本项目经过研究认为，选择从循环矩阵中寻找轻量级MDS矩阵是一种合适的做法，因为其有如下几个优势：

* 与随机方阵相比，循环矩阵更有可能找到 MDS 矩阵。
* 设k为方阵的阶。它可以在最多只能有k个不同条目的情况下维持MDS的性质，并且可以包含重复的轻量级条目。这使得循环矩阵往往比像 Hadmad矩阵和Cauchy矩阵等必须至少有 k 个不同条目的矩阵实现成本低。
* 它可以灵活地支持基于轮次和序列化的实现。

**轻量级矩阵的衡量与筛选**

衡量矩阵是否对乘法计算高效的指标是矩阵的异或运算总个数，这是一个易于理解和计算的效率指标，实践中MDS矩阵经常选择降低异或和来简化矩阵乘法计算，从而降低硬件面积，提高计算效率。值得注意的是，这个指标容易和 Hamming 重量相混淆。虽然低 Hamming 重量在实现中通常需要更少的硬件资源。例如，AES的 MDS矩阵的系数被设置成1、2、3。然而，一些研究表明，虽然这个启发式方法通常是正确的，但并不总是适用。由于一些减少效应，并且根据定义计算有限域的不可约多项式，一些不那么低 Hamming 重量的矩阵也可能用只有非常少的异或运算总个数。

本项目找到了使用公式计算循环矩阵的异或运算总个数的方法:

首先，定义单个元素的异或运算个数：对某个有限域GF()上的元素a，a的异或运算个数就是实现a和任意GF()上的元素b相乘所需要的异或运算的数量。

然后，我们给出矩阵M中某一行异或运算总个数N的计算公式:

其中yi是M中那一行的第i个元素的异或运算个数，n是那一行中非零元素的个数，而r是有限域的维度。根据哈希算法的设计，这个有限域是GF()，所以r = 8

由于本项目中计算的矩阵都是循环矩阵，显然M的异或运算总个数为4N。

接下来需要计算yi，也就是单个元素的异或运算个数。这里存在一个问题：有限域上的某个元素对于模不同的不可约多项式乘法，具有不同的异或运算个数。有趣的是，可以证明：虽然有限域上每个元素模不同的多项式时具有不同的异或运算个数，但一个有限域上所有元素的异或运算个数和是一个定值，与不可约多项式无关。计算某个有限域GF()下所有元素异或运算个数和的公式如下：

虽然如此，这个结论并不代表不同的不可约多项式对于矩阵乘法的计算效率没有影响。我们在研究过程中发现：因为最后被选取的轻量级矩阵具有的异或运算总个数较少，它们每一个元素具有的异或运算个数也较少。所以，在选取不可约多项式的时候，比起使得所有元素中异或运算个数分布较为均匀的不可约多项式，那些使得异或运算个数分布较不均匀的不可约多项式是更优的选择，因为它们的异或运算个数更多的分布于一些特定的元素上，而总的来说，有限域的阶数要远大于矩阵的阶数，只有一小部分元素会出现在矩阵当中，则那些异或运算个数较大的元素在矩阵中更不可能出现。这样在所有元素总的异或运算个数相同的情况下，标准差较大的不可约多项式下更有可能找到更轻量级的矩阵。这项发现使得我们能够在搜索时优先使用最好的不可约多项式，节约了搜索时间。为此，我们计算了GF()上所有不可约多项式下每个元素异或运算个数的标准差并制成了以下三张表格：

**GF() 上各不可约多项式下每个元素异或运算个数的标准差（1）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| x | 0x165 | 0x18B | 0x163 | 0x11B | 0x13F | 0x15F |
| σ | 6.868 | 7.362 | 6.415 | 7.530 | 5.577 | 5.058 |

**GF() 上各不可约多项式下每个元素异或运算个数的标准差（2）**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| x | 0x1C3 | 0x139 | 0x11D | 0x177 | 0x1F3 | 0x169 |
| σ | 7.463 | 6.758 | 7.396 | 5.300 | 5.754 | 6.757 |

**GF() 上各不可约多项式下每个元素异或运算个数的标准差（3）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 0x1BD | 0x1E7 | 0x12B | 0x1D7 |
| σ | 5.253 | 5.884 | 6.175 | 5.798 |

随后，我们计算了GF()上各个不可约多项式对0-255每个元素进行乘法时所需要的异或运算个数并制成异或运算个数表。通过这个异或运算总个数表，就能够计算得到给定矩阵的异或运算总个数，从而筛选出具有较小异或运算总个数的矩阵，也就是在GF()下进行矩阵乘法的计算开销更少的矩阵。

**MDS矩阵构造**

本项目选择从循环矩阵中搜索MDS矩阵，在选择最轻量级的循环矩阵并检查其MDS属性的方法中，主要存在两个挑战：

首先，对于一个阶数为k的通用循环矩阵（不考虑条目的值），circ(C0,C1.….,Cx-1)，有k!种排列条目的方式。显然，随着k的增大，排列条目的方式迅速增加，很快就变得难以处理。

其次，选择k个轻量级非零条目的方式并不需要是独特的，这可能导致拽索空间远大于仅选择k个不同条目并排列它们。

本项目引入了一个等价关系，定义为：当存在两个置换矩阵P和Q使得M′ = PMQ时，我们称两个矩阵M和M′是置换等价的，记作M ∼B M′。循环矩阵的等价类是满足等价关系 ∼B 的一组循环矩阵。

这样，可以用∼B来将k！个循环矩阵划分为等价类。其中同一等价类内的循环矩阵共享相同的分支数。这使我们能够通过检查每个等价类中的一个代表来减少搜索空间。本项目经过研究，证明对于k阶循环矩阵，一共有个循环矩阵等价类。其中大部分等价类都不含有MDS矩阵。我们分析了循环结构，并证明了对于阶数k ≤8，最多有5种类型的MDS循环矩阵，根据其第一行不同条目的个数，可以分成：

* 类型1：没有重复条目。
* 类型2：1对重复条目。
* 类型3：2对重复条目。
* 类型4：3对重复条目。
* 类型5：3个重复条目。

这样，可以通过枚举重复条目的个数和位置来生成循环矩阵，进而验证它们是否具有MDS性质。下表列举了从3-8阶下循环MDS矩阵可能属于的类型：

**3-8阶下循环MDS矩阵的可能类型**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 阶数 | 没有重复条目 | 1对重复条目 | 2对重复条目 | 3对重复条目 | 3个重复条目 |
| 3 | ✓ | ✓ |  |  |  |
| 4 | ✓ | ✓ |  |  |  |
| 5 | ✓ | ✓ | ✓ |  |  |
| 6 | ✓ | ✓ | ✓ |  |  |
| 7 | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8 | ✓ | ✓ | ✓ | ✓ | ✓ |

然后，就可以对上节选出的轻量级矩阵进行快速筛选以选出具有MDS性质的轻量级矩阵。本项目通过这种方法，在 上选择了最佳的模多项式0x11B，搜索并筛选得到了10个轻量级的MDS矩阵用于后续杂凑函数的设计当中。

**对AES S盒的改进**

本项目对于目前在密码学理论与应用上占据重要地位的AES算法的S盒进行了分析，发现其存在一个问题：尽管其S盒的代数表达式具有较高的次数，但其项数过少。具体来说，AES的S盒的代数表达式如下：

这个表达式次数为254，项数为9。较低的项数会带来较低的表达式复杂度，从而影响S盒的安全性。虽然迄今为止AES本身依然被认为是安全和可靠的，但为了更长期的安全性考虑，有必要寻找更加安全的S盒。

为了获得更好的 S 盒，我们进行了大量的实验和模拟。以下是从模拟中得出的结果：AES S 盒简单的代数表达式与取乘法逆和应用仿射变换的变换顺序相关：仿射变换周期与所选的仿射变换相关，迭代周期也与所选的仿射变换相关。因此，可以通过修改S盒的仿射变换并添加一个仿射变换。改进 S 盒的密码学特性，但是，我们不能通过仅应用一次仿射变换，实现 S 盒和相应的逆 S 盒的代数表达式包含更多项的目的。针对上述问题，我们提出了一种改进的 AES S 盒方案。在这一改进方案中，我们选择了新的仿射变换以代替原来的AES的仿射变换。改进后的 AES S 盒通过以下四个步骤构建：

首先我们将S盒中的仿射记为，简记作。这个仿射变换的定义如下：，其中，，。例如，AES S盒的仿射变换就可以记作，其中，，令，，，这个仿射变换就可以记为：

然后，我们使用仿射变换,的定义如下：

+

这样之后，我们取乘法逆：

最后，我们再次应用仿射变换就可以得到改进后的S盒：

对于每一个仿射变换，都存在一个逆向的仿射变换 使得。对于上文使用的仿射变换，对应的逆向仿射变换是，具有以下性质：

这个逆向的仿射变换也可以通过类似上文的步骤处理，从而构建出一个改进的AES 逆向S盒。

经过改进之后的S盒消除了之前提到的AES的S盒的缺点。我们经过分析，得到了改进后S盒代数表达式。由于表达式过长，现将每一项的系数列表如下：

**改进后S盒代数表达式的系数（16进制下）**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | FA | A6 | A9 | E5 | DE | 5A | 05 | FB | 5C | AA | 64 | CB | A1 | 87 | 6A | 6C |
| 1 | E4 | 87 | 93 | 6A | 76 | E5 | 66 | 09 | 43 | 0C | 91 | 92 | 03 | 8F | 63 | 30 |
| 2 | 54 | 99 | E9 | 30 | EE | BF | F2 | E6 | 71 | 4E | 90 | D5 | 18 | 85 | 45 | FA |
| 3 | 7E | 73 | 0E | 13 | 8B | 5B | 08 | 8B | C8 | 3B | 6A | 10 | 87 | 09 | FB | 47 |
| 4 | 28 | AF | C5 | 20 | 0B | 8D | 74 | D5 | 59 | 37 | 19 | C9 | 2A | 4F | 02 | C1 |
| 5 | 91 | F1 | 50 | 83 | 9B | 42 | 87 | 4A | 42 | F2 | 74 | 0C | 4F | 2D | 49 | AE |
| 6 | DA | 25 | 64 | 58 | CD | FE | 1B | D2 | 7D | F8 | 66 | A8 | 6D | 2A | A9 | 7E |
| 7 | 21 | C3 | 3F | D2 | EC | C9 | 7A | AC | 0D | CC | 7F | D3 | 35 | 25 | C2 | 6F |
| 8 | BF | 86 | 07 | 91 | 5E | C5 | 75 | 4E | C1 | 83 | E8 | CA | B0 | E9 | 75 | B6 |
| 9 | 07 | 16 | F8 | 7B | 49 | D5 | FA | AD | 5B | 5F | C2 | 19 | 12 | 13 | C5 | 20 |
| A | 99 | B3 | 44 | 9F | F2 | 71 | 9B | 38 | 7A | AE | A5 | 0D | 48 | C4 | EF | 1E |
| B | F4 | 09 | 8F | D1 | 2F | 88 | 60 | 9B | E9 | 9F | 75 | 66 | 69 | 7C | 23 | 62 |
| C | 05 | DE | 30 | DE | BB | D5 | 96 | 4D | 52 | AB | 77 | 32 | 09 | B4 | 3F | AB |
| D | FF | 97 | D6 | FB | FE | 59 | DB | BA | 15 | 1A | 64 | 2D | 02 | F8 | 5D | 3B |
| E | 74 | EE | 1B | 82 | C8 | 63 | F8 | F4 | BF | 49 | 50 | 5A | 7C | FC | 46 | 47 |
| F | C9 | 97 | ED | 52 | 59 | D3 | 01 | 56 | 79 | DB | C7 | 9A | E7 | 26 | 12 | 00 |

类似地，改进后的逆向S盒的代数表达式项数也从9提升到了253。

许多代数攻击依赖于表达式的稀疏性及其特定结构。由于改进后的S 盒代数表达式涉及近 255 项，有效避免了表达式稀疏与过度定义的可能性。将改进后的 AES S 盒应用于项目的杂凑函数中，其抵抗代数攻击的能力将大大增强。