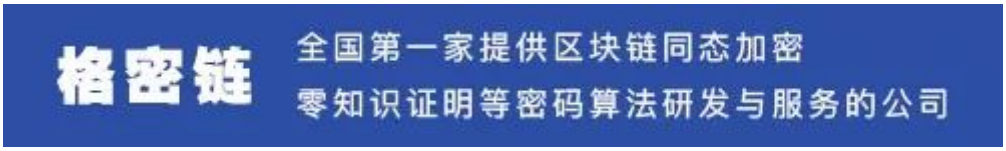
全同态加密算法深入解析-1

原创 致远博士 格密链 2018-11-29 10:00



同态加密方案提供了一种惊人的能力——能够在不解密的情况下，对密文数据进行计算。这使得您无需破坏敏感源数据，同时可以对数据进行处理。我们将分两次内容对全同态加密算法做深入地解析。

目前在同态加密方案中，最有影响力的一个方案(也是最近一些标准化工作的主题)被称为Fan-Vercauteren (FV)方案(或称为Brakerski-Fan-Vercauteren方案)，我们将在这里深入地进行说明，同时你也可以尝试使用以下实现该方案的算法库：

Library	URL	License	Language
HEANN	https://github.com/kimandrik/HEAAN	CC non commercial	C++
NFLib	https://github.com/quarkslab/NFLlib	GPLv3	C++
FV-NFLib	https://github.com/quarkslab/NFLlib	GPLv3	C++
cuHE	https://github.com/vernamlab/cuHE	MIT	CUDA C++
PALISADE	https://git.njit.edu/palisade/PALISADE/tree/master	Liberal	C++
SEAL	https://www.microsoft.com/en-us/research/project/simple-encrypted-arithmetic-library/	Microsoft research only	C++
HELib	https://github.com/shaih/HElib	Apache 2.0	C++
jLBC	http://gas.dia.unisa.it/projects/jlbc/download.html	LGPLv3	Java

这些加密方案看起来很复杂，也有一点神秘，但希望本文能让您清楚地了解它们的工作原理和驱动因素。

本文的整体结构包括（今天内容为前两部分）

- 一点点数学介绍
- 加密和解密是如何工作的
- 同态加法和乘法

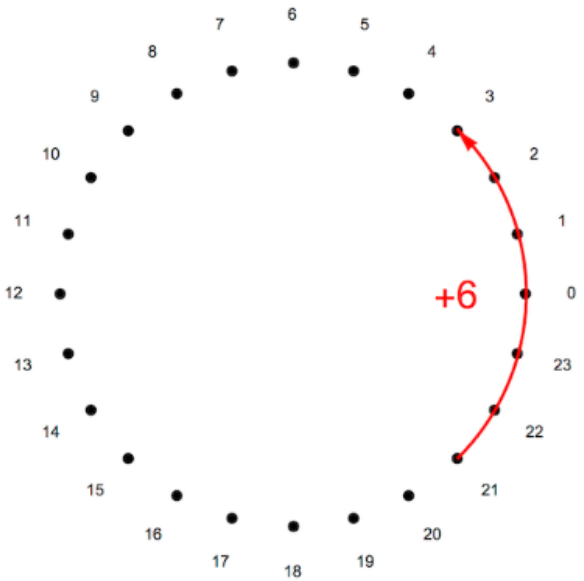
1
数学介绍

这些同态加密方案是基于Ring Learning with Errors问题。本质上，这些方案中的数据在加密时(密文)和未加密时(明文)都以多项式表示。

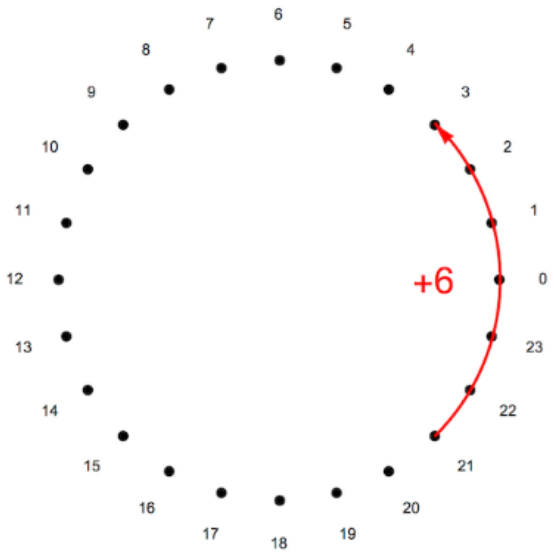
这些几乎是学校里每个人都学过的多项式。像

$$4x^2 + 2x + 1$$

但有一些区别，第一个是系数都是整数，并且需要mod t。假设t = 24，这就像一个24小时的时钟，21加6得到3。多项式的所有系数都是这样处理的。



或者，我们可以将数字考虑在-11到12之间，这样我们就可以方便地求负数。注意，这只是一个方便系数——余数为-1和余数为23(除以24时)之间没有区别。



第二点，也是比较棘手的一点，在于这种使用余数的思想不仅适用于多项式的系数，也适用于多项式本身。

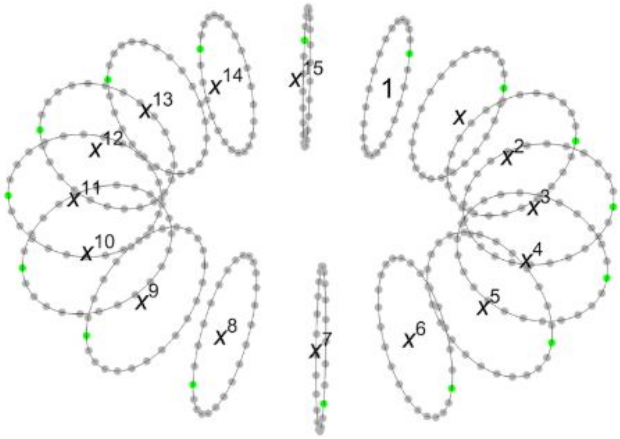
我们定义了一个特殊的多项式，称为多项式模，并且只考虑多项式乘以该多项式模后的余数。FV方案中该多项式模的具体形式为$x^d + 1$，其中对于某些n，有$d = 2^n$。为了说明这一点，我们取$n = 4$，因此多项式为$x^{16} + 1$。

因为我们考虑的是关于模$x^{16} + 1$之后的余数，所以我们只需要考虑幂从x^0到x^{15}的多项式。任何更高次的幂都会因乘以该多项式模而消去。这也可以被理解为，$x^{16} \equiv -1 (\mathrm{mod}\ x^{16}+1)$，这意味着$x^{16}$可以被-1替换，以将$x$的更高次幂归约到0到15的范围内。

所以我们考虑的多项式都是这种形式的

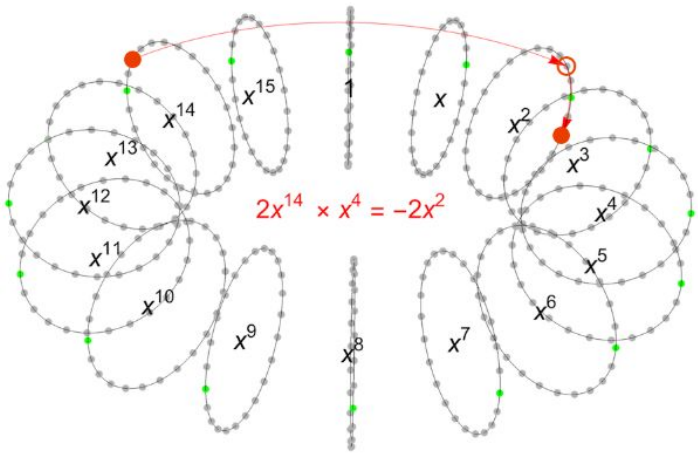
$$a_{15}x^{15} + a_{14}x^{14} + a_{13}x^{13} + a_{12}x^{12} + a_{11}x^{11} + a_{10}x^{10} + a_9x^9 + a_8x^8$$
$$+a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$

其中这16系数(即a_i)中的每一个的范围都是从0到$t - 1$。我们可以用系数的环面来说明，如下所示：



在这个图中，每个循环表示多项式中x的幂次方前的系数的取值范围（包含24个可能值）。绿点代表系数取0时所处的位置。这为我们提供了一种很好的方法来可视化多项式，这在我们考虑加密和解密步骤如何工作时将会有所帮助。

FV加密方案涉及大量的多项式乘法。当我们把x的两个幂次方相乘，比如$2x^{14}$和x^4时，我们把它们的指数相加，得到$2x^{18}$。有人可能会假设，求这个多项式关于多项式模的余数可能需要在x^{16}处将指数旋转回0，得到$2x^2$，就像上面所示的整数系数那样。如果多项式模是x^{16}，情况就是这样。然而，我们的多项式模是$x^{16} + 1$ - 如上所述，额外的+1因子引入了一个符号变化，这有助于进一步干扰乘法的结果。



如上图所示，当$2x^{14}$乘以x^4后模$x^{16} + 1$时，取这个项(由上面的红点表示)，向前旋转环面4个幂，然后从0处调整系数的值，得到$22x^2$(或$-2x^2$，如果我们认为数字是从-12到11而不是从0到23时)。

这种形式的多项式具有非常丰富的结构和许多不错的特性。它们是分圆多项式的子集。使用其中一个作为多项式模并不是严格必需的，但是这样做会更加方便快速。

— 2 —
使用环上的多项式加密

我们已经介绍了FV加密方案中使用的环上的多项式的一些属性，现在我们可以讨论加密和解密的工作原理。首先，我们需要讨论如何生成私钥和公钥，然后讨论如何使用它们进行加密和解密。

私钥和公钥

加密采用明文，并使用从私钥派生的公钥将其转换为密文。从明文到密文的转换是通过一种只有在您知道私钥的情况下才容易可逆的方式完成的。

更具体地说，明文是环上的多项式，其具有多项式模$x^d + 1$，其中$d = 2^n$，以及系数模t。明文加密后为密文，其是由两个环上的、具有相同多项式模的多项式构成的，但系数模为q，通常q远大于t。

例如，多项式模为$x^{4096} + 1$，这意味着明文和密文中的多项式都有$d = 4096$个系数。明文多项式的系数需要模$t = 290764801$，密文多项式的系数需要模$q = 92143472475614740 48$或更大。

为了便于说明，我们将使用较小的数字，但希望这些数字能够更好地说明方案的各个步骤中发生了什么。在第一部分中，为了更直观，我们将使用$d = 16$、$t = 7$和$q = 874$。注意，这些参数是不安全的!!

对于私钥或密钥，我们用s表示，它是我们随机生成的一个系数为-1、0或1的多项式。例如，

$$s = x^{15} - x^{13} - x^{12} - x^{11} - x^9 + x^8 + x^6 - x^4 + x^2 + x - 1$$

接下来，我们从密文空间中随机生成一个多项式(用于生成公钥)，其系数模为q，我们用a表示。

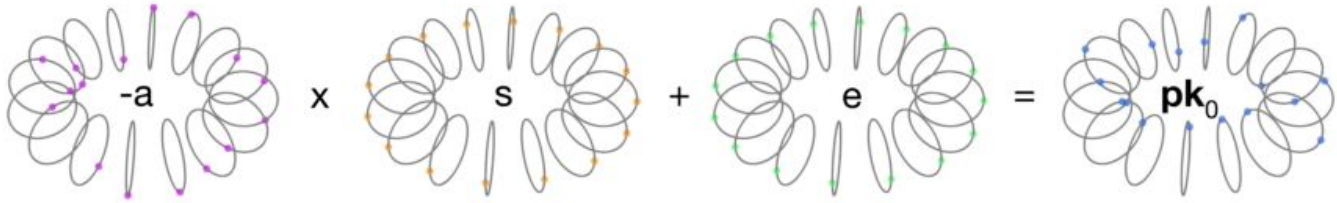
$$a = 42x^{15} - 256x^{14} - 393x^{13} - 229x^{12} + 447x^{11} - 369x^{10} - 212x^9 + 107x^8$$
$$+52x^7 + 70x^6 - 138x^5 + 322x^4 + 186x^3 - 282x^2 - 60x + 84$$

我们还定义了一个噪音多项式，它是“小”的，因为它是从离散高斯分布中取的一个小系数。这个多项式只在这里使用一次，然后丢弃。

$$e = -3x^{15} + x^{14} + x^{13} + 7x^{12} - 6x^{11} - 6x^{10} + x^9 + 4x^8$$
$$-x^6 + 3x^5 - 4x^4 + 4x^3 + 4x + 1$$

然后将公钥定义为一对多项式，即**pk** = ($[-as + e]_q, a$)，其中多项式都是模多项式模和系数模q的。

对于上面给出的示例，公钥的第一个多项式被构造为



$$\mathbf{pk}_0 = -285x^{15} - 431x^{14} - 32x^{13} + 86x^{12} - 83x^{11} - 142x^{10} - 41x^9$$
$$+430x^8 + 26x^7 - 158x^6 - 281x^5 + 377x^4 + 110x^3 - 234x^2 - 113x + 252$$

其中第一个乘法取多项式a，它的系数mod q，然后乘以系数为-1 0或1的s。由于模上多项式模的多项式的乘法具有“旋转和反射”性质，有效地混合和打乱了a的所有系数，并进一步增加了小的噪音。多项式a有效地掩盖了公钥中的私钥。

通过从公钥中找到s的方式来破解加密方案，其主要涉及的计算为($[-as + e]_q, a$)。唯一的因素是该方案中包含了噪音——如果e为零，则很容易从公钥中计算出s。当e足够大，但又不太大时，这是一个难题。

本文的示例中，私钥可以通过暴力攻击恢复——–尝试每个可能的s(只有3 ^16 = 43046721组合)，然后计算$as + e$来寻找出一个接近公钥的第一项的答案。对于真正的参数，这种暴力攻击的方法是完全不可行的。3 ^ 4096是一个很大的数字，但有更聪明的方法，然后定义给定的一组参数的安全性。

加密

加密过程看起来有点像公钥生成过程。

加密明文的过程是将一个系数模为t的多项式转换为一对系数模为q的多项式。本例中，我们将加密一个非常简单的多项式(称为消息) - $m = 3 + 4x^8 \equiv 3-3x^8$ – 只有两个不为零的系

数。

加密还需要三个小的多项式。两个噪音多项式来自于相同的离散高斯分布(即和公钥中的噪音多项式的取法一样)，另一个多项式我们称之为*u*，它的系数为-1、0或1，就像私钥一样。

$$e_1 = -5x^{15} - 2x^{14} + 3x^{13} - x^{12} - 4x^{11} + 3x^{10} + x^9 + 4x^8$$
$$+4x^7 + 5x^6 - 4x^5 - 3x^4 - 3x^3 + 2x^2 - 6x + 4$$
$$e_2 = -7x^{15} + 2x^{14} - 4x^{13} + 5x^{11} + 2x^{10} - x^9 + 4x^8$$
$$-4x^7 - 3x^6 + 2x^5 - 2x^4 + x^3 - 4x^2 - 2x + 2$$

和

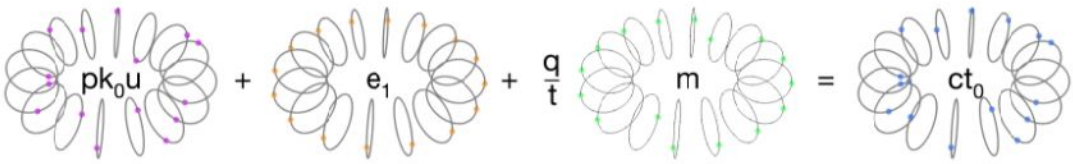
$$u = x^{14} + x^{13} + x^{12} - x^8 - x^5 - x^3 + 1$$

这些多项式只在加密过程中使用，然后丢弃。

密文是由两个多项式组成的，通过如下计算得到

$$\mathbf{ct} = ([\mathbf{pk}_0 u + e_1 + qm/t]_q, [\mathbf{pk}_1 u + e_2]_q)$$

请注意消息中的值是在mod *t*的范围内，而在我们的示例中，它们被缩放为*q*/*t*(即128)，使它们覆盖mod *q*的范围。这是消息被插入到密文时的唯一更改。这些值通过添加到第一项来掩盖，第一项的值是在mod *q*的范围内，与随机的噪音没有区别。*u*的随机性改变了每次加密中使用的掩码，从而确保相同的明文在每次加密时产生不同的密文。



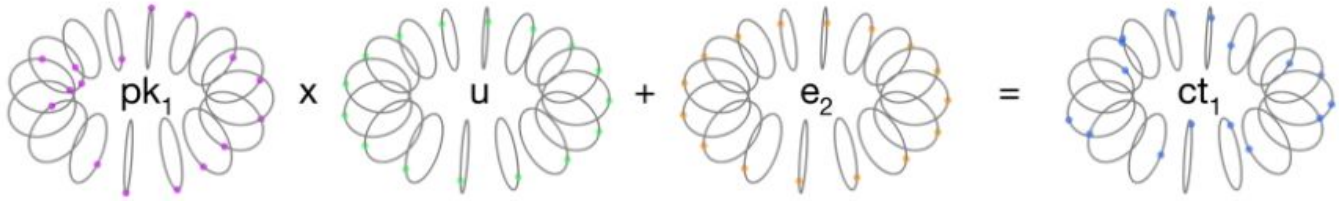
同态加法和乘法之所以有效，是因为消息在密文中以比例来表示。其他项用于掩盖消息，而且可以证明它们是有效的，只有在您知道私钥的情况下才能删除它们。

使用上面给出的多项式显式地计算密文的第一个元素

$$\mathbf{ct}_0 = 217x^{15} - 53x^{14} + 13x^{13} - 249x^{12} - 392x^{11} - 238x^{10} + 252x^9 + 115x^8$$
$$+5x^7 + 184x^6 - 201x^5 - 258x^4 - 247x^3 + 144x^2 + 23x + 42$$

代入公钥，我们可以看到密文的第一个元素展开为$\mathbf{ct_0} = [e_1 + eu - aus + qm / t]_q$。在这个表达式中，前两项是“小”的，与噪音成比例，后两项是“大”的。第一个大项有效地掩盖了第二个大项，即消息。

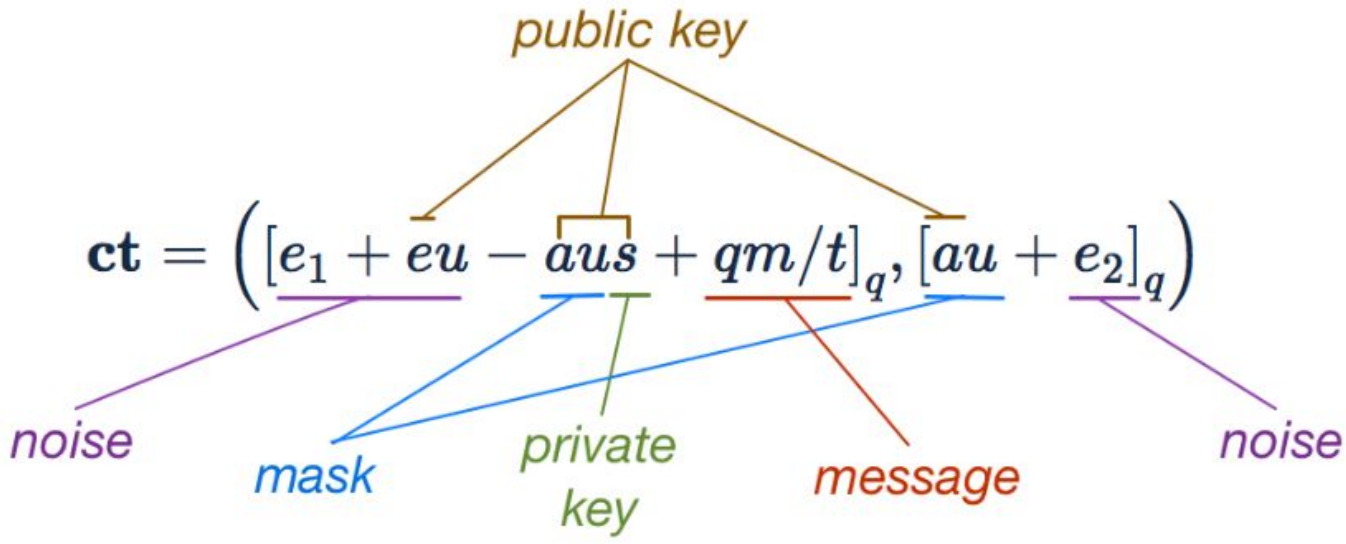
密文的第二个元素是这样计算的:



$$\mathbf{ct}_1 = 25x^{15} + 225x^{14} - 12x^{13} + 270x^{12} + 350x^{11} - 24x^{10} + 56x^9 - 330x^8$$
$$+386x^7 + 225x^6 - 332x^5 + 68x^4 - 20x^3 - 26x^2 - 91x + 380$$

代入公钥，我们看到密文的第二个元素展开为$\mathbf{ct_1} = [au + e_2]_q$。这说明了解密是如何工作的——如果我们知道$s$，就可以计算出$\mathbf{ct_1}s = [aus + e_2s]_q$，它可以用来消除密文的第一个元素中的非消息大项。

综上所述，密文可以用公钥(public key)、私钥(private key)、掩码(mask)、噪音(noise)和消息(message)表示为



解密

如上所述，解密相对简单。首先，我们计算$[\mathbf{ct_0} + \mathbf{ct_1}s]_q$，它将从消息中完全移除掩码。这给我们一个多项式，它可以展开为$[qm / t + e_1 + eu + e_2s]_q$-也就是说，缩放后的信息加上一些噪声。因此，只要噪声不太大，我们就可以恢复消息。

明确地，

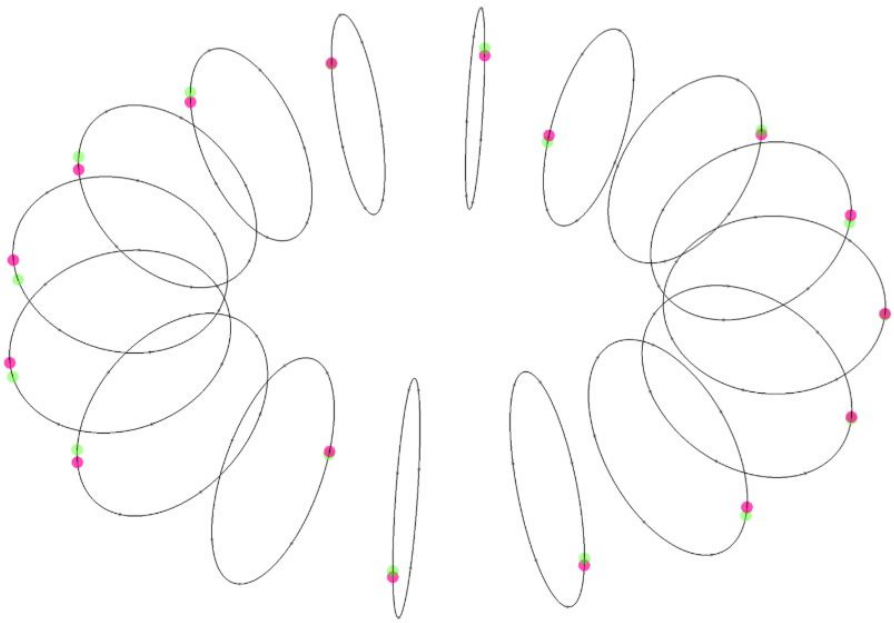
$$\mathbf{ct}_1 s + \mathbf{ct}_0 = 13x^{15} - 2x^{14} + 17x^{13} + 22x^{12} - 32x^{11} - 23x^{10} + 19x^9 - 380x^8$$
$$+9x^7 + 10x^6 - 13x^5 - 3x^4 - 2x^3 - 12x^2 + 7x + 393$$

在这里您可以看到，除了明文的两个非零系数(x^8和x^0)之外，所有的系数都小于$q/t = 128$。如果我们把这个多项式缩放回mod t范围内的值，那么我们就得到

$$\frac{13x^{15}}{128} - \frac{x^{14}}{64} + \frac{17x^{13}}{128} + \frac{11x^{12}}{64} - \frac{x^{11}}{4} - \frac{23x^{10}}{128} + \frac{19x^9}{128} - \frac{95x^8}{32}$$
$$+\frac{9x^7}{128} + \frac{5x^6}{64} - \frac{13x^5}{128} - \frac{3x^4}{128} - \frac{x^3}{64} - \frac{3x^2}{32} + \frac{7x}{128} + \frac{393}{128}$$

四舍五入这些系数可以恢复我们的消息$m = 3 - 3x^8$。

我们通过将系数四舍五入，来舍入到最接近的整数后得到我们的信息:



绿色 – 噪音变换　粉色 – 舍入结果

把它们放在一起，我们通过如下计算来解密密文

$$m' = \left\lfloor \left\lfloor \frac{t}{q} [\mathbf{ct}_0 + \mathbf{ct}_1 s]_q \right\rceil \right\rceil_t$$

⌊⌉表示舍入到最接近的整数(四舍五入)。

如果系数中噪音太大，那么它们最终会更接近一个与正确整数不同的整数，然后解密会(悄无声息地)失败并产生错误的结果。在上面的示例中，最大的噪音为13 / 128，所以仍然有一些空间允许产生更多的噪音，并且能够正确解密。噪音的含量可以通过将*q* / *t*的比值变大或变小来调节。

你可能还会喜欢：

黎曼猜想是否会对密码学的安全产生影响
比特币必须灭亡
50+区块链如何引领世界的例子
Token化如何将传统资产搬上区块链
Token的价值
区块链是糟糕的技术
谁将赢得区块链比赛 ——中国正在赢得500年来最重要比赛
全同态加密：从理论到实践-1
给六岁小孩讲区块链
我为什么受够了Chrome
传统资产正迈入Token化时代
解析比特币白皮书之交易
Coin和Token间的区别到底是什么?
区块链实力哪国强

欢迎收听“区块链杂谈”节目，国内最有质量的区块链知识分享节目。

宁波格密链网络科技有限公司，专注于区块链上的密码技术研发。

格密链

专注于区块链上的密码学技术

长按扫码可关注