---------------------------------------------------------------------

# 文档声明

## 文档性质说明

本文档是FIPS 197 Advanced Encryption Standard (AES)标准的部分内容的中文译本。翻译内容并非该标准的完整官方版本，仅覆盖原文作者选择或认为相关的特定章节或主题。本译文旨在方便中文读者理解标准的核心概念，不可替代原始英文标准。

## 合规性提醒

本译文是非正式的、由个人提供的翻译，不具任何官方效力、法律效力或认证意义。仅用于信息参考与学习目的。任何涉及密码实现、产品认证、合规性评估或法律事务的场景，必须查阅、引用并遵循NIST官方网站发布的原始、最新版本的英文标准。

## 版权与使用说明

原始标准版权：FIPS 197 Advanced Encryption Standard (AES)的版权由美国国家标准与技术研究院(NIST)所有。NIST 出版物通常被视为美国政府的作品，通常属于 公有领域(Public Domain)。用户可自由查阅、下载和使用原始标准。

​ 衍生译文版权：本译文的编排、呈现及可能的注释（如有）由译者（作者）完成。使用者可基于知识共享原则（如 CC BY 或 CC BY-SA）免费阅读、参考、分享本译文，但需注明来源（即包含本声明或链接）。

​ 禁止商业性直接利用：不得将本译文（包括格式排版）用于任何形式的直接商业性出售或牟利。若需商业使用，应基于原始NIST文档。

## 内容免责

​​ “作者个人见解”强调：翻译本身涉及对技术术语和复杂概念的理解与转换。译者在力求准确反映原文意图的基础上，包含其个人的理解与诠释。因此，译文可能与原始标准在措辞、细节侧重或细微含义上存在差异。​​

​​非权威性：译者与发布者不声明或保证本译文的精确性、完整性、可靠性或适用性。

​​ **无责任承担**：使用者理解并同意，译者、发布者及相关方对因使用、误用或无法使用本译文所载信息而导致的任何损失、损害或后果（包括技术、财务或法律后果）概不负责。

​​最终依据：所有技术实现、决策和争议解决应且仅应以NIST官方发布的原始英文标准为准。

内容完善说明​​

本译文基于特定版本的FIPS 197 Advanced Encryption Standard (AES)进行翻译。原文标准可能被修订和更新。译者不承诺本译文会持续追踪或反映NIST标准的最新修订。译文内容可能存在疏漏或错误。欢迎读者提供建设性反馈以便改进，但译者保留是否采纳修改的最终决定权。

---------------------------------------------------------------------

**联邦信息处理标准出版物——高级加密标准（AES）​**

# 0 摘要

2000年，NIST宣布选择Rijndael分组密码家族作为高级加密标准（AES）竞赛的获胜者。分组密码是许多密码服务的基础，尤其是那些确保数据保密性的服务。

本标准规定了Rijndael家族的三个成员：AES-128、AES-192和AES-256。它们均对128位的数据块进行转换，数字后缀表示相关加密密钥的位长。

# 1引言

块是指给定固定长度的位序列。分组密码是由称为密钥的位序列参数化的块置换家族。

1997年，NIST启动了高级加密标准（AES）开发工作[1]，并呼吁公众提交分组密码候选算法。分组密码是许多密码服务的基础，尤其是那些确保数据保密性的服务。2000年，NIST宣布选择Rijndael[2,3]作为AES算法。

本标准规定了Rijndael的三种实现：AES-128、AES-192和AES-256，其中后缀表示密钥的位长。每种情况下，块大小（即数据输入和输出的长度）均为 128 位。Rijndael支持本标准未采用的其他块大小和密钥长度。

本标准的结构如下：

* 第2节定义了本标准中的术语、缩略语、算法参数、符号和函数。
* 第3节描述了位、字节和字的排序及索引的符号和约定。
* 第4节解释了AES规范的一些数学组件：有限域算术和有限域元素的固定矩阵乘法。
* 第5节规定了AES-128、AES-192和AES-256。
* 第6节提供了关于密钥长度要求、密钥限制、参数扩展以及不同平台实现建议的实施指南。
* 附录A给出了AES-128、AES-192和AES-256 的密钥扩展例程示例。
* 附录B给出了AES-128调用的分步示例。
* 附录C提供了NIST网站的参考，该网站包含AES-128、AES-192和AES-256的大量示例向量。
* 附录D总结了对本出版物原始版本的更新。

# 2 定义

## 2.1 术语和缩略语

本标准使用以下定义：

1. AES

高级加密标准

1. 仿射变换（Affine transformation）

由矩阵乘法followed by向量加法组成的变换

1. 数组（Array）

一种固定大小的数据结构，用于存储元素集合，其中每个元素由其整数索引标识

1. 位（Bit）

二进制数字：0或1

1. 块（Block）

给定固定长度的位序列。在本标准中，块由128位组成，有时表示为字节或字的数组

1. 分组密码（Block cipher）

由密钥参数化的块置换家族

1. 字节（Byte）

8位的序列

1. 等效逆密码（Equivalent inverse cipher）

CIPHER()的逆的替代规范，其结构与CIPHER()相似，并以修改后的密钥调度作为输入

1. 密钥（Key）

确定从分组密码家族中选择置换的分组密码参数

1. 密钥调度（Key schedule）

通过KEYEXPANSION()从密钥生成的轮密钥序列

1. Rijndael

NIST选为AES竞赛获胜者的分组密码

1. 轮（Round）

在CIPHER()、INVCIPHER()和EQINVCIPHER() 的规范中迭代Nr次的状态变换序列。该序列由四个变换组成，但在其中一次迭代中，会省略其中一个变换

1. 轮密钥（Round key）

使用密钥扩展例程从分组密码密钥导出的Nr+1个四字数组之一；每个轮密钥是AES分组密码中ADDROUNDKEY()实例的输入

1. 状态（State）

AES分组密码的中间结果，表示为4行Nb列的二维字节数组

1. S盒（S-box）

在SUBBYTES()和KEYEXPANSION()中使用的非线性substitution表，用于执行字节值的一对一替换

1. 字（Word）

32位的组，可视为单个实体或4字节的数组

## 2.2 函数列表

本标准规定了以下函数：

1. ADDROUNDKEY()

将轮密钥与状态组合的状态变换

1. AES-128()

本标准规定的使用128位密钥的分组密码

1. AES-192()

本标准规定的使用192位密钥的分组密码

1. AES-256()

本标准规定的使用256位密钥的分组密码

1. CIPHER()

作为AES-128、AES-192和AES-256基础的块变换；密钥调度和轮数是该变换的参数

1. EQINVCIPHER()

CIPHER()的逆，其中dw替代w作为密钥调度参数

1. INVCIPHER()

CIPHER()的逆

1. INVMIXCOLUMNS()

MIXCOLUMNS()的逆

1. INVSBOX()

SBOX()的逆

1. INVSHIFTROWS()

SHIFTROWS()的逆

1. INVSUBBYTES()

SUBBYTES()的逆

1. KEYEXPANSION()

从密钥生成轮密钥的例程

1. KEYEXPANSIONEIC()

为等效逆密码生成修改后的轮密钥的例程

1. MIXCOLUMNS()

对状态的每一列数据进行混合（彼此独立）以生成新列的状态变换

1. ROTWORD()

对字的四个字节进行循环置换的字变换

1. SBOX()

由S盒定义的字节变换

1. SHIFTROWS()

对状态的后三行按不同偏移量进行循环移位的状态变换

1. SUBBYTES()

对状态的每个字节独立应用S盒的状态变换

1. SUBWORD()

对字的四个字节各应用一次S盒的字变换

1. XTIMES()

将输入字节的多项式表示乘以x（模m (x)）以生成输出字节的多项式表示的字节变换

## 2.3 算法参数和符号

1. b⁻¹：GF (2⁸) 中元素b的乘法逆元
2. b̃：AES S盒中仿射变换的输入
3. dw：用于等效逆密码的密钥调度字数组
4. GF (2)：具有两个元素的有限域
5. GF (2⁸)：具有256个元素的有限域
6. in：CIPHER()或INVCIPHER()的数据输入，表示为索引从0到15的16字节数组
7. m (x)：本标准中规定的用于将字节表示为GF (2⁸)元素的多项式表示的模数
8. key：组成AES-128、AES-192或AES-256密钥的Nk个字的数组
9. Nb：组成状态的列数，其中每列是一个32位字。在本标准中，Nb=4
10. Nk：组成密钥的32位字数。对于AES-128、AES-192和AES-256，Nk分别为4、6和8（见第 6.3 节）
11. Nr：轮数。对于AES-128、AES-192和AES-256，Nr分别为10、12和14
12. out：CIPHER()或INVCIPHER()的数据输出，表示为索引从0到15的16字节数组
13. Rcon：轮常数的字数组
14. state：状态，表示为16字节的二维数组，行和列的索引从0到3
15. u [i]：对于字或字节的一维数组u，数组中由非负整数i索引的元素
16. u [i..i+3]：对于字数组u，序列u [i]、u [i+1]、u [i+2]、u [i+3]
17. w：密钥调度的字数组
18. ⊕：位上的异或运算、字节上的按位异或运算或字上的按位异或运算
19. {}：十六进制或二进制表示的字节分隔符
20. ・：GF(2⁸)中的乘法
21. \*：整数乘法
22. ←：伪代码中变量的赋值

# 3 符号和约定

## 3.1 输入和输出

位是二进制数字——0或1。块是128位的序列；AES分组密码的数据输入和输出是块。AES分组密码的另一个输入称为密钥，是事先建立并在分组密码的多次调用中保持不变的位序列。AES-128、AES-192和AES-256的密钥长度分别为128位、192位和256位。

## 3.2 字节

AES算法中的基本处理单元是字节——8位的序列。

字节值用大括号之间的八个位的连接表示，例如{10100011}。当用索引变量表示字节的位时，本标准中的约定是索引从左到右递减，即{b₇b₆b₅b₄b₃b₂b₁b₀}。

也可以方便地使用十六进制表示法表示字节值。16个十六进制字符表示4位序列，如表1所示。一个字节由一对有序的十六进制字符表示，其中对中的左字符表示最左边的四位（即b₇、b₆、b₅、b₄），右字符表示最右边的四位（即b₃、b₂、b₁、b₀）。例如，字节10100011的十六进制形式是a3。

表 1 4位序列的十六进制表示

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序列 | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
| 字符 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 序列 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 字符 | 8 | 9 | a | b | c | d | e | f |

## 3.3 字节序列的索引

为了明确地将数据和密钥输入表示为字节序列，本标准采用以下索引约定。给定一个8k位的序列：r₀r₁r₂…r₍₈ₖ₋₃₎r₍₈ₖ₋₂₎r₍₈ₖ₋₁₎（3.1）

（对于某个正整数 k），0≤j≤k-1 的字节 aⱼ定义如下：aⱼ={r₈ⱼr₍₈ⱼ₊₁₎…r₍₈ⱼ₊₇₎}（3.2）。

因此，例如，数据块：r₀r₁r₂…r₁₂₅r₁₂₆r₁₂₇（3.3）

由字节序列表示：a₀a₁a₂…a₁₃a₁₄a₁₅（3.4）

其中:

a₀={r₀r₁…r₇}；

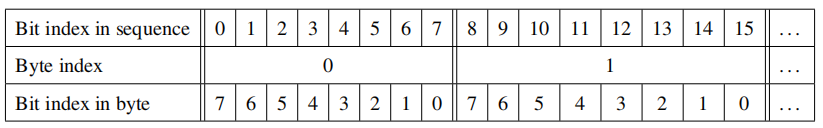
a₁={r₈r₉…r₁₅}；

……

a₁₅={r₁₂₀r₁₂₁…r₁₂₇}。

如第3.2节所述，任何单个字节内的位按从左到右的顺序递减索引。这种顺序对于第4节中描述的字节上的有限域算术来说更为自然。表2说明了字节序列的两种位索引类型。

表2 字节和位的索引



## 3.4 状态

AES分组密码的算法在内部对一个称为状态的二维（4×4）字节数组进行操作。在状态数组（记为s）中，每个字节有两个索引：行索引r（范围 0≤r<4）和列索引c（范围 0≤c<4）。状态的单个字节表示为sᵣ,ₑ或s[r,c]。

在第5节的AES分组密码算法规范中，第一步是将输入字节数组in₀、in₁、…、in₁₅复制到状态数组s中，如下所示：s[r,c]=in[r+4c]，其中0≤r<4且0≤c<4（3.6）。

然后对状态数组应用一系列变换，之后将其最终值复制到输出字节数组out₀、out₁、…、out₁₅中，如下所示：out [r+4c]=s [r,c]，其中0≤r<4且0≤c<4（3.7）。

输入和输出的索引与状态数组的索引之间的对应关系如图1所示。

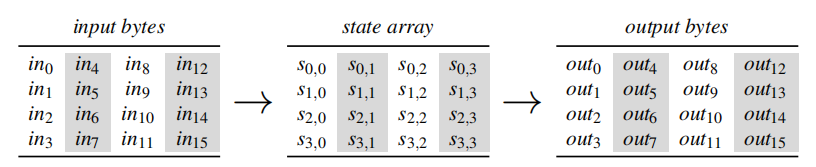


图 1 状态阵列输入与输出

## 3.5 字数组

字是4字节的序列；一个块由4个字组成。状态数组s的四列被解释为4个字的数组v，按照图1的符号表示如下：v₀=（s₀,₀；s₁,₀；s₂,₀；s₃,₀）ᵀ，v₁=（s₀,₁；s₁,₁；s₂,₁；s₃,₁）ᵀ，v₂=（s₀,₂；s₁,₂；s₂,₂；s₃,₂）ᵀ，v₃=（s₀,₃；s₁,₃；s₂,₃；s₃,₃）ᵀ（3.8）。

因此，s的列索引c成为v的索引，s的行索引r成为每个字中四个字节的索引。

给定字的一维数组u，u [i]表示由i索引的字，四个字的序列u [i]、u [i+1]、u [i+2]、u [i+3]表示为u [i..i+3]。

# 4 数学预备知识

对于第5节中规定的AES算法的某些变换，状态数组中的每个字节被解释为有限域（也称为伽罗瓦域）的256个元素之一，记为GF (2⁸)。

为了定义GF (2⁸)中的加法和乘法，每个字节{b₇b₆b₅b₄b₃b₂b₁b₀} 被解释为多项式，记为b (x)，如下所示：b (x)=b₇x⁷+b₆x⁶+b₅x⁵+b₄x⁴+b₃x³+b₂x²+b₁x+b₀（4.1）。

例如，01100011由多项式x⁶+x⁵+x+1表示。

## 4.1 GF(2⁸)中的加法

为了求有限域GF (2⁸)中两个元素的和，将表示元素的多项式的系数按模2相加（即使用异或运算，记为⊕），因此1⊕1=0，1⊕0=1，0⊕0=0。

等效地，两个字节可以通过对字节中每对对应位应用异或运算来相加。因此，{a₇a₆a₅a₄a₃a₂a₁a₀}与{b₇b₆b₅b₄b₃b₂b₁b₀}的和是{a₇⊕b₇ a₆⊕b₆ a₅⊕b₅ a₄⊕b₄ a₃⊕b₃ a₂⊕b₂ a₁⊕b₁ a₀⊕b₀}（在第5.1.4节中，此定义扩展到字）。

例如，以下三种加法表示是等效的：

（x⁶+x⁴+x²+x+1）+（x⁷+x+1）=x⁷+x⁶+x⁴+x²（多项式）

{01010111}⊕{1000011}={11010100}（二进制）

{57}⊕{83}={d4}（十六进制）。

由于多项式的系数按模2约简，系数1等同于系数-1，因此加法等同于减法。例如，x⁴+x²与x⁴-x²、-x⁴+x²和-x⁴-x²表示相同的有限域元素。同样，任何元素与自身的和是零元素。

## 4.2 GF(2⁸)中的乘法

符号・表示GF (2⁸)中的乘法。从概念上讲，这种乘法在两个字节上的定义分为两步：1）将表示字节的两个多项式作为多项式相乘；2）将所得多项式按以下固定多项式约简：m (x)=x⁸+x⁴+x³+x+1（4.3）。

在这两个步骤中，多项式的各个系数都按模2约简。

因此，如果b (x)和c (x)表示字节b和c，则b・c由其多项式乘积的以下模约简表示：b (x) c (x) mod m (x)（4.4）。

对中间步骤的计算b (x) c (x)应用m (x)的模约简可能是有用的；因此，考虑c (x)=x（即c={02}）的特殊情况。特别是，乘积b・{02}可以表示为b的函数，记为XTIMES(b)，如下所示：XTIMES(b)={ b₆b₅b₄b₃b₂b₁b₀0 }，若 b₇=0;

{ b₆b₅b₄b₃b₂b₁b₀0 }⊕{000011011 }，若 b₇=1。

乘以x的更高次幂（如{04}、{08}和{10}）可以通过重复应用XTIMES()来实现。例如，设b={57}：

{57}・{01}={57}

{57}・{02}=XTIMES ({57})={ae}

{57}・{04}=XTIMES ({ae})={47}

{57}・{08}=XTIMES ({47})={8e}

{57}・{10}=XTIMES ({8e})={07}

{57}・{20}=XTIMES ({07})={0e}

{57}・{40}=XTIMES ({0e})={1c}

{57}・{80}=XTIMES ({1c})={38}（4.6）。

这些乘积有助于计算57的任何倍数。例如，因为{13}={10}⊕{02}⊕{01}，所以：{57}・{13}={57}・({01}⊕{02}⊕{10})={57}⊕{ae}⊕{07}={fe}。

## 4.3 字与固定矩阵的乘法

AES分组密码算法中的两个变换:MIXCOLUMNS()和INVMIXCOLUMNS()可以用矩阵乘法表示。具体而言，每个变换都规定了一个不同的固定矩阵。对于这两个矩阵，矩阵的16个元素中的每个元素都是单个指定字的一个字节，这里记为 [a₀,a₁,a₂,a₃]。

给定变换的输入字[b₀,b₁,b₂,b₃]，输出字[d₀,d₁,d₂,d₃] 由有限域算术确定如下：

d₀=(a₀・b₀)⊕(a₃・b₁)⊕(a₂・b₂)⊕(a₁・b₃)

d₁=(a₁・b₀)⊕(a₀・b₁)⊕(a₃・b₂)⊕(a₂・b₃)

d₂=(a₂・b₀)⊕(a₁・b₁)⊕(a₀・b₂)⊕(a₃・b₃)

d₃=(a₃・b₀)⊕(a₂・b₁)⊕(a₁・b₂)⊕(a₀・b₃)（4.8）。

等式（4.8）的矩阵形式为：[d₀；d₁；d₂；d₃]ᵀ = [[a₀,a₃,a₂,a₁], [a₁,a₀,a₃,a₂], [a₂,a₁,a₀,a₃], [a₃,a₂,a₁,a₀]] × [b₀；b₁；b₂；b₃]ᵀ（4.9）。

## 4.4 GF(2⁸)中的乘法逆元

对于字节b≠{00}，其乘法逆元是唯一的字节，记为b⁻¹，满足：b・b⁻¹={01}（4.10）。

AES分组密码规范中的SUBBYTES()变换的定义涉及GF(2⁸)中的乘法逆元，可按如下方式计算：b⁻¹=b²⁵⁴（4.11）。

或者，设b (x)是表示b的多项式。可以将扩展欧几里得算法[5]应用于b (x)和m (x)，找到多项式a (x)和c (x)，使得：b (x) a (x)+m (x) c (x)=1（4.12）。

因此，a (x)是表示b⁻¹的多项式。

# 5 算法规范

执行AES-128、AES-192或AES-256 的通用函数记为CIPHER()；其逆记为 INVCIPHER()。

CIPHER()和INVCIPHER()算法的核心是称为轮的一系列固定状态变换。每轮需要一个额外的输入，称为轮密钥；轮密钥是一个块，通常表示为四个字的序列（即16字节）。

扩展例程（记为KEYEXPANSION()）以分组密码密钥作为输入，生成轮密钥作为输出。具体而言，KEYEXPANSION() 的输入表示为字数组（记为key），输出是称为密钥调度的字的扩展数组（记为w）。

分组密码AES-128、AES-192和AES-256 在三个方面有所不同：1）密钥长度；2）轮数（决定所需密钥调度的大小）；3）KEYEXPANSION()中的递归规范。对于每种算法，轮数记为Nr，密钥的字数记为Nk（对于一般的Rijndael，状态中的字数记为Nb；在本标准中，Nb=4）。Nk、Nb和Nr的具体值如表3所示。Rijndael的其他配置均不符合本标准。

表 3 密钥 - 块 - 轮组合

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nk：密钥字数 | 密钥位数 | Nb：块的字数 | 块位数 | Nr：轮数 |
| AES-128 | 4 | 128 | 4 | 128 | 10 |
| AES-192 | 6 | 192 | 4 | 128 | 12 |
| AES-256 | 8 | 256 | 4 | 128 | 14 |

关于密钥长度、块大小和轮数的实现问题，见第6.3节。

CIPHER()的三个输入是：1）数据输入in，表示为16字节的线性数组；2）实例的轮数Nr；3）轮密钥。因此：

AES-128 (in,key)=CIPHER (in,10,KEYEXPANSION (key))

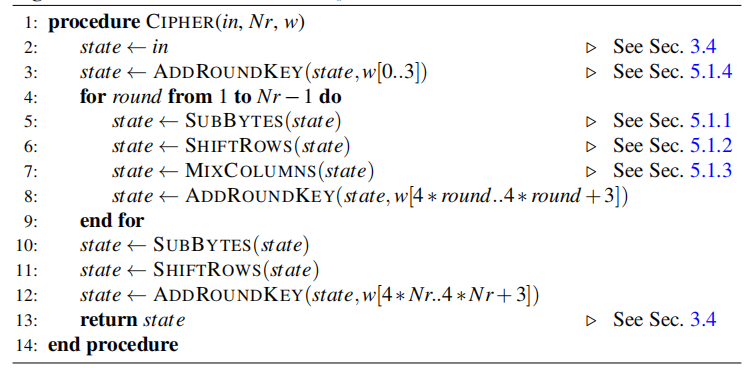
AES-192 (in,key)=CIPHER (in,12,KEYEXPANSION (key))

AES-256 (in,key)=CIPHER (in,14,KEYEXPANSION (key))。

逆置换通过在等式5.1中用INVCIPHER()替换CIPHER()来定义。

CIPHER()、KEYEXPANSION()和INVCIPHER()的规范分别在第5.1、5.2和5.3节中给出。

**算法1 CIPHER()**



## 5.1 CIPHER()

CIPHER()规范中的轮由以下四种面向字节的状态变换组成：

1. SUBBYTES()：对每个字节应用substitution表（S盒）。
2. SHIFTROWS()：按不同偏移量移位状态数组的行。
3. MIXCOLUMNS()：混合状态数组每列内的数据。
4. ADDROUNDKEY()：将轮密钥与状态组合。

这四种变换在第5.1.1–5.1.4节中规定。在这些规范中，变换后的位、字节或块通过在原始变量上添加上标符号0来表示，即bᵢ'、b'、sᵢ,ⱼ'或s'。

ADDROUNDKEY()的轮密钥由KEYEXPANSION()生成，其在第5.2节中规定。具体而言，密钥调度表示为4\*(Nr+1)个字的数组w。

CIPHER()在算法1的伪代码中规定。

**算法 1 CIPHER()的伪代码**

1: 过程CIPHER(in, Nr, w)

2: state ← in（见第3.4节）

3: state ← ADDROUNDKEY(state, w[0..3]) （见第5.1.4节）

4: 对于round从1到Nr-1：

5: state ← SUBBYTES(state)（见第5.1.1节）

6: state ← SHIFTROWS(state)（见第5.1.2节）

7: state ← MIXCOLUMNS(state)（见第5.1.3节）

8: state ← ADDROUNDKEY(state, w [4round..4round+3])

9: 结束循环

10: state ← SUBBYTES(state)

11: state ← SHIFTROWS(state)

12: state ← ADDROUNDKEY(state, w[4Nr..4Nr+3])

13: 返回state（见第3.4节）

14: 结束过程

第一步（第2行）是使用第3.4节中的约定将输入复制到状态数组中。在初始轮密钥加法（第3行）之后，状态数组通过Nr次轮函数应用（第4-12行）进行变换；最后一轮（第10-12行）有所不同，因为省略了MIXCOLUMNS()变换。然后将最终状态作为输出返回（第13行），如第3.4节所述。

### 5.1.1 SUBBYTES()

SUBBYTES()是一种可逆的非线性状态变换，其中substitution表（称为S盒）被独立地应用于状态中的每个字节。AES S盒记为SBox ()。

设b表示SBox ()的输入字节，c表示常数字节{01100011}。输出字节b'=SBox(b)由以下两个变换组合构成：

定义中间值b̃如下，其中b⁻¹是b的乘法逆元，如第4.4节所述：b̃={00}，若b={00}；b⁻¹，若b≠{00}。

对b̃的位应用以下仿射变换以生成b'的位：bᵢ'=b̃ᵢ⊕b̃₍ᵢ₊₄₎mod 8⊕b̃₍ᵢ₊₅₎mod 8⊕b̃₍ᵢ₊₆₎mod 8⊕b̃₍ᵢ₊₇₎mod 8⊕cᵢ（5.3）。

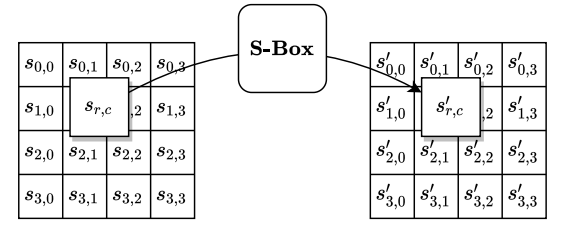
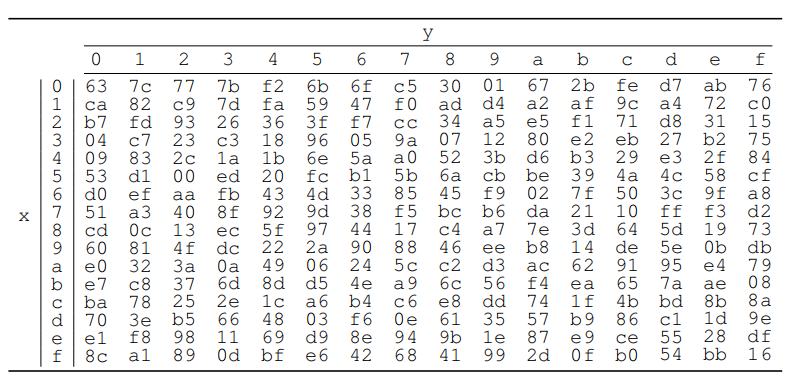


图 2 SUBBYTES()示意图

表 4 S 盒矩阵



### 5.1.2 SHIFTROWS()

SHIFTROWS()是一种状态变换，其中状态的后三行中的字节被循环移位。字节移位的位数取决于行索引r，如下所示：sᵣ,ₑ'=sᵣ,₍ₑ₊ᵣ₎mod 4，其中0≤r<4且0≤c<4（5.5）。

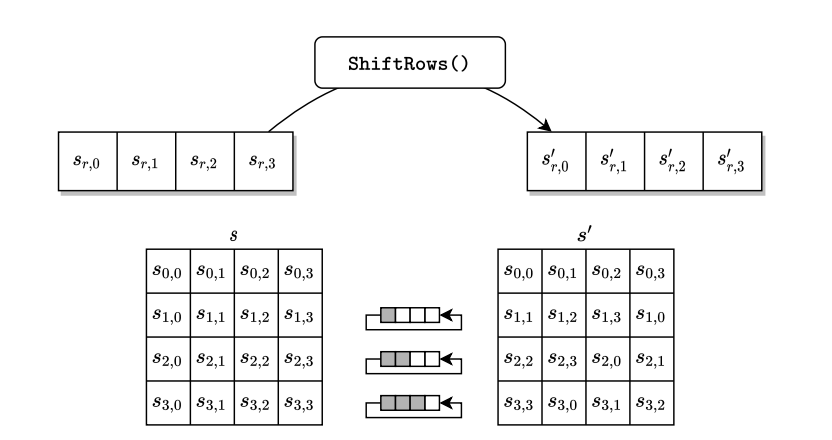


图 3 SHIFTROWS()示意图

### 5.1.3 MIXCOLUMNS()

MIXCOLUMNS()是一种状态变换，它将状态的四个列中的每个列与单个固定矩阵相乘，如第4.3节所述，其元素取自以下字[a₀,a₁,a₂,a₃]=[{02},{01},{01},{03}]（5.6）。

因此，[s₀,ₑ'；s₁,ₑ'；s₂,ₑ'；s₃,ₑ']ᵀ = [[02,03,01,01], [01,02,03,01], [01,01,02,03], [03,01,01,02]] × [s₀,ₑ；s₁,ₑ；s₂,ₑ；s₃,ₑ]ᵀ，其中0≤c<4（5.7），

因此，各个输出字节定义如下：

s₀,ₑ'=({02}・s₀,ₑ)⊕({03}・s₁,ₑ)⊕s₂,ₑ⊕s₃,ₑ

s₁,ₑ'=s₀,ₑ⊕({02}・s₁,ₑ)⊕({03}・s₂,ₑ)⊕s₃,ₑ

s₂,ₑ'=s₀,ₑ⊕s₁,ₑ⊕({02}・s₂,ₑ)⊕({03}・s₃,ₑ)

s₃,ₑ'=({03}・s₀,ₑ)⊕s₁,ₑ⊕s₂,ₑ⊕({02}・s₃,ₑ)（5.8）。

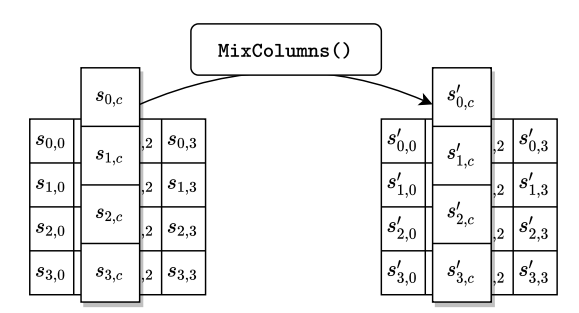


图 4 MIXCOLUMNS()示意图

### 5.1.4 ADDROUNDKEY()

ADDROUNDKEY()是一种状态变换，其中轮密钥通过应用按位异或运算与状态组合。具体而言，每个轮密钥由密钥调度（第5.2节中描述）中的四个字组成，每个字与状态的一列组合如下：[s₀,ₑ',s₁,ₑ',s₂,ₑ',s₃,ₑ']=[s₀,ₑ,s₁,ₑ,s₂,ₑ,s₃,ₑ]⊕[w₍₄₎\*round +c₎]，其中0≤c<4

其中round是0≤round≤Nr范围内的值，w[i]是第5.2节中描述的密钥调度字数组。在CIPHER()的规范中，ADDROUNDKEY()被调用Nr+1次——一次在轮函数的第一次应用之前（见算法1），以及在Nr轮中的每一轮内一次，此时1≤round≤Nr。

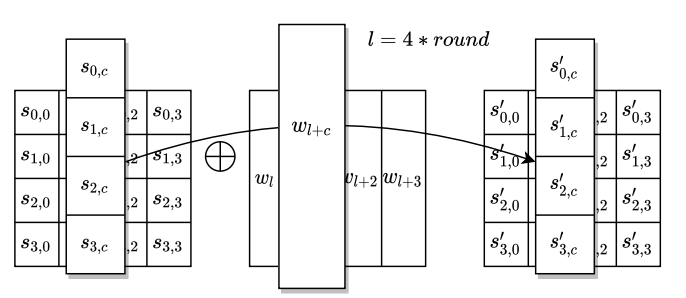


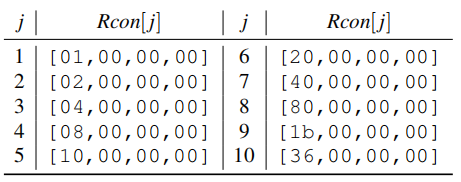
图 5 ADDROUNDKEY()示意图

## 5.2 KEYEXPANSION()

KEYEXPANSION()是一种应用于密钥以生成4\*(Nr+1)个字的例程。因此，为CIPHER()规范中ADDROUNDKEY()的Nr+1次应用中的每一次生成四个字，如第5.1.4节所述。该例程的输出由字的线性数组组成，记为w[i]，其中i的范围是0≤i<4\*(Nr+1)。

KEYEXPANSION()调用10个固定字，记为Rcon[j]，其中1≤j≤10。这10个字称为轮常数。对于AES-128，在每个10轮密钥的生成中调用一个不同的轮常数。对于AES-192和AES-256，密钥扩展例程分别调用这些相同常数中的前8个和前7个。Rcon[j]的值在表5中以十六进制表示法给出：

表 5 轮常数

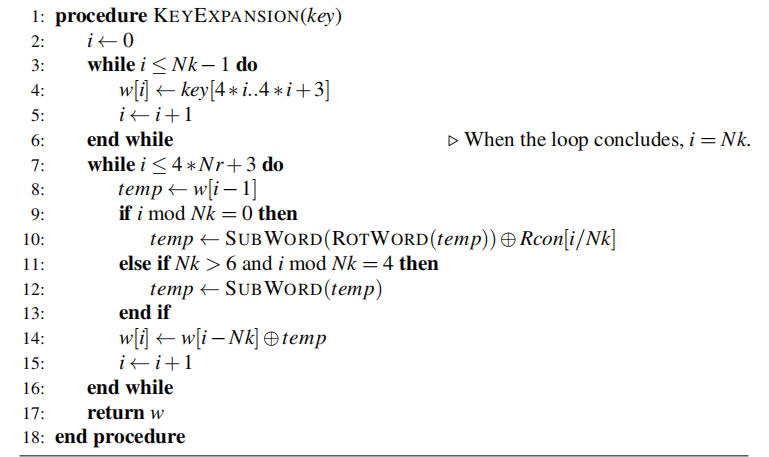


KEYEXPANSION()中调用了两种字变换：ROTWORD()和SUBWORD()。给定表示为四个字节的序列 [a₀,a₁,a₂,a₃] 的输入字，ROTWORD ([a₀,a₁,a₂,a₃])=[a₁,a₂,a₃,a₀]（5.10），且SUBWORD ([a₀,…,a₃])=[SBox (a₀),SBox (a₁),SBox (a₂),SBox (a₃)]（5.11）。

密钥的扩展根据算法2中的伪代码进行。扩展密钥的前Nk个字是密钥本身。每个后续字w [i]由前一个字w [i-1]和Nk个位置之前的字w [i-Nk]递归生成，如下所示：

1. 如果i是Nk的倍数，则w [i]=w [i-Nk]⊕SUBWORD (ROTWORD (w[i-1])) ⊕ Rcon [i/Nk]。
2. 对于AES-256，如果i+4是8的倍数，则w[i]=w[i-Nk]⊕SUBWORD (w[i-1])。
3. 对于所有其他情况，w[i]=w[i-Nk]⊕w[i-1]。

**算法 2 密钥扩展（KEYEXPANSION()）**



## 5.3 INVCIPHER()

为了实现INVCIPHER()，CIPHER()规范（第5.1节）中的变换被求逆并按相反顺序执行。状态的逆变换——记为INVSHIFTROWS()、INVSUBBYTES()、INVMIXCOLUMNS()ADDROUNDKEY()——在第5.3.1–5.3.4节中描述。

INVCIPHER()在算法3的伪代码中描述，其中数组w表示第5.2节中描述的密钥调度。

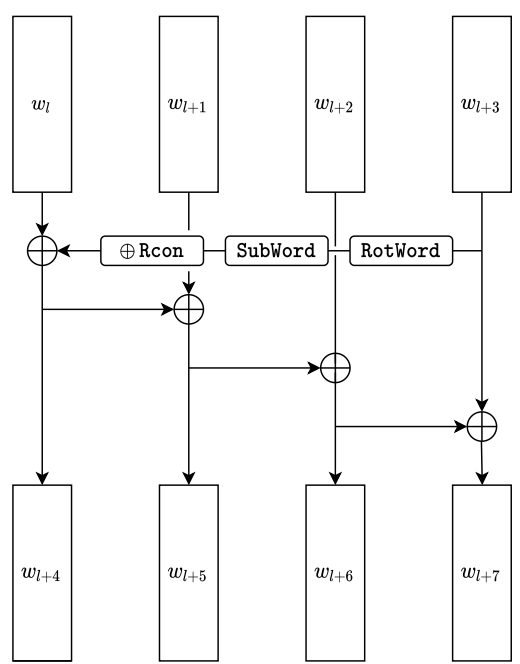


图 6 AES-128的密钥扩展（KEYEXPANSION()），用于生成4 ≤i< 44的字w [i]，其中l的取值范围是0到36之间4的倍数

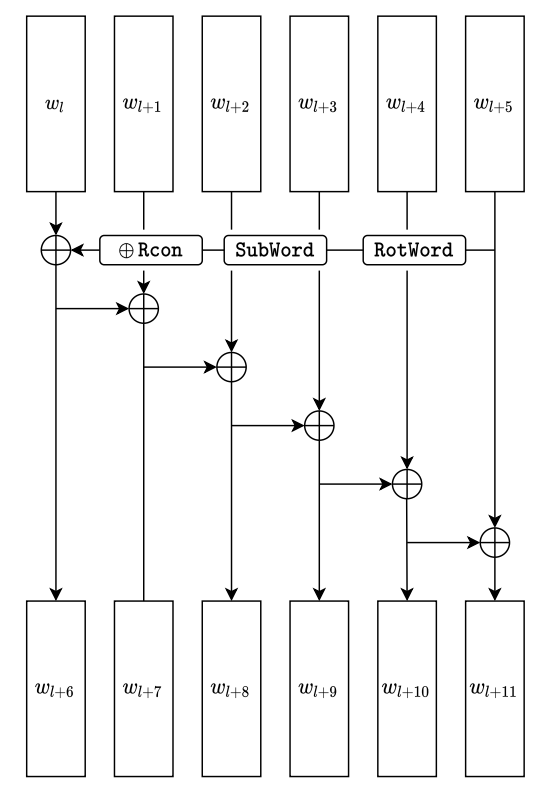


图 7 AES-192的密钥扩展（KEYEXPANSION()），用于生成6≤i< 52 的字w[i]，其中l的取值范围是0到42之间6的倍数

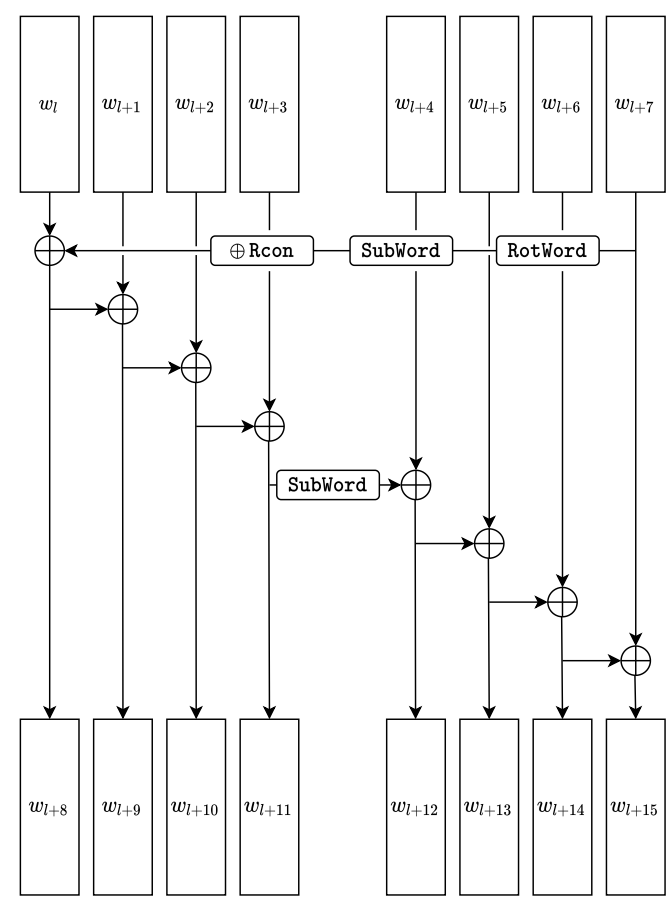
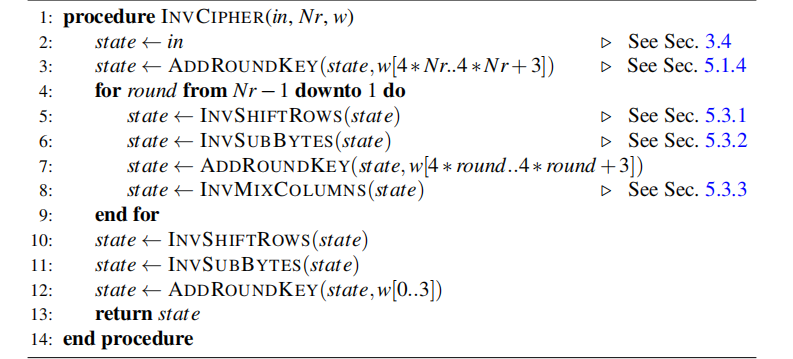


图 8 AES-256 的密钥扩展（KEYEXPANSION()），用于生成 8 ≤i< 60 的字 w[i]，其中l的取值范围是0到48之间8的倍数

**算法3 逆密码（INVCIPHER()）**



### 5.3.1 INVSHIFTROWS()

INVSHIFTROWS()是行移位变换（SHIFTROWS()）的逆变换。具体来说，状态矩阵最后三行中的字节按如下方式进行循环移位：sᵣ,ₑ = sᵣ,(c−r) mod 4，其中 0 ≤ r < 4且0 ≤ c < 4。（5.12）

图9展示了INVSHIFTROWS()的过程。在该状态矩阵的表示中，其作用是将每行中的每个字节向右移动r个位置，同时将最右侧的r个字节循环移至该行的左端。r=0的第一行保持不变。

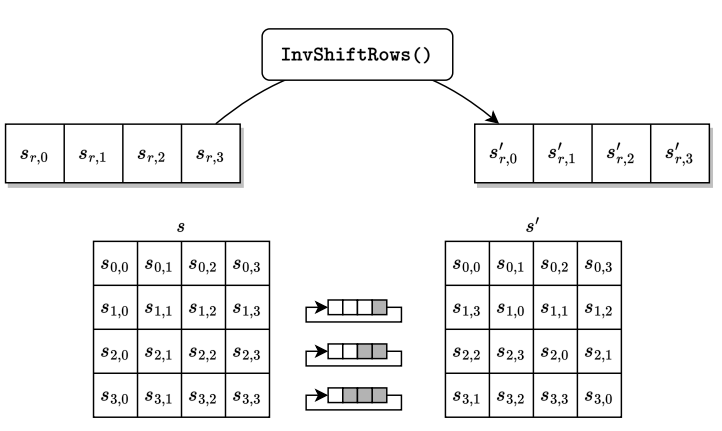
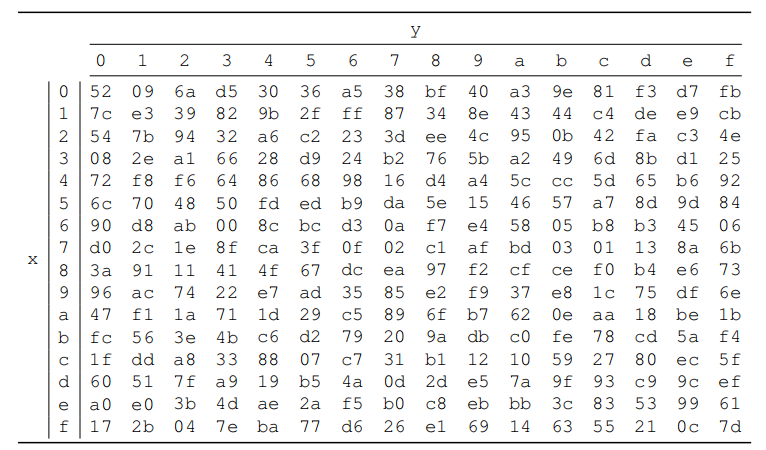


图 9 INVSHIFTROWS()示意图

### 5.3.2 INVSUBBYTES ()

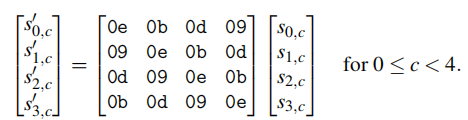
INVSUBBYTES()是字节替代变换（SUBBYTES()）的逆变换，在该变换中，S盒（SBOX()）的逆变换（记为INVSBOX()）会作用于状态矩阵的每个字节。INVSBOX()由表4导出，通过交换输入与输出的角色得到，如表6所示：

表 6 INVSBOX ()：针对字节xy的替代值（以十六进制格式）

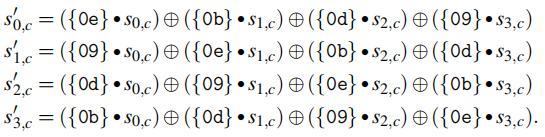


### 5.3.3 INVMIXCOLUMNS ()

INVMIXCOLUMNS()是列混合变换（MIXCOLUMNS()）的逆变换。具体而言，INVMIXCOLUMNS() 会将状态矩阵的四个列分别与一个固定矩阵相乘（如4.3节所述），该矩阵的元素来自以下字：[a0,a1,a2,a3] = [{0e},{09},{0d},{0b}] （5.13）因此，

（5.14）

通过该矩阵乘法，一列中的四个字节被替换为以下结果：

（5.15）

### 5.3.4 INVADDROUNDKEY ()

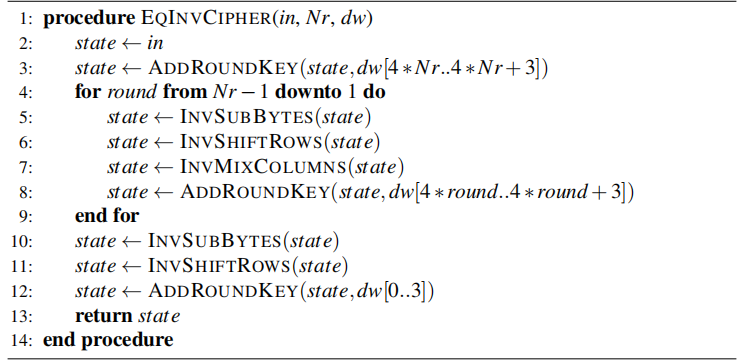
5.1.4节所述的ADDROUNDKEY ()的逆变换是其自身。

### 5.3.5 EQINVCIPHER ()

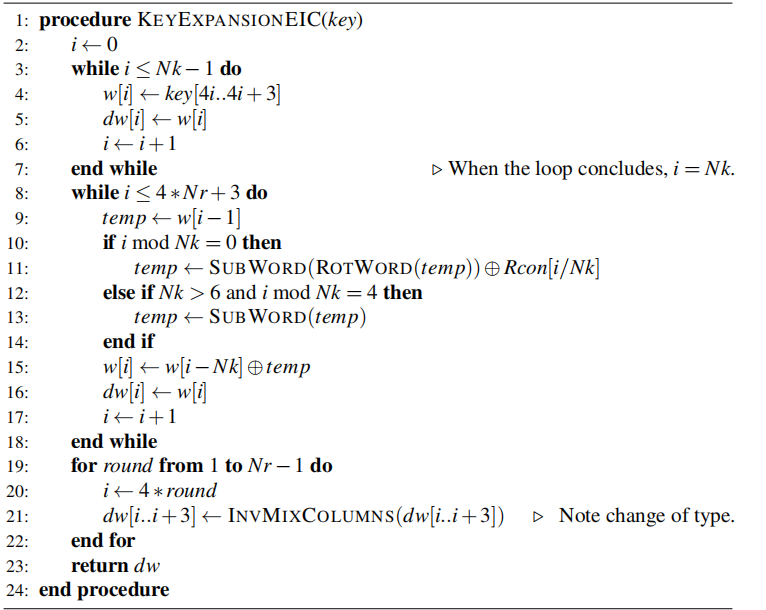
AES算法的若干特性使其可以采用另一种逆密码（CIPHER()的逆）的规范说明，称为等效逆密码，记为EQINVCIPHER()。在EQINVCIPHER()的规范中，算法1中密码轮函数的各个变换在EQINVCIPHER()中直接替换为其逆变换，且顺序保持不变。与算法3中INVCIPHER()的规范相比，这种结构的效率在Rijndael提案文档[2]中已有说明。

算法4给出的等效逆密码伪代码使用了一个经过修改的密钥编排（用字数组dw表示）。生成dw的程序是密钥扩展（KEYEXPANSION()）的扩展，记为 KEYEXPANSIONEIC()，其伪代码见算法5。

**算法4 等效逆密码（EQINVCIPHER ()）**



**算法5 等效逆密码KEYEXPANSIONEIC()**



dw 中的第一个和最后一个轮密钥与 w 中的相同；其他轮密钥的修改如第 19-22 行所述。第 21 行的注释涉及逆列混合变换（INVMIXCOLUMNS ()）的输入：一维字数组会转换为二维字节数组（如图 1 所示）。

# 6 实现考虑

## 6.1 密钥长度要求

AES算法的实现应至少支持第5节中规定的三种密钥长度之一：128位、192位或256位（即Nk=4、6或8）。实现可选择支持两种或三种密钥长度，这可能会促进算法实现的互操作性。

## 6.2 密钥限制

当加密密钥已适当生成时（参见NIST特别出版物800-133，修订2[6] 中的指南），使用生成的密钥进行AES算法操作时没有限制。

## 6.3 参数扩展

在表3中，本标准明确定义了密钥长度（Nk）、块大小（Nb）和轮数（Nr）的允许值。然而，本标准的未来修订可能会对这些参数的允许值进行更改或添加。因此，实现者可选择在设计其AES实现时考虑未来的灵活性。

## 6.4 关于各种平台的实现建议

可能存在多种实现变体，在许多情况下可能提供性能或其他优势。对于相同的输入密钥和数据（明文或密文），任何产生与本标准规定的算法相同输出（密文或明文）的实现都是 AES 算法的等效实现。

AES提案文件[2]和AES页面[7] 上的其他资源包含关于如何在各种平台上高效实现AES算法的建议。建议的实现旨在解释AES算法的内部工作原理，但不提供针对各种实现攻击的保护。

物理实现可能通过侧信道泄露密钥相关信息，例如执行计算所需的时间，或当故障注入计算时。当此类攻击是非侵入性的时，即使存在检测设备物理篡改的机制，它们也可能是有效的。例如，缓存定时攻击可能会影响在使用缓存加速从主内存访问数据的软件平台上的AES实现。

应考虑保护AES算法的实现在适用情况下免受实现攻击。此类考虑超出了本文档的范围，但在根据NIST开发的验证程序（参见https://nist.gov/cmvp）测试是否符合本标准中的算法时会予以考虑。

## 6.5 操作模式

分组密码操作模式是使用分组密码提供信息服务（如保密性和认证）的密码函数。NIST推荐的操作模式在NIST特别出版物800-38系列中规定。更多信息可在https://csrc.nist.gov/Projects/block-cipher-techniques/BCM获取。

# 7 其他

## 参考文献

1. James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, and Edward Roback. Report on the Development of the Advanced Encryption Standard (AES). Journal of Research of NIST (NIST JRES), May 2001. https://doi.org/10.6028/jres.106.023.
2. Joan Daemen and Vincent Rijmen. AES Proposal: Rijndael Document Version 2. AES Algorithm Submission, September 1999. Available at https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf.
3. Joan Daemen and Vincent Rijmen. The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition. Information Security and Cryptography. Springer, 2020. https://doi.org/10.1007/978-3-662-60769-5.
4. Michael Artin. Algebra. Pearson Modern Classic. Pearson, second edition, 2017.
5. Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. Handbook of Applied Cryptography. CRC Press, Inc., USA, 1st edition, 1997. https://doi.org/10.1201/9780429466335.
6. Elaine Barker, Allen Roginsky, and Richard Davis. Recommendation for Cryptographic Key Generation. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-133, Rev. 2, June 2020. https://doi.org/10.6028/NIST.SP.800-133r2.
7. National Institute of Standards and Technology. AES Development, 2022. Available at https://csrc.nist.gov/projects/aes.
8. National Institute of Standards and Technology. Cryptographic Standards and Guidelines: Examples with Intermediate Values, 2022. Available at https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/example-values.
9. National Institute of Standards and Technology. Crypto Publications Review Board, 2022. Available at https://csrc.nist.gov/projects/crypto-publication-review-project.
10. Nicky Mouha. Review of the Advanced Encryption Standard. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency Report (IR) 8319. https://doi.org/10.6028/NIST.IR.8319.

## 附录

参见本文件目录下的NIST-AES-Appendix.md文件

**本译文归属 github 用户 WhatTheFuck-cyber**