# 零知识证明于区块链中的落地应用

作者：Gate.io 研究院 Eric Fu，Jill Chow

## 摘要

现今区块链技术的发展速度愈发加快，区块链应用落地伴随而来的是用户对隐私安全性的要求愈发提高。基于此情况，众多区块链开发团队提出了多种不同的用户隐私安全保护机制。其中零知识证明与区块链技术相结合作为一种新的方案为提高区块链隐私安全性提供了更多的可能。本文将结合使用零知识证明的区块链系统—“Zcash”—对其加密技术以及零知识证明进行深入的探讨。

要点总结：

- 零知识证明是上世纪 80 年代提出的一项加密学理论，而其应用在区块链技术出现后才真正实现，该理论阐述了证明者如何在不提供有效信息的情况下通过验证者的验证。
- 目前零知识证明在区块链中的应用大致可以分为三类，分别为可信任初始化设置、通信设置以及无需可信初始化设置，相对而言可信初始化设置的应用性较差，无需可信初始化设置扩展性较好，现阶段零知识证明协议偏向于无需可信初始化设置方向开发。
- Zcash 作为第一个使用零知识证明的区块链系统，通过 zk-Snarks 实现了证明者与验证者之间零交互验证，目前交易模式分为透明交易、Sprout 模式以及升级后的 Sapling 模式。
- zk-Snarks 通过将普通验证问题转化为复杂多项式的方式实现证明者与验证者之间的零知识证明验证。
- 平衡隐私安全保护与利用保护隐私“作弊”将是未来零知识证明重要发展方向之一。

## 1 零知识证明介绍

零知识证明（Zero-Knowledge Protocols）是上个世纪 80 年代由麻省理工研究人员 S.Goldwasser、S.Micali 及 C.Rackoff 提出的一种理论。该理论的核心是阐述证明者如何在不向验证者提供任何有效证据的条件下证明某个论断是正确的。《How to Explain Zero-Knowledge Protocols to Your Children》中举了一个经典的例子，文中以《阿里巴巴与四十大盗》的故事为例，阿里巴巴想在不告诉强盗打开宝库大门方法的前提下证明他是可以打开大门的，这个时候可以采取的方式是让强盗与门保持足够远的距离，他念动咒语开启宝库大门，这时即使强盗不知道开门的方式仍然可以证明阿里巴巴知道开门的方式。

通过上述例子可以了解到零知识证明特征属性主要为三点：

- 完备性：证明者如果能够可以通过重重验证，验证者则可以判断证明者的真实性。
- 零知识性：通过验证的过程中，证明者不可以想验证者透露具体有用的相关信息。
- 合理性：这个验证方法是独一无二不可伪造的，除此之外没有其他方式可以通过验证。

满足这三点则可以实现零知识证明的验证方式。

## 2 零知识证明协议分类对比

虽然零知识证明并非新概念，但是实际上一直未出现契合的应用场景。直到区块链的出现，零知识证明才真正显示出其发展潜力。

自比特币问世以来，区块链技术一直处于蓬勃发展阶段。作为去中心化的账本，区块链的安全性始终都是人们最为关注的。尽管地址匿名化对用户隐私起到了一些保护作用，然而由于链上

数据透明化，攻击者依旧可以通过对数据的分析、对网络的拓扑盗取用户的隐私。零知识证明与区块链相结合使得用户隐私安全性得到大幅度地提升。

目前密码学的专家们已经通过零知识证明开发出很多种协议，包括最早被使用于 Zcash 中的 zk-Snarks（Groth16）以及后来开发的 Plonk、Sonic、Starks、DARKs、Bulletproof、SuperSonic 等。部分特点对比如下图所示：

| 协议 | 设置 | 验证成本 | 验证时间 | 验证体积 | 备注 |
|---|---|---|---|---|---|
| Snarks | 可信设置 | 高 | 短（1-10ms） | 小（0.2kb） | 验证昂贵、需可信第三方、证明时间在1-2min |
| Sonic | 通用设置 | 较高 | 中 | 固定 | |
| PLONK | 通用设置/可更新可信设置 | 较高 | 较短（比Sonic快5倍） | 较大（1kb） | 升级后多方参与可信设置、使用多项式承诺 |
| Starks | 无需可信设置 | 较高 | 弹性 | 大（100-600kb） | 扩展性好，证明时间与数据集的关系为线性增加，验证时间则呈对数关系增加 |
| Supersonic | 无需可信设置 | 较高 | 较短（75ms） | 较小（10-20kb） | 可解决复杂度高的交易 |
| Bulletproof | 无需可信设置 | 较高 | 复杂度越高消耗时间越长 | 小 | 复杂度不高的交易，性能比较好 |

*来源：公开资料 表格制作：Gate.io 研究院*

从表格中可知，目前的零知识证明协议根据可分为三类：

- 可信设置协议：最早的 Snarks 协议使用的是可信设置，例如 Groth16 协议。Groth16 可以以很小的证明高效的完成验证。而其缺点是使用的公用参考字符串通过一次性可信设置创建，这也意味着参考字符串无法升级，一旦程序出现问题，哪怕只是很小的问题也需要重新部署整个电路。另外新的应用也无法直接使用原本的可信设置，需要建立新的可信设置。同时可信设置会生成一些“有毒废料”（toxic waste），如果“有毒废料”没有被及时处理的话有可能对协议的安全性产生影响。
- 通用设置协议：Sonic 是最早的通用设置协议，在传统 Snarks 协议之上加以改进，只需初始时进行一次可信设置即可。这也就意味着参考字符串可以升级无需重新部署整个电路，新的应用也可以重复使用这个可信设置。然而，在验证成本同样不低的情况下，验证时间与验证体积都要超过 Groth16。
- 无需可信设置协议：无需可信设置协议在 Sonic 的基础上进一步改善。公用参考字符串被

公开同时不会产生“有毒废料”，提高安全性的同时，降低了使用的复杂度。然而取消可信初始化设置会导致证明的数据量增大，Starks 的验证体积更是可以超过 100kb。虽然 SuperSonic 与 Bulletproof 比 Starks 好很多，但是在验证时间和验证体积上仍要超出 Snarks。

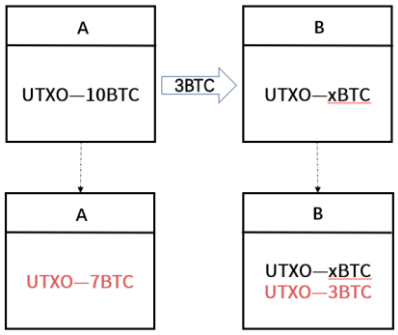
针对不同的应用场景，各类零知识证明协议都有其优点，下文将以 zk-Snarks 为例详细介绍零知识证明在区块链系统中的实现原理。

## 3 零知识证明于区块链中的应用场景 Zcash

### 3.1 Zcash 使用的交易模型

Zcash 是首个使用零知识证明的区块链系统。作为以保护隐私为重点的区块链系统，Zcash 的交易方式与比特币的交易方式是不同的，比特币系统当中使用的是 UTXO（Unspent Transaction Output）的方式进行记账。

举例来说 A 拥有 10 个比特币，此时 A 想转 3 个比特币给 B，那么转账的方式并不是以“10-3”、“x+3”的方式进行交易。假设 A 手中的只拥有一个价值 10 比特币的 UTXO，那么首先这个 UTXO 将被销毁并生成一个价值等于 7 比特币的 UTXO 与一个价值等于 3 的 UTXO 并将价值等于 7 的 UTXO 发放给 A，价值等于 3 的 UTXO 发放给 B，这样交易就完成了。




来源：公开资料 表格制作：Gate.io 研究院

虽然 UTXO 模型很稳定的记录了比特币的每一笔交易记录，然而所有的交易数据却也直接公布到了链上，也就是说所有人都可以看到这些交易数据，这在隐私安全性上来说是存在着隐患

的。

为了防止用户交易数据被“攻击者”利用，同时保证交易的真实可信性，Zcash 在原本的 UTXO 模型上进行了改进，使用 Note 作为其交易结构。如下表所示：

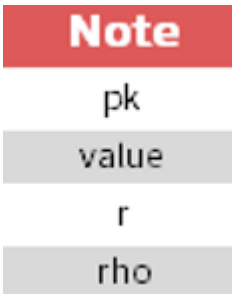
| Note commitments | nullifier set |
|---|---|
| h1=HASH (note1) | nf1 = HASH (r2) |
| h2=HASH (note2) | nf2 = HASH (r3) |
| h3=HASH (note3) | |

来源：公开资料 表格制作：Gate.io 研究院

与 UTXO 不同，Zcash 使用的 Note 并没有直接将交易后作废的原 UTXO 销毁，而是创建了一个作废列表对原来的作废后的 Note 进行存储记录。

Zcash 中通过两个表对交易进行记录，票据表（Commitment）负责储存有效的、可使用的票据（Note），与之相对作废表（Nullifier）表中则负责记录已经失效的票据。Note 当中包含如下四个元素：

- 用户的公钥 pk：由用户私钥 sk 生成，负责充当收款人地址
- Value ：记录 Note 的价值
- 随机数 rho：当 Note 被交易之后作废时，充当该 Note 在作废列表中的唯一编号（nf）。
- 随机数 r：对 r 进行 Hash 后可以生成 rho。




来源：公开资料 图片制作：Gate.io 研究院

当交易者想要将自己手中的资产交易出去的时候，可以通过发布作废 Note 的“编号”并将其放入 Nullifier 表中的方式进行交易。

以下图为例，A 想要给 B 转 7 个 ZEC：

a) 首先 A 找到了一张自己的有效 Note3，并使用自己的私钥 a_sk（A 的 Spending key）对

这张加密（Hash）的 Note 进行解密并发现这个 Note 中有 10 个 ZEC。

b) 接下来以与 UTXO 模型相同的方式把这张 Note3 分为一张 value 为 7 ZEC 的 Note4 和一张 value 为 3 ZEC 的 Note5。

c) 新的 Note4 与 Note5 中包含了生成的新随机数 r4 与 r5、通过 r4 与 r5 生成 rho4 和 rho5 以及在 Note4 当中置入的 B 的公钥 b_pk（B 的 Paying key）和 Note5 中置入的 A 的公钥 a_pk（A 的 Paying Key）。

d) 此时 Note3 不会被销毁。A 需要将 Note3 的 rho3（Hash(r3)）、经过签名加密 Note4 与 Note5 广播到全网，由矿工进行验证。

e) 矿工收到消息后在 Nullifier 表中进行查询，当没有发现 Nullifier 表中并不存在 A 广播的 rho3 并验证了 A 对 Note3 的使用权益后将 rho3（Hash(r3)）放入 Nullifier 表中，Note4 与 Note5 放入 Commitment 表中完成交易。

f) B 收到 Note4 后可使用 b_sk（B 的 Spending key）对 Note4 进行解密并获取其使用权。

| Note commitments | nullifier set |
|---|---|
| h1=HASH (note1) | nf1 = HASH (r2) |
| h2=HASH (note2) | nf2 = HASH (r3) |
| h3=HASH (note3) | |
| h4=HASH (note4) | |
| h5=HASH (note5) | |

*来源：公开资料 表格制作：Gate.io 研究院*

上文中简单的描述了 Zcash 中的交易过程，其中有几个问题需要注意。

首先，当使用 Zcash 进行匿名交易时，就意味着 Note 中的信息全部都是加密过的，也就是说公钥 pk 信息、value、r 以及 rho 都是隐秘的，除了交易方之外无法被他人看到。在上述交易过程中，当交易进行到步骤（d）时，A 需要将 Note4 与 Note5 分别使用签名加密才可以广播

到全网，如此一来 A 的 a_pk（A 的 Paying key）和 B 的 b_pk（B 的 Paying key）则会公开，这不符合上述描述中 a_pk（A 的 Paying key）与 b_pk（B 的 Paying key）的信息也是隐藏的特点。

其次，如果 Note 中的信息全部匿名化了那么验证者则无法通过常规的方式验证交易的真实性。

下文将就通过 Zcash 的交易模式对这两个问题进行说明。

## 3.2 Zcash 的交易模式

目前 Zcash 的交易方式分为三种：透明交易、Sprout 模式交易、Sapling 模式交易。

### 3.2.1 透明交易

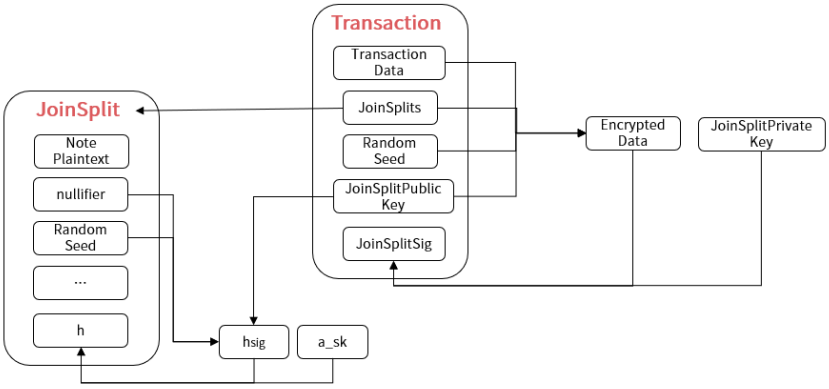
在 Zcash 的透明交易模式中，交易时无需对交易信息匿名化，用户可以直接使用透明的地址以及私钥完成交易的验证。

### 3.2.2 Sprout 模式

#### 3.2.2.1 Sprout 内部交易结构

Sprout（萌芽期）则是 Zcash 早期使用的匿名化交易方式。下图为 Sprout 内部交易结构：

Sprout内部交易结构



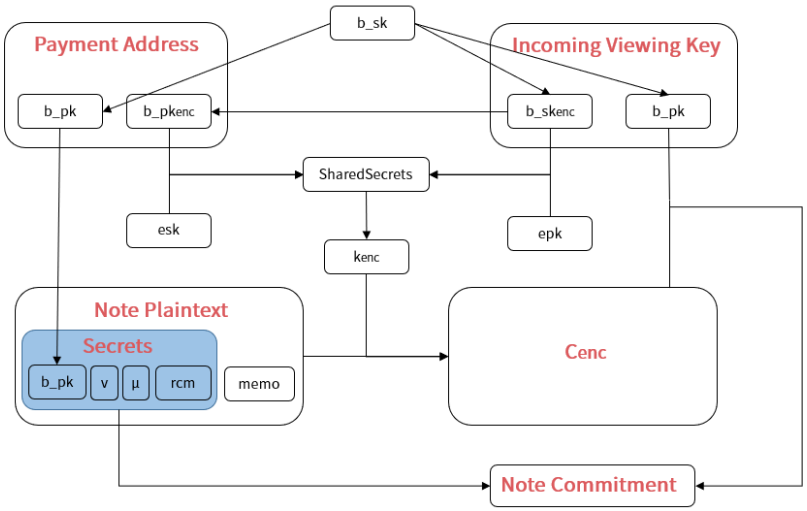

*来源：公开资料 图片制作：Gate.io 研究院*

仍以上文为例，A 与 B 进行交易，此时为了对交易添加签名同时不暴露公钥，Sprout 中通过产生随机交易密钥对“JoinSplitPublicKey/ JoinSplitPrivateKey”的方式对交易内部数据进行签名（JoinSplitSig)。由于密钥对是一次性随机生成的，所以生成的密钥对只能保证数据的完整性而不能保证数据安全性，如果攻击者重新生成一对密钥对的话那么就有可能对交易进行重新签名篡改交易数据，为了防止这种情况发生 Sprout 设计了 $h_{sig}$ 机制。$h_{sig}$ 是通过 JoinSplitPublicKey 与 JoinSplit 内部数据结合算出的 Hash 值，生成的 $h_{sig}$ 再与发送方 a_sk（A 的 Spending Key）进行加密保存在 JoinSplit 当中以确保其他人无法篡改随机生成的签名，从而保证了交易内容的安全性。

3.2.2.2 Sprout“线上”信息传递方式

除此以外 Sprout 还需要保证接收者能够在不暴露信息的情况下验证发送者发送信息的正确性。如下图所示：

Sprout“线上”信息传递




*来源：公开资料 图片制作：Gate.io 研究院*

图中交易地址中 b_pk（B 的 Paying key）是 B 通过私钥 b_sk（B 的 Spending Key）生成的公钥，$b\_pk_{enc}$（B 的 Encrypted Paying Key）则是 B 通过 $b\_sk_{enc}$（B 的 Encrypted Spending Key）生成的公钥。Incoming Viewing Key 中的 $b\_sk_{enc}$（B 的 Encrypted Spending Key）则是通过 b_sk（B 的 Spending Key）生成的私钥。

当 A 要将隐秘信息传递给 B 时，首先需要获取 B 的公钥 b_pk（B 的 Paying key）与 $b\_pk_{enc}$（B 的 Encrypted Paying Key），随后 A 将 b_pk（B 的 Paying key）置入到隐秘信息中（也就是上文中的 Note）并通过 Hash 加密后放入到 Note Commitment 中等待验证。接下来，A 随机生成密钥对“esk/epk”，其中 esk 与交易地址中的 $b\_pk_{enc}$（B 的 Encrypted Paying Key）结合生成 shared secrets，再通过 shared secrets 生成对称密钥 $K_{enc}$。

接下来 Note Plaintext 会通过 $K_{enc}$ 进行加密生成 $C_{enc}$ 并与之前生成的 epk 同时存放在 JoinSplit 中。此时 B 想要得到加密信息的内容就需要使用 $b\_sk_{enc}$（B 的 Encrypted Spending Key）与 epk 进行结合生成 shared secrets，并通过 shared secrets 获得 $K_{enc}$ 解密 $C_{enc}$。这个过程中由于只有 B 拥有 $b\_sk_{enc}$（B 的 Encrypted Spending Key），所以其他人无法通过

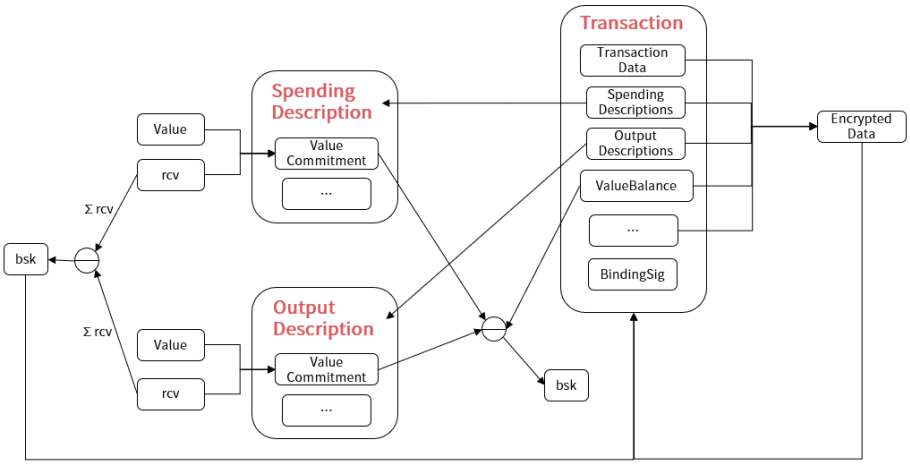
epk 获取密钥 $K_{enc}$ 从而保障了信息传递的安全性。最后 B 可以通过将自己 b_pk（B 的 Paying key）代入隐秘信息中的方式求得 Note 并对其进行 Hash 运算与 Commitment 中发送者上传的 Hash 值进行对比确认信息的真实性。

### 3.2.3 Sapling 模式

#### 3.2.3.1 Sapling 内部交易结构

Sapling（茁壮期）则为 Sprout 的升级版。交易速度提高的同时，内部结构变得更为复杂。Sapling 内部交易结构如下图所示：

Sapling内部交易结构①



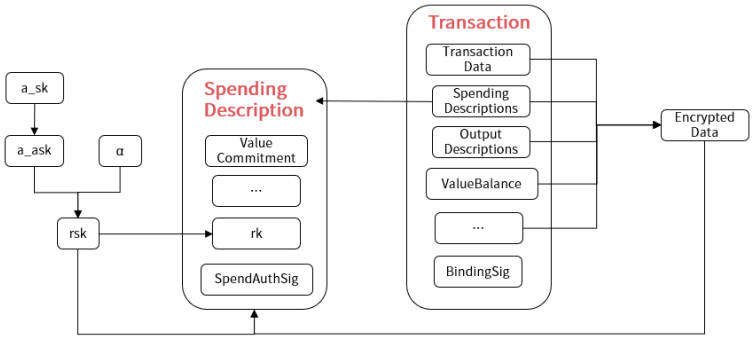

*来源：公开资料 图片制作：Gate.io 研究院*

Sapling 不采用 Sprout 中的 JoinSplit 结构进行交易，转而使用 SpendDescription 与 OutputDescription 分别表示消费与入账。其中 SpendingDescription 中包含了作废列表（Nullifier），OutputsDescription 中包含 NoteCommitment 以及 B 验证时需要的 $C_{enc}$。同样，为了保障交易的完整性与不可篡改性。A 同样需要对交易进行签名。图中 value commitment 是 value 同态隐藏得出的值，rcv 则是生成 value commitment 的随机数，通过

对 SpendDescription 中所有 rcv 的和与 OutputDescription 中所有 rcv 的和进行求差值的方式得到签名私钥 bsk；通过 SpendDescription 中所有 rcv 的和与 OutputDescription 中所有 rcv 的和求差再与 ValueBalance 做差的方式得到签名公钥 bvk。此时将交易中的信息进行加密再与 bsk 结合生成 BindingSig 以保证交易信息中数据的真实性与不可篡改性。

Sapling 中不止对交易进行签名，同时还会对 SpendDescriptions 中的信息进行签名。如下图所示：

Sapling内部交易结构②




*来源：公开资料 图片制作：Gate.io 研究院*

图中我们可以看到 A 通过 a_sk（A 的 Spending Key）生成了 a_ask（A 的 Authorization Spending Key），a_ask（A 的 Authorization Spending Key）与随机数α结合生成密钥 rsk，rsk 生成 rk 并将其置入 SpendDescriptions 中，随后 rsk 通过与交易信息中数据的 Hash 值进行结合生成 SpendAuthSig 如此一来变增强了 SpendDescriptions 的安全性。

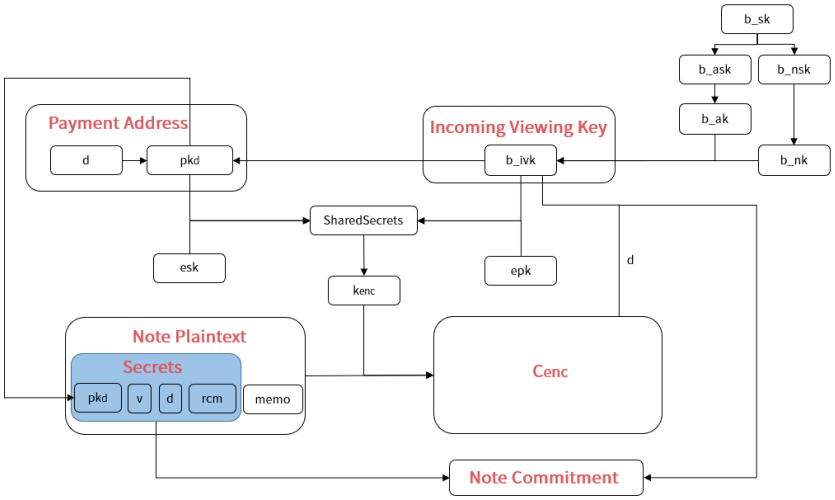
### 3.2.3.2 Sapling“线上”信息传递方式

与 Sprout 不同，A 传送信息给 B 时没有采用固定 b_pk（B 的 Paying Key）与 $b\_pk_{enc}$（B 的 Encrypted Paying Key）而是采用了更具有流动性方式提高消息传递的匿名性。同时 Sapling 对于密钥的权限做了进一步分化。

■ Incoming Viewing Key

B 通过 Incoming Viewing Key 可以查看 Note Plaintext 中的 A 传递过来的信息，如下图所示：

Sapling“线上”信息传递①




*来源：公开资料 图片制作：Gate.io 研究院*

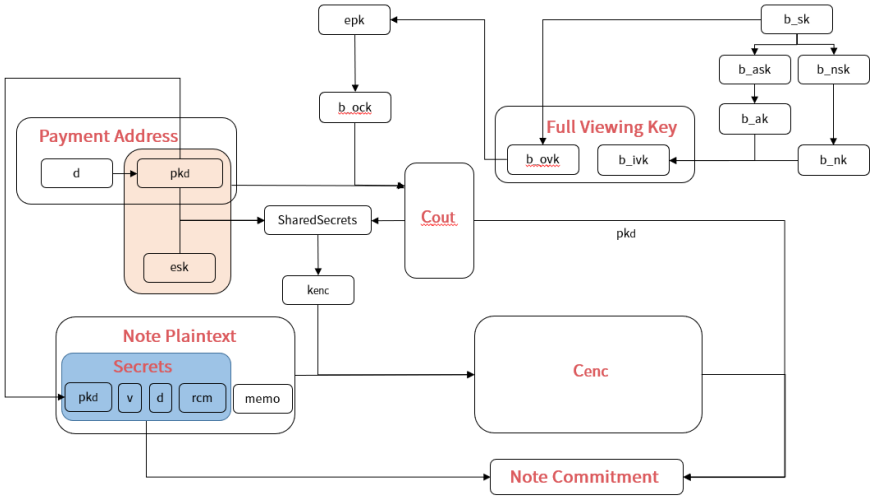
首先，B 会通过 b_sk（B 的 Spending Key）生成两个加密私钥 b_ask（B 的 Authorization Spending Key）和 b_nsk（B 的 Nullifier Spending Key），接下来 b_ask（B 的 Authorization Spending Key）和 b_nsk（B 的 Nullifier Spending Key）会分别生成 b_ak（B 的 Authorization Key）与 b_nk（B 的 Nullifier Key）并结合会生成 b_ivk（B 的 Incoming Viewing Key），b_ivk（B 的 Incoming Viewing Key）通过与随机数 d 的结合生成 pkd（Diversified Paying Key）作为 B 的接收地址。现在 A 将 pkd（Diversified Paying Key）置入到隐私信息当中，并通过 Hash 加密将 Hash 值放入 Note Commitment 当中以待 B 的验证。A 继续生成随机密钥对“esk/epk”，pkd（Diversified Paying Key）与 esk 的结合会生成 shared secrets，通过 shared secrets 可以生成对称密钥 $K_{enc}$ 随即对 Note Plaintext 进行加密形成 $C_{enc}$，接下来 A 将 epk 与 $C_{enc}$ 同时放入 OutputsDescription 中即完成了他的工作。B 接收到 OutputsDescription 中信息后可以使用 b_ivk（B 的 Incoming Viewing Key）与 epk 进行结合得到 shared secrets，并与 A 相同得到对称密钥 $K_{enc}$，$K_{enc}$ 通过对 $C_{enc}$ 的解密得到 Note Plaintext 的内容。有了随

机数 d，B 可以通过 b_ivk（B 的 Incoming Viewing Key）求得 pkd（Diversified Paying Key），并将其置入 Note Plaintext 中便可与 Note Commitment 中的 Hash 值对比验证 A 的交易信息是否真实。

■ Full Viewing Key

与 Incoming Viewing Key 不同，Full Viewing Key B 不仅可以查看 Note Plaintext 中的信息，也可以使用 b_fvk（B 的 Full Viewing Key）验证 OutputsDescription 中的全部信息。如下图所示：

Sapling“线上”信息传递②




*来源：公开资料 图片制作：Gate.io 研究院*

Full Viewing Key 中包含 b_ovk（B 的 Outgoing Viewing Key）、b_ak、b_nk 三个密钥，上文中提到，b_ak（B 的 Authorization Key）与 b_nk（B 的 Nullifier Key）结合会生成 b_ivk（B 的 Incoming Viewing Key），也就是说 Full Viewing Key 具备直接解锁 OutputsDescription 的能力。不仅如此，b_ovk（B 的 Outgoing Viewing Key）通过与 epk 结合可以直接生成对称密钥 b_ovk（B 的 Outgoing Viewing Key），A 便可以使用 b_ovk（B 的 Outgoing Viewing Key）对 pkd（Diversified Paying Key）与 esk 进行加密生成 Cout，接下来 B 可以使用 b_ovk（B 的 Outgoing Viewing Key）对 Cout 进行解密从而直接得到 pkd

（Diversified Paying Key）与 esk。至此，B 已经得到所有密钥并可查看 OutputsDescription 中的所有信息。

## 3.3 zk-Snarks 验证机制

通过复杂的交易机制 Zcash 成功的隐藏了用户之间所有的有用信息，保障了用户隐私的安全性，然而这也是对验证者验证交易真实性提出了挑战，此时便凸显出了零知识证明的重要性。零知识证明可以使验证者无需获得证明者有效信息的情况下并可证明交易信息的真实性与否，也就是说即使证明者可以在无需提供密钥解锁的情况下证明自己对于某笔资产的使用权。Zcash 中使用的零知识证明为 zk_Snarks（zero knowledge Succinct Non-interactive Argument of Knowledge），中文为“零知识简洁非交互式证明”。Zk_Snarks 可以在证明者与验证者无交互的情况下高效的完成某些验证。

### 3.3.1 验证过程

如上文中提到，证明者若想验证对某一笔资产的使用权时需要提供一份 Proof 给验证者。根据 Christian Lundkvist 对 Proof 描述，Proof 的生成过程如下：

首先证明者需要提供一个隐秘参数 w 并通过电门 C 转换成多项式：

$$\text{circuit } C(x, w)$$

接下来系统中 Generator 将会使用产生的多项式与随机参数λ生成对应密钥对 pk， vk 作为验证时使用的密钥对：

$$\text{Generator (C circuit, } \lambda \text{ is ? ):}$$

$$(pk, vk) = G(\lambda, C)$$

证明需要通过 Prover 进行计算生成 Proof π并交与验证者：

Prover(x pub inp, w sec inp):

π = P(pk, x, w)

最后 Verifier 拿到 Proof π以后会对多项式进行验证：

Verifier:

V(vk, x, π) = (∃ w s.t. C(x, w))

若验证成功则证明证明者确实拥有这笔资产的支配权，验证者则会确认交易，反之则失败。在这个验证过程中隐秘参数 w 只有证明者自己知道，验证者无法得到所谓的有用信息。λ参数的保密性同样至关重要，如果证明者得到λ参数，无论他是否知道隐秘参数 w 都可以生成假的 proof 并通过验证者的验证，也就是上文中提到的“有毒废料”。

### 3.3.2 验证逻辑

通过上文的操作可以得知 zk-Snarks 中大概的验证过程。下文将对其具体的验证逻辑进行简单的介绍。

zk-Snarks 的验证逻辑是清晰的，它的目的是“验证”而并非“求解”。基于此，zk-Snarks 并没有直接使用求解简单数学方程来验证一个问题的正确性，因为证明者求出解也就相当于暴露了自己的有用信息。zk-Snarks 采取的方式是将普通计算方程 P（Polynomial）问题[1]通过 QAP（Quadratic Arithmetic Programs，二次算术程序）的方式转化为 NP（Non-deterministic Polynomial，非确定性多项式）问题[2]。简单来说电门 C 的作用是将一个验证等式转换为多项式的一个过程：

V 神用例：w**3 + w + 5 =f= 35

（注：原文参数为 x，使用 w 为与上文统一）

这个方程很容易便可得到答案 w 等于 3。问题是现在证明者不想把 w 等于 3 这个答案（也就是秘密参数）告诉验证者。所以这个等式最终通过 QAP 转换成了向量积表示形式：

$$w.a(n) * w.b(n) - w.c(n) = 0$$

如此一来 w 便成功被隐藏了，但是想要用来进行验证仍然不够所以继续转化：

$$w.a(n) * w.b(n) - w.c(n) = x * Z(n)$$

其中 Z(n)表现形式为：

$$Z(n) = (n-1)* (n-2)* (n-3)*\ldots*(n-n)$$

n 的取值范围为 1, 2, 3…n。

至此多项式转化的步骤就算完成了。N 的问题通过电门 C 成功转化为了 NP 问题。为了方便下文的表示我们将多项式设为：

$$P(n) = w.a(n) * w.b(n) - w.c(n)$$

$$H(n) = x$$

$$\Downarrow$$

$$P(n) = H(n) * Z(n)$$

接下来证明者便需要通过验证者给出的 n 来计算 P(n)与 H(n)，并将计算值交与验证者验证 P(n) = H(n) * Z(n)的正确性。如果正确，验证者有很大的几率可以确认证明者确实知道隐秘参数 w，反之则判定证明者提供的 Proof 是假的。

目前为止验证的逻辑可以理解为验证者需要通过验证多项式左右关系来确定证明者是否真的对一笔资产拥有着使用权，但是这种验证方式若想安全的实现，有些问题仍需要解决，否则这种程度的验证带来安全隐患是非常大的。

首先 n 值的选取是多样性的，在 n 值不确定的情况下，证明者还是有几率通过制造假的 P`(n)使多项式左右成立进而完成验证。对于这个问题 zk-Snarks 采取的方式是对选取 n 的方式增加限制，验证者选取的 n 是通过随机抽样取得的，如此一来即使证明者撞到了 P`(n`) = P(n) = H(n) * Z(n)这种情况，n`恰好也等于 n 的几率还是很小的。

通过 n 值随机抽样使得证明者无法直接伪造 P`(n`) = H(n) * Z(n)，然而证明者在得到 n 的情况下，仍可以通过伪造出 H`(n)使得 P`(n) = H`(n) * Z(n)的方式通过验证者的验证，因为验证者的任务只是验证 P(n) /H(n) ?= Z(n)来确定结果对错。为了防止这种情况的发生 zk-Snarks 对 n 值进行了同态隐藏（Homomorphic Hiding）[3]。同态隐藏的目的是为了让 n 转换成 E(n)的同时仍然满足 P(E(n)) = H(E(n)) * Z(E(n))多项式的验证，只能得到 n 值同态值的证明者自然无法通过真实的 n 值构造假的 H`(n)通过验证。

至此问题仍未解决完，尽管对 n 值进行了同态隐藏，证明者还是有可能在不知道 w 的情况下制造伪证使得 P(E(n)) = H(E(n)) * Z(E(n))。由于 P(E(n)) = w . a(E(n)) * w . b(E(n)) - w . c(E(n)), 证明者可以通过伪造 P(E(n)) = w` . a`(E(n)) * w` . b`(E(n)) - w` . c`(E(n))的方式完美伪造 P(E(n))来通过验证者的验证。此时 zk-Snarks 采取了 KCA（Knowledge of Coefficient Test and Assumption）机制来阻止证明者伪造 a()、b()、c()等向量积。以 a(E(n))为例：

$$a(E(n)) \rightarrow a(E(n)), \lambda a(E(n))$$

原来的 a(E(n))转换成了 a(E(n))与λa(E(n))的向量对，并且λ无法通过λa(E(n))/ a(E(n))的除法计算出值。这样便限制了证明者伪造 a`(E(n))的可能，因为自己伪造的 a`(E(n))无法确定λa`(E(n)的值。

除此之外证明者在向验证者提交 Proof 时，Proof 中还需要计算 sum(λa(E(n))，λb(E(n))，λc(E(n)))的值以确保证明者不会使用 ya(E(n))，yλa(E(n))（倍化）的方式伪造向量积对。

向量积对解决了证明者伪造 a(E(n))的可能，但是同样引入了新的问题。

$$P(E(n)) = w.a(E(n)) * w.b(E(n)) - w.c(E(n))$$

⇩

$$P(E(n)) = w.(a(E(n)),\ \lambda a(E(n))) * w.(b(E(n)),\ \lambda b(E(n))) - w.c((E(n)),\ \lambda c(E(n)))$$

如上述公式所示，新的验证方式于单纯的加法同态运算之外又引入了加减法，此时 zk-Snarks 必须引入乘法同态运算法则才能解决验证问题。针对此问题 zk-Snarks 采用了双线性映射（bilinear map）[4]的方式使多项式的验证变为可能。

最终简化版验证多项式如下文公式所示：

$$w.(a(E(n)),\ \lambda a(E(n))) * w.(b(E(n)),\ \lambda b(E(n))) - w.c((E(n)),\ \lambda c(E(n))) = H(E(n)) * Z(E(n))$$

现在回到上文描述的 zk-Snarks 的验证过程来看验证逻辑便清晰了很多，其过程如下：

a) circuit C(x, w)的作用是将 w**3 + w + 5 == 35（例子）转换为 w . a(n) * w . b(n) - w . c(n) = H(n) * Z(n)的一个多项式。参数 x 为 H(n)。

b) Generator (C circuit, λ is ? ): (pk, vk) = G(λ, C)则是将 C circuit 中的 a(E(n))、(E(n))、c(E(n))通过λ转换为 a(E(n))、λa(E(n))，b(E(n))、λb(E(n))，c(E(n))、λc(E(n))。pk、vk 则为对应公共参数。

c) Prover(x pub inp, w sec inp): π = P(pk, x, w)则是通过 pk、w、x 生成了包含 P(E(n))、H(E(n))的 Proof π。

d) Verifier: V(vk, x, π) = (∃ w s.t. C(x, w))则是验证者最终接收到π，再利用哈希值 x、公共参数 vk 验证多项式的真伪。

综上所述，zk-Snarks 的目的是通过多次转换将能隐藏的信息全部隐藏起来并保证证明者在无法篡改被隐藏的验证内容的情况下验证最终的结果。

## 4 零知识证明对未来区块链隐私性发展的作用

隐私始终是区块链研当中的热门话题，其中如混币机制、二次加密等很多技术已经被广泛使用。然而这些技术始终绕不开需要通过密钥进行加解密的方式对证明者进行验证，零知识证明与区块链的结合犹如一扇新的大门，为区块链隐私保护提供了新的发展方向。证明者无需向验证者提供有用的信息便可验证真实性使得攻击者很难通过对具体交易数据以及资金流向进行分析从而找到用户真实的身份，这使得未来区块链安全性发展愈发有前景。

与此同时，零知识证明的愈发成熟也就代表着链上个人隐私保护与匿名性愈发强大。这为用户带来可靠安全性的同时也会给一些想利用区块链“作弊”的人提供机会。零知识证明为区块链隐私性提供安全保障的同时，如何平衡隐私性保护与“违规”也将是未来区块链发展道路上需要解决的重要问题之一。

## 5 总结

零知识证明与区块链技术的结合为用户隐私保护机制的发展打开了一扇新的大门。长久以来零知识证明难以实现落地应用，而区块链技术的出现使零知识证明的优点得到了充分发挥。Zcash 作为第一个结合零知识证明的区块链系统实现了证明者与验证者之间的零交互验证，验证者无需证明者提供任何有用信息便可实现对其的验证，提高了交易用户之间的隐私安全性。Zcash 将交易方式分为透明交易与隐私交易两种，隐私交易经过发展从早期的 Sprout 模式到升级后的 Sapling 模式不断进行优化，性能上得到了显著提高，用户体验感亦得到了改善。由于 Zcash 的隐私交易模式是通过对用户密钥进行不断加密实现的，验证者无法得到证明者的地址以及验证密钥，因此零知识证明协议是 Zcash 区块链中核心的技术模块。Zcash 中使

用 zk-Snarks 作为其零知识证明协议，zk-Snarks 则是通过将普通验证问题转化为复杂多项式的方式完成证明者与验证者之间的零交互验证。

未来零知识证明的研发更倾向于无需可信初始化设置方向，这可以使得使用零知识证明的区块链系统在提高隐私保护机制的同时拥有更好的扩展性。然而，零知识证明为保护区块链隐私提供了可行性的同时亦有可能为一些想利用区块链“作弊”的人提供了机会，如何平衡隐私保护与安全监控将是零知识证明未来发展道路上的重点之一。

# 6 参考资料

## 6.1 参考资料

1. Zcash Protocol Specification
   https://github.com/zcash/zips/blob/master/protocol/sapling.pdf
2. 赵晓柯. (2010). 浅析零知识证明. *硅谷, 000*(016), 36-37.
3. 李康, 孙毅, 张珺, 李军, 周继华, & 李忠诚. (2018). 零知识证明应用到区块链中的技术挑战. *大数据, 4*(1), 2018006.
4. Quisquater, J. J. , Quisquater, M. , Quisquater, M. , Quisquater, M. , & Berson, T. A. . (1989). How to Explain Zero-Knowledge Protocols to Your Children. *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. DBLP.
5. Hopwood, D., Bowe, S., Hornby, T., & Wilcox, N. (2016). Zcash protocol specification. *GitHub: San Francisco, CA, USA*.
6. Khovratovich, D. (2019). Bulletproofs.
7. Quadratic Arithmetic Programs: from Zero to Hero
   https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649
8. How Transactions Between Shielded Addresses Work
   https://electriccoin.co/blog/zcash-private-transactions/

9. Zcash – 图解 Transaction 结构

https://www.liankexing.com/notetwo/29131

10. Zcash - 各种密钥和签名，你懂吗?

https://blog.csdn.net/turkeycock/article/details/97571552

## 6.2 名词解释

[1] P 问题：Polynomial，P 问题就是可以有一个确定型图灵机在多项式时间内解决的问题。

[2] NP 问题：Nondeterministic Polynomially，中文译为非确定性多项式问题，与 P 问题相反，是指一个复杂问题不能确定是否在多项式时间内找到答案。

[3] 同态隐藏：Homomorphic Hiding，是指将某个值通过特定的函数生成新的数值，新的数值具有原来数值的特性，即带入原数值成立的等式，使用同态函数加密后数值进行带入同样成立。

[4] 双线性映射：Bilinear map，在数论中，一个双线性映射是由两个向量空间上的元素，生成第三个向量空间上一个元素之函数，并且该函数对每个参数都是线性的。使用双线性映射可实现乘法同态。

因出具该研究报告，特做出如下声明：

- 本研究报告是内部成员通过尽职调查和客观分析得出的结论，旨在对零知识证明于区块链中的落地应用进行分析总结，并不能完全以此来预测未来零知识证明于区块链隐私安全机制中的发展情况。
- 本研究报告非衡量研究对象本身价值、以及其发行代币价值的工具，不构成投资者做出最终投资决策的全部依据。

本研究报告中引用的项目资料来源自内部认为可靠、准确的渠道，因为存在人为或机械错误，信息均以获取时态为准。内部成员对研究报告中所依据的相关资料的真实性、准确度、完整性以及及时性进行了必要的核查与验证，但对其不做任何明示或暗示的陈述或担保。