Building a Powerful Edge Using Blockchain

Zhao Zhenlong, Jing Shuaishuai
TrustChain Technology

AGENDA

- About Us
- How to build a Powerful Edge Using Blockchain
- Two-layered Intelligence Solution
- Cloud-Edge Infrastructure
- Blockchain Technologies in Edge

About Us

世纪诚链
Trust Chain Technology

University

Open Source

Community

BH-Chain
private

Experimental field

Core Arch / Algo

**TrustChain
Enterprise**

Contribution

Components

Hyperledger
Community

From Open Source to Open Ecosystem

HYPERLEDGER

Use CVT-Network Enhancing Edge

世纪诚链
Trust Chain Technology

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

- Trusted Exec Environment
- Blockchain
- Smart Contract

Two-layered Intelligence BIoT

世纪诚链
Trust Chain Technology

AI / BigData / Cloud

Core Intelligence：Mind & Decision

Blockchain provides Consensus-Trusted Info to AI

Trust from Consensus

Smart Contract

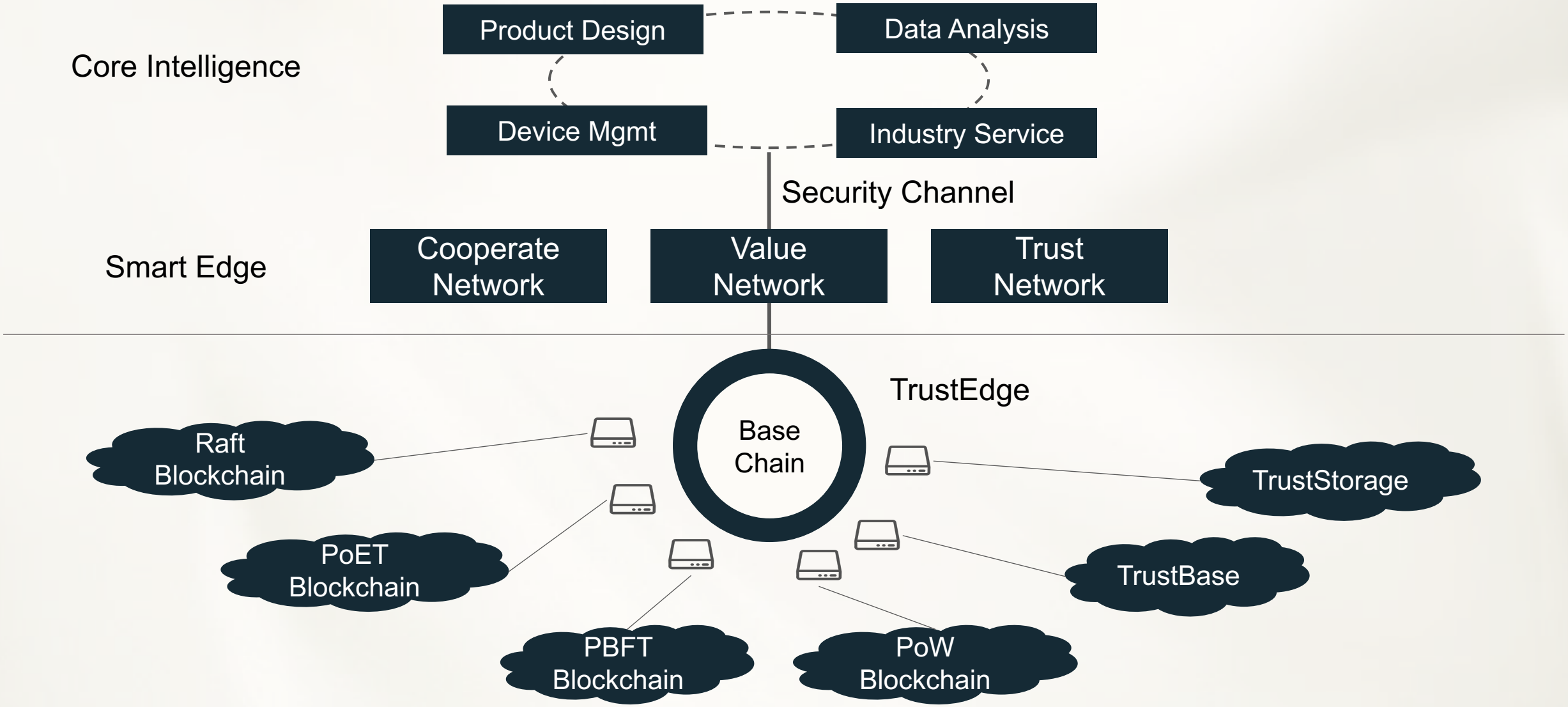
Edge Intelligence：Basic Rule

IoT provides Objective-Trusted Info to Blockchain

Trust from Objective

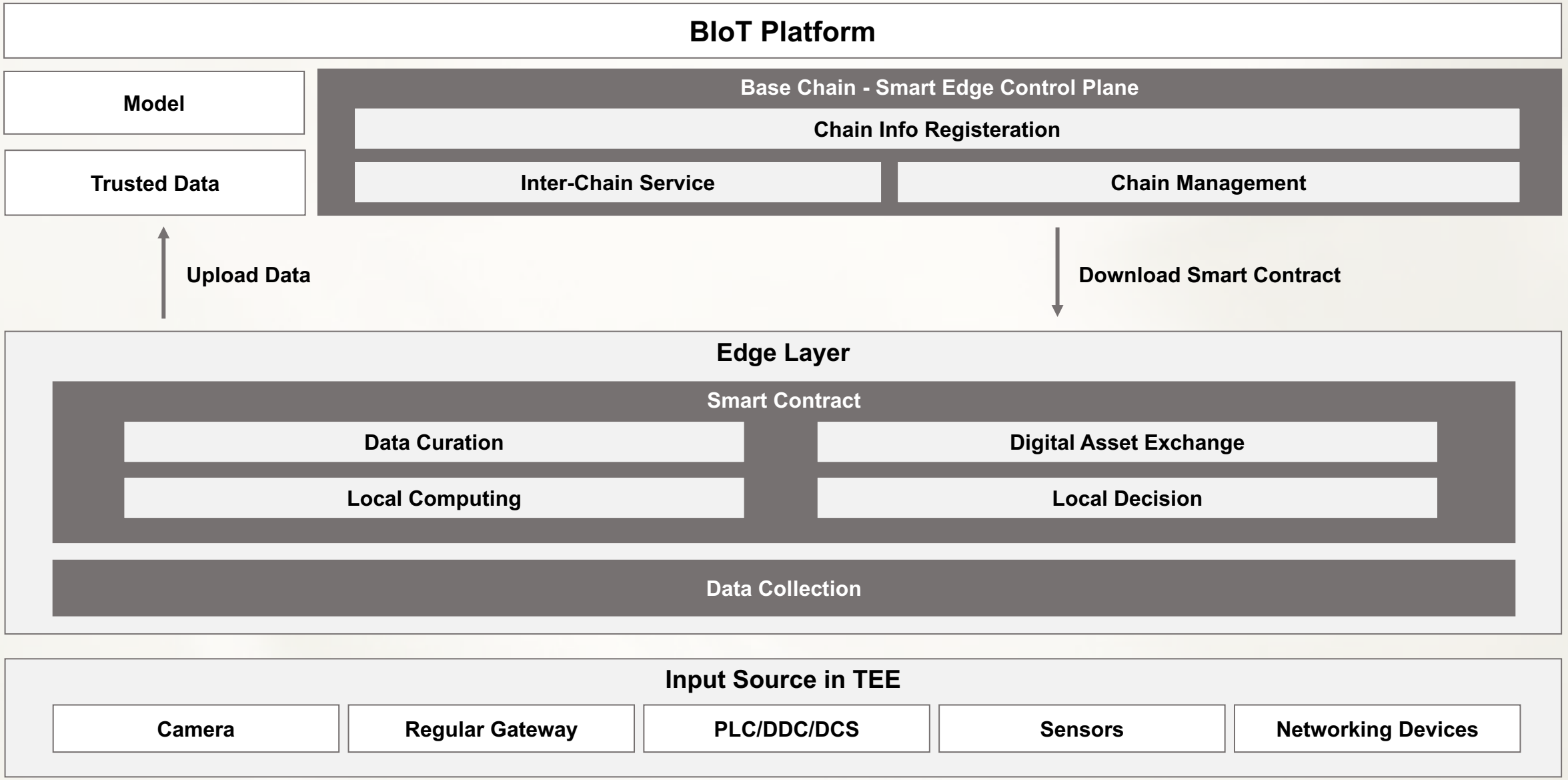
Two-layered Intelligence BIoT

世纪诚链
Trust Chain Technology



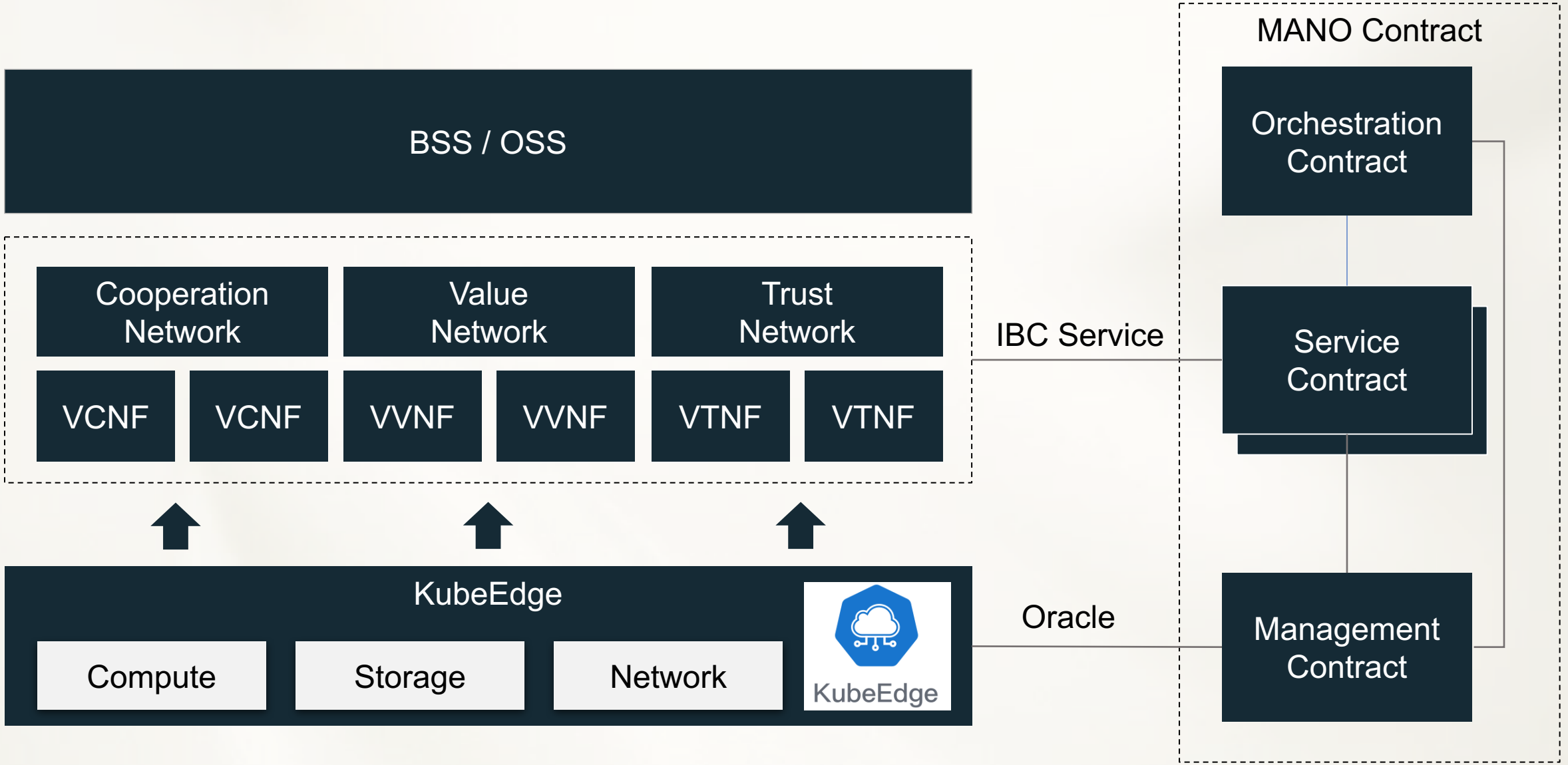
BIoT Platform

世纪诚链
Trust Chain Technology



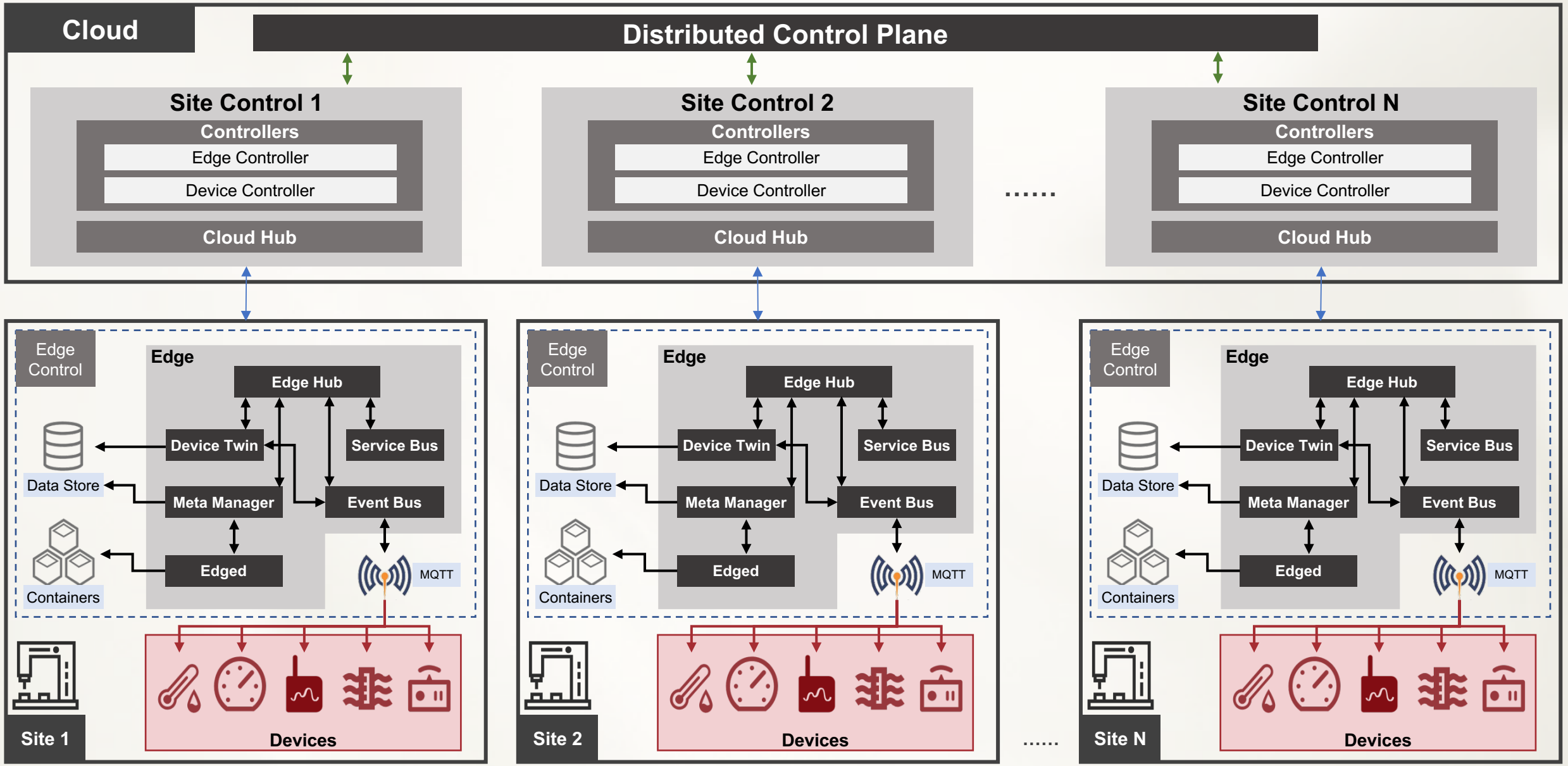
Edge Infra from NFV to CVT-NFV

世纪诚链
Trust Chain Technology



Edge Infra from NFV to CVT-NFV

世纪诚链
Trust Chain Technology



BIoT Blockchain Architecture

世纪诚链
Trust Chain Technology

TrustEdge | TrustBase | TrustStorage | TrustStream | TrustPay

HW-Oracle | Database-Oracle | Storage-Oracle | Cloud-Oracle | Pay-Oracle

Identity

Base Chain

Local Chain

HYPERLEDGER SAWTOOTH

Local Chain

Tendermint

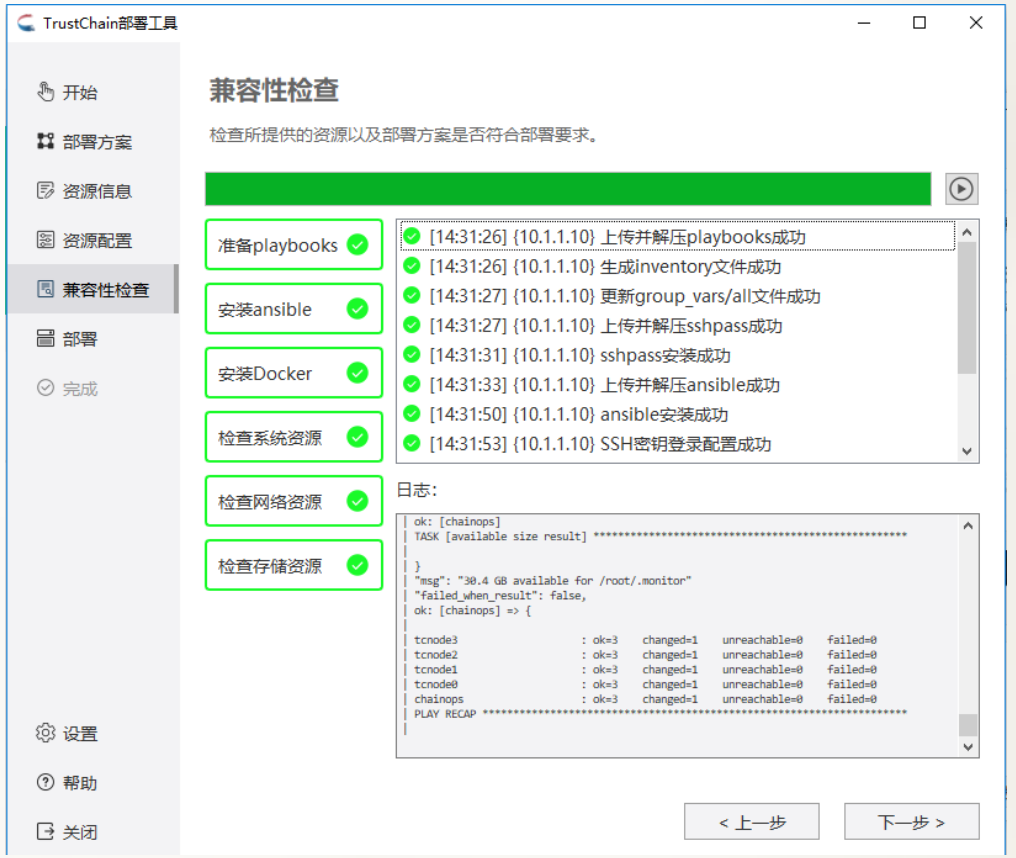
Local Chain

TrustChain

TrustOps Developer Toolstack

世纪诚链 Trust Chain Technology

- 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)



HYPERLEDGER SAWTOOTH

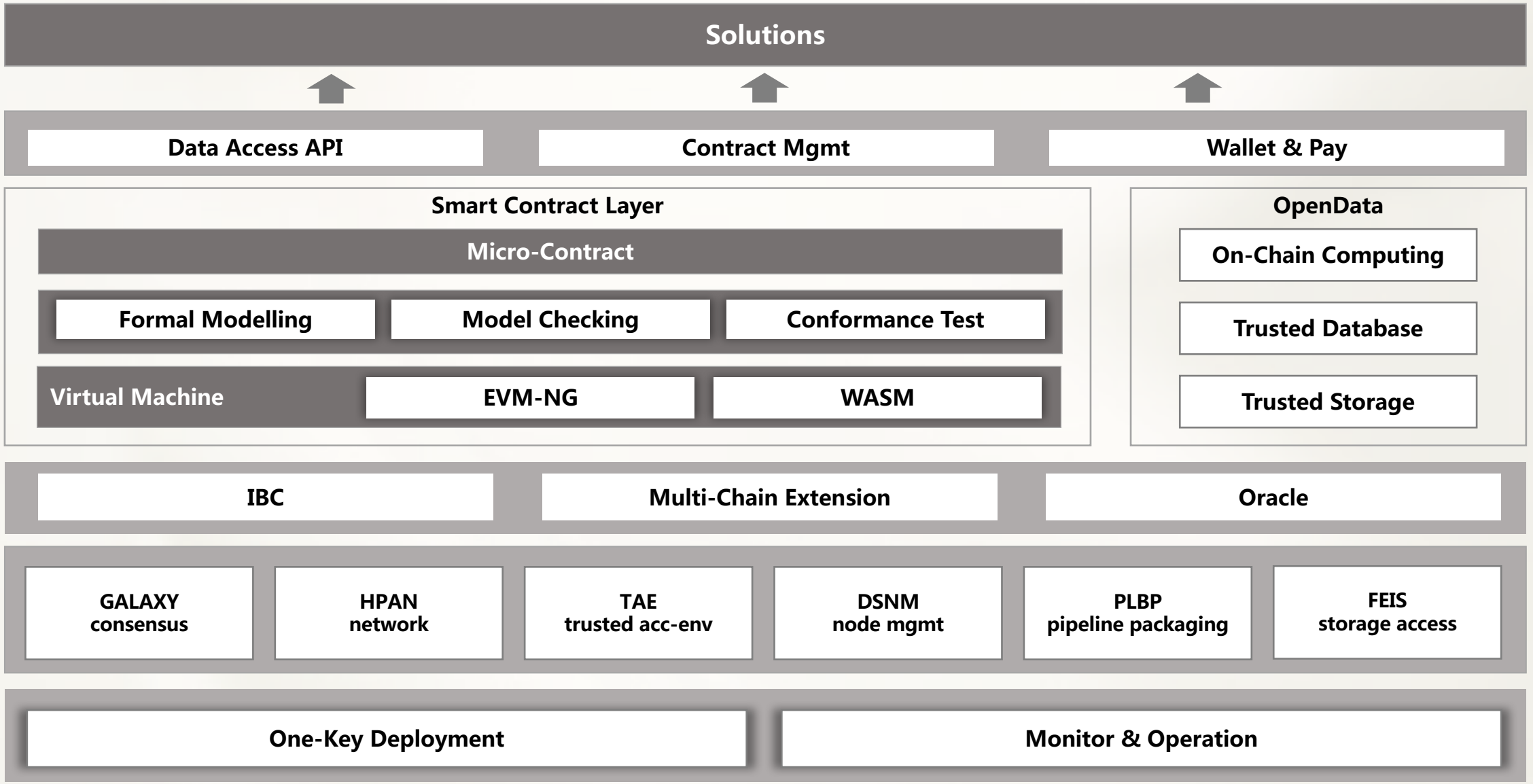
HYPERLEDGER BURROW

Tendermint

TrustChain

TrustChain Tech Arch

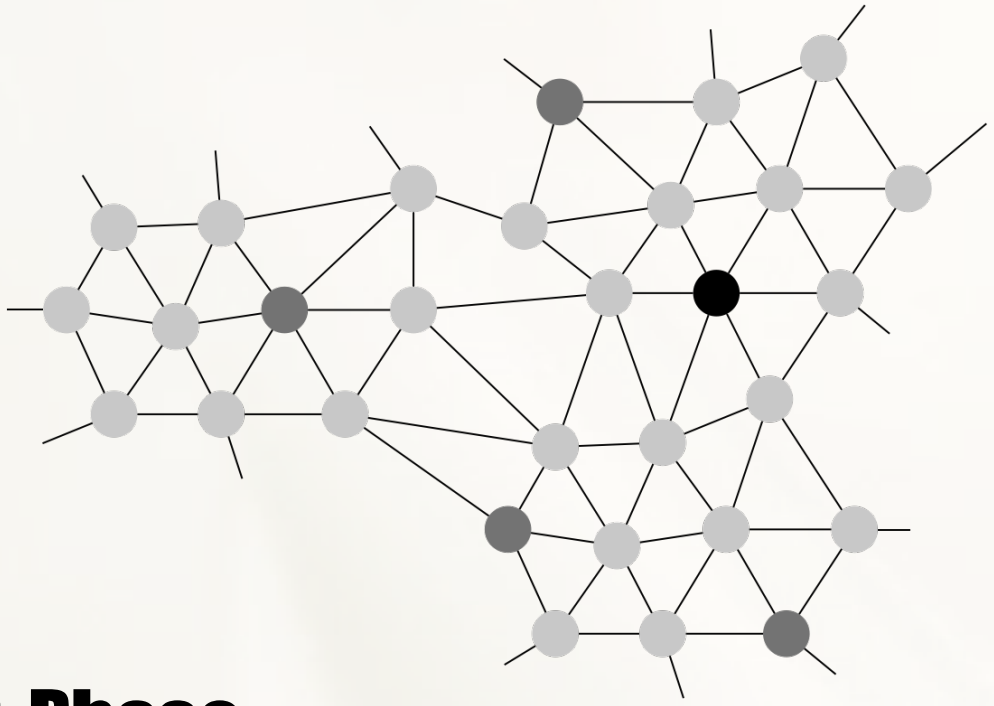
世纪诚链
Trust Chain Technology



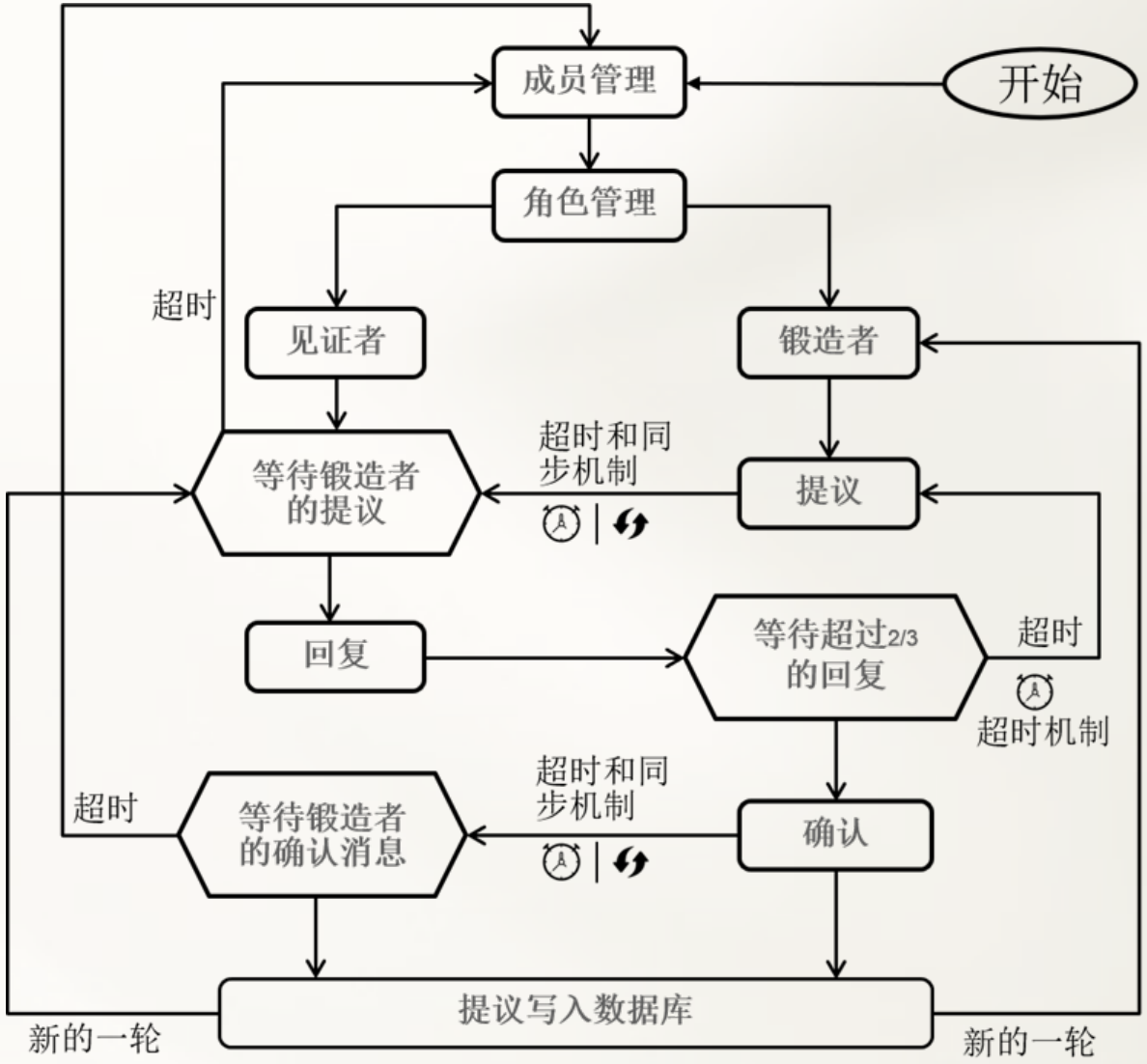
Galaxy Pluggable Consensus Framework

世纪诚链 Trust Chain Technology

PoET | Raft | PoW | FBFT

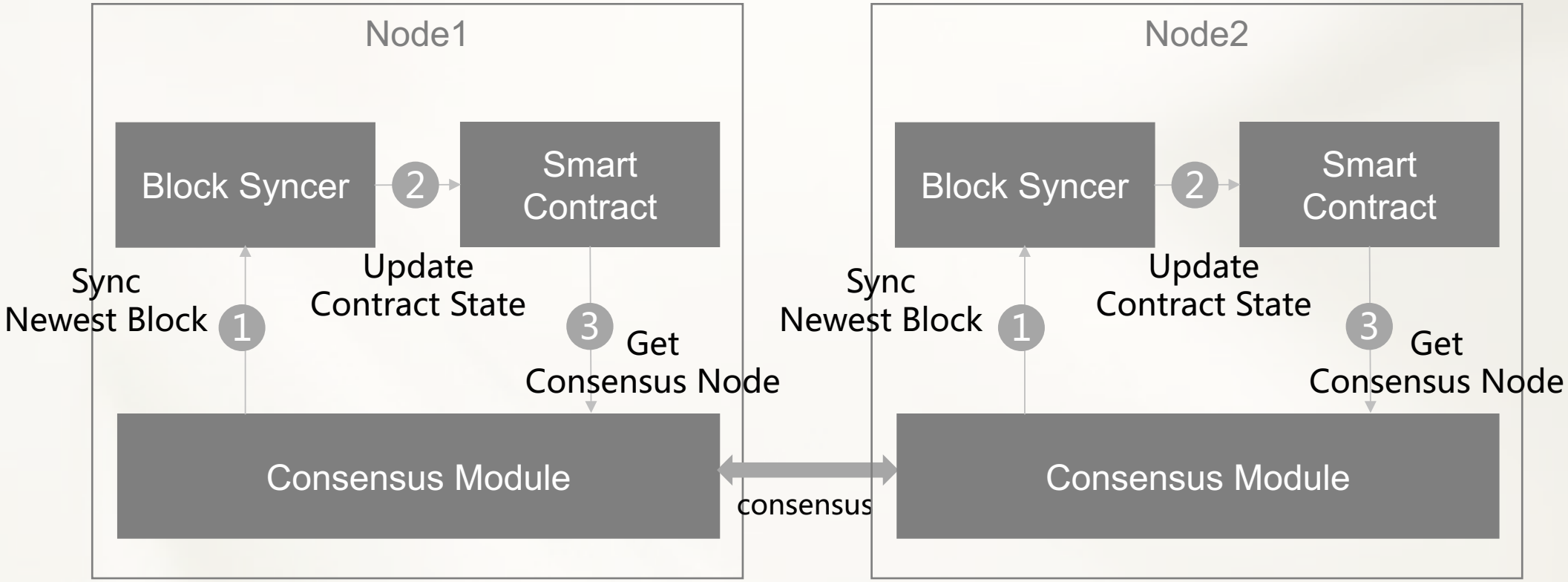


2-Phase State-Machine Replication



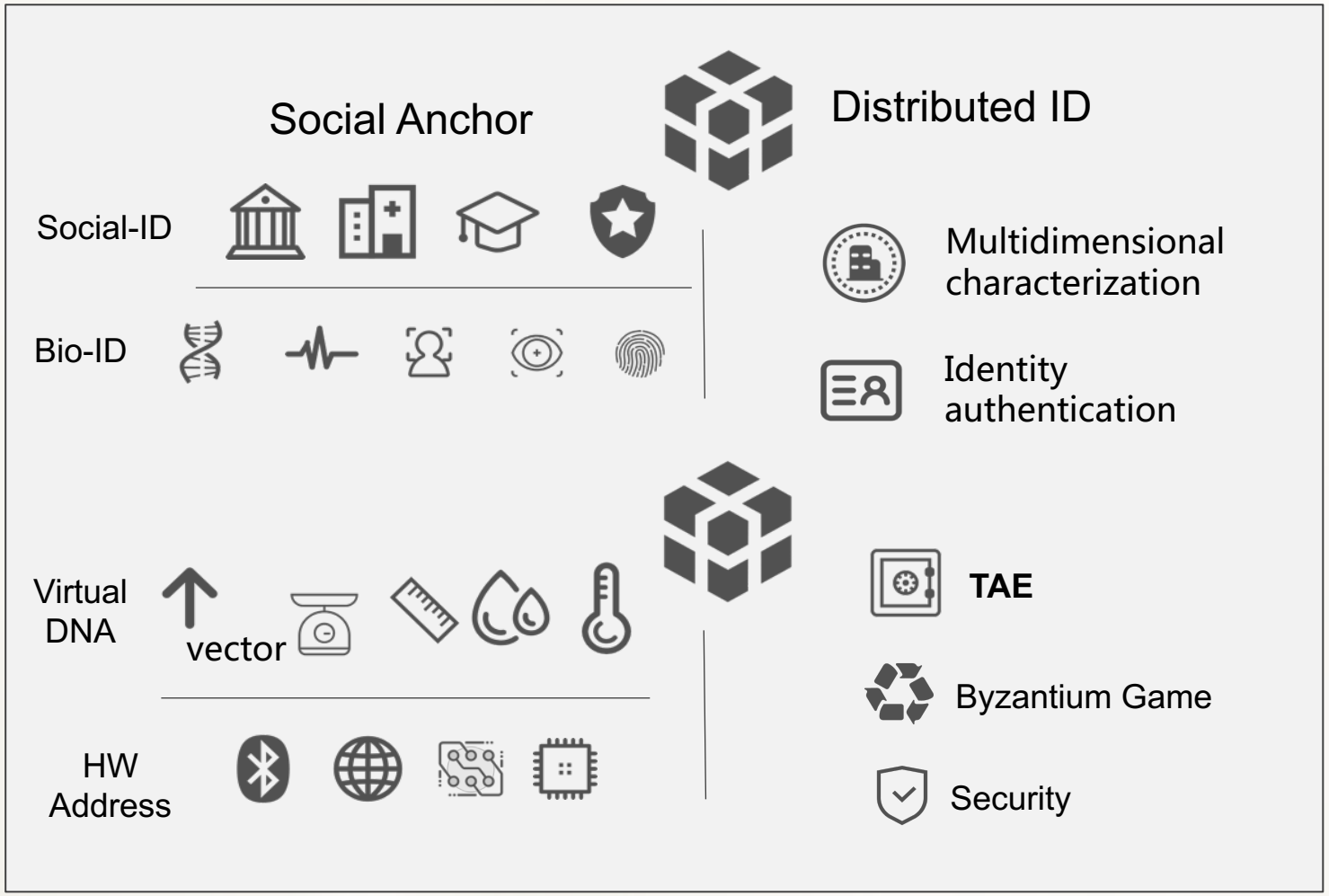
Dynamic SuperNova Management

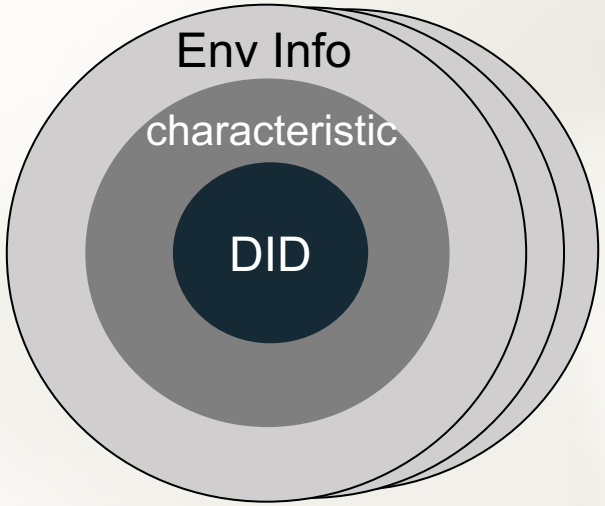
世纪诚链
Trust Chain Technology



Trust Identity

世纪诚链
Trust Chain Technology

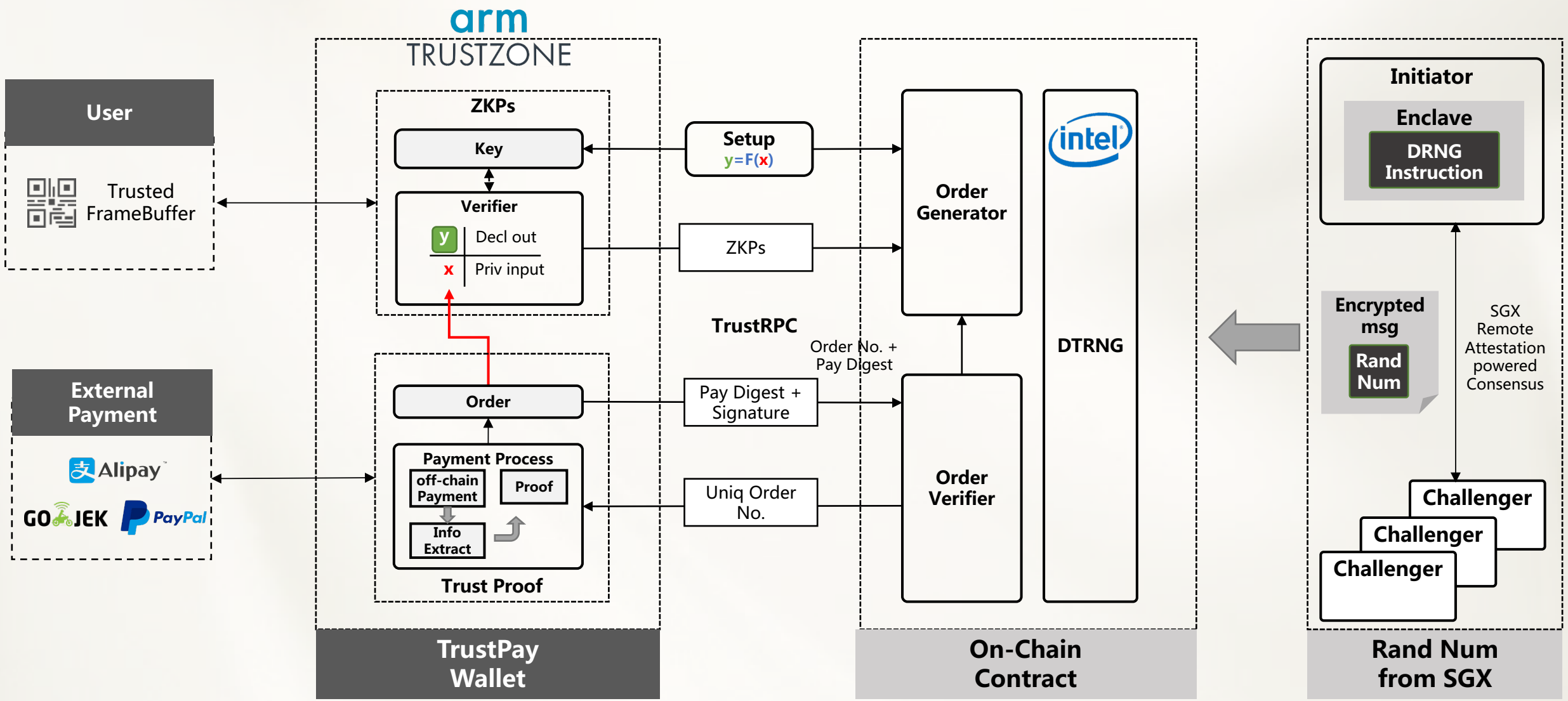




Time Series Record

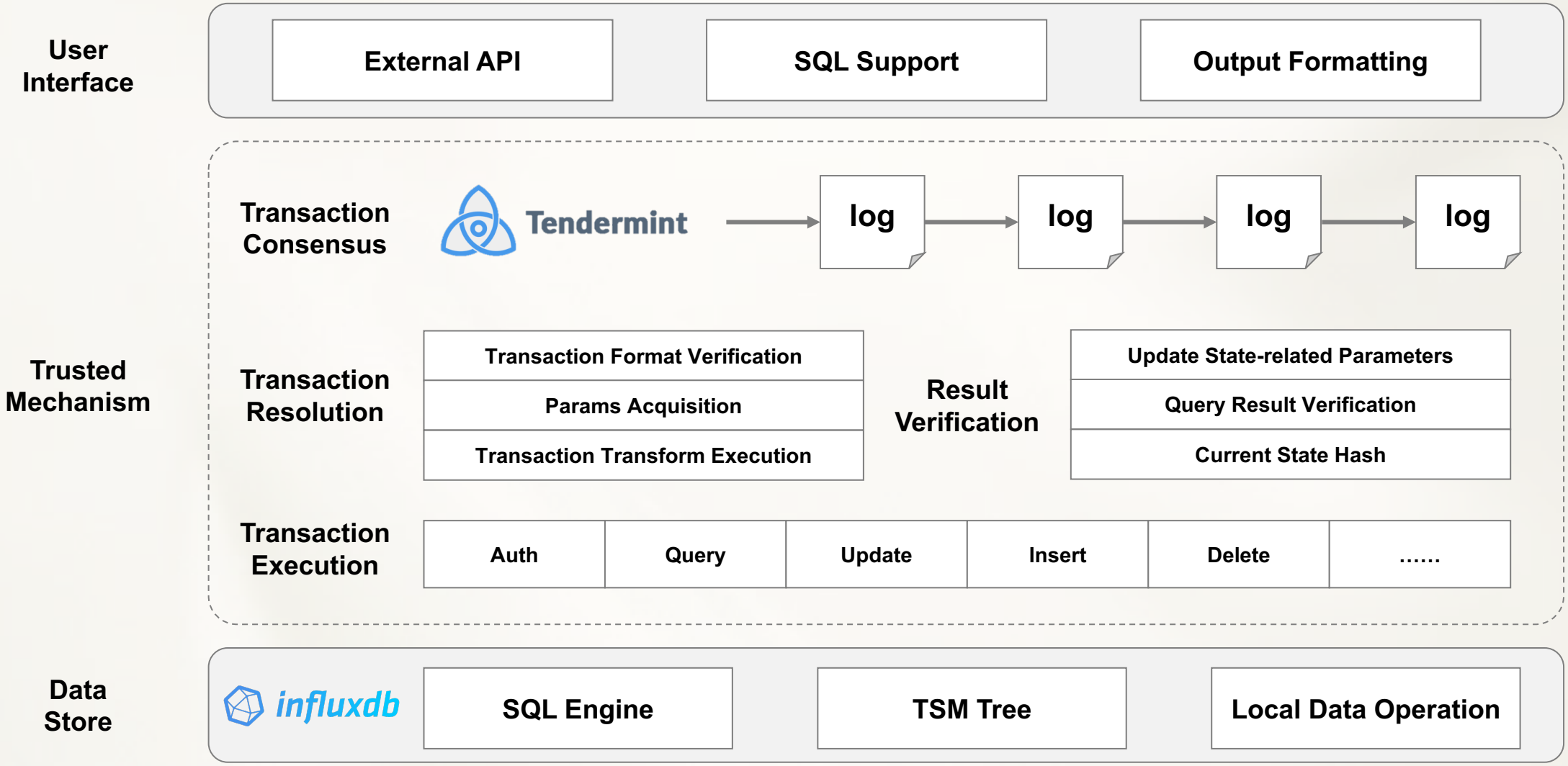
Trusted Access Environment

世纪诚链 Trust Chain Technology



Distributed Time-Series Database

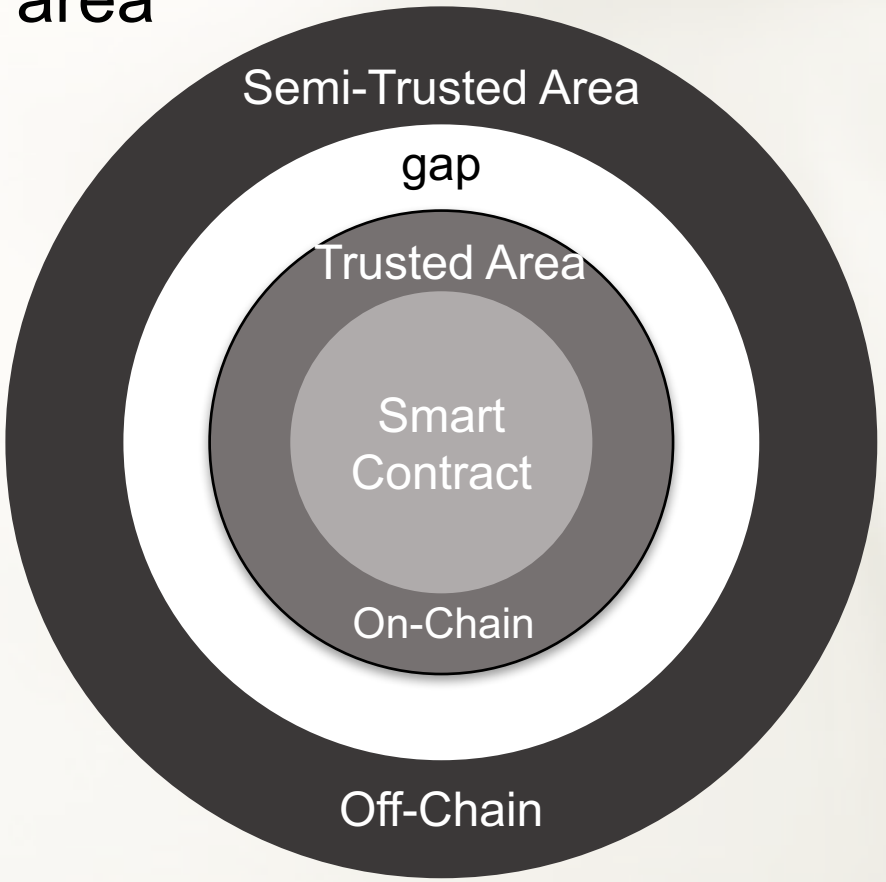
世纪诚链 Trust Chain Technology



What is Oracle ?

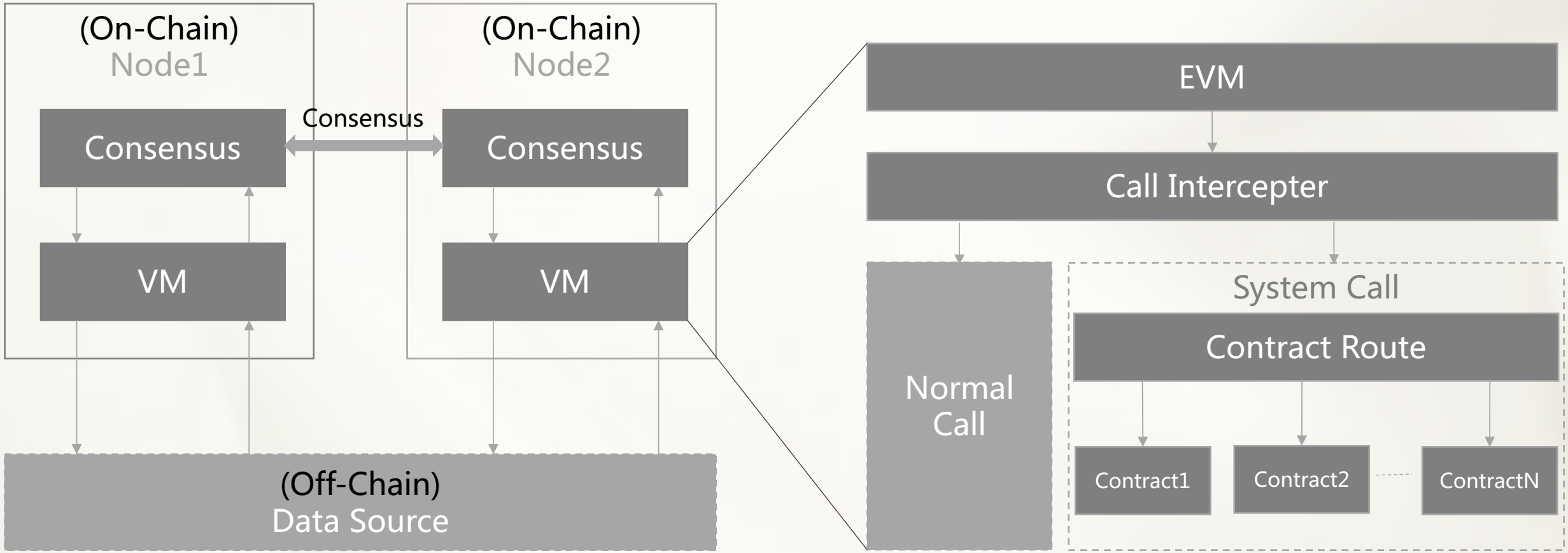
世纪诚链
Trust Chain Technology

- 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

世纪诚链
Trust Chain Technology



Off-Chain Access Channel

世纪诚链 Trust Chain Technology

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

Call Instruction:

```
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
}
```

Execute Called Contract

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

Call Instruction:

```
func opCall(pc *uint64, interpreter *EVMInterpreter, contract *Contract, memory *Memory, stack *Stack) ([]byte, error) {
    {...} // prepare call params

    ret, returnGas, err = localCall(contract, toAddr, args, gas, value)

    {...} // store return value to mem
    return ret, nil
}
```

Replace the contract call with local func call

Off-Chain Access Channel

世纪诚链
Trust Chain Technology

- **Call Intecepter**:
  - every contract have an unique address, so the contract address is a good way to identify `Normal contract` and `System Contract`

```
func opCall(pc *uint64, interpreter *EVMInterpreter, contract *Contract, memory *Memory, stack *Stack) ([]byte, error)
    {...}

    if IsSystemContract(toAddr) {
        ret, returnGas, err = sysContractCall(interpreter.evm, contract.self, toAddr, args, gas, value)
    } else {
        ret, returnGas, err = interpreter.evm.Call(contract, toAddr, args, gas, value)
    }

    {...}

    return ret, nil
}
```

Go System Contract

Normal Contract

Off-Chain Access Channel

世纪诚链 Trust Chain Technology

- **Contract Router**:
  - As there have many different system contracts, we use a contract router to route system contract Call.

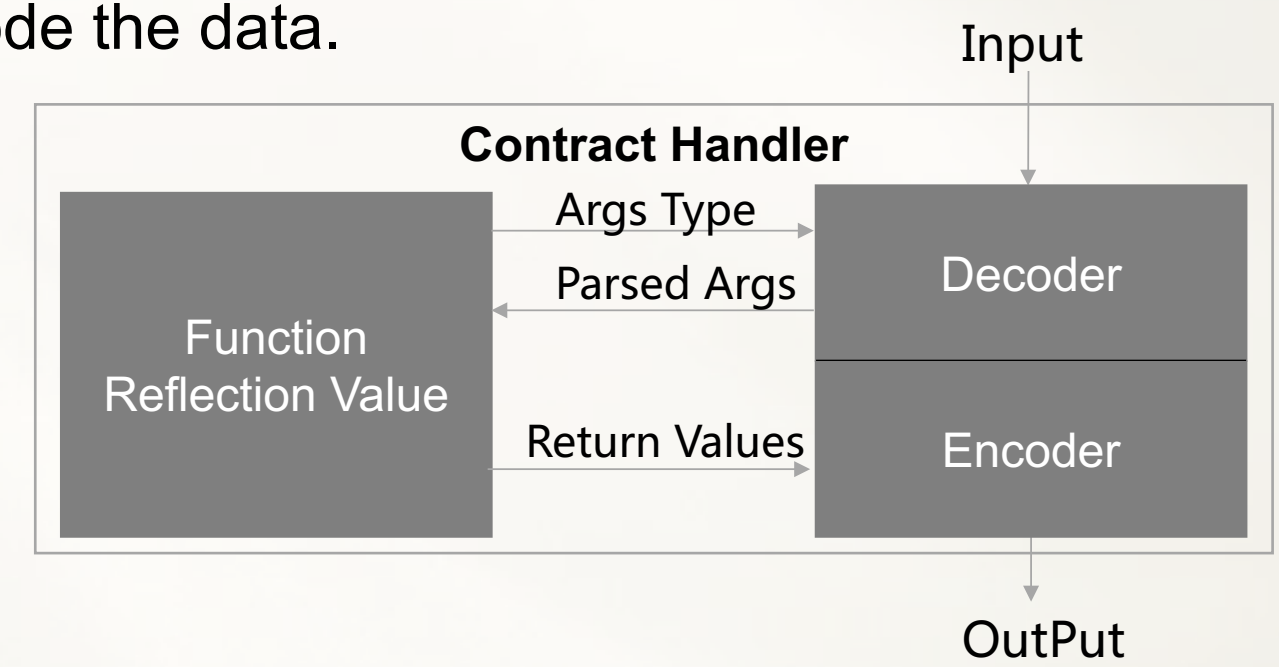
  - Contract Router API：

① **Registe (contractAddr, ContractHandler)**		// register a system contract
② ***routeTable***		// update route table
③ **GetHandler(contractAddr)**	**ContractHandler**	// retrive contract handler

Off-Chain Access Channel

世纪诚链 Trust Chain Technology

- **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:



Off-Chain Access Channel

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

  [Function Slector] [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

世纪诚链 Trust Chain Technology

ABI Decoder:

Reflect
Get Data Type

```
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
}
```

Encode/Decode
Specified Type Data

ABI Encoder:

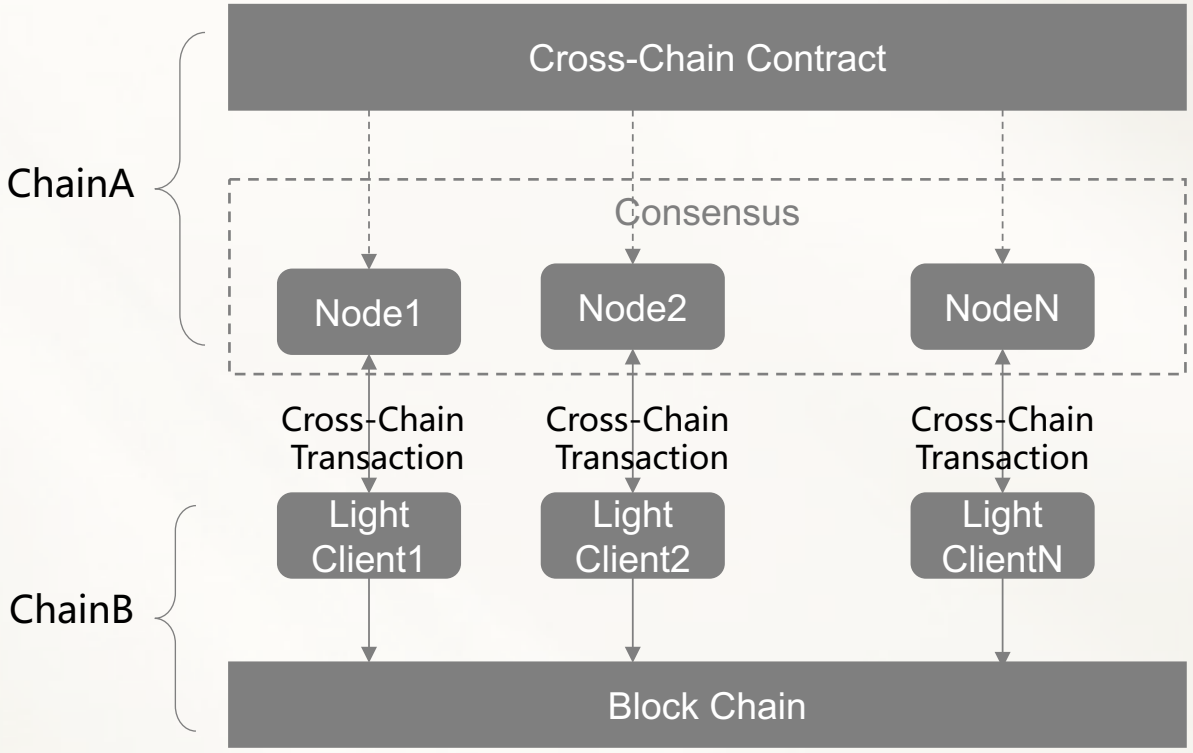
```
func EncodeReturnValue(retVals ...interface{}) ([]byte, error) {
    retPre := make([]byte, 0)
    retData := make([]byte, 0)
    preOffsetPadding := len(retVals) * constant.EvmWordSize
    for _, retVal := range retVals {
        retType := reflect.TypeOf(retVal)
        switch retType.Kind() {
        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

世纪诚链
Trust Chain Technology

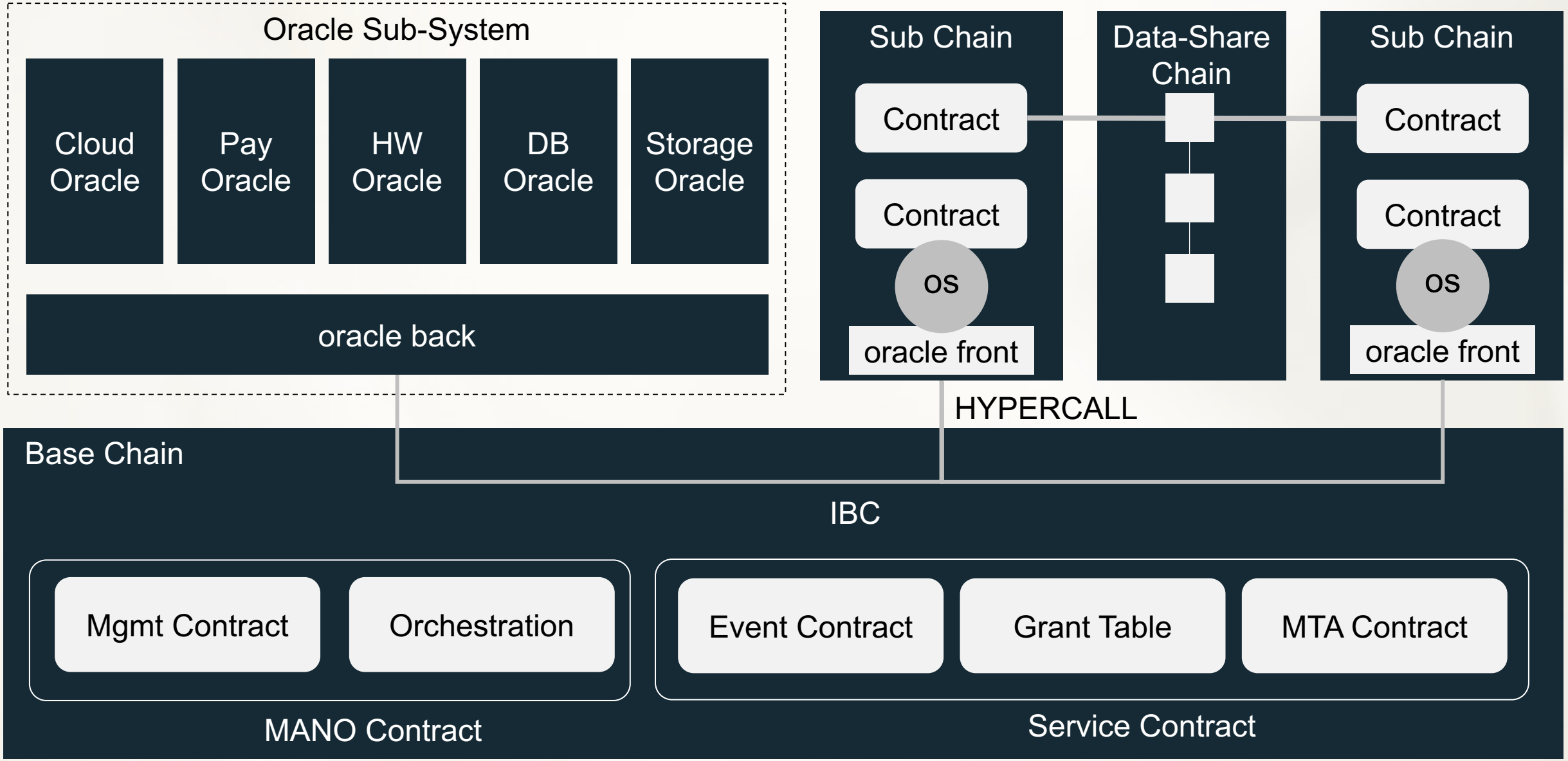
- As the oracle can directly access the off-chain data, we can use Smart Contract as the cross-chain transaction agent, and finally all cross-chain transactions will be verified independly by every node

- Cross-Chain:



The Art of Blockchain Extension

世纪诚链
Trust Chain Technology



Conclusion

世纪诚链
Trust Chain Technology

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

https://github.com/DSiSc


OPEN SOURCE SUMMIT
China 2019