Hochschule Konstanz
Fakultät Informatik

Introduction to IT Security

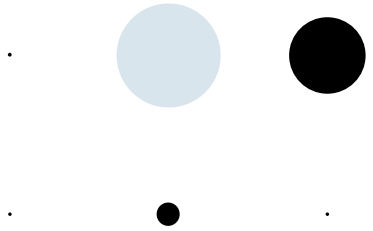
WIN+AIN

Hanno Langweg

05 Cryptographic Primitives and Algorithms

Goals of cryptography

- Protection of data in transfer over insecure channel
- Protection of data in storage on untrusted media
- Confidentiality (prevent attacks)
- Integrity (detect attacks)
- Authenticity, origin of data (detect attacks)



Perfect confidentiality

- There is perfect encryption (unbreakable): the one-time pad
  - Random key
  - Key length equal to plaintext length
  - XOR plaintext with key
  - Ciphertext looks completely random
- The problem:
  - Key length
  - Key distribution
- Not often used, e.g. some diplomatic applications, ANC Operation Vula (1988++)

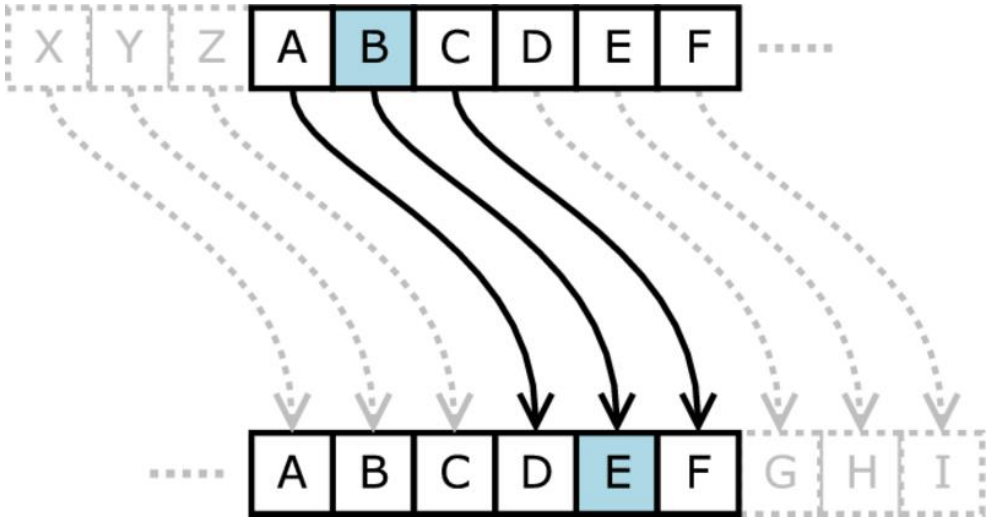
https://en.wikipedia.org/wiki/File:Red_phone.jpg

Some history

- Scytale - Greece ca. 700 B.C.
- Transposition/diffusion
- Confidentiality, authentication (same diameter?)

https://en.wikipedia.org/wiki/Scytale#/media/File:Skytale.png

Some history

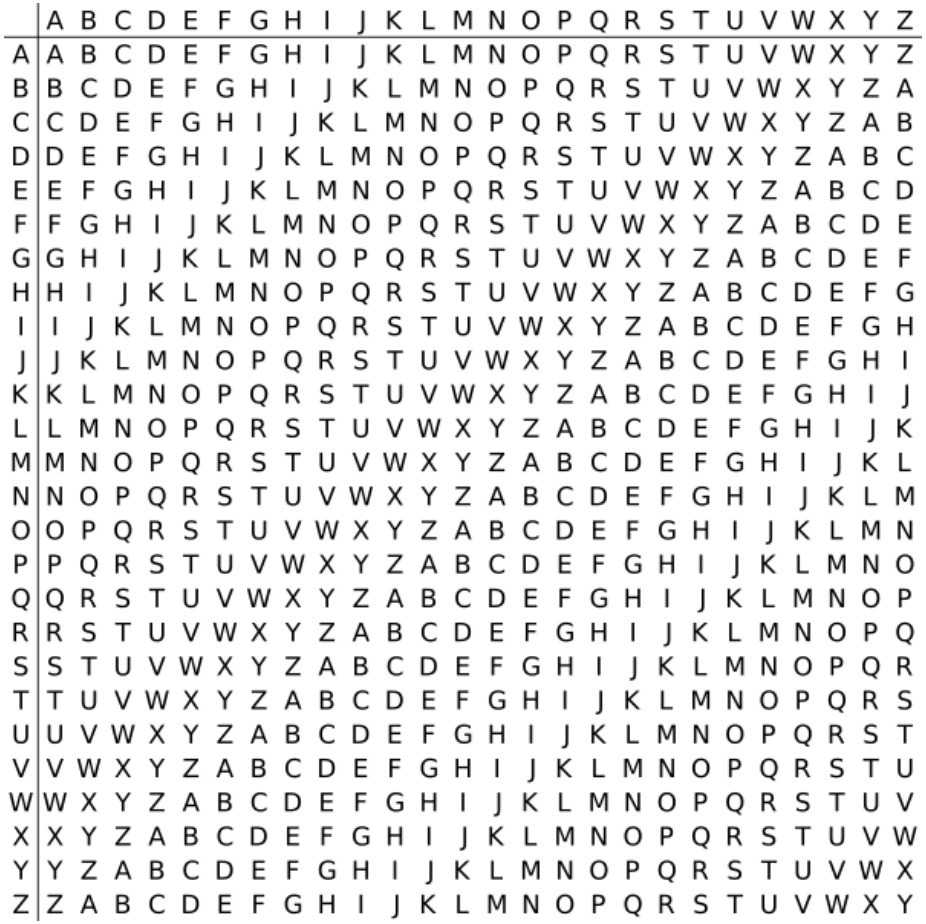


- Caesar cipher
- Monoalphabetic substitution
- Attackable by brute force, frequency analysis

https://commons.wikimedia.org/wiki/File:Caesar3.svg

Some history

- Vigenère cipher ca. 1500s
- Polyalphabetic substitution
- Attackable by frequency analysis, known plaintext parts



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Some history

- Enigma 1920s-1940s
- Polyalphabetic substitution
- Attackable by predictable plaintext parts, reduced number of rotor combinations

http://users.telenet.be/d.rijmenants/en/enigma.htm

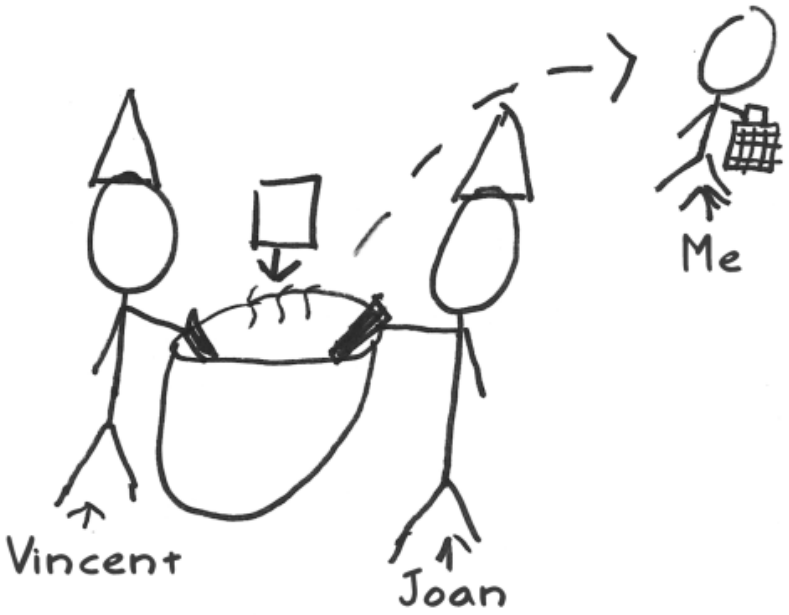
AES
Advanced Encryption Standard

AES Advanced Encryption Standard

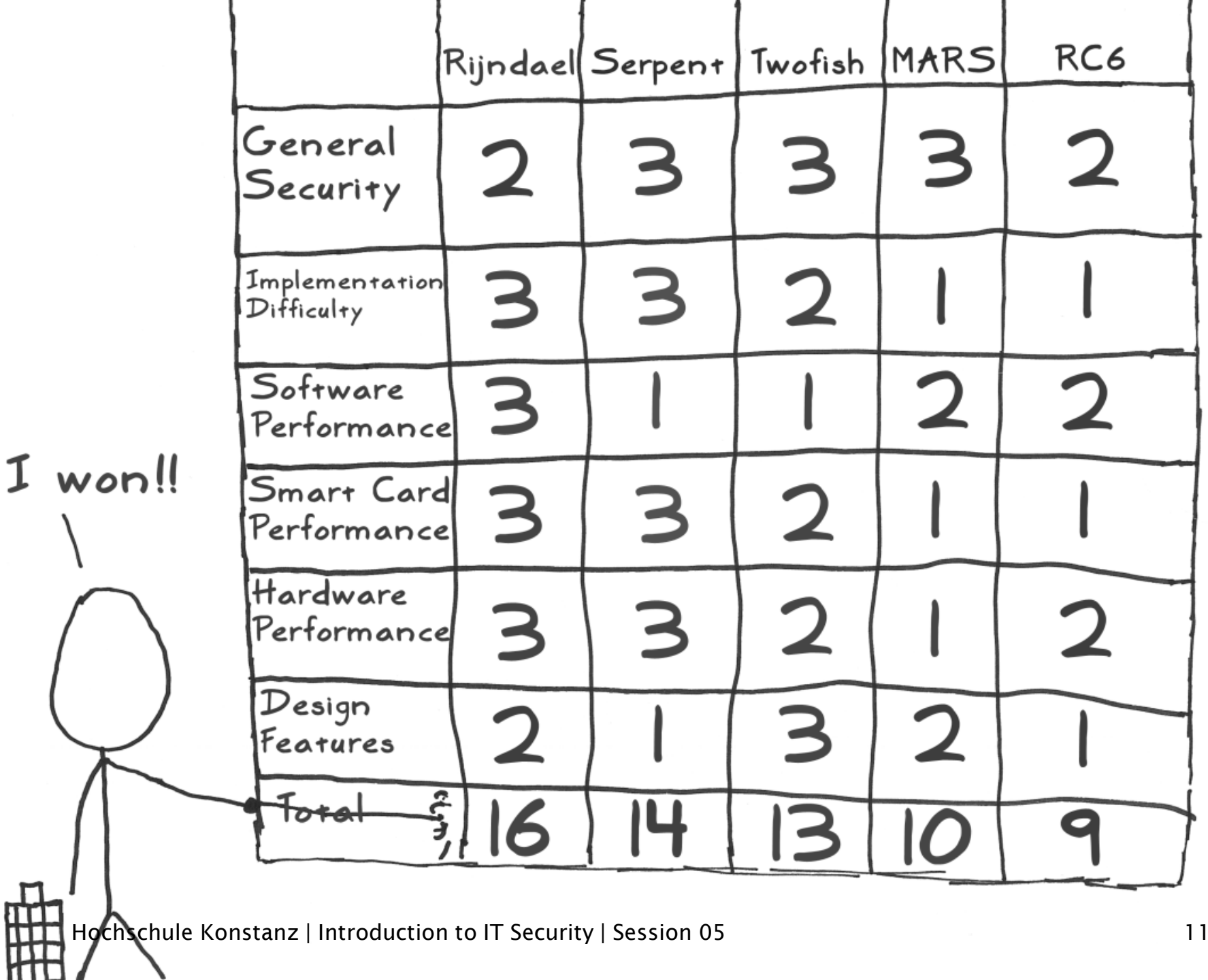
- **Symmetric** block cipher
  - Same **key** for encryption/decryption
  - **Block size** 128 bits (=16 bytes)
  - Key lengths 128, 192, 256 bits
- Selected by U.S. NIST National Institute of Standards and Technology, FIPS PUB 197, ISO/IEC 18033-3

- Public selection process 1997-2001
- Needed as successor to DES (broken), 3DES (slow)

My creators, Vincent Rijmen and Joan Daemen, were among these crypto wizards. They combined their last names to give me my birth name: Rijndael.*



* That's pronounced "Rhine Dahl" for the non-Belgians out there.



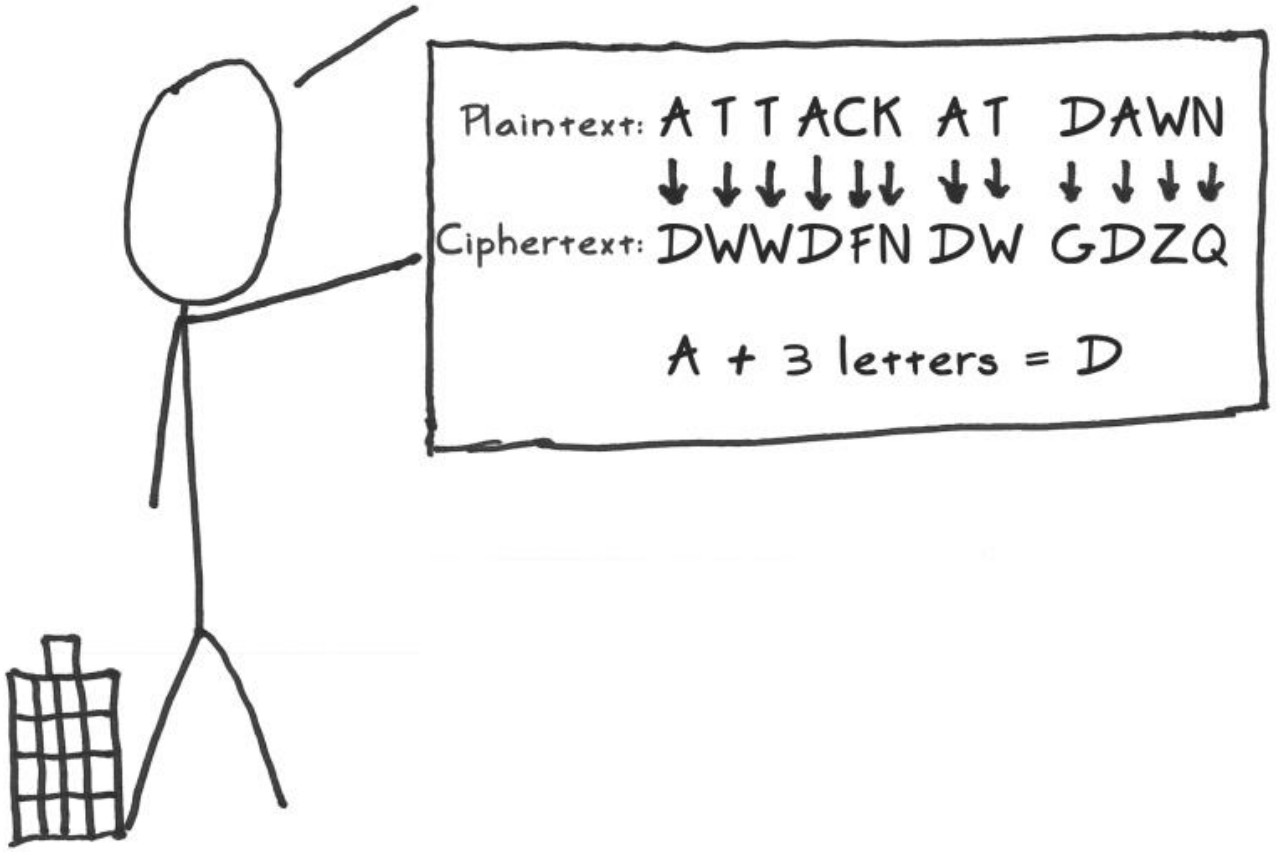
	Rijndael	Serpent	Twofish	MARS	RC6
General Security	2	3	3	3	2
Implementation Difficulty	3	3	2	1	1
Software Performance	3	1	1	2	2
Smart Card Performance	3	3	2	1	1
Hardware Performance	3	3	2	1	2
Design Features	2	1	3	2	1
~~Total~~	16	14	13	10	9

3 main principles

- Confusion
  - Little relationship between original message (plaintext) and encrypted message (ciphertext)
- Diffusion
  - Changes in plaintext should affect large parts of ciphertext
  - Ideally: 1 bit change in plaintext ➔ 50% changes in ciphertext
- Secrecy only in the key
  - Algorithm should be secure even when published

Big Idea #1: Confusion

It's a good idea to obscure the relationship between your real message and your "encrypted" message. An example of this "confusion" is the trusty ol' Caesar Cipher:

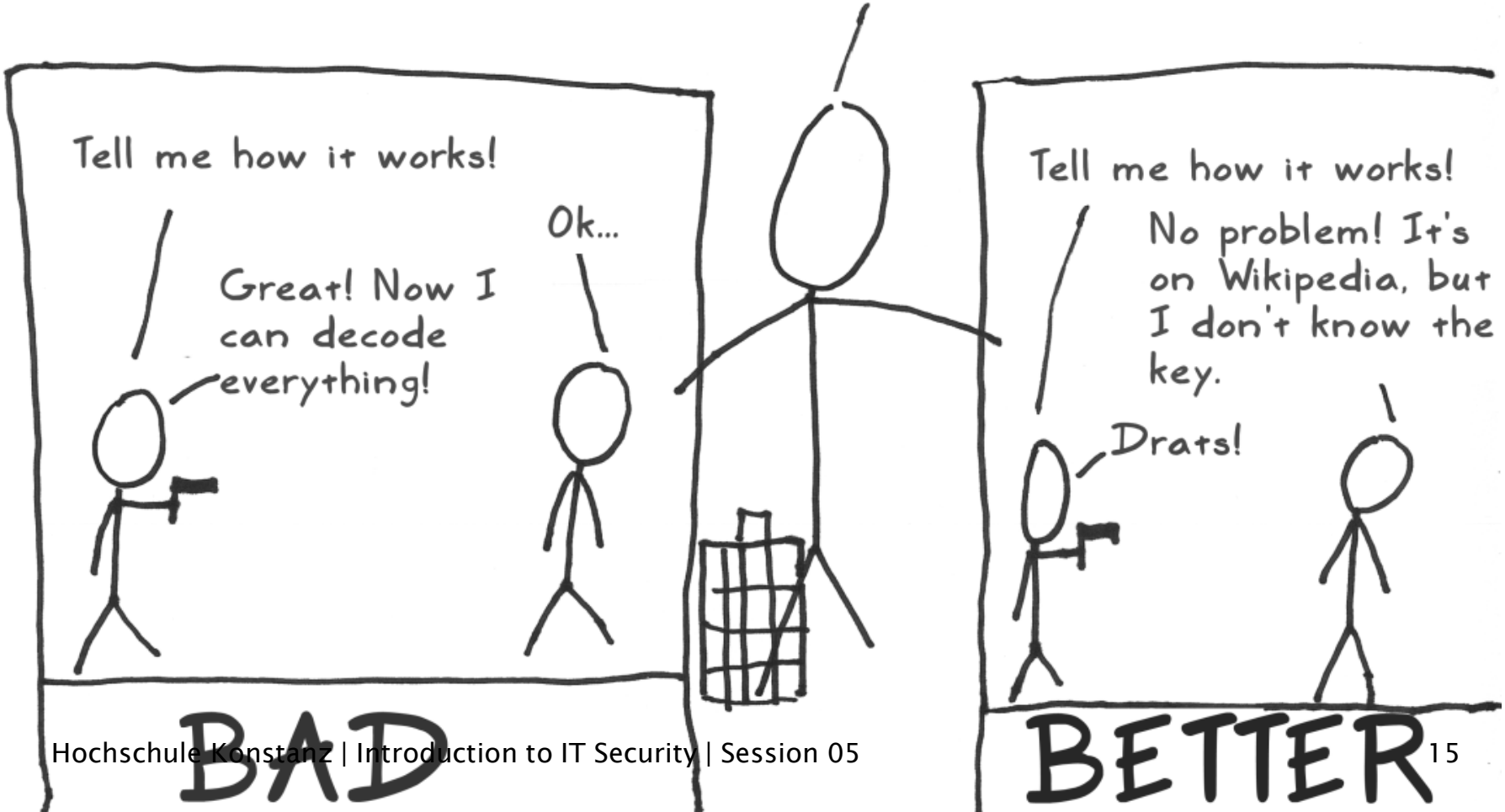


Big Idea #2: Diffusion

It's also a good idea to spread out the message. An example of this "diffusion" is a simple column transposition:

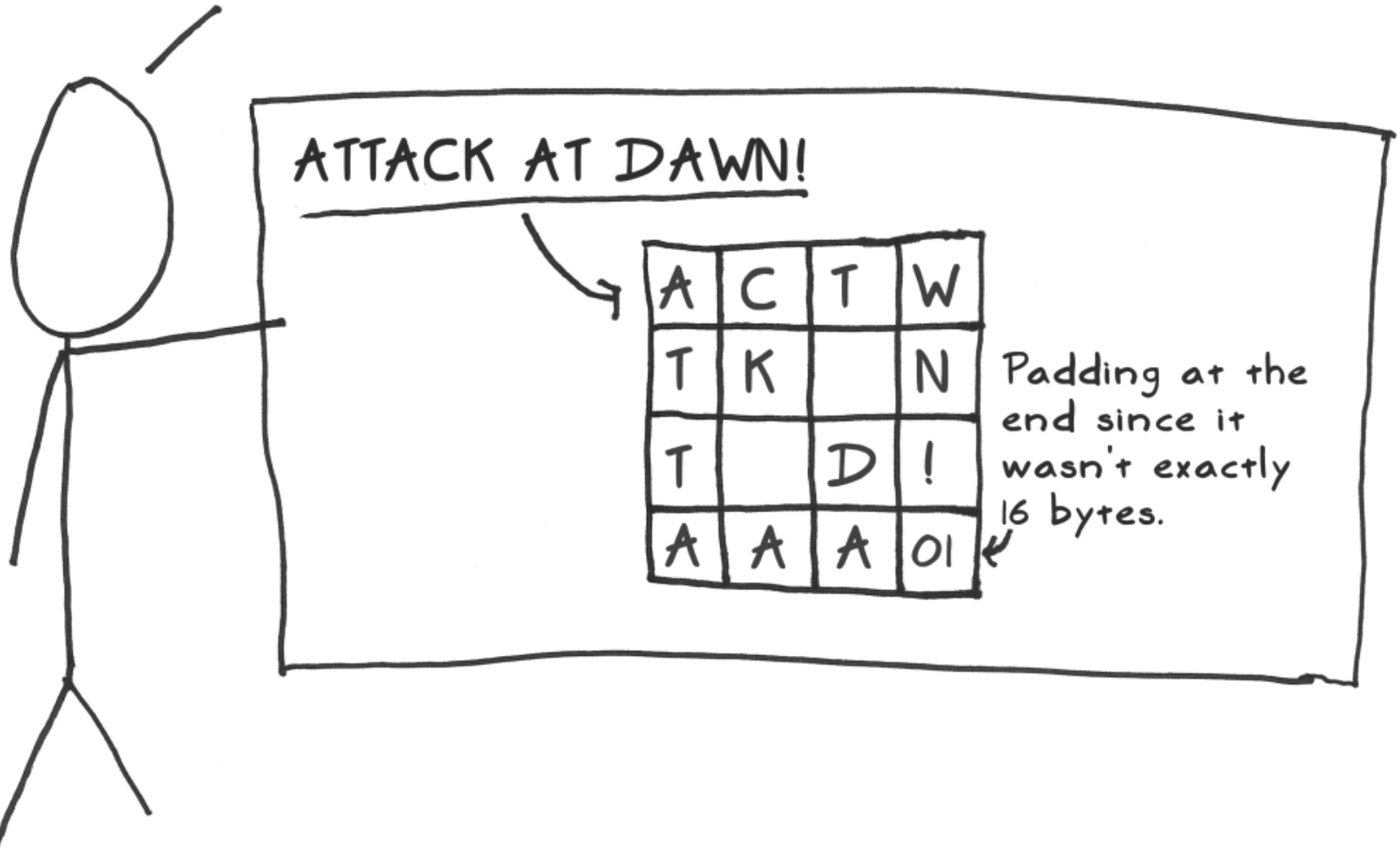
Big Idea #3: Secrecy Only in the Key

After thousands of years, we learned that it's a bad idea to assume that no one knows how your method works. Someone will eventually find that out.



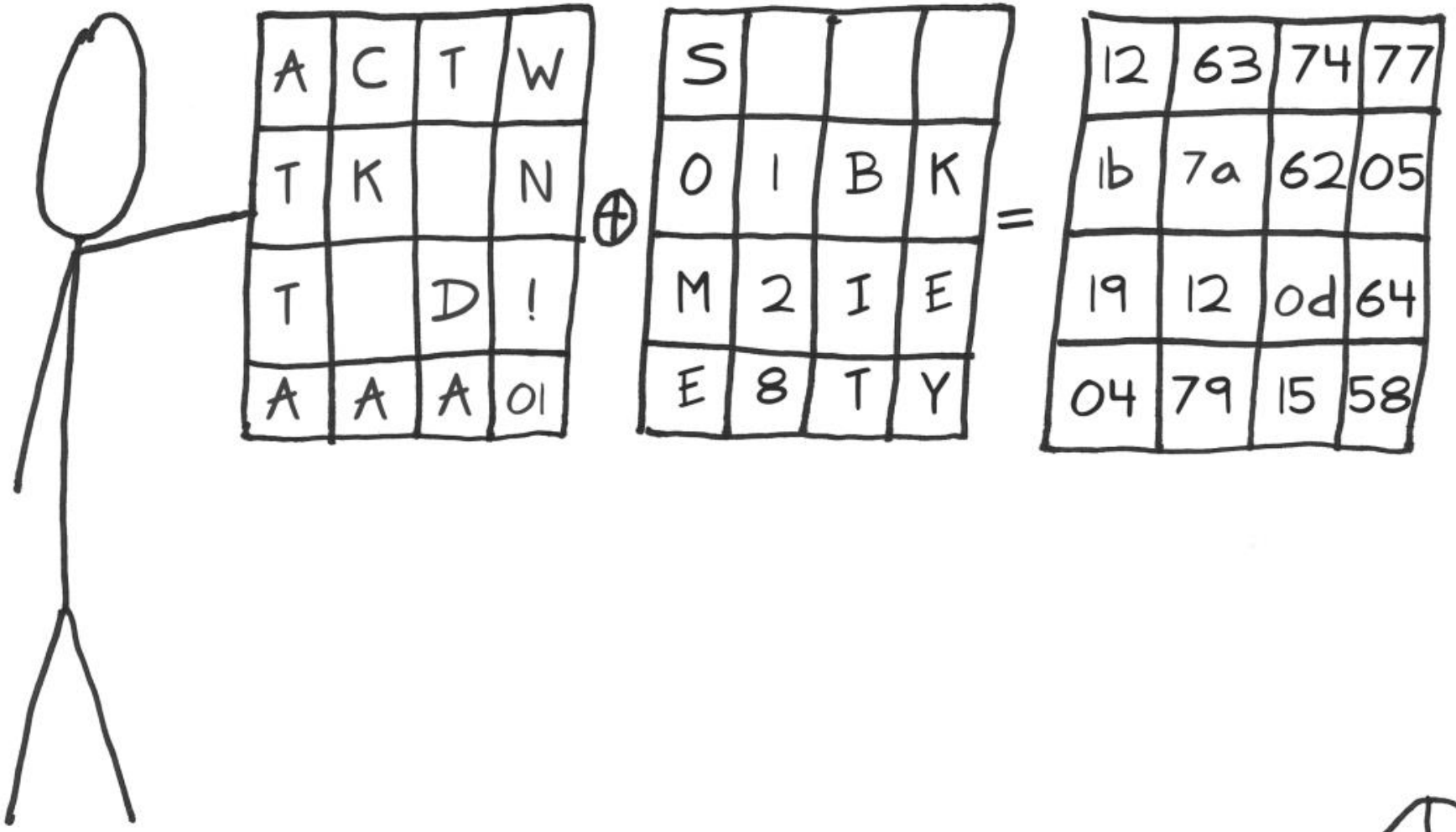
AES: Details

I take your data and load it into this 4x4 square.*



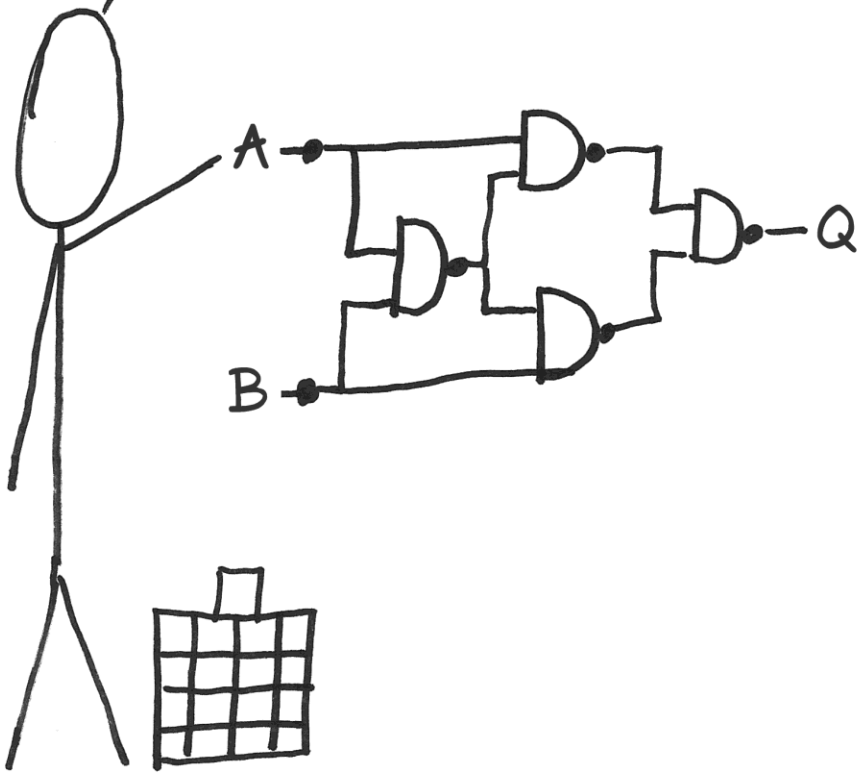
* This is the "state matrix" that I carry with me at all times.

The initial round has me xor each input byte with the corresponding byte of the first round key.



A Tribute to XOR

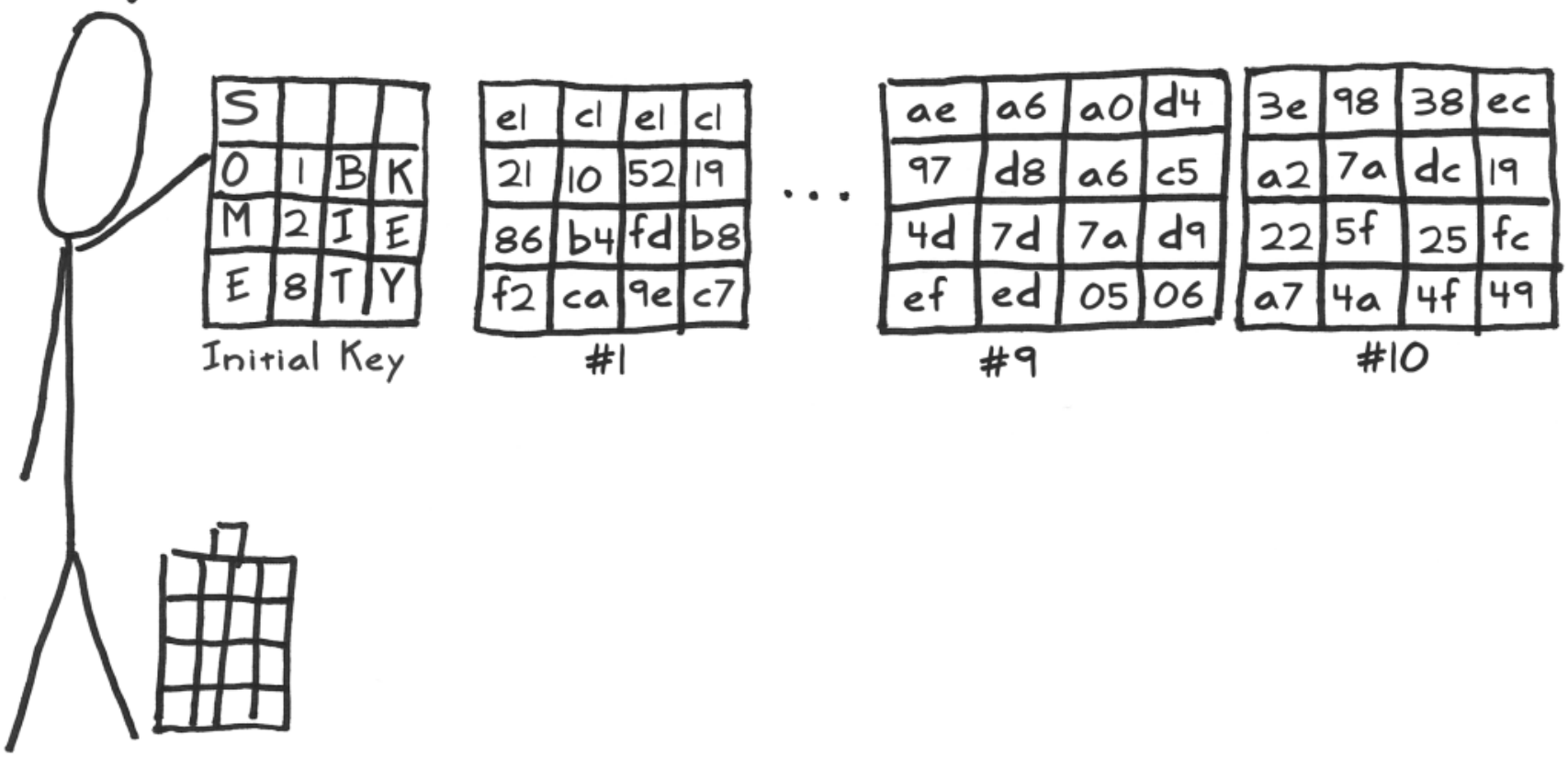
There's a simple reason why I use xor to apply the key and in other spots: it's fast and cheap – a quick bit flipper. It uses minimal hardware and can be done in parallel since no pesky "carry" bits are needed.



AES ♡ ⊕

Key Expansion: Part 1

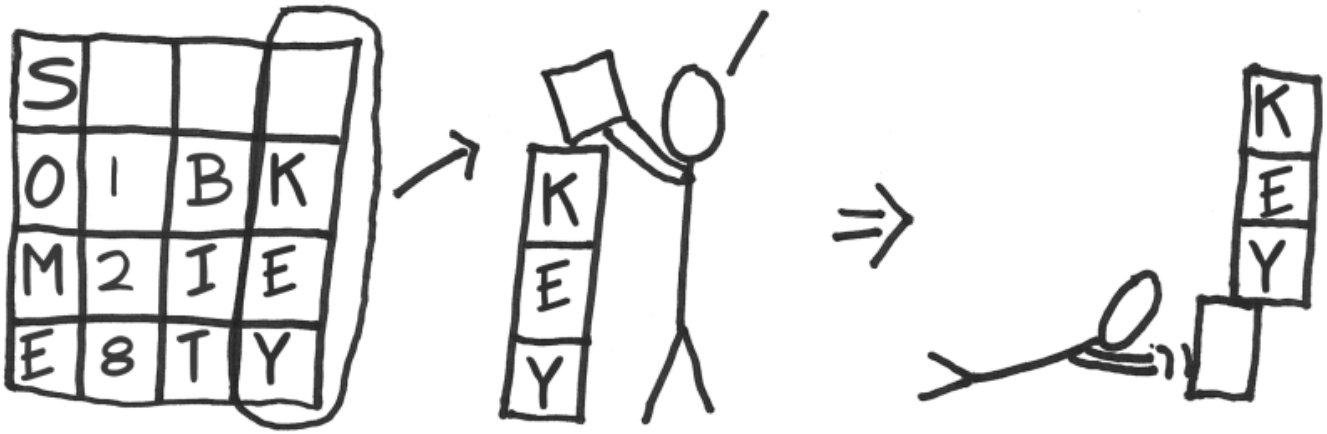
I need lots of keys for use in later rounds. I derive all of them from the initial key using a simple mixing technique that's really fast. Despite its critics,* it's good enough.



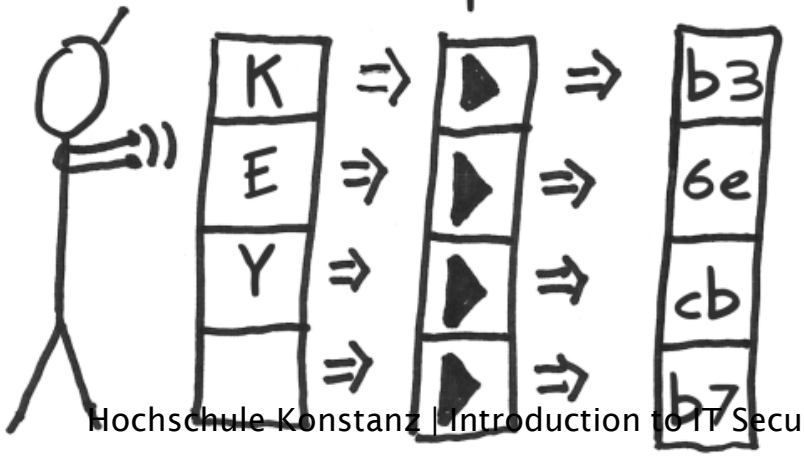
* By far, most complaints against AES's design focus on this simplicity.

Key Expansion: Part 2a

(1) I take the last column of the previous round key and move the top byte to the bottom:



(2) Next, I run each byte through a substitution box that will map it to something else:



		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

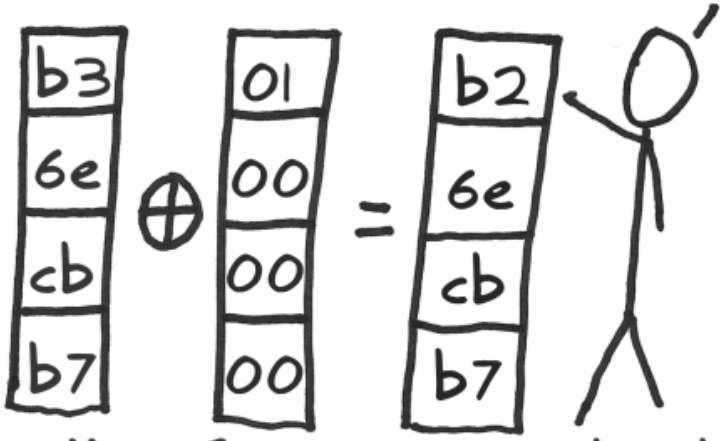
Stallings/Brown table 20.2a AES S-box

		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

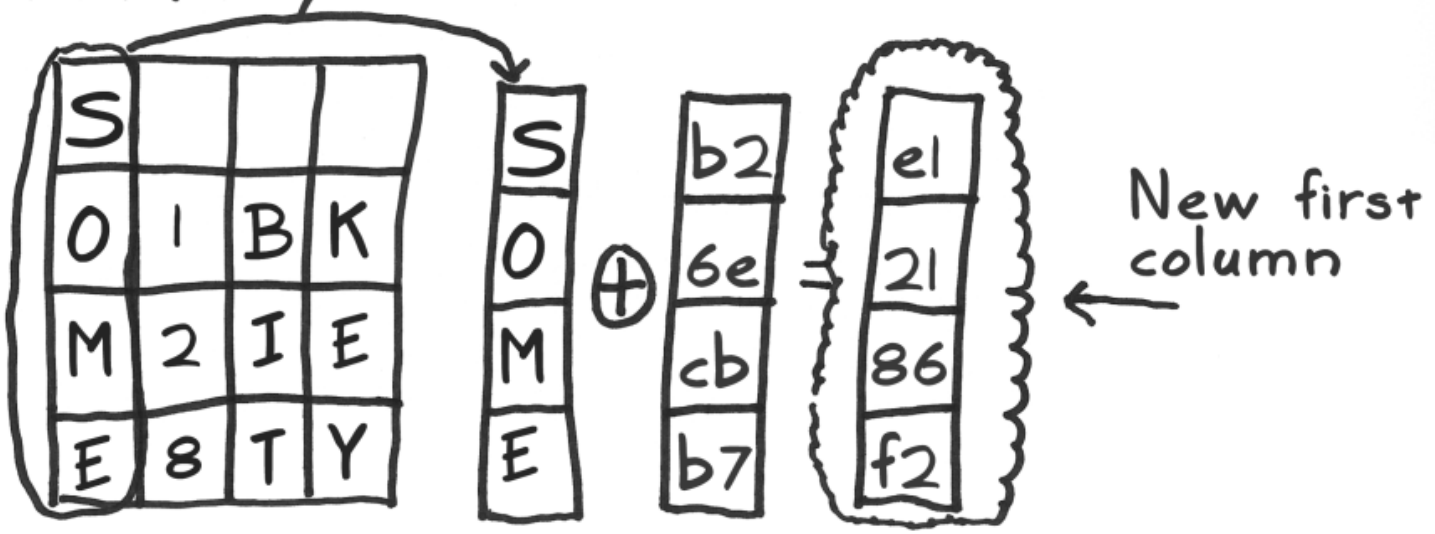
Stallings/Brown table 20.2b AES inverse S-box

Key Expansion: Part 2b

(3) I then xor the column with a "round constant" that is different for each round.

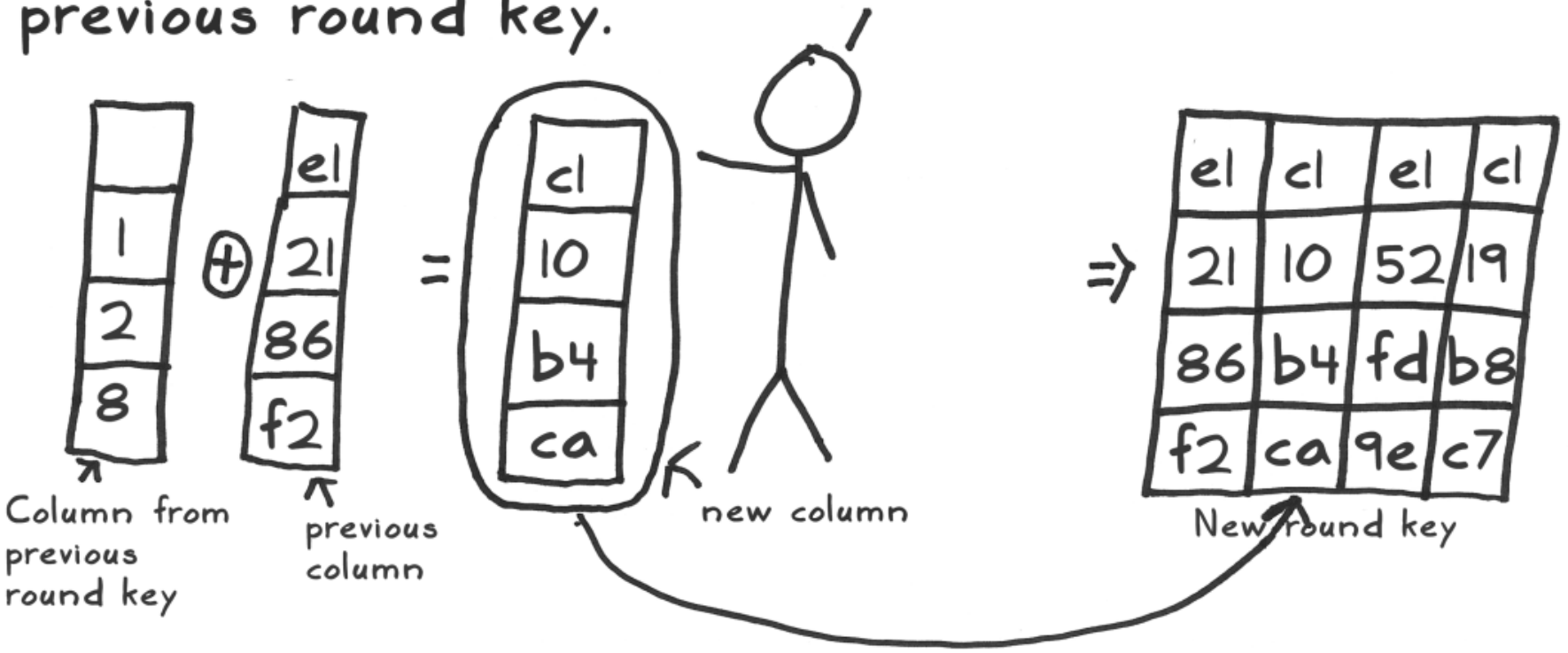


(4) Finally, I xor it with the first column of the previous round key:



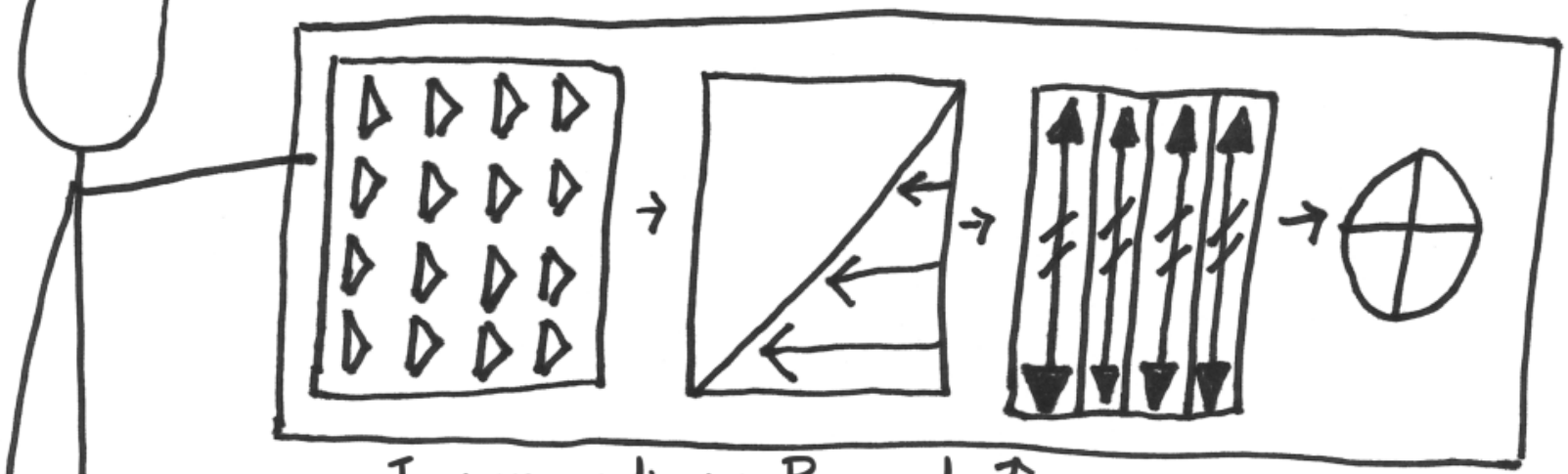
Key Expansion: Part 3

The other columns are super-easy,* I just xor the previous column with the same column of the previous round key.

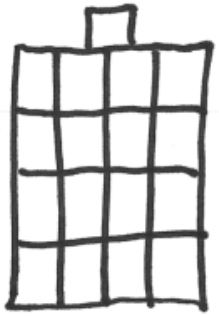


* Note that 256 bit keys are slightly more complicated.

Next, I start the intermediate rounds. A round is just a series of steps I repeat several times. The number of repetitions depends on the size of the key.



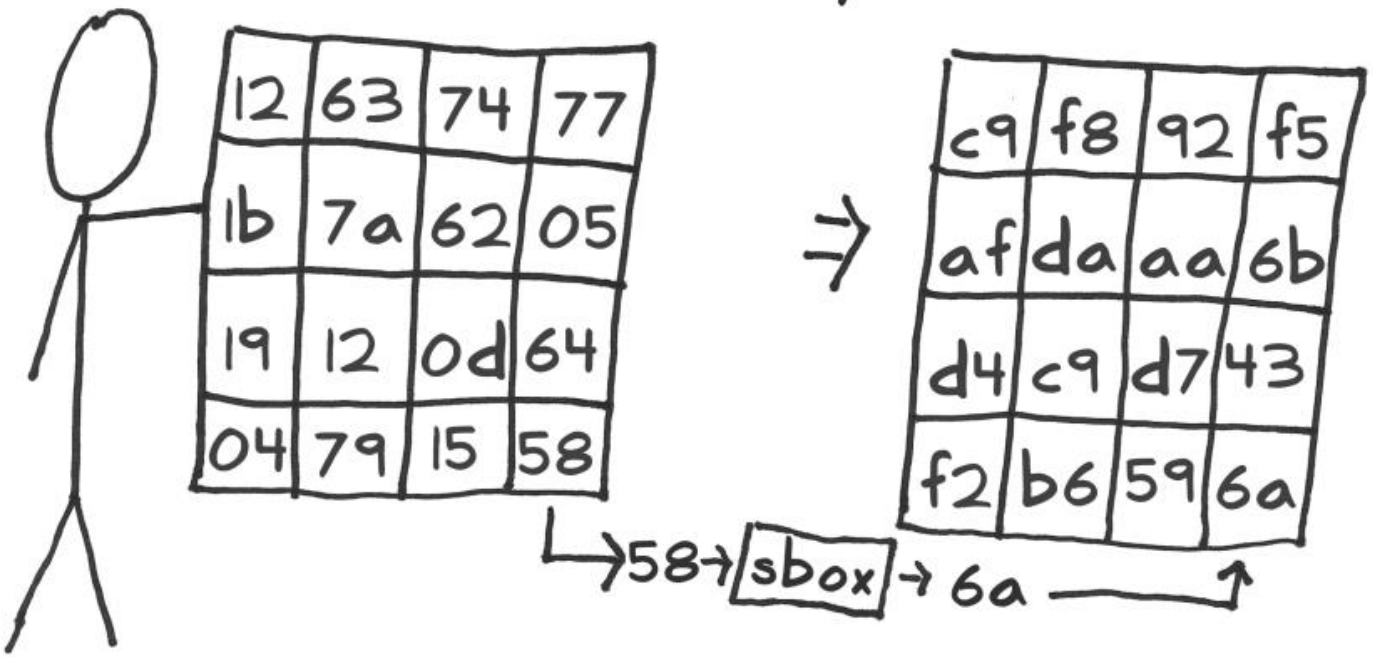
Intermediate Round

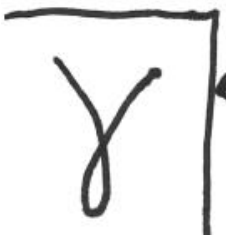


Round Repetitions	Key Size
9	128
11	192
13	256

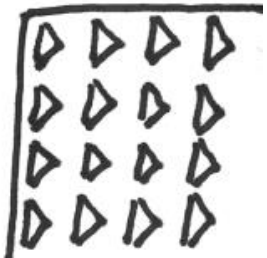
Applying Confusion: Substitute Bytes

I use confusion (Big Idea #1) to obscure the relationship of each byte. I put each byte into a substitution box (sbox), which will map it to a different byte:

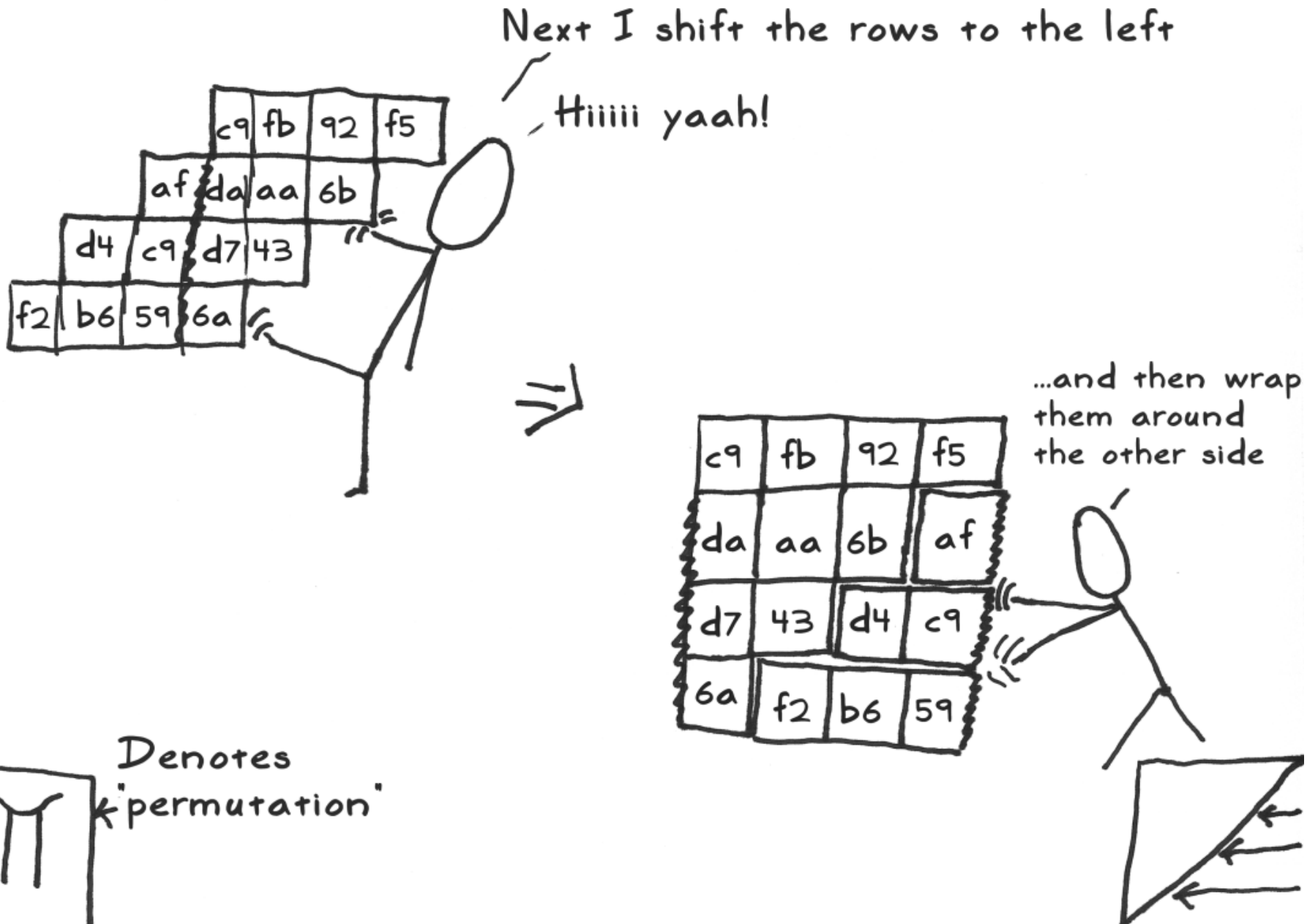




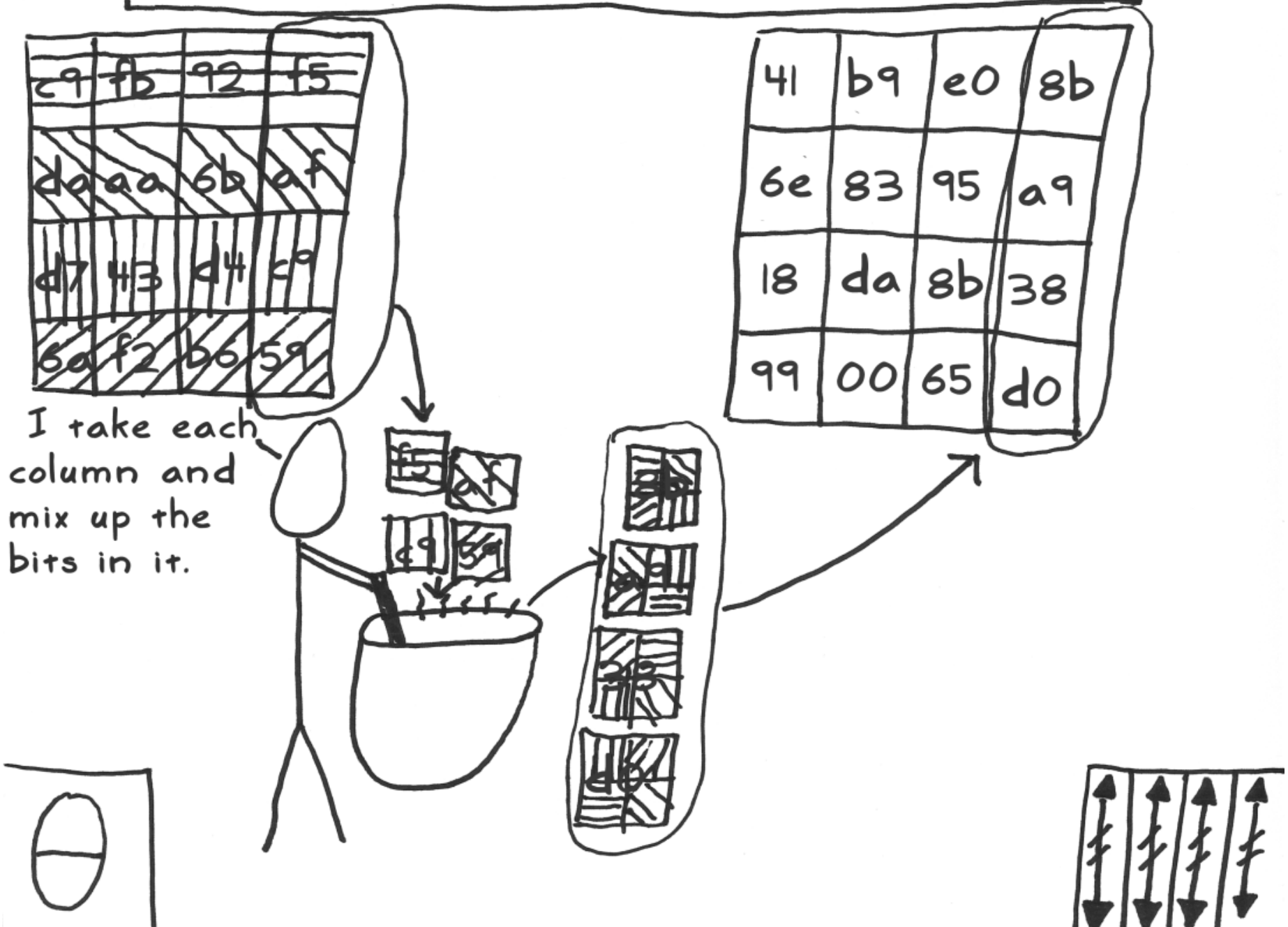
Denotes "confusion"



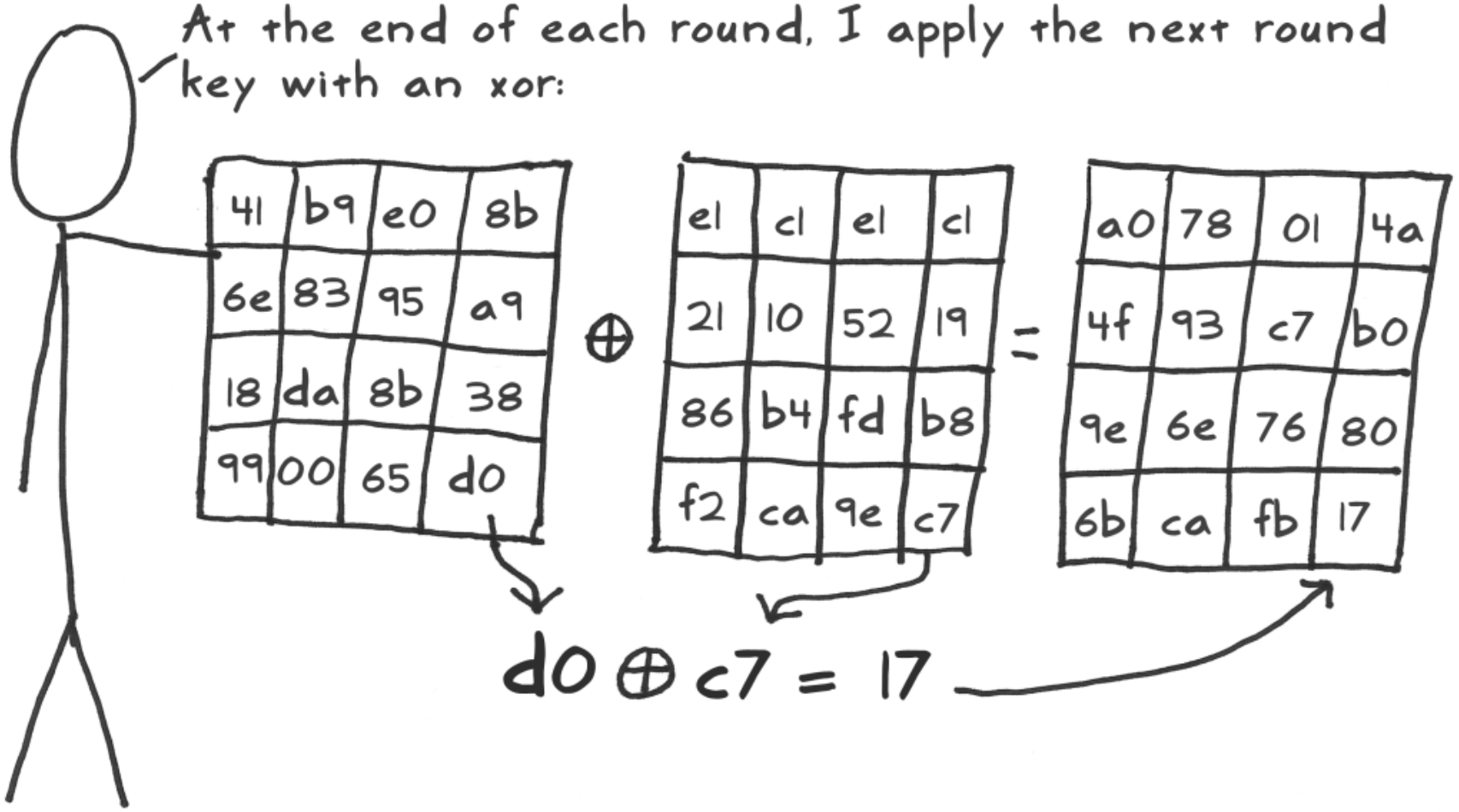
Applying Diffusion, Part 1: Shift Rows



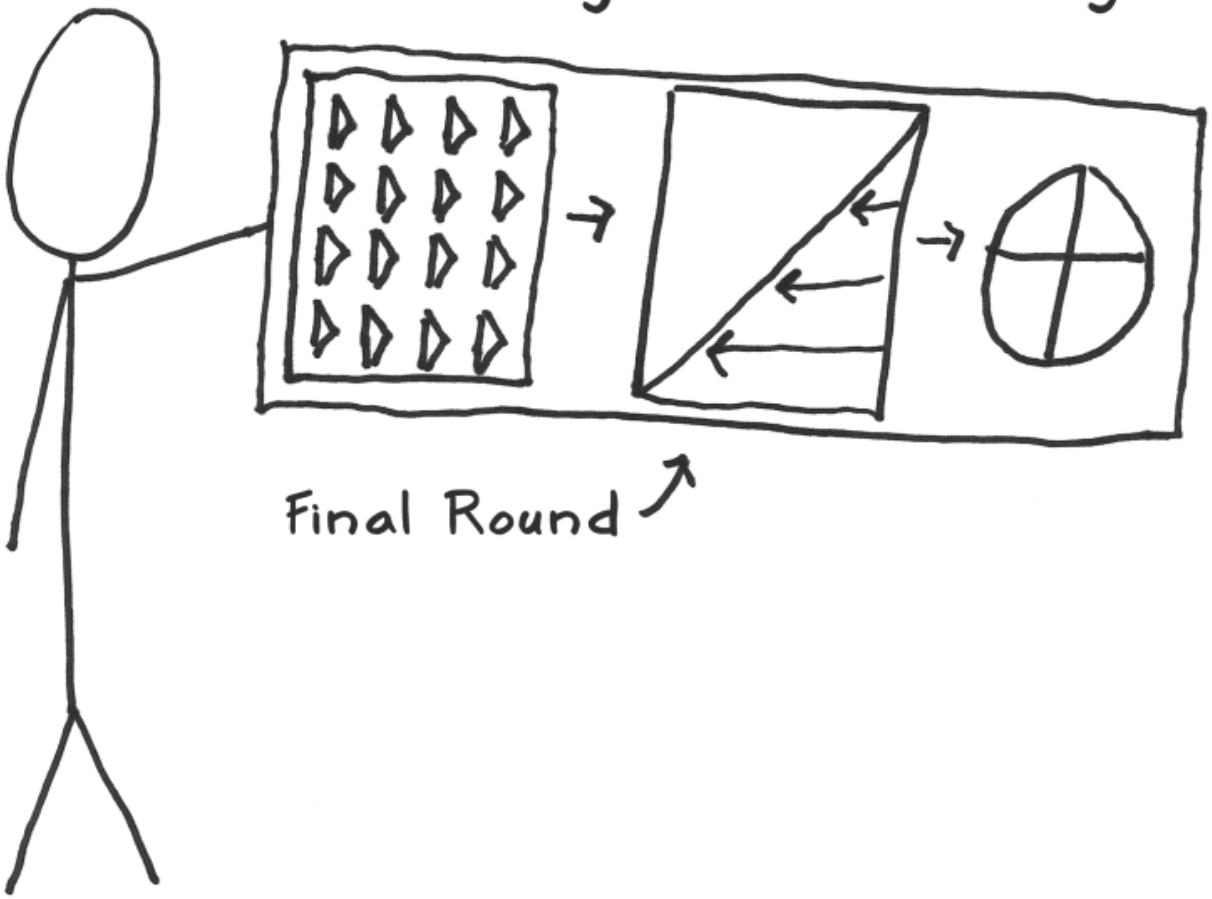
Applying Diffusion, Part 2: Mix Columns



Applying Key Secrecy: Add Round Key

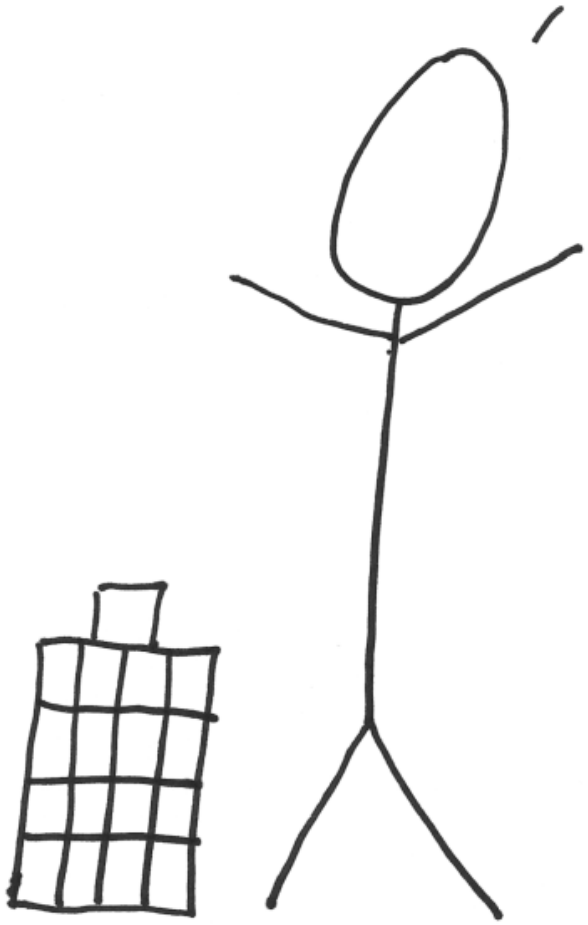


In the final round, I skip the "Mix Columns" step since it wouldn't increase security* and would just slow things down:

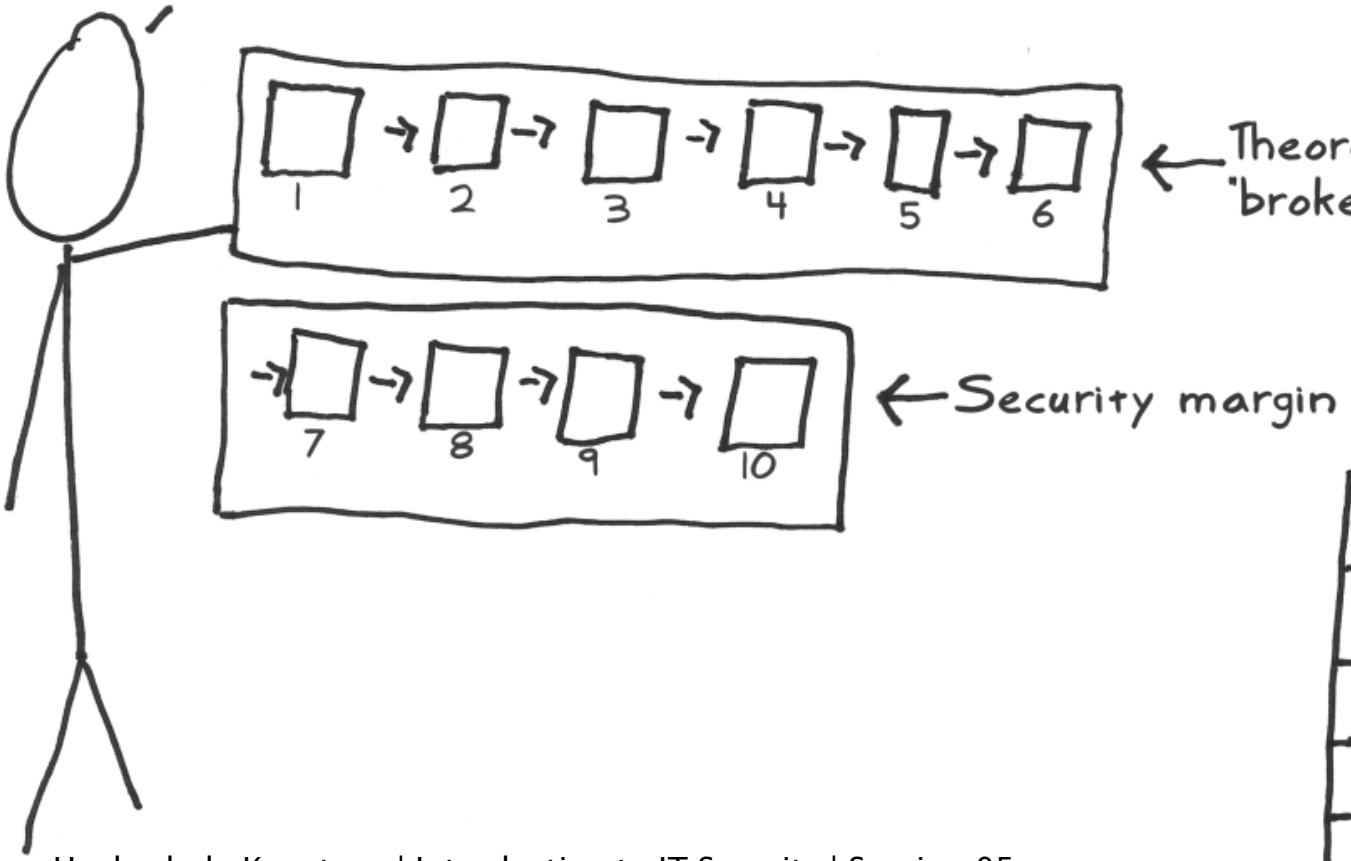


*The diffusion it would provide wouldn't go to the next round.

...and that's it. Each round I do makes the bits more confused and diffused. It also has the key impact them. The more rounds, the merrier!

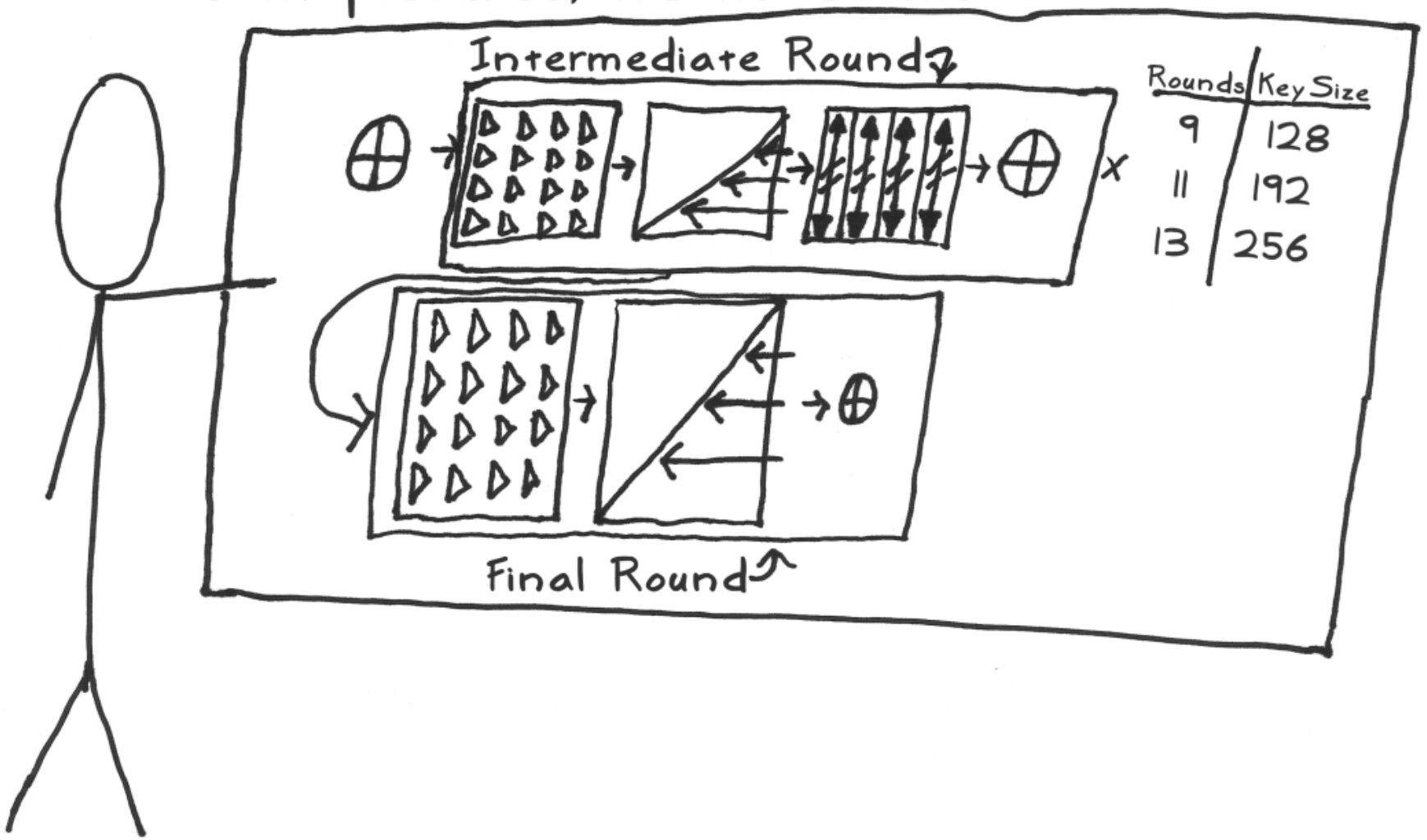


When I was being developed, a clever guy was able to find a shortcut path through 6 rounds. That's not good! If you look carefully, you'll see that each bit of a round's output depends on every bit from two rounds ago. To increase this diffusion "avalanche," I added 4 extra rounds. This is my "security margin."

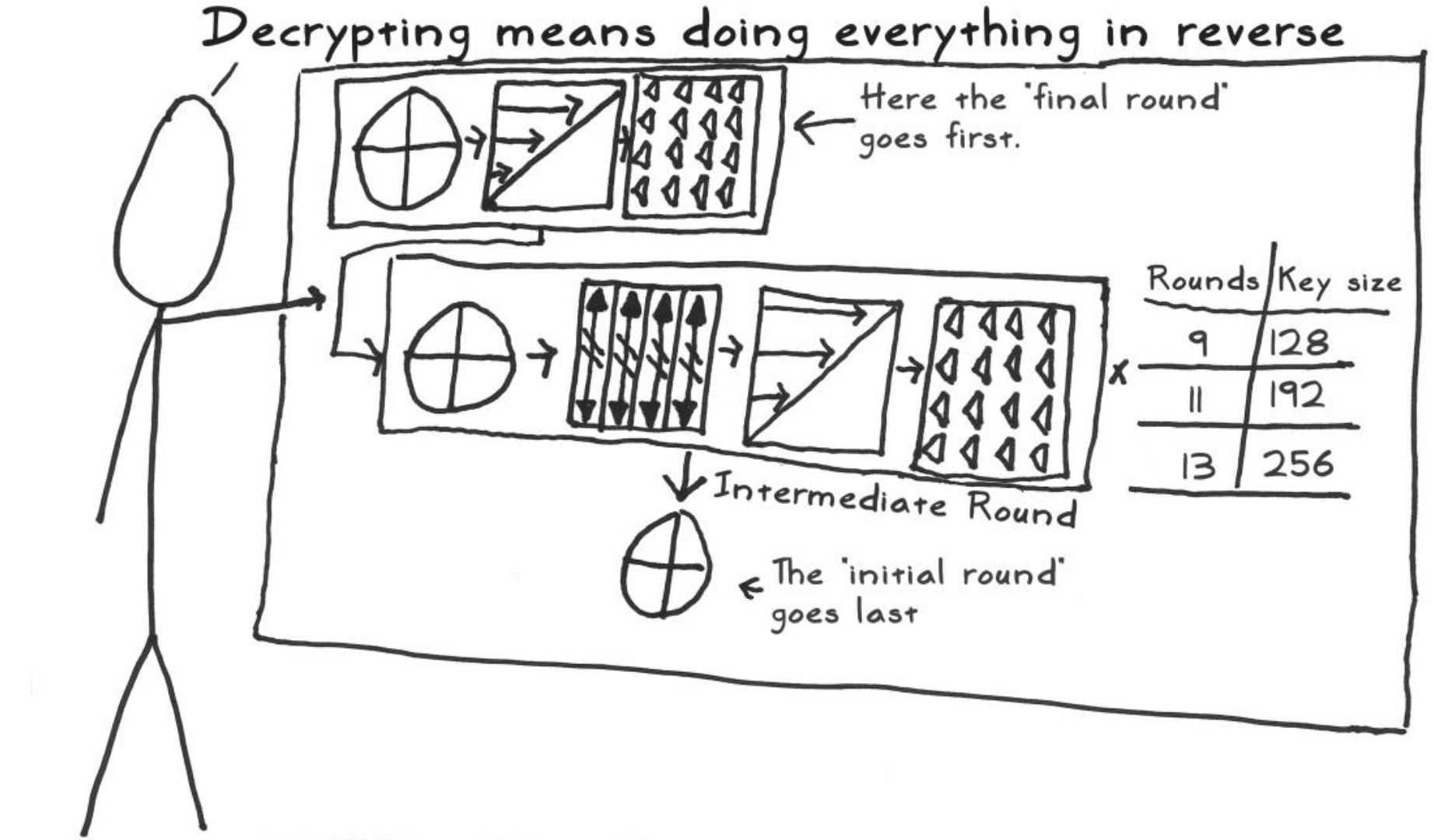


FIPS 197 Spec	
Key Size	Rounds
128	10
192	12
256	14

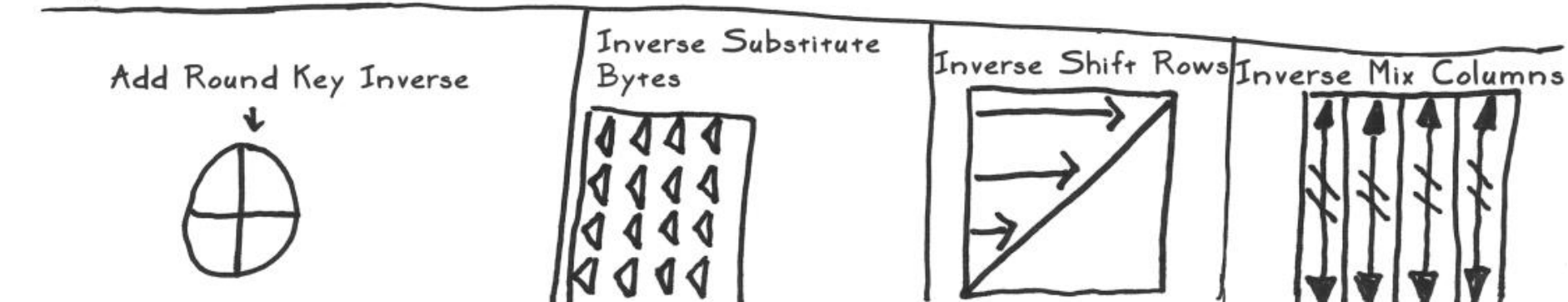
So in pictures, we have this:



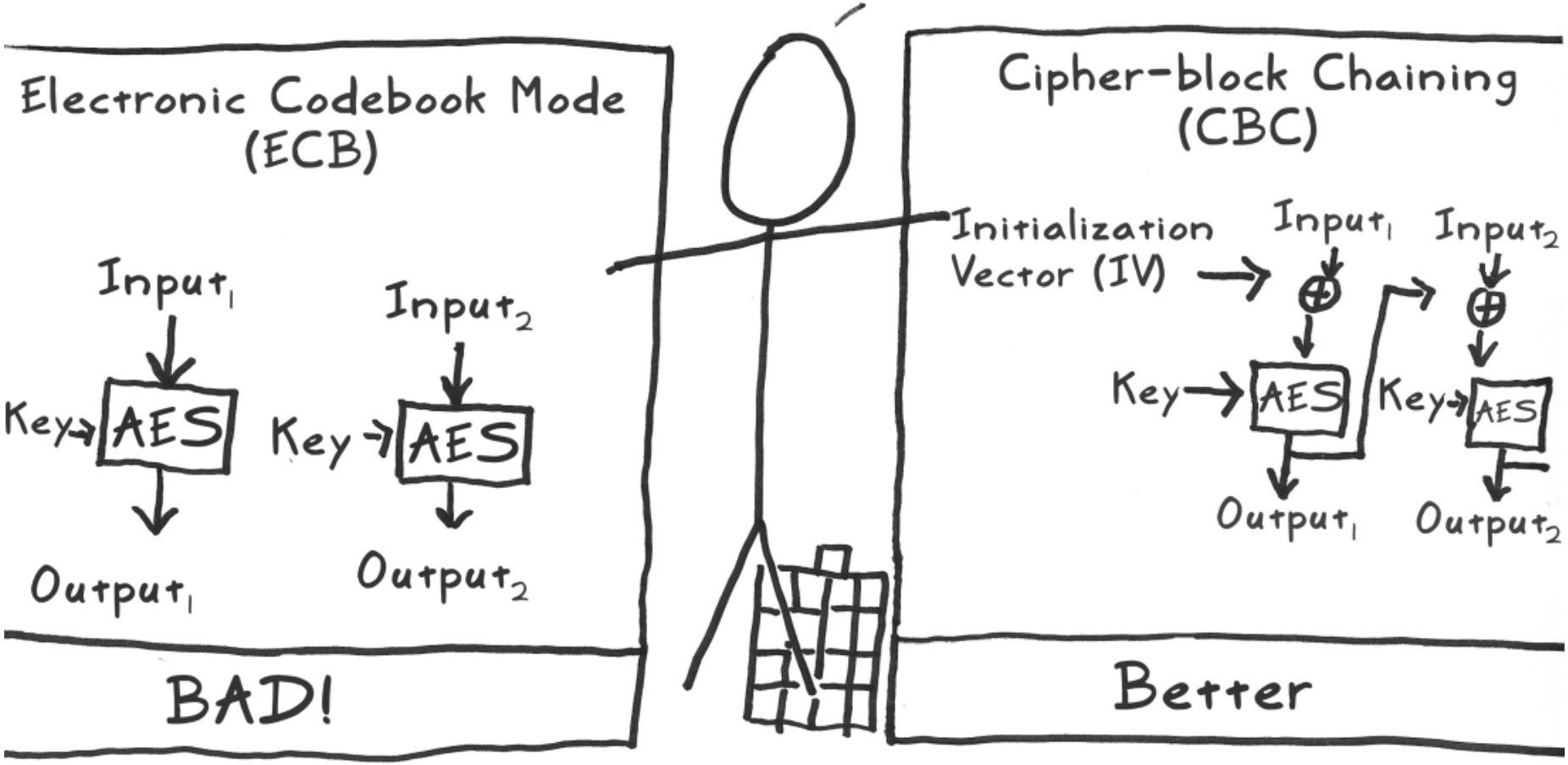
Decrypting means doing everything in reverse



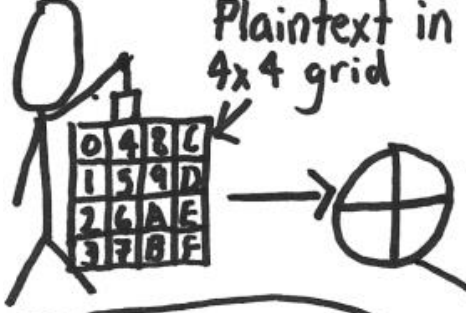
Rounds	Key size
9	128
11	192
13	256



One last tidbit: I shouldn't be used as-is, but rather as a building block to a decent "mode."



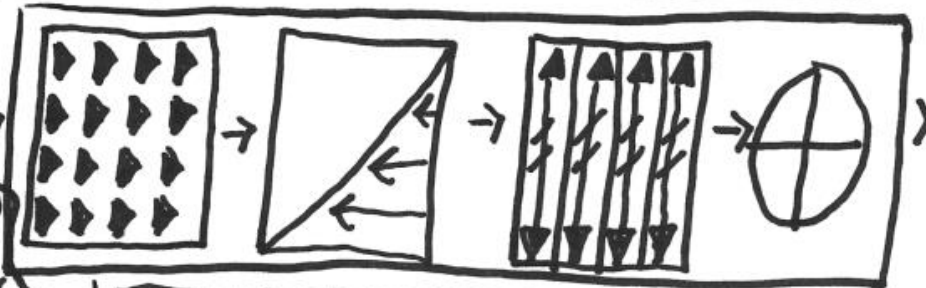
AES Crib Sheet
(Handy for memorizing)



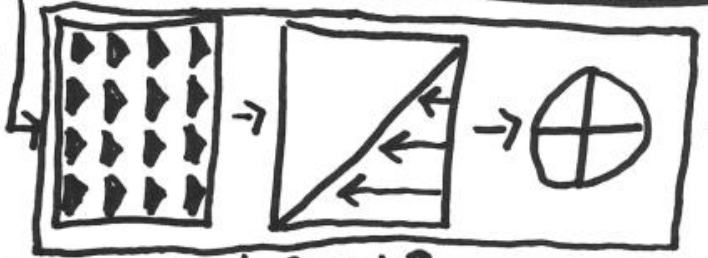
Initial Round

Intermediate Rounds

#	Key
9	128
11	192
13	256



X



Final Round

Ciphertext

General Math

11B = AES Polynomial = m(x)

$X^8 + X^4 + X^3 + X + 1$

Fast Multiply

$x \cdot a(x) = (a << 1) \oplus (a_7 == 1)\ ?\ 1B : 00$

$\log(x \cdot y) = \log(x) + \log(y)$

use $(x+1) = 03$ for log base

S-Box (SRD)

$SRD[a] = f(g(a))$

$g(a) = a^{-1} \bmod m(x)$

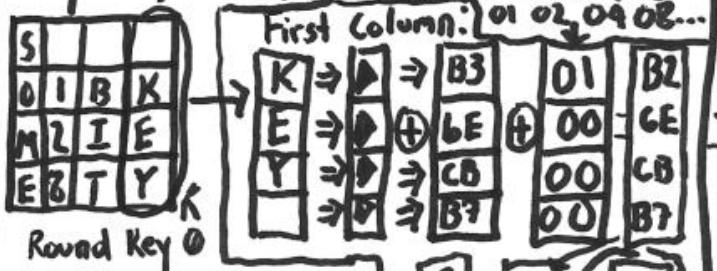
$f(a)$ Think $\vec{53} \oplus 63^T$

5 1's and 3 0's $[0110\ 0011]^T$

Key Expansion:

Round Constants

First Column: 01 02 04 08 ...



Other Columns:

Prev Col ⊕ Col from Previous round key

Mix Columns:

$\vec{2113}$

$$\begin{bmatrix} 2113 \\ 3211 \\ 1321 \\ 1132 \end{bmatrix} \cdot \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

Inverse Mix

$\vec{EBD9}$

$$\begin{bmatrix} E\ B\ D\ 9 \\ 9\ E\ B\ D \\ D\ 9\ E\ B \\ B\ D\ 9\ B \end{bmatrix} \cdot \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

Symmetric vs. asymmetric cryptosystems

- Symmetric cryptosystem
  - Both parties use **same (secret) key**
    Trusted channel needed to distribute key
  - **Fast**
- Asymmetric cryptosystem
  - Parties have a **public key** and a **private key**
    Public key can be announced in a public directory
  - **Slow**
- Can be combined ("hybrid")
  - Generate a **secret session key** for a symmetric cryptosystem
  - Use **asymmetric encryption** to **transmit session key**
  - Encrypt further **messages** with received **session key**

General recommendations for application of cryptography

Do not invent your own system

- Designing a cryptosystem is hard
  - Encryption/decryption algorithms, protocols
  - Underlying mathematical problems, complexity theory
  - Random number generation, e.g. for keys
  - Implementation, data leakage from side channels
- **Use existing algorithms and implementations**
  - Compatibility/standards, key management
  - Performance, memory consumption (especially for embedded systems)
- Check for known weaknesses in implementations, use current versions

Random number generation

- Many cryptographic protocols rely on parties choosing **random numbers** ("nonces")
- Pseudo-random number generators (PRNGs)
  - Start with a random value, "**seed**"
  - **Compute sequence** of values based on start value, e.g. by hashing start value and subsequent values
  - **Efficient**
  - **Predictable sequence** based on start value
- True **randomness hard** to obtain
  - Measure unpredictable **physical** phenomenon
  - Mouse movements, microphone, etc.
- BSI AIS 20/31 Functionality classes for random number generators

Random number generation

- Attackers may try to predict start value of PRNG
- Netscape 1.1 browser 1996

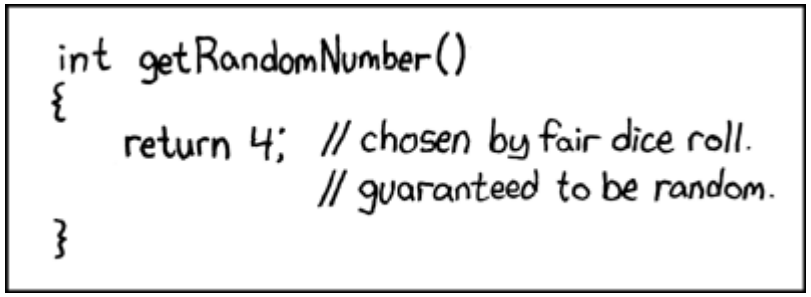
```
global variable seed;

RNG_CreateContext()
    (seconds, microseconds) = time of day; /* Time elapsed since 1970 */
    pid = process ID;  ppid = parent process ID;
    a = mklcpr(microseconds);
    b = mklcpr(pid + seconds + (ppid << 12));
     seed = MD5(a, b);

mklcpr(x) /* not cryptographically significant; shown for completeness */
    return ((0xDEECE66D * x + 0x2BBB62DC) >> 1);
```

```
RNG_GenerateRandomBytes()
    x = MD5(seed);
    seed = seed + 1;
    return x;

global variable challenge, secret_key;

create_key()
    RNG_CreateContext();
    tmp = RNG_GenerateRandomBytes();
    tmp = RNG_GenerateRandomBytes();
    challenge = RNG_GenerateRandomBytes();
    secret_key = RNG_GenerateRandomBytes();
```

- `seconds`, `pid`, `ppid` known or small variation
- Need to brute force 1 million possibilities for `microseconds` value
- Took 25 s with 1996 hardware

Random number generation

http://xkcd.com/221/



http://dilbert.com/strip/2001-10-25

Summary

- Goals of cryptography:
  - **Confidentiality, integrity** of **data** in transit/rest
- Designing a cryptosystem is hard
  - **Use existing algorithms and implementations**
- **Symmetric** cryptography
  - Same secret key for sender/receiver
  - Example: AES - key expansion, confusion (substitution), diffusion, XOR with round key, #rounds based on key size
- **Asymmetric** cryptography
  - Public key used by sender for encryption, private key used by receiver for decryption