2 基于 CUDA 并行架构的代数次数求解

最直接的确定布尔函数代数次数的方法是计算布尔函数的代数正规型,进而得到布尔函数的代数次数.如果已知布尔函数的真值表,可以基于莫比乌斯变换计算布尔函数的代数正规型.

对于一个 n 变量的布尔函数 $f(x)$,$f(x)$的真值表存储在长度为 2^n 的数组 v 中,则可以利用算法 1[18]求解布尔函数的代数正规型.算法 1 使用了两个长度为 2^{n-1} 的辅助数组 t 和 u,||表示连接运算.

算法 1. 利用真值表计算代数正规型.

输入:真值表数组 v.

输出:f 的代数正规型.

for i=1,2,…,n

  for j=1,2,…,2^{n-1}

    $t_j=v_{2j-1}$

    $u_j=v_{2j-1}\oplus v_{2j}$

  end for

  $v=t||u$

end for

遍历算法 1 得到代数正规型的系数数组,即可知道布尔函数的代数次数.算法 1 的时间复杂度 $O(n2^n)$,空间复杂度 $O(2^n)$,运行时间和所需空间随着分组长度的增加呈指数级增长.在计算资源有限的情况下,求解得到的代数次数有限.为了提高算法的运行效率,充分利用计算资源实现算法的并行处理,我们构建了基于 CUDA 并行架构的代数次数的求解模型,协同利用 GPU 和 CPU 同时进行数据计算,极大地降低了代数次数的求解时间.

2.1 GPU与CUDA架构

GPU(graphic processing unit)最早的设计是用来对大量的图形图像数据进行并行处理,以减轻 CPU 的工作负担.随着 GPU 设计理念的不断完善与制作工艺的飞速发展,GPU 的内存带宽与浮点运算能力已远远超过了同时代的 CPU 性能.用 GPU 做通用计算已经成为研究的热点,相比于 CPU 的更适合对复杂的运算指令、多路的分支与判断的逻辑控制程序的处理,GPU 则更适合处理具有大数据量、高并行性、低耦合性、高计算密度的数据计算程序.

在 GPU 基础上发展起来的 CUDA(compute unified device architecture)[19]并行计算架构,在高性能计算方面得到广泛应用.NVIDIA 公司为 CUDA 应用程序开发人员提供了一整套程序开发与调试环境组件,包括 NVCC 编译器、GDB 调试器、CUDA 库函数、CUDA 运行时库(CUDA runtime API)以及 CUDA 驱动程序(CUDA driver API).在上述的 CUDA 编程的软件体系中,其核心部分是 CUDA C 语言,它实质上是标准 C 语言的一个极小扩展集和一个运行时库,源文件可以通过 NVCC 编译器编译得到调用这些扩展和运行时库的目标程序;使用 NVCC 编译器得到的目标程序只是在 GPU 设备上运行的代码,而要管理 GPU 的计算资源,就需要借助 CUDA 运行时库和 CUDA 驱动程序来实现.与传统的 GPU 相比,CUDA 编程模型[20]中属于同一个线程块内的线程之间不仅能够相对独立的执行并行程序,而且还可以通过共享存储器和线程同步技术实现同一个线程块内不同的线程间相互通信.考虑到 CUDA 并行架构的高效性和便捷性,我们构建了基于 CUDA 架构的代数次数的求解模型.

2.2 基于CUDA架构代数次数的求解

利用 CUDA 实现高性能计算,实质上是通过 CPU 与 GPU 的分工合作、并行运行来完成的.CUDA 编程模型可以分为 Host 端(主机端)和 Device 端(设备端):Host 端为 CPU 部分,主要在计算机内存中执行,负责处理逻辑性较强的任务和执行串行部分的计算;Device 端为 GPU 部分,主要在计算机显卡内存中执行,负责处理高度线程化的并行任务,又称为核函数(kernel).CUDA 程序是由许多 Device 端内核函数并行执行步骤和许多 Host 端串行执行步骤共同完成的,从而提高了程序的整体运行性能.

对于分组密码算法,轮函数可以表示为关于该轮输入的布尔函数.利用真值表求解布尔函数的代数次数时,

首先需要求解密码算法的真值表,其次利用真值表求解布尔函数的代数正规型,进而得到对应轮数的代数次数.对于分组长度为 n 的算法,一方面,求解真值表需要遍历 2^n 个全部的输入状态,可以并行化执行;另一方面,真值表的求解过程不需要做复杂的运算和逻辑判断,因而将求解真值表的部分指定为设备端程序_device_,由 GPU 调用运行.然而,对于利用快速莫比乌斯变换求解代数正规型的过程,由于涉及多路的分支与复杂的逻辑判断,不适用于 GPU 计算,因而将实现快速莫比乌斯变换的过程指定为主机端程序_host_,由 CPU 运行.此外,为了缩短 CPU 的运行时间,我们利用 Antoine Joux[21]提出的求解代数正规型的并行优化算法,采用 CPU 多核并行技术,做到多个比特同时运算,从而实现了算法加速.

整个 CUDA 程序的实现模型是串行和并行任务的交互完成.当有并行任务时,Host 端调用 kernel 函数,将执行算法真值表求解的任务交给 Device 端解决.当 kernel 函数映射到 GPU 上后,分配到网格(grid)上,网格中的线程又被细分为一维的线程块(block),每个线程块分解为多个线性(tread),在同一个多处理器上运行,提高了数据处理的效率,极大地降低了密码算法真值表的生成时间.基于 CUDA 架构实现密码算法代数次数求解的模型如图 2 所示.

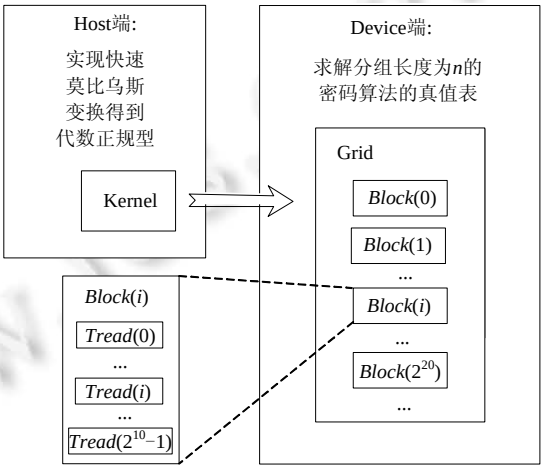

Fig.2 Model of algebraic degree estimate based on CUDA

图 2 基于 CUDA 代数次数的求解模型

2.3 代数次数的求解结果

基于 CUDA 架构计算密码算法的代数次数虽然可以极大地缩短运行时间,提高求解效率,但是当计算和存储资源有限时,仍然无法准确求解分组长度较大的密码算法的代数次数.作为应用,我们在个人 PC 机上(Intel(R) Core(TM) i7-7700HQ CPU@2.8GHz 16.00GB,NVIDIA GeForce GTX1060)配置了 VS 2012 和 CUDA 9.1.85 的环境,编写了相关的程序,在较短的时间内求解了轻量级分组密码算法 SIMON32 和 SIMECK32 全轮布尔函数的代数次数,结果分别见表 1 和表 2.

我们从算法第 1 轮开始算起,多次运行程序求解平均值,给出了计算布尔函数代数正规型和代数次数的时间.由表 1 可知,对于 SIMON32 来说,轮函数的代数次数增长比较缓慢,并没有达到算法迭代所期望的代数次数,从第 13 轮开始,SIMON32 代数次数达到上界 31.我们利用 CUDA 架构可以在不到两分钟内求解任意轮 SIMON 32 算法的代数正规型,且代数正规型的求解时间随轮数的增长没有特别明显的增大.此外,计算代数次数由于需要遍历代数正规型的单项得到最大值,因而求解代数次数的时间比代数正规型大,且随代数正规型的复杂程度而变化.

由表 2 可知,SIMECK 算法代数次数随轮数的变化与 SIMON 算法相同,从第 13 轮开始,SIMON32 代数次数达到上界 31.此外,由于 SIMECK 算法结构与 SIMON 相同,计算真值表的时间花销与 SIMON 相同,因而

SIMECK32 代数次数的计算时间与表 1 大致相同,同样可以在不到两分钟内求解任意轮 SIMECK 32 算法的代数正规型,在 6 分钟内给出全轮 SIMECK32 算法的代数次数.

Table 1 Algebraic degree of SIMON32

表 1 SIMON32 的代数次数

轮数	布尔函数代数次数	ANF 平均求解时间(s)	代数次数平均求解时间(s)
1	1	64	64
2	2	64	64
3	3	66	66
4	5	68	68
5	8	69	70
6	13	70	70
7	18	72	73
8	24	72	124
9	27	73	290
10	29	74	309
11	30	74	309
12	30	74	310
13	31	74	310
…	…	…	…
32	31	90	328

Table 2 Algebraic degree of SIMECK32

表 2 SIMECK32 的代数次数

轮数	布尔函数代数次数	ANF 求解时间(s)	代数次数求解时间(s)
1	1	65	66
2	2	66	66
3	3	68	68
4	5	69	69
5	8	70	70
6	13	71	71
7	18	72	72
8	24	74	109
9	27	75	282
10	29	75	311
11	30	76	319
12	30	76	315
13	31	78	317
…	…	…	…
32	31	87	325

本节在利用真值表求解代数正规型算法的基础上构造了基于 CUDA 架构的代数次数的求解模型.随后,作为应用,求解了对分组长度较小的算法 SIMON 和 SIMECK 任意轮数的代数次数.对于已知体制和结构的密码算法,基于 CUDA 架构的代数次数求解模型的关键是利用 GPU 遍历求解算法的真值表.在此基础上,协同利用 CPU 求解代数正规型.与传统算法实现相比,有效分配了计算资源,极大地提高了求解代数正规型算法的运行效率,缩短了计算代数次数的时间.但是对于分组长度较大的算法,由于计算和存储资源有限,无法准确求解密码算法的代数次数,但仍然可以给出密码算法代数次数上界的估计.下面,我们利用 Cube 攻击理论与代数次数之间的关系,设计估算代数次数上界的概率算法.

3 基于 Cube 理论的代数次数的估计

Cube 攻击是由 Dinur 和 Shamir[9]在 2009 年欧密会上提出的一种基于代数思想的新型攻击方法,其攻击的主要原理是:将密码算法的输出可以描述为有限域 GF(2)上关于密钥 k 和公开变量 v(明文比特或者 IV 比特)的多项式 $f(v,k)$,攻击者通过选择部分公开变量组成 Cube,并询问密码系统获得输出值,从而得到关于密钥 k 的低次方程组,通过求解方程恢复一定数目的密钥 k.Cube 攻击在序列密码、分组密码和 Hash 函数的分析中具有广泛的应用.本节在 Cube 攻击理论的基础上分析了 Cube 攻击中超多项式的取值和代数次数之间的关系,设计了代

数次数的估计算法.

3.1 Cube理论基础

对于任意 n 元布尔函数 $f(x_1,x_2,\ldots,x_n)$,对任意指标集 $I=\{i_1,i_2,\ldots,i_k\}\subseteq\{1,2,\ldots,n\}$,记 $t_I=x_{i_1}x_{i_2}\ldots x_{i_k}$,则布尔函数 $f(x_1,x_2,\ldots,x_n)$总可以表示为如下的形式.

$$f(x_1,x_2,\ldots,x_n)=f_{S(I)}\cdot t_I\oplus q(x_1,x_2,\ldots,x_n) \tag{2}$$

其中,$f_{S(I)}$不含 t_I中的变量,$q(x_1,x_2,\ldots,x_n)$中不含能被 t_I整除的项.

称 $x_{i_1},x_{i_2},\ldots,x_{i_k}$ 为 k 个 Cube 变元,集合 $C_I=\{(x_1,x_2,\ldots,x_n)\in F_2^n \mid x_i\in F_2, i\in I; x_i=0, i\notin I\}$ 为一个 k 维 Cube,$f_{S(I)}$为指标集 I 对应的超多项式.

遍历 C_I所有的取值,对公式(2)求和可得:

$$\sum_{C_I} f(x_1,x_2,\ldots,x_n)=\sum_{C_I} t_I f_{S(I)}\oplus\sum_{C_I} q(x_1,x_2,\ldots,x_n) \tag{3}$$

由于 $q(x_1,x_2,\ldots,x_n)$中的项成对出现和为 0,故有下面的等式成立.

$$f_{S(I)}=\sum_{C_I} f(x_1,x_2,\ldots,x_n)\bmod 2 \tag{4}$$

由公式(4)可知,超多项式 $f_{S(I)}$的取值为布尔函数 $f(x_1,x_2,\ldots,x_n)$在 k 维 Cube C_I上的异或和.超多项式的取值与布尔函数的代数次数之间具有以下关系.

引理 1. 对于任意 n 元布尔函数 $f(x_1,x_2,\ldots,x_n)$,若对$\{1,2,\ldots,n\}$的任意 d 元子集 I,$f_{S(I)}$取值都为常数,则 $f(x_1,x_2,\ldots,x_n)$的代数次数至多为 d 次.

证明:若函数 $f(x_1,x_2,\ldots,x_n)$的代数次数大于 d,设其次数为 $s\geqslant d+1$,不妨设 $x_{i_1}x_{i_2}\ldots x_{i_{d+1}}\ldots x_{i_s}$ 为某个 s 次项,则存在指标集 $I=\{i_1,i_2,\ldots,i_d\}$,使其对应的超多项式 $f_{S(I)}$至少含有非线性项 $x_{i_{d+1}}\ldots x_{i_s}$,即 $f_{S(I)}$不是常数,这与已知条件矛盾. 证毕. □

引理 2. 对于任意 n 元布尔函数 $f(x_1,x_2,\ldots,x_n)$,若存在$\{1,2,\ldots,n\}$的某个 d 元子集 I,其对应的超多项式 $f_{S(I)}$不是常数,则 $f(x_1,x_2,\ldots,x_n)$的代数次数至少为 $d+1$ 次.

证明:对于布尔函数 $f(x_1,x_2,\ldots,x_n)$,由公式(2)可知,

$$f(x_1,x_2,\ldots,x_n)=f_{S(I)}\cdot t_I\oplus q(x_1,x_2,\ldots,x_n).$$

其中,t_I 为 d 次单项式,$f_{S(I)}$不是常数,$f_{S(I)}\cdot t_I$ 次数至少为 $d+1$,而 $q(x_1,x_2,\ldots,x_n)$中单项式不是 t_I 的倍式,因而不会和 $f_{S(I)}\cdot t_I$ 中的项抵消,所以 $f(x_1,x_2,\ldots,x_n)$代数次数至少为$d+1$. □

利用引理 1 和引理 2,通过测试某些Cube 集合对应的超多项式 $f_{S(I)}$的取值是否为常数,可以得到 $f(x_1,x_2,\ldots,x_n)$代数次数的上界和下界.下面我们给出定理 1,给出计算布尔函数代数次数的充要条件.

定理 1. 布尔函数 $f(x_1,x_2,\ldots,x_n)$的代数次数为 d 次,当且仅当对$\{1,2,\ldots,n\}$的任意 d 元子集 I,都有 $f_{S(I)}$都为常数,并且存在$\{1,2,\ldots,n\}$的某个 $d-1$ 元子集 I',其对应的超多项式 $f_{S(I')}$不是常数.

证明:充分性的证明由引理 1 和引理 2 可得,只需要证明必要性.若布尔函数 $f(x_1,x_2,\ldots,x_n)$的代数次数为 d 次,则由超多项式的定义和性质可知,对$\{1,2,\ldots,n\}$的任意 d 元子集 I,$f_{S(I)}$都为常数.另一方面,不妨设 $x_{i_1}x_{i_2}\ldots x_{i_d}$ 为 $f(x_1,x_2,\ldots,x_n)$的某个 d 次项,则存在 $d-1$ 元指标集$I'=\{i_1,i_2,\ldots,i_{d-1}\}$,使得 I'对应的超多项式 $f_{S(I')}=x_{i_d}$,$f_{S(I')}$不是常数. 证毕. □

根据定理 1 可知,利用 Cube 方法所找到的临界值 d,就是布尔函数 $f(x_1,x_2,\ldots,x_n)$的代数次数.定理 1 需要计算 d 元子集 I 对应的超多项式 $f_{S(I)}$的取值,为了计算方便,可以利用性质 1 进行计算.

性质 1[9]. 对于布尔函数 $f(x_1,x_2,\ldots,x_n)$,选择$\{1,2,\ldots,n\}$中的任意 d 元子集 $I=\{i_1,i_2,\ldots,i_d\}$,记 $L[\alpha_1,\alpha_2,\ldots,\alpha_d]$是一组基$\alpha_1,\alpha_2,\ldots,\alpha_d$ 生成的线性空间,其中,$\alpha_j=(0\ldots a_{i_j}\ldots 0)$.当 $i_j\in I$ 时,$a_{i_j}=1$, $j=1,2,\ldots,d$,那么子集 I 对应的超多项式 $f_{S(I)}$的值可以通过如下方式计算:

$$f_{S(I)}=\sum_{v\in C_I} f\mid_v=\sum_{\beta\in L[\alpha_1,\alpha_2,\ldots,\alpha_d]} f(x\oplus\beta).$$

3.2 基于Cube理论的代数次数的估计算法

对于分组密码算法,轮函数可以表示为关于该轮输入的布尔函数.当分组长度 $2n$ 较大,选取的 Cube 变元较多时,需要遍历所有的Cube变元集合,计算超多项式的值,计算量太大.我们采用随机选取Cube点的方法,设计了概率算法进行次数估计.算法的基本思路是:从低到高依次检测布尔函数代数次数是否为 d 次($1<d<n$),而在检测代数次数是否是 d 时,根据定理 1,随机选择不同的输入点进行测试,利用性质 1 计算不同测试点的超多项式的值,判断取值是否相等,进而估计代数次数,具体算法如算法 2 所示.

算法 2. 利用 Cube 理论估计代数次数.

输入:待估计次数的分组密码算法 E,最大检测次数 max.

输出:代数次数 d.

1. 设置 d=1,$count$=0;
2. **While** ($d<n$)
   {
3. 选择 d 个线性无关的 n 维向量$\{\alpha_1,\alpha_2,\ldots,\alpha_d\}$;
4. 随机选择输入 x,计算 $tmp=\sum_{\beta\in L[\alpha_1,\alpha_2,\ldots,\alpha_d]} E(x\oplus\beta)$;
5. **While** ($count$<max)
   {
6. 随机选择 $y\neq x$,计算 $sum=\sum_{\beta\in L[\alpha_1,\alpha_2,\ldots,\alpha_d]} E(y\oplus\beta)$;
7. **if** $sum\neq tmp$ **break**;
8. $count=count+1$;
   }
9. **if** $count$=max 输出代数次数 d;
10. **else** $d=d+1$;
    }

只要测试次数 max 的取值比较大,则可以较大的正确性保证估计结果是对的.算法 2 的时间复杂度为 $O(n2^d)$,空间复杂度为 $O(1)$,可以忽略不计.算法 2 的前提是随机选取 d 维 Cube 变元,遍历所有 Cube 变元的集合后,在现有有限的计算资源下,可以计算对应超多项式的值,因而特别适合于分组密码算法输入变元个数比较大而实际轮函数布尔函数的代数次数比较小的情况.

3.3 在SIMON-like算法中的应用

我们利用算法 2 针对循环移位常数(a,b,c)取值为全为奇数和全为偶数的 Simon-like 算法的代数次数进行估计,以 SIMON[1,7,3]为例,不同分组长度的求解结果具体见表 3.

由表 3 可知,分组长度为 $2n$ 的 Simon-like 算法在参数选择(1,7,3)时,不同轮代数次数增长缓慢,且达不到代数次数的上界 $2n-1$.分组长度为 32 比特的算法从第 9 轮开始代数次数保持 15 不变,分组长度为 48 比特的算法从第 11 轮开始代数次数保持 23 不变,分组长度为 64 比特的算法从第 13 轮开始代数次数保持 31 不变,代数次数的上确界为 $n-1$.事实上,对于参数选择全是奇数或者偶数的 SIMON-like 算法,都有以下定理成立.

定理 2. 对于分组长度为 $2n$ 的 SIMON-like$[a,b,c]$算法,布尔函数 $f:F_2^{2n}\to F_2$ 为迭代 r 轮的函数,代数次数记为 $\deg(f)$,若循环移位参数(a,b,c)同为奇数或同为偶数,则对任意的轮数 r,都有 $\deg(f)\leqslant n-1$.

证明:以分组长度为32的SIMON-like为例,不妨假设循环移位参数(a,b,c)同为奇数.首先利用数学归纳法证明:迭代 r 轮输出的每一比特的值至多只由初始输入的 n 比特决定.

记 SIMON-like 32 第 1 轮的输入为 $x_0,x_1,\ldots,x_{15},x_{16},\ldots,x_{31}$,输出为 $y_0,y_1,\ldots,y_{15},y_{16},\ldots,y_{31}$,第 3 轮的输入为 $z_0,z_1,\ldots,z_{15},z_{16},\ldots,z_{31}$,迭代轮数为 r 轮.根据算法结构,第 1 轮的输出按比特表示成输入比特的布尔函数为

$$\begin{cases} y_i = x_{(i+a)\bmod 16} \cdot x_{(i+b)\bmod 16} \oplus x_{(i+c)\bmod 16} \oplus x_{15+i}, & 0 \leqslant i \leqslant 15 \\ y_i = x_{i-16}, & 16 \leqslant i \leqslant 31 \end{cases}.$$

因为(a,b,c)同为奇数,所以 $y_i(i=1,3,\ldots,15,16,\ldots,30)$能够表示成输入比特集合$(x_0,x_2,\ldots,x_{14},x_{17},\ldots,x_{31})$的布尔函数,$y_i(i=0,2,\ldots,14,17,\ldots,31)$能够表示成输入比特集合$(x_1,x_3,\ldots,x_{15},x_{16},\ldots,x_{30})$的布尔函数.

对于第 2 轮的输出,表示为第 2 轮输入的布尔函数为

$$\begin{cases} z_i = y_{(i+a)\bmod 16} \cdot y_{(i+b)\bmod 16} \oplus y_{(i+c)\bmod 16} \oplus y_{15+i}, & 0 \leqslant i \leqslant 15 \\ z_i = y_{i-16}, & 16 \leqslant i \leqslant 31 \end{cases}.$$

因为(a,b,c)同为奇数,所以 $z_i(i=1,3,\ldots,15,16,\ldots,30)$能够表示成输入比特集合$(y_0,y_2,\ldots,y_{14},y_{17},\ldots,y_{31})$的布尔函数,$z_i(i=0,2,\ldots,14,17,\ldots,31)$能够表示成输入比特集合$(y_1,y_3,\ldots,y_{15},y_{16},\ldots,y_{30})$的布尔函数.综合两轮的结果可知,$z_i(i=1,3,\ldots,15,16,\ldots,30)$能够表示成输入比特集合$(x_1,x_3,\ldots,x_{15},x_{16},\ldots,x_{30})$的布尔函数,$z_i(i=0,2,\ldots,14,17,\ldots,31)$能够表示成输入比特集合$(x_0,x_2,\ldots,x_{14},x_{17},\ldots,x_{31})$的布尔函数.

假设对于任意的 $r \leqslant k \in \mathbf{N}$,SIMON-like 32 输出的每一比特至多只由初始输入的 n 比特决定(第 1,3,…,15,16,…,30 比特或第 0,2,…,14,17,…,31 比特),则当 $r=k$ 时,不妨设 SIMON-like 32 算法的第 $k-1$ 轮的输出分别为$(w_0,w_1,\ldots,w_{15},w_{16},\ldots,w_{31})$,第 k 轮的输出分别为$(v_0,v_1,\ldots,v_{15},v_{16},\ldots,v_{31})$.若 k 为偶数,$k-1$ 为奇数,第 k 轮的输出表示为第 k 轮输入的布尔函数为

$$\begin{cases} v_i = w_{(i+a)\bmod 16} \cdot w_{(i+b)\bmod 16} \oplus w_{(i+c)\bmod 16} \oplus w_{15+i}, & 0 \leqslant i \leqslant 15 \\ v_i = w_{i-16}, & 16 \leqslant i \leqslant 31 \end{cases}.$$

因为(a,b,c)同为奇数,所以 $v_i(i=1,3,\ldots,15,16,\ldots,30)$能够表示成输入比特集合$(w_0,w_2,\ldots,w_{14},w_{17},\ldots,w_{31})$的布尔函数,$v_i(i=0,2,\ldots,14,17,\ldots,31)$能够表示成输入比特集合$(w_1,w_3,\ldots,w_{15},w_{16},\ldots,w_{30})$的布尔函数.由归纳假设可知,$v_i(i=1,3,\ldots,15,16,\ldots,30)$能够表示成输入比特集合$(x_1,x_3,\ldots,x_{15},x_{16},\ldots,x_{30})$的布尔函数,不存在的项即系数为 0;$v_i(i=0,2,\ldots,14,17,\ldots,31)$能够表示成输入比特集合$(x_0,x_2,\ldots,x_{14},x_{17},\ldots,x_{31})$的布尔函数,不存在的项即系数为 0.若 k 为奇数,同理可以证明.因而当循环移位参数(a,b,c)同为奇数时,对任意的 r 轮的 SIMON-like 32 算法,输出的每一比特的值至多只由初始输入的 n 比特决定.所以输出比特表示成输入比特的布尔函数 f 时,代数次数 $\deg(f)$至多为 $n-1$ 次.循环移位参数同为偶数时证明类似.定理 2 得证. □

Table 3 Algebraic degree estimation of SIMON[1,7,3]

表 3 SIMON[1,7,3]代数次数的估计

轮数	分组 32 比特	分组 48 比特	分组 64 比特
1	1	1	1
2	2	2	2
3	3	3	3
4	5	5	5
5	8	8	8
6	12	13	13
7	13	17	19
8	14	20	24
9	15	21	27
10	15	22	29
11	15	23	30
12	15	23	30
13	15	23	31
…	…	…	…
32	15	23	31

本节在 Cube 攻击理论基础上,研究了代数次数和超多项式取值之间的关系,将代数次数的估计问题转化为超多项式值的求解.在此基础上,设计了概率算法估计一般 SIMON-like 算法的代数次数.作为应用,求解了参数选取全为奇数或者全为偶数的 SIMON-like 算法任意轮数的代数次数,并从理论上证明了参数选择全为奇数和偶数的 SIMON-like 算法代数次数的上确界.下面,我们从布尔函数代数次数的角度出发,进一步讨论循环移位参数的选择对 SIMON-like 算法的影响,给出算法设计时参数的选择依据.

4 SIMON-like 算法循环移位参数选取分析

目前,很多不同结构的分组密码算法,不管是密钥扩展算法还是算法运行的轮函数,都会采用循环移位参数防止对密钥进行逐段破译以及隐蔽明文的统计特性,选择不好的循环移位参数会对算法的安全性产生致命的影响.目前公开的算法,设计者大多没有给出循环移位参数的选择依据和设计标准.对于非线性部件简单的SIMON-like 算法,选择不好的循环参数会导致算法分解为几个独立的简单算法,从而减少算法分析过程需要的复杂度.由定理 2 的证明过程可知,分组长度为 $2n$ 的 Simon-like 算法,当循环移位参数(a,b,c)全为奇数或者全为偶数时,算法可以分解为两个分组长度为 n 的小算法,无法保证原算法的安全强度,算法攻击的难度大幅减少.因而为了安全起见,SIMON-like 算法在选择循环移位参数时,应该避免参数选择全为奇数或者全为偶数.

SIMON 和 SIMECK 算法循环移位参数的选择既有奇数也有偶数,但设计者并没有给出其他的选取依据.Kölbl 等人[22]在 CRYPTO 2015 上,从抵抗差分和线性攻击的角度对循环移位常数的选取进行了分析,通过研究 SIMON 循环移位常数对其扩散性的影响,证明了 SIMON 算法的原始参数并不具有最强的抗差分和线性的能力.此外,文献[22]根据相关的评判准则,从所有可能的移位常数(a,b,c)中选择了 3 个优于原参数(8,1,2)的参数 SIMON[12,5,3],SIMON[7,0,2],SIMON[1,0,2].随后,Kondo 等人[23]从抵抗不可能差分攻击和积分攻击的能力评判了参数选择的优劣,发现参数 SIMON[12,5,3]总体的安全性比原始参数高.我们从布尔函数代数次数的角度出发,对 SIMON-like 型算法轮函数循环移位参数的选取进行分析.利用第 2 节和第 3 节的算法,对 Kölbl 等人选出的表现优良的 SIMON[1,0,2],SIMON[12,5,3],SIMON[7,0,2]算法的代数次数进行求解,以分组长度为 32 的算法为例,求解结果见表 4~表 6.

由表 4 可知,与 SIMON 算法相比,SIMON[1,0,2]轮函数的代数次数增长缓慢.从第 16 轮开始,SIMON[1,0,2]代数次数才达到上界 31.从积分攻击的角度来看,如果代数次数增长缓慢,达到上界所需的轮数越大,则可以找到更长轮数的积分区分器,进而攻击更长轮数的算法.因此,与原参数相比,SIMON[1,0,2]算法抵抗积分攻击的能力明显较弱.

由表 5 可知,与 SIMON 算法相比,SIMON[12,5,3]轮函数的代数次数增长缓慢.从第 14 轮开始,SIMON[1,0,2]代数次数达到上界 31.从积分攻击的角度来看,SIMON[12,5,3]可以找到至少比 SIMON 多 1 轮的积分区分器.因此,从抵抗积分攻击的角度来看,SIMON[12,5,3]算法没有原始参数安全.

由表 6 可知,虽然 SIMON[7,0,2]与 SIMON 算法轮函数的代数次数都是从第 13 轮开始上界 31,但是 SIMON[7,0,2]每一轮布尔函数的代数次数没有对应轮 SIMON 算法的大,从算法攻击所需的复杂度来看,攻击原始参数的 SIMON 算法难度更大,需要的复杂度也更大.因而原始参数具有更高的安全性.

Table 4 Algebraic degree of SIMON[1,0,2]

表 4 SIMON[1,0,2]的代数次数

轮数	布尔函数代数次数
1	1
2	2
3	3
4	5
5	8
6	11
7	14
8	17
9	20
10	24
11	27
12	29
13	29
14	30
15	30
16	31
…	…
32	31

Table 5 Algebraic degree of SIMON[12,5,3]

表 5 SIMON[12,5,3]的代数次数

轮数	布尔函数代数次数
1	1
2	2
3	3
4	5
5	8
6	12
7	16
8	22
9	26
10	28
11	29
12	30
13	30
14	31
…	…
32	31

Table 6 Algebraic degree of SIMON[7,0,2]

表 6 SIMON[7,0,2]的代数次数

轮数	布尔函数代数次数
1	1
2	2
3	3
4	5
5	8
6	12
7	17
8	21
9	26
10	28
11	29
12	30
13	31
…	…
32	31

5 总 结

本文主要给出了两种布尔函数代数次数的求解模型:一种在利用真值表求解代数正规型算法的基础上建立了基于 CUDA 的并行求解架构,协同利用 CPU 和 GPU 的计算资源,极大地缩短了求解代数次数的时间,在较短的时间内求解了 SIMON32,SIMECK32 算法任意轮数的代数正规型和代数次数;另一种求解模型根据代数次数和 Cube 攻击中超多项式取值之间的关系设计了概率算法,估计了一般 SIMON-like 型算法的代数次数.作为应用,从布尔函数代数次数的角度出发,给出了 SIMON-like 算法轮函数在选择不同循环移位参数时,代数次数变化的差异性,进而给出循环移位参数的选取依据.实验结果表明,与其他循环移位参数相比,SIMON 算法在原始参数下达到最大代数次数所需的轮数最短,说明原始参数具有更高的安全性.下一步将研究代数次数的求解模型在其他类型算法设计与分析中的应用.

References:

[1] Li N, Qi W. Symmetric Boolean functions depending on an ODD number of variables with maximum algebraic immunity. IEEE Trans. on Information Theory, 2006,52(5):2271−2273.

[2] Peng J, Wu Q, Kan H. On symmetric Boolean functions with high algebraic immunity on even number of variables. IEEE Trans. on Information Theory, 2011,57(10):7205−7220.

[3] Wang H, Peng J, Li Y, Kan H. On 2*k*-variable symmetric Boolean functions with maximum algebraic immunity *k*. IEEE Trans. on Information Theory, 2012,58(8):5612−5624.

[4] Wang Q, Peng J, Kan H, Xue X. Constructions of cryptographically significant Boolean functions using primitive polynomials. IEEE Trans. on Information Theory, 2010,56(6):3048−3053.
[5] Carlet C, Feng K. An infinite class of balanced functions with optimal algebraic immunity, good immunity to fast algebraic attack and good nonlinearity. In: Proc. of the ASIACRYPT 2008. LNCS 5350, Berlin: Springer-Verlag, 2008. 425−440.
[6] Yang J, Zhang W. Generating highly nonlinear resilient Boolean functions resistance against algebraic and fast algebraic attacks. Security and Communication Networks, 2015,8(7):1256−1264.
[7] Courtois N, Meier W. Algebraic attacks on stream ciphers with linear feedback. In: Proc. of the EUROCRYPT 2003. LNCS 2656, Berlin: Springer-Verlag, 2003. 345−359.
[8] Lai X. Higher order derivatives and differential cryptanalysis. In: Proc. of the Communications and Cryptography. Kluwer Academic Press, 1994. 227−233.
[9] Dinur I, Shamir A. Cube attacks on tweakable black box polynomials. In: Proc. of the EUROCRYPT 2009. LNCS 5479, Berlin: Springer-Verlag, 2009. 278−299.
[10] Knudsen L, Wagner D. Integral cryptanalysis. In: Proc. of the FSE 2002. LNCS 2365, Berlin: Springer-Verlag, 2002. 112−127.
[11] Climent H, Garca F, Requena V. Computing the degree of a Boolean function from its support. In: Proc. of the ISITA 2010. IEEE, 2010. 123−128.
[12] Canteaut A, Videau M. Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis. In: Proc. of the EUROCRYPT 2002. LNCS 2332, Berlin: Springer-Verlag, 2002. 518−533.
[13] Todo Y. Structural evaluation by generalized integral property. In: Proc. of the EUROCRYPT 2015. LNCS 9056, Berlin: Springer-Verlag, 2015. 287−314.
[14] Todo Y, Morii M. Bit-based division property and application to Simon family. In: Proc. of the FSE 2016. LNCS 9783, Berlin: Springer-Verlag, 2016. 357−377.
[15] Liu M. Degree evaluation of NFSR-based cryptosystems. In: Proc. of the CRYPTO 2017. LNCS 10403, Berlin: Springer-Verlag, 2017. 227−249.
[16] Beaulieu R, Shors D, Smith J, *et al*. The SIMON and SPECK families of lightweight block ciphers. IACR Cryptology ePrint Archive, Report, 2013/404, 2013. http://eprint.iacr.org/2013/404
[17] Yang G, Zhu B, Suder V, Aagaard MD, Gong G. The Simeck family of lightweight block ciphers. In: Proc. of the CHES 2015. LNCS 9293, Berlin: Springer-Verlag, 2015. 307−329.
[18] Johansson T. A framework for chosen IV statistical analysis of stream ciphers. In: Proc. of the INDOCRYPT 2007. LNCS 4859, Berlin: Springer-Verlag, 2007. 268−281.
[19] John C, Max G, Ty M. Professional CUDA C Programming. Indianapolis: Wrox, 2014.
[20] Cook S. CUDA Programming: A developer's Guide to Parallel Computing with GPUs. Newnes, 2012.
[21] Joux A. Algotithmic Cryptanalysis. Chapman & Hall/CRC, 2009.
[22] Köbl S, Leander G, Tiessen T. Observations on the SIMON block cipher family. In: Proc. of the CRYPTO 2015. LNCS 9215, Berlin: Springer-Verlag, 2015. 161−185.
[23] Kondo K, Yu S, Iwata T. On the design rationale of Simon block cipher: Integral attacks and impossible differential attacks against Simon variants. In: Proc. of the ACNS 2016. LNCS 9696, Berlin: Springer-Verlag, 2016. 518−536.

任炯炯(1995－),男,博士,讲师,主要研究领域为对称密码设计与分析.

林键(1989－),男,博士生,主要研究领域为信息安全.

李航(1995－),男,硕士,主要研究领域为对称密码设计与分析.

陈少真(1967－),女,博士,教授,博士生导师,主要研究领域为密码学与信息安全.