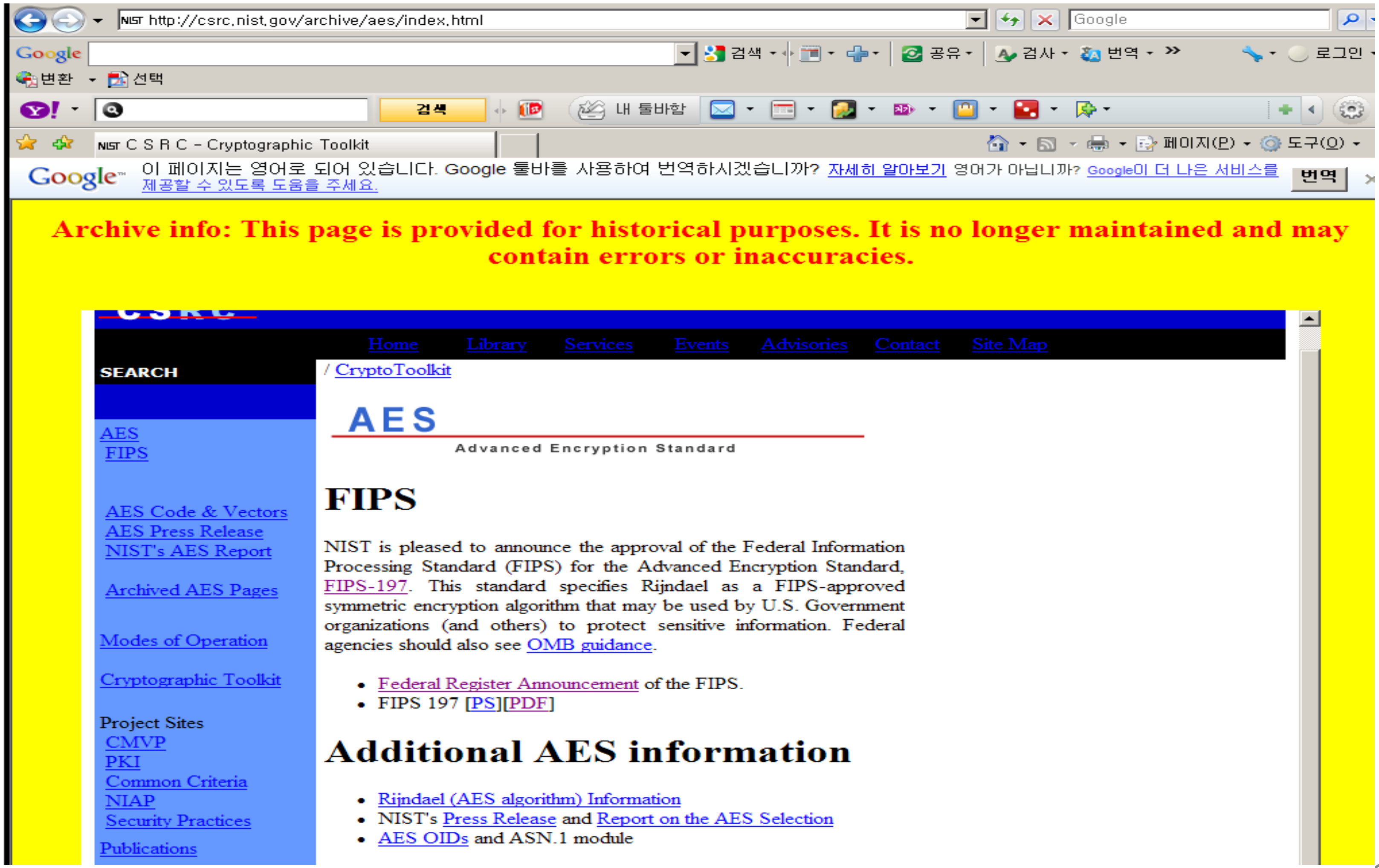
Week 5: Advanced Encryption Standard

Click http://www.nist.gov/aes


http://csrc.nist.gov/archive/aes/index.html
CSRC - Cryptographic Toolkit
이 페이지는 영어로 되어 있습니다. Google 툴바를 사용하여 번역하시겠습니까? 자세히 알아보기 영어가 아닙니까? Google이 더 나은 서비스를 제공할 수 있도록 도움을 주세요.
번역
Home Library Services Events Advisories Contact Site Map
SEARCH
AES
FIPS
AES Code & Vectors
AES Press Release
NIST's AES Report
Archived AES Pages
Modes of Operation
Cryptographic Toolkit
Project Sites
CMVP
PKI
Common Criteria
NIAP
Security Practices
Publications

Archive info: This page is provided for historical purposes. It is no longer maintained and may contain errors or inaccuracies.

/ CryptoToolkit

AES
Advanced Encryption Standard

FIPS

NIST is pleased to announce the approval of the Federal Information Processing Standard (FIPS) for the Advanced Encryption Standard, FIPS-197. This standard specifies Rijndael as a FIPS-approved symmetric encryption algorithm that may be used by U.S. Government organizations (and others) to protect sensitive information. Federal agencies should also see OMB guidance.

- Federal Register Announcement of the FIPS.
- FIPS 197 [PS][PDF]

Additional AES information

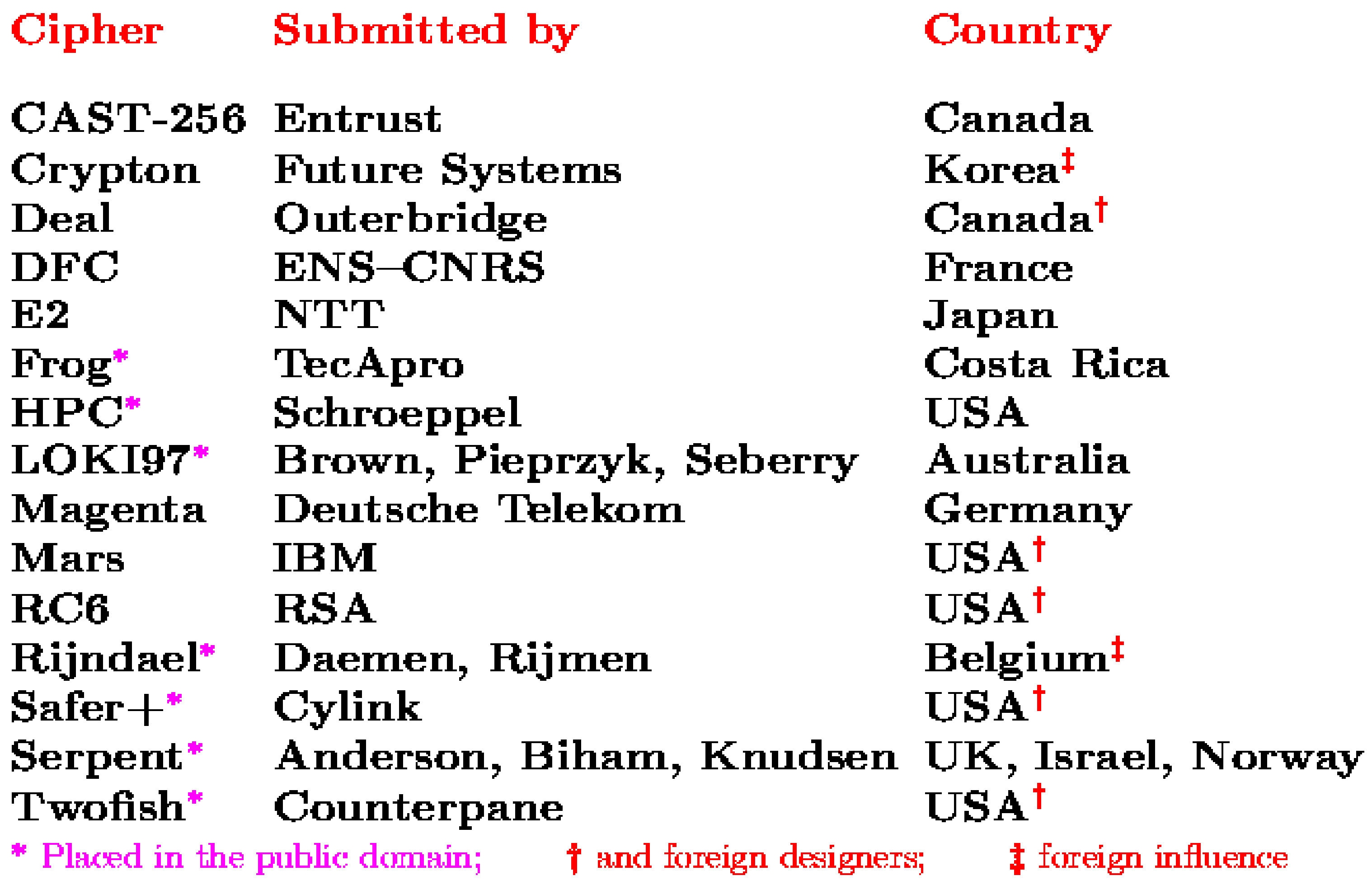
- Rijndael (AES algorithm) Information
- NIST's Press Release and Report on the AES Selection
- AES OIDs and ASN.1 module

History of AES

- Calendar
  - 1997 : Call For AES Candidate Algorithms by NIST
    - 128-bit Block cipher
    - 128/192/256-bit keys
    - Worldwide-royalty free
    - More secure than Triple DES
    - More efficient than Triple DES
  - 1998 : 1st Round Candidates – 15 Algorithms
    - Mars, Twofish, RC6, SAFER+, HPC, CAST256, DEAL, Frog, Magenta, Rijndael, DFC, Serpent, Crypton, E2, LOKI97
  - 1999 : 2nd Round Candidates – 5 Algorithms
    - MARS, RC6, Rijndael, Serpent, and Twofish
  - 2000. 10 : Rijndael selected as the finalist
  - 2001. 12: official publication as FIPS PUB 197

* NIST: National Institute of Standards and Technology

AES Contest – 1st Round Candidates



Cipher	Submitted by	Country
CAST-256	Entrust	Canada
Crypton	Future Systems	Korea‡
Deal	Outerbridge	Canada†
DFC	ENS–CNRS	France
E2	NTT	Japan
Frog*	TecApro	Costa Rica
HPC*	Schroeppel	USA
LOKI97*	Brown, Pieprzyk, Seberry	Australia
Magenta	Deutsche Telekom	Germany
Mars	IBM	USA†
RC6	RSA	USA†
Rijndael*	Daemen, Rijmen	Belgium‡
Safer+*	Cylink	USA†
Serpent*	Anderson, Biham, Knudsen	UK, Israel, Norway
Twofish*	Counterpane	USA†

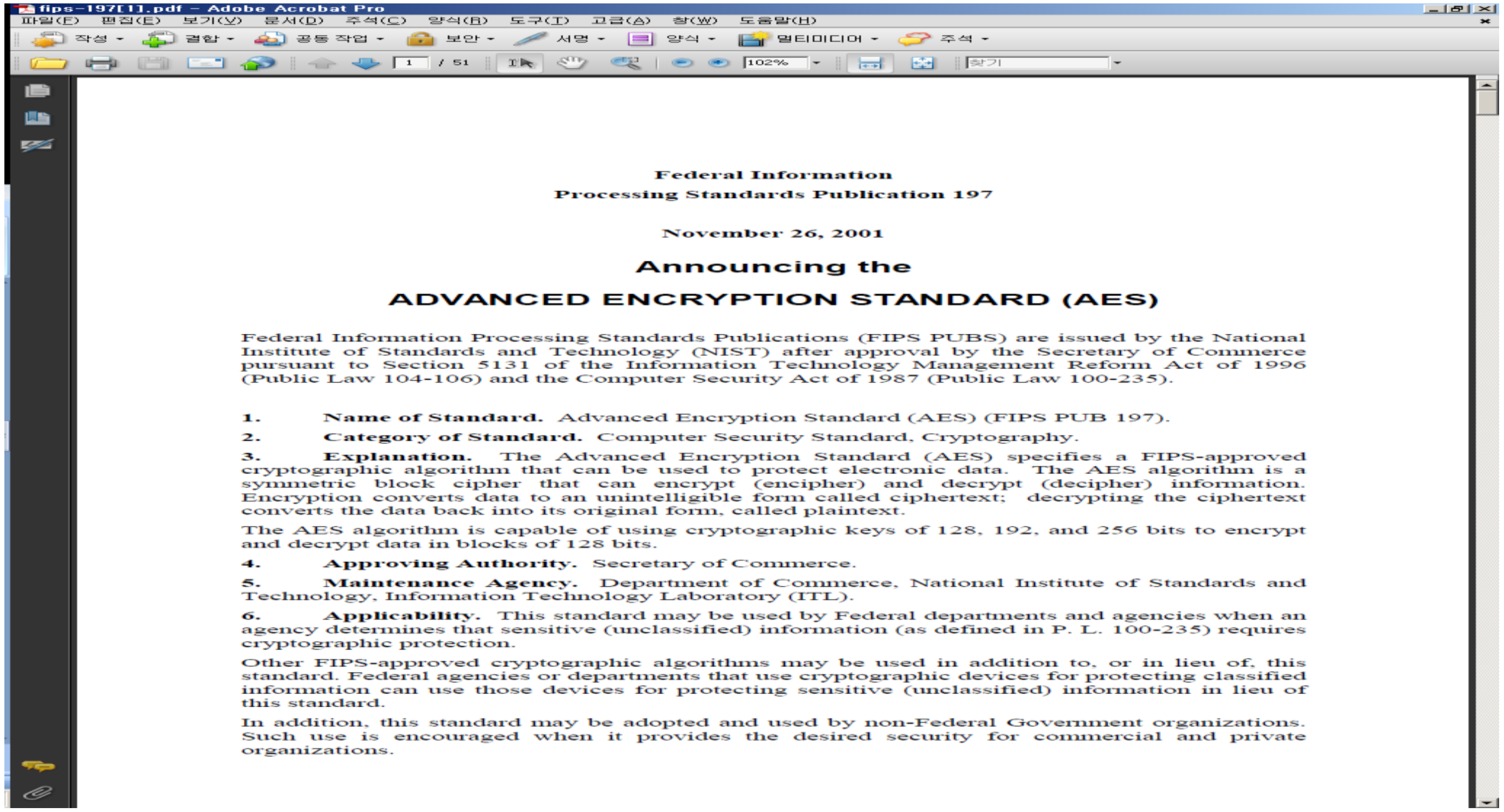
* Placed in the public domain; † and foreign designers; ‡ foreign influence

AES Contest – 2nd Round Candidates

Cipher	Submitter	Structure	Nonlinear Component
MARS	IBM	Feistel structure	Sbox DD-Rotation
RC6	RSA Lab.	Feistel structure	Rotation
Rijndael	Daemen, Rijmen	SPN structure	Sbox
Serpent	Anderson, Biham, Knudsen	SPN structure	Sbox
Twofish	Schneier et. al	Feistel structure	Sbox

AES Contest – Finalist (1/2)

- 2000. 10 : Rijndael selected as the finalist
- 2001. 12: official publication as FIPS PUB 197


fips-197[1].pdf - Adobe Acrobat Pro
파일(F) 편집(E) 보기(V) 문서(D) 주석(C) 양식(R) 도구(T) 고급(A) 창(W) 도움말(H)
작성 결합 공동 작업 보안 서명 양식 멀티미디어 주석
1 / 51
102%
찾기

Federal Information
Processing Standards Publication 197

November 26, 2001

Announcing the

ADVANCED ENCRYPTION STANDARD (AES)

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104-106) and the Computer Security Act of 1987 (Public Law 100-235).

1. **Name of Standard.** Advanced Encryption Standard (AES) (FIPS PUB 197).

2. **Category of Standard.** Computer Security Standard, Cryptography.

3. **Explanation.** The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext.

The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

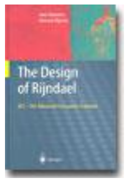
4. **Approving Authority.** Secretary of Commerce.

5. **Maintenance Agency.** Department of Commerce, National Institute of Standards and Technology, Information Technology Laboratory (ITL).

6. **Applicability.** This standard may be used by Federal departments and agencies when an agency determines that sensitive (unclassified) information (as defined in P. L. 100-235) requires cryptographic protection.

Other FIPS-approved cryptographic algorithms may be used in addition to, or in lieu of, this standard. Federal agencies or departments that use cryptographic devices for protecting classified information can use those devices for protecting sensitive (unclassified) information in lieu of this standard.

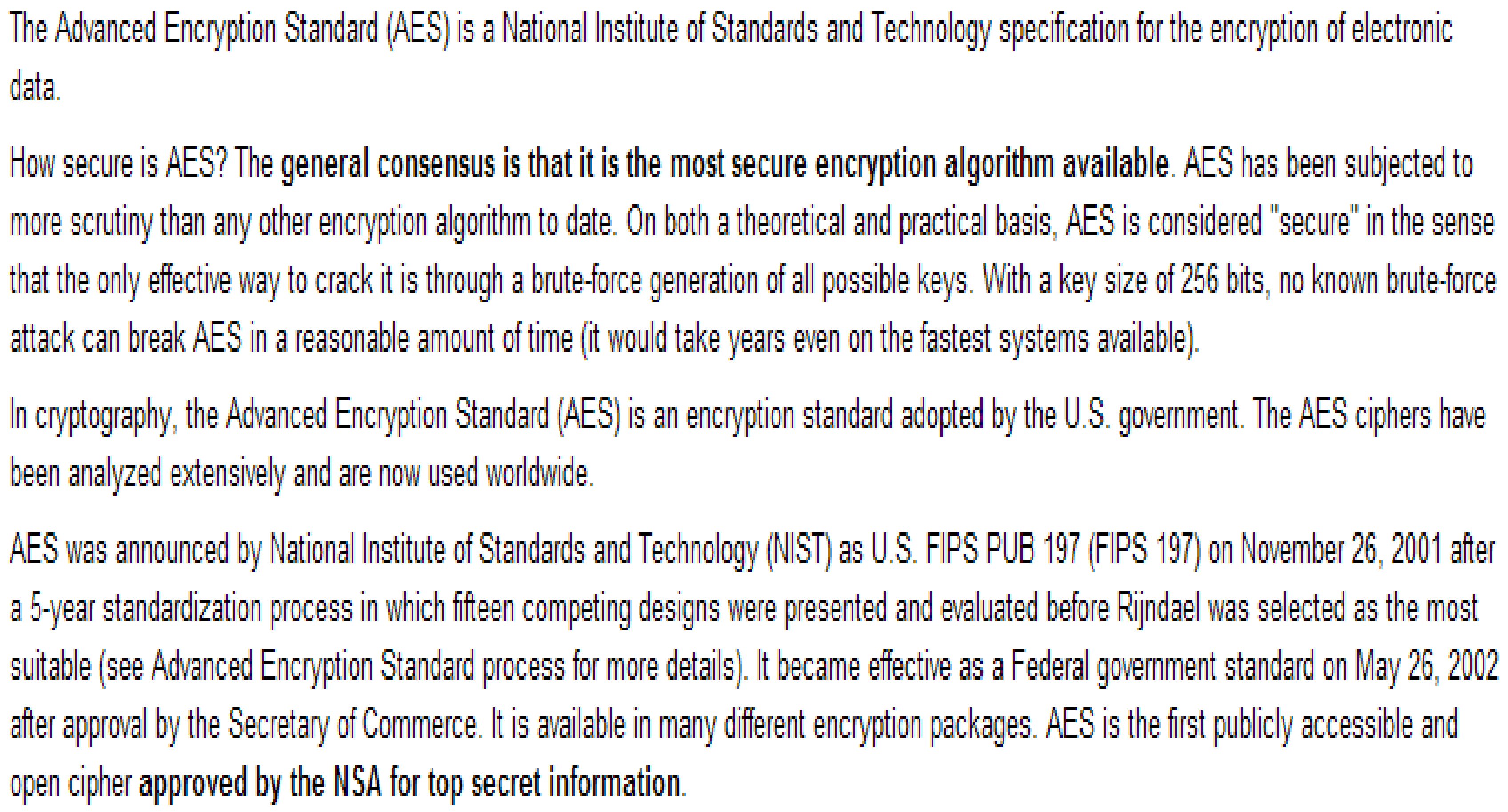
In addition, this standard may be adopted and used by non-Federal Government organizations. Such use is encouraged when it provides the desired security for commercial and private organizations.



Joan Daemen and Vincent Rijmen, "The Design of Rijndael, AES – The Advanced Encryption Standard", Springer, 2002, ISBN 3-540-42580-2

Vincent Rijmen

AES Contest – Finalist (2/2)



The Advanced Encryption Standard (AES) is a National Institute of Standards and Technology specification for the encryption of electronic data.

How secure is AES? The **general consensus is that it is the most secure encryption algorithm available**. AES has been subjected to more scrutiny than any other encryption algorithm to date. On both a theoretical and practical basis, AES is considered "secure" in the sense that the only effective way to crack it is through a brute-force generation of all possible keys. With a key size of 256 bits, no known brute-force attack can break AES in a reasonable amount of time (it would take years even on the fastest systems available).

In cryptography, the Advanced Encryption Standard (AES) is an encryption standard adopted by the U.S. government. The AES ciphers have been analyzed extensively and are now used worldwide.

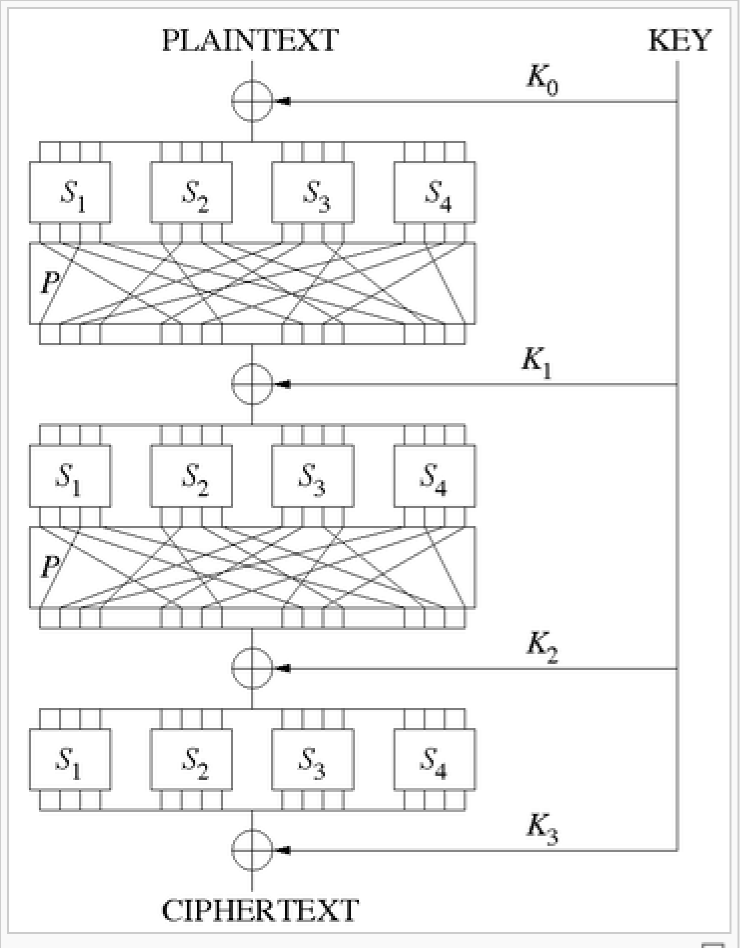
AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a 5-year standardization process in which fifteen competing designs were presented and evaluated before Rijndael was selected as the most suitable (see Advanced Encryption Standard process for more details). It became effective as a Federal government standard on May 26, 2002 after approval by the Secretary of Commerce. It is available in many different encryption packages. AES is the first publicly accessible and open cipher **approved by the NSA for top secret information**.

Security Goals of Rijndael

- K-secure
  - No shortcut attacks key-recover attack faster than key-exhaustive search
  - No symmetry property such as complementary in DES
  - No non-negligible classes of weak key as in IDEA
  - No Related-key attacks
- Hermetic
  - No weakness found for the majority of block ciphers with same block and key length
- *Rijndael is k-secure and hermetic*

SPN- Structure Block Cipher

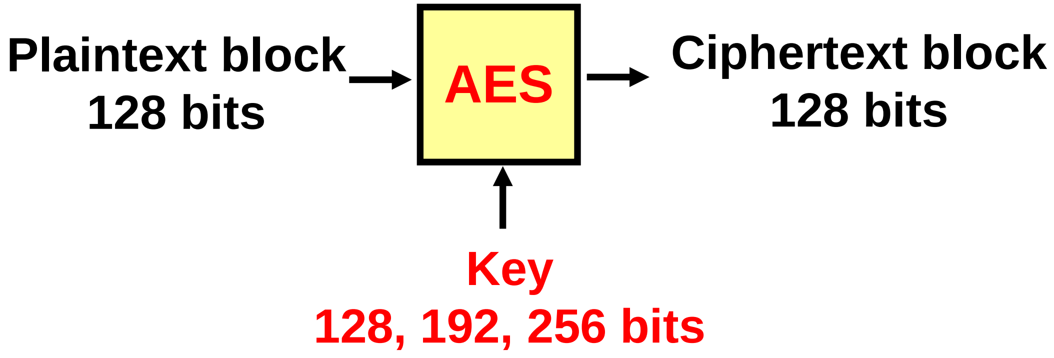
- **More constraints on the round function: must be invertible**
  - ✓ **(Invertible) Nonlinear Layer: Confusion**
  - ✓ **(Invertible) Linear Layer: Diffusion**
- **Relatively new architecture and faster than Feistel-structure**
- **Parallel computation**
- **Typically E $\neq$ D**



A sketch of a Substitution-Permutation Network with 3 rounds,

Advanced Encryption Standard (AES)

AES Parameters

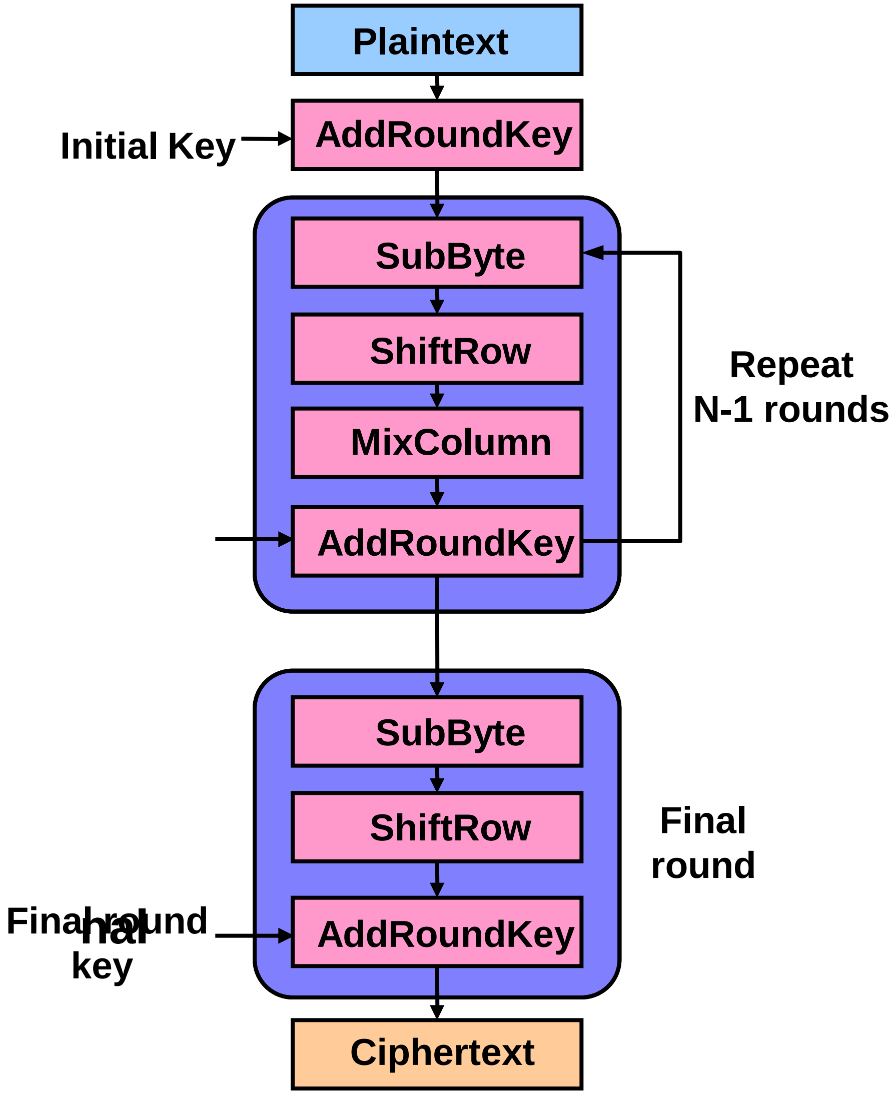


**H/W : Get AES program (AES-128, 192 or 256) and check the Encryption and Decryption by choosing a plaintext and key at random.
Submit your source code and test vector in hardcopy.**

AES Architecture

- SPN-type block cipher
- Block size = 128 bits
- Key size / No. round
  - 128 bits → 10 rounds
  - 192 bits → 12 rounds
  - 256 bits → 14 rounds

- Round transformation
  - SubBytes
  - ShiftRow
  - MixColumn
  - AddRoundKey

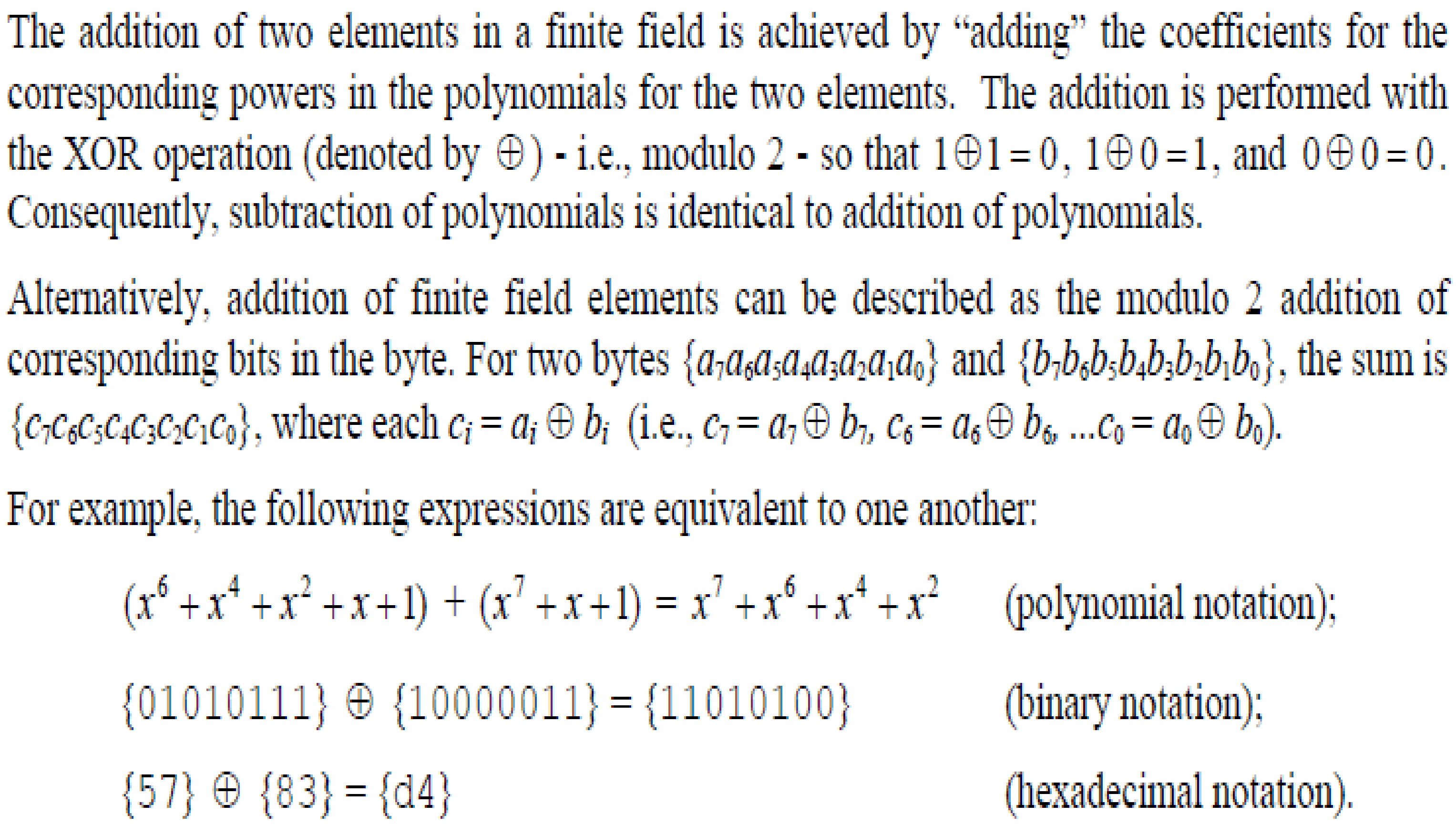


Finite Field

- Every nonzero element has multiplicative inverse.
- Prime number, Zp
  - gcd, Extended Euclidean Algorithm
- Representation of n-bit data
  - Binary
  - Hexadecimal
  - Polynomial
- GF(p), GF(2^n) over irreducible polynomial
  - Ex) GF(2^3) /$x^3 + x + 1$
  - Addition and multiplication

  - AES uses GF(2^8) /$f(x)= x^8 + x^4 + x^3 + x +1$

Finite Field - Addition



The addition of two elements in a finite field is achieved by "adding" the coefficients for the corresponding powers in the polynomials for the two elements. The addition is performed with the XOR operation (denoted by $\oplus$) - i.e., modulo 2 - so that $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, and $0 \oplus 0 = 0$. Consequently, subtraction of polynomials is identical to addition of polynomials.

Alternatively, addition of finite field elements can be described as the modulo 2 addition of corresponding bits in the byte. For two bytes $\{a_7a_6a_5a_4a_3a_2a_1a_0\}$ and $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$, the sum is $\{c_7c_6c_5c_4c_3c_2c_1c_0\}$, where each $c_i = a_i \oplus b_i$ (i.e., $c_7 = a_7 \oplus b_7$, $c_6 = a_6 \oplus b_6$, ...$c_0 = a_0 \oplus b_0$).

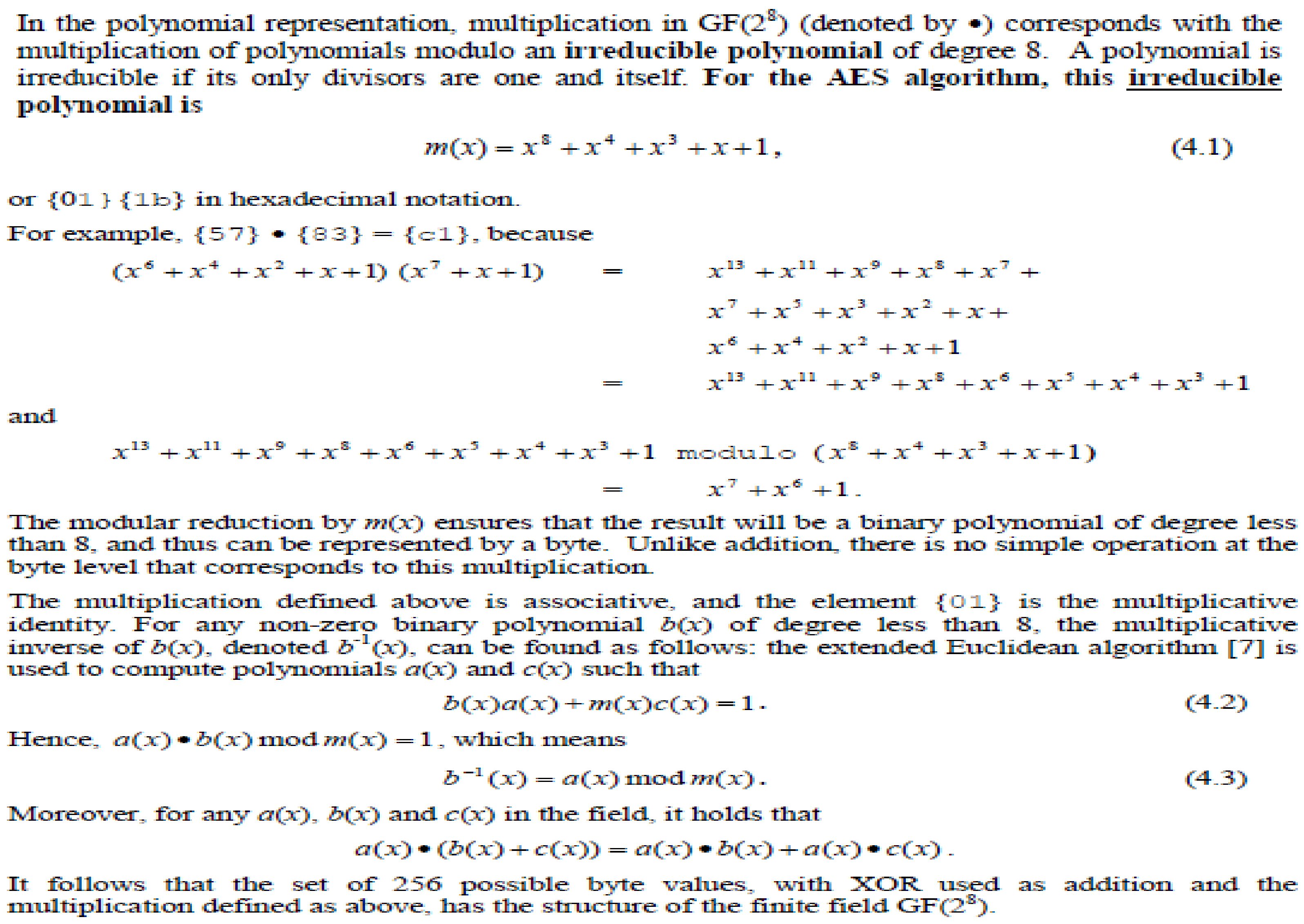
For example, the following expressions are equivalent to one another:

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$ (polynomial notation);

$\{01010111\} \oplus \{10000011\} = \{11010100\}$ (binary notation);

$\{57\} \oplus \{83\} = \{d4\}$ (hexadecimal notation).

Finite Field - Multiplication



In the polynomial representation, multiplication in GF(2^8) (denoted by •) corresponds with the multiplication of polynomials modulo an **irreducible polynomial** of degree 8. A polynomial is irreducible if its only divisors are one and itself. **For the AES algorithm, this <u>irreducible</u> polynomial is**

$$m(x) = x^8 + x^4 + x^3 + x + 1, \qquad (4.1)$$

or {01}{1b} in hexadecimal notation.

For example, {57} • {83} = {c1}, because

$$\begin{aligned}(x^6 + x^4 + x^2 + x + 1)\,(x^7 + x + 1) &= x^{13} + x^{11} + x^9 + x^8 + x^7 + \\ &\quad x^7 + x^5 + x^3 + x^2 + x + \\ &\quad x^6 + x^4 + x^2 + x + 1 \\ &= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1\end{aligned}$$

and

$$\begin{aligned}x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \ \text{modulo}\ (x^8 + x^4 + x^3 + x + 1) \\ = x^7 + x^6 + 1.\end{aligned}$$

The modular reduction by $m(x)$ ensures that the result will be a binary polynomial of degree less than 8, and thus can be represented by a byte. Unlike addition, there is no simple operation at the byte level that corresponds to this multiplication.

The multiplication defined above is associative, and the element {01} is the multiplicative identity. For any non-zero binary polynomial $b(x)$ of degree less than 8, the multiplicative inverse of $b(x)$, denoted $b^{-1}(x)$, can be found as follows: the extended Euclidean algorithm [7] is used to compute polynomials $a(x)$ and $c(x)$ such that

$$b(x)a(x) + m(x)c(x) = 1. \qquad (4.2)$$

Hence, $a(x) \bullet b(x) \bmod m(x) = 1$, which means

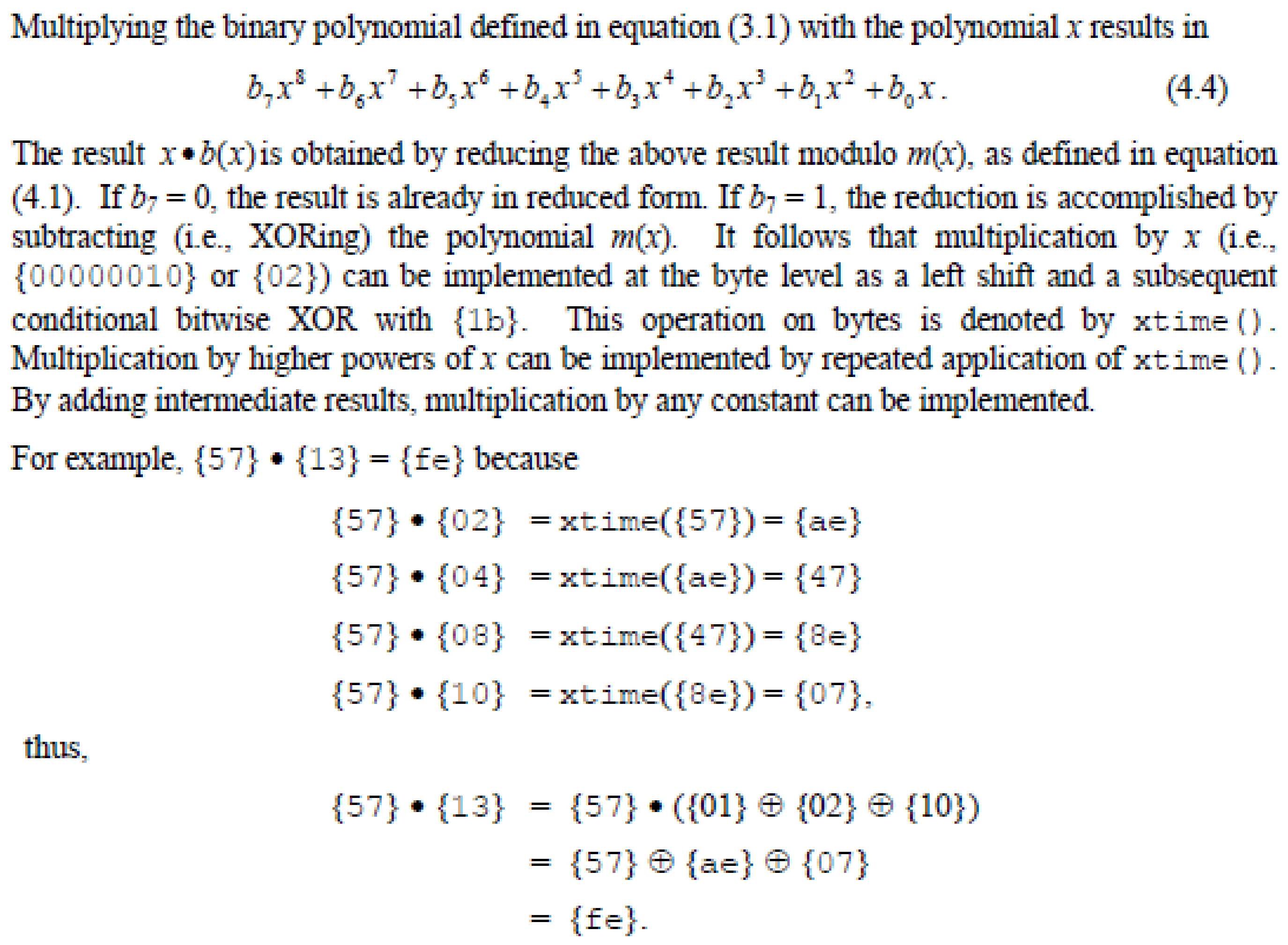
$$b^{-1}(x) = a(x) \bmod m(x). \qquad (4.3)$$

Moreover, for any $a(x)$, $b(x)$ and $c(x)$ in the field, it holds that

$$a(x) \bullet (b(x) + c(x)) = a(x) \bullet b(x) + a(x) \bullet c(x).$$

It follows that the set of 256 possible byte values, with XOR used as addition and the multiplication defined as above, has the structure of the finite field GF(2^8).

Finite Field – Multiplication by x



Multiplying the binary polynomial defined in equation (3.1) with the polynomial x results in

$$b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x. \qquad (4.4)$$

The result $x \bullet b(x)$ is obtained by reducing the above result modulo $m(x)$, as defined in equation (4.1). If $b_7 = 0$, the result is already in reduced form. If $b_7 = 1$, the reduction is accomplished by subtracting (i.e., XORing) the polynomial $m(x)$. It follows that multiplication by x (i.e., {00000010} or {02}) can be implemented at the byte level as a left shift and a subsequent conditional bitwise XOR with {1b}. This operation on bytes is denoted by `xtime()`. Multiplication by higher powers of x can be implemented by repeated application of `xtime()`. By adding intermediate results, multiplication by any constant can be implemented.

For example, {57} • {13} = {fe} because

$$\begin{aligned}
\{57\} \bullet \{02\} &= \mathtt{xtime}(\{57\}) = \{\mathtt{ae}\} \\
\{57\} \bullet \{04\} &= \mathtt{xtime}(\{\mathtt{ae}\}) = \{47\} \\
\{57\} \bullet \{08\} &= \mathtt{xtime}(\{47\}) = \{8\mathtt{e}\} \\
\{57\} \bullet \{10\} &= \mathtt{xtime}(\{8\mathtt{e}\}) = \{07\},
\end{aligned}$$

thus,

$$\begin{aligned}
\{57\} \bullet \{13\} &= \{57\} \bullet (\{01\} \oplus \{02\} \oplus \{10\}) \\
&= \{57\} \oplus \{\mathtt{ae}\} \oplus \{07\} \\
&= \{\mathtt{fe}\}.
\end{aligned}$$

Round Transformation – S-box

The `SubBytes()` transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). This S-box (Fig. 7), which is invertible, is constructed by composing two transformations:

1. Take the multiplicative inverse in the finite field GF(2^8), described in Sec. 4.2; the element {00} is mapped to itself.
2. Apply the following affine transformation (over GF(2)):

$$b_i' = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i \qquad (5.1)$$

for $0 \le i < 8$, where b_i is the i^{th} bit of the byte, and c_i is the i^{th} bit of a byte c with the value {63} or {01100011}. Here and elsewhere, a prime on a variable (e.g., b') indicates that the variable is to be updated with the value on the right.

In matrix form, the affine transformation element of the S-box can be expressed as:

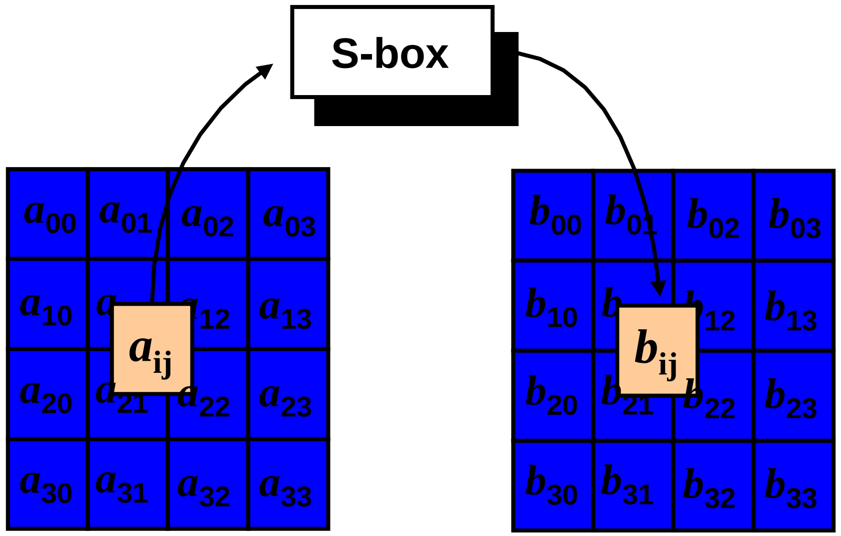
$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

The S-box used in the `SubBytes()` transformation is presented in hexadecimal form in Fig. 7.

For example, if $s_{1,1} = \{53\}$, then the substitution value would be determined by the intersection of the row with index '5' and the column with index '3' in Fig. 7. This would result in $s'_{1,1}$ having a value of {ed}.

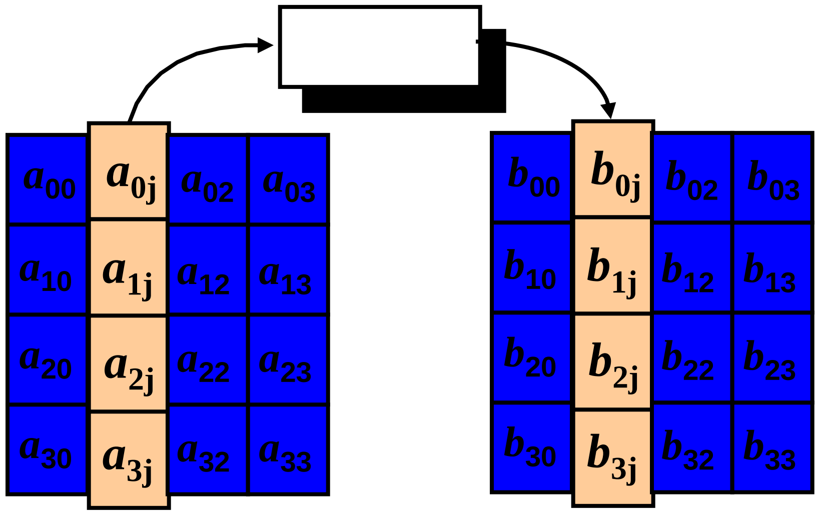
		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	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
x	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

Figure 7. **S**-box: substitution values for the byte **xy** (in hexadecimal format).

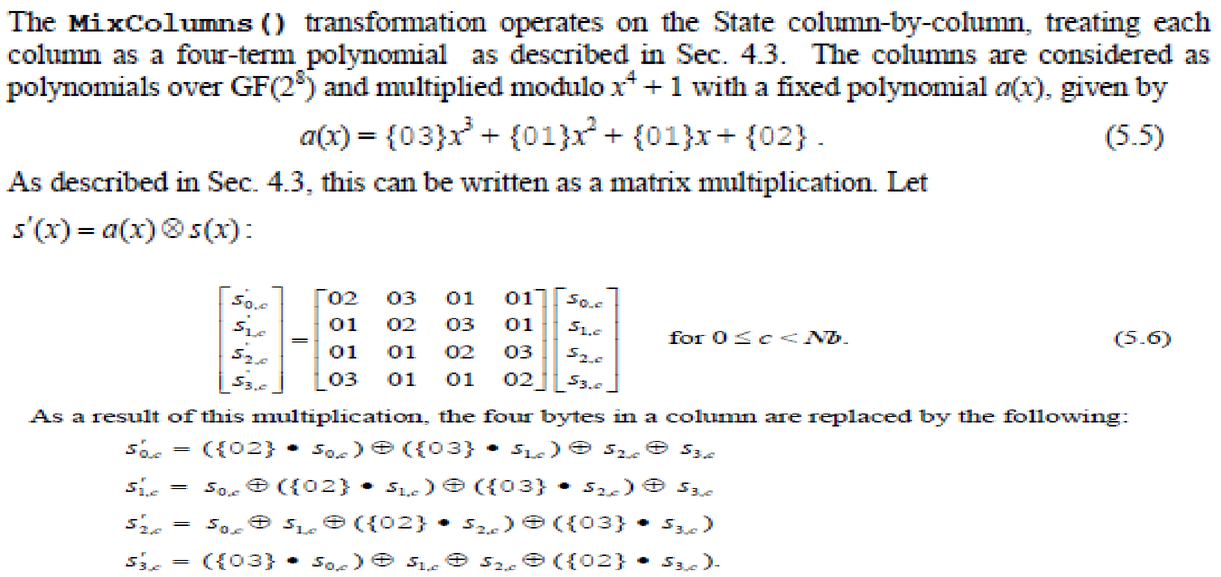

S-box
a_{00} a_{01} a_{02} a_{03}
a_{10} a_{12} a_{13}
a_{ij}
a_{20} a_{21} a_{22} a_{23}
a_{30} a_{31} a_{32} a_{33}
b_{00} b_{01} b_{02} b_{03}
b_{10} b_{12} b_{13}
b_{ij}
b_{20} b_{21} b_{22} b_{23}
b_{30} b_{31} b_{32} b_{33}

SubBytes

Round Transformation – Mixcolumns



a_{00} a_{0j} a_{02} a_{03}
a_{10} a_{1j} a_{12} a_{13}
a_{20} a_{2j} a_{22} a_{23}
a_{30} a_{3j} a_{32} a_{33}
b_{00} b_{0j} b_{02} b_{03}
b_{10} b_{1j} b_{12} b_{13}
b_{20} b_{2j} b_{22} b_{23}
b_{30} b_{3j} b_{32} b_{33}



The **`MixColumns()`** transformation operates on the State column-by-column, treating each column as a four-term polynomial as described in Sec. 4.3. The columns are considered as polynomials over GF(2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial $a(x)$, given by

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} . \qquad (5.5)$$

As described in Sec. 4.3, this can be written as a matrix multiplication. Let

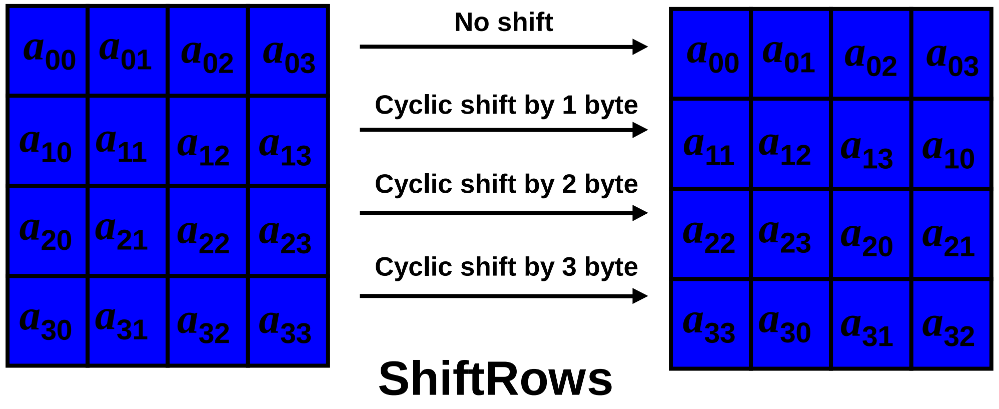
$s'(x) = a(x) \otimes s(x)$:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < Nb. \qquad (5.6)$$

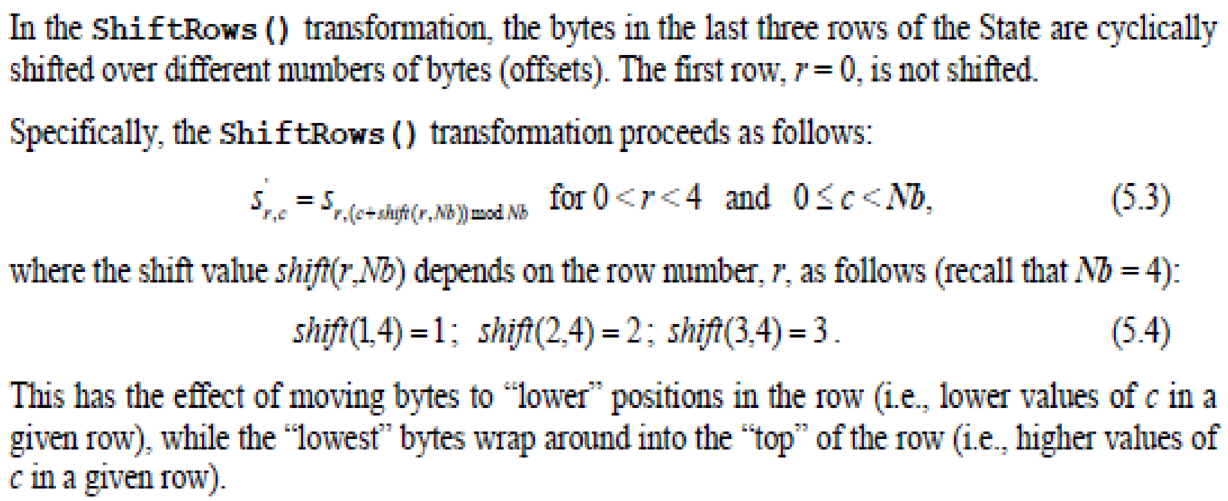
As a result of this multiplication, the four bytes in a column are replaced by the following:

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

Round Transformation – ShiftRows



a_{00} a_{01} a_{02} a_{03}
a_{10} a_{11} a_{12} a_{13}
a_{20} a_{21} a_{22} a_{23}
a_{30} a_{31} a_{32} a_{33}
No shift
Cyclic shift by 1 byte
Cyclic shift by 2 byte
Cyclic shift by 3 byte
a_{00} a_{01} a_{02} a_{03}
a_{11} a_{12} a_{13} a_{10}
a_{22} a_{23} a_{20} a_{21}
a_{33} a_{30} a_{31} a_{32}
ShiftRows



In the `ShiftRows()` transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, $r = 0$, is not shifted.

Specifically, the `ShiftRows()` transformation proceeds as follows:

$$s'_{r,c} = s_{r,(c+shift(r,Nb)) \bmod Nb} \quad \text{for } 0 < r < 4 \quad \text{and} \quad 0 \le c < Nb, \qquad (5.3)$$

where the shift value $shift(r,Nb)$ depends on the row number, r, as follows (recall that $Nb = 4$):

$$shift(1,4) = 1; \quad shift(2,4) = 2; \quad shift(3,4) = 3. \qquad (5.4)$$

This has the effect of moving bytes to "lower" positions in the row (i.e., lower values of c in a given row), while the "lowest" bytes wrap around into the "top" of the row (i.e., higher values of c in a given row).

AddRoundKey

The input block is XOR-ed with the round key

$$\begin{array}{|c|c|c|c|c|c|}\hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & a_{0,4} & a_{0,5} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|c|c|}\hline k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} & k_{0,4} & k_{0,5} \\ \hline k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} & k_{1,4} & k_{1,5} \\ \hline k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} & k_{2,4} & k_{2,5} \\ \hline k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} & k_{3,4} & k_{3,5} \\ \hline \end{array} = \begin{array}{|c|c|c|c|c|c|}\hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} & b_{0,4} & b_{0,5} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} & b_{1,5} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} & b_{2,5} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} & b_{3,5} \\ \hline \end{array}$$

Figure 5: In the key addition the Round Key is bitwise EXORed to the State.

Best Known Attack

- Related-key attack can break 256-bit AES with a complexity of 2^{119}, which is faster than brute force but is still infeasible. 192-bit AES can also be defeated in a similar manner, but at a complexity of 2^{176}. 128-bit AES is not affected by this attack.
- Chosen-plaintext attack can break 8 rounds of 192- and 256-bit AES, and 7 rounds of 128-bit AES, although the workload is impractical at $2^{128} \sim 2^{119}$. (Ferguson et al., 2000).

Feistel-Newtork vs. SP Network (Summary)

Feistel-Network

64-bit block	128-bit block
• DES/3DES	• SEED
• BLOWFISH	• TWOFISH
• CAST128	• CAST256
• RC5	• RC6
	• MARS

- ➢ Fewer constraints on the round function
- ➢ More cryptanalytic experience
- ➢ Serial in nature
- ➢ Typically E = D with round keys in reverse order

SP Network

64-bit block	128-bit block
• SAFER	• AES
• SAFER+	• CRYPTON
• IDEA	• SERPENT

- ➢ More constraints on the round function: must be invertible
- ➢ Less cryptanalytic experience: relatively new architecture
- ➢ more parallel computation
- ➢ Typically E $\neq$ D