unit. Although Nodes are independent from each other when generating new Units, and there is no existence of any possible coordination, we still expect the intersection of the MCs to be as close to the Childless Unit as possible.

Once a Unit has selected an MC, it can establish an index for two conflicting Units that haven't been indexed. First, indexing the Units that lie directly on the MC, the index for the Genesis Unit is 0, the index for the Genesis Unit's Child on the MC is 1, and so on, until all Units lying on the MC are indexed. For those Units not lying on the MC, we can always find the first Unit lying on the MC and reference the Unit directly or indirectly, thus assigning a MCI to each Unit. Consequently, given any two Units, the Unit with a smaller MCI must be generated earlier. If the MCIs of two Units happen to be the same and these two Units are in conflict, the Unit with the smaller hash value is considered the valid one. TrustNote will keep all double-spending Units including those deemed to be invalid.

The process of building the MC applies the Parent Selection Algorithm recursively. By participating in the TrustME consensus, Super Nodes gain the opportunity to become Attestor Nodes that can send Attestor Units. By comparing the number of Attestor Units among the available paths, the Parent Selection Algorithm will pick-up one of the Parent Units as the "Best Parent Unit". For different Nodes, the MC building processes are completely independent and only rely on the DAGs that each Node sees. Starting from a DAG’s Childless Unit, following the path of the Best Parent Unit, a Node can build an MC through to the Genesis Unit.

4.4 Transaction Confirmation

As new Units are created, each Node keeps track of its current MC as if they are going to create a new Unit for every live Childless Unit. Current MCs may be different for different Nodes because they may see different sets of unstable Units. The current MC will constantly change itself as new Units arrive. However, certain parts of the MC that are old enough will remain unchanged.

When traveling back, all MCs will come to some point, this point and any previous Units are stable and won’t be changed by the arrival of new Units. In fact, the Genesis Unit is a natural initial stable point. Assuming we have built a current MC based on the current set of unstable Units, and there are some Units that lie on this MC that were previously believed to be stable, this means that all future current MCs believe they will meet the same stable Units and travel back along the same path. If we can find a way of advancing this stable point forward in the direction against the Genesis Unit, then we should be able to prove the existence of such stable point by Mathematical Induction. Those Units referenced by this stable point will get a definite MCI, and all messages contained in these Units will also get confirmed.

4.5 Transaction Fees and Mining Reward

Transaction fees must be paid for publishing transactions, the Node calculates the transaction fee based on the number of bytes generated. The transaction fee is divided into 2 parts, 60% as Referencing Fee and 40% as Attestation Fee. The Referencing Fee will be obtained by the Child of the Unit, and the Attestation Fee will be added to the attestation bonus pool of the consensus round, to which the Attestor Unit who is on the MC and has the closest MCI belong. The sending Attestor Unit also needs to pay the transaction fee, the calculation of transaction fee is the same as sending ordinary Units. Since Attestor Units usually contain more information and take up more storage space than ordinary Units, so its transaction fees are relatively higher, thus encouraging other Units to reference the Attestor Units.

In the TrustNote world, the growth of DAG-ledger and the TrustME consensus are asynchronous, each round of consensus will elect a number of attestors who have the authority to submit Attestor Units. Before the MC become stable, there is no way to decide which Attestor Units lie on the MC, or to evaluate the effectiveness of the Attestor Units' references, it is therefore impossible to give an Attestation Reward directly like Bitcoin does. After each consensus round, and when every Attestor Unit issued by all Attestor Nodes becomes stable, the Attestation Reward amount for each Attestor Node can be determined. Note ordinary Units may also appear in the MC, they may get Referencing Fee from their Parent Unit, but they can't get a share of Attestation Reward.

TrustME consensus is carried out periodically with a certain number of Attestor Nodes always elected for each round. A consensus round will become stable only if all Attestor Units sent by all Attestor Nodes elected in the round become stable. The first message of each Attestor Unit is the Attestor message of the Attestor Node itself. Each time when the TrustME consensus is reached, the first Attestor Unit generated by the current round's Attestor Node must contain the Attestation Reward of the latest stable consensus round. Note the Attestation Reward of the latest stable consensus round has been defined when a new round of TrustME consensus starts, and it is one of the input variables of the TrustME consensus algorithm. In the same consensus round, Attestor Units generated by other Attestor Nodes no longer contain the Attestation Reward, instead they validate and reference the first Attestor Unit to confirm the Attestation Reward of stable consensus round. By doing so, the capabilities of Attestor Nodes are weakened thus preventing malicious Super Nodes from disturbing the Attestation Reward income of Attestor Nodes of stable consensus round by obtaining the attestation authority multiple times.

4.6 TrustME-PoW

TrustME-PoW is a consensus mechanism which selects a small number of Nodes as Attestor Nodes using proof of work at each round, and determines the priority of Attestor Nodes accordingly. The TrustME-PoW consensus algorithm is executed every 5 minutes, each time when a consensus is reached, no more than 20 Super Nodes will be selected as Attestor Nodes. These Attestor Nodes have the authority to send Attestor Units and are rewarded accordingly.

TrustME-PoW is based on the Equihash algorithm, using BLAKE2 as the underlying hash function, to reduce the unfair advantages of ASIC mining, and to encourage equitable participation from more Super Nodes, thus making the probability distribution of Super Nodes becoming Attestor Nodes more reasonable. The input of the Equihash algorithm include Current Round Number, Seeds, and the Difficulty Factor, etc. The Current Round Number begins at 0, incrementing by 1 after each round. The Seeds of each round of consensus are calculated from the Seeds of the last round of consensus and the consensus results, which can be retrieved publicly and verified. The Difficulty Factor is calculated from the average computing power of the whole network, and the average time interval of consensus is controllable by adjusting the Difficulty Factor.

Attestor Units must comply with the previously mentioned Unit inter-reference rules. The Attestor Unit can reference unstable Unit only and must validate the Units it references, and the correctness of the "Child-Parent" relationship, until the stable MC unit is verified. Attestor Units are encouraged to reference multiple Best Parent Units that are not stable yet, thus to accelerate the stabilization of the Units and promote DAG-ledger's forward advancement and convergence.

Only when the Attestor Unit becomes the Unit on the MC, the corresponding attestation reward can be obtained. In a consensus round, the Attestor Units on the MC calculate their proportion of current consensus round's attestation reward according to the number of effective references. When each Attestor Unit becomes the MC Unit and stabilizes, the Attestor Unit's effective references are calculated.

4.7 TrustME-BA

TrustME-BA is a consensus mechanism based on Verifiable Random Function (VRF) and Byzantine Agreement (BA) algorithm, it randomly selects a small number of Super Nodes as Attestor Nodes, and determines the priority of the Attestor Nodes.

TrustME-BA is executed once every minute, and every time when a consensus is reached, a number of Super Nodes will be selected as Attestor Nodes in random. Attestor Nodes have the authority to send Attestor Units which must comply with DAG consensus' Parent-Child inter-reference rule. Once the Attestor Unit sent by the Attestor Node stabilizes on the MC, the Attestor Node will get the attestation reward. When transactions are active and new Units continue to be generated, Attestor Nodes will receive their Attestation rewards timely. When the transactions are less active or even for extreme cases when there is no new Units been generated in the last one-minute time window, Attestor Node will receive its attestation reward after the Attestor Unit it sends becomes stabilized MC unit, and those Nodes who haven't sent Attestor Unit will not get the Attestation reward.

4.7.1 Design Goals

TrustME-BA consensus mechanism is designed to achieve the following 2 goals:

- **Security**

With overwhelming probability, all Super Nodes will agree on the set of selected Attestor Nodes, which means when most honest Super Nodes accept a consensus result, then any consensus processes in the future can be traced back to this previous consensus result. TrustME-BA assumes: 1) Honesty Super Nodes hold more than two thirds of total TTT in circulation; and 2) Attackers can participate in consensus and receive the appropriate rewards. The rationale for this assumption is that in order to attack TrustME-BA successfully, attackers must invest enough TTT tokens. TrustME-BA assumes an attacker can control a certain amount of target Super Nodes, but he cannot control a large quantity of Super Nodes to hold more than two thirds of total TTT in circulation.

- **Robustness**

Beyond Security goals, TrustME-BA has assumptions on network reachability to rigorously determine the priorities among Attestor Nodes. This goal is that all Super Nodes can reach a consensus on a new set of Attestor Nodes selected within one (1) minute. To be robust, TrustME-BA makes a strong synchronization assumption that all honest Super Nodes send messages to most of other honest Nodes within a known time frame. This assumption acknowledges that an attacker may control some of the honest Super Nodes, but he cannot control the entire network in a large scale nor divide the network.

4.7.2 Final Consensus and Tentative Consensus

TrustME-BA has two types of consensus status: final consensus and tentative consensus.

When a Super Node reaches final consensus, it means that any other Super Nodes also reach final consensus, or Super Nodes in the same round must agree on the same consensus result (tentative consensus), regardless of the strong synchronization assumption. Tentative consensus means that some Super Nodes may have reached a tentative consensus on other Attestor Units, and no Super Node has reached the final consensus. All Attestor Units must directly or indirectly reference the Attestor Units that were generated before, which ensures the security of TrustME-BA.

There are 2 cases where TrustME-BA may reach tentative consensus. In the first case, if the network is strongly synchronized, an attacker may, with small probability, let TrustME-BA reach tentative consensus. In this case, TrustME-BA will not reach final consensus, and will not confirm that the network has strong synchronization. But after a few rounds, it is highly probable that the final consensus will be reached. In the second case, if the network is weakly synchronized and the entire network is compromised by the attacker, in such case TrustME-BA can reach tentative consensus and elects different sets of Attestor Nodes, multiple consensus forks are formed. This will prevent TrustME-BA from reaching final consensus, because the Super Nodes are divided into different groups, and the groups do not agree with each other. To regain activity, TrustME-BA will be executed periodically until the disagreement is resolved. Once the network returns to strong synchronization status, final consensus will be reached in a short period of time.

4.7.3 Lottery Algorithm

The lottery algorithm is constructed on the basis of a Verifiable Random Function (VRF) that selects a random subset of these Nodes based on the weightings of each Super Node participating in the TrustME-BA consensus. The probability of a Super Node being selected is approximately the same as the ratio of its own weighting to total weighting. The randomness of the lottery comes from the VRF and a publicly verifiable random seed. Each Super Node can verify whether it is selected using the random seed.

Definition of VRF: Given an arbitrary string, the VRF outputs the hash value and the result of the proof.

$$\left\langle hash,\pi\right\rangle \leftarrow VRF_{sk}\left(seed\,||\,role\right)$$

The hash value *hash* is uniquely determined by the private key *sk* and the given string (*seed*||*role*), *hash* is not distinguishable from a random number without knowing the *sk*. The result of the proof *π* enables those Nodes who know the public key corresponding to the *sk* can verify whether *hash* is associated with *seed* or not. *seed* is randomly selected and publicly available, the seed of each round is generated from the seed of previous round. The lottery algorithm supports role assignment, such as selecting participants at a certain point during the consensus process.

All Super Nodes execute the lottery algorithm to determine whether they are authorized attestors. The selected Super Nodes broadcast their lottery results to other Super Nodes through the P2P network. Note in order to defend against a Sybil attack, the probability of selecting a Super Node by lottery is directly proportional to the Super Node's own weighting. A Super Node with a high weighting may be selected multiple times, for which the lottery algorithm will report the number of the Super Node been selected. If a Super Node is selected multiple times, it will be treated as multiple different Super Nodes.

4.7.4 Byzantine Agreement

The Byzantine Agreement (BA) negotiates and decides the attesting priority for each selected Super Node and provides such proof. There are several steps that need to be taken to reach an agreement and the BA algorithm will be executed multiple times. Each negotiation starts with a lottery and all Super Nodes check if they are selected to participate in the current BA; the participants broadcast a message containing the choice of attesting priority; then each Super Node initializes the BA algorithm with the set of Attestor Nodes they collected. These steps are repeated until there are enough participants to reach consensus at a given step. The above steps are not synchronized among the Super Nodes, and each Super Node immediately checks the result of the new participant’s selection after the previous steps end.

An important feature of the BA algorithm is that participants do not maintain its private state except saving the private keys, therefore, the participants can be replaced after each step to reduce the attack to the participants. When the network is strongly synchronized, the BA algorithm ensures that if all the honest Super Nodes are initialized with the same content, final consensus can be reached with very few steps. In the case of a strongly synchronized network, if there is a small number of attackers, all honest users can still reach final consensus within limited steps.

5. SMART CONTRACT

TrustNote has non-Turing-complete declarative Smart Contracts designed to interpret the expectations of the contracts, to support Boolean operations while increasing the support for variable operations and contract data access. It does not support stacks and jump instructions, so that not only retains the benefits of a declarative contracting language such as being easy to understand, and having strong security, but also enhances the expression of the contracting language. TrustNote also improves the storage capabilities for Smart Contracts' internal data, hence greatly improving the support to complex application scenarios. Comparing with Turing-complete Smart Contracts (e.g. Solidity), TrustNote enjoys the advantages of low complexity, light weight and high performance, while making it easier to program with less probability of making errors.

There is no "account" in TrustNote, TTT is stored in the form of Unspent Transaction Output (UTXO) at the address of a tamper-resistant distributed ledger. In the TrustNote Smart Contract language, an address definition is an expression that evaluates to "true" or "false". If the signature provided by the transaction is valid and generated by the private key corresponding to this public key, the result of this expression evaluates to "true". All expressions in a Smart Contract eventually result in a Boolean value, and multiple Boolean expressions can be combined using Boolean operations.

For example, the following definition requires two signatures:

```
 ["and", [
  ["sig", {pubkey: "one pubkey"}],
  ["sig", {pubkey: "another pubkey"}]
]]
```

To spend funds from the address equal to the hash of the above definition, two signatures must be provided. As you have noticed, we use JSON to construct the language expressions, this allows us to use existing, well-supported, well-optimized JSON parsers without having to create a new one.

The "Or" operation can be used to request the signature from one of the listed public keys.

```
["or", [
  ["sig", {pubkey: "laptop pubkey"}],
  ["sig", {pubkey: "smartphone pubkey"}],
  ["sig", {pubkey: "tablet pubkey"}]
]]
```

The above expression is useful when you want to control 3 devices from the same address, these devices may be your computer, cell phone, and tablet.

Operations can be nested:

```
["and", [
  ["or", [
    ["sig", {pubkey: "laptop pubkey"}],
    ["sig", {pubkey: "tablet pubkey"}]
  ]],
  ["sig", {pubkey: "smartphone pubkey"}]
]]
```

A definition can require that a minimum number of members in a large collection must be true, see the 2-of-3 signature below for example:

```
["r of set", {
  required: 2,
  set: [
    ["sig", {pubkey: "laptop pubkey"}],
    ["sig", {pubkey: "smartphone pubkey"}],
    ["sig", {pubkey: "tablet pubkey"}]
  ]
}]
```

The above expression implies the security and reliability requirement of any two signatures. If one key is missing, the address is still usable, and the definition can still be modified to set a new value for the missing third key.

Also, different entries can be given different weightings, of which a minimum requirement is set:

```
 ["weighted and", {
  required: 50,
  set: [
    {weight: 40, value: ["sig", {pubkey: "CEO pubkey"}] },
    {weight: 20, value: ["sig", {pubkey: "COO pubkey"}] },
    {weight: 20, value: ["sig", {pubkey: "CFO pubkey"}] },
    {weight: 20, value: ["sig", {pubkey: "CTO pubkey"}] }
  ]
}]
```

A definition can reference other addresses:

```
["and", [
  ["address", "ADDRESS 1 "],
  ["address", "ADDRESS 2"]
]]
```

Delegates signing to other addresses is useful for building shared control address (addresses controlled by multiple users). This syntax gives users the flexibility of changing the definition of their own address, without bothering other users.

A sub-definition enables transactions to be jointly signed by other addresses.

```
["cosigned by", "ANOTHER ADDRESS"]
```

A very useful operation can be used to query the data previously stored in TrustNote:

```
["in data feed", [
  ["ADDRESS1", "ADDRESS2", …],
  "data feed name",
  "=",
  "expected value"
]]
```

If there is at least one message that is already stored in the database, and has "data feed name" equal to "expected value", the calculation results of the operation are “true”. The data feed sent to the distributed database must be from a trusted third-party data source (oracle) whose addresses are "ADDRESS1", "ADDRESS2", ... The oracle on the chain is a very powerful thing and we call them “on-chain oracles”.

For example, this address definition represents a binary option.

```
["or", [
 ["and", [
   ["address", "ADDRESS 1"],
   ["in data feed", [["EXCHANGE ADDRESS"], ["EURUSD", "+", "0.200"], ">", "1.1500"]]
 ]],
 ["and", [
   ["address", "ADDRESS 2"],
   ["in data feed", [["TIMESTAMPER ADDRESS"], "datetime", ">", "2016-10-01 00:00:00"]]
 ]]
]]
```

The above expression relies upon two oracles, one is the euro/dollar exchange rate, the other is the release time. Initially, both parties prepare funds for the addresses defined by this expression and provide them with their respective share of funds; then, if the euro/dollar exchange rate announced by the exchange address plus 0.200 ever exceeded 1.150, address 1 takes away all the funds. Before 1st October 2016, and after the timestamp issued by the oracle, if the above-mentioned condition does not happen, address 2 takes away the entire funds.

In another example, a consumer buys goods from an online merchant who they didn’t trust, if the goods are not sent to him and he wants a refund, the consumer can pay the money to a shared address defined as follows:

```
["or", [
 ["and", [
   ["address", "MERCHANT ADDRESS"],
   ["in data feed", [["FEDEX ADDRESS"], "tracking", "=", "123456"]]
 ]],
 ["and", [
   ["address", "BUYER ADDRESS"],
   ["in data feed", [["TIMESTAMPER ADDRESS"], "datetime", ">", "2016-10-01 00:00:00"]]
 ]]
]]
```

This definition relies on all tracking numbers issued by the FedEx oracle. If the goods are dispatched, the merchant can unlock the funds according to article 1. If the goods are not dispatched before the agreed date, the consumer can take back his money.

A definition can also include enquires to the transaction itself. For example, it can be used to program restricted orders on a distributed market. Assumes a user want to buy 1200 units of asset, but he won't pay anything more than 1000 Notes (tokens), and he don't want to be sitting online waiting for the seller. What he would like to do is simply post an order onto the market so it will get executed when the matching seller arrives. He can create a restricted order, sending 1000 Notes to the address defined by the following expression:

```
["or", [
  ["address", "USER ADDRESS"],
  ["and", [
    ["address", "EXCHANGE ADDRESS"],
    ["has", {
      what: "output",
      asset: "ID of alternative asset",
      amount_at_least: 1200,
      address: "USER ADDRESS"
    }]
  ]]
]]
```

The first "or" alternative allows the user to get back his Note and cancel his order whenever he wants. The second alternative commissions the market and authorizes it to spend money, provides another output on the same transaction to pay least money for 1,200 units alternative asset to the user's address. The market will publish the list of orders, allowing sellers to discover the list of orders, make a transaction that can exchange the assets, and co-sign with the market.

6. ECOSYSTEM AND APPLICATIONS

Issuing and trading tokens are one of the key applications in blockchain technology, but from a practical point of view, trading any tokens must be supported by low-level wallet software, and the wallet software must be upgraded when a new type of token is issued, or new transaction types are added, because the old wallet may not support new tokens or new features. Ethereum is a very good platform which supports issuing new tokens, defining transaction types, etc., however writing Smart Contracts for Ethereum requires specialized expertise which is not easy, Turing-complete Smart Contracts appear to be too heavy and error-prone for many simple asset-based applications such as issuing and trading tokens. "The DAO Attack" incident is not by accident, the security of Smart Contracts and the ease of writing applications need more attention, the recent “CryptoKitties congestion” incident also echo the evidence that Ethereum's transaction performance needs to be further improved.

The TrustNote Development Team is committed to moving forward along the roadmap established in the early stages of the project. Driven by technological innovation and supported by the open-source community, TrustNote will keep focus on creating an easy-to-use, decentralized, low-level blockchain allowing new innovative ideas to run smoothly on the blockchain network, making user friendly blockchain applications accessible to everyone.

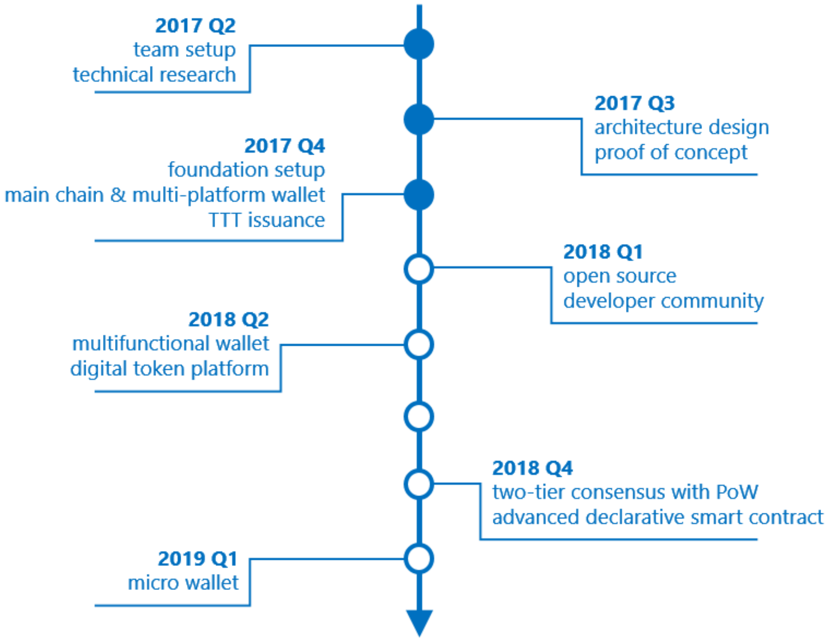


Figure 6-1 Roadmap

- Main Chain and the Multi-Platform Wallet: Build the Main Chain, develop basic version of wallet and release it to the public, and token issuance;

- Open Source: TrustNote is an open-source project and will be hosted on Github. The first

Github release is scheduled on Q1 2018. A development team will be set up including TrustNote employees and community developers. Every developer can contribute to the project and the changes will be submitted by the development team upon code review;

- Developer Community: Communicate with developers, receive questions and bug reports, publish software releases, technical articles and roadmaps, integrate the most advanced blockchain technologies by interacting with the world blockchain development community;
- Wallet Evolution:
  - Multi-platform: Use cross-platform language Node.js for development, support multiple platforms such as Windows, MacOS, Linux, Android, iOS, Chrome, Firefox, ensure safety usage of wallet inside web browser;
  - Micro-wallet: TrustNote team is committed to develop the wallet for Internet-of-Things (IoT) including micro-wallet protocols and micro-wallet application clients for IoT devices. Rust, a concurrent and efficient system programming language is selected to develop micro-wallet protocols and micro-wallet clients. It could fundamentally resolve the problems exists on IoT devices, such as lack of computing and storage resources, and inability to complete autonomous value exchange between devices etc., to ultimately empower IoT devices with blockchain technology.
- Crypto-Token Platform:
  - Customized token issue, distribute, trade, destroy, manages the entire life cycle of crypto-token;
  - Decentralized Token exchange, registration free, supports anonymous transactions;
  - Support Bitcoin, Ethereum, Litecoin, Dash and many other mainstream crypto-currency wallets;
  - Unify Token Wallet (TrustNote wallet) manages user issued assets, the transaction consumes TTT, the actual transaction costs equal to the bytes the transaction consumed;
  - Two-layer consensus mechanism ensures security while supports fast transaction confirmation. The higher the concurrency is, the shorter the time of transaction confirmation is;
  - Support third-party application extensions, asset-based application scenarios can be created using Smart Contracts and data provided by trusted third-party. App extensions can be downloaded from the wallet application market.

The crypto-token platform focuses on creating a decentralized platform for creating, issuing and operating crypto-tokens. Users can easily customize their personal crypto-token and launch crowdfunding or ICOs without the need to write sophisticated Smart Contracts, thus democratizing the access to the crypto-token issuance world. Users can also securely manage Bitcoin, Ethereum, Litecoin and other mainstream crypto-token wallets through the corresponding crypto-token gateways. A unified entrance for crypto-token wallet software across operating systems and platforms are provided, and there is no need to switch between wallet software anymore.

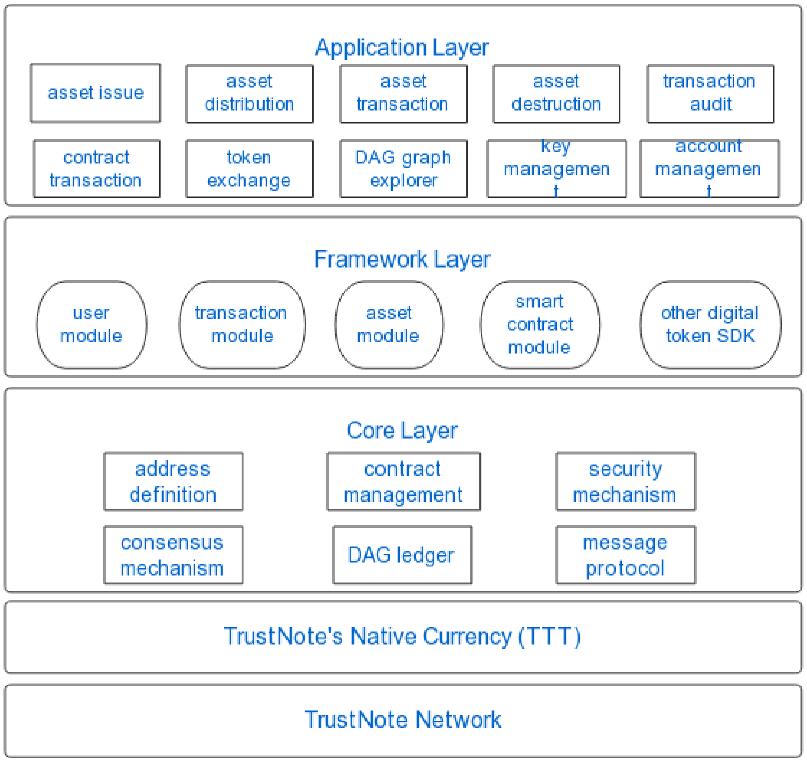


Figure 6-2 Architecture of Crypto-Token Platform

- Typical Application Scenario:
  - Virtual Assets: Game equipment, live-streaming reward;
  - Conditional Payment: Knowledge payment, API calls, decentralized insurance;
  - Cryptographic Matching: Predictive market, Charity, etc.;
  - Off-Exchange Trading (Over-the-Counter): Token Exchange;
  - Social Trading: Group red envelopes, group collection;
  - Resource Sharing: Make full use of idle resources.

7. ISSUANCE AND DISTRIBUTION

- TrustNote's crypto-currency is called "TTT", and "Note" is the unit of TTT, often specified in Mega Notes (MN);
- Total Issuance: 1,000,000,000 (10^9) MN, fixed supply;
- Initial Offering: 500,000,000 (5×10^8, 50% ratio) MN, distributed by means of "coin to coin" exchange;
- Total Attestation Rewards: 500,000,000 (5×10^8, 50% ratio) MN, Attestation Rewards are available for miners who participate in the TrustME consensus;
- The Main Chain (MC) supporting PoW mining is expected to launch on Q4 2018, while users can download the mining client and apply to become a Super Node, Super Nodes can get attestation authority by participating in the MC consensus and then get Attestation Rewards by issuing valid Attestor Units;
- The Attestation Reward Policy allocates 6.79% of Total Attestation Rewards for the first year, after which the yearly allocated Attestation Rewards decays year by year according to Figure 6-1. Of these, 90% of Attestation Rewards are allocated to the Super Nodes who provide valid Attestor Units, 10% of Attestation Rewards are allocated to TrustNote Foundation to support community operations, project incubation, and rewards to contributors etc.

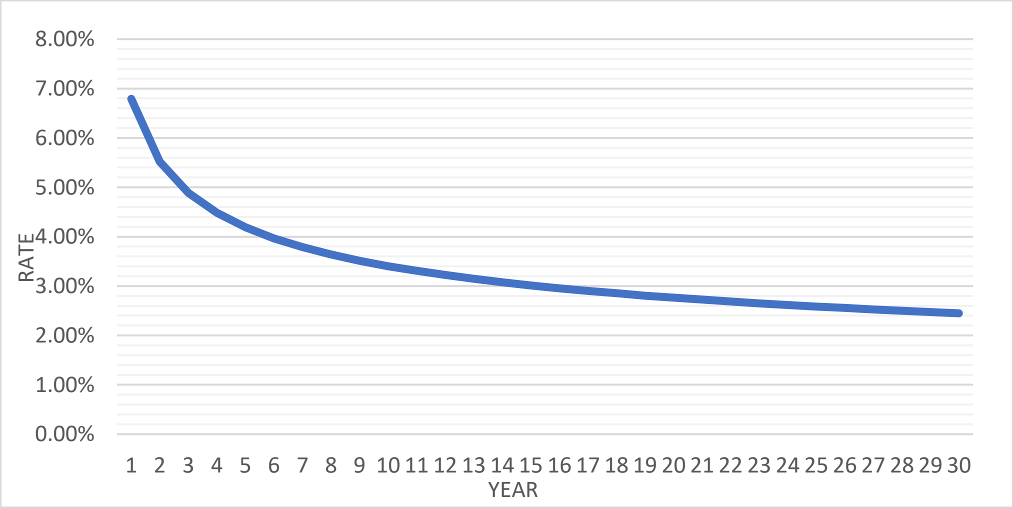


Figure 7-1 Attenuation Chart of Attestation Rewards

- TrustME-PoW reaches consensus at about every 5 minutes per round, with about 100,000 rounds each year, the total amount of Attestation Rewards allocated for each round equals

Attestation Rewards * 90% + Attestation Fees.

- In 1st year, Attestation Reward for each round is about 323.04 MN;
- In 2nd year, Attestation Reward for each round is about 262.39 MN;
- In 3rd year, Attestation Reward for each round is about 232.34 MN.

8. References

[1] Bitcoin Computation Waste, http://gizmodo.com/the-worlds-most-powerful-computer -network-is-being-was-50403276. 2013.

[2] Bitcoin wiki. Proof of Stake. http://www.blockchaintechnologies.com/blockchain-applications. As of 11 Aug 2017.

[3] Coindesk.com. Bitcoin: A Peer-to-Peer Electronic Cash System.

[4] http://www.coindesk.com/ibm-reveals-proof-concept-blockchain-powered-internet -things/ As of 11 Nov 2017.

[5] Ethereum. Ethereum. https://github.com/ethereum/. As of 12 Nov 2017.

[6] IOTA. IOTA. https://github.com/iotaledger/. As of 10 Nov 2017.

[7] Byteball. Byteball. https://github.com/byteball/. As of 10 Sep 2017.

[8] Bernstein, Daniel J, et al. High-speed high-security signatures. Journal of Cryptographic Engineering 2.2(2012), 77-89.

[9] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, Louisiana, USA,1999, pp. 173–186.

[10] Biryukov, Alex, and D. Khovratovich. Equihash: Asymmetric Proof-of-Work Based on the Generalized Birthday Problem. Network and Distributed System Security Symposium 2016.

[11] Gilad Y, Hemo R, Micali S, et al. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. The Symposium 2017, 51-68.

[12] C. Decker and R. Wattenhofer. Information Propagation in the Bitcoin Network. 13-th IEEE Conference on Peer-to-Peer Computing, 2013.

[13] D. Dolev and H.R. Strong. Authenticated algorithms for Byzantine agreement. SIAM Journal on Computing 12 (4), 656-666.

[14] A. Kiayias, A. Russel, B. David, and R. Oliynycov.. Ouroburos: A provably secure proof-of-stake protocol. Cryptology ePrint Archive, Report 2016/889, 2016. http://eprint.iacr.org /2016/889.

[15] S. King and S. Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake, 2012.

[16] S. Micali, M. Rabin and S. Vadhan. Verifiable Random Functions. 40th Foundations of Computer Science (FOCS), New York, Oct 1999.