Elastos DID Programming Guide（draft）

Data Model and Syntaxes for DID

Song SJun

2018

目录

[概述 2](#_Toc526645743)

[Elastos DID 2](#_Toc526645744)

[DID的属性 2](#_Toc526645745)

[增加和修改Elastos DID的属性 3](#_Toc526645746)

[读取Elastos DID的属性 3](#_Toc526645747)

[删除/废弃Elastos DID的属性 3](#_Toc526645748)

[DID的原理 3](#_Toc526645749)

[PKI (Public Key Infrastructure)：身份和签名 3](#_Toc526645750)

[Elastos DID Side-Chain：数字身份侧链 4](#_Toc526645751)

[签名上链 4](#_Toc526645752)

[主张 (Claim)：自主声明 4](#_Toc526645753)

[证明 (Proof)：第三方证明 4](#_Toc526645754)

[DID属性的公开和保密 5](#_Toc526645755)

[DID的体系结构 6](#_Toc526645756)

[DID编程规范 6](#_Toc526645757)

[DID的自我声明 6](#_Toc526645758)

[删除/废弃一个DID 7](#_Toc526645759)

[DID属性的隐私保护 7](#_Toc526645760)

[证明内容的要素 8](#_Toc526645761)

[双向身份验证：三次握手 9](#_Toc526645762)

[DID应用场景举例 9](#_Toc526645763)

[第三方登录 10](#_Toc526645764)

[身份证明 10](#_Toc526645765)

[App用户绑定DID 11](#_Toc526645766)

# 概述

DID (Decentralized Identifier) 是指可以由用户自主发行、自主证明拥有权的数字身份。

在传统互联网的技术实现里，需要由中心发行数字身份，即可以避免命名冲突，也可以完成身份验证。再通过颁发身份的中心完成陌生人之间的身份验证。

受比特币的去中心化钱包的启发，我们可以使用钱包地址作为用户的ID，通过地址对应的公钥进行验签，完成身份验证。陌生人之间不再需要第三方来确认身份。从而为互联网的独立、自主运行提供了一种数字身份的解决方案。

Elastos在上述基础之上，实现了Elastos DID Side-Chain，为互联网世界提供去中心数字身份的解决方案。让每个人都可以免费拥有属于自己的DID。通过拥有DID，每个人都可以在互联网世界里自我证明 (self-sovereign)；每个人都可以为自己的数字资产确权；每个人都可以基于可信身份安全通信；

本文档基于Elastos DID Side-Chain提供的基本接口，针对DID提供进一步的编程建议和参考，希望可以帮助DID开发者更好使用DID编程。

# Elastos DID

Elastos DID是基于Elastos DID Side-Chain生成的数字身份。首先按照Elastos DID Side-Chain的私钥规则生成私钥，再由该私钥生成公钥，再通过公钥生成DID。

其中私钥用来代表数字身份的拥有权。公钥用来验证数字身份的拥有权。DID是这个数字身份的符号化代表，类似于传统帐号体系中的用户名。

Elastos DID Side-Chain提供了DID的创建、读和写接口，请参考《Elastos DID Specification》。

# DID的属性

DID采用Key-Value方式保存属性。多次写入相同的Key，建议只取最新的作为有效值，忽略旧的。考虑到一个Key可能存在多个Value的情况，在DID Side-Chain的数据结构上，Value采用JSON格式的数组，无论是一个、多个或者是Empty的情况，都采用数组方式保存内容。

如果需要更新某个Key的Value，需要先获取旧的Value，在旧的Value内容基础上做修改。修改完成以后再次调用Write方法写入DID的新属性。

当读取DID时，Elastos DID Side-Chain的Read方法默认只返回最新的Value。从而使得对Value的修改生效。

## 增加和修改Elastos DID的属性

请参见《Elastos DID Specification》中描述的DID的Write方法。

## 读取Elastos DID的属性

请参见《Elastos DID Specification》中描述的DID的Read方法。

## 删除/废弃Elastos DID的属性

请参见《Elastos DID Specification》中描述的DID的Remove方法。

# DID的原理

## PKI (Public Key Infrastructure)：身份和签名

DID来自Elastos DID Side-Chain的公钥和私钥，跟传统PKI体系一样，一个私钥代表一个身份。就像银行卡的密码一样，通过密码授权的每一笔交易都代表本人意愿。密码是中心化系统的身份凭证，私钥则是自主发行的身份凭证。

传统中心系统通过密码来验证身份是否正确。在PKI帮助下，我们可以生成一个随机数交给需要验证的用户，它用私钥对这个随机数进行签名，我们再通过它的公钥来验证签名是否能正确匹配。这就可以实现双方直接互验身份，不需要中介。

虽然我们可以从传统中心系统取得身份的认可，但如何使用这个身份并不由用户自己做主。如果想使用自己的身份签署一份授权、发出一份声明、提供一份认证，这些都依赖于中心系统，如果中心系统没有提供相应服务，就无法完成。最简单的例子，我们无法用银行卡去乘飞机。虽然银行已经对我做了最高等级的身份验证（KYC），但因为银行与航空公司并没有互通信息，航空公司无法确认我是我，最终我没办法凭借我的银行卡去乘飞机。

基于目前的密码学知识，签名内容不可仿冒、不可伪造，也不可抵赖。我们可以使用私钥对任意内容进行签名。这种签名操作不需要任何人认可和许可。只要接收方（应用）认可既可。如果我在购买机票时附加一份我的公钥，用来告知航空公司我的数字身份。当我在乘飞机时，只需要使用我的私钥对检票员的随机数进行一次签名即可验证我是我。（当然这个例子里故意忽略了安检、是否本人等一系列现实中的细节）

这个世界的应用场景非常繁多，而且每天都在产生新的应用。如果用中心化身份验证方式，这种指数级的互通工作量几乎无法想象。基于PKI的方式显然可以简化验证方式，不必烦劳已有业务开发者去再次升级系统。

这就是PKI给我们带来的好处：身份和验证方法。

## Elastos DID Side-Chain：数字身份侧链

众所周知，区块链具有公开、透明、不可篡改的特性。只能不停追加，不能修改已有内容。

同时，作为公链，任何人都可以接入它，可以遍历所有历史记录，可以查看所有信息。

基于这些特性，我们实现了Elastos DID Side-Chain，它与BTC联合挖矿，同时具有公开和不可改的特性。

我们可以将与身份相关的DID属性信息保存在DID Side-Chain，让这些信息永不消失，并且不受任何人控制。

## 签名上链

DID的拥有者可以在DID的属性里写入任何内容，这些写入的内容必须附带签名，表示对内容的认可。内容一旦被写入DID Side-Chain就无法被修改、被删除。通过签名我们可以保证DID属性的真实性，不可仿冒和伪造；通过区块链我们保证DID属性不能被删除、修改，也就无法抵赖。

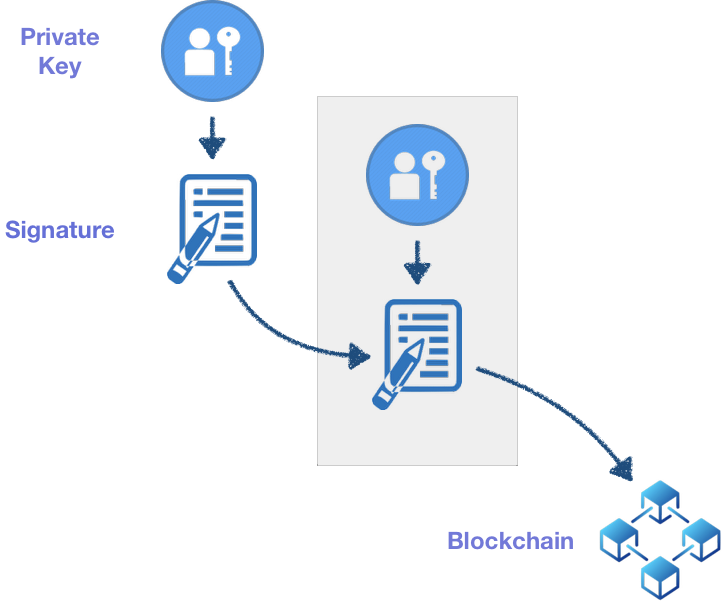
## 主张 (Claim)：自主声明

在DID属性里写入的任何内容都可以把它看做是一种“主张/声明”。比如：nickname，email，收款钱包地址，等等此类。类似社交平台上填写的各种个人属性信息。通过DID记录下来是为了表示这些内容都是“本人”认可的、可信的。

## 证明 (Proof)：第三方证明

如果在应用场景中，不能完全采信DID属性里自主声明的内容，就需要第三方出具证明，从旁佐证内容的真实性。日常生活中这些第三方证明是以证书的方式存证，比如：政府颁发的身份证件，学校颁发的毕业证，职业机构/行会颁发的执照。

对于这种情况，第三方机构可以申请自己的DID，同时用自己的DID对相应的内容进行签名，表示认可。在第三方签名的基础上，用户再用自己的DID进行签名并写入DID Side-Chain，从而完成针对DID属性的证明上链。



例如，学校通过自己的DID为学生的毕业证明文件签名，学生再将此签名写入自己的DID属性里。第三方可以通过从链上读取这个签名内容，用学生和学校的DID的公钥进行验证是否匹配。

## DID属性的公开和保密

DID的属性保存在区块链上，虽然方便全互联网用户读取，导致属性内容可见范围过大，会造成隐私内容泄露。为了解决这个问题，可以对属性内容加密存储。只对需要的人开放访问权，可以有效控制访问范围。

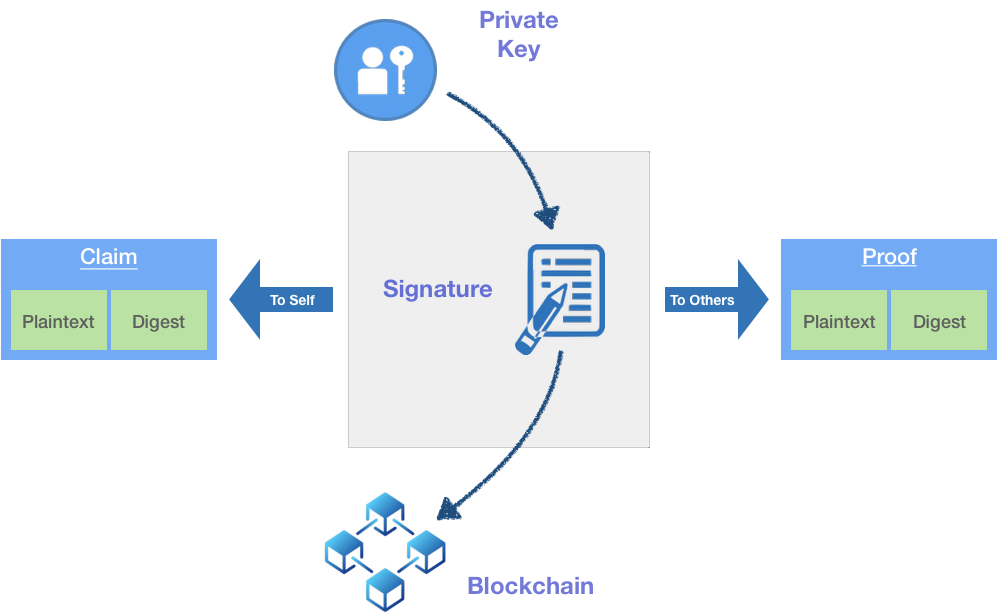
在这里，有两种加密方式：

1. 全文加密保存上链，用户自己管理密码。向需要内容的第三方提供密码，第三方从链上获得数据并解密以后获取明文内容。
2. 将内容的摘要保存上链，用户自己保管内容的明文。向需要内容的第三方提供明文，第三方从链上获取摘要，通过与用户的明文进行验证匹配来证实用户提供内容的真伪。

我们推荐使用第二种，它至少具有两方面好处：

* 从性能角度，区块链读写速度和区块链容量都有限，并且按使用量付费，不适合写入过多数据，使用摘要上链的方式可以解决区块链对内容长度和速度的限制，也更省手续费。
* 从安全角度，虽然是加密内容，但仍存在被破解的可能性，如果用户采用的是弱密码，更增加了隐私泄露的风险。

## DID的体系结构



*私钥、签名和区块链*

要点如下：

* 通过Elastos DID Side-Chain生成私钥、公钥和DID。
* 通过私钥签名，使用公钥验证签名，检查公钥与DID是否匹配。
* 签名内容写入DID Side-Chain。
* 签名内容可以是自我声明，也可以是为其它DID提供的证明。

# DID编程规范

为了更好地使用用户的DID属性，在保护用户隐私地前提下使用用户数据，Elastos提出如下编程规范，请应用开发者参考。

## DID的自我声明

我们完全可以在离线的、链下的状态下生成DID和验证DID。但在是使用过程中经常需要获取DID的公钥，用于验证DID的签名。为了方便、可信地获取DID的公钥，同时在链上实现DID信息的完整性，建议应用将DID对应的公钥写入Elastos DID Side-Chain，为了方便其它应用使用，建议采用统一的属性名“PublicKey”。

举例：

已有DID ：“IHLhCEbwViWBPwh1VhpECzYEA7jQHZ4zLv”

它的公钥：

“02C3F59F337814C6715BBE684EC525B9A3CFCE55D9DEEC53E1EDDB0B352DBB4A54”

自我声明：

DID.Write (

“IHLhCEbwViWBPwh1VhpECzYEA7jQHZ4zLv/PublicKey”,

“{ ‘values’ : [

‘02C3F59F337814C6715BBE684EC525B9A3CFCE55D9DEEC53E1EDDB0B352DBB4A54’

] }”,

“E6BB279CBD4727B41F2AA8B18E99B3F99DECBB8737D284FFDD408B356C912EE21AD478BCC0ABD65246938F17DDE64258FD8A9684C0649B23AE1318F7B9CEEEC7”);

第三方通过从Elastos DID Side-Chain上读取

“IHLhCEbwViWBPwh1VhpECzYEA7jQHZ4zLv/PublicKey”

即可得到DID的公钥，并验证其是DID的拥有者。

**建议**：使用PublicKey属性保存DID的公钥

## 删除/废弃一个DID

当一个DID不再使用，甚至发生盗用的情况时，如果用户还掌握DID的私钥，可以声明废弃DID，避免身份冒用带来的损失。

我们可以约定，“修改”PublicKey的属性值为“Destroy”表示DID持有人废弃这个DID。即便PublicKey之后被设置为其它值，也一律忽略。

任何应用如果发现一个DID的PublicKey字段**曾被**写入过“Destroy”，都应该立即停止为它提供服务。

**建议**：针对PublicKey属性写入字符串“Destroy”表示删除/废弃/停用这个DID

## DID属性的隐私保护

由于区块链上的信息是完全公开的状态，任何人都可以爬取用户的全部信息，为了避免用户的信息泄露，建议写入属性时不要带有不需要公开的信息。

通常情况下，建议DID的属性名称和属性内容都采用摘要的方式上链。并且，对于不同用户来说，同一个用途的属性的名称也不应相同，避免被猜测。

例如，如果使用WeChatID作为属性名，很容易猜到用户在使用微信。即便使用WeChatID的hash值作为属性名，因为它的hash值的固定不变的，一旦被攻击者猜中hash的明文，即可反向推测有哪些使用了微信的用户。

针对这个案例，建议采用“属性+内容”的hash作为属性名称，而属性的Value则为该hash的签名。

同时，一条属性所对应的信息尽量少，最好能具有类似原子不可分割的特性。

不好的属性设计举例：

UserProfile = [user name] + [user age] + [user location] + [user email] + [user phonenumber]

在一个属性中包含太多信息，为了提供user name的证明，需要同时提供user age、user location、user email、user phonenumber等多余信息，这会造成用户信息不必要的泄露。

建议按照使用场景对上述信息进行拆分，将通常在一起使用的、不可分割的信息作为一条属性。

**建议**：

* 非公开的属性名称和内容都使用摘要内容。
* 名称不要采用固定内容，避免反向猜测。
* 属性的内容使用场景一致，保持内容的原子性，既每个属性都是操作的最小信息粒度，避免无关信息被泄露。

## 证明内容的要素

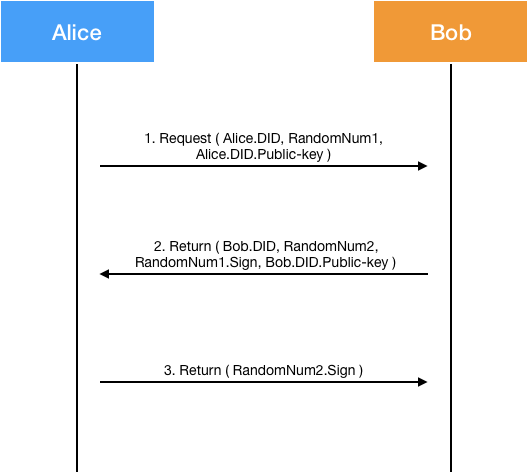
当需要使用DID为第三方提供证明时，证明内容要尽量保持原子性。这与之前建议属性内容使用尽量小颗粒度的信息一样，避免暴露无关信息、泄露用户隐私。

证明内容的设计要充分考虑：内容是否存证二义性、是否存在被冒用或者挪用的可能性。

**建议**：

* 证明内容包含有效期，从何时起到何时止。
* 证明内容包含颁发者的DID。
* 证明内容包含授予者的DID。
* 证明内容包含具体证明事项的唯一性描述信息。

## 双向身份验证：三次握手



对于两个陌生人而言，如何确认对方的有效身份，是所有业务开始之前最重要的步骤。这里提议采用三次握手来完成交换彼此的DID信息和验证工作。

1. Alice发送一个随机数RandomNum1给Bob，并附带自己的DID和公钥；
2. Bob返回随机数RandomNum1的签名、DID和公钥，并返回另一个随机数RandomNum2给Alice。
3. Alice验证通过以后，再对RandomNum2进行签名，并返回给Bob。Bob进行验证，确认Alice的身份。

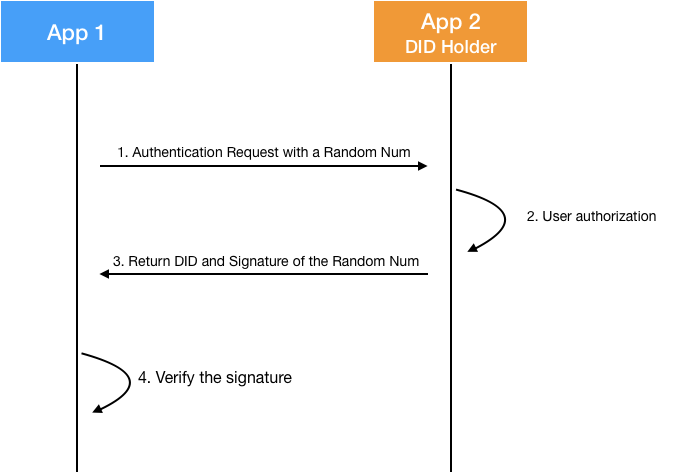
至此完成了无中介/中心的环境下身份的验证。

# DID应用场景举例

DID在跨应用共享用户身份信息的应用领域具有很好的使用场景。同时也可以为应用数据提供所属权证明。让用户可以对自己的信息、数据进行确权。

在确权基础上，不同应用可以参考下面的场景举例中的技术方案实现打通用户业务流，实现跨多个应用业务流程。

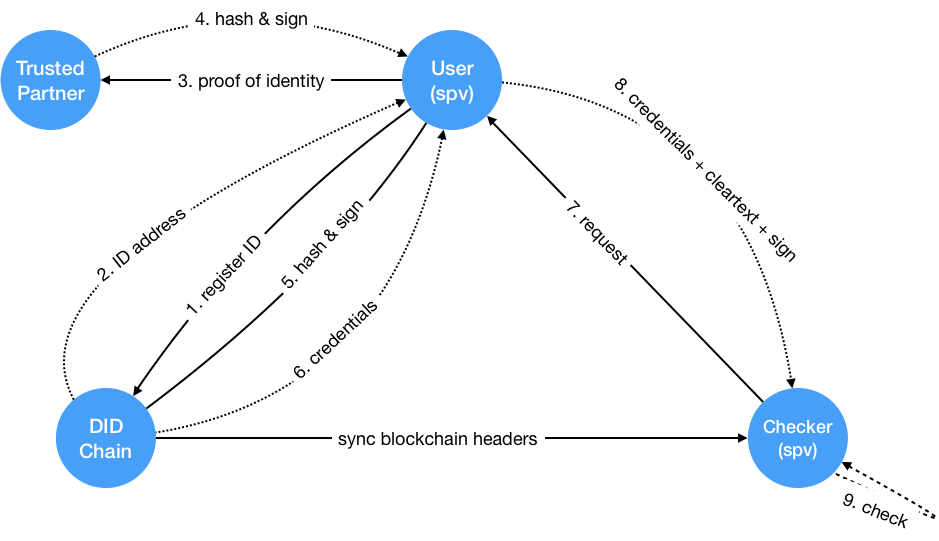
## 第三方登录



登录流程：

1. App1向App2发起请求，并附带App1的信息和一个随机数；
2. 用户授权允许访问，并对随机数进行签名；
3. 返回用户的DID和随机数的签名结果；
4. App1对返回签名进行验签。

## 身份证明



创建DID流程：

1. 创建DID；
2. 返回DID；

第三方机构验证身份流程：

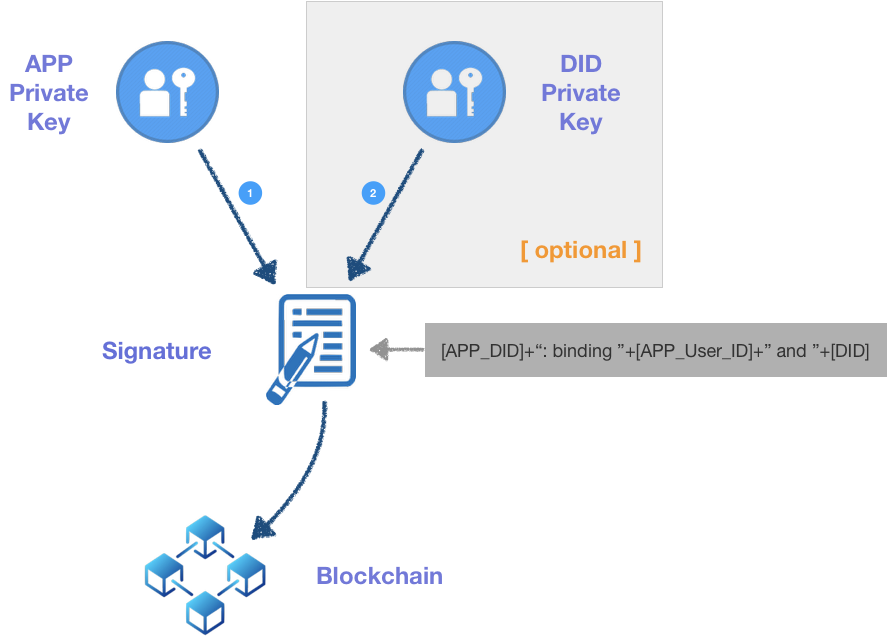
1. 向第三方机构提供证明材料；
2. 第三方机构验证通过后，对材料进行签名并返回给用户；
3. 用户将签名过的证明材料写入DID Side-Chain；
4. 返回写入交易的TXID；

第三方检查身份流程：

1. 第三方App向用户请求提供证明材料；
2. 用户提供明文信息、第三方机构签名过的证明材料和写入区块链的TXID；
3. 第三方App对用户和第三方机构的签名进行检验、证明材料的明文和链上存证进行检验。

## App用户绑定DID

APP应用可以将系统的用户与DID绑定。从而将App应用内产生的用户数据与DID形成关联关系。



App使用系统里用户唯一标识，比如UserID，与用户的DID一起生成一条绑定信息。再使用App的DID对绑定信息进行签名。App可以将签名后的信息写入区块链存证。

用户也可以将签名后的绑定信息用自己的DID写入区块链，作为与该App用户关联关系的存证。这个步骤是可选的。

在App用户与DID绑定之后，App可以直接对系统内用户的活动记录签名、存证到区块链，从而实现对用户数据的确权。