BAISS: a truly decentralized computational cloud on blockchain *(Beta version)*

BAISS LAB[*]

Abstract

BAISS has demonstrated the possibility of building a complete cloud infrastructure in blockchain, and this document formally discusses the methodology behind BAISS, especially its design and implementation.

I. Introduction

BAISS is an AI project which attempts to establish a decentralized computational cloud on blockchain [BAISS, 2020]. Traditional IaaS service providers like Google, as commonly known, have already monopolized the centralized cloud computing market [BAISS, 2020], and BAISS was born to break this situation. In this document, we discuss the theoretical principles behind BAISS, its design and implementation issues.

II. Preliminary

Let us introduce some fundamental concepts that are frequently used throughout the whole paper. BAISS can be viewed as a specific state machine: begin with a genesis state and incrementally execute *some specific operations* $\hat{V}$ to morph it into the current state. This current state we will accept it as the canonical version of BAISS. States can include any information such as account balances, reputations, trust arrangements, trust records, etc. A valid state transition $\hat{v} \in \hat{V}$ is represented as an arc between two states:

$$s_n \overset{\hat{v}}{\hookrightarrow} s_{n+1} \tag{1}$$

And a BAISS state transition function $\Im$ with a valid transition can be generalized as:

$$\Im(\kappa_n, \hat{v}) \equiv \kappa_{n+1} \tag{2}$$

$\Im$ allows components to carry out arbitrary computation, and κ allows components to store arbitrary state between legit operations.

Definition 1 *(**Operation-Based-Machine**) A (non-deterministic) operation based state machine is a tuple* $\mathtt{A} = (S, \Sigma, s_0, \hookrightarrow, F)$ *where:*

- S is a finite non-empty set of states, with $s_0 \in S$ being the start state;
- Σ is an alphabet notating operations;
- $\hookrightarrow \subseteq S \times \Sigma \times S$ is the set of labelled transitions over S;
- $F \subseteq S$ is the set of accept states.

[*]https://github.com/BAISS-Network

So now what we got is for any a $\in \Sigma$, we write $\overset{a}{\hookrightarrow}$ for $\{(s,s') \mid (s,a,s') \in \hookrightarrow\}$. Let Σ^* be the set of finite strings of labels in Σ, for any z = (a_0, a_1, ... , a_n) $\in \Sigma^*$, we have $s \overset{z}{\hookrightarrow} s'$ if there is a path $s \overset{a_0}{\hookrightarrow} s_1 \overset{a_1}{\hookrightarrow} \dots \overset{a_n}{\hookrightarrow} s'$ in A.

Given $\hookrightarrow$, we let the induced transition function δ: $S \times \Sigma \vdash 2^S$ be defined as δ (s, a) = $\left\{s' \mid s \overset{a}{\hookrightarrow} s'\right\}$. Note that δ (s, a) may be $\varnothing$ for some s and a. A is said to be deterministic, if for any s $\in S$ and a $\in$ Σ: δ (s, a) is a singleton. We can extend the transition function δ to δ^*: $S \times \Sigma^* \rightarrow 2^S$ such that δ^* (s, z) = $\left\{s' \mid s \overset{z}{\hookrightarrow} s'\right\}$. It is clear that a deterministic operation based state machine has the property that for any z $\in \Sigma^*$, δ^* (s, z) is a singleton.

Given A = (S, Σ, s_0, $\hookrightarrow$, F) and z = (a_0, a_1, ... , a_n) $\in \Sigma^*$, we call a sequence r = (s_0, s_1, ... , s_n) a run of A over z if for $0 \leq i \leq n$: $s_i \overset{a_{i+1}}{\hookrightarrow} s_{i+1}$. A run r = ($s_0$, s_1, ... , s_n) is said to be accepting if $s_n \in$ F. We say A accepts z if there exists an accepting run of A over z.

III. Transition Types

Under a real-valued time series $\{...x_{-1}, x_0, x_1, x_2...\}$, transition functions [1] in blockchain is rather simple. Transaction informations are collated into blocks and blocks are chained together using a cryptographic hash as a means of reference. Blocks records series of transactions, a ledger, together with the previous block and an identifier for the final state. The recording process forms the incentive mechanism for nodes to mine, which take places as a state transition function, adding value to a nominated account. Mining is the process of dedicating effort to bolster one series of transactions (a block) over any other potential competitor block. With the block level state transition function Ξ, transaction T, we can formalize the block B as follow:

$$\mathtt{B} \equiv (..., (\mathtt{T}_0, \mathtt{T}_1, ...), ...) \tag{3}$$

So the transaction-type state transition, the key component for building a decentralised consensus-based transaction system, can be expressed as:

$$\Xi(s_n, \mathtt{B}) \equiv s_{n+1} \tag{4}$$

The so called the PoW reward function Ω can then be treated as a kind of the block finalisation function, which is characterized as:

$$\Xi(s, \mathtt{B}) \equiv \Omega(\mathtt{B}, \Im(\Im(s, \mathtt{T}_0), \mathtt{T}_1)...) \tag{5}$$

The formalization above is essentially about a ledger maintained by a network of participants called miners which maintain the ledger through the internet. Since the whole structure is restricted by the transaction-parameter, so it is not surprise to see that the formalizations (3)-(5) are too weak to characterize a framework for building most of the traditional tech applications. Here we will going to break this monistic dogma and provide a pluralistic account to the concept *transition-type* in order to enrich the structure of the whole blockchain language.

BAISS provides a decentralised computing cloud on blockchain, so a new PoW reward mechanism is needed to motivate all the nodes in the network to maintain the BAISS ecosystem through

[1] Transition functions here are a kind of global mechanical model focusing on specific operations that cause instantaneous state change, rather than the state-preference model related to the theory of agency and choice.

transmitting values. And apparently the legit operations that BAISS require should include more than just a ledger recording process, therefore other than just letting the nodes to run the brute force and compete for the block finalisation, we adopt a hybrid consensus to promote the BAISS nodes to feed our BAISS system with different types of computing resources. As a result, we got a modified version to the definitions (3) and (5), as what follows:

$$\mathtt{B}^* \equiv (..., (\mathtt{RES}_0, \mathtt{RES}_1, ...), ...) \tag{3'}$$

$$\Xi^*(s, \mathtt{B}^*) \equiv \Omega(\mathtt{B}^*, \Im(\Im(s, \mathtt{RES}_0), \mathtt{RES}_1)...) \tag{5'}$$

IV. Isolatability

BAISS has only one goal, which is to make blockchain truly applicable to every single corner of the world. However, present-day blockchain technology is too simple to run any complex projects, and therefore the real practical E-commerce project such as Ebay, will never fit in any current blockchain framework. We believe, isolatability is one of the main obstacles hindering the adoption of blockchain technology[BAISS, 2020].

Isolatability measures the level to compatible with different types of demands of multiple parties and applications at a near optimal level under the same framework [BAISS, 2020]. The way for BAISS to achieve isolatability is to build a decentralized version of commercial IaaS on top of BAISS. Once we got the decentralized IaaS, we can then build any decentralized PaaS, SaaS, and write AI-Dapp on BAISS, and therefore meets the isolation requirements. BAISS believes that products that are needed in the reality are not driven by fancy technical terms but by great user experiences. The genuine challenge to blockchain is to develop decentralized solutions with at least the same quality as the best existing products [BAISS, 2020].

Currently, one of the most widely deployed public cloud is Amazon's cloud services product, which is Amazon Simple Storage Service. Its cloud storage protocol is widely deployed in the existing tech world, and therefore explains the reason why many cloud storage products provide some form of compatibility with the Amazon storage service API architecture. For saving the switching costs and improve the user experience for our users, our implementation allows applications previously built on Amazon to work with BAISS with minimal changes. The compatibility adds extra requirements for feature set, performance, and durability [2].

At the minimal level, we implemented:

V. Medium of Value

To make the entire BAISS system work requires sufficiently large storage spaces and powerful computing resources. So to address this problem, we have to incentivise decent amount of nodes to provide their resources for supporting the network. Based on the **BAISS − 20 protocol**, BAISS will issue a token called $BIGO_\alpha$ for the node motivation, i.e., to tell us what kind of resources can be substituted in parameter $\mathtt{RES}$ to get the PoW rewards in BAISS.

[2] From a metadata storage system standpoint, the compatibility also requires us to support: hierarchical objects, per-object key/value storage, arbitrarily large files, arbitrarily large amounts of files, and so forth. And objects should be able to be stored and retrieved by arbitrary key. Every time an object is added, edited, or removed, entries in this metadata storage system have to be adjusted. As a result, there could be heavy churn in this metadata system. Fortunately, the metadata can be heavily partitioned by the user.

Implementation 1: Minimum Amazon API

```
// Bucket operations
CreateBucket ( bucketName )
DeleteBucket ( bucketName )
ListBuckets ()

// Object operations
GetObject ( bucketName , objectPath , offset , length )
PutObject ( bucketName , objectPath , data , metadata )
DeleteObject ( bucketName , objectPath )
ListObjects ( bucketName , prefix , startKey , limit , delimiter )
```

.1 BIGO alpha

BIGO$_\alpha$ is the type of BAISS token that miners will get when they provide the storage functions or resources, which means the token is produced by processing token mining.

Implementation 2: Reduction 3.5 years each; Each reduction * 0.65; Initial: 200 millions

```
// Initial Circle: 200,000,000 * 3.5 = 700,000,000 $BIGO_\alpha$
130,000,000 * 3.5 = 455,000,000
84,500,000 *3.5 = 295,750,000
54,925,000 *3.5 = 192, 237,500
35,701,250 * 3.5 = 124,954,375
23,205,812.5* 3.5 = 81,220,343.75
...
```

So sixty years later, the total amount of the mining output will reach around 2 billion, which is exactly the total amount of $BIGO_\alpha$. For the miners of BAISS, the token generating process of $BIGO_\alpha$ is to (I) provide storage resources; or (II) provide AI computational resources. Therefore, from the perspective of the entire BAISS economic system, $BIGO_\alpha$ is an incentive mechanism used to motivate all the nodes in the network to maintain the whole BAISS ecosystem [3].

In order to maintain the dynamics of BAISS system operation, BIGO$_\alpha$ must satisfies the price volatility property, which means when we design BIGO$_\alpha$ we allow its price to change over time.

.2 BIGO beta

We know that BAISS is designed for real world business, so we need a token which is not involved in price fluctuation, to be used to pay for business services on BAISS. Therefore we have $BIGO_\beta$, which is the type of BAISS token that is needed when the users experience the services on BAISS system. The key feature of BIGO$_\beta$ is to resist the token price fluctuations. Since BIGO$_\beta$ has this unique property that against inflation and volatility, therefor business clients will worry-free about payments stability instead facing token price change frequently.

We now know that BIGO$_\beta$ is a kind of stable coin which can be used in BAISS. So how do we ensure the stability of it? BIGO$_\beta$ takes the *algorithmic stablecoins approach*, which do not require any collateral backing their system, and rely only on algorithms to get the coin price to remain

[3] BIGO$_\alpha$ also serves as the battery of the BAISS system.

stable. In the quantity theory of money, we have two central principles for controlling the price of money:

- **Money Supply Expansion**: if prices are going up, then expand the money supply to bring them back down.

- **Money Supply Contraction**: if prices are going down, then contract the money supply to bring them back up.

Inspired by these two classical principles, BAISS proposed **BIGO$_\beta$-GEN System**. According to the **BIGO$_\beta$-GEN Protocol** which defines a target price for BIGO$_\beta$ in the pegged asset, we now have 1 India Rupee (INR) for 1 BIGO$_\beta$.

It is easy to deal with the case of token supply expansion. However, we still have to further explain how to deal with the case of token supply contraction. There are two types of assets in the BIGO$_\beta$-GEN System. The first type is the BIGO$_\beta$, which is a stable currency that is pegged to the India Rupee price. On the other sides, BIGO$_\beta$-GEN Protocol defines a subsidy asset called BIGO$_\beta$-bond. When the price of BIGO$_\beta$ is less than 1 INR, BIGO$_\beta$-bond will be issued by the BIGO$_\beta$-GEN System for repurchase BIGO$_\beta$. When BIGO$_\beta$ returns to 1 INR, BIGO$_\beta$-bond will be redeemed for 1 INR.

.3 Together

BIGO$_\alpha$ can be used for exchange the BIGO$_\beta$ at certain ratio. With the growth of the BAISS ecosystem, the services in BAISS become thrillingly complete, also it attracts more users participate in the system. Therefore, the demand of BIGO$_\beta$ will continually grow, push up the price of BIGO$_\alpha$, which it will create a decent return for the BIGO$_\alpha$ holders [4]

VI. Hybrid Consensus

So now we will explain further about the concept of the legit operations $\hat{v}$ between states, and the resouses `RES`. Here we take storage providing as an important example, and the *computational recourses providing* takes a similar process.

.1 Storage

- The distributed storage system of the BAISS chain uses a data integrity certification scheme based on the Merkel tree. After the user encrypts the file, the user generates and saves a random challenge, which corresponds to the block data one-to-one. The block data and the random challenge are hashed together to form a Merkel leaf node, which is constructed into a Merkel tree, and the user saves the node information of the Merkel tree leaves and the height of the Merkel tree.

[4] In this case the price prediction can be just simplified as a simple single variable linear regression model:

$$Y = M(X) + e \tag{6}$$

Actually even from an analytical angle and try not to be too optimistic, if we have to figure out the details about the market factors set $\psi = (m_1, m_2, ..., m_n)$ that affect the token price of BIGO$_\alpha$, because of the embedding with the business layer of BAISS, the whole price prediction process will tend to be more realistic than all the other air-coins in the crypto market because we can get real business performance data for helping us to adjust our views about the pricing.

In the verification phase, the user randomly selects a challenge from the random challenge and sends it to the BAISS storage engine. The engine sends the challenge to the storage provider for verification and then returns the user result. The storage system of the BAISS chain includes the roles of users, storage engines, and storage providers. There are three core stages of the entire process: data storage, data retrieval, and transaction payment. In the data storage stage, users submit data and files to the storage engine of the BAISS chain, and the storage engine of the BAISS chain will distribute the data to the storage provider after the encryption.

Process

1. *Application Stage*
   a. The user sends a storage request to the BAISS chain, and the user provides information to the storage engine.
   b. The BAISS storage engine checks the user's qualifications. The audit result is written into the blockchain and after the network node is confirmed if the BAISS storage engine issues an access certificate to the user; if the audit fails, the access will not issue a certificate.

2. *Registration Stage*
   a. Send the license to the user for storage request.
   b. After BAISS confirms the user's identity, the user encrypts the file private key and sends it to the storage engine.

3. *Preparation Stage*
   a. After the user encrypts the data, a series of challenge factors are randomly generated and verified, then the challenge factors are saved. The user hashes the challenge factor with the data and constructs the file Merkel tree. The user saves the file tree locally and sends the block file data to the file intermediary.
   b. The intermediary distributes the file data to the storage provider, and informs the user about the distribution.

4. *Challenge and Retrieval Stagee*
   a. The user randomly selects a challenge factor and sends it to the storage supplier.
   b. The storage provider generates a data integrity certificate and sends evidence of data integrity to the user, and then the user verifies whether the proof of completeness is correct.

5. *Payment Stage*
   A sequence expired revocable contract channel has been established between the user and the storage provider. When the system verifying the data integrity certificate in the challenge stage, the user pays storage fees to the storage provider through the baiss network.

.2 Formal Theory

[*censorship*]

.3 Mining Machines

The Delegated Proof of Work (DPoW) mechanism used by the BAISS chain is an improved version of the normal PoW consensus algorithm. Compared with traditional PoW, BAISS imposes the resource providing structures on the mining process and embeds an off-chain governance on the consensus. Since the core of the BAISS chain is distributed storage computation, the DPow mechanism allows storage nodes to frequently perform backup (snapshots) operation to ensure that the entire network can be quickly restored after a system failure or intrusion. Data, so any attempt to destroy all the data backups in BAISS would become much difficult.

There are 21 super nodes and 100 ordinary nodes in the BAISS system; the main network relies on unforeseen random number algorithms to select 6-15 (to be determined) nodes, so that there are only a small number of nodes in the entire network can complete the calculation of simple hash sharding tasks, get rid of the high-energy-consuming competitive method (PoW), so that the remaining nodes can handle more tasks (such as storage, elastic computing, etc). The process that DPoW randomly selects some nodes can greatly avoid the waste of the computing power consumption, so that more micro-computing power devices (in the early stage) can participate. Also the unpredictable random number algorithm makes all nodes in the network have the same probability to be selected.

All nodes in the BAISS chain have the right to mine. But this concept is only for nodes from the same level, which is not applicable between super nodes and ordinary nodes. Also because we have different roles and resources existing in the BAISS network, we can foresee that there will be different types of mining machines produced for processing different missions.

VII. History of Trust and Risk Control Ranking

.1

Given $n \in N$, consider $x = (x_1, x_2, \dots , x_n)$, the key here is the probability $P = (y = 1 \mid x)$; we have:

$$P = (y = 1 \mid x) = 1/1 + e^{-g(x)} \text{ ; } in\ which\ g(x) = \beta_0 + \beta_1 x_1 + ...\beta_n x_n \tag{7}$$

Consider: $P = (y = 0 \mid x) = 1/1 + e^{g(x)}$. what we care: $ODD = P/1 - P$ and $P = ODD\ /\ 1 + ODD$

$$g(x) = In(P/1 - P) = \beta_0 + \beta_1 x_1 + ...\beta_n x_n \tag{8}$$

.2

[*censorship*]

VIII. VM and Smart Contract

.1 Virtual Machine

BAISS Virtual Machine (BVM) is used to compile smart contract code into machine code that can be executed on the BAISS system, and to provide a running environment for smart contracts. It is a sandbox environment completely isolated, which cannot access the network and files during operation, even between different contracts there are only limited access rights.

- **Features**

  I. BVM is a stack-based virtual machine, which different from the register-based virtual machine setting, used to compile and execute smart contracts.

II. BVM is Turing-complete [5].

III. BVM is a completely isolated environment, and cannot access the network and files during operation, even between different contracts there are only limited access rights.

IV. Call depth of the operand stack is 1024.

V. Length of the machine code is one byte, and there can be up to 256 opcodes.

- **OpCode**

  All opcodes are defined in the file **opcodes.bi**. Opcodes are divided into 9 groups according to the functions (block operation, encryption etc).

- **Instruction Sets**

  Four instruction sets are defined in **jump.table.bi**: Frontier Instruction Set, Home Stead Instruction Set, Byzantine Instruction Set, Constantinople Instruction Set (*Heath*, *Farm*, *Byzantium*, *Constantinople*); correspond to four development stages of BVM. The instruction set is forward compatible.

- **Interpreter**

  The entry function of interpreter *run* are in **interpreter.bi**, which cyclically analyzes the code in the smart contract according to the input given by the user, and translates it into a function in the corresponding instruction function set, then run and execute.

- **Battery**

  About **battery_table.bi**, **battery.bi**, calculate the consumed battery according to different operations, and all the specifications are defined in **battery_table**.

- **Smart Contract**

  About **contract.bi**, a contract is a storage unit of the BVM smart contract, and also the basic unit for an interpreter execution. It contains information about the code, caller, owner, and battery. **contracts.bi** contains some BVM pre-compiled contracts.

- **Memory**

  About **memory.bi**, memory is used for operations like (*MLOAD*, *MSTORE*, *MSTORE8*) and parameter copy of contract calls (*CALL*, *CALLCODE*). Data structure for memory maintains a byte array. When *MLOAD* and *MSTORE* are read and stored, the position and length must be specified.

- **Stack**

  About **stack.bi**, stack in BVM is used to save the operands.

[5] Or more precisely, a *quasi*−Turing−complete machine. The quasi qualification comes from the fact that the computation is intrinsically bounded through a parameter, *battery*, which limits the total amount of computation done.

- **Statedb**

  About **go** $-$ **baiss/core/state/statedb.bi**, the contract itself does not save data. The contract and its calls are similar to the log of a database. It saves the related definitions of a contract and a series of related operations. As long as these operations are executed once, the current result can be obtained. However, if we have to execute it every time then it's too slow, so this part of the data will be persisted to **statedb**. Instructions $SSTORE, SLOAD$ are defined in code to process the current state of the contract from **db**.

- **Log**

  About **logger.bi**, it records the running process of BVM. It also cover **memory**, **stack**, **statedb**, and some related information.

.2 Smart Contract

When a pre-condition is triggered, the smart contract executes the corresponding contract terms. In BAISS, we have an independent smart contract system: BAISS Contract.

- **Features**

  I. Deterministic. II. High Performance. III. Expandability.

- **Contract Types**

  I. Verificational. II. Functional. III. Applicational.

BAISS uses a lightweight BVM as its smart contract execution environment. Its startup speed is fast enough and only takes up tiny resources, making it so suitable for running small programs like smart contracts. Significant improvement can be acheived through the static compilation and caching of smart contracts with JIT (just-in-time compiler). The BVM instruction set provides a series of built-in cryptographic instructions, to optimize the efficiency for executing cryptographic algorithms used in smart contracts. In addition, data manipulation instructions directly support arrays and complex data structures and improve the whole performance of BAISS smart contract system.

The scalability of BAISS smart contracts is achieved through combined a high concurrency and dynamic partitioning with its low coupling design. The low-coupling contract program is executed in a virtual machine (BVM) and do the outer communicating through the interactive service layer. Therefore, most of the upgrades to the functions of a smart contract can be achieved by adding APIs in the interactive service layer.

.3 Execution Model

Given a *battery status* $\mho = < bat, cal >$, for any state s and z, if there is a way α such that $(s,t) \in \hat{V}(\alpha)$, then we should know (1) the operation α is one of the legit operations; and (2) after running cal, bat is not empty and enough for the system to consume it for supporting the whole state transition process from s to z.

IX. Security analysis

.1 Quantum Algorithm: Shor's Algorithm

With the help of Shor's algorithm, it allows the factorization to be performed in polynomial time, rather than exponential time achieved after using classical algorithms. For factorizing integer numbers, Shor's algorithm determines the prime factors of a composite l-bit number N in $O[l^2 \ log \ l \ log \ log \ l]$ steps, while the best classical algorithm requires $O[exp \left\{ cl^{1/3} log^{2/3} l \right\}$ steps, for some c, and therefore cracking RSA. We also know that the discrete logarithm problem (DLP) based on the hyperelliptic curve of genus 2 (HCDLP) is widely used in industry. However, the activation of the Shor's algorithm can solve the DLP and therefore invalidate those popular digital signatures such as DSA, ECDSA, and EdDSA. The race to build a quantum computer has begun. Companies such as Google, Microsoft, IBM, D-Wave, and Intel are leading the way, significant progress has been made in this area. In the next 5 to 10 years, we will be able to build a computer with thousands of stable qubits to make classical public key cryptography obsolete. When a quantum computer is able to crack asymmetric encryption, the impact to the blockchain security will be huge. If the security of RSA or ECC has no guarantee, the market value of Bitcoin and similar blockchains that are not resistant to quantum will all collapse. Since the whole mining process of BISO$_\alpha$ takes 60 years, we cannot let similar problem happens on BAISS.

NTRU encryption algorithm, a cryptosystem alternative to RSA and ECC, is adopted here. NTRU is based on polynomial rings, and the underlying mathematical problems are the shortest vector problem (SVP) and the closest vector problem of lattices. It is fully accepted to IEEE P1363 standards under the specifications for lattice-based public-key cryptography. Compared with the methods that rely on the difficulty of factoring integers, finding the discrete log of a number, the security of NTRU is based on intractability of hard problems in certain type of lattices, called convolutional modular lattices, which gives the algorithm the ability to resist quantum computing attacks, while the RSA and ECC algorithms cannot. On the other hands, we pick NTRU, rather than McEliece or MQ, because (1) McEliece creates a public key cryptosystem based on error-correcting codes and the difficult problem of decoding a message with random errors. It has has strong security potential but low computational efficiency. (2) The security of MQ is based on the intractability of multivariate quadratic polynomial equations over a finite field, which has obvious disadvantages on security side. In contrast, the NTRU public key encryption system has simple algorithms that compute fast enough, spend only small storage space, and easy to implement.

- **Key generation**

  We use the polynomial mathematic addition and multiplication on a truncated polynomial ring $R = Z[X]/(X^N - 1)$. The coefficient of polynomial reduced by modulo on two integers p and q and the degree of polynomial determine by N. Parameters involve:

  N: the degree of the truncated polynomial ring is $N - 1$.
  q: large prime modular that use in reduce the coefficient of the truncated polynomial.
  p: small prime modular that use in reduce the coefficient of the message and use in last operation in decryption.

The secret key is consist of two small polynomials in the truncated polynomial ring f and g. Over R_q and R_p, the f polynomial should be invertible and have: df numbers of coefficients equal to $+1$ in f, $df - 1$ numbers of coefficients equal to -1 in f, the rest coefficients equal to 0; and the g polynomial have: dg numbers of coefficients equal to $+1$ in g, dg numbers of coefficients equal to -1 in g, the rest coefficients equal to 0. Calculate for f polynomial the inverse modulo q and modulo p that means, $f * f_q \bmod q = 1$, $f * f_p\ mod\ p = 1$ (* means polynomial multiplication in ring, cyclic convolution).

Compute the public key by:

$$H = p * f_q * g\ modulo \qquad (xxx)$$

H: public key. f, g: private key. N, p, q: public parameters.

- **Encryption**

  m: plain text that should be converts to polynomial their coefficients modulo p.
  r: blinding polynomial chooses randomly $r \in R$.
  e: cipher text which calculates by:

$$e = r * h + m\ modulo\ q \qquad (xxx)$$

- **Decryption**

  I. Calculate a polynomial: f: private key, coefficients of a must be between $-q/2$, $q/2$.

$$a = f * e\ modulo\ q \qquad (xxx)$$

  II. Calculate b polynomial: coefficients of b must be between $-p/2$, $p/2$.

$$b = a\ modulo\ p \qquad (xxx)$$

  III. Calculate c polynomial: c represents the plain text, coefficients of c must be between $-p/2$, $p/2$.

$$c = f_p * b\ modulo\ p \qquad (xxx)$$

.2 Quantum Algorithm: Grover's Search

It is computationally difficult to take a modified block content and a given hash, add trivial data to the content to make the given hash consistent with the block content. It assumes a brute force search through the possible source data with sufficient additional bits to exhaust the hash space until a case is found that matches the known hash value. For ideal hash, this requires linear time in the size of the hash space. A run time of order $O(n)$ is expected for this classical attack. However due to a quantum algorithm known as Grover's search, which runs in time $O(\sqrt{n})$, so would give a speedup of $O(\sqrt{n})$ compared to classical collision search. It also speed up the generation of nonces, making the reconstruction of the chain from a modified block forward much faster.

- The hybrid consensus settings invalidate the simple brute force search mining model.

.3 Stability: Sybil Attack

In a peer-to-peer network, nodes will join and exit at any time, so to maintain the stability of the network, the same data usually needs to be backed up to multiple distributed nodes. Sybil attack is an effective means to attack this kind of data redundancy mechanism.

- While adopt the PoW scheme **S/Kademlia** proposes to partially address Sybil attacks.
- BAISS conducts mining based on the storage resources and computing resources providing, so even if a malicious node can create many addresses, the created addresses basically are meaningless for our hybrid consensus.
- Strange behaviors will be detected by the risk control mechanism.

X. Conclusion

We are in a world which all the centralized tech giants are bullying us by raising their service fees, but we have no other choice. In this yellow paper, we introduced some important theoretical considerations behind the design and implementation of BAISS, which we believe will be useful for readers to take a leap to the details about this amazing decentralized cloud system. We believe that freedom is the most essential key of human's society, therefor any formalities of monopoly structure system will be obstacle to human freedom of choice. In fact, blockchain technology is definitely not a toy for few geeks. BAISS hopes that innovative blockchain technology breakthroughs can be a whole product to compete with any tech giants' centralized systems, period.

References

[BAISS, 2020] BAISS Lab. (2020). BAISS: a new standard of blockchain technology (Beta version), [Online]. Available: https://github.com/BAISS-Network.