基于区块链的

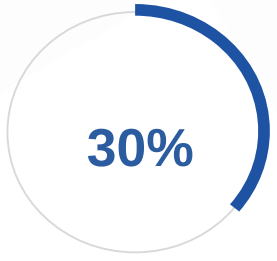
实体身份认证 与 可信数据交换

WeBank

微众愿景：科技.普惠.连接

大众消费者　　小微企业

腾讯持股



腾讯发起设立
中国内地首家互联网银行

IT人员占比

科技及创新
驱动的银行

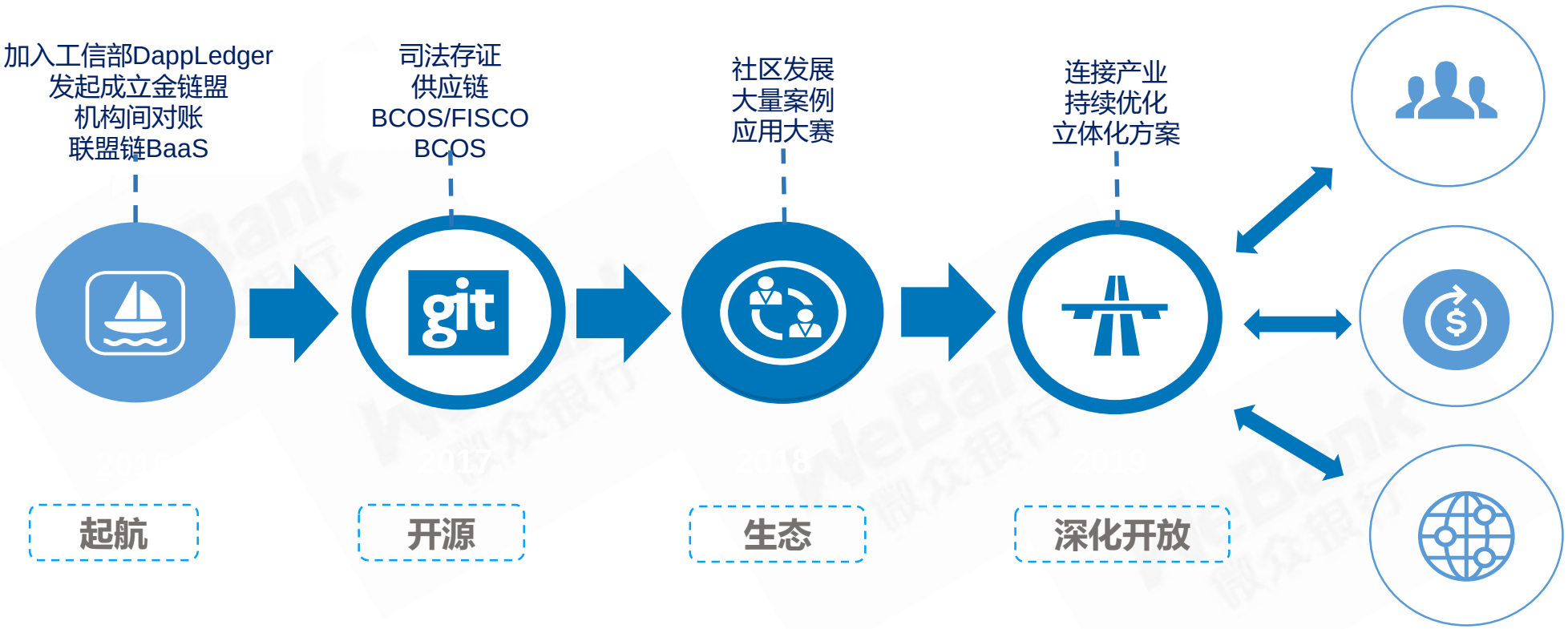
运营

提供随时随地
金融服务的银行

最快项目投产时间

基于模块化
构建的敏捷银行

区块链开源时间线



50+代码仓库	数十万行代码	20万字文档
多领域实体应用落地	500+参与机构	10000+社区成员

一揽子开源解决方案


业务可行，持续创新
对等可信，高效合作
银银合作
供应链
新零售
文化版权
司法存证
智慧城市
更多…
More…
身份标识
WeIdentity
WeEvent
消息协作
WeBASE
中间件
FISCO BCOS
联盟链技术开源社区
高安全性
隐私保护
大规模网络
海量数据
More…

痛点：现阶段数据交换面临的问题

数据孤岛

- 敏感数据不能离开机房
- PDF，视频文件怎么发给对方
- 数据范围和维度各有差异
- 如何与对方共同分析数据
- 数据交易是否合规

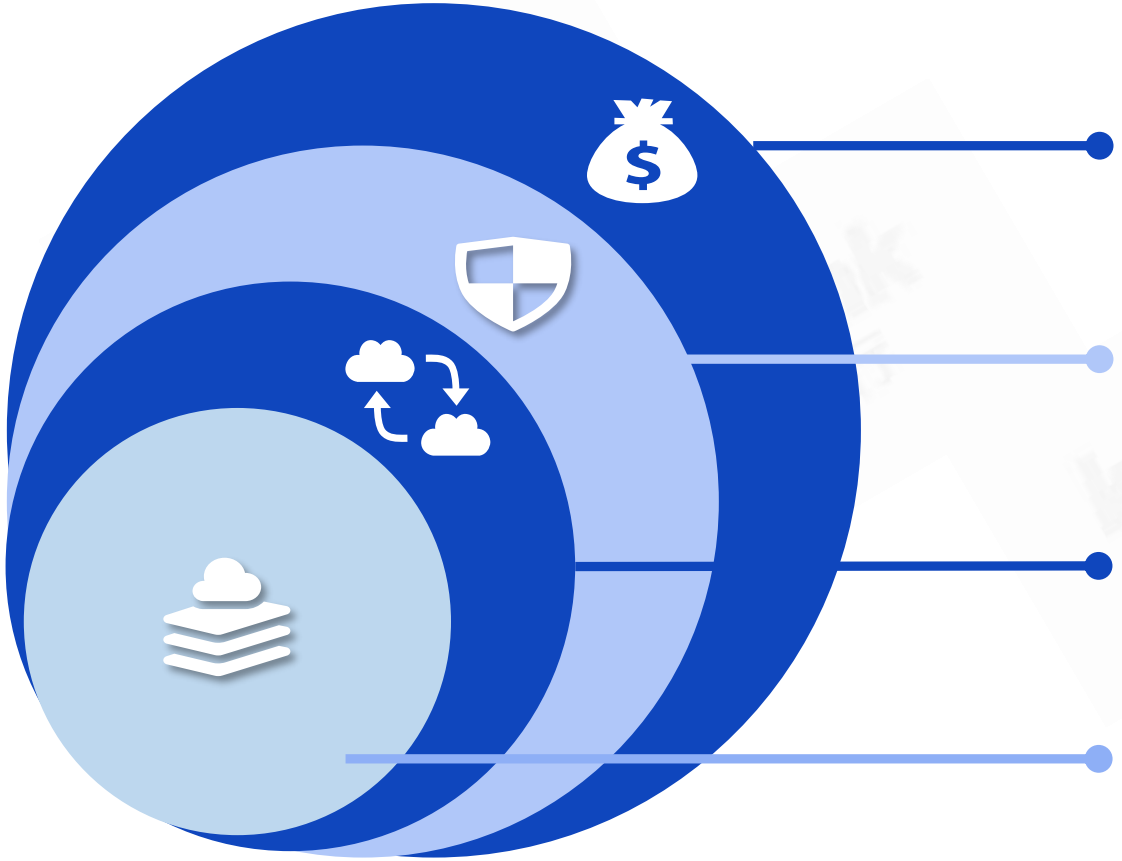
….

- 数据被在不知情的情况下被收集
- 数据的使用没有经过授权
- 数据复制和分析范围并不受控
- 数据分析的结果用于针对个人
- 数据交易对所有者无回报

….

数据滥用

数据管理和交换要考虑的问题



如何合规的获取经济利益

证明付费，访问收费，计算付费，流转付费

如何验证数据合法性?

增信，存证，确权，验证

如何授权和访问?

访问权限，按需获取，最小披露

数据存在哪里?

链上，链下，云，IPFS?

身份和数据的概念

聚焦个人的场景

- **知道是谁，符合要求，知道全部数据**

证明**小明**是**本校计算机系学生，本科学历，男，21岁**，可以办证

- **知道是谁，符合要求，知道部分数据**

证明**小明**是**本校学生且是21岁**,所以可以办证

- **知道是谁，且符合要求，不知道任何数据**

证明**小明是本校学生且已经年满18岁**符合办理大学借书证的资格

- **只知道符合要求，不知道是谁**

证明**某人**已经满足办理大学借书证的资格(零知识)

行业发展情况

W3C（The World Wide Web Consortium ）提出了DID(Decentralized Identifier)的方案，核心是多中心协同和由自己完全掌控自己的数据。

Decentralized Identity Foundation（DIF） 是推动DID方案的开源基金会，制定具体技术标准，促进互交流。

这个基金会目前已经有70家会员，包括IT巨头IBM，微软，NEC，埃森哲；区块链组织超级账本，R3，以太坊企业联盟，ID解决方案组织evernym，sovrin，SecureKey，Validated ID，Anth0，LifeID；金融机构MasterCard，微众银行等等。DIF的多个成员都有自己DID的产品，包括IBM有和银行合作推出的Trust Identity，微众银行有WeIdentity，超级账本有Hyperledger Indy，Sovrin是Indy项目的代码贡献者和Sponsor，IBM也是Sovin的成员。

https://identity.foundation/

业界的其他设计

uPort

- Selective Disclosure Request Flow
- Off-Chain Claims
- uPort-Compliant JWT's
- uPort Public Key Infrastructure
- uPort DID Resolver
- DID Document Storage (IPFS)
- On-Chain Claims Registry (ERC 780)
- Ethereum Account Abstraction
- *(Optional) uPort Proxy (Identity)*
- *(Optional) Access Control Layer*
- Ethereum

ERC 725

ERC 725 Compliant Contract

(On-Chain Claims, Identity, and Access Control)

ERC 735

```
contract ERC735 {

    event ClaimRequested(uint256 indexed
    event ClaimAdded(bytes32 indexed cla
    event ClaimRemoved(bytes32 indexed c
    event ClaimChanged(bytes32 indexed c

    struct Claim {
        uint256 topic;
        uint256 scheme;
        address issuer; // msg.sender
        bytes signature; // this.address
        bytes data;
        string uri;
    }

    function getClaim(bytes32 _claimId)
    function getClaimIdsByTopic(uint256
    function addClaim(uint256 _topic, ui
    changeClaim(bytes32 _claimId, uint25
```

ERC 780

setClaim

Used by an `issuer` to set the claim `value` with

```
function setClaim(address subject, bytes32
```

setSelfClaim

Convenience function for an `issuer` to set a clai

```
function setSelfClaim(bytes32 key, bytes32
```

getClaim

Used by anyone to get a specific claim.

```
function getClaim(address issuer, address s
```

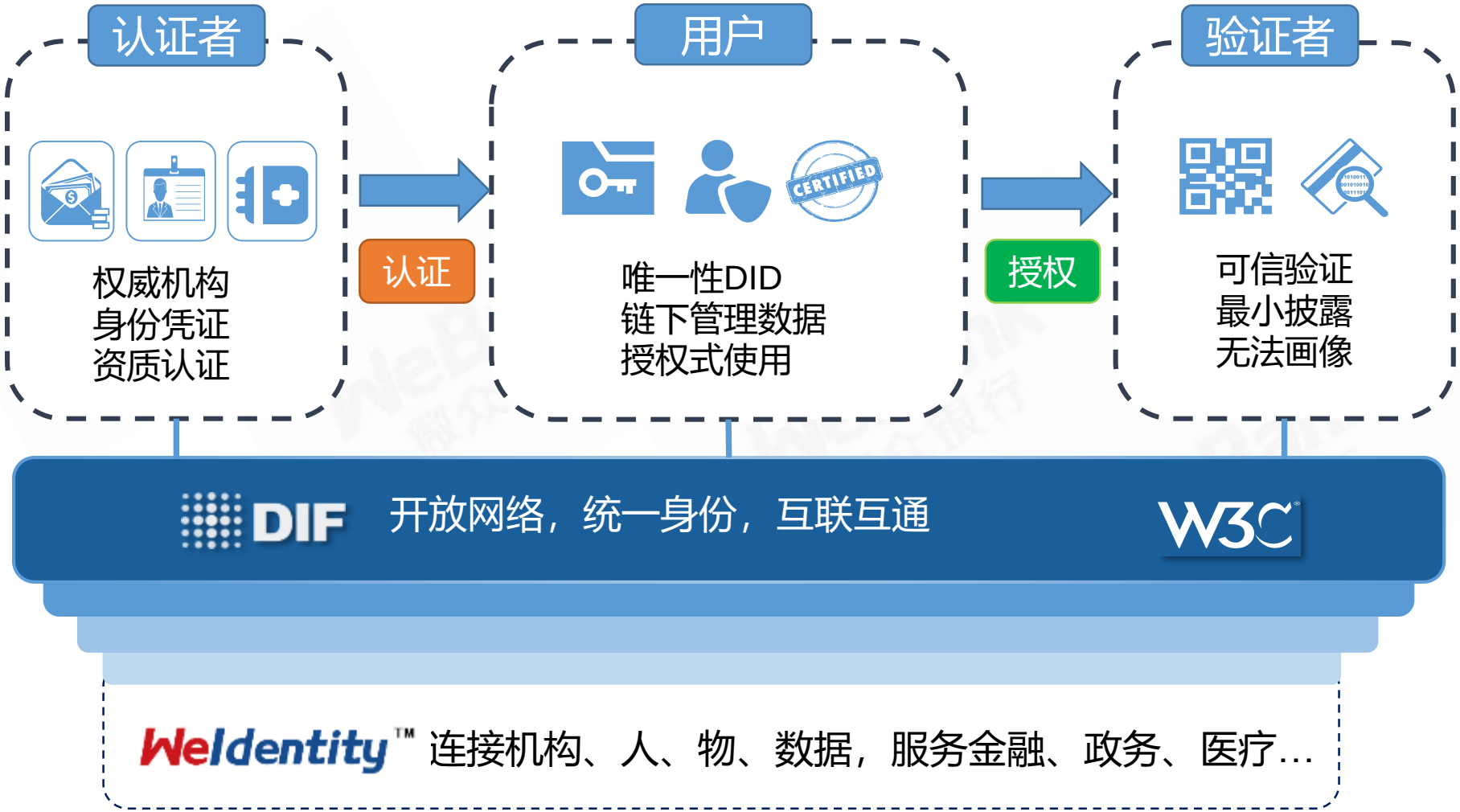
removeClaim

Used by an `issuer` to remove a claim it has mac

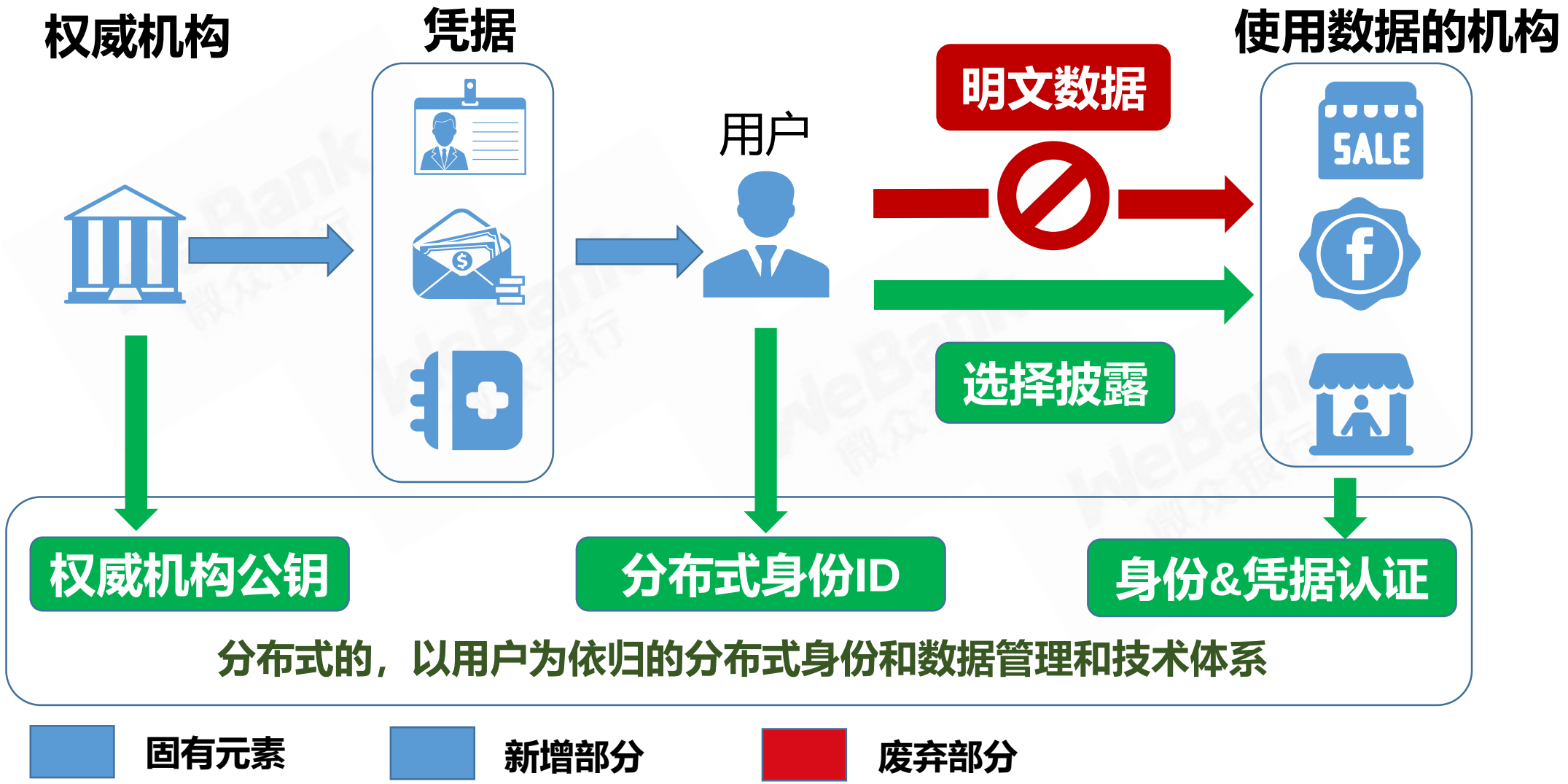
```
issuer, addres
```

ERC1056：更轻的725，ERC734：Key Manager Standard

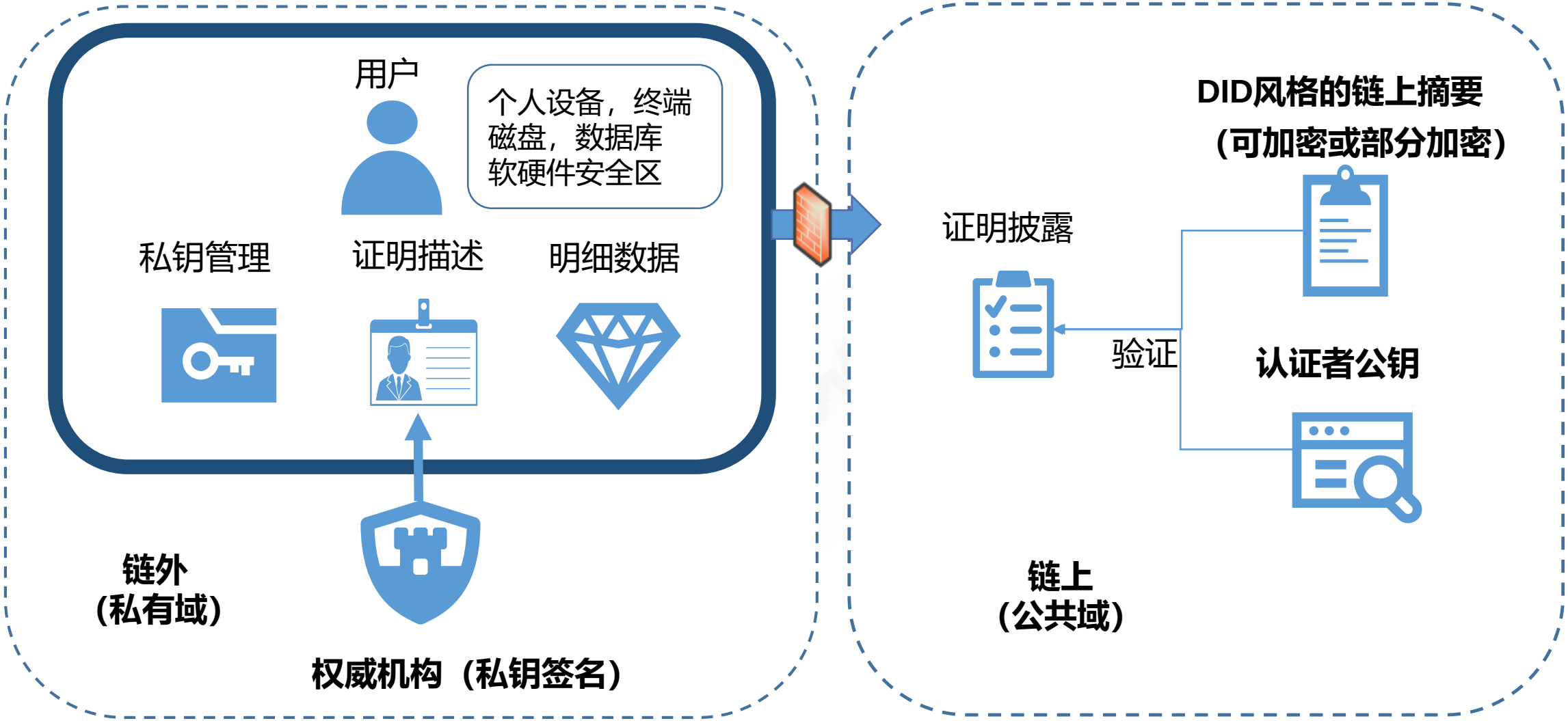
分布式身份和可信数据交换



分布式身份的“变”和”不变”


权威机构
凭据
用户
明文数据
使用数据的机构
SALE
选择披露
权威机构公钥
分布式身份ID
身份&凭据认证
分布式的，以用户为依归的分布式身份和数据管理和技术体系
固有元素
新增部分
废弃部分

身份标识数据的使用方式



理解DID


四要素
刷脸
面签
公私钥，地址
UUID，Ecode，
OID
格式，证明
存储，管理
策略，样式
授权
场景，隐私
KYC
标识
凭据
披露
验证
DID是多方沟通的【协议】
WeID是DID的一个参考实现
区块链A
区块链B
身份场景A
物联场景B

基本设定

管理视角

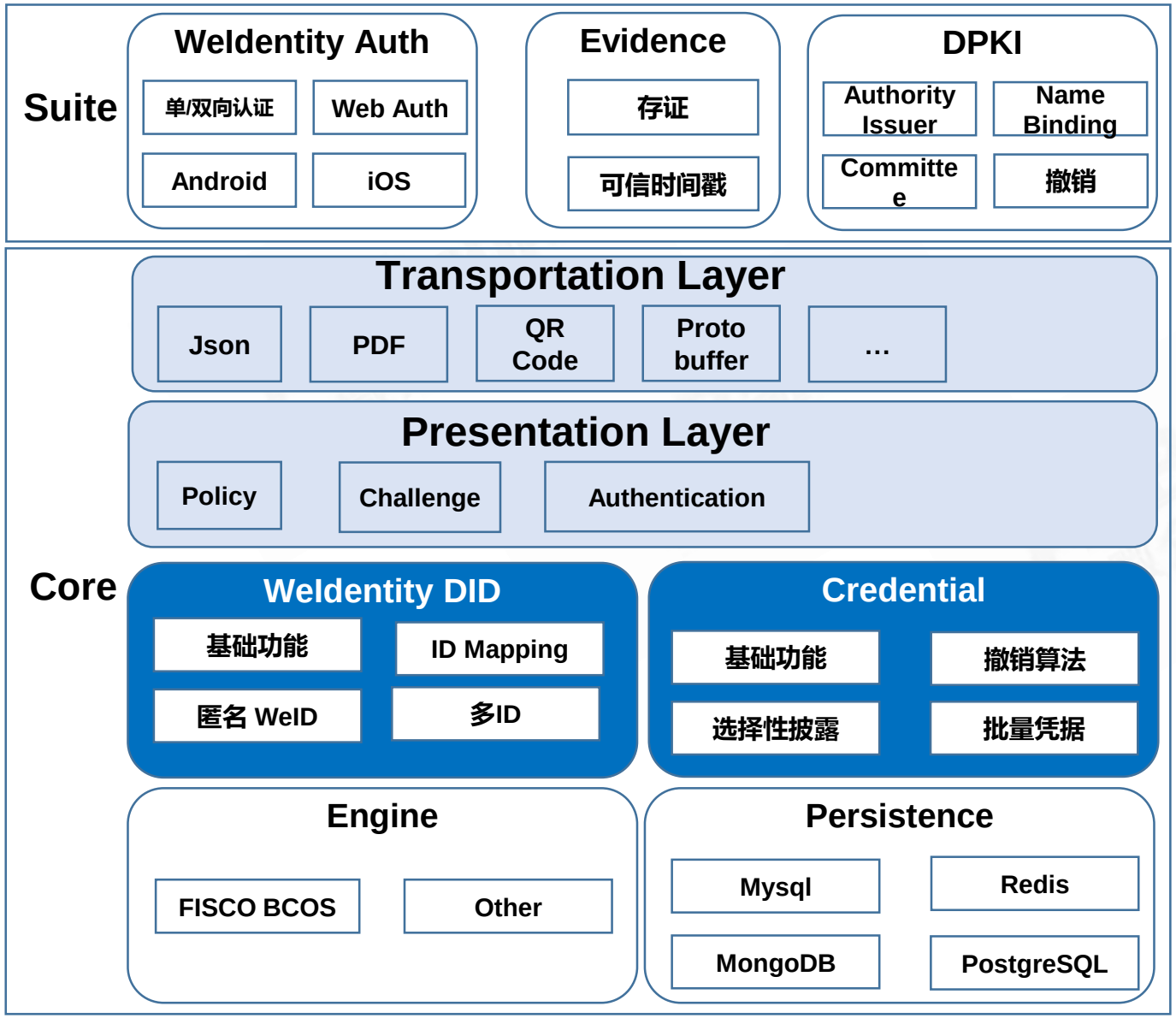
1. DID是一**个难以重复的全局唯一ID**
2. 链上维护数据索引，公钥信息，连接点描述等，**没有详细数据**
3. 发证方有一个DID，**公布了自己的公钥**，供验证方来验证，被**“信任根”**认证过的发证方是权威的。

用户视角

1. 一个自然人可以**有多个DID**
2. 每个DID都可以对应一些**“证书”或数据**，这些证书由有资质的机构发放，或者由用户自己生成
3. 用户可以通过交互协议将证书和数据**选择性披露**给验证方

WeIdentity整体架构

WeIdentity 整体架构



WeIdentity DID: 链上身份标识相关功能。

Credential: 凭证相关功能。

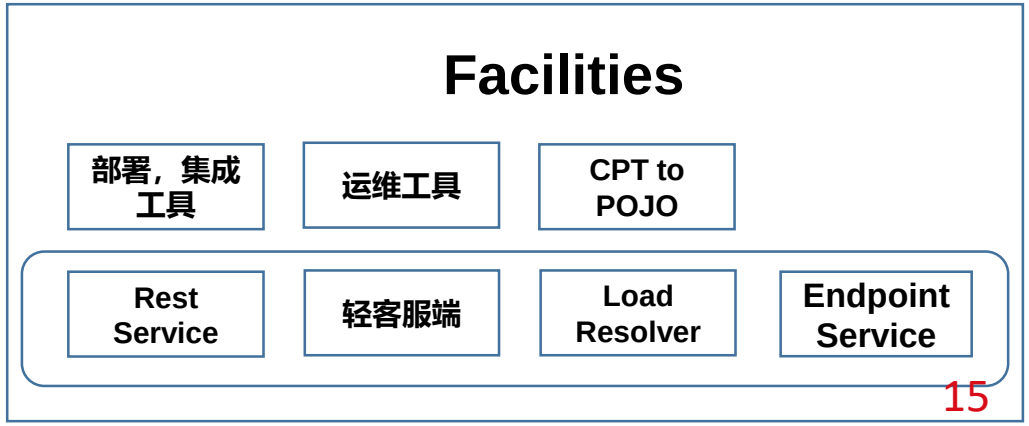
Weld Auth: 帮助Weld 之间进行认证的模块。

Evidence: Credential 链上的存证模块。

DKMS: 多中心的密匙管理系统。

DPKI: Distributed PKI。

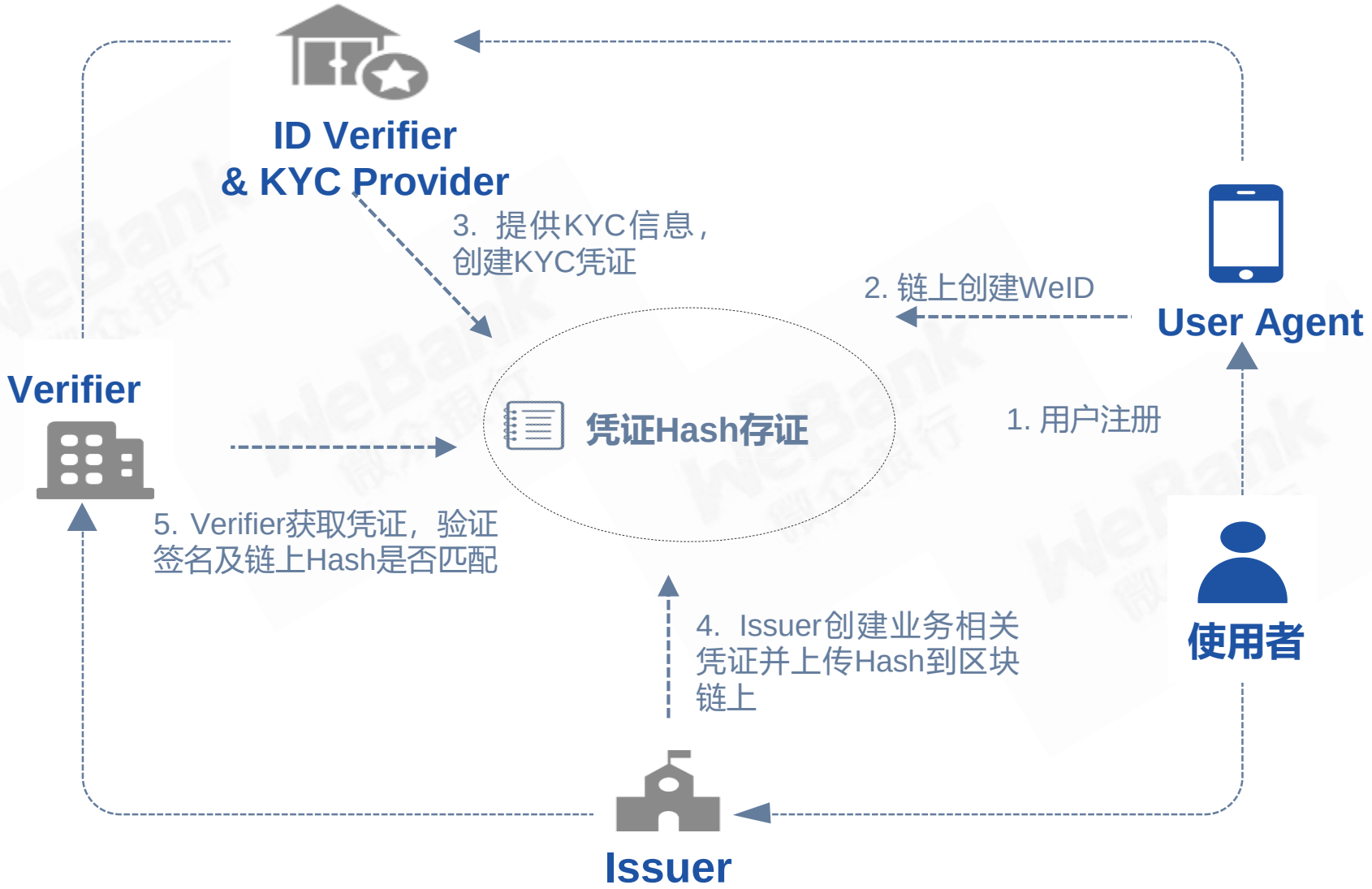
Engine: 适配不同底层平台的引擎。



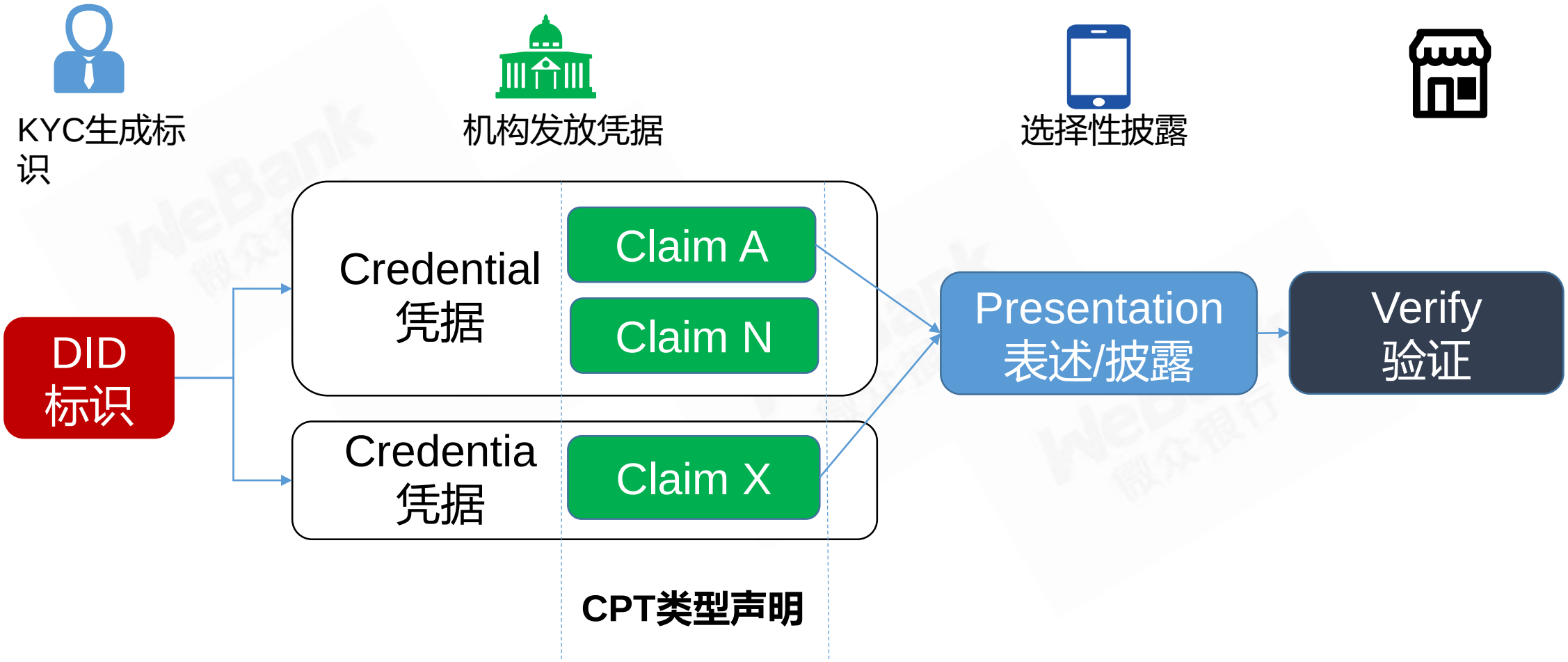
链上角色

角色	说明
Entity	WeIdentity Document或Credential描述的实体，即拥有WeIdentity DID的人或者物，可以是人或者物。
Issuer	对Claim进行认证的权威机构或者可信机构，例如下发驾照的交通局。
Verifier	使用凭证的第三方，会验证这个凭证是否经过权威机构认证，例如车管所办理业务时需要用户提供驾照，这时车管所就是Verifier。
User Agent	1.提供面向用户使用的APP或者终端。 2.用户私钥的托管机构，例如身份证明局的客服端凭证钱包APP，也可以是某个客服端digital wallet，或者是某个云上托管私钥的digital wallet。 3.完成对用户的KYC（跟进场景需要）。 4.代理用户，与其他用户或者机构交互。
Data Repository	数据的托管机构，例如用户把Credential托管在某个的客服端凭证钱包APP。

角色间关系


ID Verifier
& KYC Provider
3. 提供KYC信息，
创建KYC凭证
2. 链上创建WeID
User Agent
Verifier
凭证Hash存证
1. 用户注册
5. Verifier获取凭证，验证
签名及链上Hash是否匹配
4. Issuer创建业务相关
凭证并上传Hash到区块
链上
使用者
Issuer

基本数据的关系


KYC生成标识
机构发放凭据
选择性披露
DID
标识
Credential
凭据
Claim A
Claim N
Credentia
凭据
Claim X
CPT类型声明
Presentation
表述/披露
Verify
验证

WeIdentity基本要素

数据模型

◆ 成熟的技术:实现数据的验证和确权

- **哈希(HASH):**表示大量数据的唯一摘要值。原数据的少量更改会在哈希值中产生不可预知的大量更改,可以作为数据的验证凭据

- **数字签名**：信息的发送者（掌握私钥）能产生的别人无法伪造的一段数字串，且可以通过其公布出去的公钥验证是由他发送。

各种数据原文

账目，音视频，证书，合同订单，医疗记录

HASH摘要

HASH:

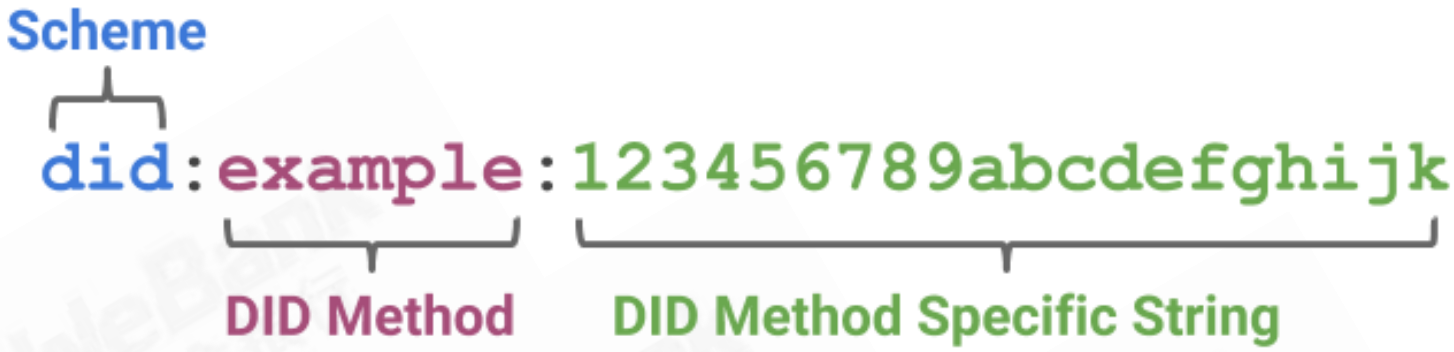
完整性,正确性

签名 → 验签

数字签名:

所有者确权

DID Specifications： 格式定义



- DID完全由创建者通过创建的私钥控制，独立于任何外部组织
- DID由算法保持唯一性
- DID的Identifier部分可以是公钥地址，也不一定是，只要是唯一的
- DID关联一个Document，Document包含DID
- **根据Zooko三角形理论，没有任何标识符能够同时实现易记忆、安全、去中心化。W3C的DID取了后两者**
- W3C 文档 :https://w3c-ccg.github.io/did-spec

链上的WeID & Document

did : weid : 101 : 0xae0b295667a9fd93d5f28d9ec85e40f4cb697bae

```
{
 "@context": ["https://w3id.org/did/v1"],
 "id": "did:weid:101:0xae0b295667a9fd93d5f28d9ec85e40f4cb697bae",
 "publicKey": [{ "id": " did:weid:101:0xae0b295667a9fd93d5f28d9ec85e40f4cb697bae#keys-1",
   "type": "RsaVerificationKey2018",
   "publicKey": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"  }],
 "service": [{  "id": " did:weid:101:0xae0b295667a9fd93d5f28d9ec85e40f4cb697bae#vcr",
   "type": "CredentialRepositoryService",
   "serviceEndpoint": https://repository.example.com/service/8377461 }]
}
```

格式模板标识

WeID

公布的公钥

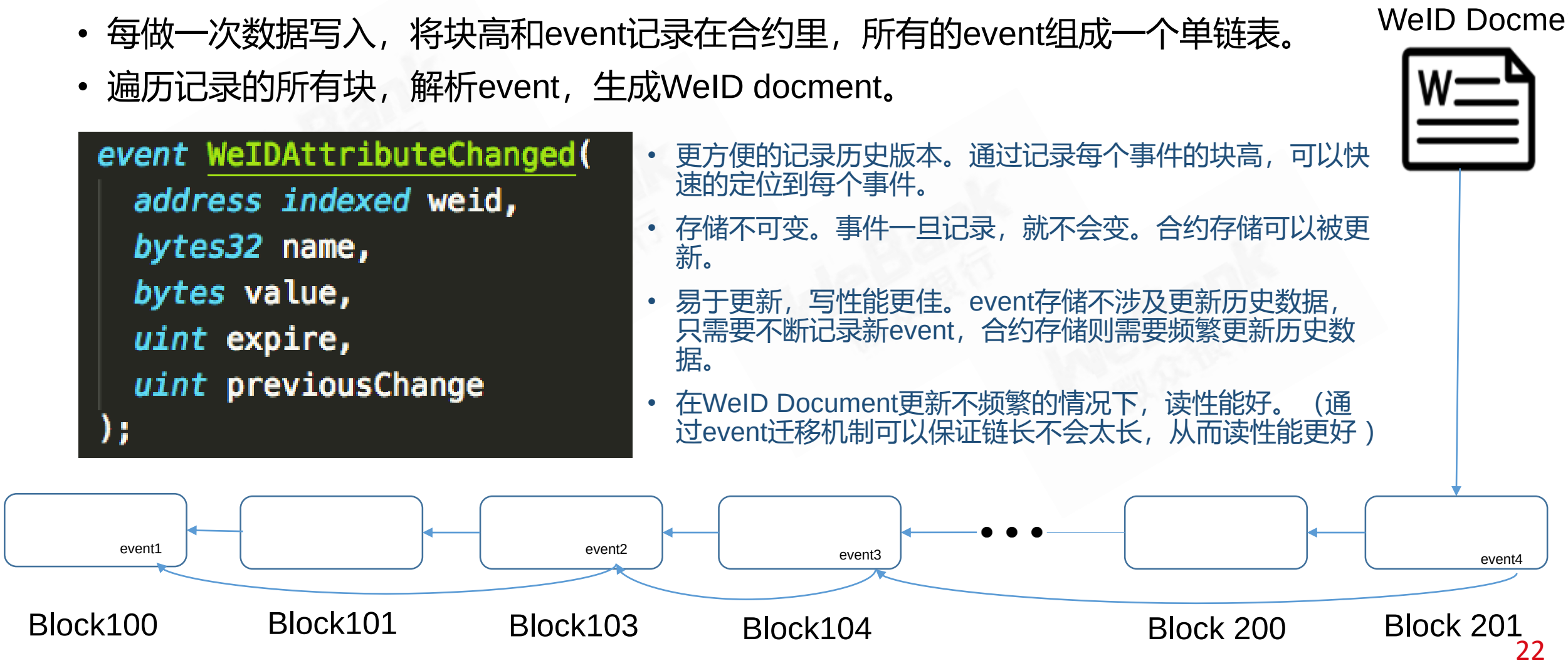
主动暴露的 Endpoint

WeID Document存储结构

- 通过linked event的方式来存储WeID相关的数据。
- 每做一次数据写入，将块高和event记录在合约里，所有的event组成一个单链表。
- 遍历记录的所有块，解析event，生成WeID docment。

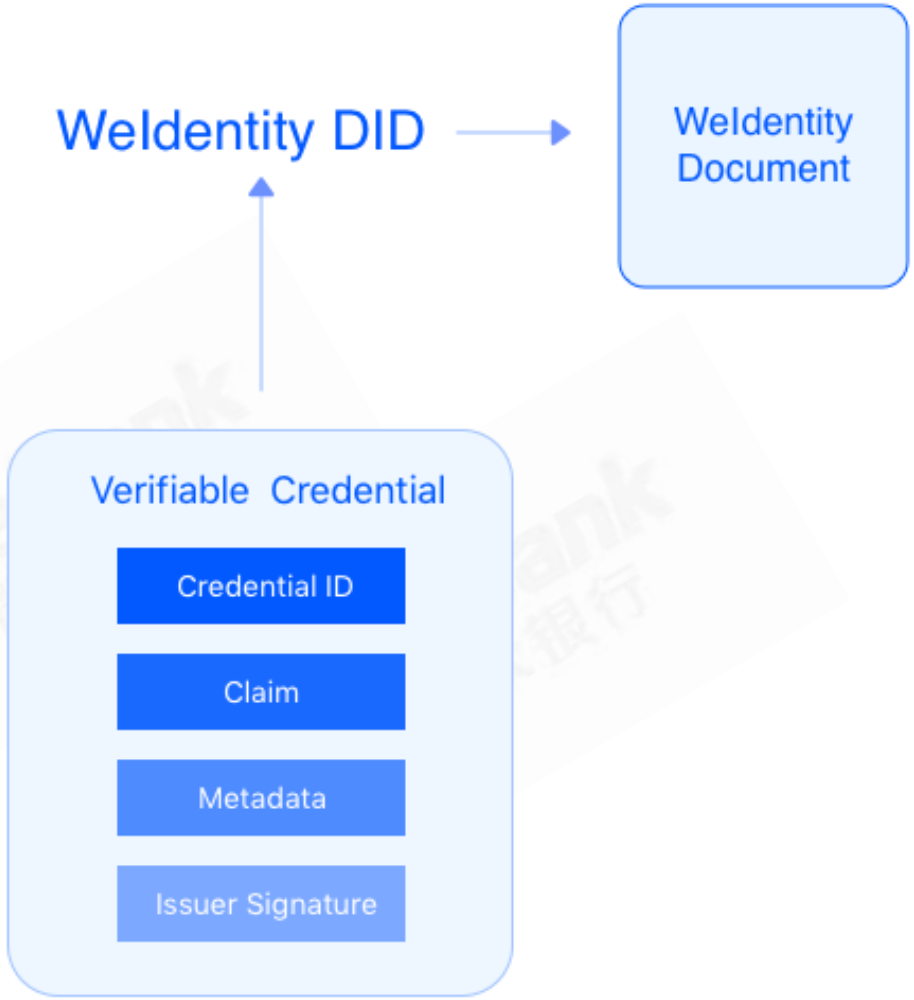
```
event WeIDAttributeChanged(
  address indexed weid,
  bytes32 name,
  bytes value,
  uint expire,
  uint previousChange
);
```

- 更方便的记录历史版本。通过记录每个事件的块高，可以快速的定位到每个事件。
- 存储不可变。事件一旦记录，就不会变。合约存储可以被更新。
- 易于更新，写性能更佳。event存储不涉及更新历史数据，只需要不断记录新event，合约存储则需要频繁更新历史数据。
- 在WeID Document更新不频繁的情况下，读性能好。（通过event迁移机制可以保证链长不会太长，从而读性能更好）



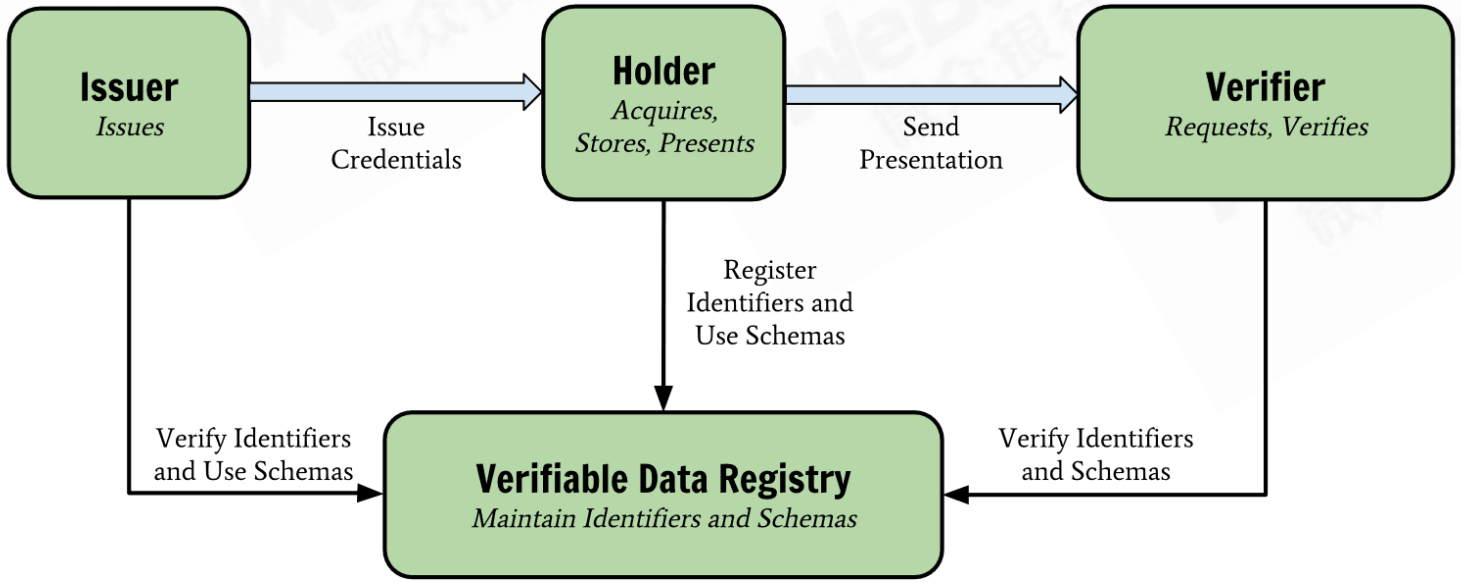
理解DID和Crendential

- Crendential简称“**凭证**”，遵循W3C Verifiable Credential规范的电子凭证，可用来抽象现实世界凭证类的对象，一个Credential可以包含一个或者多个Claim。例如电子驾照，电子学历等；
- Crendential里会包含一个DID定义，表示是哪个DID拥有它。
- 发行者用签名证明这个凭证。

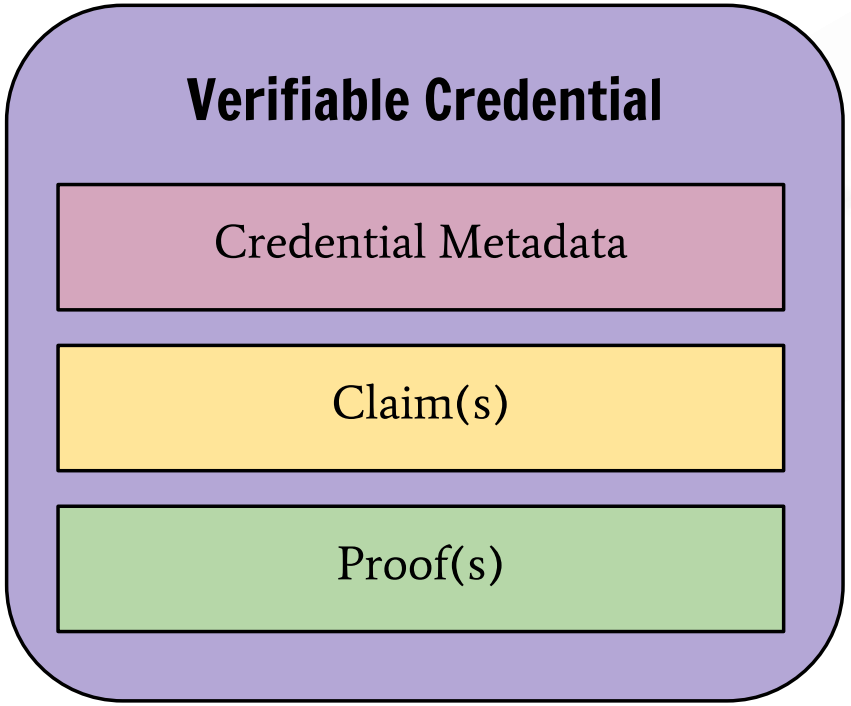


DID Verifiable Claim（可验证Claim）

- e.g. 证书、个人简历、处方、借条，需要验证：
  - Signature (Proof)：Issued from whom? To whom?
  - Metadata：When? Already expired? Revoked?
  - Claim：Data format OK?
- W3C 文档：https://w3c.github.io/vc-data-model/



理解Credential和Claim(s)：证件包模式



- 一个Credential可以有一到多个Claim（声明），每个Claim由相同或不同的issuer发放。
- 为了简洁起见，目前weid实现为一个Credential对应一个Claim，如果多个Claim，即多个Credential。

理解Claim和CPT（ Claim Protocol Type）

- 不同的证件有不同的字段
- 不同的证件由不同（或相同）的机构发放

CPT：类型定义
Claim：对应某一个类型的一个实例

可以定义、注册、注销CPT

Credential 凭证案例 (1/2): Simple ID


```
{
  “@context":
"https://github.com/WeBankFinTech/WeIdentity/blob/master/context/v1",
  "id": "04a3e89d-825a-49fe-b8f5-8ccb9f487a52",
  "cptId": 1000,
  "issuer":
"did:weid:101:0x39e5e6f663ef77409144014ceb063713b65600e7",
  "issuanceDate": "2019-06-19T19:30:44Z",
  "expirationDate": "2019-06-19T23:30:44Z",
  "claim": {
    "gender": "M",
    "name": "Average Joe",
    "age": 22,
    "socialid": 11010219970630 2013
    "id": “did:weid:101:0xae0b295667a9fd93d5f28d9ec85e40f4cb697bae" },
  "proof": {
    "creator": "did:weid:101:0x39e5e6f663ef77409144014ceb063713b65600e7#keys-1",
    "salt": {
      "gender": "ibu7f",
      "name": "el1w8",
      "age": "ajqkr",
      "socialid": "bq6ce",
      "id": "a98bz",
    }
    "created": "2019-06-19T19:30:44Z",
    "type": "EcdsaSignature",
    "signatureValue":
"G7UPiw08P5E9dEcSJEo9zpKu/nsUrpn00xDE+mwDXn9gJEohIlRUX5XTGQB4G1w3yThp6R/2RqjUYkuQTaUXbIU="
  },
  "type": ["VerifiableCredential"]
}
```


Metadata

Claim

Signature

Credential 凭证案例 (2/2): 授权凭证

Metadata

Claim

Signature


```
{
  "@context":
'https://github.com/WeBankFinTech/WeIdentity/blob/master/context/v1",
  "id": "c4f8ca00-7c1b-4ba0-993f-008106075d9c",
  "cptId": 101,
  "issuer":
'did:weid:101:0x39e5e6f663ef77409144014ceb063713b65600e7",
  "issuanceDate": "2019-05-19T09:10:24Z",
  "expirationDate": "2019-05-19T17:10:24Z",
  "claim": {
    "delegator":
'did:weid:101:0xae0b295667a9fd93d5f28d9ec85e40f4cb697bae",
    "receiver":
'did:weid:101:0x0231765e19955fc65133ec8591d73e9136306cd0
',
    "credentialid": ["04a3e89d-825a-49fe-b8f5-8ccb9f487a52”]
  }
 "proof": {
    "creator":
"did:weid:101:0x39e5e6f663ef7740914401
4ceb063713b65600e7#keys-1",
    "salt": {
      "delegator": "p98ba",
      "receiver": "4r7cv",
      "credentialid": "i7obx"
    }
    "created": "2019-05-19T09:10:24Z",
    "type": "EcdsaSignature",
    "signatureValue":
"HHQwJ9eEpyv/BgwtWDveFYAPsKOPtEE
Wt6ieb28PS76pDwlpFKtbh9Ygog8SUPIXU
aWNYS2pLkk4E91hpP8IdbU=“
  }
  "type": ["VerifiableCredential"]
}
```

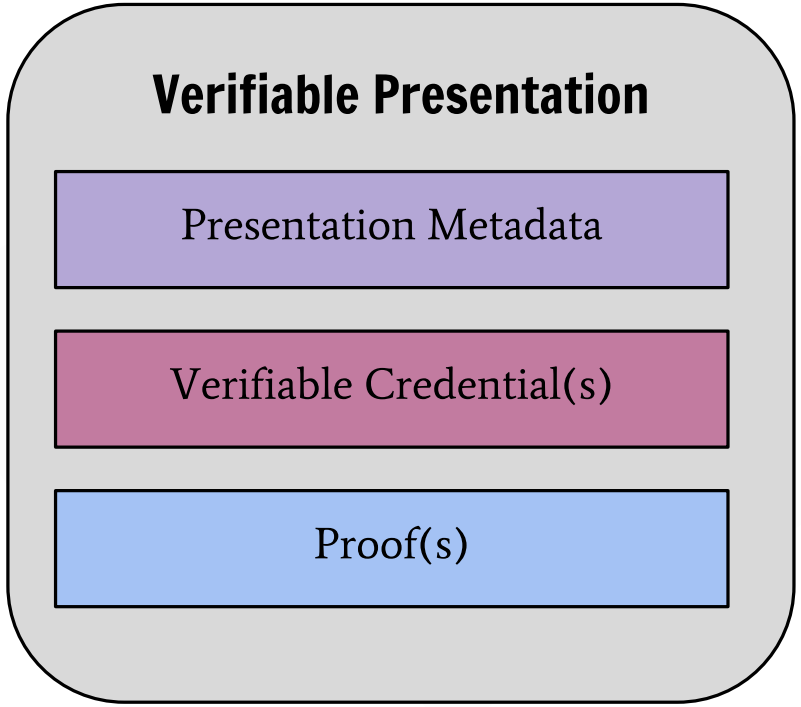
Credential模块

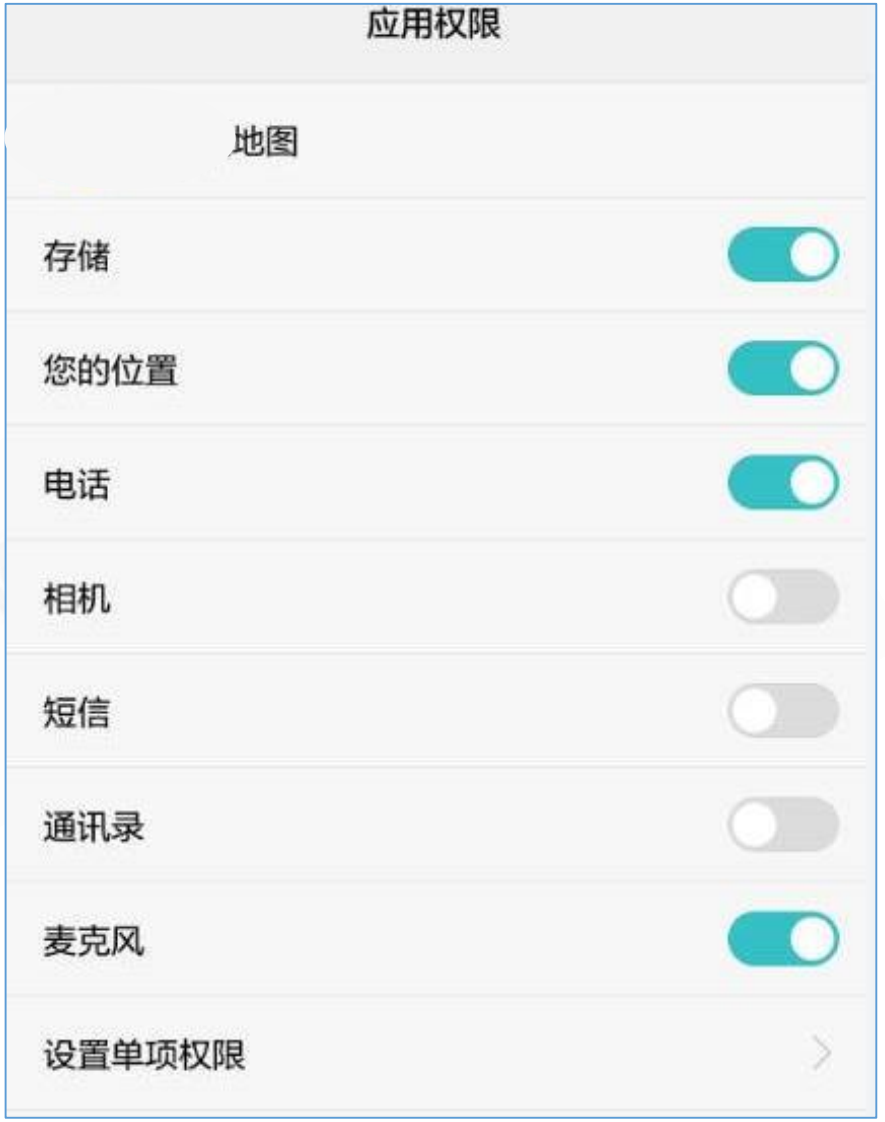
- 基本功能
  - Create：创建一个凭证
  - Revoke：撤销一个凭证
  - Verify：验证一个凭证
  - AddSignature：为一个凭证增添额外的Issuer签名

- 选择性披露：why important?
  - “Age > 18”
  - 只披露凭证的一部分属性，仍然可以验证通过

Presentation与Transportation

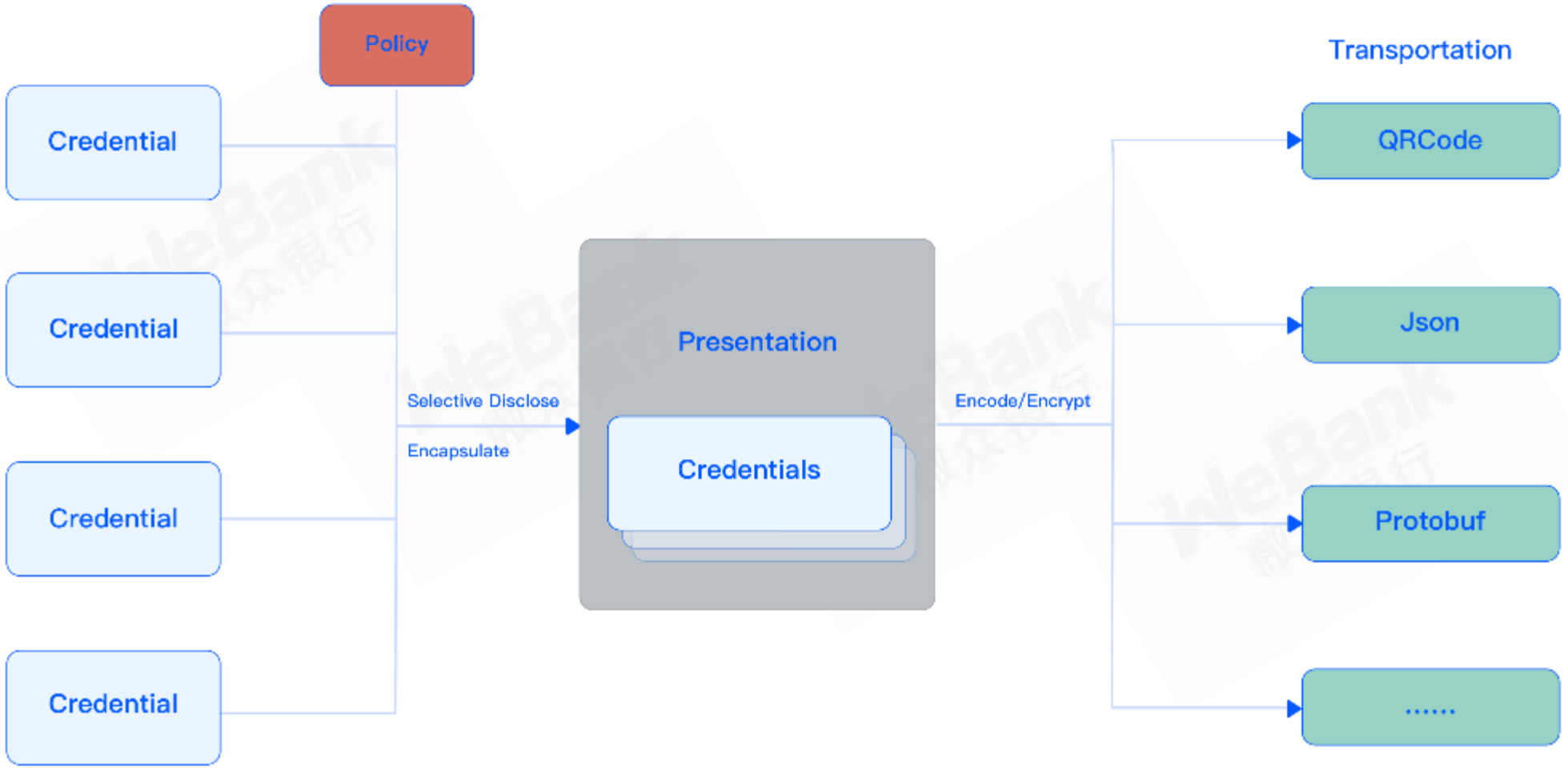
理解Presentation和Credential



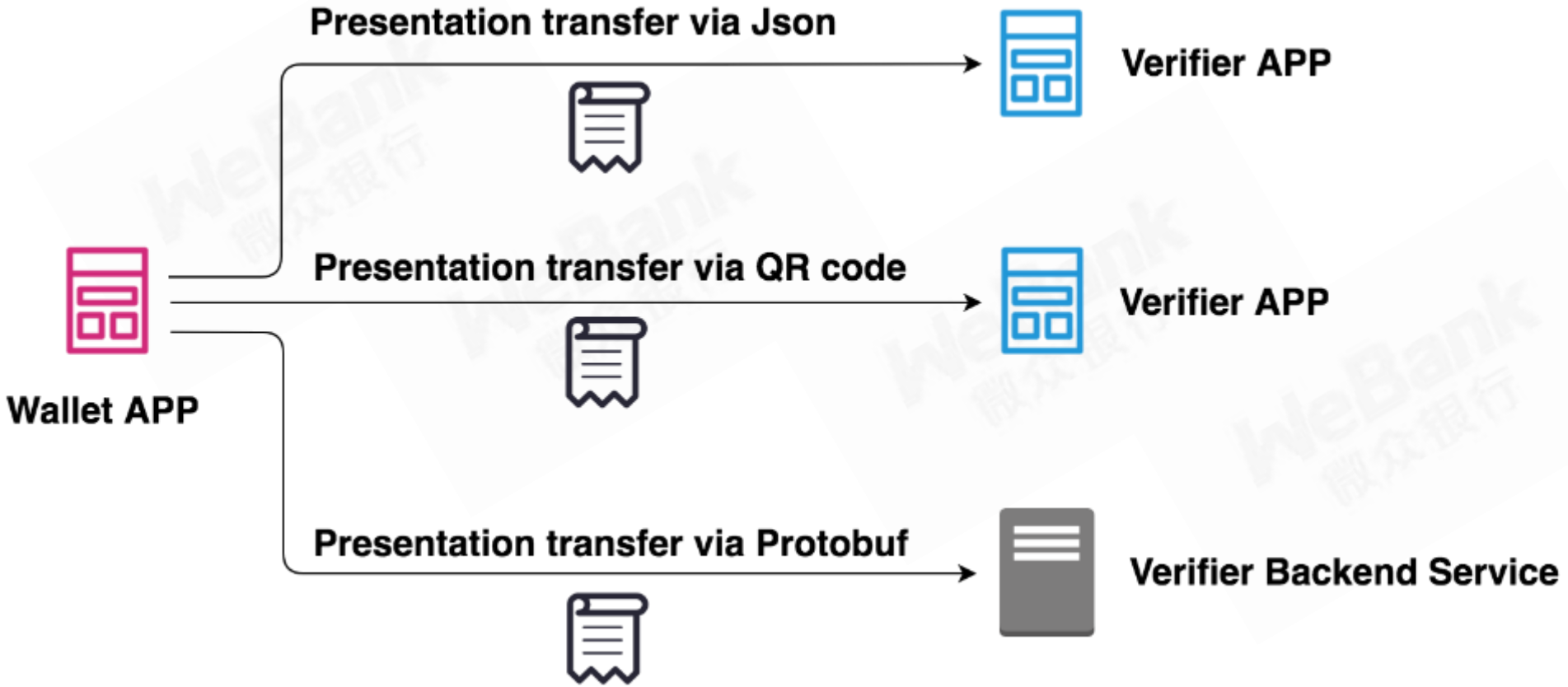


- 遵循W3C Verifiable Presentation规范的凭证打包集。一个**Presentation**可以包括一个或多个Credential
- Policy则是对Presentation的定义和约束

Data Model

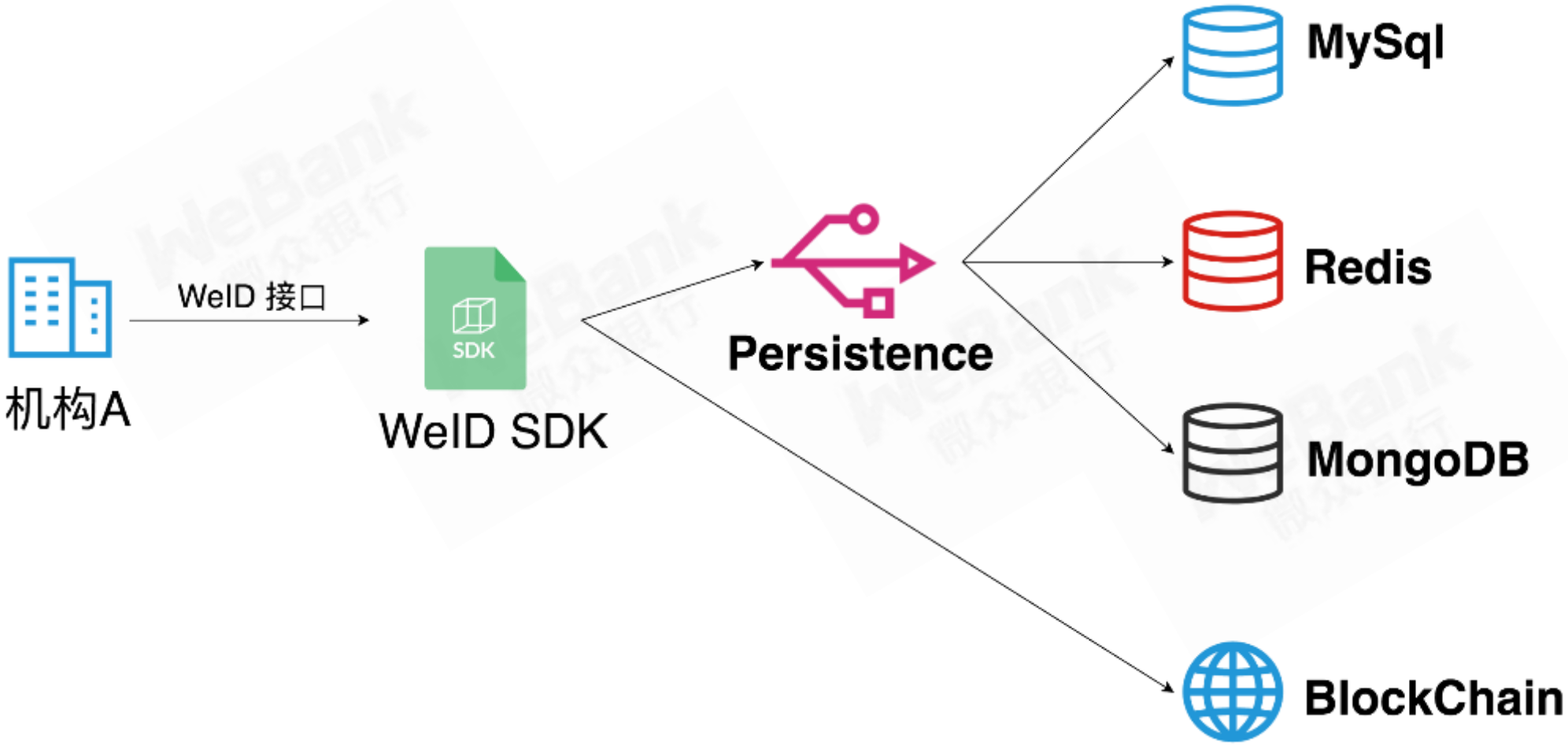

Policy
Credential
Credential
Credential
Credential
Selective Disclose
Encapsulate
Presentation
Credentials
Encode/Encrypt
Transportation
QRCode
Json
Protobuf
.....

Presentation传输（Transportation）方式



相关模块支持

WeIdentity 多种存储模型



用户数据托管

- **便利性：**

拓展用户数据容量，可跨设备，可提供经授权的查询服务

- **安全性：**

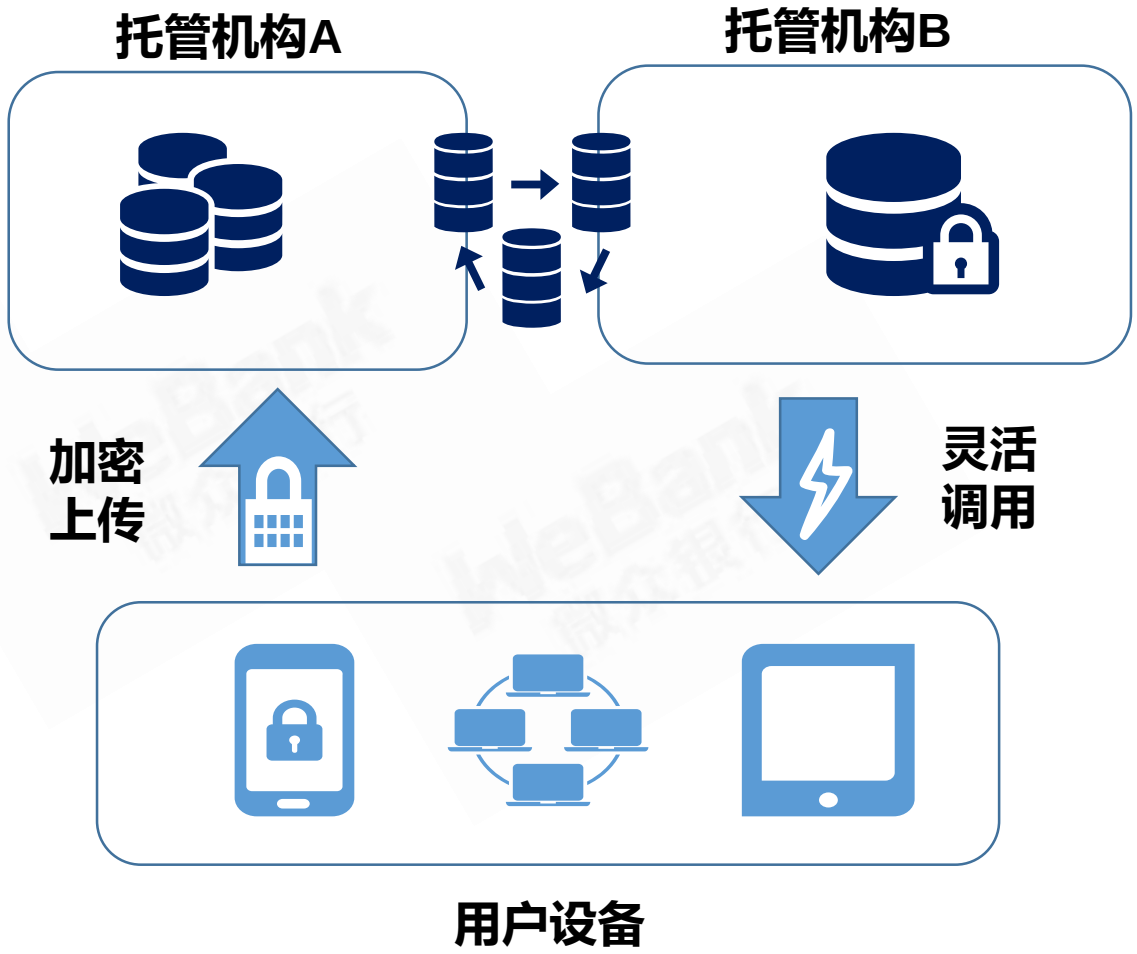
数据加密保存，密钥分片保存

- **可恢复：**

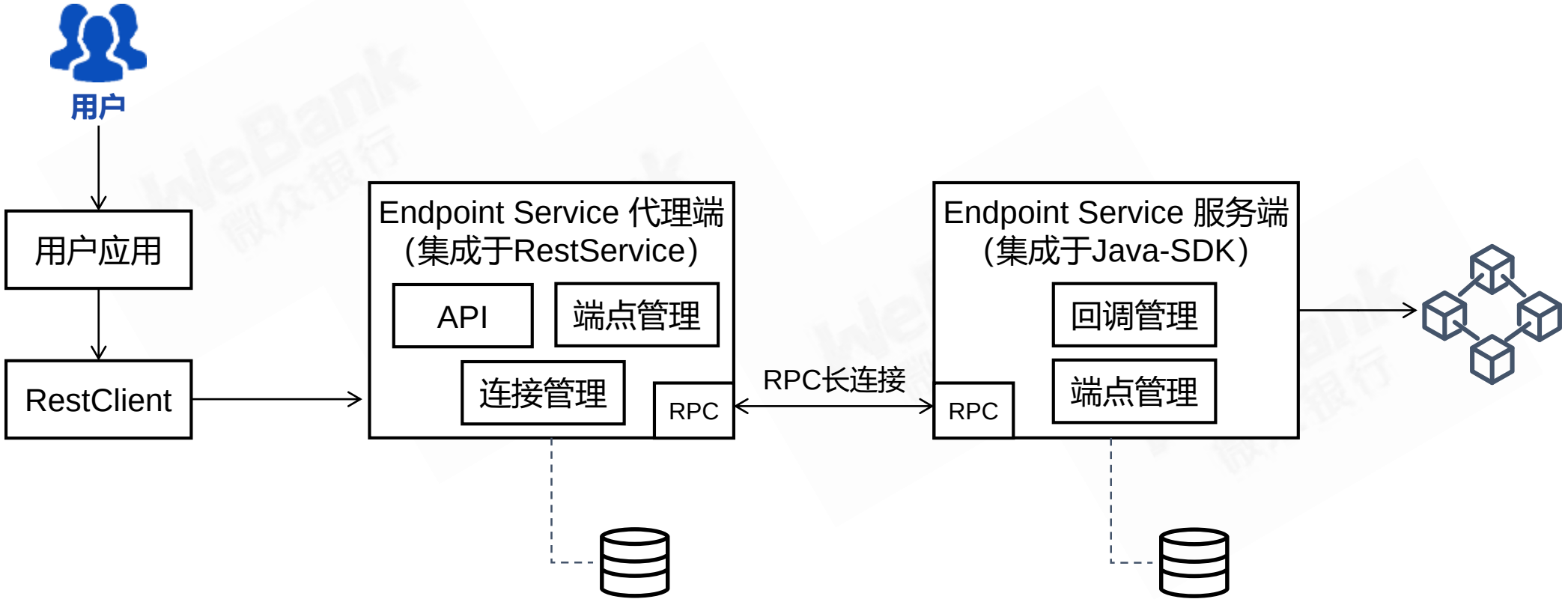
助记词，阈值签名，权威恢复

- **可迁移：**

托管机构不能滥用、控制、禁止用户迁移或删除数据



服务接入Endpoint Service Architecture



3. Endpoint Service模块

- 痛点
  - 在多机构场景下，业务方需要以Endpoint Service的形式，动态暴露自己在服务端的某些业务供他方调用
  - 提供业务接入方动态地暴露Endpoint的标准实现
- 解决方案
  - Endpoint Service代理端（1）
    - 集成于RestService，只需一个额外配置项，便可周期从各个业务方拉取配置
    - 提供业务方多活、断链重连、负载均衡、双向发送机制
  - Endpoint Service服务端（N）
    - 集成于WeID Java SDK，每个业务方只需引入jar包即可使用
    - 只需注册回调方法入Endpoint，配置一个监听端口，便可使用

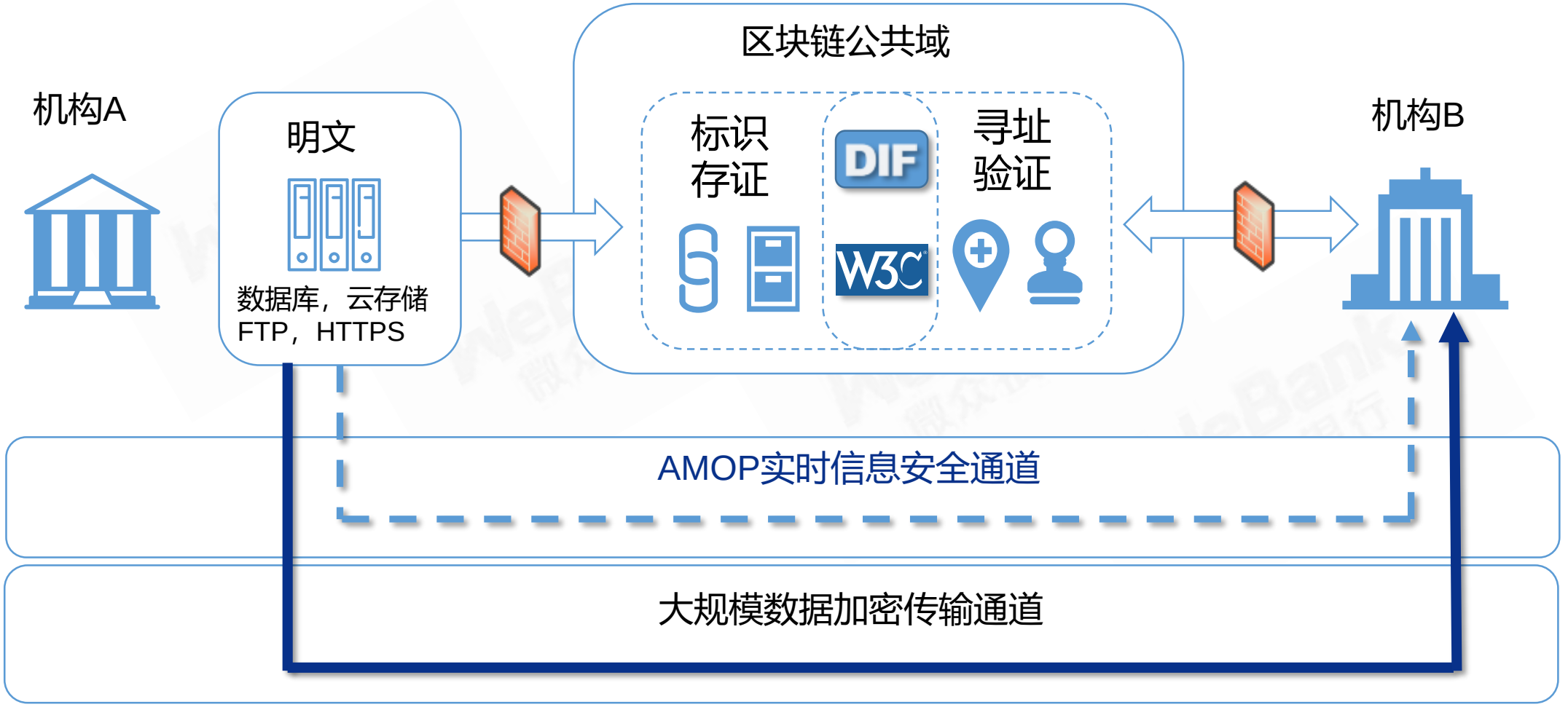
Evidence存证模块

数据的存证可以起到“增信”的作用，强权威机构对其他机构、用户、证书、数据进行存证

- 基本功能
  - Create：为一个凭证创建一个链上存证（生成hash并用机构私钥签名）
  - Verify：验证凭证是否和链上存证相一致
  - AddSignature：为一个链上存证增添额外的签名机构

- 链上存证 = hash + [机构签名...]
  - 机构签名可以由机构的WeID公钥进行验证
  - 一般为权威机构

可信数据传输的概要模型



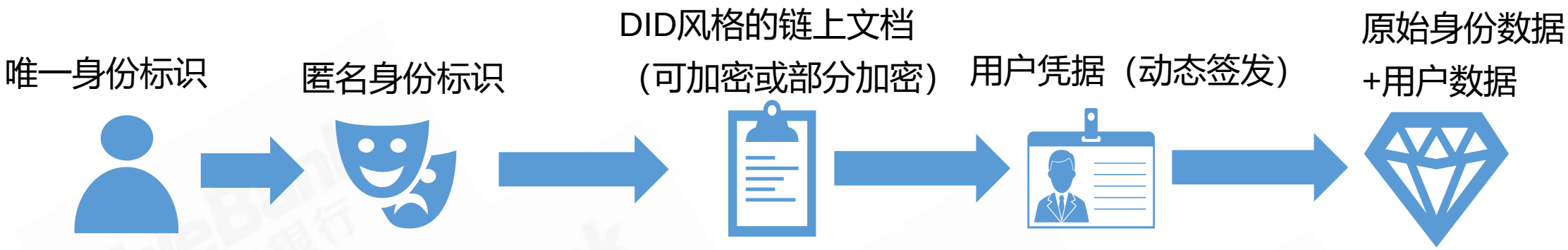
WeID开源项目支持

1. **项目官网：** https://fintech.webank.com/weidentity
2. **在线演示：** https://sandbox.webank.com/weid/#/status
3. **开源Github：** https://github.com/WeBankFinTech/WeIdentity
4. **技术和概念文档：** https://weidentity.readthedocs.io/zh_CN/latest/README.html
5. **Java SDK ：** https://weidentity.readthedocs.io/zh_CN/latest/docs/weidentity-installation.html
6. **智能合约**：https://weidentity.readthedocs.io/projects/javasdk/zh_CN/latest/docs/weidentity-contract-design.html
7. **开发参考样例：** https://github.com/WeBankFinTech/weid-sample

一些关键问题

匿名WeID和多WeID间关联

全链条的数据结构



唯一身份标识： 链上唯一的ID，通过全局一致性算法生成，做为用户在链上的统一身份标识。考虑到安全性，某种场景下，需要给用户一个匿名的一次性伪码ID，和唯一ID能挂钩又能做到匿名和反追踪（可监管）。

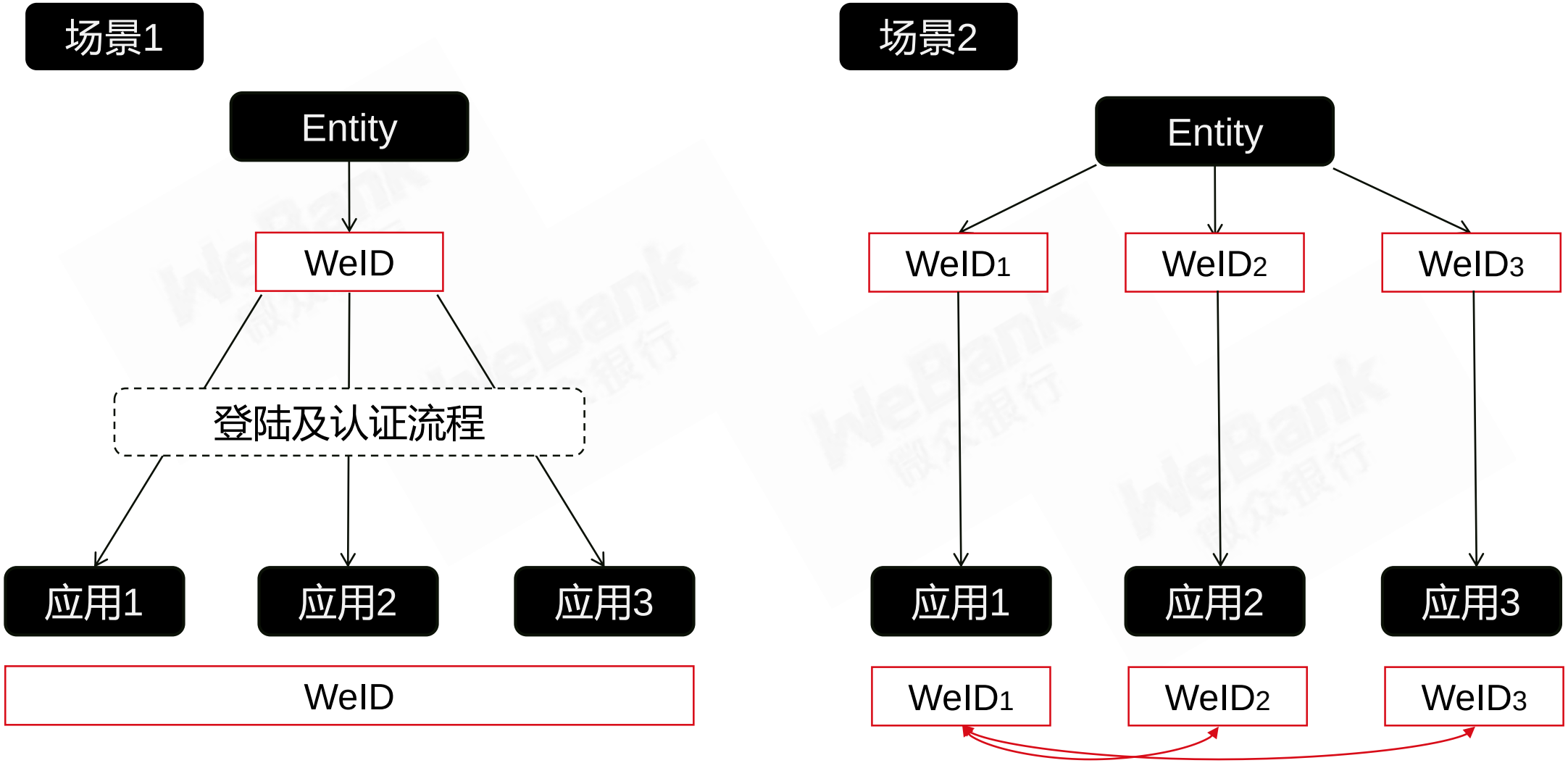
链上实体： 采用DID(W3C Decentralized ID）描述，说明用户都有哪些身份证明信息和此ID挂钩，每种身份证明的具体寻址路由等

原始身份数据： 验证者或发行者需要指导的信息，如四要素，视频，学习经历和成绩单，详细财务信息。高度隐私，一般不泄漏。

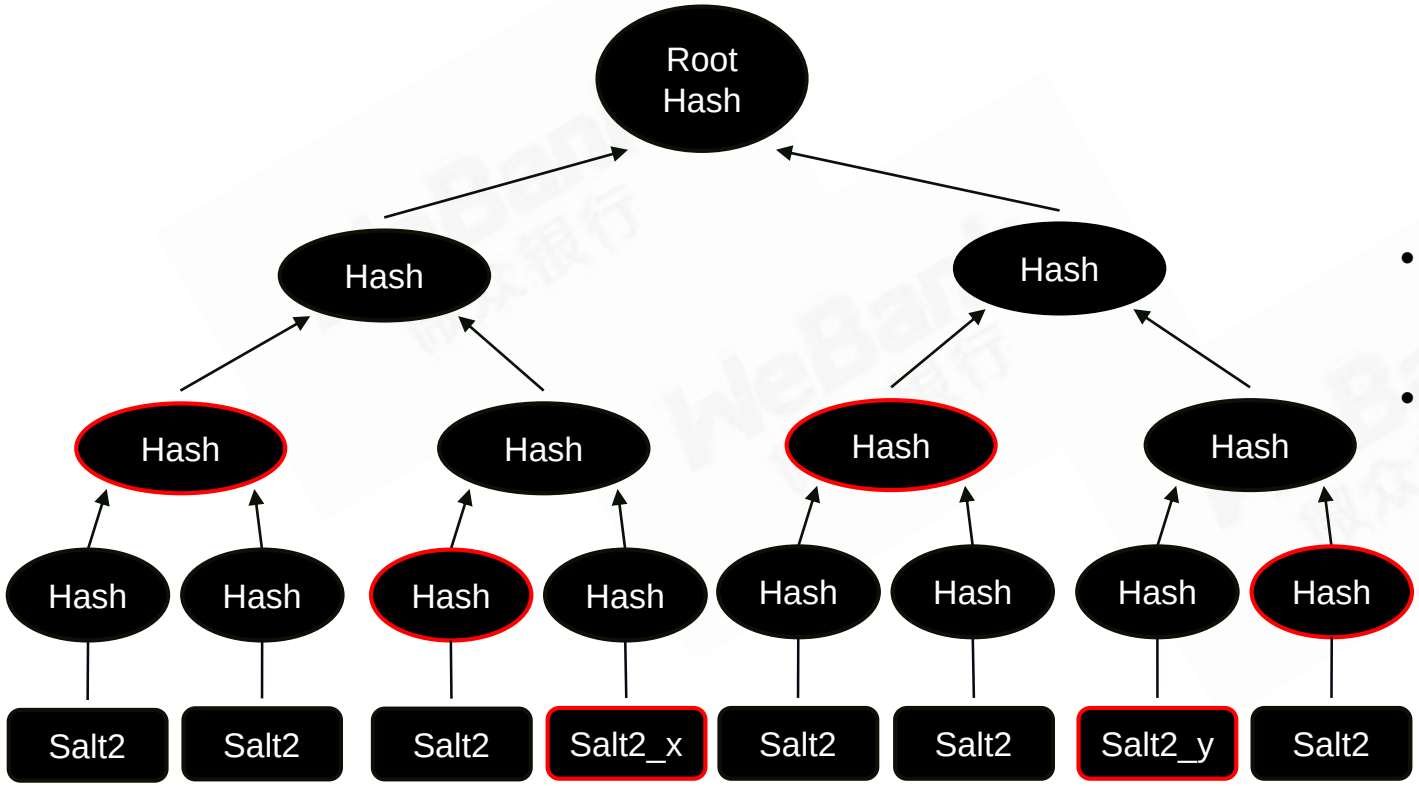
用户凭证： 有机构签名（数字证明，对应公章）背书的证明信息，通过授权式访问，可以不上链，或在链上进行加密保护，或者在需要时，点对点的动态签发（用户授权->[用户+机构A+机构B]三元关系签发->使用）

用户数据：和身份标识挂钩的各种个人数据，包括原始身份数据，行为数据，凭证数据

用户使用多个WeID代替一个WeID



WeID匹配方案：Merkle Tree



核心步骤

- 生成与修改Mapping
  - 选择$Salt_2$生成新WeID
  - 生成标准二叉树形态的Merkle Tree，并将Root Hash、树高公布于链上
  - 如果不够2^N个$Salt_2$，以0补齐叶子
- 取消Mapping
  - 从底向上重新计算生成Root Hash并更新
- 验证Mapping
  - 出示需验证的WeID，及足以构建Merkle Tree的所有必须Hash值（见左图）
  - 验证者使用$Salt_2$确认可以生成此WeID
  - 验证者从底向上计算Root Hash，如果和公布的Root Hash一致，则验证成功

Credential
选择性披露实现

选择性披露

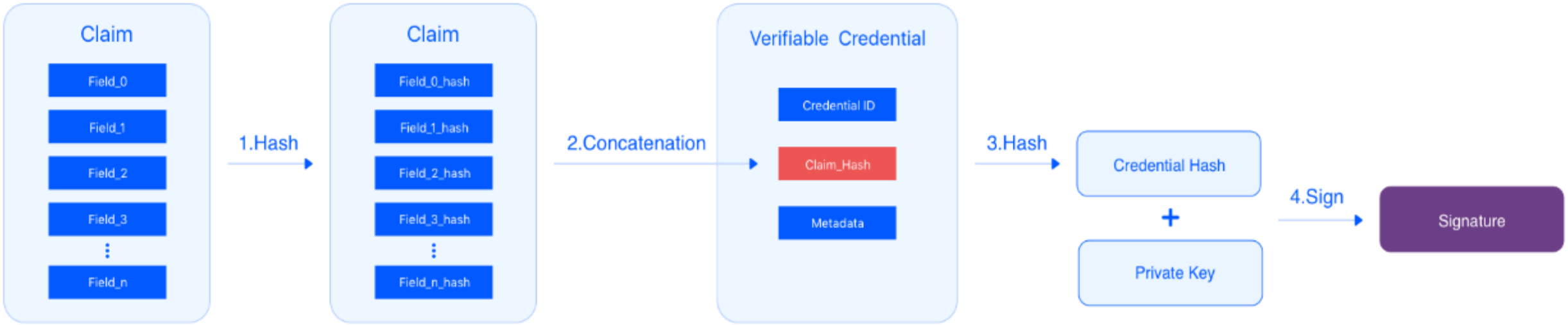
- 范例：Claim中的某项内容变成了无法反向破解的hash值


```
{
  "@context":
"https://github.com/WeBankFinTech/WeIdentity/blob/master/context/v1",
  "id": "c4f8ca00-7c1b-4ba0-993f-008106075d9c",
  "cptId": 101,
  "issuer":
"did:weid:101:0x39e5e6f663ef77409144014ceb063713b65600e7",
  "issuanceDate": "2019-05-19T09:10:24Z",
  "expirationDate": "2019-05-19T17:10:24Z",
  "claim": {
    "delegator":
"did:weid:101:0xae0b295667a9fd93d5f28d9ec85e40f4cb697bae",
    "receiver":
"did:weid:101:0x0231765e19955fc65133ec8591d73e9136306cd0",
    "credentialid": ["04a3e89d-825a-49fe-b8f5-8ccb9f487a52”]
  }
```

```
{
  "@context":
"https://github.com/WeBankFinTech/WeIdentity/blob/master/context/v1",
  "id": "c4f8ca00-7c1b-4ba0-993f-008106075d9c",
  "cptId": 101,
  "issuer":
"did:weid:101:0x39e5e6f663ef77409144014ceb063713b65600e7",
  "issuanceDate": "2019-05-19T09:10:24Z",
  "expirationDate": "2019-05-19T17:10:24Z",
  "claim": {
    "delegator":
”0x1D59CAB179FC78324A7B4D73B1BDFDA8CF1E2FC61DAAA8D284E8860D5D0AFC25",
    "receiver":
"did:weid:101:0x0231765e19955fc65133ec8591d73e9136306cd0",
    "credentialid": ["04a3e89d-825a-49fe-b8f5-8ccb9f487a52”]
  }
```

选择性披露签名过程

1. Claim中的每个字段计算生成一个对应的hash值。
2. 将Claim中的每个字段的hash值以某种形式拼接起来形成一个字符串Claim_Hash，然后跟Credential原有的其他字段组成一个新的用于计算hash的Credential结构。
3. 对这个包含Claim_Hash的Credential结构计算hash，得到Credential Hash。
4. 使用Private Key对这个Credential Hash进行签名，得到签名的值Signature。

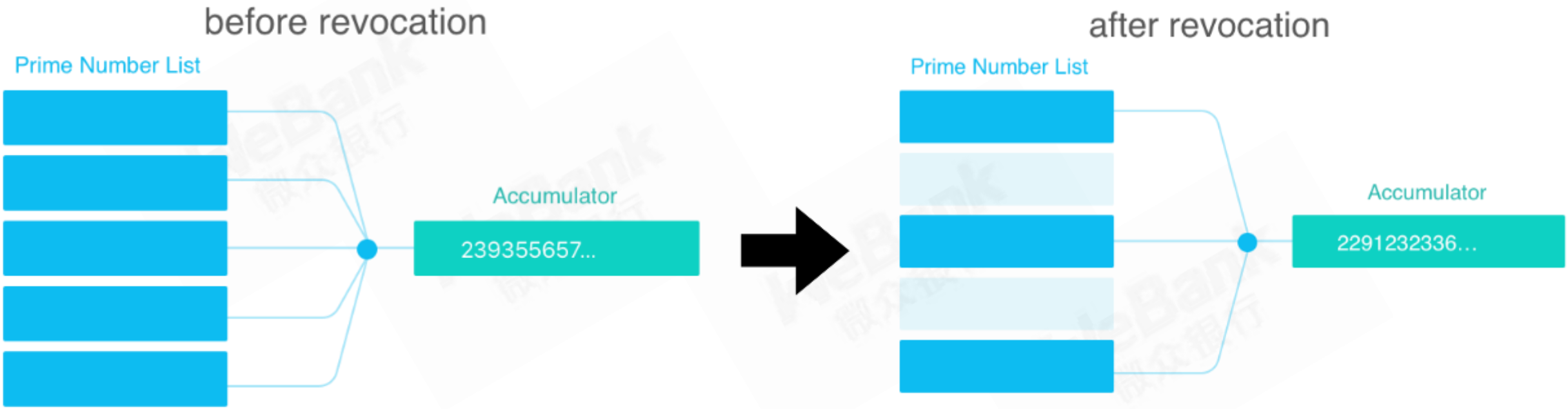


选择性披露验证过程

1. Verifier从Credential提取用户披露的Claim。
2. Verifier对用户披露的字段分别计算hash（这个例子中是Field_1,计算出Field_1_hash）,然后得到一个包含所有字段hash值的Claim结构。
3. 对这个Claim结构中的每个字段的hash值以某种形式拼接起来形成一个字符串Claim_Hash，然后跟Credential原有的其他字段组成一个新的用于计算hash的Credential结构。
4. 对这个包含Claim_Hash的Credential结构计算hash，得到Credential Hash。
5. 使用Credential的Signature和issuer的public key进行decrypt，得到一个签名的计算值
6. 比较Credential的Signature与签名的计算值，看是否相等，确认这个Credential的合法性。

Credential撤销实现

Credential的撤销实现



使用大素数作为撤销机制的优势

2* 3* 5 * 7 = 210（Accumulator）

2* 5 * 7 = 70

- 隐私性更好，Issuer无需暴露所有撤销的Credential，而只需暴露一个Accumulator。
- 无需维护一个无限膨胀的撤销列表大文件， Accumulator 可以公开在区块链上。
- 单次（写操作）交易，可以撤销多个Credential。
- 可以直接根据Credential ID进行sharding。

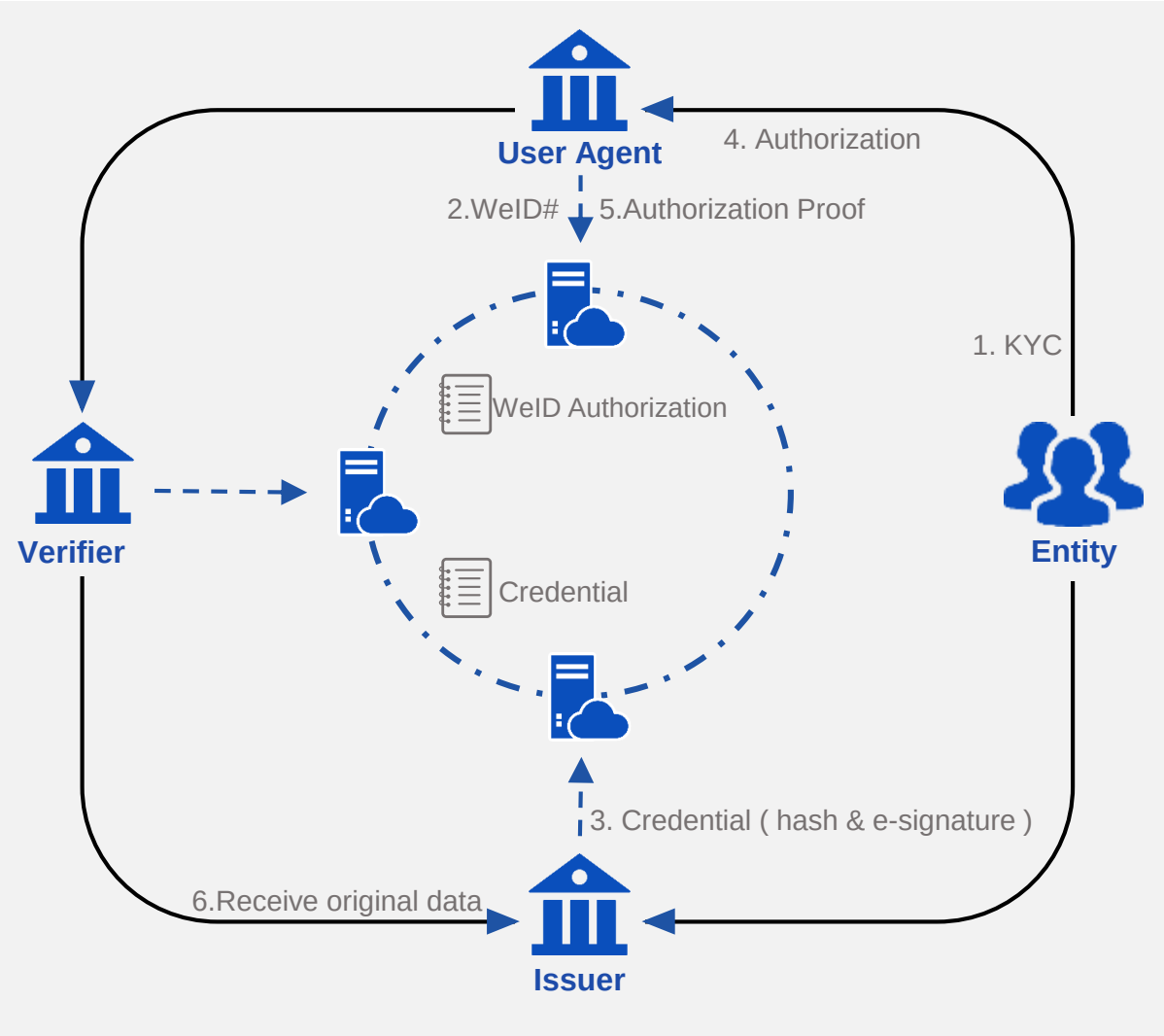
- 已有的机制
  - Credential Revocation List
  - Credential Revocation Tree (Merkle Tree, B-Tree …)

回顾和展望：一些关键问题

- **身份互通**：　身份ID规范，DID网络路由规范
- **选择性披露**：披露明文，披露证明，场景性屏蔽身份
- **屏蔽画像**：　屏蔽群体画像，对个人的画像
- **数据分离**：　UserAgent(个人端，有身份无数据），数据托管(数据银行，有数据无身份)
- **访问即计费**：（C->B），（C->B->B），对数据付费，对托管方付费，隐藏个人身份
- **防止复制和传播**：数据和资产有很大不同，明文无法防止复制，数据水印?
- **TEE和加密机：** 选择什么样的硬件安全方案取决于安全级别和预算
- **设备硬件指纹：** 数据没有暴露你，设备暴露了

案例和演示

方案介绍



实体标识化(WeID)

为每个实体（人或物）在区块链上生成符合国际规范（DID）的全球唯一ID。

电子化凭证(Credential)

将物理世界中的纸质证明文件电子化，并利用区块链不可篡改的特性，将原始数据的Hash上链，并附上权威机构(Issuer)的签名，确保数据不可伪造，可验证权威性。

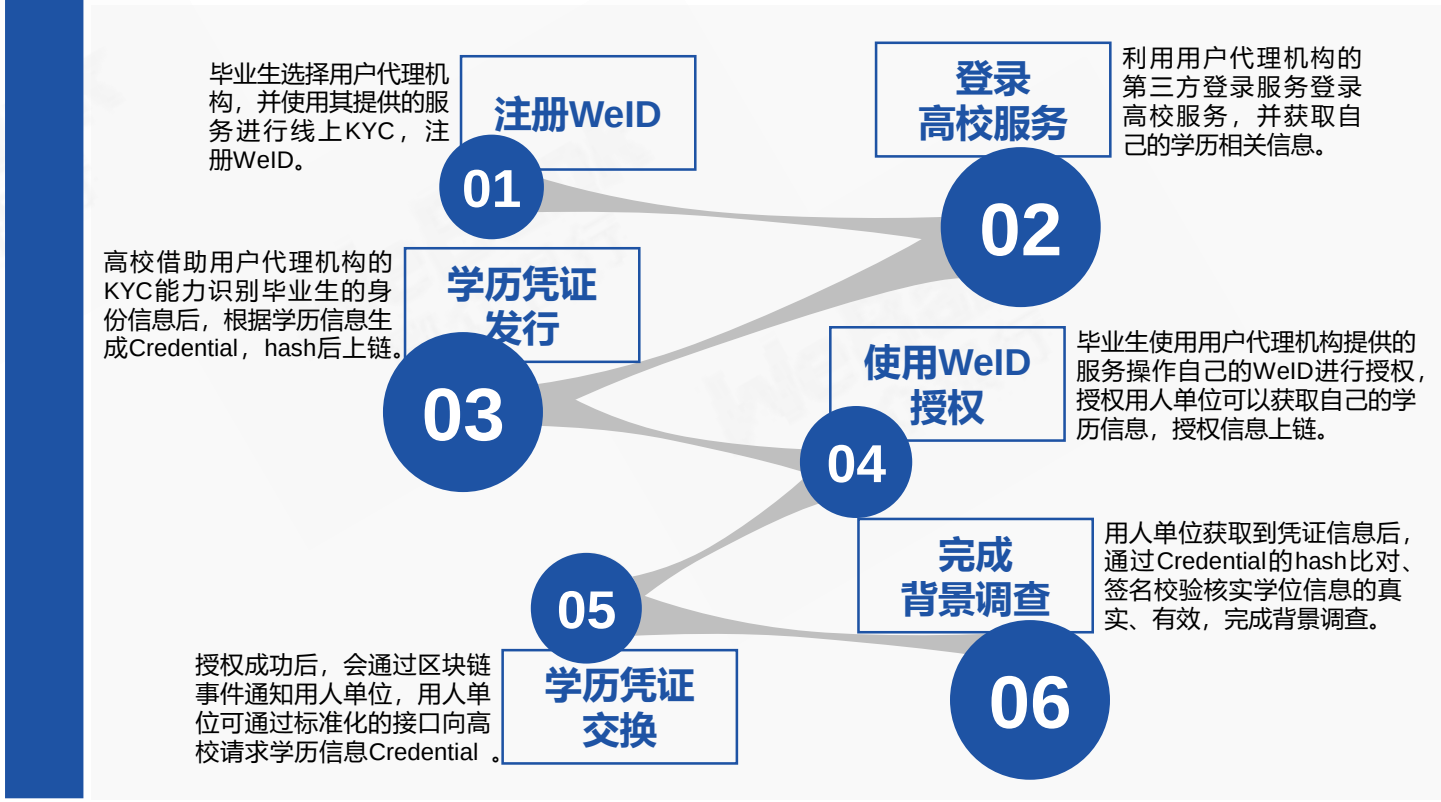
用户授权即交易(Authorization)

原始数据的跨机构传输需要得到用户的授权，授权记录由User Agent上链，符合GDPR。

实体 Entity	实体对象（人或者物），拥有链上身份ID，可授权相关机构使用自身相关数据。
凭证发行方 Issuer	对数据进行发行和认证的机构或个体。权威机构发行的数据具备权威性，个体发行的数据不具备权威性，权威机构的认定取决于具体业务场景及参与角色。
凭证验证方 Verifier	使用数据的机构，可验证数据是否被篡改、是否经过凭证发行方认证。
用户代理 User Agent	为用户生成WeID及提供KYC服务，一般为权威可信机构，实体通过该机构与链上身份或数据进行交互。

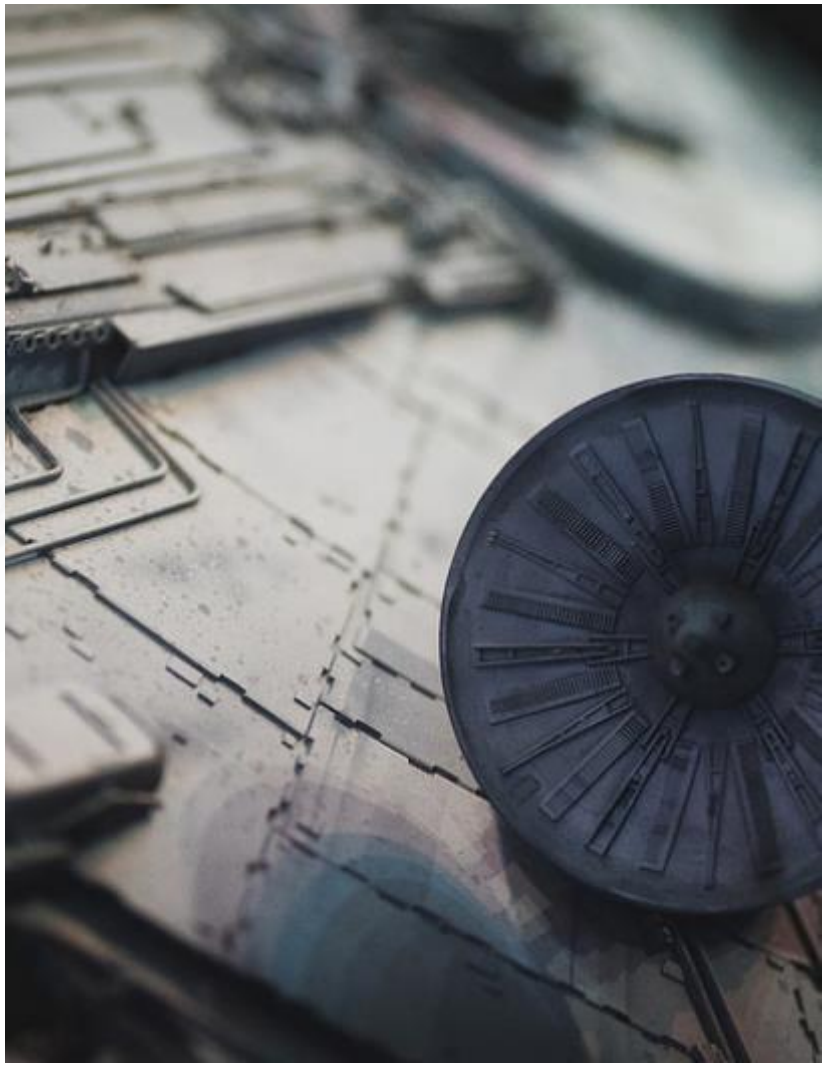
应用案例一：毕业生入职教育背景调查

毕业生入职教育背景调查



WeIdentity方案与传统方案对比：毕业生入职教育背景调查

WeIdentity方案	VS	传统方案
电子化学历凭证成本低，易于管理；	**成本与便利性**	纸质学历证明文件管理成本高，易丢失，不易更新；
密码学算法确保真实性和有效性，区块链记录确保不可篡改性；	**真实性**	纸质学历证明文件有造假可能性，验真周期长、效率低；
全流程无第三方机构参与，用户授权后，用人单位方能获取学历信息，隐私保护性强；	**隐私保护**	第三方背调公司参与可能导致用户隐私泄露问题；
方案可扩展性强，增加高校或用人单位数量，背调工作量不会相应增加。	**可扩展性**	方案可扩展性差，每增加一所高校或用人单位，第三方背调工作量会线性增加。



应用案例二：居民信息管理与政务办理

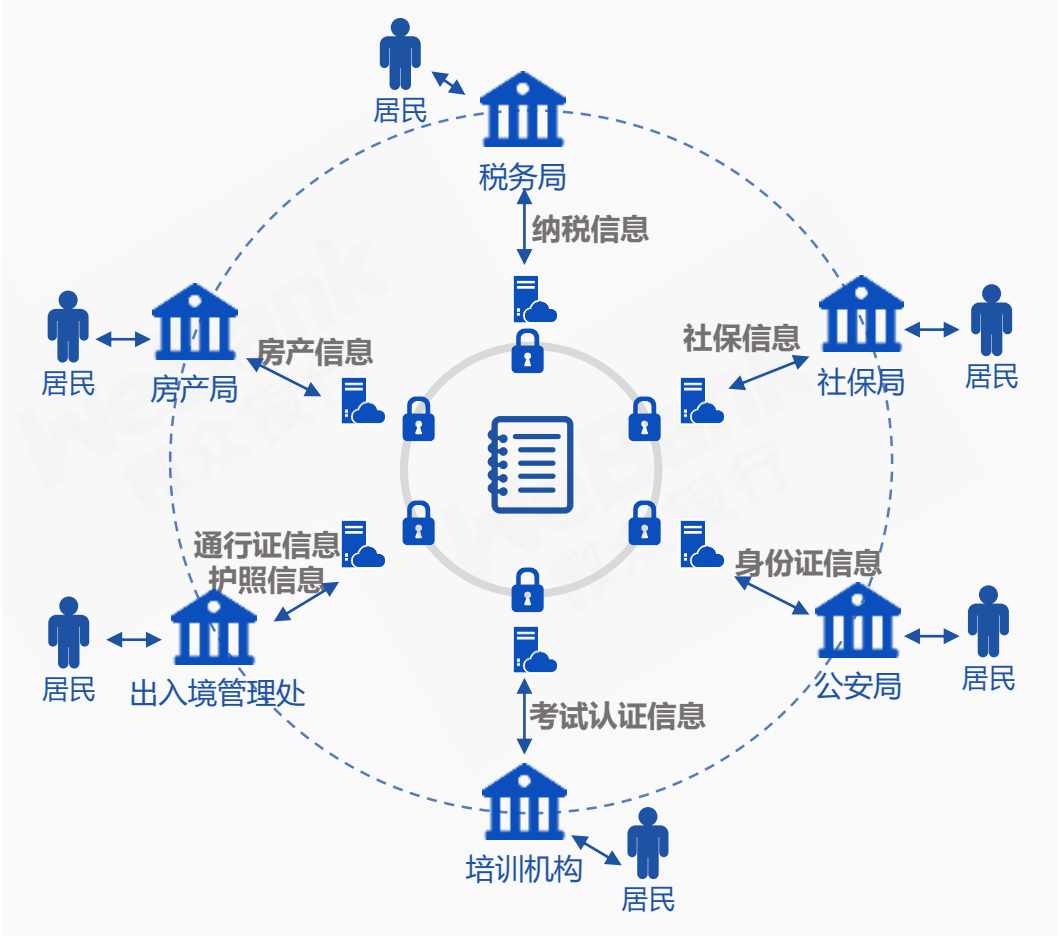
创建身份： 建立居民在链上的唯一身份标识，与真实可验的居民证件（身份证号、护照号、通行证号）映射。

关联映射： 居民在各部门应用中，可维护原有账号不变，通过真实证件映射到链上身份，达成账号间关联。

构建路由： 各部门对自身的居民数据摘要上链，并进行签名认证，形成每个居民的链上数据路由。

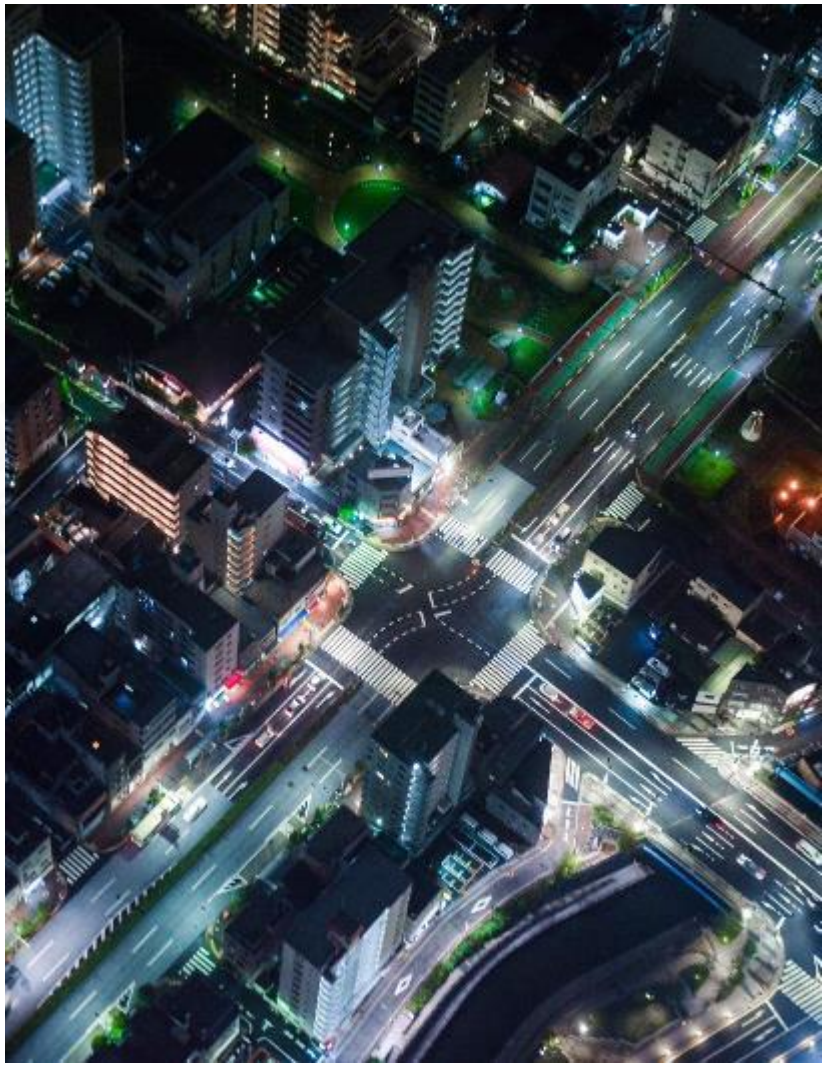
政务应用： 跨部门政务应用时，当居民授权，部门A可通过链上路由发现部门B有所需数据，并使用合法合规方式进行获取，实现快速验证或政务办理。

居民信息管理与政务办理



WeIdentity方案与传统方案对比：居民信息管理与政务办理

WeIdentity方案	VS	传统方案
证明文件转为Credential，可信存储，支持多终端应用；	**成本与便利性**	纸质文件管理成本高，使用次数、流程及场景均受限；
链上身份与现实身份一一对应；密码学算法确保真实性和有效性，区块链记录确保不可篡改性，通过链上摘要信息验真、增信；	**真实性**	对申请人身份真实性核验以及对文件数据的真实性核验，成本高、周期长、效率低；
数据归属于用户，机构使用数据需得到用户授权，隐私保护性强；可验证数字凭证的内容均在链下存储；	**隐私保护**	多部门间的信息传递，道德风险及操作风险均可能导致用户隐私泄露；
ID及凭证均遵循国际标准；方案可扩展性强，增加参与机构不会对已有业务造成影响。	**可扩展性**	方案的兼容性、可扩展性差。



标杆案例：澳门智慧城市建设之“证书电子化”项目

证书电子化管理

高额的纸质文件制作和管理成本以及受限的证书使用场景对系统提出更高的要求：

纸质证书电子化	文件可信存储与安全传输
数据归属权交还于用户	在线服务须支持多终端应用

跨机构信息交互

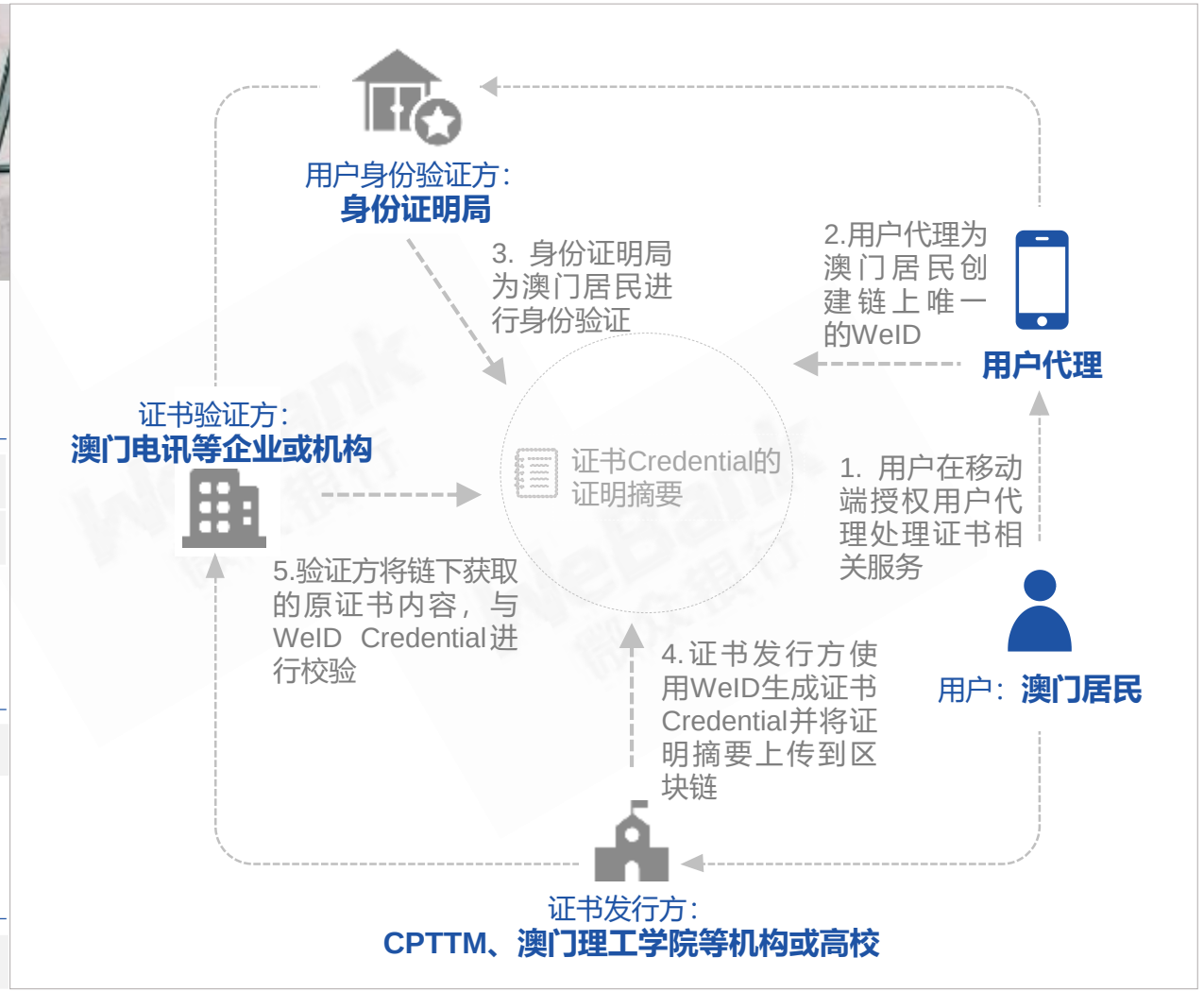
趋于严格的隐私保护法律及跨机构数据交换壁垒促进机构使用：

基于区块链的数据传输	链上存证数据传输记录及用户授权凭证

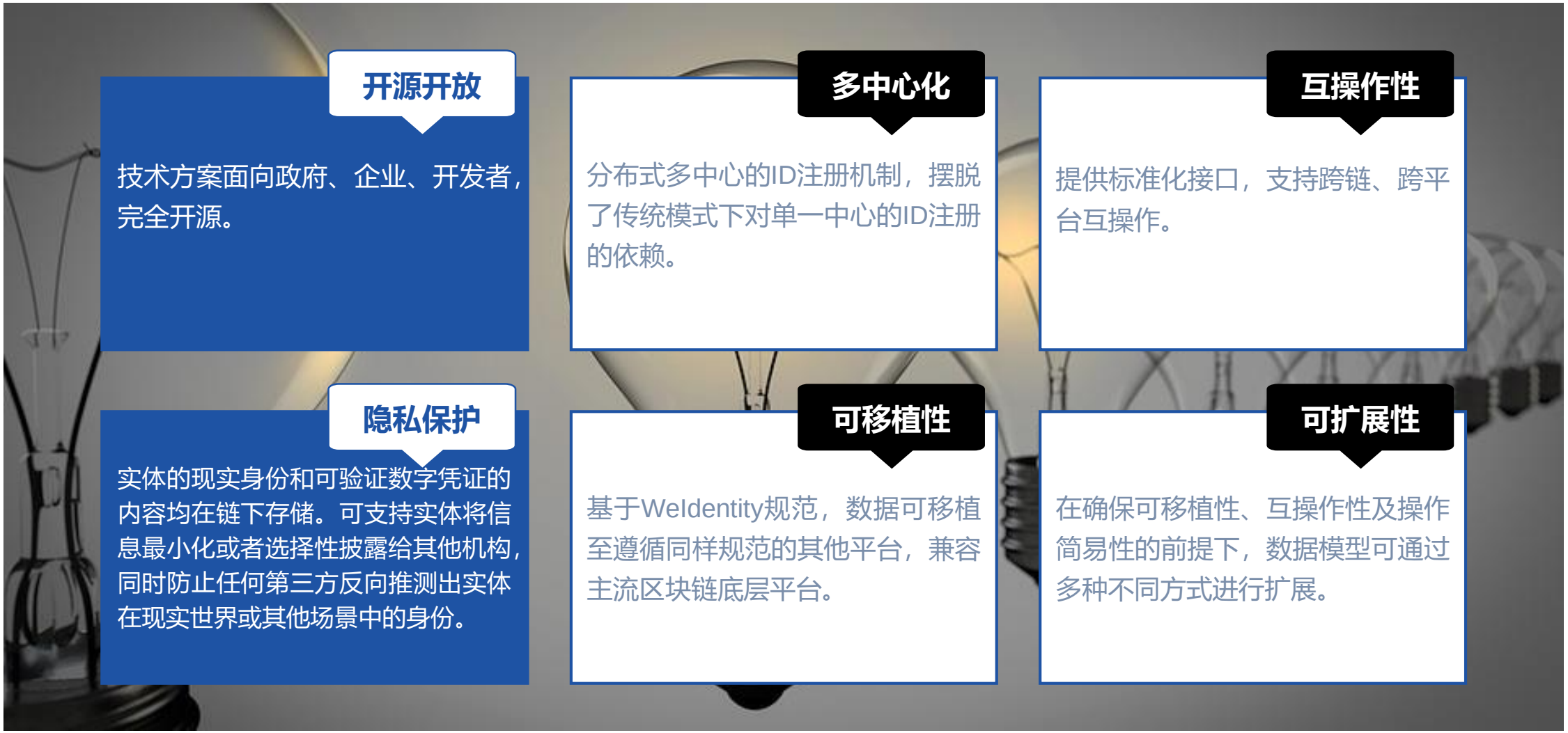
信息真实性验证

证书申请人的真实身份或证书内容本身难以确认，解决方法是：

现实身份与区块链ID一一对应	通过链上hash和电子签名验证证书



技术优势



开源开放

技术方案面向政府、企业、开发者，完全开源。

多中心化

分布式多中心的ID注册机制，摆脱了传统模式下对单一中心的ID注册的依赖。

互操作性

提供标准化接口，支持跨链、跨平台互操作。

隐私保护

实体的现实身份和可验证数字凭证的内容均在链下存储。可支持实体将信息最小化或者选择性披露给其他机构，同时防止任何第三方反向推测出实体在现实世界或其他场景中的身份。

可移植性

基于WeIdentity规范，数据可移植至遵循同样规范的其他平台，兼容主流区块链底层平台。

可扩展性

在确保可移植性、互操作性及操作简易性的前提下，数据模型可通过多种不同方式进行扩展。

相关文档

规范参考

- W3C DID Spec：**(约1.3万单词)**
- W3C Verifiable Credentials：**(约2.5万单词)**
- Linked Data Signatures 1.0 Draft：(约0.3万单词)
- RSA Signature Suite 2018：(0.1万单词)

其他

- **ERC725,734，735，780，1056 ….**

https://eips.ethereum.org/erc
https://github.com/ethereum/EIPs
https://docs.ethhub.io/built-on-ethereum/identity/ERC-EIP/

WeID项目文档

- **文档：** https://weidentity.readthedocs.io/zh_CN/latest/README.html **(约5万字)**

Step by step体验

环境准备和联盟链搭建

1.环境准备

配置	说明
操作系统	CentOS （7.2 64位）或Ubuntu（16.04 64位）。
JDK	要求JDK1.8+，推荐使用jdk8u141。
gradle	WeIdentity JAVA SDK使用gradle进行构建，您需要提前安装好gradle，版本要求不低于4.3。
网络连通	检查WeIdentity JAVA SDK部署环境是否能telnet通FISCO BCOS节点的channelPort端口，若telnet不通，需要检查网络连通性和安全策略。

2. 参照FISCO BCOS 教程搭建联盟链
https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/installation.html

体验WeIdentity

3.使用部署工具部署和集成WeIdentity https://weidentity.readthedocs.io/zh_CN/latest/docs/weidentity-build-with-deploy.html

4.运行 WeIdentity Sample
我们提供了一整套的流程演示，可以帮您快速理解 WeIdentity 的运行机制，您也可以参考该样例程序，开发您的 Java 应用。具体参考：
https://weidentity.readthedocs.io/zh_CN/latest/docs/weidentity-sample.html

5.更多，请见总文档：
https://weidentity.readthedocs.io/zh_CN/latest/README.html

THANKS