uBlock算法的低延迟一阶门限实现方法

姚 富[1,2],陈 华[1*],范丽敏[1]
(1. 中国科学院软件研究所可信计算与信息保障实验室,北京 100190;2. 中国科学院大学,北京 100049)

摘 要: 目前已有文献给出了uBlock分组密码算法的侧信道防护方案,但是这些方案不仅延迟较高,难以适用于低延迟高吞吐场景,而且在毛刺探测模型下缺乏可证明安全性. 针对这一问题,本文给出了在毛刺探测模型下具有可证明安全性的uBlock算法的低延迟门限实现方案. 此外,我们引入了Changing of the Guards技术来避免防护方案在执行过程中需要额外随机数. 对于防护方案的安全性,我们用自动化评估工具SILVER验证了S盒的毛刺探测安全性,并用泄露评估技术TVLA(Test Vector Leakage Assessment)验证了防护方案的整个电路的安全性. 最后,我们用Design Compiler工具对防护方案的性能消耗情况进行了评估. 评估结果显示,与序列化实现方式的uBlock防护方案相比,我们的防护方案的延迟能够减少约95%.

关键词: 密码芯片;uBlock算法;侧信道攻击与防护;掩码技术;门限实现;Changing of the Guards
基金项目: 国家自然科学基金(No.62172395)
中图分类号: TN918;TP309 **文献标识码:** A **文章编号:** 0372-2112(2024)04-1250-10
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20231031

Low-Latency First-Order Threshold Implementation of uBlock

YAO Fu[1,2], CHEN Hua[1*], FAN Li-min[1]
(1. *Trusted Computing and Information Assurance Laboratory*, *Institute of Software*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*;
2. *University of Chinese Academy of Sciences*, *Beijing* 100049, *China*)

Abstract: The existing side-channel protection schemes for the uBlock algorithm suffer from high latency, making them unsuitable for low-latency and high-throughput scenarios. Additionally, these schemes lack provable security under the glitch-extended probing model. To address these issues, this paper presents a low-latency Threshold Implementation of the uBlock algorithm with provable security under the glitch-extended probing model. Furthermore, we introduce the Changing of the Guards technique to eliminate the need for additional random numbers during the execution of the protection scheme. To validate the security of our protection scheme, we employ the automated evaluation tool SILVER to assess the glitch-extended probing security of the S-box and utilize the leakage evaluation technology TVLA (Test Vector Leakage Assessment) to verify the security of the entire circuit. Finally, we evaluate the performance overhead of our protection scheme using the design compiler tool. The evaluation results demonstrate that our scheme achieves a significant reduction in latency, approximately 95% less compared to serialized implementations of uBlock protection schemes.

Key words: cryptographic chip; uBlock algorithm; side-channel attack and protection; masking; threshold implementation; changing of the guards
Foundation Item(s): National Natural Science Foundation of China (No.62172395)

1 引言

密码芯片在实际运行环境中,不仅受到黑盒模型下的攻击[1,2],还容易遭受灰盒模型[3]和白盒模型[4]下的攻击. 在灰盒模型下,攻击者具有获取密码芯片执行过程中的侧信息[3,5,6]以及主动干扰密码芯片执行过程的能力[7]. 其中,攻击者利用密码芯片的物理特征来恢复其内部敏感数据的攻击方法被称为旁路攻击,也称侧信道攻击. 侧信道攻击不需要了解被攻击算法的结构,具有很强的可操作性和可复现性,因此侧信道攻击对于密码芯片的安全实现造成了严重威胁.

针对侧信道攻击,算法层的防护方法有隐藏技术

收稿日期:2023-11-01;修回日期:2024-01-21;责任编辑:覃怀银
*通讯作者:陈华

和掩码技术[8]. 掩码技术的核心思想是利用随机数通过布尔运算将密码算法的中间敏感值随机化为若干个掩码分量,使得密码芯片产生的侧信息表现为掩码分量的侧信息. 由于掩码技术具有可证明安全性和良好的扩展性[8],因此其成为了算法级主流的防护方法之一.

由于早期的掩码方案[9]均基于一个理想假设,即中间运算要么完全执行,要么完全不执行,因此这些方案能够安全地应用在软件实现中. 大多数硬件电路是由CMOS元件构成的,很容易产生毛刺现象,使得早期的掩码方案不再适用于硬件实现[10]. 为了解决这个问题,Nikova等人提出了门限实现[11]. 在门限实现中,中间运算电路被拆分成一些运算子电路,在理论上保证了其在硬件实现中的安全性. 由于门限实现具有可证明安全性、同时抵抗侧信道攻击和毛刺攻击等优点,因此被广泛应用在AES[12],PRESENT[13]、PRINCE[14]等分组密码算法上.

uBlock算法[15]是由中国科学院软件所吴文玲团队设计的分组密码算法,具有安全性高、可扩展性好、适应性强等特点. uBlock算法的非线性运算是一个代数次数为3次的4 bit S盒,适用于设计门限实现防护方案. 文献[16]指出该S盒存在3-share且无需额外随机数的一阶门限实现方案. 在此基础上,文献[17]对uBlock算法设计了3-share且无需额外随机数的一阶门限实现方案和2-share且无需额外随机数的一阶门限实现方案. 然而,文献[17]中给出的2-share的门限实现方案仅在探测模型下具有可证明安全性,无法应用在硬件实现环境中. 此外,该文献中给出的基于串行和流水线思想的uBlock算法低代价实现方案完成一个分组的加密至少需要656个时钟周期,难以应用在低延迟应用场景. 综上所述,如何给出能适用于硬件环境的uBlock算法一阶门限实现方案,使其能够应用于低延迟场景是一个有待解决的关键问题. 针对这一问题,我们给出了uBlock算法的低延迟一阶门限实现方案. 其中,方案的硬件实现代码,SILVER评估脚本以及面积测试结果均上传到GitHub(https://github.com/GitHub-lancel/uBlockTIScheme).

2　基础知识

2.1　uBlock算法

uBlock算法整体采用SPN结构,分组长度为128或256 bit,密钥长度为128或256 bit,因此有三个算法版本,分别记为uBlock-128/128,uBlock-128/256,uBlock-256/256,对应的迭代轮数分别是16、24、24轮. 在本文中,我们只考虑uBlock-128/128加密算法的防护方案,其他版本的防护方案只需要在uBlock-128/128加密算法的防护方案基础上,稍加修改即可.

uBlock-128/128加密算法的分组长度是128 bit,密钥长度是128 bit,迭代轮数是16轮. 该算法的轮函数包含轮密钥异或,S盒替换,移位操作和向量置换这四种运算. 设第i个轮函数的输入为$\boldsymbol{p}_i=\left(p_i^0,p_i^1\right)$,轮密钥为$\boldsymbol{k}_i=\left(k_i^0,k_i^1\right)$,其中$p_i^j$和$k_i^j$的大小均为64 bit. 下面给出轮函数每个运算的详细介绍:

(1)轮密钥异或:轮函数的输入$\boldsymbol{p}_i=\left(p_i^0,p_i^1\right)$与轮密钥$\boldsymbol{k}_i=\left(k_i^0,k_i^1\right)$进行比特异或运算,具体表达式为$\left(a_i^0,a_i^1\right)=\left(p_i^0,p_i^1\right)\oplus\left(k_i^0,k_i^1\right)$.

(2)S盒替换:该运算是轮函数中的唯一非线性运算,其功能是将轮密钥异或运算的输出$\boldsymbol{a}_i=\left(a_i^0,a_i^1\right)$的每个半字节通过S盒查表运算,得到其输出$\boldsymbol{b}_i=\left(b_i^0,b_i^1\right)$,表达式为$\boldsymbol{b}_i=\left(b_i^0,b_i^1\right)=\left(S_{16}(a_i^0),S_{16}(a_i^1)\right)$,其中$S_{16}$表示16个S盒并行运算,每个S盒如表1所示.

表1　uBlock算法的S盒

x	0	1	2	3	4	5	6	7
$S(x)$	7	4	9	c	b	a	d	8
x	8	9	a	b	c	d	e	f
$S(x)$	f	e	1	6	0	3	2	5

(3)移位操作:首先执行一个异或运算$b_i^1=b_i^0\oplus b_i^1$,然后依次执行4个以32 bit为单位的循环左移操作:$b_i^0=b_i^0\oplus\left(b_i^1<<<4\right)$,$b_i^1=b_i^1\oplus\left(b_i^0<<<8\right)$,$b_i^0=b_i^0\oplus\left(b_i^1<<<8\right)$,$c_i^1=b_i^1\oplus\left(b_i^0<<<8\right)$,最后再执行一个异或运算$c_i^1=b_i^0\oplus c_i^1$.

(4)向量置换:对移位操作运算输出$\left(c_i^0,c_i^1\right)$的每个字节内部进行置换操作. 设任意的一个字节为$\left(x_0,\cdots,x_7\right)$,那么置换操作的具体表达式为$\left(x_0,\cdots,x_7\right)\rightarrow\left(x_1,x_3,x_4,x_6,x_0,x_2,x_7,x_5\right)$.

2.2　毛刺探测模型

Ishai等人提出了第一个简单且抽象的安全性评估模型——探测模型[9]. 在d阶探测模型中,敌手能够同时获取掩码方案所在电路的任何d根线的信息,并根据这d根线的返回值来判断能否恢复掩码方案所使用的密钥. 如果无法恢复密钥,说明该掩码方案具有抗侧信道攻击的能力. 该模型评估掩码方案安全性的前提条件是掩码方案在电路中运行时,每个信号均不会发生意外的翻转,即不会产生毛刺现象. 由于软件实现具有原子性,该模型非常适用于评估掩码方案在软件实现中的安全性.

在由CMOS元器件构成的硬件电路中,不平衡的路径延迟,很容易产生毛刺现象,因此探测模型无法准确地评估掩码方案在这种环境下的安全性[18]. Faust等人考虑到硬件电路中毛刺、耦合和寄存器翻转等物理特

征对掩码方案的安全性影响，提出了探测模型的扩展版本——毛刺探测模型[19]. 在毛刺探测模型中，敌手不仅可以获取探测点的信息，还能够获取到探测点所在组合逻辑的所有输入信号的信息.

为了降低掩码方案的安全性证明的复杂性，Knichel，Barthe对探测模型、毛刺探测等模型提出了自动化评估工具，如SILVER[20]. 我们需要强调的是在本文中，我们主要关注的是uBlock算法在硬件实现中的一阶安全性，因此我们在理论上考虑和分析uBlock算法的掩码方案在毛刺探测模型下的一阶安全性，并使用SILVER来验证我们构造出的掩码方案的一阶毛刺探测安全性.

2.3 门限实现

设输入为$\boldsymbol{x}=(x_0,\cdots,x_{n-1})$，输出为$\boldsymbol{y}=(y_0,\cdots,y_{m-1})$，输入掩码分量为$\bar{\boldsymbol{x}}=((x_0^0,\cdots,x_0^{S_{\text{in}}-1}),\cdots,(x_{n-1}^0,\cdots,x_{n-1}^{S_{\text{in}}-1}))$，输出掩码分量为$\bar{\boldsymbol{y}}=((y_0^0,\cdots,y_0^{S_{\text{out}}-1}),\cdots,(y_{m-1}^0,\cdots,y_{m-1}^{S_{\text{out}}-1}))$，其中，$S_{\text{in}}$和$S_{\text{out}}$分别表示输入和输出掩码分量的个数，$x_i,y_i,x_i^j,y_i^j\in\text{GF}(2)$. 设目标函数$F(\bullet):F_2^n\to F_2^m$，那么它的门限实现方案为$\bar{\boldsymbol{F}}(\bullet):F_2^{nS_{\text{in}}}\to F_2^{mS_{\text{out}}}$，其中每个坐标函数$f_i$的门限实现表达式是由$S_{\text{out}}$个掩码分量函数$(f_i^0,\cdots,f_i^{S_{\text{out}}-1})$构成. 门限实现方案$\bar{\boldsymbol{F}}(\bullet)$需要满足正确性、$d$阶不完整性和(输入/输出)均匀性，其中正确性和(输入/输出)均匀性保证了门限实现方案具有抗侧信道攻击的能力，d阶不完整性保证了门限实现方案具有抗毛刺攻击的能力.

定义1(正确性) 对于每个二元输入(输出)变量$x_i(y_i)$，所有掩码分量的异或运算结果应是变量$x_i(y_i)$本身，即$\oplus_{j=0}^{S_{\text{in}}-1}x_i^j=x_i(\oplus_{j=0}^{S_{\text{out}}-1}y_i^j=y_i)$.

定义2(d阶不完整性[11]) 结合每个分量函数f_i的任何d个掩码分量表达式f_i^j都至少与一个输入掩码分量相互独立.

定义3(均匀性) 对于每个输入变量$\boldsymbol{x}$和输出变量$\boldsymbol{y}$，如果每个掩码分量都以相同的概率产生，那么该掩码过程是一个均匀的掩码过程.

2.4 Changing of the Guards

假设在一个加密算法的轮函数中，同时执行m个相同S盒运算. 第i个S盒的输入掩码分量为(x_i^0,x_i^1,x_i^2)，输出掩码分量为(y_i^0,y_i^1,y_i^2)，S盒的一个3-share非均匀的掩码方案为(S^0,S^1,S^2)，其中S^j的输入为$x_i^{(j+1)\bmod 3}$和$x_i^{(j+2)\bmod 3}$，输出为y_i^j，那么第i个S盒的第j个掩码分量函数的表达式为$y_i^j=S^j(x_i^{(j+1)\bmod 3},x_i^{(j+2)\bmod 3})$.

Changing of the Guards技术在这m个并行运算的S盒上的应用：在算法运行时，第i个S盒的3个输出掩码分量与前一个S盒的2个输出掩码分量x_{i-1}^1和x_{i-1}^2进行异或运算，确保该S盒的输出掩码分量满足均匀性，具体表达式如下所示：

$$\begin{cases} y_i^0=S^0(x_i^1,x_i^2)+x_{i-1}^1+x_{i-1}^2, & i>0\\ y_i^1=S^0(x_i^2,x_i^0)+x_{i-1}^2, & i>0\\ y_i^2=S^0(x_i^0,x_i^1)+x_{i-1}^1, & i>0\\ y_0^1=x_m^2\\ y_0^2=x_m^1 \end{cases} \tag{1}$$

值得注意的是在算法运行前，我们需要用2 bit随机数来初始化x_0^1和x_0^2.

2.5 基于毛刺探测的(d+1)-share门限实现的搜索方法

利用(d+1)-share门限实现对目标函数构建的一阶门限实现方案被寄存器分为两个独立的部分，分别称为扩散层和压缩层. 例如对一个二输入与门运算$x=f(a,b)=ab$构建的2-share门限实现方案为

$$\begin{cases} f^0(a^0,b^0)=a^0b^0+r, & \to x^a,\\ f^1(a^0,b^1)=a^0b^1, & \to x^b, \quad x^a+x^b=x^0\\ f^2(a^1,b^0)=a^1b^0, & \to x^c, \quad x^c+x^d=x^1\\ f^3(a^1,b^1)=a^1b^1+r, & \to x^d, \end{cases} \tag{2}$$

其中，a^0,a^1,b^0,b^1是输入掩码分量；r为随机数；x^0,x^1是输出掩码分量. 每个掩码分量函数$f^i(\bullet)$的结果被存储到寄存器中来阻止毛刺向后传播.

在毛刺探测模型下，目标函数的(d+1)-share门限实现除了需要满足门限实现的三个基本性质外，还需要满足“压缩层的每个输出掩码分量对应的所有输入的联合分布都应独立于输入变量”. 文献[21]提出一个搜索方法，可以在不添加随机数r的情况下，令设计出的门限实现方案满足毛刺探测安全性. 在该搜索方法中，为每个分量函数$f^i(\bullet)$都添加一次项a^0,a^1,b^0,b^1中的若干项，动态寻找特殊的掩码分量函数$f^0(\bullet),f^1(\bullet),f^2(\bullet)$和$f^3(\bullet)$的组合来满足毛刺探测安全模型的安全性要求. 例如对二输入与门运算$x=f(a,b)=ab$构建的无需额外随机数2-share门限实现方案如下所示：

$$\begin{cases} f^0(a^0,b^0)=a^0b^0, & \to x^a,\\ f^1(a^0,b^1)=a^0b^1+b^1, & \to x^b, \quad x^a+x^b=x^0\\ f^2(a^1,b^0)=a^1b^0, & \to x^c, \quad x^c+x^d=x^1\\ f^3(a^1,b^1)=a^1b^1+b^1, & \to x^d, \end{cases} \tag{3}$$

文献[21]为了确保其的搜索方法寻找的门限实现方案具有可组合性，每个目标函数的输出掩码分量x^0，x^1都必须用寄存器储存，需要更多的芯片面积且延迟也增大一倍.

在本文中，我们通过在扩散层引入额外随机数来对文献[21]提出的搜索算法进行优化，使得对三次函数设计的门限实现方案不需要存储输出掩码分量就具

有可组合性,降低门限实现方案所需的芯片面积和延迟. 在引入额外随机数时,我们采用环形结构[22],该结构与 Changing of the Guards 结合来确保设计出 uBlock 算法防护方案在加密过程中不需要引入额外的随机数.

3　方案设计

对 uBlock 算法的轮函数设计防护方案时,对于线性运算部件(如移位操作、向量置换等),只需要将这些运算应用到每个掩码分量上即可,关键在于如何对非线性部件S盒设计具有一阶毛刺探测安全的防护方案.

3.1　对直接型S盒设计防护方案

3.1.1　每个坐标函数的门限实现方案

由于S盒的四个坐标函数的代数次数最高是3次,因此我们将这四个坐标函数的布尔表达式抽象为四输入三次布尔函数 $x=f(a,b,c,d)$,其中,$<a,b,c,d>$ 为输入变量,x 为单输出变量. 根据2.5节的介绍,对四输入三次布尔函数 $f(\bullet)$ 构建的 2-share 门限实现方案可以分为两层:扩散层和压缩层,如图1所示.

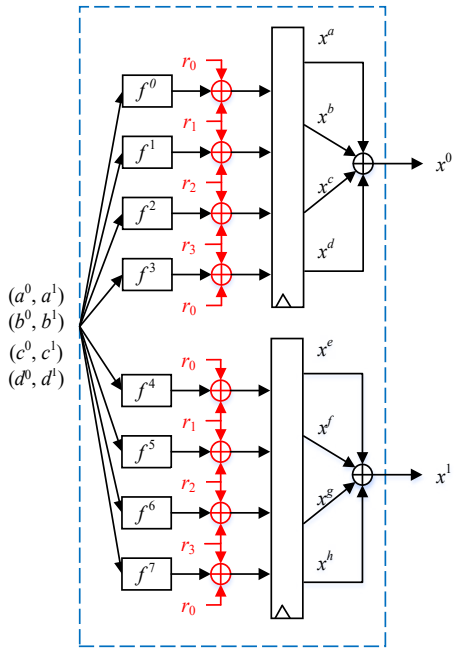

图1　单输出布尔函数的一阶门限实现结构图

结构介绍:在扩散层,每个输入变量被拆分为2个输入掩码分量,因此 $f(\bullet)$ 的每个三次项有 $2^3=8$ 个不同的掩码项. 这8个掩码项需要被安置在不同的掩码分量函数中,确保设计出的掩码方案满足一阶不完整性. 因此在扩散层,函数 $f(\bullet)$ 需要被拆分为8个掩码分量函数 $f^0(\bullet),\cdots,f^7(\bullet)$. 对于所有的掩码分量函数,我们采用表2的赋值方式来确定输入变量. 在表2中,I_i 表示输入变量,0、1分别表示 I_i 的两个掩码分量,即 I_i^0 和 I_i^1,$f^j(\bullet)$ 表示函数 $f(\bullet)$ 的第 j 个掩码分量函数. 例如,我们将 $<a,b,c,d>$ 赋值给 $<I_0,I_1,I_2,I_3>$,那么 $<a^0,b^0,c^0,d^0>$ 作为第一个掩码分量函数 $f^0(\bullet)$ 的输入,$<a^0,b^0,c^1,d^1>$ 作为第二个掩码分量函数 $f^1(\bullet)$ 的输入,以此类推. 这样可以确保函数 $f(\bullet)$ 被正确地拆分为8个掩码分量函数的形式. 然后,利用4个额外随机数 $r_0,\cdots,r_3$,对这8个掩码分量函数进行环式重掩码(如图1中的红色部分所示),得到8个中间掩码分量 $x^a,\cdots,x^h$. 如式(4)所示:

$$\begin{cases} x^a=f^0(\bullet)+r_0+r_1 \\ x^b=f^1(\bullet)+r_1+r_2 \\ x^c=f^2(\bullet)+r_2+r_3 \\ x^d=f^3(\bullet)+r_3+r_0 \\ x^e=f^4(\bullet)+r_0+r_1 \\ x^f=f^5(\bullet)+r_1+r_2 \\ x^g=f^6(\bullet)+r_2+r_3 \\ x^h=f^7(\bullet)+r_3+r_0 \end{cases} \tag{4}$$

为了阻止毛刺向后传输,这8个中间掩码分量需要用寄存器存储. 在压缩层阶段,前4个中间掩码分量 x^a,x^b,x^c,x^d 进行异或运算,得到输出掩码分量 x^0,后4个中间掩码分量 x^e,x^f,x^g,x^h 进行异或运算,得到输出掩码分量 x^1.

表2　输入掩码分量赋值给掩码分量函数的方式

掩码分量函数	I_0	I_1	I_2	I_3
$f^0(\bullet)$	0	0	0	0
$f^1(\bullet)$	0	0	1	1
$f^2(\bullet)$	0	1	0	1
$f^3(\bullet)$	0	1	1	0
$f^4(\bullet)$	1	0	0	1
$f^5(\bullet)$	1	0	1	0
$f^6(\bullet)$	1	1	0	0
$f^7(\bullet)$	1	1	1	1

安全性分析:在毛刺探测模型下,敌手探测电路中的任何一个位置时,可以同时获取该位置所在组合逻辑电路的所有输入信息. 由于在我们提出的门限实现方案中,中间掩码分量 $x^a,\cdots,x^h$ 存储到寄存器中,因此敌手探测该方案中的任何一个位置时,只能获得该位置所在层的一些相关信息,不会同时获取扩散层和压缩层的信息. 因此该方案的安全性需要分两种情况来讨论:

(1)敌手探测扩散层的输出值:敌手能够获取探测点以及探测点所在的掩码分量函数的输入信息. 由于每个掩码分量函数 $f^i(\bullet)$ 只包含输入变量的一个掩码分量,且输出值被重掩码后服从均匀分布,因此敌手无法根据探测点以及探测点所在的掩码分量函数的输入信

息来获取输入变量$<a,b,c,d>$的任何信息.

(2)敌手探测压缩层的输出值:敌手探测一个输出掩码分量x^0时,同时获取中间掩码分量x^a,x^b,x^c,x^d的信息,因此中间掩码分量x^a,x^b,x^c,x^d的联合概率分布必须独立于输入变量$<a,b,c,d>$. 对于中间掩码分量x^e,x^f,x^g,x^h,同理. 在我们提出的门限实现方案中,由于采用环式重掩码的方式,对于任意给定的8个掩码分量函数,(x^a,x^b,x^c,x^d)和(x^e,x^f,x^g,x^h)的统计分布不一定独立于输入变量$<a,b,c,d>$,因此我们需要用搜索算法1来搜索满足条件的特殊掩码分量函数.

算法1 对任意四输入三次函数搜索2-share门限实现方案

输入: $x=f(a,b,c,d)$ //目标函数

输出: $M^{0,\cdots,7}:\{f^0(\bullet),\cdots,f^7(\bullet)\}$ //掩码分量函数

1. FOR $i=0$ to 7
2. $M^i\leftarrow\forall f^i(\bullet):F_2^n\rightarrow F_2$ //获取可能的掩码分量函数
3. $M^{0,1,2,3}\leftarrow\varnothing$
4. FOR $((f^0(\bullet),f^1(\bullet),f^2(\bullet),f^3(\bullet))\in M^0\times M^1\times M^2\times M^3)$
5. IF $(\exists\alpha;\forall a,b,c,d;P(f^0(\bullet)+r_0+r_1,f^1(\bullet)+r_1+r_2,f^2(\bullet)+r_2+r_3,f^3(\bullet)+r_3+r_0)=\alpha)$ //联合独立性
6. IF $(f^0(\bullet)+f^1(\bullet)+f^2(\bullet)+f^3(\bullet)$是均匀的) //均匀性
7. $M^{0,1,2,3}\leftarrow M^{0,1,2,3}\cup(f^0(\bullet),f^1(\bullet),f^2(\bullet),f^3(\bullet))$
8. $M^{4,5,6,7}\leftarrow\varnothing$
9. FOR $((f^4(\bullet),f^5(\bullet),f^6((\bullet)),f^7((\bullet)))\in M^4\times M^5\times M^6\times M^7)$
10. IF $(\exists\alpha;\forall a,b,c,d;P(f^4((\bullet))+r_0+r_1,f^5(\bullet)+r_1+r_2,f^6(\bullet)+r_2+r_3,r_3+f^7(\bullet)+r_0)=\alpha)$ //联合独立性
11. IF $(f^4(\bullet)+f^5(\bullet)+f^6(\bullet)+f^7(\bullet)$是均匀的) //均匀性
12. $M^{4,5,6,7}\leftarrow M^{4,5,6,7}\cup(f^4(\bullet),f^5(\bullet),f^6(\bullet),f^7(\bullet))$
13. $M^{0,\cdots,7}\leftarrow\varnothing$
14. FOR $((f^0(\bullet),\cdots,f^7(\bullet))\in M^{0,1,2,3}\times M^{4,5,6,7})$
15. IF $(\forall a,b,c,d;f^0(\bullet)+\cdots+f^7(\bullet)=f(a,b,c,d))$ //正确性
16. $M^{0,\cdots,7}\leftarrow M^{0,\cdots,7}\cup(f^0(\bullet),\cdots,f^7(\bullet))$
17. RETUREN $M^{0,\cdots,7}$

在算法1中:函数$x=f(a,b,c,d)$中的任何一个三次项的所有掩码项被安置在8个掩码分量函数中的方法只有一种,而任何一个二次项的安置方法有$2^4=16$种,一次项有$4^2=16$种,因此第1、2行用于生成每个掩码分量函数$f^i(\bullet)$的所有可能形式,形成的集合记为M^i,其中M^i中的每个元素与函数$f(a,b,c,d)$有相同的三次项. 第3~5行和第8~10行用于判断前4个中间掩码分量和后4个中间掩码分量的联合分布是否分别独立于输入变量$<a,b,c,d>$. 第6、7行和第11、12行用于判断输出掩码分量x^0和x^1是否分别服从均匀分布,确保下一个目标函数的输入变量被均匀掩码. 第14~16行用于判断8个掩码分量函数构成的门限实现方案是否满足门限实现的正确性.

3.1.2 四个坐标函数的门限实现方案的可组合性

利用算法1对每个坐标函数$f_i(\bullet)$搜索到满足条件的门限实现方案有很多个,设其构成的集合记为M_i. 从$M_0,\cdots,M_3$的每个集合中都选一个元素,可以构成S盒的一个防护方案,那么该防护方案整体是否具有毛刺探测安全性需要进一步分析.

在S盒替换操作之后的线性运算中,总会存在这样的情况,即每个坐标函数的一个输出掩码分量进行布尔运算,如四个坐标函数的第一个掩码分量进行异或运算,即$y=x_0^0\oplus x_1^0\oplus x_2^0\oplus x_3^0$. 由于每个输出掩码分量没有存储到寄存器中,因此在毛刺探测模型下,敌手探测该异或运算的输出y时,他能够同时获取四个坐标函数的中间掩码分量$(x_0^a,\cdots,x_0^d)$,$(x_1^a,\cdots,x_1^d)$,$(x_2^a,\cdots,x_2^d)$和$(x_3^a,\cdots,x_3^d)$的信息. 算法1确保了这4组中间掩码分量的分布分别独立于输入变量$<a,b,c,d>$,但这并不意味着这4组中间掩码分量的联合分布独立于输入变量$<a,b,c,d>$. 因此我们还需要算法2来搜索不同坐标函数的掩码分量函数的组合不会泄露输入变量$<a,b,c,d>$. 由于这四个坐标函数的任何一个输出掩码分量进行布尔运算的组合存在$2^4=16$种可能,因此我们在算法2中需要判断这16种可能的组合是否均不会泄露输入变量$<a,b,c,d>$,这对应于算法2的第4行.

算法2 寻找uBlock算法S盒的2-share门限实现方案

输入: M_0,M_1,M_2,M_3

输出: $\{\bar{f}_0(\bullet),\bar{f}_1(\bullet),\bar{f}_2(\bullet),\bar{f}_3(\bullet)\}$

1. FOR $i=0$ to 3
2. FOR $(\bar{f}_i(\bullet)\in M_i)$
3. $M_i\leftarrow M_i-\bar{f}_i(\bullet)$
4. IF $(\forall a,b,c,d$: 16种掩码分量组合的联合分布独立于输入变量$<a,b,c,d>)$
5. IF $(\bar{f}_0(\bullet),\cdots,\bar{f}_i(\bullet)$的输出满足联合均匀)
6. BREAK
7. RETUREN $\{\bar{f}_0(\bullet),\bar{f}_1(\bullet),\bar{f}_2(\bullet),\bar{f}_3(\bullet)\}$

我们利用算法1、2对S盒成功搜索到1 672个具有毛刺探测安全的2-share防护方案. 由于文章篇幅有限,我们将一个具体方案上传到GitHub.

3.2 对分解型S盒设计防护方案

该S盒属于等价类[16]C_{223}^4,可以分解为两个代数次数为2的4 bit S盒F和G的级联形式$S(x)=G(F(x))$. 设F的输入变量为$(x_3,\cdots,x_0)$,输出变量为$(a_3,\cdots,a_0)$,那么F的代数表达式为

$$\begin{cases}a_3=x_1+x_2+x_3+x_1x_2\\a_2=x_2\\a_1=x_1\\a_0=1+x_0+x_2x_3\end{cases}\tag{5}$$

设 G 的输入变量为 $(a_3,\cdots,a_0)$，输出变量为 $(y_3,\cdots,y_0)$，那么 G 的代数表达式为

$$\begin{cases} y_3=a_3 \\ y_2=1+a_2+a_0a_1 \\ y_1=a_0+a_1+a_3+a_0a_3 \\ y_0=a_0 \end{cases} \tag{6}$$

将 F 和 G 视为三次项为0的三次S盒，我们可以用算法1和算法2对其搜索具有一阶毛刺探测安全的门限实现方案. 对每个坐标函数应用算法1时，由于一个二次项包含 $2^2=4$ 个掩码项，一次项包含2个掩码项，因此一次项和二次项的掩码项的不同组合，可使得在8个掩码分量函数中，存在0~4个掩码分量函数的表达式恒为0. 由于掩码分量个数与方案所占的芯片面积之间成正比，因此我们选择只保留有4个掩码分量函数的表达式恒为0的门限实现方案. 最终，我们对 F 和 G 均能搜索到上百万个满足条件的门限方案. F 的代数表达式(5)的四个坐标函数的一个门限实现方案的扩散层表达式为式(7)~(10).

$$\begin{cases} a_3^0=x_1^0x_2^0+r_0 \\ a_3^1=x_1^0+x_3^1+x_1^0x_2^1+r_0 \\ a_3^2=x_2^0+x_3^0+x_1^1x_2^0+r_0 \\ a_3^3=x_1^1+x_2^1+x_1^1x_2^1+r_0 \end{cases} \tag{7}$$

$$\begin{cases} a_2^0=x_3^1 \\ a_2^1=x_3^0 \end{cases} \tag{8}$$

$$\begin{cases} a_1^0=x_2^0 \\ a_1^1=x_2^1 \end{cases} \tag{9}$$

$$\begin{cases} a_0^0=x_2^0x_3^0+1+r_1 \\ a_0^1=x_0^0+x_2^1x_3^1+r_1 \\ a_0^2=x_2^0x_3^1+r_1 \\ a_0^3=x_0^1+x_2^1x_3^0+r_1 \end{cases} \tag{10}$$

G 的代数表达式(6)的四个坐标函数的一个门限实现方案的扩散层表达式为式(11)~(14).

$$\begin{cases} y_3^0=a_2^1+a_3^1 \\ y_3^1=a_2^1+a_3^0 \end{cases} \tag{11}$$

$$\begin{cases} y_2^0=a_0^0a_1^0+1+r_2 \\ y_2^1=a_3^1+a_0^0a_1^1+r_2 \\ y_2^2=a_2^0+a_0^1a_1^0+r_2 \\ y_2^3=a_2^1+a_3^1+a_0^1a_1^1+r_2 \end{cases} \tag{12}$$

$$\begin{cases} y_1^0=a_0^0a_3^0+r_3 \\ y_1^1=a_0^0+a_1^0+a_0^0a_3^1+r_3 \\ y_1^2=a_3^0+a_0^1a_3^0+r_3 \\ y_1^3=a_0^1+a_1^1+a_3^1+a_0^1a_3^1+r_3 \end{cases} \tag{13}$$

$$\begin{cases} y_0^0=a_0^0 \\ y_0^1=a_0^1 \end{cases} \tag{14}$$

3.3　硬件实现结构

在加密准备阶段，将两个128 bit的明文掩码分量与两个128 bit的主密钥掩码分量进行异或运算，然后执行16轮掩码型轮操作. 每个掩码型轮操作包含加密运算和密钥扩展两个部分，如图2中的蓝色和红色虚线框内. 加密运算的掩码型轮函数包含5个部分，分别是多路选择器0，S盒替换，移位操作，向量置换和轮密钥加.

(1)多路选择器0：当加密轮数是第一轮时，该选择器将明文掩码分量与主密钥掩码分量的异或值(共256 bit)馈入到S盒替换模块；当加密轮数是其他轮时，该选择器将轮函数的两个输出掩码分量(共256 bit)馈入到S盒替换模块.

(2)S盒替换：32个掩码型S盒并行运算，其中掩码型S盒分为直接型S盒防护方案和分解型S盒防护方案. 由于扩散层的输出需要用寄存器存储，因此直接型S盒防护方案需要消耗1个时钟周期，而分解型S盒防护方案需要消耗2个时钟周期.

(3)移位操作、向量置换、轮密钥加：这三个模块是线性运算，用一个组合逻辑来实现即可.

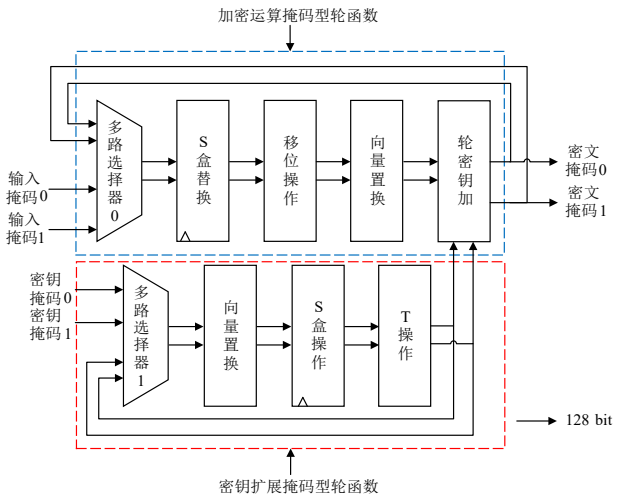

图2　基于轮函数的uBlock算法硬件实现结构

密钥扩展的掩码型轮函数包含4个部分，即多路选择器1，向量置换、S盒操作和T操作. 多路选择器1的作用与多路选择器0相同，用来选择正确的输入馈入到向量置换模块. S盒操作采用的防护方案与加密运算部分相同. T操作属于线性运算，用组合逻辑来实现.

3.4　基于Changing of the Guards的随机数优化

在我们的uBlock算法防护方案中，每个S盒都需要一定量的额外随机数来确保其门限实现方案的扩散层输出掩码分量的毛刺探测安全性. 如果利用Changing of the Guards技术来优化这些额外随机数，那么关键之

处在于随着轮函数的迭代，我们能否为每个掩码型S盒找到足够的Guards来代替这些额外随机数.

不管是基于直接型S盒的uBlock算法防护方案，还是基于分解型S盒的uBlock算法防护方案，每个掩码型S盒额外随机数的引入方式均为环式(见式(4))，因此在压缩层的计算过程中，它们会被异或掉，不会对掩码型S盒的输出掩码分量以及后续函数的数据分布产生影响. 这意味着轮函数的迭代并不会造成Guards数量的减少，因此每个轮函数的每个掩码型S盒都会有足够的Guards来代替这些额外随机数. 在这种情况下，如果对于一个轮函数的每个掩码型S盒，可以在电路中找到与该掩码型S盒的运算过程不相关的足够中间变量，那么它们可以用来作为该掩码型S盒的额外随机数，确保该掩码型S盒的扩散层输出分量的毛刺探测安全性. 这样，我们就可以用Changing of the Guards技术来对uBlock算法防护方案的额外随机数进行优化.

对于基于直接型S盒的uBlock算法防护方案的第i个掩码型S盒，我们发现第$(i+1)\bmod 16$，$(i+2)\bmod 16$，$(i+3)\bmod 16$个掩码型S盒的输入掩码分量与该掩码型S盒的计算过程是相互独立的. 故可以从第$(i+1)\bmod 16$，$(i+2)\bmod 16$，$(i+3)\bmod 16$个掩码型S盒的输入掩码分量(共12 bit)中选择10 bit来代替第i个掩码型S盒所需的10 bit额外随机数(如图3所示)，从而避免了为每个S盒引入额外的随机数.

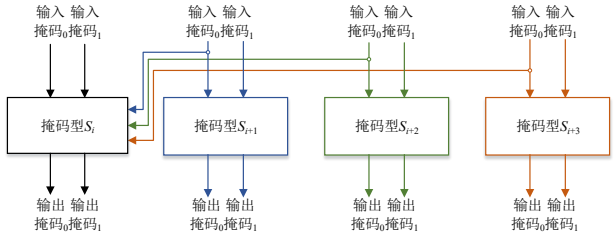

图3 直接型S盒防护方案中Guards的添加方式

对于基于分解型S盒的uBlock算法防护方案来说，每个掩码型S盒需要4 bit额外随机数来确保其的毛刺探测安全性. 同样地，这些随机数在压缩层中被异或掉，因此我们可以采用相同的方法Changing of the Guards技术来优化这些额外随机数. 具体来说，对于第i个掩码型S盒，我们用第$(i+1)\bmod 16$个掩码型S盒的4 bit输入掩码分量来代替该S盒所需的4比额外随机数，如图4所示.

4 安全性评估与性能评估

4.1 S盒安全性的评估

SILVER是用于在各种安全概念(如探测安全，毛刺探测安全)下，评估加密电路安全性的主流形式化验证工具. 该工具使用最简有序二分决策图(Reduced Order Binary Decision Diagrams，ROBDD)来验证敌手探测的观测集合与敏感变量之间的统计独立性来评估方案的安全性. 由于SILVER对敏感变量集和观测集的联合分布进行符号和详尽的分析，因而避免了假阴性情况.

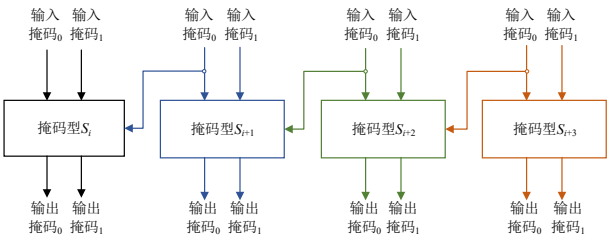

图4 分解型S盒防护方案中Guards的添加方式

由于该评估方法的对象是硬件实现的网表文件，因此在评估前，我们需要用Synopsis Design Compiler在NanGate 45 nm工艺库下将这两个S盒防护方案的RTL级代码综合生成对应的网表文件. 在综合过程中，我们禁用了不同模块之间的优化，确保网表文件满足门限实现的d阶不完整性.

对于直接型S盒防护方案，我们将10 bit Guards配置为随机数. 整个评估过程耗时约0.137 s，评估结果显示该防护方案具有一阶毛刺探测安全性(如图5的第3行所示)，且输出掩码分量满足联合均匀性(如图5的第4行所示).

```
1  [     0.000] Netlist: /home/uBlock/nodecompose/nl/unflattened/SharedSbox.nl
2  [     0.044] probing.standard (d ≤ 1) -- [1;32mPASS[0m.
3  [     0.094] probing.robust   (d ≤ 1) -- [1;32mPASS[0m.
4  [     0.137] uniformity               -- [1;32mPASS[0m.
```

图5 直接型S盒防护方案的SILVER评估结果

对于分解型S盒防护方案，我们将4 bit Guards配置为随机数. 整个评估过程消耗0.072 s，评估结果表明该防护方案具有一阶毛刺探测安全性(如图6的第3行所示)，且输出掩码分量满足联合均匀性(如图6的第10行所示).

```
1  [     0.000] Netlist: /home/uBlock/decompose/nl/unflattened/SharedSbox.nl
2  [     0.023] probing.standard (d ≤ 1) -- [1;32mPASS[0m.
3  [     0.061] probing.robust   (d ≤ 1) -- [1;32mPASS[0m.
4  [     0.072] uniformity               -- [1;32mPASS[0m.
```

图6 分解型S盒防护方案的SILVER评估结果

4.2 整个电路安全性的评估

当目标电路是uBlock算法防护方案的整个硬件实现电路时，若电路实现采用本文给出的迭代式结构，SILVER无法评估整个电路的安全性. 为了确保整个电路的安全性，我们在Sakura_G开发板上用泄露评估技术TVLA对其进行全面的泄露检测.

Sakura_G开发板是一款专用于执行侧信道攻击的开发板. 该开发板的信噪比较高，有利于分析待检测电路是

否存在信息泄露. 我们启用了“keep_hierarchy”约束,阻止了Xilinx ISE对不同模块之间的优化,确保了综合过程不会破坏防护方案的安全性. 我们用示波器采集芯片运行过程产生的功耗曲线. Sakura_G开发板运行的时钟频率是3 MHz,示波器的采集频率是500 M/s.

用TVLA对整个电路进行安全性评估时,首先采集明文分别是固定和随机情况下的N条功耗曲线,然后进行Welch t-test计算以评估明文分别是固定和随机的两个曲线集之间的差异性,其统计量的计算公式如下所示:

$$T=\frac{m_1-m_2}{\sqrt{\frac{s_1^2}{N}+\frac{s_2^2}{N}}} \tag{15}$$

其中,m_1和m_2分别是两个曲线集的均值;s_1^2和s_2^2分别是两个曲线集的方差;T是Welch t-test的统计量. 最后根据计算出的T值来判断被评估的电路是否存在信息泄露. 当采集的曲线量$N>5\,000$,计算出的T值不超过±4.5时,两个曲线集之间不存在差异的置信度大于99.999%. 因此目前工业界和学术界均设定T=±4.5作为统计量的阈值,即计算出的T值超过±4.5时,表明整个电路不存在信息的泄露. 反之,则存在信息的泄露.

为了评估整个电路的安全性,我们进行了两个实验:关闭随机数发生器和打开伪随机数发生器情况下的整个电路的TVLA信息泄露评估. 在关闭伪随机数发生器的情况下,明文和密钥不会被拆分为两个掩码分量的形式,此时电路运行的是无防护的uBlock密码算法. 通过一阶TVLA泄露检测,理论上该电路应该产生信息泄露,即T值超过阈值±4.5. 而在打开伪随机数发生器的情况下,整个电路运行的是添加防护方案后的uBlock密码算法. 对其进行一阶TVLA泄露检测,理论上该电路不会产生信息的泄露. 通过这两个对比实验,我们证明我们设计的uBlock算法防护方案不会产生一阶信息泄露,即具有一阶安全性.

伪随机数发生器关闭:对基于直接型S盒和分解型S盒的uBlock算法防护方案分别采集了500万条功耗曲线. 图7和图8分别给出了两个防护方案的一阶TVLA结果. 在500万条功耗曲线下,两个防护方案的T值均超过4.5,说明在未防护情况下,两个方案均存在信息泄露,达到预期目标.

伪随机数发生器打开:对两种uBlock算法防护方案分别采集1 000万条功耗曲线. 图9和图10分别给出了1 000万曲线量下,两个防护方案的一阶TVLA结果,图11和图12是对应的二阶TVLA结果. 从图中可以发现,两种防护方案的一阶T值均没超过4.5,而二阶T值均出现超过4.5的情况. 该实验结果说明,在1 000万条功耗曲线量下,这两种防护方案均没有一阶信息泄露,因而具有一阶安全性,达到预期目标.

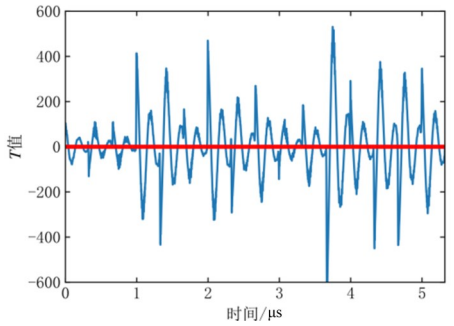

图7　直接型uBlock防护方案的一阶TVLA(PRNG关闭)

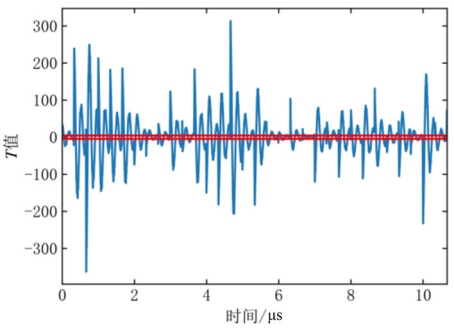

图8　分解型uBlock防护方案的一阶TVLA(PRNG关闭)

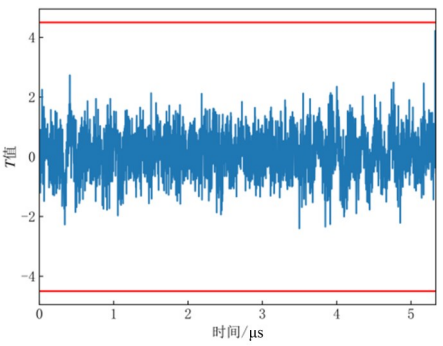

图9　直接型uBlock防护方案的一阶TVLA(PRNG打开)

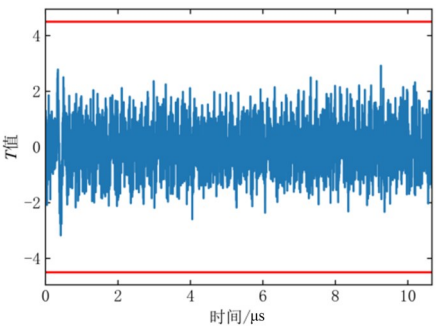

图10　分解型uBlock防护方案的一阶TVLA(PRNG打开)

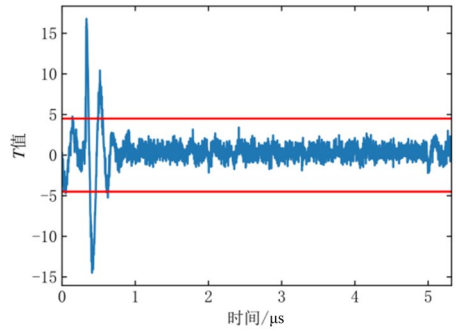

图11 直接型uBlock防护方案的二阶TVLA(PRNG打开)

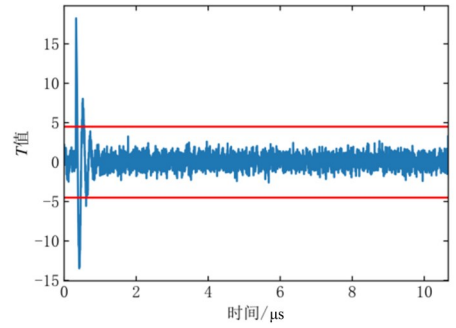

图12 分解型uBlock防护方案的二阶TVLA(PRNG打开)

4.3 性能评估

在本节中,我们在UMC 180 nm和NanGate 45 nm工艺库下,分别对uBlock算法的两个防护方案进行性能评估.表3给出了已有uBlock算法防护方案在安全性、面积、随机数和延迟等方面的比较.其中方案1指的是本文提出的基于直接型S盒的uBlock算法防护方案,方案2指的是本文提出的基于分解型S盒的uBlock算法防护方案.

表3 已有uBlock算法防护方案的安全性和资源消耗对比

设计	安全模型		面积		随机数	延迟
	探测	毛刺探测	kGE	工艺库	bit/S盒	clk
文献[15]	×	×	6.0	TSMC 180 nm	0	16
方案1	√	√	25.6	UMC 180 nm	0	16
方案2	√	√	20.4		0	32
3-share方案[17]	√	√	10.3	NanGate 45 nm	0	656
2-share方案[17]	√	×	7.3		0	672
方案1	√	√	20.7		0	16
方案2	√	√	16.8		0	32

相比于文献[15],基于直接型S盒和分解型S盒的uBlock算法防护方案分别在4.2倍和3.4倍面积的代价下,获得了抵抗一阶侧信道攻击的能力.相比于文献[17]的3-share方案,基于直接型S盒和分解型S盒的uBlock算法防护方案将延迟由656 clk分别降低至16 clk(约97.6%)和32 clk(95.1%).相比于文献[17]的2-share方案,我们不仅将两个方案的延迟缩减为16 clk和32 clk,还将方案的安全模型由探测模型提升到毛刺探测模型.

5 总结

针对uBlock算法缺乏低延迟的防护方案这一问题,本文提出了一种自动化搜索方法来寻找具有一阶毛刺探测安全且低延迟的门限实现方案.基于该方法,我们对uBlock算法构建了两种新型的一阶门限实现方案:基于直接型S盒和基于分解型S盒.在硬件实现方面,我们给出了基于轮函数的硬件实现结构.与现有的uBlock防护方案相比,我们的方案将延迟分别降低97.6%和95%,显著提升了加解密的效率.

参考文献

[1] BIHAM E, SHAMIR A. Differential cryptanalysis of DES-like cryptosystems[C]//Advances in Cryptology — CRYPTO'90. Berlin: Springer, 1990: 2-21.

[2] MATSUI M. Linear cryptanalysis method for DES cipher [C]//Advances in Cryptology — EUROCRYPT'93. Berlin: Springer, 1994: 386-397.

[3] KOCHER P C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems[C]//Advances in Cryptology — CRYPTO'96. Berlin: Springer Berlin Heidelberg, 1996: 104-113.

[4] XIE N J, GONG Z, TANG Y F, et al. Protecting white-box block ciphers with galois/counter mode[C]//2022 IEEE Conference on Dependable and Secure Computing (DSC). Piscataway: IEEE, 2022: 1-7.

[5] KOCHER P, JAFFE J, JUN B. Differential power analysis [C]//Advances in Cryptology — CRYPTO'99. Berlin: Springer Berlin Heidelberg, 1999: 388-397.

[6] QUISQUATER J J, SAMYDE D. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards [C]//Smart Card Programming and Security. Berlin: Springer Berlin Heidelberg, 2001: 200-210.

[7] BONEH D, DEMILLO R A, LIPTON R J. On the importance of checking cryptographic protocols for faults[C]// Advances in Cryptology — EUROCRYPT'97. Berlin: Springer Berlin Heidelberg, 1997: 37-51.

[8] CHARI S, JUTLA C S, RAO J R, et al. Towards sound approaches to counteract power-analysis attacks[C]//Advances in Cryptology — CRYPTO'99. Berlin: Springer Berlin Heidelberg, 1999: 398-412.

[9] ISHAI Y, SAHAI A, WAGNER D. Private circuits: Securing

hardware against probing attacks[C]//Advances in Cryptology - CRYPTO 2003. Berlin: Springer, 2003: 463-481.

[10] MANGARD S, PRAMSTALLER N, OSWALD E. Successfully attacking masked AES hardware implementations[C]//Cryptographic Hardware and Embedded Systems — CHES 2005. Berlin: Springer, 2005: 157-171.

[11] NIKOVA S, RECHBERGER C, RIJMEN V. Threshold implementations against side-channel attacks and glitches [C]//Information and Communications Security. Berlin: Springer, 2006: 529-545.

[12] MORADI A, POSCHMANN A, LING S, et al. Pushing the limits: A very compact and a threshold implementation of AES[C]//Advances in Cryptology — EUROCRYPT 2011. Berlin: Springer, 2011: 69-88.

[13] POSCHMANN A, MORADI A, KHOO K, et al. Side-channel resistant crypto for less than 2, 300 GE[J]. Journal of Cryptology, 2011, 24(2): 322-345.

[14] MORADI A, SCHNEIDER T. Side-channel analysis protection and low-latency in action[C]//Advances in Cryptology — ASIACRYPT 2016. Berlin: Springer, 2016: 517-547.

[15] WU W L, ZHANG L, ZHENG Y F, et al. The block cipher uBlock[J]. Journal of Cryptologic Research, 2019, 6(6): 690-703.

[16] BILGIN B, NIKOVA S, NIKOV V, et al. Threshold implementations of all 3×3 and 4×4 S-boxes[C]//Cryptographic Hardware and Embedded Systems — CHES 2012. Berlin: Springer, 2012: 76-91.

[17] 焦志鹏, 陈华, 姚富, 等. uBlock算法的低代价门限实现侧信道防护方法[J]. 计算机学报, 2023, 46(3): 657-670.
JIAO Z P, CHEN H, YAO F, et al. The low cost threshold implementation method of uBlock algorithm against side-channel attacks[J]. Chinese Journal of Computers, 2023, 46(3): 657-670. (in Chinese)

[18] MANGARD S, POPP T, GAMMEL B M. Side-channel leakage of masked CMOS gates[C]//Lecture Notes in Computer Science. Berlin: Springer, 2005: 351-365.

[19] FAUST S, GROSSO V, MERINO DEL POZO S, et al. Composable masking schemes in the presence of physical defaults & the robust probing model[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018: 89-120.

[20] KNICHEL D, SASDRICH P, MORADI A. SILVER — statistical independence and leakage verification[C]//Advances in Cryptology — ASIACRYPT 2020. Cham: Springer International Publishing, 2020: 787-816.

[21] REZAEI SHAHMIRZADI A, MORADI A. Re-consolidating first-order masking schemes[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020: 305-342.

[22] REPARAZ O, BILGIN B, NIKOVA S, et al. Consolidating masking schemes[C]//Lecture Notes in Computer Science. Berlin: Springer, 2015: 764-783.

作者简介

姚　富　男,1990年10月出生于山西省朔州市. 现为中国科学院软件研究所博士研究生. 研究方向为密码算法侧信道分析与防护.
E-mail: yaofu2020@iscas.ac.cn

陈　华　女,1976年10月生于山东省日照市. 现为中国科学院软件研究所正高级工程师, 博士生导师. 研究方向为侧信道分析与防护、密码检测.
E-mail: chenhua@iscas.ac.cn

范丽敏　女,1978年12月生于内蒙古自治区赤峰市. 现为中国科学院软件研究所高级工程师,硕士生导师. 研究方向为侧信道分析与防护、密码检测.
E-mail: fanlimin@iscas.ac.cn