ZUC 算法的量子电路实现

孙 壮[1,2], 黄震宇[1,2]

[1] 中国科学院信息工程研究所 信息安全国家重点实验室 北京 中国 100093
[2] 中国科学院大学 网络空间安全学院 北京 中国 100049

摘要 近年来, 量子计算相关技术在不断的发展, 其对传统密码算法体制产生了一定的冲击。例如, Shor 算法使得基于大整数分解、求离散对数等数学问题设计的公钥密码算法不再安全。相较而言, 量子计算对对称密码算法的冲击要小一些, 能对对称密码算法产生威胁的攻击方法主要有两类。第一类是在经典询问的条件下, 使用 Grover 算法进行密钥的搜索。第二类是在量子询问的条件下, 使用 Simon 算法进行秘密信息的恢复。以上两类攻击方法, 都需要用到被攻击算法所对应的量子预言机(quantum oracle)。其主要组成部分是基于 Clifford + Toffoli 等通用容错量子门集实现的对称密码算法的量子电路。因此密码算法量子电路的规模与攻击的复杂度密切相关。研究对称密码算法的量子电路有助于更好地量化对称密码算法抵抗量子攻击的能力。本文

基于 Clifford + Toffoli 的量子电路模型将 ZUC 算法实现为量子电路, 这是首次将流密码算法实现为量子电路。本文的主要目标是通过构造消耗量子比特数较少的量子电路模型, 给出一个基于量子算法攻击 ZUC 算法的量子比特数的下界, 从而说明当量子计算机能够支持的逻辑量子比特数达到何种规模时, 可能会对 ZUC 算法产生实质性的威胁。针对此目标, 本文的电路设计准则是, 优先考虑减少量子比特的消耗, 并在此基础上优化 Toffoli 门的消耗。本文针对 ZUC 算法的各个关键部件, 如有限域

$GF(2^{31}-1)$ 上的模加器, 有限域 $GF(2^{31}-1)$ 上的线性反馈移位寄存器以及 S 盒等, 给出了详细的量子电路实现方案。基于这些关键组件的实现, 本文给出了 ZUC 算法的整体量子电路实现方案。基于此方案, 要实现 ZUC 算法的初始化过程并生成 128 比特的密钥流, 我们需要 752 个量子比特, 109770 个 Toffoli 门, 348117 个 CNOT 门, 以及 26912 个 Pauli X 门。

关键词 量子密码分析; 量子电路; 流密码; ZUC

中图法分类号 TP309.5 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2023.06.08

Implementing Quantum Circuit of ZUC Algorithm

SUN Zhuang[1,2], HUANG Zhenyu[1,2]

[1]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract In recent years, the development of quantum computing has had an impact on the classical cryptographic algorithms. Shor's algorithm makes some asymmetric cryptographic algorithms, which are based on factorization or discrete logarithm, no longer secure. On the other hand, the impact of quantum computing on symmetric cryptography algorithms is less. There are two main attck methods that can pose threats to symmetric cryptographic algorithms. The first method is using Grover's algorithm to search the key in the classical query condition. The second method is using Simon's algorithm to recover the secret information in the quantum query condition. Both of these methods need the corresponding quantum oracles of the attacked algorithm. Its main component is quantum circuits of symmetric cryptographic algorithms imple-mented with universal fault-tolerant quantum gate set. This makes the scales of such quantum circuits greatly influence the complexity of these attacks. Therefore, investigating the quantum circuits of symmetric cryptographic algorithms can help us quantify the resistance of symmetric cryptographic algorithms against these quantum attacks. In this paper, the stream cipher ZUC is implemented as a quantum circuit based on the “Clifford + Toffoli” quantum gate set. This is the first time that a stream cipher is implemented as a quantum circuit. By constructing this quantum circuit that consumes as few qubits as possible, we give a lower bound on the number of qubits that quantum algorithms can attack the ZUC algorithm. When the number of logical qubits that a quantum computer can support exceeds this lower bound, it may pose a substantial threat to the ZUC algorithm. According to this goal, our circuit design criterion is to consider reducing the consumption of qubits firstly, and on this basis to optimize the consumption of Toffoli gates. we give detailed quantum circuit design for

each key component of ZUC, such as the modulo adder over the finite field $GF(2^{31}-1)$, the linear feedback shift register over $GF(2^{31}-1)$, and the S-boxes. Based on the quantum circuits of these components, we present the overall quantum

circuit design of ZUC. According to the design in this paper, 752 qubits, 109770 Toffoli, 348117 CNOT, and 26912 Pauli X gates are needed for ZUC to complete its initialization process and generate 128-bit keystream.

Key words quantum cryptanalysis; quantum circuit; stream cipher; ZUC

通信作者 孙壮, 硕士, E-mail: ...@ii...

1 引言

随着量子计算技术的不断发展, 越来越多的学者开始关注经典密码算法在后量子情景下的安全性。众所周知, Shor 算法[1-2]可以被用来解决因子分解和离散对数问题。它的问世使得诸如 RSA、ECDSA 以及 ECDH 等非对称密码算法变得不再安全。相较而言, 对称密码算法受到的影响较小, 能对对称密码算法产生威胁的主要攻击方法有两类。一是在经典询问条件下运用 Grover 算法进行密钥搜索[3], 二是在量子询问条件下运用 Simon 算法进行秘密信息的恢复[4-5]。不论上述哪种攻击方法, 密码算法所对应的量子预言机的实现都是必不可少的。具体来说, 在经典询问模型下, 攻击者为了运用 Grover 算法来进行密钥搜索, 需要通过量子电路实现利用密钥叠加态对指定明文加密的过程。在量子询问模型下, 攻击者需要一个基于固定密钥加密的黑盒量子电路, 其能够对明文叠加态的输入返回相应的密文叠加态输出。这两类量子黑盒电路的主要部分都是基于量子门实现的密码算法加密过程。因此, 将对称密码算法的加密过程实现为量子电路的最低量子资源消耗, 能够帮助说明量子计算机的物理规模达到何种程度时, 基于量子计算的密码攻击才能对对称密码算法产生真正的威胁。

近年来, 有许多文献在关注如何利用更少的量子计算资源将分组密码算法 AES 实现为量子电路的问题, 如Grassl等人[6]给出了三种AES算法的量子电路, 以及将其应用于Grover算法所消耗资源的估计。Almazrooie 等人[7]通过借鉴文献[8]的方法, 减少了非线性运算所需的辅助量子比特数量, 给出了新的 AES-128 算法的量子电路。Kim 等人[9]给出了在进行 AES 算法的密钥搜索时, 权衡时间与空间的一种度量方法。Langenberg 等人[10]通过借鉴经典电路的设计方法, 给出了优化后的 AES 算法 S 盒的量子电路, 减少了量子比特和 Toffoli 门的消耗。Zou 等人[11]进一步减少了 AES 算法 S 盒量子电路消耗的量子比特, 并将 S 盒的逆运算引入电路, 减少了 zig-zag 设计下 AES 算法量子电路中量子比特的消耗。Jaques 等人[12]在考虑 NIST 提出的最大电路深度限制的前提下, 以深度和宽度的乘积作为衡量指标, 给出了较优的 AES 算法的量子电路。但是, 关于如何利用较少的量子计算资源将流密码算法实现为量子电路, 还没有公开的研究成果。因此, 本文选取了流密码中具有代表性的 ZUC 算法作为研究对象, 探究如何用较少的量子计算资源将该算法的初始化与密钥流生成过程实现为量子电路。

现如今量子计算机的发展还不成熟, 已知的比较著名的量子计算机原型有国内的“九章”量子计算原型机, 以及谷歌和 IBM 的一系列量子计算机原型。这些量子计算机在介绍与宣传时, 往往会强调量子比特数。从“悬铃木”的 53 个量子比特, 到“九章”的 76 个量子比特, 随着技术的不断进步, 量子计算机拥有的量子比特数也在逐渐增多。同时, 有许多相关文献[6-7,9-11], 在进行电路设计时以量子比特作为优化目标, 力求用较少的量子比特实现相关电路。在本文的量子电路设计中, 我们同样考虑尽可能地减少量子比特的消耗。我们希望通过本文的电路设计, 给出一个实现 ZUC 算法加密过程的量子比特数的较紧下界, 从而说明当现实中的量子计算机所支持的逻辑量子比特数大于该下界时, 量子计算才可能会对 ZUC 算法产生实质性的威胁。

就量子计算资源来说, 除了量子比特数, 量子门数也是一个重要的度量指标。从现有的物理研究结果来看, 构建非 Clifford 门的物理代价远高于 Clifford 门。此外由于退相干等因素造成错误, 我们需要基于纠错模型来实现正确的量子逻辑门操作, 而在这些纠错模型中非Clifford门的纠错代价也远高于 Clifford 门。例如, 在现在常用的 Surface Code 纠错模型下, T 门的纠错代价是 Clifford 门的一百倍左右[13]。因此, 在量子门消耗的研究中, Toffoli 门(或 T 门)作为非 Clifford 门中的元素, 其数量的多少更受到关注。除了量子比特与量子门数外, 量子电路的深度是另一个与量子计算机物理能力密切相关的量。量子电路的深度, 即将能够并行执行的量子门放在同一层同时运行, 而运行整个电路所需要的层数则为电路的深度。若一个量子电路拥有较少的深度, 则理论上电路运行的实际时间也会较短, 从而可以降低退相干的影响。同样, 由于非 Clifford 门的实现时间和纠错代价更大, 因此非 Clifford 门的深度(T-depth 或 Toffoli-depth), 是基于深度研究量子计算资源的主要度量指标。在设计加密算法量子电路的已有研究中, 与基于降低量子比特数进行设计的文献相比, 基于降低电路深度的研究较少, 只有在文献[12]中, 作者基于尽量降低 T 深度与总电路深度的设计准则给出了 AES 的另一种量子电路实现。

由于量子比特数、电路深度以及量子门数, 属于不同的优化角度(指标), 在设计电路时侧重考虑的因素不同, 电路总体框架与每个环节的设计会有很大差别。如上所述, 基于本文的目的, 我们首先考虑尽量减少量子比特的消耗, 在此基础上尽量减少物理

资源与纠错代价消耗更大的 Toffoli 门的数量, 之后再考虑优化 CNOT 门与 Pauli X 门的数量。对基于降低量子门数量或基于降低电路深度来设计 ZUC 算法量子电路的问题, 我们将在以后的工作中探究。这里我们需要特别指出, 这几个指标之间存在一定的权衡关系, 降低部分指标的消耗, 可能会增加其他指标的消耗。但一般设计者会保证其他指标也在一个合理的范围内, 即保证电路的深度乘以宽度(即量子比特数)不发生大的变化。例如对于 AES 的 S 盒电路, 文献[6]中给出了只消耗 9 个量子比特, 但同时需消耗 9000 多个 T 门的电路。而文献[11]中的 S 盒电路需要消耗 22 个量子比特以及 364 个 T 门。可以看出, 虽然文献[6]中给出的 S 盒设计消耗的量子比特非常少, 但因其消耗的 T 门数极大, 因此未被采纳到之后任何的关于降低量子比特数的电路设计中。在本文中, 我们同样遵循这个设计原则, 优先考虑减少电路消耗的量子比特数, 但同时会将电路深度和量子门数保持在一个合理的范围。

具体来说, 我们采用量子容错电路通用的 Clifford + Toffoli 门集来构造 ZUC 算法的量子电路。Clifford 门集主要包括 Hadamard 门、CNOT 门、相位门和 Pauli 门等。对于对称密码算法的量子预言机的实现一般只涉及 Clifford 门集中的 CNOT 门与 Pauli X 门, 我们将利用这两类门完成 ZUC 算法量子电路的线性部分的实现, 同时我们会借鉴文献[14]中针对线性可逆电路的优化合成方法来构造电路的线性部分。对于 ZUC 算法中非线性部分的量子电路实现, 我们主要用到了 Toffoli 门, 其可被分解为 Clifford 门与 T 门, 相关的分解方法可以参考文献[15-16]。本文将直接使用 Toffoli 门作为量子电路所耗资源的统计指标, 不再对其进一步分解。有关量子门的更多知识请参考文献[17]。本文将针对 ZUC 算法中的主要部件, 包括有限域 GF($2^{31}-1$) 上的模加器、GF($2^{31}-1$) 上的线性反馈移位寄存器, 以及 S 盒进行探究。

本文将首先介绍 ZUC 算法的具体流程, 接着给出 ZUC 算法中关键组件的量子电路设计, 然后给出 ZUC 算法的整体量子电路设计, 同时对所消耗的量子计算资源进行估算, 最后, 我们对本文工作进行总结。

2 预备知识

在进行 ZUC 算法量子电路的设计之前, 我们首先简要介绍该算法。ZUC 算法是由国内学者自主设计的流密码算法, 其经过多次国内和国际的评估与研讨, 具有公认的较好的安全性。ZUC 算法与 128-EIA3 及 128-EEA3 算法一起, 被 3GPP 推荐为第三套 4G 无线通信的国际加密和完整性标准的候选算法。ZUC 算法是面向字节的流密码, 其以 128 比特的初始密钥和 128 比特的初始向量作为输入, 输出若干个 32 比特长的字节[18]。ZUC 算法的执行分为两个阶段, 初始化阶段和工作阶段。每个阶段都包含三部分: 线性反馈移位寄存器、比特重组以及非线性函数 F。

2.1 线性反馈移位寄存器

ZUC 算法的线性反馈移位寄存器(Linear feedback shift register, LFSR)含有 16 个寄存器单元 $(s_0,s_1,\ldots,s_{15})$, 每个单元都能存储 31 比特的信息。该线性反馈移位寄存器的运算是定义在有限域 $\mathrm{GF}\left(2^{31}-1\right)$ 上的[19]。寄存器单元的初始值由初始密钥、初始向量以及特定常数组成。线性反馈移位寄存器的执行流程如过程 1 和过程 2 所示:

过程 1. 线性反馈移位寄存器(初始化模式)

a) $c_i=\begin{cases}a_0\mathrm{sum}_0\oplus\mathrm{sum}_0 & i=1\\ \left(a_{i-1}\oplus c_{i-1}\right)\left(\mathrm{sum}_{i-1}\oplus c_{i-1}\right)\oplus\mathrm{sum}_{i-1} & 1<i\le 31\end{cases}$

b) $s_{16}=\left(v+u\right)\ \mathrm{mod}\left(2^{31}-1\right)$

c) IF $s_{16}=0$, THEN $s_{16}=2^{31}-1$

d) $\left(s_1,s_2,\ldots,s_{15},s_{16}\right)\to\left(s_0,s_1,\ldots,s_{14},s_{15}\right)$

其中, u 是由非线性函数 F 产生的 31 比特的字串。我们需要特别说明, 与一般的 $\mathrm{mod}\left(2^{31}-1\right)$ 运算不同, 在 ZUC 算法的寄存器单元进行 $\mathrm{mod}\left(2^{31}-1\right)$ 运算过程中, 我们总是用 $2^{31}-1$ 来表示 $\mathrm{mod}\left(2^{31}-1\right)$ 等于 0 的元素。在本文中, 如无特殊说明, $\mathrm{mod}\left(2^{31}-1\right)$ 均指上述特殊的模运算。

过程 2.线性反馈移位寄存器(工作模式)

a) $s_{16}=2^{15}s_{15}+2^{17}s_{13}+2^{21}s_{10}+2^{20}s_4+\left(1+2^8\right)s_0\ \mathrm{mod}(2^{31}-1)$

b) IF $s_{16}=0$, THEN $s_{16}=2^{31}-1$

c) $\left(s_1,s_2,\ldots,s_{15},s_{16}\right)\to\left(s_0,s_1,\ldots,s_{14},s_{15}\right)$

我们注意到, 计算 $2^j s_i\ \mathrm{mod}\left(2^{31}-1\right)$, 等价于将

31 比特的 s_i 循环向左移动 j 比特。也就是说, 线性反馈移位寄存器的第一步, 可以等价地写成公式(1)的形式。其中 $s_i \lll_j k$ 表示 j 比特的 s_i 循环向左移动 k 比特。

$$\begin{aligned} v = &\left(s_{15} \lll_{31} 15\right) + \left(s_{13} \lll_{31} 17\right) + \\ &\left(s_{10} \lll_{31} 21\right) + \left(s_4 \lll_{31} 20\right) + \\ &\left(s_0 \lll_{31} 8\right) + s_0 \mod\left(2^{31}-1\right) \end{aligned} \tag{1}$$

2.2 比特重组

在比特重组阶段, 算法会从 LFSR 的寄存器中选取 128 比特, 组成 4 个 32 比特大小的字。具体过程如下:

过程 3. 比特重组

a) $X_0 = s_{15\mathrm{H}} \| s_{14\mathrm{L}}$

b) $X_1 = s_{11\mathrm{L}} \| s_{9\mathrm{H}}$

c) $X_2 = s_{7\mathrm{L}} \| s_{5\mathrm{H}}$

d) $X_3 = s_{2\mathrm{L}} \| s_{0\mathrm{H}}$

上述表达式中, $s_{i\mathrm{H}}$ 代表寄存器 s_i 中的高 16 位比特($s_{i\mathrm{L}}$ 代表低 16 位比特)。$\|$ 表示将两个字符串进行拼接。X_0, X_1, X_2 会在非线性函数 F 中被使用。X_3 会与 F 的输出进行异或, 产生最终输出。

2.3 非线性函数 F

F 以 X_0, X_1, X_2 为输入, 生成 32 比特长的输出 W。其运算流程如下:

过程 4. 非线性函数 $F\left(X_0, X_1, X_2\right)$

a) $W = \left(X_0 \oplus R_1\right) \boxplus R_2$

b) $W_1 = R_1 \boxplus X_1$

c) $W_2 = R_2 \oplus X_2$

d) $R_1 = S\left(L_1\left(W_{1\mathrm{L}} \| W_{2\mathrm{H}}\right)\right)$

e) $R_2 = S\left(L_2\left(W_{2\mathrm{L}} \| W_{1\mathrm{H}}\right)\right)$

$\boxplus$ 代表有限域 $\mathrm{GF}\left(2^{32}\right)$ 上的模加操作。R_1 和 R_2 是两个 32 比特的寄存器。L_1 和 L_2 是两个线性变换, 其接收 32 比特长的字作为输入, 并生成 32 比特长的输出。L_1 和 L_2 定义如下:

$$\begin{aligned} L_1(x) = &x \oplus \left(x \lll_{32} 2\right) \oplus \left(x \lll_{32} 10\right) \\ &\oplus \left(x \lll_{32} 18\right) \oplus \left(x \lll_{32} 24\right) \end{aligned} \tag{2}$$

$$\begin{aligned} L_2(x) = &x \oplus \left(x \lll_{32} 8\right) \oplus \left(x \lll_{32} 14\right) \\ &\oplus \left(x \lll_{32} 22\right) \oplus \left(x \lll_{32} 30\right) \end{aligned} \tag{3}$$

过程4中的 S 表示一个 32×32 的S盒, 其是由四个 8×8 的 S 盒并行组成的, 即 $S = \left(S_0, S_1, S_0, S_1\right)$。$S_0$ 是基于 Feistel 结构及有限域 $\mathrm{GF}(16)$ 上的置换操作构造的 S 盒, 其接收两个 4 比特的字符串作为输入。我们用 x_1 和 x_2 代表 S_0 的输入, 用 P_1、P_2 和 P_3 来代表 S_0 用到的置换操作。那么 S_0 的运算过程, 可以表示成如下形式:

$$S_0\left(x_1 \| x_2\right) = \left(\left(x_1 \oplus P_1(x_2) \oplus P_3\left(P_2\left(x_1 \oplus P_1(x_2)\right) \oplus x_2\right)\right) \| \left(P_2\left(x_1 \oplus P_1(x_2)\right) \oplus x_2\right)\right) \lll_{32} 5$$

S_1 的结构类似于 AES 算法的 S 盒。但与 AES 算法不同, 该 S 盒中生成有限域 $\mathrm{GF}(2^8)$ 的不可约多项式为 $x^8 + x^7 + x^3 + x + 1$。具体来说, S_1 可以表示成:

$$S_1 = Ax^{-1} + B$$

其中, A 是有限域 GF(2) 上的一个 8×8 的矩阵, x^{-1} 表示有限域 $\mathrm{GF}(2^8)$ 上元素的逆, $B = 01010101$。

2.4 ZUC 的执行过程

在 ZUC 的初始化阶段, 非线性函数 F 的输出是 32 比特长的字 W。我们将 W 最右边的一位移除后, 就得到了 u。u 会在 LFSR 的初始化模式中被用到。ZUC 初始化阶段的运算过程, 如下所示。

过程 5. ZUC 的初始化阶段

a) 比特重组

b) $u = W \gg 1 = \left(F\left(X_0, X_1, X_2\right)\right) \gg 1$

c)线性反馈移位寄存器(初始化模式)

在初始化阶段, 算法会重复执行过程 5, 共执行 32 次。当初始化阶段完成后, 算法进入工作阶段。算法首先会执行 1 次过程 6, 并丢弃生成的输出。

过程 6.

a) 比特重组

b) $F\left(X_0, X_1, X_2\right)$

c) 线性反馈移位寄存器(工作模式)

然后, 算法会重复执行过程 7, 不断生成输出。

过程 7. 工作阶段

a) 比特重组

b) $F\left(X_0, X_1, X_2\right) \oplus X_3$

c) 线性反馈移位寄存器(工作模式)

若要获悉 ZUC 算法更多的细节, 请参考文献[18-19]。

3 ZUC 算法的量子电路

正如第 2 节所介绍的, ZUC 算法主要包含线性反馈移位寄存器、比特重组及非线性函数 F 三个部分, 我们需要将每个部分都实现为量子电路。从量子电路模型来看, 其中的比特重组部分只涉及变量下标的变化, 这可以通过量子电路的换线操作来完成, 一般来说, 该操作不消耗量子门。因此, 我们的电路设计主要针对算法的线性反馈移位寄存器以及非线性函数 F 两部分。如引言中所述, 本文给出的量子电路构造将主要基于 Clifford + Toffoli 模式, 具体来说, 本文涉及的通用量子门包括 CNOT 门, Pauli X 门以及 Toffoli 门三类。这三类门的具体效果以及在电路图中的表示如下。

如图 1 所示, CNOT 门以 $|a\rangle$ 所在的量子位为控制位, 以 $|b\rangle$ 所在的量子位为受控位, 将计算基态 $|a\rangle|b\rangle$ 转化为 $|a\rangle|a\oplus b\rangle$。

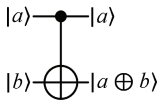


图 1 CNOT 门

Figure 1 The CNOT gate

如图 2 所示, Pauli X 门将计算基态 $|a\rangle$ 转化为 $|a\oplus 1\rangle$。

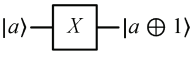


图 2 Pauli X 门

Figure 2 The Pauli X gate

如图 3 所示, Toffoli 门以 $|a\rangle$ 和 $|b\rangle$ 所在的量子位作为控制位, 以 $|c\rangle$ 所在的量子位作为受控位, 将计算基态 $|a\rangle|b\rangle|c\rangle$ 转化为 $|a\rangle|b\rangle|c\oplus ab\rangle$。

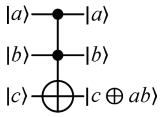


图 3 Toffoli 门

Figure 3 The Toffoli gate

在给出整体的量子电路设计之前, 我们先给出各个部件的量子电路设计。

3.1 量子模加器

首先我们给出量子模加器的设计。在现有的文献中, 已有一些关于量子模加器的研究成果, 其中主要分为两类构造方案, 一类主要使用了 controlled-R_n 门来构造量子模加器[20], 另一类主要使用 Toffoli 和 CNOT 门来构造量子模加器[21], 本文将借鉴第二种设计思路。ZUC 算法的运算过程中, 涉及两种模加操作, 分别是在有限域 $\mathrm{GF}\left(2^{31}-1\right)$ 上的模加和在有限域 $\mathrm{GF}\left(2^{32}\right)$ 上的模加。我们将分别介绍两种模加运算对应的量子电路。

3.1.1 有限域 $\mathrm{GF}\left(2^{31}-1\right)$ 上的量子模加器

我们首先来介绍在有限域 $\mathrm{GF}\left(2^{31}-1\right)$ 上的量子模加器的设计。如之前所述, ZUC 算法中的 $\mathrm{mod}(2^{31}-1)$ 运算与传统的模运算不同。因此, 针对这种特殊的模加运算, 我们在量子比特数优先的设计原则下, 给出了一种高效的量子电路设计。

该量子模加器的总体结构如图 4 所示。其由两个 31 量子比特的量子寄存器和一个辅助量子位组成。图 4 左边的电路展示了该量子模加器的主要三个组成部分, 接下来我们会对各部分进行详细说明。图 4 右边的电路是该量子模加器的简图, 主要用于 ZUC 算法整体电路的示意图中。我们需要注意, 此时简图的黑色条框在右侧, 当黑色条框在左侧时, 其表示原电路的酉逆。简图上的“I”、“O”、“A”分别表示 input、output(保存输出结果)与 ancilla(辅助量子位)。

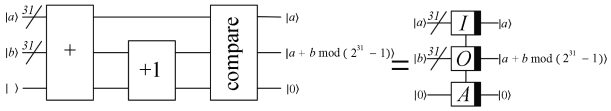


图 4 有限域 $\mathrm{GF}\left(2^{31}-1\right)$ 上的量子模加器

Figure 4 The quantum modulo adder over $\mathrm{GF}\left(2^{31}-1\right)$

如 ZUC 算法的设计文档所述[19], 对于寄存器单元上的值 $0<a\le 2^{31}-1$ 与 $0<b\le 2^{31}-1$, 若 $a+b=p+c2^{31}$, 其中 $0\le p\le 2^{31}-1$, $c\in\{0,1\}$, 则有 $a+b \bmod\left(2^{31}-1\right)=p+c$。从上述关系, 我们可以知道, 通过传统计算模式计算 $a+b \bmod\left(2^{31}-1\right)$, 可以分为两步。

1) 将 a,b 看作两个 31 位的二进制数并计算它们

的和 $a+b$，记录下 $a+b$ 最高位的进位 c，以及后 31 位组成的二进制数 p；

2) 计算 $p+c$。显然，由于 $0<a\le 2^{31}-1$ 与 $0<b\le 2^{31}-1$，我们有 $p+c\le 2^{31}-1$，因此 $p+c$ 仍为一个 31 位的二进制数。

图 4 左侧的前两个部分是对应了上述两个步骤的量子电路，而第三个部分则是为了将辅助量子比特清零。下面，我们对该量子模加器的具体运算步骤进行详细介绍。

量子模加器的输入是状态 $|a\rangle$ 和 $|b\rangle$，首先我们将 $|a\rangle$ 和 $|b\rangle$ 相加。在图 4 中，我们用标有“+”的方块表示实现相加过程的电路，即量子加法器。其具体实现可以参考文献[21]中的图 2。我们将 $|a\rangle$ 和 $|b\rangle$ 相加的结果保存在第二个寄存器上，加法产生的进位保存在辅助量子位上，进位用 c 表示。此时，我们可以将第二个寄存器上的状态表示为 $\left|a+b-c \bmod\left(2^{31}-1\right)\right\rangle$，辅助量子位的状态表示为 $|c\rangle$。

之后，我们需要把辅助量子位上的值加到第二个量子寄存器上。也即将第二个量子寄存器的状态变为 $\left|a+b \bmod\left(2^{31}-1\right)\right\rangle$。对于这个步骤，最直接的方法是将 $\left|a+b-c \bmod\left(2^{31}-1\right)\right\rangle$ 与 $|c\rangle$ 看成两个 31 位的输入再运行一次量子加法器，但是这个过程需要 30 个辅助比特来将 $|c\rangle$ 扩充为 31 位。显然，该方案与量子比特数优先的设计原则相违背。在本文中，我们利用了文献[22]中的量子增量器电路(即实现一个整数加 1 的操作)，只需要利用 ZUC 算法量子电路其余部分的一个此时空闲的量子比特，即可完成上述运算。该运算过程在图 4 中被表示为标有“+1”的方块，其具体实现如图 5 所示。

图 5 的运算过程大致如下，我们将 1 位的 $|c\rangle$ 和 31 位的 $\left|a+b-c \bmod\left(2^{31}-1\right)\right\rangle$ 看作一个整体 $|x\rangle=|c,x_1,\cdots,x_{31}\rangle$，其中 $|c\rangle$ 看作最低有效位。从整个 ZUC 算法量子电路的其余部分，选取一个此时空闲的量子比特，记其为 $|g_0\rangle$。需要注意，此时我们不要求 $|g_0\rangle=|0\rangle$，即使其为混合态仍然不影响我们的运算。这时的 $|a\rangle$ 与 $|g_0\rangle$ 被称为 borrowed dirty qubits，它们的初值不影响电路的输出，并且在该过程结束后，它们将恢复为初值。我们将 $|a\rangle$ 与 $|g_0\rangle$ 整体看作状态 $|g\rangle=|g_0,\cdots,g_{31}\rangle$，并将 $|x\rangle$ 与 $|g\rangle$ 作为图 5 电路的输入。从图 5 下方的具体电路图中，我们可以看出该电路实现了 $|x\rangle|g\rangle\to|x+1\rangle|g\rangle$。其中，图中的 g' 表示 g 的二进制补码。经过图 5 所示的运算后，图 4 的第二个寄存器上的值，就是我们想要的结果 $\left|a+b \bmod\left(2^{31}-1\right)\right\rangle$。

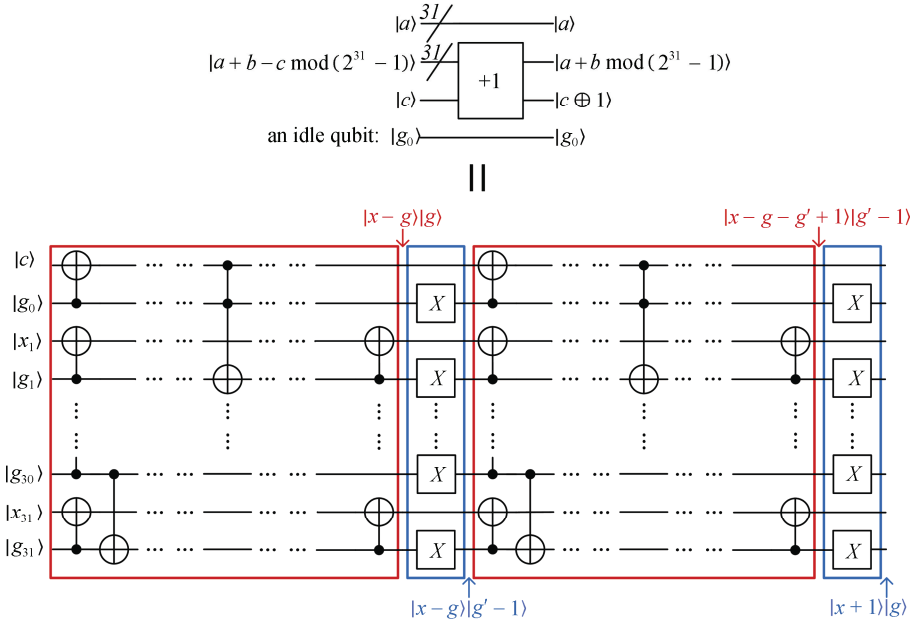


图 5 增量器

Figure 5 The incrementer

最后，我们需要利用图 4 中的 compare 部件将辅助量子位的状态 $|c\oplus 1\rangle$ 清零。此部件接收图 4 中前两

个部件的输出$|a\rangle\left|a+b \bmod (2^{31}-1)\right\rangle|c\oplus 1\rangle$作为输入。电路前半部分的作用是从第一个寄存器中减去第二个寄存器的值, 减完后第一个寄存器的状态变为$\left|a-\left(a+b \mod\left(2^{31}-1\right)\right)\right\rangle$。我们用$c_i\,(i=1,\cdots,31)$表示减法产生的第$i$位的借位, 其中$c_{31}$表示最高位的借位。我们考虑两种情况:

1) $a+b\le 2^{31}-1$。则 compare 部件的输入状态为$|a\rangle|a+b\rangle|1\rangle$;

2) $a+b>2^{31}-1$。则 compare 部件的输入状态为$|a\rangle\left|a+b-(2^{31}-1)\right\rangle|0\rangle$。

由于$0<b\le 2^{31}-1$, 因此, 对于情况 1), 我们有$a<a+b$, 从而$c_{31}=1$; 对于情况 2), 我们有$a\ge a+b-(2^{31}-1)$, 此时$c_{31}=0$。综上所述, 我们得知$c_{31}=c\oplus 1$。因此, 我们可以通过 CNOT 门将c_{31}异或到辅助量子位上, 从而清零辅助量子位。

compare 部件的详细实现如图 6 所示。在图 6 中, 我们将$\left|a+b \bmod\left(2^{31}-1\right)\right\rangle$记作$|\mathrm{sum}_0,\cdots,\mathrm{sum}_{30}\rangle$。此时, 我们有

$$c_i=\begin{cases}a_0\mathrm{sum}_0\oplus\mathrm{sum}_0 & i=1\\ (a_{i-1}\oplus c_{i-1})(\mathrm{sum}_{i-1}\oplus c_{i-1})\oplus\mathrm{sum}_{i-1} & 1<i\le 31\end{cases}$$

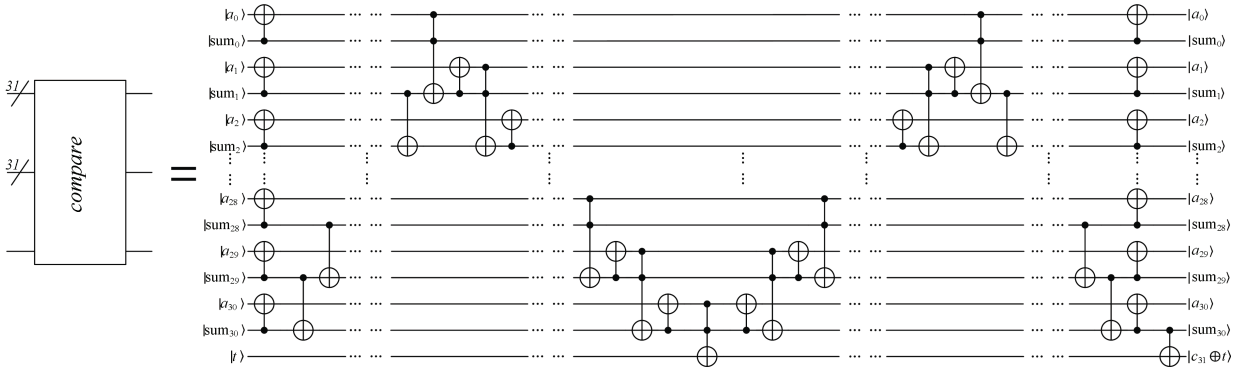


图 6 compare 电路

Figure 6 The "compare" circuit

若我们将$|a_i\rangle$所在的量子位记为$A[i]$, 将$|\mathrm{sum}_i\rangle$所在的量子位记为$S[i]$, 则图 6 所示的量子电路的运算可以用如下步骤表示:

(1) 对于$i=0,\cdots,30$, 以$S[i]$为控制位, $A[i]$为受控位, 应用 CNOT 门。经过本步后, 电路状态变为

$$|a_0\oplus\mathrm{sum}_0\rangle|\mathrm{sum}_0\rangle|a_1\oplus\mathrm{sum}_1\rangle|\mathrm{sum}_1\rangle\left(\bigotimes_{i=2}^{30}|a_i\oplus\mathrm{sum}_i\rangle|\mathrm{sum}_{i-1}\oplus\mathrm{sum}_i\rangle\right)|t\rangle$$

(3) 对于$i=0,\cdots,29$, 以$A[i]$和$S[i]$为控制位, 以$S[i+1]$为受控位, 应用 Toffoli 门; 并且以$S[i+1]$为控制位, 以$A[i+1]$为受控位, 应用 CNOT 门。经过本步后, 电路状态变为

$$|a_0\oplus\mathrm{sum}_0\rangle|\mathrm{sum}_0\rangle\left(\bigotimes_{i=1}^{30}|a_i\oplus c_i\rangle|\mathrm{sum}_i\oplus c_i\rangle\right)|t\rangle$$

(4) 以$A[30]$和$S[30]$为控制位, 以辅助量子位为受控位, 应用 Toffoli 门。经过本步后, 电路状态变为

$$|a_0\oplus\mathrm{sum}_0\rangle|\mathrm{sum}_0\rangle\left(\bigotimes_{i=1}^{30}|a_i\oplus c_i\rangle|\mathrm{sum}_i\oplus c_i\rangle\right)|c_{31}\oplus\mathrm{sum}_{30}\oplus t\rangle$$

(5) 分别执行步骤 3、步骤 2 及步骤 1 的酉逆操作, 将

$$\left(\bigotimes_{i=0}^{30}|a_i\oplus\mathrm{sum}_i\rangle|\mathrm{sum}_i\rangle\right)|t\rangle$$

(2) 对于$i=30,\cdots,2$, 以$S[i-1]$为控制位, $S[i]$为受控位, 应用 CNOT 门。经过本步后, 电路状态变为

$$\left(\bigotimes_{i=0}^{30}|a_i\oplus\mathrm{sum}_i\rangle|\mathrm{sum}_i\rangle\right)|t\rangle$$

除辅助量子位外的其余量子比特, 恢复为初始状态。经过本步后, 电路状态变为

$$\left(\bigotimes_{i=0}^{30}|a_i\rangle|\mathrm{sum}_i\rangle\right)|c_{31}\oplus\mathrm{sum}_{30}\oplus t\rangle$$

(6) 最后, 以$S[30]$为控制位, 以辅助量子位为受控位, 应用 CNOT 门。经过本步后, 电路状态变为

$$\left(\bigotimes_{i=0}^{30}|a_i\rangle|\mathrm{sum}_i\rangle\right)|c_{31}\oplus t\rangle$$

因为$c_{31}=c\oplus 1$, 所以我们可以通过上述电路来清零图 4 所示的量子模加器的辅助量子位。

有限域$\mathrm{GF}\left(2^{31}-1\right)$上的量子模加器共消耗 63 个

量子比特, 246 个 Toffoli、639 个 CNOT 及 64 个 Pauli X 门。

3.1.2 有限域 $\mathrm{GF}\left(2^{32}\right)$ 上的量子模加器

接下来, 我们介绍有限域 $\mathrm{GF}\left(2^{32}\right)$ 上的量子模加器的设计。参与模加运算的两个加数都是 32 位的数。因此, 我们直接丢弃最高有效位产生的进位, 就可以保证运算满足需求。我们参考了文献[21]中图 2 的设计, 将两个 32 量子比特大小的加数相加, 并且不保留最高有效位产生的进位。这样, 我们就构造出了基于有限域 $\mathrm{GF}\left(2^{32}\right)$ 的量子模加器。其具体构造如图 7 所示, 图 7 右边是该模加器的简图。图 7 所示模加器共消耗 64 量子比特, 62 个 Toffoli、154 个 CNOT 门。

3.2 LFSR 的量子电路

ZUC 算法中 LFSR 的主要工作是通过模加等运算产生新的状态 $|s_i\rangle$, 并更新寄存器单元。由于在初始化模式和工作模式下, LFSR 的运算流程非常相似, 因此我们以初始化模式下的运算为例, 展示如何构造 LFSR 的量子电路。

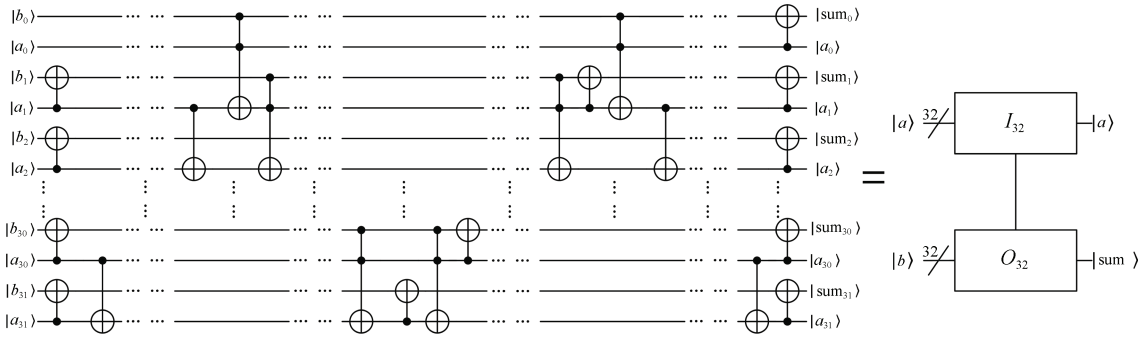


图 7 有限域 $\mathrm{GF}\left(2^{32}\right)$ 上的量子模加器

Figure 7 The quantum modulo adder over $\mathrm{GF}\left(2^{32}\right)$

初始化模式下 LFSR 的量子电路如图 8 所示。此量子电路共包括 17 个大小为 31 量子比特的寄存器及一个辅助量子位, 前 16 个寄存器存储量子态 $|s_0\rangle,\ldots,|s_{15}\rangle$, 第 17 个寄存器是初态为零的辅助寄存器, 而辅助量子位会在量子模加器(或其逆电路)的运算过程中被用到。图 8 中的单个 CNOT 门代表一组(31 个)并行工作的 CNOT 门。符号“$\lll_i$”代表量子寄存器中的量子比特循环左移 i 位, 符号“$\ggg_j$”代表量子寄存器中的量子比特循环右移 j 位。显然, 循环移位可以由换线操作来完成, 因此在本文的模型中, 我们忽略循环移位的资源消耗。

为了方便说明, 我们在图 8 的不同位置, 用数字进行了标注。

首先用一组 CNOT 门将状态 $|s_0\rangle$ 拷贝至辅助寄存器, 即图 8 位置 1 处的状态为 $|s_0\rangle$。接着将第 1 个寄存器的量子位循环左移 8 位并模加到辅助寄存器, 即位置 2 处的状态为: $\left|\left(1+2^8\right)s_0 \bmod\left(2^{31}-1\right)\right\rangle$。我们令 $y\equiv\left(1+2^8\right)s_0 \bmod\left(2^{31}-1\right)$, 也就是, $x\equiv\left(2^{24}-2^{16}+2^8-1\right)y \bmod\left(2^{31}-1\right)$。位置 2 处的状态也可以表示为: $\left|y \bmod\left(2^{31}-1\right)\right\rangle$。同理, 位置 3 状态为: $\left|\left(2^{24}-2^{16}+2^8\right)y \bmod\left(2^{31}-1\right)\right\rangle$。位置 4 处应用的是 $\mathrm{GF}\left(2^{31}-1\right)$ 上量子模加器的逆电路, 实现了模减的效果。位置 4 处的状态为 $\left|\left(2^{24}-2^{16}\right)y \bmod\left(2^{31}-1\right)\right\rangle$。位置 5 处的状态为 $\left|2^{24}\bullet y \bmod\left(2^{31}-1\right)\right\rangle$。对于辅助寄存器来说, 其经过三次向左循环移位后, 在位置 6 处的值为: $\left|2^{24}\bullet y \bmod\left(2^{31}-1\right)\right\rangle$。于是, 我们可以用一组 CNOT 门将第一个寄存器的状态清零。接着, 我们将辅助寄存器的量子位向右循环移 24 位, 即在位置 7 处的值为: $\left|\left(1+2^8\right)s_0 \bmod\left(2^{31}-1\right)\right\rangle$。接着根据公式(1), 我们通过循环移位得到其余加数, 并分别模加到辅助寄存器上, 得到相应结果。工作模式下 LFSR 的单轮运算到此完成。但在初始化模式下, 新状态 $|s_{16}\rangle$ 是由模加的结果 $|v\rangle$ 和非线性函数 F 生成的输出 $|u\rangle$ 再次模加得到的。因此, 在图 8 中的位置 8, 我们用带有虚线的模加器来表示模加 $|u\rangle$ 的过程。虚线表示图 8 所示量子电路与 ZUC 算法整体电路的其余部分(即 F 的量子电路)有交互。模加 $|u\rangle$ 所消耗的量子资源, 被统计在 3.4 节中。

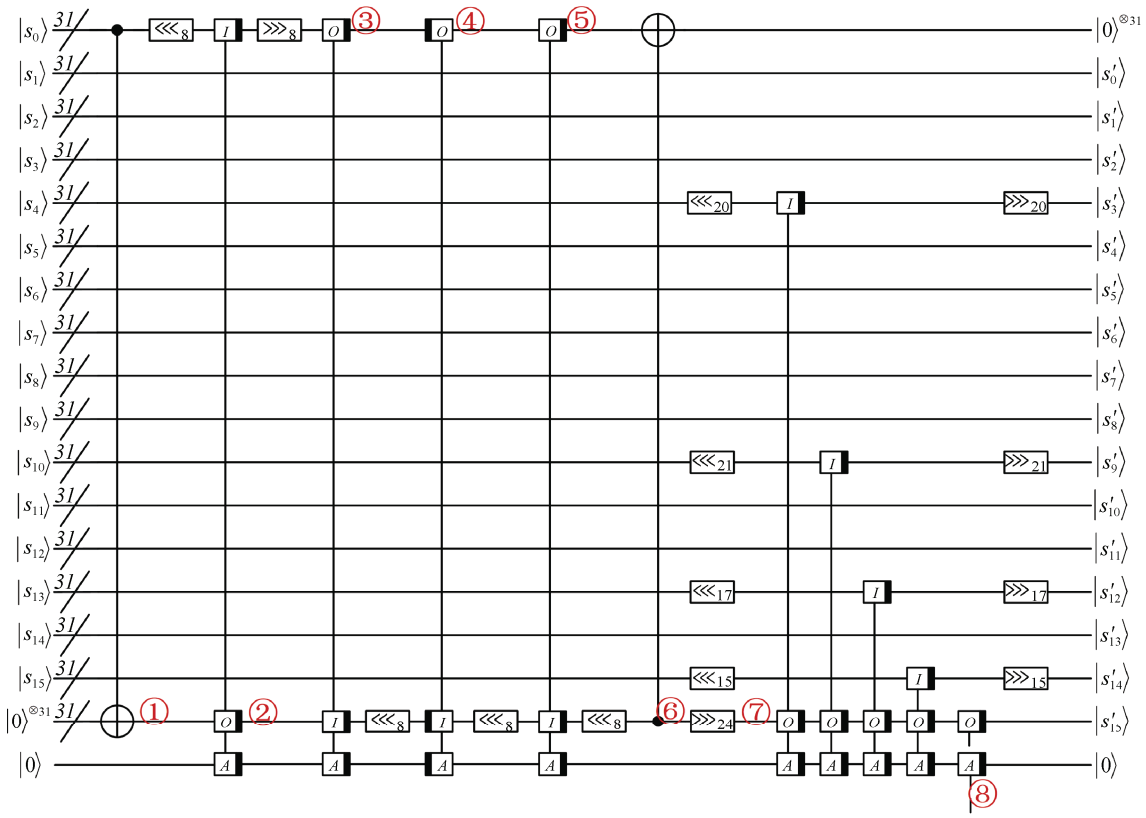


图 8 LFSR 的初始化模式量子电路

Figure 8 The quantum circuit of LFSR in initialization mode

综上所述, 利用图 8 所示的各个量子门, 我们可以完成 LFSR 单轮的操作, 由原状态$|s_0\rangle,\cdots,|s_{15}\rangle$, 变为新状态$|s_0'\rangle,\cdots,|s_{15}'\rangle$。图 8 所示的 32 个辅助量子比特可以重复使用, 在之后的轮运算中, 无须额外消耗量子比特。并且 LFSR 与非线性函数 F 的运算是交替进行的, 因此只要进行及时清零, 非线性函数 F 也可以使用图 8 所示的 32 个辅助量子比特。图 8 所示的量子电路共消耗了 528 个量子比特, 1968 个 Toffoli、5174 个 CNOT 以及 512 个 Pauli X 门。

3.3 比特重组的量子电路

比特重组阶段, 算法从LFSR的寄存器单元中选取 128 个量子比特, 组成四个大小为 32 量子比特的字 X_0, X_1, X_2, X_3。比特重组的具体过程展示在 2.2 节中。X_0, X_1, X_2, X_3将会在接下来的流程中进行异或、模加等运算。我们并不显示地组成 4 个 32 量子比特的字, 而是以寄存器的相应量子位作为量子门的控制位, 以此来让 X_0, X_1, X_2, X_3参与到相应运算中。以 X_0 为例, 我们以$|s_{15}\rangle$的高 16 位与$|s_{14}\rangle$的低 16 位作为一组 CNOT 门的控制位, 并选取适当的受控位, 使 X_0 与相应的值进行异或运算。X_2 与 X_3 同理。对于 X_1, 我们以$|s_{11}\rangle$的低 16 位与$|s_9\rangle$的高 16 位作为 $\mathrm{GF}(2^{32})$ 上量子模加器的输入, 并选取适当的输出位, 使 X_1 模加上 R_1。比特重组阶段消耗的资源将会在 3.4 节中被统计。

3.4 非线性函数 F 的量子电路

此节将分为三个部分来介绍非线性函数 F 的量子电路。首先给出线性变换 L_1 与 L_2 的量子电路的设计思路, 然后给出非线性部分 S_0 与 S_1 的量子电路, 最后给出 F 整体的量子电路构造及相应的资源估计。如 3.2 节指出的那样, F 与 LFSR 共用图 8 所示的辅助量子比特, 所以我们在图 10 和图 11 中没有显示地表示出 F 所用的辅助量子比特。

3.4.1 L_1 与 L_2 的量子电路

L_1 和 L_2 是 F 中的线性变换, 其运算如公式(2)和公式(3)所示。由于 L_1 和 L_2 均为 $\mathrm{GF}(2^{32})$ 上的线性变换, 因此变换矩阵可以转化为基本行变换矩阵的复合。从量子电路角度来看, 该变换的量子电路可以 in place(即不需要辅助量子比特)的写成 CNOT 门与换线操作的复合, 其中 CNOT 门对应了将矩阵的一行加到另一行, 换线操作则对应了将矩阵的两行进行交换。对于上述分解问题, 最直接的方法是通过高斯消去法或者 LU 分解来解决, 在文献[23]中, 作者给

出了一种基于分块的 LU 分解方法, 能够减少电路中出现的 CNOT 门的数量。在文献[14]中, 作者给出了一种在经典计算模型下启发式的计算矩阵的最佳 s-Xor 数的算法。我们发现寻找矩阵的最佳 s-Xor 计数等价于量子计算模型下寻找该线性变换的最少 CNOT 门实现, 并且此算法得到的分解中 CNOT 门的数量远远少于高斯消去法以及 LU 分解得到的结果。因此, 我们采用了文献[14]中的算法来构造 L_1 和 L_2 的量子电路, 具体实现请看附录 A。该量子电路总共消耗 173 个 CNOT 门。

3.4.2 S_0 与 S_1 的量子电路

非线性函数 F 中的 S 盒是由 S_0 与 S_1 并行组成的, 即 $S=(S_0,S_1,S_0,S_1)$。我们将分别构造 S_0 与 S_1 的量子电路。

S_0 是基于 Feistel 结构设计的, 其包含三个置换 P_1、P_2 和 P_3。每个置换都接收 4 比特长的字串作为输入, 并产生 4 比特的输出。S_0 的具体结构请参考文献[19]。我们将三个置换转化为布尔表达式的形式, 如下所示(I_i 表示输入的第 i 个比特, O_j 表示输出的第 j 个比特)。

$$P_1\begin{cases}O_0=I_0I_2+I_2I_3+I_0+I_2+1\\O_1=I_1I_2+I_1I_3+I_1+I_3\\O_2=I_0I_3+I_1I_3+I_1+I_3\\O_3=I_0I_1+I_0I_2+I_0+I_2+1\end{cases}$$

$$P_2\begin{cases}O_0=I_0I_2I_3+I_1I_2I_3+I_0I_1+I_0I_2+I_0I_3+I_1+I_2+1\\O_1=I_0I_1I_2+I_0I_1I_3+I_0I_3+I_1I_2+I_2I_3+I_1+I_2+I_3\\O_2=I_0I_1I_2+I_0I_1+I_0I_2+I_0I_3+I_1I_3+I_2I_3+I_0+I_1+I_2\\O_3=I_1I_2I_3+I_0I_1+I_0I_2+I_0I_3+I_0+I_1+I_3+I_1I_2\end{cases}$$

$$P_3\begin{cases}O_0=I_1I_3+I_2I_3+I_2\\O_1=I_0I_2+I_0I_3+I_3\\O_2=I_0I_2+I_1I_2+I_1+1\\O_3=I_0I_1+I_1I_3+I_0\end{cases}$$

我们根据以上的布尔表达式来设计 S_0 的量子电路, 详细设计请看附录 C。此电路共消耗 9 个量子比特, 33 个 Toffoli、31 个 CNOT 及 4 个 Pauli X 门。

S_1 类似于 AES 中 S 盒的结构, 先对域 GF(256) 上的元素 x 进行求逆, 再进行仿射变换, 得到最终结果, 即 $S_1=Mx^{-1}+B$。此前已经有许多针对 AES 算法 S 盒的成熟的优化设计[11,24-26], 而文献[11]中设计的电路优化效果较好, 使用了较少的量子比特, 所以我们将利用该文实现的 AES 算法的 S 盒电路, 通过添加少许 CNOT 门来构造 S_1 的量子电路。可以看出在我们的电路设计准则下, 添加少量的 CNOT 门对整体量子计算资源消耗的影响很小。

S_1 的整体设计如图 9 所示。该电路接收 8 量子比特的输入, 产生 8 量子比特的输出, 运算过程中共使用了 15 个辅助量子比特。因为 AES 算法的 S 盒与 S_1 中, 构造有限域 GF(256) 所用的不可约多项式不同, 前者的不可约多项式是 $x^8+x^4+x^3+x+1$, 后者的不可约多项式是 $x^8+x^7+x^3+x+1$。所以我们首先对 8 量子比特输入进行线性转换, 将其变为 AES 算法 S 盒的域元素的表示方式, 这一步对应图 9 中位置 1 处的电路。接着利用文献[11]中的电路, 完成 AES 算法 S 盒的运算以及清零辅助量子比特的逆运算, 这一步对应图 9 位置 2 处的电路。经过前两处电路的作用后, 我们得到了原输入值经 AES 算法 S 盒运算后的结果值。然后我们进行 AES 算法 S 盒的逆仿射变换, 此步对应图 9 中位置 3 处的电路。此时我们得到原输入值在有限域 GF(256) 上求逆后的值。之后我们将元素表示转换回 S_1 的域元素表示方式, 此步对应图 9 位置 4 处的电路。最后进行 S_1 的仿射变换(对应图 9 位置 5 处的电路), 即可得到结果值。图 9 位置 1、4 处电路对应的线性矩阵及推导过程, 请看附录 B。S_1 的量子电路共消耗 23 个量子比特, 117 个 Toffoli、689 个 CNOT 及 36 个 Pauli X 门。

3.4.3 F 的量子电路

在完成非线性函数 F 各个部件的量子电路设计后, 我们现给出 F 的量子电路。有关 F 运算过程的具体描述, 请看 2.3 节。图 10 和图 11 分别展示了在初始化阶段和工作阶段的 F 的量子电路, 两者仅有细微差别。为了方便说明, 我们在图 10 中的某些位置标记了数字序号。

下面, 我们以图 10 为例, 来解释 F 的量子电路。此量子电路由三个 32 量子比特大小的寄存器组成。分别保存量子态 $|R_1\rangle$ 和 $|R_2\rangle$, 以及用作辅助量子位。图中量子电路上的虚线, 代表其与 ZUC 算法量子电路的其他部分有交互。例如图 10 的位置 1 处, 代表以电路其余部分的量子位为一组 CNOT 门的控制位, 以图 10 中所示的 32 个辅助量子比特为这组 CNOT 门的受控位。在图 10 中, 我们首先用一组 CNOT 门, 将状态 $|R_1\rangle$ 异或到辅助寄存器上, 接着分别以 LFSR 中 s_{15} 的高 16 位与 s_{14} 的低 16 位, 作为 32 个 CNOT

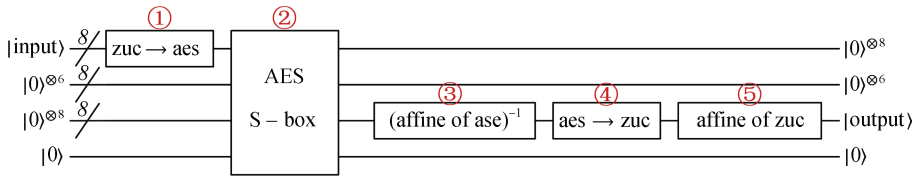


图 9　S_1 的量子电路

Figure 9　The quantum circuit of S_1

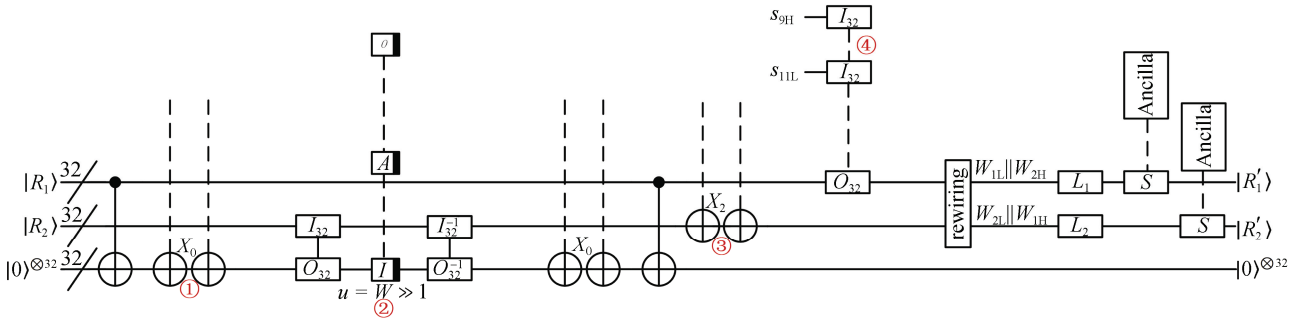


图 10　初始化阶段下 F 的量子电路

Figure 10　The quantum circuit of F (initialization stage)

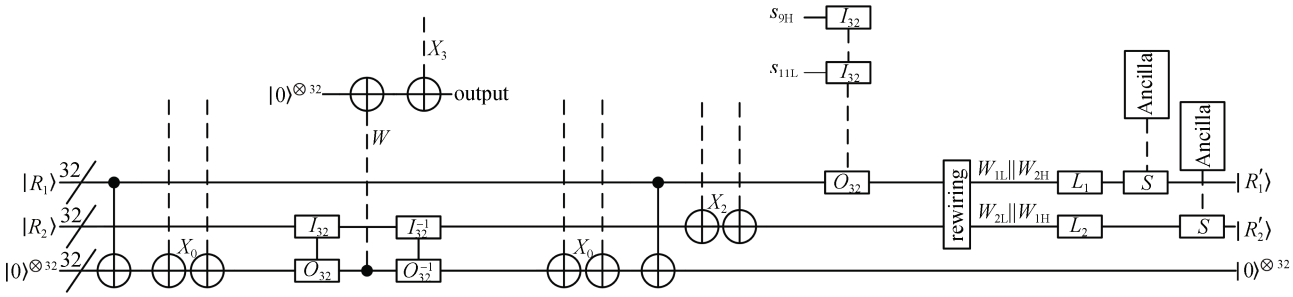


图 11　工作阶段下 F 的量子电路

Figure 11　The quantum circuit of F (working stage)

门的控制位，以辅助寄存器为这 32 个 CNOT 门的受控位。经过这些 CNOT 门的作用后，图 10 中辅助寄存器的状态变为 $|X_0 \oplus R_1\rangle$。接着，我们以 $|R_2\rangle$ 作为 GF(2^{32}) 上量子模加器的输入，将其模加到辅助寄存器上，就得到了非线性函数 F 的输出 W。

在初始化模式下，W 在右移舍弃一位后，要模加到 LFSR。如图 10 的位置 2 处所示，将 $u = W \gg 1$ 作为 GF($2^{31}-1$) 上的量子模加器的输入，将其模加到 LFSR 的特定位置(图 8 中的位置 8)。而在工作模式下，W 将与 X_3 异或产生最终的输出。因此在图 11 中，我们用 CNOT 门将 $|W\rangle$ 拷贝到状态为 $|0\rangle^{\otimes 32}$ 的辅助量子比特上。再令 $|X_3\rangle$ 与其进行异或，产生最终输出。

在图 10 所示的初始化阶段下，当将 u 模加到 LFSR 后，我们用 GF(2^{32}) 上量子模加器的逆电路以及 CNOT 门来清零辅助量子比特。接着我们来更新 $|R_1\rangle$ 和 $|R_2\rangle$ 的状态。图 10 位置 3 处的操作类似位置 1 处，其将量子态 $|R_2\rangle$ 更新为 $|X_2 \oplus R_2\rangle$。位置 4 处表示，我们以 LFSR 中 s_9 的高 16 位与 s_{11} 的低 16 位，作为 GF(2^{32}) 上的量子模加器的输入，将其模加到图 10 的第一个寄存器上，把状态 $|R_1\rangle$ 更新为 $|X_1 \boxplus R_1\rangle$。再经过换线操作、线性变换 L_1 和 L_2 以及 S 盒的作用，得到更新后的状态。需要注意的是，由于 $S=(S_0,S_1,S_0,S_1)$，而 S_0 和 S_1 运算时分别需要 1 个、15 个辅助量子比特。即单个 S 盒运行需要 32 个辅助量子比特，正好使用了图 8 中全部的辅助量子比特。因此，为了最大限度地减少量子比特的消耗，图 10 所示的两个 S 盒是串行运算的。图 10 共消耗 96 个量子比特，1032 个 Toffoli、4314 个 CNOT 及 224 个 Pauli X 门。图 11 共消耗 128 个量子比特，786 个

Toffoli、3739 个 CNOT 及 160 个 Pauli X 门。

4 ZUC 算法量子电路的资源估算

在初始化阶段, ZUC 算法的单轮量子电路如图 12 所示。工作阶段下的 ZUC 算法的单轮量子电路与图 12 相似。

可以注意到, 在我们单轮实现的输出结果中仍然有 64 量子比特为$|0\rangle$状态, 因此, 可以在下一轮操作中继续使用。在初始化阶段, ZUC 算法会重复执行 32 次过程 5。接着进入工作阶段, 首先执行 1 次过程 6, 丢弃产生的输出, 然后重复执行过程 7, 不断生成输出。每执行 1 次过程 7, 产生 32 比特的输出。各个过程单轮的资源消耗如表 1 所示(因为过程 6 不保存产生的输出, 因此其相较于过程 7, 少了用于生成和存储输出的量子比特与 CNOT 门的消耗)。

我们可以将本文设计的 ZUC 算法的量子电路作为 Grover 算法的黑盒, 来遍历搜索 ZUC 算法的密钥。因为 ZUC 算法的密钥长度是 128 比特, 我们考虑在有 128 比特密钥流的情况下, 利用 Grover 算法搜索密钥。在这种情况下, 我们除执行 32 次过程 5 及 1 次过程 6 外, 还需要执行 4 次过程 7, 从而产生 128 比特的输出。这个过程消耗的资源总数为 752 个量子比特, 109770 个 Toffoli、348117 个 CNOT、26912 个 Pauli X 门。

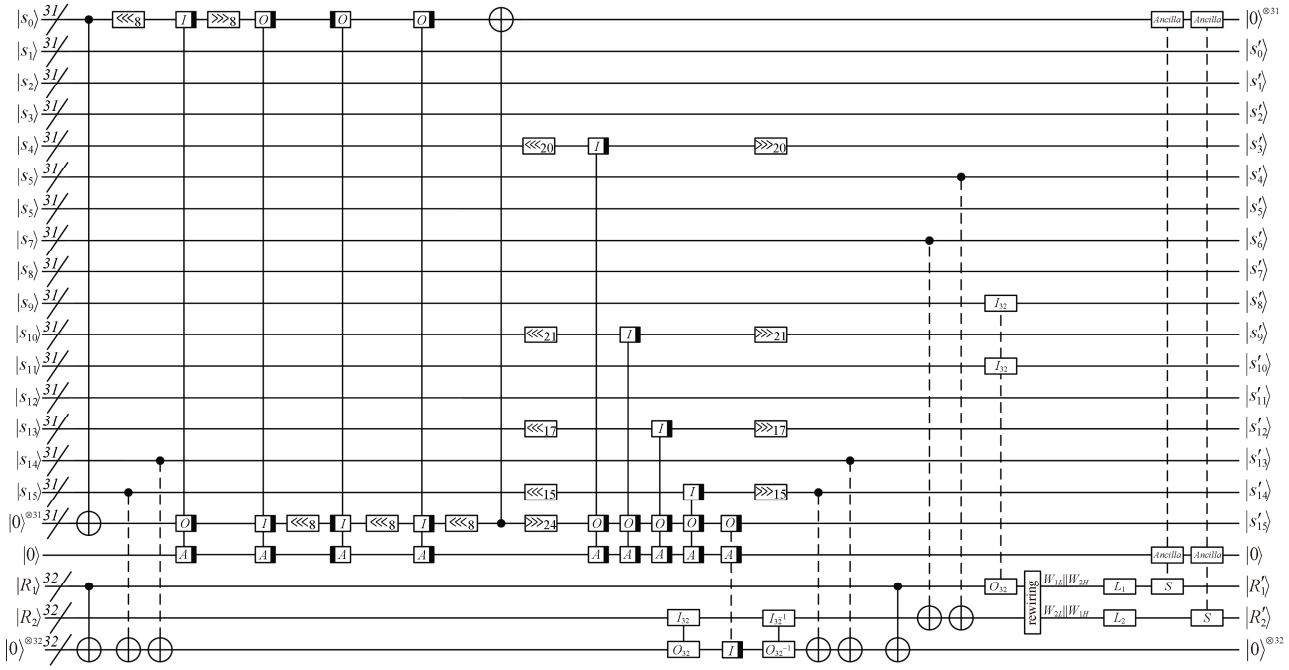


图 12 ZUC 算法单轮量子电路(初始化阶段)

Figure 12 The quantum circuit of ZUC (Initialization stage)

表 1 单轮的资源消耗

Table 1 Single-round resource consumption

	Toffoli	CNOT	Pauli-X	qubits
过程 5	3,000	9,488	736	624
过程 6	2,754	8,849	672	624
过程 7	2,754	8,913	672	656

5 结论

本文基于 Toffoli、CNOT 以及 Pauli X 门, 在优先减少量子比特消耗, 再尽量减少 Toffoli 门数量的设计原则下将 ZUC 算法的密钥流生成过程实现为量子电路。同时本文也首次给出了有限域 $GF(2^{31}-1)$ 上模加器以及线性反馈移位寄存器的量子电路设计, 这些组件的设计未来可以用于其他密码算法的量子电路实现。

参考文献

[1] Shor P W. Algorithms for Quantum Computation: Discrete Logarithms and Factoring[C]. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 2002: 124-134.

[2] Shor P W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer[J]. *SIAM Journal on Computing*, 1997, 26(5): 1484–1509.

[3] Grover L K. A Fast Quantum Mechanical Algorithm for Database Search[C]. *The twenty-eighth annual ACM symposium on Theory of Computing*, 1996: 212–219.

[4] Kaplan M, Leurent G, Leverrier A, et al. Breaking Symmetric Cryptosystems Using Quantum Period Finding[C]. Annual Interna-

tional Cryptology Conference. Berlin, Heidelberg: Springer, 2016: 207-237.

[5] Dong X Y, Dong B Y, Wang X Y. Quantum Attacks on some Feistel Block Ciphers[J]. *Designs*, *Codes and Cryptography*, 2020, 88(6): 1179-1203.

[6] Grassl M, Langenberg B, Roetteler M, et al. Applying Grover's Algorithm to AES: Quantum Resource Estimates[C]. *Post-Quantum Cryptography*. Cham: Springer, 2016: 29-43.

[7] Almazrooie M, Samsudin A, Abdullah R, et al. Quantum Reversible Circuit of AES-128[J]. *Quantum Information Processing*, 2018, 17(5): 112.

[8] Guajardo J, Paar C. Itoh-Tsujii Inversion in Standard Basis and Its Application in Cryptography and Codes[J]. *Designs*, *Codes and Cryptography*, 2002, 25(2): 207-216.

[9] Kim P, Han D, Jeong K C. Time–Space Complexity of Quantum Search Algorithms in Symmetric Cryptanalysis: Applying to AES and SHA-2[J]. *Quantum Information Processing*, 2018, 17(12): 339.

[10] Langenberg B, Pham H, Steinwandt R. Reducing the Cost of Implementing the Advanced Encryption Standard as a Quantum Circuit[J]. *IEEE Transactions on Quantum Engineering*, 2020, 1: 1-12.

[11] Zou J, Wei Z H, Sun S W, et al. Quantum Circuit Implementations of AES with Fewer Qubits[C]. *International Conference on the Theory and Application of Cryptology and Information Security*. Cham: Springer, 2020: 697-726.

[12] Jaques S, Naehrig M, Roetteler M, et al. Implementing Grover Oracles for Quantum Key Search on AES and LowMC[C]. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Cham: Springer, 2020: 280-310.

[13] Fowler A G, Mariantoni M, Martinis J M, et al. Surface Codes: Towards Practical Large-Scale Quantum Computation[J]. *Physical Review A*, 2012, 86(3): 032324.

[14] Xiang Z J, Zeng X, Lin D, et al. Optimizing Implementations of Linear Layers[J]. *IACR Transactions on Symmetric Cryptology*, 2020: 120-145.

[15] Jones C. Low-Overhead Constructions for the Fault-Tolerant Toffoli Gate[J]. *Physical Review A*, 2013, 87(2): 022328.

[16] Amy M, Maslov D, Mosca M, et al. A Meet-in-the-Middle Algorithm for Fast Synthesis of Depth-Optimal Quantum Circuits[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2013, 32(6): 818-830.

[17] Nielsen M A, Chuang I L. Quantum Computation and Quantum Information[M]. Cambridge: : Cambridge University Press, 2012.

[18] Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3[R]. Document 2: ZUC Specification. ETSI / SAGE Specification, Version: 1.5, 2011.

[19] Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3[R]. Document 4: Design and Evaluation Report. ETSI / SAGE Specification, Version: 1.3, 2011.

[20] Beauregard S. Circuit for Shor's Algorithm Using 2n+3 Qubits[J]. *Quantum Information & Computation*, 2003, 3(2): 175–185.

[21] Takahashi Y, Tani S, Kunihiro N. Quantum Addition Circuits and Unbounded Fan-out[J]. *Quantum Information & Computation*, 2010, 10(9): 872–890.

[22] Häner T, Roetteler M, Svore K M. Factoring Using 2n + 2 Qubits with Toffoli Based Modular Multiplication[J]. *Quantum Information & Computation*, 2017, 17(7/8): 673–684.

[23] Patel K N, Markov I L, Hayes J P. Optimal Synthesis of Linear Reversible Circuits[J]. *Quantum Information & Computation*, 2008, 8(3): 282–294.

[24] Boyar J, Peralta R. A New Combinational Logic Minimization Technique with Applications to Cryptology[C]. *International Symposium on Experimental Algorithms. Berlin*, Heidelberg: Springer, 2010: 178-189.

[25] Boyar J, Peralta R. A Small Depth-16 Circuit for the AES S-Box[C]. *IFIP International Information Security Conference*. Berlin, Heidelberg: Springer, 2012: 287-298.

[26] Canright D. A very Compact S-Box for AES[M]. Cryptographic Hardware and Embedded Systems – CHES 2005. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005: 441-455.

附　　录

附录 A　线性变换 L_1 及 L_2 的量子电路

Appendix A Quantum circuits of L_1 and L_2

$Q[i]=Q[i]\oplus Q[j]$ 意为，以第 j+1 个量子比特为控制位，以第 i+1 个量子比特为受控位，应用 CNOT 门。

L_1 的量子电路:

$Q[27] = Q[27] \oplus Q[19];$
$Q[8] = Q[8] \oplus Q[0];$
$Q[25] = Q[25] \oplus Q[1];$
$Q[26] = Q[26] \oplus Q[10];$
$Q[17] = Q[17] \oplus Q[9];$
$Q[4] = Q[4] \oplus Q[12];$
$Q[30] = Q[30] \oplus Q[22];$
$Q[31] = Q[31] \oplus Q[23];$
$Q[0] = Q[0] \oplus Q[24];$
$Q[10] = Q[10] \oplus Q[2];$
$Q[23] = Q[23] \oplus Q[7];$
$Q[21] = Q[21] \oplus Q[27];$
$Q[2] = Q[2] \oplus Q[8];$
$Q[29] = Q[29] \oplus Q[13];$
$Q[28] = Q[28] \oplus Q[12];$
$Q[12] = Q[12] \oplus Q[20];$
$Q[13] = Q[13] \oplus Q[5];$

$Q[20] = Q[20] \oplus Q[10]$;
$Q[5] = Q[5] \oplus Q[27]$;
$Q[24] = Q[24] \oplus Q[8]$;
$Q[10] = Q[10] \oplus Q[0]$;
$Q[27] = Q[27] \oplus Q[11]$;
$Q[11] = Q[11] \oplus Q[17]$;
$Q[10] = Q[10] \oplus Q[18]$;
$Q[18] = Q[18] \oplus Q[26]$;
$Q[19] = Q[19] \oplus Q[3]$;
$Q[18] = Q[18] \oplus Q[8]$;
$Q[0] = Q[0] \oplus Q[22]$;
$Q[22] = Q[22] \oplus Q[14]$;
$Q[8] = Q[8] \oplus Q[16]$;
$Q[3] = Q[3] \oplus Q[27]$;
$Q[27] = Q[27] \oplus Q[1]$;
$Q[1] = Q[1] \oplus Q[31]$;
$Q[27] = Q[27] \oplus Q[9]$;
$Q[0] = Q[0] \oplus Q[6]$;
$Q[14] = Q[14] \oplus Q[30]$;
$Q[14] = Q[14] \oplus Q[4]$;
$Q[8] = Q[8] \oplus Q[30]$;
$Q[30] = Q[30] \oplus Q[12]$;
$Q[1] = Q[1] \oplus Q[17]$;
$Q[9] = Q[9] \oplus Q[7]$;
$Q[17] = Q[17] \oplus Q[25]$;
$Q[9] = Q[9] \oplus Q[15]$;
$Q[15] = Q[15] \oplus Q[13]$;
$Q[7] = Q[7] \oplus Q[13]$;
$Q[13] = Q[13] \oplus Q[19]$;
$Q[30] = Q[30] \oplus Q[6]$;
$Q[25] = Q[25] \oplus Q[23]$;
$Q[23] = Q[23] \oplus Q[15]$;
$Q[12] = Q[12] \oplus Q[26]$;
$Q[15] = Q[15] \oplus Q[31]$;
$Q[23] = Q[23] \oplus Q[29]$;
$Q[31] = Q[31] \oplus Q[5]$;
$Q[6] = Q[6] \oplus Q[4]$;
$Q[4] = Q[4] \oplus Q[26]$;
$Q[26] = Q[26] \oplus Q[16]$;
$Q[4] = Q[4] \oplus Q[20]$;
$Q[5] = Q[5] \oplus Q[29]$;
$Q[29] = Q[29] \oplus Q[3]$;
$Q[16] = Q[16] \oplus Q[22]$;
$Q[26] = Q[26] \oplus Q[24]$;
$Q[24] = Q[24] \oplus Q[22]$;
$Q[20] = Q[20] \oplus Q[28]$;
$Q[22] = Q[22] \oplus Q[28]$;
$Q[28] = Q[28] \oplus Q[2]$;
$Q[3] = Q[3] \oplus Q[11]$;
$Q[31] = Q[31] \oplus Q[21]$;
$Q[28] = Q[28] \oplus Q[18]$;
$Q[19] = Q[19] \oplus Q[17]$;
$Q[11] = Q[11] \oplus Q[19]$;
$Q[16] = Q[16] \oplus Q[0]$;
$Q[2] = Q[2] \oplus Q[26]$;
$Q[26] = Q[26] \oplus Q[10]$;
$Q[19] = Q[19] \oplus Q[27]$;
$Q[0] = Q[0] \oplus Q[8]$;
$Q[13] = Q[13] \oplus Q[21]$;
$Q[17] = Q[17] \oplus Q[9]$;
$Q[6] = Q[6] \oplus Q[22]$;
$Q[9] = Q[9] \oplus Q[25]$;
$Q[21] = Q[21] \oplus Q[29]$;
$Q[28] = Q[28] \oplus Q[4]$;
$Q[7] = Q[7] \oplus Q[31]$;
$Q[29] = Q[29] \oplus Q[13]$;
$Q[31] = Q[31] \oplus Q[23]$;
$Q[22] = Q[22] \oplus Q[30]$;
$Q[12] = Q[12] \oplus Q[28]$;
$Q[25] = Q[25] \oplus Q[1]$;

L_2 的量子电路:

$Q[27] = Q[27] \oplus Q[19]$;
$Q[30] = Q[30] \oplus Q[22]$;
$Q[24] = Q[24] \oplus Q[8]$;
$Q[31] = Q[31] \oplus Q[7]$;
$Q[15] = Q[15] \oplus Q[23]$;
$Q[29] = Q[29] \oplus Q[5]$;
$Q[10] = Q[10] \oplus Q[18]$;
$Q[13] = Q[13] \oplus Q[29]$;
$Q[12] = Q[12] \oplus Q[28]$;
$Q[22] = Q[22] \oplus Q[14]$;
$Q[21] = Q[21] \oplus Q[13]$;
$Q[17] = Q[17] \oplus Q[9]$;
$Q[9] = Q[9] \oplus Q[25]$;
$Q[18] = Q[18] \oplus Q[2]$;
$Q[2] = Q[2] \oplus Q[26]$;
$Q[25] = Q[25] \oplus Q[27]$;
$Q[26] = Q[26] \oplus Q[28]$;
$Q[28] = Q[28] \oplus Q[30]$;
$Q[1] = Q[1] \oplus Q[27]$;
$Q[30] = Q[30] \oplus Q[6]$;
$Q[19] = Q[19] \oplus Q[3]$;
$Q[26] = Q[26] \oplus Q[20]$;
$Q[6] = Q[6] \oplus Q[8]$;
$Q[6] = Q[6] \oplus Q[0]$;

$Q[11] = Q[11] \oplus Q[27]$;
$Q[11] = Q[11] \oplus Q[29]$;
$Q[8] = Q[8] \oplus Q[16]$;
$Q[20] = Q[20] \oplus Q[4]$;
$Q[27] = Q[27] \oplus Q[13]$;
$Q[4] = Q[4] \oplus Q[22]$;
$Q[22] = Q[22] \oplus Q[24]$;
$Q[16] = Q[16] \oplus Q[24]$;
$Q[13] = Q[13] \oplus Q[15]$;
$Q[24] = Q[24] \oplus Q[0]$;
$Q[3] = Q[3] \oplus Q[29]$;
$Q[29] = Q[29] \oplus Q[7]$;
$Q[29] = Q[29] \oplus Q[23]$;
$Q[23] = Q[23] \oplus Q[31]$;
$Q[7] = Q[7] \oplus Q[15]$;
$Q[0] = Q[0] \oplus Q[2]$;
$Q[27] = Q[27] \oplus Q[5]$;
$Q[24] = Q[24] \oplus Q[10]$;
$Q[5] = Q[5] \oplus Q[31]$;
$Q[15] = Q[15] \oplus Q[9]$;
$Q[16] = Q[16] \oplus Q[2]$;
$Q[31] = Q[31] \oplus Q[9]$;
$Q[10] = Q[10] \oplus Q[26]$;
$Q[2] = Q[2] \oplus Q[10]$;
$Q[10] = Q[10] \oplus Q[20]$;
$Q[4] = Q[4] \oplus Q[12]$;
$Q[12] = Q[12] \oplus Q[20]$;
$Q[7] = Q[7] \oplus Q[1]$;
$Q[14] = Q[14] \oplus Q[30]$;
$Q[30] = Q[30] \oplus Q[8]$;
$Q[8] = Q[8] \oplus Q[18]$;
$Q[29] = Q[29] \oplus Q[21]$;
$Q[20] = Q[20] \oplus Q[14]$;
$Q[14] = Q[14] \oplus Q[6]$;
$Q[9] = Q[9] \oplus Q[1]$;
$Q[6] = Q[6] \oplus Q[22]$;
$Q[18] = Q[18] \oplus Q[12]$;
$Q[12] = Q[12] \oplus Q[28]$;
$Q[1] = Q[1] \oplus Q[17]$;
$Q[23] = Q[23] \oplus Q[17]$;
$Q[22] = Q[22] \oplus Q[30]$;
$Q[17] = Q[17] \oplus Q[19]$;
$Q[9] = Q[9] \oplus Q[3]$;
$Q[28] = Q[28] \oplus Q[20]$;
$Q[20] = Q[20] \oplus Q[4]$;
$Q[19] = Q[19] \oplus Q[11]$;
$Q[7] = Q[7] \oplus Q[25]$;
$Q[9] = Q[9] \oplus Q[19]$;
$Q[19] = Q[19] \oplus Q[21]$;
$Q[21] = Q[21] \oplus Q[5]$;
$Q[26] = Q[26] \oplus Q[18]$;
$Q[25] = Q[25] \oplus Q[17]$;
$Q[0] = Q[0] \oplus Q[8]$;
$Q[3] = Q[3] \oplus Q[27]$;
$Q[27] = Q[27] \oplus Q[19]$;
$Q[5] = Q[5] \oplus Q[29]$;
$Q[15] = Q[15] \oplus Q[23]$;
$Q[29] = Q[29] \oplus Q[13]$;
$Q[8] = Q[8] \oplus Q[24]$;
$Q[18] = Q[18] \oplus Q[10]$;
$Q[31] = Q[31] \oplus Q[7]$;
$Q[17] = Q[17] \oplus Q[9]$;

附录 B S_1 和 AES 算法 S 盒的元素表示的转换矩阵及其推导过程

Appendix B The transformation matrix of the element representation(the S-box of AES and S_1) and its derivation

转换矩阵的推导思路:

AES 算法的 S 盒及 S_1 的元素均在有限域 GF(256) 上, 但前者(AES 算法的 S 盒)用于构造有限域的不可约多项式是 $x^8 + x^4 + x^3 + x + 1$, 而后者(S_1)用于构造有限域的不可约多项式是 $x^8 + x^7 + x^3 + x + 1$。若假设前者在有限域 GF(256) 上的一组基为 $a^7, a^6, \cdots, a, 1$, 后者在有限域 GF(256) 上的一组基为 $b^7, b^6, \ldots, b, 1$。那么可得

$$\begin{cases} a^8 + a^4 + a^3 + a + 1 = 0 \\ b^8 + b^7 + b^3 + b + 1 = 0 \end{cases}$$

同时, 假设

$$b = c_7 a^7 + c_6 a^6 + c_5 a^5 + c_4 a^4 + c_3 a^3 + c_2 a^2 + c_1 a + c_0$$

其中 $c_i \in \{0,1\}, 0 \le i \le 7$。那么通过联立以上等式, 我们可以解得

$$\begin{cases} c_0 = 0 \\ c_1 = 1 \\ c_2 = 0 \\ c_3 = 0 \\ c_4 = 0 \\ c_5 = 0 \\ c_6 = 0 \\ c_7 = 1 \end{cases}$$

通过进一步计算, 我们可以得到$b^7, b^6, \ldots, b$,1在$a^7, a^6, \ldots, a$,1下的表示, 即得到两组基之间的线性关系。从而得到, 将S_1的域元素表示转化为 AES 算法 S 盒的域元素表示的转换矩阵:

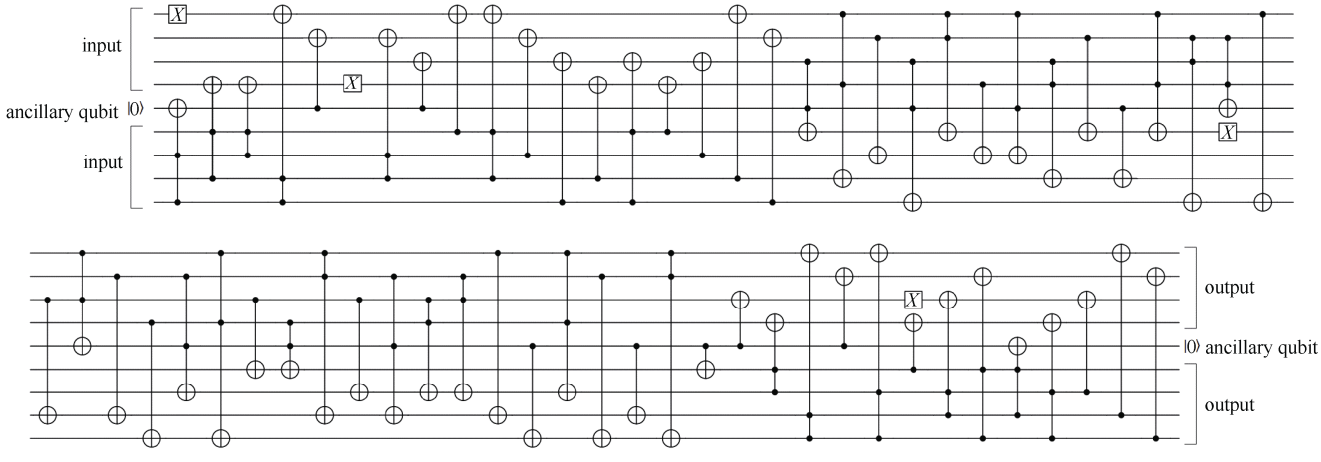
$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

相应的, 我们能得到将 AES 算法 S 盒的域元素表示转化为S_1的域元素表示的转换矩阵:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

附录 C S_0的量子电路

Appendix C The quantum circuit of S_0



孙壮 于 2018 年在山东大学软件工程专业获得学士学位。现在中国科学院大学网络空间安全专业攻读硕士学位。研究领域为密码理论与技术。研究兴趣包括量子电路优化。Email: sunzhuang@iie.ac.cn

黄震宇 于 2010 年在中国科学院数学与系统科学研究院应用数学专业获得博士学位。现任信息工程研究所信息安全国家重点实验室副研究员。研究领域为密码理论与技术。研究兴趣包括密码分析、计算机代数。Email: huangzhenyu@iie.ac.cn