MANX 状态通道技术

课题 **1**：区块链隐私保护的需求：

现行的区块链平台确认交易的方式过于冗余，导致每笔交易的确认会有一定的延时，不太符合现阶段快节奏的交易方式，并且每笔交易都需要消耗一笔不菲的手续费，如果交易过程过于繁杂，则交易手续费也将增大交易成本，加之区块链会把交易的详细过程公之于众，这无法满足交易的保密性，这些是参与交易的多方都不愿看到的。

课题 **2**：**MANX** 状态通道设计概念：

MANX 提出了面向隐私保护的状态通道（State Channel），将区块链中的合约层单独剥离出来，具体的交易过程放到链下进行，最后只需要把最终的交易结果推送到链上的 “根合约” 进行结算即可。这样既能减少区块链网络的延时对繁杂的交易过程带来的影响，又能减少交易过程中产生的手续费，还能对具体的交易细节进行保密。

课题 **3**：**MANX** 状态通道定义：

MANX 在这提出两个定义：

链上合约（on-chain-contract），这包括参与的各方共同拟定并部署运行在链上的用于最终结算的合约，在收到参与的各方打入的代币后方可生效，链下交易完成后将代币按结算结果转给参与的各方，而后自动作废；另外也包括了在交易的过程中任何参与的一方对交易结果有异议而提起“仲裁申请”的仲裁合约。

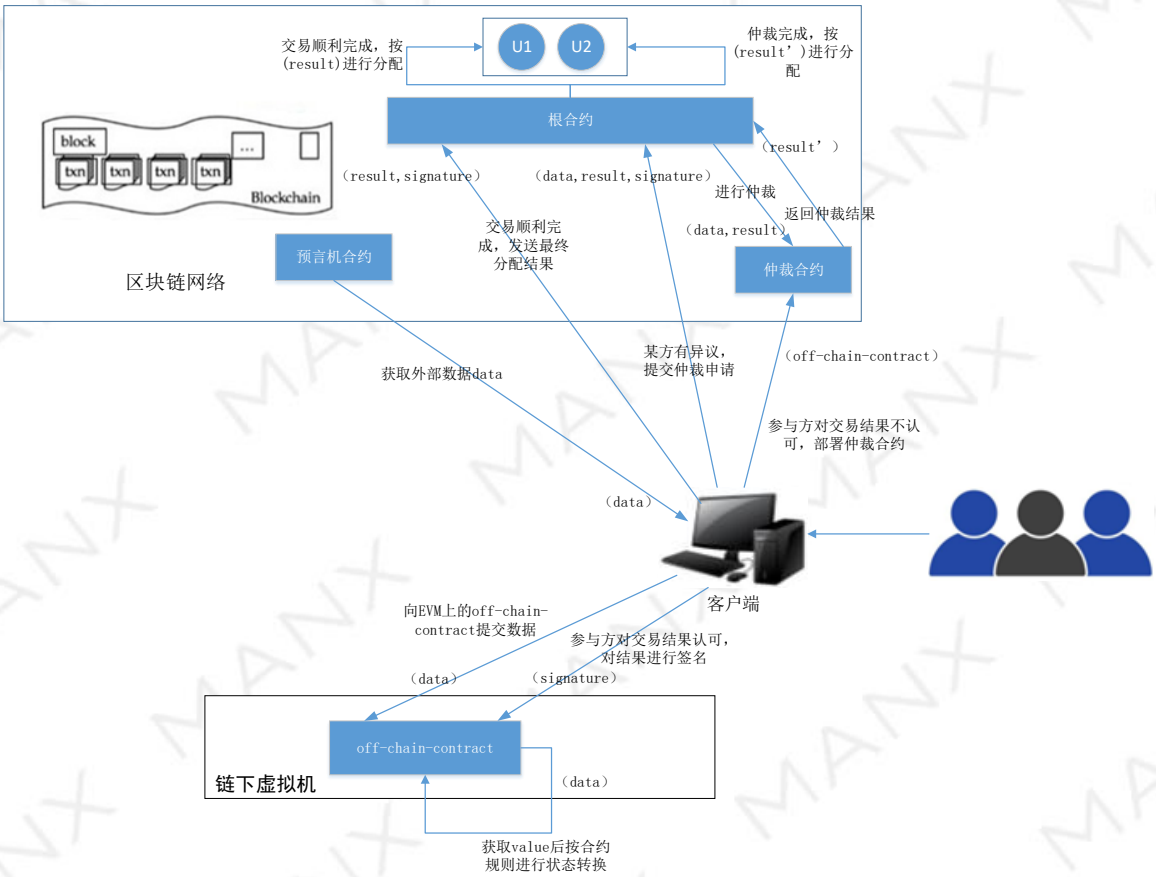
链下合约（off-chain-contract），这是参与的各方共同拟定的交易合约，其中写明了交易的规则和允许参与交易的各方的公钥地址，链下合约由链下虚拟机或者私链运行和维护，保证链下合约自动、安全地运行。

课题 **4**：**MANX** 隐私状态通道协议具备的特点：

1) 隐私性：区块链在这个过程中看不到中间的支付或合约信息,除了最终的结算和争议解决过程;

2) 即时性。因为参与多方之间的通道更新几乎是即时的，比任何区块链上的解决方案都快(无论是公有链还是私有链), 甚至可能比中心化的解决方案都快，因为 A 与 B 之间的通道更新无需中心化服务器也能实现；

3) 低成本：Dapp 的参与者将消息与事务相互发送，以更新状态，但不会将消息发布到链中。



MANX 状态通道结构

课题 **5**：状态通道的全生命周期管理

1) 打开通道：首先由至少两个参与者就初始状态达成一致，并且建立 on-chain-contract 根合约和 off-chain-contract，并向 on-chain-contract 根合约中放入一些代币进行托管，则状态通道开启。预言机 oracle 负责向 off-chain-contract 发送外部数据。

MANX 为了规范和限定用户自定义合约，创建了一组根合约，预言机接口根合约是所有根合同之一。任何希望在 MANX 外部调用数据的合约必须继承预言机接口根合约。预言机接口根合约包含四个部分：

- 在 MANX 之外获得的数据;
- 数据提供者的地址;
- 数据使用者的地址;
- 数据提供者的奖惩机制，保证金机制等规则。

由于根据不同机制和原理存在多种不同类型的预言机，事实上，MANX 并不关心数据提供者从外部获取数据的具体方式，我们将创建预言机接口根合约来连接各种类型的外部预言机。

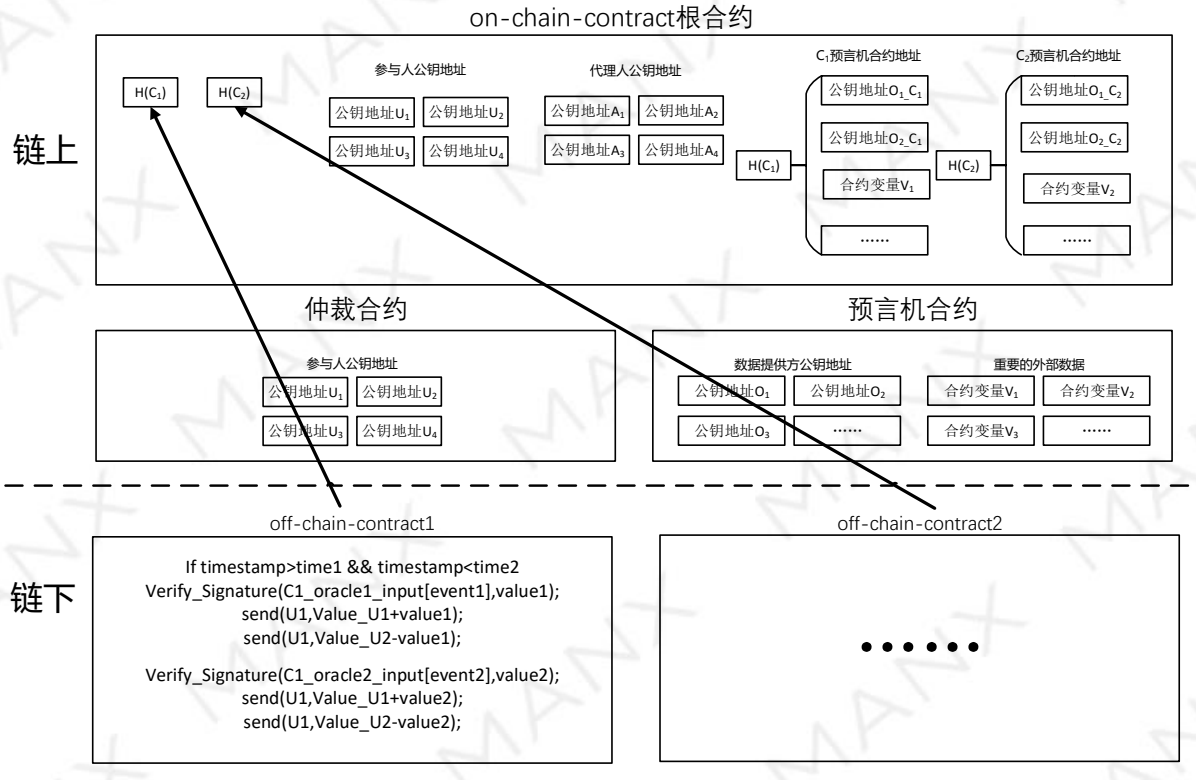


图1:On-Chain-Contract 根合约

链上合约 on-chain-contract 主要分为三类：根合约、仲裁合约和预言机合约。

(1) 根合约的主要内容如下：$H(C_i)$：对C_i做哈希运算得到的数字摘要；参与人公钥地址：状态通道参与人的公钥地址；代理人公钥地址：经参与人授权的代理人的公钥地址；预言机公钥地址：被 off-chain-contract 所信任的预言机的公钥地址。根合约用于最终结算。

(2) 仲裁合约的主要内容如下：参与人公钥地址：状态通道参与人的公钥地址。仲裁合约用于某一方在对链下交易有异议时的检验操作，其内容与对应的 off-chain-contract 完全一致。

(3) 预言机合约的主要内容如下：数据提供方公钥地址：为链下合约提供外部数据的公钥地址，其是经过认证的、可信的第三方公钥；重要的外部数据：链下合约所须的重要数据，这是状态通道能否顺利运行的关键。预言机合约为状态通道提供必要的外部数据，其可信度被参与的各方认可。

链下合约 off-chain-contract 合约的主要内容如下：C_i：合约第 i 个 off-chain-contract 的源码，包括交易规则、参与者的公钥地址和各地址的余额状态；

2) 运行通道：当状态通道建立完毕后，参与人根据预言机按照规则运行状态通道内各个 off-chain-contract 的状态机：$f(state, action) => state'$。

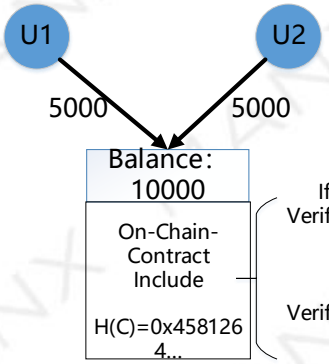
状态通道的工作机制：

(1) off-chain-contract 合约的执行流程。

(2) off-chain-contract 合约的更改。对于 off-chain-contract 的任意修改，必须全体通道参与人认可并签名，同时将 on-chain-contract（根合约）中的链下合约的哈希列表中相应的$H(C_i)$更新。

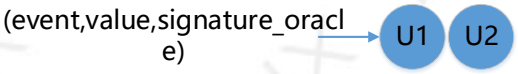
(3) off-chain-contract 合约增删用户。对于状态通道中增删参与人，必须全体通道参与人认可并签名，同时将 on-chain-contract（根合约）中的参与人公钥地址进行有效或者失效的状态更新。

(4) off-chain-contract 用户代理设置。在状态通道模式下不能保证各个参与人任意时间均保持在线，因此各参与人可以通过签名方式设置通道下的代理人，并且在 on-chain-contract（根合约）中更新对应的代理人公钥地址。



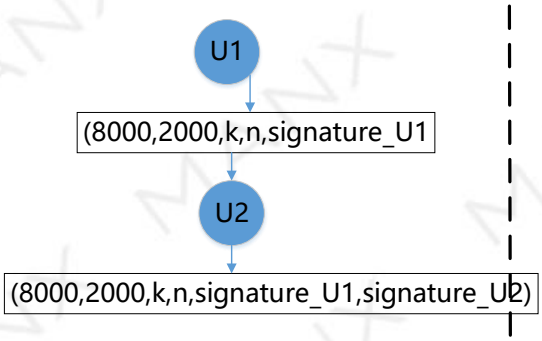
1. 参与人达成一致建立on-chain-contract根合约,并向其中存入保证金，同时建立off-chain-cotract。

2. 预言机向状态通道的参与人报送外部数据(event,value)与签名。



3. U1将(event,value,signature)输入off-chain-contract计算保证金的分配比例并签名得到(8000,2000,k,n,signature_U1)，发送给U2,k为计数器，每次加1,n为off-chain-contract的编号。

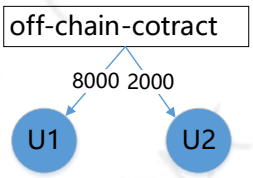
4a. U2同意U1的结果，并签名得到(8000,2000,k,n,signature_U1,signature_U2)。



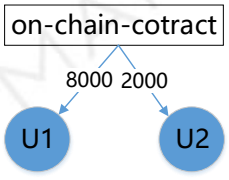
4b. U2不同意U1的结果。则U1将off-chain-contract部署至链上作为仲裁合约，并把仲裁合约的地址address和hash值提交给根合约。根合约验证hash值的有效性，如果有效则记录下address。

5b. U1将(event,value,signature)提交到根合约，根合约调用并将(event,value,signature)传入仲裁合约，仲裁合约将交易结果返回给根合约。

5a. 根据U2的签名结果，由off-chain-contract分配保证金。



6b. 根合约按照仲裁合约返回来的交易结果将保证金转出到U1和U2。



状态通道的工作机制

3) 关闭通道：当状态通道 off-chain-contract 从其中一个通道参与者接收到有效的状态更新时，它将进入挑战期，在此期间另一个通道参与者可以提交更高序列号的状态更新。在挑战周期结束之后，具有最高序列号的有效状态更新被接受为最终状态。

4) 结算：当参与方的任何一方想要关闭交易通道，则更新到最近的状态后，将数据上链进行结算，并关闭通道。

状态通道对于状态是否有效的判断依据如下：

a) 状态更新必须由所有通道参与者签署。

b) 每个连续的状态更新必须高于 sequence 最后一个。

c) off-chain-contract 在通道关闭后只能提交关闭前的最近状态更新。

课题 **6**：实施案例

假设有两个参与方 U1、U2，双方要进行一个“预测气温”的游戏，如果当天最高气温低于 25℃，则 U1 账户给 U2 转 1 个代币，反之 U2 账户给 U1 转 1 个代币，十天为止，之后链上合约按照 U1 和 U2 的账户余额将代币转出给 U1 和 U2。

```
contract C1 {        //根合约
   address U1; //U1 的公钥地址
address U2; //U2 的公钥地址
address add_C3; //“仲裁合约”C3 的地址
mapping(address => uint) balance;    // U1、U2 的余额
function recharge() payable              //U1、U2 分别向合约充值
{           balance[msg.sender] = msg.value;}
function returnEther () payable
{           //将合约中的代币转出到 U1、U2 公钥地址上，此交易发起者只能是本合约
                    If(msg.sender == this){       //判断交易发起者地址是否是本合约的地址
                              U1.send(balance[U1]);
           U2.send(balance[U2]);
                    }
}
function settlement(uint balance_U1,uint balance_U2) payable     {           //最终结算
           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)
{           //进行仲裁，仲裁会调用部署在链上的“游戏合约”C2(即“仲裁合约”C3)
           If(msg.sender == U1 || msg.sender == U2){
                    uint A12,A22;
                    (A12,A22) = C2(add_C3). changeState(temp);
                    settlement(A12,A22);
           }
}
}
```

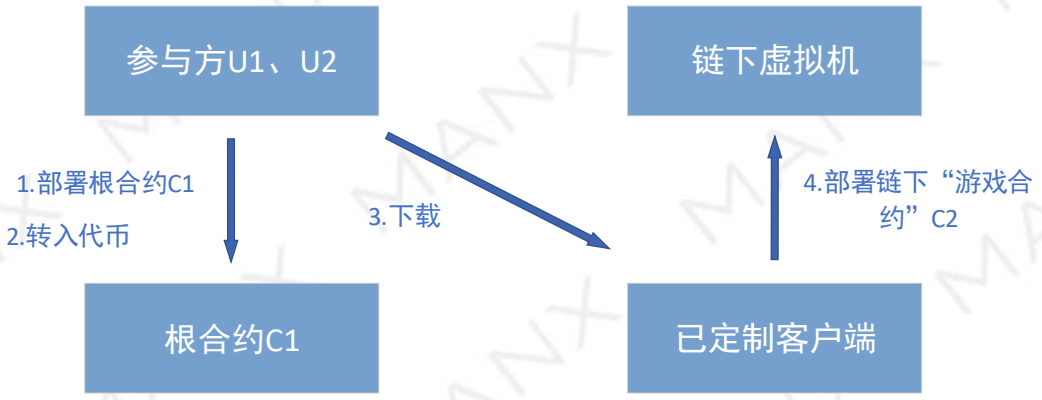
```
contract C2 {        //游戏合约
        address U1; //U1 的公钥地址
        address U2; //U2 的公钥地址
        uint stage = 0; //当前交易的阶段，在这个例子中共有十个阶段（即十天）
mapping(uint => mapping(address => uint)) balance;    //某阶段中 U1、U2 的余额
mapping(uint => mapping(address => uint)) signature; //U1、U2 对某阶段的签名
function changeState(uint temp) returns(uint balance_u1,uint balance_u2,uint temp)
{       //获取当天的最高温度，进而改变余额状态
        if (temp <= 25) {   //如果 temp<=25，则 U1 向 U2 转 1 个代币，反之同理。
                stage++;
                balance[stage][U1] -= 1;
                balance[stage][U2] += 1;
        } else {
                stage++;
                balance[stage][U1] += 1;
                balance[stage][U2] -= 1;
        }
        balance_u1 = balance[stage][U1];      //返回交易后的余额状态
        balance_u2 = balance[stage][U2];
}
function signing(uint attitude)
{       //客户对当前阶段的交易结果签名，“1”为认可，“2”为不认可
        signature[stage][msg.sender] = attitude;
        }
}
```

在此游戏的过程中，状态通道工作流程如下：

（1） U1 和 U2 共同拟定好两个智能合约，一个部署在链上的智能合约 C1 称为“根合约”，一个在链下运行的智能合约 C2 称为“游戏合约”（也作为后续用于仲裁的“仲裁合约”C3），随后 U1 或 U2 将 C1、C2 两个合约通过我们平台给定的接口上传。平台获得合约后，将两个合约转化成 Java 版的智能合约（用 web3J 可以转化），并将两个 Java 智能合约集成入已经开发好的 Java 客户端，制成特定的“预测气温”状态通道客户端；

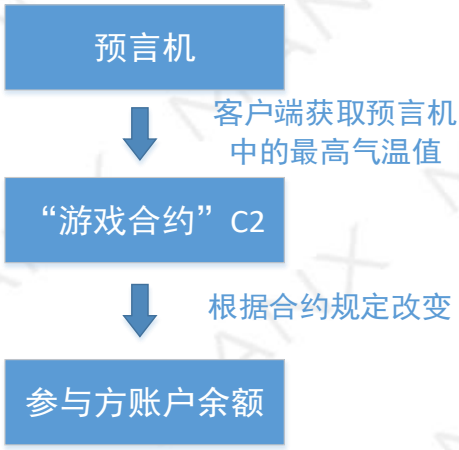
状态通道工作流程（1）

（2） U1 或 U2 将“根合约”C1 部署到平台提供的公链上，随后 U1 和 U2 都往 C1 中转入事先约定好的代币，使合约 C1“生效”（由合约代码控制），“根合约”C1 会事先存有“游戏合约”C2 的 hash 值，以备后续检验“仲裁合约”C3 的合法性。随后 U1 和 U2 下载定制的客户端，通过客户端提供的链下虚拟机接口部署链下“游戏合约”C2，至此，状态通道开启；



状态通道工作流程（2）

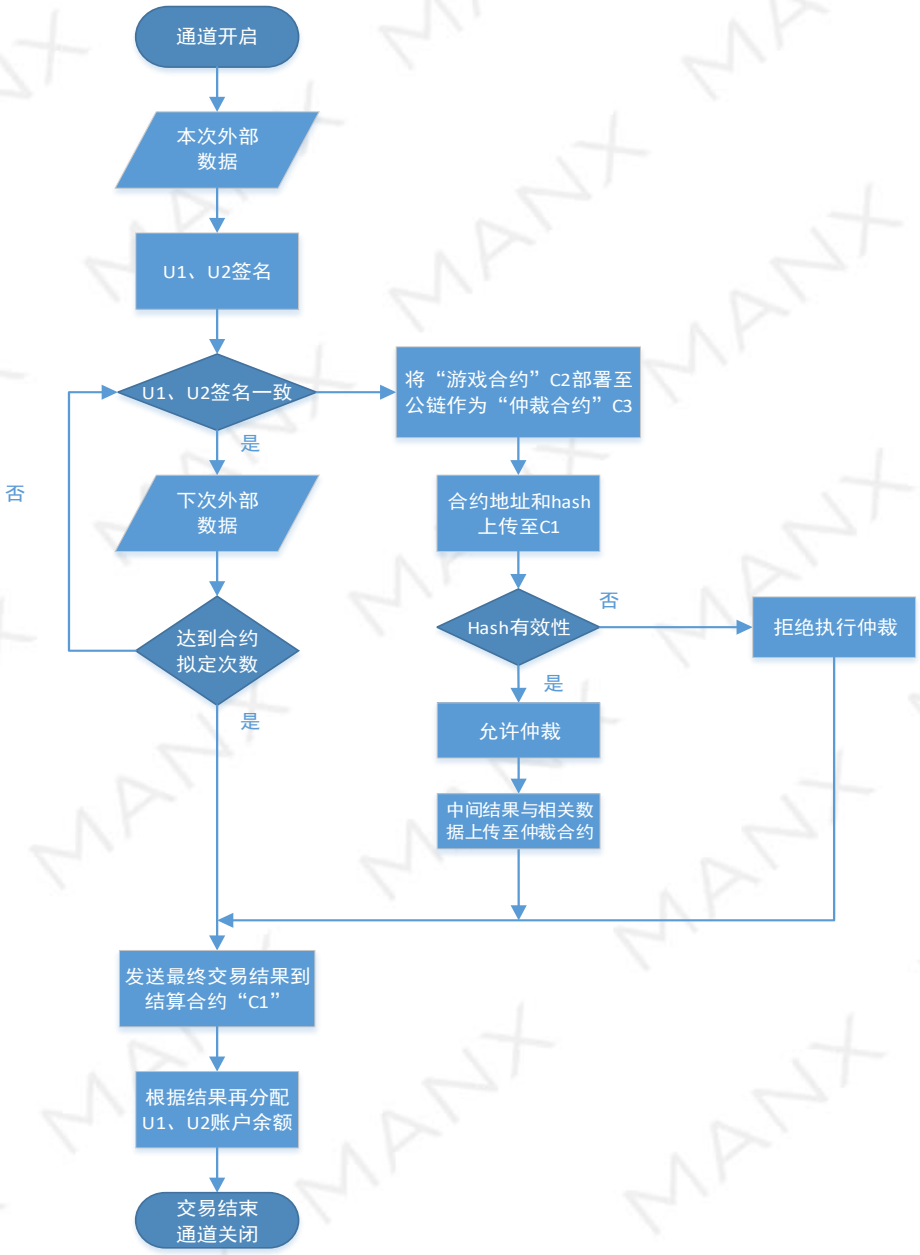
（3） “预言机”可以理解成线上的某个合约，这个合约用于提供外部数据，在 C1 和 C2 拟定之时需在合约内容中确定线上的某个合约作为外部数据的“提供方”。在这个预测游戏中可表现为一个“天气合约”每天固定时间定好当天的最高气温，而客户端也在每天的固定时间自动将“天气合约”中的最高气温值上传至虚拟机中的“游戏合约”C2，而 C2 中的某个函数接口接收这个天气数值，根据之前拟定的规则，改变 U1 和 U2 的余额状态；



状态通道工作流程（3）

（4） 在第(3)步完成后，U1 和 U2 双方各自用私钥对交易结果进行签名，如果双方都认可结果，则签名认可。如果每次双方都对当天交易结果认可，则状态通道将循环执行第(3)步的操作，直到执行完之前拟定的次数为止（在整个过程中，双方的签名可以理解成对某个“标志变量”的修改，例如一个变量 uint x，如果认同，就将 x 改为 1,不认同则改为 2 即可）。之后 U1 或 U2 将最终的交易结果发送至“根合约”C1（结果包括 U1 和 U2 的最终余额），C1 根据交易结果将代币转发至 U1 和 U2 的账户中，整个交易结束，状态通道关闭；若交易进行到某天，U1 或 U2 对某天的交易结果有异议，则其中一方将“游戏合约”C2 部署至公链上作为“仲裁合约”C3，并将合约的地址和 hash 值送至 C1，C1 判断 hash 值是否有效，有效则允许仲裁并认可“仲裁合约”C3 的仲裁结果。有异议的一方将当天的交易结果（交易完成前的双方余额状态值 A11、A21）和当天最高温度值 temp 发送至“根合约”C1，C1 调用已经部署在链上的“仲裁合约”C3，并将 A11 和 A21 也传入 C3 进行仲裁。C1 获取 C3 的

仲裁结果得到最终余额状态 A12 和 A22，并将 A12 和 A22 作为最终交易结果将代币转发至 U1 和 U2 的账户中，整个交易结束，状态通道关闭。



状态通道工作流程（4）

课题 **7**：区块链隐私保护技术对比**:**

项目名称	**MANX**	雷电网络/ **TRINITY**	**Zcash**	**Monero**
原理	多方自治状态通道	双节点的链下通道	零知识证明	环签名
隐私保护对象	任何业务	仅用于微支付	仅保护交易	仅保护交易
隐私保护效果	完全保密 (数据不上链)	仅用于微支付	部分保密 (链上数据保护)	部分保密 (链上数据保护)
隐私保护组策略	支持	不支持	不支持	不支持
支持参与节点数	2~无限	2	不支持	不支持
即时性	高 (交易即确认)	中	低	低

隐私保护性能比较