uBlock算法的低代价门限实现侧信道防护方法

焦志鹏[1),2)] 陈 华[1)] 姚 富[1),2)] 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 在传统的基于黑盒模型的密码分析中,攻击者仅可以利用密码算法的输入输出信息进行攻击,现有密码算法在黑盒模型下的安全性已经得到较为充分的论证.但是在灰盒模型下,攻击者的能力得到提高,其不仅可以获取密码算法的输入输出信息,还可以获得密码算法实际执行过程中泄露的功耗、电磁、光等物理信息,这些物理信息和密码算法的中间状态具有相关性,敌手可以利用这种相关性进行秘密信息的恢复,这种攻击被称为侧信道攻击.侧信道攻击自提出以来,由于其相对低的实现代价以及较高的攻击效率对于密码算法的实现安全性造成了严重的威胁.uBlock算法是2019年全国密码算法设计竞赛分组密码一等奖获奖算法,同样受到了侧信道攻击的威胁.目前针对uBlock算法的研究较少,在硬件实现方面主要考虑低延迟高吞吐量的实现,缺乏针对资源受限情况下的低代价优化实现,不利于侧信道防护方案的构造.目前公开的文献中指出其S盒适用于基于门限实现的侧信道防护方案构造,存在3-share的无需新随机数的门限防护方案,但是没有给出具体的实现方案.针对这样的现状,本文首先基于流水线和串行化的思想设计并实现了一种适用于uBlock算法的低代价硬件实现方案;其次在上述低代价实现方案的基础上构造并实现了一种适用于uBlock算法的3-share无需新随机数的门限防护方案;最后针对3-share门限实现面积消耗较大的问题进一步优化实现代价,构造并实现了一种2-share的无需新随机数的门限防护方案.为了验证上述防护方案的实际安全性,本文在FPGA开发板中进行了实际实现,并基于测试向量泄露评估技术进行侧信道评估实验,实验结果验证了3-share和2-share uBlock算法门限防护方案对于侧信道攻击均具有相应的防护能力.为了进一步对比防护方案之间的实现代价,本文从寄存器消耗、随机数消耗以及时延等角度对各个方案进行了对比,然后又通过实验评估了不同方案在FPGA资源、等效门等方面的消耗,总而言之上述代价对比说明2-share无需新随机数的门限实现方案的资源消耗相对于其他防护方案较少,其中在面积消耗上其相对于3-share的无需新随机数的门限实现方案降低了约30%.

关键词 uBlock算法;侧信道攻击;侧信道防护;门限实现;硬件安全

中图法分类号 TP309 **DOI号** 10.11897/SP.J.1016.2023.00657

The Low Cost Threshold Implementation Method of uBlock Algorithm Against Side Channel Attacks

JIAO Zhi-Peng[1),2)] CHEN Hua[1)] YAO Fu[1),2)] FAN Li-Min[1)]

[1)](*Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing* 100190)

[2)](*University of Chinese Academy of Sciences, Beijing* 100049)

Abstract In traditional cryptographic analysis based on black box model, attackers can only use the input and output information of cryptographic algorithm to attack, and the security of existing cryptographic algorithms under black box model has been fully demonstrated. However, under the grey box model, the ability of the attacker is improved. It can not only obtain the input and output information of the cryptographic algorithm, but also obtain the power consumption, electromagnetic, optical and other physical information leaked during the actual implementation

收稿日期:2021-08-30;在线发布日期:2022-02-10.本课题得到国家自然科学基金项目(62172395)资助.**焦志鹏**,博士研究生,主要研究领域为侧信道分析与防护.E-mail:zhipeng2017@iscas.ac.cn. **陈 华(通信作者)**,博士,正高级工程师,主要研究领域为侧信道分析与防护、密码检测.E-mail:chenhua@iscas.ac.cn. **姚 富**,博士研究生,主要研究领域为侧信道分析与防护. **范丽敏**,博士,高级工程师,主要研究领域为侧信道分析与防护、密码检测.

of the cryptographic algorithm. The correlation between the physical information and the intermediate state of the cryptographic algorithm can be used by the adversary to recover the secret information, which is called the side-channel attack. Since the side-channel attack was proposed, it has posed a serious threat to the security of the implementation security of cryptography algorithm because of its relatively low implementation cost and high attack efficiency. uBlock algorithm is the first prize winning algorithm of block cipher in 2019 National Cryptographic Algorithm Design Competition, which is also threatened by side channel attacks. Now study of uBlock algorithm is less, in terms of hardware implementation mainly consider low delay implementation and high throughput, and lack of low-cost optimization implementations for resource-constrained situations, and it is not conducive to the construction of protection scheme against side-channel attack. The current public literature points out its S-box is suitable for the construction of side channel protection scheme based on threshold implementation, and there is a 3-share threshold implementation scheme without new random number during the execution of the algorithm, but does not give a concrete implementation scheme. In view of this situation, this paper firstly designs and implements a low cost hardware implementation scheme suitable for uBlock algorithm based on pipeline and serialization. Secondly, based on the above low cost implementation scheme, a 3-share threshold implementation scheme without new random numbers is constructed and implemented for uBlock algorithm. Finally, aiming at the problem that the implementation area of 3-share threshold protection scheme consumes a large amount, the implementation cost is further optimized, and a 2-share threshold protection scheme without new random numbers is constructed and implemented. In order to verify the actual security of the above protection schemes, this paper carried out the actual implementation in FPGA development board, and carried out the side-channel evaluation experiments based on the test vector leakage evaluation technology. Experimental results verify that 3-share and 2-share threshold protection schemes of uBlock algorithm have the corresponding protection ability against side channel attacks. In order to further compare the realization cost of protection schemes, this paper compares each scheme from the perspective of register consumption, random number consumption and delay, and then evaluates the consumption of different schemes in FPGA resources, equivalent gates through experiments. All in all, the cost comparison above shows that the resource consumption of 2-share threshold implementation without new random number is the lowest compared with other protection schemes, and the area consumption of 2-share threshold implementation without new random number is reduced by about 30% compared with 3-share threshold implementation without new random number.

Keywords uBlock algorithm; side channel attack; side channel protection; threshold implementation; hardware security

1 引 言

密码算法是信息安全的重要保障.在传统的密码学研究中,主要考虑黑盒模型下的密码算法安全性,通过数学分析的方法评估密码算法的安全性.随着灰盒攻击的出现[1],密码算法的实现安全性受到了严重威胁.在灰盒攻击模型下,攻击者在黑盒模型攻击者的能力的基础上,还具有获取密码设备执行过程中侧信道信息[2-3]、甚至干扰设备执行的能力[4].其中利用密码设备执行过程中的侧信道信息而不干扰密码设备正常运行的灰盒攻击方法被称为侧信道攻击.侧信道攻击因为实现代价相对其他灰盒攻击较低,攻击效率相对于黑盒攻击方法更高,是密码算法实现安全性的重要威胁.

针对侧信道攻击的威胁,不同的防护策略被提

出.如文献[5]所述,可以从芯片级、系统级、算法级、门级以及晶体管级实现相应的防护策略,每一类的防护策略都有各自的优缺点和应用场景,本文主要研究算法级别的防护策略.在算法级别,按照防护思想的不同,可以分为隐藏技术和掩码技术.侧信道攻击的本质是利用密码设备执行过程中泄露的侧信道信息与执行算法的相关性进行相应的攻击,防护方法的本质就是破坏这种相关性.隐藏技术[6-7]通过随机化或者是均衡化密码设备的侧信道信息来减弱甚至消除密码算法的中间操作和设备侧信道信息之间的依赖关系从而实现对于侧信道攻击的防护,由于其实现方法相对经验化,一些研究表明部分基于隐藏技术设计的防护方案[8-9]存在相应的安全缺陷.掩码技术通过随机化密码设备处理的中间值来达到破坏侧信道信息和中间操作之间关系的目的,相对于隐藏技术,掩码技术理论上更加完备,目前已成为侧信道攻击防护方案中的主流技术.

早期掩码防护方案[10-12]默认相应操作按照特定顺序执行,并没有考虑到电路中毛刺现象对防护方案安全性的影响,后续研究[13-14]证实了在这种假设条件下实现的防护方案是不安全的.为了应对毛刺攻击对于防护方案的影响,门限实现防护方案(Threshold Implementation, TI)[15]以及 CHES 2011 中 Prouff 等人的基于安全多方计算的防护方案[16]被相继提出.其中门限防护方案具有以下优点:首先它能够提供可证明的侧信道安全性;其次它可以保证毛刺环境下的侧信道安全性;最后相对于其他具有相似安全性能的防护方案,它具有相对低廉的实现代价.因此门限防护方案自提出以来就得到了学术界更为广泛的研究和应用,其被陆续应用到 AES[17]、PRESENT[18]和 KECCAK[19]等国际标准密码算法中.

然而相对于原始的未防护密码算法,门限实现防护方案仍然会带来较大的资源消耗,因此如何设计安全性与资源消耗平衡的侧信道防护方案是门限防护方案设计中的关键问题.对于具有不同结构特征的算法,如何设计实现代价与安全性兼顾的门限实现侧信道防护方案仍然有待进一步的研究.uBlock 算法[20]是全国密码算法设计竞赛分组密码算法中的一等奖获奖算法,具有良好的安全特性.文献[20]中指出 uBlock 算法 S 盒适用于门限实现防护方案设计,其 S 盒是一个代数次数为 3 的 4 比特输入输出非线性运算,根据文献[21]可以分解为度为 2 的 S 盒,存在一个 3-share 的抗一阶侧信道攻击的无需新随机数的门限实现,但没有给出具体的实现方法.而且一个门限实现方案的防护代价和中间变量划分的 share 个数成正比关系,3-share 的门限实现方法存在着实现代价较大的问题,构造适用于 uBlock 算法的 2-share 的门限实现防护方案可以有效降低防护方案的面积消耗,但是在执行过程中需要添加新的随机数保证方案的安全性,因此如何在降低面积消耗的同时避免随机数的消耗是本文需要解决的关键问题.

针对这一现状,我们首先考虑使用 2-share 的门限实现防护方法降低 uBlock 算法门限防护的面积消耗.在此基础上,通过分解项组合的方式构造不同的门限实现,最终形成了适用于 uBlock 算法的无需添加新随机数的 2-share 门限实现构造方案.本文的主要贡献有以下几点:

(1) 结合 uBlock 加密算法与密钥扩展结构的特点,设计并实现了一种基于串行和流水线思想的 uBlock 算法低代价实现方案,将 S 盒进行了分解,分解为了代数次数为 2 的便于门限防护实现的 2 个 S 盒的级联.

(2) 在 uBlock 低代价实现方案的基础上,设计并实现了 3-share 的无需新随机数的门限实现防护方案,在 FPGA 上进行了实际实现,并通过实验评估了实现代价以及侧信道安全性.

(3) 为了进一步降低实现代价,构造了一种适用于 uBlock 算法的 2-share 门限防护方案,同样无需添加新的随机数,而且其面积消耗相对于 3-share 的无需新随机数的门限防护方案降低了约 30%.然后在 FPGA 上对其进行了实际实现,并通过实验评估了实现代价以及侧信道安全性.

本文第 2 节主要介绍门限实现和 uBlock 算法的相关工作以及基础知识;第 3 节重点论述本文的主要工作,包括 uBlock 算法的串行实现,uBlock 算法的 3-share 的门限防护实现,uBlock 算法的 2-share 的门限防护方案,对于 2-share 门限实现的改进以及对于防护方案的安全性分析;第 4 节在 FPGA 开发板上对于 uBlock 算法门限实现方案进行了侧信道评估实验,并且对于各个方案的实现代价进行了评估和对比;第 5 部分对于本文工作进行了总结.

2　相关工作

2.1　门限实现

门限实现是以安全多方计算以及秘密共享原理

为基础进行构造的一类侧信道防护方法.门限实现原理于 ICICS 2006[15]被提出,主要用于抵抗毛刺现象对侧信道防护方案的影响,这一防护方案陆续应用到 AES 等国际标准算法中[17],验证了其具有有效的安全性以及相对可接受的代价.这个时候的防护方案主要考虑对于一阶侧信道攻击的防护,随着侧信道攻击技术的发展,高阶侧信道攻击的发展威胁着密码设备的安全性,为应对高阶侧信道攻击的威胁,Bilgin 等人在 ASIACRYPT 2014[22]将门限实现原理扩展到高阶的能够抵抗高阶侧信道攻击的情况.

本文主要考虑抵抗一阶侧信道攻击的情况,以 $y=f(x)$ 为例,其实现步骤如下所示:

(1)输入变量的分解

使用布尔掩码将函数输入分解为 S_{in} 份,称之为对输入的 S_{in} -share 划分,为实现对于一阶侧信道攻击的防护,需满足 $S_{in}\geqslant 2$;首先利用 $n\times(S_{in}-1)$ 比特的随机数生成前 $S_{in}-1$ 份输入 $(x_1^1,\dots,x_n^1,\dots,x_1^{S_{in}-1},\dots,x_n^{S_{in}-1})$,其中 n 为输入变量数据位宽,则第 S_{in} 份输入可以由前 $S_{in}-1$ 份输入和原始输入的异或生成,具体可以表示为 $(x_1^{s_{in}},\dots,x_n^{s_{in}})=(x_1\oplus(\oplus_{i=1}^{s_{in}-1}x_1^i),\dots,x_n\oplus(\oplus_{i=1}^{s_{in}-1}x_n^i))$,最终构成需要的掩码输入 $(x_1^1,\dots,x_n^1,\dots,x_1^{S_{in}},\dots,x_n^{S_{in}})$,由于随机数的存在,$S_{in}$ 份输入满足输入均匀性,也就是说每一种可能的输入都是均匀出现的.

(2)中间运算的分解

将步骤(1)中分解好的输入变量带入到目标运算中,可以得到由掩码输入构成的新的函数表达式 $y=f(x_1,\dots,x_n)$,拆分其组成项构成相应的输出函数 $(f_1,\dots,f_{s_{out}})$,其需要满足正确性:$y=\oplus_{j=1}^{S_{out}}f_j$,其中 S_{out} 表示输出函数的个数.另外为了达到安全防护的目的,一阶门限实现需要满足非完备性,即任意一个输出函数都应和至少一个输入份额相互独立.待防护的密码算法往往包含多轮的运算,因此会出现本阶段运算输出作为下一阶段输入的情况,为满足下一阶段的输入均匀性的要求,需要门限实现满足输出均匀性,这个性质称之为函数均匀性.

如以上步骤所示,实现一阶门限实现的安全性需要满足 4 大属性:

(1) 输入均匀性,本属性是实现安全防护的基础,和非完备性共同构成了方案的安全性,只需要使用随机数按照步骤(1)的形式进行构造即可满足;

(2)正确性,本属性是运算正确执行的保障,对于运算进行分解时,不遗漏相应的组成项即可自然满足本属性;

(3) 非完备性,本属性是门限实现安全性的重要保障,这里需要将各个组成项小心地划分到输出函数中,另外需要在非线性运算之间添加寄存器;

(4) 函数均匀性,本属性主要是为了满足下一阶段非线性运算的输入均匀性,在一阶防护中可以通过巧妙的构造使得输出自然满足均匀性.但是在高阶实现中,为了达到高阶安全性的目的,往往采用重掩码的方式实现.

门限实现有着不同的变种,在步骤(1)输入变量分解阶段,针对一阶侧信道防护的情况,按照 S_{in} 设定方法的不同,可以主要分为 $S_{in}=t+1$ [15],以及 $S_{in}=1+1=2$ 两类[23],其中 t 为待防护目标运算的代数次数,两种门限实现方法对比而言,前者相对而言更容易实现函数输出的均匀性,但是它的实现代价也相对较大.

2.2 uBlock 算法

uBlock 算法是一种基于 SPN 结构的分组密码算法,分组长度和密钥长度分别支持 128 和 256 比特,本文主要针对 128 比特的分组长度与密钥长度的加密算法进行研究,其他解密算法以及 256 比特的情况具有相似性.

2.2.1 加密算法

uBlock 加密算法的加密轮数和分组长度有关,在 128 比特分组长度的情况下,其加密轮次是 16 轮.uBlock 算法主要有密钥异或操作、S 盒操作、移位操作以及向量置换操作四种运算,加密算法和密钥扩展算法使用相同的 4 比特 S 盒,S 盒是密码算法中唯一的非线性部件,如表 1 所示.

表 1 uBlock 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

设 128 比特明文输入表示为 (X_0^0,X_1^0),分别为 64 比特.(X_0^i,X_1^i) 表示第 i $(1\leqslant i\leqslant 16)$ 轮状态,需要 16 轮的轮操作,则第 i 轮的轮状态更新需要依次执行如下操作:

(1)密钥异或操作:RK^{i-1} 代表轮密钥,则 $X_0^i||X_1^i=RK^{i-1}\oplus(X_0^{i-1}||X_1^{i-1})$;

(2)S 盒操作:$X_0^i=S_{16}(X_0^i),X_1^i=S_{16}(X_1^i)$,$S_{16}$ 代表 16 个 S 盒操作并置;

(3) 移位操作:$X_1^i=X_1^i\oplus X_0^i$,然后依次执行

32 比特为单位的循环左移操作 $X_0^i = X_0^i \oplus (X_1^i \underset{32}{<<<} 4)$,$X_1^i = X_1^i \oplus (X_0^i \underset{32}{<<<} 8)$,$X_0^i = X_0^i \oplus (X_1^i \underset{32}{<<<} 8)$,$X_1^i = X_1^i \oplus (X_0^i \underset{32}{<<<} 20)$;

(4)向量置换操作:$X_0^i = X_0^i \oplus X_1^i$,$X_0^i = PL(X_0^i)$,$X_1^i = PR(X_1^i)$,$PL$ 表示 8 个字节的向量置换操作 {1,3,4,6,0,2,7,5},PR 表示 8 个字节的向量置换操作 {2,7,5,0,1,6,4,3}.

最终密文为 $RK^{16} \oplus (X_0^{16} \,||\, X_1^{16})$.

2.2.2　密钥扩展算法

128 比特 uBlock 的轮密钥扩展算法需要经过 16 轮的轮操作产生轮密钥,设 K 为 128 比特主密钥,RK^i ($0 \leqslant i \leqslant 16$) 为轮密钥,$K_0^i, K_1^i, K_2^i, K_3^i$ 均为 32 比特数据,则 $RK^0 = (K_0^0 \,||\, K_1^0 \,||\, K_2^0 \,||\, K_3^0) = K$,则第 1 至 16 轮密钥产生需要依次执行如下操作:

(1)向量置换操作:$K_0^i \,||\, K_1^i = PK(K_0^{i-1} \,||\, K_1^{i-1})$,$PK$ 为 16 个半字节的向量置换操作{6,0,8,13,1,15,5,10,4,9,12,2,11,3,7,14};

(2)S 盒操作:$K_2^i = K_2^i \oplus S_8(K_0^i \oplus RC_i)$,其中 S_8 表示 8 个 4 比特 S 盒并置;RC_i 为 32 比特轮常数,由 8 级线性反馈移位寄存器(Linear Feedback Shift Register, LFSR)生成,初始状态为 $c_7c_6c_5c_4c_3c_2c_1c_0 = (00110110)$,当 $j \geqslant 8$,状态更新为 $c_j = c_{j-2} \oplus c_{j-3} \oplus c_{j-7} \oplus c_{j-8}$,$RC_i = a_i \,||\, a'_i \,||\, a''_i \,||\, a'''_i$,$(a_i, a'_i, a''_i, a''')$ 的计算如公式(1)所示;

$$\begin{aligned} a_i &= c_i\bar{c}_{i+1}c_{i+2}c_{i+3}c_{i+4}c_{i+5}c_{i+6}c_{i+7},\\ a'_i &= c_i\bar{c}_{i+1}c_{i+2}\bar{c}_{i+3}c_{i+4}\bar{c}_{i+5}c_{i+6}\bar{c}_{i+7},\\ a''_i &= c_ic_{i+1}c_{i+2}\bar{c}_{i+3}c_{i+4}c_{i+5}c_{i+6}\bar{c}_{i+7},\\ a'''_i &= c_ic_{i+1}c_{i+2}c_{i+3}c_{i+4}\bar{c}_{i+5}c_{i+6}\bar{c}_{i+7} \end{aligned} \tag{1}$$

(3)T 操作:$K_3^i = K_3^i \oplus T(K_1^i)$,其中 T 运算,是半字节为单位与 2 进行不可约多项式为 $m(x) = x^4 + x + 1$ 的有限域 $GF(2^4)$ 上的乘法运算.

第 i 轮的轮密钥为 $RK^i = K_2^i \,||\, K_3^i \,||\, K_1^i \,||\, K_0^i$.

3　方案设计

本部分针对 uBlock 算法目前缺乏侧信道防护的现状,基于门限实现技术对于 uBlock 算法进行了相应的防护研究.

首先针对 uBlock 算法加密算法和密钥扩展算法共用相同 S 盒的特点基于串行和流水线思想设计了 uBlock 算法的串行实现结构.在此基础上设计实现了 uBlock 算法 3-share 的无需新随机数的门限实现防护方案.其次基于面积优化的考虑设计了适用于 2-share 的门限防护方案.最后对于随机数进一步优化设计了适用于 uBlock 算法的 2-share 无需新随机数的门限防护实现方案.此外对于防护方案进行了相应的安全性分析.

3.1　uBlock 串行实现

文献[20]主要考虑了低延时高吞吐量下的算法实现,在资源受限的环境下,需要考虑面积优先的实现方式,此外门限实现侧信道防护方法也会增加算法的实现代价,低代价的算法有利用侧信道防护方案的构造与实现.针对于这种现状,本文设计了一种 uBlock 加密算法和密钥扩展算法相互结合的低代价实现框架,如图 1 所示.具体的,加密算法和密钥扩展共用一个 S 盒实现,也就是说整个算法实现一个 S 盒实例,通过流水线和串行的方式实现对于 S 盒的调用以及整体实现代价的降低.整轮实现需要 41 个时钟周期.整个算法的实现是在 4 比特的数据通路下进行.

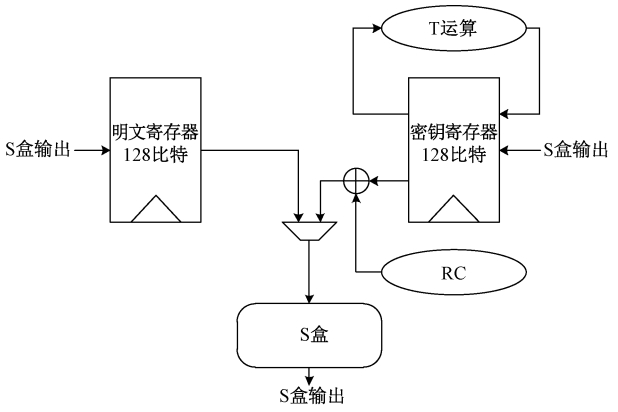


图 1　uBlock 算法串行实现

在加密准备阶段,将 128 比特初始明文输入分别存储在($S_{00}, \cdots, S_{07}, \cdots, S_{30}, \cdots, S_{37}$)等 32 个 4 比特寄存器中,然后执行 16 轮加密轮操作,如图 2 所示,每轮状态寄存器按照如下步骤更新:

(1)密钥异或操作:在第 0 至 31 个时钟周期,状态寄存器进行 4 比特寄存器为单位的左移操作,以 $K_{00} \oplus S_{00}$ 作为 S 盒的输入,其中 K_{00} 是轮密钥寄存器,在下文轮密钥相关部分将进行详细描述;

(2)S 盒操作:第 1 至 32 时钟周期完成 S 盒运算,将 S 盒输出结果赋给 S_{37};

(3)移位操作:第 34 个时钟周期,使用($S_{00} \oplus S_{21}, \cdots, S_{06} \oplus S_{27}$, $S_{07} \oplus S_{20}$)更新($S_{00}, \cdots, S_{07}$),使用($S_{10} \oplus S_{31}, \cdots, S_{16} \oplus S_{37}$, $S_{17} \oplus S_{30}$)更新($S_{10}, \cdots, S_{17}$),完成循环左移 4 比特操作;第 35 个时钟周期,

使用($S_{02}\oplus S_{20},\cdots,S_{07}\oplus S_{25}$, $S_{00}\oplus S_{26}$, $S_{01}\oplus S_{27}$)更新($S_{20},\cdots,S_{27}$),使用($S_{12}\oplus S_{30},\cdots,S_{17}\oplus S_{35}$, $S_{10}\oplus S_{36}$, $S_{11}\oplus S_{37}$)更新($S_{30},\cdots,S_{37}$),完成循环左移 8 比特操作;第 36 个时钟周期,使用($S_{00}\oplus S_{22},\cdots,S_{05}\oplus S_{27}$, $S_{06}\oplus S_{20}$, $S_{07}\oplus S_{21}$)更新($S_{00},\cdots,S_{07}$),使用($S_{10}\oplus S_{32},\cdots,S_{15}\oplus S_{37}$, $S_{16}\oplus S_{30}$, $S_{17}\oplus S_{31}$)更新($S_{10},\cdots,S_{17}$),完成循环左移 8 比特操作;在第 37 个时钟周期,使用($S_{20}\oplus S_{05}$, $S_{21}\oplus S_{06}$, $S_{22}\oplus S_{07}$, $S_{23}\oplus S_{00},\cdots,S_{27}\oplus S_{04}$)更新($S_{20},\cdots,S_{27}$),使用($S_{30}\oplus S_{15}$, $S_{31}\oplus S_{16}$, $S_{32}\oplus S_{17}$, $S_{33}\oplus S_{10},\cdots$, $S_{37}\oplus S_{14}$)更新($S_{30},\cdots,S_{37}$),完成循环左移 20 比特操作;

(4) 向量置换操作:在第 38 个时钟周期,以 4 比特的寄存器为单位,对($S_{00},\cdots,S_{37}$)进行向量置换操作.

执行 16 轮后的状态寄存器($S_{00},\cdots,S_{37}$)与 RK_{16} 进行异或运算,即可产生最后的密文输出.

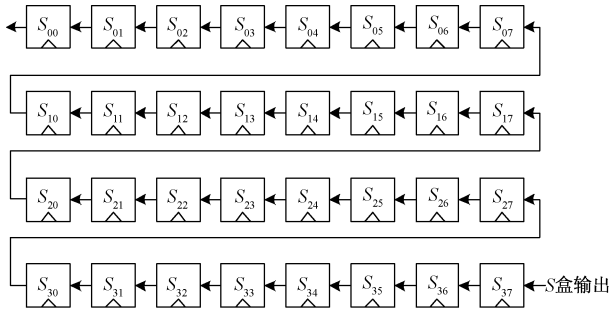


图 2 uBlock 明文寄存器

密钥扩展运算和加密运算交叉进行,生成所需轮密钥.轮密钥寄存器类似于状态寄存器,初始时主密钥存入($K_{00},\cdots,K_{07},\cdots,K_{30},\cdots,K_{37}$)等 28 个 4 比特寄存器中,然后进行 16 轮轮密钥更新操作,如图 3 所示,每轮的密钥状态寄存器按如下步骤进行更新:

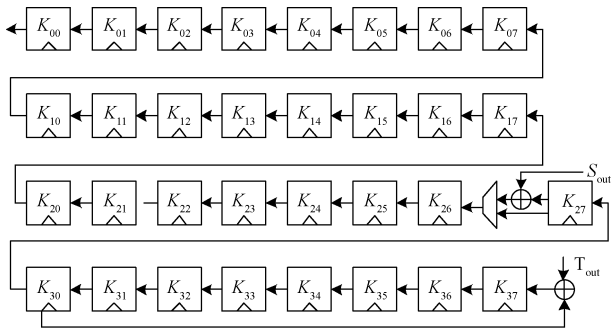


图 3 uBlock 密钥寄存器

0 到 30 个时钟周期,轮密钥寄存器进行 4 比特寄存器为单位的循环左移操作,K_{00} 参与加密运算的密钥异或操作;

(1)向量置换操作:在第 31 个时钟,轮密钥寄存器进行 4 比特寄存器为单位的左移操作,与此同时,($K_{00},\cdots,K_{17}$)进行 4 比特为单位的置换操作,置换操作与移位操作相互结合;

(2) S 盒操作:在第 32 到 39 个时钟,($K_{00},\cdots,K_{07}$),($K_{10},\cdots,K_{17}$),($K_{20},\cdots,K_{27}$),($K_{30},\cdots,K_{37}$)分别进行以 4 比特寄存器为单位的循环移位操作,将 RC$\oplus K_{00}$ 赋给 S 盒运算,进行 S 盒运算,其中 RC 使用 LFSR 进行连续 8 个时钟周期的更新,输出 4 比特所需常数值;在第 33 到 39 个时钟,产生 S 盒操作输出 S_{out},以 $K_{27}\oplus S_{out}$ 更新 K_{26},第 40 个时钟周期,以 $K_{27}\oplus S_{out}$ 更新 K_{07},完成 S 盒操作,具体的 S 盒运算公式将在下文详细描述;

(3)T 操作:在第 32 到 39 个时钟,K_{10} 作为输入进行 T 操作,在 32 到 39 个时钟,产生 T 操作输出 T_{out},使用 $K_{30}\oplus T_{out}$ 更新 K_{37},完成 T 操作,具体 T 操作运算公式将在下文详细描述.

第 40 轮,以($K_{20},\cdots,K_{27}\oplus S_{out}$),($K_{30},\cdots,K_{37}$),($K_{10},\cdots,K_{17}$),($K_{00},\cdots,K_{07}$)更新($K_{00},\cdots,K_{37}$)组成轮密钥输出,完成一轮的计算.

其中 uBlock 算法的 S 盒可以看作为代数度为 3 的 4 比特输入,4 比特输出的函数,可以分解为度为 2 的两个 S 盒 G,F 的复合,S=G(F(x)),如图 4 所示.

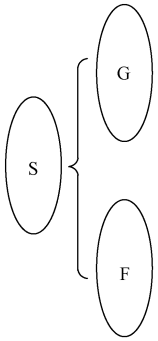


图 4 S 盒分解示意图

将 S 盒分解为两个代数度为 2 的 S 盒级联有利于接下来的侧信道防护方案设计.F,G 两个 S 盒的代数表达如公式(2)和(3)所示,其中 $x=(x_3,x_2,x_1,x_0)$,x_3 代表 S 盒输入 x 的最高比特位:

$$\begin{aligned} F(x_3,x_2,x_1,x_0) &= (f_3,f_2,f_1,f_0),\\ f_3 &= x_1\oplus x_2\oplus x_3\oplus x_1x_2,\\ f_2 &= x_2,\\ f_1 &= x_1,\\ f_0 &= 1\oplus x_0\oplus x_2x_3 \end{aligned} \tag{2}$$

$$\begin{aligned} G(f_3,f_2,f_1,f_0) &= (g_3,g_2,g_1,g_0),\\ g_3 &= f_3,\\ g_2 &= 1\oplus f_2\oplus f_0f_1,\\ g_1 &= f_0\oplus f_1\oplus f_3\oplus f_0f_3,\\ g_0 &= f_0 \end{aligned} \tag{3}$$

其中 T 运算是以 $m(x)=x^4+x+1$ 为不可约多项式的有限域 $GF(2^4)$ 上的乘 2 运算,如公式(4)所示,运算可以表示为:$T_{out}=T(t_{in})$,$t_{in}=(x_3,x_2,x_1,x_0)$,x_3 代表 t_{in} 的最高比特位.

$$\begin{aligned} T(x_3,x_2,x_1,x_0) &= (t_3,t_2,t_1,t_0)=T_{out},\\ t_3 &= x_2,\\ t_2 &= x_1,\\ t_1 &= x_0\oplus x_3,\\ t_0 &= x_3 \end{aligned} \tag{4}$$

3.2　uBlock 3-share 门限实现

对于 uBlock 算法,为抵抗一阶侧信道攻击,设计了 3-share 的门限实现方案,其具有 3 输入,3 输出,也被称为(3,3)－TI.(3,3)－TI 需要将输入变量随机化为三个掩码分量,整个算法的实现是在 12 比特的数据通路下进行,也就是说 12 比特的寄存器存储着原始 4 比特输入值的三个掩码分量.算法执行的时序逻辑与无防护情况下的时序逻辑框架基本相同,其他不同之处主要在于 S 盒运算部分和 T 运算部分添加了相应的防护,以及 RC 运算参加密钥扩展运算的方式.

常数项 RC:以 8 级的 LFSR 进行计算,在第 32～39 个时钟周期产出所需的 4 比特 RC 值,异或 12 比特 K_{00} 其中半个字节作为 S 盒的输入,参与密钥扩展运算.

T 操作:在第 32-39 个时钟周期,进行 T 运算.以 $(T_{in1},T_{in2},T_{in3})$ 作为 T 运算的输入分量,满足 $T_{in}=T_{in1}\oplus T_{in2}\oplus T_{in3}$,T 运算的门限实现分函数和原始无防护实现的 T 运算相同,$(T_{out1},T_{out2},T_{out3})$ 代表运算的输出分量,则其门限实现如公式(5)所示.

$$\begin{aligned} T_{out1} &= T(T_{in1}),\\ T_{out2} &= T(T_{in2}),\\ T_{out3} &= T(T_{in3}) \end{aligned} \tag{5}$$

S 盒操作:相比于串行无防护实现,3-share uBlock 算法的门限实现最大的不同在于 S 盒的构造实现.其中 S 盒的门限实现运算如图 5 所示,以 (In_1,In_2,In_3) 作为 S 盒输入分量,满足 $x=In_1\oplus In_2\oplus In_3$,以 F' 表示 F S-box 的门限实现分函数,以 (f'_3,f'_2,f'_1) 表示其输出分量,G' 表示 G S-box 的门限实现分函数,(Out_1,Out_2,Out_3) 表示 S-box 的输出分量,如 2.1 节门限实现的要求,门限实现要求各个分函数和原始中间状态相互独立,每个分函数都只有两个分量参与运算,和第三个输入分量保持独立,从而保证和中间状态的相互独立性,则其门限实现如公式(6)和(7)所示.

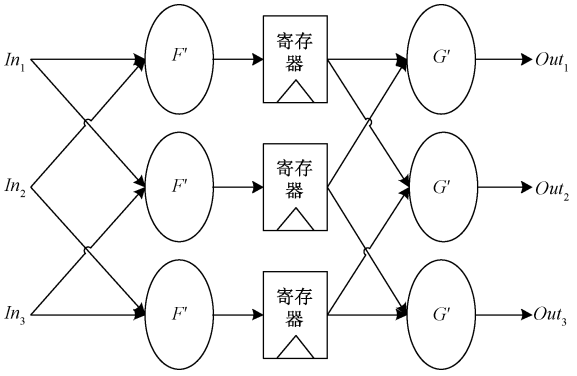


图 5　S 盒 3-share 结构图

$$\begin{aligned} f'_1 &= F'(In_1,In_2),\\ f'_2 &= F'(In_1,In_3),\\ f'_3 &= F'(In_2,In_3) \end{aligned} \tag{6}$$

$$\begin{aligned} Out_1 &= G'(f'_1,f'_2),\\ Out_2 &= G'(f'_1,f'_3),\\ Out_3 &= G'(f'_2,f'_3) \end{aligned} \tag{7}$$

以 (x_3,x_2,x_1,x_0),(y_3,y_2,y_1,y_0) 代表函数 F',G' 的两个 4 比特输入,x_3,y_3 分别代表输入的最高比特位,则函数表达式如公式(8)和(9)所示.

$$\begin{aligned} &F'(x_3,x_2,x_1,x_0,y_3,y_2,y_1,y_0)=(f_3,f_2,f_1,f_0),\\ &f_3=x_2\oplus x_3\oplus x_1x_2\oplus x_1y_2\oplus y_1x_2,\\ &f_2=x_2,\\ &f_1=x_1,\\ &f_0=1\oplus x_0\oplus x_2x_3\oplus x_2y_3\oplus y_2x_3 \end{aligned} \tag{8}$$

$$\begin{aligned} &G'(x_3,x_2,x_1,x_0,y_3,y_2,y_1,y_0)=(g_3,g_2,g_1,g_0),\\ &g_3=x_3,\\ &g_2=1\oplus x_2\oplus x_0x_1\oplus x_0y_1\oplus y_0x_1,\\ &g_1=x_0\oplus x_1\oplus x_3\oplus x_0x_3\oplus x_0y_3\oplus y_0x_3,\\ &g_0=x_0 \end{aligned} \tag{9}$$

3.3　uBlock 2-share 门限实现

(3,3)－TI 需要将输入变量掩码为 3 份随机分量,需要寄存器增加到无防护实现的 3 倍,面积消耗较大,为节省面积消耗,本节构造了 2-share 的门限实现方案,其需要的寄存器约为无防护实现的 2 倍,具有 2 输入,4 输出,被称为(2,4)－TI 的实现方

式.整个算法的实现是在 8 比特的数据通路下进行,也就是说 8 比特的寄存器存储着原始 4 比特输入值的两个掩码分量.算法执行的逻辑与(3,3)－TI 情况下的逻辑框架基本相同,不同之处主要在于 S 盒运算部分,相比于无防护情况,有一个从 4-share 压缩到 2-share 的过程,为防止毛刺现象带来安全泄露,需要使用寄存器对 4-share 输出进行暂存,然后压缩为 2-share,因此相比于 3-share 的门限实现多使用了一个时钟周期.此外对于 T 运算的防护也从 3-share 变为 2-share,其具体的实现过程如下所述.

首先加密过程中 S 盒操作涉及的寄存器状态更新描述如下:

S 盒操作(状态寄存器):在 0 到 31 时钟,$K_{00}\oplus S_{00}$ 作为 S 盒输入,在此期间$(K_{00},\cdots,K_{37})$做循环左移运算,$(S_{00},\cdots,S_{37})$做左移运算;在 1 到 32 时钟,将 S 盒 2-share 输出赋值给 S_{37},另外 2-share 输出使用寄存器进行暂存,$(S_{00},\cdots,S_{37})$做左移运算;在 2 到 32 时钟,将 S_{37} 与之前暂存的 2-share 相互异或赋给 S_{36};在 33 时钟,S_{37} 异或之前暂存的 2-share S 盒运算输出并参与之后的线性运算.

S 盒操作(密钥扩展寄存器):在第 32-39 个时钟周期,RC 异或 K_{00} 其中半个字节作为 S 盒的输入,在此期间,$(K_{00},\cdots,K_{07})$进行以 8 比特的寄存器为单位的循环左移操作以保障 K_{00} 为 S 盒运算所需的数据.在 33-40 个时钟周期,将 S 盒其中 2-share 输出参与下一阶段的异或运算,另外 2-share S 盒输出使用寄存器暂存一个时钟周期后,在 34-41 个时钟周期异或到前两个 share 中去.

其中 S 盒的运算如下所述.在 2-share 的 S 盒实现中,和 3-share 的最大区别是非线性运算的比特位与只有线性运算的比特位的门限实现方式不同,需要按比特分别进行处理.对于分解出来的两个 S-box G 和 F 的门限实现方式是类似的,这里只介绍一次项以及二次项的分解方式,F 函数以及 G 函数均由一次项以及二次项组成,其门限实现可以类比构造.只有线性运算比特位的运算是 2-share 的输入,2-share 的输出,较为简单,以 $f(x)=x\oplus 1$ 为例,输入掩码为 (x_1,x_2),满足 $x=x_1\oplus x_2$,(f_1,f_2)代表 2-share 掩码输出,其具体表示如公式(10)所示.

$$\begin{aligned} f(x_2,x_1)&=(f_2,f_1),\\ f_2&=x_1\oplus 1,\\ f_1&=x_2 \end{aligned} \tag{10}$$

而带有非线性运算的比特位,其是 2-share 的输入,4-share 的输出,需要先将 4-share 的输出使用寄存器暂存,然后压缩为 2-share 以供下一阶段的输入使用,这里输出的 2-share 需要满足均匀性.以 $f(a,b)=ab$ 为例,其非线性运算的门限实现方式如图 6 所示,而带有非线性运算的比特位中线性运算的 2-share 异或到 4-share 输出的任意两项即可.

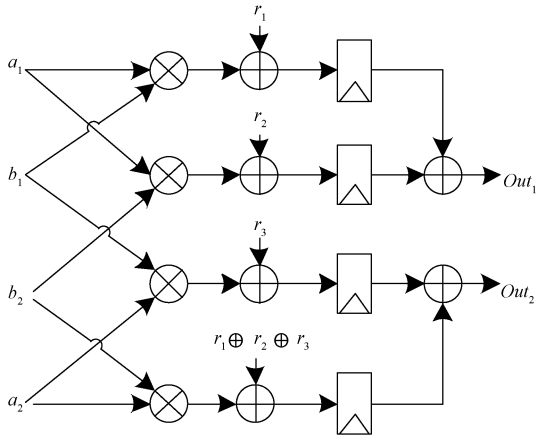


图 6 2-share S 盒实现示意图

假设非线性运算 $f(a,b)=ab$ 的输入掩码为 $a=a_1\oplus a_2$,$b=b_1\oplus b_2$,(f_1,f_2,f_3,f_4)为 4-share 输出,这里使用随机数(r_1,r_2,r_3)对输出进行重掩码,然后使用寄存器暂存后压缩为 2-share 输出,使得满足均匀性同时保持毛刺环境下安全性,(Out_1,Out_2)是输出的 2-share,则其门限实现如公式(11)和(12)所示.

$$\begin{aligned} &f(a_2,a_1,b_2,b_1)=(f_4,f_3,f_2,f_1),\\ &f_4=r_3\oplus a_1b_1,\\ &f_3=r_2\oplus a_1b_2,\\ &f_2=r_1\oplus a_2b_1\\ &f_1=r_1\oplus r_2\oplus r_3\oplus a_2b_2, \end{aligned} \tag{11}$$

$$\begin{aligned} Out_1&=f_4\oplus f_3,\\ Out_2&=f_2\oplus f_1 \end{aligned} \tag{12}$$

T 操作:在第 32-39 个时钟周期,进行 T 运算.以 (T_{in1},T_{in2}) 为 T 运算的输入分量,满足 $T_{in}=T_{in1}\oplus T_{in2}$,T 运算的门限实现分函数和原始无防护实现的 T 运算相同,(T_{out1},T_{out2})代表运算的输出分量,则其门限实现如公式(13)所示.

$$\begin{aligned} T_{out1}&=T(T_{in1}),\\ T_{out2}&=T(T_{in2}) \end{aligned} \tag{13}$$

3.4 2-share 门限实现的改进

(2,4)－TI 掩码实现相对于(3,3)－TI 的面积消耗有所降低,但是在执行过程中需要添加新的随机数,这也是不小的资源消耗.为解决这一问题,在(2,4)－TI 防护方案的基础上,借鉴文献[23]的思

想,可以通过巧妙地构造,使得压缩过程中的一次项的掩码分量对于二次项的掩码分量形成掩码的效果,从而使得 S 盒门限实现在无需增加新的随机数情况下即可满足函数输出的均匀性,从而实现无新随机数的 2-share 门限实现防护.

不同于之前的设计中,主要考虑非线性部分的门限实现方式,通过随机数重掩码的方式实现门限输出的均匀性,这里将线性运算和非线性运算结合起来进行考虑,可以构造出无需新随机数的但是满足输出均匀性的门限实现方式,以函数 $f(a,b)=ab+c$ 为例,其门限实现的构造方式如图 7 所示.

非线性运算输入的掩码分量分别为 (a_1,a_2),(b_1,b_2),(c_1,c_2),满足 $a=a_1\oplus a_2$,$b=b_1\oplus b_2$,$c=c_1\oplus c_2$,其中输入 c 的两个掩码分量起到了重掩码的作用,(f_1,f_2,f_3,f_4) 为 4-share 输出,使用寄存器暂存后按照下面公式的方式压缩为 2-share 输出,从而使得 2-share 输出分别包含 c_1 和 c_2,从而满足均匀性,Out_1 和 Out_2 是输出的 2-share,则其门限实现如公式(14)和(15)所示.

$$\begin{aligned}&f(a_2,a_1,b_2,b_1)=(f_4,f_3,f_2,f_1),\\&f_4=c_1\oplus a_1b_1,\\&f_3=a_1b_2,\\&f_2=a_2b_1,\\&f_1=c_2\oplus a_2b_2\end{aligned}\tag{14}$$

$$\begin{aligned}Out_1&=f_4\oplus f_3,\\Out_2&=f_2\oplus f_1\end{aligned}\tag{15}$$

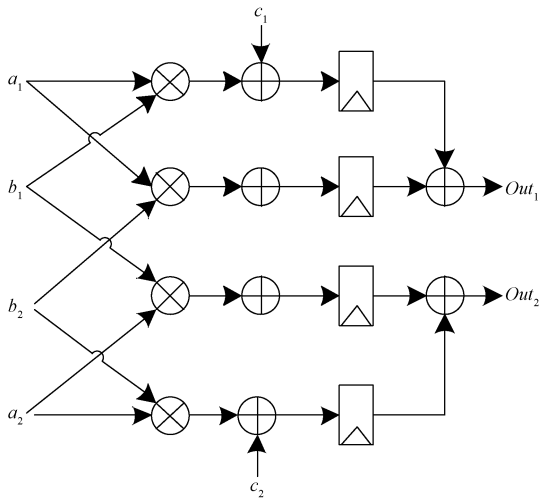


图 7　S 盒 2-share 均匀门限实现

3.5　安全性分析

本文基于门限实现原理针对 uBlock 算法进行侧信道防护的方案设计,对于侧信道攻击的防护能力由门限实现原理保证. 运算输入的均匀性和门限实现非完备性可以保障已防护算法的中间值和原始输入保持相互独立的关系,运算输出的均匀性保证了密码运算中级联时门限实现的安全性. 通过破坏侧信道攻击的前提,也就是运行算法的侧信道信息和运行算法的中间值要具有相关性,从而实现了对于一阶侧信道攻击的防护,具体针对每个实现方案的安全性分析如下所述.

首先是 3-share 的门限实现安全性分析.

(1)均匀性:明文输入和密钥输入都使用随机数掩码划分为了 3-share,满足输入均匀性;

(2)正确性:线性运算如 T 操作,$T_{out}=T_{out1}\oplus T_{out2}\oplus T_{out3}$,满足正确性;非线性操作也就是 S 盒,公式(6)中输出分量 (f'_3,f'_2,f'_1) 相互异或等于 F 盒原始输出,公式(7)中 S 盒输出分量 (Out_1,Out_2,Out_3) 相互异或等于原始 S 盒输出,满足正确性;

(3)非完备性:线性操作如 T 操作,每个分函数只涉及一个掩码分量,和另外两个掩码分量相互独立,因此和原始中间值相互独立;非线性操作中,如公式(8)中 F 盒操作的分函数 F' 以及公式(9)中 G 盒的分函数 G' 涉及均只涉及两个掩码分量,和另外一个掩码分量相互独立,因此和原始中间值相互独立,而且非线性运算之间均有寄存器分隔,可抵抗毛刺带来的安全性泄露;综上所述经过防护后的中间值与原始中间值相互独立,因此满足一阶侧信道安全性;

(4)函数均匀性:线性运算天然满足函数均匀性,非线性运算如公式(6)和(7)所示,输出的 (f'_3,f'_2,f'_1),(Out_1,Out_2,Out_3) 可验证均满足均匀性,每种可能的取值都均匀出现,作为下一阶段运算的输入时满足输入均匀性,因此整个防护实现的安全性都满足四大属性,实现了一阶侧信道安全防护.

其次是 2-share 的带随机数门限实现安全性分析.

(1)均匀性:明文输入和密钥输入都使用随机数掩码划分为了 2-share,满足输入均匀性;

(2)正确性:线性运算如 T 操作,$T_{out}=T_{out1}\oplus T_{out2}$,满足正确性;非线性操作也就是 S 盒,其线性组成项如公式(10)所示,其非线性组成项如公式(11)和(12)所示,结果分量相互异或等于原始输出,G 盒函数的门限实现类似,同样满足正确性;

(3)非完备性:线性操作如 T 操作,每个分函数只涉及一个掩码分量,和另外一个掩码分量相互独立,因此和原始中间值相互独立;非线性操作中,如公式(11)所示,F 盒操作的分函数 (f_1,f_2,f_3,f_4)

均只涉及输入的一个掩码分量，和另外一个掩码分量相互独立，因此和原始中间值相互独立，而且非线性运算之间均有寄存器分隔，可抵抗毛刺带来的安全性泄露.综上所述经过防护后的中间值与原始中间值相互独立，因此满足一阶侧信道安全性；

(4)函数均匀性：线性运算的均匀性天然满足，非线性运算如公式(11)和(12)所示，输出的(Out_1，Out_2)由于添加的随机数，满足均匀性，即每种可能的取值都均匀出现，因此作为下一阶段运算的输入时满足输入均匀性，因此整个防护实现的安全性都满足四大属性，实现了一阶侧信道安全防护.

最后是 2-share 无需新随机数的门限实现安全性分析.

(1)均匀性、正确性以及非完备性：类似于 2-share 的带随机数的门限实现，三个性质可以满足；

(2)函数均匀性：线性运算的均匀性天然满足，非线性运算如公式(14)和(15)所示，输出的(Out_1，Out_2)由于 c_1 和 c_2 线性掩码分量起到了类似于随机数掩码的作用，使得输出的 2-share 满足均匀性，即每种可能的取值都均匀出现，因此作为下一阶段运算的输入时满足输入均匀性.综上所述整个防护实现的安全性都满足四大属性，实现了一阶侧信道安全防护.

4 实 验

4.1 安全性评估实验

本文在 SAKURA-X 评估板上实现了我们的 uBlock 算法侧信道防护方案，SAKURA-X 是一种验证侧信道防护方案的专用开发板，主要包含两个 FPGA，一个是评估 FPGA，用于加载待测方案；另一个是控制 FPGA，用来控制相关通信.本文通过采集防护方案执行过程中能量信息的方式验证侧信道防护效果.

本文采集能量曲线的实验设置如下：1、为了降低评估 FPGA 中的噪声，提高能量曲线的信噪比，本文将随机数发生器和 uBlock 算法的门限实现分别在控制 FPGA 和评估 FPGA 中进行实现；2、为了能够更加精确地采集待测曲线，我们使得 uBlock 算法门限实现工作在 375kHz 低频时钟下.总体来讲，本文的实验设置有利于攻击者实现侧信道攻击，防护方案的安全性在便于攻击者的实验环境中如果得到验证，那么其安全性也将适用于其他应用场景.

本文采用基于 t-test 的测试向量泄露评估技术(Test Vector Leakage Assessment，TVLA)[24]对 uBlock 算法门限防护方案的侧信道安全性进行评估.TVLA 技术是一种从信息泄露角度评估防护方案侧信道安全性的通用评估技术.在具体的评估实验中，首先收集两组分别是固定输入和随机输入的能量曲线，然后通过 t-test 评估两组曲线分布的均值是否有差异来判断是否有泄露的产生.本文选择正负 4.5 作为统计量的阈值，其置信度大于 99.999%，当统计值超过阈值时，则表明其存在泄露点，相应的防护方案可能存在侧信道防护缺陷，当其统计值低于阈值时，则表明通过了测试.

本文使用 32MHz 的采集频率对于待测防护方案进行能量曲线的采集，进行了无防护情况下(关闭随机数发生器)以及有防护情况下(打开随机数发生器)的评估对比实验，验证设备的运行状况.图 8 表示未添加防护时采集 1 万条能量曲线的一阶 TVLA 结果，图中横轴表示 uBlock 算法执行过程中不同时刻采集到的样本点，从左到右表示 uBlock 算法从开始执行到执行结束，纵轴代表每个样本点对应的 TVLA 测试的 t-test 统计量，图中统计量超过了 4.5，表示在未添加防护时存在着相应的侧信道泄露.图 9 表示添加了防护后采集 1 万条能量曲线的一阶 TVLA 结果，没有了泄露，符合预期，验证了设备运行状况正常.

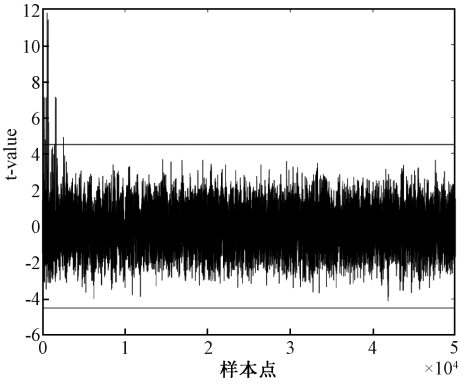


图 8 一阶 TVLA 结果(1 万曲线量，无防护)

2-share 无需新随机数的 uBlock 算法门限防护方案侧信道评估结果如图 10，11，12，13 所示.其中图 10、图 11 表示固定值选取为全 0 时的 TVLA 评估结果.图 10 表示添加了防护后采集 100 万条能量曲线一阶 TVLA 测试结果，统计量未超过 4.5，表明防护方案起到了相应的防护效果；然而对于评估实验曲线量是否足够可能存在疑问，图 11 为二阶 TVLA 测试结果，统计量超过了 4.5，表明进行防护方案存在着二阶的侧信道泄露，符合预期，证明了本

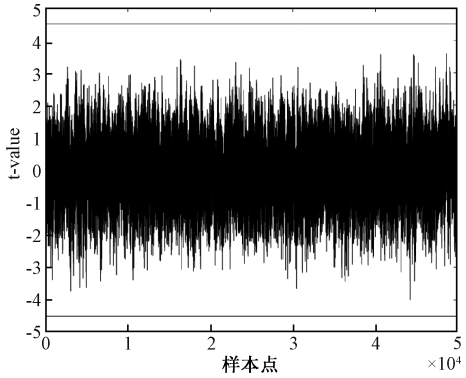


图 9　一阶 TVLA 结果(1 万曲线量,有防护)

次 TVLA 实现中采集的曲线量是足够的,若有一阶泄露存在,那么应可以检测到.在图 12,13 为排除不同固定值选取带来的安全评估误报差而进行的另一组实验,固定值输入从全 0 改为随机选取的一个固定值,同样进行了 100 万条曲线量的一阶、二阶 TVLA 测试,结果同样符合预期.综上所述,实验结果验证了 uBlock 算法门限实现方案一阶侧信道安全防护的有效性.

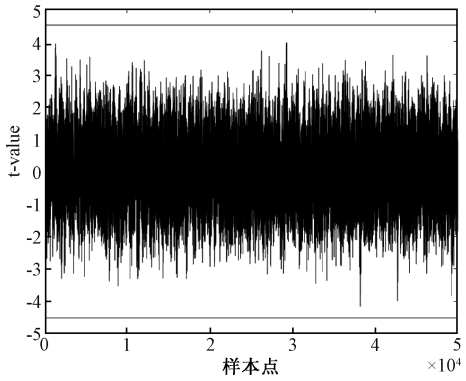


图 10　一阶 TVLA 结果(100 万曲线量,全 0 固定值)

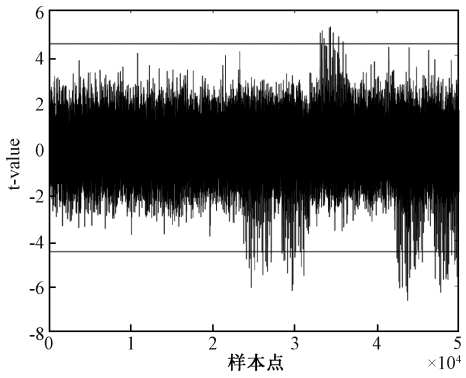


图 11　二阶 TVLA 结果(100 万曲线量,全 0 固定值)

3-share uBlock 算法门限防护方案侧信道评估结果如图 14,15,16,17 所示,其中图 14,15 为固定值输入为全 0 时的实验结果,图 16,17 为固定值输入为随机选取的一个值时的实验结果.图 14 表示采

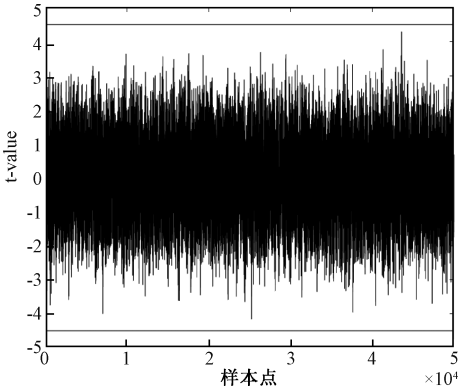


图 12　一阶 TVLA 结果(100 万曲线量,随机固定值)

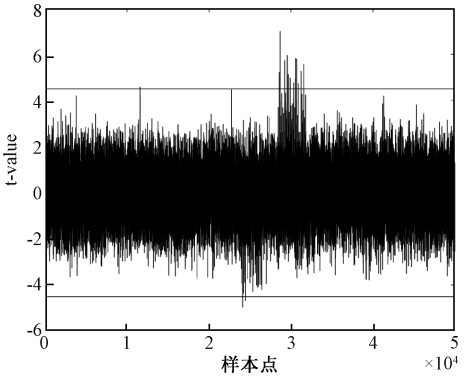


图 13　二阶 TVLA 结果(100 万曲线量,随机固定值)

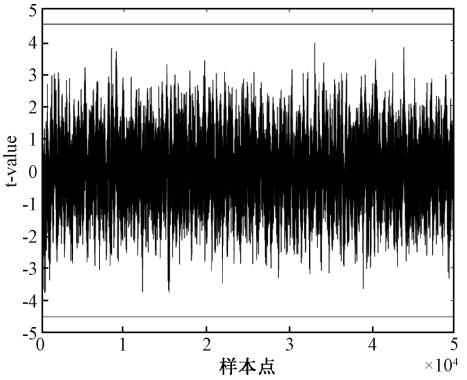


图 14　一阶 TVLA 结果(500 万曲线量,全 0 固定值)

集 500 万条能量曲线一阶 TVLA 测试结果,统计量未超过 4.5,表明防护方案起到了相应的防护效果;图 15 表示添加了防护后的采集 500 万条能量曲线二阶 TVLA 测试结果,统计量超过了 4.5,表明防护方案存在着二阶的侧信道泄露,符合预期,证明了本次 TVLA 实现中采集的曲线量是足够的.图 16,17 的一阶、二阶测试结果同样符合预期,排除了特定固定值可能带来的实验评估误差.综上所述,实验结果验证了 uBlock 算法 3-share 门限实现方案一阶侧信道安全防护的有效性.

综上所述,2-share 的无需新随机数的 uBlock

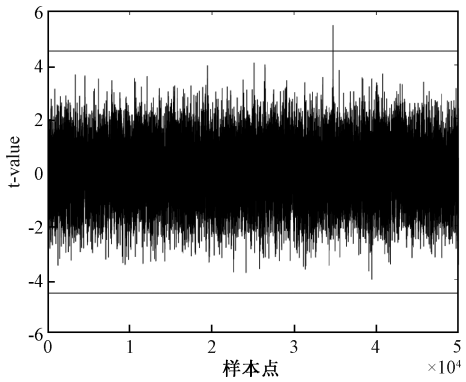


图 15 二阶 TVLA 结果(500 万曲线量,全 0 固定值)

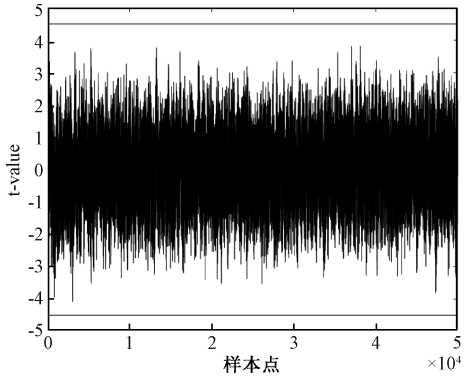


图 16 一阶 TVLA 结果(500 万曲线量,随机固定值)

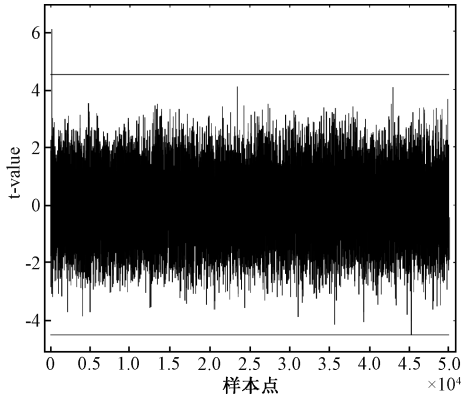


图 17 二阶 TVLA 结果(500 万曲线量,随机固定值)

防护方案和 3-share 的 uBlock 防护方案都具有一阶侧信道防护安全性.此外,虽然二者均不能实现二阶侧信道安全性,但是 3-share 防护方案检测到二阶侧信道泄露需要更多的曲线量,3-share 防护方案相对于 2-share 方案更不易被二阶侧信道攻击,当然相对应的 3-share 防护方案代价更大.

4.2 代价评估实验

在本节中,从不同角度对上述方案的实现代价进行了评估和对比.

表 2 中对串行无防护的原始方案、3-share 的门限实现防护方案、2-share 的需要新随机数的门限实现方案以及最终的 2-share 无需新随机数的门限实现方案的实现代价进行了相应的对比.在对比结果中,第 2 列显示,优化后 2-share TI 相对于 3-share-TI 寄存器消耗降低 1/3;第 3 列显示,初始阶段明文密钥随机掩码所需的随机数降低 1/2;第 4 列表示算法执行过程中保证函数均匀性所需新添加的随机数,优化后 2-share TI 在加密过程中和 3-share TI 同样无需添加新的随机数,而优化之前的 2-share TI 每执行一次 S 盒操作需要消耗 12 比特的随机数;第 5 列表示 2-share 的 TI 相对于串行无防护实现以及 3-share TI 需要多消耗 1 个时钟周期,相对来说时钟周期的增加可以忽略.综上所述,改进后的无需新随机数 2-share TI 的实现代价有明显降低.

表 2 uBlock 实现资源消耗对比表

项目	明文密钥寄存器	明文密钥掩码随机数	新随机数/S 盒	时钟周期/轮
串行无防护	128 * 2	0	0	41
3-share TI	128 * 6	128 * 4	0	41
2-share TI	128 * 4	128 * 2	12	42
无新随机数的 2-share TI	128 * 4	128 * 2	0	42

表 3 为各个 uBlock 算法实现方案在 Xilinx Kintex7 XC7K160T 型号 FPGA 上进行实现,在 Xilinx 公司 ISE 14.7 集成开发工具上评估得到的资源消耗情况,从 Slice、Reg 以及 LUT 消耗情况均可以看出无新随机数 2-share TI 资源消耗最少.

表 3 uBlock FPGA 资源消耗对比表

项目	Slice	Reg	LUT
串行无防护	336	271	934
3-share TI	651	780	1981
2-share TI	496	528	1561
无新随机数 2-share TI	494	528	1561

表 4 中使用 Synopsys 2016.03 在 NanGate 45nm 公开元件库下对各个 uBlock 算法方案的面积消耗情况进行了相应的评估.评估均显示,无新随机数 2-share TI 方案在面积占用上达到了最低,相对于 3-share TI 实现降低了约 30%.综上所述,本文实现了一种针对 uBlock 算法的资源消耗较少的一阶侧信道防护方案.

表 4 uBlock 面积消耗对比表

项目	面积消耗(GE)	面积比(相对无防护)
串行无防护	3646.8	1
3-share TI	10279.0	2.818
2-share TI	7285.5	1.998
无新随机数 2-share TI	7243.4	1.986

5　结束语

针对 uBlock 算法在硬件低代价实现方面研究的不足,本文首先结合串行实现的思想与流水线的思想构造并实现了一种适用于资源受限环境下的低代价硬件实现方案;其次在此基础上构造并实现了 3-share 的无需新随机数的门限实现方法;最后进一步优化了门限实现的面积消耗,实现了一种 2-share 的需要新随机数的门限实现方案,并且通过优化组合构造了一种适用于 uBlock 算法的 2-share 的无需新随机数的门限实现方案.在文章的第 4 部分,本文评估了各种方案的实现代价,并且验证了门限实现方案的侧信道防护效果.本文主要针对 uBlock 算法的一阶侧信道防护需求进行了相应的门限方案设计,未来计划设计并实现适用于 uBlock 算法的高阶门限实现方案,进一步提高 uBlock 算法抵抗侧信道攻击的能力.

参　考　文　献

[1] Kocher P C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems//Proceedings of the 16th Annual International Cryptology Conference. Santa Barbara, USA, 1996: 104-113

[2] Messerges T S. Using second-order power analysis to attack DPA resistant software//Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Worcester, USA, 2000: 238-251

[3] Ferrigno J, Hlaváč M. When AES blinks: introducing optical side channel. IET Information Security, 2008, 2(3): 94-98

[4] Boneh D, DeMillo R A, Lipton R J. On the importance of checking cryptographic protocols for faults//Proceedings of International Conference on Theory and Application of Cryptographic. Konstanz, Germany, 1997:37-51

[5] Gornik A, Stoychev I, Jürgen Oehm. A novel circuit design methodology to reduce side channel leakage//Proceedings of International Conference on Security, Privacy, and Applied Cryptography Engineering. Chennai, India, 2012:1-15

[6] Tiri K, Akmal M, Verbauwhede I. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards//Proceedings of the 28th European Solid-State Circuits Conference. Florence, Italy, 2002: 403-406

[7] Tiri K, Verbauwhede I. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation//Proceedings of the Design, Automation and Test in Europe Conference and Exhibition. Paris, France, 2004: 246-251

[8] Suzuki D, Saeki M. Security evaluation of DPA countermeasures using dual-rail pre-charge logic style//Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Yokohama, Japan, 2006: 255-269

[9] Tiri K, Hwang D, Hodjat A, et al. Prototype IC with WDDL and differential routing-DPA resistance assessment//Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Edinburgh, UK, 2005: 354-365

[10] Chari S, Jutla C S, Rao J R, et al. Towards sound approaches to counteract power-analysis attacks//Proceedings of the 19th Annual International Cryptology Conference. Santa Barbara, USA, 1999: 398-412

[11] Ishai Y, Sahai A, Wagner D. Private circuits: Securing hardware against probing attacks//Proceedings of the 23th Annual International Cryptology Conference. Santa Barbara, USA, 2003: 463-481

[12] Trichina E, Korkishko T, Lee K H. Small size, low power, side channel-immune AES coprocessor: design and synthesis results//Proceedings of International Conference on Advanced Encryption Standard. Bonn, Germany, 2004: 113-127

[13] Oswald E, Mangard S, Pramstaller N, et al. A side-channel analysis resistant description of the AES S-box//Proceedings of International Workshop on Fast Software Encryption. Paris, France, 2005: 413-423

[14] Canright D, Batina L. A very compact "perfectly masked" S-box for AES//Proceedings of International Conference on Applied Cryptography and Network Security. New York, USA, 2008: 446-459

[15] Nikova S, Rechberger C, Rijmen V. Threshold implementations against side-channel attacks and glitches//Proceedings of International Conference on Information and Communications Security. Raleigh, USA, 2006:529-545

[16] Prouff E., Roche T. Higher-order glitches free implementation of the AES using secure multi-party computation protocols//Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Nara, Japan, 2011:63-78

[17] Bilgin B, Gierlichs B, Nikova S, et al. Trade-offs for threshold implementations illustrated on AES. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015, 34(7): 1188-1200

[18] Poschmann A, Moradi A, Khoo K, et al. Side-channel resistant crypto for less than 2,300 GE. Journal of Cryptology, 2011, 24(2): 322-345

[19] Bilgin B, Daemen J, Nikov V, et al. Efficient and first-order DPA resistant implementations of keccak//Proceedings of International Conference on Smart Card Research and Advanced Applications. Berlin, Germany, 2013: 187-199

[20] Wu Wen-Ling, Zhang Lei, Zheng Ya-Fei, Li Ling-Chen. The block cipher uBlock. Journal of Cryptologic Research, 2019,6(06):690-703(in Chinese)
(吴文玲,张蕾,郑雅菲,李灵琛.分组密码 uBlock.密码学报,

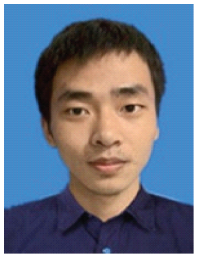
2019,6(06):690-703)

[21] Bilgin B, Nikova S, Nikov V, et al. Threshold implementations of all 3 × 3 and 4 × 4 S-boxes//Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Leuven, Belgium, 2012: 76-91

[22] Bilgin B, Gierlichs B, Nikova S, et al. Higher-order threshold implementations//Proceedings of International Conference on the Theory and Application of Cryptology and Information Security. Taiwan, China, 2014: 326-343

[23] Reparaz O, Bilgin B, Nikova S, et al. Consolidating masking schemes//Proceedings of the 35th Annual Cryptology Conference. Santa Barbara, USA, 2015:64-783

[24] Schneider T, Moradi A. Leakage assessment methodology//Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems. Saint-Malo, France, 2015:495-513



JIAO Zhi-Peng, Ph. D. candidate. His main research field is side channel analysis and protection.

CHEN Hua, Ph. D., professor-level senior engineer. Her main research fields include side channel analysis and protection, cryptography detection.

YAO Fu, Ph. D. candidate. His main research field is side channel analysis and protection.

FAN Li-Min, Ph. D., senior engineer. Her main research fields include side channel analysis and protection, cryptography detection.

Background

As a mainstream attack method of grey box attack model, side channel attack has been widely concerned and studied since it was proposed because of its simplicity and effectiveness, which poses a great threat to the implementation security of cryptography algorithm. Different protection strategies have been proposed against the threat of side channel attacks, among which the threshold implementation technology has been widely studied by academia and industry because of its provable security, ability against glitch attack and acceptable implementation cost. At present, the construction method of threshold implementation has been studied in depth, such as the optimization of the use of random numbers, the optimization of the area realization, etc. However, how to achieve the balance between the security and the realization cost of the specific algorithm is still to be studied, which needs to be analyzed on a case-by-case basis.

uBlock algorithm is the first prize winning block cipher algorithm of the National Cryptographic Algorithm Design Competition in 2019. It is of great significance to the construction of China's cryptographic system and is also threatened by side channel attack. However, there are few researches on side channel protection at present. In order to solve this problem, this paper constructs an area-priority hardware implementation architecture for uBlock algorithm based on serial and pipelining ideas. Then, based on this hardware architecture, a 3-share threshold implementation scheme for uBlock algorithm without new random numbers is constructed. Finally, a 2-share threshold implementation method is constructed to solve the problem that the 3-share threshold area cost is high. Aiming at the problem of random number consumption caused by 2-share threshold implementation, a 2-share threshold implementation scheme without new random number is constructed for uBlock algorithm by means of optimization and reorganization. At the end of the paper, the above schemes are implemented on FPGA and the implementation cost of each scheme is evaluated. The area consumption of the 2-share threshold implementation is reduced by about 30% compared with the 3-share one, and the security of the side-channel protection scheme is verified by the actual side-channel evaluation experiment.

This research is supported by the National Natural Science Foundation of China (No. 62172395).