THE BTN TECHNICAL WHITEPAPER

[Version 1.0]

March 3, 2020

The BTN Team

Contents

Abstract

BTN has created a **POE (Proof of Existence)** consensus algorithm mechanism in the world, innovatively developing **Asymmetric secret key drawing, Multiple level seals** and other core solutions. Taking asymmetric encryption algorithm and IBC identity cryptographic algorithm as a mixed underlayer, a decentralized blockchain trusted platform which realizes high-performance concurrence and unlimited expansion in the complete smart contract is formed through **Non-Interactive Zero-Knowledge (NIZK)**and **Trusted Platform Module(TPM)**. It integrates the hardware scheme for trusted computing chips. In the OSI (Open System Interconnect), IBC (Identity-Based Cryptograph) is adopted for point-to-point trusted identity authentication and digital asset transaction in IBS (Identity-Based SignCrypt). The mutual authentication and key negotiation protocol within and between domains based on CL-PKC is guaranteed with flexible security and privacy.

The content of this white paper is version 1.0, which only covers core consensus algorithm, system architecture and organization mechanism. Other network designs, technologies, systems and applications will be gradually updated in later versions. In addition to blockchain applications, BTN has the characteristics of absorbing secondary nodes, by which we can build content, advertising, distribution, private cloud, edge computing, distributed computing and storage ecosystems.

1 Introduction: Blockchain and BTN core values

Under the profound impact of digital transformation, a parallel digital world has become more and more similar to our physical world. We are in a great era of migrating from the physical world to the digital world! In the wave of digital migration, blockchain technology is more like a catalyst, which plays a crucial role in the digital transformation. As a trust protocol, it can be regarded as a link between the physical world and the digital world. With the characteristics of its trust, distribution and value, blockchain is used for identification and recognition of connected physical objects as well as digital representation. In fact, blockchain is more like the trust game of key and lock in the digital world, and our existence is being digitalized unconsciously.

Bitcoin, which appeared in 2009, is a great invention. It has successfully launched a social experiment with its ingenious technical algorithms and codes, mapping the value of money into the digital world. With the explosive growth of system decentralization, it also brings about computing power monopolization and mining pooling. A few people become rich in this barbaric new world, while most people have nothing to do with it. Many people marvel that Bitcoin is a miracle, but the inequity of the system ecology caused by its computing power competition is becoming more and more obvious. After more than ten years of development, the barbaric empire of computing power has risen, and computing power has become more and more concentrated. The system that should have been decentralized is turned out to be the one under the control of one or several player(s). The blockchain network is manipulated at will, which goes against the original intention of the decentralization of the blockchain. Meanwhile, the continuous forking events are terrible for the capital system of hundreds of billions of dollars.

Bitcoin appeared earlier than the mobile internet, nodes were designed as PCs, and all nodes competed equally with computing power. However, mobile internet and network devices featured diversification and mobile portability, and network infrastructure also made great changes. Blockchain technology has been innovated continuously, and smart contract ETH is adopted. Although it launched the peak of currency issuance, only over 1,000 TPS appeared and serious network congestion was caused. This is far beyond the commercial standard. In the innovation of later mainstream consensus algorithms, POS and DPOS seem to meet the performance standards, but it is difficult to get rid of the suspicion of the monopolization by a few nodes.

Blockchain elites are always searching for solutions in “the Impossible Triangle” of performance, decentralization and expandability. How to let the general public get involved in it, have it applied and share the dividends brought by blockchain will indicate the arrival of the golden age of blockchain.

As a new technology system, blockchain is valued for achieving such goals as decentralization and non-tampering by smartly using computers, networks and mathematical methods. It dramatically reduces the non-transparency and unreliability of the whole society, which greatly reduces transaction cost, changes the social trust system and strongly affects the existing finance, business, intermediary agents and fairness supervision systems. This will involve cash, banks, insurance companies, securities traders, exchanges, international and domestic trade, commercial contracts, share certificates, various certificates (real estate certificate, diploma, degree certificate, driving certificate, etc.), product certification, copyright registration and other aspects of our life.

Blockchain, as a new thing which is complex and not easily understood but closely related to finance and our daily life, has attracted an extensive attention and even been used as a tool for underground finance and

network sales. Blockchain has financial attributes such as financing tools and trading tools, but at the initial stage of barbaric growth, it lacks supervision and thus causes many problems. However, this does not mean that blockchain itself is problematic. Blockchain is an infrastructure technology similar to water, electricity, telephone and internet. Compared with the financial industry, this characteristic is particularly obvious. Because of these characteristics, the regulatory system is essential. As the regulatory system becomes mature, our understanding and applications of blockchain become mature. Blockchain will eventually become one of the infrastructures of the whole society.

We hold that to become an ideal general infrastructure system for blockchain, the following requirements shall be met:

1. There are more than a million TPS with high concurrency and low latency.
2. There should be no abnormal hashrate competition, and zero threshold involvement should be guaranteed.
3. Besides communication bookkeeping, network nodes can reasonably utilize resources and share bandwidth and storage.
4. The system can be expanded easily and infinitely.
5. There should be extensive decentralization, and there should be no monopolization by mining monopolist.

In response to the above requirements, the team of BTN has made its long-term efforts to redesign the blockchain structure and various underlying protocols. BTN has created a POE (Proof of Existence) consensus algorithm mechanism in the world, innovatively developing Asymmetric secret key drawing, multiple level seals and other core solutions. Taking RSA cryptographic algorithm and IBC identity cryptographic algorithm as a mixed underlayer, a blockchain trusted platform which realizes high-performance concurrence and unlimited expansion in the complete Turing smart contract is formed through Non-Interactive Zero-Knowledge (NIZK) and TPM Intelligent Hardware Seal System. It aims at providing a convenient, safe and trusted blockchain application gateway for different roles and institutions on the network and connecting various trust value islands to the value network.

"BTN" is a trusted blockchain network based on decentralization, which integrates the hardware scheme for trusted computing chips. In the OSI (Open System Interconnect), IBC (Identity-Based Cryptograph) is adopted for point-to-point trusted identity authentication and digital asset transaction in IBS (Identity-Based SignCrypt). The mutual authentication and key negotiation protocol within and between domains based on CL-PKC is guaranteed with flexible security and privacy.

Blockchain mainly solves the problem of trust cost, and it is difficult to create a burst application relying on blockchain technology alone. Blockchain itself is not a technology that can directly empower people. The real breakthrough in funds and applications should be a combination of blockchain, AI and IOT. This is also the vision of BTN.

The aim of BTN is to effectively solve the performance and reliability problems faced by blockchain in its practical applications by using its own core technologies and market methods. Specific methods will be described in the next section.

The profitability of an enterprise comes from its capability to solve problems and its position in people's minds (i.e. brand value). The method invented by BTN is suitable for large scale. On the one hand, it provides better reliability and disaster recovery for BTN. On the other hand, it gets rid of the problems of pooling and monopolization of the existing blockchain technology which are criticized by people. It can enter into ordinary families easily, so it is abler to seize people's cognitive resources. With its growth and increasing applications, BTN will become the first choice to carry all kinds of credit applications, just as Google are the first choices for people to search, and Facebook are the first choices for people to communicate. BTN will become a representative for credit application.

2 POE consensus algorithm

2.1 Starting from the problem: The conflict between performance and decentralization

Blockchain technology provides a simple and reliable trust mechanism, which is a great progress. Trust and verification mechanism is an important part of the human society. When we choose creditworthy products, banks, hotels and insurance companies, trust is also an important factor besides products and services, and we also pay a premium for it. In some fields, the transparency is very low for ordinary people due to their high monopolization or professionalism, and the trust cost will be greater. Blockchain technology is very complex, but its algorithm is open, and professionals can check it. Therefore, this is a great progress.

However, this technology is not perfect. There is a conflict between performance and decentralization. For the systems like Bitcoin and ETH, the more nodes, the more bookkeepers and the more reliable the accounts. In turn, the more nodes, the more communications and the less processing capacity.

This can be regarded as one of the core problems from blockchain technology at present. The problem is a challenge, but it is also an opportunity. Let's start from this problem.

2.2 Technical methods: Detailed explanation of core technologies

How can we solve these problems?

- Keep decentralized and don't be monopolized or controlled by the miners. Nodes can be expanded infinitely to let more people join us and make our system more reliable.
- Maintain good performance and keep the processing capacity strong regardless of excessive nodes.

Our core methods include drawing, two-level mining and extensive validation. Through the combination of drawing and two-level mining, the system will obtain an unlimited expansion capacity and maintain high processing performance. The following diagram summarizes the classification mechanism and the operation process of the system. Next, we will explain the key points of the system in detail.

2.2.1 Two-level system

In the system of Bitcoin and ETH, all nodes are equal and have a capacity of point-to-point communication. All transactions are broadcast directly between nodes. All nodes have a capability to generate blocks. All nodes compete with each other equally, and the final bookkeeping right is obtained through POW. This mechanism is a great invention, in which each node can reach an agreement after independent operation without a voting process. However, it has the following problems:

- POW itself does not make to much sense. It is just a meaningless operation to get a specific hash value. This is criticized by people for Bitcoin and ETH.

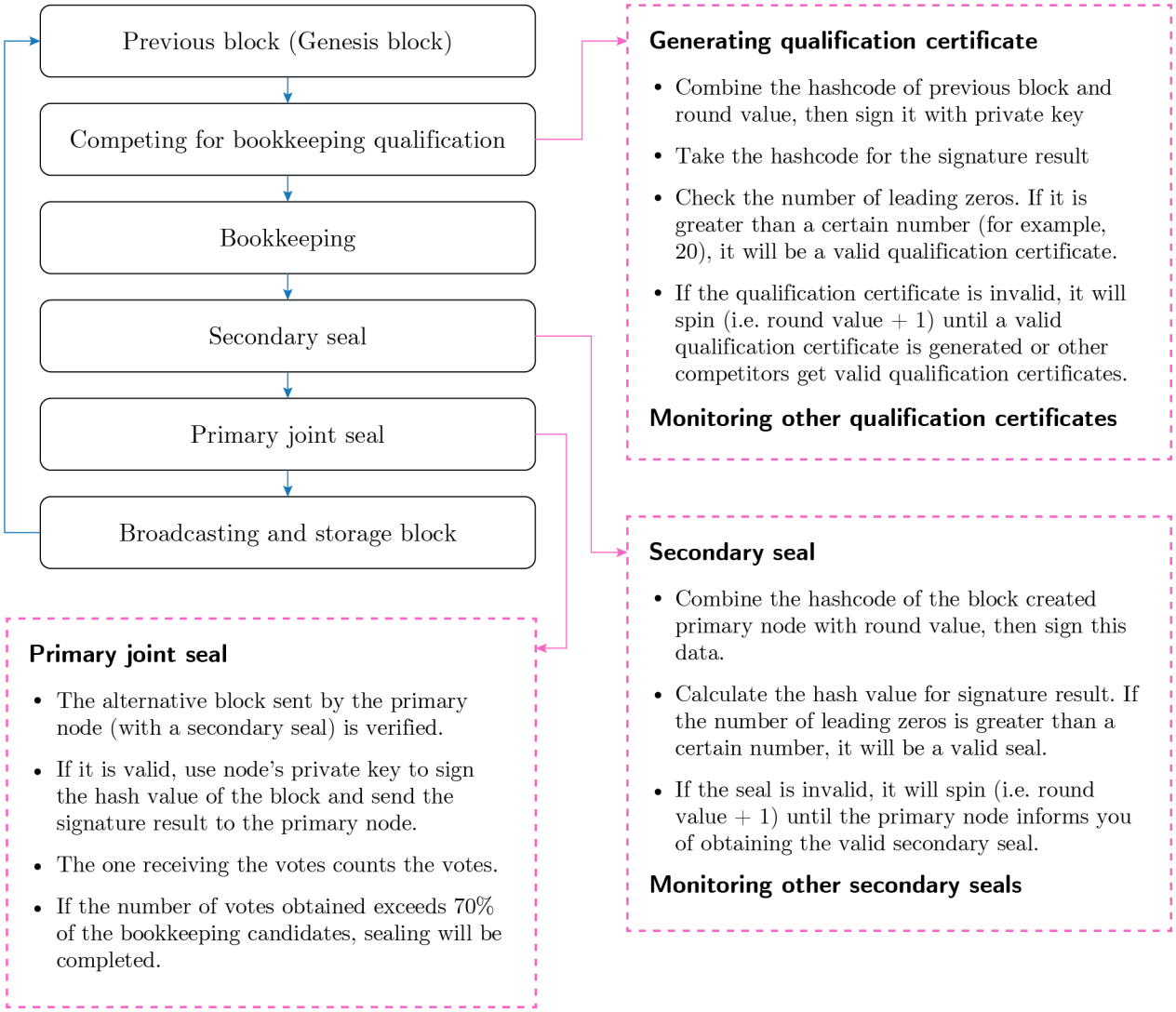


Figure 1: Technical methods

- POW leads to the appearance of speculators, namely, mining rigs. A mining rig focuses on POW operation and does nothing else, which has squeezed out PC and other general-purpose computing devices.
- Due to the limitations of routers and firewalls, nodes have to be deployed in the IDC, which leads to the formation of the mining pool.

We adopt a two-level mechanism to avoid these problems. This grading mechanism is different from Bitcoin, ETH, EOS and Algorand. Primary node is a public network node which mainly undertakes bookkeeping and communication. Secondary node can be placed at home and is used to witness and test the bookkeeping results of the primary node.

2.2.2 Competition among alternative bookkeepers

A. Drawing algorithm

Bitcoin and ETH adopt P2P technology. If a transaction occurs, it needs to be broadcast to the whole network. It is actually unnecessary to do so. EOS adopts some mechanisms to select specific nodes to process all transactions in a centralized way, which improves TPS rapidly. It also brings a problem. EOS selects bookkeeping nodes via POS, but it is suspected of monopolization. Only truly reliable random algorithm based on large cardinality can get rid of this suspension.

The method we adopt is to use a private key to sign the hash code, take the hash value (256 bits) and then determine the number of leading zeros. We can obtain the result we need only by simple mathematical derivation. The basis is as follows: if 1 leading zero is required and other bits are unlimited, the probability will be 0.5. If 2 leading zeros are required, the probability will be 0.25. In this similar way, if n leading zeros are required, the probability will be:

$$\frac{1}{2^n} \ (n = 1, 2, 3, ..., 256)$$

For example, if 20 leading zeros are required, the probability will be 1/1048576. According to this rule, we can select potential bookkeeping nodes, that is, we can adjust the number of leading zeros according to the total number of nodes in the whole system to determine how many nodes will be eligible for bookkeeping. If our system has a total of 1 million primary nodes and we need 20 leading zeros, there will be about 1 potential bookkeeping node, and so on:

If we only need 19 leading zeros, there will be about 2 potential bookkeeping nodes;

If we only need 18 leading zeros, there will be about 4 potential bookkeeping nodes;

If we only need 17 leading zeros, there will be about 8 potential bookkeeping nodes;

...

In this process, it is also an important step to use a private key to sign the hash code of the previous block. It is obviously unrealistic to sign the whole block with a private key. Only by signing hash code with a private key can we generate data that is only available for this node but related to the previous block. This can reliably add the information of the node to the history block, greatly reducing the possibility of being counterfeited.

After the hash value is encrypted with a private key, the non-fixed-length data will be obtained. Taking hash value for the data again, we can turn non-fixed-length data into fixed-length data again to simplify our mathematical calculation and form simple and consistent drawing rules.

Take the hash code (sha1 160 bit) of all numbers (converted into string) from 1 to 10 million as an example, the number of leading 0 is more than 20, only 7 numbers, respectively:

```
4233739
leading zero count=21
4836969
leading zero count=22
5370398
leading zero count=23
5871620
leading zero count=21
6132845
leading zero count=25
8520920
leading zero count=21
9057261
leading zero count=22
```

Part of the demo code as below:

```
fun hash(data:ByteArray):ByteArray{
  val md = MessageDigest.getInstance(SHA_KEY)
  val hash = md.digest(data)
  return hash
}

fun leadingZeroCount(hash: ByteArray): Int {
  var count = 0
  val zero:Byte = 0
  var curByte:UByte = 0xffu
  for ((index, b) in hash.withIndex()){
    if(b == zero)
      count += 8
    else{
      curByte = b.toUByte()
      break
    }
  }
  return count + curByte.countLeadingZeroBits()
}


fun sign(privBytes:ByteArray?,data:ByteArray):ByteArray{
  return rsaEncrypt(privBytes,data)
}

fun rsaEncrypt(privBytes:ByteArray?,data:ByteArray, keyIsPub:Boolean = false):
    ByteArray{
```

```
    val key = getRsaKey(privBytes,keyIsPub)
    val cipher = Cipher.getInstance(RSA_ALG)
    cipher.init(Cipher.ENCRYPT_MODE, key)
    return cipher.doFinal(data)
  }

  private fun getRsaKey(bytes:ByteArray?,isPub:Boolean): Key {
    lateinit var keySpec: EncodedKeySpec
    val keyFactory = KeyFactory.getInstance(RSA_KF)
    return if(isPub){
      keySpec = X509EncodedKeySpec(bytes)
      val k = keyFactory.generatePublic(keySpec)
      k
    }else{
      keySpec = PKCS8EncodedKeySpec(bytes)
      keyFactory.generatePrivate(keySpec)
    }
  }

  ...

  val signedData = sign(privateKey,bytes)
  val signedHash = hash(signedData)
  val zeros = leadingZeroCount (signedHash)
```

B. Competition among alternative bookkeepers

All the primary nodes encrypt the hash codes of the previous block with their own private keys as well as the result sha256. If the number of leading zeros meets the requirements (e.g. m zeros), it will be eligible for bookkeeping.

All the nodes eligible for bookkeeping broadcast to the whole network. All the nodes to be submitted for transaction and the good and evil of the bookkeeping nodes are verified. If verification is passed, the bookkeeping requests will be submitted to the bookkeeping nodes.

C. Spinning

As the drawing algorithm is probabilistic, it may not be able to generate legal candidates. At this moment, candidates can join the rounds and then resign for hash code until they can obtain the qualifications or receive better competition results.

The spinning method is applicable to primary and secondary nodes.

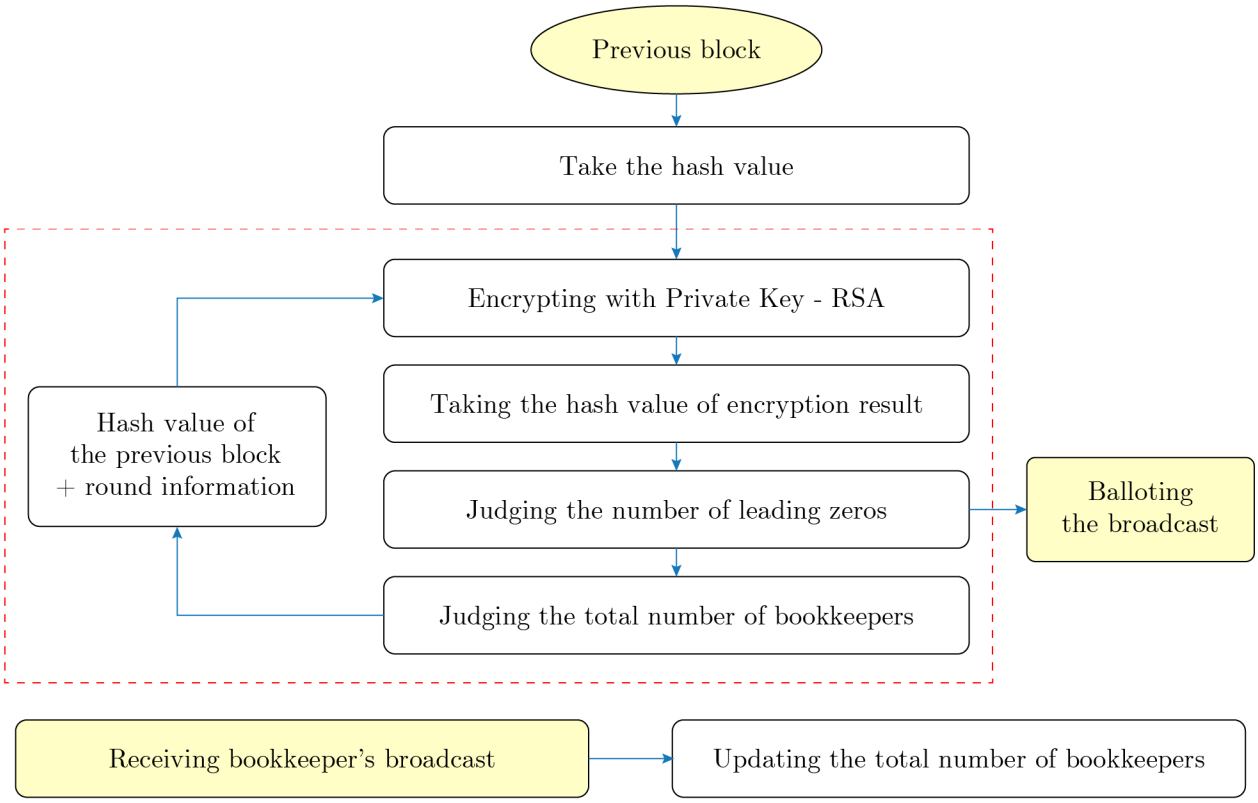


Figure 2: Drawing algorithm

Part of the demo code as below:

```
abstract class MasterNode:NodeInstance(){
    fun compete(block:Block){
        val hash = block.hashCode
        var round = 0
        while(true){
            val bytes = combineByteArray(hash,round)
            val signedData = sign(privateKey,bytes)
            val signedHash = hash(signedData)
            val zeros = leadingZeroCount(signedHash)

            if(btnNetwork.clerkIsReady)
                break

            if(zeros < btnNetwork.clerkZeros){
                round++
                continue
            }
```

```
18
19                  btnNetwork.broadcast(hash,signedData,publicKey)
20                  break
21              }
22          }
23      }
```

2.2.3 Block generation

A. First alternative state

After the primary node is bookkept, the block will enter the first alternative state.

B. Second alternative state: Secondary seal

The primary node broadcasts the block in the first state generated by itself to the whole network. The secondary node verifies the the first state candidate block. If the block is valid, the hash value of the candidate block will be encrypted with its own private key, and then the encryption result hash code will be taken. If the number of leading zeros meets the requirements (e.g. m zeros), the block will enter the second state.

The motivation for the primary node to give its own block to others for inspection comes from the quantity. The more people know and identify its blocks, the more likely it is to obtain a secondary seal.

Part of the demo code as below:

```
1   abstract class SupervisorNode:NodeInstance(){
2       fun sealBlock(block:Block){
3           if(!block.verify())
4               return
5
6           val signedData = sign(privateKey,block.hashCode)
7           val hash = hash(signedData)
8           val precedingZeroCount = leadingZeroCount (hash)
9           if(precedingZeroCount <= btnNetwork.sealPrecedingThreshold)
10              return
11
12          block.clerk.seal(signedData,publicKey)
13      }
14  }
15
16  abstract class MasterNode:NodeInstance(){
17  fun onBlockSealed(sealedBlock:Block){
```

```
18              btnNetwork.broadcastSealedBlock(sealedBlock)
19      }
20      }
```

C. Third state: Primary joint seal

All bookkeeping candidates are also the members of the voting committee. When a bookkeeping candidate gets a qualified sealed block in the second state, he will send the information to the voting committee. As every bookkeeping candidate will broadcast his bookkeeping qualification to the whole network, every node in the whole network has a list of bookkeeping committees maintained by itself.

After the voting committee (original bookkeeping candidates) receives enough blocks in the second state, it will calculate the priority of each block (comparing the number of leading zeros) and find the candidate block with the top priority. The voter sign the hash code of the block, and then sends the signed result and it's public key back to the bookkeeper of the block in the second state. The bookkeeper with a block in the second state counts how many votes (i.e. signatures) he has received. As the number of alternative bookkeepers is known in the whole network, the number of signatures will be known correspondingly and he can enter the third state.

This algorithm is more concise than PBFT and does not require complex point-to-point communications. Meanwhile, it can avoid long-term waiting for final transaction like Bitcoin.

2.2.4 Chain formation

After the block in the bookkeeping node enters the third state, the block will be broadcast immediately with all the votes this node got. All the primary nodes know who is the candidate bookkeeper, so they are able to judge whether a block can really enter the third state. When the primary node confirms that the block in the third state it receives is valid, it will start the next round of drawing, sign the hash code of the block with its private key and get the hash code of this result, by checking the count of leading zeros, the node can know whether it can become a new candidate bookkeeper. Thus, the linking process of a blockchain is completed.

2.3 Other key technologies

2.3.1 Programming languages and development platforms

BTN uses Kotlin as a programming language to compile into JVM bytecodes and run them on the Java virtual machine.

Java provides an excellent running platform, but the syntax has not been updated too much for a long time. Kotlin is a statically typed programming language for modern multiple platform applications. It is more secure than Java and can detect common traps statically. Meanwhile, it is more concise than Java and supports variable type information, high order functions (closures), extension function and first-class

delegation. This will make our development more secure and efficient. It will also facilitate others to join the BTN community, improve basic codes or expand BTN.

2.3.2 Network communication

BTN uses Netty as the basis of network communication. Netty is a NIO framework which is driven by high-performance and asynchronous events, providing a support for TCP and UDP. As an asynchronous NIO framework, all the IO operations of Netty are asynchronous and non-blocking. With the Future-Listener mechanism, users can easily obtain the IO operation results actively or through the notification mechanism.

As the most popular NIO framework, Netty has been widely applied in internet, big data distributed computing, game industry, communication industry, etc. Some famous open source components in the industry are also built based on Netty, such as RPC framework and zookeeper.

Netty's high performance, maturity, reliability and flexibility are an important guarantee for BTN to support tens of thousands of or even higher TPS.

Part of master node's code as below:

```
fun main() {
    val mBoss = NioEventLoopGroup(1)
    val mWorker = NioEventLoopGroup(20)

    val bs = ServerBootstrap()
    try {
        bs.group(mBoss, mWorker)
            .channel(NioServerSocketChannel::class.java)
            .option(ChannelOption.SO_BACKLOG, 128)
            .option(ChannelOption.SO_REUSEADDR, true)
            .option(ChannelOption.SO_REUSEADDR, true)
            .childOption(ChannelOption.SO_KEEPALIVE, true)
            .childHandler(SupervisorChannelInitializer())

        val f = bs.bind(PRIMARY_PORT).sync()
        if (f.isSuccess) {
            println("Primary node is started")
            f.channel().closeFuture().addListener {
                mWorker.shutdownGracefully()
                mBoss.shutdownGracefully()
            }
        } else {
            mWorker.shutdownGracefully()
            mBoss.shutdownGracefully()
        }
    } catch (e: Exception) {
```

```
            e.printStackTrace()
        }
    }
```

2.3.3 Communication protocol

BTN uses Json as the underlying communication protocol. The upper layer is encapsulated as a unified object-oriented command btncmd, which is encoded and decoded by btncmdencoder and btncmddecoder.

Part of the demo code as below:

```
    open class BtnCmd(open val body:String ){
        val cmd:String
        val param:JsonObject?
        val paramString:String?
        init {
            val index = body.indexOf("{")
            if(index == -1){
                cmd = body
                paramString = ""
                param = null
            }else{
                cmd = body.substring(0,index)
                paramString =  body.substring(index)
                param = try{
                    JsonParser().parse(paramString).asJsonObject
                }catch (e:Exception){
                    e.printStackTrace()
                    null
                }
            }
        }
        fun getStringParam(name:String):String?{
            return param?.get(name)?.asString
        }

        fun getLongParam(name:String):Long?{
            return param?.get(name)?.asLong
        }

        fun getIntParam(name:String):Int?{
            return param?.get(name)?.asInt
        }
```

```
33
34          fun getJsonParam(name:String):JsonObject?{
35              return param?.get(name)?.asJsonObject
36          }
37          fun getBooleanParam(name:String):Boolean?{
38              return param?.get(name)?.asBoolean
39          }
40
41          override fun toString(): String {
42              return "BtnCMD:[$cmd]$body"
43          }
44          fun getJsonArray(s: String): JsonArray? {
45              return param?.getAsJsonArray(s)
46          }
47
48          fun getReturnCode(): Int? {
49              return getIntParam("r")
50          }
51      }
52
53      open class BtnCmdEncoder: MessageToByteEncoder<BtnCmd>() {
54          init {
55              enclog.info("BtnCmd Encoder Created")
56          }
57          override fun acceptOutboundMessage(msg: Any?): Boolean {
58              return msg is BtnCmd
59          }
60          override fun encode(ctx: ChannelHandlerContext?, msg: BtnCmd?, out: ByteBuf?) {
61              enclog.info("try to encode message$msg")
62              if(out == null)
63                  return
64
65              if(msg?.body == null){
66                  enclog.info("read a message ,but body is null")
67                  return
68              }
69
70              enclog.info("Encoding btncmd ${msg.body}")
71              out.writeInt(BTN_CMD_FLAG)
72              val bytes = msg.body.toByteArray(Charsets.UTF_8)
73              out.writeInt(bytes.size)
74              out.writeBytes(bytes)
75              enclog.info("Encode end")
76          }
77      }
```

```
78
79  open class BtnCmdDecoder:ChannelInboundHandlerAdapter(){
80      private enum class State {
81          INIT,
82          READ_COMMAND
83      }
84
85      private var state = State.INIT
86      private var buffer:ByteBuf? = null
87      private var byteRead = 0
88      private var messageLen = 0
89      private var readCount = 0
90      private var cmdId = 0
91
92      init {
93          this.resetState()
94      }
95      private fun resetState(){
96          this.buffer?.release()
97          this.buffer = null
98          this.byteRead = 0
99          this.state = State.INIT
100         this.messageLen = 0
101         this.readCount = 0
102     }
103
104     override fun channelRead(ctx: ChannelHandlerContext?, msg: Any?) {
105         if(ctx == null){
106             declog.info("read a message ,but context is null")
107             return
108         }
109
110         if(msg !is ByteBuf){
111             declog.info("read a message ,not byte buffer")
112             ctx.fireChannelRead(msg)
113             return
114         }
115
116         if(msg.refCnt() == 0){
117             declog.err("Command reference count = 0",Error("Wrong input message,
    Command is not retained."))
118             return
119         }
120
121         while (true) {
```

```
122                 if(state == State.INIT){
123                     if(msg.readableBytes() < 8){
124                         ctx.fireChannelRead(msg)
125                         return
126                     }
127 
128                     msg.markReaderIndex()
129                     declog.info("before try to read btn cmd flag")
130 
131                     val flag = try{
132                         msg.readInt()
133                     }catch (e:java.lang.Exception){
134                         declog.err("message=${msg.hashCode()} channelId=${ctx.channel().
        id()}")
135                         declog.err("decode err",e)
136                         msg.release()
137                         throw e
138                     }
139 
140                     declog.info("after read btn cmd flag")
141                     if(flag != BTN_CMD_FLAG){
142                         msg.resetReaderIndex()
143                         ctx.fireChannelRead(msg)
144                         return
145                     }
146 
147                     declog.info("will change btncmd read state")
148                     messageLen = msg.readInt()
149 
150                     state = State.READ_COMMAND
151                     cmdId++
152                     buffer = ctx.alloc().buffer(messageLen)
153                     if(buffer == null) {
154                         msg.release()
155                         throw Exception("Can not allocate buffer for BtnCmd")
156                     }
157 
158                     declog.info("Ready for a new btn cmd=$cmdId")
159                 }
160 
161                 var leftLen = messageLen - byteRead
162                 if(msg.readableBytes() <= leftLen)
163                     leftLen = msg.readableBytes()
164 
165                 readCount ++
```

```
166                 buffer?.writeBytes(msg,leftLen) //notice
167                 byteRead += leftLen
168                 if(byteRead >= messageLen){
169                     val btncmd = BtnCmd(buffer?.readCharSequence(messageLen,CHARSET).
        toString())
170                     declog.info("read btn cmd finished cmd=${btncmd.body} remained
        bytes=${msg.readableBytes()}")
171                     ctx.fireChannelRead(btncmd)
172                     this.resetState()
173                 }
174 
175                 if (msg.readableBytes() == 0) {
176                     msg.release()  //notice wrong btng, not fired, must be released
177                     return
178                 }else{
179                     //notice continue until no byte.
180                 }
181             }
182         }
183 
184     }
185 
186     class BtnCmdCodec : CombinedChannelDuplexHandler<BtnCmdDecoder, BtnCmdEncoder>() {
187         init {
188             init(BtnCmdDecoder(), BtnCmdEncoder())
189         }
190     }
```

2.3.4 Remote asynchronization

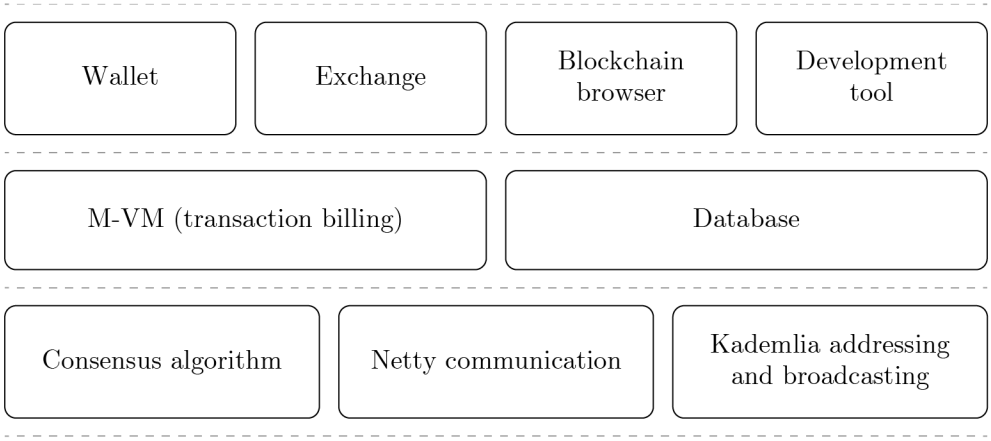
In the network and multithreading software, asynchronous call often brings many problems. Waiting for asynchronous results often requires polling, events and other means. This will make the codes become rather complicated and prone to errors. Based on the modern syntax characteristics and Netty provided by Kotlin, the Boss Chain team has designed a new remote call method supporting async /await.

Part demo code as below:

```
1     private val requestMap = HashMap<String, CompletableDeferred<Any?>>()
2 
3     suspend fun requestRemote(name:String, param:String? = null,paramIsString:Boolean =
          false):Any?{
4             if(globalCtx == null)
5                 throw Exception("Global context is null, can not send remote request")
```

```
6
7           val uuid = UUID.randomUUID().toString()
8           val deferred = CompletableDeferred<Any?>()
9           requestMap[uuid] = deferred
10          var param1 = param
11          if(param != null && paramIsString){
12              param1 = "\"$param\""
13          }
14
15          val remoteRequest = """${ServerUtil.REMOTE_REQUEST}{"reqid":"$uuid","name":"
        $name", "param":$param1 }"""
16          log.info("RemoteRequest=$remoteRequest")
17          sendMsCmdStr(globalCtx, remoteRequest)
18
19          val remoteResponse = withTimeoutOrNull(TimeUnit.SECONDS.toMillis(10L)) {
20              deferred.await()
21          }
22          requestMap.remove(uuid)
23          if(remoteResponse == null){
24              val remoteRequest1 = remoteRequest.replace("\"","'")
25              return jsonResultString(2,"Remote api timeout. Remote request=
        $remoteRequest1")
26          }
27
28          return remoteResponse
29      }
```

2.4 Overall architecture of BTN



3 Market Advantage

3.1 Number advantage of secondary nodes

With the two-level system, we can deploy a large number of secondary nodes without slowing down the performance of the whole network. It is easy for the mechanism of primary and secondary nodes to organize the sales force. The primary node is responsible for bookkeeping and transfer. It is placed in IDC and held by the institution or early initiators, and its service quality is guaranteed. The secondary node is cheap and can be held by ordinary family users, so it is easy to expand the number. The larger the number, the more validators in our system and the higher its credibility.

If we have 1,000 primary (bookkeeping and diversion service) nodes, there will be a total of 1 million secondary (verification and signature) nodes. If we have 100,000 primary nodes, there will be 100 million secondary nodes. There is no any other public chain can compare with this on this planet. Compared with the public blockchains like Bitcoin, ETH and EOS, the more nodes they have, the more complex their communication and the lower their processing capacity.

Our primary node is encouraged for expansion. The more nodes we have, the information we saved is more safe, more reliable. If there is a problem, the other nodes can be selected more easily to ensure the availability of the whole system. Due to the concepts of bookkeeping pool and proxy, the increase of the number of nodes will not lead to the decrease of processing rate. Any bookkeeping request received by a primary node will be recorded when two levels are available.

3.2 Every household device has a chance to generate the blocks

Mine field and mining pool are no longer required for mining, so every device has a chance to generate the blocks. We can use the private key of a single device to encrypt the current block and check whether the encrypted result is consistent with the eigenvalue. There can be countless devices competing for the generation of this block and the distributed concurrent operation. This is similar to the nonce in Bitcoin and can avoid the appearance of the mining rigs as speculators. As a node has a chance to obtain a seal, it does not depend on its specific operational capability but on its identity information. There is no significant difference between 100 operations and one operation within a second, for the priority of the seal in the low round is higher than that in the high round.

3.3 Summary of core advantages

Based on the technologies and market methods, the core advantages of BTN are as follows:

1. It is highly decentralized, and the quantity can be expanded infinitely.
2. High performance can be maintained.
3. Everyone can get involved in it. IDC mining rigs are not necessary any more.

4. peculators will not prefer to build mining pools and monopolists, since it is not as profitable as BTC or ETH.

4 Risks

4.1 Technical risks

4.1.1 Methods, possibilities and consequences of doing evil

It is very easy for any good node to judge whether the node to be submitted for transaction is good or evil. You do not need to ensure whether others are good or bad but just make sure that you are good.

The transaction initiator knows exactly about his position. The steps are as follows:

Steps	Bitcoin (primary node only)	BTN	Consequences, solutions and remarks
Initiate a request	The Full wallet submits the request to any primary node. It does not distinguish between good and evil nodes. Good node will broadcast transactions. Your transfer and balance will be obtained from your verified chain.	The bookkeeping request of the secondary node is directly sent to the bookkeeper. It can identify whether the bookkeeper is legal (based on block and signature).	Request initiation does not require a primary transfer node. Why is identification required comparing with Bitcoin? The reason is that we do not need to transfer it.
Transfer a transaction request	The node receiving the transaction may be an evil node and might not be transferred.	It is possible that the evil primary transfer node might not be transferred, and the influence is only limited to this.	The transaction is invalid and does not enter into the valid block.
Bookkeeping: generate a bad block.	Transfer others' money to yourself. It is created out of nothing. The commission charge is too high.	The same as the left column	These are illegal blocks and will not be authenticated, so they are meaningless.
Seal	POW sealing is conducted for the claimed bad block.	It is signed at the secondary node and voted by the voting committee. The signed secondary nodes and the voted primary nodes must be evil nodes, otherwise they cannot generate the blocks.	POW is not required, which is the core difference. Sealing is conducted based on identity.
SuBTNequent transactions	No good node will generate blocks based on this bad block.	No good node will generate blocks based on this.	

4.1.2 Pledge mechanism

In the future, pledge mechanism can be taken into consideration. Nodes compete for bookkeeping or sealing, but they require some pledge. The pledged amount should not be too large, otherwise monopolization will be caused easily. If nodes do evil, their pledged assets will be deducted so as to further ensure the reliability of the system.

4.1.3 Double spend

Voting is unanimous without double spend. Subsequent blocks are incapable of comeback.

4.1.4 Witch attack

If a large number of nodes are owned, witch attack (identity disguising) is conducted, and the probability of selecting malicious nodes will increase. Is this founded? If so, what can malicious nodes do? What is the impact on the system?

In fact, the Bitcoin system can also create the currency out of nothing as long as it continues to do evil. In other words, most nodes do evil. In this way, the whole system will lose its value and credibility.

An honest node can easily verify which chain is legal, for each block can be verified. Therefore, good nodes can continue to generate good chains. If a node is to be verified, we will only need to verify the block it currently receives. If those bad nodes do evil, they can continue to do evil in their bad chains. No matter how long or short a good chain is, it will always exist and grow. Good primary nodes will continue to generate the next block based on good chain and block.

4.2 Competition risk: Comparison with Bitcoin, ETH and EOS

	POE	Bitcoin/ETH(POW)	EOS(POS)
Bookkeeping rate	✓ The committee conducts bookkeeping, and TPS is close to EOS.	✗ Bookkeeping is conducted in the whole network.	✓ The committee conducts bookkeeping.
Forking	✓ None. It is solved by voting	6 blocks are confirmed.	✓ None, PBFT
Authenticity, reliability	✓ Based on random number, the larger the network is, the more authentic it is. It is the whole network verification.	✓ Based on POW, it is the whole network verification.	✗ Verification mechanism is non-transparent and controversial.
Network base and election system	✓ The bigger, the better; the faster, the more stable. Drawing is conducted with a reliable random mechanism.	✗ The bigger, the slower. Bookkeeping is conducted in the whole network. Hashrate competition is conducted in the whole network.	The election mechanism is not transparent, and POS cannot justify itself.
Participants	✓ It is primary and secondary networks, in which the latter is for home users.	✗ It cannot be used at home. P2P networking is not possible.	✗ P2P networking is not possible.
Monopolization	Identity authentication by a real person, TPM device	Mine field, mining pool	Being monopolized by the committee, it is not transparent.
Sales	Private placement for primary node, masses participation for secondary node	Mine field investment	Private placement, small scale
Broadcasting times	2 times (campaign and successful broadcasting)	Account and block broadcasting	No account broadcasting, block broadcasting

4.3 Organizational and capital risks

If BTN wants to succeed, it needs to be fully prepared. Once it is started , it needs to organize enough funds and market resources to quickly reach consensus ,form a certain scale and establish its own protection pattern. Once the leading position is formed, it will have a very strong self-growth force and continue to maintain its technical, scale and organization advantages.

5 Prospect

5.1 Complete application

The content in this file only covers core consensus algorithm, system architecture and organization mechanism but does not cover virtual machine, wallet, blockchain browser, exchange and other contents. These will be gradually improved with the growth of the ecosystem, and eventually Mass Chain will develop a set of complete and independent global computers that surpass ETH.

5.2 Ecosystem

Besides the applications of blockchain, BTN can absorb the secondary nodes and give us an opportunity to build an ecosystem. We can use blockchain as a breakthrough and cornerstone to build an ecosystem covering content, advertising, distribution, private cloud, edge computing, distributed computing and storage.

Our home computing devices can verify and seal the blocks in the primary nodes, and they can also be used as a normal intelligent terminal. As mentioned above, computing is not the core task of the secondary node. The secondary node has enough idle computing power and storage space for other applications, including but not limited to the following possibilities:

- Home intelligent terminal: such as private cloud, intelligent home host
- Content distribution platform: use secondary nodes to distribute film, television and other contents.
- Distributed computing: use secondary nodes for applications such as image recognition, big data storage and searching
- Distributed storage and bandwidth sharing: We have a large number of secondary and primary nodes which can carry some distributed storage and broadband sharing services without affecting the core businesses.
- Edge network: Our secondary node can be started within 24 hours with a storage function, which can serve as an edge storage device in the family, such as TV, mobile phone and pad.

If the network is large enough and our position is high enough, we will be able to virtually decompose several subnets, generate sub-chains to carry specific services and meet the needs of customers or industries.

6 Q&A

6.1 Quantum computing

Most of the existing public key cryptography algorithms (RSA, Diffie-Hellman, elliptic curve, etc.) can be broken through by quantum computers that are large and stable enough. The potential impact of quantum computing needs to be considered. Quantum computer is very powerful, but the premise of using its powerful computing power is that there should be a quantum algorithm which can solve the problems efficiently, otherwise quantum computer will be useless and bring disadvantages because of its high cost.

National Institute of Standards and Technology (NIST) started the research of post-quantum cryptography in 2012 and launched the global collection of post-quantum cryptography standards in February 2016. The post-quantum cryptography algorithms are mainly constructed by the following four mathematical methods:

- Lattice-based
- Code-based
- Multivariate-based
- Hash-based

After the new algorithm becomes an industrial standard, the RSA algorithm of the system can be replaced and all the historical data and business logic can be kept unchanged without changing the whole system architecture.

As for symmetric cryptography and hash functions (such as AES, SHA1, Sha2, etc.), although there are quantum algorithms that can be broken through in theory, the impact of this algorithm is limited and there are many restrictions. The famous quantum algorithm is Grover's algorithm in 1996. In this section, the length of key or hash can be doubled, for example, updating from AES-128 to AES-256, from SHA-256 to SHA-512, etc..

6.2 Witch attack and TPM

If the file certificate is used during the block sealing in the second alternative state, the evil node can seal the block with a large number of legal certificates collected by itself so as to obtain the sealing revenue. This is similar to witch attack.

TPM can be used to control this behavior. As a hardware, the possibility of privately manufacturing TPM is too low. All TPMs must be filed and activated through the root node before taking effect. The relationship between TPM and quantum computing is shown in the previous section.

TPM does not need to be implemented immediately, for all file certificates are still legal certificates that cannot be counterfeited. Malefactors need to spend a lot of cost to obtain these certificates.

6.3 TPS

The core of blockchain is reliability rather than TPS. However, TPS can be solved through side chain and mortgage mechanism.

6.4 Failure to package transactions in time

If a secondary node sends out a bookkeeping request and the primary node that finally gets the signature fails to package in time, the requester can resend it. There should be a certain number of bookkeeping nodes, otherwise some accounts may get lost. It is not feasible to leave it to a node for bookkeeping.

7 Terminology

- **TPS:** Trade per second
- **UTXO:** Unspent output from bitcoin transactions
- **TPM:** Trusted platform module

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, http://bitcoin. org/bitcoin. pdf. 2008.

[2] E. Foundation. Ethereum's white paper. Technical report, https://github.com/ethereum/wiki/wiki/White-Paper, 2014.

[3] Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. Pixel: Multi-signatures for consensus. Technical report, Cryptology ePrint Archive, Report 2019/514, 2019. https://eprint.iacr.org/2019/514.

[4] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th Security Symposium Security 15)*, pages 129–144, 2015.

[5] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News*, 33(2):51–59, 2002.

[6] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *23rd Security Symposium (Security 14)*, pages 781–796, 2014.

[7] Jing Chen, Sergey Gorbunov, Silvio Micali, and Georgios Vlachos. Algorand agreement: Super fast and partition resilient byzantine agreement. *IACR Cryptology ePrint Archive*, 2018:377, 2018.

[8] Ethash. Ethereum's white paper. Technical report, https://github.com/ethereum/wiki/wiki/Ethash, Accessed on June 27, 2017., version 23.

[9] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[10] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[11] SECG SEC. 2: Recommended elliptic curve domain parameters. *Standards for Efficient Cryptography Group, Certicom Corp*, 2000.

[12] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th security symposium (security 16)*, pages 279–296, 2016.

[13] Daniel J Bernstein. Multi-user schnorr security, revisited. *IACR Cryptology ePrint Archive*, 2015:996, 2015.

[14] Andrew Poelstra. Schnorr signatures are non-malleable in the random oracle model, 2014.