

Advanced Encryption Standard

"It seems very simple."

"It is very simple. But if you don't know what the key is it's virtually indecipherable."

—Talking to Strange Men, Ruth Rendell



Origins

- ❑ clear a replacement for DES was needed
 - have theoretical attacks that can break it
 - have demonstrated exhaustive key search attacks
- ❑ can use Triple-DES – but slow, has small blocks
- ❑ US NIST (*National Institute of standards and technology*) issued call for ciphers in 1997
- ❑ 15 candidates accepted in Jun 98
- ❑ 5 were shortlisted in Aug-99
- ❑ Rijndael was selected as the AES in Oct-2000
- ❑ issued as FIPS PUB 197 standard in Nov-2001

AES Requirements

- ❑ private key symmetric block cipher
- ❑ 128-bit data, 128/192/256-bit keys
- ❑ stronger & faster than Triple-DES
- ❑ active life of 20-30 years (+ archival use)
- ❑ provide full specification & design details
- ❑ both C & Java implementations
- ❑ It would be available royalty free anywhere in the world

AES Evaluation Criteria

□ initial criteria:

- security – effort for practical cryptanalysis
- cost – in terms of computational efficiency
- algorithm & implementation characteristics

□ final criteria

- general security
- ease of software & hardware implementation
- implementation attacks
- flexibility (in en/decrypt, keying, other factors)

AES Evaluation Criteria

□ Initial criteria:

- security – effort to practically cryptanalyze an algorithm. Emphasis on the practicality of attack. Brute force attacks with current and projected technology need not be considered.
- cost – high computational efficiency so as to be usable in high speed applications such as broadband links.
- algorithm & implementation characteristics – flexibility, suitability for variety of h/w and s/w implementations, simplicity to make the analysis of security more straightforward.

■ Final criteria

- General security- To assess general security, NIST relied on the public security analysis conducted by the cryptographic community.
- Software & hardware implementation ease
- Attacks on implementations – Analysis attacks
- Flexibility (in en/decrypt, keying, other factors)
Parameter flexibility includes ease of support for other key and block sizes and ease of increasing number of rounds in order to cope up with the newly discovered attacks.
Implementation flexibility refers to the possibility of optimizing cipher elements for particular environments.
- Key Agility– This refers to the ability to change keys quickly and with a minimum of resources.

AES Shortlist

- after testing and evaluation, shortlist in Aug-99:
 - MARS (IBM) - complex, fast, high security margin
 - RC6 (USA) - v. simple, v. fast, low security margin
 - Rijndael (Belgium) - clean, fast, good security margin
 - Serpent (Euro) - slow, clean, v. high security margin
 - Twofish (USA) - complex, v. fast, high security margin
- then subject to further analysis & comment
- saw contrast between algorithms with
 - few complex rounds verses many simple rounds
 - which refined existing ciphers verses new proposals

The AES Cipher - Rijndael

- ❑ designed by Rijmen-Daemen in Belgium
- ❑ has 128/192/256 bit keys, 128 bit data
- ❑ an **iterative** rather than **feistel** cipher
 - processes data as block of 4 columns of 4 bytes
 - operates on entire data block in every round
- ❑ designed to be:
 - resistant against known attacks
 - speed and code compactness on many CPUs
 - design simplicity



AES parameters:


Key size (words/bytes/ bits)	4/16/128	6/24/192	8/32/256
Plaintext block size	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

Rijndael has variable number of rounds=

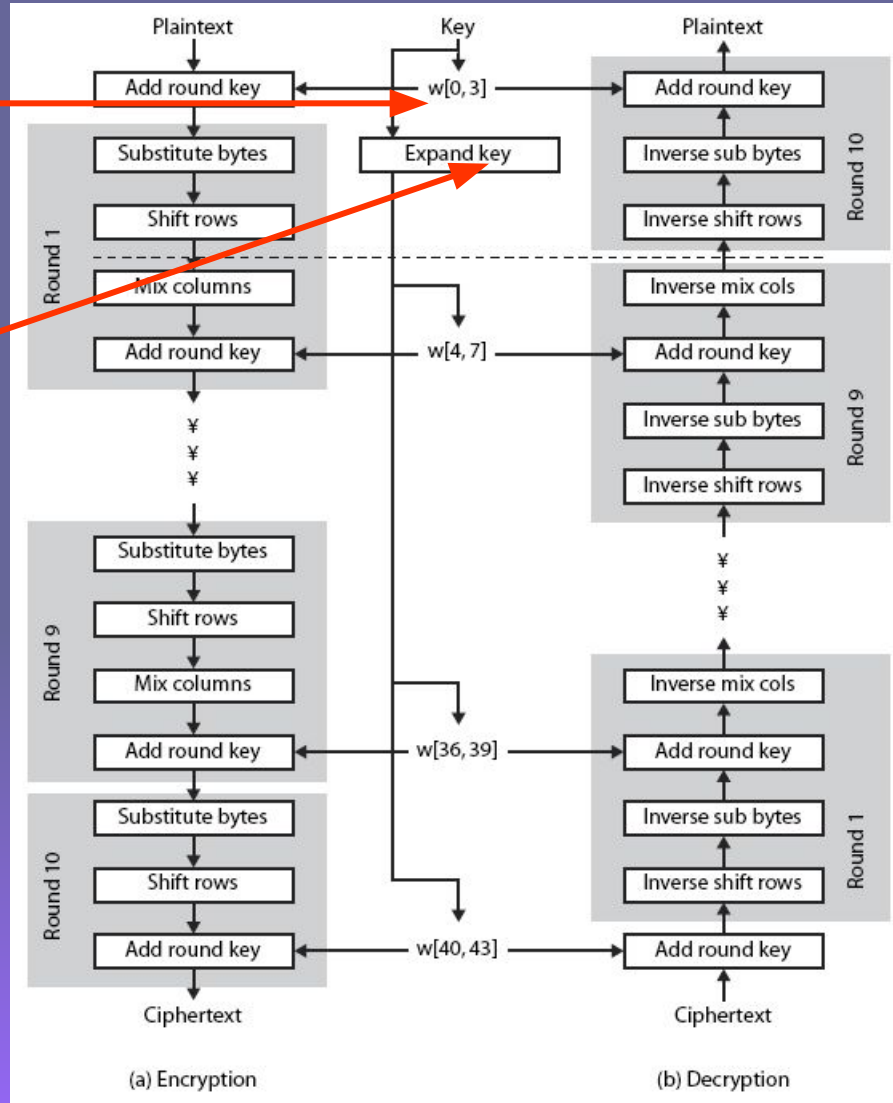
- 9 if both the block and the key are 128 bits long.
- 11 if either the block or the key is 192 bits long, and neither of them is longer than that.
- 13 if either the block or the key is 256 bits long



Each round involves four steps:
(3 substitutions + 1 permutation)

- 1) The **Byte Sub** step, where each byte of the block is replaced by its substitute in an S-box
 - 2) The **Shift Row** step- simple permutation
 - 3) The **Mix Column** step- A substitution that makes use of some arithmetic (Matrix multiplication)
 - 4) Final step is **Add Round Key**- Simple bitwise XOR of the current block with portion of the expanded key.
- 
- Decorative concentric circles in the bottom right corner of the slide.

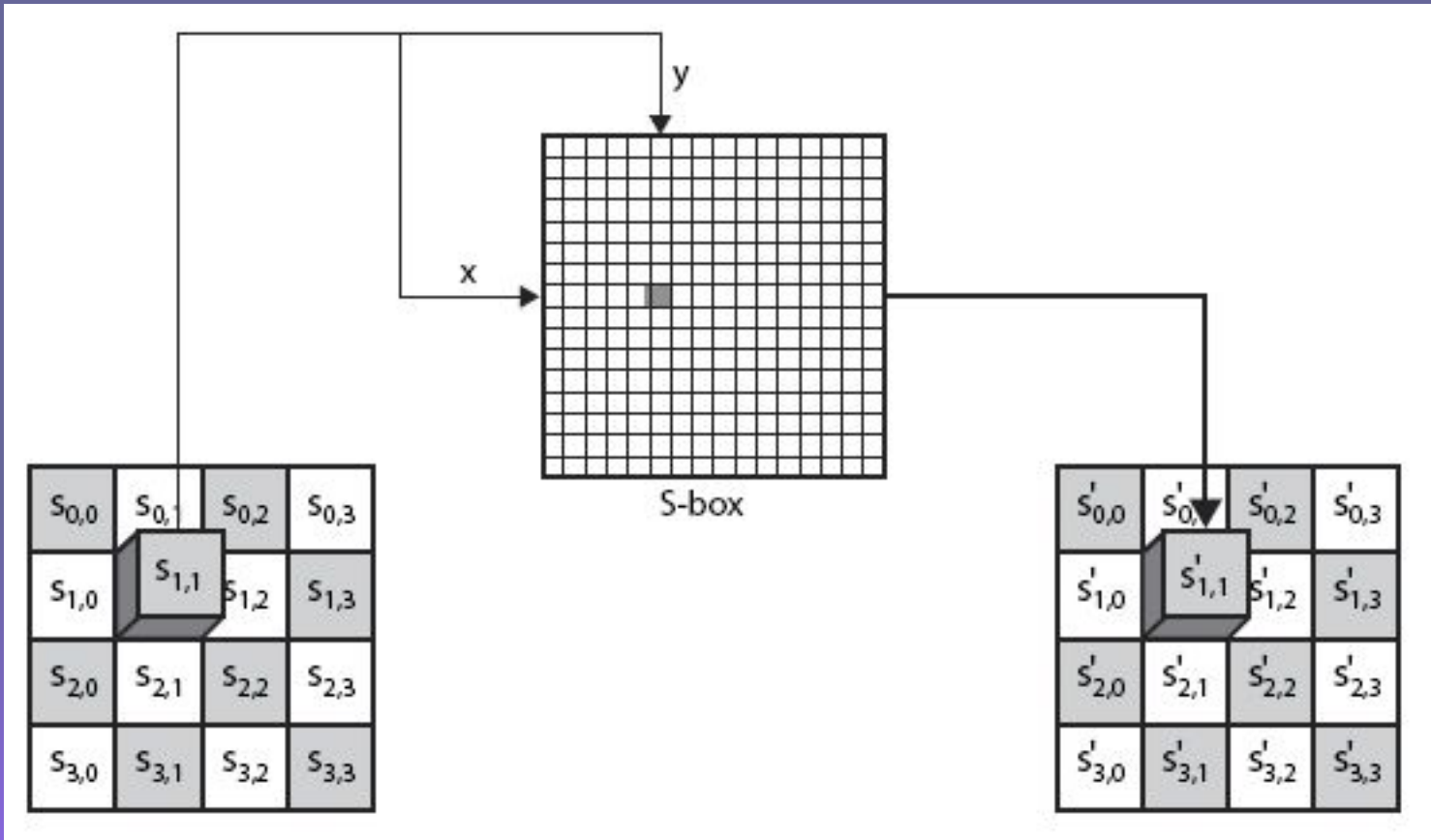
Rijndael



Byte Substitution

- a simple substitution of each byte
- uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
- designed to be resistant to all known attacks

Byte Substitution

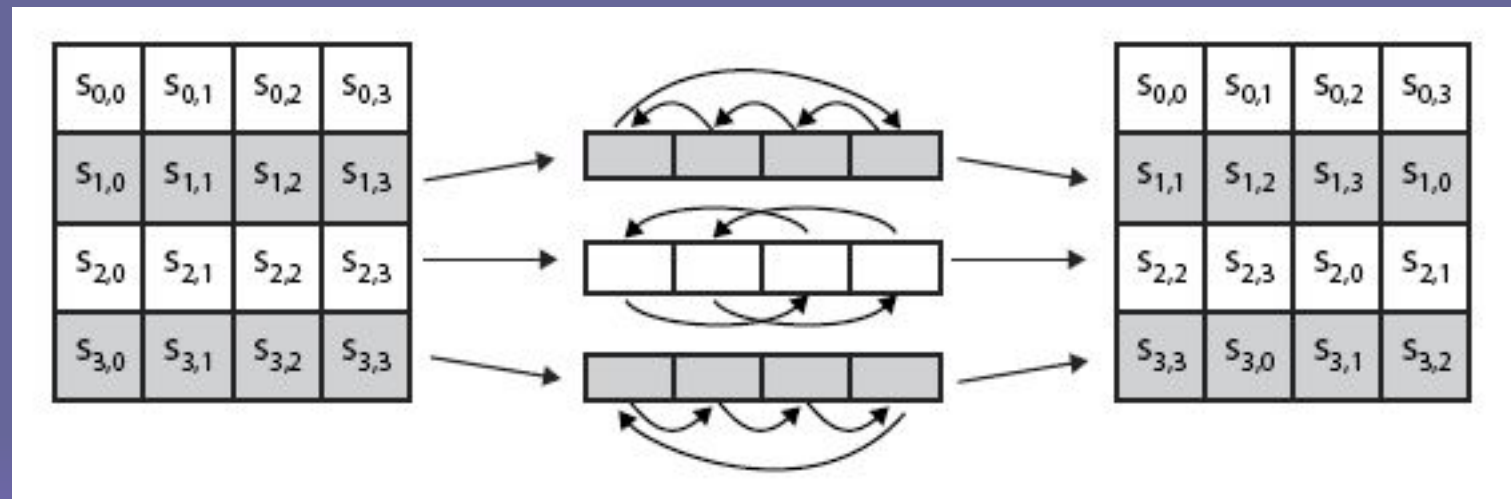


Shift Rows

- a circular byte shift in each each
 - 1st row is unchanged
 - 2nd row does 1 byte circular shift to left
 - 3rd row does 2 byte circular shift to left
 - 4th row does 3 byte circular shift to left
- decrypt inverts using shifts to right
- since state is processed by columns, this step permutes bytes between the columns



Shift Rows



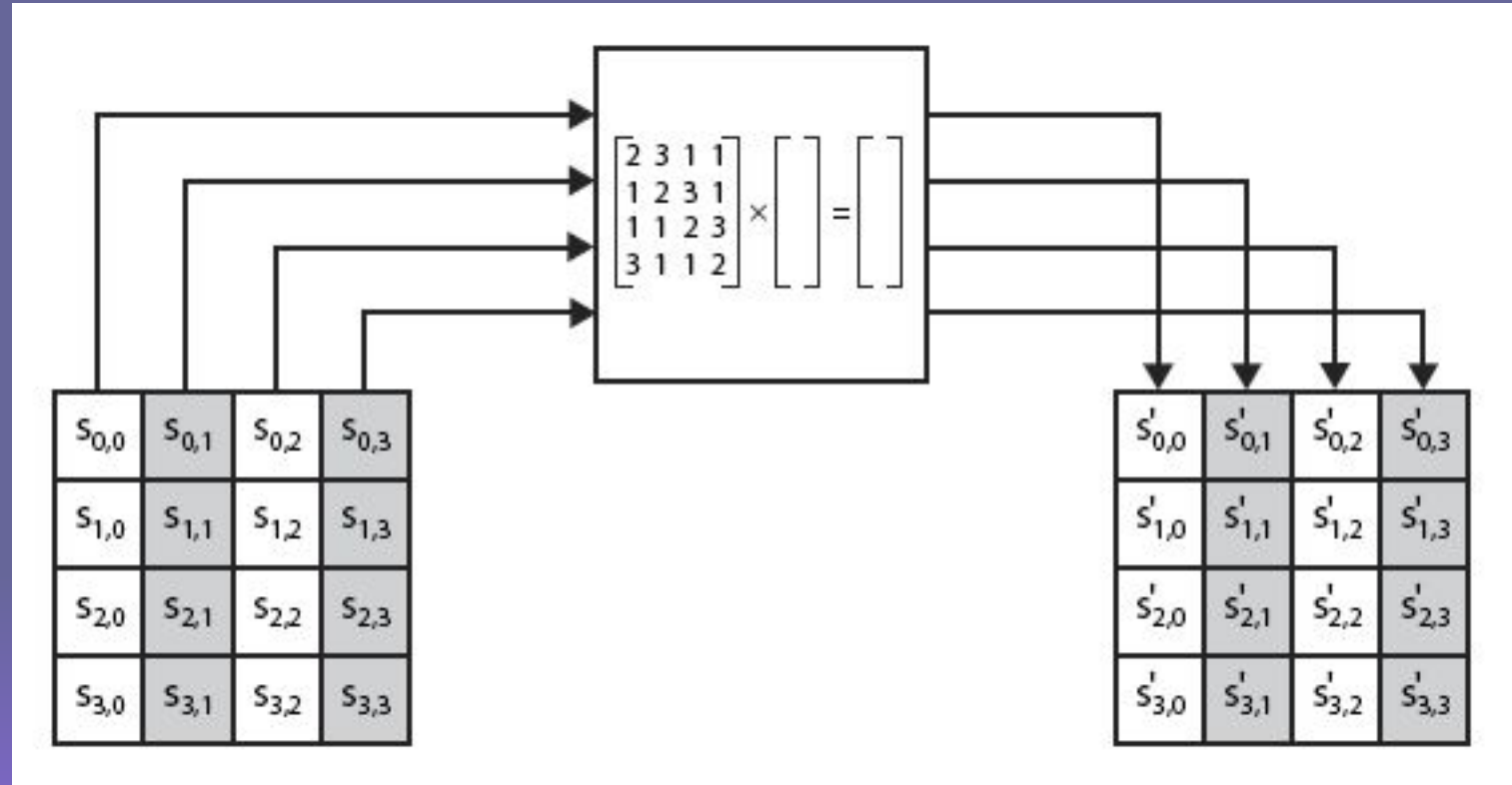
Mix Columns

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in $GF(2^8)$ using prime poly $m(x) = x^8 + x^4 + x^3 + x + 1$

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

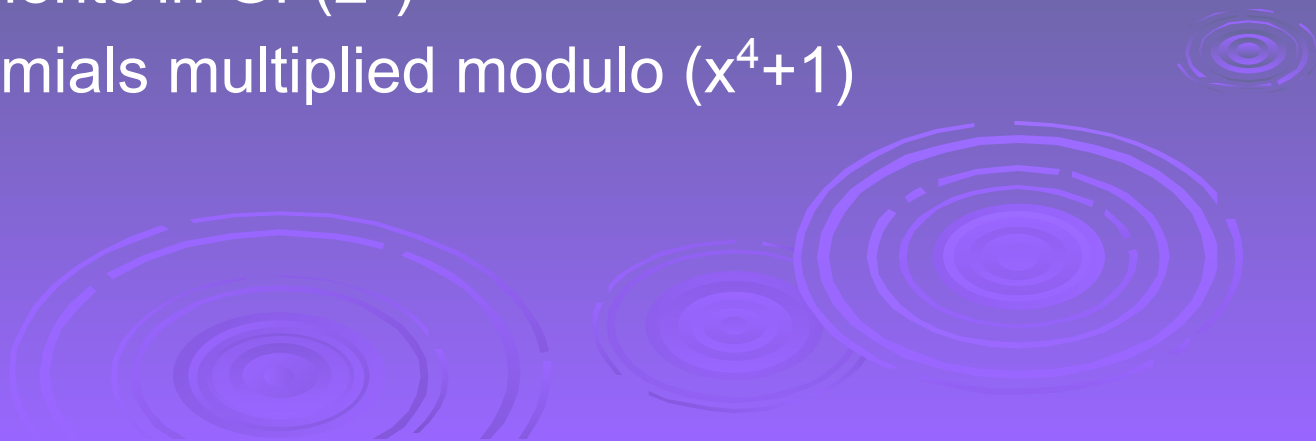
$GF(p^n)$ = Finite Field of order p^n

Mix Columns



Mix Columns

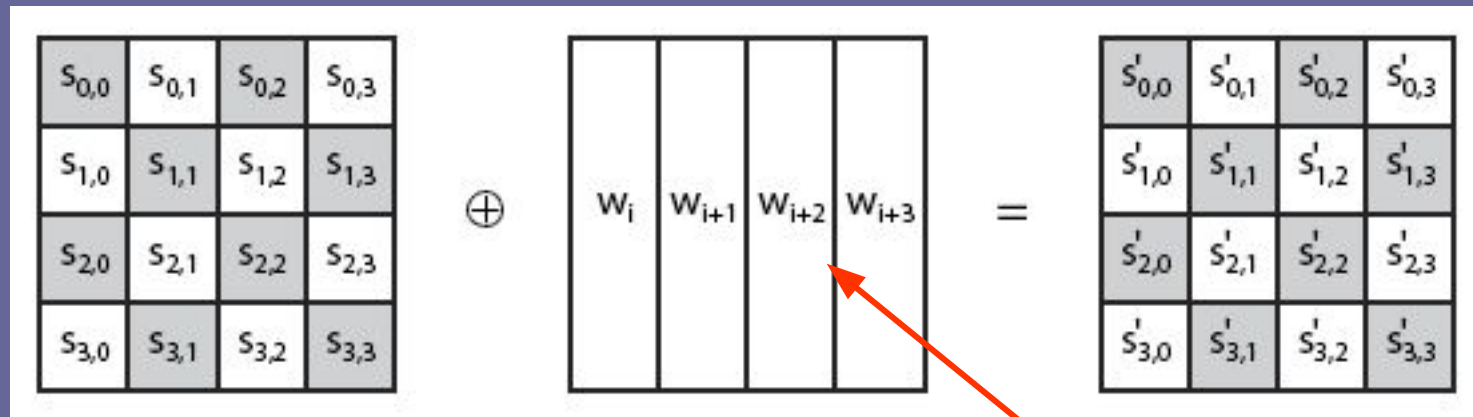
- can express each col as 4 equations
 - to derive each new byte in col
- decryption requires use of inverse matrix
 - with larger coefficients, hence a little harder
- have an alternate characterisation
 - each column a 4-term polynomial
 - with coefficients in $GF(2^8)$
 - and polynomials multiplied modulo (x^4+1)



Add Round Key

- XOR state with 128-bits of the round key
- again processed by column (though effectively a series of byte operations)
- inverse for decryption identical
 - since XOR own inverse, with reversed keys
- designed to be as simple as possible
 - a form of Vernam cipher on expanded key
 - requires other stages for complexity / security

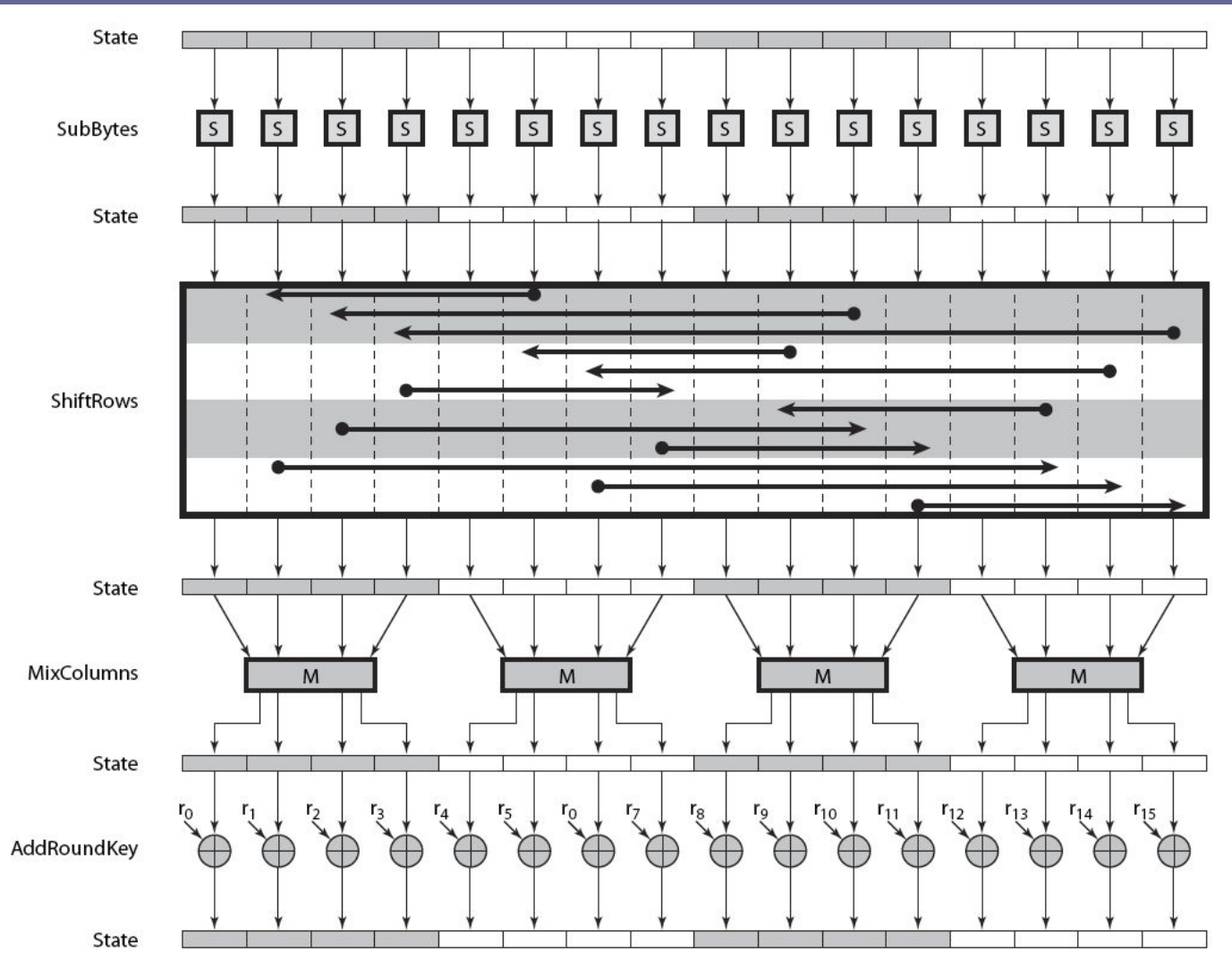
Add Round Key



4 words = 16 bytes = 128 bits

- 128 bits of State are bitwise XORed with 128 bits of the round key
- Figure indicates column-wise operation with words in round key

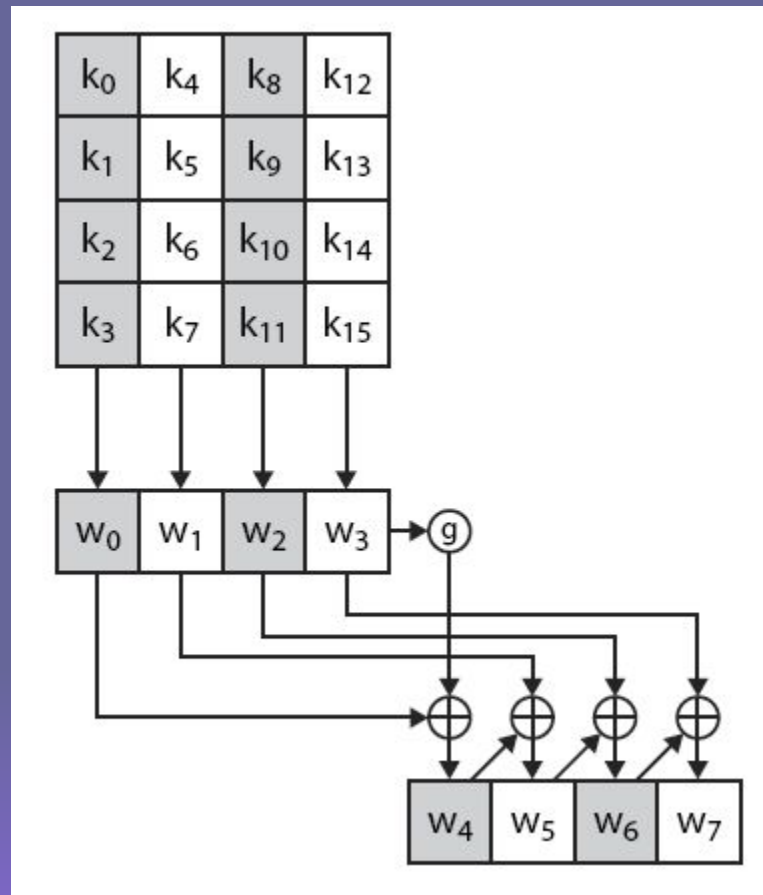
AES Round



AES Key Expansion

- takes 128-bit (16-byte) key and expands into array of 44/52/60 words (*32-bit words*)
- start by copying key into first 4 words of expanded key
- then loop creating words that depend on values in previous & 4 places back
 - in 3 of 4 cases just XOR these together
 - 1st word in 4 has rotate + S-box + XOR round constant on previous, before XOR 4th back

AES Key Expansion



- Generation of first 8 words of Expanded key
- g – complex function

Key Expansion Rationale

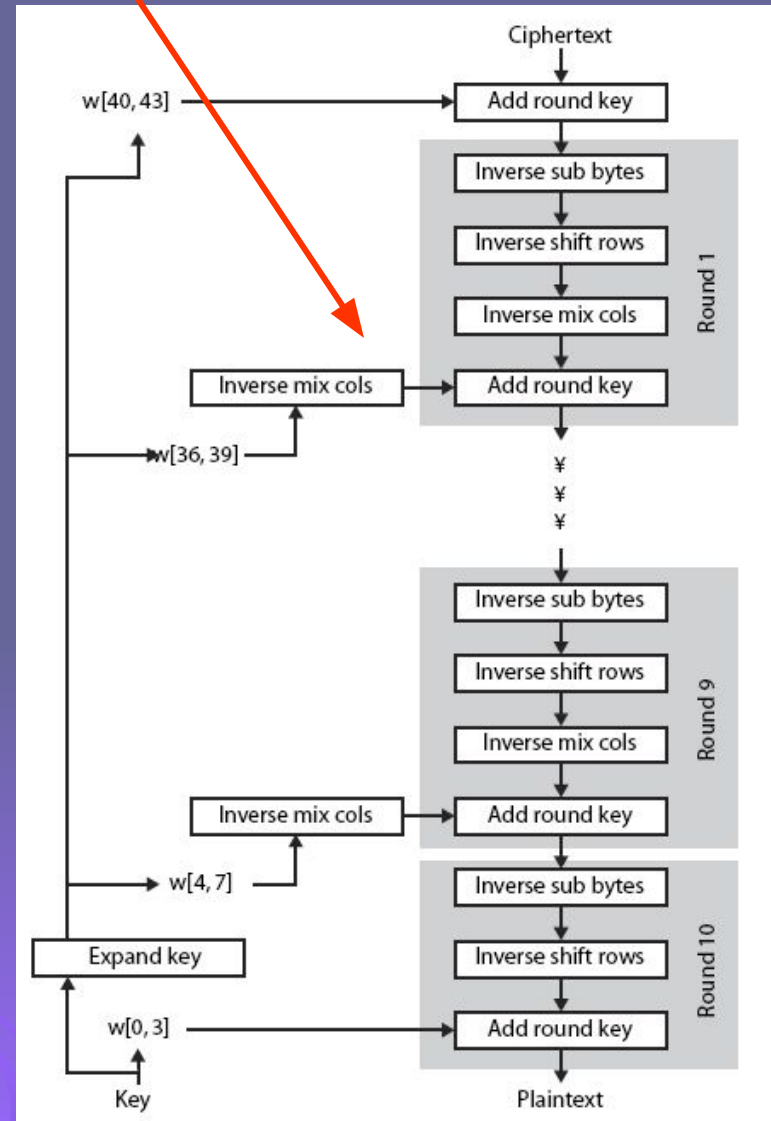
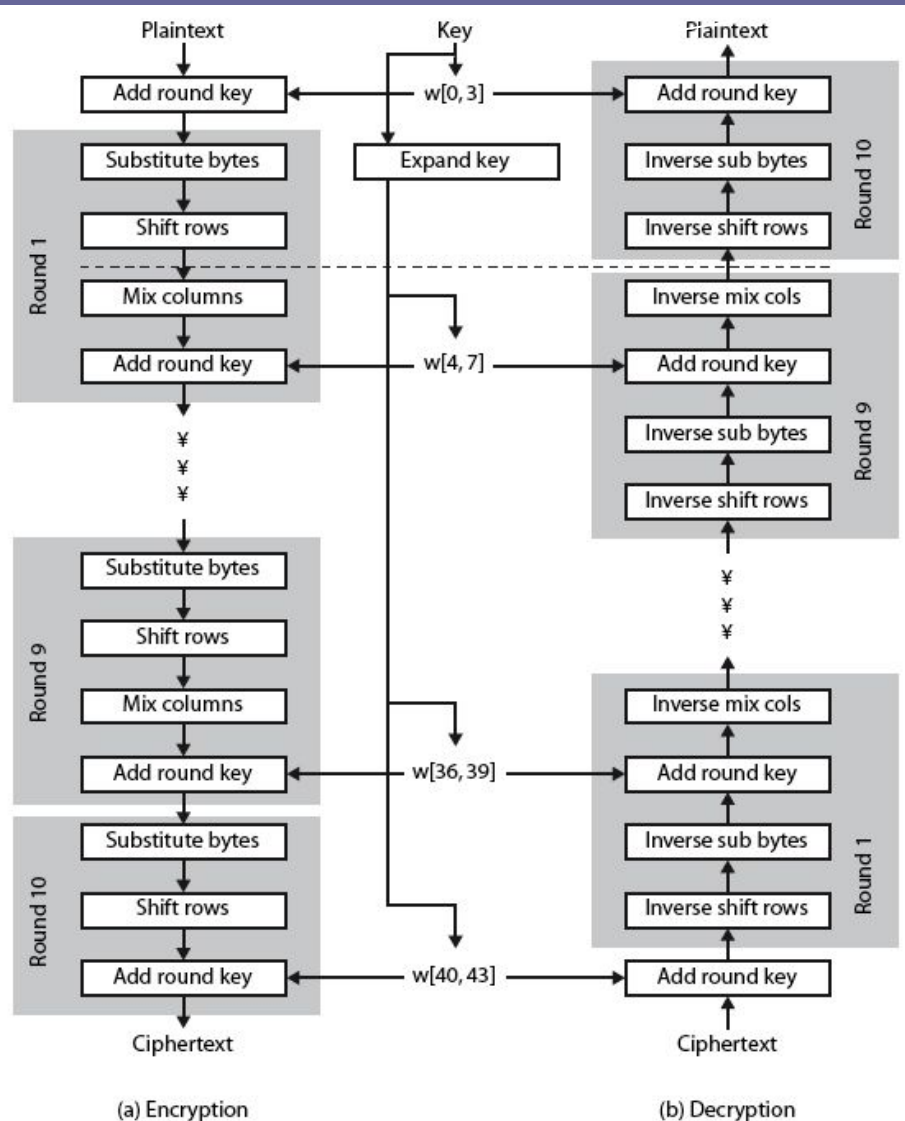
- designed to resist known attacks
- design criteria included
 - knowing part/round key insufficient to find remaining
 - invertible transformation
 - fast on wide range of CPU's
 - use round constants to break symmetry
 - diffuse key bits into round keys
 - enough non-linearity to make analysis
 - simplicity of description



AES Decryption

- ❑ AES decryption is not identical to encryption since steps done in reverse
- ❑ but can define an equivalent inverse cipher with steps as for encryption
 - using inverses of each step
 - with a different key schedule
- ❑ works since result is unchanged when
 - swap byte substitution & shift rows
 - swap mix columns & add round key

AES Decryption



Implementation Aspects

- can efficiently implement on 8-bit CPU
 - byte substitution works on bytes using a table of 256 entries
 - shift rows is simple byte shift
 - add round key works on byte XOR's
 - mix columns requires matrix multiply in $GF(2^8)$ which works on byte values, can be simplified to use table lookups & byte XOR's

Implementation Aspects

- can efficiently implement on 32-bit CPU
 - redefine steps to use 32-bit words
 - can precompute 4 tables of 256-words
 - then each column in each round can be computed using 4 table lookups + 4 XORs
 - at a cost of 4Kb to store tables
- designers believe this as a key factor for efficient implementation and preference in its selection as the AES cipher

Summary

- 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

