基于毛刺探测模型的低随机数 SM4 一阶门限实现

祝汉鹏，胡晓婷*，张露露
（江苏师范大学人工智能与计算机学院，徐州 221000）

摘要：门限实现作为一种具有可证明安全性的抗侧信道攻击方法，已被广泛应用于各类密码算法中。尽管已有研究提出了多种针对 SM4 算法的侧信道攻击防护方案，但现有方案在面对毛刺扩展探测等新型攻击模型时仍存在安全缺陷。因此，本文提出一种改进的 SM4 算法 S 盒一阶门限实现方案。该方案基于塔域 $GF((2^2)^2)^2$ 结构设计 S 盒，重构满足门限特性的乘法器，并引入 COTG（changing of the guards）技术优化随机数添加策略，将单个 S 盒的随机数消耗降低至 10 bit。本文借助形式化验证工具 SILVER，证明了该方案在毛刺扩展探测模型下的理论安全性，并通过测试向量泄露评估（test vector leakage assessment，TVLA）验证了整体物理电路的实际防护能力。实验结果表明，该方案可有效抵御一阶侧信道攻击。

关键词：SM4 算法；抗侧信道攻击；毛刺探测模型；低随机数；一阶门限实现
中图分类号：TP309.2 **文献标志码：**A
DOI: 10.20172/j.issn.2097-3136.260602

Low-randomness first-order threshold implementation for SM4 based on the glitch-extended probing model

Zhu Hanpeng, Hu Xiaoting*, Zhang Lulu
（School of Artificial Intelligence and Computer Science, Jiangsu Normal University, Xuzhou 221000, China）

Abstract: Threshold implementation (TI), as a countermeasure against side-channel attacks with provable security, has been widely applied in cryptographic algorithms. Although various side-channel attack protection schemes for the SM4 algorithm have been proposed, existing schemes still exhibit security vulnerabilities when facing novel attack models such as the glitch-extended probing model. To this end, this paper proposes an improved first-order threshold implementation scheme for the S-box of the SM4 algorithm. This scheme designs the S-box based on the tower field $GF((2^2)^2)^2$ structure, reconstructs a multiplier that satisfies threshold properties, and introduces COTG (changing of the guards) technique to optimize the randomness addition strategy, reducing the randomness consumption of a single S-box to 10 bits. Furthermore, the theoretical security of the proposed scheme under the glitch-extended probing model is proven by utilizing the formal verification tool SILVER, and the actual protection capability of the overall physical circuit is validated through test vector leakage assessment (TVLA). The results demonstrate that the proposed scheme can effectively resist first-order side-channel attacks.

Keywords: SM4 algorithm; countermeasure against side-channel attack; glitch-extended probing model; low-randomness; first-order threshold implementation

0 引言

自 1999 年 Kocher 等[1]首次提出差分功耗分析（differential power analysis，DPA）并成功攻破 DES（data encryption standard）算法以来，DPA 已成为嵌入式设备面临的主要物理安全威胁。此类攻击通过监测密码设备在运行时的功耗、电磁辐射等物理特征，并结合统计分析手段提取敏感信息，进而实现

* **通信作者：**E-mail: hxt@jsnu.edu.cn
基金项目：江苏师范大学博士基金（20XSRX014）

引用格式：祝汉鹏，胡晓婷，张露露. 基于毛刺探测模型的低随机数 SM4 一阶门限实现 [J]. 网络空间安全科学学报，2026，4（3）：80 - 91.
Citation Format: Zhu H P，Hu X T，Zhang L L. Low-randomness first-order threshold implementation for SM4 based on the glitch-extended probing model[J]. Journal of Cybersecurity，2026，4（3）： 80 - 91.

密钥恢复。目前，常见的侧信道攻击已涵盖简单功耗分析（simple power analysis，SPA）、差分功耗分析、相关功耗分析[2]（correlation power analysis，CPA）、模板攻击[3]（template attack，TA）以及故障攻击[4]等。这些物理攻击手段的不断演进，使得集成加密功能的嵌入式设备在实际部署中面临严峻的安全风险。

为抵御侧信道攻击，研究者提出了多种防护策略，其中掩码技术是当前主流的防护技术。根据运算类型的不同，掩码技术可分为布尔掩码和算术掩码：布尔掩码适用于涉及布尔操作的场景，如分组密码 AES（advanced encryption standard）、SM4 等；而算术掩码则更适合处理涉及算术运算的场景，如后量子密码算法等[5]。传统的掩码技术通过在运算过程中引入随机掩码来掩盖秘密信息与功耗之间的统计相关性，从而提升抗 DPA 的能力。但在硬件实现中，由于毛刺现象的存在，这种相关性无法完全被消除，从而使得安全方案面临失效风险。Nikova 等[6]提出的门限实现（threshold implementation，TI）是首个在硬件加密中具备可证明安全性的掩码方案。该方案要求将代数次数为 t 的函数分解成 $(t\times d)+1$ 个份额（d 为安全阶数），从而实现对 d 阶攻击的防护。然而，在高阶防护需求下，份额数量的增加会导致硬件面积开销显著增加。为降低总体开销，Reparaz 等[7]提出一种仅需 $d+1$ 个份额即可抵抗 d 阶攻击的 TI 方案，有效减少了面积开销。然而，该方案需要引入额外的新随机数以满足均匀性要求。为进一步降低随机数消耗，Daemen 等[8]将 COTG（changing of the guards）技术与门限实现结合，提出了一种随机数复用机制：在掩码计算过程中，利用与当前敏感变量计算无关的、已缓存的中间值来替代原本需要实时生成的新随机数，从而降低整体随机数的消耗。门限方案作为一种被证实的有效抗侧信道攻击技术，已被广泛应用于 PRESENT[9]、AES[10] 和 PRINCE[11] 等分组密码算法中。

在安全性验证方面，早期主要使用探针模型[12]进行理论评估，但该模型未能考虑实际硬件电路中由毛刺导致的泄露。因此，Faust 等[13]提出了毛刺扩展探测模型，用于评估存在毛刺及耦合错误场景下门限实现的安全性。随着毛刺扩展探测模型的提出，后续的门限方案均在此理论框架下开展安全性验证与优化设计。Shahmirzadi 等[14]提出了一种无需额外随机数的毛刺扩展探针安全方案。该方案基于 2 个份额构建一阶门限实现，并已成功应用于 AES、PRINCE 等算法的 S 盒构造。Yao 等[15]进一步优化了该方法，提出了一种针对替换层中二次函数的优化设计策略，在保证一阶毛刺扩展探测安全的前提下，有效降低了时钟周期与硬件面积开销。

SM4 算法作为我国自主研发的首个分组对称加密算法，广泛应用于金融、物联网、通信等领域。随着应用范围的扩大，侧信道攻击对其安全性的威胁日益凸显，如何抵御侧信道攻击成为国内外学者关注的研究热点。2014 年，谭锐能等[16]从架构优化角度出发，提出了一种 SM4 多路径乘法掩码方法，通过在轮函数中构建多条数据路径并改进 S 盒变换，在低硬件资源和功耗开销的前提下，有效消除了关键中间信息的泄露。2015 年，梁浩等[17]提出了基于复合域求逆的 S 盒掩码防护方案。该方案通过理论分析和 Matlab 仿真验证，证明了其具备抗一阶相关功耗攻击的能力。2016 年，裴超[18]设计了一种基于 S 盒查找表的随机掩码方案，该方法无须掌握 S 盒的具体代数结构，仅需对 S 盒进行随机行列变换即可实现，具有良好的普适性。2018 年，李新超等[19]采用门限实现技术，构造了秘密共享函数来替代 S 盒仿射变换，并基于复合域重新设计了 S 盒结构，有效实现了对一阶 DPA 的防护。但该研究缺乏实际的侧信道安全验证实验。2022 年，武小年等[20]基于 S 盒的复合域分解，提出了一种二共享门限掩码方案，通过电路重构与 S 盒复用降低了硬件开销，具备有效抵抗 CPA 攻击的能力。2023 年，蒲金伟等[21]在门限实现的基础上，针对 S 盒非线性求逆运算提出了无随机数共享方案，引入面向域乘法掩码技术，显著降低了随机数消耗。同时，设计了一种 8 位数据位宽的 SM4 串行架构，具有抗一阶 DPA 攻击的防护能力。然而，多数现有设计仍主要针对传统探针模型，部分方案虽具备一定的抗毛刺能力，但缺乏在毛刺扩展探测模型下的形式化安全验证。因此，本文设计实现了一种在毛刺扩展探测模型下可证安全的 SM4 掩码防护方案。

1　相关理论和技术

1.1　SM4 算法

SM4[22] 是一种对称分组密码算法，分组长度和密钥长度均为 128 bit。算法包含 32 轮非线性迭代运算和 1 次反序变换。

图 1 给出 SM4 算法加密流程，在加密时，128 bit 明文 X 被分为 32 bit 分量 (X_0,X_1,X_2,X_3)，在 32 bit 轮密钥 $rk_i\in(Z_2^{32})$ 的作用下，经 32 次轮函数 F 和最后的反

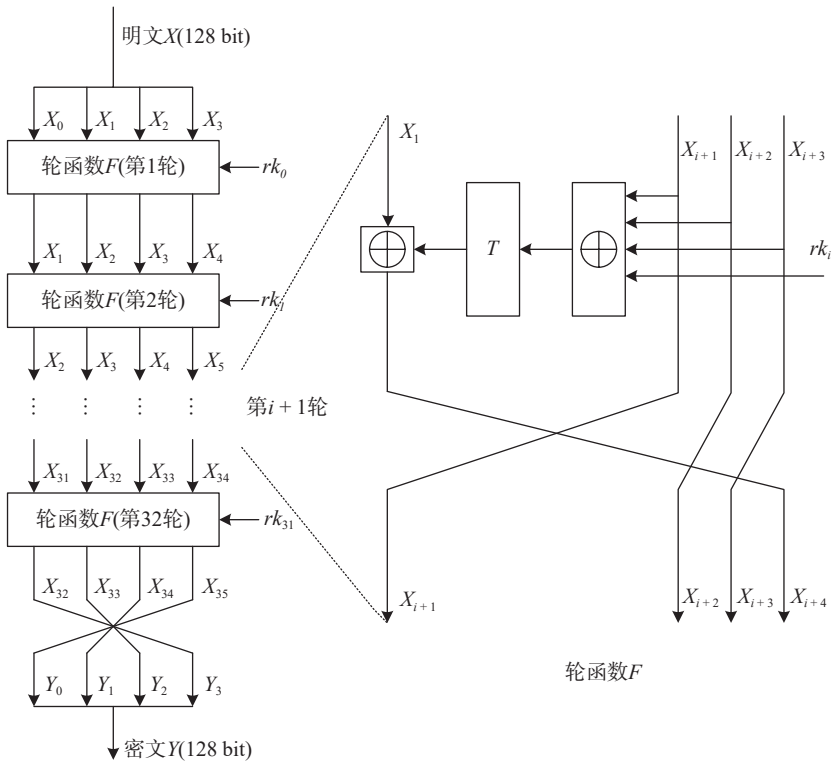

图 1 SM4 算法加密流程

Fig. 1 SM4 algorithm encryption process

序变换，输出式（1）所示的加密结果 Y。

$$Y=(Y_0,Y_1,Y_2,Y_3)=(X_{35},X_{34},X_{33},X_{32}) \tag{1}$$

轮函数 F 结构见图 1，其逻辑表达式见式（2）。

$$\begin{aligned}X_{i+4}&=F(X_i,X_{i+1},X_{i+2},X_{i+3})\\&=X_i\oplus T(X_{i+1}\oplus X_{i+2}\oplus X_{i+3}\oplus rk_i)\end{aligned} \tag{2}$$

其中，T 是可逆变换，非线性变换 τ 和线性变换 L 可表示为 $T(\cdot)=L(\tau(\cdot))$。τ 变换包含 4 个并行的 8 bit S 盒变换，每个 S 盒变换完成一个字节变换操作。具体操作如公式（3）所示。

$$\begin{aligned}B&=\tau(A)\\&=((S\text{box}(a_0),S\text{box}(a_1),S\text{box}(a_2),S\text{box}(a_3))\end{aligned} \tag{3}$$

首先将 32 bit 的输入 A 分成 4 个 8 bit 分量的 a_0,a_1,a_2,a_3，然后对每个 8 bit 做 S 盒变换即可输出 32 bit 的 B。

SM4 解密流程和加密流程相同，仅轮密钥使用顺序相反。

线性变换 L 由异或操作和循环移位操作构成。L 的具体形式如式（4）所示：

$$\begin{aligned}L(B)=&B\oplus(B<<<2)\oplus(B<<<10)\oplus\\&(B<<<18)\oplus(B<<<24)\end{aligned} \tag{4}$$

1.2 毛刺探测模型

毛刺探测模型是 Faust 等[13]提出的一种形式化安全模型，用于评估掩码算法在侧信道攻击下的安全性。该模型与 Ishai 等[12]提出的传统探测模型类似，均假设攻击者可在电路任意位置放置至多 d 个探测点，以获取相关电路信号信息。然而，与传统探测模型不同的是，毛刺扩展探测模型充分考虑了硬件电路中可能产生的毛刺信号。

在传统探测模型中，攻击者重点关注电路理想状态下的信号行为，该模型主要适用于软件实现的安全性分析场景。然而，在 CMOS 器件构成的硬件电路中，信号时延不一致引发的毛刺与瞬态逻辑翻转问题，导致传统安全分析模型在评估底层硬件实现物理安全性时存在准确性不足的缺陷。相比之下，毛刺扩展探测模型充分考虑了这一硬件实际特性。该模型假定攻击者不仅能够获取电路稳态信号，还可捕获因电路时序偏差产生的瞬时毛刺信号。通过将单点探测扩展为对目标连线及其所有依赖计算信号（直至寄存器）的同步探测，可更为全面地评估毛刺等物理现象对电路安全性的影响，为硬件实现提供了更严苛、贴合实际的安全性评估标准。

基于该攻击模型，研究人员开发了 SILVER[23]、MaskVerif[24]、Prover[25] 等多款形式化验证工具，实现了对底层小规模电路安全性的自动化评估。本文提出的 SM4 一阶门限实现方案基于毛刺扩展探测模型设

计，并借助 SILVER 工具完成了 S 盒的安全性验证。

1.3 门限实现

门限实现结合了多方计算和秘密共享的思想，是一种已被证明安全的侧信道防护方案，即使在毛刺环境下也能保证电路安全性。其核心思想是将输入变量 x 拆分成 m 个份额，并将输出函数 f 分解为 n 个分量函数，若这些函数满足以下条件，即可抵御 d 阶侧信道攻击。

1）正确性：假定输入变量 x 被拆分成 m 个份额 $x_1,x_2,\cdots,x_m$，输出函数 f 被分解为 $f_1,f_2,\cdots,f_n$，则它们必须满足 $x=\sum_{i=1}^{m}x_i$，$y=f(x)=\sum_{i=1}^{n}f_i$。

2）d 阶不完整性[2]：如果输入变量被拆分为 d+1 个份额，那么输出函数 $f(x)$的任意 d 个分量最多仅依赖于 d 个输入部分，即任意 d 个分量函数的组合至少独立于输入变量的某一份额。这意味着攻击者即便获取了 d 个分量函数的信息，也无法据此恢复原始的敏感数据，从而确保了敏感数据的安全。

3）均匀性：输入 x 的每个份额都以相同的概率出现，同时输出函数 $f(x)$ 的每个分量也呈均匀分布特性，即各份额在统计上是独立且均衡的。

1.4 COTG 技术

在门限实现中，通常需要在分量函数中引入随机数，以确保输出的均匀性。然而，随机数的生成本身会消耗大量的资源，因此减少对随机数的依赖一直是门限实现中的重要研究方向。COTG 技术，是 Daemen 等[8]在 Keccak 的门限实现方案中首次提出的一种优化方案，其核心思想是：通过使用与当前中间值计算无关的非敏感变量，替代部分或全部随机数的使用，从而有效减少门限方案对随机数的依赖。该方法被广泛应用于 AES[26]、Ascon[27] 等算法中。

例如，在 S 盒的门限实现中，假定 S 盒变换被分解为 3 个分量函数 (S_0，S_1，S_2)。那么，在 m 个 S 盒并行实现方案中，其 3 份额输入 (a_i，b_i，c_i) 经 S 盒映射为 3 份额输出 (A_i，B_i，C_i) 的过程可以表示为式（5）。

$$\begin{cases}A_i=S_0(b_i,c_i),i>0\\B_i=S_1(a_i,c_i),i>0\\C_i=S_2(a_i,b_i),i>0\end{cases}\tag{5}$$

然而，通过式（5）直接计算得到的输出份额 (A_i，B_i，C_i) 可能无法满足均匀性要求。为了确保均匀性，就需要在式（5）的分量函数中引入随机数，这显然会额外增加随机数消耗成本。然而，若采用 COTG 技术（该过程的分量函数可表示为式（6）），可以将原本需独立生成的随机数替换为相邻 S 盒的一个输入份额。由于该份额在代数上不参与当前 S 盒的掩码计算，因而与当前输出份额的分布统计相互独立，从而保证了输出的均匀性，同时有效降低了对额外随机数的依赖。

$$\begin{cases}A_i=S_0(b_i,c_i)+b_{i-1}+c_{i-1},i>0\\B_i=S_1(a_i,c_i)+c_{i-1},i>0\\C_i=S_2(a_i,b_i)+b_{i-1},i>0\end{cases}\tag{6}$$

1.5 基于毛刺探测的 (*d*+1)-share 门限实现

文献 [7] 证明，采用最少 d+1 个输入份额的门限实现方案，就可以有效抵御 d 阶 DPA 攻击。该方案引入随机数，并通过寄存器将运算分隔为扩散层和压缩层两个部分，以防止毛刺现象的传播，从而提高系统的毛刺探测安全性。

例如，在一个简单的两输入与门 $y=f(a,b)$的门限实现中，假定其两份额输入为 (a_0,a_1,b_0,b_1)，输出份额为 (y_0,y_1)，则其分量函数 f_i 可以写成式（7）所示的形式。

$$\begin{aligned}&f_0(a_0,b_0)=a_0b_0\rightarrow x_0,\\&f_1(a_0,b_1)=a_0b_1+r\rightarrow x_1,x_0+x_1=y_0\\&f_2(a_1,b_0)=a_1b_0+r\rightarrow x_2,x_2+x_3=y_1\\&f_3(a_1,b_1)=a_1b_1\rightarrow x_3,\end{aligned}\tag{7}$$

其中，r 为随机数。

在实际实现时，分量函数 f_i 的运算部分，被称为扩散层，其结果 x_0,x_1,x_2,x_3 被存储到寄存器。然后在压缩层将分量函数压缩成 y_i。由式（7）可知，分量函数中必须添加随机数 r，若缺少该随机数 r，压缩后的各输出份额 y_0，y_1 无法与输入份额 (a_0,a_1,b_0,b_1) 保持概率分布独立性。

为减少随机数消耗，文献 [14] 提出了一种搜索方法，通过为各分量函数 f_i 动态添加一次项 a_0,a_1,b_0,b_1 中的若干项，搜索出满足均匀性要求的分量函数组合 f_0,f_1,f_2,f_3，可以在不引入随机数 r 的情况下确保门限方案满足毛刺探测安全性。例如，式（8）就是对式（7）优化后的形式，能够在不依赖随机数的情况下实现二份额与逻辑门限计算。

$$\begin{aligned}&f_0(a_0,b_0)=a_0b_0\rightarrow x_0,\\&f_1(a_0,b_1)=a_0b_1+b_1\rightarrow x_1,x_0+x_1=y_0\\&f_2(a_1,b_0)=a_1b_0\rightarrow x_2,x_2+x_3=y_1\\&f_3(a_1,b_1)=a_1b_1+b_1\rightarrow x_3,\end{aligned}\tag{8}$$

2 低随机数依赖的 SM4 一阶门限实现方案设计

由于 S 盒是 SM4 算法中唯一的非线性操作，也

是整体算法重点防护的部分。因此，本节先重点介绍基于二份额门限实现的 S 盒设计方法，随后，简要阐述 SM4 密码算法的整体防护架构。由于本节涉及变量较多，为方便理解，相关符号说明见表 1 所示。

表 1 符号说明

Table 1 Symbol description table

符号说明	含义
$x=(x_0, x_1)$	表示x由两个份额x_0, x_1构成
x_i^k	表示x_i的第k bit
x'	表示x是1 bit变量
$x\|\|y$	x和y拼接
f_{x_i}	函数f_x的i个分量函数
m_i	用于刷新函数以满足联合均匀性的随机数，且该随机数取自S盒或经复用得到 其中，m_0~m_7为1 bit，m_8和m_9为4 bit
r_i	用于刷新函数以满足联合均匀性的新随机数，均为1 bit

2.1 一阶毛刺探测安全的 S 盒防护方案

本文提出的一阶毛刺探测安全的 S 盒防护方案，基于 Canright[28] 提出的塔域分解 S 盒结构（如图 2 所示）进行构建。该结构利用线性映射，将 GF(2^8) 域上的求逆运算转化为 GF(2^4) 域的运算，并进一步降阶为 GF(2^2) 域，可以有效降低运算复杂度并减少硬件实现面积。在此结构的基础上，本文采用 2 份额的门限实现方案，在毛刺探测模型下实现对一阶侧信道攻击的有效防护。如图 3~图 5 所示，所提出的 S 盒结构包含 6 个处理阶段，各阶段之间以虚线进行分隔。

第 1 阶段：如图 3 所示，该阶段接受 2 份额 8 bit 数据 $a=(a_0, a_1)$，通过线性映射输出 2 份额 8 bit 数据 $b=(b_0, b_1)$。若将线性映射后的数据直接作为下一阶段的输入，可能被毛刺探针探测到输入的相关信息，从而造成信息泄露。为保障线性运算的安全性，需要将映射结果存储于寄存器中。

第 2 阶段：此阶段的输入为 2 份额的 8 bit 数据 $b=(b_0, b_1)$，输出为 16 个 1 bit 数据 x_0', x_1', x_2', x_3', y_0', y_1', y_2', y_3', z_0', z_1', z_2', z_3', t_0', t_1', t_2', t_3'。由于本阶段需在 GF(2^4) 域上完成平方缩放运算和乘法运算，需按高低位将 8 bit 输入变量 $b_i(i\in[0, 1])$ 拆分为两个 4 bit 数据 c_i 和 d_i。

为减少随机数的消耗，本方案将输入为 c_i 和 d_i 的乘法运算与平方缩放运算进行合并，推导出式 (9) 所示的综合布尔表达式，并以此来计算 4 bit 输出 $x||y||z||t$。

$$\begin{cases} x = f_x(c_i^3,c_i^2,c_i^1,c_i^0,d_i^3,d_i^2,d_i^1,d_i^0) \\ \quad = c_i^3d_i^0 + c_i^1d_i^2 + c_i^1d_i^0 + c_i^2d_i^1 + c_i^0d_i^3 + c_i^0d_i^1 + c_i^2d_i^2 \\ \quad + c_i^3d_i^3 + c_i^3d_i^1 + c_i^1d_i^3 + c_i^1d_i^1 + c_i^2 + c_i^0 + d_i^2 + d_i^0 \\ y = f_y(c_i^3,c_i^2,c_i^1,c_i^0,d_i^3,d_i^2,d_i^1,d_i^0) \\ \quad = c_i^3d_i^2 + c_i^2d_i^3 + c_i^2d_i^2 + +c_i^3d_i^1 + c_i^2d_i^0 + c_i^0d_i^2 \\ \quad + c_i^1d_i^3 + c_i^1d_i^1 + c_i^0d_i^0 + c_i^3 + c_i^1 + +d_i^3 + d_i^1 \\ z = f_z(c_i^3,c_i^2,c_i^1,c_i^0,d_i^3,d_i^2,d_i^1,d_i^0) \\ \quad = c_i^3d_i^2 + c_i^3d_i^0 + c_i^1d_i^2 + +c_i^2d_i^3 + c_i^2d_i^1 + c_i^0d_i^3 + c_i^3d_i^3 \\ \quad + c_i^3d_i^1 + c_i^1d_i^3 + c_i^1d_i^1 + c_i^0d_i^0 + +c_i^1 + c_i^0 + d_i^1 + d_i^0 \\ t = f_t(c_i^3,c_i^2,c_i^1,c_i^0,d_i^3,d_i^2,d_i^1,d_i^0) \\ \quad = c_i^3d_i^3 + c_i^2d_i^2 + c_i^3d_i^1 + c_i^2d_i^0 + c_i^0d_i^2 + c_i^1d_i^3 \\ \quad + c_i^0d_i^0 + c_i^1d_i^0 + c_i^0d_i^1 + c_i^0 + d_i^0 \end{cases} \tag{9}$$

其中，x_i^k 表示 x_i 的第 $k$bit。

针对由 4 个代数表达式 x，y，z，t 构成的综合表达式，设计满足毛刺探测安全的防护方案，需同时满足两个核心要求：一是确保扩散层的分量函数不泄露敏感信息，二是保证压缩层输出份额在无寄

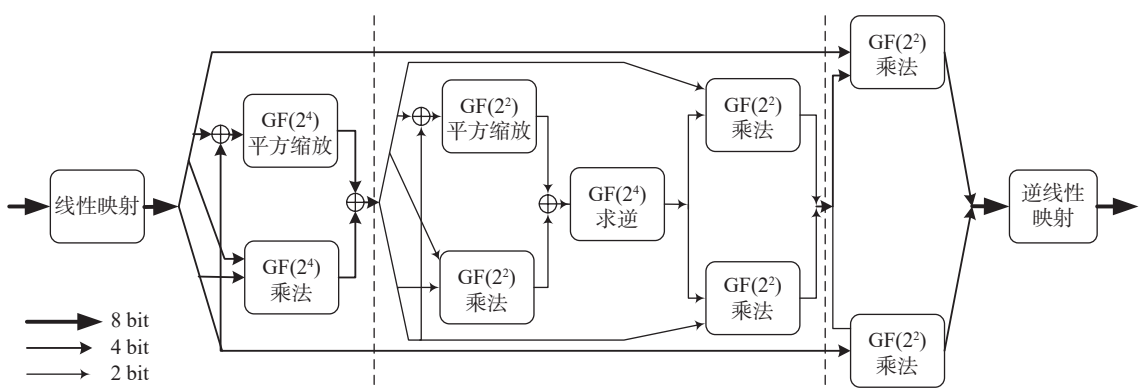

图 2 基于塔域分解的 S 盒结构

Fig. 2 S-box structure based on tower field decomposition

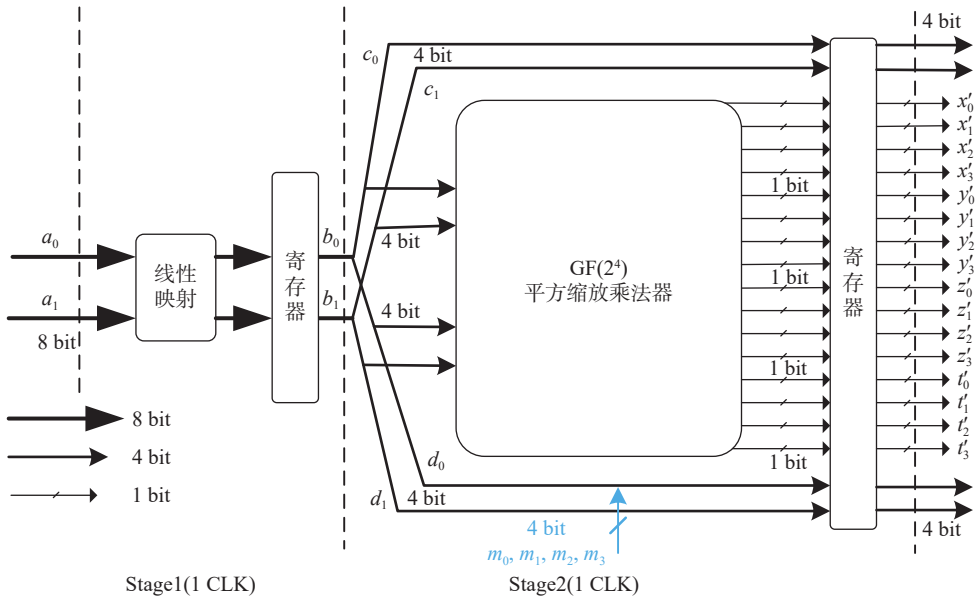

图 3　S 盒门限实现第 1 和第 2 阶段

Fig. 3　First and second stages of the threshold implementation for S-box

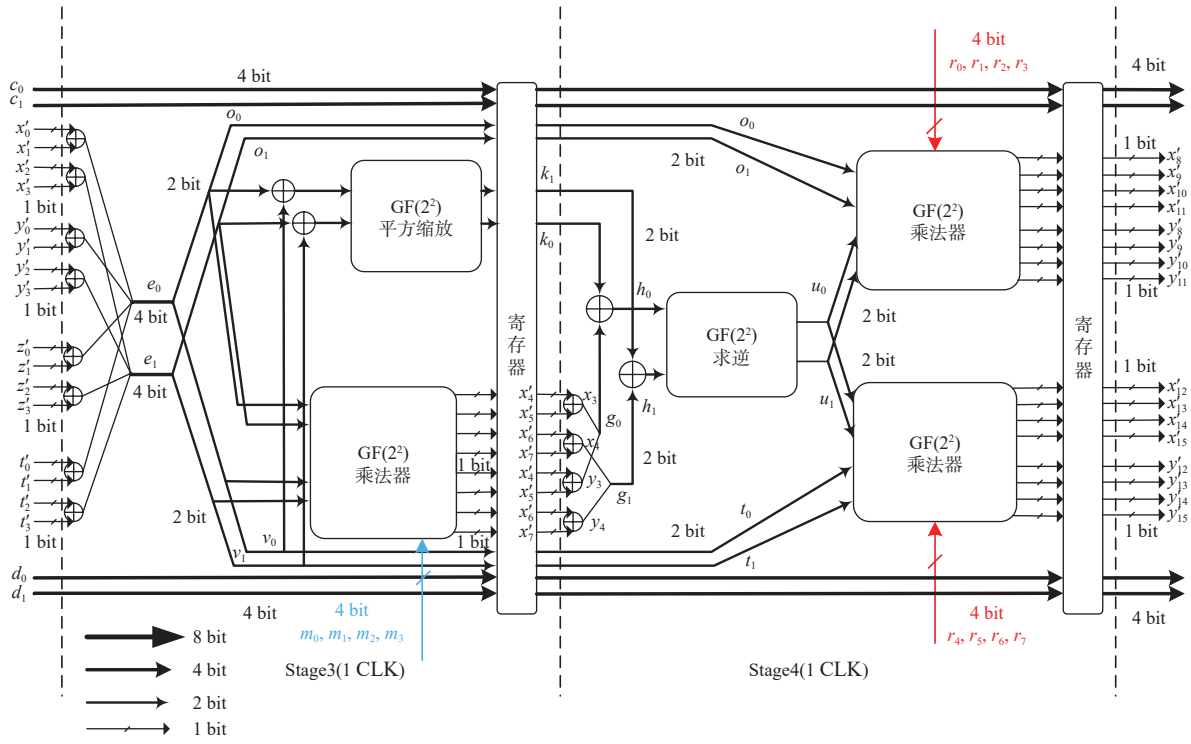

图 4　S 盒门限实现第 3 和第 4 阶段

Fig. 4　Third and fourth stages of the threshold implementation for S-box

存器存储条件下满足联合均匀分布特性，确保输出变量独立于输入变量。

文献 [14] 提出的搜索算法可用于对表达式 x、y、z、t 进行分解，可以确保每个分量函数最多只包含各输入变量的一个份额，从而防止敌手通过探测单个分量函数获取完整输入变量的信息。该搜索算法首先将二次函数拆分成 4 个包含特定二次项的分量函数 f_i，再将这 4 个分量函数两两分组，每组对应一个输出份额。对于每个分量函数，算法会遍历搜索其代数范式中所有可能的一次项（含一个变量的项）和二次项（含两个变量的项）的组合。在筛选过程中，输出分组必须同时满足安全性和均匀性要求。

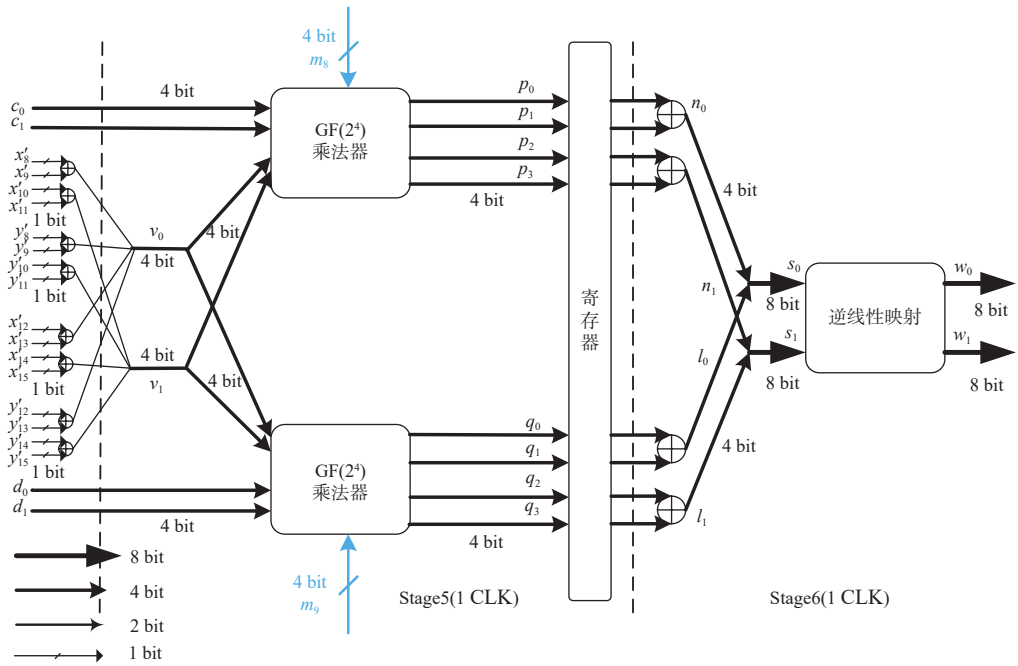

图 5 S 盒门限实现第 5 和第 6 阶段

Fig. 5 The fifth and sixth stages of the threshold implementation for S-box

以式（9）中 x 的计算为例。通过搜索算法可以将函数 f_x 分解为式 $f_{x_0},f_{x_1},f_{x_2},f_{x_3}$ 然后将分组 f_{x_0},f_{x_1} 和 f_{x_2},f_{x_3} 分别压缩生成 x 的两个份额 x_0 和 x_1。具体实现时，压缩阶段放在第 3 阶段。

然而，该算法生成的表达式虽满足单组函数的均匀性（如 x_0 和 x_1），但无法保证在无寄存器存储条件下整体输出份额的联合均匀性，即压缩后的输出份额（如 x_0，x_1，y_0，y_1 等）的概率分布是不一致的，这导致其不满足一阶毛刺安全性要求。因此，本方案采用文献 [15] 中的方法，为 x 添加随机数 m_0 对分量函数进行刷新，以满足联合均匀性要求。这样，x 最终的分量函数可以写成式 (10) 所示的 x'_0,x'_1,x'_2,x'_3 的形式。

$$\begin{cases} x'_0 = f_{x_0} + m_0 \\ x'_1 = f_{x_1} + m_0 \rightarrow x_0 = x'_0 + x'_1 \\ x'_2 = f_{x_2} + m_0 \\ x'_3 = f_{x_3} + m_0 \rightarrow x_1 = x'_2 + x'_3 \end{cases} \tag{10}$$

同理，对表达式 y、z、t 的分解函数分别引入独立随机数 m_1、m_2、m_3 进行刷新处理，确保函数输出 x，y，z，t 满足联合均匀分布条件，从而实现一阶毛刺探测安全。

进一步，为降低随机数的生成和存储开销，本方案基于 COTG 技术，从同轮运算另一 S 盒的 16 bit 输入 (该输入与当前 S 盒运算无逻辑依赖关系) 中，提取 4 个不相关的 1 bit 数据作为图 3 中的 m_0，m_1，m_2，m_3，取代了原方案中需独立生成的随机数，从而有效减少了随机数的使用量。

第 3 阶段：该阶段如图 4 所示，接收 16 个 1 bit 数据 $x'_0,x'_1,x'_2,x'_3,y'_0,y'_1,y'_2,y'_3,z'_0,z'_1,z'_2,z'_3,t'_0,t'_1,t'_2,t'_3$ 作为输入，经处理后，输出主要分为两部分：k 的 2 个 2 bit 分量 k_0、k_1；8 个 1 bit 数据 $x'_4,x'_5,x'_6,x'_7,y'_4,y'_5,y'_6,y'_7$。

具体处理流程如下：首先将第 2 阶段输出的 16 个分量函数，按图 4 所示每两个为一组进行异或压缩，生成 8 个 1 bit 的中间结果，接着将这些结果拼接为两组 4 bit 数据 e_0 和 e_1。随后对 4 bit 数据 e_i（其中 $i\in[0,1]$）按高低 2 bit 进行切分，得到 o_i 和 t_i。

完成拆分后，在 GF(2^2) 域上并行计算 o_i 和 t_i 的平方缩放和乘法运算。其中，GF(2^2) 平方缩放运算生成 k=(k_0，k_1)。本阶段的核心在于 GF(2^2) 上的乘法运算，其代数表达式是一个二次布尔函数，如式（11）所示。

$$\begin{cases} x' = f_{x'}(a',b',c',d') = b'c' + a'd' + b'd' \\ y' = f_{y'}(a',b',c',d') = a'c' + b'c' + a'd' \end{cases} \tag{11}$$

其中，$a'\|b'$ 和 $c'\|d'$ 分别表示 2 bit 乘数和被乘数，$x'\|y'$ 为 2 bit 输出。

与前一阶段类似，首先对表达式进行搜索，以找到满足均匀性条件的 (x'，y') 的分量函数。以 x' 为例，可以搜索得到式 (12) 中所示的 $f_{x'0},f_{x'1},f_{x'2},f_{x'3}$ 分量函数，随后，向各分量函数中添加随机数 m_4，并将结果存储至相应的寄存器中，以确保这两个函数

输出能满足联合均匀性的要求。

$$\begin{cases} f_{x'0}(a'_0,b'_0,c'_0,d'_0)=b'_0c'_0+a'_0d'_0+b'_0d'_0 \\ f_{x'1}(a'_1,b'_1,c'_0,d'_0)=b'_1+b'_1c'_0+a'_1d'_0+b'_1d'_0 \\ f_{x'2}(a'_0,b'_0,c'_1,d'_1)=b'_0c'_1+a'_0d'_1+b'_0d'_1 \\ f_{x'3}(a'_1,b'_1,c'_1,d'_1)=b'_1+b'_1c'_1,+a'_1d'_1+b'_1d'_1 \\ x'_4=f_{x'0}+m_4+m_5 \\ x'_5=f_{x'1}+m_4 \\ x_3=x'_4+x'_5 \\ x'_6=f_{x'2}+m_4+m_5 \\ x'_7=f_{x'3}+m_4 \\ x_4=x'_6+x'_7 \end{cases} \tag{12}$$

需要注意的是，由于添加的随机数在份额异或运算后会相互抵消，而第 3 阶段中乘法器的输入包含了第 2 阶段的输入与输出份额，因此第 3 阶段中该乘法器的输入变量并不相互独立。为解决此问题，需要在 x' 的分解方案中添加额外的随机数 m_5 进行刷新，以确保第 3 阶段的乘法器输入变量完全独立。对于 y' 表达式，采用与 x' 表达式相同的防护方案，同样引入随机数 m_6 和 m_7 进行刷新处理。注意，上述 4 个单比特随机数 (m_4，m_5，m_6，m_7) 同样基于 COTG 技术产生，这里不再赘述。

第 4 阶段：该阶段对应图 4 右侧部分，由 GF(2^2) 求逆模块和两个并行的 GF(2^2) 乘法模块构成，共消耗一个时钟周期。

该阶段的输入包括 3 个两份额变量 k=(k_0,k_1)，o=(o_0,o_1)，t=(t_0,t_1)(每个份额为 2 bit)，以及 8 个 1 bit 数据 x'_4,x'_5,x'_6,x'_7,y'_4,y'_5,y'_6,y'_7，其中 o_i 和 t_i 为第 3 阶段输入数据拆分后的值。

为防止毛刺现象，数据 o_i 和 t_i 两组值在每个时钟周期内均通过寄存器进行重新存储。经过本阶段处理后，最终输出 16 个 1 bit 数据 x'_8,x'_9,x'_{10},x'_{11},y'_8,y'_9,y'_{10},y'_{11},x'_{12},x'_{13},x'_{14},x'_{15},y'_{12},y'_{13},y'_{14},y'_{15}。

具体实现时，首先将输入的 8 个 1 bit 分量函数每两个为一组进行异或压缩，生成 4 个 1 bit 的中间结果。随后按图 4 所示的方式拼接为两个 2 bit 份额 g_0，g_1，并将其与输入 k_0，k_1 进行异或得到 h_0，h_1。接着将 h_0，h_1 送入至 GF(2^2) 求逆模块中进行求逆，得到求逆的结果 u_0，u_1，并将其作为后续乘法器的输入。

在进入乘法器运算环节后，本阶段的两个并行 GF(2^2) 乘法器采用与前一阶段相同的防护策略，采用满足联合均匀性的分解函数方法，并分别引入随机数 (r_0,r_1,r_2,r_3) 与 (r_4,r_5,r_6,r_7) 对其进行刷新，将刷新后结果存入寄存器中。注意，这里的随机数 r_i 和第 1 和第 2 阶段中的 m_i 不同，它不是来自 S 盒的其他输入，而是新添加的随机数。

第 5 阶段：该阶段的操作如图 5 所示，主要包括两个 GF(2^4) 乘法运算单元，其运算仅需一个时钟周期即可完成。

该阶段输入为两个两份额变量 c=(c_0,c_1)，d=(d_0,d_1)，以及 16 个 1 bit 数据 x'_8,x'_9,x'_{10},x'_{11},y'_8,y'_9,y'_{10},y'_{11},x'_{12},x'_{13},x'_{14},x'_{15},y'_{12},y'_{13},y'_{14},y'_{15}。输出为两组，共 8 个 4 bit 数据 p_0，p_1，p_2，p_3 和 q_0，q_1，q_2，q_3。

这里需要说明，为防止毛刺传播，数据 c 和 d 自第 2 阶段起需通过寄存器逐级暂存，因此在进入第 5 阶段前共经历三级缓冲。

核心实现流程如下：首先对输入的 16 个 1 bit 数据进行异或压缩，分别得到中间结果。随后按图 5 所示进行拼接，得到两组 4 bit 数据 v_0，v_1，组成后续乘法器的输入。

接着，完成 v 与 c 以及 v 与 d 的 GF(2^4) 乘法运算。由于后续逆线性映射阶段为线性运算，各变量间无直接运算关系，故此阶段乘法器仅需满足联合均匀性要求即可。因此，本设计只需在两个乘法器中分别引入 4 bit 随机数 m_8 和 m_9 即可满足安全性要求。

值得注意的是，最初通过 COTG 引入的 1 bit 随机数 m_0，m_1，m_2，m_3，m_4，m_6 已分别在第 2 和第 3 阶段被异或操作抵消，且这些随机数与本阶段计算无关，故可安全复用。具体而言，m_8 由 m_0，m_1，m_2，m_3 拼接而成。类似地，m_9 可由 m_4、m_6 与新添加的随机数 r_9，r_{10} 拼接构成。

第 6 阶段：此阶段的输入为第 5 阶段寄存器缓存的 8 个 4 bit 数据，即 p_0，p_1，p_2，p_3 和 q_0，q_1，q_2，q_3。首先将输入数据按组进行异或压缩，输出得到 2 份额的 4 bit 数据 n=(n_0,n_1) 和 l=(l_0,l_1)。具体如式（13）、（14）所示。

$$\begin{cases} p_0=v_0c_0 \\ p_1=v_0c_1+m_8 \\ p_2=v_1c_0 \\ p_3=v_1c_1+m_8 \\ n_0=p_0+p_1 \\ n_1=p_2+p_3 \end{cases} \tag{13}$$

$$\begin{cases} q_0=v_0d_0 \\ q_1=v_0d_1+m_9 \\ q_2=v_1d_0 \\ q_3=v_1d_1+m_9 \\ l_0=q_0+q_1 \\ l_1=q_2+q_3 \end{cases} \tag{14}$$

接着，将压缩后的 4 bit 份额按位拼接，得到 2 个 8 bit 数据 s_0=(n_0||l_0)，s_1=(n_1||l_1)。最后，对 s=(s_0,s_1) 进行

逆线性映射操作，将数据从复合域转换回 GF(2^8) 域，从而获得 S 盒最终的 2 份额输出 w=(w_0，w_1)。

2.2 基于 2-share 的 SM4 一阶门限实现结构

本文提出的 SM4 门限实现方案采用二份额结构，整体架构如图 6 所示，主要包括密钥生成单元、加密处理单元，以及分别用于密钥生成和数据加密的两个独立部署的门限 S 盒。算法初始化时，系统引入随机数对初始的 128 bit 明文及密钥执行掩码操作，将其分别拆解为两份 128 bit 的数据份额，以此作为后续模块的输入。在加密操作启动前，密钥生成单元需经历 9 个时钟周期完成初始密钥的生成，随后将生成的密钥传递至加密处理单元，经过 32 轮循环运算后，最终输出加密结果。

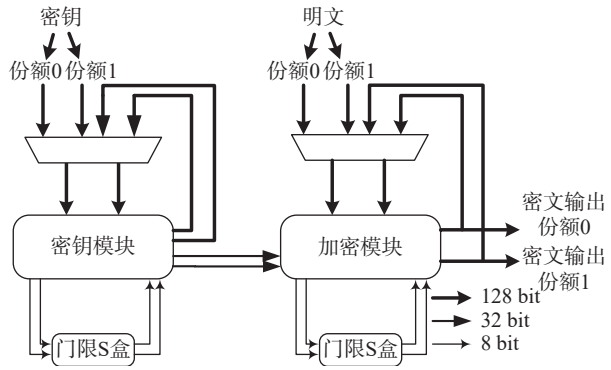

图 6 2-shares 的 SM4 一阶门限实现整体架构

Fig. 6 Architectural framework of the first order threshold implementation for SM4 with two shares

2.3 基于 COTG 技术的随机数使用

为降低 S 盒防护方案中随机数的消耗，本文引入了 COTG 技术。由于本文仅使用单个 S 盒进行 S 盒变换，无法通过传统方法提取相同轮次并行的 S 盒输入来充当随机数。因此，本文采用多路选择器实现跨周期的掩码提取。在执行非线性变换 τ 阶段时，整体 32×2 bit 的输入数据被等分为 4 份 8×2 bit 的数据，并在第 0~3 个时钟周期内依次送入门限 S 盒模块进行处理。基于不同周期下 S 盒输入掩码与当前中间运算结果之间的数据独立性特征，本方案直接截取相邻时钟周期 S 盒输入的单路掩码（如第一份额的 8 bit 数据）充当 Guards 使用。具体如图 7 所示。此方法在有效减少随机数调用的同时，确保了单份额提取操作不会导致敏感信息的泄露。

3 安全性评估与硬件性能分析

3.1 S 盒实现的安全性评估

为评估所设计方案的安全性，本文使用形式化验证工具 SILVER 对提出的 SM4 算法 S 盒实现进行安全性验证。该工具能够基于不同安全模型 (如探测安全、毛刺探测安全等) 评估加密电路的安全性。SILVER 通过对现有设计网表中各逻辑门和连线的信号行为进行概率建模，在不修改原始设计的前提下，利用统计方法验证特定连线子集与输入敏感变量之间的统计独立性，从而判断是否存在敏感信息泄露的路径。

在 SILVER 验证前，本文采用 NanGate 45 nm 标准单元库，通过 Synopsys Design Compiler 进行逻辑综合并生成验证网表。为保证门限实现的非完备性特征，禁用了综合工具的跨模块优化选项。由图 8 SILVER 验证结果可知，本文设计实现的 S 盒一阶门限防护方案满足一阶毛刺探测安全性标准，且输入和输出掩码份额均具有联合均匀分布的性质。

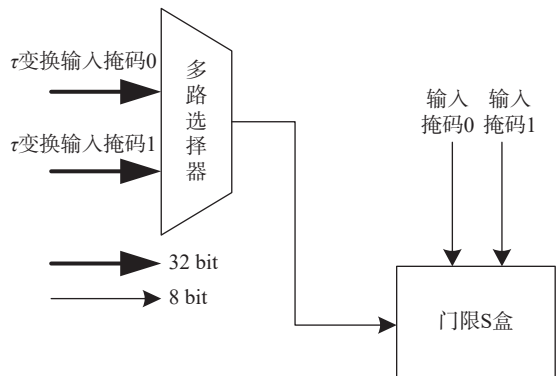

图 7 S 盒防护方案中 Guards 的添加方式

Fig. 7 Method of adding Guards in the S-box protection scheme

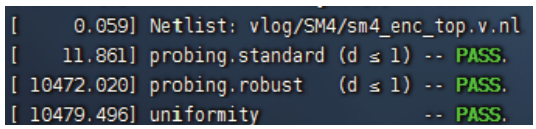

图 8 SM4 算法 S 盒实现的 SILVER 评估结果

Fig. 8 Evaluation results of the SM4 algorithm S-box implementation using SILVER

3.2 SM4 整体电路安全性评估

由于 SM4 算法采用迭代式结构，且其硬件电路复杂度较高，SILVER 无法直接对其整体加密方案进行有效的安全性评估。为了评估 SM4 算法一阶防护方案的整体安全性，本文采用基于 Welch's t 检测的 TVLA（test vector leakage assessment）[29] 技术对其进行验证。本实验采用搭载 Spartan-6 FPGA 的 SAKURA-G 评估板作为硬件测试平台，设置工作时钟频率为 2 MHz。在波形采集阶段，利用 PicoScope 3406D 数字示波器以 500 Msample/s 的采样率，对密码模块加密时的功耗波形进行实时采集。

在使用 TVLA 评估电路整体安全性时，首先需要获取两种情况下的功耗数据：一种是明文固定不变的情况，另一种是明文随机变化的情况。随后，使用公式 (15) 计算检验统计量 t 的值。然后将计算得到的 t 值与预设的置信阈值进行比较，若某采样点的 t 值超出该范围，则可判断该点存在敏感信息泄露的风险。

$$t=\frac{X_A-X_B}{\sqrt{\frac{s_A^2}{N_A}+\frac{s_B^2}{N_B}}} \tag{15}$$

在公式 (15) 中，符号中的下标 A 和 B 分别代表待测的两个功耗样本集，X_A 和 X_B 分别表示对应的样本均值，S_A 和 S_B 是样本方差，N_A 和 N_B 是样本数量。

目前，Welch's t 检验在评测抗侧信道功耗攻击安全性时，业界普遍采用的置信阈值为 $|t|>4.5$。即如果 t 值在 [−4.5，4.5] 内，可认为系统具有抗侧信道攻击的能力，其置信度大于 99.999 9%。否则认为是不安全的。

为评估电路的整体安全性，为全面评估电路整体安全性，本文设置两组 TVLA 信息泄露对比实验：一组关闭伪随机数发生器（pseudo-random number generator，PRNG），一组开启 PRNG。每组评估实验均采集 10 万条功耗曲线。图 9 和图 10 分别给出了两组 t 值检验结果。

3.3　硬件性能对比分析

本文硬件实现性能评估基于 Synopsys Design Compiler 工具，并采用 NanGate 45 nm 工艺库进行逻辑综合。表 2 给出 SM 算法不同门限实现的安全性和硬件性能对比。

由表 2 实验结果可知，基于 COTG 技术设计的 S 盒电路，对一阶毛刺探测模型具备可证明的安全防护能力。在随机数消耗方面，单轮 S 盒的随机数消耗量仅需 10 bit，相比文献 [19] 和文献 [21] 的 58 bit 和 12 bit 方案，分别降低了 82.8% 和 16.7%。然而，安全性提升也带来了相应的硬件开销增加。

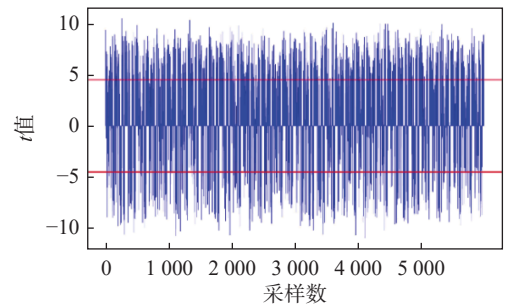

图 9　关闭 PRN 情况的一阶门限实现 SM4 防护方案的 TVLA 实验结果

Fig. 9　TVLA results of a first-order threshold implementation of SM4 with PRNG disabled

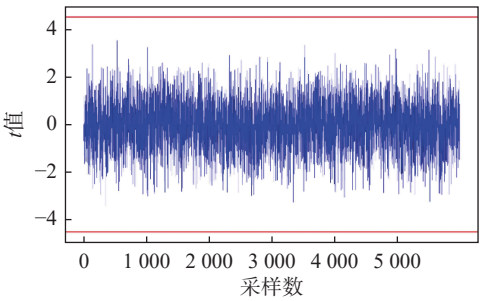

图 10　打开 PRNG 情况的一阶门限实现 SM4 防护方案的 TVLA 实验结果

Fig. 10　TVLA test results of a first-order threshold implementation of SM4 with PRNG enabled

时钟开销的增加主要源于硬件面积的折中设计，为有效缩减面积，各模块内部采用单 S 盒替代了传统的 4 个 S 盒并行架构，数据的分批输入与输出额外消耗了时钟周期。此外，考虑到 SM4 每轮加密均需实时调用轮密钥，若加密与密钥生成模块共享单一 S 盒将导致严重的串行等待，因此本方案选择在两模块中分别独立配置门限 S 盒。在面积开销方面，

表 2　SM4 算法不同门限实现的安全性和硬件性能对比

Table 2　Comparison of security and hardware performance for different threshold implementations for SM4

方案	防护阶数	单个S盒面积(kGE)	SM4面积(kGE)	S盒随机数/bit	时延(CLK周期数)	加入密钥防护	安全模型	
							一阶探测	一阶毛刺探测
文献[17]	1	—	18.7	—	160	是	√	×
文献[19]	1	—	28.0	58	160	否	√	×
文献[21]	1	2.19	6.79	12	292	否	√	×
本文	1	2.26	14.10	10	297	是	√	√

表现有所差异：单个 S 盒的面积较文献 [21] 增加了 3.2%，而 SM4 整体面积相较文献 [17] 和文献 [19] 分别减少了 24.6% 和 49.6%，但相较文献 [21] 则增加了 107.7%。需要说明的是，文献 [21] 未对密钥生成模块提供防护，且其采用 8 bit 数据位宽的串行结构设计更为紧凑，因此面积较小。相比之下，本文方案虽然在面积上有所增加，但提供了对一阶毛刺探测的防护能力，有效提升了算法的抗侧信道攻击性能。

4 结束语

面向毛刺探测模型，本文构建了一种 SM4 算法的一阶二份额门限实现方案。该方案通过对乘法器进行分解构造，结合 COTG 技术，将 S 盒的随机数消耗压缩至 10 bit，从而减少了系统对复杂随机数生成器的依赖。尽管在硬件实现层面引入了部分面积增加与时延开销，但 SM4 算法在该模型下的安全防护能力得到了显著增强。安全性验证结果表明，所提方案不仅在毛刺探测模型下具备可证明的安全性，且通过了基于 Welch's t 检验的一阶 TVLA 测试，可有效抵御一阶侧信道攻击。

后续研究将致力于在维持低随机数开销的前提下，进一步缩减整体防护方案的硬件实现面积。

参考文献：

[1] Kocher P，Jaffe J，Jun B. Differential power analysis[C]// Advances in Cryptology—CRYPTO'99，19th Annual International Cryptology Conference，Santa Barbara，CA，USA，1999：388-397.

[2] Bellizia D，Scotti G，Trifiletti A. Fully integrable current-mode feedback suppressor as an analog countermeasure against CPA attacks in 40nm CMOS technology[C]//13th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME). IEEE，2017：349-352.

[3] Chari S，Rao J R，Rohatgi P. Template attacks[C]//International workshop on cryptographic hardware and embedded systems. Berlin，Heidelberg: Springer Berlin Heidelberg，2002：13-28.

[4] Fuhr T，Jaulmes É，Lomné V，et al. Fault attacks on AES with faulty ciphertexts only[C]//2013 Workshop on Fault Diagnosis and Tolerance in Cryptography. IEEE，2013：108-118.

[5] 张舒琪，李延斌，王蓬勃，等. 抗侧信道攻击的后量子密码掩码转换方案 [J]. 网络空间安全科学学报，2024，2（5）：44-56.
Zhang S Q，Li Y B，Wang P B，et al. Mask conversion scheme on post quantum cryptographic for resisting side channel attacks[J]. Journal of Cybersecurity，2024，2（5）：44-56.

[6] Nikova S，Rechberger C，Rijmen V. Threshold implementations against sidechannel attacks and glitches[C]//International Conference on Information and Communications Security. Berlin，Heidelberg：Springer Berlin Heidelberg，2006：529-545.

[7] Reparaz O，Bilgin B，Nikova S，et al. Consolidating masking schemes[C]//Annual Cryptology Conference. Berlin，Heidelberg：Springer Berlin Heidelberg，2015：764-783.

[8] Daemen J. Changing of the guards：a simple and efficient method for achieving uniformity in threshold sharing[C]//International Conference on Cryptographic Hardware and Embedded Systems. Cham：Springer International Publishing，2017：137-153.

[9] Liu B，Tang M. MS - LW - TI：Primitive - based first - order threshold implementation for 4× 4 S - boxes[J]. IET Information Security，2024（1）：8851878.

[10] Dhooghe S，Shahmirzadi A R，Moradi A. Second-order low-randomness d+1 hardware sharing of the AES[C]//Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022：815-828.

[11] Dhooghe S，Ovchinnikov A. Threshold implementations with non-uniform inputs[C]//International Conference on Selected Areas in Cryptography. Cham：Springer Nature Switzerland，2023：97-123.

[12] Ishai Y，Sahai A，Wagner D. Private circuits：Securing hardware against probing attacks[C]//CRYPTO 2003：Advances in Cryptology，Santa Barbara，USA，2003：463-481.

[13] Faust S，Grosso V，Del Pozo S M，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（3）：89-120.

[14] Shahmirzadi A R，Moradi A. Reconsolidating first-order masking schemes：nullifying fresh randomness[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems，2021：305-342.

[15] Yao F，Chen H，Wei Y，et al. Optimizing AES threshold implementation under the glitch-extended probing model[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems，2024，43（7）：1984-1997.

[16] 谭锐能，卢元元，田椒陵. 抗侧信道攻击的 SM4 多路径乘法掩码方法 [J]. 计算机工程，2014，40（5）：103-108.
Tan R N，Lu Y Y，Tian J L. Multi-path multiplicative masking method of SM4 against side-channel attacks[J]. Computer Engineering，2014，40（5）：103-108.

[17] 梁浩，乌力吉，张向民. 基于复合域的 SM4 算法的设计与实现 [J]. 微电子学与计算机，2015，32（5）：16-20.
Liang H，Wu L J，Zhang X M. Design and implementation of SM4 block cipher based on composite field[J]. Microelectronics & Computer，2015，32（5）：16-20.

[18] 裴超. 一种 SM4 掩码方法和抗 DPA 攻击分析 [J]. 密码学报，2016，3（1）：79-90
Pei C. A masking method of SM4 and analysis of anti-DPA attack[J]. Journal of Cryptologic Research，2016，3（1）：79-90.

[19] 李新超，钟卫东，张帅伟，等. 一种基于门限实现的 SM4 算法 S 盒实现方案 [J]. 计算机工程与应用，2018，54（17）：83-88.
Li X C，Zhong W D，Zhang S W，et al. A new threshold implementation of the S-box in SM4[J]. Journal of Cryptologic Research. 2018，5（6）：641-650.

[20] 武小年，李金林，潘晟，等. SM4 算法门限掩码方案设计与实现 [J]. 计算机应用研究，2022，39（2）：572-576.
Wu X N，Li J L，Pan S，et al. Design and implementation of threshold masking scheme for SM4 algorithm[J]. Application Research of Computers，2022，39（2）：572-576.

[21] 蒲金伟，高倾健，郑欣，等. SM4 抗差分功耗分析轻量级门限实现 [J]. 计算机应用，2023，43（11）：3490-3496.
Pu J W，Gao Q J，Zheng X，et al. SM4 resistant differential power analysis lightweight threshold implementation[J]. Journal of Computer Applications，2023，43（11）：3490-3496.

[22] 吕述望，苏波展，王鹏，等. SM4 分组密码算法综述 [J]. 信息安全研究，2016，2（11）：995-1007.
Lü S W，Su B Z，Wang P，et al. A review of the SM4 block cipher algorithm[J]. Information Security Research，2016，2（11）：995-1007.

[23] Knichel D，Sasdrich P，Moradi A. SILVER-statistical independence and leakage verification[C]//International Conference on the Theory and Application of Cryptology and Information Security. Cham：Springer International Publishing，2020：787-816.

[24] Barthe G，Belaïd S，Cassiers G，et al. maskverif：Automated verification of higher-order masking in presence of physical defaults[C]//European Symposium on Research in Computer Security. Cham：Springer International Publishing，2019：300-318.

[25] Zhou F，Chen H，Fan L. Prover toward more efficient formal verification of masking in probing model[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems，2025（1）：552-585.

[26] Gigerl B，Klug F，Mangard S，et al. Smooth passage with the guards：Second-order hardware masking of the AES with low randomness and low latency[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems，2024（1）：309-335.

[27] Samwel N，Daemen J. DPA on hardware implementations of Ascon and Keyak[C]//Proceedings of the Computing Frontiers conference. 2017：415-424.

[28] Canright D. A very compact S-box for AES[C]//International Workshop on Cryptographic Hardware and Embedded Systems. Berlin，Heidelberg：Springer Berlin Heidelberg，2005：441-455.

[29] 郑震，严迎建，刘燕江. 侧信道能量信息测试向量泄漏评估技术 [J]. 电子与信息学报，2023，45（9）：3109-3117.
Zheng Z，Yan Y J，Liu Y J. Test vector leakage assessment technique of side-channel power information[J]. Journal of Electronics & Information Technology，2023，45（9）：3109-3117.