www.platon.network

PlatON

PlatON: A High-Efficiency Trustless Computing Network

V0.6.1

August 29, 2018

“ Hence the world is not only the most admirable machine,but also insofar as it consists of minds it is the best commonwealth,through which there is conferred on minds as much happiness or joy as possible,and it is in this that their physical perfection consists.”

Gottfried Wilhelm Leibniz
-“On the Ultimate Origination of Things”

Summary

The scalability issue of blockchain is intrinsic to the consensus algorithm that it relies on to validate transaction or computation in a trustless way. The more nodes participate in replicating a single computation, the more confidence is bestowed on the correctness of the results. However, the time to consensus and resources committed on the work increase drastically with the number of nodes prohibiting blockchain being adopted in a large scale. The flipside is consensus is not the essence of blockchain's appeal but the trustless feature which can be attained cryptographically.

Verifiable computation (VC) is originally a scheme enabling a client or verifier to verify the correctness of the results computed by an untrusted cloud provider. The computing party is responsible to conjure a cryptographic proof for the verifier to easily evaluate the correctness of the output. Compared to consensus algorithm, VC, in theory, guarantees correctness without replicated computation, tremendously reduces the time and cost otherwise consumed in running thousands of nodes and synchronizing them for block production. In most circumstances, VC can potentially be the more efficient infrastructure for smart contract execution and diminish the role of blockchain to a tamper-free database for the storage and public access of proofs.

PlatON uses VC to achieve scalability, verifiability and privacy in trustless computing. The proprietary technology developed by the team in the past few years further reduces the overhead time and resources for the prover by several magnitude turning it into a more practical solution for general trustless computing. Moreover, as privacy is concerned in many trustless computing cases, secure multi-party computation (MPC) and homomorphic encryption (HE) are employed to prevent exposure throughout the entire lifetime of the data. The goal of PlatON is to construct a full stack infrastructure for decentralized apps, while individual module such as VC will be available to connect other blockchains to PlatON for scalable and privacy-preserving computing.

Contents

Chapter 1

Trustless Computing

1.1 Introduction to Trustless Computing

In the digitalized world, any individual can only get hold of a small subset of the massive amount of data generated daily. It is impossible for any entity to stream the whole set of data valuable to them, handicapping their full grasp of the landscape.

Every participant of the digitalized world is partially blind, blocked in a certain angle towards the full picture. Collaborators thus have the need to exchange data or engage in collaborative computation, in the process of which value, information and asset are traded. Traditionally, untrusted collaborators tend to use a “trusted third party” to collect the data and verify its validity, e.g. outsourcing to the fast ascending cloud computing service. However, trusting a third party inevitably exposes users to potential issues with scalability and privacy.

The advancement in modern cryptography, especially fueled by the recent rise of blockchain, proposed a new collaborative computing model, many have called it “trustless computing”, meaning the integrity of the computing results is assured without reliance on a third party.

1.2 Trustless Computing Schemes

1.2.1 Consensus-based Strategy

A decade ago, Bitcoin, for the first time, made trust trivial between parties involved in a transaction using cryptocurrency. Transactions can be validated after decentralized nodes agreeing to a current state. Ethereum inherited the consensus algorithm from Bitcoin and went beyond being a ledger by attesting to the computation of smart contract, hence enabling trustless computing of an arbitrary program. However, to assure correctness the computation must be replicated by all the nodes, manifesting the intrinsic contradiction between efficiency and trustlessness. Although Ethereum was called a world computer for large scale trustless computing, it fails the mission due to poor scalability. In addition, the lack of confidentiality also prohibiting blockchain to be used to process private data and hinder the further growth of decentralized applications.

Large-scale trustless computing is the foundation for decentralized applications as cloud computing is to the millions of centralized applications. There are more than a hundred projects working on the consensus layer, taking different sets of trade-offs, to develop a scalable blockchain for decentralized applications to run fast and securely and yet this direction is proven to be extremely complicated and challenging. In consensus-based schemes, there are three major common issues, described below.

1.2.1.1 The Verifier's Dilemma

Researchers from National University of Singapore mentioned a concept of the Verifier's Dilemma [**The Verifier's Dilemma**] in the paper "Demystifying Incentives, in the Consensus Computer". According to this concept, complex computation task will break the integrity of the network, facilitating malicious attacks.

The blockchain system rewards block producers who win the mining contest, however providing no direct economic incentive to the other miners that verify the transactions in the block. The issue may seem trivial when majority of the transactions in the network are simple computation tasks, requiring very limited computing power. As the transactions become more demanding for computing power, such as complex smart contract, the verifiers may spend large amount of resources but end up getting no reward. Many verifiers are likely to skip the verification step to save resources for the race to the mine the next block. With less verifiers, the possibility of launching a successful attack on the network goes up.

A typical attack resulted from Verifier's dilemma could start with a malicious actor broadcasting a series of complex computing intensive transactions to the networks to exhaust the resource of other miners. While rational miners compete to solve these puzzles, the bad actor could gain an advantage in mining the next block by starting early.

1.2.1.2 The Scalability Trilemma

The Scalability Trilemma is a huge challenge for the blockchain scalability [**The Scalability Trilemma**] which can be presented in the form of a triangle, with scalability, decentralization and safety each occupying a leg. It is theoretically impossible to maximize the performance of blockchain in all three dimensions at the same time. A certain system must choose a set of trade-offs in pursuit of scalability.

Bitcoin and Ethereum are designed to achieve the most permissionless participation of block producers for decentralization and safety, however, at the cost of scalability. This fully decentralized scheme requires every node to process every computation, limiting the throughput of the network within the capacity of each single node. As the number of nodes accrue when the network grows, the overhead time required for consensus increases gradually, resulting a long latency to finality.

1.2.1.3 Security and Privacy Dilemma

On-chain consensus scheme entails complete copies of ledger or state stored on every full node and publicity of all transaction data, whereby giving up protection of privacy. Transparency is important, but so is privacy when it matters. As long as privacy is concerned, the users would not release the data on the blockchain, putting a roadblock in the utility of decentralized applications.

1.2.2 Off-Chain Strategy

To avoid the trade off in efficiency, people in the industry have increasingly come to an agreement: the proper use of blockchain is for verification only; the computing tasks must be separated from the consensus layer and migrated off-chain, where a scalable, privacy-preserving and verifiable computing infrastructure can be constructed to allow untrusted entities to engage in collaborative computation.

Currently a few projects are under development to realize off-chain trustless computing, such as Oasis, Truebit, Stark, etc. Based on the verification mechanism, there are three main strategies.

1.2.2.1 Trusted Hardware

Trusted hardwares, such as SGX/Intel, provides a trusted execution environment (TEE), to host computing nodes. Computing nodes built on TEE perform trustless computing over private data and attest to the correctness of the results on chain. The TEE, however, used as another trusted third party, establishes a trust boundary, inside of which code and data are deciphered, substantiating a security threat.

In fact, SGX, the most widely used trusted hardware, has very serious security flaws due to the very recently research results. As the Foreshadow [**Foreshadow**], the recent state-of-the-art attack on intel SGX, it says that "Foreshadow demonstrates how speculative execution can be exploited for reading the contents of SGX-protected memory as well as extracting the machine's private attestation key". What making things worse is that "it only takes a single compromised SGX machine to erode trust in the entire SGX ecosystem".

1.2.2.2 Interactive Game

On-chain computation is expensive and restricted for complex programs on Ethereum. Truebit suggested an interactive proof protocol to enforce off-chain node perform the computation correctly. The solver, who solve the computation task must play an interactive verification game with the verifier, when challenged. The game narrows down to the problematic step in the computation process through rounds of interactions and eventually settles the disagreement on-chain with the minimum amount of computing and expense. The dynamics between solver and verifier add uncertainty to the Truebit system and the interactive mechanism implies long latency to finality.

1.2.2.3 Non-interactive Proof

In non-interactive verifiable computing, no interaction between the prover and verifier is necessary, nor is there any constraint on the identity of the verifier. The proof, once posted on chain, is public verifiable by anyone remotely on their local machine. Verifiable computing was originally studied to enable client to outsource the computation of a function to untrusted entities with stronger computation capacity and verify the correctness of the results with much less resource compared to what is required for the native computation.

SNARK and STARK are the two most well-known implementations of VC, both provide zero knowledge proof, preserving privacy of the prover and reducing the proof size and verification time, thus practical in certain applications. SNARK has very short proof and fast proof verification. However, it requires a complicated trusted setup procedure which is the biggest obstacle in deployment. The major advancement of STARK, over SNARK, is transparency, hence no initial setup is necessary. Although STARK is a very impressive construction, the concrete efficiency, such as proof generation time and proof length is not practical enough for real applications.

In a VC scheme based on SNARK or STARK, it requires a witness in the proof generation, if that is anyone other than the data owner, the private data is revealed. SNARK/STARK do not protect the privacy of the client who outsource the computation to an untrusted third party, although the privacy of the prover is guaranteed.

PlatON network, in contrast, deploys a high-efficiency VC algorithm overlaid upon homomorphic encryption (HE) and secure multi-party computation (MPC), provide a complete solution for trustless computing while maintaining privacy of the client's data in the whole process.

1.2.2.4 Trustless Computing in PlatON

PlatON is a realization of a practical VC algorithm, that requires no external setup, and compared to previous VC or Zero-Knowledge Proof (ZKP) solutions, significantly shortens proof size and enables accelerated proof generation and verification. In the workflow of PlatON, confidentiality is accomplished using full homomorphic encryption (FHE) and secure multi-party computation (MPC), which jointly guarantees the privacy of the input data and computing logic. The trustless computing on PlatON, in contrast to trusted computing that relies on trusted hardware or TEE (e.g. SGX) provided by third party manufacturer for computing integrity, only depends on falsifiable cryptographic assumptions, thereby providing unprecedented security of private data through its lifetime without a trust boundary.

On PlatON, computation is separated from the on-chain consensus, thus not bounded by the scalability trilemma of blockchain, attain excellent scalability and decentralization without compromising safety and privacy. By breaking down the computation to basic computation elements, individual gate or circuit, PlatON sends these to multiple computing nodes to be executed parallelly, vastly elevating the horizontal scalability and range of distribution of participation. Moreover, no redundancy is necessary once the proof is quickly verified to validate a transaction, expediting the process and overall throughput.

The traditional zk-SNARK/STARK technology could not address the input privacy problem when outsourcing the computation to other nodes. This is why we can build VC with a deterministic prover/worker procedure on the core idea of Bulletproof, instead of the full mechanism of Bulletproof or some zero knowledge variants of SNARK (zk-SNARK). We would like to stress that all currently known constructions of zk-SNARK relies on some ad-hoc and quite non-standard complexity assumptions, such as even stronger variants of knowledge-of-exponent assumption, and they are often complex and hard to interpret. We don't know if these new assumptions will stand the test of time.

As a well-known VC scheme, SNARK generates constant-sized proofs for any NP statement, and have extremely fast verification time. However, the provers/workers of such a system need to spend quasi-linear time $O(n \log n)$ in generating a proof. In our VC, though size of the proof is slightly larger (logarithm in the size of the target function, which is the same asymptotic complextity as STARK, but with shorter concrete length), but it admits much more efficient workers (generating a proof in linear time), and this lower time complexity of the worker is more desirable in our scenario. In the future, we will explore the possibility of parallelizing the computation of the worker, and make proof-generation more efficient.

Chapter 2

Strategic Position

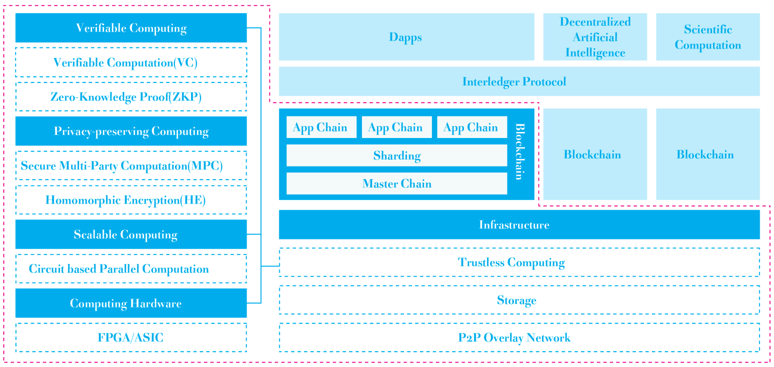

Figure 2.1: PlatON's overall architecture

PlatON's overall architecture is summarized within the red dotted line in Figure 2.1. PlatON provides a complete trustless computing infrastructure that can be used for Dapp or scale other blockchains. PlatON's own blockchain is the base chain that incentivize the computing nodes and store states and computation proofs.

PlatON is designed as an infrastructure including computing, storage, service discovery, etc. for diverse decentralized systems that covers other blockchains, decentralized AI, scientific computation and more. The initial goal is to construct a trustless computing platform with cryptographic based verifiable computing, as a solution for the scalability, privacy and verifiability challenges that impeded blockchain. The following are the key components of the system.

- **Off-chain verifiable computing**, that scales horizontally with the number of nodes, is used to perform heavy and complex computation, while generating computation proof to attest to the correctness of the results.
- **Secure multi-party computing and homomorphic encryption** guarantee the private data in an encrypted form or jointly compute different functions (applications) with multi-party inputs through entire computation process, eliminating the trust boundary.
- **Compilers transform computation functions into circuits** which are distributed by the blockchain to different computation nodes. Functions represented by circuits can be executed with better concurrency and decentralization.
- **Computing hardware**. The circuit based computation naturally fits the architecture of FPGA, thus gain advantage in capacity and cost using specifically developed FPGA/ASIC hardware.

On top of the trustless computing platform, PlatON deploys its own blockchain as an immutable database to store states and proofs. The blockchain will also be used as a settlement layer to facilitate transactions and to incentivize participation as computing nodes and verification nodes. The main features of the blockchain are described below.

- **Appchains** (sidechains) can be created by sharding specifically to serve diverse applications.
- **Energon** is the native cryptocurrency on PlatON to support all economic activities especially purchasing large scale computing service. The cryptocurrency will be used as the basic unit for computing power and main form of payment.
- **Smart contract** is available on the base chain to provide diverse data service to dapps or other blockchain

Chapter 3

Technical Solution

3.1 Computing

3.1.1 Verifiable Computing

A publicly Verifiable Computation(VC) scheme allows a computationally limited client to outsource to a worker the evaluation of a function F on input u. The client can then verify the correctness of the returned $F(u)$ while performing less work than required for the function evaluation.

More formally, publicly VC is defined as follows.

Definition. A publicly verifiable computation scheme VC consists of a set of three polynomial-time algorithms.

- The randomized key generation algorithm $KeyGen(F, \lambda)$: On input the function F and a security parameter λ, it outputs a public evaluation key EK_F and a public verification key VK_F .
- The deterministic worker algorithm $Compute(EK_F, u)$: On input the evaluation key EK_F and input u, it outputs the result y (expecpted to be $F(u)$) and a proof π_y of y's correctness.
- The verification algorithm $Verify(VK_F, u, y, \pi_y)$: On input the verification key VK_F, u, the expected result y and the proof π_y , it outputs 1 if $y = F(u)$, and 0 otherwise.

The triple of algorithms of a VC are required to satisfy the following properties:

- **Correctness.** For any function F and any input u, if y is computed as expected, then the Verify algorithm will accept the proof.
- **Unforgeability.** For any function F and any probabilistic polynomial-time adversary, it is computational infeasible to produce a accepted proof π ,while $y \neq f(u)$.
- **Efficiency.** It is required that the complexity or the running time of Verify is cheaper than evaluating F .

In PlatON, we additionally require two notable features of our VC scheme:

- **Worker-efficiency.** It is relatively efficient for a worker to generate a correctness proof for a function value.
- No Trusted Setup procedure involved.

The state-of-the-art constructions follows the paradigm by transforming the target function F into an arithmetic circuit and then representing the circuit as Hadamard-product relation and linear constraints. We adopt the core idea of Bulletproof to build our VC scheme.

More specifically, any multiplication gate of fan-in 2 has three wires; 'left' and 'right' for the input wires, and 'output' for the output wires. In the relation a_L, a_R, a_O are the vectors in $\mathbb{Z}_p$ of left inputs, right inputs and outputs for each multiplication gate, respectively. It should satisfy the relation $a_L \circ a_R = a_O$, where $\circ$ is pair-wise multiplication. Other addition gates and constant gates will be encoded into matrices W_L, W_R, W_O and a vector c in $\mathbb{Z}_p$. The additional constraints are of the form:

$$W_L \cdot a_L + W_R \cdot a_R + W_O \cdot a_O = c.$$

The worker will generate a short proof according to the Hadamard-product relation with the constraints under the discrete logarithm assumption. The client could verify it very efficiently. More details of the VC scheme are as follows.

The algorithm $KeyGen(F, \lambda)$ taking as input a circuit that computes the target function F and the security parameter λ, outputs the vectors $g, h \in \mathbb{G}^n, W_L, W_R, W_O, c$ and a cryptographic hash function, which serve as evaluation/verification key.

In the algorithm $Compute(EK_F, u)$, the worker first computes the function F on a given input and then generates a correctness proof in the following way. The worker computes the vectors a_L, a_R, a_O from the structure of the circuit. The worker generates random values y, z from using the publicly known hash function, then computes two vectors Y, Z based on y, z respectively. Instead of proving the relations and directly, the worker proves the following relation (3.1):

$$\langle a_L, a_R \circ Y \rangle - \langle a_O, Y \rangle + \langle Z, W_L \cdot a_L + W_R \cdot a_R + W_O \cdot a_0 \rangle \\ +k(y, z) = \langle Z, c \rangle + k(y, z) \qquad (3.1)$$

where $k(y, z)$ is a polynomial can be computed by both parties.The worker generates two polynomials:

$$\begin{aligned} l(x) &= a_L \cdot x + a_O \cdot x^2 + Y^{-1} \circ (Z \cdot W_R) \\ r(x) &= Y \circ a_R \cdot x - Y + Z \cdot (W_L \cdot x + W_O) \end{aligned}$$

and encodes the left side of (3.1) into the second coefficient of the inner product polynomial $t(x) = \langle l(x), r(x) \rangle$, leaving the right side for the client to compute. The worker randomly generates a uniform value s from $\mathbb{Z}_p^*$ and use the publicly known hash function to computes the vectors $l = l(s), r = r(s)$, and the value $t = \langle l, r \rangle$. Finally the worker sends the proof $\pi = (a_L, a_R, a_O, t_1, t_3, l, r)$ to the client.

In the algorithm $Verify(VK_F, u, y, \pi_y)$, upon receiving the proof, the client first gets y, z using the hash function and then obtains the vectors Y, Z , along with the polynomial $k(y, z)$,and then computes the random value s from hash function. Finally the client checks the proof for the relation (3.1) by verifying the following conditions.

$$\begin{aligned} l &= a_L \cdot s + a_O \cdot s^2 + Y^{-1} \circ (Z \cdot W_R) \cdot s \\ r &= Y \circ a_R \cdot s - Y + Z \cdot (W_L \cdot s + W_O) \\ \langle l, r \rangle &= t_1 \cdot s + (\langle Z, c \rangle + k(y, z)) \cdot s^2 + t_3 \cdot s^3 \end{aligned}$$

In order to reduce the proof size, the worker can store the vectors a_L, a_R, a_O, l, r into three group elements:

$$A_I = g^{a_L} h^{a_R} \in \mathbb{G},\ A_O = g^{a_O} \in \mathbb{G},\ P = g^l h'^r \in \mathbb{G}$$

where h' can be computed from h, and $P = A_I^s \cdot A_O^{s^2} \cdot h'^{-Y} \cdot w_l^s \cdot w_r^s \cdot w_o$ with:

$$w_l = h'^{Z \cdot W_L},\ w_r = g^{Y^{-1} \circ (Z \cdot W_R)},\ w_o = h'^{Z \cdot W_O}$$

After that, the worker executes an inner product argument system (which can be done in a short and non-interactive proof by applying Fiat-Shamir heuristic) with the client in which the worker proves the following relation:

$$\{(g, h' \in \mathbb{G}^n, P \in \mathbb{G}, t \in \mathbb{Z}_p;\ l, r \in \mathbb{Z}_p^n):\ P = g^l h'^r \wedge t = \langle l, r \rangle\}$$

On PlatON, VC is coupled with computing channel to guarantee computation integrity. As shown in the figure below.

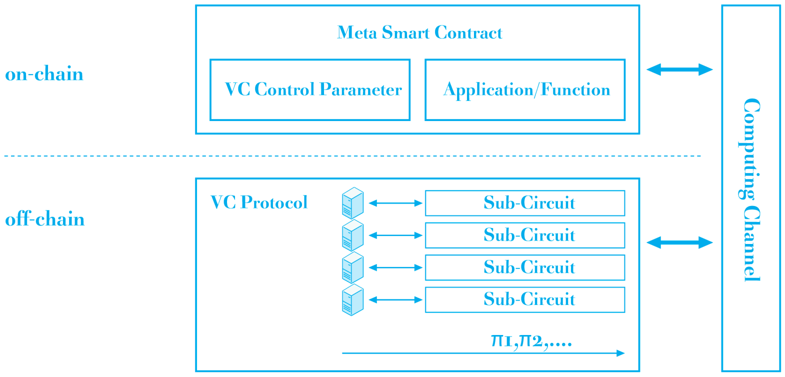

Figure 3.1: Overview of off-chain VC workflow

As shown in Figure 3.1, Clients broadcast computation tasks in the form of a smart contract containing all necessary parameters required for VC and methods of circuit compilation and other corresponding economic incentives. The rest of the meta smart contract would be the specific application or function.

The computing channel transfers the on-chain meta smart contract off-chain. The compiler converts the computing/application part of the meta smart contract to circuits then further splits them into sub-circuits as required by the algorithm. The sub-circuits are distributed to random computing nodes where the computing takes place in parallel for efficiency. To control the computation integrity, the computing node must, during the computing process, also generates a proof at the same time according to the VC algorithm. After submitting the computing results, the proof generated by VC algorithm cryptographically is evaluated to check the correctness of the computation output. In the nature of VC, the time and cost spent on the verification must be much lower than the native computation, plus the significantly reduced overhead on the computing nodes by recent advances in the research of VC, off-chain VC is overall more scalable and cheaper than consensus based on-chain computing.

Through contract as computation, PlatON compiles computation/smart contracts into circuits. Complex computations are broken down into sub-tasks in circuit form. An incentive mechanism together with computation as a contract is then used to persuade idle computing power on the network to compute the sub-tasks. Through VC technology, the results of sub-tasks computed by heterogeneous computing power can be verified for a tiny amount of computing cost. VC links "contract as computation" with "computation as contract" to truly realize the harnessing of global heterogeneous computing power for parallel computing.

Unlike conventional blockchain technology, there is no need for each individual to repeatedly verify transactions through smart contracts. When using VC and smart contracts in circuit form, nodes only need to verify the legitimacy of transactions in a very short amount of time in order to verify whether the new state was computed from the old state through a smart contract.

3.1.2 Privacy-Preserving Computing

On PlatON, secure MPC and HE are combined to achieve complete private-preserving computing, thereby maintain the privacy of the code and data during any operation. Different from other efforts in off-chain computing based on TEE/SGX, data on PlatON is protected through the full cycle and lifetime without any trust boundary.

3.1.2.1 Secure Multi-Party Computation

Secure multi-party computation(MPC) was formally introduced by Andrew Yao in 1982 and 1986. It is defined as the problem of n players to compute an agreed function of their inputs while guaranteeing the correctness of the output as well as the privacy of the players' inputs. Concretely, for n players, each player i knows input x_i ,and they want to jointly compute a pre-agreed function $f(x_1, ..., x_n) = y$, such that all the players learn y , but can get nothing more than that.

One of the most efficient paradigm to build general-purpose MPC protocols is based on the garbled circuits method introduced by Yao for the two-party case and further extended by Beaver, Micali and Rogaway for the multi-party case.

More details of the garbled circuit technique is as follows. For general-purpose constructions, the function f is first represented as a Boolean circuit, without loss of generality, consisting of AND and XOR gates. Since most real-world programs written in advanced language contain complex data structures, this is a highly non-trivial task.

Taking two-party computation as an example, it involves two parties called the generator and evaluator. The generator takes as input the circuit, and writes down all the truth table of each gate. Then For each wire i in the circuit, the generator chooses two uniformly random string (X_i^0, X_i^1), which is called labels, to represent 0 or 1.

After that, the generator will transfer the truth table of each gate g into a garbled gate. Define $g(.,.) : \{0,1\} \times \{0,1\} \to \{0,1\}$ the function of the gate g . Take the AND gate as an example, the function g_{AND} is defined as $g_{AND}(0,0) = 0$, $g_{AND}(0,1) = 0$, $g_{AND}(1,0) = 0$, $g_{AND}(1,1) = 1$. Let x, y, z be the left input, right input and output wire respectively. The generator computes four ciphertexts of each gate g as follows.

$$\begin{aligned}
c_0 &= H(X_x^0 \| X_y^0 \| g) \oplus X_z^{g(0,0)} \\
c_1 &= H(X_x^0 \| X_y^1 \| g) \oplus X_z^{g(0,1)} \\
c_2 &= H(X_x^1 \| X_y^0 \| g) \oplus X_z^{g(1,0)}
\end{aligned}$$

$$c_3 \quad = \quad H(X_x^1 \| X_y^1 \| g) \oplus X_z^{g(1,1)} \tag{3.2}$$

Where H is a cryptographic hash function and g is represented as a gate sequence string when it feeds to the hash function.

After garbling all the gates in the circuits, the generator sends all the garbled ciphertexts together with the input labels related to his input bits.(i.e., 0 for X_i^0 and 1 for X_i^1). Before the evaluator proceeds the garbled circuit, he has to obtain the input labels according to his input information. Since the labels are sampled by the generator, he has to run an oblivious transfer with the generator to get the labels without telling her the inputs in plain. Oblivious transfer is a very fundamental cryptographic primitive, and satisfies the following properties.

1. The generator takes labels X_0, X_1 as input, and The evaluator takes bit $b = 0/1$ as input. At the end of the protocol, the evaluator will get X_b.

2. The generator does not know which label is chosen by the evaluator.

3. The evaluator does not know the other label X_{1-b}.

For each garbled gate, The evaluator tries to decrypts all four ciphertexts with the input labels as keys, note that he could only get one meaningful output label, and then iteratively decrypts the garbled gate to get the output label of the circuit. Since the meaning of the output label of the circuit will reveal by the generator at the very beginnig, the evaluator could get the output of the computation and then send it back to the generator.

Many optimized versions of the garbled circuit are proposed in the literature. The Free-XOR technique enables to garble the XOR gates (almost) for free, thus do not need any encryption on XOR gates. Row reduction and half-gate techniques could reduce the number of ciphertext of AND gates from 4 to 2. Very recent research result using Authenticated Garbled Circuit to achieve very efficient protocols for both two-party and multi-party case in the malicious model.

One common use case is that one party has an algorithm A and the other party has input data x, they want to jointly compute $A(x)$ without compromising other information of A and x. In this case, one should first generate a universal circuit U with the size of A and x as the input. The universal circuit satisfies the property that $U(A, x) = A(x)$, when representing A as a bit string.

Garbled circuit is one of the most efficient techniques for secure multi-party computation. The state-of-the-art implementations could handle millions gates per second in personal computer, which is sufficient for a lot of applications.

MPC provides the fundamental technology module for collaborative computing over private data, integrated with the control layer in meta smart contracts and economic measure applied in the computing channel on PlatON, the technology preserves privacy in applications streaming data from multiple sources, implementing the true private-preserving computation.

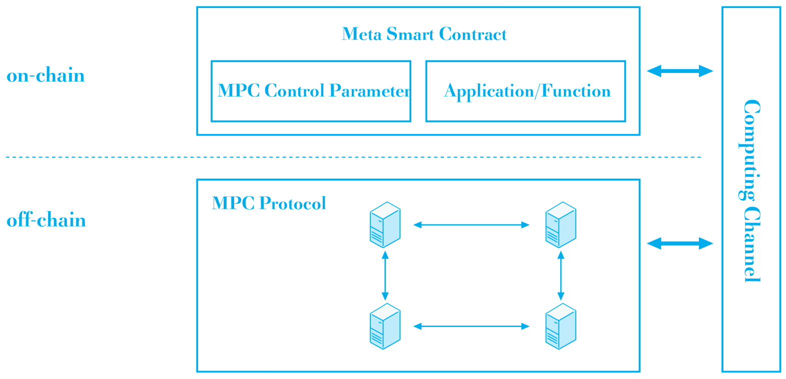

Figure 3.2: Overview of MPC workflow

As shown in Figure 3.2, clients broadcast computation tasks in the form of a smart contract containing all necessary parameters required for MPC and corresponding economic incentives. The rest of the meta smart contract contains information about the specific application or function. The computing channel transfers the on-chain meta smart contract off-chain, where the compiler converts the computing/application part of the meta smart contract to circuits then dispatch them to data providers who simultaneously operate under the MPC protocol, thus keeping the data locally and only the (encrypted) result posted on-chain.

3.1.2.2 Homomorphic Encryption

Homomorphic encryption is a form of encryption that allows computation on ciphertexts. Besides the original components of traditional encryption schemes, there is another evaluation algorithm that takes an aimed function F and encrypted data as input. It will generate an encrypted result, when decrypted, the message is as if performing F on the plaintext in the encrypted data. A cryptosystem that supports arbitrary computation on ciphertext is known as fully homomorphic encryption (FHE). Formally speaking, a homomorphic encryption consists of the following algorithms.

- The $KeyGen(\lambda)$ algorithm takes as input the security parameter, it outputs a public key PK and a secret key SK.
- The $Enc(PK, m)$ algorithm takes as input the public key PK and a message m, it outputs a ciphertext c.
- The $Dec(SK, c)$ algorithm takes as input the secret key SK and a ciphertext, it outputs a message m.
- The $Eval(PK, f, c_1, ..., c_l)$ algorithm takes as input the public key PK, a function f and ciphertexts $c_1, ..., c_l$, it outputs a ciphertext c_f.

Besides the properties of traditional public key encryption schemes. It should satisfy the following property: If $c_1 = Enc(PK, m_1), ..., c_l = Enc(PK, m_l)$, $m_f = f(m_1, ..., m_l)$ and $c_f = Eval(PK, f, c_1, ..., c_l)$, then it should have: $m_f = Dec(SK, c_f)$.

An encryption is called fully homomorphic if for any function f, the above properties are satisfied.

Existing FHE schemes follows the blueprint of Craig Gentry's bootstrapping framework. Gentry's amazing "Bootstrapping theorem" says that if a scheme is homomorhpic enough to evaluate its own decryption circuit, then it can be turned into fully homomorphic encryption that can evaluate any function.

Based on Gentry's Bootstrapping theorem, the main goal to design FHE scheme is to construct a somewhat homomorphic encryption which is capable to handle it's own decryption circuit. One of the state-of-the-art constructions is based on the (ring) learning with errors (LWE) problem. More specifically, let n be a power of 2, define a cyclotomic ring $R = \mathbb{Z}[X]/(X^n + 1)$, for an integer $q \geq 2$, let $R_q = R/qR$. Define the message space as R_2. The public key is a pair of ring element $(a, b) \in R_q^2$, where $b = -a \cdot s - 2e$, the secret key is $\vec{s} = (s, -1) \in R^2$, s, e are sampled from discrete Gaussian distribution.

Given a message $m \in R_2$, the ciphertext is of the form $\vec{c} = (c_0, c_1)$, where $c_0 = a \cdot r + 2e_0$, $c_1 = b \cdot r + 2e_1 + m$, r, e_0, e_1 are sampled from discrete Gaussian distribution.

The decryption of the ciphertext is as follows:

$$m = [[\langle \vec{c}, \vec{s} \rangle]_q]_2,$$

where $[\cdot]_q$ means all operations are done in $\mathbb{Z}_p$ for $q \geq 2$. Note that, as long as the norm of $m + 2e_1 - 2e_0 \cdot s$ is sufficiently smaller than q, then

$$[\langle \vec{c}, \vec{s} \rangle]_q = [m + 2e_1 - 2e_0 \cdot s]_q = m + 2e_1 - 2e_0 \cdot s$$

and the decryption equation holds.

To handle the homomorphic evaluation of the ciphertexts, one should first represent the function f as a arithmetic circuit or Boolean circuit.Without loss of generality, arithmetic circuit over R_2 is used here. Therefore, one needs only consider addition gate and multiplication gate in R_2.

Given two ciphertext ,$\vec{c}_0, \vec{c}_1$, to evaluate the addition gate, one just needs to add the vectors, i.e.,$\vec{c}_{add} = \vec{c}_0 + \vec{c}_1 \in R_q$. This is because

$$\langle \vec{c}_{add}, \vec{s} \rangle = \langle \vec{c}_0 + \vec{c}_1, \vec{s} \rangle = \langle \vec{c}_0, \vec{s} \rangle + \langle \vec{c}_1, \vec{s} \rangle$$

To evaluate the multiplication gate, it is more involved. Notice that for a ciphertext $\vec{c}$, it should satisfy that $[\langle \vec{c}, \vec{s} \rangle]_q = m + 2e'$ for some "error" e' . Then we have that

$$\langle \vec{c}_0 \otimes \vec{c}_1, \vec{s} \otimes \vec{s} \rangle = \langle \vec{c}_0, \vec{s} \rangle \cdot \langle \vec{c}_1, \vec{s} \rangle = (m_0 + 2e_0') \cdot (m_1 + 2e_1') = [m_0 m_1 + 2e'']_q$$

for some "error" e'' , where $\otimes$ is tensor product. This means that $\vec{c}_0 \otimes \vec{c}_1$ is a "ciphertext" of $m_0 m_1$ under secret key $\vec{s} \otimes \vec{s}$. To shrink the length and transform it back to a "normal" ciphertext, a Key Switching technique is used. Basically, the Key Switching technique transforms ciphertext $\vec{c}_0 \otimes \vec{c}_1$ under secret key $\vec{s} \otimes \vec{s}$ into a ciphertext $\vec{c}_{mult}$ under secret key $\vec{s}$ with the same plaintext. Basically $\vec{c}_{mult}$ is obtained as $\vec{c}_{mult} = W \cdot (\vec{c}_0 \otimes \vec{c}_1)$, where W is a matrix and could be viewed as ciphertexts of each element in $\vec{s} \otimes \vec{s}$ under secret key $\vec{s}$.

In the (R)LWE construct, error management is the most important issue to do homomorphic operations. After a few operations (especially for multiplication gate), the error will increase and close to q and are too large to decrypt. To handle this problem, another approach called Modulus Switching is proposed. Briefly speaking, it transforms a ciphertext $\vec{c}$ under modulus q into a new ciphertext $\vec{c}'$ under modulus q' with the same plaintext, and $q' < q$. This is simply obtained by $\vec{c}' = \left\lfloor \frac{q'}{q} \cdot \vec{c} \right\rceil$. Using smaller modulus will shallow the decrypt circuit and also is capable to homomorphically evaluate it with the modulus switching technique.

On PlatON, the privacy of a single source data provider can be protected. Under FHE, the data contributed by the provider in the controller layer of the meta smart contract and computing channel are endowed with complete confidentiality when transferred off-chain for verifiable computation.

Figure 3.3 depicts the workflow of verifiable computing of a meta smart contract under FHE. The client writes two parts of information into the smart contract, the first part contains HE control parameters and incentive mechanism, the other part describes the computation task. The information from the smart contract, once transferred off-chain through the computing channel, is in turn compiled to circuits, dispatched in the form of sub-circuits along with data encrypted under FHE from data providers to multiple computing nodes or a single computing node for execution. VC protocol must be performed to assure the computation integrity before the encrypted results is posted on-chain.

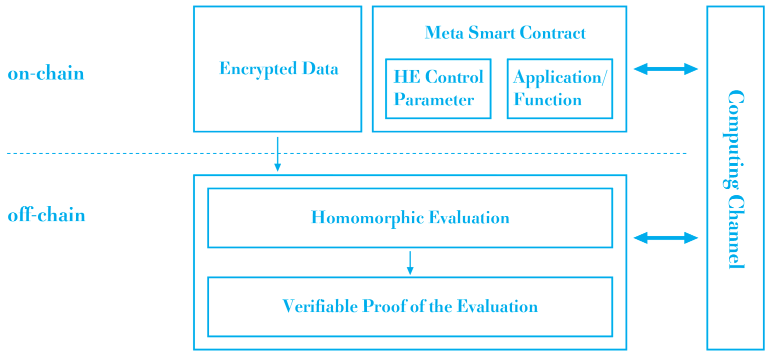

Figure 3.3: Overview of the off-chain verifiable computing workflow under FHE

3.1.3 Parallel Computing

PlatON breaks computation functions in the meta smart contract down into smaller computation tasks that are then distributed to different computing nodes to be executed in parallel, providing high performance and linear scalability. To guarantee execution of the computation task within a certain time, every sub-circuit will be assigned to multiple nodes to mitigate the risk of disconnection or overtime error, given the prevailing availability of the participating nodes.

3.2 Circuits

In a circuit, input and output wires are connected through different types of gates to form a "complex directed acyclic network". A circuit made up of logic gates (e.g. AND, OR, NOT, and NAND) is referred to as a Boolean Circuit; a circuit made up of arithmetic gates (e.g. addition, multiplication) is referred to as an Arithmetic Circuit.

All forms of computation can be translated into circuits made up of limited types of gates for any level of complexity. Circuits are widely used as computing basis in cryptography due to the simplicity of its basic components.

Circuits serve as the nexus of complex networks in PlatON. It is the common computing format shared by MPC, ZKP, VC and HE, connecting all kinds of algorithms and hardware through its universality.

Circuits are the basic units of measurement for ”computation” in PlatON. Gates are the basic elements of circuits, the energy consumption for different gates varies. The total energy consumption for each computation can be the expressed as the sum of energy consumed by all the gates in the circuit, providing a benchmark to measure and price computation.

3.3 Computing Specific Hardware

Every meta smart contract in PlatON is compiled into corresponding circuits. Gates of circuits are a natural fit with the hardware architecture. Converting functions into circuits and executing these gates on a hardware platform will greatly increase calculation efficiency and reduce power consumption/cost.

PlatON will roll out computing specific hardware based on FPGA/ASIC at a certain stage of the project, as a key component in the next generation architecture. Once implemented, we expect a great improvement in the performance of the platform.

3.4 Base Chain

3.4.1 Decoupling Computation from Consensus

As shown in Figure 3.4, PlatON decouples the contract execution and consensus of the blockchain, building an off-chain scalable trustless computing network. Such design further dissipates the power in the blockchain in a more decentralized manner thus escalate the security of the network, while allowing independent upgradation of both functions for continuous improvement.

The smart contract on PlatON uses IO logic to operate on-chain and computation function to depict the computation task which is in turn compiled into Boolean Circuits, divided into multiple sub-circuits then sent to computing nodes to be executed in parallel. The computing nodes are randomly selected for the computation tasks, and for

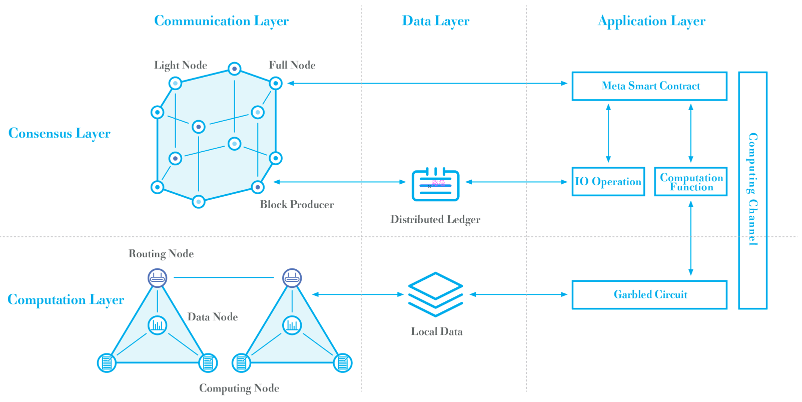

Figure 3.4: Decoupling of Computing and Consensus Layers in PlatON

availability guarantee, each sub-circuit is always sent to multiple nodes to maintain certain level of redundancy.

3.4.2 Computing Channel

Computing nodes are required to deposit in the computing channel a corresponding amount of crypto asset to the value of computation tasks that they are given. The deposit will be returned if the computational proof is verified or slashed if not.

Computing channel is a system smart contract, or regarded as computation state machine, in charge of tracking the state of the computation tasks. It serves as a deterministic program to assure termination and settle of computation and issues reward or punishment according to the correctness of results.

3.4.3 Multichain Architecture

Multiple appchains can be created on PlatON base chain by sharding. The appchains supports independents business activities and extend in parallel to each other. The transactions from multiple chains are packaged parallelly with their block head store on the base chain to achieve consensus.

Chapter 4

Technical Architecture

4.1 PlatON Network Protocol

4.1.1 Network Protocol Stack

The basic implementation of PlatON network is a decentralized structured topology completely based on RELOAD (Resource LOcation And Discovery) based protocol and the Kademlia protocol [**Kademlia**]. Shown in Figure 4.1 are the layers of the overall PlatON network structure.

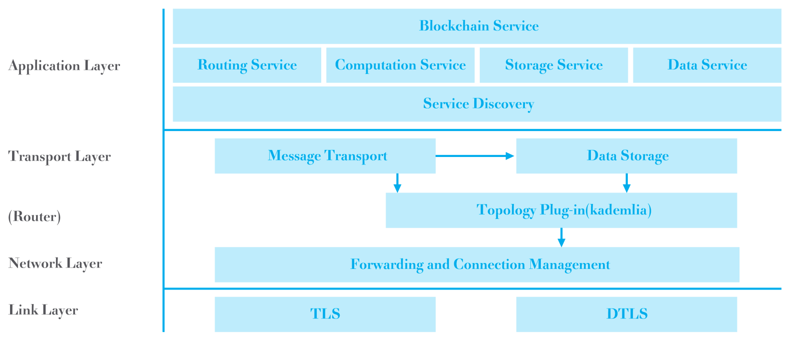

Figure 4.1: PlatON network protocol stack (based on[RFC6940])

4.1.1.1 Link Layer

The Link Layer ensures the secure transfer of data. A variety of transmission protocols are employed to prevent eavesdropping, tampering and spoofing; to provide secure and authenticated connections; and to verify the source of messages and ensure the integrity of the data.

Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) are implemented on this layer.

PlatON offers a plug-and-play mechanism for encryption algorithms that supports standard international algorithms (including SHA256, SHA3, ECDSA, RSA, 3DES, AES, RSA-OAEP, and ECIES) on demand. China's national cryptographic algorithms (SM2, SM3, SM4, SM9 etc.) will also be supported.

4.1.1.2 Forwarding and Connection Management

The Forwarding and Connection Management layer stores and implements the Routing Table by providing packet forwarding services between nodes. It also handles establishing new links between nodes, setting up connections for overlay links across NATs using ICE.

4.1.1.3 Topology Plug-in

RELOAD is a P2P network framework that supports the development of different topology algorithms for implementing a fully-distributed non-structured topological or fully-distributed structured topological network.

The Topology Plug-in is responsible for implementing the specific overlay algorithm being used. It uses the Message Transport component to send and receive overlay management messages, the Storage component to manage data replication, and the forwarding and connection management layer to control hop-by-hop message forwarding.

The Topology Plug-in allows RELOAD to support a variety of overlay algorithms. PlatON implements a DHT based on Kademlia algorithm.

4.1.1.4 Data Storage

The Data Storage Layer is responsible for processing messages relating to the storage and retrieval of data. It talks directly to the Topology Plug-in to manage data replication and migration, and it talks to the Message Transport component to send and receive messages.

The base RELOAD protocol currently defines three data models: single value, array and dictionary.

4.1.1.5 Message Transport

The Message Transport layer is responsible for handling end-to-end reliability, managing request state for the usages, and forwarding Store and Fetch operations to the Data Storage layer. It delivers message responses to the component initiating the request.

PlatON uses RELOAD as the basis for developing a Regional Flooding algorithm that broadcasts messages quickly throughout the entire network.

4.1.1.6 Application Layer

The communication and storage capabilities of the RELOAD base layer are used to provide service discovery and scaling as well as routing, computing, data, storage and blockchain services based on service discovery.

The following chapters describe the network protocols of each service on the application layer.

4.1.2 Service Discovery

In a peer-to-peer (P2P) overlay network such as PlatON, the peers forming the overlay share their resources in order to provide the service the system has been designed to provide. The peers in the overlay both provide services to other peers and request services from other peers. Possible services peers in PlatON can offer to each other include a TURN relay service, an audio/video service, a computing service, a SIP agent service, and so on.

A peer that wishes to use a particular service faces the problem of finding peers that are providing that service from the Overlay Instance. It is here that service discovery plays a key role.

4.1.2.1 ReDiR Tree

A naive way to perform service discovery is to store the Node-IDs of all nodes providing a particular service into the DHT under a well-known key. The limitation of this approach is that it overloads the nodes storing pointers to service providers since all service lookup requests for services will need to be answered by the node responsible for services.

PlatON uses the ReDiR (Recursive Distributed Rendezvous) [RFC7374] to implement a service discovery mechanism. The ReDiR-based service discovery mechanism is suitable for use even in overlay networks where the number of end users that may make service discovery requests can be very high (e.g., tens of thousands of nodes or even higher) and where a large fraction of the peers can offer the service.

The ReDiR-based service discovery mechanism is implemented as a tree structure of the nodes that provide a particular service. The nodes embed the ReDiR tree into the RELOAD Overlay Instance using RELOAD Store and Fetch requests. Only several service lookups can get the closest service provider node.

Each tree node in the ReDiR tree contains a dictionary of entries of peers providing a particular service. Each tree node in the ReDiR tree also belongs to some level in the tree. The root node of the ReDiR tree is located at level 0. The child nodes of the root node are located at level 1 of the ReDiR tree. The children of the tree nodes at level 1 are located at level 2, and so forth.

The ReDiR tree has a branching factor b. At every level lvl in the ReDiR tree, there is room for a maximum of tree nodes. Each tree node in the ReDiR tree is uniquely identified by a pair, where lvl is a level in the ReDiR tree and j is the position of the tree node (from the left) at that level. Each level in the ReDiR tree contains b^{lvl} key spaces.

All services providers are mapped into corresponding key space. A tree node is responsible for the storage of each key space. Tree node contains key space

$$(2^{\texttt{BitsInKEY}} b^{-lvl}(j + \frac{b'}{b}),\ 2^{\texttt{BitsInKEY}} b^{-lvl}(j + \frac{b'+1}{b}))$$

for $0 \le b' < b$. Tree node stores resource $id = \texttt{hash}(service, lvl, j)$.

Figure 4.2 shows an example of a ReDiR tree whose branching factor is 2.

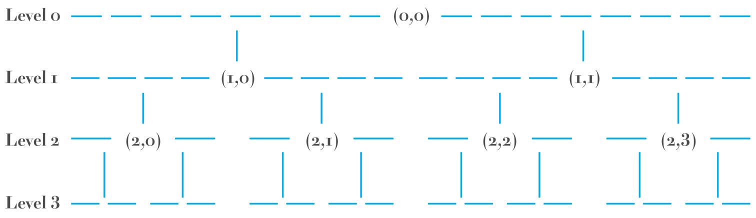

Figure 4.2: A ReDIR tree with a branching factor of 2 (based on [RFC7374])

4.1.2.2 Service Registration

A node n with key k use the following procedure to register as a service provider in the RELOAD Overlay Instance:

- Starting at some level $l = l_{start}$. This is generally 2.
- Node n sends a RELOAD Fetch request to fetch the contents of the tree node responsible for key space $I(l, k)$.
- Node n sends a RELOAD Store request add its entry to the dictionary stored in the tree node responsible for key space $I(l, k)$.
- If node n's key is the lowest or highest key stored in the tree node responsible for key space, node n MUST reduce the current level by one , and continue up the ReDiR tree towards the root level (level 0), repeating steps 2 and 3 above. Node n continue in this way until it reaches either the root of the tree or a level at which k is not the lowest or highest key in the key space.

- In the same way, node n also performs a downward walk from level $l = l_{start}$ recursively until the following condition is satisfied: node n is the only service provider in the tree node responsible for key space $I(l, k)$.

4.1.2.3 Service Refresh

All state in the ReDiR tree is soft. Therefore, a service provider needs to periodically repeat the registration process to refresh its Resource Record. If a record expires, it must be dropped from the dictionary by the peer storing the tree node.

4.1.2.4 Service Lookup

A service lookup is similar to service registration. We again start at some level . At each step we get the current key space and determine where to look next as follows:

- If there is no service provider stored in the tree node associated with , then service provider must occur in a larger range of the keyspace, so we set and repeat, or fail if level is equal to 0.
- If k is sandwiched between two client entries in , then the service provider must lie somewhere in a sub-space of . We set and repeat.
- Otherwise, the returned result must be the service provider closest to key (k) and the lookup is done.

4.1.3 Computing Service

Effective high-speed communication between computing nodes becomes very important in PlatON. Computing services based on the RELOAD protocol implements the publication of computing services, the discovery of computing services, and the creation of computing sessions.

4.1.3.1 Publication of Computing Service

When a computing node joins the PlatON to provide its computing service, before it registers as a computing service provider it must use the STUN protocol [RFC5389] to determine whether it is located behind a NAT device. If it is already behind a NAT device then it must use service discovery to find a TURN service provider in order to obtain a relay address. Computing nodes must register its own IP address or the relay address with the PlatON network.

When a node publishes its computing service, it uses the service discovery protocol to carry out service publication and refreshes. The node publishes itself to the key space $I(l, \texttt{power})$ with `power` being the computing power provided.

4.1.3.2 Discovery of Computing Service

PlatON computations must lookup matched computing service provider with required computing power in key space , using the service discovery protocol.

4.1.3.3 Distrubution of Computation Tasks

PlatON is packed with a RELOAD-based computation task distribution protocol. Computation tasks are distributed to computing nodes that provide computing services through P2P communications. To ensure computation reliability and performance, a certain degree of redundancy is maintained during the distribution of computation tasks. In other words, computation tasks are distributed to multiple computing nodes at the same time.

4.1.3.4 Secure Multi-Party Computation Protocol

PlatON is packaged with RELOAD-based GC and OT protocols to support MPC. Computing sessions are created by multiple parties over the SIP protocol.

4.1.4 Blockchain Service

Blockchain nodes use the message transport component of the RELOAD framework to synchronize transaction and block. Block producers also use the base RELOAD protocol to communicate with each others.

4.1.4.1 Adding Blockchain Nodes

Multiple blockchain nodes can operate concurrently on the PlatON network. Blockchain nodes exist as a special type of ”service” within the PlatON network. When a node chooses to join a designated blockchain, it must use the service service discovery mechanism to publish itself as a provider of the designated blockchain service (service name must be the name of the designated blockchain).

Client-initiated transactions can use the service discovery mechanism with the blockchain name to find and launch a blockchain transaction with a node for the designated blockchain.

4.1.4.2 Forwarding of Transaction

The rapid broadcast capability of the RELOAD message transport component is used to quickly disseminate blockchain transactions throughout the entire network and pack them into a block.

4.1.4.3 Block Synchronization

Every full node in PlatON maintains a complete copy of blocks data. Once block consensus is achieved, it is broadcast throughout the entire RELOAD overlay network. Once it has been received and verified by each node, it is stored locally.

Synchronization of blocks data is also based upon the base RELOAD protocol.

4.1.4.4 Proof of Verifiable Computation – Giskard

In the Giskard consensus algorithm, block producers are elected using their weighted computingl contribution. This election action is a blockchain transaction. block producers produce and verify blocks through the asynchronous BFT protocol which is based upon the base RELOAD protocol.

4.2 PlatON Network Structure

As shown in Figure 4.3, the PlatON network is a RELOAD overlay network and all nodes must join the RELOAD overlay network,each

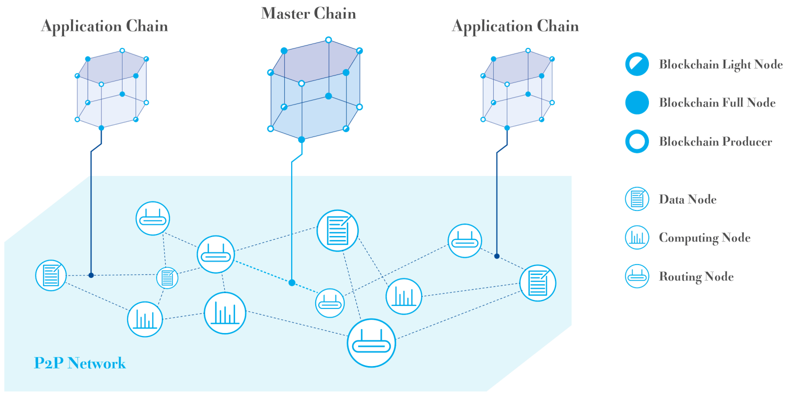

Figure 4.3: PlatON Network Structure

node has its own node ID , So, the routing information for any node in the RELOAD overlay network can be found using its node ID.

There are multiple logical blockchains on the RELOAD overlay network. Each node can choose to join one or more blockchains and become a blockchain node. Each blockchain in the RELOAD overlay network is a type of service; joining a blockchain means the node has registered itself as a provider of the corresponding blockchain service on that RELOAD overlay network.

Nodes in the PlatON network can also provide certain services independent of the blockchain, including TURN, SIP, computing, data and storage services. These services can be used by all blockchains on the network. Service-providing nodes can obtain a variety of different measurable economic and community incentives.

4.2.1 PlatON Nodes

Nodes in the PlatON network can be divided into two types based on the kind of service they provide.

4.2.1.1 Basic Services Nodes

- **Computing Nodes**
  Nodes that provide computing services to the PlatON network.

They are responsible for completing all kinds of computation tasks.

- **Data Nodes**
  Nodes that provide data services to the PlatON network. They are responsible for providing data to all kinds of computation tasks.

- **Routing Nodes**
  Nodes in the PlatON network can be deployed on private networks. Nodes within a private network can implement NAT Traversal through routing nodes. Routing nodes provide STUN and TURN services.

4.2.1.2 Blockchain Nodes

- **Light Nodes**
  They only store block header and data related to the node itself rather than storing all block data. They rely on full nodes for rapid transaction verification. Light nodes participate in the network-wide broadcast of transactions and blocks.

- **Full Nodes**
  They store all block data. Transactions can be verified locally. Full nodes participate in the network-wide broadcast of transaction and blocks.

- **Block Producers**
  They are responsible for carrying out transactions and packaging transactions data into block. In the Giskard consensus protocol, block producers are elected through their weighted computing contributions and produce and verify blocks through the asynchronous BFT protocol.

4.2.2 Multi-Chain Routing Mechanism

In the PlatON network, blockchain services adopt the concept of "Single access point for network-wide service". In other words, a PlatON user can send a transaction from any node on the network to any application chain without having to actually establish a direct connection with the target application chain node. This implies that blockchain routing is transparent to the user.

In the PlatON network, each blockchain is a type of service. When a blockchain node joins the designated blockchain, it registers itself as a service provider for the corresponding application chain. The blockchain node uses the service discovery protocol for service publication and refreshing. The node publishes itself to the key space $I(l, \texttt{NodeID})$ with `NodeID` being the blockchain node ID. Each blockchain ultimately forms a ReDIR tree in the PlatON network.

PlatON users can connect to any node on the network and use the node to lookup the key space $I(l, random)$ of the designated blockchain's ReDIR tree, in which random is the random hash value that is used to randomly find a node in the specified blockchain. If the lookup is successful then a transaction can be launched directly with the blockchain of the corresponding node.

4.3 Meta Computing Framework– Monad

The computation essentially consists of the algorithm, data, and computing power.

The monad aims to effectively integrate heterogeneous algorithms, data and computating resources all over the world and then engage in extensive and in-depth transactions of data and computing power.

The monad not only uses parallel computing and computing specific hardware to improve computation performance, a variety of cryptographic algorithms such as MPC, VC, HE, and ZKP are also integrated to ensure verifiable computation and data privacy.

4.3.1 What is Monad

The monad is a scaling solution that brings scalable computation and transaction throughput to blockchain. The monad implements off-chain computing for linear computing scalability, and integrates a variety of cryptographic algorithms for privacy and verifiability, and uses parallel computing and computing specific hardware for high performance.

4.3.2 Participants

Monad participants are divided by their ability type into: computing requester, algorithm provider, data provider, computing power provider and computing collaborator.

4.3.2.1 Computing Requester

This generally refers to external users that call the meta smart contract through the client terminal to trigger computation.

4.3.2.2 Algorithm Provider

Refers to the publisher of the meta smart contract. The algorithms contained within the meta smart contract defines computation functions and input/output format. Computation functions are compiled into circuit. The algorithms are published along with the meta smart contract to the PlatON platform.

4.3.2.3 Data Provider

In the PlatON platform, the data provider is also referred to as the data node that stores data in local database.

The data provider provides the data used for computation based on the input format defined by the algorithm.

The block producer is a special type of data provider and is in fact the provider of on-chain data.

4.3.2.4 Computing Power Provider

Accepts and executes computation tasks (including algorithm and data). Also known as the computing node in the PlatON network.

4.3.2.5 Computing Collaborator

The computing collaborator is responsible for obtaining the data and then combining the data with the computation function to form computation tasks. Computation tasks are then distributed to computing power providers for computation. The computing collaborator is generally a data provider as well.

4.3.3 Computation Tasks

4.3.3.1 Data Source

The data of computation tasks can be divided into sole-source data and multi-source data.

- **Sole-Source Data**
  Data comes from a single data provider. If the data provider is a block producer then the data source is on-chain.

- **Multi-Source Data**
  Data comes from multiple on-chain and off-chain data providers.

4.3.3.2 Computation Classification

Different types of computation tasks are supported by monad. These can be divided into verifiable proxy computation and privacy-preserving proxy computation.

- **Verifiable Proxy Computation**
  The privacy of the data does not need to be guaranteed during the computation process. Once the computation is completed by the computing power provider, the proof of correctness is returned along with the results for verification by the computing requester.

  In the case of multi-source data, each data provider carries out their own verifiable proxy computation.

- **Privacy-Preserving Proxy Computation**
  The privacy of the data must be guaranteed during the computation process. Sole-source and multi-source data can both be used with privacy-preserving proxy computation though different cryptographic algorithms are used. These will be described separately in a later chapter.

4.3.4 Computing Channels

Computing channels in PlatON are implemented as a special contract that ensures the validity of the computation through cryptographic proof and timelocks.

Computing channels will usually deduct a set amount of the transaction requester's digital assets to serve as a deposit for the computation task. Once the task has ended and settlement has been completed the remaining digital assets are returned to the transaction requester.

At the end of computation, all participants can send a transaction to the computing channel contract. The channel can then be closed and the settlement process triggered. The computing channel does not carry out settlement right away however. A timed window is open, during which time participant can submit a dispute over the computation process.

The blockchain can therefore be considered to be a kind of "cryptographic court" and serve as an arbitration method with higher costs on the PlatON network. As the final method for providing transaction security, this greatly reduces the incentive for fraud and forgery. Proof of work and proof of transaction results verified over a long period of time already provide an excellent guarantee for validity of computation. This particular arbitration method therefore complements and provides an ultimate guarantee of transaction security.

4.3.5 Execution of Computation Tasks

4.3.5.1 Allocation of Computing Resources

When a computing node joins the network, it publishes itself as a computing service. The node will automatically detect and measure its computing power that will be published as computing **service capacity**.

When computation tasks are being distributed by **computing collaborator**, the first step is to lookup matched computing nodes with enough computing power. The **service discovery** protocol is used to lookup computing nodes that match the computing power requirements at random on the network. Computing nodes with high contribution scores are then selected based on their computating contribution ranking. To guarantee fairness and encourage more nodes to provide computing services, some of the newly added computing nodes will be selected at random as well.

4.3.5.2 Verifiable Proxy Computation

The algorithm provider can specify "verifiable" computational requirement when releasing the computation function. When the contract is being executed, the computing collaborator sets up the parameters of the VC algorithm and breaks the circuits in the contract down into multiple sub-circuits.

The computing collaborator combines the VC algorithm parameters, sub-circuits, and input data into multiple sub-asks that are then distributed to multiple computing nodes. A single sub-task will be simultaneously distributed to multiple computing nodes to guarantee the soundness of the computation. A certain amount of redundancy in computation is retained for cross-validation.

While the sub-task has been computed, a proof of correctness is generated through the VC algorithm by computing nodes and returned to the computing collaborator. Once the proof verified, the gates number of sub-circuit is translated into the value of computing contribution that will be credited to the account of the computing power provider. The computing power provider then is rewarded proportionately.The Verifiable Proxy Computation Process is shown in Figure 4.4.

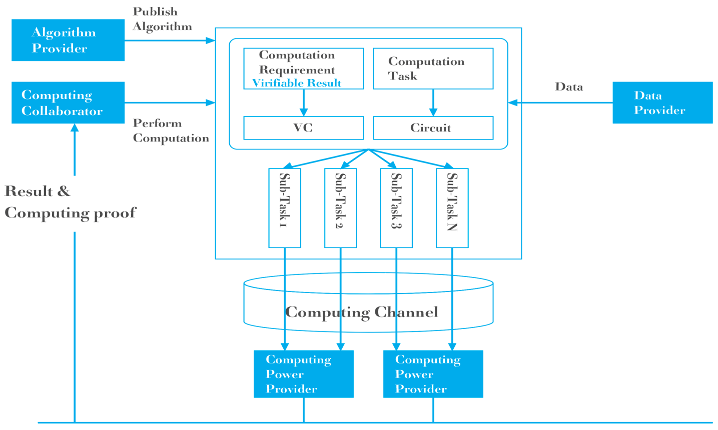

Figure 4.4: Verifiable Proxy Computation Process

4.3.5.3 Sole-Source Data Privacy-Preserving Computation

Sole data sources can make use of all computing resources on the network without revealing the raw data. When the contract is being executed, the **computing collaborator** sets up the parameters of the homomorphic and VC algorithms then breaks the circuits in the contract down into multiple sub-circuits.

The input data encrypted using homomorphic encryption algorithm is combined with the VC algorithm parameters, sub-circuits into multiple sub-asks that are then distributed to multiple computing nodes. A single sub-task will be simultaneously distributed to multiple computing nodes to guarantee the soundness of the computation. A certain amount of redundancy in computation is retained for cross-validation.

The computing node performs homomorphic computation on the ciphertext based on the sub-circuits. The VC algorithm is also used to prove the correctness of execution. The encrypted computation results and proof of correctness are returned to **the computing collaborator**. The computing collaborator first verifies the proof before decoding the ciphertext to obtain the result. Once the proof verified, the gates number of sub-circuit is translated into the value of computing contribution that will be credited to the account of the computing power provider, the computing power provider then is rewarded proportionately.The Privacy-Preserving proxy computation process for sole-source data is shown in Figure 4.5.

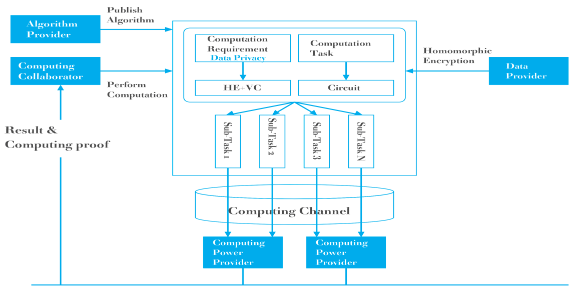

Figure 4.5: Privacy-Preserving proxy computation process for sole-source data

4.3.5.4 Multi-Source Data Privacy-Preserving Proxy Computation

Multi-Source computation tasks can be computed collaboratively by multiple data providers to obtain the results while also guaranteeing data ownership and privacy through the MPC algorithm.

The classic Yao's Garble Circuit algorithm involves generator and evaluator.

The generator select a random number as label for each wire in the circuit, and encrypts the output label with the input label for each gate, and then finally, circuit is converted into garbled circuit. Evaluator obtains the labels correlated with both input data through OT protocol, and deciphers the garbled circuit to get the ultimate result.

As GC generator, The block producer distributes computation tasks, setting up GC and VC algorithm parameters when the contract is being executed. The block producer selects labels randomly and breaks the circuits down into multiple sub-circuits. Sub-circuit is combined with labels into multiple sub-asks that are then distributed to multiple computing nodes. Computing nodes performing the sub-task computation must also use the VC algorithm to generate a proof of correctness and return it to the block producer for verification.

At the same time, as a GC evaluator, the data provider obtains the labels related to its inputs data through the OT protocol with the GC generator. This process can be conducted concurrently with the distributing of computation tasks by the GC generator.

After obtaining the labels and garbled circuits, GC evaluator can publish a computation task for deciphering the circuit as the GC generator publish„ except that it must provide a GC decryption algorithm for its computational requirement.

An incentive mechanism is also used to encourage worldwide computing nodes to perform the resource-consuming encryption and decryption of circuits concurrently, and then idle computing resources can be fully utilized, and the data flowing and privacy protection can get a massive boost. The Privacy-preserving proxy computation process for multi-source data is shown in Figure 4.6.

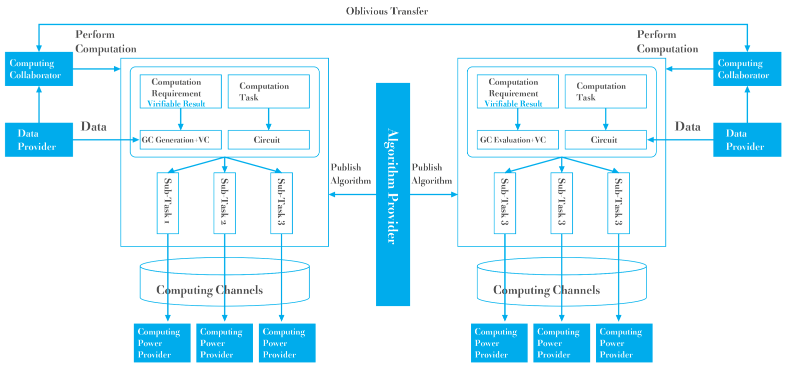

Figure 4.6: Privacy-Preserving proxy computation process for multi-source data

4.3.6 Computing Specific Hardware

Every meta smart contract in PlatON is compiled into corresponding circuits. Gates of circuits are a natural fit with the hardware architecture. Converting functions into circuits and executing these gates on a hardware platform will greatly increase calculation efficiency and reduce power consumption/cost.

PlatON will divide development and rolling-out of computing specific hardware into the following stages:

- **Stage 1:** Design the IP for suitable hardware and implement the corresponding hardware devices through FPGA.
- **Stage 2:** License IP to ASIC/ASSP/SOC manufacturers and help them incorporate support for meta computing framework acceleration into their designs.
- **Stage 3:** Implement the ASIC chip to provide ecology partners with an IC-grade solution.

To promote the full-scale roll-out of the next-generation computing framework, PlatON will maintain an open approach to cooperating with all types of eco-system partners. We strongly believe that true improvements to network performance can only be achieved through collaborative hardware evolution. Only then can our vision for a next-generation computing framework be realized.

4.4 Proof of Verifiable Computation – Giskard

To avoid wasting computing power and improve consensus performance, the PoW method is not used in Giskard. PlatON will select some nodes to participate in the consensus by means of election and randomization.

To select nodes that are truly active and honest, Giskard elections are based on the joint weighted stake of the value of computing contribution. The value of computing contribution can be used to measure the contribution and honesty of a node.

4.4.1 The Value of Computing Contribution

The account in PlatON has multiple attributes such as fund balance, computing contribution, and so forth. Such an account have contributed to depict user's behaviors throughout the network more accurately.

The value of computing contribution is the amount of verified computing power provided by computing nodes. All computations in PlatON must be interpreted as circuits. Cost of computation is directly proportional to the number of gates whose weights are different. Corresponding to different weights, every gate has different computing contribution score. The value of computing contribution of a computing node is the total score of all gates in circuits the computing node executed.

4.4.2 Proof of Verifiable Computation

"Those who labor with their minds become rulers; those who labor with their bodies become the ruled." The fully digital world will give a completely new twist and interpretation to such a governance structure.

Computation is the basic process and essence of everything. Though those who labor with their minds (the thinkers) or bodies (the laborers) appear to belong to different layers, they have all essentially participants or initiators of computation processes.

We believe the value is created through work in the PlatON network. All the power that thinkers possess originate from the laborers and they should be generated from the laborers as well. All of the decisions they generate should and must satisfy the natural interest of the laborers as well. There is of course a variety of ways for creating value from work. An inclusive network should be compatible with different scales of measurement, including calculating the dimensions of computing contribution when assessing their overall weighting. Such an ideal can be guaranteed through the use of a suitable algorithmic mechanism.

In PlatON, some block producers are elected through continuous real-time voting, another block producers are randomly selected by Verifiable Random Functions(VRF). Every node has the right to vote, the number of votes is determined by the joint weighted value of the stake, the value of computing contribution, and so forth. The election process is repeated in the next cycle.

Block producers use an asynchronous BFT algorithm to generate blocks and form a consensus. Malicious block producers will be disqualified and a certain amount of computing contribution and Energon will be deducted as punishment. Block producers split and distribute the computation functions in the contract to multiple computing nodes when generating blocks. Each computing node returns the result as well as a proof of the computation. The result and proof is packaged by the producer to the block. Other nodes only need to verify the proof to determine the validity of the block. Block verification time is therefore greatly reduced and transaction performance improved.

The election, voting and block generation mechanism for candidate blocks is not only a technical problem but also a multiple-choice question on community ideals and governance model. This is an area which this technical white paper will not look into too great detail here. Community feedback will instead form the basis for proposals on ecology governance that will be released at appropriate times.

4.5 Meta Smart Contract – Sophia

Smart contract in PlatON is completely different from the conventional one. An innovative ”Meta Smart Contract” is proposed by PlatON as a computing-oriented smart contract for the computing world.

4.5.1 Types of Meta Smart Contract

PlatON is essentially a distributed service platform with a server-less, decentralized topology. The meta smart contract is then a Functions as a Service (FaaS) application deployed on the network.

Users can publish their services to the outside world by simply submitting their meta smart contract code to the PlatON platform. They will only need to pay for the resources consumed during the execution of that code.

PlatON divides the meta smart contract into the three following types.

4.5.1.1 Stateful Contract

A stateful contract is similar to the conventional smart contract.

This type of smart contract needs to preserve its state on the chain. When a stateful contract is executed, the input data comes from the distributed ledger on the chain. Every time the contract is executed it changes the state of the contract. All changes are also recorded in the distributed ledger.

As shown in Figure 4.7, contract computations are broken down into multiple sub-tasks and distributed to multiple computing nodes. The contract developer can choose the privacy-preserving computation

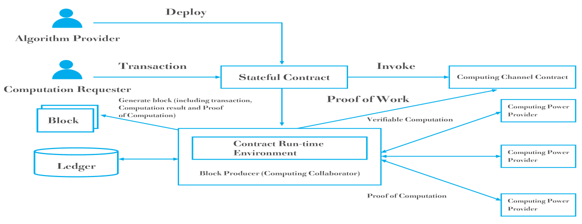

Figure 4.7: Stateful Contract

mode to ensure that the data is not revealed to the computing nodes.

4.5.1.2 Stateless Contract

Stateless contract does not preserve any states on the chain.

As shown in Figure 4.8, when a stateless contract is executed, the input data comes from the local database of the off-chain data provider. It can be a single source data provider or a multiple source data providers.

The computing process for a single source data providers is identical to that of the stateful contract; if there are multiple data providers, then the MPC protocol is used for multi-party collaborative computing to guarantee ownership over their respective data. Actual computations are broken down into multiple sub-tasks and distributed to multiple computing nodes. The contract developer can choose the privacy-preserving computation mode to ensure that the data is not revealed to the computing node.

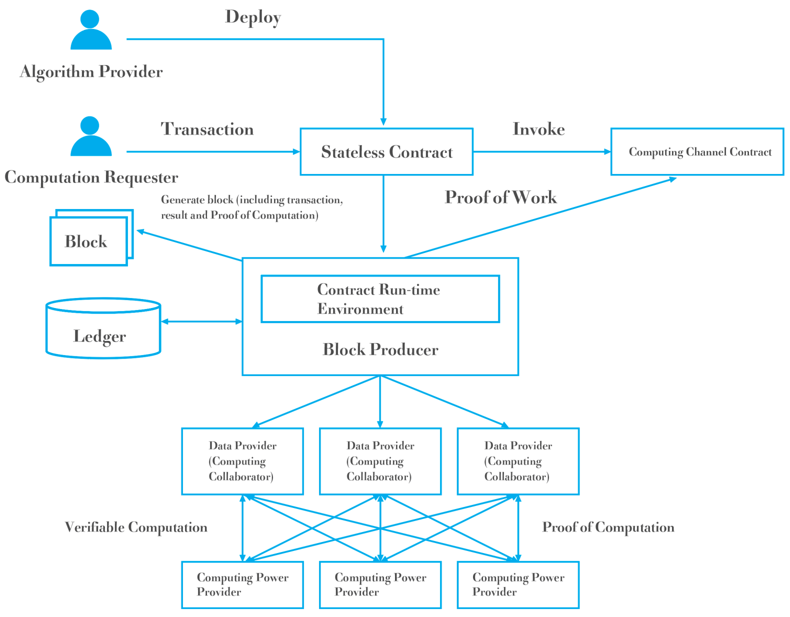
Deploy
Algorithm Provider
Transaction
Computation Requester
Stateless Contract
Invoke
Computing Channel Contract
Proof of Work
Generate block (including transaction, result and Proof of Computation)
Block
Contract Run-time Environment
Block Producer
Ledger
Data Provider (Computing Collaborator)
Data Provider (Computing Collaborator)
Data Provider (Computing Collaborator)
Verifiable Computation
Proof of Computation
Computing Power Provider
Computing Power Provider
Computing Power Provider

Figure 4.8: Stateless Contract

4.5.1.3 Hybrid Contract

PlatON also allows smart contracts to both store their state on the chain and involve off-chain data in the computation. This type of contract is known as a hybrid contract.

Hybrid contracts are essentially a form of multi-source data computation where the block producer participates in the computation as the provider of on-chain state data. A classic application for the hybrid contract is to write the results of off-chain data to the chain for the use in the next computation.

4.5.2 Virtual Machine of Meta Smart Contract

PlatON compiles meta smart contracts into circuits so that computation performance can be accelerated in conjunction with parallel computing and computing specific hardware. Developers can use popular programming language such as java to develop meta smart contract. A specialized compiler will be provided to compile meta smart contracts into circuits.

In addition to providing computing specific hardware that can execute meta smart contracts directly, PlatON also provides meta smart contract virtual machines that can carry out meta smart contracts under different software and hardware environments or for integration into other blockchain projects.

Chapter 5

Energon

PlatON consists of the master chain and multiple application chains. Each chain operates independently and are logically parallel to each other. The master chain is the initial chain of the PlatON network. Application chains are vertical chains that spring from the master chain to solve industry-specific problems.

5.1 On-chain Digital Energon

PlatON is a service-oriented computing architecture. In addition to providing computing, data, storage, network and other basic services, application developers can also publish their application services in PlatON. Each application running on PlatON needs to consume a certain amount of resources (including computing power, bandwidth, storage, data, etc.).To achieve the fair and reasonable use of resources and avoid the abuse of resources, PlatON realizes the reasonable dispatch and validation of resources through a series of algorithms and uses Energon to measure the use of resources. Energon is also the energy that drives PlatON to compute.

ATP is the Energon of the PlatON master chain. Each application chain can also create its Energons. In the PlatON network, users need only a single unified account to manage and use their Energons. Energons of different chains can be transferred freely across the chain.

For basic questions about the definition, value, issuance, recycling, and cost of Energon in PlatON, please refer to the PlatON economic and ecological red book, which will be further elaborated.

5.2 Cross-Chain Transfer of Energon

The cross-chain transfer of Energon across the PlatON network is carried out through a special Contract called the Energon Exchange Contract (EEC), which is built into the master chain and each application chain. The EEC smart contract uses "two-way peg" and "atomic swap" technologies to achieve Energon cross-chain transfer.

5.3 Decentralized Exchange

In the future, with the continuous application and development of PlatON, the need for conversion and exchange between PlatON's internal Energon, Energon and other digital assets (such as USDT, BTC, ETH, etc.) will also increase. Central exchanges provide efficient trading performance, but they also face huge risks of internal fraud and external hacker attacks, which bring asset security problems to users. PlatON built-in a decentralized exchange, employ the "order on-chain, match off-chain, settle on-chain" way, not only to ensure the safety of users assets, but also guarantee the performance of deal making and user experience.

- Internal cross-chain Energon conversion
  PlatON implements internal cross-chain energon exchange by establishing Energon reserve pool. Energon reserve pool adjusts conversion rate dynamically according to market conditions, the user can convert Energon of a chain to Energon of another chain through exchange with the reserve pool. The reserve pool uses [**two-way peg**] to ensure the security of cross-chain conversion.

  PlatON can enhance Energon liquidity, monitor market changes, and quell market speculation through the Energon reserve pool.

- External digital asset trading
  PlatON supports the creation of a reserve pool anchored to external digital assets (such as USDT, BTC, ETH, etc.) by mortgaging ATP on the master chain. The external digital asset reserve pool

dynamically adjusts the exchange rate according to market conditions. PlatON can trade ATP with external digital assets such as USDT, BTC ,and ETH through the external digital asset reserve pool.

Chapter 6

Technology Roadmap

The technology roadmap for PlatON is shown in Figure 6.1

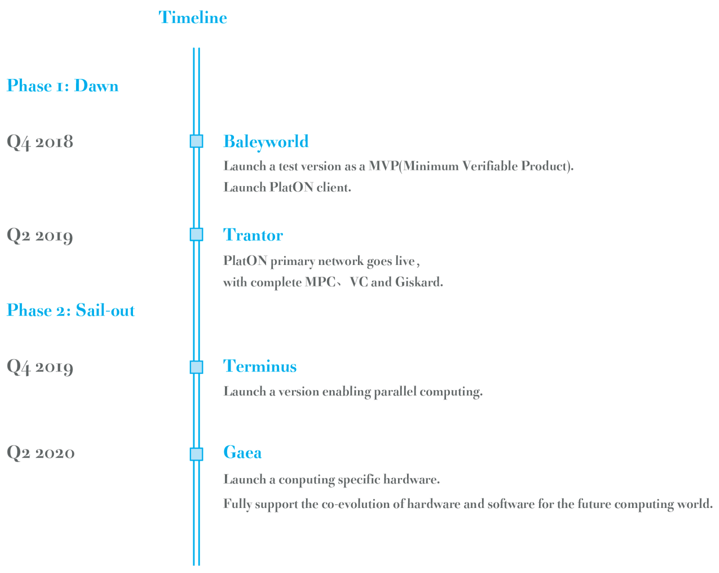
Timeline
Phase 1: Dawn
Q4 2018
Baleyworld
Launch a test version as a MVP(Minimum Verifiable Product).
Launch PlatON client.
Q2 2019
Trantor
PlatON primary network goes live,
with complete MPC、VC and Giskard.
Phase 2: Sail-out
Q4 2019
Terminus
Launch a version enabling parallel computing.
Q2 2020
Gaea
Launch a conputing specific hardware.
Fully support the co-evolution of hardware and software for the future computing world.

Figure 6.1: PlatON technology roadmap

- **Baleyworld:** Implement a complete RELOAD overlay network and blockchain services with support for service discovery, meta smart contracts and VC.

- **Trantor:** Support a complete MPC, updated VC and Giskard consensus.
- **Terminus:** Enable off-chain parallel computing.
- **Gaea:** Feature the version of software and hardware integration. Release computing specific hardware.

Chapter 7

Community Evolution

7.1 Technology Evolution

PlatON network is a system still in its early infancy which is not comprehensive enough. What it provides now is the outline of PlatON working as the next-generation computing architecture. Considering the great technical challenge of the complex network PlatON built, it must be noted that there still remains some problems with distributed architecture, cryptographic algorithms, mechanism design of game theory, hardware implementation and network building, some of which are even common intellectual challenges of all mankind, waiting for further breakthroughs in academia as well as progressive exploration in engineering. With the power of the global community, we will be dealing with these problems one by one and evolving step by step. Thus, improvement and iterations would be continued to be updated in our roadmap for the future.

7.2 Organizational Evolution

The evolution of the organization could be considered as the evolution of the governance model to a certain degree. We believe in the power of improvement rather than putting our faith in revolution. Human society experiments and makes decisions through different organizational structures and governance models repeatedly to achieve progress.

The open source blockchain community was struck by many problems such as the splitting and forking of the community, hacker attack, contract loophole, and regulatory challenges in the past few years. Nevertheless, the community retains its vitality.

As a globalized self-organizing community, the governance model design and practice of PlatON will run into similar challenges. However, PlatON will keep dealing with all governance challenges with the help of joint governance, sharing, and consensus.

PlatON will be progressively involving more and more participants such as developer communities providing for smart contracts, academic communities providing algorithms and theory, computing communities providing computing power, data communities providing data and requirement requestors. As the interests among them will be definitely diverse, the conflicts generated cannot be resolved through technology or algorithms for certain. PlatON will collect the feedback from each community to gradually refine and put forward community governance proposals for the future.

As a complex network, PlatON, instead of favoring any party on purpose, will encourage and support even more parties, institutions, stakeholders, and individuals to participate in this network as always. The more complex it is, the more prosper and stronger the network grows.

PlatON, in the age where data sovereignty becomes increasingly significant, has faith in the idea of “The part is unaware of the whole, while the whole cannot know the part it contains.” , which is not only the practice of privacy protection with cryptography but the value of the consensus mechanism as well.

The logical foundation driving the organizational evolution of the PlatON derives from consensus and “The Calculus of Consent”.

7.3 Network Evolution

On the basis of network-wide consensus, PlatON depends on the sharing of global computing power, data and algorithms, built on the basis of joint governance in the community among which network infrastructure plays a decisive role.

Starting from the Internet nowadays, PlatON will keep pace with the building and operation the global mobile Internet and space-based Internet to support, adapt to, or even launch different types of network and computing infrastructure whose ultimate goal is building a space-air-ground-based network infrastructure for driving the development of PlatON.

The network evolution of PlatON will greatly expand the community's horizons and continue to boost community consensus. The following work plan, updated regularly, will be published in the white paper on website building at the right time.

Always down-to-earth, we are sometimes looking at the stars.

Chapter 8

Glossary

Term	Acronyms	Definition
Algorithm Provider	-	The algorithm provider specifically refers to the publisher of the meta-smart contract in PlatON network. The algorithm is implemented in a meta smart contract that defines computation functions and input and output parameters.
Application Specific Integrated Circuit	ASIC	Application Specific Integrated Circuit is the integrated circuit designed and manufactured specifically for the needs of a complete machine or system.
Application Specific Standard Parts	ASSP	Application Specific Standard Parts is the integrated circuit designed for use in special applications.

Atomic Swaps	-	An atomic swap is a proposed feature in cryptocurrencies, that allows for the exchange of one cryptocurrency for another cryptocurrency without the need for a trusted third party. PlatON uses atomic swap to achieve Energon cross-chain transfer.
ATP	-	ATP is Energon released in PlatON master chain. It can be exchanged with the Energon in PlatON appchains and external digital assets. The abbreviation ATP can be further expressed as ATP adenosine triphosphate, which is a combination of adenine, ribose and three phosphate groups, and is the most direct source of energy in the organism. ATP is also known as the ”energy currency” of the living body.
Block Producer	-	Block Producer is responsible for executing the transaction and packaging the transaction data into blocks.In the Giskard consensus agreement, the consensus node is based on the calculated contribution weighted equity elections and consensus is reached through the asynchronous BFT protocol.

Byzantine Fault Tolerance	BFT	Byzantine refers to the Byzantine Generals' Problem described by Leslie Lamport, Robert Shostak and Marshall Pease in 1982. Byzantine Fault Tolerance is the ability of a distributed network to function as desired and correctly reach a sufficient consensus despite malicious nodes of the system failing or propagating incorrect information to other peers. Several BFT protocols were introduced to improve its robustness and performance, Practical Byzantine Fault Tolerance (PBFT) is one of these optimizations and was introduced by Miguel Castro and Barbara Liskov in 1999.
Circuit	-	A common form of computation expression made up of different gates. Known as a Boolean Circuit if made up of logic gates, or an Arithmetic Circuit if made up of arithmetic gates.
Computing Requester	-	The computing requester is an external client that initiates a call to a meta smart contract.
Computing Channels	-	Computing channel is a system smart contract, or regarded as computation state machine, in charge of tracking the state of the computation tasks. It serves as a deterministic program to assure termination and settle of computation and issues reward or punishment according to the correctness of results.

Computing Node	-	A node that provides computing services for the Platon network and is responsible for performing various computing tasks.
Computation Task	-	In PlatON, the computation function is compiled into Boolean Circuits and broken down into subcircuits for parallel computation. Each sub-circuit and its inputs are packed into a computation task.
Computing Collaborator	-	The computing coordinator is responsible for obtaining the data and distributing the data and algorithm to the computing provider for computing.
Computing Power Provider	-	The computing power provider accepts and executes computation tasks (including algorithm and data). Also known as the computing node in the PlatON network.
Data Node	-	A node that provides data services for the Platon network and is responsible for providing data for various computing tasks.
Data Provider	-	The data provider provides the corresponding data for computing based on the input data format defined by the algorithm.
Decentralized Application	Dapp	Dapp is a distributed application running on blockchain. Dapp is consisting of smart contract and front-end.

Decentralized Structured Topology	-	Based on how the nodes are linked to each other within the overlay network, and how resources are indexed and located, we can classify networks as unstructured or structured . The most common type of structured P2P networks implement a distributed hash table (DHT),in which peers can search for resources on the network using a hash table.
Delegated Proof-of-Stake Consensus	DPOS	DPOS is a consensus mechanism that requires coin holders to vote for ”delegates”, who are then responsible for validating transactions and maintaining the blockchain. DPOS is an alternative to Proof-of-Stake (PoS) model.
Energon	-	Energon is a measure of the use of Platon resources and is the energy that drives PlatON's ”computing factory”. Each application chain on PlatON can create its own Energon independently.
Field-Programmable Gate Array	FPGA	FPGA is an integrated circuit designed to be configured by a customer or a designer after manufacturing - hence “field-programmable”.
Full Node	-	A full node downloads every block and transaction and check them against consensus rules.

Functions as a Service	FaaS	FaaS is a “serverless” architecture that allows customers to develop, run, and manage application functionalities without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app.In a FaaS system, the functions are expected to start within milliseconds in order to allow handling of individual requests, and charge per execution time of the function.
(Fully) Homomorphic Encryption	(F)HE	Computation of the ciphertext protects privacy while providing operability. Fully homomorphic means all operations are supported during computation.
Garbled Circuit	GC	A method for encrypting circuits first proposed by Andrew C.C. Yao. It is one of the tools most commonly used in MPC.
Interactive Connectivity Establishment	ICE	ICE is a NAT traversal protocol for use in offer/answer mode. It is mainly used for the establishment of multimedia sessions under UDP. It uses STUN and TURN protocol, and can be used by other programs that implement the offer/answer model, such as SIP.

Kademlia	-	Kademlia is a P2P overlay network transport protocol designed by Petar Maymounkov and David Mazières to build a distributed P2P network. It is a P2P information system based on XOR operation. It establishes the structure of the network and regulates the way in which nodes communicate and exchange information.
Light Node	-	A light Node does not download the complete blockchain. Instead, it downloads the block headers only to validate the authenticity of the transactions. Lightweight nodes use a method called Simplified Payment Verification (SPV) to verify transactions.
Malicious model	-	A security model where the attacker does not follow the rules during execution. They actively tamper with the protocol and probe honest participants in hopes of obtaining more information.

Meta Computing Framework	-	The meta computing framework effectively integrates heterogeneous algorithm resources, data resources, and computing resources on a global scale, thereby promoting data and power transactions profoundly and extensively.While the meta computing framework uses parallel computing and computing specific hardware to improve computing performance, it also integrates multiple cryptographic algorithms to ensure verifiable and data privacy for computing.
Meta Smart Contract	-	Meta smart contract is smart contract in PlatON, it can access to onchain and offchain data.
Oblivious Transfer	OT	A type of secure transmission protocol that allows the two parties to safely select tags based on the input bits. It is one of the tools commonly used in MPC.
Proof-of-Stake Consensus	PoS	PoS is a type of algorithm by which a cryptocurrency blockchain network aims to achieve distributed consensus. In PoS-based cryptocurrencies, the creator of the next block is chosen via various combinations of coin age-based selection.

Proof-of-Verifiable-Computation Consensus	Giskard	As PlatON's consensus mechanism, Giskard elects block producers based on the value of computing contribution, and block producers use an asynchronous BFT algorithm to generate blocks and form a consensus. Giskard solves the problem of wasted power.
Proof-of-Work Consensus	PoW	Proof of work is the first distributed consensus mechanism, pioneered by bitcoin's pseudonymous creator, Satoshi Nakamoto. In PoW, the miners compete to find a hash with specific properties. The miner that finds the answer first is allowed to add a new block of transactions to the blockchain and is rewarded with a tranche of newly-minted bitcoins.
Recursive Distributed Rendezvous	ReDiR	ReDiR defines a service discovery mechanism based on RELOAD and was standardized as [RFC7374].
REsource LOcation And Discovery	RELOAD	The RELOAD protocol is a P2P network protocol framework proposed by the IETF P2PSIP (Peer-to-Peer Session Initiation Protocol) working group and was standardized as [RFC6940].The RELOAD protocol defines a unified overlay network peer and client protocol for abstract storage and message routing services.
Routing Node	-	The routing node provides the STUN and TURN services.

Secure Multi-Party Computation	MPC	Participate in multi-party computation where there is no trusted third party to obtain the computation results without revealing their respective inputs.
Session Initiation Protocol	SIP	SIP was standardized as [RFC2543] in 1999.The SIP is a signalling protocol used for initiating, maintaining, and terminating real-time sessions that include voice, video and messaging applications.
Session Traversal Utilities for NAT	STUN	STUN is a light-weight communications protocol to detect and traverse network address translators that are located in the path between two endpoints of communication.
System-on-a-Chip	SoC	A SoC is an integrated circuit that integrates all components of a computer or other electronic systems.
Traversal Using Relay NAT	TURN	TURN is an extension of STUN/[RFC5389], which defines a relay protocol that enables hosts behind Symmetric NAT to use the relay service to transmit packets to the peer.
Two-way Peg	-	Two-way Peg is a sidechain technology proposed by Adam Back et al in 2014. In PlatON, Two-way Peg is used to achieve Energon's secure cross-chain transfer.
value of computing contribution	-	The value of computing contribution is the validated effective power provided by the user.

Verifiable Computation	VC	Can effectively verify whether the returned data was derived using the logical computation specified in the original data.
Verifiable Random Function	VRF	the concept of a Verifiable Random Function (VRF) was introduced by Micali, Rabin, and Vadhan. It is a pseudo-random function that provides publicly verifiable proofs of its outputs' correctness.
Zero-Knowledge Proof	ZKP	The Prover convinces the Verifier that a certain fact is true without revealing any other information (zero knowledge).

Chapter 9

References

[**Trustless Computing Schemes**]
[**The Verifier's Dilemma**] L. Luu, J. Teutsch, R. Kulkarni, P. Saxena. "Demystifying Incentives in the Consensus Computer". CCS, 2015

[**The Scalability Trilemma**] https://github.com/ethereum/wiki/wiki/Sharding-FAQs this-sounds-like-theres-some-kind-of-scalability-trilemma-at-play-what-is-this-trilemma-and-can-we-break-through-it

[**Foreshadow**] https://foreshadowattack.eu/

[**PlatON Network Protocol**]
[**RFC6940**] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", RFC 6940, January 2014,
http://www.rfc-editor.org/info/rfc6940.

[**RFC5245**] J. Rosenberg , "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010,
http://www.rfc-editor.org/info/rfc5254.

[**RFC7374**] J. Maenpaa, and G. Camarillo, Ericsson, "Service Discovery Usage for REsource LOcation And Discovery (RELOAD)", RFC 7374, October 2014, http://www.rfc-editor.org/info/rfc7374.

[**RFC5766**] R. Mahy, P. Matthews, and J. Rosenberg, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)”, RFC 5766, April 2010,
http://www.rfc-editor.org/info/rfc5766.

[**RFC7890**] D. Bryan, P. Matthews, E. Shim, D. Willis, and S.Dawkins, “Concepts and Terminology for Peer-to-Peer SIP (P2PSIP)”, RFC 7890, June 2016,
http://www.rfc-editor.org/info/rfc7890.

[**Kademlia**] P. Petar, Maymounkov, David Mazieres. Kademlia: A peer-to -peer information system based on the XOR metric[DB/OL]. www.cs.rice.edu/conferences IPTPS02/109.pdf.

[**Energon**]
[**Atomic Swaps**] T. Nolan, Re: Alt chains and atomic transfers, https://bitcointalk.org/index.php topic=193281.msg2224949 msg2224949, 2013.

[**Two-way Peg**] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, Enabling Blockchain Innovations with Pegged Sidechains,
<https://blockstream.com/sidechains.pdf>, 2014.

[**Homomorphic Encryption**]
[1] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, 1978.
[2] C. Gentry. Fully Homomorphic Encryption Using Ideal Lattices. STOC, 2009.

[**Zero-Knowledge Proof**]
[1] S. Goldwasser, S. Micali, C. Rackoff , “The knowledge complexity of interactive proof systems” , SIAM Journal on Computing, 1989.

[2] B. Manuel, F. Paul, M. Silvio “Non-Interactive Zero-Knowledge and Its Applications”. STOC, 1988.

[**Multi-Party Computation**]

[1] A. C. Yao, Protocols for Secure Computations (Extended Abstract). FOCS, 1982.

[2] A. C. Yao, How to Generate and Exchange Secrets (Extended Abstract). FOCS, 1986.

[3] O. Goldreich, S. Micali, A. Wigderson:How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. STOC, 1987.

[**Verifiable Computation**]

[1] S. Micali, “Computationally Sound Proofs”. SIAM Journal on Computing, 2000.

[2] B. László, F. Lance, L. A. Leonid, S. Mario, “Checking Computations in Polylogarithmic Time”. STOC, 1991.

[3] S. Goldwasser, Y. T. Kalai, G. N. Rothblum. “Delegating Computation: Interactive Proofs for Muggles”.STOC, 2008.

[4] G. Rosario, G. Craig, P. Bryan. “Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers”. CRYPTO, 2010