The IRIS Network

Inter-chain service infrastructure and protocol for building trustworthy and distributed business applications

Haifeng Xi haifeng@bianjie.ai

Harriet Cao harriet@bianjie.ai

NOTE: If you can read this on GitHub, then we're still actively developing this document. Please check regularly for updates!

Table of Contents

Disclaimer

This whitepaper and any other documents published in association with this whitepaper relate to the intended development and use of the IRIS network. They are information purposes only and may be subject to change.

This whitepaper describes a developing project

This whitepaper contains forward-looking statements that are based on the beliefs of IRIS Foundation Limited, as well as certain assumptions made by and information available to IRIS Foundation Limited.

The IRIS network as envisaged in this whitepaper is under development and is being constantly updated, including but not limited to key governance and technical features. The IRIS token involves and relates to the development and use of experimental platforms (software) and technologies that may not come to fruition or achieve the objectives specified in this whitepaper.

If and when the IRIS network is completed, it may differ significantly from the network set out in this whitepaper. No representation or warranty is given as to the achievement or reasonableness of any plans, future projections or prospects and nothing in this document is or should be relied upon as a promise or representation as to the future.

No offer of regulated products

The IRIS tokens are not intended to represent a security or any other regulated product in any jurisdiction.

This document does not constitute an offer or solicitation of securities or any other regulated product, nor a promotion, invitation or solicitation for investment purposes. The terms of the purchase are not intended to be a financial service offering document or a prospectus of any sort.

The IRIS tokens do not represent equity, shares, units, royalties or rights to capital, profit, returns or income in the platform or software or in IRIS Foundation Limited or any other company or intellectual property associated with the platform or any other public or private enterprise, corporation, foundation or other entity in any jurisdiction.

This whitepaper is not advice

This whitepaper does not constitute advice to purchase any IRIS tokens. It must not be relied upon in connection with any contract or purchasing decision.

Risk warning

The purchase of IRIS tokens and participation in the IRIS network carries with it significant risks.

Prior to purchasing IRIS tokens, you should carefully assess and take into account the risks, including those listed on https://www.irisnet.org/ and in any other documentation.

Views of IRIS Foundation Limited only

The views and opinions expressed in this whitepaper are those of IRIS Foundation Limited and do not necessarily reflect the official policy or position of any government, quasi-government, authority or public body (including but not limited to any regulatory body of any jurisdiction) in any jurisdiction.

Information contained in this whitepaper is based on sources considered reliable by IRIS Foundation Limited but there is no assurance as to their accuracy or completeness.

English is the authorised language of this whitepaper

This whitepaper and related materials are issued in English only. Any translation is for reference purposes only and is not certified by IRIS Foundation Limited or any other person. No assurance can be made as to the accuracy and completeness of any translations. If there is any inconsistency between a translation and the English version of this whitepaper, the English version prevails.

No third party affiliation or endorsements

References in this whitepaper to specific companies and platforms are for illustrative purposes only. The use of any company and/or platform names and trademarks does not imply any affiliation with, or endorsement by, any of those parties.

You must obtain all necessary professional advice

You must consult a lawyer, accountant and/or tax professional, as well as any other professional advisors, as necessary prior to determining whether to purchase IRIS tokens or otherwise participate in the IRIS network.

IRIS OVERVIEW

The IRIS network is named after the Greek goddess Iris, said to be the personification of the rainbow and the faithful messenger between heaven and humanity.

Contractual relationships are a fundamental building block of human society and the importance of blockchain technology lies in providing a very efficient and cost effective way of realizing reliable contractual relationships: for the first time, trust (which is also very costly to establish) is not needed when multiple parties participate in sophisticated business interactions. It has been said that blockchain technology provides the most important elements for distributed business to take place: lifting network effect and very low transaction cost. More and more people see the potential of blockchains as the new internet of value and will gradually transform the current business models into more efficient distributed ones. Especially the token mechanism embedded in most modern blockchain emphasizes each network participant's right and will disrupt business in its current form [1].

However, blockchain technology is still in its early stages. As with any new technology, there are drawbacks. These include limited performance and undeveloped governance mechanisms. Presently, these drawbacks make it difficult for blockchains to support real-world distributed business collaboration. Consortium chains, such as Hyperledger Fabric and R3 Corda, and organisations such as the Ethereum Enterprise Alliance, have tried to address those performance and governance issues to make blockchain technology more suitable for enterprises. However, today consortium chains are dominated by huge enterprise companies. Furthermore their close-form off on-chain governance model is very inefficient. Without a token economics model and the openness and the excitement in public chains, consortium chains may be viewed as lacking vitality.

We would like to enhance the current blockchain technology and make it possible to enable thousands and millions of Small Medium Businesses ("SMBs") and even individual freelance business service providers to provide their services and enjoy the rewards in an open network. To achieve this, we have identified the following challenges and consequent opportunities for technology innovations:

Not all computation could or should be implemented as on-chain computations such as smart contracts

The Turing-complete virtual machine provided by Ethereum [2] runs Smart Contracts gives people a lot of hope of developing decentralized applications. However Smart contracts can only handle deterministic logic (so every node can reach an identical state after processing every transaction and block) while huge amount of existing business logic that is not deterministic and might vary at different time and under different environmental parameters. Especially these days, business systems have placed an increasing amount of reliance on computer algorithms, including Natural Language Processing ("NLP"), machine learning, and operation research algorithms, for decision optimization. In those algorithms, very often we purposely add some randomness to make the decision not to get stuck at local optimal states while trying to find a better sub-optimal result.

On the other hand, some of the real world business logics are meant to be run once off-chain and shouldn't be implemented as smart contracts this type of replicated computing. Integration and collaboration of off-chain services and resources with a distributed ledger is key to further advance the adoption of blockchain technology for more real-world use scenarios.

How to reuse the existing blockchain resources, including both public chains and consortium chains

It is infeasible to use one public chain to address all use cases. Every day there are different chains going live which focus on one aspect of problem solving such as distributed storage, asset ownership or predict market etc. According to the coinmarketcap.com, there are more than 1000 cryptocurrencies currently active on various exchanges.

While building business applications involve handling storage and also different source of data feeds. Another motivation of our work involves how to support building distributed business applications by reusing some of the existing work like storage (IPFS, SIA, Storj.io etc.), data feed (Augur, Gnosis, Oraclize etc.) and IoT (IOTA etc.) provided by those dedicated blockchains and not reinventing the wheel.

Besides, there are many (near) real-time business transactions do need more close form consortium/permission/private chains to address performance, security and business governance requirements. Our vision of distributed business infrastructure needs to have the Interoperability of many heterogeneous chains including public/consortium/permission/private chains.

Inter-chain technology is a very nature answer to the requirement. However, till today, the existing Inter-chain technologies are mainly designed to provide interoperability among existing blockchains and focus on token value transfer. The question of how to consume the resource provided in different chains still remains unanswered.

Comparing the proposed inter-chain technologies like Cosmos [3] and Polkadot [4], we find out that Cosmos provides more mature base for interoperability and scalability. Especially, we found the design of "`many hubs and many zones`" and "`each zones are independent blockchains having independent governance models`" from Cosmos provides a very suitable architecture for modeling the real world complexity in a SOC way. To best reuse the existing framework, we present the IRIS Network, a decentralized inter-chain network composing hub and zones with implementing a layer of service infrastructure based on Cosmos/Tendermint [5], with enhanced usage of token .

Since the IRIS network is designed on top of Cosmos/Tendermint, we will first discuss Cosmos/Tendermint, summarize the features we inherit from Cosmos/Tendermint and summarize the innovations we have created.

Cosmos and Tendermint

Cosmos [3] intends to build the 'internet of blockchains'. It is a network of many independent blockchains, called "zones". Each zone is powered by classical Byzantine fault-tolerant ("BFT") consensus protocols like Tendermint.

Tendermint provides a high-performance, consistent, secure BFT consensus engine, where strict fork-accountability guarantees hold over the behavior of malicious actors. Tendermint is well suited for scaling heterogeneous blockchains including public blockchains such as Ethermint [6], which is a blazing fast Proof-of-Stake implementation of Ethereum, as well as performance critical permission/consortium chains. The successful stories on using Tendermint in the permission/consortium chain domain including Oracle [7], CITA [8] and Hyperledger Burrow [9].

Tendermint is used as the consensus protocol for building the first zone on the Cosmos Hub. Hub can connect to many different kinds of zones, and the communication is achieved via an inter-blockchain communication ("IBC") protocol, a kind of virtual UDP or TCP for blockchains. Tokens can be transferred from one zone to another securely through the Cosmos Hub, without the need for an exchange or a trusted third party between zones.

To develop robust interoperable blockchains and blockchain applications with Cosmos Hub, Cosmos SDK provides blockchain development 'starter-kit' of common blockchain modules while not enforcing user stories thus giving maximum flexibility for application customization.

IRIS Innovations

IRIS network aims to build technology foundation which facilitate construction of distributed business applications. It goes beyond today's blockchain systems which are mainly for digitalized assets.

The key challenges that we aim to address via the IRIS network are two-fold:

- Integration and collaboration of off-chain computing and resources on a distributed ledger;
- interoperability of the services across heterogeneous chains.

We address those challenges through incorporation of a service oriented infrastructure into Cosmos/Tendermint.

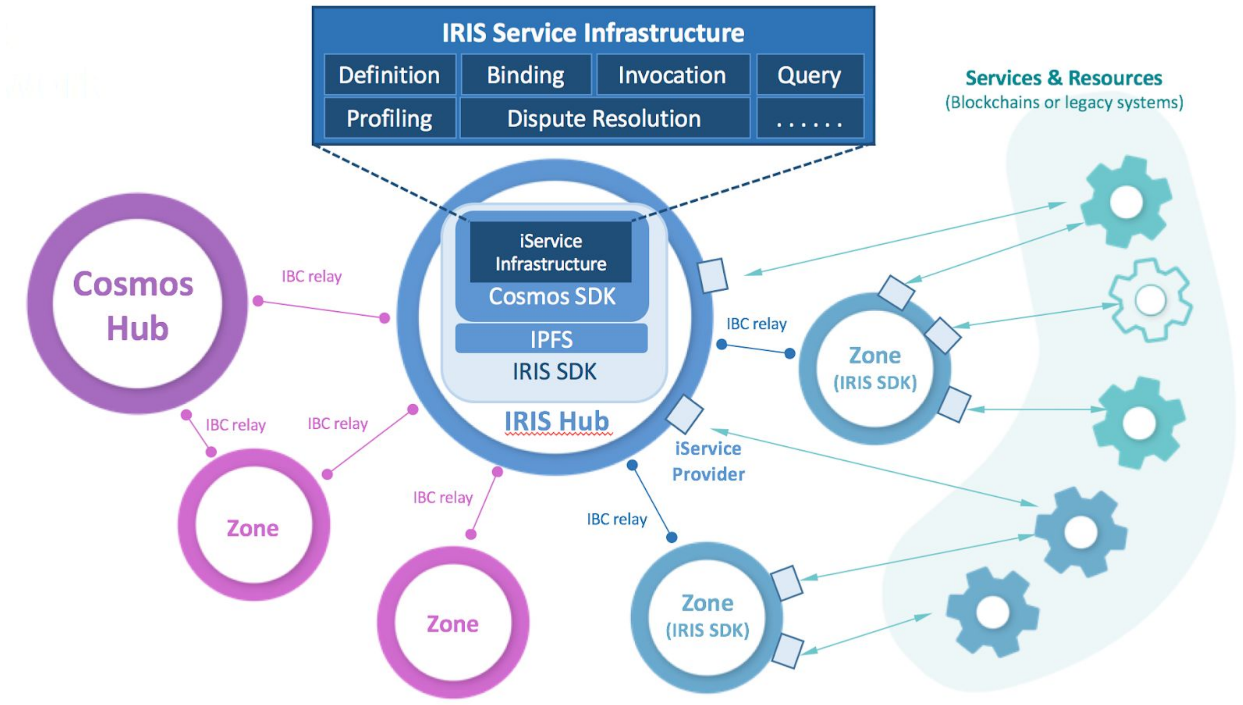
Our design inherits the thinking pattern from many years of service-oriented architecture ("SOA") practices. SOA is an architectural approach to create systems built from autonomous services which have explicit boundaries, share schemas and contracts [13]. Earlier practice of SOA focused on the implementation of Enterprise Service Bus ("ESB") which enables communication among services via a common communication bus which consists of a variety of point-to-point connections between providers and consumers. However, centralized management of services through ESB could trigger a single point of failure, also adds dependency of service deployment. The recent surge of micro-services architectural style can be seen as a development of SOA without focusing on the ESB rather using light message queues for inter service communication. In IRIS network, the inter service communication is intended to be implemented over blockchain to leverage blockchain as a trusted machine for mediating business collaborations. It runs without prerequisite of existing trust among service provider and service consumer which is very hard to establish.

The IRIS network uses Tendermint protocol as a high-performance consensus engine. Leveraging the flexibility provided by tendermint's Application BlockChain Interface ("ABCI"), we define a set of service infrastructure transaction types including service provisioning, service consumption and service governance. As explained earlier, most business logic is not suitable for implementation as deterministic smart contracts on blockchain, we are using this service layer to move the business application specific logics and transaction processing off the blockchain itself and use the blockchain only to get consensus on the results generated through those services. This idea is also inspired by existing work from blockchain community when address performance issues of moving some complicated computation off the main chain, such as Lightning Network's off-chain state channels [10] as well as Plasma's fraud proof side chains [11]. Although we are not implementing side chains, we rip traditional business logic computation off the blockchain and use it as a trustworthy mediation bus for complicated business collaboration.

For interchain communication, Cosmos IBC [12] defines a protocol for transferring values from an account on one chain to an account on another chain. The IRIS network designs new semantics to allow cross-chain computation to be invoked by leveraging IBC. We believe this capability is very important when building scalable business applications. Further details of potential use cases are set out below.

The IRIS network is intended to provide the service infrastructure for handing and coordinating on-chain transaction processing with off-chain data processing and business logic execution. Enhanced IBC capability allows those off-chain processing to be invoked cross chain, if required. The IRIS network, as presently envisaged, will also include client-side tools, including a smart wallet enabling cross-chain multi-asset storage, as well as consume and provide iServices. We plan to provide SDKs for easy construction of iServices. For example, for a specific service definition, the Client SDK would generate the provider side skeleton as well as consumer side stub for major programming languages.

IRIS Network Design



As illustrated in the figure above, the IRIS network is intended to have the same topology as the Cosmos network. We plan to connect the IRIS Hub to the Cosmos Hub as one of its zones and regional hubs. IRIS full nodes, developed with the IRIS SDK (which is itself a planned extension of the Cosmos SDK), are proposed to provide a service infrastructure as well as offer integration with an embedded InterPlanetary File System ("IPFS") node.

IRIS Services (a.k.a. "iServices") intend to bridge the gap between the blockchain world and the conventional business application world, by mediating a complete lifecycle of off-chain services -- from their definition, binding (provider registration), invocation, to their governance (profiling and dispute resolution). By enhancing the IBC processing logic to support service semantics, the IRIS SDK is intended to allow distributed business services to be available across the internet of blockchains.

While the IRIS network focuses on providing an innovative solution for distributed business applications, it is still part of the broader Cosmos network. All zones connected to our proposed IRIS hub would be able to interact with any other zone in the Cosmos network over the standard IBC protocol. Furthermore, by introducing a layer of service semantics, which we believe could enable a whole new set of business scenarios, the planned IRIS network would represent an increase in scale and diversity of the Cosmos network.

Network Actors

1. Consumers are those users who may consume off-chain services by sending requests to the network and receiving responses from the network.
2. Providers are those users who may offer the implementation of one or more iService definitions and often act as *adaptors* of off-chain services and resources located in other public and consortium chains, as well as in enterprise legacy systems. Providers monitor and process incoming requests and send responses back to the network. A provider could at the same time act as a consumer by sending requests to other providers. As planned, providers would be required to charge a fee for any services they might offer, and the service fee, by default, would be priced in the IRIS network's native fee token known as "IRIS"; providers could also price their services in other whitelisted Cosmos fee tokens, to be considered in due course.
3. Profiler is the special user who works on behalf of the IRIS Foundation Limited ("Foundation"), a Hong Kong incorporated company limited by guarantee. The Foundation will take the lead in building the IRIS network. The profiler is the sole user authorized to invoke iServices in the profiling mode, which is intended to help create and maintain objective *provider profiles* that consumers use to select suitable providers.

IRIS Services

In this section, we set out the intended technical parameters for deploying iServices on the IRIS network.

Service Definition

A Method is composed of:

- ID (int): The unique ID of this method in the encompassing iService
- Name (string): The unique name of this method in the iService
- Description (string): A description of this method
- Input (string): A structured definition of the input parameters
- Output (string): A structured definition of the output result
- Error (string): A structured definition of all possible error conditions
- OutputPrivacy (enum): Can be one of NoPrivacy or PubKeyEncryption

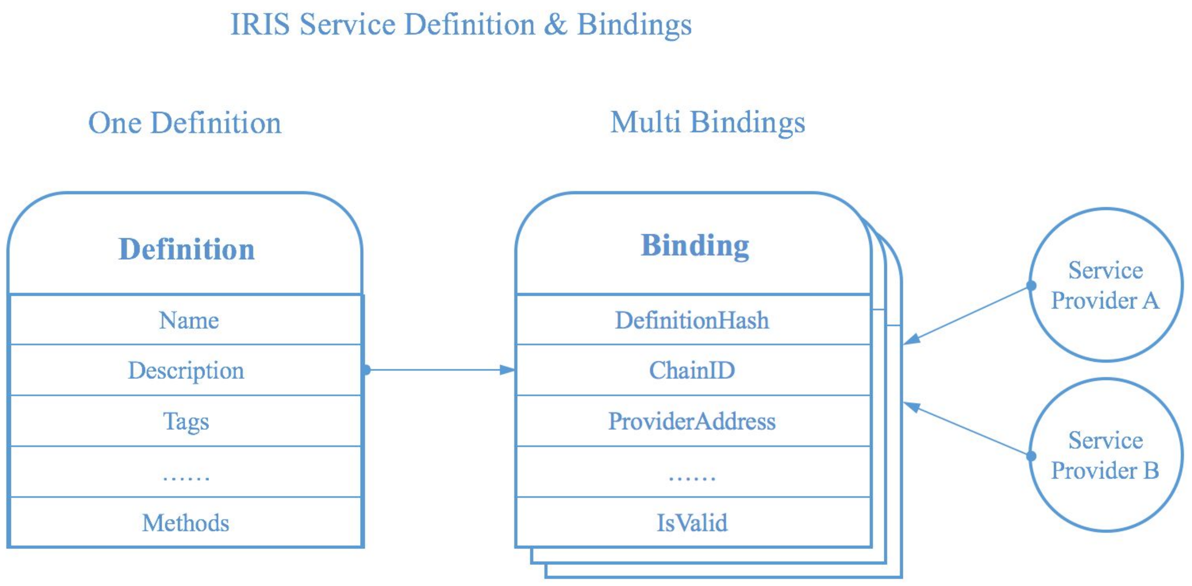
- A ServiceDefinition is composed of:
- Name (string): The name of this iService
- Description (string): A description of this iService
- Tags (string): Comma separated keywords about this iService
- Creator (string): A self-description of the iService creator. Optional
- ChainID (string): The ID of the blockchain where this iService was originally defined
- Messaging (enum): Can be one of Unicast or Multicast
- Methods ([]Method): The definition of methods available in this iService
- A CreateServiceDefinitionTx transaction is composed of:
- Definition (ServiceDefinition): The service definition to be created

Service Binding:

A CreateServiceBindingTx transaction is composed of:

- DefinitionHash ([]byte): The hash of the service definition that the provider is binding to
- ChainID (string): The ID of the blockchain where the provider is connected
- ProviderAddress ([]byte): The provider's blockchain address
- BindingType (enum): Can be one of Local or Global; choose Global if a provider wants the binding to be exposed to outside of its own chain as defined inBindingExposedChainIDs ([]enum) ; otherwise, use Local which means only the binding is only exposed within the local blockchain
- BindingExposedChainIDs ([]enum): The ID of the blockchains where the binding can be exposed. This parameter can be empty if the BindingType (enum) takes value as Local
- ProviderDeposit (int64): To create an effective binding, the provider must put down a deposit (in terms of IRIS token amount) that is greater than the value of the system parameter MinProviderDeposit; a larger deposit may imply more trustworthiness of the provider
- ServicePricing (string): A structured definition of the service pricing model on a per method basis, including cost per call, volume discount, promotional terms etc.; service fee is by default listed in IRIS token but could also be quoted in other whitelisted fee tokens.
- ServiceLevel (string): A structured definition of service level the provider agrees to bind himself to, in terms of response time, availability etc.

- BindingExpiration (int64): The blockchain height where this binding expires; a negative number means "never expire"
- An UpdateServiceBindingTx transaction is composed of:
- DefinitionHash ([]byte): The hash of the service definition the provider has bound to
- ChainID (string): The ID of the blockchain where the provider is connected
- ProviderAddress ([]byte): The provider's blockchain address
- ChangeSet (string): A structured definition of desired changes to an existing binding identified by the preceding three fields



A provider can update ServicePricing, ServiceLevel and BindingExpiration at any time, but a small amount of their deposit will be slashed for changing the latter two (specified by ServiceLevelUpdateSlash and BindingExpirationUpdateSlash respectively). Setting BindingExpiration to a height that is lower than the current height will be interpreted as invalidating the binding immediately.

Updates to ProviderDeposit will always be treated as adding to the current deposit balance. Whenever the balance drops below MinProviderDeposit, the binding will be disabled until the provider increases the balance above the threshold. Upon expiration or invalidation of a binding, the provider will automatically get back the remaining balance of its deposit.

BindingType can be changed from Local to Global, but not the other way around. To downgrade a binding from Global to Local, a provider must first invalidate the binding in question and then create a new Local binding.

If a provider somehow needs to move the binding to a new address, it is not allowed to update ProviderAddress directly; instead, the provider should invalidate the current binding and create another one with the desired new ProviderAddress.

Upon successful execution of these two transactions by the application (i.e., iService business logic in the IRIS SDK), a ServiceBinding object will be created or updated accordingly.

A ServiceBinding is composed of:

DefinitionHash ([]byte)

ChainID (string)

ProviderAddress ([]byte)

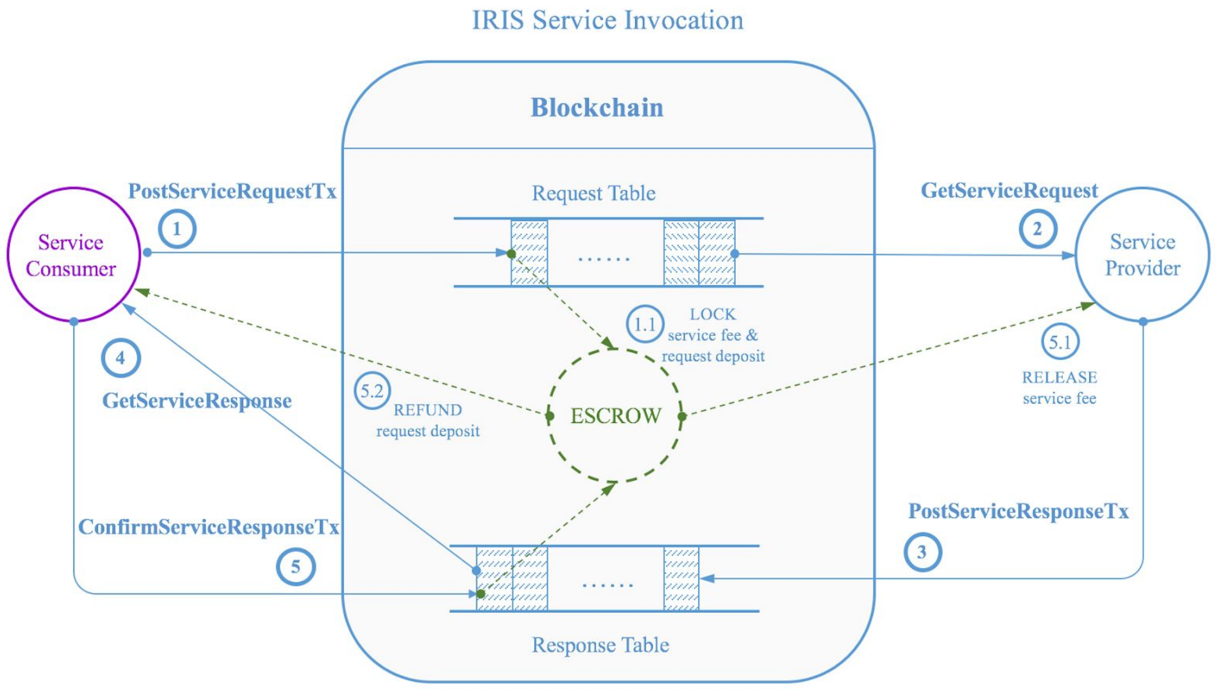
ServiceLevel (string)

ServicePricing (string)

BindingExpiration (int64)

IsValid (enum): Can be one of True or False

Service Invocation



Consumers and providers are proposed to interact with each other through endpoints. There are two kinds of endpoints -- request table and response table (see Figure above). Service requests are posted to request tables monitored by interested provider(s) which pick up and process requests addressed to them; service results (or errors) are posted back to response tables monitored in turn by matched consumers.

For a Multicast service, all of its bindings share one request table; for a Unicast service, however, a separate request table is created and maintained for each of its bindings. As for the other direction, a dedicated response table would be created and managed for each consumer.

A ServiceRequest is composed of:

- ChainID (string): The ID of the blockchain where the consumer is connected
- ConsumerAddress ([]byte): The blockchain address of the consumer
- DefinitionHash ([]byte): The hash of the service definition
- MethodID (int): The ID of the method to be invoked
- InputValue (string): A structured representation of input values
- BindingHash ([]byte): The hash of the target binding, in case of a Unicast service. Optional

- MaxServiceFee (int64): The max amount of service fee the consumer is willing to pay for a Multicast request. Optional
- Timeout (int): The max number of blocks the consumer is willing to wait for response(s) to come back
- A PostServiceRequestTx transaction is composed of:
- Requests ([]ServiceRequest): The service requests to be posted
- RequestDeposits ([]int64): The consumer must put down for each request a deposit (in terms of IRIS amount) that is greater than the value of MinRequestDeposit; this deposit is meant to incentivize the consumer to acknowledge receipt of service responses in a timely manner (see ConfirmServiceResponseTx).

The application will verify that each request is coming from a consumer with matching ChainID and ConsumerAddress, the targeted binding is valid, the request deposit is sufficient, the consumer's account balance is enough to cover the request deposits and service fees, and that the total number of requests in the transaction is less than MaxRequestPostBatch.

When a verified request is appended to the request table, the related service fee (MaxServiceFee in case of a Multicast request) will be deducted from the consumer's account and locked up in escrow.

A GetServiceRequest query is composed of:

- DefinitionHash ([]byte): The hash of the service definition
- BindingHash ([]byte): The hash of this provider's binding to the service in question; the application will verify that the binding is valid and the caller matches the binding's ChainID and ProviderAddress
- BeginHeight (uint64): The blockchain height from where the application should start to retrieve requests for the provider, up to a total number that is the lesser of BatchSize and MaxRequestGetBatch
- BatchSize (int): The max number of requests to be returned

A ServiceResponse is composed of:

- RequestHash ([]byte): The hash of the matched request
- BindingHash ([]byte): The hash of this provider's service binding
- OutputValue ([]byte): A structured (potentially encrypted) representation of output result. Optional
- ErrorMsg (string): A structured representation of error messages. Optional

A PostServiceResponseTx transaction is composed of:

- Responses ([]ServiceResponse): The service responses to be posted
- The application will verify that each response is coming from a provider with matching ChainID and ProviderAddress, and that the number of responses in the transaction is less than MaxResponsePostBatch. A verified request will be appended to the response table for the intended consumer.
- A GetServiceResponse query is composed of:
- RequestHash ([]byte): The hash of the original request; the application will verify that the caller matches the request's ChainID and ConsumerAddress
- BeginHeight (uint64): The blockchain height from where the application should start to retrieve responses for the consumer, up to a total number that is the lesser of BatchSize and MaxResponseGetBatch
- BatchSize (int): The max number of responses to be returned

A ConfirmServiceResponseTx transaction is composed of:

- ResponseHash ([][]byte): The hash of responses to be confirmed

The application will verify that the each response to be confirmed is indeed for a request originated by the caller, and that the number of responses in the transaction is less than MaxResponseConfirmBatch.

Responses that fall out of the Timeout window (and, in case of Multicast responses, when MaxServiceFee runs out as more responses come back) will not be accepted by the application. A consumer starts processing a Unicast response immediately upon receiving it. However, for Multicast responses, a consumer will need to wait until the Timeout window elapses before starting to process all responses received, if any.

When a Unicast response is confirmed by the consumer, the associated service fee will be released from escrow to the matched provider account -- after a small tax (defined by ServiceFeeTaxRate) is deducted and added to the system reserve; and the associated request deposit will be returned to the consumer as well.

In the case of a Multicast request, the situation is a bit more complex: when a response is confirmed, only part of the request deposit is returned to the consumer, in proportion to the response related service fee vs MaxServiceFee; and after all responses are confirmed, the remaining escrow balance for the request will be returned to the consumer.

If a request timeouts without seeing any response come back, the application will refund the associated balance held in escrow plus the request deposit, in full, back to the consumer. However, if the consumer does not confirm a response in time (before ResponseConfirmTimeout + blockchain height of the response), a small penalty (defined by ResponseConfirmDelayPenaltyRate) will be applied before the request deposit is refunded to the consumer, while the associated service fee will be released to the provider as usual.

Dispute Resolution

In any case where a consumer is unsatisfied with a service response, a mechanism should exist allowing the consumer to issue a complaint and consequently, to receive an acceptable solution to that complaint, without having to resort to a centralized authority such as the legal system. Also, this mechanism should avoid introducing subjective evaluation, which could be abused by either side.

The process to resolve a dispute that arises on the IRIS network resembles that of service invocation, except that a consumer sends a Complaint to the provider, and the provider responds with a Resolution. These interactions are intended to happen through a pair of global endpoints known as complaint table and resolution table.

Under the present design for the IRIS network, a consumer deposit is required for filing a complaint. Where a consumer does not confirm a resolution in a timely manner, a penalty will be deducted from this deposit. Similarly, a provider's deposit will be partially slashed if he fails to respond to a complaint in a timely manner.

A Complaint is composed of:

- ResponseHash ([]byte): The hash of the response in dispute
- Problem (string): A description of the problem with the service response
- PreferredDisposal (enum): Can be one of Refund or Redo

A Resolution is composed of:

- ComplaintHash ([]byte): The hash of the matched complaint
- Disposal (enum): Can be one of Refund or Redo
- Refund (uint64): Service fee refund. Optional
- OutputValue ([]byte): A structured (potentially encrypted) representation of output result. Optional

Our intended dispute resolution process, as outlined above, may not be legally binding. Nonetheless, we believe that it will provide an efficient means of resolving common disputes on the IRIS network.

Service Profiling

Bootstrapping the iService ecosystem presents a few challenges. A major challenge is finding a way to make it easy for consumers to discover suitable providers - consumers need performance metrics to evaluate and select a provider, yet without consumer usage no performance metrics will be available.

With the intention to solve this circular issue, we plan to introduce a profiling mechanism where a privileged system user, the profiler, invokes all the active services on a regular schedule. This would leave objective performance data in the network (such as response time, availability, complaint handling etc.) that are useful for real consumers.

Service profiling calls would be exempt from service fees and consumer deposits, but they would incur network transaction fees. These calls would originate from a few reserved addresses that are intended to be recognized and honored by the application.

Profiling activities would stay at a relatively stable level for new services and gradually decline for individual services as they start to attract real consumer calls, which is expected to generate more performance data on their own.

Transaction fees incurred during profiling would be paid out from the system reserve by default, and the Foundation reserve would step in if necessary.

Query

All the service related lifecycle objects described above can be queried using the ABCI Query interface [3]. These queries would be executed over the Query connection and do not participate in the consensus process. We have already seen how GetServiceRequest and GetServiceResponse queries work as part of the service invocation process.

Below is a non-exhaustive summary of our currently planned queries:

Object	Commonly Used Filters	Authorization
Service Definition	Name, keywords, source (chain ID), messaging type, with active bindings...	Anyone can query
Service Binding (for a given definition)	Location (local or remote), pricing, service level, expiration...	Anyone can query
Service Request	Service definition and binding, blockchain height, batch size	Only matched provider(s)
Service Response	Service request, blockchain height, batch size	Only matched consumer

Performance Metrics

Area	**Metrics**	**Authorization**
Provider (address)	Number of services provided (ever and active), response time (min, max and average), requests served (local and remote), requests missed, complaints received, complaints ignored, ...	Anyone can query
Provider (binding)	Active time, response time (min, max and average), requests served (local and remote), requests missed, complaints received, complaints ignored, ...	Anyone can query
Consumer	Number of services ever used, requests made, requests confirmed (in time and missed), complaints made, resolutions confirmed, ...	Anyone can query
Consumer (service, binding)	Requests made, requests confirmed (in time and missed), complaints made, resolutions confirmed, ...	Anyone can query

IBC Enhancement

One unique advantage of establishing our service infrastructure on top of Cosmos is the potential for services to be deployed once and invoked everywhere, over an internet of blockchains. Specifically, our plan is to accomplish the following:

1. Service definitions are broadcast to every zone in the IRIS network;
2. Global service bindings are broadcast to every zone in the IRIS network;
3. Service requests or complaints targeting a remote provider are routed to the blockchain to which the provider is connected;
4. Service responses or resolution meant for a remote consumer are routed back to the blockchain to which the consumer is connected.

When processing a CreateServiceDefinitionTx transaction, the application is designed to first validate and store the ServiceDefinition object locally, before creating an IBCPacket containing the definition for each neighboring chain.

Each neighbor eventually receives -- from the corresponding relay process -- an IBCPacketTx containing the packet; if the definition does not already exist in the receiving chain, the latter will pass on the definition by creating an IBCPacket for each of its neighbors -- except the source chain from which it received the packet in the first place; if the definition already exists, the receiving chain stops passing on the definition.

Similarly, when a ServiceBinding is created or updated with its BindingType set or updated to Global, an IBCPacket containing the binding is created for each neighboring chain, and gets propagated across the entire IRIS network.

An IBCPacket described above is composed of:

- Header (IBCPacketHeader): The packet header
- Payload (ServiceDefinition or ServiceBinding): The bytes of the service definition or binding

The IBCPacketHeader above is composed of:

- SrcChainID (string): The ID of the blockchain creating this packet
- DstChainID (string): The ID of the neighboring blockchain this packet is destined for
- Number (int): A unique number for all packets
- Status (enum): NoAck
- Type (string): "iris-service-definition" or "iris-service-binding"

Now let's take a look at how interchain service invocation happens through IBC. When a request is made for a Unicast service, the application checks if the target binding is Local; where this is true, the ServiceRequest is appended to the corresponding request table as explained in 2.2; otherwise, an IBCPacket containing the ServiceRequest will be created instead.

An IBCPacket containing a ServiceRequest is composed of:

- Header (IBCPacketHeader): The packet header
- Payload (ServiceRequest): The bytes of the service request

The IBCPacketHeader above is composed of:

- SrcChainID (string): The ID of the blockchain creating this packet
- DstChainID (string): The ID of the blockchain where the remote provider is located, i.e., ServiceRequest.ServiceBinding.ChainID
- Number (int): A unique number for all packets
- Status (enum): AckPending
- Type (string): "iris-service-request"
- MaxHeight (int): Current height + ServiceRequest.Timeout

As a remote request finally arrives at the destination chain, the application would append it to the corresponding endpoint (the request table) for the targeted binding. A response to this remote request would be wrapped in a receipt IBCPacket that is routed all the way back to the source chain and appended to the remote endpoint (the response table) for the originating consumer.

Request for a remote Multicast service is treated in the same way except that more than one IBCPacket could be generated in the source chain.

Remote complaints and resolutions are expected to work in the same manner as requests and responses, and therefore will not be elaborated here.

Below is a complete list of application-dependent IBCPacket types:

Type	iService Object
"iris-service-definition"	ServiceDefinition
"iris-service-binding"	ServiceBinding
"iris-service-request"	ServiceRequest
"iris-service-response"	ServiceResponse
"iris-complaint"	Complaint
"iris-resolution"	Resolution

Use Cases

In this section, we have set out some potential use cases for the IRIS network.

Distributed AI for privacy preserving data analysis

The proposed service infrastructure has been prototyped by Bianjie AI, a Shanghai based startup, into its permission product BEAN (Blockchain Edge Analytics Network) to solve the longstanding challenge of getting data for running analytics models. Although homomorphic encryption is one of the key methods allowing computing to be achieved over encrypted data, it is said to be unable to practically solve real world machine learning problems due to its slow performance. As a result, BEAN was created to take a different approach. This approach uses the idea of model parallelism taken from the traditional distributed AI study [14] and utilizing SOA design patterns to develop distributed analytics services as an additional layer to the blockchain.

To protect data access, the (partial) model that runs on the data side needs to be open sourced to the client and specified in the service definition. Since only the partial model is released to the client, the model developers do not have to worry about someone stealing their idea; equally, the data owners never need to worry about losing control of data usage as their data will not be leaving its origin.

Other potential benefits could include the following:

Only a small amount of parametric data being exchanged on-chain, which can help enhance performance.

A more practical way for data usage auditing, which is often needed in the healthcare domain.

Healthcare data is highly private, involving numerous security requirements. This puts forward the challenge for healthcare data to be used for the purposes of cross-organization collaboration (such as a cross-hospital clinic records search for diagnosis assistance, new drug clinic test patient identification, health insurance automatic claim processing etc.). This minimum viable product ("MVP") service layer implementation is built on top of Ethermint in attempt to connect hospitals, insurance companies and analytics service providers to provide privacy preserving healthcare data analytics capability.

Smart contracts have been implemented to support on-chain service registration and invocation. One example of the off-chain data processing could be to support a Diagnosis Related Group ("DRG") grouping analytics service. More specifically, when a hospital user invokes the DRG service, the raw medical record is processed off-chain using service provider provided client side NLP (implemented as SQL and Python) code stub to exact structured data inputs for receiving DRGs service over blockchain without passing the highly confidential raw medical records.

The BEAN scenario demonstrates a more complicated service use case including implementing distributed analytics, and connecting service providers as well as service consumers, utilizing blockchain to provide audible transaction ledge as well as trustworthy distributed computing foundation.

Data and analytics e-marketplace

From studying several proposed AI+Blockchain projects, it seems that most of the projects aim to provide data exchange markets and analytics API markets. With proposed IRIS infrastructure, those networks could potentially be built with ease through publishing data as data services and wrapping analytics API as analytics services utilizing the IRIS service provider SDK.

Distributed e-commerce

With the proposed IRIS infrastructure, integration with traditional systems like ERP to obtain inventory information, or inter-chain query on trusted data sources to obtain information such as transportation and weather data, will be quite similar to the approach with which many enterprise application developers are already familiar. With those services integrated to support distributed e-commerce applications, it could be possible for distributed e-commerce applications to provide a similar user experience as centralized systems, such as Amazon or Alibaba.

Combining public chains & consortium chains

For many business scenarios, taking a hybrid architecture of combining the good features of a public chain and a consortium chain can provide beneficial results, particularly with regards to performance, security and economic incentives.

For example, hospitals and insurance companies could form a consortium blockchain to support high performance medical insurance transactions, whilst identifying other information such as statistics regarding certain diseases as a global service, which can be invoked from other public chains. The tokens received from public chains can be awarded back to those information providers in the consortium chain, which motivate the system participants to improve and promote services. With this infrastructure provided by IRIS, large-scale spontaneous collaboration could be made possible while still supporting stringent performance and security requirements.

There are many use cases that could be supported by the IRIS service infrastructure, such as more efficient asset based security systems, distributed regulation technology such as due diligence, mutual aid marketplace etc. One of IRIS project plans is also working closely with such application project teams to support and enable them with needed blockchain infrastructure and allow them to focus on delivering the envisioned business value more efficiently.

Token Economics

Similar to the Cosmos Network, the IRIS network, as presently designed, is intended to support a multi-token model. The tokens will be held on the various zones, and can be moved from one zone to another via the IRIS Hub. There are two types of tokens that are expected to support IRIS network's operation:

- staking token
- fee token

Staking token

Adopting the same staking mechanism design used in the Cosmos network [15], the IRIS Hub will have its own special native token for staking. This token will be called "IRIS". We have a number of ideas in mind regarding the specific functionality of the IRIS token, including:

- integration of the IRIS token in the IRIS network's consensus engine validators, through a system of validators and delegators;
- voting power to participate in the IRIS network's governance

Fee token

There are two types of fee tokens in IRIS network:

- Network fee token is for spam-prevention and payment to validators in maintaining the ledger;
- Service fee token is used for payment to service providers who deploy iServices and the default payment service token is IRIS token.

The IRIS network is intended to support all whitelisted fee tokens from the Cosmos network, e.g Photon, plus the IRIS token.

Supporting a variety of whitelisted fee tokens is a feature that we plan to adopt from Cosmos. It can provide an enhanced experience for network participants. In Cosmos, for `network fee token`, each validator has a config file defines his personal weighting of how much they value each fee token. Validator can run a separate cron job to update the config file based on validator preferred live market data or maybe just use a default config value.

For the `service fee token` design, similarly a multi-token model is planned to be supported. A service provider on the IRIS network should therefore have the freedom to charge for their services in their preferred tokens, provided that it appears on the whitelist.

To help IRIS network participants mitigate cryptocurrency price volatility, the Foundation intends to facilitate the deployment of global iServices for retrieving market data from different exchanges, or through the potential introduction of oracles.

Both staking and fee tokens are subject to further refinement to ensure compliance with legal and regulatory obligations.

Initial Token Distribution

On Genesis, the initial token supply will be 2,000,000,000 IRIS tokens. The distribution of IRIS tokens is planned to be as follows:

- Private Sale: 25%
- Bianjie Developer Team: 15% (4-year vesting period starting from IRIS Hub launch, during which the team will vest 1/48th of its IRIS tokens each month)
- Tendermint Developer Team: 10% (2-year vesting period starting from IRIS Hub launch, during which the team will vest 1/24th of its IRIS tokens each month)
- IRIS Foundation: 15% (reserved to support the operations of the Foundation)
- Ecosystem Development: 30% (swap with zones connecting to IRIS Hub; grant to potential users; awards to outstanding partners)
- Cosmos Hub Airdrop: 5% (The goal for this airdrop is to support the long term successs of both Cosmos and IRIS hubs. One of the design thinking is to have this airdrop to ATOM holders arranged through a special airdrop to a wallet owned by the Cosmos Hub for ATOM holders and stake to IRISnet. This could enhance security for both Hubs also ATOM holders can enjoy the block rewards from another HUB)

If and when the IRIS network is fully deployed, the annual inflation rate of IRIS tokens will be adjusted to account for the fact that a substantial portion of IRIS tokens in circulation may be voluntarily staked by participants to participate in the consensus engine.

Proceeds from the private sale of IRIS tokens will be used, first and foremost, for the development of the IRIS network. The planned usage distribution is as follows:

- Foundation Operations: 10% (including service providers and contractors fees, for example, auditing, consulting, legal and other third party fees, and other overheads)
- Software Development: 50% (including costs, fees and expenses directly attributable to the development of launch)
- Developer Enablement: 10% (including funding hackathons, awards to volunteers and training programs)
- Research and Development Sponsorships: 10% (including conference, research programs and university outreach)
- Marketing and Promotion: 20% (including business development, community programs and outreach, together with related travel, communication, publication, distribution and other expenses)

Roadmap

The expected IRIS project is set out below. We reiterate that the roadmap is indicative only, and subject to change. The IRIS project has already accomplished the Pangu stage and is currently in Nuwa stage.

PANGU (January 2018 ~ March 2019) The first stage of the IRIS project focused on having the IRIS Hub up and running. Initial version of the mobile client for the IRIS network was released. In this stage we also focused on building the fundamental IRIS Service Layer which was released to IRIS Hub. The fundamental IRIS Service layer enables service definition, binding, invocation and query.

NUWA (April 2019 ~ October 2019) In this stage we are adding more fundational modules to support application development especially DeFi applications. The new modules planned to release in this stage including multi-asset management, Coinswap, multi-sig account etc. We plan to collabrate with 1-2 ecosystem parteners to develop applications using those modules. We also plan to accomplish the test connection with Cosmos Hub through IBC at this stage.

KUAFU (Nov 2019 ~ Sep 2020) In this stage we are aiming to accomplish the connection of IRIS Hub with applications blockchains through IBC. We plan to upgrade IRISnet mobile client to support those applications. The third stage will also focus on incremental upgrades to the IRIS network in order to support our planned advanced IRIS Service features.

HOUYI (Beyond Oct 2020) The fourth stage will focus on further technology innovations to the IRIS network, SDK and mobile client, as well as developer engagement.

The Team

Bianjie is the core development team for the IRIS network, leveraging the team's experience established from building distributed applications. Bianjie is a Shanghai-based start-up established in 2016. It focuses on developing innovative products and solutions for healthcare and financial industries, using advanced Blockchain and AI technologies. Besides IRISnet, Bianjie's also building another core product --- `BEAN (Blockchain Edge Analytics Network)` BEAN (Blockchain Edge Analytics Network), which is a permission chain which delivers distributed data analytics services for privacy preserving healthcare data analysis and exchange using NLP and machine learning technologies. Bianjie is also the operation and service partner of Cosmos Network in China.

Tendermint (the team that developed the Tendermint consensus engine and is currently building Cosmos), Wancloud (a subsidiary of Wanxiang Blockchain are strategic parteners working together with Bianjie AI building the IRIS network's infrastructure.

Tendermint's intended role to give technical advice and development support to the IRIS project team in extending the Tendermint ABCI and the Cosmos IBC technologies. Wancloud is envisaged as the key strategy partner to both the Cosmos and IRIS ecosystems, and we understand that it intends to participate in Cosmos and IRIS development across Asia.

Core Members

Haifeng Xi

Haifeng is the co-founder of IRISnet proejct. He is a senior technologist and entrepreneur. Haifeng has an M.S degree in ECE from the University of Maryland. Haifeng worked as CTO for Wanxiang Blockchain Wancloud before starting IRISnet project. He also worked as senior architect for two leading financial companies In US (Tudor Investment & RBS Sempra), then he came back to China worked in the capacity of CTO for three companies, one of which is NASDAQ listed (China Finance Online).

Harriet Cao

Harriet is the co-founder of IRISet project. She is also the co-founder of Bianjie, which the core development team for IRISnet. Bianjie was founded in 2016 focusing on building smart services for blockchain that enable trustworthy and efficient business collaborations. Harriet is an award-winning practitioner of distributed computing, data analytics and artificial intelligence technologies, and was the recipient of 2010 INFORMS Daniel H. Wagner Prize. Prior to establishing Bianjie, Harriet worked for IBM Research for more than 16 years in various capacities including Director of IBM Research Shanghai Lab and Big Data Analytics Leader for IBM Global Labs. Harriet has an M.S degree in Robotics from Carnegie Mellon University and an M.S. degree in Automation Control from Tsinghua University.

Jeffrey Hu

Jeffrey Hu is the research director of IRISnet and heads the strategy and technology research as well as ecosystem development for IRISnet. He is also the former chief technical analyst at Huobi Research. He has extensive experience in the blockchain industry. He published many in-depth Blockchain research reports which made great influence in China.

Jae Kwon

After graduating from Cornell in 2005 with an undergraduate degree in computer science, Jae worked as a software developer in Silicon Valley, first at Amazon (where he worked on the Alexa digital assistant), then later at Yelp, where he led their mobile app development team. After getting the blockchain bug, Jae created the Tendermint BFT consensus algorithm and the Tendermint Core blockchain engine, with the intent of creating a provably secure proof-of-stake algorithm. In addition to Tendermint, Jae is also the creator of Cosmos.

Tom Tao

Since joining Wanxiang in August 2016, Tom is responsible for Wanxiang Blockhain Group's consulting service, Wancloud BaaS Platform as well as the ChainBase accelerator and incubator service. Before Wanxiang, Tom worked in service management and business management for over 18 years in a number of global leading companies. Tom has spearheaded the introduction of cloud services, IoT data service platforms, and creative accelerator technologies into the Chinese market. Tom has been tracking trends in the blockchain, cloud computing, IoT and smart manufacturing industries since 2013. Tom has a Master's degree in Physics from Fudan University and a Bachelor's degree in Electrical Engineering from Nankai University.

Advisors

Dr. Shuo Bai

Dr. Bai is the director of ChinaLedger Technical Committee, and former Chief Architect of Shanghai Stock Exchange. He is a senior blockchain professional who graduated from Peking University with doctorate of science. He worked in various capacities including researcher, doctoral student advisor, director of software department, and chief scientist in the Institute of Computing Technology, Chinese Academy of Sciences. He also led the establishment of China National Internet Emergency Center (CNCERT/CC) since 2000. Dr. Bai has rich experiences in theoretical research and technical practices in the fields of financial exchanges, consortium and public blockchains.

Jim Yang

Jim Yang runs Strategy for Tendermint. He was the founder and CEO at ChatX, mobile messaging studio. ChatX developed various mobile messaging/social apps. He also co-founded Identyx, where he served as CEO until its acquisition by Red Hat. Identyx developed an open source enterprise identity management software.

Zaki Manian

Zaki Manian, Executive Director of Trusted IoT Alliance, is a prolific contributor to the development of blockchain and cryptocurrency technology. Zaki has deep expertise in cryptography and distributed consensus system. He is also an advisor to the Cosmos project, and several other investment funds and startup in the space.

Adrian Brink

Adrian Brink, Core Developer & Head of Community of Tendermint / Cosmos Network.

Michael Yuan

Dr. Michael Yuan is the Director of the CyberMiles Foundation. Michael received a PhD in Astrophysics from University of Texas at Austin. He is the author of 5 books on software development, published by Prentice Hall, Addison-Wesley, and O'Reilly. Michael was an active code committer in large Open Source projects such as Firefox, Fedora, JBoss, and others. He is an expert on enterprise and mobile software, and was a Principle Investigator on multiple research projects funded by the US government.

Yubo Ruan

Yubo is the founder of 8 Decimal Capital. The fund invested in IRISnet，0x、Kyber、Ontology、Fcoin、Zilliqa、ICON、Wanchian、Bibox、BiShiJie. Yubo is the co-founder of Skylight Investment, a boston based venture fund backed by New Oriental(NYSE:EDU). Previously, Yubo started two highly successful companies, including Alisimba (Acquired by TopHacker Group) held 4 national patents and won the 2017 AACYF 30 under 30, Silver Medal Winner, iENA International Inventions Competition, 2012.

References

- 1 Wanxiang Blochchain Inc., Distributed Business Value Research Institute, "Blockchain and Distributed Business Whitepaper", September 2017.
- 2 Ethereum Foundation, "Ethereum Homestead Documentation", http://ethdocs.org/en/latest/
- 3 Jae Kwon, Ethan Buchman，"Cosmos, A Network of Distributed Ledgers", https://cosmos.network/whitepaper
- 4 Gavin Wood, "Polkadot: Vision For a Heterogeneous Muilti-chain Framework", https://polkadot.io/
- 5 Tendermint, https://tendermint.com/docs/
- 6 Ethermint, https://ethermint.zone/
- 7 Oracle International Corporation, "Accountability and Trust in Distributed Ledger Systems", USA Patent Application 20170236120, August 17, 2017, http://www.freepatentsonline.com/y2017/0236120.html
- 8 Jan Xie, "CITA Technical Whitepaper", https://github.com/cryptape/cita-whitepaper/blob/master/en/technical-whitepaper.md
- 9 Hyperledger Burrow, https://github.com/hyperledger/burrow
- 10 Joseph Poon, Thaddeus Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments", January 14, 2016, https://lightning.network/lightning-network-paper.pdf
- 11 Joseph Poon, Vitalik Buterin, "Plasma: Scalable Autonomous Smart Contracts", August 11, 2017, https://www.plasma.io/plasma.pdf
- 12 Ethan Frey, "Cosmos IBC Specification", Sep. 29, 2017, https://github.com/cosmos/ibc/blob/master/README.md

- 13 Thomas Erl, "SOA: Principles of Service Design", Prentice Hall; 1st edition (July 28, 2007)
- 14 Dean, J., Corrado, G.S., Monga, R., et al, Ng, A. Y. "Large Scale Distributed Deep Networks". In Proceedings of the Neural Information Processing Systems (NIPS'12) (Lake Tahoe, Nevada, United States, December 3--6, 2012). Curran Associates, Inc, 57 Morehouse Lane, Red Hook, NY, 2013, 1223-1232.
- 15 Tendermint Blog, "Cosmos Validator Economics -- Revenue Streams", January 2018, https://medium.com/@tendermint/b5b2c682a292
- 16 Sunny Aggarwal, "Cosmos Token Model", December 2017, https://drive.google.com/file/d/1jtyYtx7t1xy9gxEi2T5lXFNd8xUY7bhJ/view