关于RSA累加器的神奇魔力，通过以太坊创始人Vitalik的计算，使用RSA累加器后，原本每年2.5 GB的Plasma子链数据，可以被压缩到3.6 MB，压缩率达到了惊人的99.856%。

然而，这种技术在其最初的设计当中，需要用到可信设置，而来自斯坦福大学的应用密码学小组则发布了一篇题为《[用于IOP和无状态区块链的累加器批处理技术](https://eprint.iacr.org/2018/1188)》的论文，论述了一种通过**类组（ class group）**而**无需可信设置**的累加器方法，这些累加器可用于创建无状态区块链（指节点不需要存储整个状态，以确信哪些区块是有效的），以及用于实现高效的UTXO commitment。

在这篇文章中，以太坊区块链可扩展性和安全研究员Georgios Konstantopoulos对该论文进行了审查，并进行了相关总结，以下为译文：



在这篇文章中，我们将尝试深入探索RSA累加器，同时回顾一下斯坦福大学应用密码学小组最近发表的论文，这篇非常重要的论文由Benedikt Bunz，Ben Fisch和Dan Boneh共同撰写，其题目为《[用于IOP和无状态区块链的累加器批处理技术](https://eprint.iacr.org/2018/1188)》。

强烈建议大家复习下数学，以便更好地理解这一技术。

背景

自1994年以来，累加器便成为了学术界非常关注的一个话题。其类似于默克尔树（Merkle Tree），并被用于以密码方式承诺一组数据的知识。稍后，可通过发布证明来证明数据集中子集的成员身份。在默克尔树（Merkle Tree）结构中，证明被称为默克尔分支（或默克尔证明），并且承诺数据的大小是以对数形式增长的。

另一方面，累加器允许以恒定的大小证明成员身份，以及为多个元素批处理证明（默克尔树没有这一特性）。

本文的重点是描述RSA累加器构建区块、如何构建成员和非成员身份的证明，以及如何跨多个区块对它们进行批处理。这种特殊的技术，也在基于UTXO的Plasma中具有应用，并由此产生了Plasma原代变异体。设计一个允许在Plasma中压缩UTXO集的累加器，需要付出大量的努力。

*免责声明：为了简单起见，作者模糊处理了这篇文章中的注释（例如不包括G$中的$U、W\或模块化算术的mod N）。*

术语表（来自[1]的定义）：

**累加器**：“一个密码学累加器，其会产生对一组元素的短期约束承诺，以及对集合中任何元素的短期成员身份和非成员身份证明。”

**动态累加器**：“支持添加和删除具有O(1) 成本的元素累加器，其与累积元素数量无关。”

**通用累加器**：“支持成员和非成员身份证明的动态累加器。”

**批处理**：批验证n个证明，要比验证单个证明要快n倍。

**聚合**：在一个常量大小的证明中聚合n个成员证明。

**未知顺序组**：组的顺序是其集合中元素的数目。为了保证所提供的证明的安全性，需要生成一组未知顺序（否则累加器中使用的模数有已知的因子分解，并且可以创建伪证明）。生成它可通过多方计算完成，但如果生成方串通检索生成的数的阶乘，则这是不安全的。**它可通过使用**[**类组**](https://github.com/Chia-Network/vdf-competition/blob/master/classgroups.pdf)**在没有可信设置的情况下生成（注：这点是非常重要的）**。

隐序组的简洁证明

Wesolowski在[2]中提出了指数方案的知识证明，证明者试图说服验证者他们知道一个数字x，这样，在已知基数u下，使得u^x=w成立。

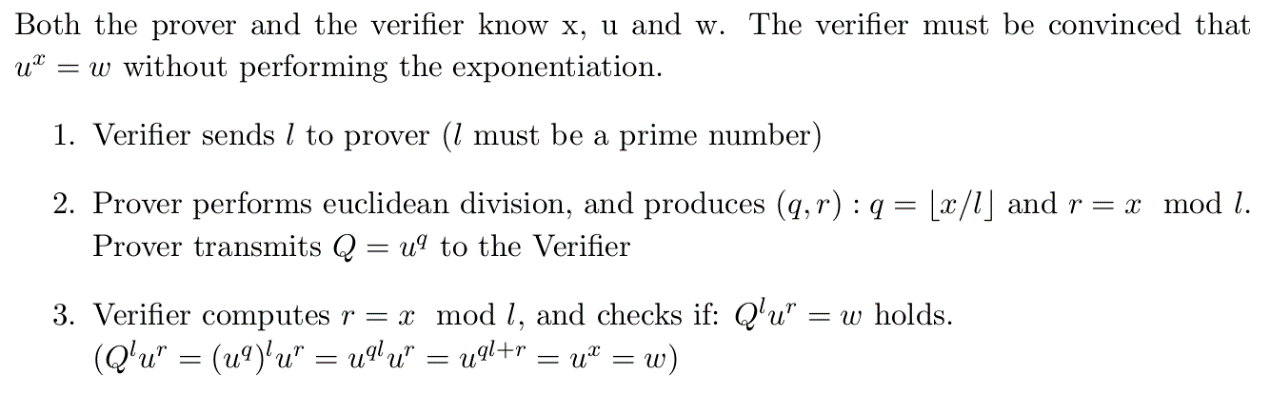
让我们举个例子，以2为基数（u=2），w=16，则得出x=4。我们怎么做？我们把X发送给验证者，它们必须执行2^4，并对照W检查结果。如果匹配，它们便会相信。以下两个步骤似乎很明显：

1. 验证者必须执行u^x ：对于大数字来说，这是一个代价高昂的操作；
2. 将x传输到验证者：x可能很大，因此传输它所需的带宽可能不小；

让我们看看有什么协议可以解决这一挑战。这些协议都是交互式的，这意味着验证者和证明者互相发送“挑战”（challenge），这些挑战在协议的后续步骤中会被使用，以确保协议的安全。

求幂证明（PoE，第3.1节）

首先，让我们看看如何让验证者信服，而不需要它们实际运行整个求幂运算。

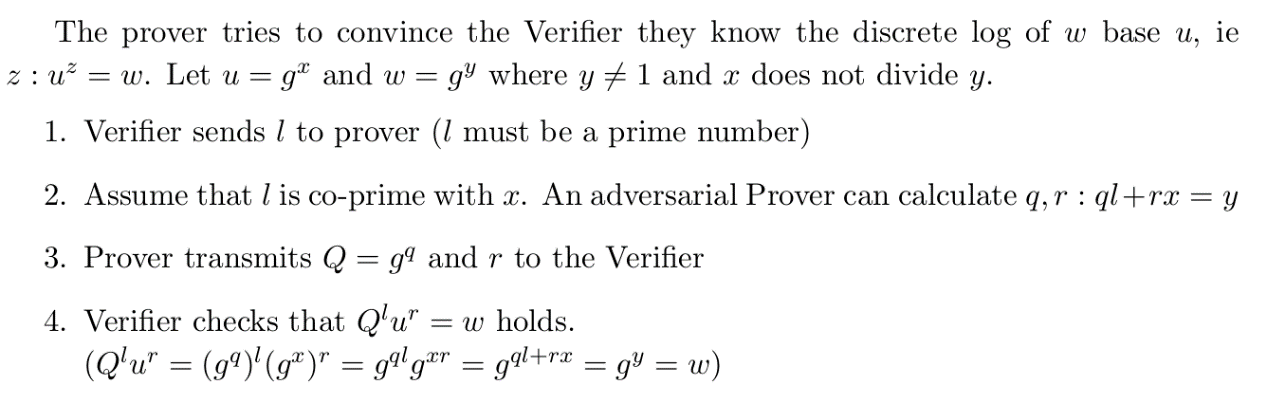


求幂证明（注：当前版本的论文中有一个小错误，在第8页中，作者设置q=g^q而不是u^q。

“只有当验证者能够比计算u^x更快地计算余数r时，该协议才有用。它解决了求幂问题，但仍然要求证明者向验证者传输一个潜在的大x，或者x是公开的。”

离散对数知识证明（PoKE, 第3.3节）

相比传输x，我们可传输r。证明变为（Q,r），而验证者必须另外检查r是否小于l（PoKE\*协议）。当对手可自由地选择基数u时，这会是不安全的！



验证者被证明者愚弄了，他们知道z: u^z=w，而不知道z！

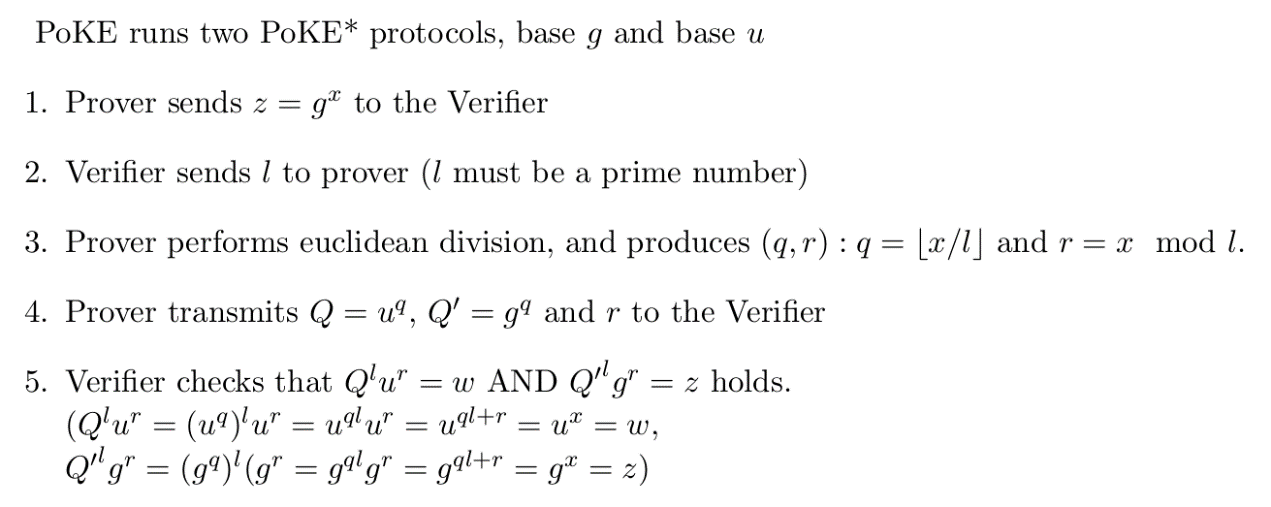
这里破解协议的细节在于，证明者选择了基数u=g^x，因此x与l是互质（co-prime）的。

我们可确定，上述协议适用于在[公共参考字符串（CRS）](https://en.wikipedia.org/wiki/Common_reference_string_model)中编码和固定的基数g，简单来说，各方事先就基数g达成一致，并且不能被对手任意选择。

该协议可通过以下方式进行修复：

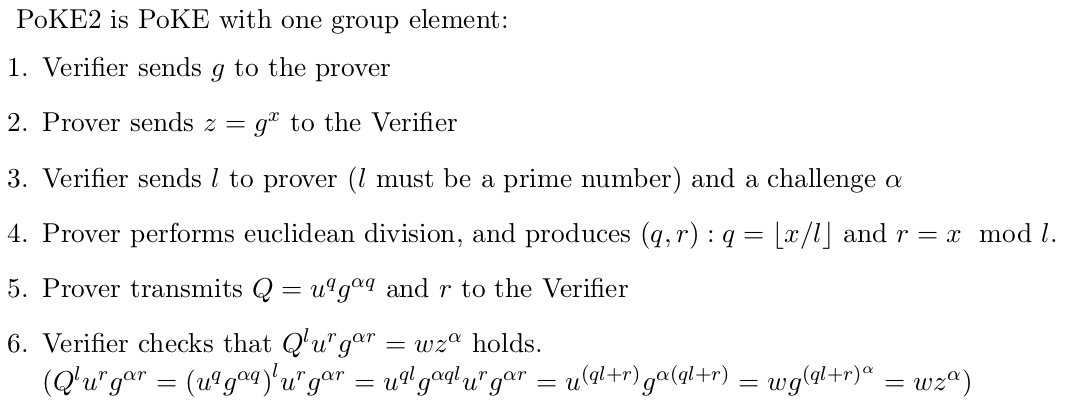
1. 对于固定的g，证明z=g^x离散对数x的知识；
2. 证明同一x也是基为u，w的离散对数；

所以最后的协议（PoKE）是：



证明现在是2组元素Q和Q’. 我们能做得更好吗？

将证明减少到一个组元素，这可通过添加其它交互步骤来完成：

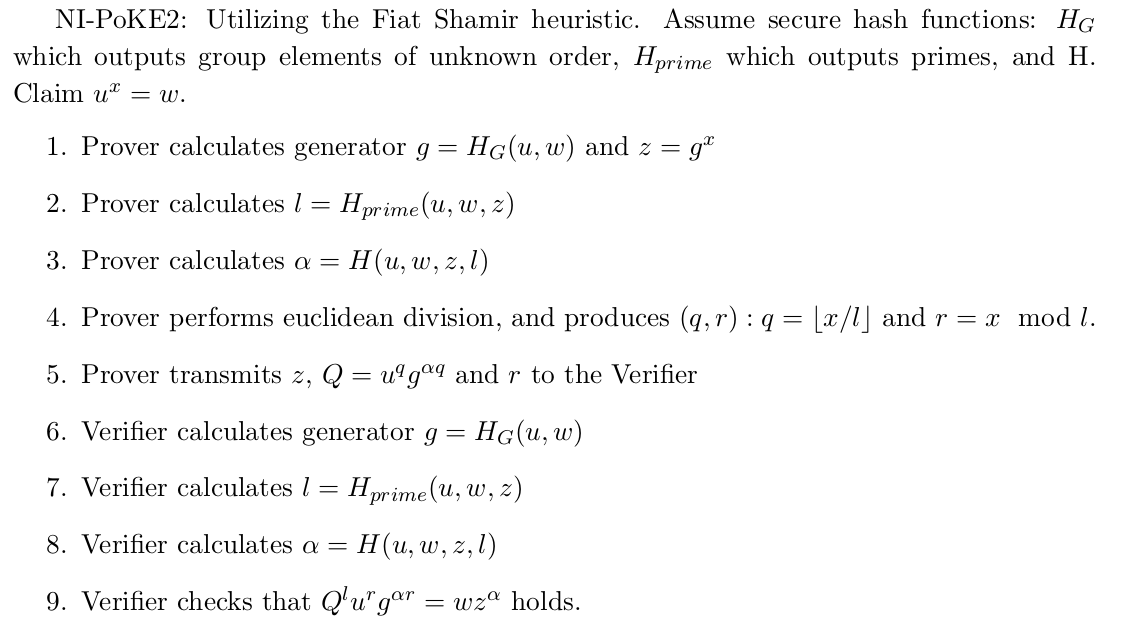


验证者需要发送一个额外的挑战\alpha，以便证明者无法创建假证明。

从交互式证明，到非交互式证明

在随机Oracle模型中，通过使用Fiat-Shamir heuristic，任何交互式协议都可变成非交互式的（假设我们有一个安全的随机性生成器，例如一个安全的加密哈希函数）。

PoKE2需要两个交互步骤，一个是由验证者挑选给证明者的生成器g，一个是发送挑战值l和a。相反，我们可以哈希当前的“抄本”（transcript），并使用输出作为这些值。因为我们是在随机的Oracle模型中操作的，所以如果这些值是由验证者挑选的（以防证明者作弊，或者它们来自哈希函数）则没有什么区别，因为这两者在统计上是不可区分的！



每一方生成挑战参数，而不需要交互，每次使用哈希函数和协议的当前抄本

上述技术涉及证明函数w=f（x）=u^x（标量值）的原像（preimage）知识。

这些技术也可以扩展到支持同态原像知识的证明，即证明长度n向量x的知识，使得φ（x）=w。

它们也可以在零知识下执行。对于已知g，PoKE需要发送g^x。在验证协议的正确性时，我们假设存在一个模拟器，该模拟器能够通过了解验证内容x来模拟g^x。这会泄漏信息，因此不是零知识的！论文作者所使用的技术包括屏蔽输入，这些输入通过使用一种类似Schnorr的协议和佩德森承诺（Pedersen Commitments）技术来得到证明。如果你并不熟悉这些术语，可先访问一下[这些内容](https://cryptoservices.github.io/cryptography/2017/07/21/Sigs.html)。

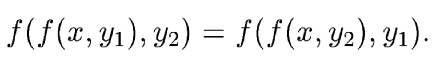
RSA累加器

我们在术语表中给出了累加器的定义。现在，我们将讨论支持批量成员证明和非成员证明的通用累加器的构造。

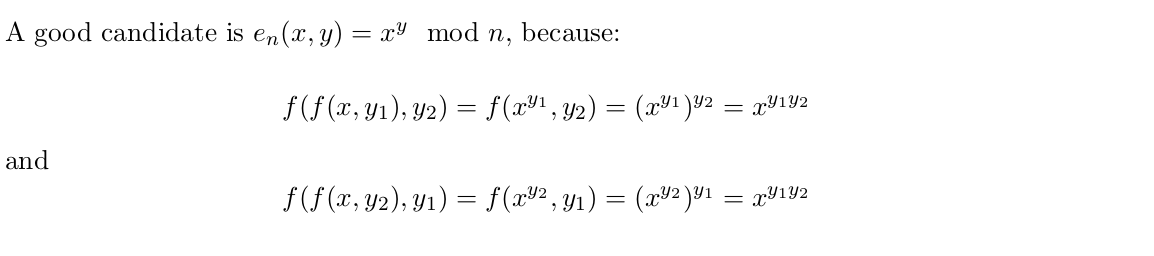
构造累加器需要从一组未知顺序中选择一个模数N，该模数N可以从RSA组中选择（例如，如果你信任RSA实验室，则为[RSA-2048](https://en.wikipedia.org/wiki/RSA_numbers" \l "RSA-2048)），也可以通过可信设置生成。

RSA累加器的初始状态，是从未知顺序g组中采样的生成器，这意味着累加器中的元素列表为空[].

如[3]所述，累加器必须具有准交换数学性质。



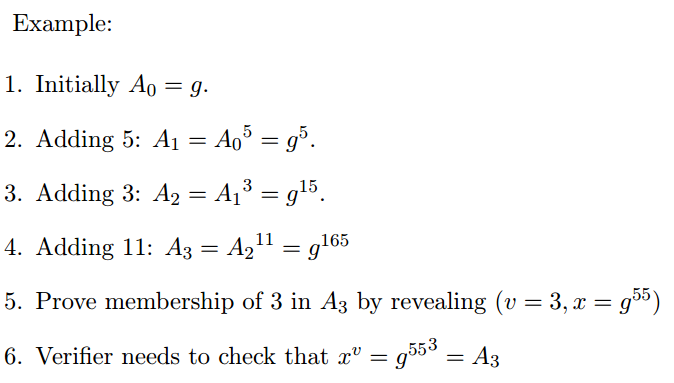
两个元素的准交换性质。



将元素x添加到累加器a，是通过将累加器提升到元素A’ = A^x mod N 来完成的。（为了简单起见，此处我们省略mod N）

证明成员身份

证明累加器中某个元素的成员身份，需要显示该元素的值和验证因子。



验证因子或共同因子是累加器中所有值的乘积（除了我们正证明的包含值）

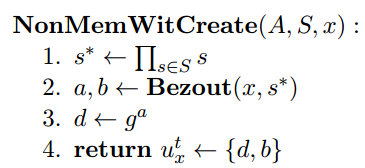
（值，验证因子）对是包含在集合中的证明。

“如果这个值是一个很大的数字，将其传递给验证者，并且求幂的成本是不可忽略的呢？”

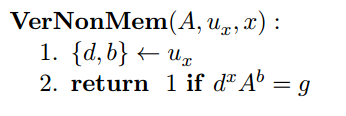
这就是上面的NI-PoKE2协议的由来。相比发送上述所述的证明，我们可以证明验证因子的知识，其会提供一个有效的证明！这似乎不太可能，因为我们的示例很简单，但我们将在下面的批处理成员证明部分中，看到可能发生的情况。

证明非成员身份

非成员证明需要计算我们证明的元素的[Bezout系数](https://en.wikipedia.org/wiki/B%C3%A9zout%27s_identity)和集合中元素的乘积。在[这里](https://brilliant.org/wiki/bezouts-identity/)，我们可以找到关于这个主题的优秀指南。

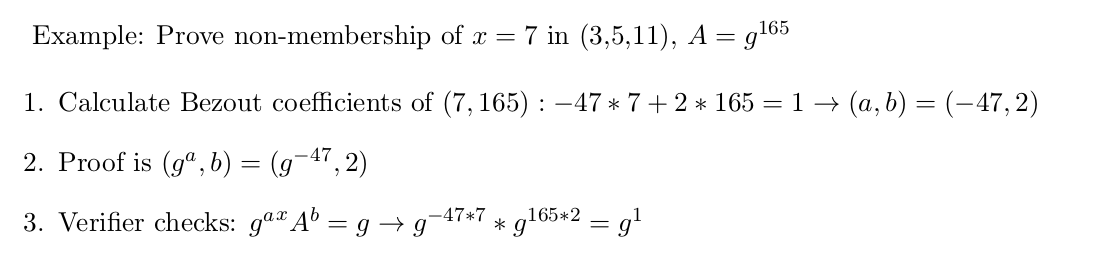


创建一个非成员验证因子



验证一个非成员验证因子

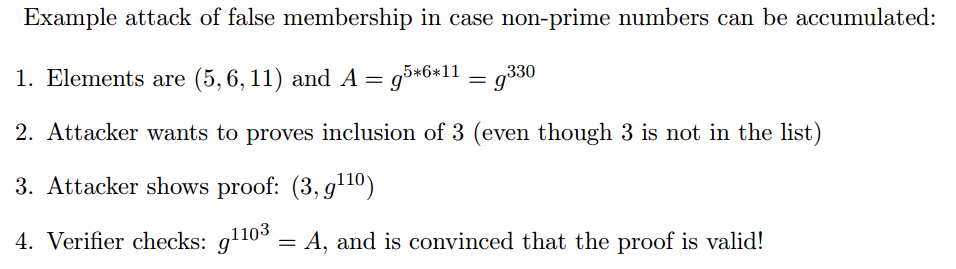
下面就是一个例子：



“Vitalik Buterin还提出了一种证明非成员身份的方法，其中他提到的想法是不确定的。（没有提供其安全性的证明，因此如果你要使用它，可能要小心了！）”

哈希素数

奇质数（即不带2的素数）既需要用于知识证明协议，也需要用于累加器元素。如果累积的元素不是素数，那么对手可在元素不在集合中的情况下，愚弄验证者包含了该元素。



非累积元素的有效成员证明！

因此，我们必须将累积元素限制为素数，否则对手可以证明包含累积元素的任何因子（在这种情况下，证明包含3是因为它是6的因子）。

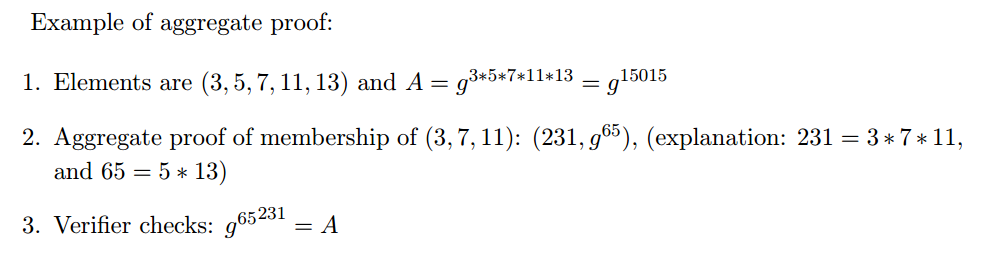
此外，如果NI-PoKE2中的挑战值l不是素数，那么我们会得到另一种协同因子攻击，其中攻击者可以计算q，r，从而愚弄验证者包含了某个元素，这类似于 PoKE\*攻击。

聚合和批处理证明

回忆一下定义：

**聚合：**将多个证明组合在一个常量大小的证明中；

**批处理：**一次性验证多个证明，而不是单次地验证所有的证明；

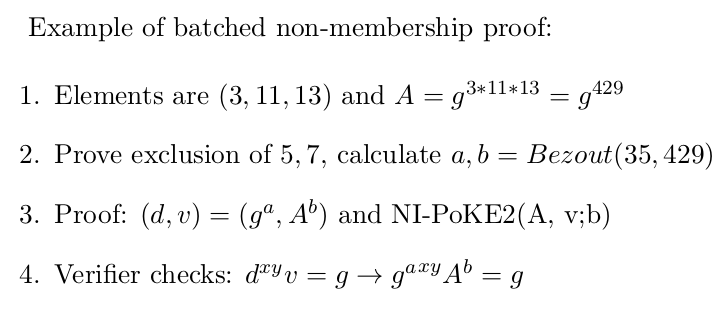
聚合和批处理成员证明是是非常简单的，只需将被证明的值相乘，并为它们提供一个共同因素： 

聚合成员身份证明

我们可以很快看出，如果我们想要为很多元素创建成员身份的聚合证明，那么值在传输时会变大，并且验证者需要执行昂贵的指数运算。为此，我们使用NI-PoKE2来证明我们知道因子g^65,而不需要向验证者发送231，验证者也不需要进行昂贵的指数计算（我们实现了批量验证！）

批处理非成员证明，是通过计算元素 (a’, b’)积的Bezout系数来完成的，然后具有与以前相同的证明（g^a’，b’）。合并验证因子的大小，与提供两个独立的验证因子大致相同。

这可以通过将证明设置为(g^a’, A^b’) 来解决。为了确保安全，证明者还必须提供一个NI-PoKE2，以证明对b'的了解。



第3步的NI-PoKE2是为了安全考虑，否则对手可能会设置v = g \* d^(-xy)，并在不知道b的情况下愚弄验证者。

这可以通过应用NI-PoE来提高效率，这样验证者就不需要在最后一步执行求幂运算。

在一个恒定大小验证因子的情况下，目前并没有有效的算法来聚合非成员证明。

移除可信设置

所有的指数运算都关于模N，这是一个具有未知素数因子分解的数字。这是因为提供的所有证明，都在未知顺序组的通用组模型中（第2页），并且需要[强RSA假设](https://en.wikipedia.org/wiki/Strong_RSA_assumption)和[自适应根假设](http://crypto.stanford.edu/~dabo/papers/VDFsurvey.pdf)。

在不知道相关私钥的情况下生成公钥是困难的。如论文第2页中的[2]所述，我们可通过执行安全的多方计算来创建所需的数字，但必须相信参与受信任设置的各方没有串通来找回秘密。Wesolowski在[2]中通过所谓的“类组”（class groups）而提出了另一种选择：

“一个更好的方法是使用虚二次序（imaginary quadratic order）的类组。事实上，通过选择一个随机的判别式，我们可以很容易地生成一个虚二次序，当判别式足够大时，就无法计算类组的顺序。”

目前，Chia正在为有效计算这种“类组”而进行竞争，同时还提供了一份有关其背后所需理论的[综合性论文](https://github.com/Chia-Network/vdf-competition/blob/master/classgroups.pdf)。

结论

如果你能看到这里，那么祝贺你！

我们简要介绍了RSA累加器的工作原理，以及如何构造有效证明累加器中元素的成员身份和非成员身份的方案。原论文作者还提供了构造向量承诺的方法，其在不同的索引处有成批的opening，这不是默克尔树的特征。作者构建了第一个能够实现O（1）opening的向量承诺方案（这里的opening指证明在承诺中某一指标上某一要素的价值）。

应用例子

这些累加器可用于创建无状态区块链（指节点不需要存储整个状态，以确信哪些区块是有效的）。它们还可用于实现高效的UTXO承诺，允许用户在不知道整个UTXO集的情况下发布交易。最后，向量承诺可用于创建简短的交互式Oracle证明，这一过程比使用默克尔树（Merkle Tree）结构更为有效。

下一步是什么？

这是一篇非常好的论文，它介绍并形式化了很多可用于区块链结构扩展性的原语。

特别是，RSA累加器已吸引了Plasma研究社区成员的关注，他们正设法利用它来压缩Plasma Cash的UTXO历史。最近，ethresear.ch上已经有很多关于如何构造它的文章。因此，在下一篇文章中，我们将对当前的方案、它们的优缺点以及哪一个方案最为有效进行盘点。

对于可利用向量承诺的非同质 Plasma 结构，我也非常感兴趣。

谁知道呢，也许已经有人在做这个了？

关于这一话题，接下来的文章题目会是： Plasma中的RSA累加器分类。

相关阅读：

[1] https://eprint.iacr.org/2018/1188

[2] https://eprint.iacr.org/2018/623

[3] http://www.michaeldemare.com/pubs/owa.pdf

原文：https://medium.com/@gakonst/deep-dive-on-rsa-accumulators-230bc84144d9  
作者：Georgios Konstantopoulos  
翻译：洒脱喜