健康认证的技术方案

ASR分析

角色\ASR	关键需求	质量属性	约束	原则
用户（客户端）	1. 申请健康证明 2. 出示健康证明二维码 3. 进行KYC认证 4. 注册 DID 5. DID 登录	POC 级别，流程跑通	1. 验证区块链核心能力，用技术手段建立信任。 2. 实现要结合当下现状，具体参考前提假设的讨论。	1. 简单快速实现，能 mock 的不实现接口，能不出UI就不出UI 2. 设计可以尽可能多考虑，实现需要排优先级。
健康认证机构（体检中心）	1. 健康认证信息收集 2. 验证用户KYC信息 3. 出示健康认证证书 4. 注册 DID 5. DID 登录			
健康检查机构（商场）	1. 扫健康码，进行健康码验证 2. 注册 DID			
KYC服务（可选）	1. 验证用户信息 2. 把信息验证信息上链 3. 注册 DID			
区块链	1. 存贮数字身份DID 2. 存贮可信声明的验证信息			
区块链服务	1. 提供注册 DID 功能 2. 提供 DID 信息检索功能 3. 提供验证可信声明功能			

前提假设的讨论

全功能情况

用户，健康认证机构，KYC服务提供商都已经上链，有自己的 DID，并且有自己的设备，可以去链上验证可信声明。

1. 所有角色的数字身份和账号依赖于 DID，需要 mock 这些 DID 或者提供注册功能（网页）。
2. 去健康认证机构需要出示 KYC 信息，并使用 DID 登录。

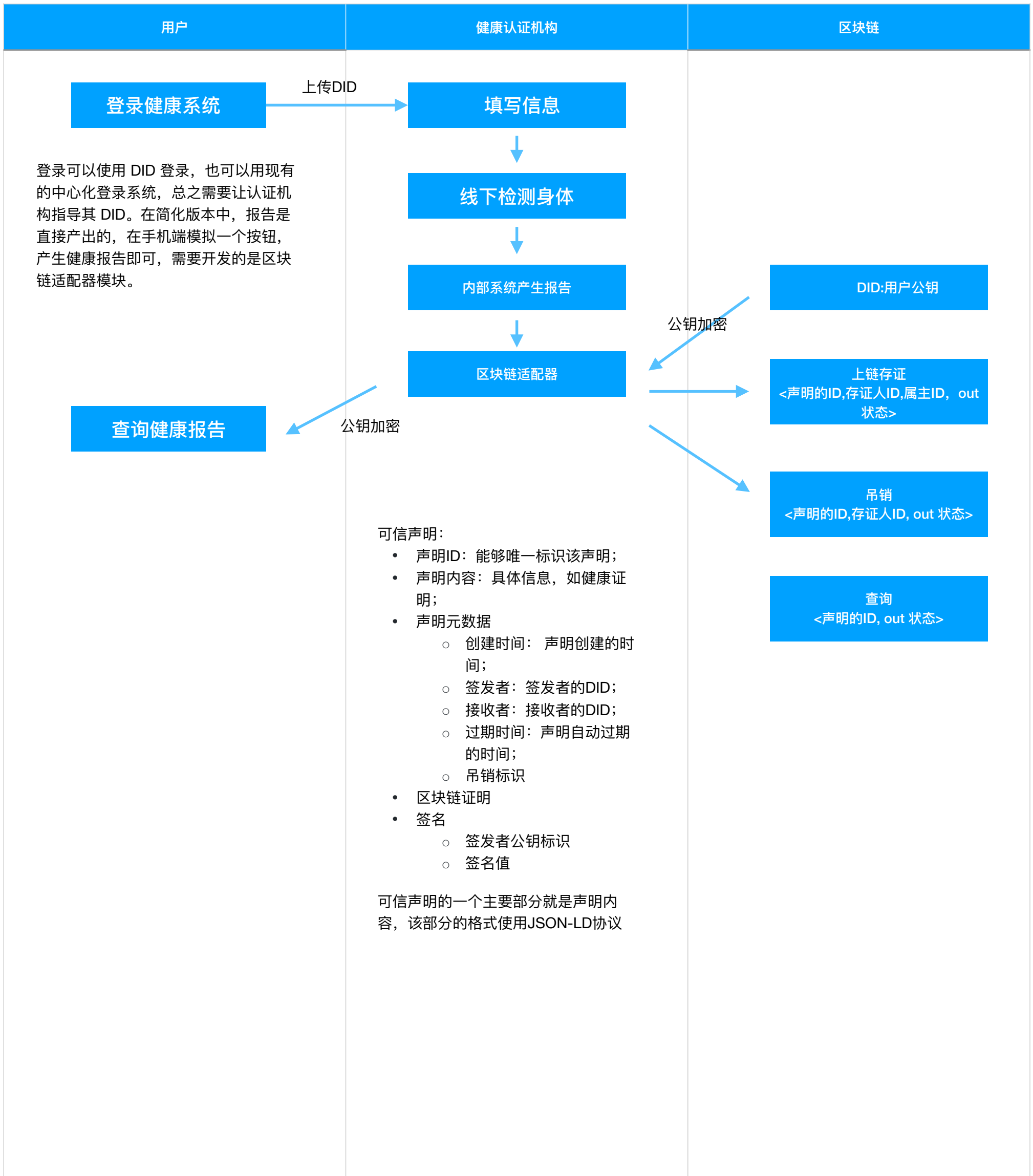
简化情况

目前情况：

1. 考虑到现有系统大部分是中心化系统，不太可能用 DID 登录。
2. 不太可能都注册 DID。在简化情况下，让健康认证机构和用户注册 DID，这是最低要求，下面的流程在此基础上讨论。
3. 非验证区块链核心功能的需求，都可以先不做，因为本 POC 主要还不是实现功能。

业务流程分析

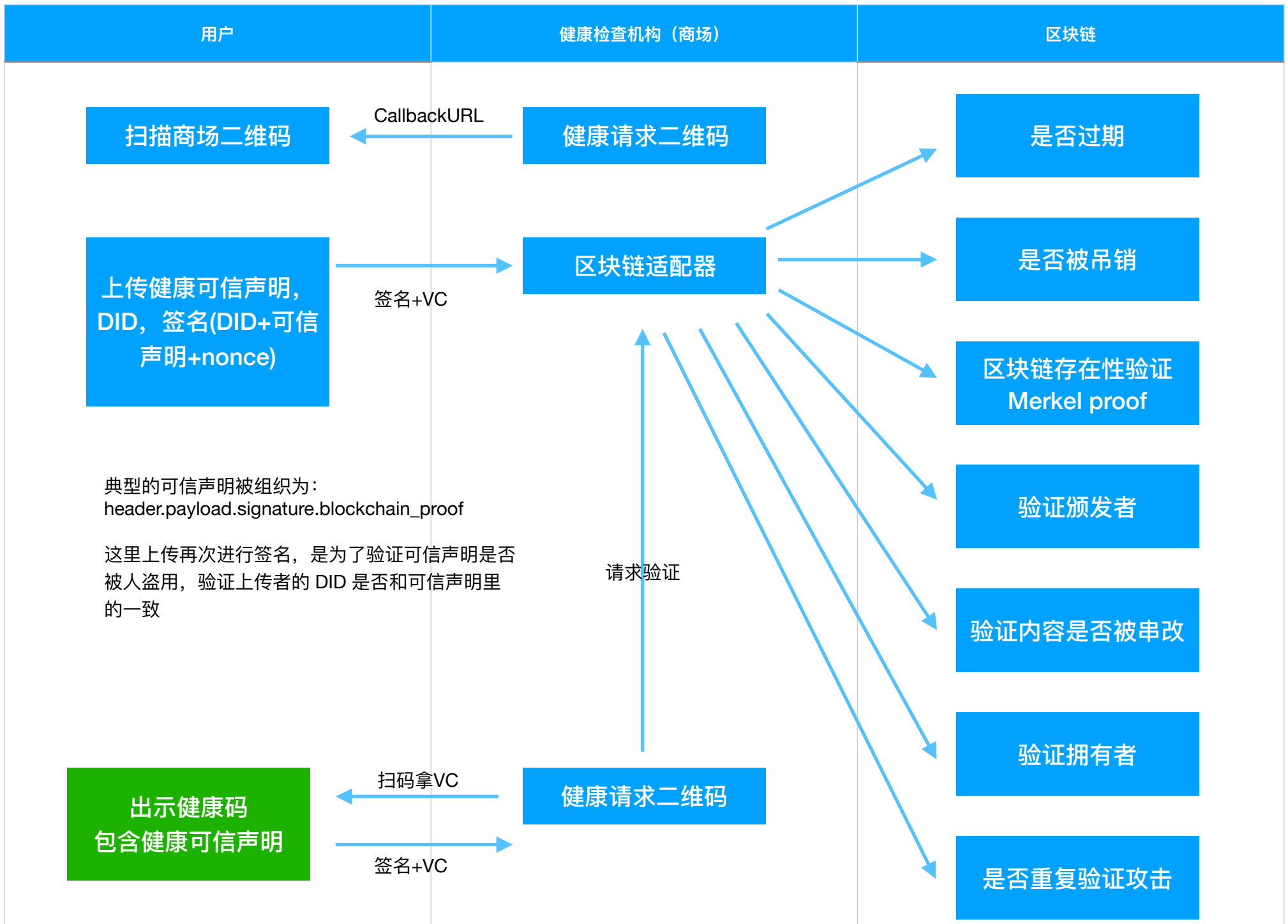
健康证明申请流程



注意：

1. 此处省略了KYC，会导致健康机构无法通过数字化手段验证身份，需要线下出示身份证。
2. 此处可选 DID 登录，详情请参见 DID 登录协议。
3. 在简化流程中，公钥加密是可以不实现的。
4. 假设上链一定是成功的。

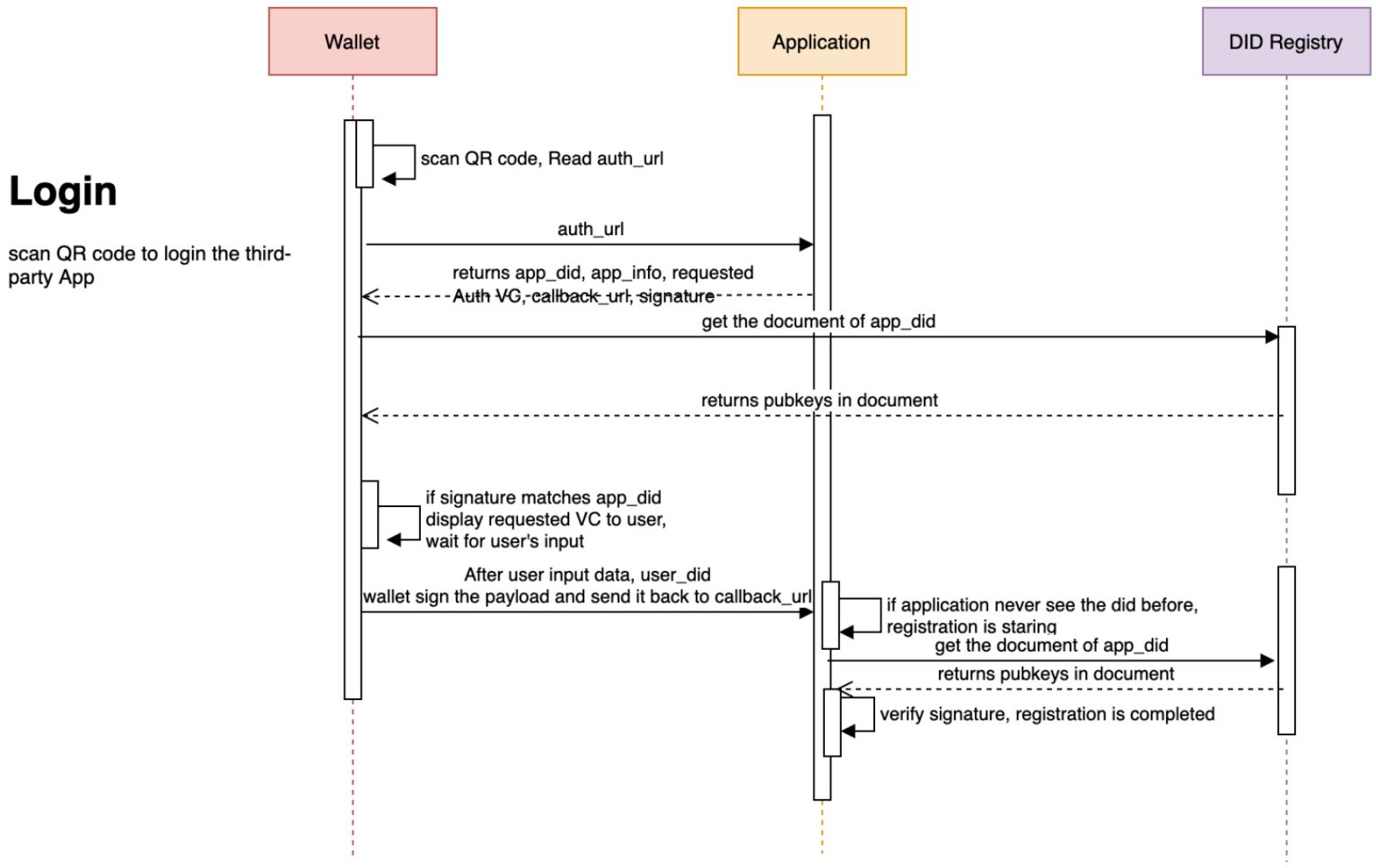
健康声明验证流程



注意：

1. 此处的验证充分说明了区块链是如何打造信任基础，说明了**“基于数学和代码的信任”**和**“基于权利和人类法律的信任”**的不同点。
2. 此处缺少了选择可信声明的过程，扫码以后，用户必须知道要出示何种类型的可信声明。
3. wallet 出示可信声明二维码，然后机器或者人验证也是可以的，此流程在可信声明比较小，可以被二维码包含的情况下比较好用。如果内容较多，会导致生成的二维码过密，难以识别。
4. 少了用户验证商场的流程。

DID 登录流程



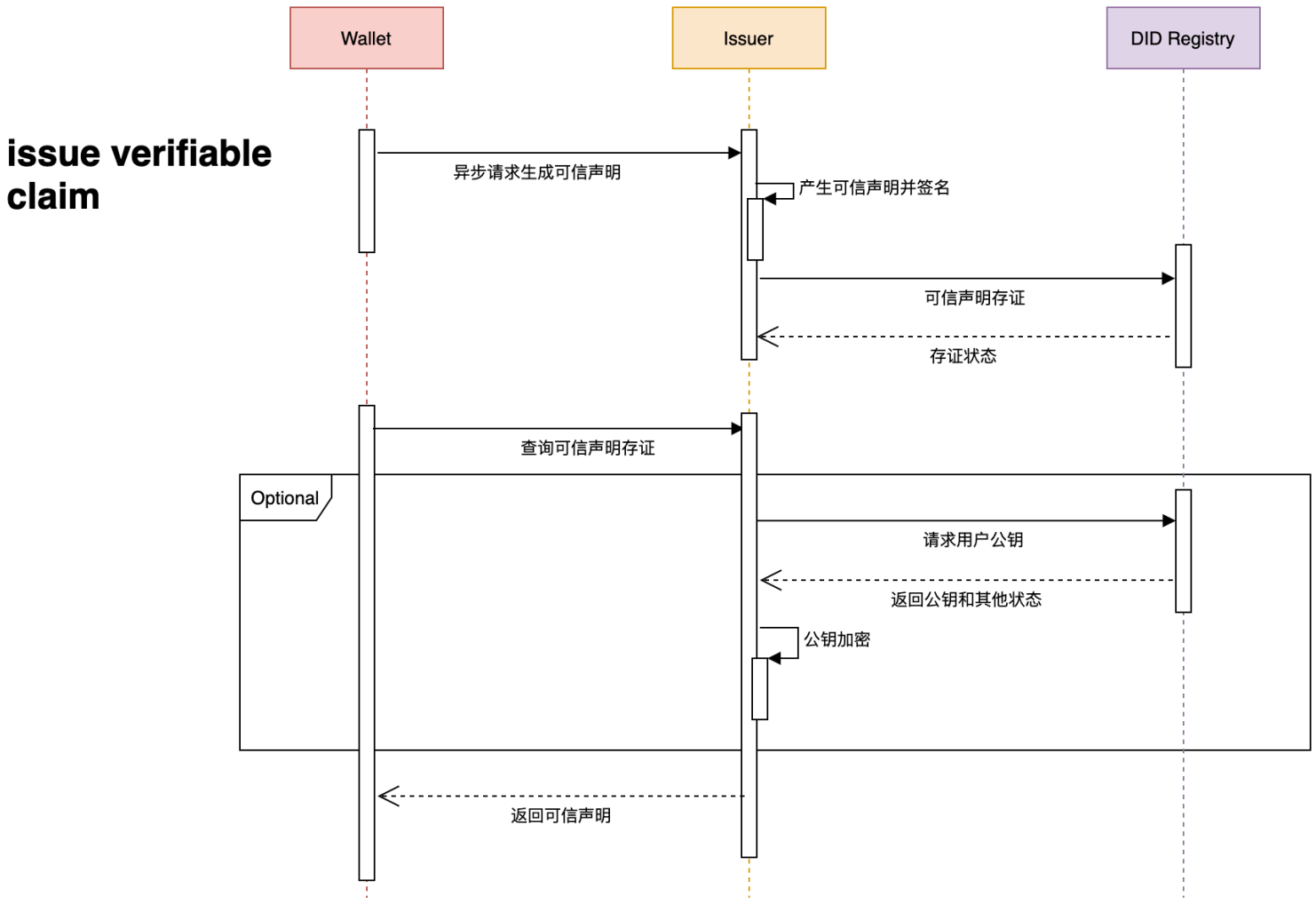
注意：

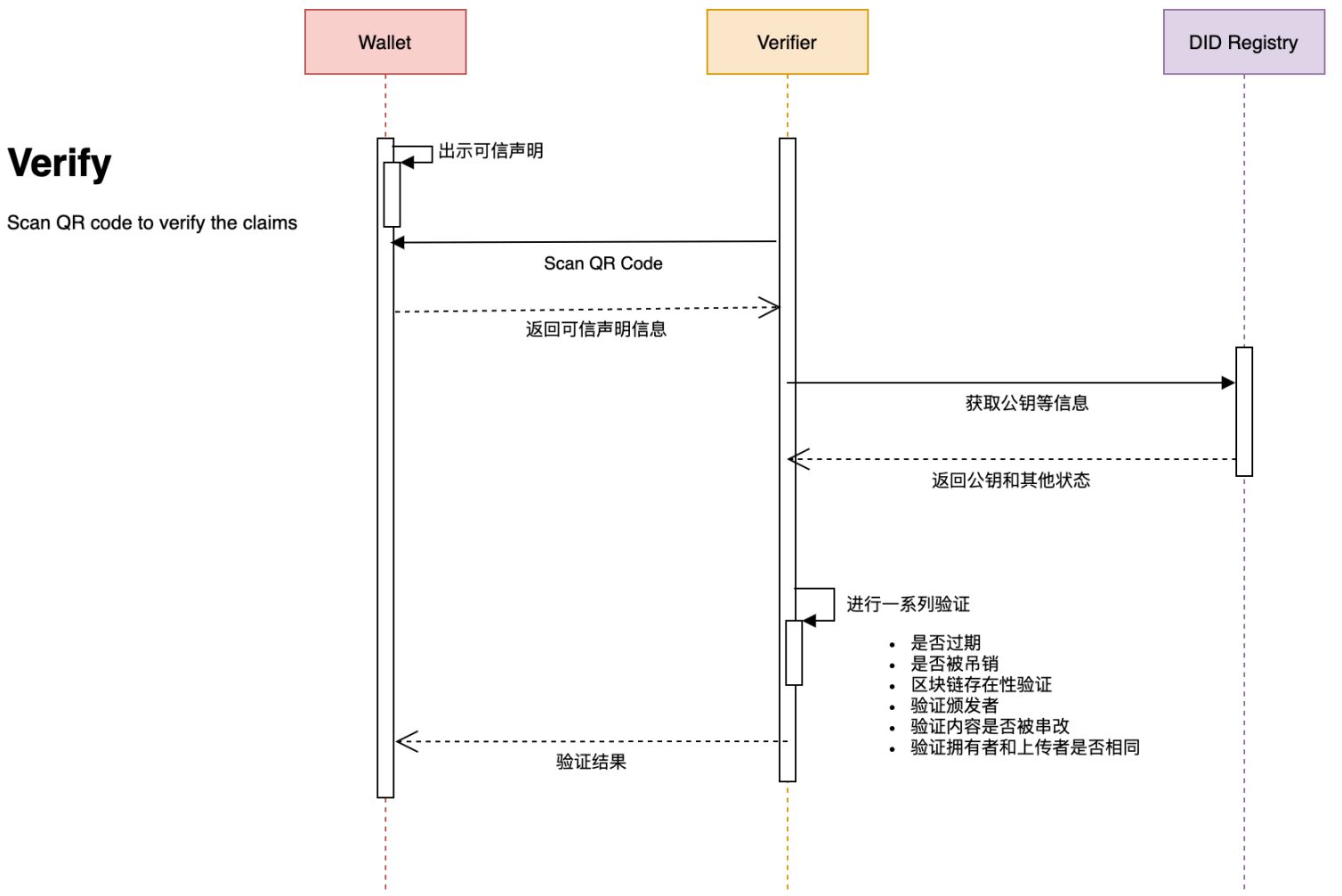
1. 在本次简化实现中，**可以不使用 DID 登录**，因为我们连第三方系统也没有，就只是模拟一下。
2. DID 注册流程，应该考虑如何绑定现有中心化账户，如果第三方系统是中心化的，就会给用户创建一个账户，DID 登录只是关联到其中心化账户。
3. 图的链接

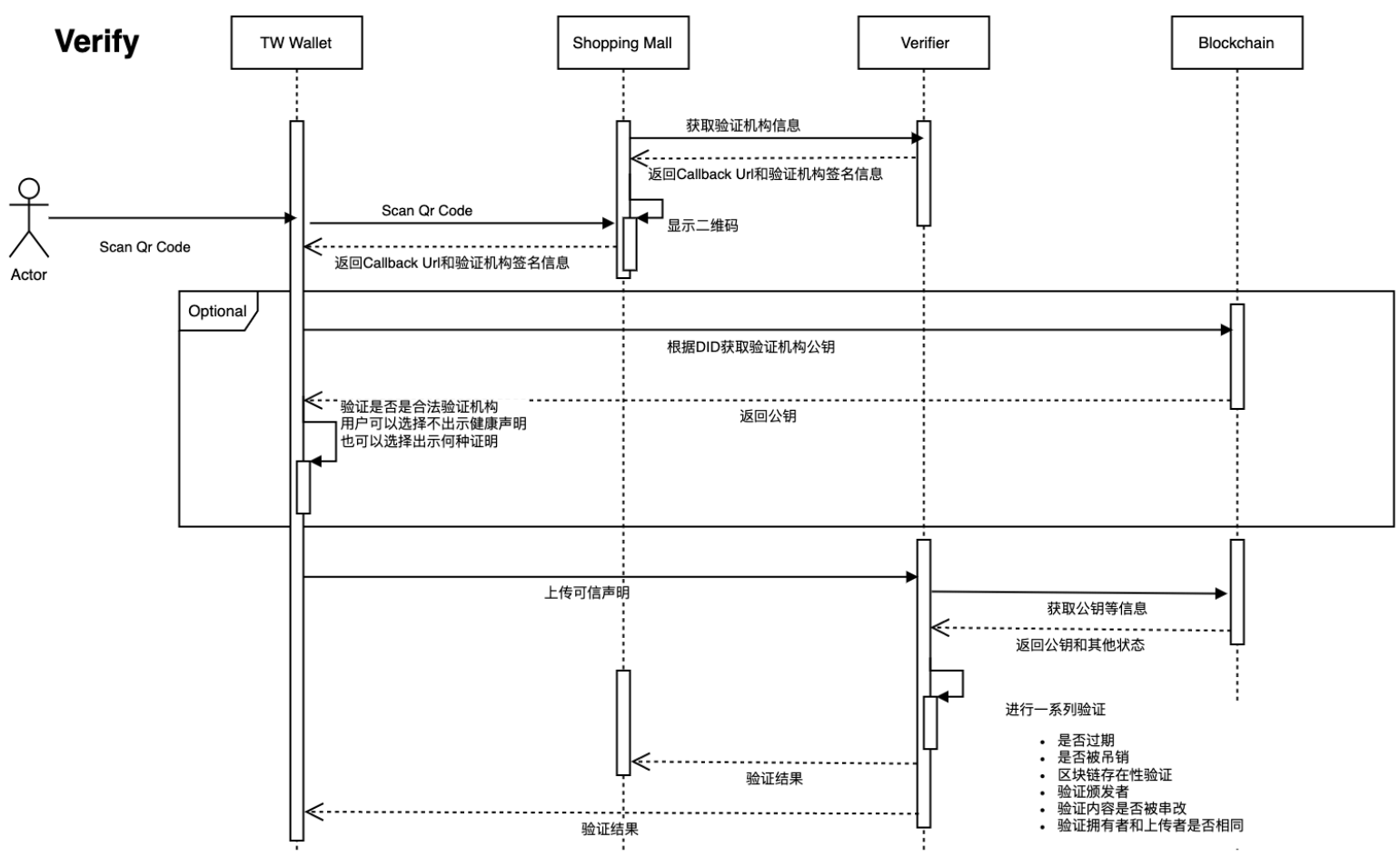
KYC流程

暂时不做

补充序列图


issue verifiable claim
Wallet
Issuer
DID Registry
异步请求生成可信声明
产生可信声明并签名
可信声明存证
存证状态
查询可信声明存证
Optional
请求用户公钥
返回公钥和其他状态
公钥加密
返回可信声明


Verify
Scan QR code to verify the claims
Wallet
Verifier
DID Registry
出示可信声明
Scan QR Code
返回可信声明信息
获取公钥等信息
返回公钥和其他状态
进行一系列验证
是否过期
是否被吊销
区块链存在性验证
验证颁发者
验证内容是否被串改
验证拥有者和上传者是否相同
验证结果


Verify
TW Wallet
Shopping Mall
Verifier
Blockchain
Actor
获取验证机构信息
返回Callback Url和验证机构签名信息
Scan Qr Code
显示二维码
Scan Qr Code
返回Callback Url和验证机构签名信息
Optional
根据DID获取验证机构公钥
验证是否是合法验证机构
用户可以选择不出示健康声明
也可以选择出示何种证明
返回公钥
上传可信声明
获取公钥等信息
返回公钥和其他状态
进行一系列验证
是否过期
是否被吊销
区块链存在性验证
验证颁发者
验证内容是否被串改
验证拥有者和上传者是否相同
验证结果
验证结果

通用设计

- 在此总结出未来可用的通用设计，然后可以设计出通用模块。
- 因为有很多简化设计，所以在此总结出大的原则，未来可以继续补充。

DID 协议

DID 协议应明确以下内容：

1. 如何生成 DID
    1. 生成算法
    2. URL 格式
    3. 如何验证
    4. 如何存贮在区块链
2. 如何管理 DID
    1. 谁控制
    2. 控制权如何转移，销毁，丢失找回
    3. 如何在企业内部适配和映射现有的权限系统
3. 和 DID 相关的支持协议如下
    1. 如何搜索 DID 下相关内容，基于语义网和Json-LD格式的高质量互联网
    2. 如何建立 DID 域名系统，建立去中心化域名系统
    3. 可信声明协议
4. DID 派生出来的协议有
    1. 登录协议
    2. 如何绑定现有中心化账号

可信声明协议

1. 如何产生，管理，销毁，验证可信声明
2. 如何查询，选择可的出示可信声明
3. 如何在保护隐私的情况下通过可信声明验证
4. 可信声明格式，JSON-LD 标识实体
5. 可信声明支持的算法簇
6. 相关的协议有
    1. DID 协议
    2. 语义网的协议
    3. 信用积分
    4. 数据交换

区块链适配器

区块链适配器需要打通和各个链的接口，包含以下内容：

1. 服务端可复用模块
    1. 接口的定义
    2. 数据格式定义
    3. 流程的定义，实现协议
    4. 区块链链接模块，支持各个链
    5. 加解密算法支持
2. SDK支持
    1. RN，Flutter，Android，iOS
    2. 加解密算法支持
    3. 可信声明查询，出示，保密等机制
3. 端到端加密通信机制
    1. 非对称加密加密对称加密秘钥
    2. 对称加密加密原文

接口和数据格式定义

此处定义的接口和协议与实现无关，主要用伪代码描述相关概念。

可信声明相关

详情请查看可信声明协议

函数作用	函数定义	说明
声明存证	bool Commit(byte[] claimId, byte[] committerId, byte[] ownerId);	在存证合约中，声明的唯一标识就是声明的ID，这个将被作为第一个参数，committerId就是存证人的身份，ownerId是可信声明属主的身份。当且仅当该声明没有被存证过，且Commit函数是由committer调用，才能存证成功；否则，存证失败。存证成功后，该声明的状态就是已存证（committed）。
吊销声明	bool Revoke(byte[] claimId, byte[] committerId);	当且仅当声明已被存证，且committerId等于存证人的身份时，才能成功吊销；否则，吊销失败。
存证查询接口	byte[] GetStatus(byte[] claimId);	返回 • 状态：”未存证“， ”已存证“， ”存证被吊销“； • 存证人 ID
可信声明格式	header.payload.signature.blockchain_proof	

可信声明 - header	{
	"alg": "ES256",
	"typ": "JWT"
	}
		• **alg** 指明使用的签名方案。
		• **typ** 格式类型，可以是以下两种值
		○ JWT:不包含区块链证明的可信声明
		○ JWT-X:包含区块链证明的可信声明

可信声明 - payload	{
	"@context": ["https://www.w3.org/2018/credentials/v1",
	"https://blockchain.thoughtworks.cn/credentials/v1"],
	"id": "xyzxyzxyz",
	"ver": "0.7.0",
	"iss": "did:tw:xxx",
	"iat": 1588059342,
	"exp": 1651131342,
	"typ": ["VerifiableCredential", "HealthyCredential"],
	"sub": {
	"id": “did:tw:xxx",
	"healthyStatus": {
	"typ": "HealthyStatus",
	"val": "HEALTHY"
	}
	}
	}
		• **ver** 指明可信声明版本
		• **iss** 可信声明签发者的ID
		• **iat** unix时间戳格式的创建时间
		• **exp** unix时间戳格式的过期时间
		• **jti** 可信声明的唯一标识符
		• **@context** 可信声明内容定义文档的uri，包含每个字段的含义和值的类型的定义，符合JSON-LD协议
		• **clm** 指明了可信声明内容
		• **sub.id** 可信声明接收者的ID
		• **clm-rev** 指明了可信声明是否被吊销
		• **Type** 文档类型

可信声明 - signature	signature := Base64(sign(Base64(header).Base64(payload)))	在构造完Header和Payload部分后，根据JWS标准计算签名。详细的描述在 RFC 7515 Section 5.1。

可信声明 - blockchain_proof	{
	"type":"MerkleProof",
	“tx_hash”:"hash",
	“contract_address": “cintract address",
	“block_height":10,
	“root":"root hash",
	“proofs”:[]
	}
		基于以太坊的一个实现
		除此之外，还需要检查声明存证状态查询接口GetStatus()，如果状态字段为”未存证“，则返回错误。

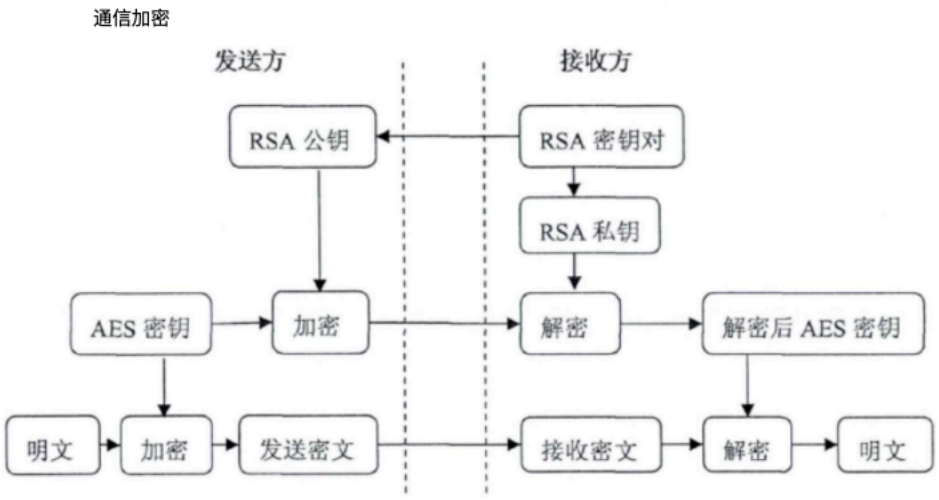
DID相关

详情请查看 TW DID 协议 和 DID 标准

函数作用	函数定义	说明
DID 定义	did:tw:12AE66CDc592e10B60f9097a7b0D3C59fce29876	• did -> URL scheme identifier • tw -> Identifier for the DID method • 12AE66CDc592e10B60f9097a7b0D3C59fce29876 -> DID method-specific identifier is a checksumed address like Etheruem.
DID Documents	{ "@context": "https://www.w3.org/ns/did/v1", "id": "did:tw:xxx", "created":"2020-04-20T10:27:27.326Z", "publicKeys": [{ "id": "did:tw:xxx#keys-1", "type": "Secp256k1", "publicKey": "xxxxx" }] }	• 描述了 DID 主体的一组数据。包含了 DID 标识符，公钥信息，服务端口以及签名等信息。有时，也会包含描述 DID 主体的属性或声明信息。以 JSON-LD 方式呈现。 • 可以有多个控制的公钥
生成 DID	DID create()	• DID是一种URI，由每个实体自己生成成 • 生成算法需要保证碰撞概率非常低
注册 DID	POST /dids { "did": "did:tw:xxx", "publicKey": "xxx" , "signature": "xxx" }	• 需要在区块链上注册之后才能生效，不能重复注册 • 需要有校验机制
读取 DID	GET /dids/{did}	• Returns 200 with Document Object in JSON-LD format • 返回值参考DID Documents
更新 DID，改变控制人	PUT /dids/{did} { "publicKey": "xxx", "signatures": [{ "id":"did:tw:owner#keys-1", "type":"Secp256k1", "signature":"owner" }, { “id":"did:tw:new#keys-2", "type":"Secp256k1", “signature”:”new signer” }] }	Returns `200` with Document Object { "@context": "https://www.w3.org/ns/did/v1", "id": “did:tw:xxx", "created":"2020-04-20T10:27:27.326Z", "updated":"2020-04-20T12:30:20.326Z", "publicKeys": [{ "id": “did:tw:xxx#keys-1", "type": "Secp256k1", "publicKey": “xxx" }] }
设置恢复人	暂无	
注销	DELETE /dids/{did} { "signatures": [{ "id":"did:tw:xxx#keys-2", "type":"Secp256k1", “signature":"xxxx" }] }	Returns `204`
增加附加属性	暂无	

区块链适配器加密通讯机制

区块链适配器可以被当做模块或者插件嵌入到手机或者服务端，适配器之间可以实现通信加密机制保护明文不被抓包。下图的RSA和AES算法都是可以替换的。



算法支持

可信声明的签名需要支持多种算法，适应不同的要求，列举如下：

- ES224 - ECDSA with SHA224,
- ES256 - ECDSA with SHA256,
- ES384 - ECDSA with SHA384,
- ES512 - ECDSA with SHA512,
- ES3-224 - ECDSA with SHA3 224
- ES3-256 - ECDSA with SHA3 256
- ES3-384 - ECDSA with SHA3 384
- ES3-512 - ECDSA with SHA3 512
- ER160 - ECDSA with RIPEMD160
- SM - SM2 with SM3 （国密）
- EDS512 - EDDSA with SHA256

服务端模块定义

下面绿色的表示建议实现的，黑色表示优先级可以低一些。

1. 服务端应实现一个通用的区块链适配器
    1. 可以方便的作为库嵌入到现有的系统中（Java SDK）
    2. 可以提供restful接口，包装库
    3. 可以提供多语言支持，比如 Java，JS
    4. 可以提供 docker 运行环境
2. 需要实现上述文档描述的通用接口
    1. DID相关
    2. 可信声明相关
3. 可以配置的点：
    1. 接入哪个链
        1. quorum
        2. 其他
    2. 配置链的节点URL
    3. 配置使用的算法
    4. 通信加密

智能合约模块定义

1. 实现 DID 合约
2. 实现存证合约
3. 需要针对不同的链实现合约所描述内容
4. 需要考虑跨链如何支持

手机端和 Web 端模块定义

1. 前端应实现一个通用的区块链适配器
    1. 可以方便的作为库嵌入到现有的系统中（Dart SDK）
    2. 可以提供多语言支持

2. 需要实现上述文档描述的通用接口
    1. DID相关
    2. 可信声明相关
3. 可以配置的点：
    1. 可以配置连接到服务端适配器
    2. 可以配置连接到不同的节点
    3. 配置使用的算法
4. 通信加密协议