Building a Powerful Edge Using Blockchain

Zhao Zhenlong, Jing Shuaishuai
TrustChain Technology
• About Us
• How to build a Powerful Edge Using Blockchain
• Two-layered Intelligence Solution
• Cloud-Edge Infrastructure
• Blockchain Technologies in Edge
From Open Source to Open Ecosystem
Use CVT-Network Enhancing Edge

CVT Network = Cooperate + Value + Trust

- edge-trusted security
- circulation of value
- smart-self-decided

- Trusted Exec Environment
- Blockchain
- Smart Contract
Two-layered Intelligence BIoT

Blockchain provides Consensus-Trusted Info to AI

Core Intelligence: Mind & Decision

AI / BigData / Cloud

Trust from Consensus

Smart Contract

Edge Intelligence: Basic Rule

IoT provides Objective-Trusted Info to Blockchain

Trust from Objective
Two-layered Intelligence BloT

Core Intelligence
- Product Design
- Data Analysis
- Device Mgmt
- Industry Service

Security Channel

Smart Edge
- Cooperate Network
- Value Network
- Trust Network
- Security Channel

Base Chain
- TrustEdge
- TrustStorage
- TrustBase
- PoW Blockchain
- PoET Blockchain
- PBFT Blockchain
- Raft Blockchain

Product Design
Data Analysis
Device Mgmt
Industry Service
Cooperate Network
Value Network
Trust Network
Security Channel
TrustEdge
TrustStorage
TrustBase
PoW Blockchain
PoET Blockchain
PBFT Blockchain
Raft Blockchain

Cooperate
Network

Value
Network

Trust
Network

Security Channel

TrustEdge

TrustStorage

TrustBase

PoW
Blockchain

PoET
Blockchain

PBFT
Blockchain

Raft
Blockchain

Smart
Edge

Core
Intelligence

Two-layered Intelligence BloT

Trust Chain Technology

世纪诚链
BloT Platform

Input Source in TEE
- Camera
- Regular Gateway
- PLC/DDC/DCS
- Sensors
- Networking Devices

Edge Layer
- Smart Contract
  - Data Curation
  - Digital Asset Exchange
  - Local Computing
  - Local Decision

Data Collection

Base Chain - Smart Edge Control Plane
- Chain Info Registration
- Inter-Chain Service
- Chain Management

Upload Data
Download Smart Contract
Edge Infra from NFV to CVT-NFV

Cloud

Distributed Control Plane

Site Control 1
- Controllers
- Edge Controller
- Device Controller
- Cloud Hub

Site Control 2
- Controllers
- Edge Controller
- Device Controller
- Cloud Hub

Site Control N
- Controllers
- Edge Controller
- Device Controller
- Cloud Hub

Cloud Hub

Edge
- Edge Hub
- Device Twin
- Service Bus
- Event Bus
- Meta Manager
- Edged
- Data Store
- Containers

Devices

MQTT

Site 1

Site 2

Site N
IoT Blockchain Architecture

TrustEdge

TrustBase

TrustStorage

TrustStream

TrustPay

HW-Oracle

Database-Oracle

Storage-Oracle

Cloud-Oracle

Pay-Oracle

Identity

Base Chain

Local Chain

Local Chain

Local Chain

Hyperledger Sawtooth

Tendermint

TrustChain
TrustOps Developer Toolstack

- Using ansible and docker
- One-Key Deployment
- Environment Pre-Check and Post-Confirm
- Resource Allocation / Network Planning
- Support
  - TrustChain
  - Tendermint
  - Hyperledger Burrow
  - Hyperledger Sawtooth
  - Hyperledger Fabric (todo)
TrustChain Tech Arch

Solutions

Data Access API  Contract Mgmt  Wallet & Pay

Smart Contract Layer

Micro-Contract

Formal Modelling  Model Checking  Conformance Test

Virtual Machine

EVM-NG  WASM

OpenData

On-Chain Computing  Trusted Database  Trusted Storage

IBC  Multi-Chain Extension  Oracle

GALAXY consensus  HPAN network  TAE trusted acc-env  DSNM node mgmt  PLBP pipeline packaging  FEIS storage access

One-Key Deployment  Monitor & Operation
Galaxy Pluggable Consensus Framework

PoET  Raft  PoW  FBFT

2-Phase State-Machine Replication
Dynamic SuperNova Management

Node1

1. Sync Newest Block
2. Update Contract State
3. Get Consensus Node

Consensus Module

Node2

1. Sync Newest Block
2. Update Contract State
3. Get Consensus Node

Consensus Module

Consensus

Sync

Newest

Block

Update

Contract

State

Get

Consensus

Node

Contract
Trust Identity

Social Anchor
- Social-ID
- Bio-ID
- Virtual DNA
- HW Address

Distributed ID
- Multidimensional characterization
- Identity authentication
- TAE
- Byzantium Game
- Security

Env Info characteristic

Time Series Record

TrustBase
Trusted Access Environment

User

External Payment

TrustPay Wallet

ZKPs

Setup \( y = F(x) \)

Order Generator

Order Verifier

On-Chain Contract

Initiator

Enclave

DRNG Instruction

DTRNG

SGX Remote Attestation powered Consensus

Encrypted msg

Rand Num

Challenger

Challenger

Rand Num from SGX

User

Trusted FrameBuffer

External Payment

Alipay

GOJEK

PayPal

TrustPay Wallet

ZKPs

Key

Verifier

\( y \) Decl out

\( x \) Priv input

TrustRPC

Order

Payment Process

off-chain

支付流程

Payment Info

Extract

Trust Proof

Order No. + Pay Digest

Signature

Uniq Order No.

Order No. + Pay Digest

Pay Digest + Signature

TrustProof

Order Generator

Order Verifier

On-Chain Contract

Initiator

Enclave

DRNG Instruction

DTRNG

SGX Remote Attestation powered Consensus

Encrypted msg

Rand Num

Challenger

Challenger

Rand Num from SGX
Distributed Time-Series Database

User Interface
- External API
- SQL Support
- Output Formatting

Transaction Consensus
- Tendermint
- log
- log
- log
- log

Trusted Mechanism

- Transaction Format Verification
- Params Acquisition
- Transaction Transform Execution
- Result Verification
- Update State-related Parameters
- Query Result Verification
- Current State Hash

Transaction Execution
- Auth
- Query
- Update
- Insert
- Delete
- ……

Data Store
- SQL Engine
- TSM Tree
- Local Data Operation
What is Oracle?

- Ability to access off-chain resources from smart contract
- Channel between trusted area and semi-trusted area
  - Storage
  - Database
  - Hardware
  - Large platform
- Many types of oracles
  - Software oracle
  - Hardware oracle
  - Consensus oracle
  - Inbound and outbound oracle
Oracle

(On-Chain) Node1
Consensus
VM

(On-Chain) Node2
Consensus
VM

Consensus
Consensus

EVM
Call Interceptor

System Call
Contract Route
Contract1
Contract2
ContractN

Normal Call

(Off-Chain) Data Source
Off-Chain Access Channel

• EVM `Call` instruction is used to call another contract; if we learn the detail of `Call` instruction:

  Call Instruction:

  ```go
  func opCall(pc *uint64, interpreter *EVMInterpreter, contract *Contract, memory *Memory, stack *Stack) ([byte, error] { 
      (...) // prepare call params
      ret, returnGas, err = interpreter.evm.Call(contract, toAddr, args, gas, value)
  
      (...) // store return value to mem
      return ret, nil
  }
  
  replace the contract call with local func call
  ```

  Execute Called Contract

• So when calling another contract, we can use `Go function` as the called contract.

  Call Instruction:
• Call Interceptor:
  • every contract have an unique address, so the contract address is a good way to identify `Normal contract` and `System Contract`
• **Contract Router:**
  • As there have many different system contracts, we use a contract router to route system contract Call.

• **Contract Router API:**

  ① Registe (contractAddr, ContractHandler)  // register a system contract

  ② routeTable  // update route table

  ③ GetHandler(contractAddr) ContractHandler  // retriue contract handler
Off-Chain Access Channel

- **Contract Handler:**
  - When processing a call request, Handler will decode the input params and execute the native function to get the off-chain data.

  - As the solidity `Input Param` / `Return Value` is already encoded, we must use a codec to decode/encode the data.

- Request processing flow:
In ABI encoding specification, a contract call input in the following format:

\[
\text{[Function Selector]} \ [\text{Encoded Params}]
\]

**Function Selector:**
- The *first four bytes* of the call data for a function call specifies the function to be called. It is the first (left, high-order in big-endian) four bytes of the Keccak-256 (SHA-3) hash of the signature of the function.
- e.g. Selector for `function test(string name) public view` is `0xf9fbd554`

**Dynamic Types:**
- For Dynamic types, the encoded value of the data will consist of two parts: *data length*, *data content*;
- Format: `[Data Length][Data Content]`
Off-Chain Access Channel

ABI Decoder:

```go
func DecodeParams(input []byte, args ...interface{}) error {
    for i := 0; i < len(args); i++ {
        rv := reflect.ValueOf(args[i])
        if rv.Kind() != reflect.Ptr || rv.IsNil() {
            return InvalidUnmarshalError
        }
    }
    switch rv.Elem().Type().Kind() {
    case reflect.String:
        ...
    case reflect.Slice:
        ...
    case reflect.Uint64:
        ...
    default:
        return UnSupportedTypeError
    }
    return nil
}
```

ABI Encoder:

```go
func EncodeReturnValue(retVals ...interface{}) ([]byte, error) {
    retPre := make([]byte, 0)
    retData := make([]byte, 0)
    preOffsetPadding := len(retVals) * constant.EvmWordSize
    for _, retVal := range retVals {
        retType := reflect.TypeOf(retVal)
        switch retType.Kind() {
        case reflect.String:
            ...
        case reflect.Slice:
            ...
        case reflect.Uint64:
            ...
        case reflect.Array:
            ...
        default:
            return nil, UnSupportedTypeError
        }
    }
    return append(retPre, retData...), nil
}
```
Oracle-Based Cross-Chain

• As the oracle can directly access the off-chain data, we can use Smart Contract as the cross-chain transaction agent, and finally all cross-chain transactions will be verified independently by every node.

• Cross-Chain:
The Art of Blockchain Extension

Oracle Sub-System
- Cloud Oracle
- Pay Oracle
- HW Oracle
- DB Oracle
- Storage Oracle

Sub Chain
- Contract
- Contract
- OS

Data-Share Chain

Sub Chain
- Contract
- OS

Base Chain

IBC
- Mgmt Contract
- Orchestration

Event Contract
- Grant Table
- MTA Contract

Service Contract

HYPERCALL
- oracle front
- oracle back
Conclusion

- Two-layered Intelligence BIoT Platform Solution
- Cloud-Edge Infrastructure Design and Implementation
- Pluggable Consensus Framework
- TAE and TrustBase
- Oracle Technology
- Chain Extension

https://github.com/DSiSc