



以低成本实现Serpent算法的量子电路

陈慧楠¹, 朱彦莹¹, 蔡彬彬^{1,2*}, 刘晓芬^{1*}

1. 福建师范大学计算机与网络空间安全学院, 福州 350117

2. 福建师范大学数字福建环境监测物联网实验室, 福州 350117

*联系人, 蔡彬彬, E-mail: cbb@fjnu.edu.cn; 刘晓芬, E-mail: liuxiaofen23@fjnu.edu.cn

收稿日期: 2024-11-13; 接受日期: 2024-12-19; 网络出版日期: 2025-03-06

国家自然科学基金(编号: 62171131, 61976053)、福建省自然科学基金(编号: 2022J01186, 2023J01533)和福建省中青年教师教育科研项目(编号: JAT231018)资助

摘要 Serpent是AES候选算法之一, 以其高安全性和抗攻击性著称, 适用于对信息安全要求较高的领域. S盒是Serpent算法中唯一的非线性结构. 本文研究如何使用较少的资源实现Serpent算法的量子电路. 以更少的量子比特和量子门实现S盒的量子电路, 结合S盒的逆量子电路, 可构造密钥扩展的量子电路. 并且通过对S盒的量子电路中出现的3控制非门进行分解, 并对分解后的量子电路进行优化, 进一步减少Toffoli门的使用. 结合所提出的量子电路, 给出需要448个量子比特、92288个T门和需要384个量子比特、118944个T门两种实现Serpent算法的量子电路. 与之前需要800个量子比特和132608个T门的研究相比, 本文实现Serpent算法量子电路需要的资源更少. 此外, 本文的S盒与密钥扩展的量子电路可减少实现Serpent时所需的资源, 进而降低Grover算法对其攻击时所需的量子电路规模.

关键词 Serpent, S盒, 量子电路, Grover算法, 资源估计

PACS: 03.67.Lx, 03.67.Dd, 03.65.Ta

1 引言

随着信息时代的飞速发展, 信息安全问题日益凸显, 成为全球关注的焦点. 作为保障和实现信息安全的核心技术, 密码学得到广泛的应用和重视. 然而, 量子计算技术的快速发展^[1,2]带来了前所未有的挑战. 在过去二十年间, 量子信息领域取得诸多技术突破, 其基于量子并行性、量子纠缠等量子力学现象, 提供一种全新的信息处理和通信模式, 具有超越经典计算系统的潜力^[3], 如Shor算法^[4,5]、Grover算法^[6]和Simon算

法^[7]等. 其中, Shor算法通过量子计算来确定函数的周期性, 并利用量子傅里叶变换^[8-10]对因式分解问题进行快速求解, 显著地加速大整数的分解过程. 相比对应经典算法的指数级复杂度, Shor算法能够在多项式时间内完成这一任务. 因此, Shor算法对基于因式分解难题的经典加密系统(尤其是目前广泛使用的RSA加密方案^[11])构成直接威胁, 使得该传统公钥密码算法在量子计算背景下不再安全. 这一发现促使密码学界重新评估现有加密技术在量子计算时代的适应能力, 同时推动量子密码学和后量子密码学的研究. 特别是量子

引用格式: 陈慧楠, 朱彦莹, 蔡彬彬, 等. 以低成本实现Serpent算法的量子电路. 中国科学: 物理学 力学 天文学, 2025, 55: 240313

Chen H N, Zhu Y Y, Cai B B, et al. Quantum circuit for implementing Serpent algorithm with low costs (in Chinese). Sci Sin-Phys Mech Astron, 2025, 55: 240313, doi: 10.1360/SSPMA-2024-0550

密钥分发^[12]和量子隐形传态^[13]作为量子通信中的关键技术,其重要性得到进一步重视.相关技术的深入发展,不仅旨在应对量子信息技术带来的潜在安全挑战,也为未来的应用需求提供坚实的理论与技术支持.

Grover算法在无结构数据库搜索问题上展示显著的量子加速能力.在经典计算环境中,在大小为 N 的无序列表中查找目标项,平均需要进行 $O(N)$ 次查询,而Grover算法利用量子并行性和干涉原理,只需要 $O(\sqrt{N})$ 次查询就能找到目标项.虽然Grover算法并不像Shor算法那样直接对公钥加密系统造成威胁,但它对对称加密算法,尤其是分组密码^[14,15]的安全性产生深远影响.在经典计算环境中,破解一个密钥长度为 n 位的对称加密算法通常需要经过 $O(2^n)$ 次尝试.但在量子环境下,应用Grover算法后,只需 $O(2^{n/2})$ 次查询就能完成.这意味着现有的私钥加密方案在量子计算时代变得更加脆弱,攻击者可以大幅度减少所需的计算资源,从而提高成功攻击的可能性.因此,为保持与经典计算环境下相同的安全性,以抵御Grover算法的攻击,分组密码的密钥长度通常需要至少增加一倍. Grover算法作为已知的能够威胁分组密码的量子算法,对其攻击方案的资源消耗进行研究,可以更清楚地认识现有分组密码在量子计算环境下的安全强度.由于量子计算机的计算能力目前尚难以预测,美国国家标准与技术研究院(National Institute of Standards and Technology, NIST)^[16]提出通过基本操作数、电路规模等指标来衡量安全性.近年来,针对量子攻击者计算资源的评估逐渐受到广泛关注,相关研究包括对Grover搜索在对称加密方案中的密钥搜索^[17-28]及哈希函数搜索的资源估计^[29]等.

Serpent^[30]是一种对称分组密码算法,由Ross Anderson, Eli Biham和Lars Knudsen在1998年设计,它是高级加密标准(Advanced Encryption Standard, AES)^[31-33]竞赛的五个最终候选算法之一.尽管Serpent未能成为AES的最终标准,但凭借其卓越的安全性和成熟的设计理念, Serpent依然在密码学界获得广泛的认可和应用. Serpent的设计目标是提供极高的安全性,能够有效抵御已知的各种密码分析攻击^[34-42].设计者在安全性上采取非常保守的策略,其保守性体现在它的加密轮数比AES的更多.这种设计确保即使在面对未来潜在的攻击手段时, Serpent仍能够保持强大的防御能力,降低被破解的风险.目

前,关于Serpent算法的量子密码分析研究仍处于探索阶段.特别是在量子计算逐渐成为现实的背景下,评估Grover算法对Serpent算法的攻击所需资源显得尤为重要.这一分析不仅能够对Serpent算法在量子计算环境下的安全性提供更深入的理解,也为设计更加安全的后量子密码算法提供参考依据.

在对分组密码的密钥进行恢复时, Grover算法相比经典搜索方法提供平方级的加速,使得破解对称加密算法变得更加高效.为实现Grover算法在对称密码上的应用,需要对密码加密过程进行可逆实现,以便能够有效地进行密钥搜索.本文针对Serpent算法设计其可逆量子电路的实现,并在此基础上评估Serpent算法的Grover搜索攻击所需的量子资源.这一分析不仅为量子攻击的可行性提供实证支持,也为未来密码学研究提供参考.具体地,本文利用DORCIS工具^[43]对Serpent算法S盒代换操作的量子电路进行设计,并对电路中出现的3控制非门进行分解和优化,提高运行效率.与文献^[44]相比,本文在Serpent加密过程的量子电路设计中使用更少的 T 门和量子比特,降低量子硬件的需求.这一优化不仅展示本文设计方法的有效性,还为实际应用中的量子计算资源管理提供新的思路.

2 预备知识

2.1 Serpent算法

Serpent算法采用32轮的代换-置换网络(Substitution-Permutation Network, SPN)结构.加密过程中的输入数据块大小为128比特,初始密钥长度有3种不同的长度:128, 192和256比特.其中,128比特的明文 M ,经过32次轮函数操作处理,生成128比特的密文 C .

Serpent算法的加密过程(图1)由以下部分组成:

(1) 初始置换.

(2) 次轮函数操作.其中,每个轮函数操作包括一次轮密钥加操作、一次S盒代换操作和一次线性变换操作.且在最后一次轮函数操作中,线性变换操作被一个额外的轮密钥加操作取代:

(3) 逆初始置换.

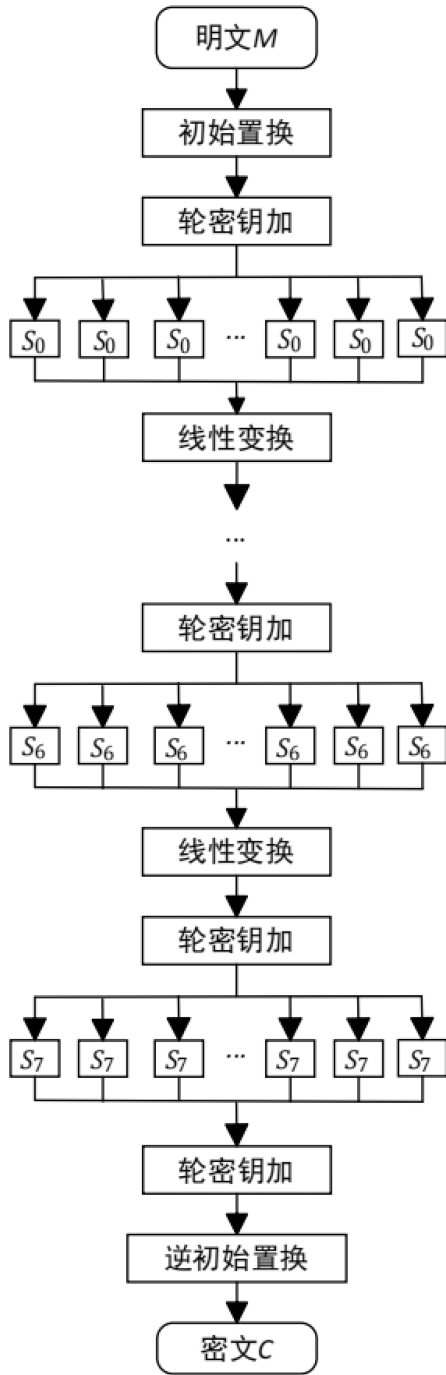
图 1 Serpent算法的加密过程^[34]

Figure 1 The encryption process of Serpent algorithm [34].

2.1.1 初始置换和逆初始置换

在加密过程的开始, 对输入的明文 M 进行一次初始置换. 在加密结束之前, 执行一次逆初始置换后, 得

到密文 C . 其中, 初始置换与逆初始置换的功能是将原有的数据按照特定位置进行置换, 可通过查询置换表得到, 该置换表的详细信息参见文献[22].

2.1.2 轮密钥加操作

在第 i 次轮函数操作过程中, 128比特的子密钥 K_i 与当前数据块 B_i 进行按位异或, 记为 $K_i \oplus B_i$, 其中 $0 \leq i \leq 31$.

2.1.3 S盒代换操作

Serpent算法使用的8个的S盒, 详见附录A1. 第 t 个S盒代换操作记为 S_t , 对应的逆S盒代换操作记为 S_t^{-1} . 这些S盒按顺序循环使用, 在第0 (1, 2, ..., 7)次轮函数操作中使用 S_0 ($S_1, \dots, S_7$), 第8次轮函数操作再次使用 S_0 , 接着第9次轮函数操作再次使用 S_1 , 以此类推. 每个S盒在整个加密过程中被使用4次, 并且在每一次轮函数操作中并行执行32次S盒代换操作(每个S盒处理4比特数据块, 而每次轮函数操作需要处理的比特数据块大小为128比特).

2.1.4 线性变换操作

线性变换操作每次处理128比特数据块, 将数据块从低位到高位划分为4个字, 每个字32比特, 分别记为 X_0, X_1, X_2, X_3 . 线性变换操作后的32比特输出 X_0, X_1, X_2, X_3 通过以下步骤生成:

$$X_0: = X_0 \lll 13,$$

$$X_2: = X_2 \lll 3,$$

$$X_1: = X_1 \oplus X_0 \oplus X_2,$$

$$X_3: = X_3 \oplus X_2 \oplus (X_0 \ll 3),$$

$$X_1: = X_1 \lll 1,$$

$$X_3: = X_3 \lll 7,$$

$$X_0: = X_0 \oplus X_1 \oplus X_3,$$

$$X_2: = X_2 \oplus X_3 \oplus (X_1 \ll 7),$$

$$X_0: = X_0 \lll 5,$$

$$X_2: = X_2 \lll 22.$$

其中, $\lll$ 表示循环左移, $\ll$ 表示逻辑左移.

2.1.5 密钥扩展

Serpent算法的输入密钥有128, 192和256比特三种长度. 对于不足256比特的密钥, 会通过扩展方式将其长度调整为256比特. 具体的扩展方案如下: 首先, 在输入密钥的最高位上补一个“1”, 然后在其后填充多个“0”, 直到密钥长度达到256比特. 因此, 可以将初始输入的密钥长度统一视为256比特. 在32次轮函数操作过程中, 算法涉及的33个(前31次轮函数操作每次均有一次轮密钥加操作, 第32次轮函数操作有两次轮密钥加操作) 128比特的子密钥通过以下步骤生成:

1. 将256比特的初始密钥从低位到高位划分为8个字, 分别记为 $\omega_{-8}, \dots, \omega_{-1}$.

2. 获得预处理密钥 ω_j :

$$\omega_j = (\omega_{-8} \oplus \omega_{-5} \oplus \omega_{-3} \oplus \omega_{-1} \oplus \phi \oplus j) \lll 11,$$

其中 j 是获得预处理密钥的次序($0 \leq j \leq 131$), $\phi = 0x9e3779b9$ 为十六进制常数.

3. 获得子密钥 K_i :

$$\begin{aligned} K_i &= \{k_{4i}, k_{4i+1}, k_{4i+2}, k_{4i+3}\} \\ &= S_{(i+3) \bmod 8} \{ \omega_{4i}, \omega_{4i+1}, \omega_{4i+2}, \omega_{4i+3} \}. \end{aligned}$$

其中 i 是获得预处理密钥的次序($0 \leq i \leq 32$).

2.2 Grover算法

在一个大小为 N ($n = \log_2 N$) 的无结构数据集中寻找满足特定条件的一个目标元素 y 的搜索问题. 假设存在函数 $f(x) : \{0, 1\}^n \rightarrow \{0, 1\}$ 满足:

$$f(x) = \begin{cases} 1 & \text{当 } x = y, \\ 0 & \text{其他.} \end{cases}$$

假设映射 f 是能够容易有效实现的, 则有Oracle O_f 用于确定元素 x 是否为目标元素. 为解决上述搜索问题, 经典穷举搜索算法需要进行 $O(2^n)$ 次Oracle O_f 查询, 而Grover算法只需要进行 $O(2^{n/2})$ 次Oracle O_f 查询.

Grover 搜索算法由以下步骤组成:

1. 准备量子态 $|0\rangle^{\otimes n}$, 对 n 个量子比特执行Hadamard操作, 得到状态: $|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$.

2. 构造黑盒 $O_f: |x\rangle \xrightarrow{O_f} (-1)^{f(x)} |x\rangle$, 使得当 x 是问题的解时, $f(x) = 1$, 否则 $f(x) = 0$.

3. 执行 R 次(其中 $R = \lfloor \frac{\pi}{4} 2^{n/2} \rfloor$) Grover迭代($2|\psi\rangle\langle\psi| - I$) O_f 得到: $[(2|\psi\rangle\langle\psi| - I)O_f]^R |\psi\rangle \approx |y\rangle$.

4. 测量得到目标 y .

2.3 基础量子门

图2展示本文使用到的基础量子门. 其中控制非门(2(b))和Toffoli门(图2(c))最后一个量子比特为目标位, 其余量子比特为控制位. 控制非门用于实现按位异或操作, 即控制非门可以将控制比特的值与目标比特按位异或, 并将结果赋值到目标比特上.

3 Serpent算法的量子电路

Serpent算法由两个子程序组成, 加密过程和密钥扩展. 下面, 首先描述这两个子程序并对其进行量子电路设计, 然后展示如何利用这两个子程序构成Serpent算法的可逆量子电路.

3.1 加密过程的量子电路

加密过程主要包括1次初始置换、32次轮函数操作和1次逆初始置换三个部分.

3.1.1 初始置换和逆初始置换的量子电路

初始置换与逆初始置换只是把原有的数据按照位置进行置换, 其量子电路只需要改变输出比特位顺序即可实现, 无需量子操作.

3.1.2 轮函数操作的量子电路

轮函数操作包含三个部分: 轮密钥加操作、S盒代换操作、线性变换操作.

(1) 轮密钥加操作的量子电路. 在Serpent算法中, 共使用33个子密钥进行轮密钥加操作. 前31轮各使用1个子密钥, 在第32次轮函数操作过程中, 用轮密钥

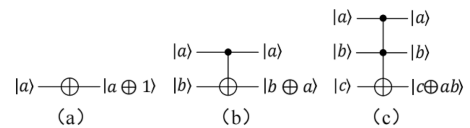


图2 基础量子门. (a) 非门; (b) 控制非门; (c) Toffoli门

Figure 2 Basic quantum gates. (a) NOT gate; (b) CNOT gate; (c) Toffoli gate.

加操作代替线性变换操作, 因此该轮次共使用2个子密钥.

在每次轮密钥加操作中, 子密钥的长度与数据块的长度均为128比特. 通过将子密钥与当前数据块按位异或来完成, 因此需要128个控制非门. 所以在整个加密过程中, 轮密钥加操作共需要 $33 \times 128 = 4224$ 个控制非门.

(2) S盒代换的量子电路. 在经典计算机中, 可以利用预定义的S盒, 根据输入值直接查找并匹配出相应的输出值. 然而, 由于量子位的叠加特性, 量子计算机可同时处理多个输入值, 因此预定义的S盒不能直接应用于量子计算, 必须将S盒实现为一个根据输入动态产生不同输出的量子电路.

通过计算, 可以推导出Serpent算法S盒的代数关系. 例如, 用 x_3, x_2, x_1, x_0 分别表示从高位到低位表示输入S盒的4比特数据, 操作的代数关系计算结果如式(1)所示:

$$\begin{aligned} S_0(x_0) &= 1 + x_0 + x_2 + x_3 + x_1x_0 + x_2x_0 + x_2x_1 \\ &\quad + x_2x_1x_0 + x_3x_2x_0 + x_3x_2x_1, \\ S_0(x_1) &= 1 + x_0 + x_2x_0 + x_2x_1 + x_3x_1 + x_2x_1x_0 \\ &\quad + x_3x_2x_0 + x_3x_2x_1, \\ S_0(x_2) &= x_1 + x_3 + x_1x_0 + x_2x_0 + x_3x_1 + x_2x_1x_0 \\ &\quad + x_3x_2x_1, \\ S_0(x_3) &= x_0 + x_1 + x_2 + x_3 + x_3x_0. \end{aligned} \quad (1)$$

根据 S_0 操作的代数关系可知, 在量子电路中实现 S_0 操作需要使用额外的辅助量子比特来存储式(1)中涉及的中间状态. 因此, 为避免等式中低效的S盒代换操作的实现, 本文选择使用DORCIS工具¹⁾来实现S盒的量子电路. DORCIS是一种设计和优化可逆电路的工具, 该工具通过高效的算法生成优化的量子电路^[43], 适用于实现复杂的非线性变换(如S盒), 在减少量子门数量和量子比特需求方面具有优越的性能. 算法1给出使用DORCIS工具的 S_0 操作的量子电路的优化实现, 其中 NOT 操作表示图2(a)中的非门, $CNOT$ 操作指图2(b)中的控制非门.

根据算法1, 可得 S_0 操作的量子电路, 如图3所示. 其中, 该量子电路中最后部分电路出现的交叉表示通过改变电路输出的顺序来实现量子交换门的功能, 从

算法1 使用DORCIS工具, 实现 S_0 操作

输入: 4比特输入 $x(x_3, x_2, x_1, x_0)$
输出: 4比特输出 $x(x_3, x_2, x_1, x_0)$

1. $x_0 = CNOT(x_3, x_0)$
2. $x_3 = Toffoli(x_1, x_0, x_3)$
3. $x_2 = CNOT(x_3, x_2)$
4. $x_3 = NOT(x_3)$
5. $x_1 = CNOT(x_2, x_1)$
6. $x_1 = Toffoli(x_3, x_0, x_1)$
7. $x_0 = Toffoli(x_2, x_1, x_0)$
8. $x_3 = CNOT(x_0, x_3)$
9. $x_0 = Toffoli(x_3, x_2, x_0)$
10. $x_2 = CNOT(x_3, x_2)$
11. $x_0 = CNOT(x_1, x_0)$
12. 返回 $x(x_0, x_1, x_3, x_2)$

而避免消耗额外的量子资源.

完整的S盒代换操作量子电路见附录A2, 逆S盒代换操作的量子电路为对应S盒代换操作量子电路的逆过程. 其中, 在量子电路中出现的3控制非门可采用图4所示的分解方式, 即1个3控制非门可通过使用1个辅助量子比特和3个Toffoli门实现.

采用图4所示的分解方式分解后的量子电路可进一步优化. 例如, S_3 操作的量子电路如图5(a)所示, 将电路中出现的3控制非门按照图4方式进行分解, 得到分解后的量子电路(见图5(b)). 观察图5(b)中方框内的

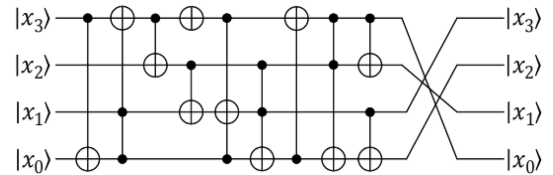


图3 Serpent算法 S_0 操作的量子电路

Figure 3 The quantum circuit for the S_0 operation of the Serpent algorithm.

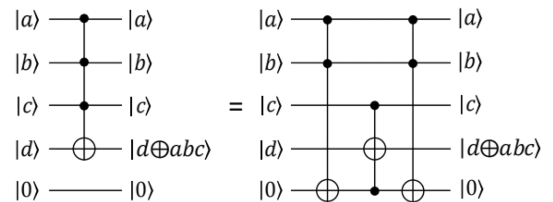


图4 1个3控制非门分解为3个Toffoli门

Figure 4 Decomposition of a 3-controlled $CNOT$ gate into 3 Toffoli gates.

1) <https://github.com/matthewchunqed/dorcis-public>

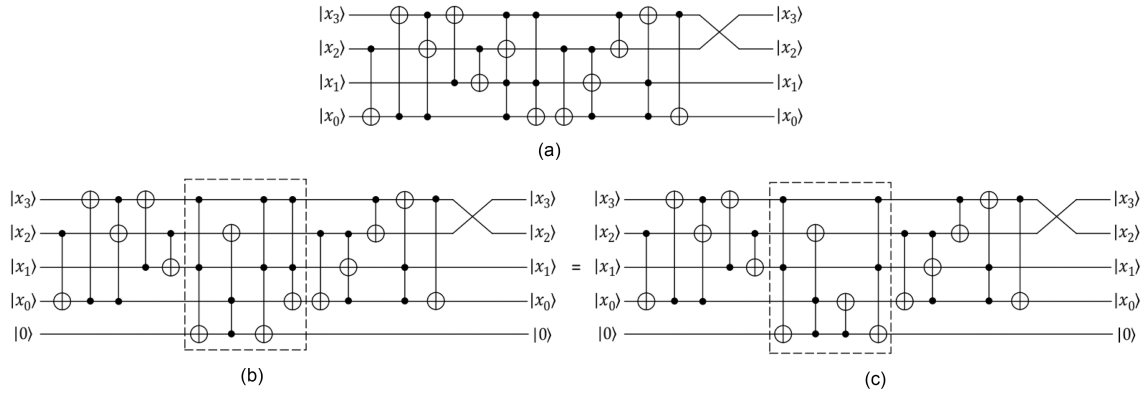


图5 S_3 操作量子电路的分解、优化过程. (a) S_3 操作的量子电路; (b) S_3 操作量子电路的分解; (c) S_3 操作量子电路分解后进行优化

Figure 5 Decomposition and optimization process for the quantum circuit of the S_3 operation. (a) The quantum circuit for the S_3 operation; (b) decomposition of quantum circuit for the S_3 operation; (c) optimization after decomposition of quantum circuit for the S_3 operation.

4个Toffoli门可以发现, 对辅助量子比特进行操作的第1、3个Toffoli门和第4个Toffoli门操作的控制量子比特一致, 仅目标量子比特不同. 并且, 在操作完图5(b)中方框内的第1个Toffoli门后, 辅助量子比特的状态与第4个Toffoli门需要对目标量子比特进行异或的状态一致. 因此, 可以在对辅助量子比特执行回退清空操作之前, 将辅助量子比特的状态通过控制非门异或到第4个Toffoli门的目标量子比特上, 实现第4个Toffoli门所需执行的功能, 即可得到优化后的量子电路(见图5(c)). 优化后的量子电路相比优化前的电路(即图5(b))减少1个Toffoli门, 增加1个控制非门, 同时降低1 Toffoli深度. 由于控制非门的实现难度通常小于Toffoli门, 因此通过增加控制非门的数量以减少Toffoli门的使用是一种有效的优化策略.

类似地, 对部分S盒的量子电路进行优化后, 表1给出Serpent算法S盒实现所需的量子资源.

在32轮的加密过程中, 每个S盒应用4轮, 每轮并行执行32次. 因此, 经过计算可得Serpent加密过程中S盒代换操作实现的量子电路宽度为160, 包括32个辅助量子比特, 需要5120个Toffoli门, 6400个控制非门和1536个非门, Toffoli深度为160.

(3) 线性变换的量子电路. 线性变换操作主要涉及循环左移和逻辑左移两个操作. 为了更清晰地描述电路结构, 首先定义两个操作 RL^j 和 LL^j , 分别代表循环左移 j 位(图6(a))和逻辑左移 j 位(图6(b)), $0 \leq j \leq 31$. 其中, 完成一次 RL^j 操作只需要改变输出量子比特的排列, 不

涉及任何量子操作. 完成一次 LL^j 操作需要 $(32-j)$ 个控制非门. 因此, 线性变换操作的量子电路可用图7表示.

表1 Serpent算法S盒实现所需的量子资源

Table 1 Quantum resources required for S-boxes implementation of the Serpent algorithm

	控制非门	Toffoli门	非门	Toffoli深度	辅助量子比特
S_0	6	4	1	4	0
S_1	5	4	2	4	0
S_2	6	4	1	4	0
S_3	8	6	0	6	1
S_4	7	6	2	6	1
S_5	6	6	2	6	1
S_6	5	4	2	4	0
S_7	7	6	2	6	1

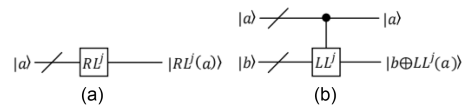


图6 线性变换操作过程中定义的两个操作. (a) 循环左移 j 位; (b) 逻辑左移 j 位

Figure 6 Two operations defined during the linear transformation operation. (a) Rotate left by j qubits; (b) shift logical left j qubits.

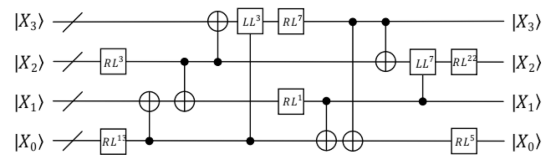


图7 线性变换操作的量子电路

Figure 7 The quantum circuit for the linear transformation operation.

经过计算可得, 一次完成的线性变换操作需要 $32 \times 6 + (32 - 3) + (32 - 7) = 246$ 个控制非门. 在前 31 次轮函数操作过程中均有 1 次线性变换操作, 第 32 次轮函数操作不需要线性变换操作. 因此, 在加密过程中所有线性变换操作共需要 $31 \times 246 = 7626$ 个控制非门.

3.2 密钥扩展的量子电路

为了更清晰地描述电路结构, 如图 8(a)和(b)所示定义两个操作 F 和 G^j , 分别表示将常数 $\phi = 0x9e3779b9$ 和预处理密钥次序 j 异或在当前状态上, 因此, 一次 F 操作需要使用 20 个非门, 而一次 G^j 操作最多需要使用 7 个非门(因为 $0 \leq j \leq 131$).

将长度为 256 比特的初始密钥由低位到高位分为两个数据块, 分别记为 W_0, W_1 , 每个数据块均为 128 比特. 密钥扩展过程的量子电路如图 9 所示, 其中 $\mathcal{KE}_i$ ($0 \leq i \leq 32$) 表示第 i 次获得预处理密钥 $\{\omega_{4i}, \omega_{4i+1}, \omega_{4i+2}, \omega_{4i+3}\}$ 的过程. 需要注意的是, 中间

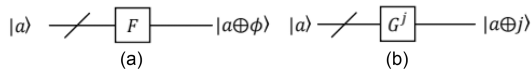


图 8 密钥扩展过程中定义的两个操作. (a) 将常数 $\phi = 0x9e3779b9$ 异或到当前状态上; (b) 将预处理密钥次序 j 异或到当前状态上

Figure 8 Two operations defined in the key expansion process. (a) XOR the current state with the constant $\phi = 0x9e3779b9$; (b) XOR the current state with j which is the sequence of prekeys.

轮次的电路结构与前两轮的电路相同, 因此在图中被省略. 而图 9 中展示的密钥扩展过程最后两轮的电路结构, 与前面轮次相比存在一定差异, 将 $\mathcal{KE}_{32}$ 提前至 $\mathcal{KE}_{31}$ 后, 再分别执行 S_2, S_3 操作, 可以得到第 31 轮的子密钥 K_{31} 和第 32 轮的子密钥 K_{32} , 无需进行 S_2 和 S_3 的逆操作. 这种差异是为在构建 Serpent 算法量子电路时减少量子门数量和电路深度.

在图 9 所示的密钥扩展过程中, 共使用 S_3 操作 $5 \times 32 = 160$ 次, S_2^{-1} 操作 $3 \times 32 = 96$ 次, 其余 S 盒与逆 S 盒代换操作各 $4 \times 32 = 128$ 次. 因此, 密钥扩展过程的量子电路宽度为 288, 包括 32 个辅助量子比特, 需要 10304 个 Toffoli 门, 25536 个控制非门和 6604 个非门, Toffoli 深度为 322.

综上, 利用加密过程与密钥扩展的量子电路, 可构建 Serpent 算法的量子电路. 电路的输入是 128 比特明文 M , 经过 1 次初始置换、32 次轮函数操作(在最后一轮中, 线性变换操作被 1 个额外的轮密钥加操作所取代)、1 次逆初始置换, 生成 128 比特的密文 C . 图 10 展示 Serpent 算法的量子电路, 其中, IP 和 FP 分别表示初始置换与逆初始置换的过程, LT 表示线性变换操作的过程(图 7), 逆 S 盒代换操作的量子电路为对应 S 盒代换操作量子电路的逆过程.

此外, 若将 S 盒代换的量子电路中出现的 3 控制非门的按照如图 11 的分解方案进行分解, 虽然会增加 Toffoli 门的数量, 但是将不再需要辅助量子比特.

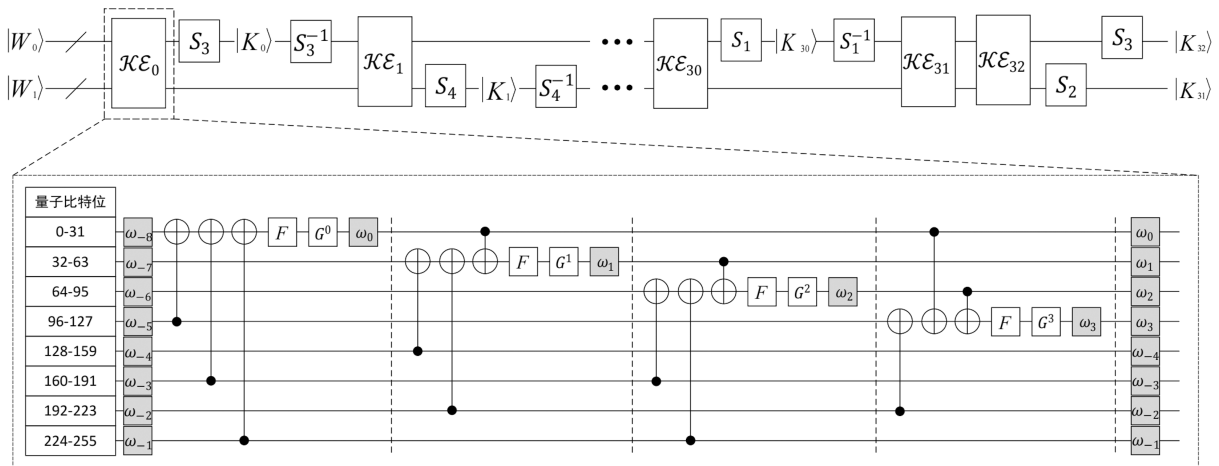


图 9 密钥扩展过程的量子电路

Figure 9 The quantum circuit for the key expansion process.

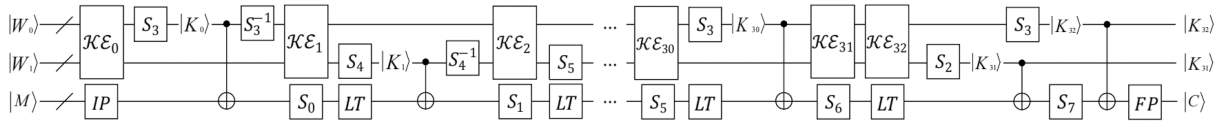


图 10 Serpent算法的量子电路

Figure 10 The quantum circuit for Serpent algorithm.

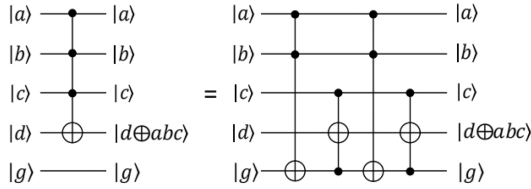


图 11 1个3控制非门分解为4个Toffoli门

Figure 11 Decomposition of a 3-controlled CNOT gate into 4 Toffoli gates.

对Serpent算法的量子电路做资源估计, 包括加密过程和密钥扩展. 对Serpent算法解密的量子电路为Serpent算法量子电路的逆过程. 初始化量子态需要的资源取决于给定的明文与初始密钥, 在资源估计中不考虑这一部分. 因此, 实现Serpent算法的量子电路所需的量子资源如表2所示, 其中方案1表示量子电路中的3控制非门按照图4的方案进行分解, 方案2表示量子电路中的3控制非门按照图11的方案进行分解.

为了与文献[44]进行比较, 需要将Toffoli门分解为T门和Clifford门的组合. 文献[45]给出T门个数为7、T深度为4和T门个数为7、T深度为3的Toffoli门分解方案. 文献[46]给出T门个数为7、T深度为1的Toffoli门分解方案. 文献[47]给出T门个数为4、T深度为1的QAND门(图12(a)), 可用于部分Toffoli门的分解, 但需要目标量子比特和辅助量子比特的初始状态为 $|0\rangle$. 使用上述4种门分解方案, 将本文设计的Serpent加密过程量子电路的资源估计与文献[44]的资源估计进行比较, 如表3所示. 并且, 在所设计Serpent加密量子电路中的逆Toffoli门可采用图12(b)所示的电路进行实现, 不需要使用T门, 可进一步降低T门个数及T深度.

表 2 Serpent算法实现所需的量子资源

Table 2 Quantum resources required for Serpent algorithm implementation

	控制非门	Toffoli门	非门	Toffoli深度	量子比特	辅助量子比特
方案1	43786	15424	8140	482	416	32
方案2	43786	16992	8140	531	384	0

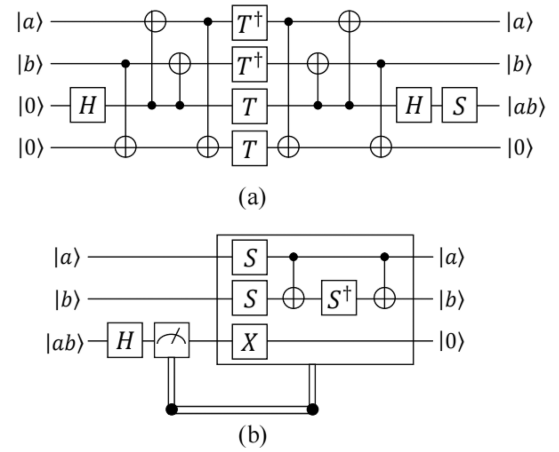


图 12 文献[47]提出的量子门. (a) QAND门; (b) QAND†门

Figure 12 Quantum gates proposed in ref. [47]. (a) QAND gate; (b) QAND† gate.

表 3 本文设计的Serpent电路所需量子资源与文献[44]的比较

Table 3 Comparison of the quantum resources required for Serpent circuit designed in this paper with the ref. [44]

	Toffoli门分解方案	Clifford门	T门	T深度	量子比特
方案1	T门个数为7、T深度为4	178454	92288	1585	448
	T门个数为7、T深度为3	190742	92288	1201	448
	T门个数为7、T深度为1	301334	92288	433	544
方案2	T门个数为7、T深度为4	187862	118944	2124	384
	T门个数为7、T深度为3	204854	118944	1593	384
	T门个数为7、T深度为1	357782	118944	531	512
文献[44]		256314	132608	—	800

由表3分析可得, 本文提出的Serpent算法的量子电路所需的T门数量和量子比特数量相较于已有方案有所减少. 其中, T门数量由132608降为92288 (118944), 降低约30.4% (10.3%); 量子比特的数量由800降为448 (384), 优化44% (52%).

4 对Serpent算法实现Grover搜索攻击

下面讨论对Serpent算法进行Grover搜索攻击所需

的量子资源.

4.1 实现Grover搜索攻击的Oracle

根据文献[47], 需要 $r = \lceil k/n \rceil$ 个已知明密文对得以寻找到唯一的解密密钥, 其中 n 是分组密码的块大小, k 是加密方案的密钥长度. 如果 k/n 是一个整数, 那么应该考虑 $r = k/n + 1$, 此时Grover搜索攻击成功的概率为 $e^{-2^{k-n}}$. 因此, 对于Serpent算法(分块大小为128, 密钥长度为256), 实现Grover搜索攻击的Oracle需要3个已知的明密文对(此时Grover搜索攻击成功的概率为 $e^{-2^{-128}} \approx 1$), 图13给出针对已知Serpent算法的3个明密文对的Grover搜索攻击的Oracle构造, 其中Serpent代表Serpent算法, $Serpent^\dagger$ 代表Serpent算法的逆过程.

Grover搜索攻击的Oracle将Serpent算法的 n 位输出与给定的 r 个密文进行比较, 此操作可以通过使用 $(2n \cdot r)$ 控制的非门来实现. 此处同样忽略一些依赖于给定密文的非门. 在资源估计中只考虑Serpent算法的深度, 而忽略在比较密文中使用的多控制非门. Grover搜索攻击的Oracle实现所需的量子资源如表4所示.

4.2 Grover搜索攻击的资源估计

根据对Grover搜索攻击的Oracle的资源估计, 可以对完整的Grover搜索攻击进行资源估计. 相较于实现Grover搜索攻击的Oracle (章节2.2中的 O_f)所需的量子资源消耗, 实现Grover迭代算子(章节2.2中的 $(2|\psi\rangle\langle\psi| - D)O_f$)的其他量子操作所需资源的量级相对较小, 因此在分析Grover搜索攻击的总体资源需求时, 可以合理地忽略这些量子资源的消耗. 所以, 对Serpent算法实现Grover搜索攻击所需的量子资源如表5所示.

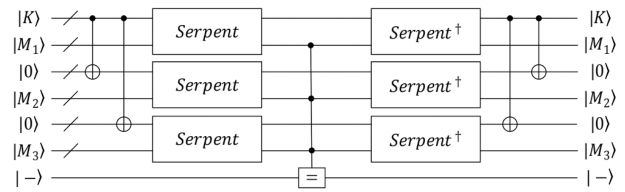


图 13 针对Serpent算法, 使用3个明密文对的Grover搜索攻击的Oracle

Figure 13 Oracle of Grover's search attack for Serpent algorithm using 3 plaintext-ciphertext pairs.

表 4 Grover搜索攻击的Oracle实现所需的量子资源

Table 4 Quantum resources required for Oracle of Grover's search attack implementation

	控制非门	Toffoli门	非门	Toffoli深度	量子比特	辅助量子比特
方案1	263740	92544	48840	964	1249	609
方案2	263740	101952	48840	1062	1153	513

表 5 对Serpent算法实现Grover搜索攻击所需的量子资源

Table 5 Quantum resources required to implement Grover's search attack for Serpent algorithm

	控制非门	Toffoli门	非门	Toffoli深度	量子比特	辅助量子比特
方案1	1.0060 $\times 2^{145.65}$	1.0055 $\times 2^{144.14}$	1.0040 $\times 2^{143.155}$	1.0090 $\times 2^{137.55}$	1249	609
方案2	1.0060 $\times 2^{145.65}$	1.0052 $\times 2^{144.28}$	1.0040 $\times 2^{143.155}$	1.0018 $\times 2^{137.7}$	1153	513

5 总结

本文提出一个改进的Serpent算法的可逆量子电路, 然后利用Grover算法对该密码算法进行攻击, 并给出实现该攻击所需的量子资源估计. 与基于S盒代数关系设计的量子电路相比, 本文设计的S盒代换操作的量子电路显著减少所需的量子比特数量, 提高电路的效率和可实施性. 因此, 本文提出的改进的Serpent算法的量子电路在所需的门数量与量子比特数量上相比现有方案有显著减少.

参考文献

- Huang H L, Xu X Y, Guo C, et al. Near-term quantum computing techniques: Variational quantum algorithms, error mitigation, circuit compilation, benchmarking and classical simulation. *Sci China-Phys Mech Astron*, 2023, 66: 250302
- Song Y, Wu Y, Wu S, et al. A quantum federated learning framework for classical clients. *Sci China-Phys Mech Astron*, 2024, 67: 250311
- Nielsen M A, Chuang I L. Quantum Computation and Quantum Information. Cambridge: Cambridge University Press, 2010

- 4 Shor P W. Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceeding of 35th Annual Symposium on Foundations of Computer Science. Santa Fe: IEEE, 1994. 124–134
- 5 Shor P W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev*, 1999, 41: 303–332
- 6 Grover L K. A fast quantum mechanical algorithm for database search. In: Proceeding of 28th Annual ACM Symposium on Applied Computing. New York: ACM, 1996. 212–219
- 7 Simon D R. On the power of quantum computation. *SIAM J Comput*, 1997, 26: 1474–1483
- 8 Cleve R, Ekert A, Macchiavello C, et al. Quantum algorithms revisited. *Proc Roy Soc Lond A Math Phys Eng Sci*, 1998, 454: 339–354
- 9 Coppersmith D. An approximate Fourier transform useful in quantum factoring. arXiv:quant-ph/0201067
- 10 Kitaev A Y. Quantum measurements and the Abelian stabilizer problem. arXiv:quant-ph/9511026
- 11 Rivest R L, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. *Commun ACM*, 1978, 21: 120–126
- 12 Li H W, Hao C P, Chen Z J, et al. Security of quantum key distribution with virtual mutually unbiased bases. *Sci China-Phys Mech Astron*, 2024, 67: 270313
- 13 Lv M Y, Hu X M, Gong N F, et al. Demonstration of controlled high-dimensional quantum teleportation. *Sci China-Phys Mech Astron*, 2024, 67: 230311
- 14 Du S H, Ruan C G. Research on block cipher (in Chinese). *Commun Technol Dev*, 1995, 4: 45–49 [杜生辉, 阮传概. 分组密码及其研究. 通信技术与发展, 1995, 4: 45–49]
- 15 Lin D J, Lin B G, Lin D Q. Status quo and trend of block's cryptography (in Chinese). *Tianjin Commun Technol*, 2002, 4: 1–7 [林德敬, 林柏钢, 林德清. 国内外分组密码理论与技术的研究现状及发展趋势. 天津通信技术, 2002, 4: 1–7]
- 16 National Institute of Standards and Technology. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. National Institute of Standards and Technology, 2016. <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>
- 17 Yamamura A, Ishizuka H. Quantum cryptanalysis of block ciphers. *RIMS Kokyuroku*, 2000, 1166: 235–243
- 18 Grassl M, Langenberg B, Roetteler M, et al. Applying Grover's algorithm to AES: Quantum resource estimates. In: Proceeding of International Workshop on Post-Quantum Cryptography. Cham: Springer International Publishing, 2016. 29–43
- 19 Almazrooe M, Samsudin A, Abdullah R, et al. Quantum reversible circuit of AES-128. *Quantum Inf Process*, 2018, 17: 1–30
- 20 Anand R, Maitra S, Maitra A, et al. Resource estimation of grovers-kind quantum cryptanalysis against fsr based symmetric ciphers. *cryptoeprint: 2020/1438*, 2020
- 21 Jang K, Choi S, Kwon H, et al. Grover on Korean Block Ciphers. *Appl Sci*, 2020, 10: 6407
- 22 Jang K, Kim H, Eum S, et al. Grover on GIFT. *cryptoeprint: 2020/1405*, 2020
- 23 Langenberg B, Pham H, Steinwandt R. Reducing the cost of implementing the advanced encryption standard as a quantum circuit. *IEEE Trans Quantum Eng*, 2020, 1: 1–12
- 24 Li Y J, Yi Z H, Wang Z, et al. Quantum cryptanalysis of lightweight cipher TWINE-128 (in Chinese). *J Cryptol Res*, 2022, 9: 633–643 [李艳俊, 易子晗, 汪振, 等. 轻量级密码TWINE-128的量子密码分析. 密码学报, 2022, 9: 633–643]
- 25 Du X N, Wang X Y, Liang L F, et al. Quantum cryptanalysis of lightweight block cipher Piccolo (in Chinese). *J Commun*, 2023, 44: 175–182 [杜小妮, 王香玉, 梁丽芳, 等. 轻量级分组密码Piccolo的量子密码分析. 通信学报, 2023, 44: 175–182]
- 26 Li Y J, Ge Y D, Wang Q, et al. Improved KLEIN algorithm and its quantum analysis (in Chinese). *J Comput Appl*, 2024, 44: 2810–2817 [李艳俊, 葛耀东, 王琦, 等. 改进的KLEIN 算法及其量子分析. 计算机应用, 2024, 44: 2810–2817]
- 27 Dong X Y. A survey of quantum cryptanalysis of symmetric cryptography (in Chinese). *J Cryptol Res*, 2024, 11: 159–173 [董晓阳. 对称密码的量子分析法综述. 密码学报, 2024, 11: 159–173]
- 28 Feng X N, Wu H Y. Quantum attacks on symmetric cryptosystems (in Chinese). *J Appl Sci*, 2024, 42: 39–52 [冯晓宁, 吴洪宇. 对称密码体制的量子攻击. 应用科学学报, 2024, 42: 39–52]
- 29 Amy M, Di Matteo O, Gheorghiu V, et al. Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. In: Proceeding of International Conference on Selected Areas in Cryptography. Cham: Springer International Publishing, 2016. 317–337
- 30 Wright M A. The advanced encryption standard. *Netw Security*, 2001, 2001: 11–13
- 31 Rijmen V, Daemen J. Advanced encryption standard. *Proc Fed Inf Process Stand Publ Nat Inst Stand Technol*, 2001, 19: 22
- 32 Dobbertin H, Knudsen L, Robshaw M. The Cryptanalysis of the AES — A Brief Survey. In: Dobbertin H, Rijmen V, Sowa A, eds. *Advanced Encryption Standard — AES*. AES 2004. Heidelberg: Springer Berlin, 2004
- 33 Joan D, Vincent R. The Design of Rijndael: AES—the Advanced Encryption Standard. Heidelberg: Springer Berlin, 2002. 196
- 34 Biham E, Dunkelman O, Keller N. Differential-linear cryptanalysis of serpent. In: Proceeding of Software Encryption, 10th International Work-

- shop. Heidelberg: Springer Berlin, 2003. 9–21
- 35 Knudsen L R, Rijmen V. Known-key distinguishers for some block ciphers. In: Proceeding of 13th International Conference on the Theory and Application of Cryptology and Information Security. Kuching: CTAC, 2007. 315–324
- 36 Dunkelman O, Indestege S, Keller N. A differential-linear attack on 12-round Serpent. In: Proceeding of Progress in Cryptology-INDOCRYPT 2008 9th International Conference on Cryptology in India. Heidelberg: Springer Berlin, 2008. 308–321
- 37 Hu Z H. Differential algebraic attack of Serpent (in Chinese). J Beijing Univ Technol, 2010, 36: 651–653 [胡志华. Serpent加密算法的差分代数攻击. 北京工业大学学报, 2010, 36: 651–653]
- 38 Tezcan C, ozbudak F. Differential factors: Improved attacks on SERPENT. Int Workshop Lightw Cryptogr Secur Privacy, 2014, 3: 69–84
- 39 Chen S Z, Fu Z X, Ren J J. New results of differential-linear cryptanalysis on reduced-round Serpent (in Chinese). J Inf Eng Univ, 2020, 21: 699–704 [陈少真, 付志新, 任炯炯. 减轮Serpent算法差分-线性分析的新结果. 信息工程大学学报, 2020, 21: 699–704]
- 40 Hadipour H, Derbez P, Eichlseder M. Revisiting differential-linear attacks via a boomerang perspective with application to AES, Ascon, CLEFIA, SKINNY, PRESENT, KNOT, TWINE, WARP, LBlock, Simeck, and SERPENT. In: Reyzin L, Stebila D, eds. Advances in Cryptology-CRYPTO 2024. CRYPTO 2024. Cham: Springer, 2024. 38–72
- 41 Florez-Gutierrez A, Todo Y. Improving linear key recovery attacks using Walsh spectrum puncturing. In: Joye M, Leander G. eds. Advances in Cryptology-EUROCRYPT 2024. Cham: Springer, 2024. 187–216
- 42 Yang Q, Song L, Zhang N, et al. Optimizing rectangle and boomerang attacks: A unified and generic framework for key recovery. J Cryptol, 2024, 37: 1–62
- 43 Chun M, Baksi A, Chattopadhyay A. Dorcis: Depth optimized quantum implementation of substitution boxes. Cryptol ePrint Arch, 2023, 2023: 286
- 44 Langenberg B W. Quantum Circuits for Symmetric Cryptanalysis. Boca Raton: Florida Atlantic University, 2018.
- 45 Amy M, Maslov D, Mosca M, et al. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Trans Comput-Aided Des Integr Circuits Syst, 2013, 32: 818–830
- 46 Selinger P. Quantum circuits of T-depth one. Phys Rev A, 2013, 87: 042302
- 47 Jaques S, Naehrig M, Roetteler M, et al. Implementing Grover oracles for quantum key search on AES and LowMC. In: Proceeding of Advances in Cryptology-EUROCRYPT 2020 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques. Zagreb, 2020. 280–310

附录

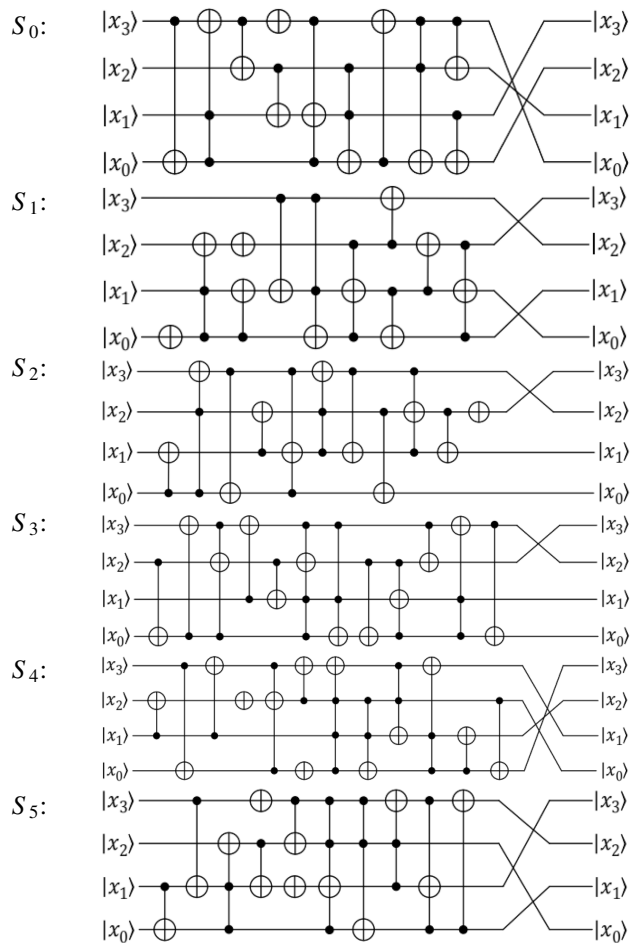
A1 S盒

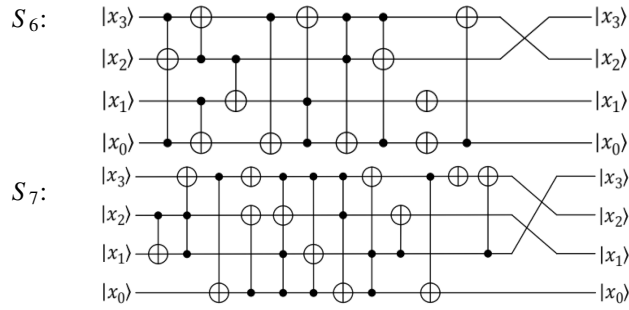
S盒输入 m , 可以查找到对应数值 n , 那么意味着输入 m , 经过一次S盒代换操作后输出 n . 例如, 输入值为3(0011), 经过操作, 输出值为1(0001). 以下是Serpent算法的8个 4×4 的S盒代换操作规则, 从 S_0 到 S_7 分别是:

S_0 :	3	8	15	1	10	6	5	11	14	13	4	2	7	0	9	12
S_1 :	15	12	2	7	9	0	5	10	1	11	14	8	6	13	3	4
S_2 :	8	6	7	8	3	12	10	15	13	1	14	4	0	11	5	2
S_3 :	0	15	11	8	12	9	6	3	13	1	2	4	10	7	5	14
S_4 :	1	15	8	3	12	0	11	6	2	5	4	10	9	14	7	13
S_5 :	15	5	2	11	4	10	9	12	0	3	14	8	13	6	7	1
S_6 :	7	2	12	5	8	4	6	11	14	9	1	15	13	3	10	0
S_7 :	1	13	15	0	14	8	2	11	7	4	12	10	9	3	5	6

A2 S盒代换操作的量子电路

以下是Serpent算法S盒代换操作的量子电路, 从 S_0 到 S_7 分别是:





Quantum circuit for implementing Serpent algorithm with low costs

CHEN HuiNan¹, ZHU YanYing¹, CAI BinBin^{1,2*} & LIU XiaoFen^{1*}

¹College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China

²Digital Fujian Internet-of-Things Laboratory of Environmental Monitoring, Fujian Normal University, Fuzhou 350117, China

*Corresponding authors (CAI BinBin, email: cbb@fjnu.edu.cn; LIU XiaoFen, email: liuxiaofen23@fjnu.edu.cn)

Serpent, one of the AES candidate algorithms, is known for its high security and attack resistance. It is suitable for areas with high information security requirements. Serpent's distinctive nonlinear structure is the S-boxes. This paper studies how to build the quantum circuit for Serpent algorithm with lower costs. By combining with the quantum circuits of S-boxes with fewer qubits and quantum gates and the inverse quantum circuits of S-boxes, the key expansion quantum circuit can be constructed. Moreover, by decomposing the 3-controlled *CNOT* gates appearing in the quantum circuits of the S-boxes and optimizing the decomposed quantum circuits, the number of *Toffoli* gates is further reduced. In conjunction with the proposed quantum circuits, two types of quantum circuits requiring 448 qubits and 92288 *T* gates and 384 qubits and 118944 *T* gates are given to realize the Serpent algorithm. In comparison with the previous study, which required 800 qubits and 132608 *T* gates, the quantum circuit in this paper requires fewer resources. Furthermore, the S-boxes and key expansion quantum circuits used in the study can reduce the quantum resources in implementing Serpent, reducing the quantum circuit scale required to attack Serpent by the Grover algorithm.

Serpent, S-box, quantum circuit, Grover algorithm, resource estimation

PACS: 03.67.Lx, 03.67.Dd, 03.65.Ta

doi: [10.1360/SSPMA-2024-0550](https://doi.org/10.1360/SSPMA-2024-0550)