MANX State Channel Technology

Topic 1: State channel-based privacy protection scheme

The transaction confirming method used by current blockchain platforms is too redundant, resulting in delays in each transaction confirmation, which is not consistent with current high frequency transaction demand. Also, each transaction consumes a service fee. In addition, the blockchain will make the transaction details public, which may have implications for privacy and confidentiality.

Topic 2: Design Concept for MANX State Channel

To solve this problem, MANX proposes a state channel for privacy protection. The state channel is separate from the contract layer and the transaction process is carried out off chain. In the end, only the transaction result is uploaded to the "root contract" on the chain for settlement. This will reduce the impact of the blockchain network's delay, reduce the processing fees incurred during the transaction process, while protecting the details of transactions.

Topic 3: MANX State Channel Concepts

Two definitions are proposed here:

(1) “on-chain-contract” is determined and deployed by all participants and is used for final settlement. The on-chain-contract will become effective after it receives the tokens from all participants. After the off-chain transactions are complete, the tokens will be transferred to each participant according to the settlement result. Then the contract will be marked completed. In addition, the process includes an arbitration contract where participants can request arbitration if they disagree with the transaction result.

(2) “off-chain-contract” is a transaction contract determined by all participants, which specifies the transaction rules and the public key addresses that are allowed to participate in the transaction. The off-chain-contract is operated and maintained by virtual machines or a private chain to ensure that the contract runs automatically and safely.

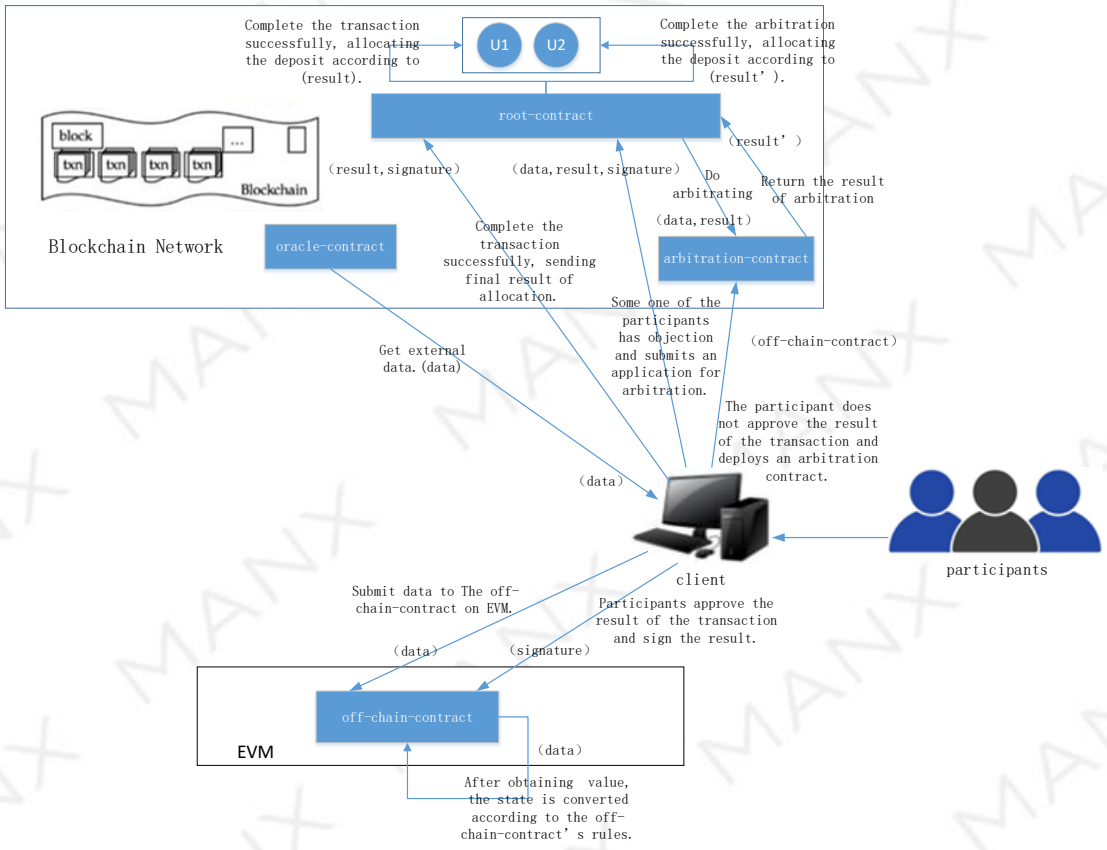
Topic 4: MANX State Channel Features

MANX's privacy status channel protocol has the following features:

- **Privacy:** The blockchain cannot see the intermediate payment or contract information during this process, except for the final settlement and dispute resolution process;

- **Speed.** Because the updates between multiple participants are almost instantaneous, the state channel solution is faster than any blockchain solution (whether public or private). It may even be faster than a centralized solution because the channel update between A and B can be implemented without a centralized server.

- **Cost Effective:** DAPP participants send messages and transactions to each other to update status, but they do not submit messages on the main chain.



MANX state channel structure

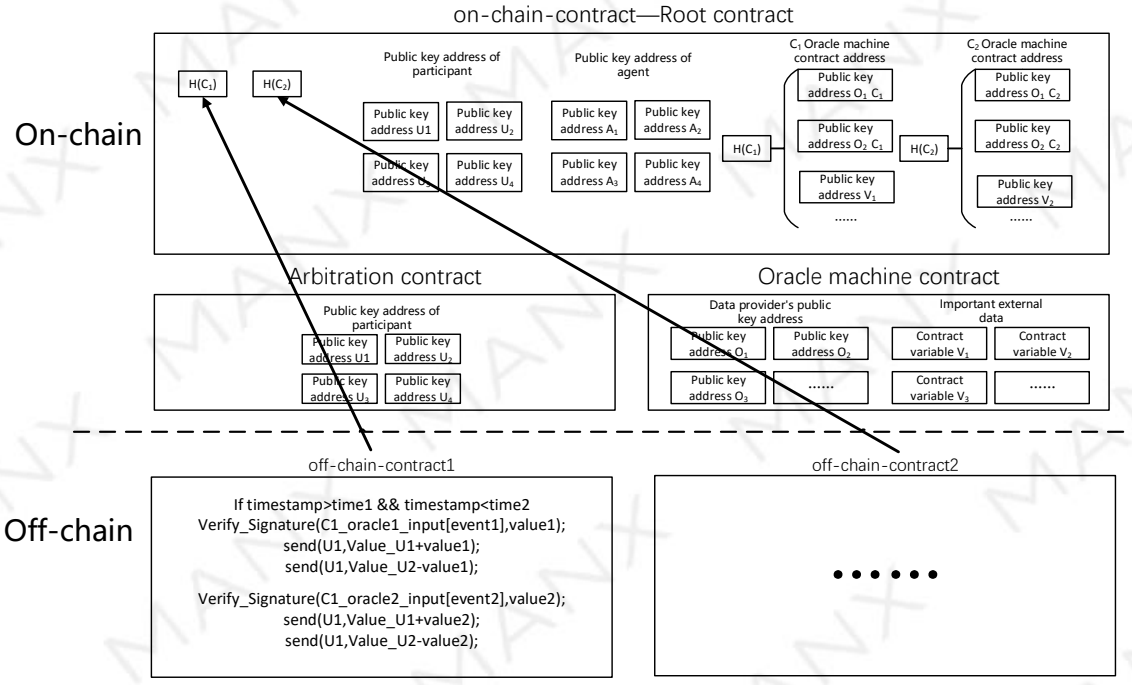
Topic 5: State Channel Life Cycle

1. Open the channel. First, at least two participants must agree on the initial state and establish both an on-chain root contract and an off-chain contract, then put tokens into the on-chain root contract as a deposit. The status channel will then be opened. The oracle is responsible for sending external data to the off-chain contract.

   MANX defines a set of root contracts for the specification and qualification of user-defined contracts, and the oracle interface root contract is one of the root contracts. Any contract that wishes to call external data outside MANX must inherit the oracle interface root contract. The oracle interface root contract contains four parts：

   - Data that is obtained outside MANX;
   - Data provider accounts that pass data into MANX;
   - Data user accounts that call data in MANX;
   - Rules such as data provider's reward and punishment mechanism, margin mechanism, etc.

   Because of the variety of oracles for different mechanisms, we don't care about the specific way data providers receive data as long as they implement the oracle interface root contracts to connect various types of oracles.



On-Chain-Contract

The on-chain-contract is divided into three categories: root contract, arbitration contract and oracle contract.

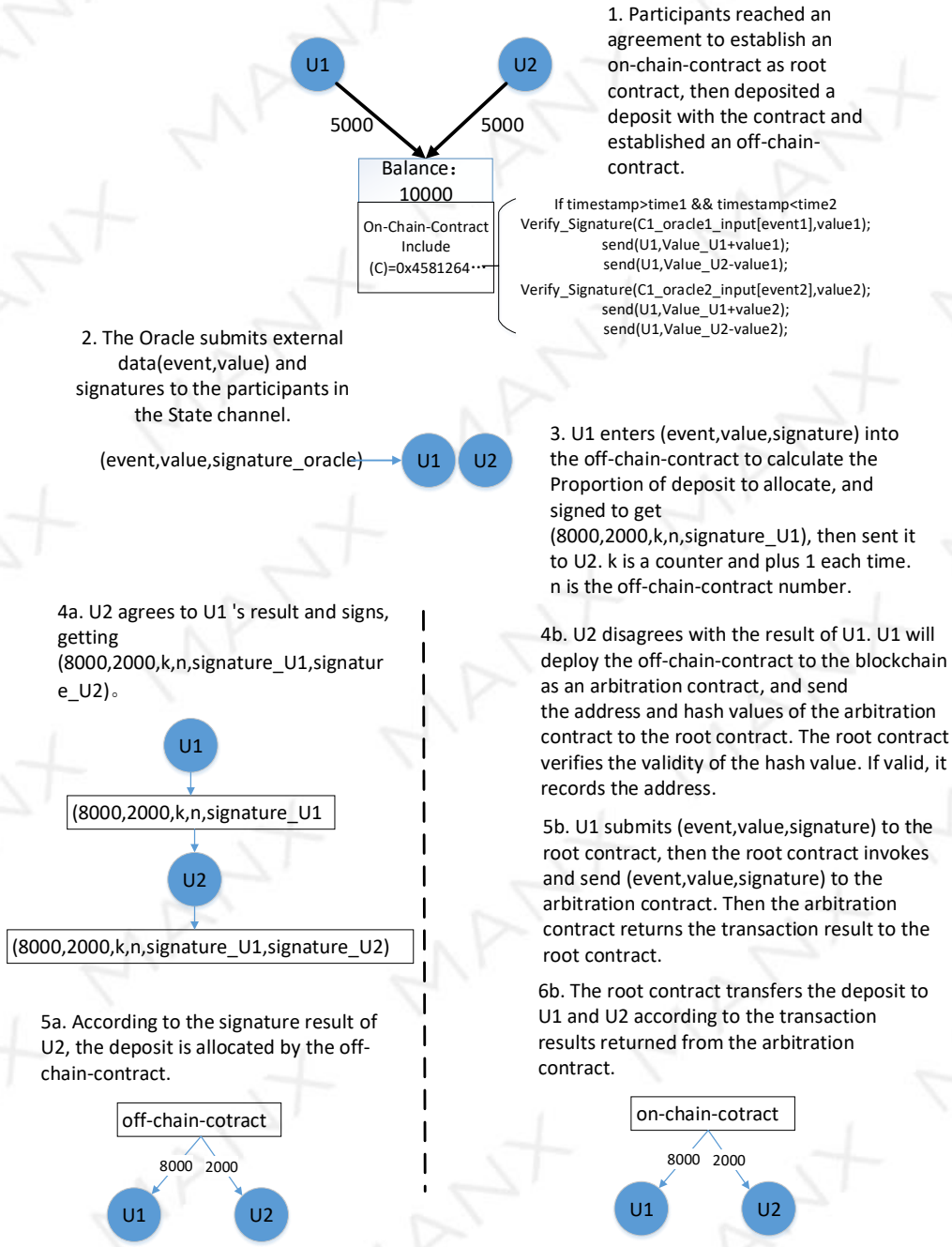
a) The main contents of the root contract are as follows: H(C_i), the digital summary obtained by hashing C_i - the public key address of the state channel participants**,** the public key address of the agent authorized by the participant and the public key address of the oracle that is trusted by the off-chain contract.
b) The main contents of the arbitration contract are as follows: The public key address of the state channel participants. Arbitration contracts are used for verification when some participants disagree with off-chain-contract transaction. Its content is exactly the same as the corresponding off-chain-contract.
c) The main contents of the oracle contract are as follows: The public key address of data providers which is a certified, trusted third-party public key and external data that is key to the successful operation of the status channel. The oracle contract provides the necessary external data for the status channel and its credibility should be recognized by all participants.

The main contents of the off-chain contract. Ci, the source code of the ith off-chain contract which contains the transaction rules, participants' public addresses and balance status.

2. Run the channel. After the establishment of the state channel, the participants operate the off-chain contract state machine according to the rules based on oracle: f (state, action) => state.

State channel working mechanism:

a) The execution flow of the off-chain contract.
b) Off-chain contract changes. Any modifications must be approved and signed by all channel participants, and the corresponding hashing summary H(C_i) in the on-chain root contract needs to be updated.
c) Adding or deleting users. The addition or deletion of participants must be approved and signed by all participants, and the public key address of the participants in the on-chain root contract needs to be updated.
d) Agent settings. In state channel mode, it can't be guaranteed that every participant stays online all the time. Therefore, each participant can designate an agent by signing and updating the corresponding agent's public key address in the on-chain root contract.



State channel working mechanism

3. Close the channel. When the status channel off-chain contract receives a valid status update from one participant, it will enter the challenge period, during which another channel participant can submit a status update with a higher serial number. After the challenge period, the valid status with the highest serial number is accepted as the final status.

4. Settlement. When any participant wants to close the transaction channel, the data will be submitted to the chain for settlement after updating to the latest state. The channel will then be closed.

   The judgment rules of whether the status is valid are as follows:

   a) Status updates must be signed by all channel participants.
   b) The serial number of each status update must be higher than the last one of the sequence.
   c) Off-chain contracts can only submit the latest status update after the channel is closed.

Topic 6: Use Case

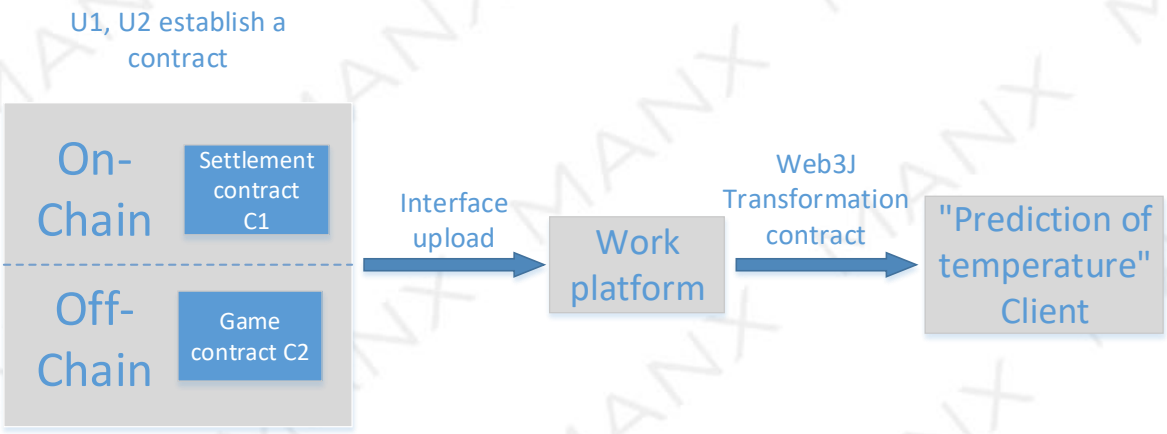
Suppose there are the participants U1, U2 who want to play a game to predict the temperature. If the highest temperature is lower than 25℃, U1 needs to pay one token to U2. Otherwise, U2 needs to pay one token to U1. The valid period is ten days. In the end, the on-chain-contract will transfer the tokens to U1 and U2 according to their account balances.

```
contract C1 {
   address U1
   address U2
   address add_C3
   mapping(address => uint) balance;
   function recharge() payable
   {balance[msg.sender] = msg.value;}
   function returnEther () payable
   {
        If(msg.sender == this){
          U1.send(balance[U1]);
          U2.send(balance[U2]);
         }
    }
   function settlement(uint balance_U1,uint balance_U2) payable
   {
          If(msg.sender == U1 || msg.sender == U2)
          {
          balance[U1] = balance_U1;
          balance[U2] = balance_U2;
          this.returnEther();
            }
    }
    function arbitration(uint A11,uint A21,uint temp)
    {
            If(msg.sender == U1 || msg.sender == U2){
          uint A12,A22;
          (A12,A22) = C2(add_C3). changeState(temp);
          settlement(A12,A22);
             }
      }
}
```

```
contract C2 {
        address U1;
        address U2;
        uint stage = 0;
        mapping(uint => mapping(address => uint)) balance;
        mapping(uint => mapping(address => uint)) signature;
        function changeState(uint temp) returns(uint balance_u1,uint balance_u2,uint temp)
        {
         if (temp <= 25) {
                stage++;
                balance[stage][U1] -= 1;
                balance[stage][U2] += 1;
         } else {
                stage++;
                balance[stage][U1] += 1;
                balance[stage][U2] -= 1;
         }
         balance_u1 = balance[stage][U1];
         balance_u2 = balance[stage][U2];
         }
        function signing(uint attitude)
       {
                signature[stage][msg.sender] = attitude;
         }
}
```

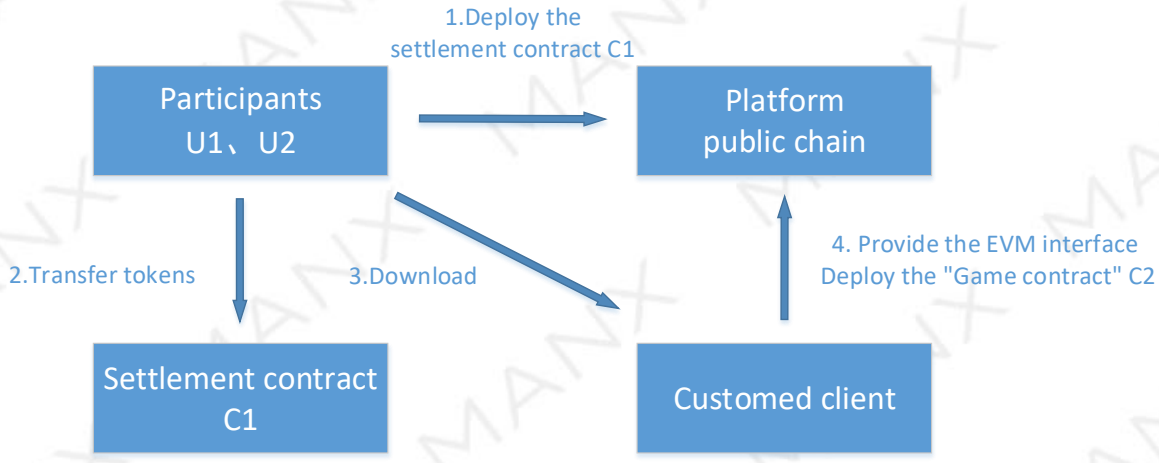
In this game, the state channel workflow is as follows:

(1) U1 and U2 jointly propose two smart contracts. The smart contract C1 deployed on the chain is called the "root contract," and the smart contract C2 running off chain is called the "game contract" (also used as arbitration contract C3). U1 or U2 then uploads the two contracts C1 and C2 through the interface provided by our platform. After the platform acquires the contract, it converts the two contracts into a Java version and integrates them into the platform Java client to create a customized "predict temperature" status channel client.



Status channel workflow (1)

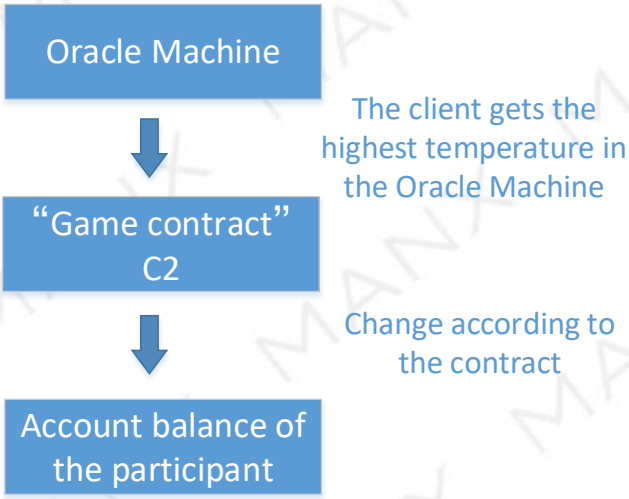
(2) U1 or U2 deploys the root contract C1 on the public chain provided by the platform, and U1 and U2 transfer the pre-agreed tokens to C1, making the contract C1 effective (controlled by the contract code). The root contract C1 has stored the hash value of "game contract" C2 in advance to examine of the legitimacy of "arbitration contract" C3. U1 and U2 then download the customized client and deploy the off-chain "game contract" C2 through the virtual machine interface. At this point, the status channel is opened.



Status channel workflow (2)

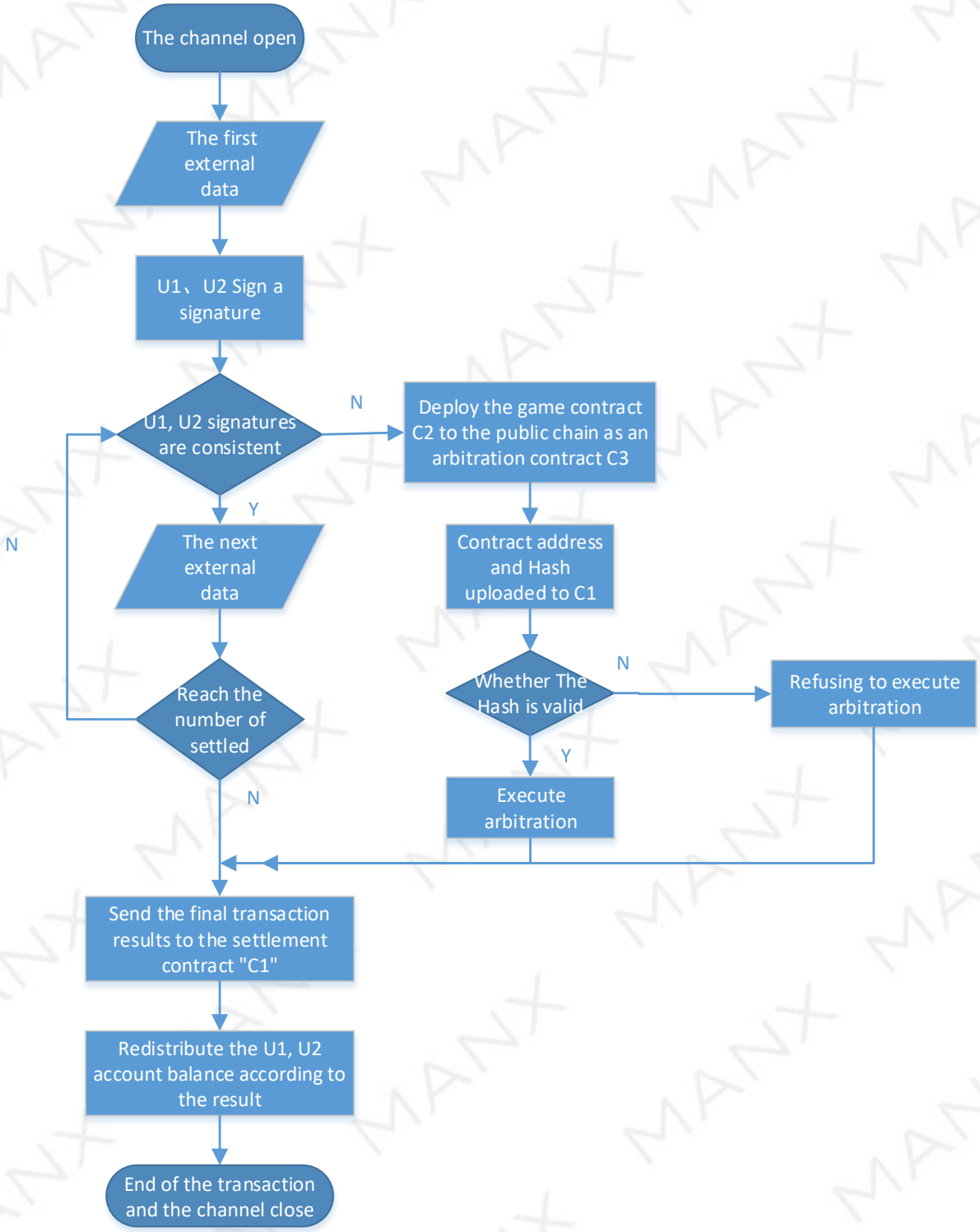
(3) "The oracle" is on-chain contract which is used to provide external data. In the preparation of C1 and C2, it is necessary to determine which on-chain contract will serve as the external data provider. In this prediction game, the oracle "weather contract" measures the highest temperature. Meanwhile, the client automatically uploads the highest temperature value from the weather contract to the game contract

C2 in the virtual machine. Contract C2 changes the account balance of U1 and U2 according to the pre-defined rules.



Status channel workflow (3)

(4) After step 3 is completed, U1 and U2 each use the private key to sign the transaction result. If they both agree with the result, the signature is approved. If they agree with the transaction results each day, the status channel will loop operations of step 3 until the contract expires. (the signature can be understood as a modification to a variable. For example, there is a variable x. If they both agree, x is modified to 1. Otherwise, it's modified to 2). U1 or U2 then sends the final transaction result to the root contract C1 (The result includes the final account balance of U1 and U2). C1 then transfers the tokens to the accounts of U1 and U2 according to the transaction results. After all these processes, the status channel is closed. If U1 disagrees with the transaction result, they can deploy the arbitration contract C3 and send the contract address and hash value to C1. C1 judges whether the hash value is valid. If it is valid, the arbitration is allowed and the arbitration result of arbitration contract C3 is accepted. Then, U1 sends the account balance values A11, A21 of previous day and the highest temperature value of current day to the "root contract" C1. C1 calls the arbitration contract C3 to obtain the actual account balance value A12 and A22, and transfer tokens to the accounts of U1 and U2. The entire transaction ends. The status channel is closed.



Status channel workflow (4)

Topic 7: Comparison of BlockChain Privacy Protection Techniques:

Project	MANX	Lightning Network/ TRINITY	Zcash	Monero
Principle:	multi-autonomous state channels	Two-node off-chain channel	Zero-knowledge Proof	ring signature
Privacy protection object	Any business	Only for micropayments	Only protect transactions	Only protect transactions
Privacy protection effect	Completely confidential (data is not on the chain)	Only for micropayments	Partially confidential (protect data on the chain)	Partially confidential (protect data on the chain)
Privacy Protection Group Policy	Support	Not support	Not support	Not support
Participating nodes number	2~infinite	2	Not support	Not support
Immediacy	High	Medium	Low	Low

Privacy Protection Capabilities: A Comparison