智能合约隐私保护方案

区块链中的隐私保护方案大多基于转账交易，而智能合约涉及的应用数据的隐私保护方法却比较有限。现有的方案大多是将数据的散列值存在区块链上，用于完整性的验证，而原始数据和数据的计算都放在链下。这种方式会导致计算过于中心化，很难保证安全性。

超级链主要针对使用 Wasm 作为智能合约虚拟机的区块链系统，设计了一种基于 TEE 的隐私保护方法，通过虚拟机内置 TEE 可信方法的方式完成隐私计算，比较容易部署。方案涉及两个模块：

1. 秘钥托管模块： 解决秘钥存储，秘钥更新等问题
2. 隐私计算流程： 解决密文计算，包括正常的四则运算

一、秘钥托管

秘钥托管采用一种基于 HMAC 秘钥衍生算法。采用一种链状的方式派生秘密信息 kds（key derivation secret）。先由某个 TTP 随机生成一个根 kds(base dks，即 bds)，然后通过 HMAC 生成一定数量的子 kds：

假设 kds(n)表示第 n 代子 kds， 那么 bds 就是 kds(0)

kds(n) = HMAC(kds(n-1), n-1 | kds(n-1))，其中 | 表示链接符号。

实际应用中，先使用最末尾的 kds(n)，需要更新就向前使用，这样的话新的 kds 就可以计算得到旧的 kds。

利用 kds 可以计算每次交易使用的秘钥：

key = HMAC(kds, address | args_hash)

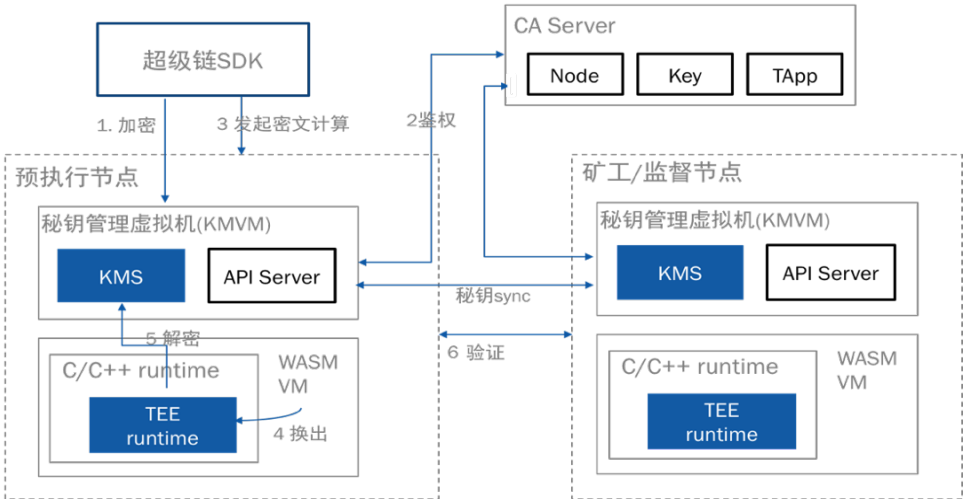
- key 是对称加密秘钥
- kds 为该轮使用的秘密信息
- 其中 address 就是当前发起交易或者合约的用户的地址
- args_hash 是当前智能合约调用的参数的原文的 hash

通过这种方式，能满足秘钥可更新、秘钥无关联。不同的合约调用使用不重复的秘钥。秘钥派生之后，不同节点可在 TEE 里面完成秘钥的同步：

可信第三方	→	**区块链节点**
TEE 里面进行生成	加密同步	最新秘钥

二、隐私计算

计算流程只将涉及到密文处理的计算部分放到 TEE 里，过程如下：

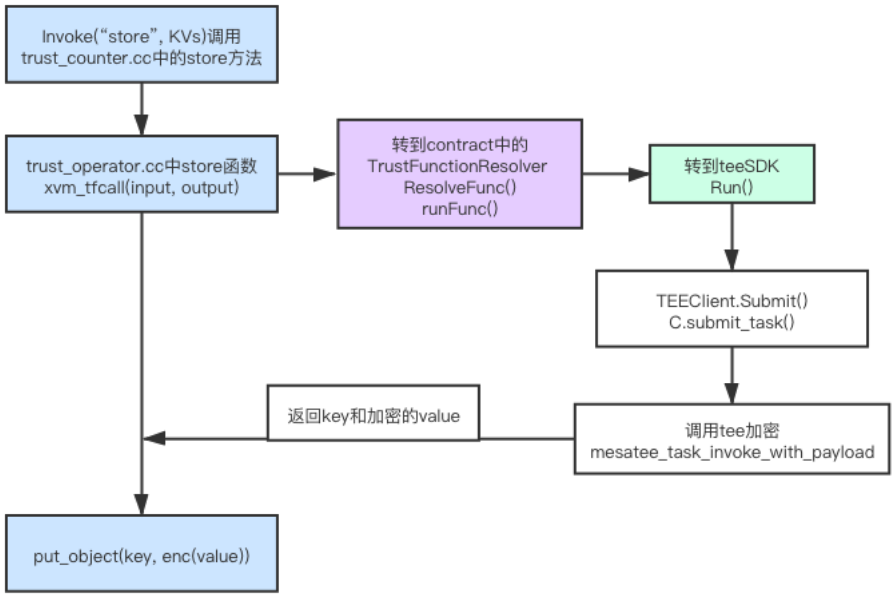


1. 超级链 SDK 跟预执行节点建立加密通道，将合约调用的明文信息传递到 TEE 环境；
2. 秘钥管理虚拟机(KMVM)去证书颁发机构(CA)验证发起者的身份以及证书合法性，合法则将合约参数加密并返回，否则拒绝请求；
3. SDK 发起合约执行，合约参数是加密之后的密文；
4. 合约虚拟机 wasmVM 拿到密文之后，通过虚拟机内置函数的方式调用 TEE 暴露的方法进行计算；
5. TEE 可信方法先调用 KMS 解密密文，然后进行明文计算，计算完成后再加密回密文；
6. 计算得到的结果经过区块链的共识，转发到网络中的其他节点，重复进行步骤 4 和 5 的计算来验证结果。

三、合约调用流程

超级链 contractsdk/cpp/example/turstops 中有一个智能合约实例 trust_counter.cc，用于隐私数据的加密存储。合约调用过程与普通的 wasm 合约调用类似，下面以调用该合约的 store 方法为例，详细讲解合约调用流程，该方法会将 KV 对的 value 加密存储：

1. sdk 通过 invoke 调用 trust_counter.cc 中 store 方法，输入若干 KV 对作为参数；
2. store 的实现在 trust_operators.cc 中，该函数会先通过 xvm_tfcall 调用外部方法，利用 teesdk 实现 value 值的加密，然后调用 put_putobject 方法，将 key 和加密过后的 value 存到区块链中，value 加密流程如下：
3. 合约外部调用会转到 contract 中的 TrustFunctionResolver，这里解析函数会调用 runFunc()函数；
4. runFunc 会通过提前设置的 pluginPath 定位到 teesdk，调用 teesdk 的 Run 函数，teesdk 是 Csdk 的 Go 封装；
5. Run 函数调用 TEEClient.Submit，这个函数会调用 C 的 submit_task 函数，然后将任务交给 TEE 进行加密计算，并得到加密的 value。



2020 年 4 月 7 日 @Hongyan Wang