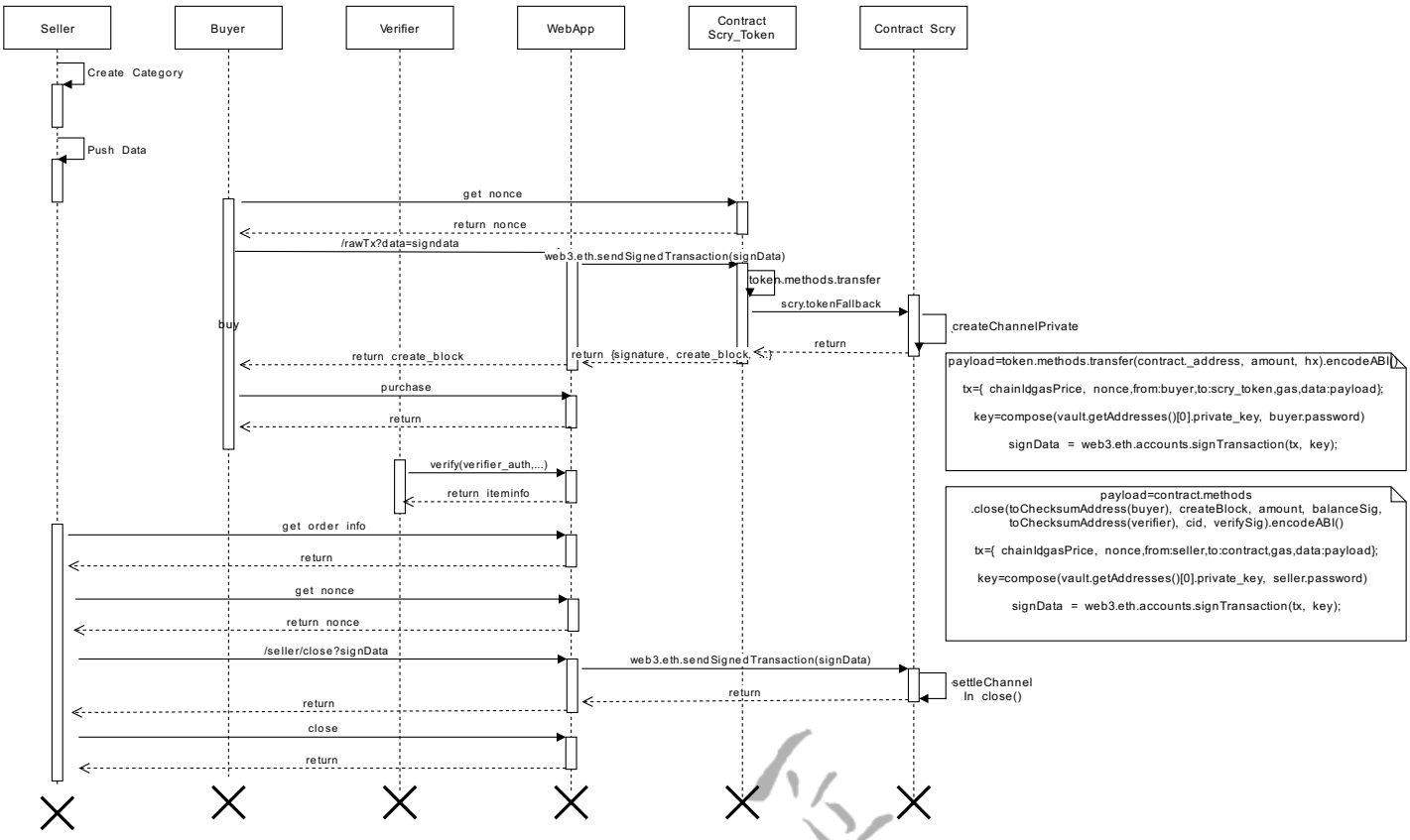
区块链交易协议层设计

目录

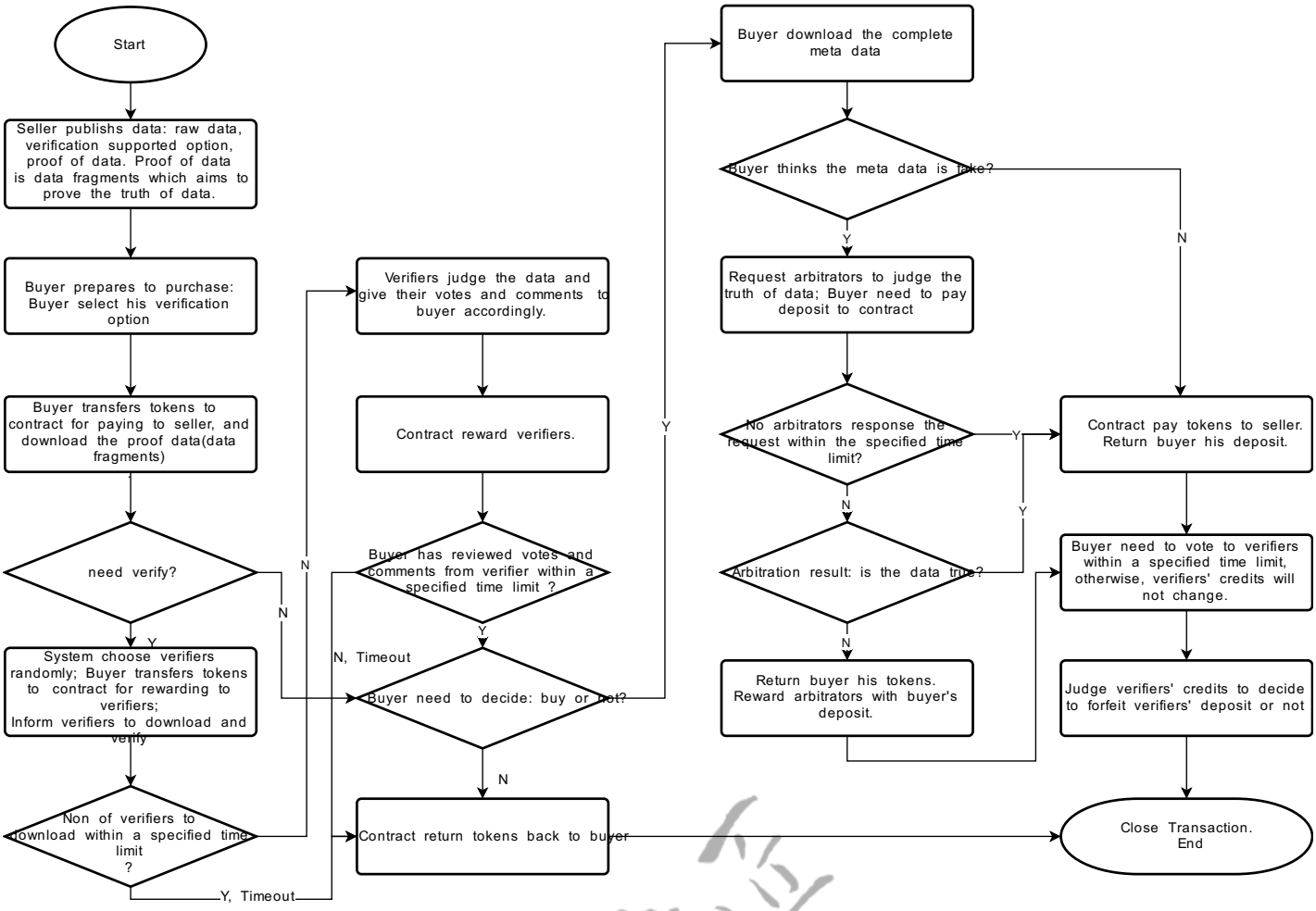
原始实现



设计原则

- 数据私密性
- 验证准确度
- 用户匿名
- 资金安全性
- 谁主张，谁举证
- 尽量简单

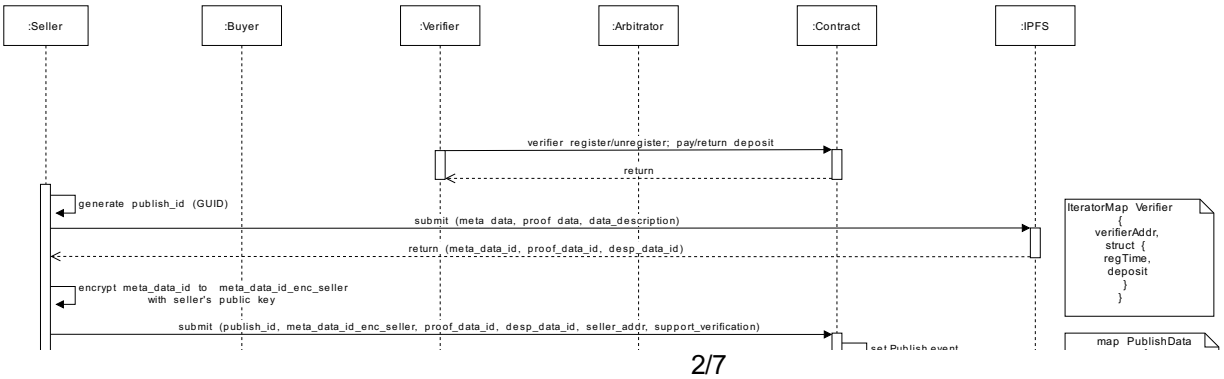
业务流程设计

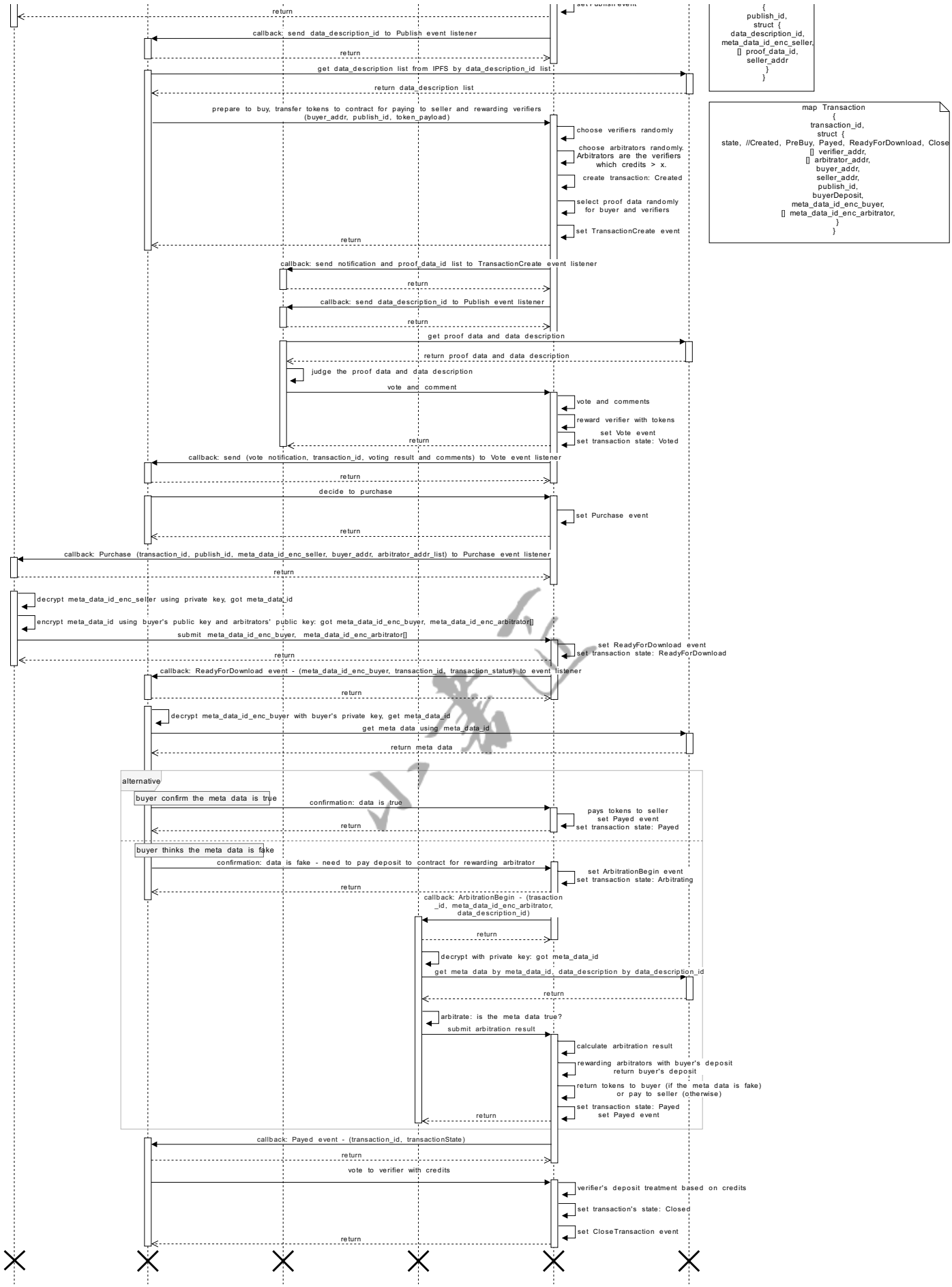


几个重要的问题及说明

1. seller可以选择是否启用verify机制，即图中verification option。buyer同样可以选择是否启用verify机制。
2. seller发布的数据，包括了元数据，数据证明。数据证明的设想是：首先，seller需要明确指出其发布的数据的类型，关键内容；其次，seller需要举出至少一处能够证明上述关键内容的元数据片段，并以文字或者图片形式发布。
3. 针对verifier，系统设计了credit机制。具体credit机制可参考滴滴打车，是一个平均分评价机制。当verifier的credit低到某个值以后，系统会没收其押金，verifier除非重新缴纳押金，否则会失去verify资格。
4. 开始验证程序后，buyer和verifier都会收到系统随机发送的data fragments。若在规定期限内没有任何verifier愿意进行verify，则系统进入取消购买流程。
5. buyer购买了data后，能够下载完整的data。这时，buyer可依据verify的质量对verify进行投票打分。
6. seller一旦发布了数据，就永远不能更改。

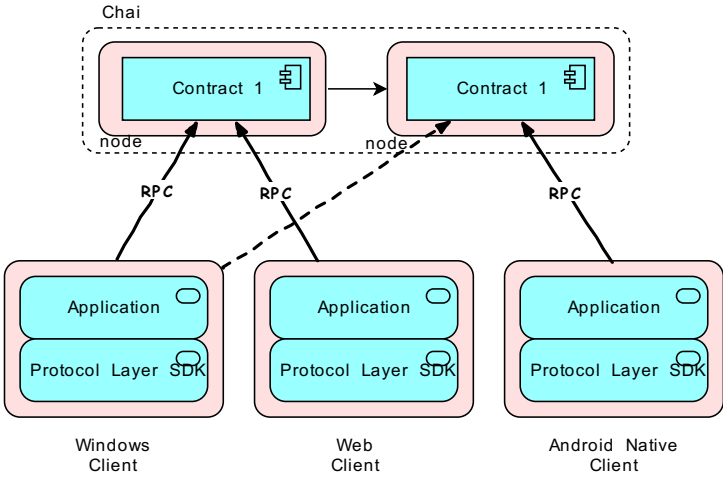
组件交互流程设计




{
publish_id,
struct {
data_description_id,
meta_data_id_enc_seller,
[] proof_data_id,
seller_addr
}
}
map Transaction
{
transaction_id,
struct {
state, //Created, PreBuy, Payed, ReadyForDownload, Close
[] verifier_addr,
[] arbitrator_addr,
buyer_addr,
seller_addr,
publish_id,
buyerDeposit,
meta_data_id_enc_buyer,
[] meta_data_id_enc_arbitrator,
}
}
return
callback: send data_description_id to Publish event listener
return
get data_description list from IPFS by data_description_id list
return data_description list
prepare to buy, transfer tokens to contract for paying to seller and rewarding verifiers (buyer_addr, publish_id, token_payload)
choose verifiers randomly
choose arbitrators randomly. Arbitrators are the verifiers which credits > x.
create transaction: Created
select proof data randomly for buyer and verifiers
set TransactionCreate event
return
callback: send notification and proof_data_id list to TransactionCreate event listener
return
callback: send data_description_id to Publish event listener
return
get proof data and data description
return proof data and data description
judge the proof data and data description
vote and comment
vote and comments
reward verifier with tokens
set Vote event
set transaction state: Voted
return
callback: send (vote notification, transaction_id, voting result and comments) to Vote event listener
return
decide to purchase
set Purchase event
return
callback: Purchase (transaction_id, publish_id, meta_data_id_enc_seller, buyer_addr, arbitrator_addr_list) to Purchase event listener
return
decrypt meta_data_id_enc_seller using private key, got meta_data_id
encrypt meta_data_id using buyer's public key and arbitrators' public key: got meta_data_id_enc_buyer, meta_data_id_enc_arbitrator[]
submit meta_data_id_enc_buyer, meta_data_id_enc_arbitrator[]
set ReadyForDownload event
set transaction state: ReadyForDownload
return
callback: ReadyForDownload event - (meta_data_id_enc_buyer, transaction_id, transaction_status) to event listener
return
decrypt meta_data_id_enc_buyer with buyer's private key, get meta_data_id
get meta data using meta_data_id
return meta data
alternative
buyer confirm the meta data is true
confirmation: data is true
pays tokens to seller
set Payed event
set transaction state: Payed
return
buyer thinks the meta data is fake
confirmation: data is fake - need to pay deposit to contract for rewarding arbitrator
set ArbitrationBegin event
set transaction state: Arbitrating
return
callback: ArbitrationBegin - (trasaction_id, meta_data_id_enc_arbitrator, data_description_id)
return
decrypt with private key: got meta_data_id
get meta data by meta_data_id, data_description by data_description_id
return
arbitrate: is the meta data true?
submit arbitration result
calculate arbitration result
rewarding arbitrators with buyer's deposit
return buyer's deposit
return tokens to buyer (if the meta data is fake)
or pay to seller (otherwise)
set transaction state: Payed
set Payed event
return
callback: Payed event - (transaction_id, transactionState)
return
vote to verifier with credits
verifier's deposit treatment based on credits
set transaction's state: Closed
set CloseTransaction event
return

系统架构

方案 1 - 基于本地前置引擎的去中心化模式(Prefered)



特点

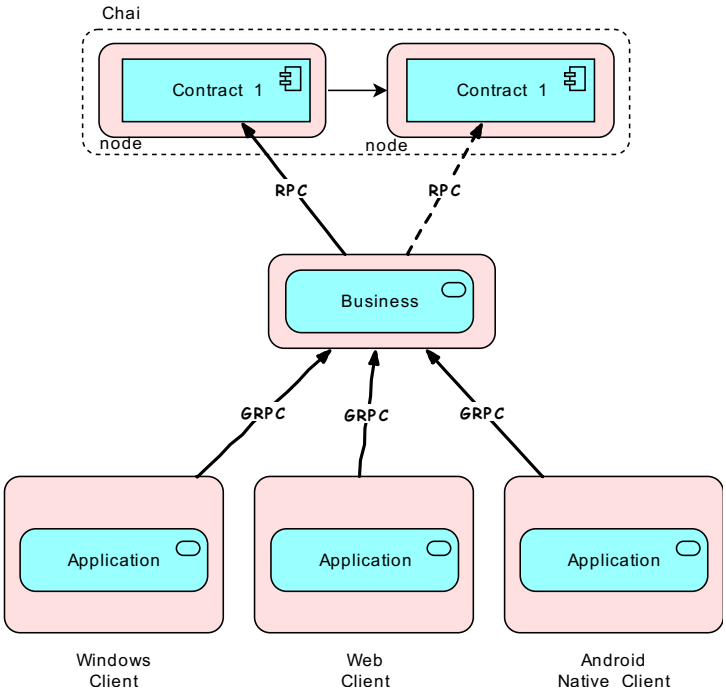
- 每个SDK实例只需管理一个用户
- SDK作为前置引擎驱动整个业务流程，上层应用调用SDK在界面展现数据
- SDK直接连接到链上，进行去中心化的操作
- SDK的用户控制：每一时刻只有一个当前用户，SDK负责驱动该当前用户的业务流，上层应用可切换当前用户
- SDK的状态控制：SDK维护一个状态机(state machine)，根据状态机来驱动客户端业务流程
- 要求客户端自行安装IPFS节点

SDK接口模式

- Candidate 1: 基于Electron的接口模式
  SDK与上层应用处于同一进程，安全性高，接口不友好，耦合度高，不方便上层应用调用

- Candidate 2: 基于GRPC通信协议的接口模式
  SDK独立进程，接口友好，与上层应用形式完全解耦，需要额外验证调用者身份，安全性不好

方案 2 - 基于Web Server的中心服务器模式(Not Prefered)

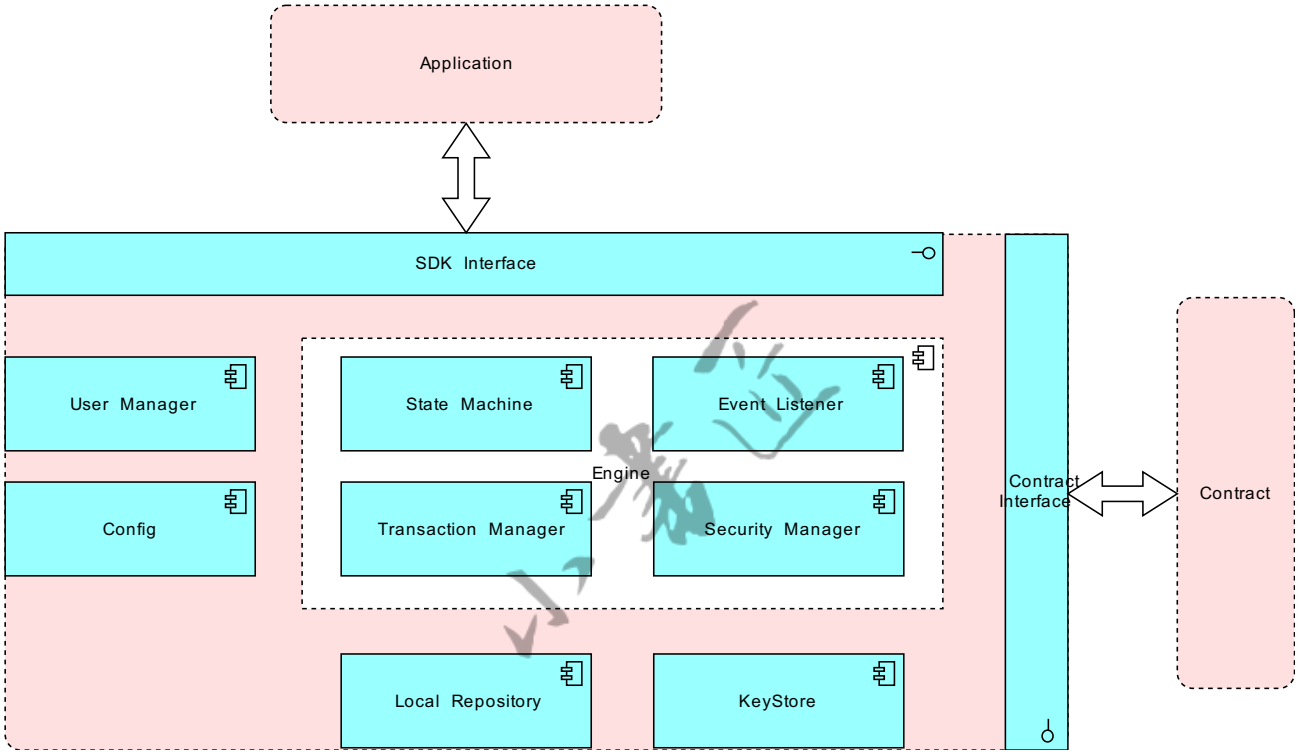


特点

- Web Server需要同时管理所有用户业务流程，逻辑较复杂
- 需要在中心节点部署IPFS和服务器软硬件和带宽，存在费用成本，此成本由谁来承担是个问题
- 轻客户端模式，用户易于接入
- 安全风险集中在中心节点，存在单点故障
- 不是完全的去中心化系统

SDK

SDK设计



- application调用sdk: grpc

SDK接口定义

接口名称	接口描述
CreateAccount	创建账户
PublishData	发布数据
RegisterVerfifier	注册一个verifier
GetDataDescriptionList	取数据描述列表
PrepareToBuy	准备买
GetProofDataList	取proof data 列表
VoteAndCommentForMetaData	依据proof data，给meta data投票和评论
GetVoteAndComments	取投票结果和评论结果
BuyData	决定购买数据

接口名称	接口描述
FeedbackMetaDataTruth	反馈数据的真实性
DoArbitration	进行仲裁
VoteToVerifier	投票给verifier

合约

合约接口定义

接口名称	功能描述	调用方法	值返回方式
isVerifierRegistered(address addr) returns (bool registered)	用户是否注册为verifier	call	同步返回
registerVerifier(address veriferAddr) returns (bool success)	用户注册为verifier	transaction	VerifierRegisteredEvent
publishDataIndexes(string publishId, hex meta_data_id_enc_seller, string[] proof_data_ids, string desp_data_id, bool supportVerify) returns (bool success)	发布数据在IPFS中的id，并指出是否支持验证机制	transaction	PublishEvent
prepareToBuy(string publishId, address buyerAddr, bool needVerify) returns (bool success)	准备购买，转账给contract，创建交易。如果seller支持验证机制，且buyer需要验证机制，随机选择verifiers，并给每个verifier和buyer随机选择proof data list，否则只为buyer随机选择proof data list。	transaction	CreateTransactionEvent
judgeData(string transactionId, bool isTrue, string comment) returns (bool success)	verifier给数据投票和评论	transaction	VoteEvent
buy(string publishId, string transactionId) returns (bool success)	购买. contract转账给seller成功后，给seller发送交易已支付事件以及meta_data_Id_enc_seller	transaction	TransactionPayedEvent
submitEncyptedMetaDataId(string transactionId, hex meta_data_id_enc_buyer) returns (bool success)	卖家提交使用买家公钥加密的元数据ID	transaction	ReadyForDownloadEvent
voteToVerifier(address verifierAddr, uint credit, string transactionId) returns (bool success)	买家在得到原始数据以后，给verifier信用打分。根据verifier的信用情况，若其信用平均分低于x分，决定是否没收押金。交易完成，关闭	transaction	CloseTransactionEvent
checkTransactionTimeout(string transactionId) returns (bool isTimeout)	检查合约中交易是否超时	call	同步返回
processTransactionTimeout(string transactionId) returns (bool success)	处理交易超时	transaction	TransactionTimeoutEvent

合约设计

- token contract - 由于ddd是erc20标准的token，为保持一致，且考虑到安全性，选择继承自OpenZepplin的StandardToken作为测试token
- 业务 contract - 包含整个交易的所有逻辑处理

交易

交易状态

Created
Voted
Payed
ReadyForDownload
Closed

交易超时

交易超时，需要对交易做相应的处理：退费，自动关闭交易等。

客户端调用

调用合约

调用transaction性质的合约，是发送一个交易到链上，所以需要对交易进行签名后再发送。