软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(8):3891−3904 [doi: 10.13328/j.cnki.jos.006543]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

对一种白盒 SM4 方案的差分计算分析*

原梓清[1], 陈 杰[1,2]

[1](综合业务网理论及关键技术国家重点实验室 (西安电子科技大学), 陕西 西安 710071)

[2](广西密码学与信息安全重点实验室 (桂林电子科技大学), 广西 桂林 541004)

通信作者: 陈杰, E-mail: jchen@mail.xidian.edu.cn

摘 要: 传统密码算法的安全性建立在黑盒攻击模型下. 在这种攻击模型下, 攻击者只能获取密码算法的输入输出, 而无法得知密码算法运行时的内部细节. 近年来白盒攻击模型的概念被提出. 在白盒攻击模型下, 攻击者既可以获取密码算法的输入输出, 也可以直接观测或更改密码算法运行时的内部数据. 为保证已有密码算法在白盒攻击环境下的安全性, 在不改变其功能的基础上通过白盒密码技术对其进行重新设计被称为已有密码算法的白盒实现. 研究白盒实现方案的设计与分析对于解决数字版权管理问题具有重要意义. 近年来, 出现了一类针对白盒实现方案的旁信道分析方法. 这类分析手段只需要知道很少白盒实现方案的内部细节, 却可以提取到密钥, 因此是一类对现有白盒实现方案具有实际威胁的分析手段. 对现有白盒实现方案进行此类分析对于确保方案安全性具有重要现实意义. 此类分析方法中的典型代表是基于差分功耗分析原理的差分计算分析. 基于差分计算分析, 对白-武白盒 SM4 方案进行了安全性分析. 基于对 GF(2) 上 n 阶均匀随机可逆矩阵统计特征的研究结果, 提出了一种改进型差分计算分析 (IDCA), 可以在分析成功率几乎不变的前提下显著提升分析效率. 结果表明, 白-武白盒 SM4 方案在面对差分计算分析时不能保证安全性, 必须对其进行进一步改进使之满足实际应用场景下的安全性需求.

关键词: 白盒密码; 白盒实现; SM4 算法; 旁信道分析; 差分计算分析

中图法分类号: TP309

中文引用格式: 原梓清, 陈杰. 对一种白盒SM4方案的差分计算分析. 软件学报, 2023, 34(8): 3891–3904. http://www.jos.org.cn/1000-9825/6543.htm

英文引用格式: Yuan ZQ, Chen J. Differential Computation Analysis of White-box SM4 Scheme. Ruan Jian Xue Bao/Journal of Software, 2023, 34(8): 3891–3904 (in Chinese). http://www.jos.org.cn/1000-9825/6543.htm

Differential Computation Analysis of White-box SM4 Scheme

YUAN Zi-Qing[1], CHEN Jie[1,2]

[1](State Key Laboratory of Integrated Services Networks (Xidian University), Xi'an 710071, China)

[2](Guangxi Key Laboratory of Cryptography and Information Security (Guilin University of Electronic Technology), Guilin 541004, China)

Abstract: The security of traditional cryptographic algorithms is based on the black-box attack model. In this attack model, the attacker can only obtain the input and output of the cryptographic algorithm, but not the internal details of the cryptographic algorithm. In recent years, the concept of white-box attack model has been proposed. In the white-box attack model, attackers can not only obtain the input and output of cryptographic algorithm, but also directly observe or change the internal data of cryptographic algorithm. In order to ensure the security of existing cryptographic algorithms under white-box attack environment, redesigning the existing cryptographic algorithms through white-box cryptography technology without changing their functions is called white-box implementation of existing cryptographic

* 基金项目: "十三五"国家密码发展基金 (MMJJ20180219); 陕西省自然科学基础研究计划 (2021JM-126); 广西密码学与信息安全重点实验室研究课题 (GCIS202125)
收稿时间: 2021-07-05; 修改时间: 2021-08-26, 2021-10-05; 采用时间: 2021-12-02; jos 在线出版时间: 2022-09-23
CNKI 网络首发时间: 2022-11-15

algorithms. It is of great significance to study the design and analysis of the white-box implementation scheme for solving the issue of digital rights management. In recent years, a kind of side channel analysis method for white-box implementation schemes has emerged. This kind of analysis method only needs to know a few internal details of white-box implementation schemes, then it can extract the key. Therefore, it is the analysis method with practical threat to the existing white-box implementation schemes. It is of great practical significance to analyze the existing white-box implementation schemes to ensure the security of the schemes. The typical representative of this kind of analysis method is the differential computation analysis (DCA) based on the principle of differential power analysis. This study analyzes the Bai-Wu white-box SM4 scheme based on DCA. Based on the research results of the statistical characteristics of *n*-order uniform random invertible matrix on GF(2), an improved DCA (IDCA) is proposed, which can significantly improve the analysis efficiency on the premise of almost constant success rate. The results also show that the Bai-Wu white-box SM4 scheme can not guarantee the security in the face of DCA, therefore, it must be further improved to meet the security requirements of practical scenarios.

Key words: white-box cryptography; white-box implementation; SM4 algorithm; side channel analysis; differential computation analysis (DCA)

传统密码算法的安全性建立在运行环境可信的假设下. 在这种假设下, 攻击者只能获取密码算法的输入输出, 而无法得知密码算法运行时的内部细节, 我们称此时的攻击模型为黑盒攻击模型 (black-box attack model). 随着信息技术的快速发展, 计算机、智能手机与互联网已被广泛应用于生产生活中, 视频、音乐、图片、软件等数字信息被广泛传播. 在这种情况下, 数字产品常常运行在不可信任终端上, 传统密码算法的运行环境已不再单纯可信, 其安全性受到了新的挑战. 例如, 用户在自己的计算机上用视频播放器播放加密过的视频. 若用户是会员, 则视频播放器使用正确的密钥对视频进行解密后播放. 这种情况下, 这个解密过程很可能是不安全的, 因为整个解密过程是在用户的计算机上进行的, 攻击者 (甚至可能是用户本人) 可以直接观测密码算法程序运行时的内部细节, 轻易获得密钥信息. 若攻击者将密钥信息泄露给非会员用户 (无授权用户), 则那些非会员用户就有可能根据所得密钥信息对加密过的视频直接进行解密, 从而跳过了身份认证过程, 这显然是视频发布方不想看到的.

上文所述针对密码算法运行终端的攻击, 有如下显著特点.

(1) 密码算法运行环境不可信, 攻击者对密码算法程序的运行具有完全的控制力, 能够直接观测或更改程序运行时的内部数据.

(2) 攻击目标是密码算法的软件实现, 攻击目的是提取密钥.

我们称这种攻击模型为白盒攻击模型 (white-box attack model). 与传统黑盒攻击模型不同, 白盒攻击模型对于攻击者能力的限制很少. 在不可信任终端环境下, 白盒攻击是更高级的安全威胁. 能够抵抗白盒攻击的密码算法及其实现被称为白盒密码 (white-box cryptography). 其中, 已有密码算法的白盒实现是指将已有的密码算法通过白盒密码技术进行设计, 使得在白盒攻击环境下, 不改变原算法的功能但原算法所希望保证的安全性不受破坏.

白盒攻击模型与白盒密码的概念由 Chow 等人于 2002 年首次提出[1]. 在文献 [1] 中, 针对白盒攻击模型, Chow 等人提出了一种对已有密码算法白盒化处理的方法, 即将原密码算法改编为嵌入密钥的查找表网络, 并对每个查找表的输入输出以及整个查找表网络的输入输出用混淆编码进行保护. 基于此, Chow 等人设计了一个 AES 的白盒实现方案[1]以及一个 DES 的白盒实现方案[2]. 此后的白盒实现方案大多沿用 Chow 等人的设计思路, 典型代表有肖雅莹设计的 AES 和 SM4 的白盒实现方案[3], Luo 等人设计的 AES 的白盒实现方案[4], Shi 等人设计的 SM4 的白盒实现方案[5], 以及 Bai 等人设计的 SM4 的白盒实现方案[6]和 AES 的白盒实现方案[7].

对白盒实现方案的安全性分析, 目前主要有两种思路: 一种是在纯白盒环境下, 通过对算法内部细节的安全性分析找到漏洞, 并采用查找表组合、仿射等价算法等手段消除混淆编码, 对密钥相关的查找表进行分析并提取密钥. 这类分析方法要求攻击者洞悉白盒实现方案的内部细节, 实际应用场景中很难满足这样的条件. 这类分析方法的典型代表有: Billet 等人提出的 BGE 攻击[8], Michiels 等人提出的 MGH 攻击[9], Mulder 等人对肖-来白盒 SM4 方案的攻击[10], 以及林婷婷等人对史扬白盒 SM4 方案的攻击[11]. 另一种对白盒实现的分析思路, 是在灰盒环境下, 利用某些软件采集算法程序运行时泄露的信息, 并采用相应的统计分析方法提取密钥. 这类分析方法属于旁信道分析方法, 攻击者不需要过多知道白盒实现的内部细节, 只需掌握白盒实现的底层算法, 能够监测算法程序的运行

状态即可进行分析. 但是这类分析方法需要收集较多的信息以确保较高的分析成功率, 因此需要算法程序能够多次反复运行. 与纯白盒环境下的分析相比, 灰盒环境下的分析所需条件较低, 在实际应用场景中对白盒实现方案具有更高的威胁, 必须得到充分重视. 针对白盒实现方案的旁信道分析方法是在 2016 年由 Bos 等人首次提出的差分计算分析 (differential computation analysis, DCA)[12]. 差分计算分析是基于差分功耗分析 (differential power analysis, DPA) 提出的分析方法, 可借助特定的软件收集算法程序运行时泄露的信息, 并对其进行分析从而提取密钥. Bos 等人[12]还运用差分计算分析对几个白盒 AES 方案进行了成功分析, 但是没有说明差分计算分析能够成功的理论依据. 其后几年的研究者针对差分计算分析做了大量的工作. 文献 [13,14] 详细分析了差分计算分析的理论依据. 文献 [15] 提出一种方法, 可以减小软件迹的尺度从而提高差分计算分析的效率. 为对抗差分计算分析, 2017 年 Banik 等人提出了控制流混淆、查找表位置随机化、虚拟操作等软件补偿措施[16], 2018 年 Biryukov 等人提出了基于掩蔽 (masking) 的改进方案[17]. 为分析包含软件补偿对策的白盒实现方案, 2018 年 Bogdanov 等人提出了高阶差分计算分析 (higher order differential computational analysis)[18]. 此后 2020 年 Lee 等人又提出了针对高阶差分计算分析的抵抗策略[19]. 2021 年, Biryukov 等人提出将 Dummy Shuffling 应用于白盒实现以抵抗旁信道分析[20]. 历经多年发展, 有关差分计算分析的理论不断得到完善, 如今该分析方法已成为检验白盒实现方案安全性的重要工具.

本文运用差分计算分析方法, 对白鲲鹏等人设计的 SM4 的白盒实现方案 (后称白-武白盒 SM4 方案) 进行了成功分析, 论证了该方案在面对差分计算分析时不能保证安全性. 此外, 基于我们对 GF(2) 上 n 阶均匀随机可逆矩阵统计特征的研究成果, 我们提出了一种改进型差分计算分析方法, 可以在分析成功率几乎不变的前提下提升分析效率.

本文第 1 节介绍了所需的预备知识, 包括 SM4 分组密码算法、白-武白盒 SM4 方案以及差分计算分析. 第 2 节描述了我们对 GF(2) 上 n 阶均匀随机可逆矩阵统计特征的理论分析结果. 第 3 节对白-武白盒 SM4 方案进行了差分计算分析. 第 4 节提出了一种改进型差分计算分析, 并与第 3 节中所用分析方法进行了对比. 最后总结全文.

1 预备知识

1.1 SM4 分组密码算法

SM4 (原名 SMS4) 分组密码算法是国家商用密码管理局于 2006 年首次公开发布的. 2012 年 3 月, SM4 被列为国家密码行业标准. 2016 年 8 月, SM4 被列为国家标准 (GB/T 32907-2016)[21]. 2017 年 10 月, SM4 被 ISO 标准列为补篇草案. 时至今日, SM4 已成为国内主流标准分组密码之一.

SM4 的分组长度与初始密钥长度均为 128 位. 与通常的分组密码算法一样, SM4 包括密钥扩展算法与数据处理算法两大部分.

密钥扩展算法[21]根据 128 位的初始密钥生成 32 个轮密钥 rk_1, rk_2, …, rk_i, …, rk_{32}, 其中 $rk_i \in Z_2^{32}$, $i=1,2,\ldots,32$, Z_2^{32} 表示定义在 GF(2) 上的 32 维向量集合.

数据处理算法包括加密变换与解密变换. 设输入的明文为 $(X_0,X_1,X_2,X_3)\in(Z_2^{32})^4$, 输出的密文为 $(Y_0,Y_1,Y_2,Y_3)\in(Z_2^{32})^4$, 轮密钥为 rk_1, rk_2, …, rk_i, …, rk_{32}, $i=1,2,\ldots,32$, 则 SM4 的加密变换可用公式 (1) 和公式 (2) 表示:

$$X_{i+3}=X_{i-1}\oplus T(X_i\oplus X_{i+1}\oplus X_{i+2}\oplus rk_i),\ i=1,2,\ldots,32 \tag{1}$$

$$(Y_0,Y_1,Y_2,Y_3)=(X_{35},X_{34},X_{33},X_{32}) \tag{2}$$

其中, $T=L\circ\tau$. τ 是非线性代换, $\tau(x)=S(x_0)\|S(x_1)\|S(x_2)\|S(x_3)$, $x=x_0\|x_1\|x_2\|x_3$, $x\in Z_2^{32}$, $x_i\in Z_2^8$, Z_2^8 表示定义在 GF(2) 上的 8 维向量集合. S 是 SM4 的 S 盒查表运算[21]. L 是线性置换, $L(x)=x\oplus(x<<<2)\oplus(x<<<10)\oplus(x<<<18)\oplus(x<<<24)$, $x\in Z_2^{32}$, L 也可用 32 阶矩阵 M 表示为公式 (3):

$$M=\begin{bmatrix} A & B & B & C \\ C & A & B & B \\ B & C & A & B \\ B & B & C & A \end{bmatrix} \tag{3}$$

其中, $A=\begin{bmatrix}1&0&1&0&0&0&0&0\\0&1&0&1&0&0&0&0\\0&0&1&0&1&0&0&0\\0&0&0&1&0&1&0&0\\0&0&0&0&1&0&1&0\\0&0&0&0&0&1&0&1\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}$, $B=\begin{bmatrix}0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\\1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\end{bmatrix}$, $C=\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\1&0&0&0&0&0&1&0\\0&1&0&0&0&0&0&1\end{bmatrix}$.

解密变换与加密变换过程相同, 只是轮密钥使用顺序相反, 在此不再赘述.

1.2 白-武白盒 SM4 方案

白-武白盒 SM4 方案[6]采用随机的仿射变换进行编码, 并应用查找表保护算法运行时的内部信息. 该方案引入了较多的随机量, 并且设计了更加复杂的内部编码解码过程, 这大大增加了该方案的分析难度. 白-武白盒 SM4 方案的实现过程如下[22].

(1) 根据密钥扩展算法, 由 128 位的初始密钥生成 32 个 32 位的轮密钥 rk_1, rk_2, …, rk_{32}.

(2) 生成随机矩阵与随机向量.

首先, 生成 32 阶随机可逆矩阵 P_0, P_1, …, P_{35}, E_1, E_2, …, E_{32}, F_1, F_2, …, F_{32} 以及 32 位随机向量 p_i, $p_{i',j}$, $p'_{r+3,j}$, $p''_{r+3,j}$, $e_{r,0}$, $e_{r,1}$, …, $e_{r,5}$, $f_{r,0}$, $f_{r,1}$, …, $f_{r,5}$. 其中, $i=0, 1, \ldots, 35$, $i'=0, 1, \ldots, 34$, $j=0, 1, 2, 3$, $r=1, 2, \ldots, 32$, $\sum_{j=0}^{3} p_{i',j}=p_{i'}$, $\sum_{j=0}^{3} p'_{r+3,j}=p'_{r+3}$, $\sum_{j=0}^{3} p''_{r+3,j}=p''_{r+3}$, $p'_{r+3}\oplus p''_{r+3}=p_{r+3}$, $E_r=diag(E_{r,0},E_{r,1},E_{r,2},E_{r,3})$, $F_r=diag(F_{r,0},F_{r,1},F_{r,2},F_{r,3})$.

然后按照以下步骤生成 16 位随机向量 $u_{r,i}$.

(a) 计算 $e_r=\sum_{i=0}^{5} e_{r,i}$, $f_r=\sum_{i=0}^{5} f_{r,i}$.

(b) 记 $e_r=e_{r,0}\|e_{r,1}\|e_{r,2}\|e_{r,3}$, $f_r=f_{r,0}\|f_{r,1}\|f_{r,2}\|f_{r,3}$.

(c) 将 $e_{r,i}$, $f_{r,i}$ 重组为 16 位向量 $u_{r,i}=e_{r,i}\|f_{r,i}$, 其中 $i=0, 1, 2, 3$, $r=1, 2, \ldots, 32$.

(3) 用轮密钥, 随机矩阵与随机向量生成查找表.

第 r 轮, 存储以下 16 个查找表 (遍历 8 位的输入值 x 存储 32 位的输出值 y), 具体过程可由公式 (4)–公式 (10) 表示:

$$TC_{r,j}: y=P_{r+3}\cdot(P_{r-1,j}^{-1}\cdot x\oplus P_{r-1}^{-1}\cdot p_{r-1,j})\oplus p'_{r+3,j},\ j=0,1,2,3 \tag{4}$$

$$TA_{r,j}: y=E_r\cdot(P_{r,j}^{-1}\cdot x\oplus P_r^{-1}\cdot p_{r,j})\oplus e_{r,j},\ j=0,1 \tag{5}$$

$$TA_{r,j+2}: y=E_r\cdot(P_{r+1,j}^{-1}\cdot x\oplus P_{r+1}^{-1}\cdot p_{r+1,j})\oplus e_{r,j+2},\ j=0,1 \tag{6}$$

$$TA_{r,j+4}: y=E_r\cdot(P_{r+2,j}^{-1}\cdot x\oplus P_{r+2}^{-1}\cdot p_{r+2,j})\oplus e_{r,j+4},\ j=0,1 \tag{7}$$

$$TB_{r,j}: y=F_r\cdot(P_{r,j+2}^{-1}\cdot x\oplus P_r^{-1}\cdot p_{r,j+2})\oplus f_{r,j},\ j=0,1 \tag{8}$$

$$TB_{r,j+2}: y=F_r\cdot(P_{r+1,j+2}^{-1}\cdot x\oplus P_{r+1}^{-1}\cdot p_{r+1,j+2})\oplus f_{r,j+2},\ j=0,1 \tag{9}$$

$$TB_{r,j+4}: y=F_r\cdot(P_{r+2,j+2}^{-1}\cdot x\oplus P_{r+2}^{-1}\cdot p_{r+2,j+2})\oplus f_{r,j+4},\ j=0,1 \tag{10}$$

以及以下 4 个查找表 (遍历 16 位的输入值 x 存储 32 位的输出值 y), 可由公式 (11) 表示:

$$TD_{r,j}: y=P_{r+3}\cdot M_j\cdot S((E_{r,j}^{-1}\|F_{r,j}^{-1})\cdot(x\oplus u_{r,j})\oplus rk_{r,j})\oplus p''_{r+3,j}, r=1,2,\ldots,32, j=0,1,2,3 \tag{11}$$

其中, $(E_{r,j}^{-1}\|F_{r,j}^{-1})$ 是 8×16 矩阵, $(E_{r,j}^{-1}\|F_{r,j}^{-1})\cdot(x\oplus u_{r,j})$ 表示将 16 位向量 $(x\oplus u_{r,j})$ 的两个 8 位分量分别经过 $E_{r,j}^{-1}$ 与 $F_{r,j}^{-1}$ 解码后再将两解码后的值异或, S 表示 SM4 的 S 盒查表运算, M_j 分别表示第 1.1 节中矩阵 M 的 4 个 32×8 分块.

(4) 输入编码如公式 (12) 所示:

$$X'_i=P_i\cdot X_i\oplus p_i,\ i=0,1,2,3 \tag{12}$$

(5) 加密:

若记 r 为加密循环轮次, 则加密过程包括如下 (a)–(e) 共 5 步:

(a) 查TA表, TB表, TC表, 对输入进行解码与编码:

由X'_{r-1}的 4 个 8 位分量分别查询查找表$TC_{r,0}$, $TC_{r,1}$, $TC_{r,2}$, $TC_{r,3}$, 得$c_{r,0}$, $c_{r,1}$, $c_{r,2}$, $c_{r,3}$.

由X'_r的 4 个 8 位分量分别查询查找表$TA_{r,0}$, $TA_{r,1}$, $TB_{r,0}$, $TB_{r,1}$, 得$a_{r,0}$, $a_{r,1}$, $b_{r,0}$, $b_{r,1}$.

由X'_{r+1}的 4 个 8 位分量分别查询查找表$TA_{r,2}$, $TA_{r,3}$, $TB_{r,2}$, $TB_{r,3}$, 得$a_{r,2}$, $a_{r,3}$, $b_{r,2}$, $b_{r,3}$.

由X'_{r+2}的 4 个 8 位分量分别查询查找表$TA_{r,4}$, $TA_{r,5}$, $TB_{r,4}$, $TB_{r,5}$, 得$a_{r,4}$, $a_{r,5}$, $b_{r,4}$, $b_{r,5}$.

(b) 重组:

计算$a_r=\sum_{i=0}^{5}a_{r,i}$, $b_r=\sum_{i=0}^{5}b_{r,i}$. 记$a_r=a_{r,0}\|a_{r,1}\|a_{r,2}\|a_{r,3}$, $b_r=b_{r,0}\|b_{r,1}\|b_{r,2}\|b_{r,3}$, 将$a_{r,i}$, $b_{r,i}$重组为 4 个 16 位的向量$y_{r,0}$, $y_{r,1}$, $y_{r,2}$, $y_{r,3}$, 其中$y_{r,i}=a_{r,i}\|b_{r,i}$, $i=0, 1, 2, 3$.

(c) 查TD表, 完成主要加密操作:

根据$y_{r,0}$, $y_{r,1}$, $y_{r,2}$, $y_{r,3}$分别查询查找表$TD_{r,0}$, $TD_{r,1}$, $TD_{r,2}$, $TD_{r,3}$, 得$d_{r,0}$, $d_{r,1}$, $d_{r,2}$, $d_{r,3}$.

(d) 获取轮输出:

$$X'_{r+3}=\sum_{i=0}^{3}(c_{r,i}\oplus d_{r,i}) \tag{13}$$

(e) 循环执行步骤 (a)–(d) 32 次, 得到未解码的加密结果.

(6) 输出解码如公式 (14) 所示:

$$X_i=P_i^{-1}(X'_i\oplus p_i),\ i=32,33,34,35 \tag{14}$$

以上加密过程未考虑反序变换.

性能方面: 白-武白盒 SM4 方案的执行速度较快, 但是内存占用较大 (整个方案的查找表尺寸共计 32.5 M). 安全性方面: 白-武白盒 SM4 方案能够抵抗多种已知的攻击方式 (BGE 攻击, MGH 攻击, Mulder 等人的攻击[10], Lepoint-Rivain 攻击, 林-来攻击), 安全性较高. 2018 年, 潘文伦等人对其进行了安全性分析[22], 将白-武白盒 SM4 方案的安全性归结于外部编码仿射常数的安全性, 并指出其密钥和外部编码的取值空间大小为 $61200\cdot 2^{128}$.

1.3 差分计算分析

差分计算分析是在 2016 年由 Bos 等人首次提出的针对白盒实现方案的旁信道分析方法[12], 是差分能量分析的软件对应版本. 进行差分能量分析需要获取密码算法执行时产生的能量迹, 而进行差分计算分析需要获取密码算法执行时产生的软件迹. 一般地, 获取软件迹的步骤如下.

(1) 输入任意一条随机明文, 执行算法程序, 记录程序执行时访问的所有地址与数据, 此即获得了一条软件迹.

(2) 将软件迹可视化, 以此来了解算法程序对内存的使用情况, 通过辨别重复模式的个数, 判断程序实现了哪个密码学原语.

(3) 保持地址空间布局随机化 (address space layout randomization, ASLR) 关闭, 输入多条随机明文, 记录多条软件迹, 可以选择采用一些记录标准.

(4) 将记录到的所有软件迹序列化为 0 和 1 组成的串.

根据文献 [13], 标准差分计算分析的步骤如下.

(1) 获取软件迹.

(2) 选取算法计算过程中的某个密钥相关的中间状态字节, 确定选择函数. 选择函数的输出为中间状态字节的某一位, 称为目标比特. 对每种情况的目标比特, 执行如下步骤 (3)–(5).

(3) 根据目标比特是 0 还是 1, 将软件迹分为两类.

(4) 分别求两类软件迹的均值迹.

(5) 求两条均值迹的差分迹.

(6) 比较各差分迹的峰值, 其中最大者对应的目标比特为最佳目标比特.

(7) 对每种情况的密钥字节猜测, 执行步骤 (2)–(6), 比较各种密钥字节猜测下的最佳目标比特对应的差分迹峰

值, 其中最大者对应的密钥字节猜测为最佳密钥字节猜测.

此处简单阐述差分计算分析的原理. 若白盒密码采用可逆矩阵作为密钥相关的查找表的输出编码, 且含有至少 1 行汉明重为 1 的行, 则会泄露至少 1 位的原始输出. 即, 若 $R[i]$ 表示矩阵 R 的第 i 行, 其汉明重为 1, 其第 j 个元素为 1, 则满足公式 (15):

$$R[i]\cdot p = p[j] \tag{15}$$

其中, $p[j]$ 表示向量 p 的第 j 个元素. 这样, 最终得到的编码结果中至少有 1 位没有经过混淆, 该位对应软件迹中某个样点, 样点值将与该位对应的原始输出值完全一致. 若选择函数恰好采用该位对应的原始输出作为目标比特, 则在密钥字节猜测正确的情况下, 对软件迹进行分类, 其中一类软件迹中该位对应的样点值将均为 0, 另一类软件迹中该位对应的样点值将均为 1. 之后对两类软件迹求均值得到两条均值迹, 其中一条均值迹中该位对应的样点值仍为 0 (多个 0 求均值仍为 0), 另一条均值迹中该位对应的样点值仍为 1 (多个 1 求均值仍为 1). 之后求得的差分迹中该位对应的样点值将为 1, 在图像上会出现一个明显尖峰, 根据这个特征可以将正确的密钥字节猜测辨别出来. 反之, 若密钥字节猜测不正确, 则对软件迹进行分类后, 每一类软件迹中该位对应的样点值都有 50% 的概率可能是 0, 也有 50% 的概率可能是 1, 由此导致之后求得的差分迹的图像较为平缓, 这表明猜测不正确.

若白盒密码采用可逆的仿射变换作为密钥相关的查找表的输出编码, 则相当于将矩阵乘法的结果再异或上一个常数. 如果矩阵满足之前提出的包含汉明重为 1 的行, 则编码后的输出的某一位将是该位对应的原始输出异或一个常数后的结果. 但异或至多只会交换分类后的两类软件迹, 即使交换发生也依然可以判别出正确的密钥字节猜测.

若白盒密码采用了可逆矩阵或可逆的仿射变换作为密钥相关的查找表的输出编码, 但是矩阵不含有汉明重为 1 的行, 则标准差分计算分析失效. 此时需要采用变体差分计算分析, 即将标准差分计算分析的步骤 (2) 中选择函数的输出 (即目标比特) 改为中间状态字节各位的某种线性组合的结果. 这样, 在密钥字节猜测正确的情况下, 至少有 1 种线性组合的结果与软件迹中某样点的值一致, 从而保证最后可以判别出正确的密钥字节猜测. 但这也导致目标比特由 8 种情况增加到 255 种情况, 大大延长了分析时间. 事实上, 若编码所用矩阵足够均匀, 则很多情况下标准差分计算分析失效, 下节将就此问题进行研究.

2 对 GF(2) 上 *n* 阶均匀随机可逆矩阵统计特征的研究

从第 1.3 节可以看出, 标准差分计算分析能否成功, 与用于编码的随机可逆矩阵各行的汉明重有密切关系. 本节围绕白盒实现方案编码时最常用的一种随机可逆矩阵——定义在 GF(2) 的 n 阶均匀随机可逆矩阵, 对其各行的汉明重的统计特征展开理论分析.

记事件 D: n 阶均匀随机矩阵至少有一行汉明重为 1. 事件 E: n 阶均匀随机矩阵是可逆矩阵. 则在事件 E 发生的条件下事件 D 发生的概率如公式 (16) 所示:

$$P(D|E) = \frac{P(DE)}{P(E)} = \frac{N(DE)}{N(E)} \tag{16}$$

其中, $N(E)$ 是 n 阶均匀随机可逆矩阵的个数, 如公式 (17) 所示:

$$N(E) = \prod_{i=0}^{n-1}(2^n - 2^i) \tag{17}$$

$N(DE)$ 是至少有一行汉明重为 1 的 n 阶均匀随机可逆矩阵的个数, 根据容斥原理有公式 (18).

$$N(DE) = \sum_{1\leqslant i\leqslant n} N(F_i) - \sum_{1\leqslant i<j\leqslant n} N(F_iF_j) + \sum_{1\leqslant i<j<k\leqslant n} N(F_iF_jF_k) - \ldots + (-1)^{n-1}N(F_1F_2\ldots F_n) \tag{18}$$

其中, $N(F_i)$ 是第 i 行汉明重为 1 的 n 阶均匀随机可逆矩阵的个数, $N(F_iF_j)$ 是第 i 行和第 j 行汉明重均为 1 的 n 阶均匀随机可逆矩阵的个数, $N(F_iF_jF_k)$ 是第 i, j, k 行汉明重均为 1 的 n 阶均匀随机可逆矩阵的个数, 以此类推.

考虑 $N(F_1)$. 先确定第 1 行, 第 1 行汉明重为 1, 共有 $\mathrm{C}_n^1 = n$ 种情况. 在第 1 行确定的情况下, 第 2 行不能与第 1 行相同, 也不能全为 0, 共有 $2^n - 2$ 种情况. 在前两行确定的情况下, 第 3 行不能是前两行的线性组合, 也不能全

为 0, 共有 2^n-2^2 种情况. 以此类推, 在前 $n-1$ 行确定的情况下, 第 n 行有 2^n-2^{n-1} 种情况. 可得公式 (19):

$$N(F_1)=n\cdot\prod_{i=1}^{n-1}(2^n-2^i) \tag{19}$$

又 $N(F_1)=N(F_2)=N(F_3)=\ldots=N(F_n)$, 可得公式 (20):

$$\sum_{1\leqslant i\leqslant n}N(F_i)=\mathrm{C}_n^1\cdot n\cdot\prod_{i=1}^{n-1}(2^n-2^i) \tag{20}$$

类似地, 可得 $\sum_{1\leqslant i<j\leqslant n}N(F_iF_j)=\mathrm{C}_n^2\cdot n\cdot(n-1)\cdot\prod_{i=2}^{n-1}(2^n-2^i)$, $\sum_{1\leqslant i<j<k\leqslant n}N(F_iF_jF_k)=\mathrm{C}_n^3\cdot n\cdot(n-1)\cdot(n-2)\cdot\prod_{i=3}^{n-1}(2^n-2^i)$, …, $N(F_1F_2\ldots F_n)=n!$. 据此整理公式 (18) 得公式 (21):

$$N(DE)=\sum_{j=1}^{n-1}(-1)^{j-1}\cdot\mathrm{C}_n^j\cdot\mathrm{A}_n^j\cdot\prod_{i=j}^{n-1}(2^n-2^i)+(-1)^{n-1}\cdot n! \tag{21}$$

根据公式 (21) 可求出 $N(DE)$ 的精确值, 根据公式 (17) 可求出 $N(E)$ 的精确值, 将其代入公式 (16) 即可求得 $P(D|E)$ 的精确值. 类似地, 可以求出在 n 阶均匀随机矩阵是可逆矩阵的情况下至少有一行汉明重为 x 的概率. 若记事件 D_x: n 阶均匀随机矩阵至少有一行汉明重为 x, 则可得公式 (22) 与公式 (23):

$$P(D_x|E)=\frac{P(D_xE)}{P(E)}=\frac{N(D_xE)}{N(E)} \tag{22}$$

$$N(D_xE)=\sum_{j=1}^{n-1}(-1)^{j-1}\cdot\mathrm{C}_n^j\cdot\mathrm{A}_{\mathrm{C}_n^x}^j\cdot\prod_{i=j}^{n-1}(2^n-2^i)+(-1)^{n-1}\cdot\mathrm{A}_{\mathrm{C}_n^x}^n \tag{23}$$

根据公式 (17), 公式 (22), 公式 (23), 对定义在 GF(2) 上的 8 阶, 16 阶和 32 阶均匀随机可逆矩阵的某些概率进行计算, 所得理论结果如表 1 所示.

表 1　8 阶, 16 阶和 32 阶矩阵的理论结果

矩阵阶数	至少有一行汉明重为1的概率 (保留7位小数)	至少有一行汉明重为2的概率 (保留7位小数)	至少有一行汉明重为3的概率 (保留7位小数)
8	0.2278958	0.6112003	0.8672067
16	0.0038996	0.0289017	0.1283123
32	0.0000001	0.0000037	0.0000370

根据表 1 中的理论结果, GF(2) 上的 8 阶均匀随机可逆矩阵至少有一行汉明重为 1 的概率为 0.2278958, GF(2) 上的 16 阶均匀随机可逆矩阵至少有一行汉明重为 1 的概率为 0.0038996, GF(2) 上的 32 阶均匀随机可逆矩阵至少有一行汉明重为 1 的概率为 0.000 000 1. 再结合第 1.3 节内容可知, 若编码所用矩阵足够均匀, 则标准差分计算分析不能保证成功, 且 n 越大标准差分计算分析的成功率越低, 当 n 为 32 时标准差分计算分析几乎不可能成功. 而实际应用中, 用于编码的矩阵常常是 GF(2) 上的 8 阶或 8 阶以上的均匀随机可逆矩阵, 这导致很多情况下标准差分计算分析失效, 必须采用第 1.3 节最后一段所述的变体差分计算分析.

3　对白-武白盒 SM4 方案的差分计算分析

根据文献 [21] 中的 SM4 密钥扩展算法, 恢复白-武白盒 SM4 方案的初始密钥, 至少需要恢复连续 4 轮的轮密钥, 每个轮密钥又由 4 个密钥字节组成. 由于差分计算分析恢复各个密钥字节的过程十分相似, 且恢复第 1 轮轮密钥后可用类似的方法依次恢复第 2, 3, 4 轮的轮密钥, 故以下所有实验只讨论如何通过差分计算分析恢复第 1 轮轮密钥的最高有效字节, 对于如何恢复其余的密钥字节不再赘述. 又由于白盒实现方案难以出现使用不均匀矩阵的情况, 故以下所有实验只讨论编码所用矩阵均匀的情况.

对白-武白盒 SM4 方案进行差分计算分析的整个实验过程如下.

(1) 编程实现白-武白盒 SM4 方案.

我们用 Java 程序语言编程实现白-武白盒 SM4 方案作为后续实验的测试对象, 程序运行环境如下: 操作系统为 Windows 10 家庭中文版 (64 位), JDK 版本为 1.8.0_221, 开发平台为 Eclipse, 处理器为 Intel(R) Core(TM) i5-8300H CPU @ 2.30 GHz, 内存 8 GB.

该程序的实现步骤如下.

(a) 依据文献 [21] 所述密钥扩展算法, 根据给定初始密钥生成轮密钥.

(b) 生成随机向量与均匀随机可逆矩阵. 其中, 在生成均匀随机可逆矩阵时, 先生成均匀随机矩阵, 再判断所得矩阵是否可逆, 若不可逆则重新生成, 直到所得矩阵可逆为止.

(c) 依据文献 [22] 所述白-武白盒 SM4 方案的查找表生成过程, 利用步骤 (a) 所得轮密钥和步骤 (b) 所得随机向量与均匀随机可逆矩阵, 生成加密所需查找表.

(d) 随机生成若干条明文.

(e) 依据文献 [22] 所述白-武白盒 SM4 方案的加密过程, 利用步骤 (c) 所得查找表, 对步骤 (d) 所得随机明文进行加密, 获得若干组随机明密文对. 其中, 外部编码与外部解码的过程不写在加密函数内.

(2) 获取软件迹.

如何获取软件迹不在本文的讨论范围内, 故我们假定已经通过某种方式得到了软件迹. 为支持后续实验进行, 可以通过如下方式模拟所需软件迹.

在 (1) 所得程序中插入辅助代码, 将加密时产生的部分中间结果以及相关的随机明密文对分别存储在两个二进制文件中, 在之后的分析中将加密中间结果对应的二进制文件的内容序列化即可模拟所需的软件迹. 又由文献 [12] 可知, 通过可视化的软件迹区分密码算法的轮边界是容易的. 故我们只需存储第一轮加密时的所有查找表的输出作为加密时产生的中间结果.

依据上述方法, 我们随机生成 200 条明文并加密, 得到一个存储着 200 条中间结果记录的二进制文件以及一个存储着 200 组明密文对的二进制文件.

(3) 编程实现差分计算分析算法.

我们用 C++程序语言编程实现差分计算分析算法, 用于对 (2) 所得的两个二进制文件进行分析, 程序运行环境如下: 操作系统为 Windows 10 家庭中文版 (64 位), 开发平台为 Visual Studio 2015, 处理器为 Intel(R) Core(TM) i5-8300H CPU @ 2.30 GHz, 内存 8 GB.

该程序的实现步骤如下.

(a) 读取 (2) 所得的两个二进制文件.

(b) 将存储着 200 条中间结果记录的二进制文件中的内容序列化, 转化为 200 条软件迹.

(c) 计算目标比特.

由第 1.2 节公式 (11) 得公式 (24):

$$TD_{1,0}: y = P_4 \cdot M_0 \cdot S((E_{1,0}^{-1} \| F_{1,0}^{-1}) \cdot (x \oplus u_{1,0}) \oplus rk_{1,0}) \oplus p''_{4,0} \tag{24}$$

$TD_{1,0}$ 是与第 1 轮密钥最高密钥字节相关的查找表. 由第 1.1 节和第 1.2 节内容可知, 若输入编码前的明文为 $(X_0, X_1, X_2, X_3) \in (Z_2^{32})^4$, 则:

$$(E_{1,0}^{-1} \| F_{1,0}^{-1}) \cdot (x \oplus u_{1,0}) = (X_1 \oplus X_2 \oplus X_3)_0 \tag{25}$$

$(X_1 \oplus X_2 \oplus X_3)_0$ 表示 X_1, X_2, X_3 三者异或结果的最高字节. 则:

$$TD_{1,0}: y = P_4 \cdot M_0 \cdot S((X_1 \oplus X_2 \oplus X_3)_0 \oplus rk_{1,0}) \oplus p''_{4,0} \tag{26}$$

令 $y' = S((X_1 \oplus X_2 \oplus X_3)_0 \oplus rk_{1,0})$, $Q = P_4 \cdot M_0$, 则:

$$TD_{1,0}: y = Q \cdot y' \oplus p''_{4,0} \tag{27}$$

若进行标准差分计算分析, 则目标比特是 y' 的某一位, 有 8 种情况; 若进行变体差分计算分析, 则目标比特是 y' 各位的某种线性组合结果, 有 255 种情况. 无论进行哪种分析, 在给定密钥字节猜测 $rk_{1,0}$ 且已知输入编码前的明

文的情况下, 都可计算出各种情况的目标比特.

依据上述方法, 我们分别计算了 200 条随机明文对应的所有密钥字节猜测下的所有情况的目标比特.

(d) 对于每种密钥字节猜测下的每种情况的目标比特, 执行步骤 (e)–(h).

(e) 根据目标比特为 0 还是 1, 将步骤 (b) 所得软件迹分为两类.

(f) 分别求步骤 (e) 所得的两类软件迹的均值迹.

(g) 求步骤 (f) 所得的两条均值迹的差分迹.

(h) 求步骤 (g) 所得的差分迹的峰值.

(i) 求每种密钥字节猜测下的每种情况的差分迹峰值的最大值, 该最大值所对应的密钥字节猜测即为最佳密钥字节猜测.

(4) 判断分析是否成功.

根据 (3) 中程序输出的最佳密钥字节猜测, 判断其是否与 (1) 中给定初始密钥所对应的第一轮轮密钥的最高有效字节相同, 若相同则分析成功, 否则分析失败.

根据上述实验过程, 对白-武白盒 SM4 方案进行标准差分计算分析.

给定初始密钥为: 0x12345678, 0xabababab, 0xcdcdcdcd, 0xffffffff. 此时第一轮轮密钥为: 0xf0b96271. 最高有效字节为 0xf0. 对 10 种不同编码的方案分别采用 200 条中间结果记录及其对应的 200 条随机明文进行标准差分计算分析, 其中只有 5 种方案被成功分析. 假设恢复每个密钥字节的成功率均为 0.5, 则恢复整个初始密钥的总成功率仅为 0.000 015. 若采用更少的样本进行分析, 成功率只会更低. 显然, 标准差分计算分析对编码所用矩阵均匀的白-武白盒 SM4 方案效果不佳, 必须采用变体差分计算分析.

根据上述实验过程, 对白-武白盒 SM4 方案进行变体差分计算分析.

(1) 固定编码所用随机矩阵和随机向量. 对 10 种不同初始密钥的方案采用数量不等的中间结果记录及随机明文进行变体差分计算分析, 并统计采用各种数量的样本进行分析时的正确率, 结果如表 2 所示.

表 2 变体差分计算分析的正确率

样本数目	10	20	23	30	100	200
分析正确率 (%)	0	0	50	100	100	100

(2) 固定初始密钥为: 0x12345678, 0xabababab, 0xcdcdcdcd, 0xffffffff. 此时第一轮轮密钥为: 0xf0b96271. 最高有效字节为 0xf0. 对 10 种不同编码的方案分别采用 200 条中间结果记录及其对应的 200 条随机明文进行变体差分计算分析, 均能得出正确的结果 (f0). 统计分析各种编码的方案时所用的时间, 结果如表 3 所示.

表 3 变体差分计算分析所需时间 (ms)

所用编码	编码1	编码2	编码3	编码4	编码5	编码6	编码7	编码8	编码9	编码10
所需时间	30667	30676	30820	30676	30435	31002	30883	30502	30567	30930

可见, 只要采用足量的软件迹进行分析, 变体差分计算分析是一定可以成功恢复密钥的, 白-武白盒 SM4 方案不能抵抗变体差分计算分析. 由表 3 可知, 变体差分计算分析恢复第一轮轮密钥最高有效字节所需时间在半分钟左右, 由此可以估计该分析恢复整个初始密钥大约需要 8 min.

图 1 与图 2 分别展示了上述变体差分计算分析 (2) 中编码 1 方案在两种情况下的差分迹图像. 当密钥字节猜测正确 (f0) 时, 根据第 1.3 节理论, 此时可由最佳目标比特对软件迹进行正确分类, 这导致部分与选择函数相关的样点值在一类软件迹中恒为 0 (或 1), 而在另一类软件迹中恒为 1 (或 0). 之后求得两条均值迹, 其中一条均值迹中这些样点的值为 0 (或 1), 另一条均值迹中这些样点的值为 1 (或 0). 然后根据均值迹求得的差分迹中, 这些样点的值总是 1, 而其他与选择函数不相关的样点值远小于 1, 这导致此时的差分迹图像出现了一个明显的尖峰. 当密钥字节猜测错误 (以 c8 为例) 时, 根据第 1.3 节理论, 此时不能由最佳目标比特对软件迹进行正确分类, 这导致分类

后的两类软件迹中的所有样点值为 0 或 1 的概率较为接近, 最终导致差分迹中所有样点的值都远小于 1, 反映在图像上就是较为平缓的折线. 根据差分迹的区别, 可以很容易地辨别出正确的密钥字节猜测.

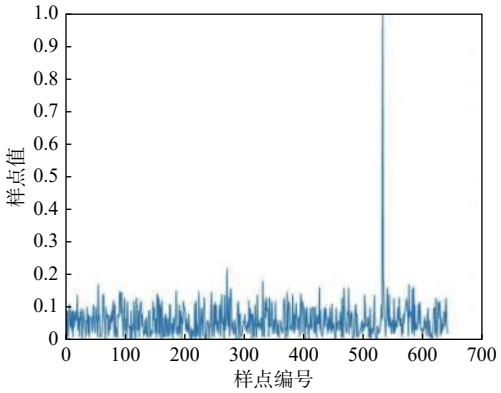

图 1 密钥字节猜测正确 (f0) 时最佳目标比特对应的差分迹

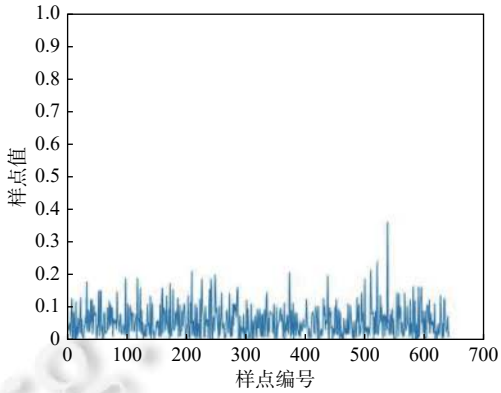

图 2 密钥字节猜测错误 (c8) 时最佳目标比特对应的差分迹

4 对白-武白盒 SM4 方案的改进型差分计算分析

文献 [13] 中提到, 由于列混合操作造成的潜在信息泄露, 实际的差分计算分析并不一定需要遍历所有线性组合情况, 只遍历其中的一部分情况也有很高的概率成功恢复单个密钥字节. 本节基于该思想, 提出一种改进型差分计算分析 (improved differential computation analysis, IDCA).

4.1 改进思路

考虑公式 (27). $Q = P_4 \cdot M_0$, Q 是 32×8 的矩阵, P_4 是 GF(2) 上的 32 阶均匀随机可逆矩阵, M_0 是 GF(2) 上的 32×8 的矩阵, 是第 1.1 节中矩阵 M 的前 8 列. 观察可知, M_0 没有任何一列的元素全为 0. 有如下定理 1.

定理 1. 设 G 是定义在 GF(2) 上的 r 阶均匀随机矩阵, H 是定义在 GF(2) 上的各列不全为 0 的 $r \times s$ 矩阵. 若矩阵 $J = G \cdot H$, 则 J 是均匀的.

证明: 因为 H 的各列不全为 0, 所以 J 中任意元素 $a_{i,j}$ 都可表示为公式 (28).

$$a_{i,j} = c_1 \oplus c_2 \oplus \ldots \oplus c_t \tag{28}$$

其中, $a_{i,j}$ 表示 J 第 i 行第 j 列的元素, c_1, c_2, …, c_t 都是矩阵 G 第 i 行的某个元素, t 是矩阵 H 第 j 列中元素 1 的个数, $1 \leqslant t \leqslant r$. 由于 G 是 GF(2) 上的均匀随机矩阵, 故 c_1, c_2, …, c_t 为 0 或 1 的概率均为 50%. 要证 J 是均匀的, 即证 $a_{i,j}$ 为 0 或 1 的概率均为 50%.

当 $t = 1$ 时, 结论是显然的.

当 $t = 2$ 时, $a_{i,j}$ 为 0 的概率为 0.5×0.5+0.5×0.5=0.5, 为 1 的概率也为 0.5.

当 $3 \leqslant t \leqslant r$ 时, 可用数学归纳法, 先证明以下两点.

(1) 若 $c' = c_1 \oplus c_2$, 则 c' 为 0 或 1 的概率均为 50%.

(2) 若 $c'' = c_1 \oplus c_2 \oplus \ldots \oplus c_{v-1}$, $c''' = c'' \oplus c_v$, $3 \leqslant v \leqslant t$, 且 c'' 为 0 或 1 的概率均为 50%, 则 c''' 为 0 或 1 的概率均为 50%.

对于 (1), c' 为 0 的概率为 0.5×0.5+0.5×0.5=0.5, 为 1 的概率也为 0.5, (1) 得证. 对于 (2), 由于 c'' 为 0 或 1 的概率均为 50%, c_v 为 0 或 1 的概率均为 50%, 故 c''' 为 0 的概率为 0.5×0.5+0.5×0.5=0.5, 为 1 的概率也为 0.5, (2) 得证. 故 $3 \leqslant t \leqslant r$ 时 $a_{i,j}$ 为 0 或 1 的概率均为 50%.

综上, 定理 1 得证, 证毕.

我们可以计算 GF(2) 上的 32×8 的均匀随机矩阵某一行汉明重为 1 的概率为 $8 \times 0.5 \times 0.5^7 = 0.03125$, 某一行汉明

重不为 1 的概率为 1–0.03125=0.96875, 所有行汉明重不为 1 的概率为 0.96875^{32}=0.36205529, 至少有一行汉明重为 1 的概率为 1–0.36205529≈0.6379447. 同理, 至少有一行汉明重为 2 的概率为 0.975 439 6, 至少有一行汉明重为 3 的概率为 0.999 629 1, 至少有一行汉明重为 4 的概率为 0.999 963 6. 根据定理 1, 假若 P_4 是 GF(2) 上任选的 32 阶均匀随机矩阵, 则此时的矩阵 Q 是均匀的. 而事实上, P_4 是 GF(2) 上的 32 阶均匀随机可逆矩阵, 并不是任选的, 此时的矩阵 Q 未必是均匀的, 因此还需进一步的计算机实验来对 Q 的性质进行分析.

我们用 Java 程序语言编写程序, 先生成 32 阶均匀随机矩阵, 再判断所得矩阵是否可逆, 若不可逆则重新生成, 直到所得矩阵可逆为止, 由此得到 GF(2) 上的 32 阶均匀随机可逆矩阵 P_4 后, 再由程序计算 $Q = P_4 \cdot M_0$, 并统计 Q 的各行汉明重的统计特征. 测试环境如下: 操作系统为 Windows 10 家庭中文版 (64 位), JDK 版本为 1.8.0_221, 开发平台为 Eclipse, 处理器为 Intel(R) Core(TM) i5-8300H CPU @ 2.30 GHz, 内存 8 GB.

当样本容量为 10 000 000 时, 至少有一行汉明重为 1 的矩阵的出现频率为 0.637 821 0, 至少有一行汉明重为 2 的矩阵的出现频率为 0.975 445 9, 至少有一行汉明重为 3 的矩阵的出现频率为 0.999 645 3, 至少有一行汉明重为 4 的矩阵的出现频率为 0.999 964 0. 可见, Q 各行汉明重的统计特征非常符合 GF(2) 上的 32×8 均匀随机矩阵的理论值. 可以猜测, 只要方案编码所用矩阵是均匀的, 那么矩阵 Q 至少有一行汉明重为 1 的概率大于 63%, 至少有一行汉明重为 2 的概率大于 97%, 至少有一行汉明重为 3 的概率大于 99.9%, 至少有一行汉明重为 4 的概率大于 99.99%. 由此, 我们得出一个提高差分计算分析的效率的改进思路: 有选择地缩小遍历空间, 在保证分析成功率几乎不变的前提下极大地缩短分析时长.

第 3 节中, 变体差分计算分析的目标比特为 y' 各位的某种线性组合的结果, 共有 255 种情况. 若记目标比特为 b, 则可得公式 (29):

$$b = C \cdot y' \tag{29}$$

其中, C 是定义在 GF(2) 上的非零 8 维行向量, y' 是定义在 GF(2) 上的 8 维列向量. 则 C 有 255 种情况, 其中有 8 种情况汉明重为 1, 28 种情况汉明重为 2, 56 种情况汉明重为 3, 70 种情况汉明重为 4. 若只遍历 C 的汉明重为 1 的情况, 则需要 Q 中有至少一行汉明重为 1 方可保证分析成功, 这个概率在 63% 以上. 故只遍历 C 的汉明重为 1 的情况时, 恢复单个密钥字节的成功率在 63% 以上, 恢复整个初始密钥的总成功率在 0.06% 以上. 同理, 只遍历 C 的汉明重为 2 的情况时, 恢复单个密钥字节的成功率在 97% 以上, 恢复整个初始密钥的总成功率在 61.4% 以上. 只遍历 C 的汉明重为 3 的情况时, 恢复单个密钥字节的成功率在 99.9% 以上, 恢复整个初始密钥的总成功率在 98.4% 以上. 只遍历 C 的汉明重为 4 的情况时, 恢复单个密钥字节的成功率在 99.99% 以上, 恢复整个初始密钥的总成功率在 99.8% 以上. 可见, 即使只遍历 C 的汉明重为 3 的情况来进行差分计算分析, 也有很高的成功率能够恢复整个初始密钥. 若需要更高的成功率, 可以选择只遍历 C 的汉明重为 4 的情况来进行差分计算分析.

由第 3 节图 1 与图 2 可知, 密钥字节猜测正确时差分迹图像会出现显著的尖峰, 而猜测错误时差分迹图像较为平缓. 事实上, 若保证分析时采用的软件迹达到一定数量 (可能不需要很多), 则在密钥字节猜测错误时差分迹图像会表现得相当平缓, 差分迹峰值会明显小于猜测正确时的差分迹峰值. 由此, 我们得出第 2 个提高差分计算分析的效率的改进思路: 直接判断每个差分迹的峰值是否大于某个门限, 若大于该门限则直接得出最佳密钥字节猜测. 根据第 1.3 节差分计算分析的原理, 密钥字节猜测正确时差分迹峰值总是等于 1 的; 而密钥字节猜测错误时差分迹峰值将小于等于 1, 分析时采用的软件迹越多则差分迹峰值越小. 我们可以根据实际情况确定合适的判决门限, 该门限通常略小于 1.

综合上述两条改进思路, 我们得出 IDCA 的步骤如下.

(1) 获取软件迹.

(2) 选取算法计算过程中的某个密钥相关的中间状态字节, 确定选择函数. 选择函数的输出 (即目标比特) 为中间状态字节各位的某种线性组合结果, 且根据需要只筛选部分情况的线性组合进行分析.

(3) 对步骤 (2) 筛选出的每种情况, 执行如下步骤 (4)–(8).

(4) 对每种情况的密钥字节猜测, 执行步骤 (5)–(8).

(5) 根据目标比特是 0 还是 1, 将软件迹分为两类.

(6) 分别求两类软件迹的均值迹.

(7) 求两条均值迹的差分迹.

(8) 判断差分迹峰值是否大于判决门限, 若大于判决门限则此时的密钥字节猜测就是最佳密钥字节猜测, 分析结束.

4.2 分析结果

与第 3 节中的实验过程类似地, 我们对白-武白盒 SM4 方案进行 IDCA. 在分析中我们只遍历 C 的汉明重为 3 的情况, 选取的判决门限为 0.9.

(1) 固定编码所用随机矩阵和随机向量. 对 10 种不同初始密钥的方案采用数量不等的中间结果记录及随机明文进行 IDCA, 并统计采用各种数量的样本进行分析时的正确率, 结果如表 4 所示.

表 4 IDCA 的正确率

样本数目	10	20	25	30	100	200
分析正确率 (%)	0	0	60	100	100	100

(2) 固定初始密钥为: 0x12345678, 0xabababab, 0xcdcdcdcd, 0xffffffff. 此时第一轮轮密钥为: 0xf0b96271. 最高有效字节为 0xf0. 对 10 种不同编码的方案分别采用 200 条中间结果记录及其对应的 200 条随机明文进行 IDCA, 均能得出正确的结果 (f0), 与变体差分计算分析所用时间的对比如表 5 所示.

表 5 IDCA 与变体差分计算分析所需时间的对比 (ms)

分析方法	编码1	编码2	编码3	编码4	编码5	编码6	编码7	编码8	编码9	编码10
IDCA	2 046	301	1 095	532	132	140	1 950	364	805	161
变体差分计算分析	30 667	30 676	30 820	30 676	30 435	31 002	30 883	30 502	30 567	30 930

由表 4 与表 5 可见, 与第 3 节中的变体差分计算分析相比, IDCA 能够极大地缩短分析时长, 且对于样本的数量需求没有明显增加, 分析的成功率几乎保持不变 (理论上, 在分析中只遍历 C 的汉明重为 3 的情况时, 恢复整个初始密钥的总成功率在 98.4% 以上, 与变体差分计算分析相比只降低了不到两个百分点). 实际应用中编码所用矩阵通常是均匀的, IDCA 对于类似的白盒实现方案同样有较高的分析效率. 设选取的中间状态字节与定义在 GF(2) 上的 $r\times s$ 矩阵 R 相乘, 且在分析中只遍历 C 的汉明重为 w 的情况, $w=1,2,\ldots,s/2$. 则 w 越大, 成功概率越高. r 与 s 相差越大, 则列混合操作造成的潜在信息泄露越严重, 成功概率的下限越高.

5 总 结

差分计算分析只需要知道少量白盒实现方案的内部细节, 因此是一种对现有白盒实现方案具有实际威胁的攻击手段. 检验现有白盒实现方案抗差分计算分析的能力对于确保方案在实际应用场景下的安全性具有重要意义. 本文基于我们对 GF(2) 上的 n 阶均匀随机可逆矩阵统计特征的研究结果, 对白-武白盒 SM4 方案进行了差分计算分析, 并在此基础上提出了改进型差分计算分析 (IDCA), 能够在分析成功率几乎不变的前提下很大程度提高分析效率, 是一种行之有效的对白盒实现方案的分析手段. 显然, 在面对差分计算分析时白-武白盒 SM4 方案不能保证安全性, 有待在未来对其进行进一步改进使之满足实际应用场景下的安全性要求.

References:

[1] Chow S, Eisen P, Johnson H, van Orschot PC. White-box cryptography and an AES implementation. In: Nyberg K, Heys H, eds. Proc. of the Int'l Workshop on Selected Areas in Cryptography. Berlin, Heidelberg: Springer, 2003. 250–270. [doi: 10.1007/3-540-36492-7_17]

[2] Chow S, Eisen P, Johnson H, van Oorschot PC. A white-box DES implementation for DRM applications. In: Feigenbaum J, ed. Proc. of the ACM Workshop on Digital Rights Management. Berlin, Heidelberg: Springer, 2003. 1–15. [doi: 10.1007/978-3-540-44993-5_1]

[3] Xiao YY. White-box cryptography and implementations of AES and SMS4 [MS. Thesis]. Shanghai: Shanghai Jiao Tong University, 2010 (in Chinese with English abstract).

[4] Luo R, Lai XJ, You R. A new attempt of white-box AES implementation. In: Proc. of the 2014 IEEE Int'l Conf. on Security, Pattern Analysis, and Cybernetics (SPAC). Wuhan: IEEE, 2014. 423–429. [doi: 10.1109/spac.2014.6982727]

[5] Shi Y, Wei WJ, He ZJ. A lightweight white-box symmetric encryption algorithm against node capture for WSNs. Sensors, 2015, 15(5): 11928–11952. [doi: 10.3390/s150511928]

[6] Bai KP, Wu CK. A secure white-box SM4 implementation. Security and Communication Networks, 2016, 9(10): 996–1006. [doi: 10.1002/sec.1394]

[7] Bai KP, Wu CK, Zhang ZF. Protect white-box AES to resist table composition attacks. IET Information Security, 2018, 12(4): 305–313. [doi: 10.1049/iet-ifs.2017.0046]

[8] Billet O, Gilbert H, Ech-Chatbi C. Cryptanalysis of a white box AES implementation. In: Handschuh H, Hasan MA, eds. Proc. of the Int'l Workshop on Selected Areas in Cryptography. Berlin, Heidelberg: Springer, 2005. 227–240. [doi: 10.1007/978-3-540-30564-4_16]

[9] Michiels W, Gorissen P, Hollmann HDL. Cryptanalysis of a generic class of white-box implementations. In: Avanzi RM, Keliher L, Sica F, eds. Proc. of the 2009 Int'l Workshop on Selected Areas in Cryptography. Berlin, Heidelberg: Springer, 2009. 414–428. [doi: 10.1007/978-3-642-04159-4_27]

[10] De Mulder Y, Roelse P, Preneel B. Cryptanalysis of the Xiao-lai white-box AES implementation. In: Knudsen LR, Wu HP, eds. Proc. of the 2013 Int'l Workshop on Selected Areas in Cryptography. Berlin, Heidelberg: Springer, 2013. 34–49. [doi: 10.1007/978-3-642-35999-6_3]

[11] Lin TT, Lai XJ. Research on white-box cryptography. Journal of Cryptologic Research, 2015, 2(3): 258–267 (in Chinese with English abstract). [doi: 10.13868/j.cnki.jcr.000077]

[12] Bos JW, Hubain C, Michiels W, Teuwen P. Differential computation analysis: Hiding your white-box designs is not enough. In: Gierlichs B, Poschmann AY, eds. Proc. of the Int'l Conf. on Cryptographic Hardware and Embedded Systems. Berlin, Heidelberg: Springer, 2016. 215–236. [doi: 10.1007/978-3-662-53140-2_11]

[13] Bock EA, Bos JW, Brzuska C, Hubain C, Michiels W, Mune C, Gonzalez ES, Teuwen P, Treff A. White-box cryptography: Don't forget about grey-box attacks. Journal of Cryptology, 2019, 32(4): 1095–1143. [doi: 10.1007/s00145-019-09315-1]

[14] Bock EA, Brzuska C, Michiels W, Treff A. On the ineffectiveness of internal encodings—Revisiting the DCA attack on white-box cryptography. In: Preneel B, Vercauteren F, eds. Proc. of the Int'l Conf. on Applied Cryptography and Network Security. Cham: Springer, 2018. 103–120. [doi: 10.1007/978-3-319-93387-0_6]

[15] Breunesse CB, Kizhvatov I, Muijrers R, Spruyt A. Towards fully automated analysis of whiteboxes: Perfect dimensionality reduction for perfect leakage. Cryptology ePrint Archive, Report 2018/095, 2018.

[16] Banik S, Bogdanov A, Isobe T, Jepsen M. Analysis of software countermeasures for whitebox encryption. IACR Trans. on Symmetric Cryptology, 2017, 3(8): 307–328. [doi: 10.13154/tosc.v2017.i1.307-328]

[17] Biryukov A, Udovenko A. Attacks and countermeasures for white-box designs. In: Peyri T, Galbraith S, eds. Proc. of the Int'l Conf. on the Theory and Application of Cryptology and Information Security. Cham: Springer, 2018. 373–402. [doi: 10.1007/978-3-030-03329-3_13]

[18] Bogdanov A, Rivain M, Vejre PS, Wang JW. Higher-order DCA against standard side-channel countermeasures. In: Polian I, Stöttinger M, eds. Proc. of the Int'l Workshop on Constructive Side-channel Analysis and Secure Design. Cham: Springer, 2019. 118–141. [doi: 10.1007/978-3-030-16350-1_8]

[19] Lee S, Kim M. Improvement on a masked white-box cryptographic implementation. IEEE Access, 2020, 8: 90992–91004. [doi: 10.1109/access.2020.2993651]

[20] Biryukov A, Udovenko A. Dummy shuffling against algebraic attacks in white-box implementations. In: Canteaut A, Standaert FX, eds. Proc. of the Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Cham: Springer, 2021. 219–248. [doi: 10.1007/978-3-030-77886-6_8]

[21] General Administration of Quality Supervision, Inspection and Quarantine of the People's Republic of China, Standardization Administration. GB/T 32907-2016 Information security technology—SM4 block cipher algorithm. Beijing: China Standards Press, 2017 (in Chinese with English abstract).

[22] Pan WL, Qin TH, Jia Y, Zhang LT. Cryptanalysis of two white-box SM4 implementations. Journal of Cryptologic Research, 2018, 5(6): 651–671 (in Chinese with English abstract). [doi: 10.13868/j.cnki.jcr.000274]

附中文参考文献:

[3] 肖雅莹. 白盒密码及AES与SMS4算法的实现 [硕士学位论文]. 上海: 上海交通大学, 2010.

[11] 林婷婷, 来学嘉. 白盒密码研究. 密码学报, 2015, 2(3): 258–267. [doi: 10.13868/j.cnki.jcr.000077]
[21] 中华人民共和国国家质量监督检验检疫总局, 中国国家标准化管理委员会. GB/T 32907-2016 信息安全技术SM4分组密码算法. 北京: 中国标准出版社, 2017.
[22] 潘文伦, 秦体红, 贾音, 张立廷. 对两个SM4白盒方案的分析. 密码学报, 2018, 5(6): 651–671. [doi: 10.13868/j.cnki.jcr.000274]

原梓清(1996－), 男, 硕士生, 主要研究领域为白盒密码的设计与安全性分析.

陈杰(1979－), 女, 博士, 副教授, 主要研究领域为密码算法的设计与安全性分析.