# 1.程序说明

在加密开发者的工具带上有一种鲜为人知的技术。密码学累加器是一种具有几种奇异性质的原语，可用于构建各种零知识证明系统。密码学累加器最早是由 Josh Benaloh 和 Michael de Mare 提出的，原始论文《One-way accumulators: A decentralized alternative to digital sinatures (extended abstract) 》 于 1993 年发表在欧洲密码学会议（EUROCRYPT）上。一个密码学上的累加器是一个单向的隶属函数。它可用于识别一个候选是否为一个集合的成员，且不会在过程中暴露集合中的成员。这个概念因 Zerocoin的提出重新受到重视。实际上，主要类型的累加器类似于RSA，并且基于模幂运算。

我们可以把加密累加器看作是一个在集合上工作的超级hash函数。像SHA-3这样的常规hash函数接受单个消息并输出固定大小的hash值。然而，累加器接受一组值，并将它们转换为一个同样大小不变的数字。从某种意义上说，累加器是Merkle树和Bloom过滤器的非对称密码学近亲。

考虑一个最简单的秘密协议，Alice发布了一个秘密消息。稍后，她将整个消息透露给Bob，Bob可以验证它是否符合承诺。现在，如果Alice使用的是累加器而不是hash，她可能会选择只显示消息的一部分或全部部分。

## 1.1成员资格证明

具有累加器值的集合。稍后我们将看到几个将数据编码为集合成员身份的示例。现在，让我们假设Alice将她的消息转换为一组单词并发布了相关的累加器（看起来像一个hash）。

她可以选择一些词，并产生一个证明，这是另一个不变大小的数字。它允许Bob验证显示的单词的完整性：它们确实属于提交的集合。但是，Bob对其他保密部分一无所知。这个属性称为零知识。

通过计算集合中所有值的乘积来处理集合。如果我们将输入数据映射到素数，它们的乘积将唯一地表示集合。否则，就会产生混淆：{2,6}会给出与{3,4}相同的12。集合将用大括号（如{x}）编写，它们的乘积用大写字母（如X）编写。

让我们选择一个模N（两个大素数的乘积），一个生成元G（任何其他素数）。

对于一组秘密值{u}及其乘积u，累加器C通过模幂运算计算。C是一个和N差不多大的数。

C=GU modN

接下来，让我们从{u}取一个子集{r}来展示。为了计算证明，我们实际上需要{u}的所有其他值，那些仍然是秘密的，记为{s}。在乘积形式中，我们有R\*S=U。证明P为：

P=GS modN

然后我们把{r}和P透露给Bob。他将计算C'并验证其是否等于承诺C：

C'=PR modN

通过替换P，我们看到C'必须等于C：

C'=GSR=GU=C modN

## 1.2非成员身份证明

令人惊讶的是，鲍伯也可以用相反的话来证明鲍伯的操作是不可能的。

举个例子：鲍勃问第一个词是“猫”，还是“狗”。爱丽丝的回答证明这两者都不是，鲍勃可以证实这一事实。但他对这个词的含义还是没有别的线索。

非成员身份的证明有点棘手，但这里有一个想法。

以一个集合{x}为例，它不属于已提交的{u}的元素。因为素数集合，所以X和U的GCD（最大公约数）将是1。Alice将使用扩展的欧几里德算法来提供系数，以允许Bob验证这一事实。

{x}{u} 素数集合 所以gcd(X,U)=1

扩展的欧几里算法：aU+bX=1

非成员证明：

d=G-b modN

(a,d)

验证：

C=GU modN

Ca=dxG modN

Ca modN=GUa modN=G1-bX modN=G-bXG modN=dXG modN

## 1.3混合成员证明

{x}中有属于{u}的元素记为{r}，也有不属于{u}的元素

因为素数集合 所以gcd(X,U)=R

扩展的欧几里算法：aU+bX=R

混合成员证明：

d=G-b modN

(a,d)

验证：

C=GU modN

Ca=dxGR modN

Ca modN=GUa modN=GR-bX modN=G-bXGR modN=dXGR modN

## 1.4安全性

该系统依赖于与RSA相同的假设：

离散对数的硬度（Bob从C求U，或从P求S）

RSA问题（Alice伪造了一个伪证明P）

整数因式分解的困难（求N的因子）

这个系统的一个缺点是它要求模N是两个素数的乘积，但是Alice一定不知道这些因子。有两种方法。

Bob可以生成N并让Alice用它来证明，他会信任他们的。但是Bob可以利用他的因子来伪造累加器和证明，因此其他人不会信任它们，而Alice具有可否认性。

但我们真正想要的是公开使用这个系统，为此我们需要一个可信的设置。这意味着有些计算机必须从随机因素中生成N，而忽略这些因素。如果一个人相信这些因素不是由任何人拯救的，那么他就可以信任这个体系。

事实上有这样一个数字：RSA-2048。嗯，很有可能。早在1991年，RSA实验室就公布了一份数字列表，并向任何人提出挑战，以20万美元的奖金计算这些数字。这是罗恩·里维斯特（RSA中的R）在谈论这件事。

N=RSA\_2048=25959084747474747474747474742432432400483985714292921262042042027777777836043662020707595555562664852558588078444444124124124124124951508218929292991747474741841841842828282828842007284848449494992689292929287878787272727272727272727272727272727187187187187187187187187187187979797187187187979718718718718718718718718718774742458686917171717151515151515151515151515151515151515151515151515151515151515151515151515151515151515151515162282828282828620435767984233871847744479207399343236584823824281198163815010674810451660377306056201619676256133844143603833904414952634432190114657544454178424020924616651572335077870774981257724676292638635638328991215483143816789988504044536402357381951378636564391212010397122822120720357；

## 1.5应用

我们如何实际实现一些有用的东西？将数据编码为集合。这里是一种方法和一个示例实现。

让我们以一个政府机构为例，该机构对其活动的机密报告进行归档。随着一些丑闻的爆发，公众要求提供一份特别的报告，以核实该机构当时的行动。然而，该机构辩称，有些细节过于敏感，比如其代理人的身份，会用一个隐喻性的黑色记号笔将其编辑掉。

让我们有一个过程，在这个过程中，该机构将与机密文件相关联的累加器作为承诺发布。当时间到了，机构会公布部分内容，并证明它是一个承诺文件的成员。

报告123 1970年1月1日

今天，探员做了正确的事。

让我们用一些编码将文档表示为一个位序列：00001101…

下一步，我们为每个位指定一个唯一的数字。我们的累加器使用素数，因此我们从2、3、5开始枚举素数……根据位的值，我们将数字分为两个列表：

文档：0 0 0 0 1 1 0 1…

零：2 3 5 7--17-…

一：---11 13-19…

使用累加器，我们提交与零相关联的数字列表（2、3、5、7、17，…），但不提交给其他数字（11、13、19不包括在内）。

一份完整文件的数字将达到数百万，但这没问题。对于n位消息，时间复杂度为O（n.log（n））。实际上，我们看到的是几个kbit/s的天真实现。

单凭承诺，我们无法判断是否包括任何数字，因此我们对文件内容一无所知。

现在，该机构希望公布该文件的部分内容，但不公布其他部分。让我们标记一些要隐藏的部分：

显示：0 0 0 0 1…

与部分文件一起，它还发布了一份证明，证明：

-2、3、5和17属于提交集，表示0。

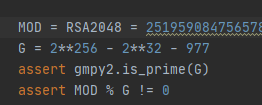
-13和19不属于集合，代表1。

-但是它没有提到7和11：它们可能属于也可能不属于集合。

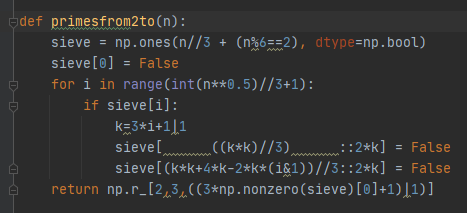
从那里，我们可以验证文件中透露的部分是真实的，但我们对仍然保密的部分一无所知。

## 1.6成员非成员证明程序

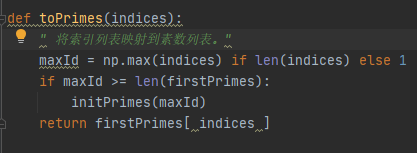
参数设置



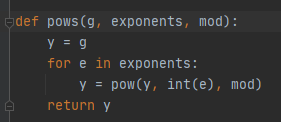
素数生成函数



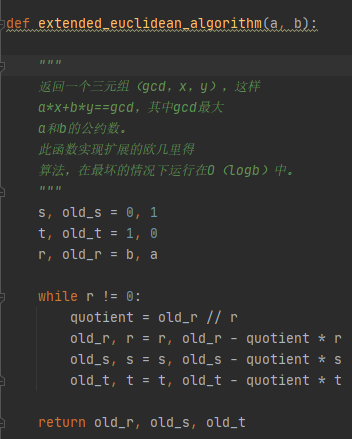
将普通数组映射到素数



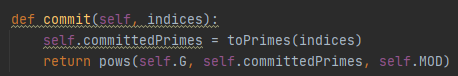
数据用其乘积表示并作指数运算



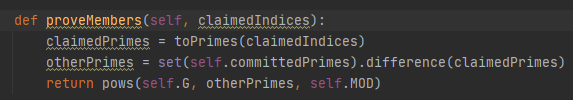
扩展的欧几里算法



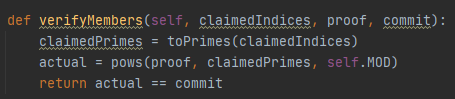
数值承诺



成员身份证明



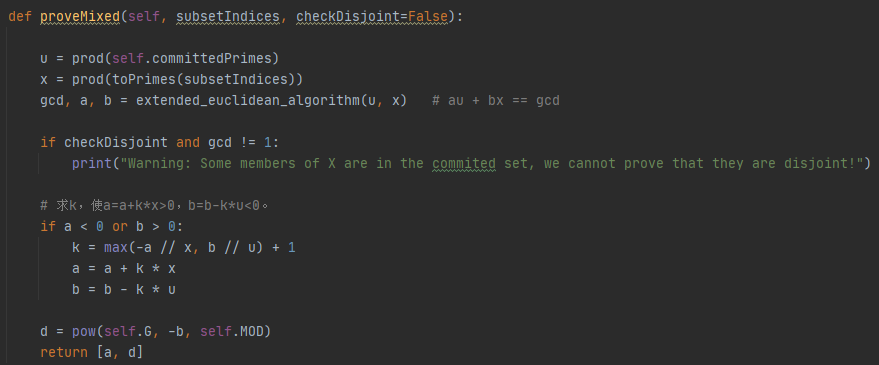
成员身份验证证明



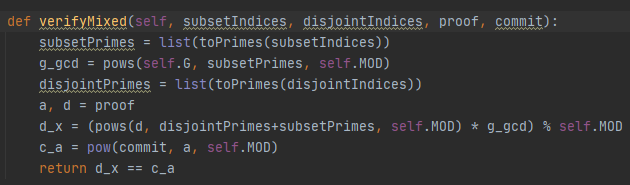
非成员身份证明

证明非成员 checkDisjoint=True

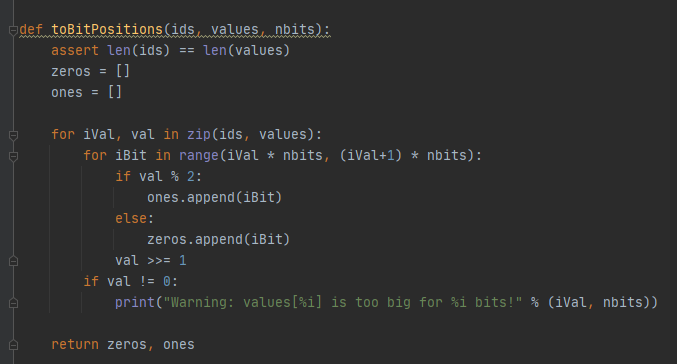
证明既有成员又有非成员checkDisjoint=False



非成员身份验证证明



数据编码函数并返回0 1列表

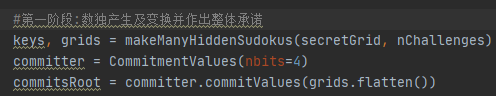


## 1.7数独游戏

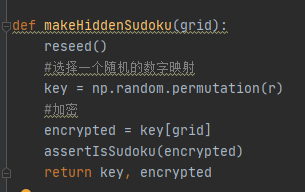
参数设置

T@7F5[OW45N8)HL5]3SLQR4

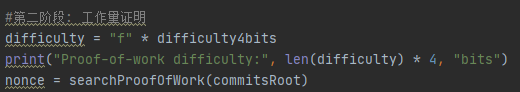
产生数独并变换并作出整体承诺



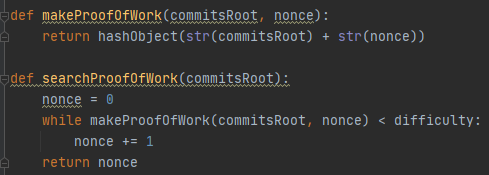
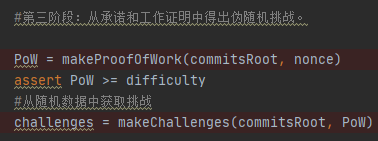
数独变换函数



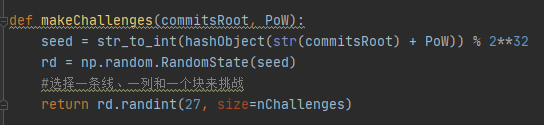
工作量证明



工作量证明函数

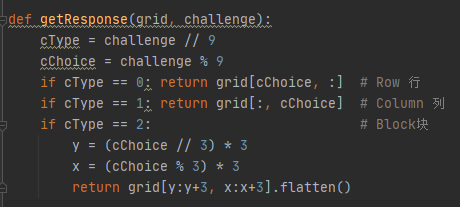
  
获得伪随机挑战  


获得挑战函数



根据挑战 获得响应 做出承诺  


响应函数



模拟发送接收  


验证  
