WEEKLY SHARING

Plonk：Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge
普世非交互式知识论证的拉格朗日基排列

2022.09 STEPH

背景

对于SNARKs算法，绕不开的一个点就是中心化的Trust Setup，也称之为CRS(the Common Reference String)。而无论是PGHR13, Groth16,还是GM17算法，它们的CRS都是一次性的，不可更新的。即，不同的问题将对应着不同的CRS，这在某些场景下，会变得比较麻烦。这些存在的问题，变成了PLONK，SONIC这类算法的一个优势，它们算法虽然也需要中心化的可信设置，但是它的CRS具有一定的普适性。即，只要电路的大小不超过CRS的上限阈值，一些证明问题就可以共用一个CRS，这种CRS称之为SRS(universal Structured Reference String)，关于SRS的定义，详细的可参考SONIC协议里的第3小节。(了解Groth16的小伙伴，很熟悉CRS - Common Reference String。SRS是类似的意义，只是这些数据是Universal的，Structured Reference String。)PLONK算法继用了SONIC算法的SRS的思想，但是在证明的效率上，做了很大的提升。

Groth

不同的zk-SNARK方案也各有所长。zk-SNARK方案可以被分为【通用】与【非通用】zk-SNARK，PLONK与Groth16分别是其中的典型代表。

更多阅读

https://eprint.iacr.org/2019/099.pdf (Sonic paper)

https://mp.weixin.qq.com/s/XfXoo3r8PiqBH_g2dqI_Qg (零知识证明算法之PLONK)

https://www.8btc.com/media/6587859 (零知识证明算法之PLONK)

ABSTRACT - 抽象的概念

虽然通用零知识证明协议的改进研究已进行了多年，但PLONK（相对于更早但更复杂的SONIC以及最近的Marlin）带来的是一系列的改进，这些改进可能会总体上大大提高这类证明的可用性及进展。它支持通用或可更新的可信设置（trusted setup），而且相比 Sonic 有显著的性能提升。这将会是在真实环境中使用零知识证明的一个巨大进步，并且不会由于可信设置而产生信任问题。

研发者提出了一个通用的 SNARK 结构，具有完全简洁的验证，并且显着降低了证明器的运行时间（在完全简洁的验证器模式下，根据电路结构，组幂运算比 [MBKM19 - SONIC] 大约少 7.5-20 倍）。

与 [MBKM19-SONIC] 类似，研发者的新方案也依赖基于 Bayer 和 Groth [BG12] 的置换参数。 但是，专注于“对子组的评估，而不是单项式的系数”； 这可以简化排列参数和算术步骤。

原PLONK协议的优化
优化1：支持通用或可更新的可信设置
优化2：多项式承诺

优化1：支持通用或可更新的可信设置

由于 zk-SNARK 的实际部署，让结构化参考字符串 (SRS) 以“通用和可更新”的方式可构造变得非常有趣。 这意味着相同的 SRS 可用于关于某个有界大小的所有电路的陈述； 并且在任何时间点，SRS 都可以由新的一方更新，因此直到那时为止，所有更新者中只有一方的诚实是健全性所必需的。 为简洁起见，让我们将这种设置过程称为通用的 zk-SNARK。

（1）通用性，即可信设置与应用无关，仅需要一次可信设置，可以满足所有（电路门数在一定限制内，目前zksync电路门数在 2^20 ~ 2^26之间）的应用；

（2）可更新： setup参数可以任意更新，只要有一个可信参与者，即可保证可信设置的安全性。

这意味着两件事：

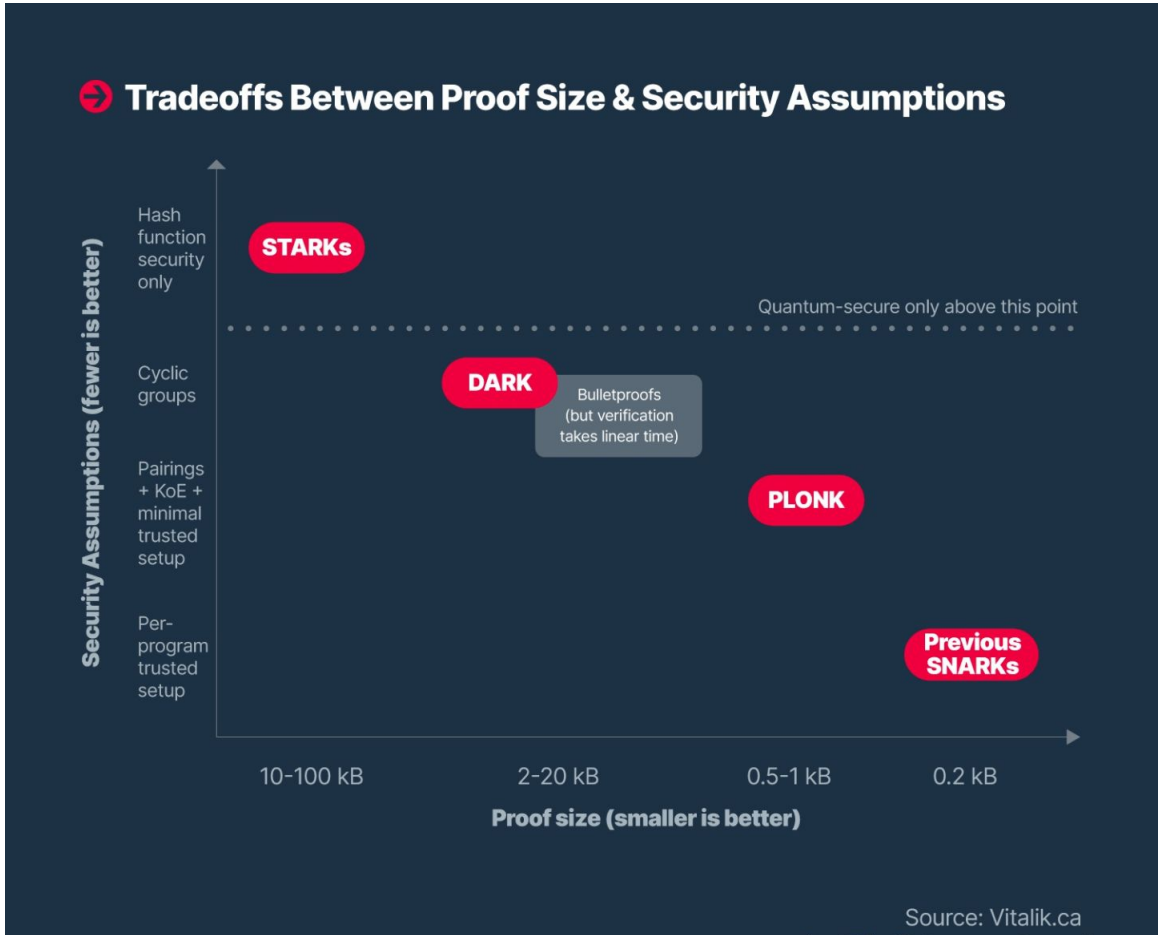
1、你想证明的不是每个程序都有一个独立的可信设置，整个方案只有一个可信的设置，在此之后，您可以将该方案用于任何程序(在进行设置时选择的最大尺寸)。

2、有一种方法可以让多方参与受信任的设置，只要其中一方是诚实的，那么该设置就是安全的，而且这种多方参与的过程是完全按顺序的:第一个人参与，然后是第二个，然后是第三个……所有参与者甚至不需要提前知道;新参与者可以把自己加到最后。这使得可信设置很容易拥有大量参与者，从而在实践中非常安全。

即使对于超过一百万个门的电路，PlonK 证明也能够在 23 秒内在消费级硬件上构建。 这标志着通用 SNARK 效率的重大进步，现在可用于广泛的现实世界用例。

优化2：多项式承诺

它所依赖的“奇特密码学”是一个单一的标准化组件，称为“polynomial commitment”（多项式承诺）。PLONK使用基于可信设置和椭圆曲线对的“Kate commitments”（Kate承诺），但你也可以用其它方案替换它，例如FRI（这将使PLONK变成一种STARK）或者DARK（基于隐藏顺序组）。这意味着该方案在理论上与证明大小和安全性假设之间的任何（可实现的）权衡兼容。

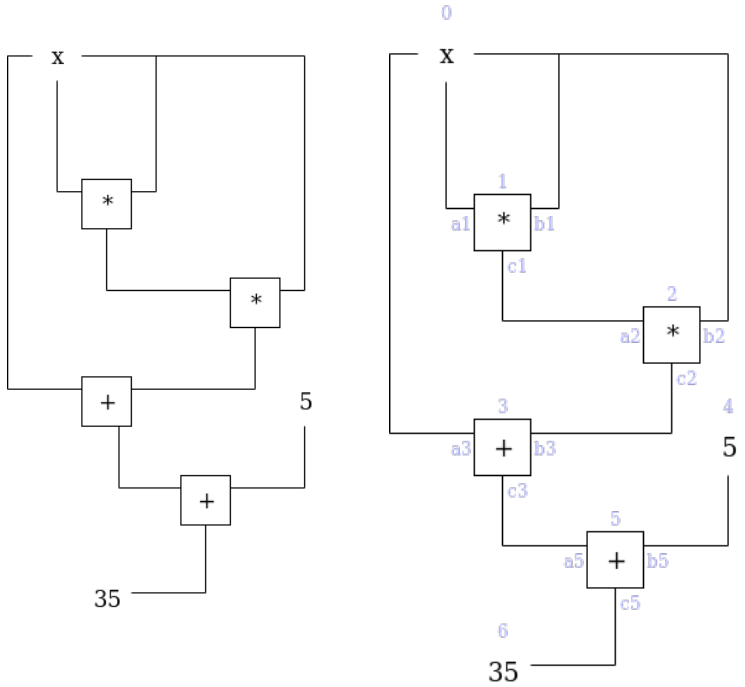


这意味着需要在证明大小与安全性假设之间进行不同权衡的用例（或者对这个问题有不同思想立场的开发人员），仍然可以为“算术化”共享大部分相同的工具（把一个程序转换成一组多项式方程的过程，然后用多项式承诺来检验）。

PLONK是如何工作的

让我们从解释PLONK的工作原理开始，我们只关注多项式方程而不立即解释如何验证这些方程。PLONK的一个关键组成部分，就像SNARKs中使用的QAP一样，这是一个转换问题的过程，形式是“给我一个值X，我给你一个特定的程序P，这样当X作为输入进行计算时，给出一些具体的结果Y，” 放到问题“给我一组满足一组数学方程的值”当中。程序p可以表示很多东西，例如，问题可能是“给我一个数独的解决方案”，你可以通过将P设置为数独验证器加上一些编码的初始值并将Y设置为1（即“是的，这个解决方案是正确的”）来对其进行编码，一个令人满意的输入X将是数独的有效解决方案。这是通过将P表示为一个带有逻辑门的加法和乘法电路，并将其转换为一个方程组来完成的，其中变量是所有线上的值，每个门有一个方程（例如，乘法为x6 = x4 * x7，加法为x8 = x5 + x9）。

下面是一个求x问题的例子，这样P(x) = x**3 + x + 5 = 35 (提示: x = 3):



我们按如下方式给门和线贴上标签：

在门和线上，我们有两种类型的约束：门约束（连接到相同门之间线的方程，例如a1 * b1 = c1）和复制约束（关于电路中任何位置的不同线相等的声明，例如a0 = a1 = b1 = b2 = a3 或者c0 = a1）。我们需要创建一个结构化的方程组，它最终将减少到一个非常少数量的多项式方程组，来表示这两个方程组。

PLONK是如何工作的

在PLONK中，这些方程的设置和形式如下（其中，L = 左，R=右，O=输出，M=乘法，C=常数）：

$$(Q_{L_i})a_i + (Q_{R_i})b_i + (Q_{O_i})c_i + (Q_{M_i})a_ib_i + Q_{C_i} = 0$$

每个Q值都是一个常数，每个方程中的常数（和方程数）对于每个程序都是不同的。每个小写字母值都是一个变量，由用户提供：ai 是第i个门的左输入线，bi是右输入线，ci是第i个门的输出线。对于加法门，我们设置：

$$Q_{L_i} = 1, Q_{R_i} = 1, Q_{M_i} = 0, Q_{O_i} = -1, Q_{C_i} = 0$$

将这些常数插入方程并进行简化，得到ai+bi-oi=0，这正是我们想要的约束条件。对于乘法门，我们设置：

$$Q_{L_i} = 0, Q_{R_i} = 0, Q_{M_i} = 1, Q_{O_i} = -1, Q_{C_i} = 0$$

对于将ai设置为某个常数x的常数门，我们设置：

$$Q_L = 1, Q_R = 0, Q_M = 0, Q_O = 1, Q_c = -x$$

你可能已注意到线的每一端，以及一组线中的每根线，显然必须具有相同的值（例如x）对应于一个不同的变量；到目前为止，没有什么能强迫一个门的输出与另一个门的输入相同（我们称之为“复制约束”）。PLONK当然有一种强制复制约束的方法，我们稍后会讨论这个问题。所以现在我们有一个问题，证明者想要证明他们有一堆Xai, Xbi以及Xci值满足了一堆相同形式的方程。这仍然是一个大问题，但不像“找到这个计算机程序的一个令人满意的输入”，这是一个非常结构化的大问题，我们有数学工具可用于“压缩”它。

更多阅读

Plonk零知识证明方案

https://www.jianshu.com/p/889b7e09ae9a

零知识证明 - 深入理解PlonK算法

https://learnblockchain.cn/article/2180

THANK YOU