现代密码学理论与实践

第5章高级数据加密标准AES

高级加密标准AES要点

- AES是一种分组密码，用以取代DES的商业应用。其分组长度为128位，密钥长度为128位、192位或256位。
- AES没有使用Feistel结构，每轮由四个独立的运算组成：字节代换、置换、有限域上的算术运算，以及与密钥的异或运算。

AES的评估准则

- AES的起源
  - 因为DES的不安全，建议用3DES，密钥168位，抵御密码分析攻击。但是3DES用软件实现速度较慢，分组短，仅64位。
  - 美国国家标准技术协会NIST在1997年征集新标准，要求分组128位，密钥128、192或256位。
  - 15种候选算法在1998年6月通过了第一轮评估，仅有5个候选算法在1999年8月通过了第二轮评估。
  - 2000年10月，NIST选择Rijndael作为AES算法，Rijndael的作者是比利时的密码学家Joan Daemen博士和Vincent Rijmen博士。
  - 2001年11月，NIST完成评估并发布了最终标准FIPS PUB 197。

AES的评估

- AES评估准则的三大类别
  - 安全性：指密码分析方法分析一个算法所需的代价
  - 成本：期望AES能够广泛应用于各种实际应用，计算效率要高
  - 算法和执行特征：算法灵活性、适合于多种硬件和软件方式的实现、简洁性，便于分析安全性

Table 5.1 NIST Evaluation Criteria for AES (September 12, 1997) (page 1 of 2)

SECURITY

- **Actual security:** compared to other submitted algorithms (at the same key and block size).
- **Randomness:** The extent to which the algorithm output is indistinguishable from a random permutation on the input block.
- **Soundness:** of the mathematical basis for the algorithm's security.
- **Other security factors:** raised by the public during the evaluation process, including any attacks which demonstrate that the actual security of the algorithm is less than the strength claimed by the submitter.

COST

- **Licensing requirements:** NIST intends that when the AES is issued, the algorithm(s) specified in the AES shall be available on a worldwide, non-exclusive, royalty-free basis.
- **Computational efficiency:** The evaluation of computational efficiency will be applicable to both hardware and software implementations. Round 1 analysis by NIST will focus primarily on software implementations and specifically on one key-block size combination (128-128); more attention will be paid to hardware implementations and other supported key-block size combinations during Round 2 analysis. Computational efficiency essentially refers to the speed of the algorithm. Public comments on each algorithm's efficiency (particularly for various platforms and applications) will also be taken into consideration by NIST.
- **Memory requirements:** The memory required to implement a candidate algorithm--for both hardware and software implementations of the algorithm--will also be considered during the evaluation process. Round 1 analysis by NIST will focus primarily on software implementations; more attention will be paid to hardware implementations during Round 2. Memory requirements will include such factors as gate counts for hardware implementations, and code size and RAM requirements for software implementations.

Table 5.1 NIST Evaluation Criteria for AES (September 12, 1997) (page 2 of 2)

ALGORITHM AND IMPLEMENTATION CHARACTERISTICS

- **Flexibility:** Candidate algorithms with greater flexibility will meet the needs of more users than less flexible ones, and therefore, inter alia, are preferable. However, some extremes of functionality are of little practical application (e.g., extremely short key lengths); for those cases, preference will not be given. Some examples of flexibility may include (but are not limited to) the following:
  - **a.** The algorithm can accommodate additional key- and block-sizes (e.g., 64-bit block sizes, key sizes other than those specified in the Minimum Acceptability Requirements section, [e.g., keys between 128 and 256 that are multiples of 32 bits, etc.])
  - **b.** The algorithm can be implemented securely and efficiently in a wide variety of platforms and applications (e.g., 8-bit processors, ATM networks, voice & satellite communications, HDTV, B-ISDN, etc.).
  - **c.** The algorithm can be implemented as a stream cipher, message authentication code (MAC) generator, pseudorandom number generator, hashing algorithm, etc.
- **Hardware and software suitability:** A candidate algorithm shall not be restrictive in the sense that it can only be implemented in hardware. If one can also implement the algorithm efficiently in firmware, then this will be an advantage in the area of flexibility.
- **Simplicity:** A candidate algorithm shall be judged according to relative simplicity of design.

AES评估准则

- 一般安全性
  - 依赖于密码学界的公共安全分析
- 软件实现
  - 软件执行速度，跨平台执行能力及密钥长度改变时速度变化
- 受限空间环境
  - 在诸如智能卡中的应用
- 硬件实现
  - 硬件执行提高执行速度或缩短代码长度
- 对执行的攻击
  - 抵御密码分析攻击
- 加密与解密
- 密钥灵活性
  - 快速改变密钥长度的能力
- 其他的多功能性和灵活性
- 指令级并行执行的潜力

Table 5.2 Final NIST Evaluation of Rijndael (October 2, 2000) (page 1 of 2)

General Security

Rijndael has no known security attacks. Rijndael uses S-boxes as nonlinear components. Rijndael appears to have an adequate security margin, but has received some criticism suggesting that its mathematical structure may lead to attacks. On the other hand, the simple structure may have facilitated its security analysis during the timeframe of the AES development process.

Software Implementations

Rijndael performs encryption and decryption very well across a variety of platforms, including 8-bit and 64-bit platforms, and DSPs. However, there is a decrease in performance with the higher key sizes because of the increased number of rounds that are performed. Rijndael's high inherent parallelism facilitates the efficient use of processor resources, resulting in very good software performance even when implemented in a mode not capable of interleaving. Rijndael's key setup time is fast.

Restricted-Space Environments

In general, Rijndael is very well suited for restricted-space environments where either encryption or decryption is implemented (but not both). It has very low RAM and ROM requirements. A drawback is that ROM requirements will increase if both encryption and decryption are implemented simultaneously, although it appears to remain suitable for these environments. The key schedule for decryption is separate from encryption.

Hardware Implementations

Rijndael has the highest throughput of any of the finalists for feedback modes and second highest for non-feedback modes. For the 192 and 256-bit key sizes, throughput falls in standard and unrolled implementations because of the additional number of rounds. For fully pipelined implementations, the area requirement increases, but the throughput is unaffected.

Table 5.2 Final NIST Evaluation of Rijndael (October 2, 2000) (page 2 of 2)

Attacks on Implementations

The operations used by Rijndael are among the easiest to defend against power and timing attacks. The use of masking techniques to provide Rijndael with some defense against these attacks does not cause significant performance degradation relative to the other finalists, and its RAM requirement remains reasonable. Rijndael appears to gain a major speed advantage over its competitors when such protections are considered.

Encryption vs. Decryption

The encryption and decryption functions in Rijndael differ. One FPGA study reports that the implementation of both encryption and decryption takes about 60% more space than the implementation of encryption alone. Rijndael's speed does not vary significantly between encryption and decryption, although the key setup performance is slower for decryption than for encryption.

Key Agility

Rijndael supports on-the-fly subkey computation for encryption. Rijndael requires a one-time execution of the key schedule to generate all subkeys prior to the first decryption with a specific key. This places a slight resource burden on the key agility of Rijndael.

Other Versatility and Flexibility

Rijndael fully supports block sizes and key sizes of 128 bits, 192 bits and 256 bits, in any combination. In principle, the Rijndael structure can accommodate any block sizes and key sizes that are multiples of 32, as well as changes in the number of rounds that are specified.

Potential for Instruction-Level Parallelism

Rijndael has an excellent potential for parallelism for a single block encryption.

NIST对AES的要求

- 对称密钥分组密码
- 128位分组，密钥长度可以分别是128/192/256位
- 要求比Triple-DES更安全和更快
- 至少能够安全工作20-30年
- 提供完整的规范说明和设计细节
- 能够用C或Java实现
- NIST公布了所有提交的算法和不保密的分析资料，最终通过评估，选择了Rijndael

AES密码

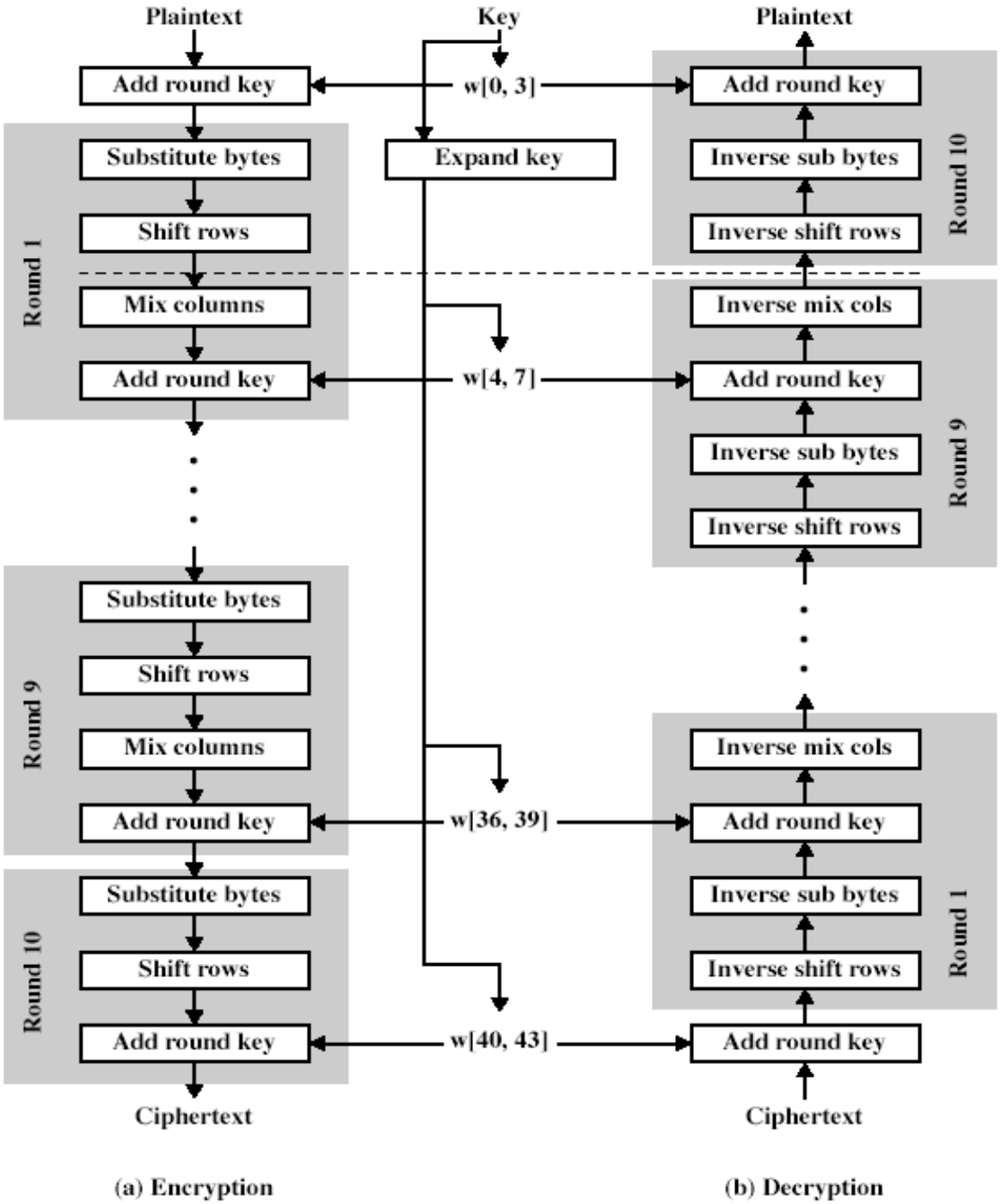
- AES的分组长度为128位，密钥长度可以是128/192/256的任意一种
- 未采用Feistel密码结构而是用迭代方式
  - 数据分成4组，每组4字节
  - 每一轮对整个分组进行操作
- Rijndael具有如下特性
  - 对所有已知的攻击具有免疫性
  - 在各种CPU平台上其执行速度快且代码紧凑
  - 设计简单

AES的参数

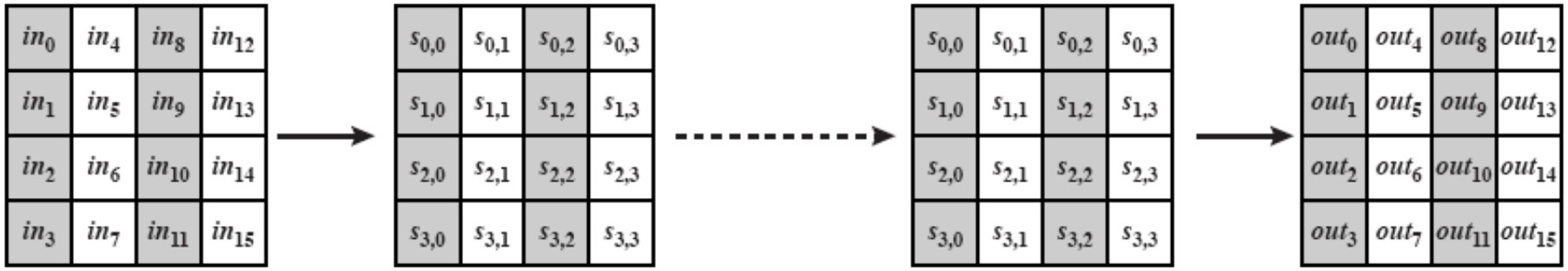
Table 5.3 AES Parameters

Key size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

AES的加密与解密


Plaintext
Key
Plaintext
Add round key
w[0, 3]
Add round key
Substitute bytes
Expand key
Inverse sub bytes
Shift rows
Inverse shift rows
Round 1
Round 10
Mix columns
Inverse mix cols
Add round key
w[4, 7]
Add round key
Inverse sub bytes
Round 9
Round 9
Inverse shift rows
Substitute bytes
Shift rows
Inverse mix cols
Mix columns
Add round key
w[36, 39]
Add round key
Substitute bytes
Inverse sub bytes
Round 10
Round 1
Shift rows
Inverse shift rows
Add round key
w[40, 43]
Add round key
Ciphertext
Ciphertext
(a) Encryption
(b) Decryption

AES的数据结构



(a) Input, state array, and output

(b) Key and expanded key

Figure 5.2 AES Data Structures

AES的结构

- 非Feistel结构
- 密钥被扩展成由44个32位字组成的数组w[i]
- 一个混淆和三个代换
  - 字节代换：用一个S盒完成分组中的按字节代换
  - 行移位：一个简单的置换
  - 列混淆：利用在域GF(2^8)上的算术特性的代换
  - 轮密钥加：利用当前分组和扩展密钥的一部分进行按位XOR
- 算法结构非常简单
- 仅在轮密钥加密阶段中使用密钥
- 轮密钥加配合其他三个混淆的交替使用提供了安全性
- 每个阶段均可逆
- 解密按逆序方式使用扩展密钥，但是算法加密不一样
- 一旦将四个阶段求逆，可以证明解密函数可以恢复明文
- 加密和解密过程的最后一轮均只包括三个阶段

Rijndael密码

- 128位数据分成4组，每组4字节(state)
- 每个state执行9/11/13轮操作:
  - 字节代换(每个字节使用一个S盒)
  - 行移位变换(在组和列之间变换)
  - 列混淆变换(使用组的矩阵乘)
  - 轮密钥加(用密钥异或state)
- 所有运算均可通过异或和查表来完成，因此非常快而且效率高。

AES的一轮加密过程

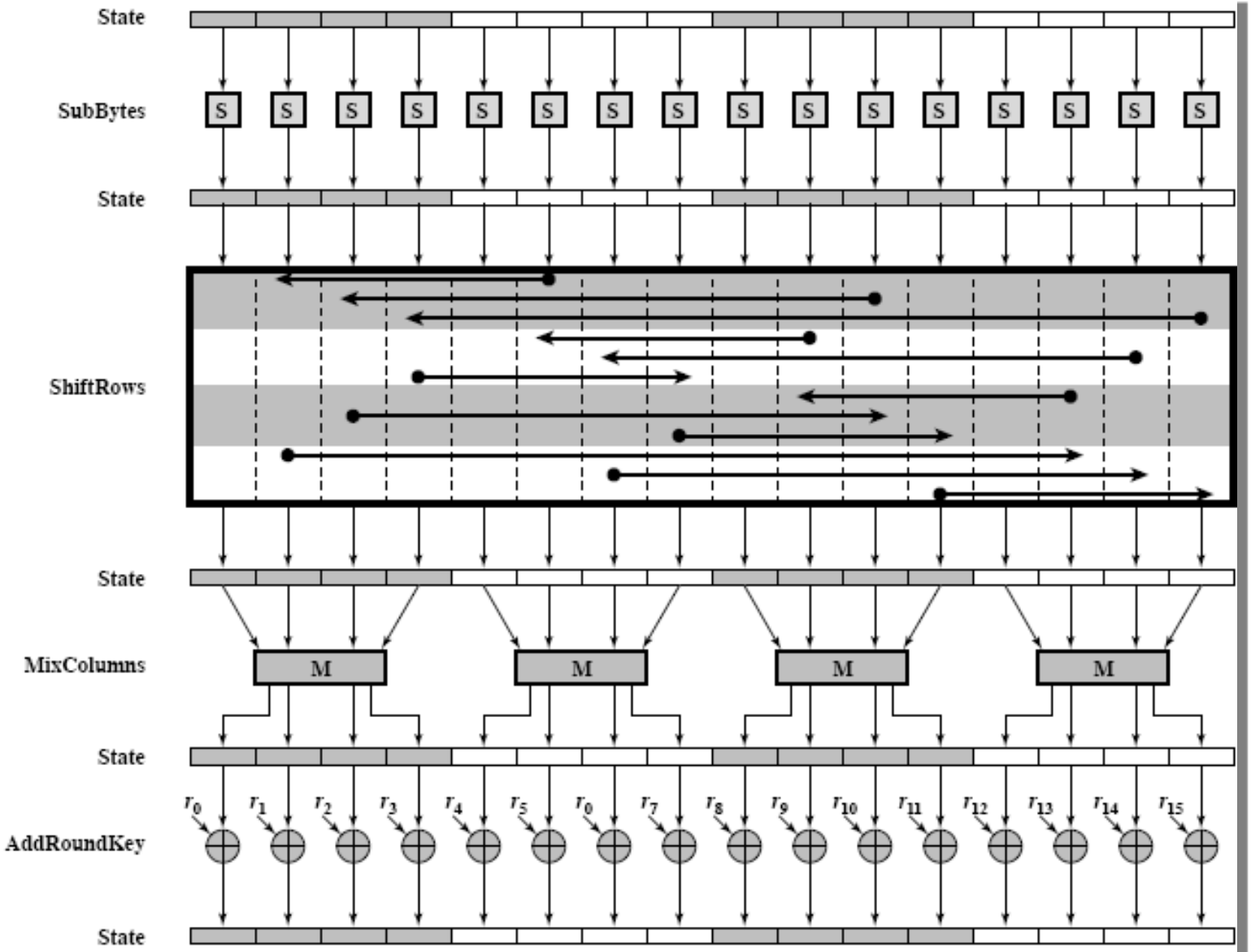
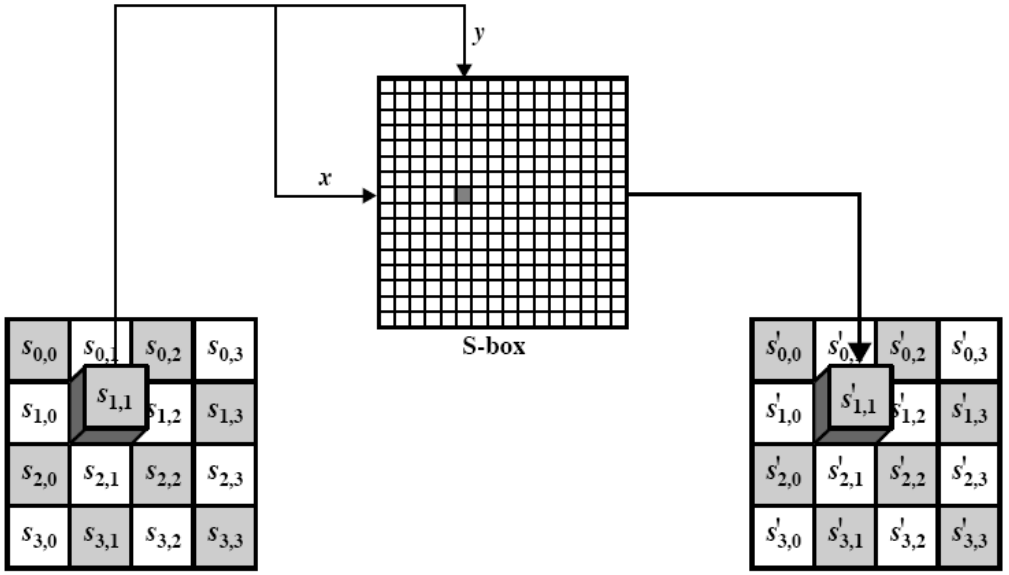


Figure 5.3 AES Encryption Round

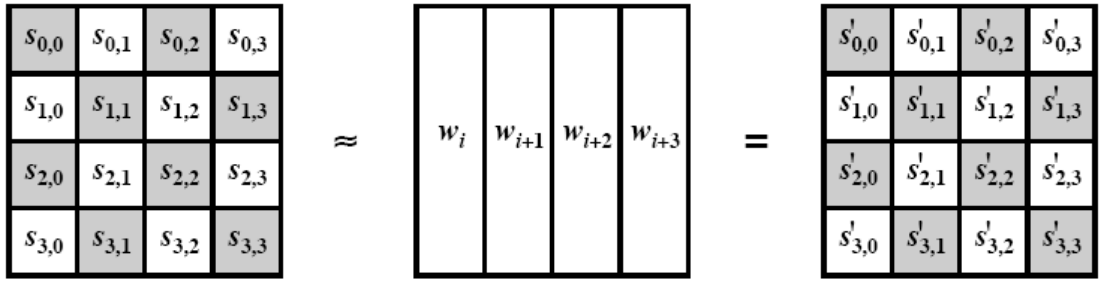
字节代替变换

- 对每个字节做一简单的替换
- 使用一个16x16字节的表，其中包含了所有256种8位的置换值
- 每个字节的左4位选择行，右4位选择列来查表替换
  - 比如字节{95}由第9行第5列所在字节替换
  - 也就是字节{2A}
- S盒使用GF(2^8)中定义的所有转换值来构建
- 能够抵御所有已知攻击

AES的字节层操作
字节代替变换和轮密钥加变换



(a) Substitute byte transformation



(b) Add Round Key Transformation

Figure 5.4 AES Byte-Level Operations

Table 5.4 AES S-Boxes

(a) S-box

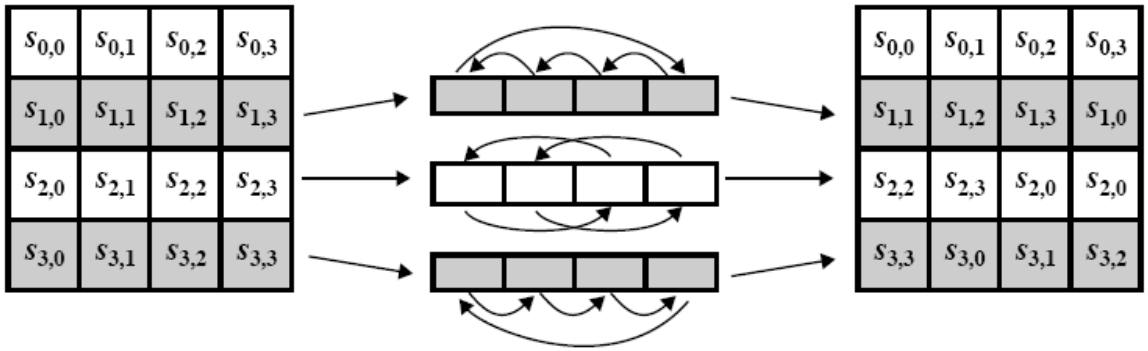
		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(b) Inverse S-box

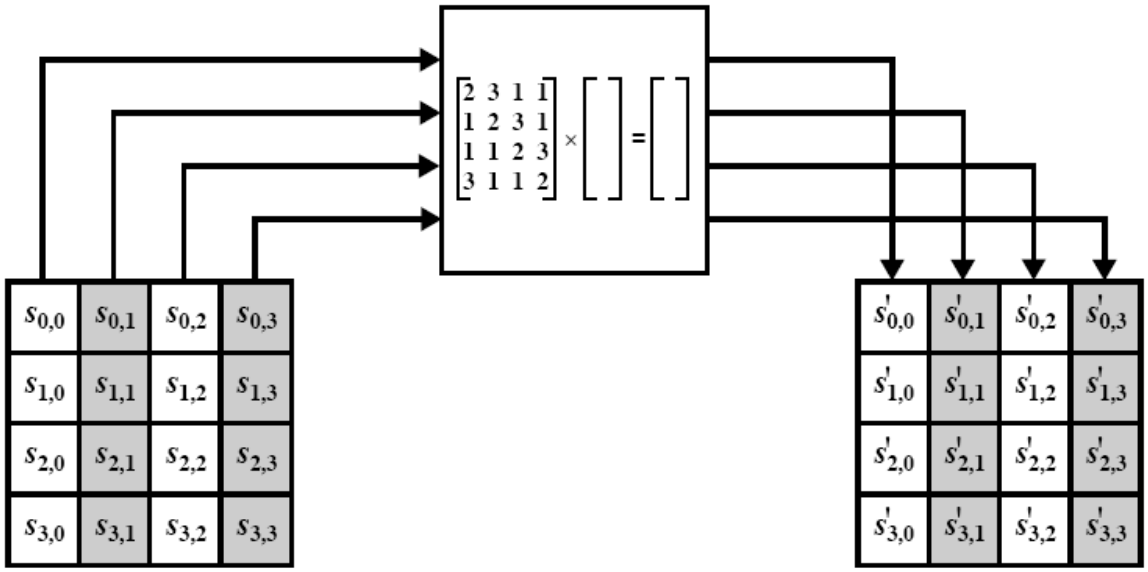
		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

行移位变换

- 正向行移位变换
  - State的第1行保持不变
  - State的第2行循环左移1个字节
  - State的第3行循环左移2个字节
  - State的第4行循环左移3个字节
- 逆向行移位变换进行相反的移位以实现解密
- 因为State是按列处理的，行移位变换就把字节在列之间进行了置换



(a) Shift row transformation



(b) Mix column transformation

Figure 5.5 AES Row and Column Operations

列混淆变换

- 正向列混淆变换对每列独立进行操作
- 每列中的每个字节被映射为一个新值，由该列中的4个字节通过函数变换得到
- 这个变换由以下基于State的矩阵乘法表示，使用素多项式$m(x) = x^8+x^4+x^3+x+1$，$GF(2^8)$
- 逆向列混淆变换也由矩阵乘法定义

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

轮密钥加变换

- 正向轮密钥加变换中128位的State按位与128位的秘密钥异或(XOR)
- 都是基于State列的操作，即把State的一列中的4个字节与轮密钥的1个字进行异或
- 逆向轮密钥加变换与正向轮密钥加变换相同，因为异或操作是其本身的逆
- 轮密钥加变换非常简单，却能影响State中的每一位

AES的密钥扩展

- AES密钥扩展算法的输入是4字(16字节)，输出是44字(或者52字, 或60字)，每字32位
- 输入密钥直接被复制到扩展密钥数组的前4个字
- 然后每次用4个字填充扩展密钥数组余下的部分
- 这样的设计能够抵御已知攻击

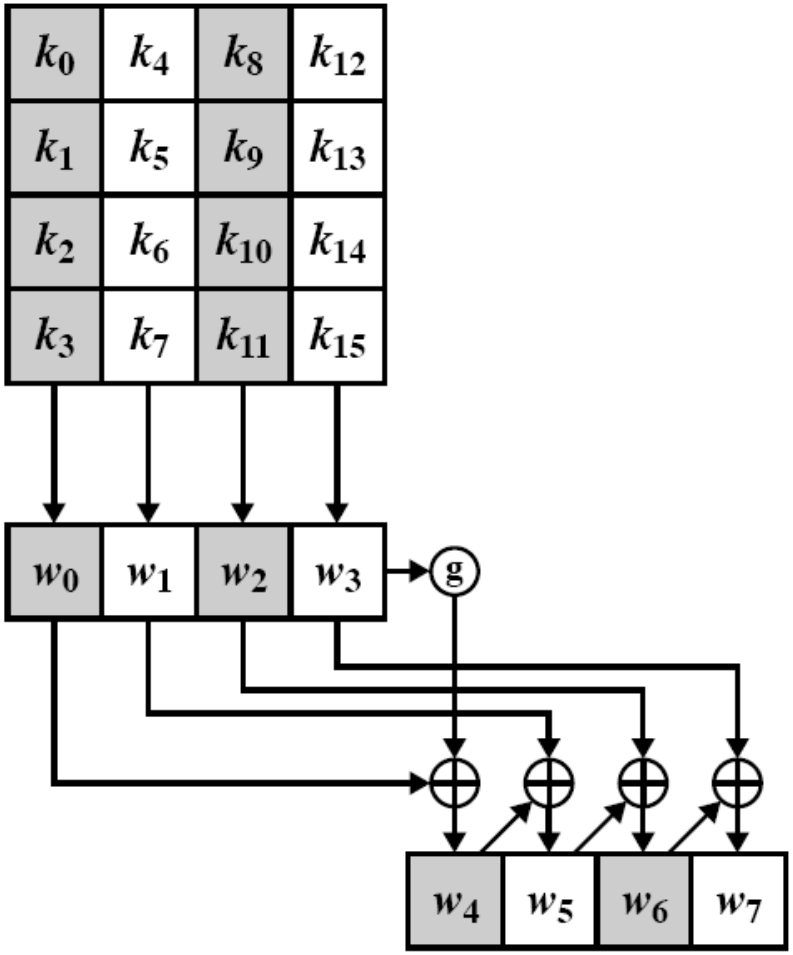
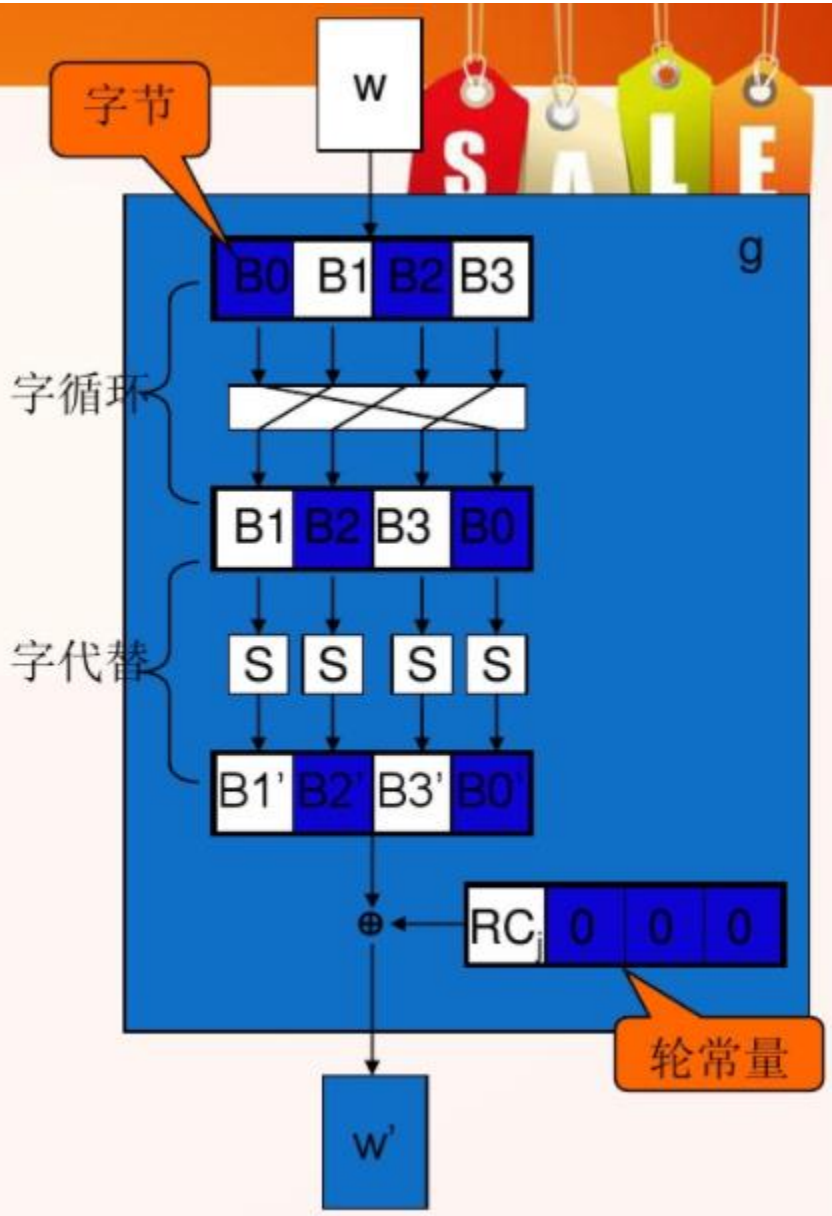


Figure 5.6 AES Key Expansion

g函数：

- 1、字循环的功能是使一个字中的4个字节循环左移一个字节，即将输入字$[B_0,B_1,B_2,B_3]$变换成$[B_1,B_2,B_3,B_0]$。
- 2、字代替利用S盒对输入字中的每个字节进行字节代替。
- 3、将结果与轮常量Rcon[j]相异或。



轮常量：

- 轮常量是一个字，这个字的最右边三个字节总为0。因此字与Rcon相异或，其结果只是与该字最左边那个字节相异或。
- 每轮的轮常量均不同，其定义为Rcon[j]=(RC[j],0,0,0),其中RC[1]=1,RC[j]=2 · RC[j-1]。
- RC[j]的值按十六进制表示为：

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

对应的逆算法

- AES的解密算法和加密算法不同，主要是解密中变换的顺序与加密中变换的顺序不同
- 但是可以定义一个等价的逆密码算法，采用与加密算法一样的步骤
  - 每一步都是逆向进行的
  - 密钥的安排不同于加密

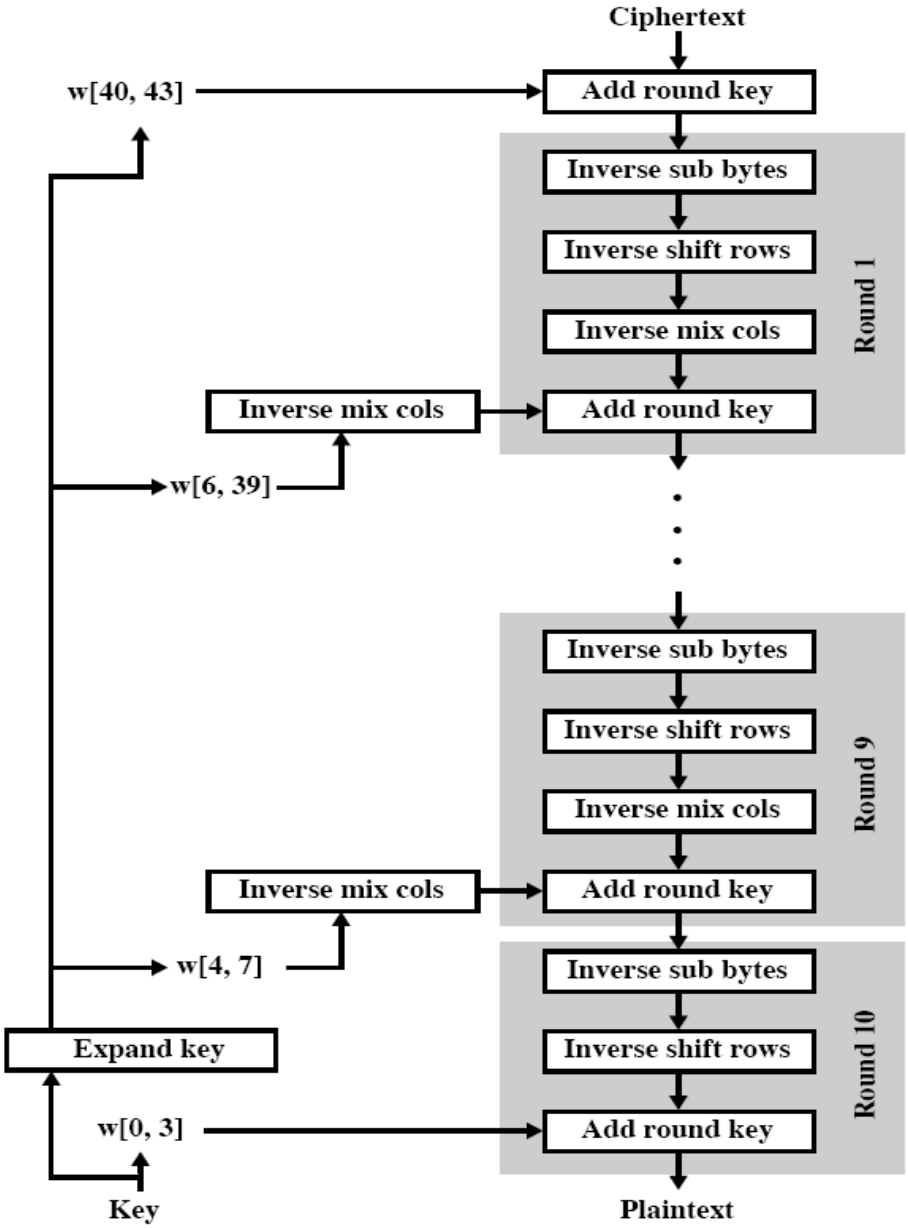


Figure 5.7 Equivalent Inverse Cipher

AES的实现

- 可以在8位处理器上非常有效地实现
  - 字节代换是在字节级别上进行操作的，只要求一个256字节的表
  - 行移位是简单的移字节操作
  - 轮密钥加是按位异或操作
  - 列混淆变换要求在域GF(2^8)上的乘法，所有的操作都是基于字节的，只要简单地查表即可
- 可以在32位处理器上非常有效地实现
  - 将操作定义在32位的字上
  - 事先准备好4张256字的表
  - 在每一轮中通过查表并加上4次异或即可计算每一列的值
  - 需要16Kb空间来存储这4张表

Summary

- We have considered:
  - the AES selection process
  - the details of Rijndael – the AES cipher
  - looked at the steps in each round
  - the key expansion
  - implementation aspects