智能合约技术与应用调研

**摘要**

区块链是一个全球性的去中心化分布式数据库账本。智能合约作为一段由事件驱动的、具有状态的、运行于区块链系统之上的程序，能够保管、处理区块链账本上的数字资产，代码的逻辑定义了智能合约的内容，运行在通用平台上的智能合约还能够实现并扩展传统应用系统的功能。区块链技术的发展为智能合约提供了很好的运行基础，区块链3.0时代，被称为区块链治理时代。是区块链技术和公共治理、实体产业相结合的时代，将链式记账、智能合约和实体领域结合起来，实现去中心化的自治，发挥区块链的价值。

**智能合约的工作原理**

基于区块链的智能合约包括事务的保存和状态处理，都在区块链上完成。事务主要包含需要发送的数据，而时间则是对这些数据的描述信息。当事务及事件信息传入智能合约后，合约资源集合中的资源状态会被更新，进而触发智能合约进行状态机判断。如果事件动作满足触发条件，则由状态机根据参与者的预设信息，选择合约动作自动、正确地执行。

**智能合约的特点**

1.自动化。智能合约可以根据触发条件，自动执行相应的下一步事务，条件判断准确、及时。传统合约需要人工判断触发条件，效率、准确性均不如智能合约。

2.主客观维度。智能合约适合客观性请求的场景，传统合约适合主观性判断指标。主观性判断指标很难纳入自动机中进行判断，从而难以执行下一步事务。

3.成本。智能合约的成本低，所有的判断条件、执行、处置都是自动执行。

4.执行时间。智能合约属于事前约定、预防执行模式，传统合约是事后执行。

5.违约责任。智能合约违约成本高，一旦违约，数字资产、保证金等抵押品将遭受损失，而传统合约的违约执行依赖于刑罚，执行成本高。

6.使用范围。智能合约技术可以全球采用，传统合约受地域影响大，不同地区的法律、人文因素会对执行过程产生影响。

**智能合约在公共事件中的应用场景**

1.资产数字化

个人和机构，无论是私营机构还是政府机构，都可以在这种跨国界和全球各地的不信任环境中安全地进行交易，其系统由区块链上的智能合约提供支持。资产交易发生在规定合同的规则内、愿景中立即发生。

2.身份验证

虽然智能合约无需额外信任，但在很多情况下智能合约仍然需要可信来源才能触发。通常必须来自权威人士或用于身份验证。智能合约在执行之前需要多次使用可信来源的数据，因此必须能够连接到这些可信来源。

3.医疗保健

电子病历已成为许多医院和医疗保健提供者的标准。目标是为患者和医生提供从任何地方即时和安全地访问患者记录的途径。目前使用的大多数系统在允许普遍访问医疗记录方面还没有达到标准。存在竞争标准以及存储和访问信息的竞争方式。这使得患者在被迫使用不同服务时受到较少的控制，这取决于提供者选择使用哪个电子记录供应商。

4.供应链管理

当货物来自全球各地时，公司需要跟踪它们，直到它们到达目的地。他们需要一个不变的，安全的记录，记录货物的存在地点和存在时间。这种需求是区块链的另一个完美用例。利用区块链的安全性和不变性来“审计”供应链的能力将使普华永道等会计师事务所能够跨境工作，并为可能不知道如何开始在其行业中使用区块链的企业客户提供更强大的服务。

5.质量控制

物联网传感器可验证产品真伪，并确保产品在整个供应链中完好无损，比如将食品保存在一定温度下，密封的容器不被中途开封，或者追踪货物的位置。智能合约可以基于物联网数据和合约中约定的质量控制标准触发赔偿和罚款。

6.公用事业

水、电力和网络是现代社会的根基。公用事业对我们的生活至关重要，然而保障公用事业稳定运营的基础设施和技术却早已过时。智能合约可以将老旧的公用事业系统转移到区块链上，为这一至关重要的基础设施带来现代化发展。

7.监管

监管机构和企业之间签署智能合约可以为双方简化监管流程。预言机可以将所需的数据上传至合约中，智能合约可以对数据进行处理，并根据处理结果给出合规或拒绝的认证。预言机可以将两个本来无法交互的系统连接起来。比如，使用传感器测量卡车重量，如果测量结果合格，就给出通过许可。再比如，使用AI算法在区块链上扫描交易活动，并基于合规状态给出认证结果。

**使用智能合约案例**

* 1. 使用 Geth 客户端，搭建测试用的本地区块链，配置私有区块链网络的初始状态

新建文件 genesis.json

 {  
   "config": {  
         "chainId": 22,  
         "homesteadBlock": 0,  
         "eip155Block": 0,  
         "eip158Block": 0  
  },  
   "alloc"     : {},  
   "coinbase"   : "0x0000000000000000000000000000000000000000",  
   "difficulty" : "0x400",  
   "extraData" : "",  
   "gasLimit"   : "0x2fefd8",  
   "nonce"     : "0x0000000000000038",  
   "mixhash"   : "0x0000000000000000000000000000000000000000000000000000000000000000",  
   "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",  
   "timestamp" : "0x00"  
 }

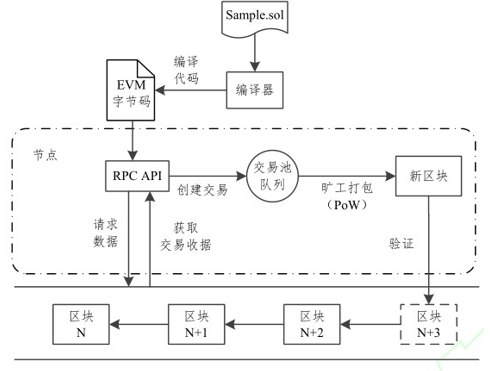
* 1. 初始化区块链，生成创世区块和初始状态

 $ geth --datadir /path/to/datadir init /path/to/genesis.json

* 1. 创建账号

 > personal.newAccount()  
 ​  
 Passphrase:  
 Repeat passphrase:  
 "0x1b6eaa5c016af9a3d7549c8679966311183f129e"

* 1. 创建和编译智能合约



* + - 用solc获得合约编译后的EVM二进制码

 $ solc --bin testContract.sol  
 ​  
 ======= testContract.sol:testContract =======  
 Binary:  
 6060604052341561000c57fe5b5b60a58061001b6000396000f30060606040526000357c0100000000000000000000000000000000000000000000000000000000900463ffffffff168063c6888fa114603a575bfe5b3415604157fe5b60556004808035906020019091905050606b565b6040518082815260200191505060405180910390f35b60006007820290505b9190505600a165627a7a72305820748467daab52f2f1a63180df2c4926f3431a2aa82dcdfbcbde5e7d036742a94b0029

* + - 再用solc获得合约的 JSON ABI (Application Binary Interface)，其中指定了合约接口，包括可调用的合约方法、变量、事件等

 $ solc --abi testContract.sol  
 ​  
 ======= testContract.sol:testContract =======  
 Contract JSON ABI  
 [{"constant":false,"inputs":[{"name":"a","type":"uint256"}],"name":"multiply",  
 "outputs":[{"name":"d","type":"uint256"}],"payable":false,"type":"function"}]

* + - 回到Geth 的 JavaScript 环境命令行界面，用变量记录上述两个值，在code 前加上 0x 前缀

 >code = "0x6060604052341561000c57fe5b5b60a58061001b6000396000f30060606040526000357c01000000000000000000000000000000  
 00000000000000000000000000900463ffffffff168063c6888fa114603a575bfe5b3415604157fe5b60556004808035906020019091905050606b565b6040518082815260200191505060405180910390f35b60006007820290505b9190505600a165627a7a72305820748467daab52f2f1a63180df2c4926f3431a2aa82dcdfbcbde5e7d036742a94b0029"  
 > abi = [{"constant":false,"inputs":[{"name":"a","type":"uint256"}],"name":"multiply","outputs":[{"name":"d","type":"uint256"}],"payable":false,"type":"function"}]

* + - 发送部署智能合约的交易

 > myContract = eth.contract(abi)  
 > contract = myContract.new({from:myAddress,data:code,gas:1000000})

* 1. 调用智能合约，用以下命令可以发送交易，其中 sendTransaction 方法的前几个参数与合约中 multiply 方法的输入参数对应。交易会通过挖矿记录到区块链中，如果涉及状态改变也会获得全网共识

 > contract.multiply.sendTransaction(10, {from:myAddress})