Wanchain Documentation

Release 1.0

WAnchain community

Jul 31, 2018

Contents

wanchain

Wanchain Documentation

CHAPTER 1

Contents

1.1 Introduction

1.1.1 What is Wanchain?

Wanchain aims to build a blockchain infrastructure that can be used to realize the connection and value exchange of different block-chain networks in a decentralized way. It uses the latest theory of cryptography to make up for the shortcomings of existing inter-chain protocol. Any blockchain network, whether public chain, private chain or alliance chain, can be access to Wanchain to complete the inter-chain transaction of digital assets at a low cost. Wanchain is not just a generic inter-chain protocol, but also a distributed ledger that records inter-chain transactions and in-chain transactions. This ledger supports not only the smart contracts, but also the privacy protection of smart contract tokens trade. It is a qualitative leap to the Interconnection of thousands of Chains and turn of Chains into Internet, and the centralized application of advanced cryptography theory in the field of blockchain, as well as a new and complete blockchain network with high efficiency and completeness. Wanchain is about to reshape the economic ecology of blockchain.

1.1.2 Differentiation of other blockchain product

From the perspective of blockchain industry's development, the ultimate vision of Wanchain is to create an "Internet" of blockchain to achieve the qualitative leap of blockchain turning from chains to internet. In Wanchain's ecosystem, thousands of chains are interconnected, where value can be transferred free from barriers, giving full play of blockchains function of carrying and transferring value. Wanchain is about to reshape the economic ecology.

From the perceptive of the development of blockchain's underlying technology, Wanchain applies secure multi-party computation, threshold secret sharing, ring signature scheme based on elliptic curve, one-off generation mechanism of account and many other cutting-edge technologies of cryptology, and solves privacy protection problem of smart contract token transaction. To some extent, Wanchain is the typical application of advanced cryptography solving the real problems of blockchain, representing the development direction of blockchain's underlying technology, that is, transforming from conditional security to provable security, from logical control to algorithm theory control. Wanchain boasts a professional cryptography research team who will continue to contribute to the industry as a whole for the cryptography applying in the field of blockchain.

From the perspective of blockchain application, Wanchain is more than a blockchain project realizing inter-chain transaction and interoperability of multiple assets, but a complete blockchain developing platform. While achieving the function of inter-chain transaction, Wanchain is also a blockchain network that can run independently: it

contains native coin, supports smart contracts, and has privacy protection mechanisms for smart contract token transactions. Any developers may develop application met their need on Wanchain in accordance with application scenario.

1.1.3 Milestones

2016/06

Research on privacy protection and cross-chain transactions

2016/12

Development on Proof of Concept

2017/06

White Paper published. Wanchain.org online

2017/09

ICO Crowd funding

2018/01

Wanchain 1.0 goes live (Privacy Protection, WAN, Wallet, Block chain Explorer)

2018/06 (ongoing)

Wanchain 2.0 (Integration with Ethereum, Multi-Coin Wallet)

2018/12 (ongoing)

Wanchain 3.0 (Integration with Bitcoin, Multi-Coin Wallet)

2019/12 (ongoing)

Wanchain 4.0 (Integration with Private Chains, Multi-Coin Wallet)

1.1.4 Community

Official Links:

- Website
- Reddit
- Telegram Ann
- Telegram Chat
- Medium
- Facebook
- Twitter
- Instagram
- Whitepaper
- Yellowpaper
- Commericalpaper
- Meet the team

1.2 Wanchain Clients

1.2.1 Go-Wancahin

1.3 Connecting to Wanchain Clients

1.3.1 GUI WAN Wallet

There are three GUI wallets for belowing type of operate systems:

- Windows OS
- Mac OS
- Linux OS

We have an universal **How to Use GUI wallet guide** to guide you step by step to quickly get familar with our wallet

Getting started with Wanchain wallet

- https://github.com/wanchain/go-wanchain/wiki/Getting-Started-with-Wanchain-Wallet

How to Use GUI Wallet guide

- https://github.com/wanchain/go-wanchain/wiki/How-To-Use-(GUI)-Guide

1.3.2 Command Line

We provide command line tools for anyone who is comfortable using CMD terminology

Please find How to Use CLI Wallet guide here

1.4 WAN Account

We have a step-by-step tutorial process on github to guide you through https://github.com/wanchain/go-wanchain/wiki/How-To-Use-(GUI)-Guide

1.4.1 Keystore

Keystore is where controls your account details, if you lose your keystore files you are at high risks to lose all your current assets, therefore we strongly recommend you backup all keystore files at a place you could trust and won't forget (ie Offline PC device or U-Disk etc)

At any circumstances, you should NOT share or reveal your keystore information with anyone you don't trust, you might get stolen by doing so . If you think your keystore has been leak to any 3rd parties, please transfer your assets to a new account !

You can always find your accounts & application data at backups

Open your Wallet GUI click: *File -> Backup -> Accounts /Application data*

Once you clicked Accounts it will pop up a folder with "Keystore" file inside.

Here is the root : *C:UsersAdministratorAppDataRoamingwanchain*

Double click Keystore files under file wanchain, you will find all your accounts information and those file/files are what you need to keep in safe all the time.

1.4.2 Creating Account

1. Have an existing account

   If you have an existing keystore file , you can simply conduct drag & drop on your wallet GUI to get it back

   File -> Import accounts -> Drag & Drop -> Done

2. Create a new account

   If you need to create a new account , simply click **"Account"** once you log in wallet GUI .

   Account -> Input New Account Name -> Enter/Re-enter password -> Done

1.4.3 Ordinary & Private Address

Ordinary address (Short address) as known as public address which allows everyone see transaction details. On the opposite, Private address (Long address) as known as MSA (Master Stealth Address) which details will not be revealed.

```
Example of Oridinary address :
0xefe000C1b9f9ca9bf063857aAF5fCb7B8A25eaA1

Example of Private address :
0x02bddd6c139a10c5c9e81d1d6438dd26bc4a26824a2c729819d21ee1fca8b2dbc203936c798596ac4adcbe89e96c883
```

1.4.4 Backup and restore accounts

The process to backup your acounts :

Click *File -> Backup -> Application Data*

You will see files under **WanWalletGui** as defaulted. Please select them all and make copy to a safety place where you can trust.

The Accounts & Application Data stores all your public transaction records, private transaction records and OTA balance. If you delete these files , you will not lose your assets but you will not be able to see your assets at your current wallet at meantime unless you back them up in advance . However, you can always call it back via "Import Account" which we have demonstrated to you in above content.

1.5 WAN

1.5.1 What is WAN?

WAN or WANCoin known as the native currancy of WAN chain. Both common intra-chain transactions and inter-chain transactions will consume a certain amount of WAN, and at the same time, WANC will be used as a margin for inter-chain verification node.

1.5.2 WAN Supply

The base unit of WAN is called Win. Blow is a list of named demonimations and their value in Win accordinly.

Unit	Win Value	Win
Win	1 win	1
Kwin	1e3 win	1,000
Mwin	1e6 win	1,000,000
Gwin	1e9 win	1,000,000,000
Szabo	1e12 win	1,000,000,000,000
Finney	1e15 win	1,000,000,000,000,000
Wan	1e18 win	1,000,000,000,000,000,000

1.6 The Wanchain network

1.6.1 Main network

1.6.2 Test network

Pass the `--testnet` argument to the client. e.g.:

```
> gwan --testnet

> wanwalletgui --network testnet
```

1.7 Contracts and Transactions

1.7.1 How To Deploy Smart Contract On Wanchain

1.7.1.1 Smart Contract Source Code Example

Note: The following smart contract code is only an example and is NOT to be used in Production systems.

```
pragma solidity ^0.4.11;

/**
* Math operations with safety checks
*/
library SafeMath {
function mul(uint a, uint b) internal pure returns (uint) {
    uint c = a * b;
    assert(a == 0 || c / a == b);
    return c;
}

function div(uint a, uint b) internal pure returns (uint) {
    assert(b > 0);
    uint c = a / b;
    assert(a == b * c + a % b);
    return c;
}

function sub(uint a, uint b) internal pure returns (uint) {
    assert(b <= a);
    return a - b;
}
```

(continues on next page)

(continued from previous page)

```
function add(uint a, uint b) internal pure returns (uint) {
    uint c = a + b;
    assert(c >= a);
    return c;
}

function max64(uint64 a, uint64 b) internal pure returns (uint64) {
    return a >= b ? a : b;
}

function min64(uint64 a, uint64 b) internal pure returns (uint64) {
    return a < b ? a : b;
}

function max256(uint256 a, uint256 b) internal pure returns (uint256) {
    return a >= b ? a : b;
}

function min256(uint256 a, uint256 b) internal pure returns (uint256) {
    return a < b ? a : b;
}
}

contract ERC20Protocol {
    /* This is a slight change to the ERC20 base standard.
    function totalSupply() constant returns (uint supply);
    is replaced with:
    uint public totalSupply;
    This automatically creates a getter function for the totalSupply.
    This is moved to the base contract since public getter functions are not
    currently recognised as an implementation of the matching abstract
    function by the compiler.
    */
    /// total amount of tokens
    uint public totalSupply;

    /// @param _owner The address from which the balance will be retrieved
    /// @return The balance
    function balanceOf(address _owner) public constant returns (uint balance);

    /// @notice send `_value` token to `_to` from `msg.sender`
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transfer(address _to, uint _value) public returns (bool success);

    /// @notice send `_value` token to `_to` from `_from` on the condition it is␣
↪approved by `_from`
    /// @param _from The address of the sender
    /// @param _to The address of the recipient
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function transferFrom(address _from, address _to, uint _value) public returns␣
↪(bool success);

    ///if you want to use privacy transaction,you need to implement this function␣
↪in your contract
    /// @notice send `_value` token to `_to` from `msg.sender`
    /// @param _to The address of the recipient
    /// @param _toKey the ota pubkey
```

(continues on next page)

(continued from previous page)

```
    /// @param _value The amount of token to be transferred
    /// @return Whether the transfer was successful or not
    function otatransfer(address _to, bytes _toKey, uint256 _value) public returns␣
↪(string);

    ///check privacy transaction
    /// @param _owner The address from which the ota balance will be retrieved
    /// @return The balance
    function otabalanceOf(address _owner) public constant returns (uint256␣
↪balance);

    /// @notice `msg.sender` approves `_spender` to spend `_value` tokens
    /// @param _spender The address of the account able to transfer the tokens
    /// @param _value The amount of tokens to be approved for transfer
    /// @return Whether the approval was successful or not
    function approve(address _spender, uint _value) public returns (bool success);

    /// @param _owner The address of the account owning tokens
    /// @param _spender The address of the account able to transfer the tokens
    /// @return Amount of remaining tokens allowed to spent
    function allowance(address _owner, address _spender) public constant returns␣
↪(uint remaining);

    event Transfer(address indexed _from, address indexed _to, uint _value);
    event Approval(address indexed _owner, address indexed _spender, uint _value);
}

//the contract implements ERC20Protocol interface with privacy transaction
contract StandardToken is ERC20Protocol {

    using SafeMath for uint;
    string public constant name = "WanToken-Beta";
    string public constant symbol = "WanToken";
    uint public constant decimals = 18;

    function transfer(address _to, uint _value) public returns (bool success) {

        if (balances[msg.sender] >= _value) {
            balances[msg.sender] -= _value;
            balances[_to] += _value;
            Transfer(msg.sender, _to, _value);
            return true;
        } else { return false; }
    }

    function transferFrom(address _from, address _to, uint _value) public returns␣
↪(bool success) {

        if (balances[_from] >= _value && allowed[_from][msg.sender] >= _value) {
            balances[_to] += _value;
            balances[_from] -= _value;
            allowed[_from][msg.sender] -= _value;
            Transfer(_from, _to, _value);
            return true;
        } else { return false; }
    }

    function balanceOf(address _owner) public constant returns (uint balance) {
        return balances[_owner];
    }
```

(continues on next page)

(continued from previous page)

```
    function approve(address _spender, uint _value) public returns (bool success) {

        assert((_value == 0) || (allowed[msg.sender][_spender] == 0));

        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true;
    }

    function allowance(address _owner, address _spender) public constant returns␣
↪(uint remaining) {
    return allowed[_owner][_spender];
    }

    mapping (address => uint) balances; mapping (address => mapping (address =>␣
↪uint)) allowed;
    // privacy balance, bytes for public key
    mapping (address => uint256) public privacyBalance;
    mapping (address => bytes) public otaKey;

    //this only for initialize, only for test to mint token to one wan address
    function initPrivacyAsset(address initialBase, bytes baseKeyBytes, uint256␣
↪value) public {
        privacyBalance[initialBase] = value;
        otaKey[initialBase] = baseKeyBytes;
    }

    // return string just for debug
    function otatransfer(address _to, bytes _toKey, uint256 _value) public returns␣
↪(string) {
        if(privacyBalance[msg.sender] < _value) return "sender token too low";

        privacyBalance[msg.sender] -= _value;
        privacyBalance[_to] += _value;
        otaKey[_to] = _toKey;
        return "success";
    }

    //check privacy balance
    function otabalanceOf(address _owner) public view returns (uint256 balance) {
        return privacyBalance[_owner];
    }
}
```

Note:

- Privacy transaction function is "otatransfer" in the ERC20 Protocol, the contract with privacy transaction need to implement ERC20 Protocol
- Privacy balance is stored in the map privacyBalance, function otabalanceOf can get this balance

1.7.1.2 How To Compile And Deploy

Requirement:

1. A working Wanchain client, go to the github site: go-wanchain to get the latest version
2. Remix which is an amazing online smart contract development IDE
3. your awesome Dapp consists of one or multiple smart contracts

Steps:

1. go to remix, copy and paste your smart contract code, make static syntax analysis, and compile it
2. click Details on the right panel of remix, copy all the code of WEB3DEPLOY section from the pop-up
3. copy the script and run it in gwan console

```
var erc20simple_contract = web3.eth.contract([
    {
        "constant": true,
        "inputs": [],
        "name": "name",
        "outputs": [
            {
                "name": "",
                "type": "string"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "_spender",
                "type": "address"
            },
            {
                "name": "_value",
                "type": "uint256"
            }
        ],
        "name": "approve",
        "outputs": [
            {
                "name": "success",
                "type": "bool"
            }
        ],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "totalSupply",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
```

(continues on next page)

(continued from previous page)

```
        "constant": false,
        "inputs": [
            {
                "name": "_to",
                "type": "address"
            },
            {
                "name": "_toKey",
                "type": "bytes"
            },
            {
                "name": "_value",
                "type": "uint256"
            }
        ],
        "name": "otatransfer",
        "outputs": [
            {
                "name": "",
                "type": "string"
            }
        ],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "_from",
                "type": "address"
            },
            {
                "name": "_to",
                "type": "address"
            },
            {
                "name": "_value",
                "type": "uint256"
            }
        ],
        "name": "transferFrom",
        "outputs": [
            {
                "name": "success",
                "type": "bool"
            }
        ],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "decimals",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
```

(continues on next page)

(continued from previous page)

```
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "name": "privacyBalance",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "_owner",
                "type": "address"
            }
        ],
        "name": "balanceOf",
        "outputs": [
            {
                "name": "balance",
                "type": "uint256"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "symbol",
        "outputs": [
            {
                "name": "",
                "type": "string"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
```

(continues on next page)

(continued from previous page)

```
            {
                "name": "initialBase",
                "type": "address"
            },
            {
                "name": "baseKeyBytes",
                "type": "bytes"
            },
            {
                "name": "value",
                "type": "uint256"
            }
        ],
        "name": "initPrivacyAsset",
        "outputs": [],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "_to",
                "type": "address"
            },
            {
                "name": "_value",
                "type": "uint256"
            }
        ],
        "name": "transfer",
        "outputs": [
            {
                "name": "success",
                "type": "bool"
            }
        ],
        "payable": false,
        "stateMutability": "nonpayable",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "_owner",
                "type": "address"
            }
        ],
        "name": "otabalanceOf",
        "outputs": [
            {
                "name": "balance",
                "type": "uint256"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
```

(continues on next page)

(continued from previous page)

```
    {
        "constant": true,
        "inputs": [
            {
                "name": "_owner",
                "type": "address"
            },
            {
                "name": "_spender",
                "type": "address"
            }
        ],
        "name": "allowance",
        "outputs": [
            {
                "name": "remaining",
                "type": "uint256"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "name": "otaKey",
        "outputs": [
            {
                "name": "",
                "type": "bytes"
            }
        ],
        "payable": false,
        "stateMutability": "view",
        "type": "function"
    },
    {
        "anonymous": false,
        "inputs": [
            {
                "indexed": true,
                "name": "_from",
                "type": "address"
            },
            {
                "indexed": true,
                "name": "_to",
                "type": "address"
            },
            {
                "indexed": false,
                "name": "_value",
                "type": "uint256"
            }
        ],
```

(continues on next page)

(continued from previous page)

```
        "name": "Transfer",
        "type": "event"
    },
    {
        "anonymous": false,
        "inputs": [
            {
                "indexed": true,
                "name": "_owner",
                "type": "address"
            },
            {
                "indexed": true,
                "name": "_spender",
                "type": "address"
            },
            {
                "indexed": false,
                "name": "_value",
                "type": "uint256"
            }
        ],
        "name": "Approval",
        "type": "event"
    }
])
```

```
var erc20simple = erc20simple_contract.new(
    {
        from: web3.eth.accounts[1],
        data: '0x6060604052341561000f57600080fd5b6111e38061001e6000396000f300606060405260043
        gas: '4700000'
    },
    function (e, contract) {
        console.log(e, contract);
        if (typeof contract.address !== 'undefined') {
            console.log('Contract mined! address: ' + contract.address +
↪' transactionHash: ' + contract.transactionHash);
        }
    }
)
```

4. the transaction id and contract address (hash values starting with '0x') will be printed out onto the console after few seconds
5. now, you can play with your Dapp

Note: You can locate a demo WANCHAIN token contract and involved scripts under contracts/demo/ directory

1.7.1.3 How To Invoke Privacy Transfer

After deployed above token contract on WANCHAIN,in the WANCHAIN console,you can invoke token privacy transaction according to following process:

Suppose there are at lease 3 accounts in your WANCHAIN node

1. Define asset function and variable

```
var initPriBalance = 10000;
var priTranValue = 888;
var wanBalance = function (addr) {
    return web3.fromWin(web3.eth.getBalance(addr));
}
var wanUnlock = function (addr) {
    return personal.unlockAccount(addr, "wanglu", 99999);
}
var sendWanFromUnlock = function (From, To, V) {
    eth.sendTransaction({ from: From, to: To, value: web3.toWin(V) });
}
var wait = function (conditionFunc) {
    var loopLimit = 130;
    var loopTimes = 0;
    while (!conditionFunc()) {
        admin.sleep(2);
        loopTimes++;
        if (loopTimes >= loopLimit) {
            throw Error("wait timeout! conditionFunc:" + conditionFunc)
        }
    }
}
wanUnlock(eth.accounts[1])
wanUnlock(eth.accounts[2])
stampBalance = 0.09;
```

2. buy stamp for token privacy transaction

```
abiDefStamp = [{ "constant": false, "type": "function", "stateMutability
↪": "nonpayable", "inputs": [{ "name": "OtaAddr", "type": "string" }, {
↪"name": "Value", "type": "uint256" }], "name": "buyStamp", "outputs": [
↪{ "name": "OtaAddr", "type": "string" }, { "name": "Value", "type":
↪"uint256" }] }, { "constant": false, "type": "function", "inputs": [{
↪"name": "RingSignedData", "type": "string" }, { "name": "Value", "type
↪": "uint256" }], "name": "refundCoin", "outputs": [{ "name":
↪"RingSignedData", "type": "string" }, { "name": "Value", "type":
↪"uint256" }] }, { "constant": false, "type": "function",
↪"stateMutability": "nonpayable", "inputs": [], "name": "getCoins",
↪"outputs": [{ "name": "Value", "type": "uint256" }] }];
contractDef = eth.contract(abiDefStamp);
stampContractAddr = "0x0000000000000000000000000000000000000c8";
stampContract = contractDef.at(stampContractAddr);
var wanAddr = wan.getWanAddress(eth.accounts[1]);
var otaAddrStamp = wan.generateOneTimeAddress(wanAddr);
txBuyData = stampContract.buyStamp.getData(otaAddrStamp, web3.
↪toWin(stampBalance));
sendTx = eth.sendTransaction({ from: eth.accounts[1], to:␣
↪stampContractAddr, value: web3.toWin(stampBalance), data: txBuyData,␣
↪gas: 1000000 });
wait(function () { return eth.getTransaction(sendTx).blockNumber != null;␣
↪});

keyPairs = wan.computeOTAPPKeys(eth.accounts[1], otaAddrStamp).split('+');
privateKeyStamp = keyPairs[0];
```

3. get stamp mix set for ring sign

```
var mixStampAddresses = wan.getOTAMixSet(otaAddrStamp, 2);
var mixSetWith0x = []
for (i = 0; i < mixStampAddresses.length; i++) {
    mixSetWith0x.push(mixStampAddresses[i])
}
```

4. define token contract ABI

```
var erc20simple_contract = web3.eth.contract([{ "constant": true, "inputs
↪": [], "name": "name", "outputs": [{ "name": "", "type": "string" }],
↪"payable": false, "stateMutability": "view", "type": "function" }, {
↪"constant": false, "inputs": [{ "name": "_spender", "type": "address" },
↪ { "name": "_value", "type": "uint256" }], "name": "approve", "outputs
↪": [{ "name": "success", "type": "bool" }], "payable": false,
↪"stateMutability": "nonpayable", "type": "function" }, { "constant":␣
↪true, "inputs": [], "name": "totalSupply", "outputs": [{ "name": "",
↪"type": "uint256" }], "payable": false, "stateMutability": "view", "type
↪": "function" }, { "constant": false, "inputs": [{ "name": "_to", "type
↪": "address" }, { "name": "_toKey", "type": "bytes" }, { "name": "_value
↪", "type": "uint256" }], "name": "otatransfer", "outputs": [{ "name": "
↪", "type": "string" }], "payable": false, "stateMutability": "nonpayable
↪", "type": "function" }, { "constant": false, "inputs": [{ "name": "_
↪from", "type": "address" }, { "name": "_to", "type": "address" }, {
↪"name": "_value", "type": "uint256" }], "name": "transferFrom", "outputs
↪": [{ "name": "success", "type": "bool" }], "payable": false,
↪"stateMutability": "nonpayable", "type": "function" }, { "constant":␣
↪true, "inputs": [], "name": "decimals", "outputs": [{ "name": "", "type
↪": "uint256" }], "payable": false, "stateMutability": "view", "type":
↪"function" }, { "constant": true, "inputs": [{ "name": "", "type":
↪"address" }], "name": "privacyBalance", "outputs": [{ "name": "", "type
↪": "uint256" }], "payable": false, "stateMutability": "view", "type":
↪"function" }, { "constant": true, "inputs": [{ "name": "_owner", "type
↪": "address" }], "name": "balanceOf", "outputs": [{ "name": "balance",
↪"type": "uint256" }], "payable": false, "stateMutability": "view", "type
↪": "function" }, { "constant": true, "inputs": [], "name": "symbol",
↪"outputs": [{ "name": "", "type": "string" }], "payable": false,
↪"stateMutability": "view", "type": "function" }, { "constant": false,
↪"inputs": [{ "name": "initialBase", "type": "address" }, { "name":
↪"baseKeyBytes", "type": "bytes" }, { "name": "value", "type": "uint256"␣
↪}], "name": "initPrivacyAsset", "outputs": [], "payable": false,
↪"stateMutability": "nonpayable", "type": "function" }, { "constant":␣
↪false, "inputs": [{ "name": "_to", "type": "address" }, { "name": "_
↪value", "type": "uint256" }], "name": "transfer", "outputs": [{ "name":
↪"success", "type": "bool" }], "payable": false, "stateMutability":
↪"nonpayable", "type": "function" }, { "constant": true, "inputs": [{
↪"name": "_owner", "type": "address" }], "name": "otabalanceOf", "outputs
↪": [{ "name": "balance", "type": "uint256" }], "payable": false,
↪"stateMutability": "view", "type": "function" }, { "constant": true,
↪"inputs": [{ "name": "_owner", "type": "address" }, { "name": "_spender
↪", "type": "address" }], "name": "allowance", "outputs": [{ "name":
↪"remaining", "type": "uint256" }], "payable": false, "stateMutability":
↪"view", "type": "function" }, { "constant": true, "inputs": [{ "name": "
↪", "type": "address" }], "name": "otaKey", "outputs": [{ "name": "",
↪"type": "bytes" }], "payable": false, "stateMutability": "view", "type
↪": "function" }, { "anonymous": false, "inputs": [{ "indexed": true,
↪"name": "_from", "type": "address" }, { "indexed": true, "name": "_to",
↪"type": "address" }, { "indexed": false, "name": "_value", "type":
↪"uint256" }], "name": "Transfer", "type": "event" }, { "anonymous":␣
↪false, "inputs": [{ "indexed": true, "name": "_owner", "type": "address
↪" }, { "indexed": true, "name": "_spender", "type": "address" }, {
↪"indexed": false, "name": "_value", "type": "uint256" }], "name":
↪"Approval", "type": "event" }]);

contractAddr = '0xa2e526a3632d225f15aa0592e00bed31a48c953d';
// this address should changed according to your contract deploy
erc20simple = erc20simple_contract.at(contractAddr)
```

5. create one time address for account1

```
var wanAddr = wan.getWanAddress(eth.accounts[1]);
var otaAddrTokenHolder = wan.generateOneTimeAddress(wanAddr);
keyPairs = wan.computeOTAPPKeys(eth.accounts[1], otaAddrTokenHolder).
↪split('+');
privateKeyTokenHolder = keyPairs[0];
addrTokenHolder = keyPairs[2];
sendTx = erc20simple.initPrivacyAsset.sendTransaction(addrTokenHolder,␣
↪otaAddrTokenHolder, '0x' + initPriBalance.toString(16), { from: eth.
↪accounts[1], gas: 1000000 });
wait(function () { return eth.getTransaction(sendTx).blockNumber != null;␣
↪});

ota1Balance = erc20simple.privacyBalance(addrTokenHolder)
if (ota1Balance != initPriBalance) {
    throw Error('ota1 balance wrong! balance:' + ota1Balance + ', except:
↪' + initPriBalance)
}
```

6. generate ring sign data

```
var hashMsg = addrTokenHolder
var ringSignData = personal.genRingSignData(hashMsg, privateKeyStamp,␣
↪mixSetWith0x.join("+"))
```

7. create one time address for account2

```
var wanAddr = wan.getWanAddress(eth.accounts[2]);
var otaAddr4Account2 = wan.generateOneTimeAddress(wanAddr);
keyPairs = wan.computeOTAPPKeys(eth.accounts[2], otaAddr4Account2).split(
↪'+');
privateKeyOtaAcc2 = keyPairs[0];
addrOTAAcc2 = keyPairs[2];
```

8. generate token privacy transfer data

```
cxtInterfaceCallData = erc20simple.otatransfer.getData(addrOTAAcc2,␣
↪otaAddr4Account2, priTranValue);
```

9. generate call token privacy transfer data

```
glueContractDef = eth.contract([{ "constant": false, "type": "function",
↪"inputs": [{ "name": "RingSignedData", "type": "string" }, { "name":
↪"CxtCallParams", "type": "bytes" }], "name": "combine", "outputs": [{
↪"name": "RingSignedData", "type": "string" }, { "name": "CxtCallParams",
↪ "type": "bytes" }] }]);
glueContract = glueContractDef.at(
↪"0x0000000000000000000000000000000000000000")
combinedData = glueContract.combine.getData(ringSignData,␣
↪cxtInterfaceCallData)
```

10. send privacy transaction

```
sendTx = personal.sendPrivacyCxtTransaction({from:addrTokenHolder,␣
↪to:contractAddr, value:0, data: combinedData, gasprice:'0x' +␣
↪(200000000000).toString(16)}, privateKeyTokenHolder)
wait(function(){return eth.getTransaction(sendTx).blockNumber != null;});
```

11. check balance

```
ota2Balance = erc20simple.privacyBalance(addrOTAAcc2);
if (ota2Balance != priTranValue) {
throw Error("ota2 balance wrong. balance:" + ota2Balance + ", expect:" +␣
↪priTranValue);
```

(continues on next page)

(continued from previous page)

```
}
ota1Balance = erc20simple.privacyBalance(addrTokenHolder)
if (ota1Balance != initPriBalance - priTranValue) {
throw Error("ota2 balance wrong. balance:" + ota1Balance + ", expect:" +␣
↪(initPriBalance - priTranValue));
}
```

1.7.2 Developer Tools

1.7.2.1 Using truffle to deploy Smart Contracts (v0.6)

1. Install truffle

Truffle is a command line tool to compile, deploy, and test smart contracts on blockchain. Before installing truffle, you want to install or upgrade the npm to the latest version. To install npm, use the following command:

```
$ sudo apt-get install npm
```

To upgrade npm, use the following command

```
$ sudo npm i -g npm
```

Run npm to install truffle

```
$ sudo npm install -g truffle
```

Note: Here -g is for global install so that truffle can be run from any directory

2. Use truffle tool

Once truffle is installed, you can type truffle to see a list of commands and options.

Usage: truffle <command> [options]

```
Commands:

    init       Initialize new and empty Ethereum project
    compile    Compile contract source files
    migrate    Run migrations to deploy contracts
    deploy     (alias for migrate)
    build      Execute build pipeline (if configuration present)
    test       Run JavaScript and Solidity tests
    debug      Interactively debug any transaction on the blockchain (experimental)
    opcode     Print the compiled opcodes for a given contract
    console    Run a console with contract abstractions and commands available
    develop    Open a console with a local development blockchain
    create     Helper to create new contracts, migrations and tests
    install    Install a package from the Ethereum Package Registry
    publish    Publish a package to the Ethereum Package Registry
    networks   Show addresses for deployed contracts on each network
    watch      Watch filesystem for changes and rebuild the project automatically
    serve      Serve the build directory on localhost and watch for changes
    exec       Execute a JS module within this Truffle environment
    unbox      Download a Truffle Box, a pre-built Truffle project
    version    Show version number and exit
```

See more at http://truffleframework.com/docs

3. Initiate, compile, and deploy a truffle project for Wanchain

Make a truffle directory

```
$ mkdir  wanchain-example
```

Initialize the truffle project

3.1 Initiate

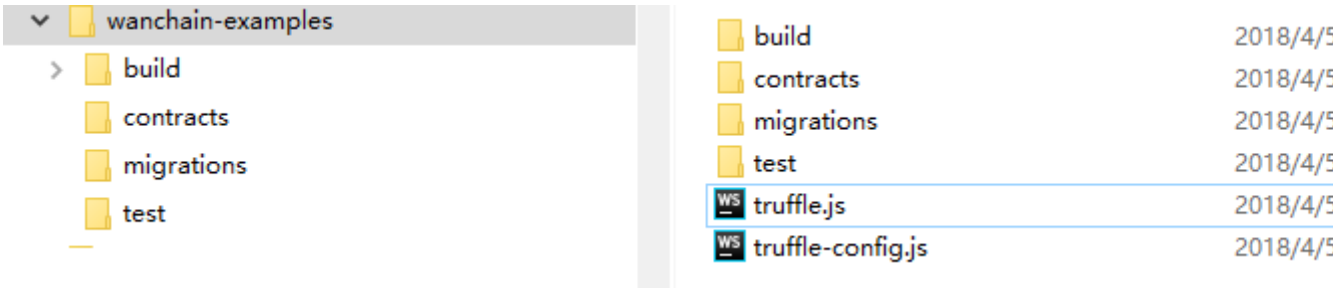
In the truffle project directory,execute:

```
$ truffle init
```

Once a truffle project is initialized, several directories will be created, including

- **contracts** : where the source contracts are supposed to reside
- **migrations** : where the deployment scripts are supposed to reside
- **test** : where the test files are supposed to reside
- **build** : the contract compiled result will be put here, created after running truffle compile or truffle migrate.
- **truffle-config.js** : The configuration file provides default setup parameters for the project
- **truffle.js** : This provide truffle smart contract parameters

The directory will be like below:



3.2 compile

In the truffle project directory, execute the command:

```
$ truffle compile
```

If the compilation is successful, a build directory will be created and abi files will be generated under **build/contracts/** directory.These abi files can then be deployed as smart contract.

3.3 Config

Setup truffle.js file to specify the network setting etc.

```
module.exports = {
    networks: {
        development: {
            host: 'localhost',
            port: 8545,
            network_id: '*',
            gas: 4000000,
            gasPrice: 180e9,
            // following address needed to be replaced with unlocked␣
↪account on gwan node
            from: '0x8f84573C8BaB4d56FDdB48cc792424E8816908fB'
        }
    }
}
```

Add deploy script for contract in the directory migrations in the truffle project:

Such as the deploy script name is 2_deploy_contracts.js which will deploy the contract PollApp.sol in the contract directory, the script will be as following:

```
var PollApp = artifacts.require("./PollApp.sol");
module.exports = function(deployer) {
    deployer.deploy(PollApp);
};
```

3.4 start gwan node on local host

Run following command in the directory which include gwan

```
$ ./gwan --rpc --testnet --rpcapi eth,net,admin,personal,wan --
↪verbosity=0 console
```

In the gwan console to unlock a existing wanchain account in gwan node and make sure there are balance in the unlocked account, this account need to be same with the from address in the file truffle.js

3.5 deploy contracts

Execute command in the truffle project directory:

```
$ truffle migrate --network development
```

The result will be as following:

```
Using network 'development'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0xde8e497406fc400274080b554f2892ab37669b824e14724286fd7c7b00be4a29
  Migrations: 0x0ae74fd3bc469a8e1a2eac981ee4016f0859e770
Saving successful migration to network...
  ... 0xad01cea5acb65cbcb28610036c8202c42727e8e9853ef2b9edae832ddef07993
Saving artifacts...
Running migration: 2_deploy_contracts.js
Wed, 03 Oct 2018 16:00:00 GMT
1538582400
  Replacing Migrations...
  ... 0x8a089b31fed57997f0c97fd1acb4fc49d4693d3cb2e9f009d172069f1a6cf72d
  Migrations: 0x2e762184f6c72a10d4828ce578587a455e472740
Saving successful migration to network...
  ... 0x5c0c784f9ef2d4dea644c34aa4fe271f8d5b56c598c9937e517d8dea44df1168
Saving artifacts...
```

4. Test deployed smart contract

4.1 test script

Truffle support test for the contract, the test script should put in the directory “test” in the truffle,the test script maybe look like this:

```
var solc = require('solc');
var Web3 = require('web3');

const PollApp = artifacts.require('./PollApp.sol');

contract('PollApp', ([owner]) => {
    let PollAppInstance;
```

(continues on next page)

(continued from previous page)

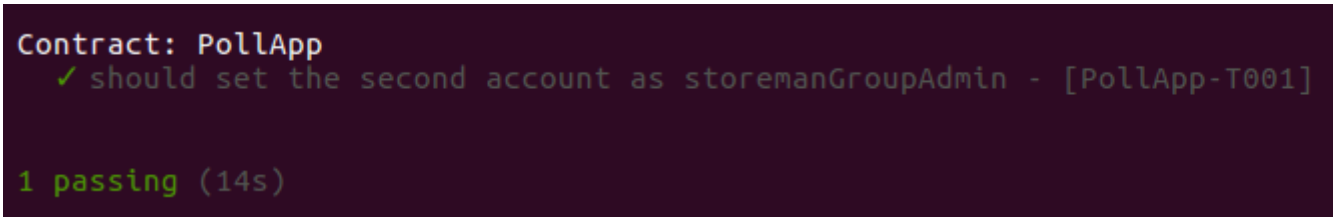
```
    before('set up contract before test', async () => {
        await web3.personal.unlockAccount(owner, '****', 9999);

        //create instance
        PollAppInstance = await PollApp.new({from: owner});
    })

    it('test case - 001', async () => {
        await PollAppInstance.XXX()({ from: owner });
        assert.equal(..., ..., 'description...');
    })
})
```

4.2 start test

In the directory of the truffle project, execute the command:

```
$ truffle test
or specify to execute one test script
$ truffle test ./test/XXX.js
```

This command will execute the test script in the directory "test" of truffle project. The result may look like this:



1.8 Hackathon Resources

1.8.1 Hackathon handbook

1.8.2 Terms and condition

1.9 Wallet Support

1.9.1 Wanwallet 2.0 Cross-chain transactions

1.9.2 Wanchain Ledger Wallet Overview

1.9.3 Wanchain Trezor Wallet Overview