摘要

本篇白皮书描述了一个开放的、建立在链下可验证服务端与链上可靠的智能合约之上的零知识证明随机数引擎zkRandom(zero knowledge random)的实现方案，zkRandom的本质是随机数预言机。本文首先通过分析现有的中心化和非中心化随机数生成机制的不足，来说明zkRandom存在的意义。然后通过逐步介绍zkRandom的工作流程、随机数计算算法、抗攻击防作弊机制让大家真正了解zkRandom。zkRandom保证了整个随机数获取过程是不可破解的、随机的、透明可追溯的。在zkRandom的作用下，服务商无须担心用户对其结果进行破解，用户也无须担心服务商作弊，且整个过程是公开透明的。在这种模式下，用户可以与服务商建立起一个相互信任的良性循环。

1.简介

区块链处在一个确定性的、封闭的系统环境中，它不允许不确定的因素存在，任何一笔链上计算都需要经过多个节点达成一致后才可生效。区块链的确定性意味着链上随机数生成成为了一大难题。

近年来，随着智能合约的兴起，众多DApp 如雨后春笋般涌现出来，如金融衍生品交易平台、稳定币、竞技游戏、保险、预测市场等等。一般智能合约的执行需要触发条件，而当智能合约的触发条件为外部数据时，就需要用到预言机来为智能合约提供服务了。一个理想的工作流程是用户通过链下API 接口将数据发送给链下服务器，链下服务器调用链上预言机合约发送数据，预言机合约对数据进行验证，在验证通过后将数据发送给用户的智能合约。

根据随机数预言机目前的基础设施来看，随机数预言机的应用前景则更深更广，因为所有需要概率性计算的场景都需要用到随机数，尤其是在游戏领域。在区块链进入智能合约时代后，尤其是各种游戏类DApp 的兴起，随机数预言机也迎来了一个新的时代。

2 当前现状

2.1 中心化随机数生成机制的不足

中心化随机数生成机制所面临的两大主要风险是: 伪随机性和缺乏透明度。

现如今大多通过计算机生成的随机数都是伪随机数，是由随机种子和程序算法也就是某些公式函数生成的。对于同一随机种子与函数，得到的随机数列是一定的，因此得到的随机数可预测且有周期，不能算是真正的随机数。一旦被黑客掌握随机数种子和算法，就极有可能预测出随机数甚至直接控制随机数的生成，从而可能会造成用户和服务商的巨额财产损失。

缺乏透明度会让用户不公地面临服务商不正当行为所带来的风险，这里的风险主要是指服务商的恶意行为。以游戏为例，用户并不知道游戏服务商是否有绝对公平的生成随机数，也不知道游戏结果是否有按生成的随机数来决策，更不知道生成的随机数是什么、决策机制是什么。对用户而言，没有任何方法去追溯、验证游戏结果。很有可能用户每次的游戏是在服务商的操控下进行，游戏的输赢完全取决于服务商在后台的操控。随着时间的推移，用户会逐渐对游戏服务商失去信任。即使存在一些良心游戏服务商，用户也难以辨别。现实生活中不乏因为游戏机制不透明而被投诉的例子，比如暴雪旗下的三款拥有战利品抽取的游戏《守望先锋》《炉石传说》以及2.0 版本的《风暴英雄》就曾因为不公平问题屡遭用户投诉。

2.2 去中心化随机数生成机制的不足

去中心化预言机制所面临的主要风险是：可被预测和作弊。

区块链上的随机数需要满足分布式的特点，但追求随机的可验证性和要求确定性的区块链因性质不同很难融合，这也就导致了在链上不能直接生成随机数，亦或生成的随机数很容易就被黑客破解（比如根据区块高度、区块时间戳计算出来的随机数可轻松被预测），造成资产的损失。

以链上卡牌竞技类游戏为例，卡牌竞技类游戏的核心是不可预测、可验证的随机数，从而决定最终结果，但是在链上是无法生成随机数或者说在链上的随机数是可以被预测和破解，一旦随机数被预测和作弊，就很容易导致资产被盗，现实中也存在因为随机数问题资产被盗的例子，比如 BetDice、Dice2.Win。由此可见一个不可预测不可作弊的随机数预言机至关重要。

2.3 zkRandom的设想

中心化随机数生成机制是伪随机的、非透明的，而去中心化随机数生成机制又存在可被预测的问题。基于以上原因，我们提出了零知识证明随机数引擎zkRandom，zkRandom是基于VRF（Verifiable Random Function）的理论基础，设计了一种透明的、不可篡改的随机数生成标准，并提供生态工具，如浏览器、SDK等，使Dapp项目方使用链上随机数的成本更低、更可信。zkRandom工作原理是：

1. Dapp用户对自己的业务数据进行签名，然后将业务数据和签名凭证发放给Dapp合约进行预登记；
2. Dapp合约登记业务数据时，将业务数据的hash和发起人等信息发送至zkRandom申请Nonce值；
3. 随机数生成方通过监听zkRandom的合约事件，根据事件内容通过业务数据hash及nonce值进行签名，将签名凭证及部分数据发放给验证合约；
4. 验证合约验证原始数据、业务Hash、随机数凭证的有效性，验证通过后生成随机数返回或发送给Dapp合约。

生成随机数种子和产生随机数凭证方根据不同的场景需求，角色会不同，不同的角色在获取随机数的方式上也会有所不同。zkRandom的方案中，不同角色产生的随机数安全级别也不相同，依据随机数安全级别的高低，提供三种随机数生成方式，从低到高分别是：

1. Dapp方管理私钥，直接生成可验证随机数；
2. zkRandom托管私钥，由zkRandom生成可验证随机数；
3. 由去中心化节点服务提供可验证随机数预言机服务。

Dapp可根据自身的需求和成本接受能力，选择最符合实际情况的接入方式。安全级别越高的随机数，使用成本越高。

zkRandom除了提供生成可验证随机数的基础链上服务外，会提供完整的生态设施，如随机数浏览器、Dapp-sdk等用于快速接入zkRandom服务及完成随机数的生成、使用和验证。

3 zkRandom运行机制介绍

3.1 公布PublicKey

随机数服务方接入zkRandom服务，需要在zkRandom的合约中公布服务商的公钥信息，zkRandom会通过该公钥，对产生的随机数种子进行合法性验证。

服务商提交PublicKey至zkRandom是服务开始的第一步，具体流程如下：

1. 服务商通过椭圆曲线加密算法（ECDSA-secp256k1）生成一对秘钥；
2. 公钥提交至zkRandom进行公开，私钥由服务商保存；
3. 公钥公布成功后，zkRandom会生成一个编号，对应该公钥。在提交业务数据生成随机数时，需要包含该参数用于找回公钥。

3.2 业务数据组建规则

业务数据由固定前缀、业务哈希、自增数Nonce和固定后缀组成。固定前缀和固定后缀与zkRandom预言机所在链有关；业务哈希生成规则由使用zkRandom的服务商自定义，需要注意的是服务商要能根据这个业务哈希区分出不同的业务场景；Nonce是通过注册PublicKey时返回的ID从zkRandom合约接口获取的，在每次获取完随机数后，该ID下对应的Nonce值自增。为了提高并发性能，zkRandom允许同一服务商注册多个PublicKey。

3.3 随机数获取

随机数生成由用户、随机数种子服务商、zkRandom共同完成。流程如下：

1. 用户发起业务请求至Dapp合约，登记业务数据，并向zkRandom发起随机数申请；
2. 随机数种子服务商监听zkRandom的生成随机数的事件，根据事件及业务数据hash、nonce等数据生成随机种子凭证；
3. zkRandom通过签名凭证恢复公钥，验证随机数种子合法性，通过后，使用签名凭证生成随机数；
4. zkRandom通过回调的方式通知到Dapp合约，并记录相关的业务事件；

在步骤1中，向zkRandom发起随机数申请后，zkRandom会返回一个nonce值，可根据nonce值是否被使用，判断系统是否存在作恶的情况。

在步骤2中，因为zkRandom是无差别服务的中间件，所以不需要完全的业务数据，只需要业务数据的hash值即可；

在步骤4中，交付随机数会有两种方式：

1. 直接通过函数的返回值获取随机数，该场景主要由Dapp服务商直接发起请求，减少中间环节的一种方式，可降低成本，提高效率；
2. 通过notify接口获取随机数，该场景用于构建更高可信的服务场景，需要用户先通过业务数据和nonce值确定随机种子，及Hash值来生成随机数；

3.3.1 如何保证结果随机性

最终的随机数取决于服务商签名凭证，而服务商的签名凭证是由业务数据、zkRandom的nonce值和服务商的私钥共同决定的。无论业务数据发生多细微的变化，在使用摘要哈希算法对其签名计算出来的签名凭证一定会发生变化，且业务数据的产生需要按照随机数引擎规定的方式来，每次产生的业务数据都不同，这也就保证了每次签名出的V 值都会是不同的。因此，即使对于同一用户而言，其每次利用zkRandom得到的随机数结果都会有很大差别，签名值生成的机制保证了结果的随机性。

3.3.2 如何保证不可破解

不可破解体现在结果不可被黑客预测，结果的不可预测是由业务数据的约定性、业务数据的变化性以及签名值的不可推测性共同保证的。

业务数据的约定性体现在黑客不可随意构造业务数据，客户方、服务方、随机数引擎都需要根据规定的规则构建业务数据。倘若任意一方不遵守规则都将导致验证不通过，这将直接对其信用造成影响，从这里就已经彻底封堵了通过捏造数据来左右最终结果这条路（根据业务数据组建规则得知业务哈希是黑客唯一可能改变的数据，但业务哈希如果被恶意篡改了zkRandom会直接校验不通过）；业务数据的变化性是由组建业务数据时的nonce保证的，同一业务数据只在一次请求中有效，即使被黑客抓取到了数据，也只能是失效了的数据；签名值的不可推测性体现在签名值是由服务方使用私钥对业务数据进行摘要哈希签名计算出来的，这对于不知道密钥的任何人来说完全不可预测。

3.3.3 如何防止服务方作弊

业务数据及zkRandom产生的nonce共同生成的hash，该hash为用户和Dapp达成共识的数据，服务方根据hash进行签名，虽然可以提前预知随机数种子，但却改变不了结果，服务方能做的只有拒绝提交数据。一旦服务方恶意拒绝用户的随机数生成请求，将会面临用户的质疑。此外服务方受随机数引擎的规则约束，一经作弊，用户通过浏览器的记录对随机数的产生进行审计，通过一定的约束可以督促服务方遵守规则。

服务方作弊主要来自两种情况：

1. 篡改数据，不修改最终随机数生成结果；
2. 提前预知结果后，通过拒绝服务的方式干扰Dapp的结果。

针对第1种情况，zkRandom会对原始数据，即用户提供的种子数据，进行合法性验证。如果服务方通过篡改原始数据，无法通过zkRandom的合法性检查。

针对第2种情况，zkRandom会通过内部的监督机制，对超时为处理的随机数请求进行补偿，并对服务方进行惩罚。同时，用户也可以通过浏览器对服务方或Dapp方进行安全审计。

3.4 定期公布私钥

服务商可通过定期更换并公布旧私钥的方式进一步提高自身的可信度。zkRandom平台在获取到服务商公布的旧私钥后可对历史数据做进一步的校验。当然，这一步并不是必须的，因为前面的步骤已经足够验证最终生成的随机数的可信度了。