Implementation Security In Cryptography

Lecture 07: The Anatomy of AES

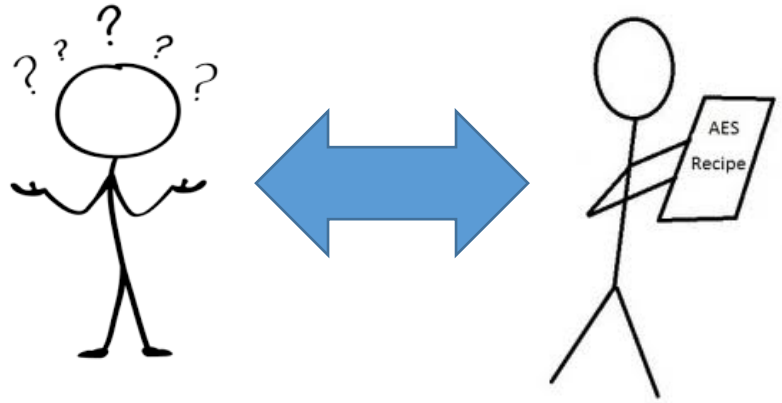
Recap

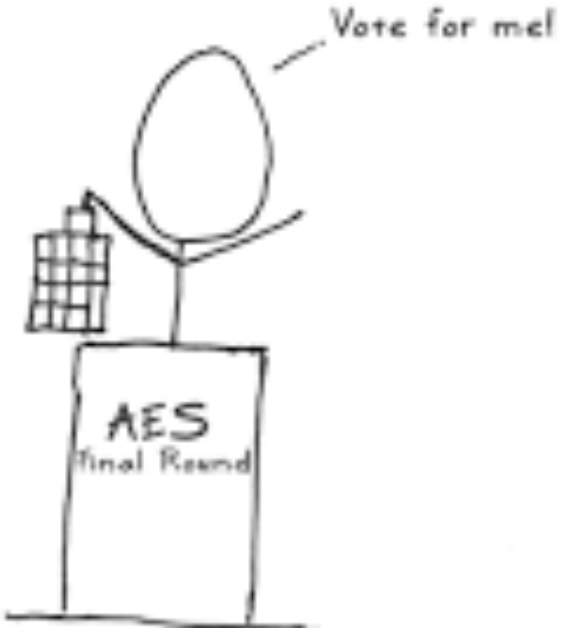
- In the last lecture
    - Finite Field

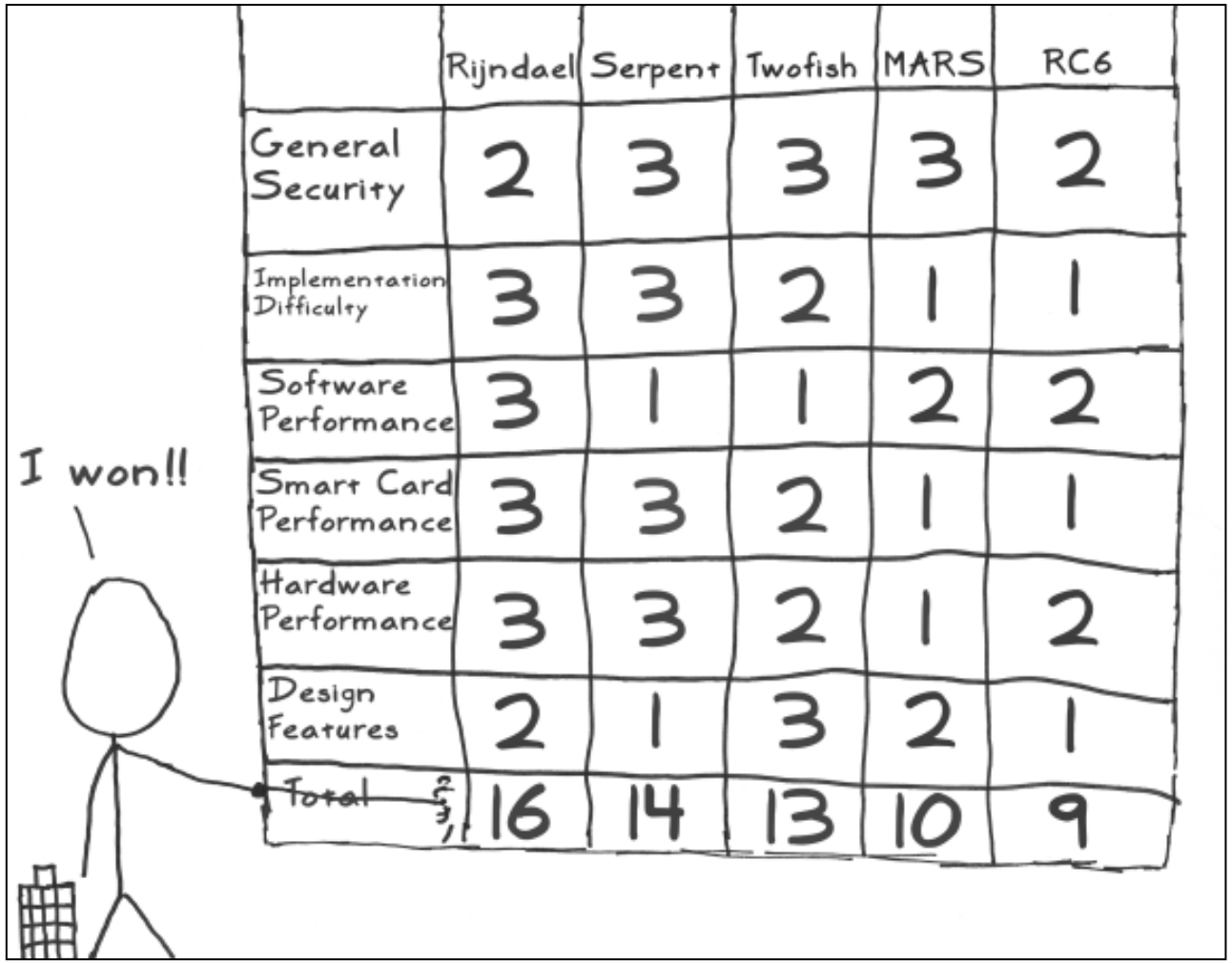
Today

- How things relate to AES

What is AES?







	Rijndael	Serpent	Twofish	MARS	RC6
General Security	2	3	3	3	2
Implementation Difficulty	3	3	2	1	1
Software Performance	3	1	1	2	2
Smart Card Performance	3	3	2	1	1
Hardware Performance	3	3	2	1	2
Design Features	2	1	3	2	1
~~Total~~	16	14	13	10	9

Ref: http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

Advanced Encryption Standard (AES)

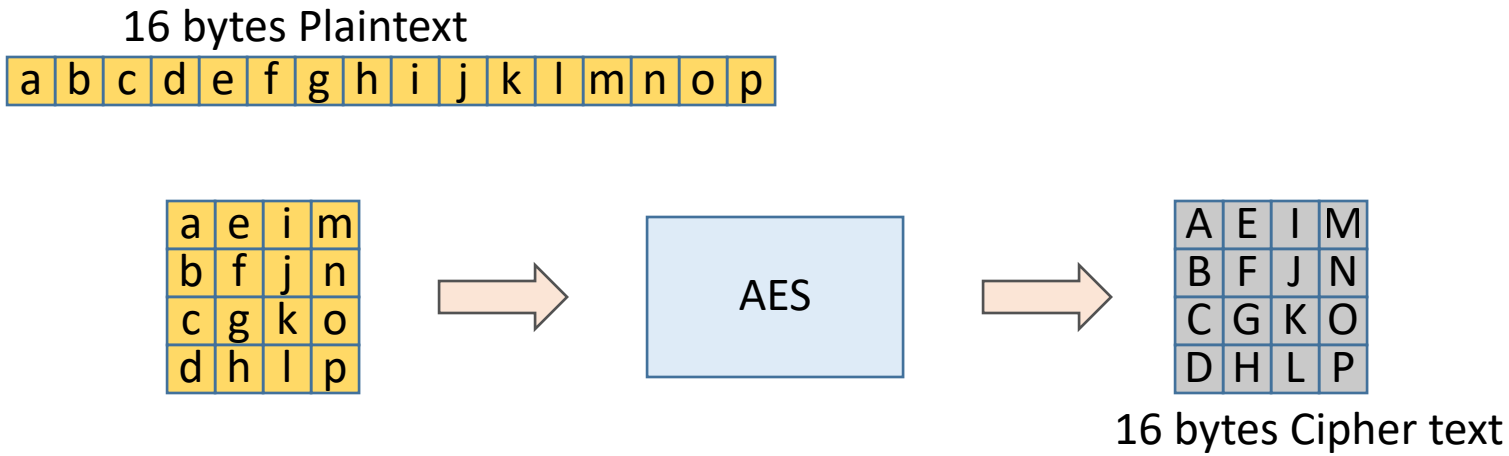
- NIST's standard for block cipher since October 2000.

	Key Length	No. of rounds
AES-128	16 bytes	10
AES-192	24bytes	12
AES-256	32bytes	14

- Each round has
  - *Round key addition*
  - Confusion Layer : *Byte Substitution*
  - Diffusion Layer : *Shift row* and *Mix column*

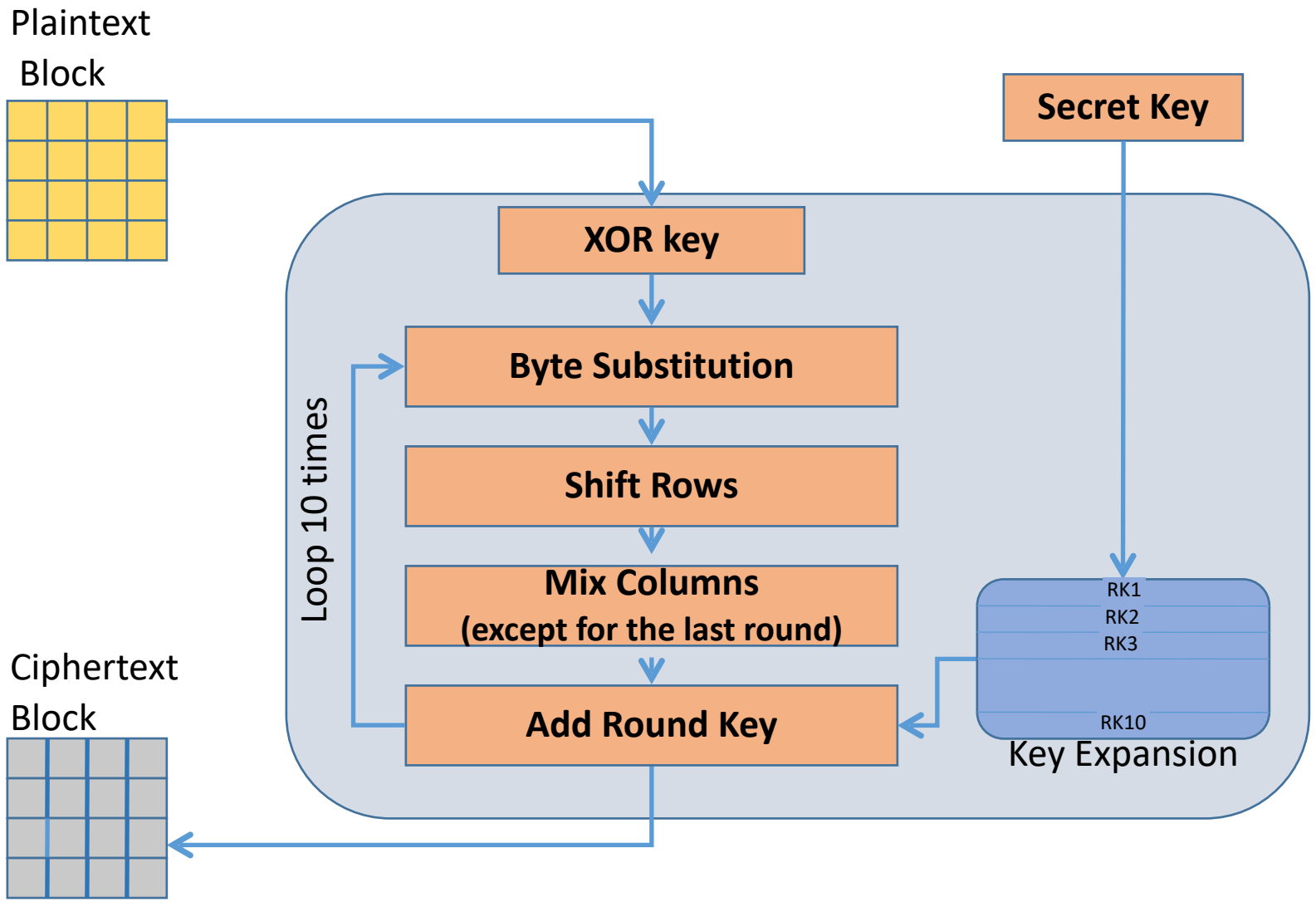
    (the last round does not have mix column step)

The AES State Representation

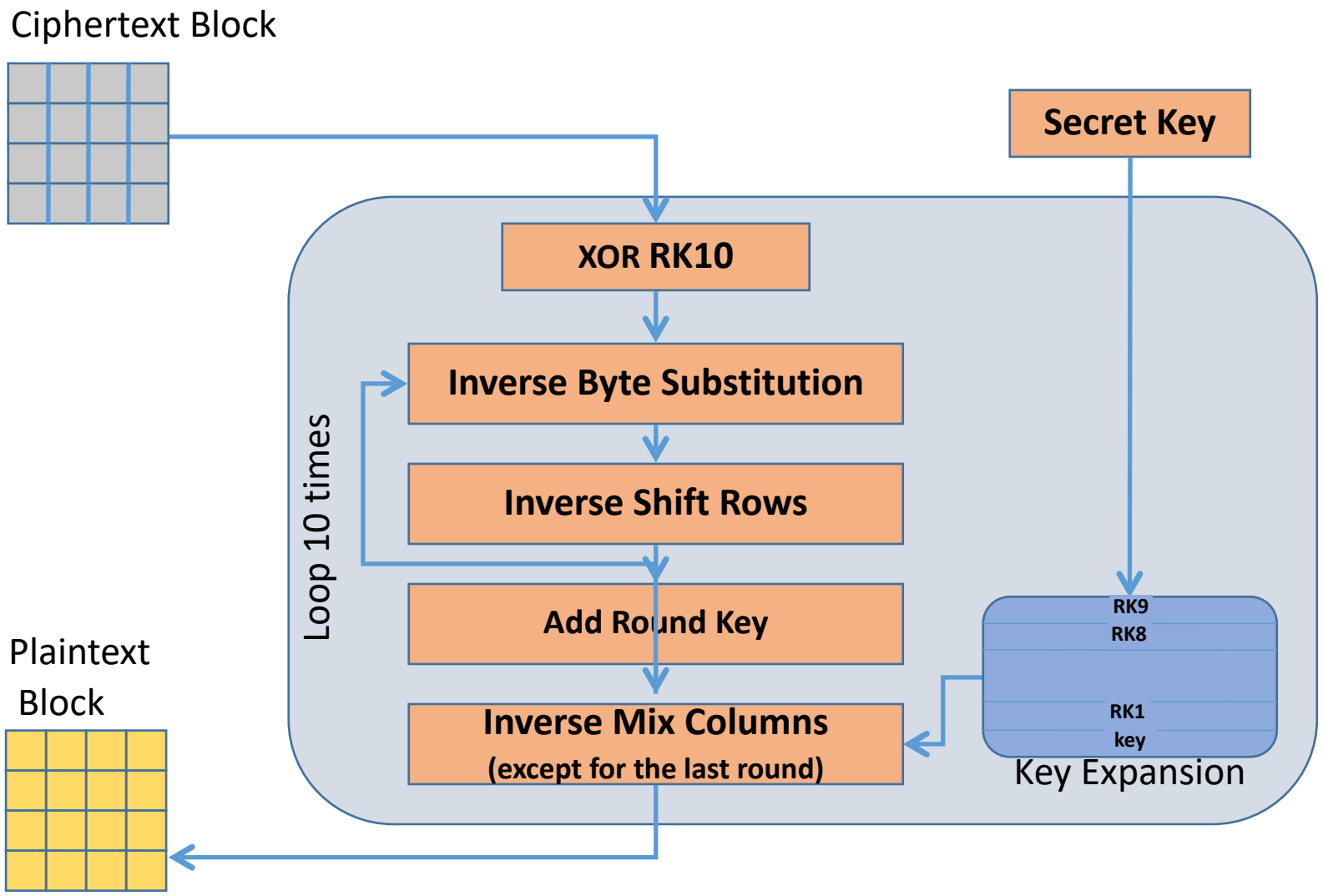


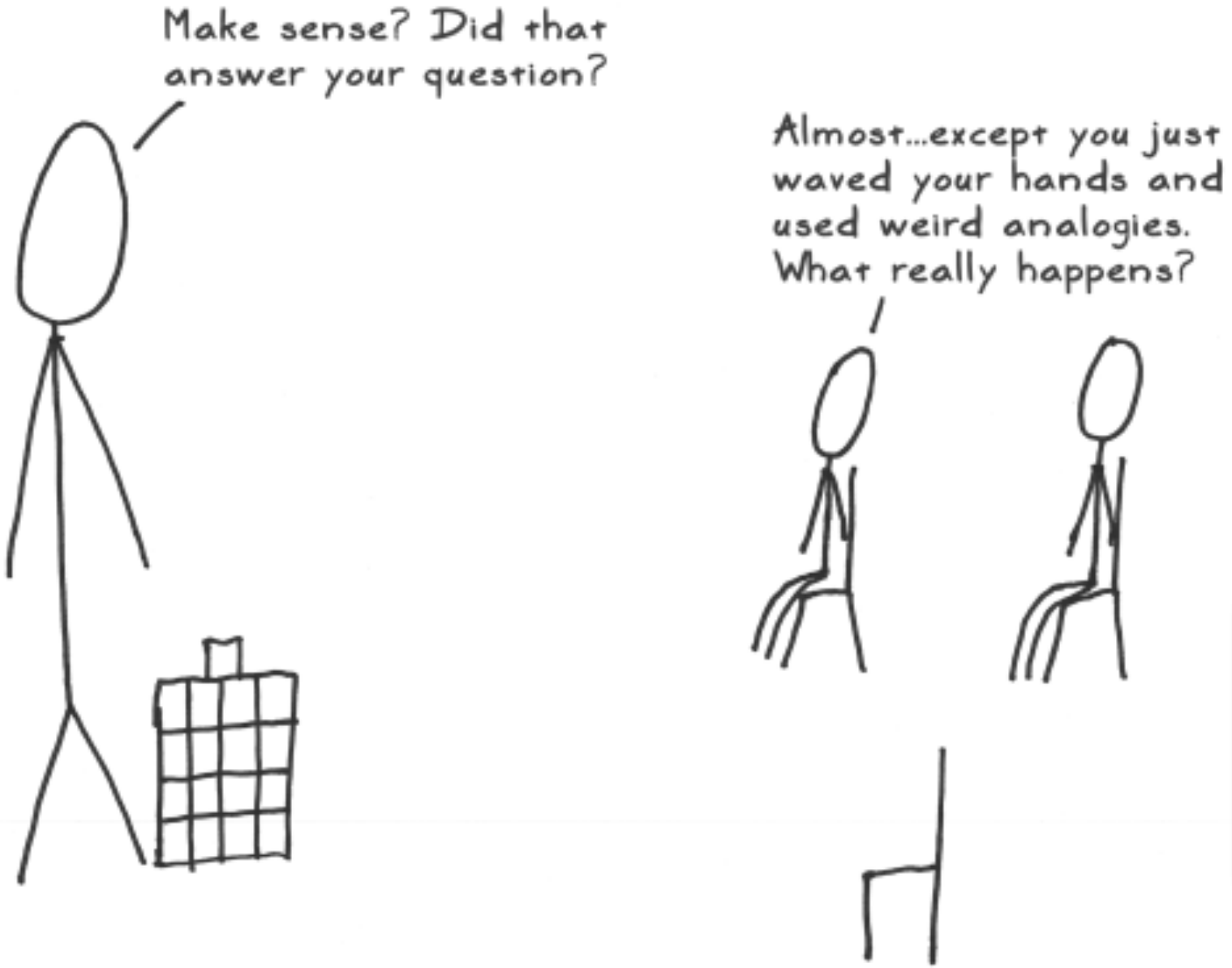
- 16 bytes arranged in a 4x4 matrix of bytes

AES-128 Encryption



AES-128 Decryption

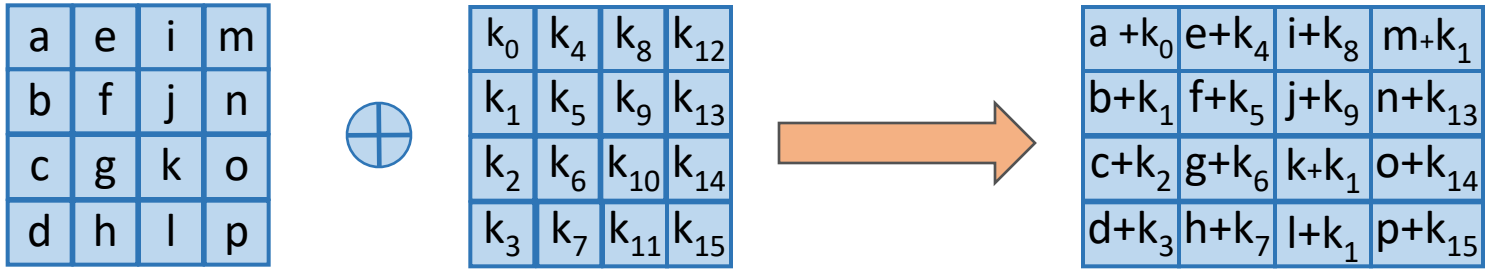




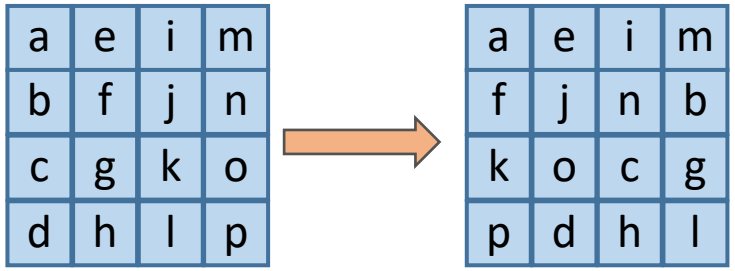
AES has a more mathematically oriented description. For that we need to understand “fields”

- We cannot avoid the maths here if we want to implement…
- We cannot avoid hardware design principles
- Hardware design principles help in software too..

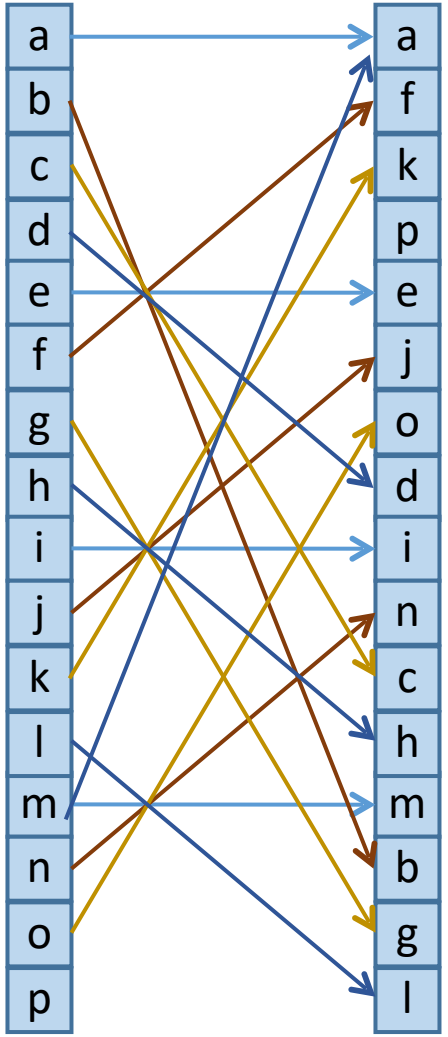
Add Round Key



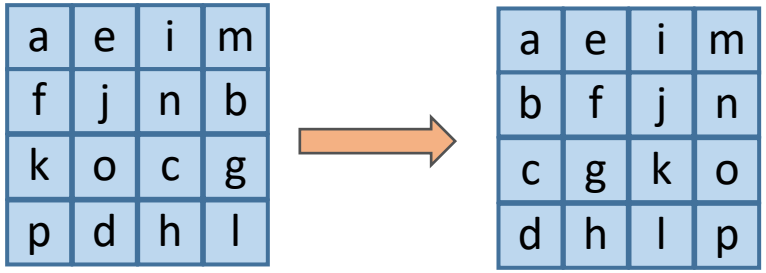
Shift Rows



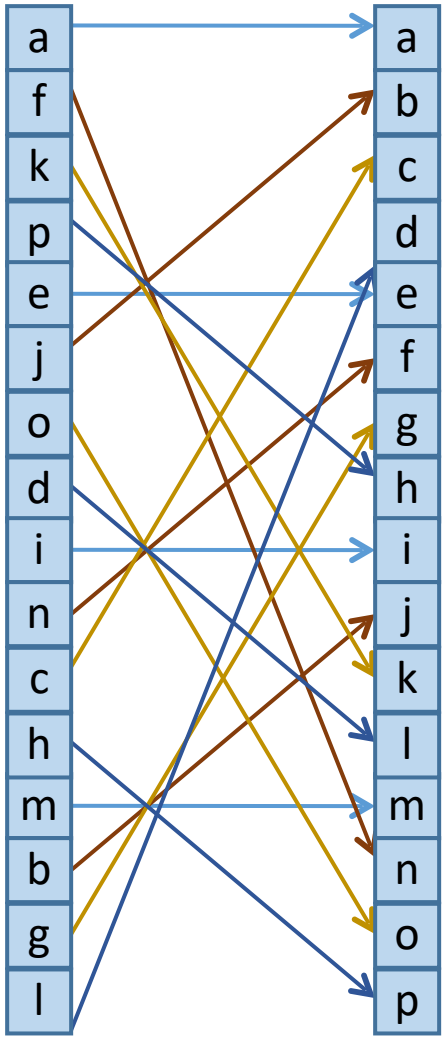
- ***ShiftRows***
  - Leave the First row untouched
  - Left Rotate (2nd Row by 8 bits)
  - Left Rotate (3rd Row by 16 bits)
  - Left Rotate (4th Row by 24 bits)
- **Implementation in Hardware**
  - No resources required, only mapping with wires



Inverse Shift Rows

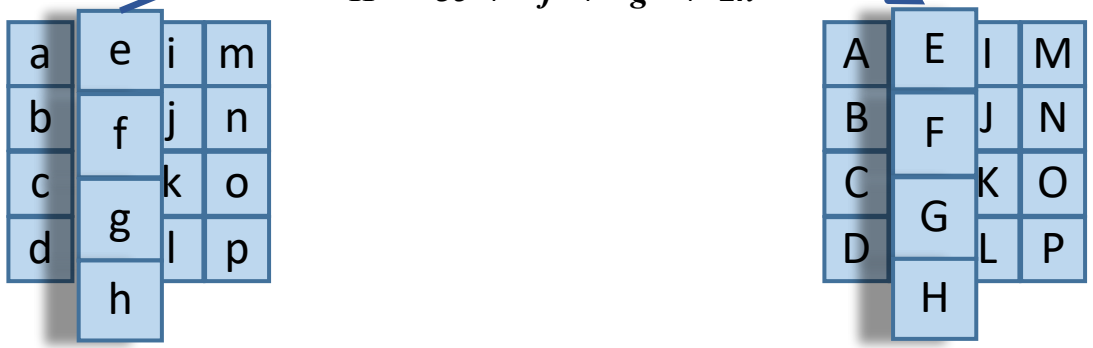


- ***ShiftRows***
  - Leave the First row untouched
  - Right Rotate (2nd Row by 8 bits)
  - Right Rotate (3rd Row by 16 bits)
  - Right Rotate (4th Row by 24 bits)



Mix Columns

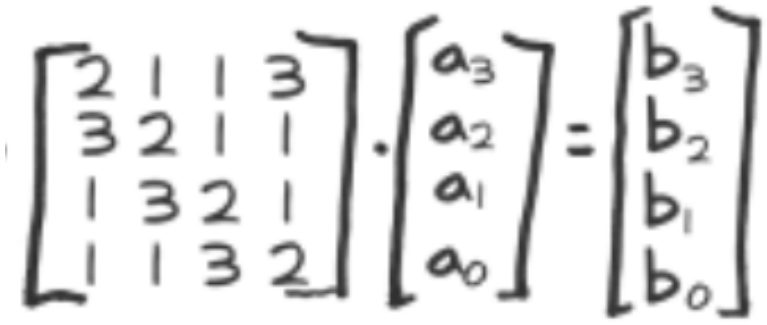
$$E = 2e + 3f + g + h$$
$$F = e + 2f + 3g + h$$
$$G = e + f + 2g + 3h$$
$$H = 3e + f + g + 2h$$



- The 4x4 matrix is multiplied with the matrix

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

How MixColumns is implemented?



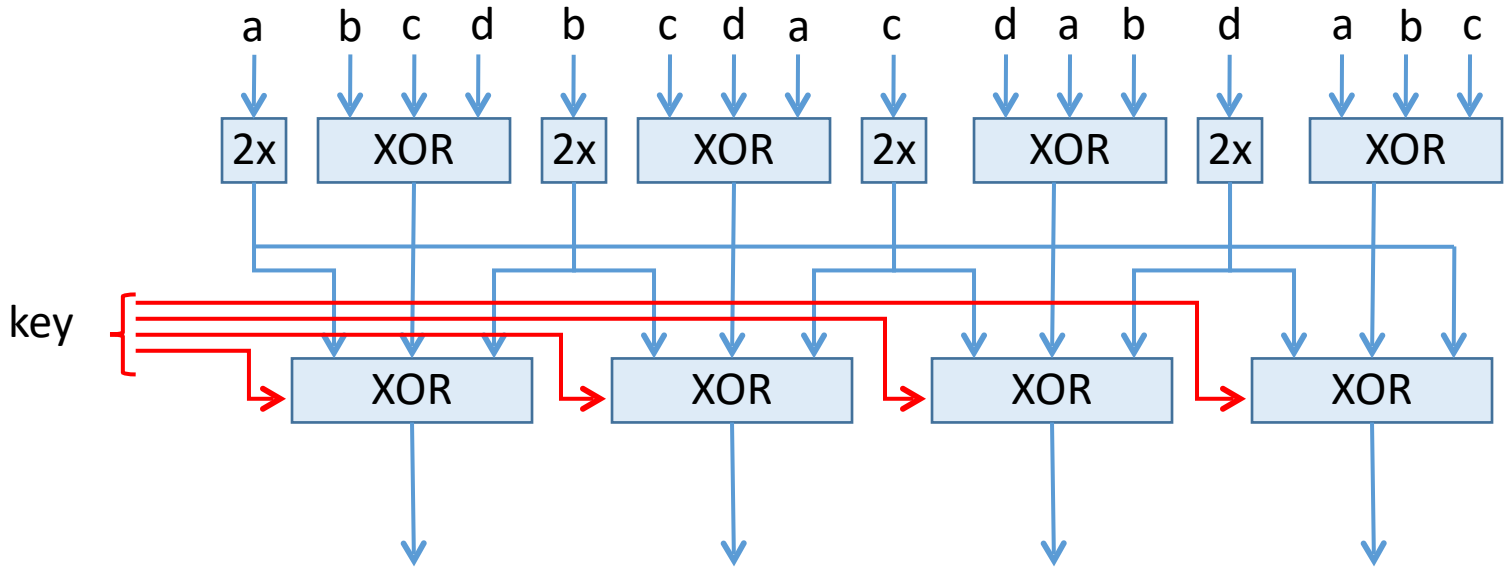

```
# An utility function for computing 02*x in AES finite field
 def xtime(self, x):
    tmp1 = x<<1
    tmp1 = tmp1&0xff
    tmp2 = x>>7
    tmp2 = tmp2&1
    tmp2 = tmp2*0x1b
    val = tmp1^tmp2
    return val
```

Mix Column Implementation

$$A = \{02\}\cdot a \oplus \{03\}\cdot b \oplus \{01\}\cdot c \oplus \{01\}\cdot d$$
$$B = \{01\}\cdot a \oplus \{02\}\cdot b \oplus \{03\}\cdot c \oplus \{01\}\cdot d$$
$$C = \{01\}\cdot a \oplus \{01\}\cdot b \oplus \{02\}\cdot c \oplus \{03\}\cdot d$$
$$D = \{03\}\cdot a \oplus \{01\}\cdot b \oplus \{01\}\cdot c \oplus \{02\}\cdot d$$

Multiplying by 2

$$2 \bullet x = \begin{cases} (x << 1) & when\ MSB(x) = 0 \\ (x << 1)\ \&\ (1B)_{16} & when\ MSB(x) = 1 \end{cases}$$



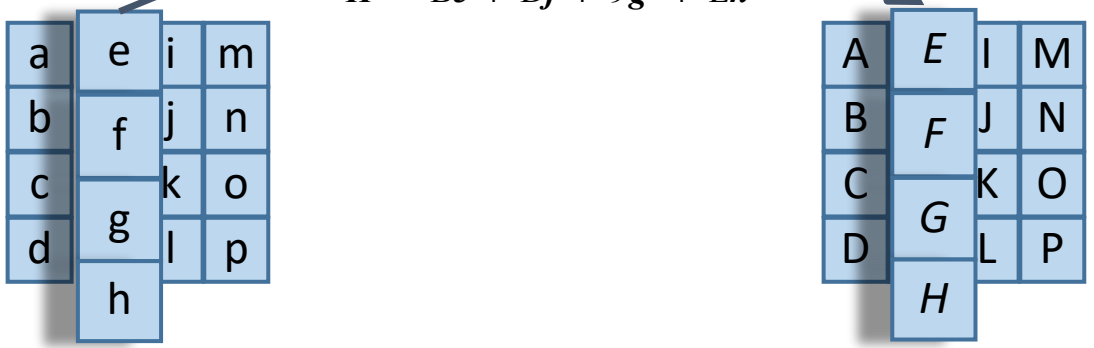
F.R.Henríquez et. l., *Cryptographic Algorithms on Reconfigurable Hardware,* Springer series on Signals and Communication Technology, ISBN 0-387-33883-7

How MixColumns is implemented?

```
# MixColumns Function
def MixColumns(self):
  for i in range(4):
    t = self.state[0][i]
    Tmp = self.state[0][i] ^ self.state[1][i] ^ self.state[2][i] ^ self.state[3][i]
    Tm = self.state[0][i] ^ self.state[1][i] ;
    Tm = self.xtime(Tm)
    self.state[0][i] = self.state[0][i] ^ Tm ^ Tmp

    Tm = self.state[1][i] ^ self.state[2][i]
    Tm = self.xtime(Tm)
    self.state[1][i] = self.state[1][i] ^ Tm ^ Tmp

    Tm = self.state[2][i] ^ self.state[3][i]
    Tm = self.xtime(Tm)
    self.state[2][i] = self.state[2][i] ^ Tm ^ Tmp

    Tm = self.state[3][i] ^ t
    Tm = self.xtime(Tm)
    self.state[3][i] = self.state[3][i] ^ Tm ^ Tmp
```

Inverse Mix Column

$$E = Ee + Bf + Dg + 9h$$
$$F = 9e + Ef + Bg + Dh$$
$$G = De + 9f + Eg + Dh$$
$$H = Be + Df + 9g + Eh$$



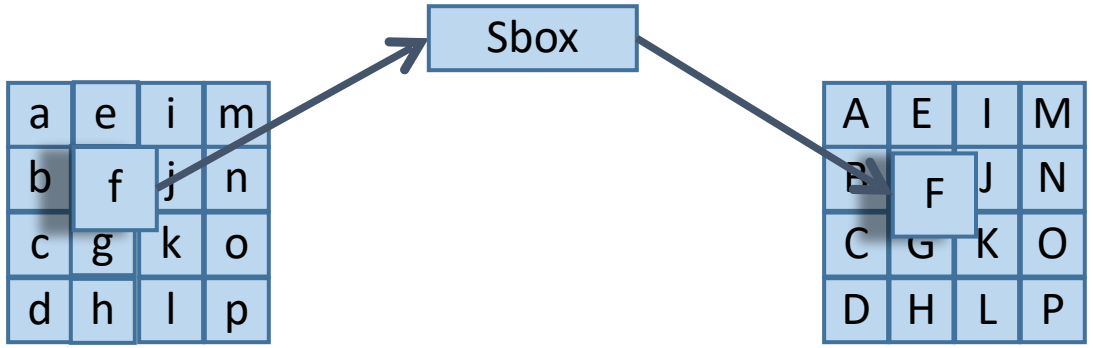
- The 4x4 matrix is multiplied with the matrix

$$\begin{bmatrix} E & B & D & 9 \\ 9 & E & B & D \\ D & 9 & E & B \\ B & D & 9 & E \end{bmatrix}$$

- The hardware implementation can be done in a similar way as mix columns

Byte Substitution

- Makes a non-linear substitution for every byte in the 4x4 matrix



Affine Transformation

$$Sbox(x) = \begin{cases} Affine(x^{-1}) & \text{if } a \neq 0 \\ Affine(0) & \text{if } a = 0 \end{cases}$$

$$\begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

S-box Encryption Table

- Use a table to do the byte substitution
- Eg. **Sbox[42] = 2c**

		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

S-box Mystery

- Use a table to do the byte substitution
  - Is this a great approach?
  - Although, used in many software implementations
  - This has quite a lot of security issues.
  - Also, AES S-Box has a nice math
    - **How to implement this in hardware/software??**

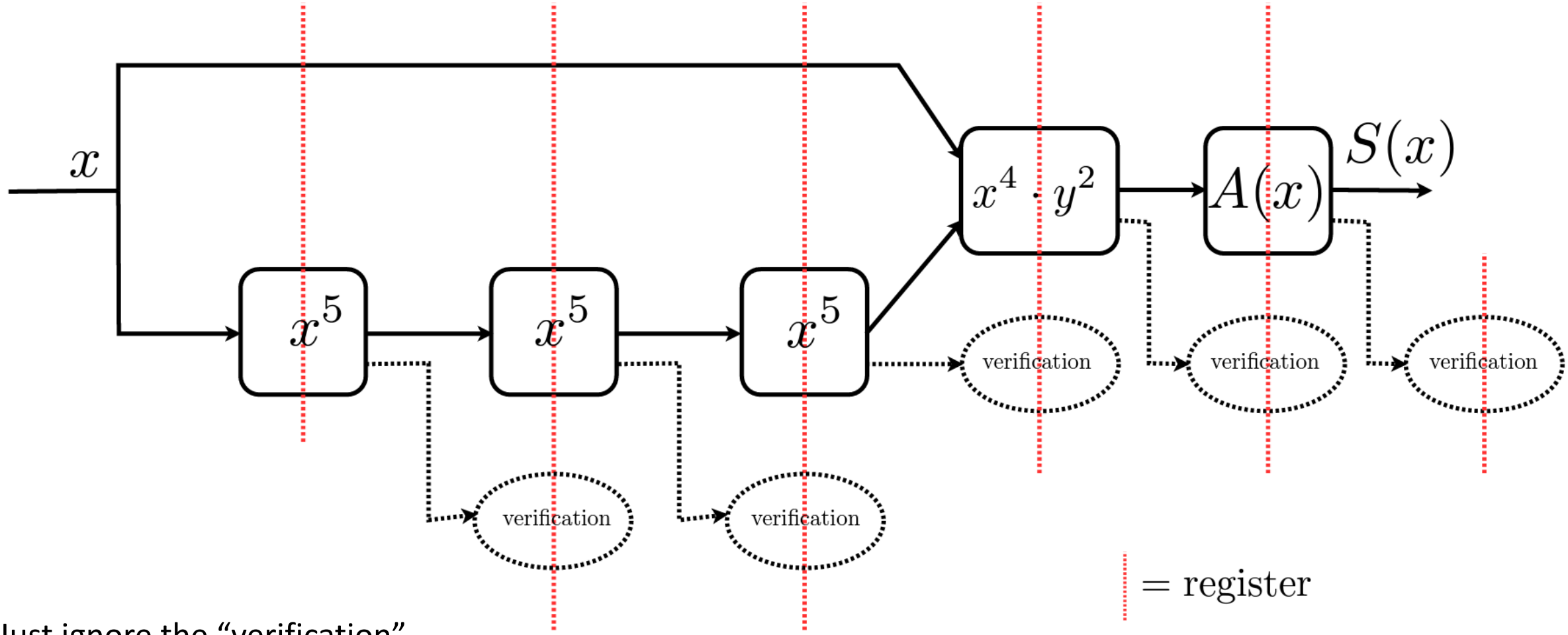
How to Compute Inverse in a Finite Field

- Several ways exist
  - Option 1: Extended Euclidean algorithm
    - Your same old GCD finder with a twist — it can find multiplicative inverse
    - Hardly used
  - Option 2: Any idea? — there was a hint in the previous class
  - Option 3: Exploiting field isomorphisms
    - **The most compact way…**

How to Compute Inverse in a Finite Field

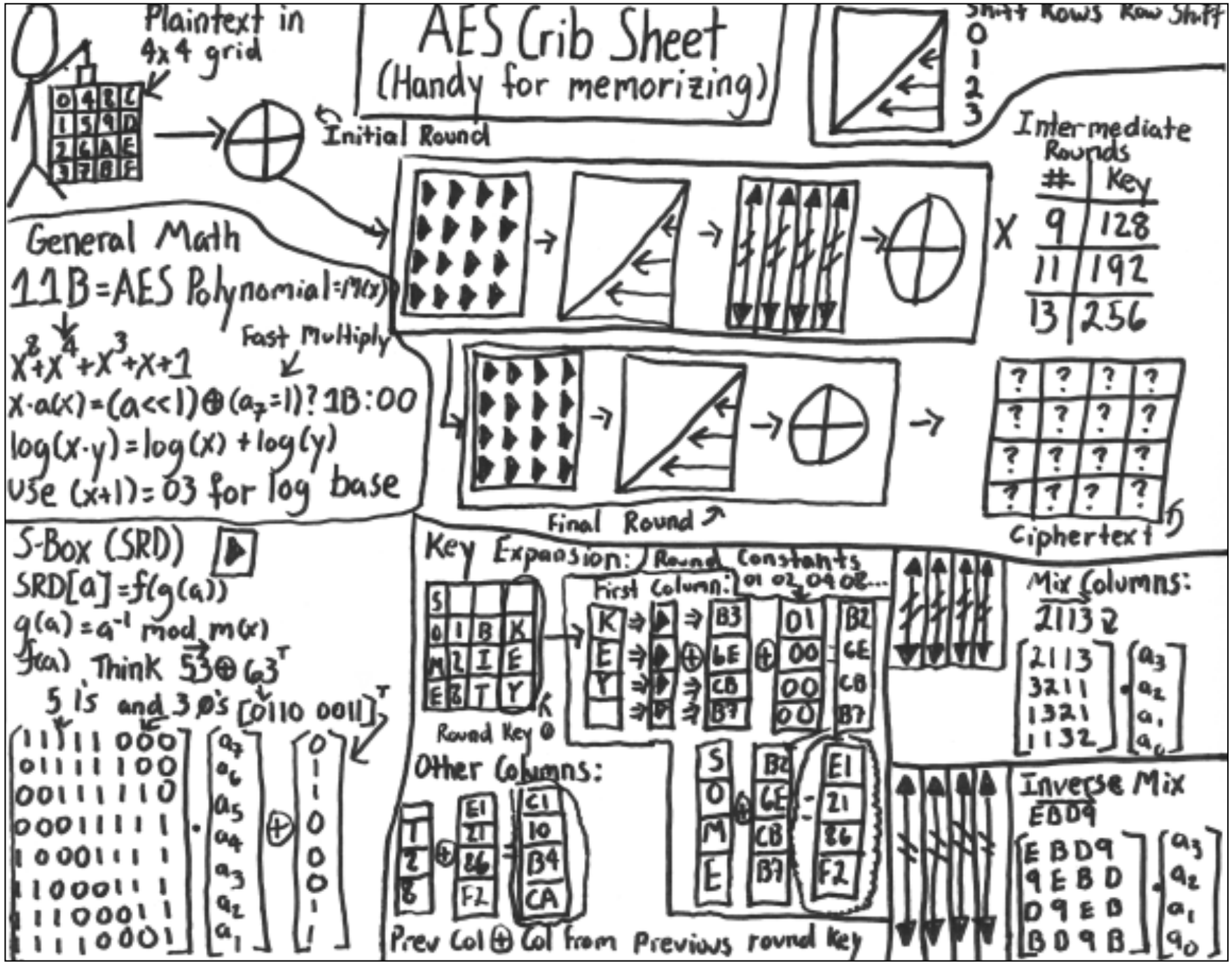
$$a^{256-2} = a^{-1} \ mod \ m(x)$$

S-Box Hardware — One Popular Way



- Just ignore the “verification”

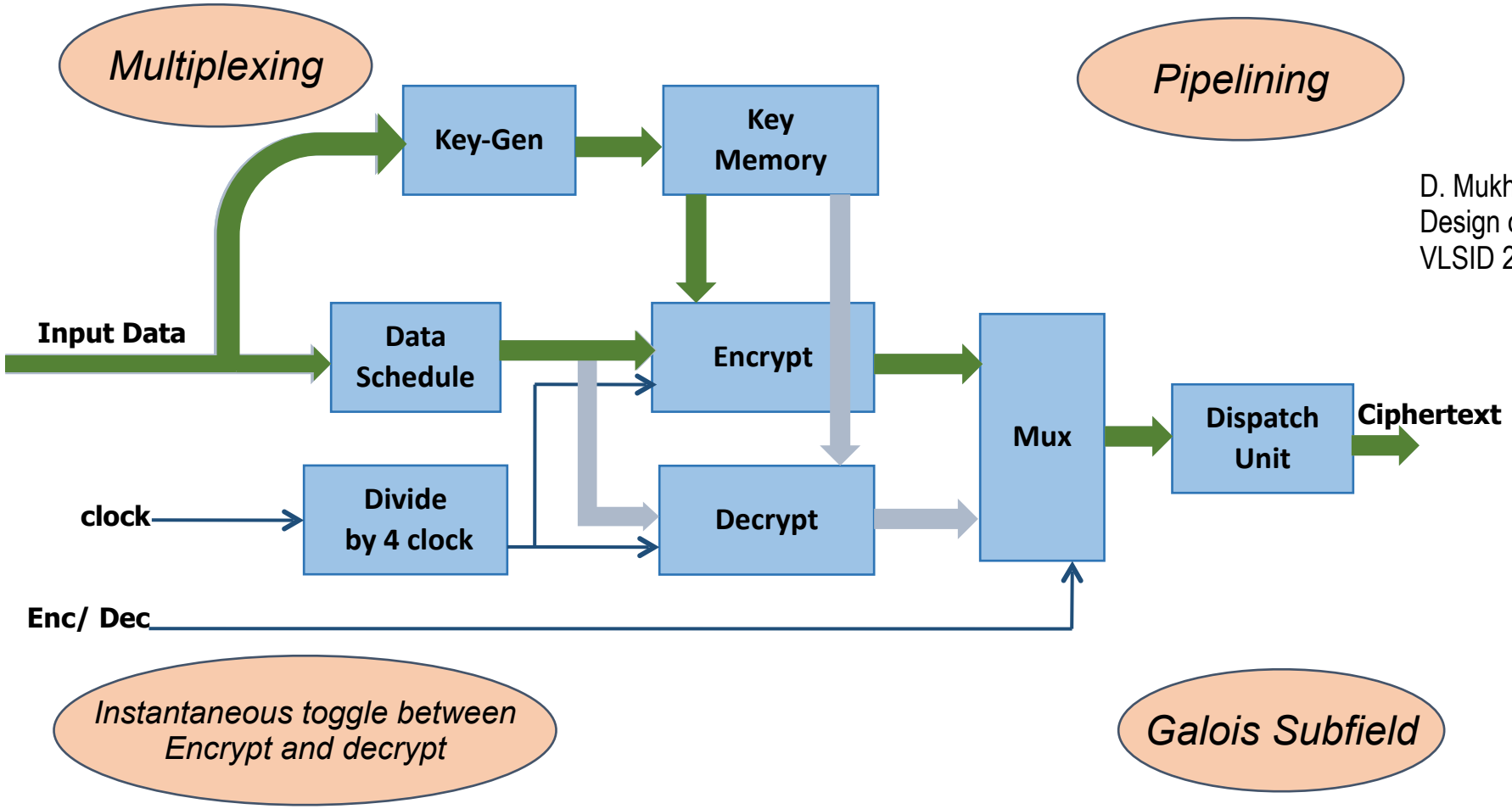
AES Crib Sheet
(Handy for memorizing)



AES Challenges for the Hardware Designer

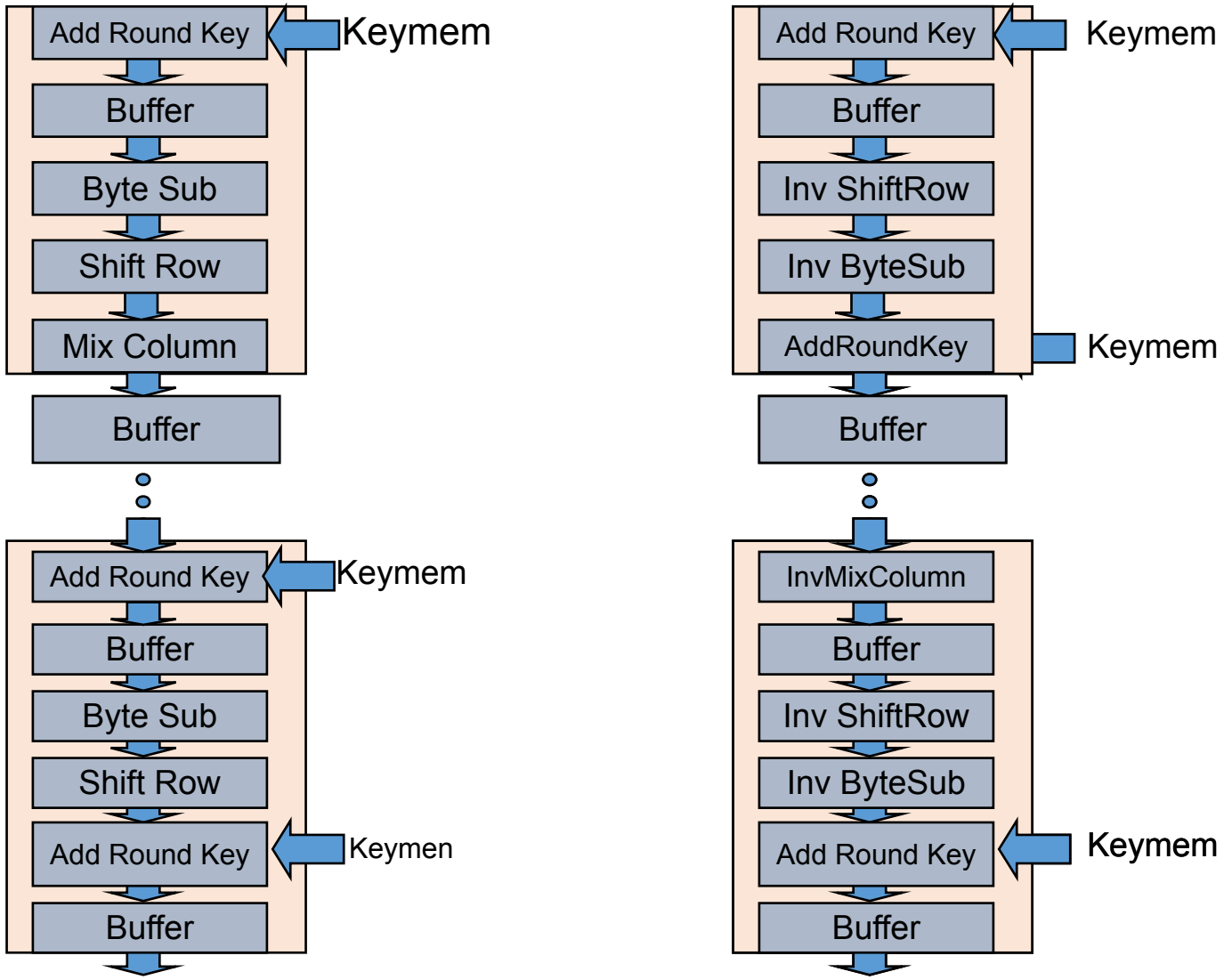
- Reduce area required
- Increase throughput
- A unified AES engine for both encryption and decryption
- Able to handle different key sizes

High Speed AES Encryption-Decryption Architecture

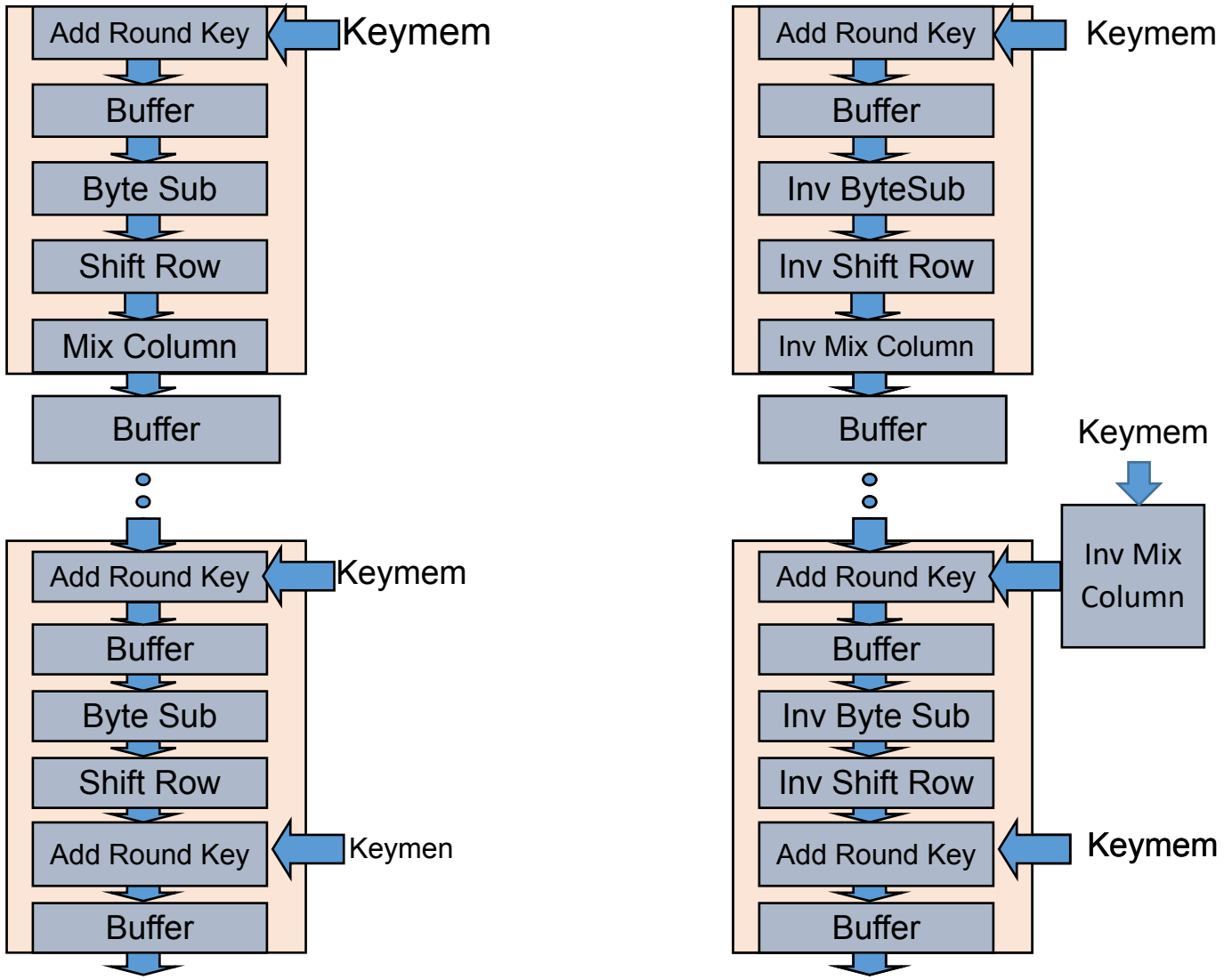


D. Mukhopadhyay et. al, An Efficient End to End Design of Rijndael Cryptosystem in 0.18µ CMOS, VLSID 2005

The Decryption Flow



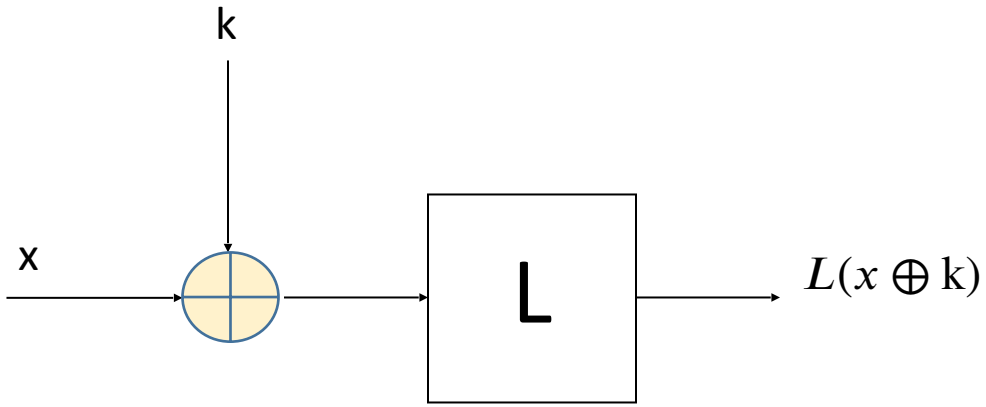
The Decryption Flow

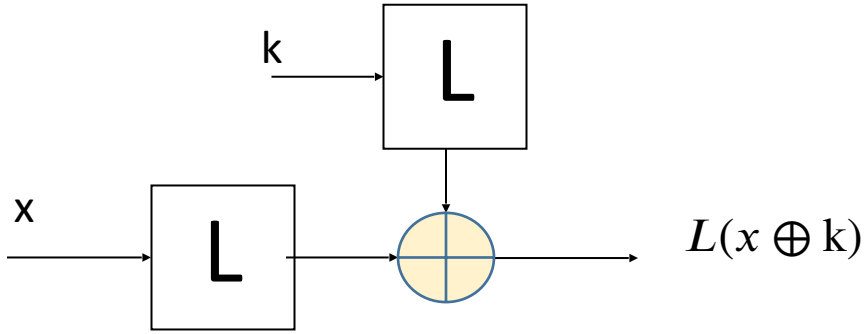


Some Points

- The order of InvShift Rows and InvSubBytes is indifferent.
- The order of AddRoundKey and InvMixColumns can be inverted if the round key is adapted accordingly.

A Linear transformation can be pushed through an XOR





Encryption steps for two round AES variant

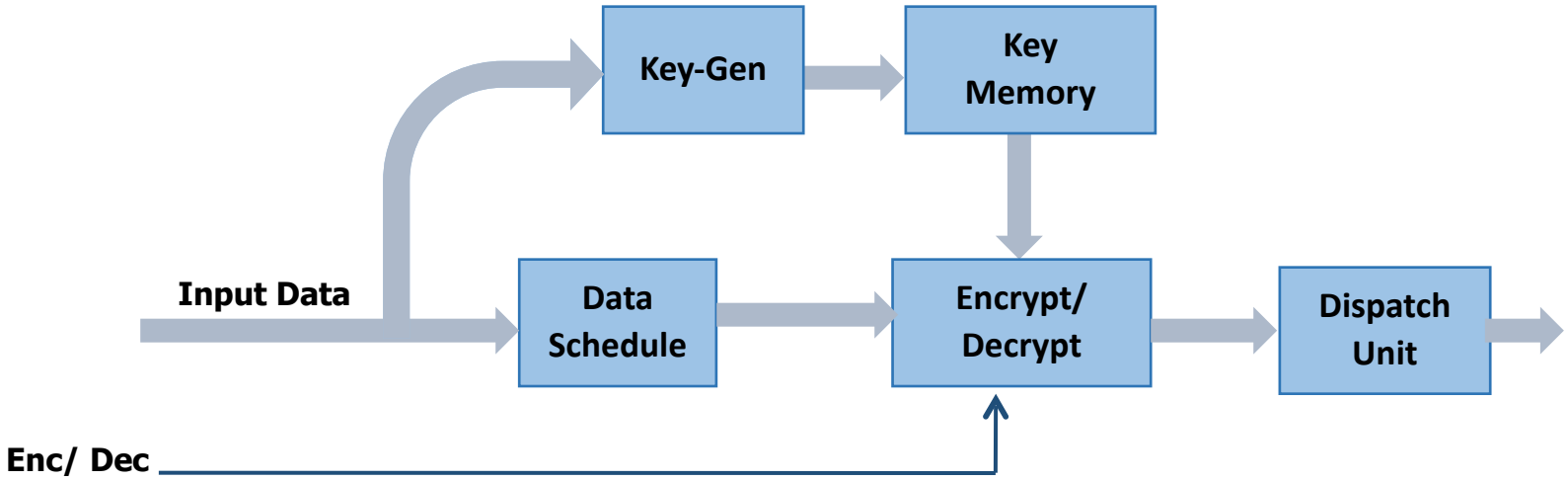
- AddRoundKey(State, ExpandedKey[0]);
- SubBytes(State);
- ShiftRow(State);
- MixColumn(State);
- AddRoundKey(State, ExpandedKey[1]);
- SubBytes(State);
- ShiftRow(State);
- AddRoundKey(State, ExpandedKey[2]);

Equivalent Decryption steps for two round AES variant

- AddRoundKey(State, ExpandedKey[2]);
- InvSubBytes(State);
- InvShiftRow(State);
- InvMixColumn(State);
- AddRoundKey(State, EqExpandedKey[1]);
- InvSubBytes(State);
- InvShiftRow(State);
- AddRoundKey(State, ExpandedKey[0]);

Compact Encryption-Decryption Architecture

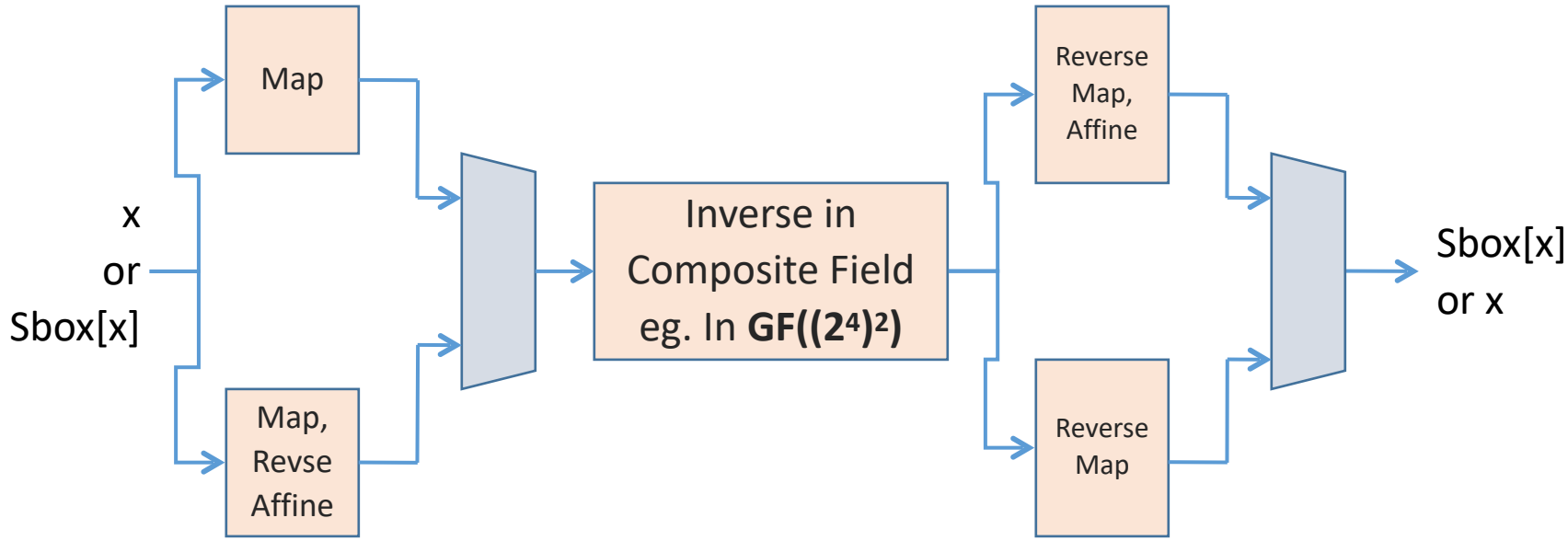
- Merge the operations in the encryption and decryption into a single unit.



Merged Round Operations

- *Addroundkey* is the same in both encryption and decryption
- *ShiftRows* and *InverseShiftRows* only differ in the direction of shifting

Merged S-boxes



Type	Total Gates in terms of NAND gates using std cell library
Sbox + InvSbox	362
Merged	234

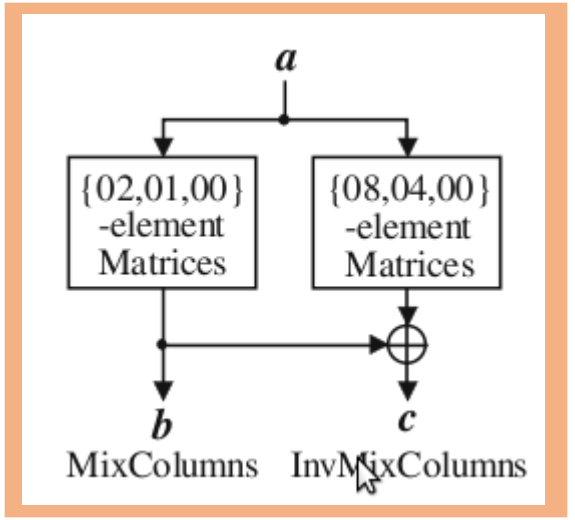
D. Canright, *A Very Compact S-box for AES*, CHES-2005

Merged Mix Columns

$$\begin{pmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} = \begin{pmatrix} 0E\ 0B\ 0D\ 09 \\ 09\ 0E\ 0B\ 0D \\ 0D\ 09\ 0E\ 0B \\ 0B\ 0D\ 09\ 0E \end{pmatrix} \cdot \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix}$$

$$= \begin{pmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{pmatrix} \cdot \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix}$$

$$+ \begin{pmatrix} 08\ 08\ 08\ 08 \\ 08\ 08\ 08\ 08 \\ 08\ 08\ 08\ 08 \\ 08\ 08\ 08\ 08 \end{pmatrix} \cdot \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} + \begin{pmatrix} 04\ 00\ 04\ 00 \\ 00\ 04\ 00\ 04 \\ 04\ 00\ 04\ 00 \\ 00\ 04\ 00\ 04 \end{pmatrix} \cdot \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix}$$



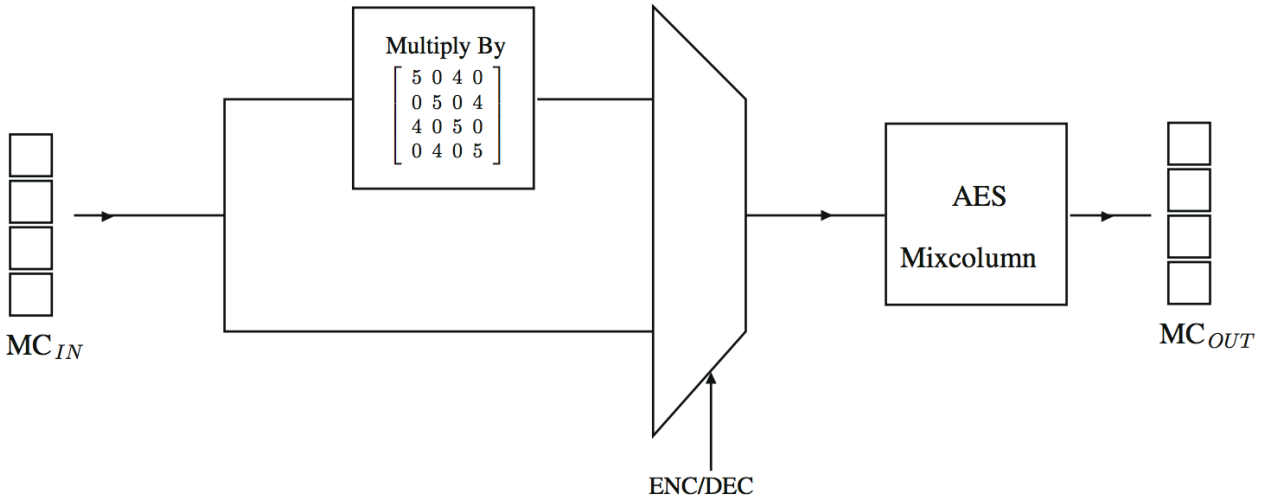
	Separate Mix (Inv) Columns	Merged Mix Columns
No. of XOR gates	152 + 440 = 592	195
Delay (gates)	5	7

A. Satoh et. al., *A compact Rijndael Hardware Architecture with S-box Optimization,* ASIACRYPT 2001

Another InvMixColumn Design

$$\begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} 05 & 00 & 04 & 00 \\ 00 & 05 & 00 & 04 \\ 04 & 00 & 05 & 00 \\ 00 & 04 & 00 & 05 \end{pmatrix}$$

Gate-count around 166 XORs and a 32-bit MUX.



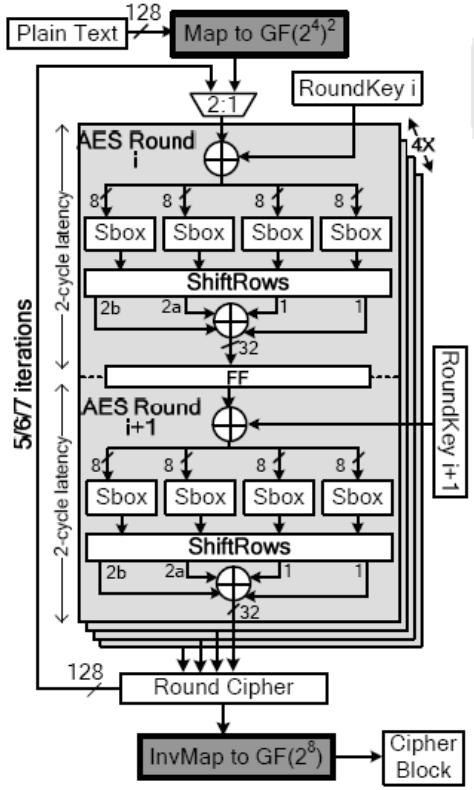
Subhadeep Banik, Andrey Bogdanov, Francesco Regazzoni:Atomic-AES: A Compact Implementation of the AES Encryption/Decryption Core. INDOCRYPT 2016: 173-190

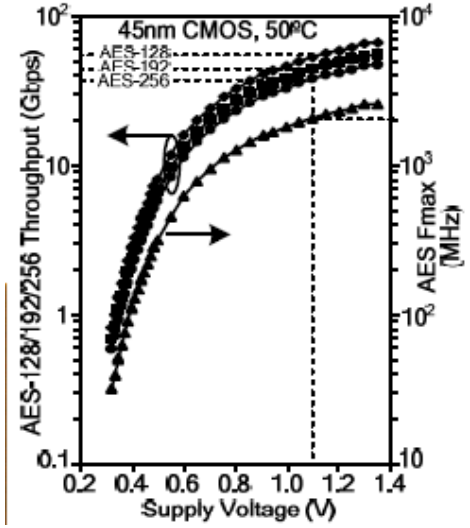
State of the Art

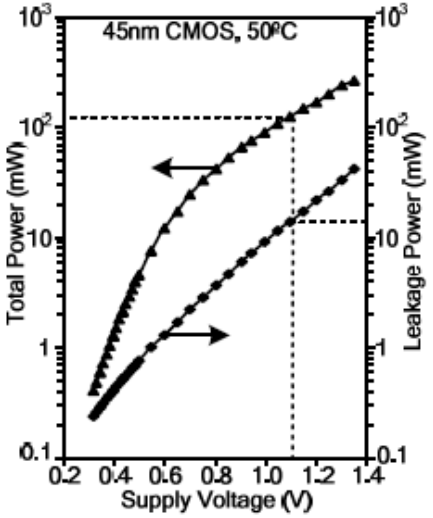
- Intel has introduced dedicated hardware for AES on its microprocessors
- These are accessed by dedicated AES instructions
- The AES design consists of:
  - Standard techniques used such as composite fields, single map – reverse map, etc.
  - Speed of 53Gbps obtained by fully custom design flow on 45nm technology.

S. Mathew, *53Gbps Native GF(2^4)^2 Composite Field AES-Encrypt/Decrypt Accelerator for Content Protection in 45nm High Performance Microprocessors,*VTS 2010

Intel's AES Hardware







S. Mathew, *53Gbps Native GF(2^4)^2 Composite Field AES-Encrypt/Decrypt Accelerator for Content Protection in 45nm High Performance Microprocessors,* VTS 2010