基于零知识证明的数字资产私密发布和授权协议

概述

零知识证明是目前区块链研究最为火热的领域之一，有望在扩容和隐私保护两个方向为区块链带来巨大的突破。

众所周知，在区块链上存储的数据是全部公开的，任何人都可以获取、解析所有的区块数据。这极大地限制了区块链的应用场景。例如，在比特币等采用 UTXO 模型设计的区块链系统中，每一笔 UTXO 的流动都是公开的，匿名性的保证是通过隐藏人和地址的对应关系来实现，一旦某一个地址和人的对应关系暴露，再配合一些链上数据分析工具，基本上可以追踪到这个人的全部交易记录和资金数量。在以太坊等智能合约区块链中，每一个合约的代码、每一次合约调用的参数也都是完全公开的，所有数据对所有人可见。

而零知识证明恰好可以用来解决这个问题。关于零知识证明的原理在本文中不再赘述。通过一些巧妙的方案设计，我们可以使用零知识证明来保证链上没有任何公开的数据，同时不影响区块链的正常功能。比如 ZCash 项目，就通过零知识证明设计了交易记录混合方案，实现了和比特币一样的功能，但是链上不存储任何一笔交易的具体信息（发送人、接收人、金额等）。

区块链的应用远远不止 Token 转账这一种。人类社会正在经历信息革命带来的巨大冲击，从企业的视角来看，大部分企业正在经历从制造型企业到创新型企业的转型。企业的核心生产资料，正在从资金、厂房、原料等转变为人、知识（信息）、知识形成的网络，专业的说法叫“社会资本”。在这样的大背景下，知识类资料、成果等数字资产的保护和有效流通对企业来说就变得尤其重要。

区块链在帮助数字资产流通上有着天生的优势，但是却在保护数字资产的隐私性上有很大的劣势。以太坊上的 ERC-721、ERC-1155 提案实现了对非同质化 Token（NFT）的标准化，使得在区块链上进行数字资产的发行、交易变得非常容易。但是资产的所有权、转移记录，对外都是完全公开的，这无疑暴露了企业的很多敏感信息。

安永（EY）之前发布的 Nightfall 项目，实现了对以太坊上 ERC-721 类资产私密转账的支持。但是 ERC-721 的使用使得资产的注册是必须公开的，在资产注册以后转入一个私密合约，之后的转移操作才是对外不可见的。另外，对于文章、图片等类

型的数字资产，最重要的流通方式不是所有权的转移，而是使用权的购买。比如文章的转载权购买、图片的购买使用等。对于这类数字资产，使用 NFT 模型抽象是不够的，也就更加无法实现使用权的私密购买。

针对这样的场景，我们设计了数字资产的私密发布和私密授权协议。基于零知识证明实现了如下的功能：

1. **可以在链上发布数字资产，但是隐藏数字资产的所有者。**

用发布图片来举例子。比如，我可以在链上注册一张图片，链上没有这张图片的原始内容，只有图片的哈希值和 ID。从链上也看不出这张图片是我发的。

下一步，假如我还需要证明我对图片的所有权，我在某个媒体上发布这张图片时可以附带一段“文本”，将这段文本拿到区块链上查询，可以证明我是这张图片的作者。但是单独通过链上数据，任何人都无法查到我还有哪些其他图片。其他人即使有了这段文本，也无法伪造所有权证明，也无法对图片进行售卖。

2. **可以在链上进行数字资产的使用授权，但是隐藏授权的购买者、所有者以及数字资产信息。**

比如，我在媒体上看到一张图片，可以付费购买使用权，但是没有人知道我从谁手里买了什么。在我的博客里使用购买的图片时我也可以附带一段“文本”，通过这段文本可以在链上确认我的付费购买的有效性，但是无法查到我的其他任何购买记录，也无法查到我是从谁手里购买了这张图片。

通过上述的功能，我们实现了图片发布、购买的全流程隐私保护。同时也支持在必要时公开一些信息，用以证明所有权，并且不暴露任何其他的额外隐私信息。

需要注意的是，虽然我们用了图片来举例子，但是这个协议实现的功能可以用在任何类型的数字资产的注册和授权上。

在本文的后半部分，我们将对实现上述功能的零知识证明协议进行详细描述。

另外，我们已经在「原本链」上完成了这个协议的开发，并且经过了一段时间的测试。后续我们会将这部分代码整理出来在 Github 开源。

在目前的协议设计中没有数字资产的所有权转移的功能，因为这部分已经有现成的方案可以参考，比如 Nightfall。而我们的协议可以很容易地和这部分方案进行融合。在后续的开源实现中我们将会对这部分设计进行补充。

在我们现有的代码实现中，使用了 ZoKrates 工具包，具体的零知识证明算法使用了 Groth 16。由于零知识证明的通用性，具体使用的算法几乎不影响上层的协议设计。因此实现协议时也可以使用 Bellman 工具包,或者将算法替换为 Bulletproofs 或者是 Sonic 算法，以实现更好的性能，以及去除对可信初始化的依赖。

变量说明

我们在协议中用到的变量如下：

符号	描述
A, B, C	组织 A,B,C
n_A	组织 A 的注册名
h	sha256 哈希函数
$pk_A^E, addr_A$	组织 A 的区块链公钥和地址
sk_A^E	组织 A 的区块链私钥
pk_A^R, sk_A^R	组织 A 的资产公钥和私钥，其中 $pk_A^R = h(sk_A^R)$
α	代表一个非同质数字资产的唯一 id
R_A^α	组织 A 对资产α的注册记录
Z_B^α	将资产授权给 B 的一条记录
π	零知识证明的证据数据
ϕ_L	Merkle Tree 上从叶子节点 L 到根节点的路径上的所有节点值
Ψ_L	Merkle Tree 上从叶子节点 L 到根节点的路径上的所有节点的兄弟节点值
M	Merkle Tree 的根节点计算函数，输入为Ψ_L和 L
MT^R	用于记录资产注册记录的 Merkle Tree
MT^Z	用于记录资产授权记录的 Merkle Tree

协议说明

组织注册

每个组织拥有一对代表组织资产所有权身份的公-私钥对，长度均为 32 字节，分别记为 $\mathrm{pk}^{\mathrm{R}}, \mathrm{sk}^{\mathrm{R}}$，满足关系：

$$\mathrm{pk}^{\mathrm{R}} = \mathrm{h}(\mathrm{sk}^{\mathrm{R}})$$

组织还拥有一对代表其区块链身份的公-私钥对和对应的区块链地址，分别记为 $\mathrm{pk}^{\mathrm{E}}, \mathrm{sk}^{\mathrm{E}}, \mathrm{addr}$，是用标准的椭圆曲线签名算法生成的。组织需要谨慎保管两组公-私钥对中的私钥，不可泄露。

组织还拥有一个组织名 n，组织名会在验证授权信息时提供给查询方，形式上应该是代表组织身份的自然语义信息。

组织需要根据协议完成组织注册流程，才能实现对数字资产的匿名授权。组织注册实际上是在区块链上记录了组织的一些重要信息，将组织的 $\mathrm{pk}^{\mathrm{R}}, \mathrm{addr}, \mathrm{n}$ 三种信息做了一个绑定。这些信息有的不可更改，例如组织的区块链地址 addr 和代表组织的资产所有权身份的 pk^{R}；还有的信息需要权限管理，例如组织的注册名信息 n。

我们需要保证以下几点：

1. 只有拥有 sk^{R}的组织才能为对应的 pk^{R}进行组织注册操作；
2. 一旦组织完成注册，只有拥有 pk^{E}对应的私钥 sk^{E}的地址才能修改组织的注册名。

注意，对于上述两点我们都有一个隐含的需求：组织的私钥不能公开。

第 1 点，我们用零知识证明来实现；第 2 点则依赖于椭圆曲线签名算法机制。

下面以组织 A 为例，具体介绍组织注册的流程：

1. 在安全环境下生成需要的两组公-私钥对和其他组织信息数据；
2. 生成零知识证明π，包含约束：
   - $\mathrm{pk}_{\mathrm{A}}^{\mathrm{R}} == \mathrm{h}(\mathrm{sk}_{\mathrm{A}}^{\mathrm{R}})$

3. 公开输入：$[pk_A^R]$；
4. 私有输入：$[sk_A^R]$；
5. 调用合约方法：$\text{Organization.register}(\pi, pk_A^R, n_A, addr_A)$

注：在 ZoKrates 的零知识证明生成过程中，我们可以设置输入变量是否公开，公开的变量会出现在生成的证明中，而私有的变量则不会出现。

资产注册

完成注册后的组织需要根据协议对自己的数字资产完成注册流程，才能对这些资产进行匿名授权。对于任意的数字资产，组织需要先为其生成一个唯一的数字 id，记为α。在本协议中，数字 id 可以是不超过 32 字节的任意数据。

资产注册实际上是在区块链上记录了组织的资产公钥 pk^R和α的对应关系，我们需要保证以下几点：

1. 只有拥有 sk^R的组织才能为对应的 pk^R注册资产；
2. 同样的α只能被注册一次。

同样的，零知识证明可以帮助我们实现第 1 点。

下面我们以组织 A 注册资产α为例，具体介绍资产注册的流程：

1. 生成一个随机数σ_A
2. 计算$R_A^\alpha = h(\alpha|pk_A^R|\sigma_A)$；
3. 生成零知识证明π，包含约束：
   - $pk_A^R := h(sk_A^R)$
   - $R_A^\alpha == h(\alpha|pk_A^R|\sigma_A)$
4. 公开输入：$[R_A^\alpha, \alpha]$；
5. 私有输入：$[sk_A^R, \sigma_A]$
6. 调用合约方法：$\text{Shield.register}(\pi, R_A^\alpha, \alpha)$。

资产授权

注册后的组织可以将注册后的资产匿名授权给另一个注册后的组织。授权的本质是在区块链上生成了一条授权记录，记录中隐含了资产α以及被授权方的信息，我们需要保证以下几点：

1. 只有经过注册的资产α才可以被授权；
2. 授权方必须拥有α在注册时对应的资产私钥 sk^R；
3. 资产α、授权方信息和被授权方信息不能公开。

其中：

- 第 1 点需要我们给出资产α的注册记录存在的零知识证明；
- 第 2 点需要我们给出授权方拥有 sk^R的零知识证明；
- 只要给出上述零知识证明，第 3 点自然就成立了。

下面以组织 A 授权资产α给组织 B 为例，具体介绍资产授权的流程：

1. 获取组织 B 的资产公钥 pk_B^R
2. 生成一个随机数σ_{AB}，这个随机数将会通过加密信道发送给组织 B
3. 计算$Z_B^\alpha = h(\alpha|pk_A^R|pk_B^R|\sigma_{AB})$；
4. 获取$\psi_{R_A^\alpha}$，即R_A^α在 MT^R上的兄弟节点路径值；
5. 获取 MT^R的根节点 $root_R$；
6. 生成零知识证明π，包含约束：
   - $pk_A^R := h(sk_A^R)$
   - $R_A^\alpha := h(\alpha|pk_A^R|\sigma_A)$
   - $root_R == M(\psi_{R_A^\alpha}, R_A^\alpha)$
   - $Z_B^\alpha == h(\alpha|pk_A^R|pk_B^R|\sigma_{AB})$
7. 公开输入：$[Z_B^\alpha, root_R]$；

8. 私有输入：$[pk_B^R, sk_A^R, \alpha, \psi_{R_A^\alpha}, \sigma_A, \sigma_{AB}]$；
9. 调用合约方法：$Shield.authorize(\pi, Z_B^\alpha, root_R)$。

授权证明

组织为了向第三方展示自己获取了某个数字资产的授权，需要生成一个授权证明。授权证明本质上是证明了组织与区块链上的授权记录的对应关系，即：

1. 区块链上存在α和 pk^R对应的授权记录Z^α；
2. 组织拥有 pk^R对应的私钥 sk^R。

下面以组织 B 生成资产α的授权证明为例，具体介绍生成授权证明的流程：

1. 获取 pk_A^R和σ_{AB}
2. 计算$Z_B^\alpha = h(\alpha|pk_A^R|pk_B^R|\sigma_{AB})$
3. 获取$\psi_{Z_B^\alpha}$，即Z_B^α在 MT^Z上的兄弟节点路径值；
4. 获取 MT^Z的根节点 $root_Z$；
5. 生成零知识证明π，包含约束：
   - $pk_B^R == h(sk_B^R)$
   - $Z_B^\alpha := h(\alpha|pk_A^R|pk_B^R|\sigma_{AB})$
   - $root_Z == M(\psi_{Z_B^\alpha}, Z_B^\alpha)$
6. 公开输入：$[pk_A^R, pk_B^R, \alpha, root_Z]$
7. 私有输入：$[sk_B^R, \psi_{Z_B^\alpha}, \sigma_{AB}]$；
8. $[\pi, pk_A^R, pk_B^R, \alpha, root_Z]$向所有人公开，作为组织 B 拥有来自 A 的资产$\alpha$授权的证明。

验证授权

对于其他组织提供的授权证明，可以通过调用智能合约的方法来验证授权证明的有效性：

$Shield.authorizeCheck(\pi, pk_A^R, pk_B^R, \alpha, root_Z)$

方法返回：

1. B 是否获得 A 的授权；
2. 授权方 A 的组织信息；
3. 被授权方 B 的组织信息。

智能合约

本协议使用了一系列智能合约来实现与区块链的交互，主要包含以下几类：

- 组织相关：用于组织注册、组织名修改等；
- Shield 合约：用于组织发起对资产的注册、授权和验证等；
- Verifier 合约：使用椭圆曲线配对函数来验证零知识证明，作为 library 供其他合约调用。

下面具体介绍两个重要的合约内容：

Organization.sol

- $\mathrm{register}(\pi, \mathrm{pk}_{\mathrm{A}}^{\mathrm{R}}, \mathrm{n}_{\mathrm{A}}, \mathrm{addr}_{\mathrm{A}})$:
  - 在区块链上注册组织，记录组织的注册名、资产公钥和区块链地址的对应关系。
- $\mathrm{resetName}(\mathrm{n}_{\mathrm{A}})$:
  - 修改组织名，只有组织注册的区块链地址有权限调用。
- $\mathrm{get}(\mathrm{pk}^{\mathrm{R}})$：
  - 获得资产公钥 pk^{R}对应的组织注册名。

Shield.sol

- $\mathrm{register}(\pi, \mathrm{R}_{\mathrm{A}}^{\alpha}, \alpha)$：
  - 在区块链上注册数字资产α，这里组织 A 的信息是隐藏的；
  - $\mathrm{R}_{\mathrm{A}}^{\alpha}$将被保存到 MT^{R}中去；

  - 更新 MT^{R}；
  - 生成一个α对应的 ERC-721token，所有者为 Shield 合约。
- $\mathrm{authorize}(\pi, \mathrm{Z}_{\mathrm{B}}^{\alpha}, \mathrm{root}_{\mathrm{R}})$：
  - 将数字资产α授权给 B，这里数字资产α的信息是隐藏的；
  - 合约需要验证 root 曾经或刚好是 MT^{R}的根节点。
- $\mathrm{verify}(\pi, \mathrm{inputs})$：
  - 通过调用 Verifier 合约的方法来验证一个证据π和对应的输入 inputs 是否匹配。
- $\mathrm{authorizeCheck}(\pi, \mathrm{pk}_{\mathrm{A}}^{\mathrm{R}}, \mathrm{pk}_{\mathrm{B}}^{\mathrm{R}}, \alpha)$：
  - 验证授权证明信息。