

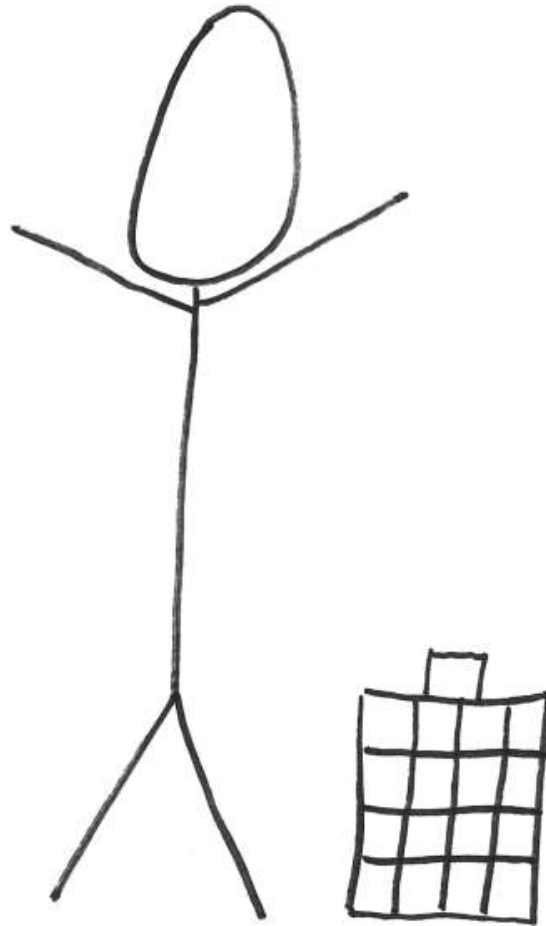
A Stick Figure Guide to the Advanced Encryption Standard (AES)



© Copyright 2009, Jeff Moser
<http://www.moserware.com/>

Act 1: Once Upon a Time...

I handle petabytes* of data every day. From encrypting juicy Top Secret intelligence to boring packets bound for your Wifi router, I do it all!

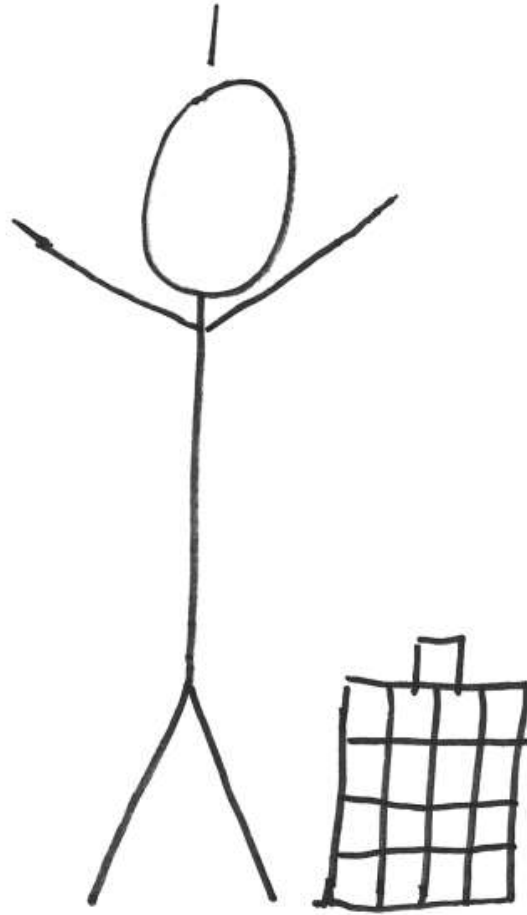


* 1 petabyte \approx a lot

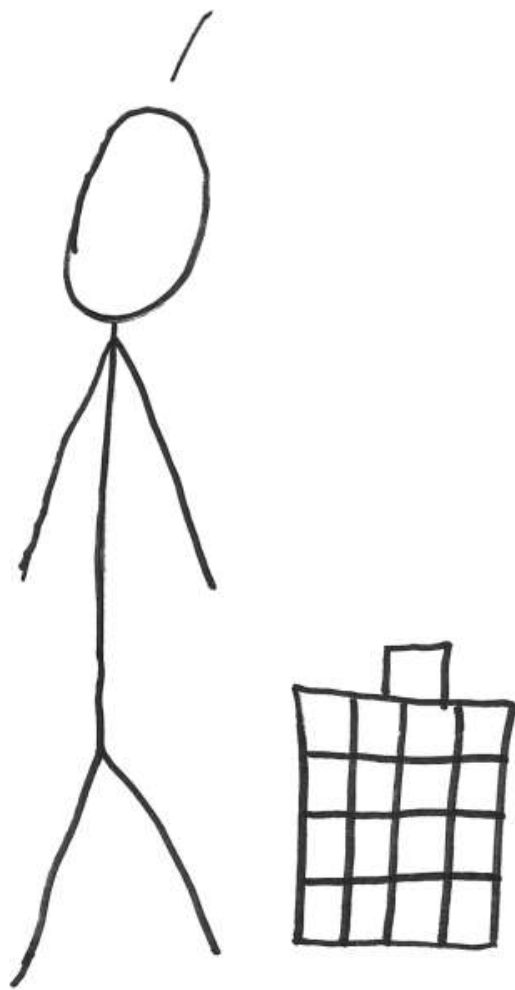
... and still no one seems to care
about me or my story.



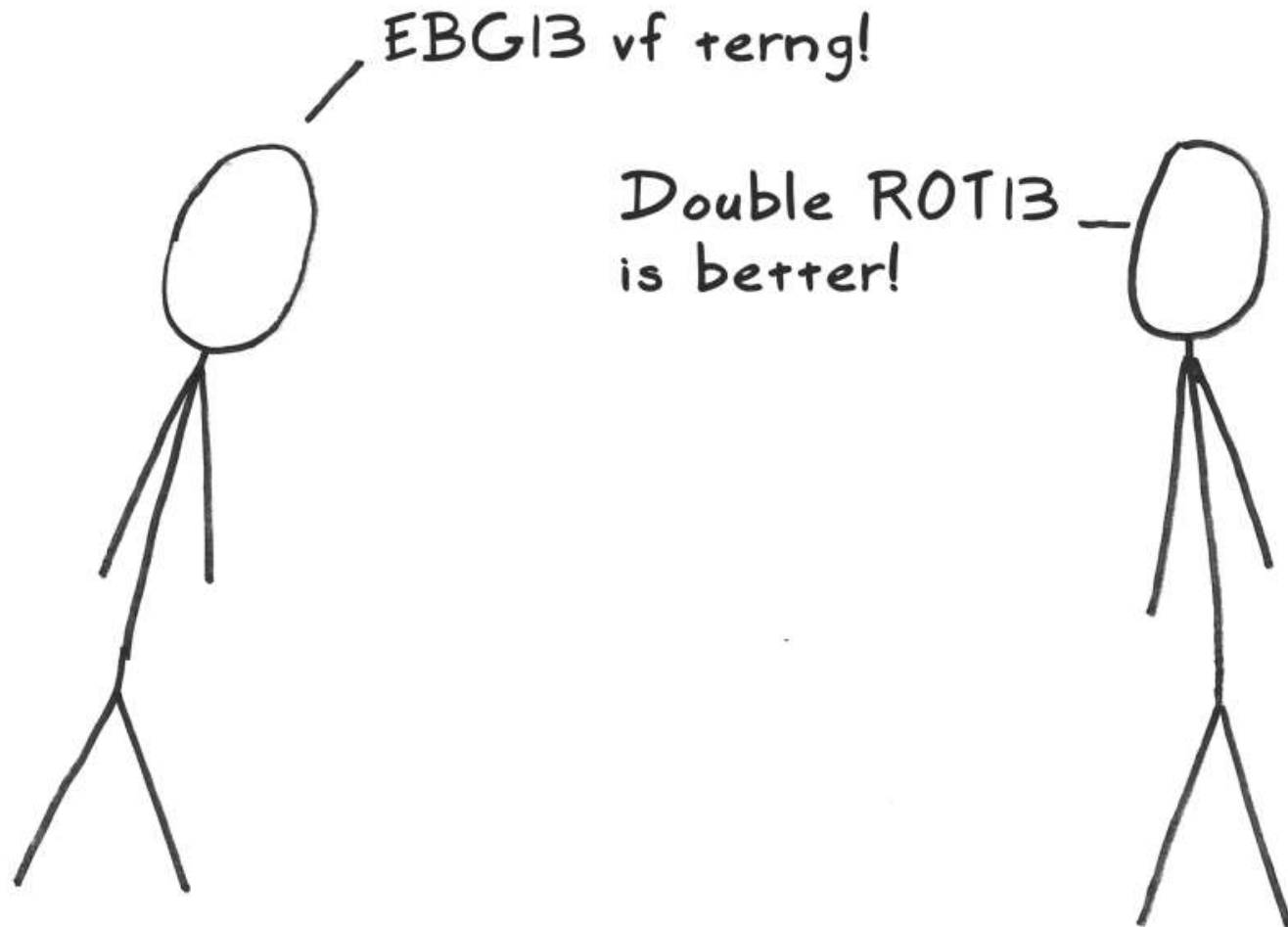
I've got a better-than-Cinderella
story as I made my way to become
king of the block cipher world.



Whoa! You're still there. You want
to hear it? Well let's get started...



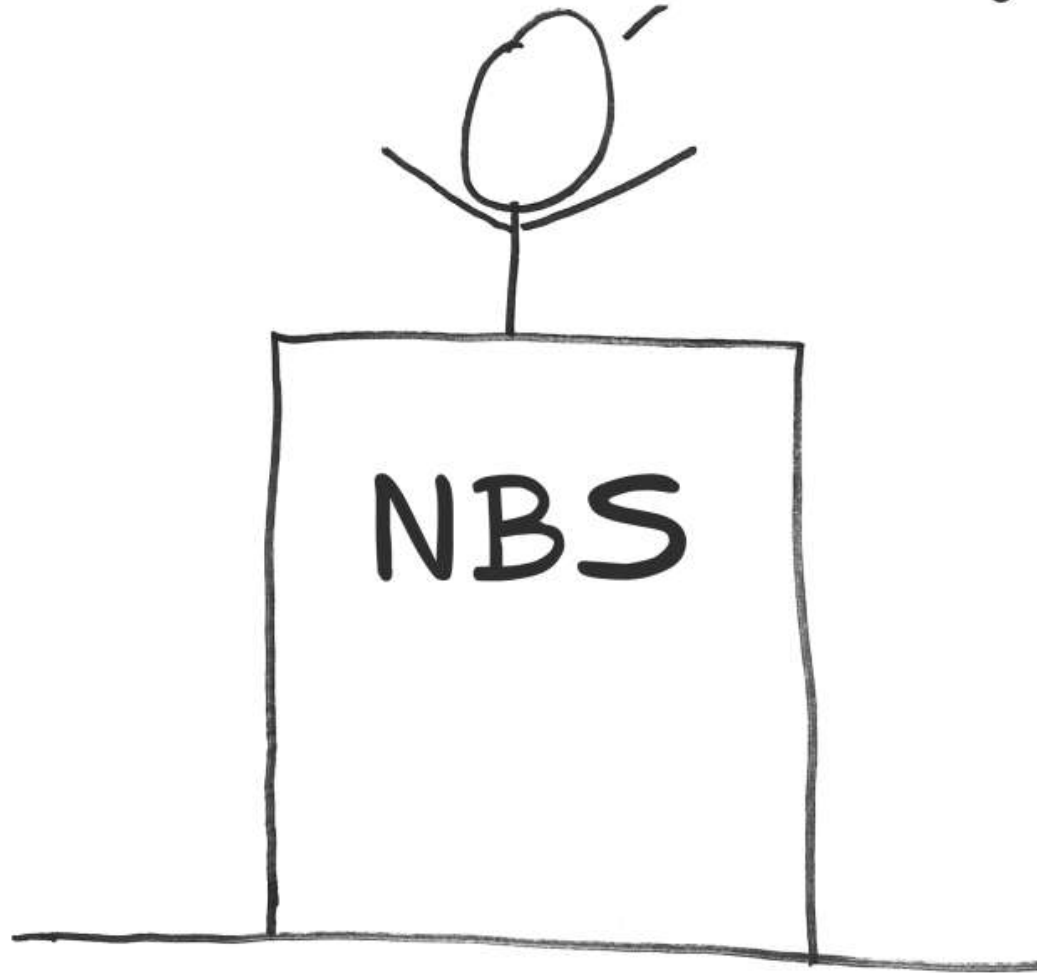
Once upon a time,* there was no good way for people outside secret agencies to judge good crypto.



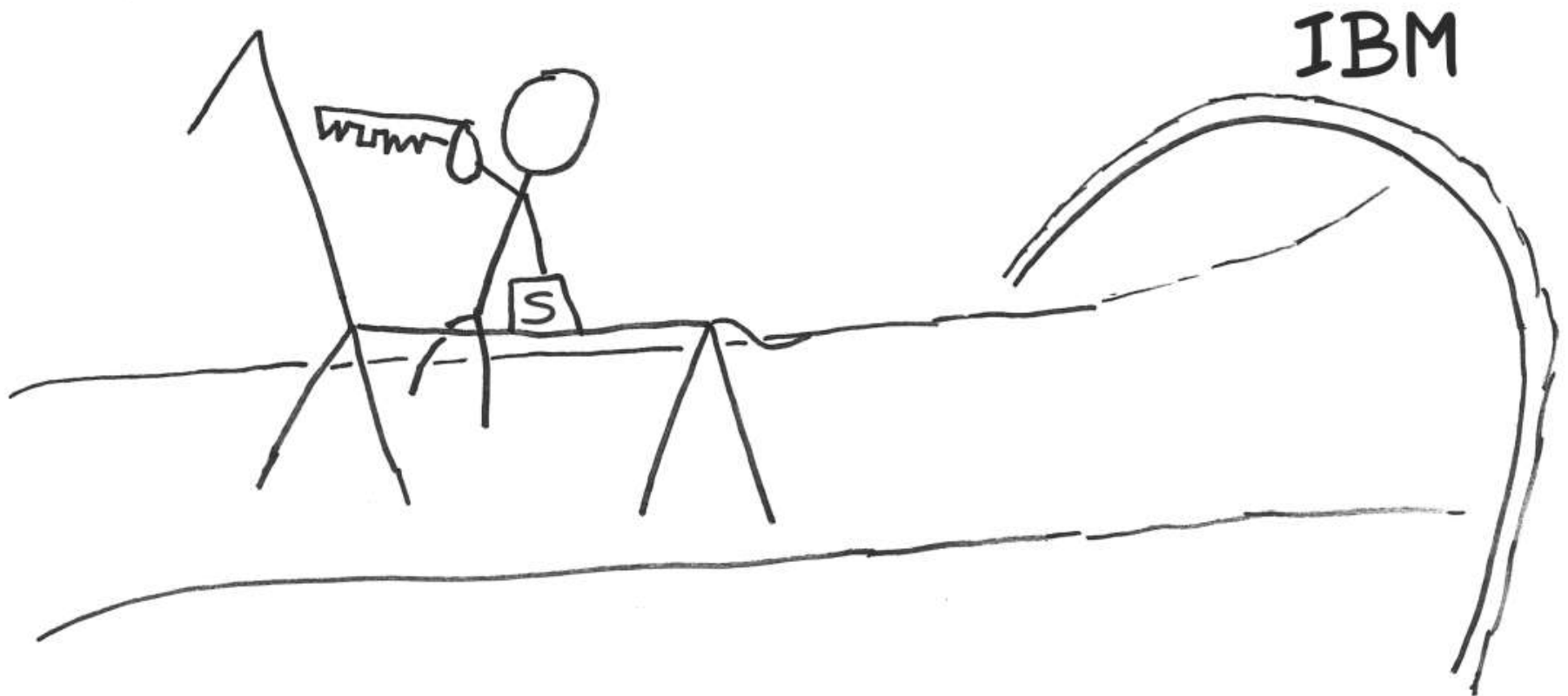
* ~ pre-1975 for the general public

A decree went throughout the land to find a good, secure, algorithm.

We need a good cipher!

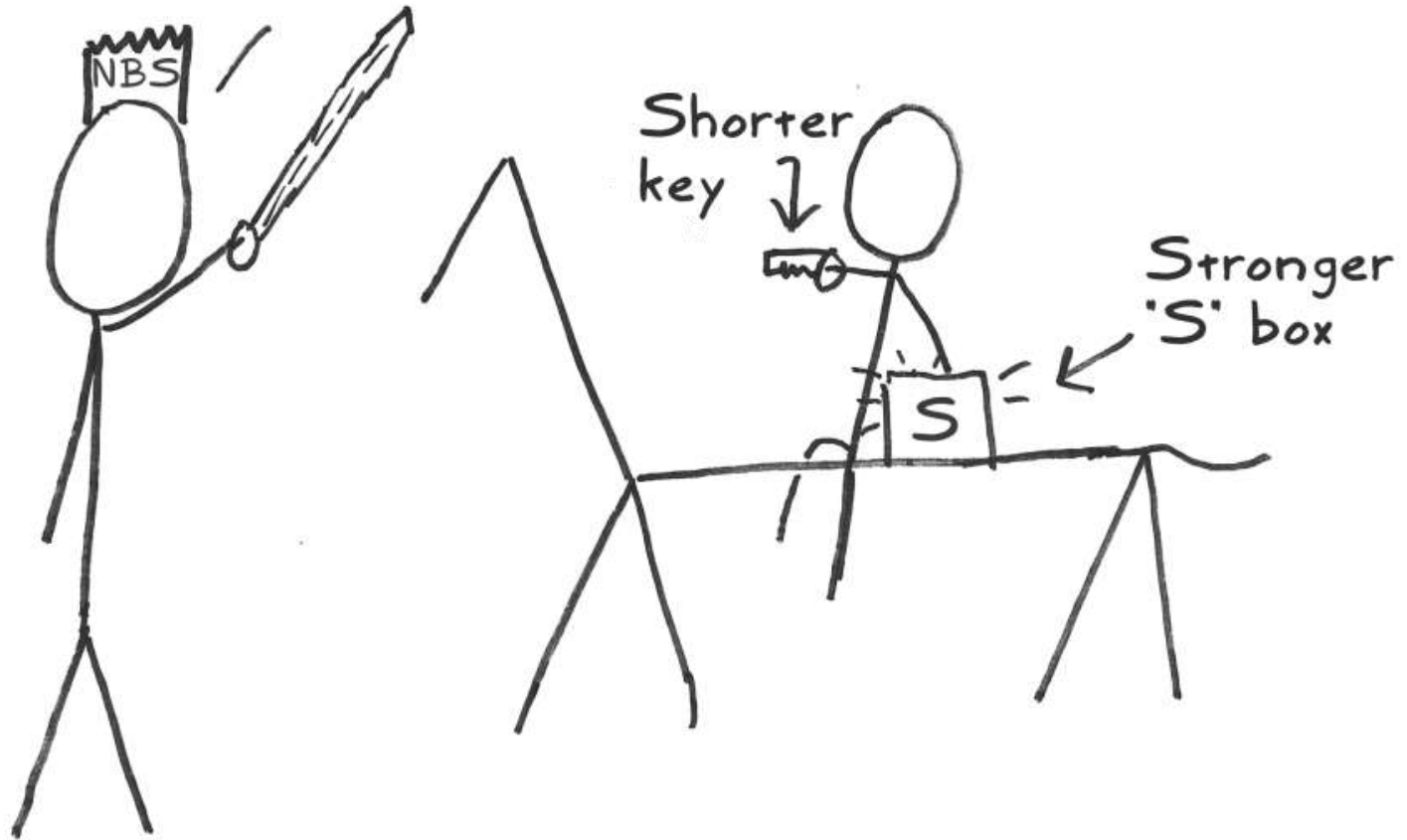


One worthy competitor named
Lucifer came forward.



After being modified by the National Security Agency (NSA), he was anointed as the Data Encryption Standard (DES).

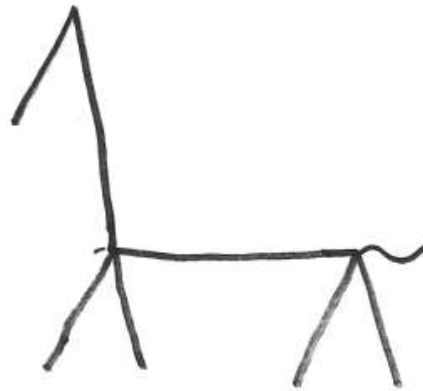
I anoint thee as DES!



DES ruled in the land for over 20 years. Academics studied him intently. For the first time, there was something specific to look at. The modern field of cryptography was born.

"... to the best of our knowledge, DES is free from any statistical or mathematical weakness."

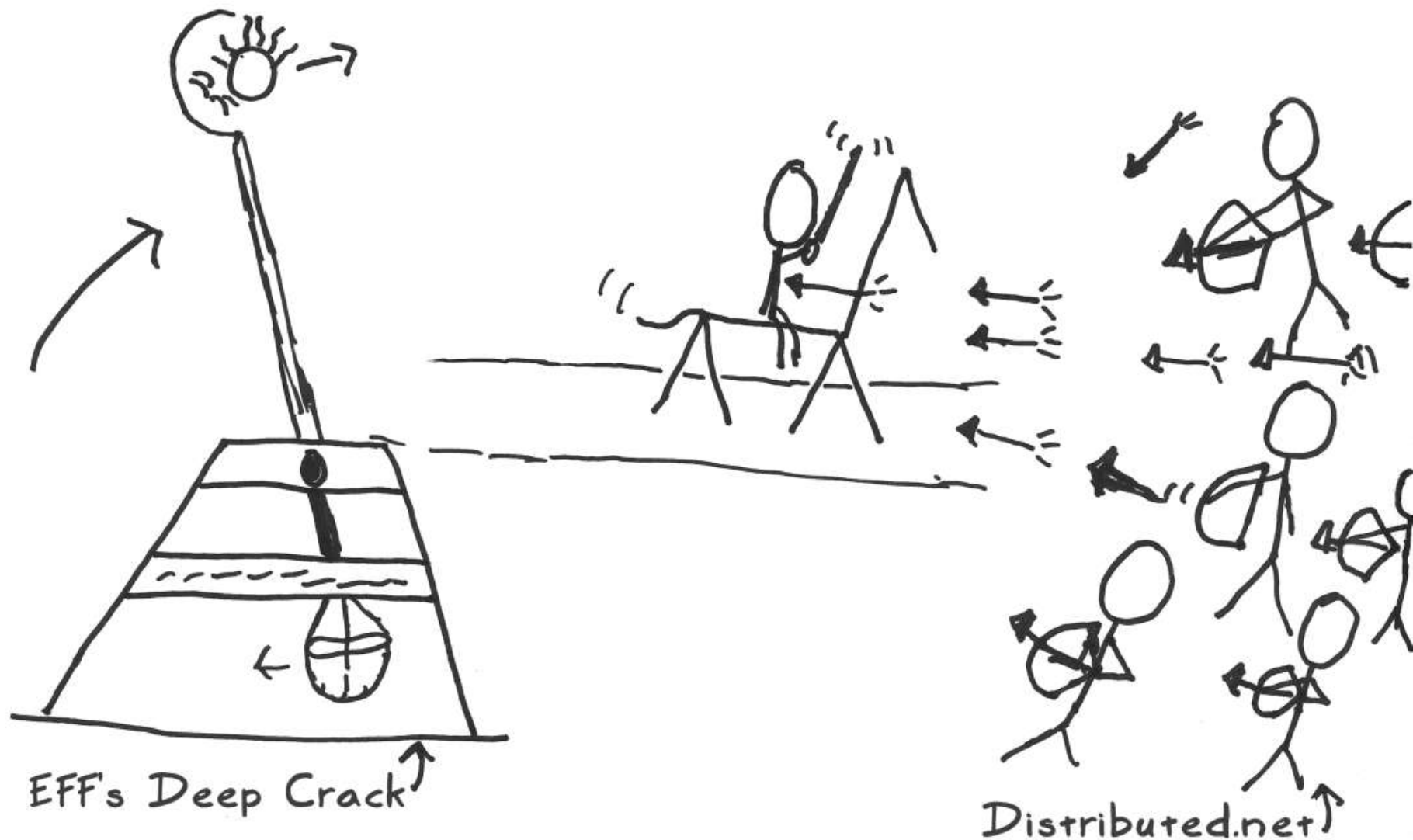
NSA



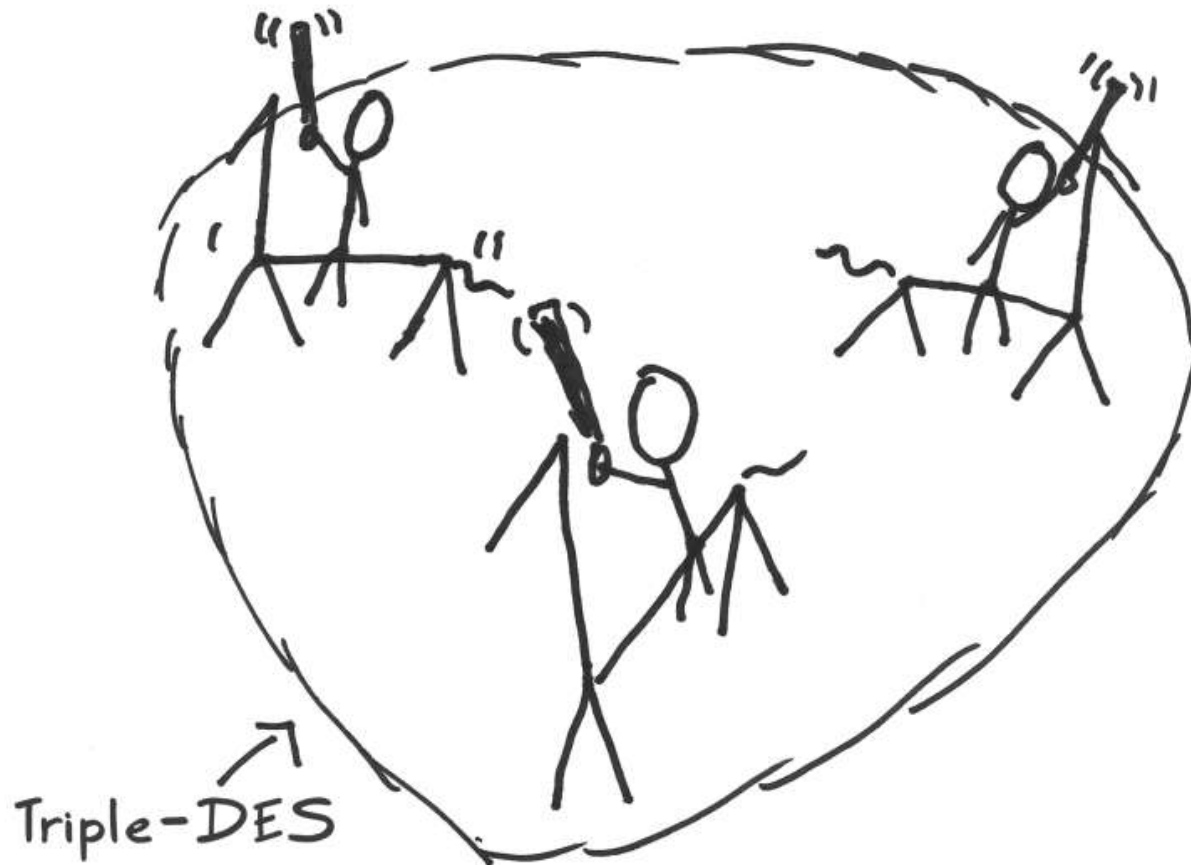
Check out that Feistel network!



Over the years, many attackers challenged DES. He was defeated in several battles.



The only way to stop the attacks was to use DES 3 times in row to form "Triple-DES." This worked, but it was awfully slow.



Another decree went out* ...

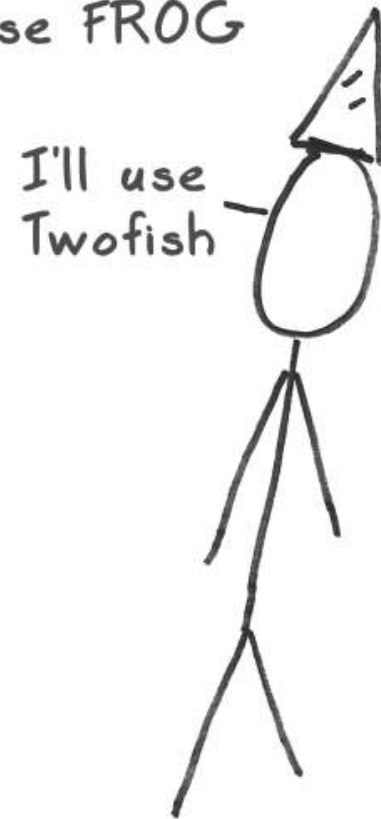
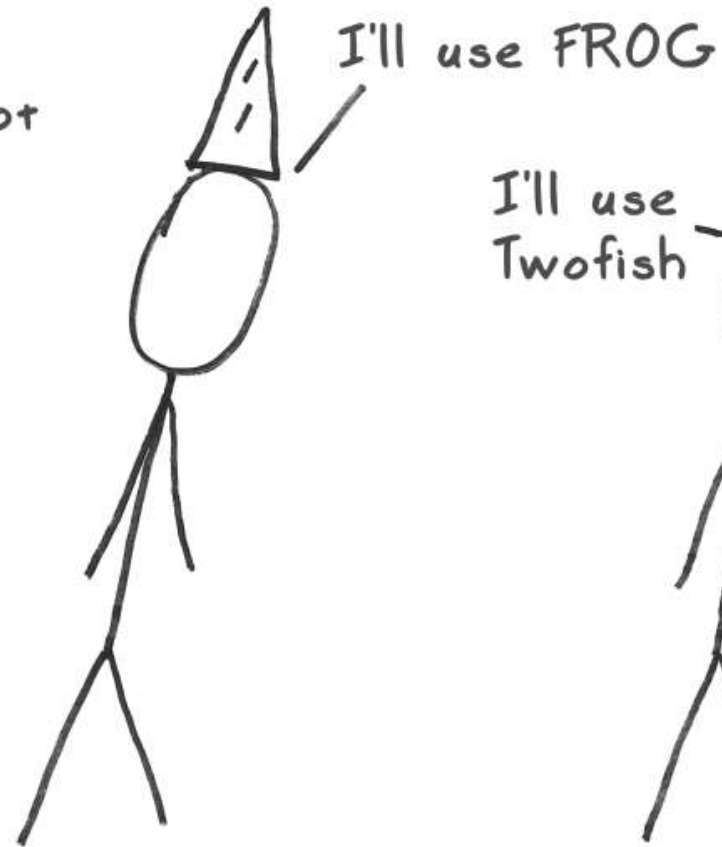
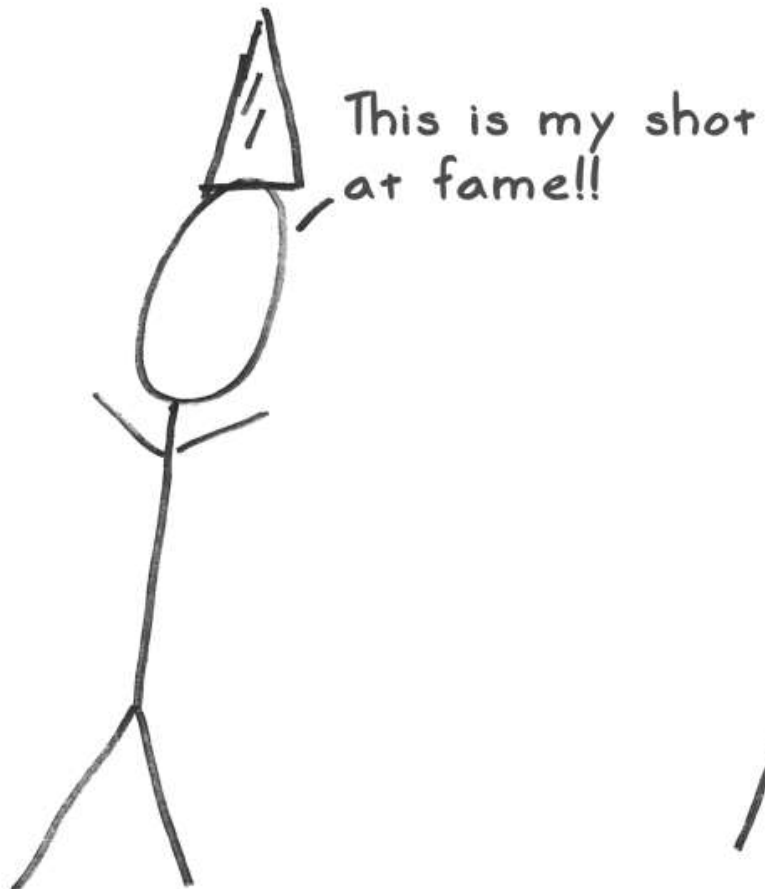
We need something at least as strong as Triple-DES, but it has to be fast and flexible.



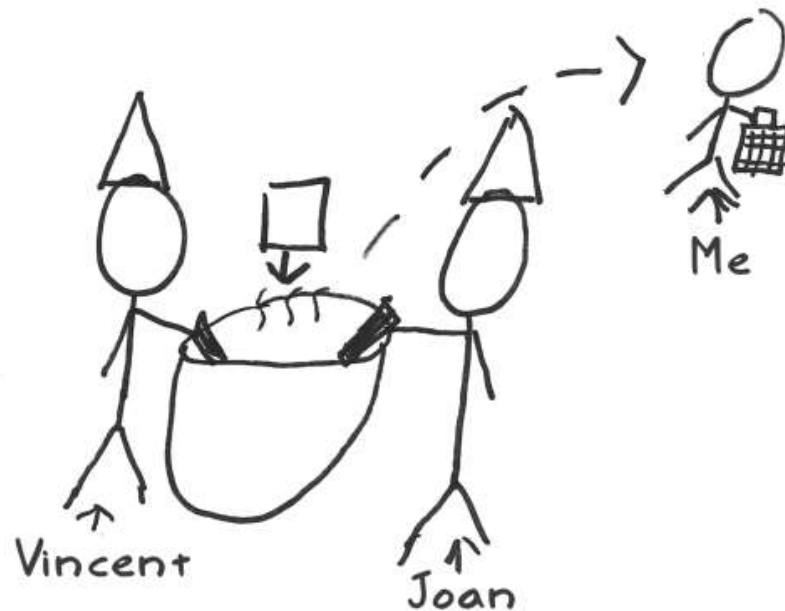
NIST

* ~ early 1997

This call rallied the crypto wizards
to develop something better.

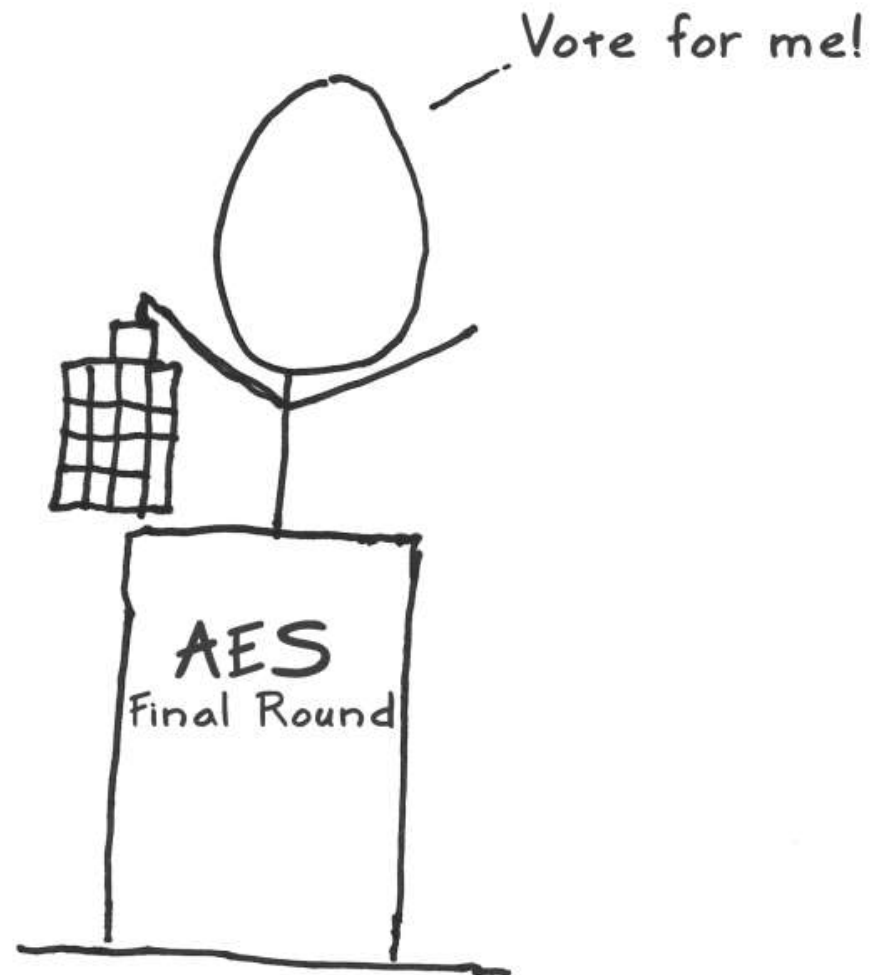


My creators, Vincent Rijmen and Joan Daemen, were among these crypto wizards. They combined their last names to give me my birth name: Rijndael.*



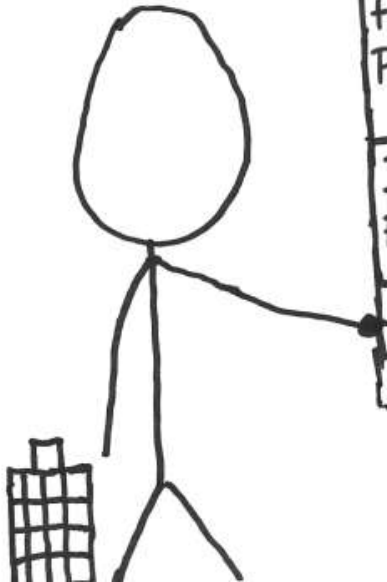
* That's pronounced 'Rhine Dahl' for the non-Belgians out there.

Everyone got together to vote and...



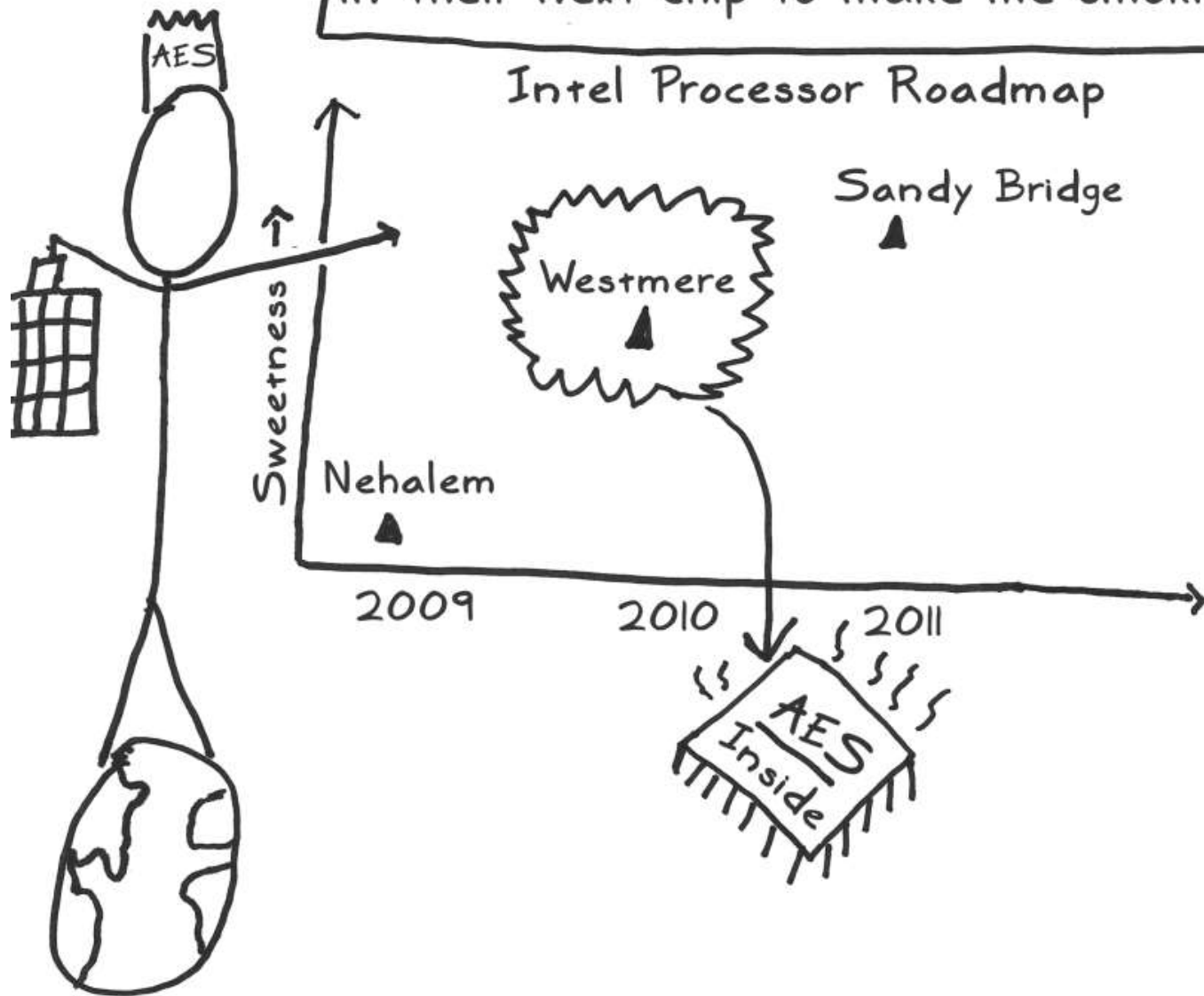
	Rijndael	Serpent	Twofish	MARS	RC6
General Security	2	3	3	3	2
Implementation Difficulty	3	3	2	1	1
Software Performance	3	1	1	2	2
Smart Card Performance	3	3	2	1	1
Hardware Performance	3	3	2	1	2
Design Features	2	1	3	2	1
Total	16	14	13	10	9

I won!!

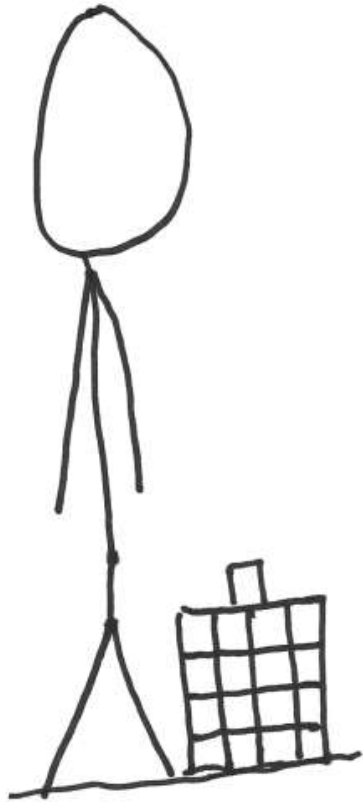


...and now I'm the new king of the crypto world. You can find me everywhere. Intel is even putting native instructions for me in their next chip to make me smokin' fast!

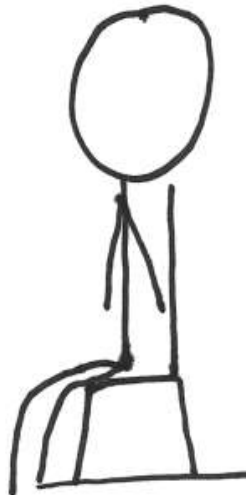
Intel Processor Roadmap



Any questions?



Nice story and
all, but how does
crypto work?



Weird. I'm out...



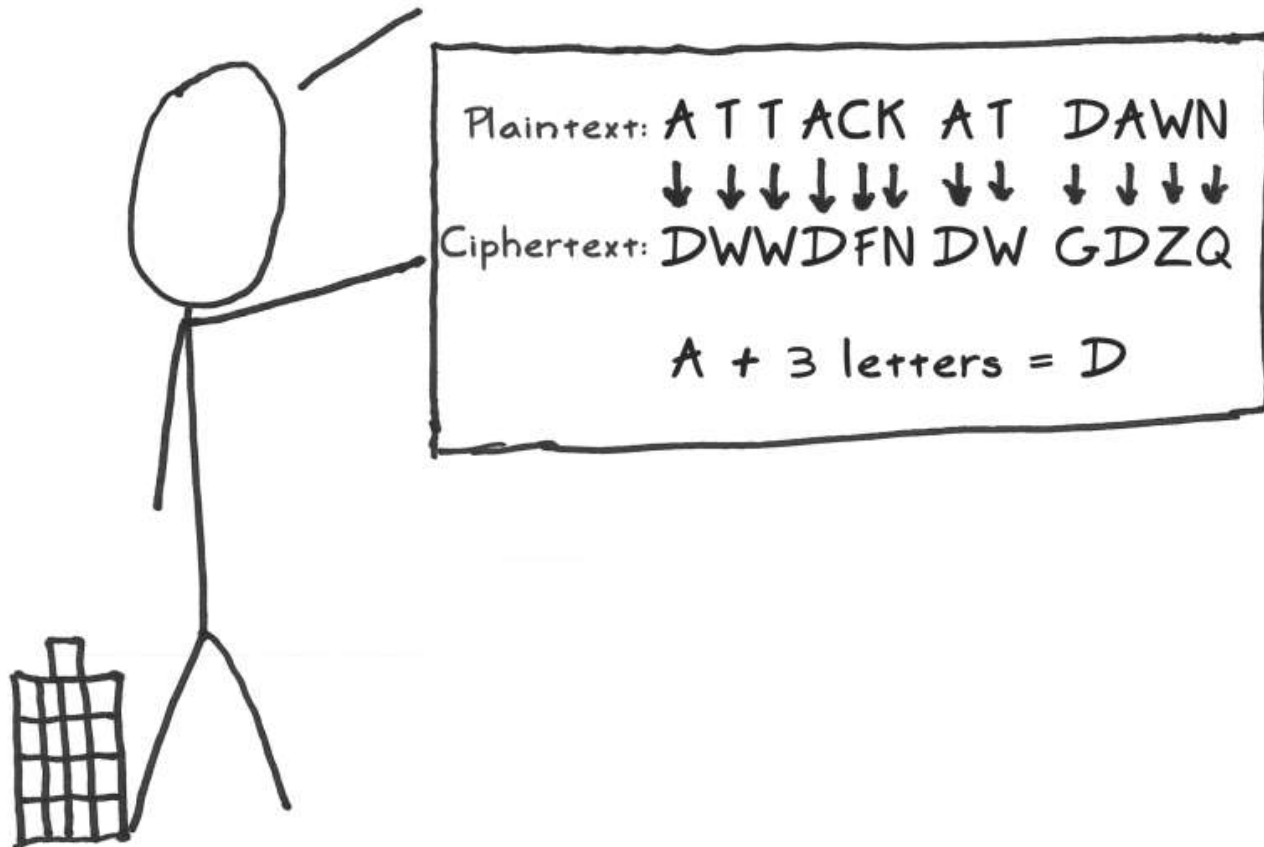
Act 2: Crypto Basics

Great question! You only need to know 3 big ideas to understand crypto.



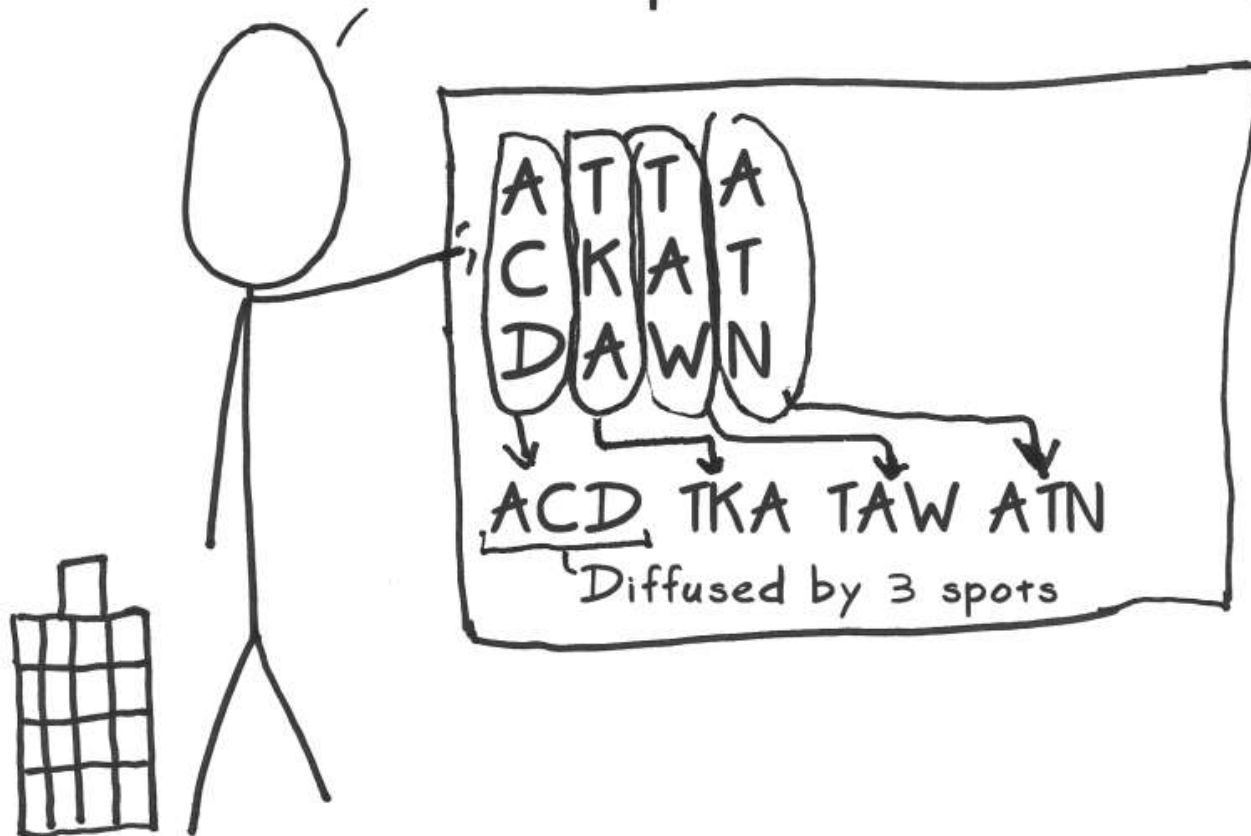
Big Idea #1: Confusion

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:



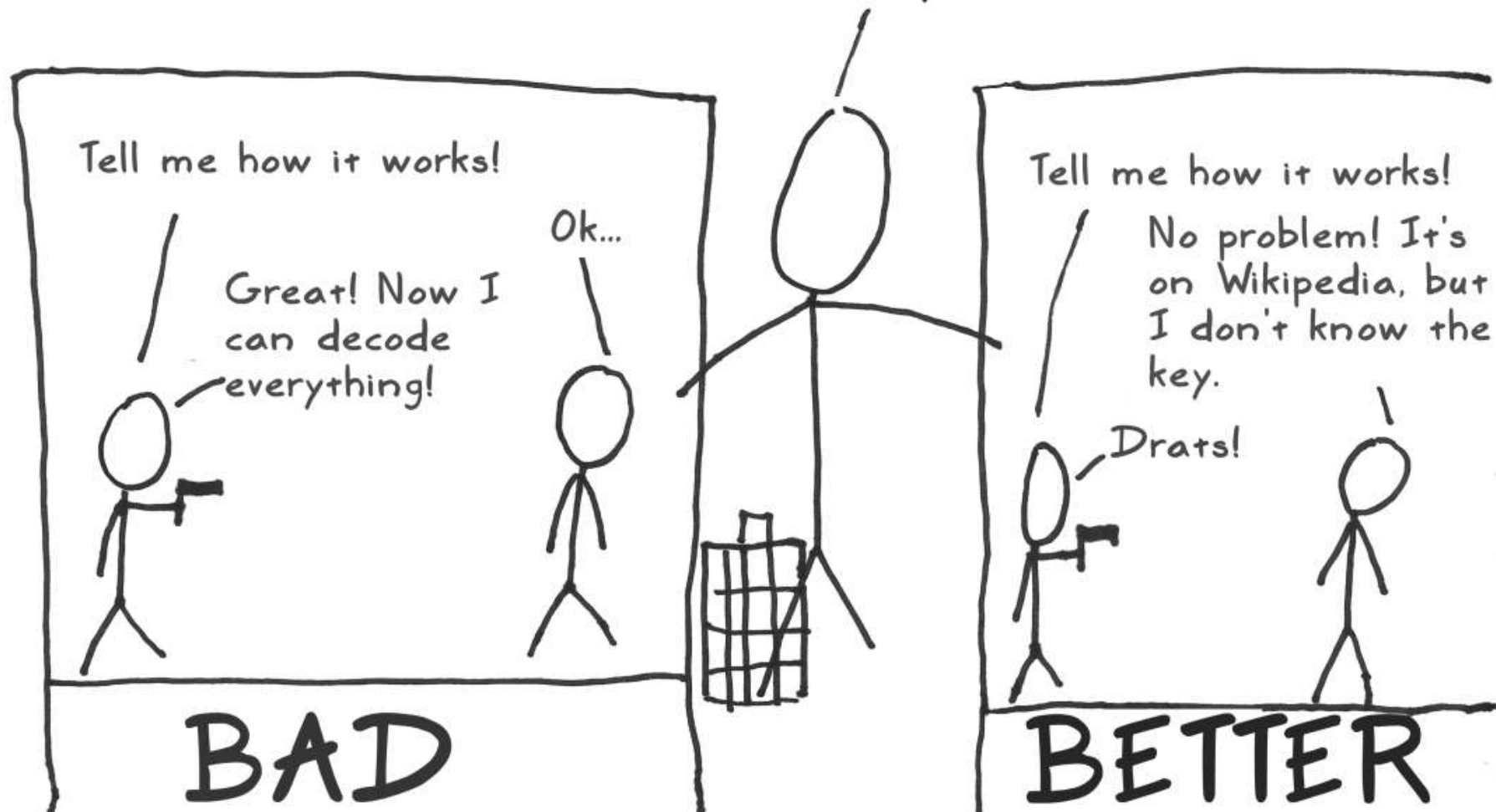
Big Idea #2: Diffusion

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

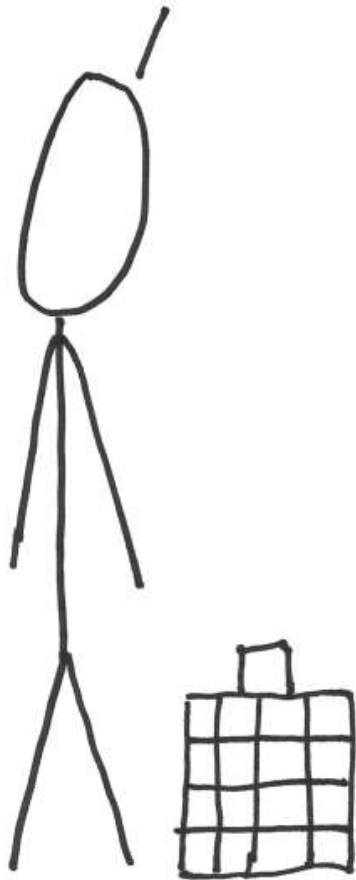


Big Idea #3: Secrecy Only in the Key

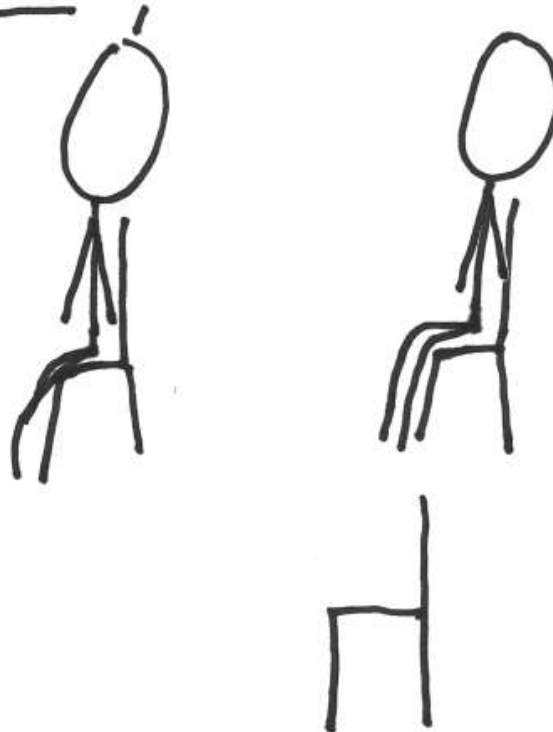
After thousands of years, we learned that it's a bad idea to assume that no one knows how your method works. Someone will eventually find that out.



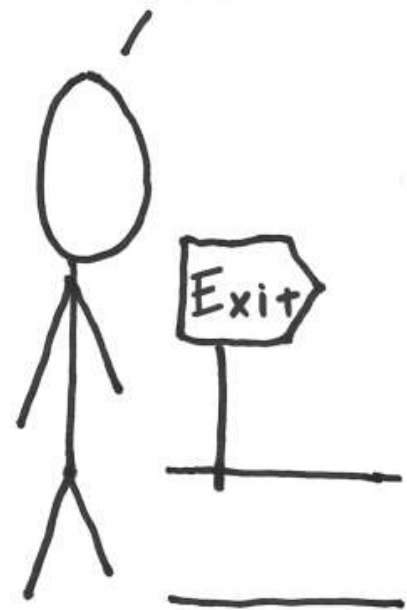
Does that answer
your question?



That helps, but was
too general. How do
you work?



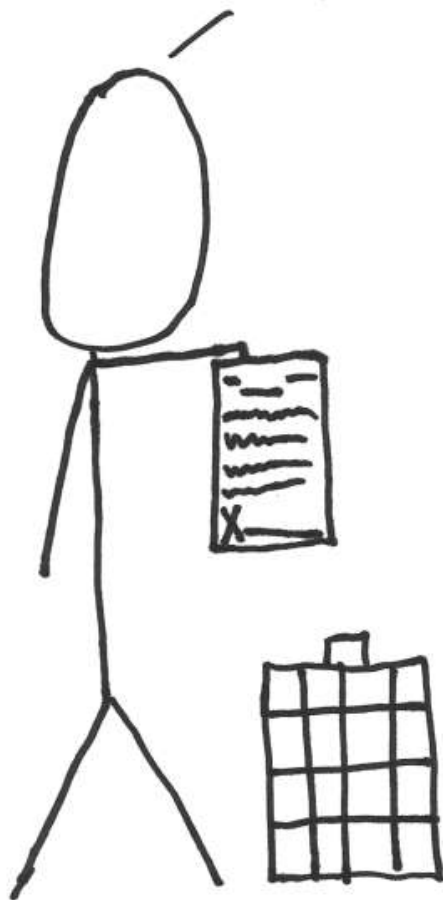
Details? I
can't handle
details!



Act 3: Details

I'd be happy to tell you
how I work, but you have
to sign this first.

Uh... what's that?



Foot-Shooting Prevention Agreement

I, _____, promise that once
Your Name

I see how simple AES really is, I will
not implement it in production code
even though it would be really fun.

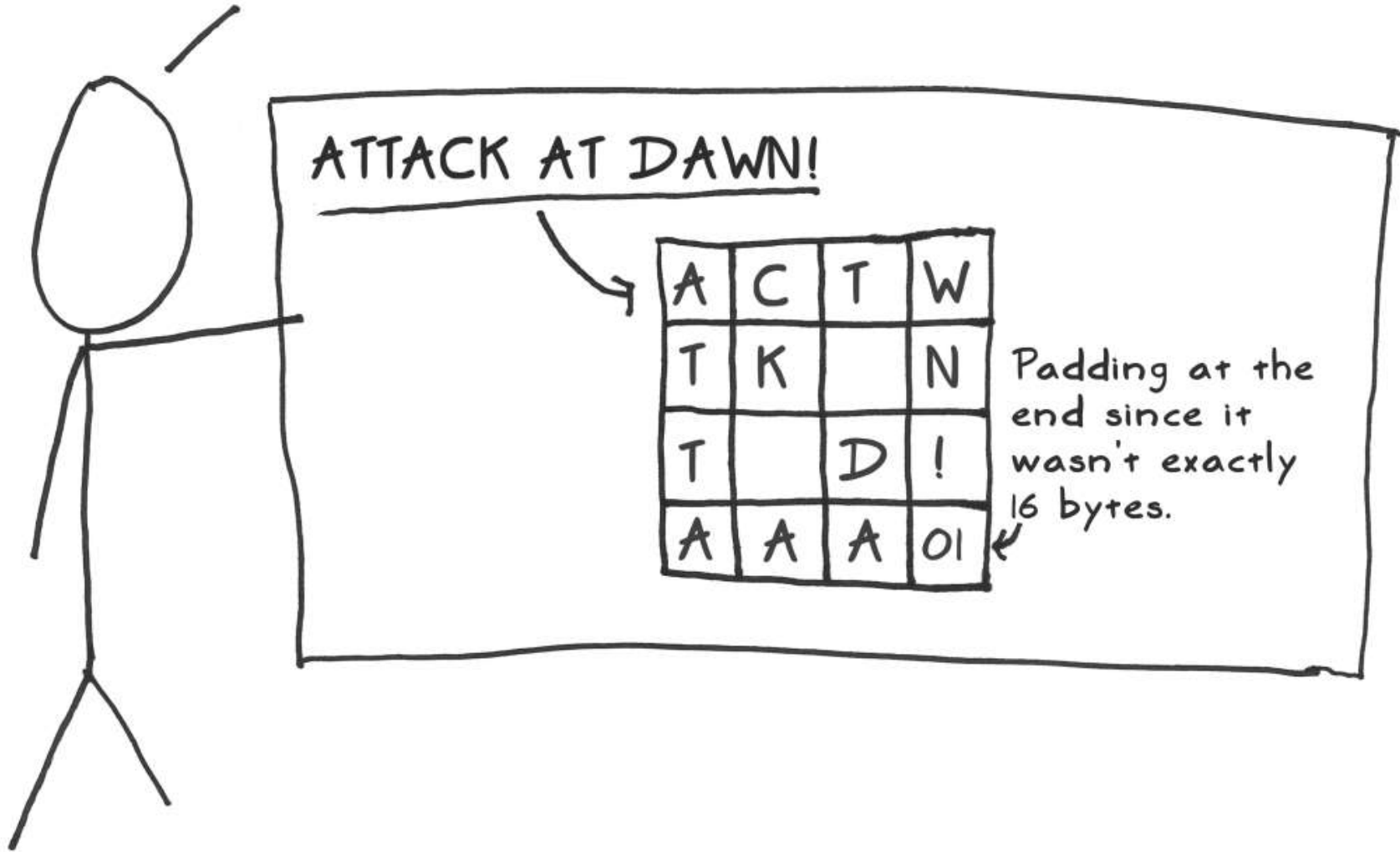
This agreement shall be in effect
until the undersigned creates a
meaningful interpretive dance that
compares and contrasts cache-based,
timing, and other side channel attacks
and their countermeasures.



Signature

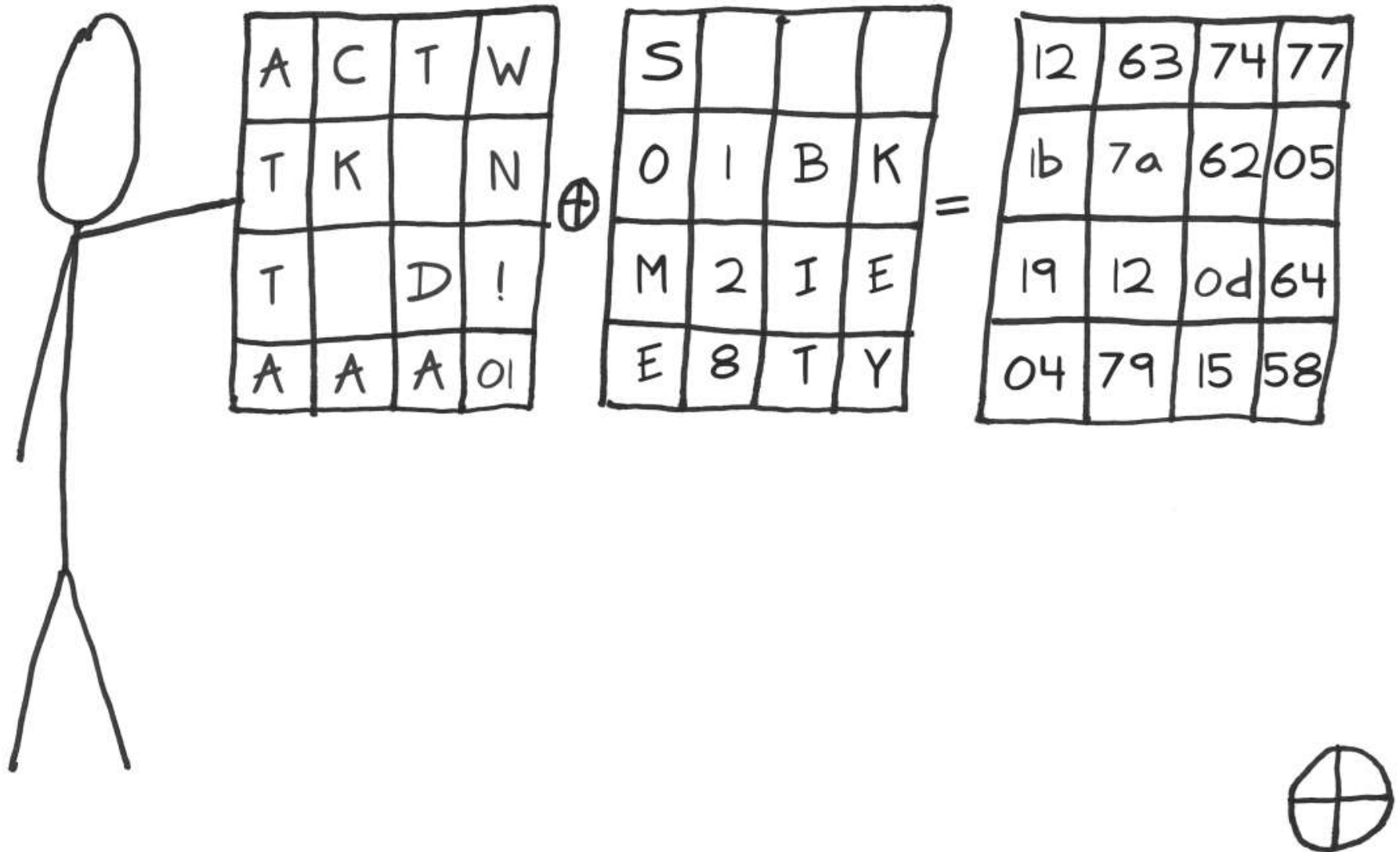
Date

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



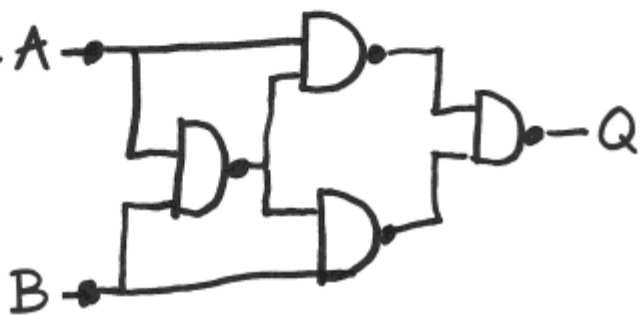
* This is the 'state matrix' that I carry with me at all times.

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.

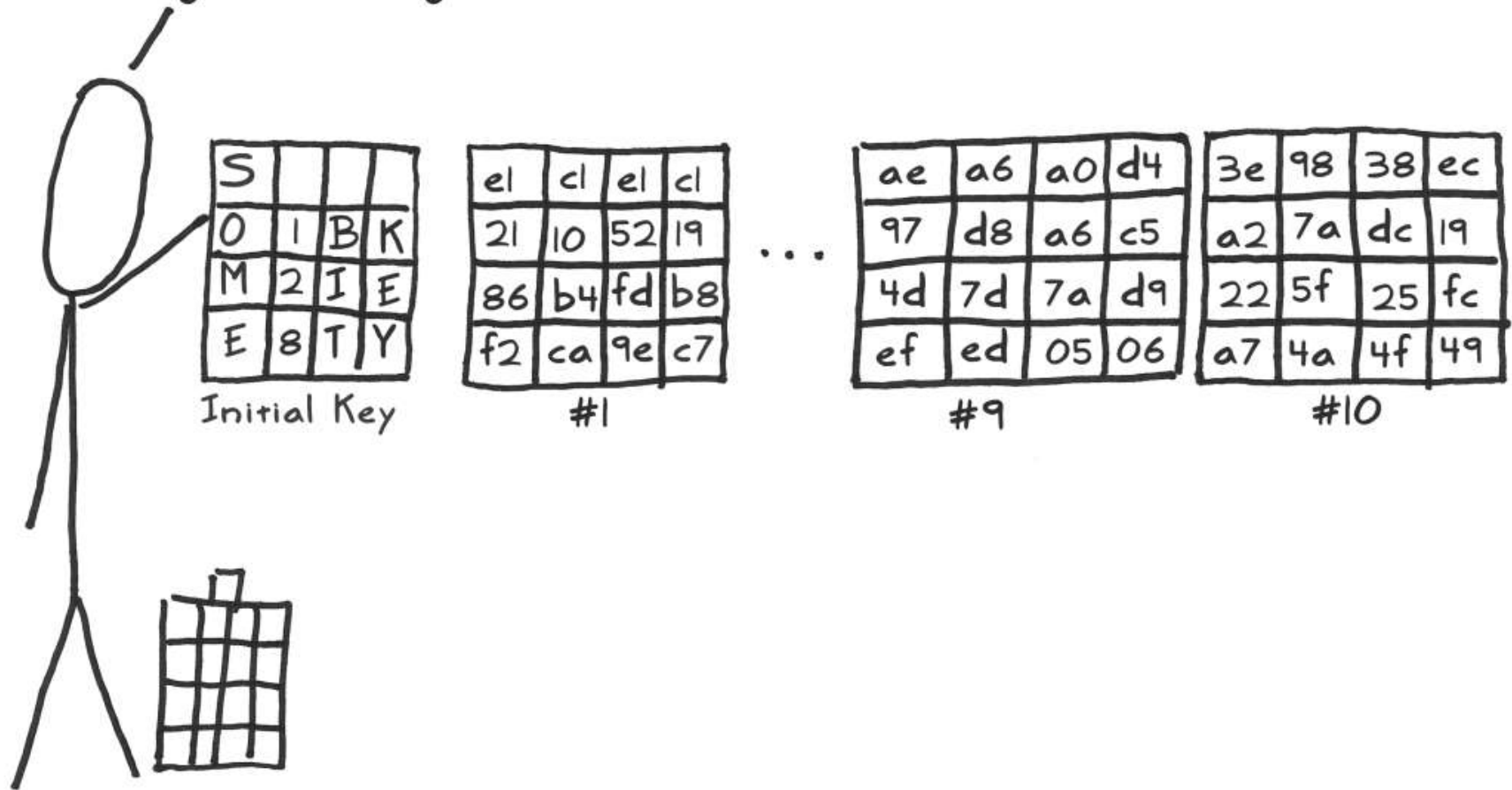


AES ♥ ⊕



Key Expansion: Part 1

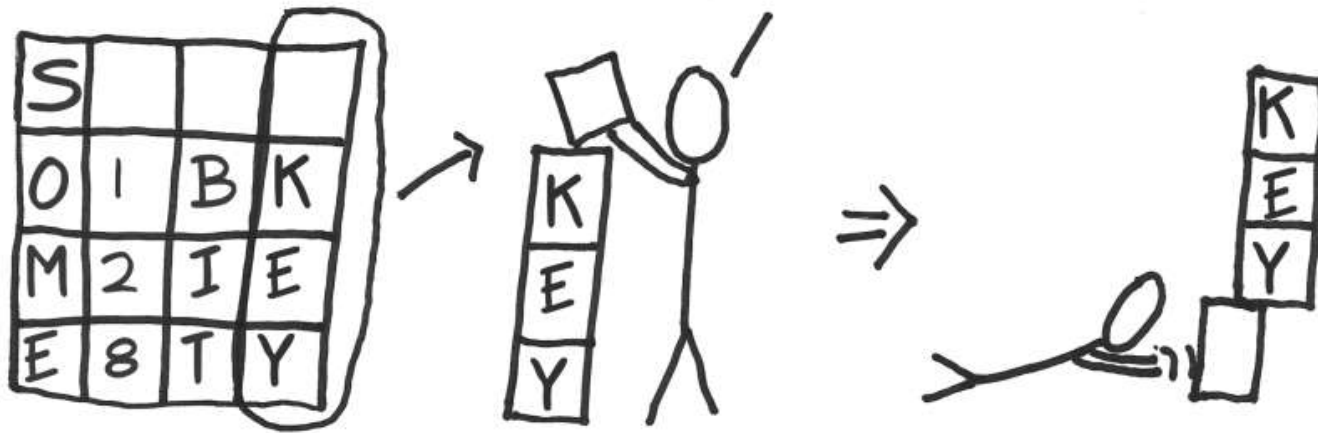
I need lots of keys for use in later rounds. I derive all of them from the initial key using a simple mixing technique that's really fast. Despite its critics,* it's good enough.



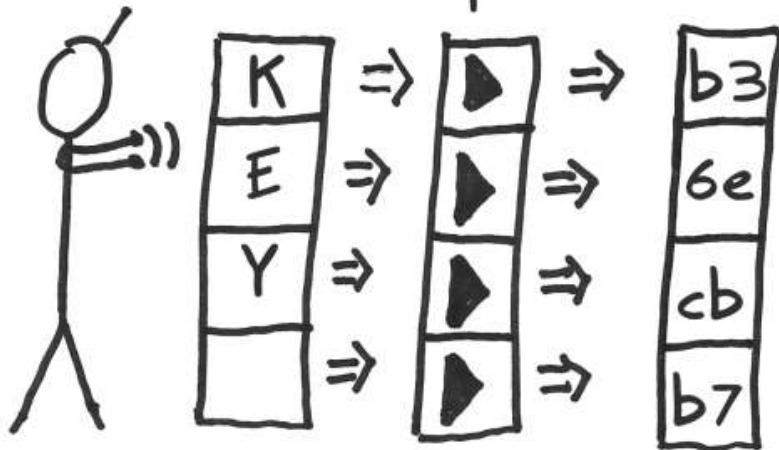
* By far, most complaints against AES's design focus on this simplicity.

Key Expansion: Part 2a

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

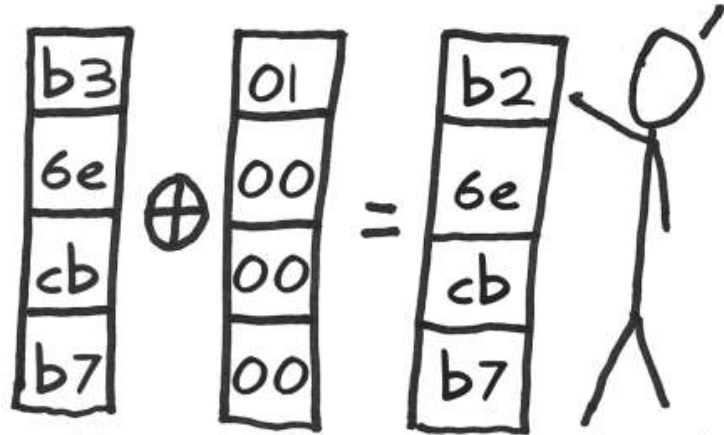


- ② Next, I run each byte through a substitution box that will map it to something else:

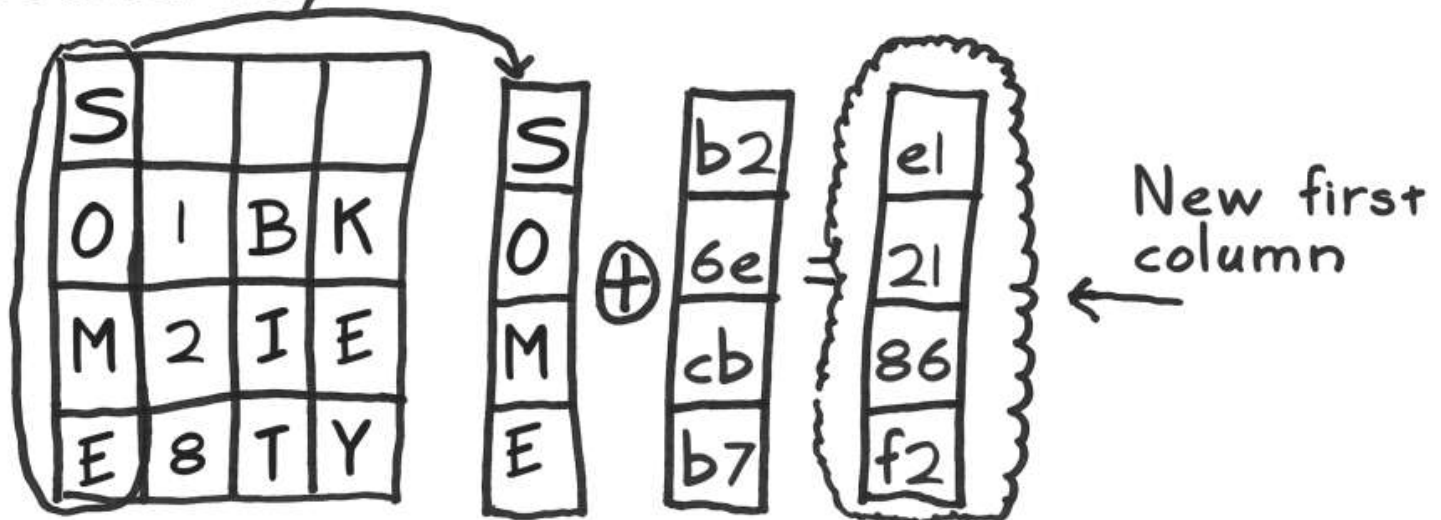


Key Expansion: Part 2b

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

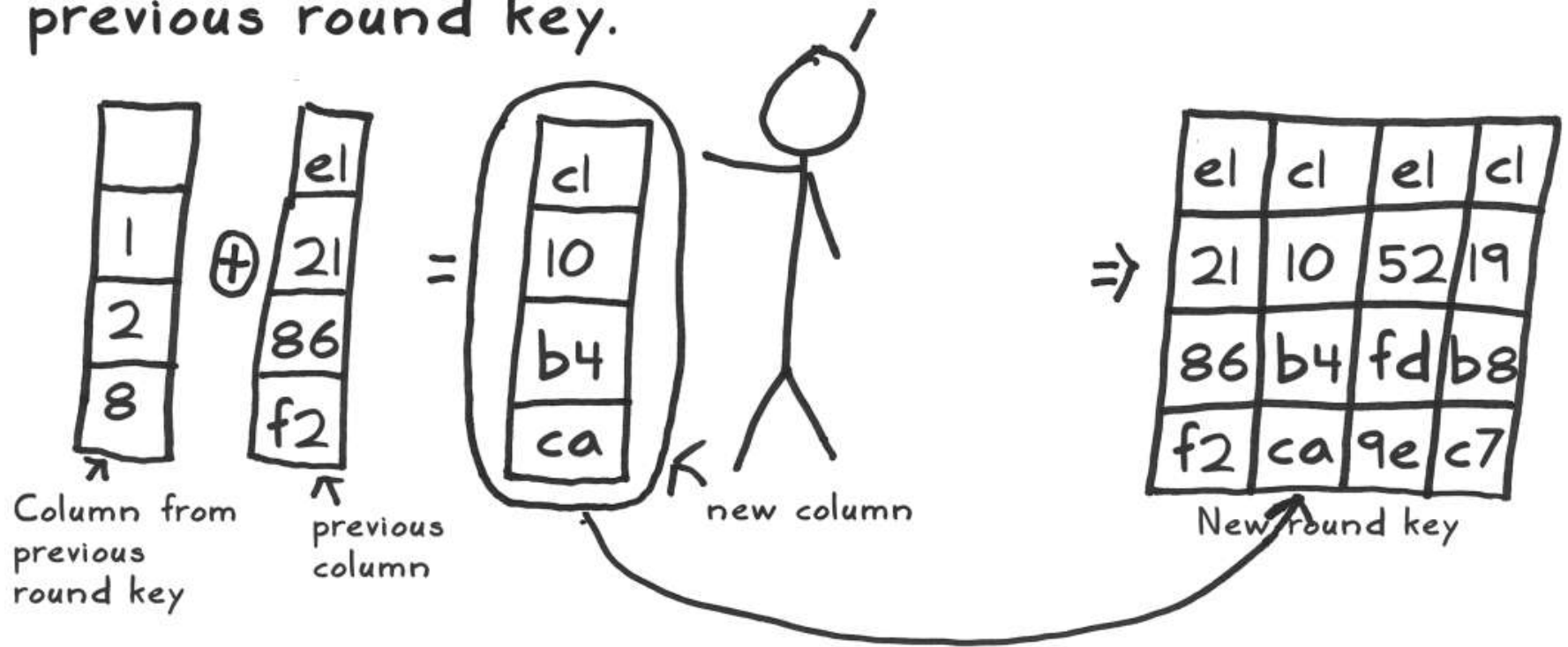


- ④ Finally, I xor it with the first column of the previous round key:



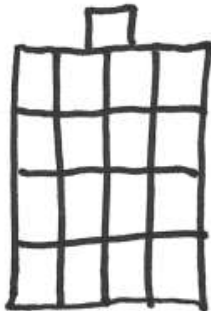
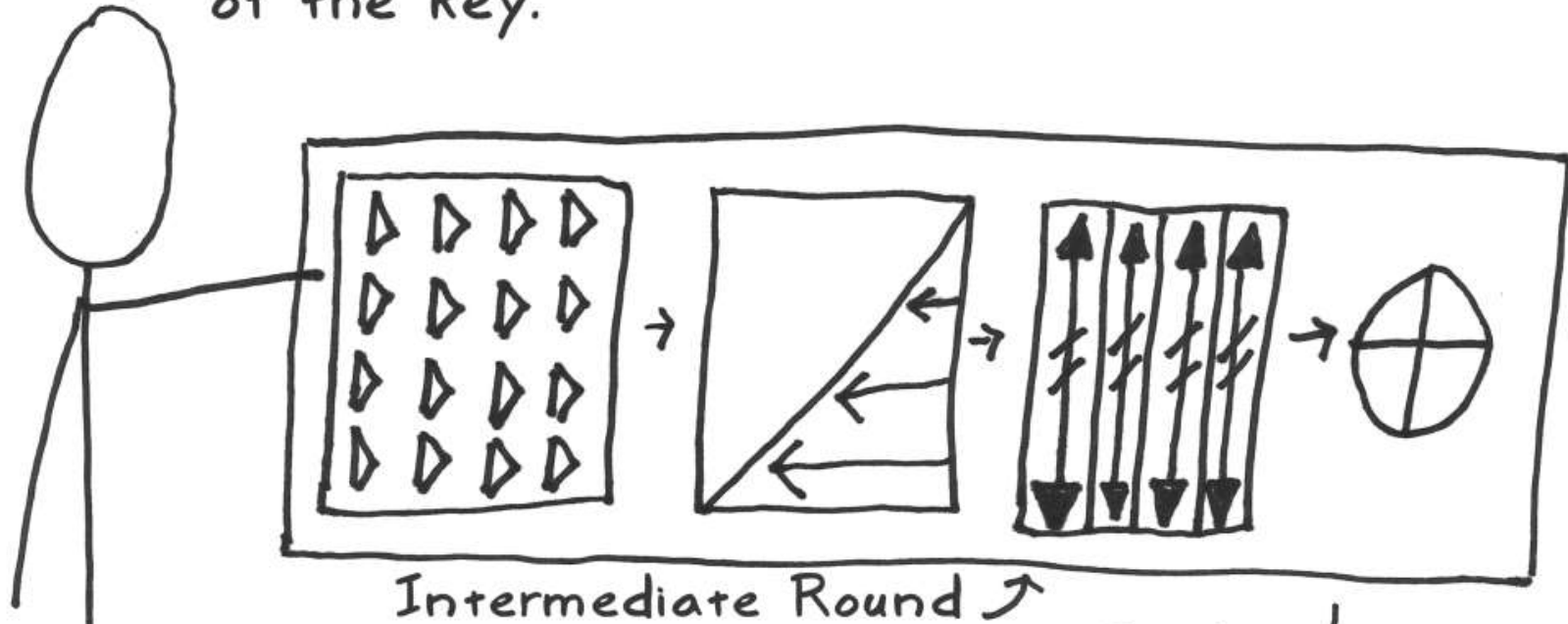
Key Expansion: Part 3

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.

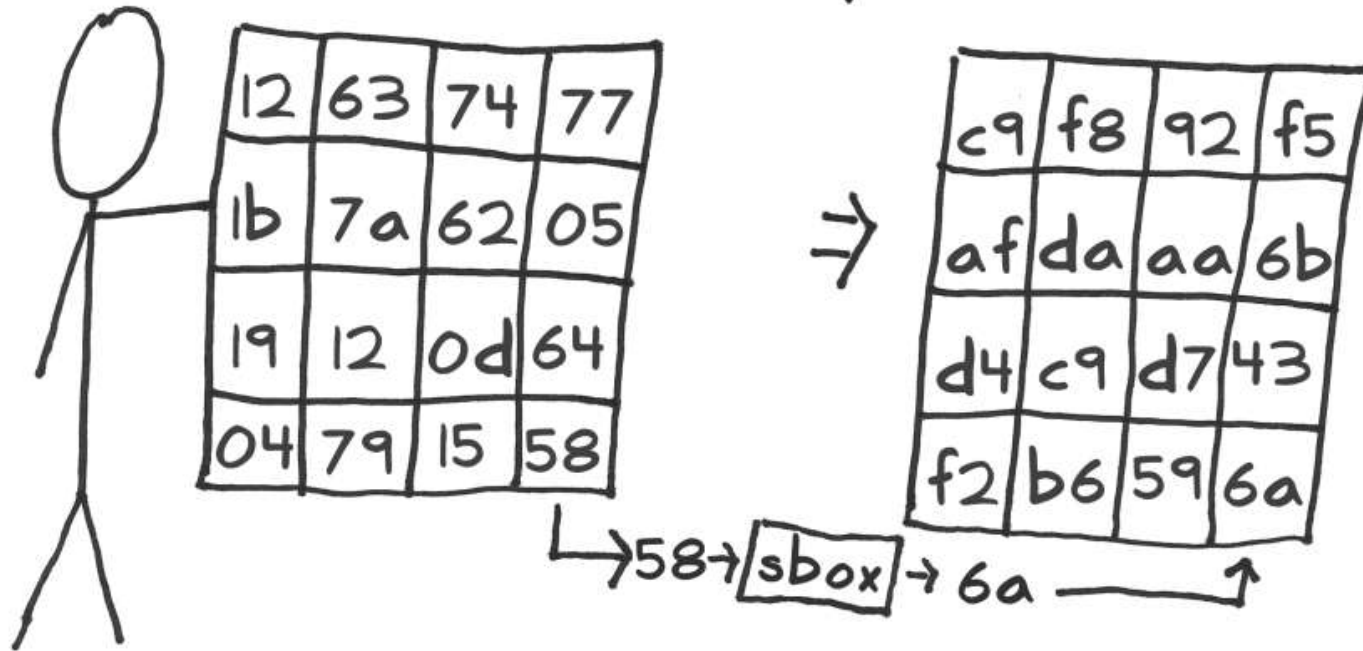
Next, I start the intermediate rounds. A round is just a series of steps I repeat several times. The number of repetitions depends on the size of the key.



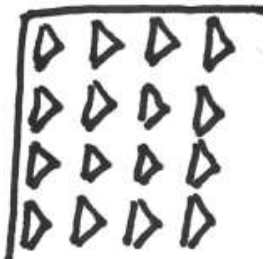
Round Repetitions	Key Size
9	128
11	192
13	256

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:



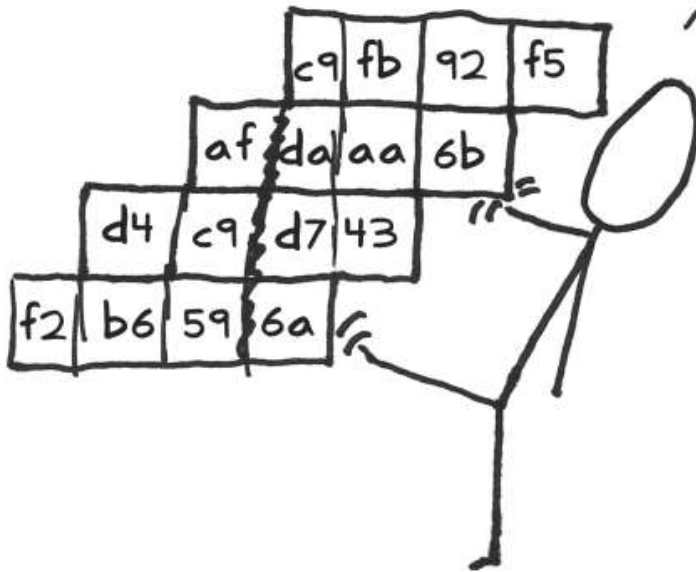
Denotes
"confusion"



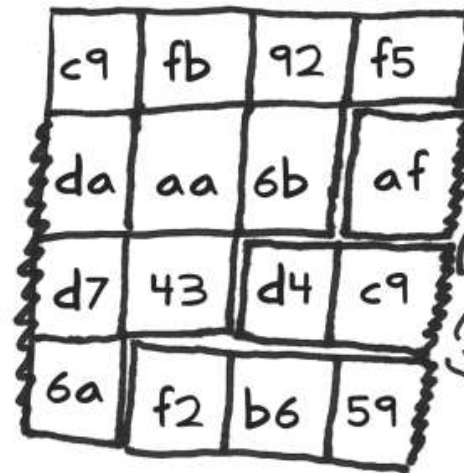
Applying Diffusion, Part 1: Shift Rows

Next I shift the rows to the left

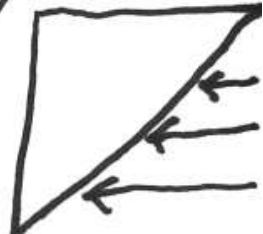
Hiiii yaah!



...and then wrap them around the other side



Denotes
permutation

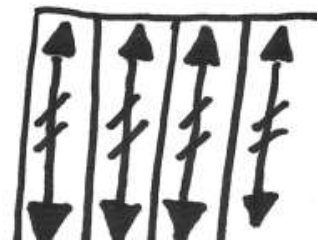


Applying Diffusion, Part 2: Mix Columns

c9	fb	92	f5
da	aa	6b	af
d7	43	d4	c9
6a	f2	b6	59


41	b9	e0	8b
6e	83	95	a9
18	da	8b	38
99	00	65	d0

I take each column and mix up the bits in it.



Applying Key Secrecy: Add Round Key

At the end of each round, I apply the next round key with an xor:



41	b9	e0	8b
6e	83	95	a9
18	da	8b	38
99	00	65	d0

 \oplus

e1	c1	e1	c1
21	10	52	19
86	b4	fd	b8
f2	ca	9e	c7

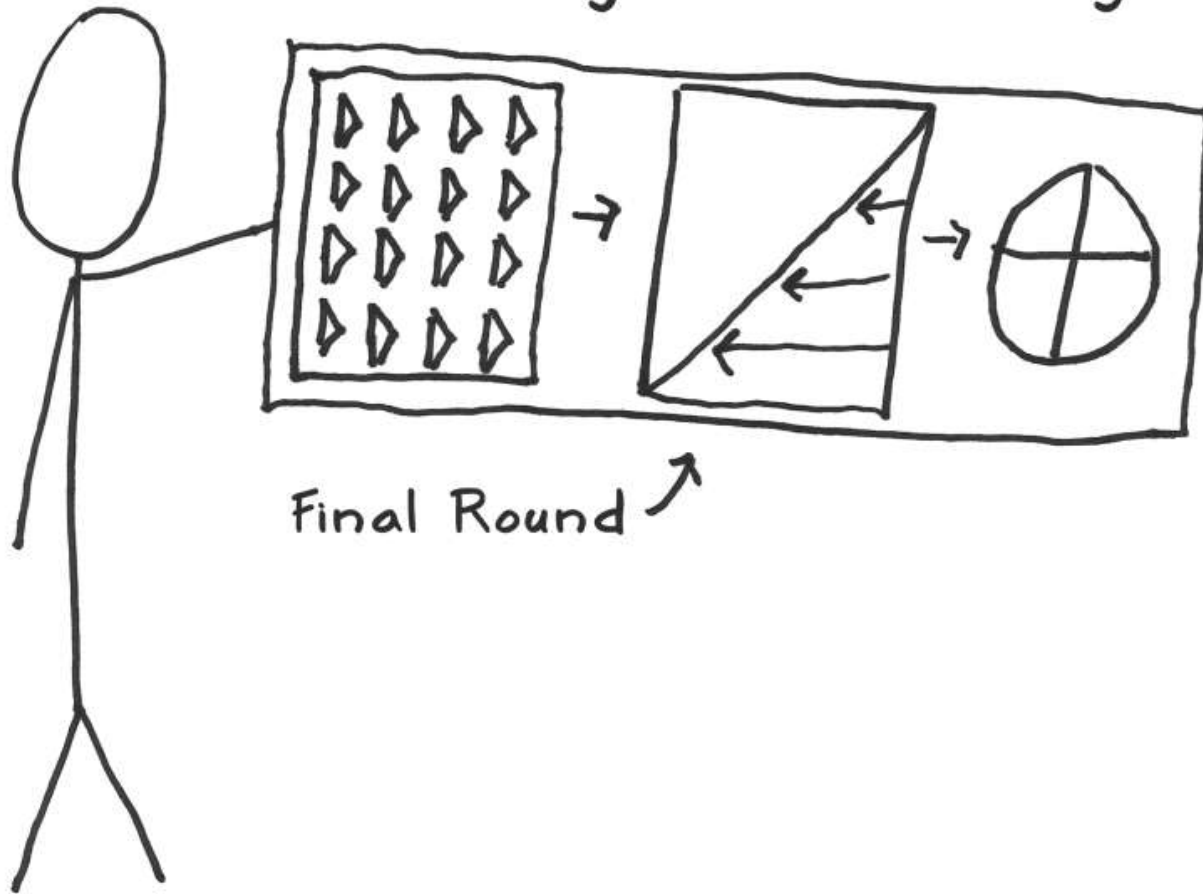
 $=$

a0	78	01	4a
4f	93	c7	b0
9e	6e	76	80
6b	ca	fb	17

$d0 \oplus c7 = 17$

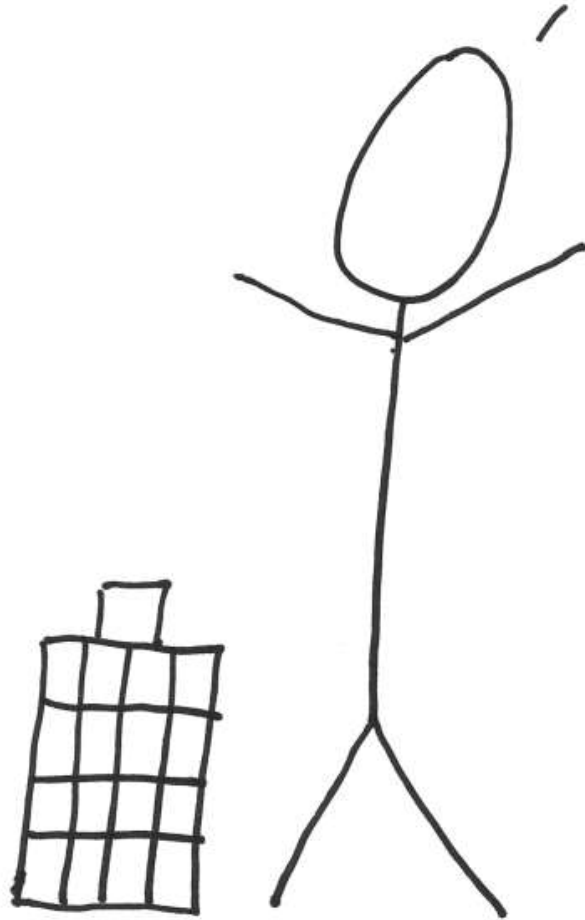


In the final round, I skip the "Mix Columns" step since it wouldn't increase security* and would just slow things down:

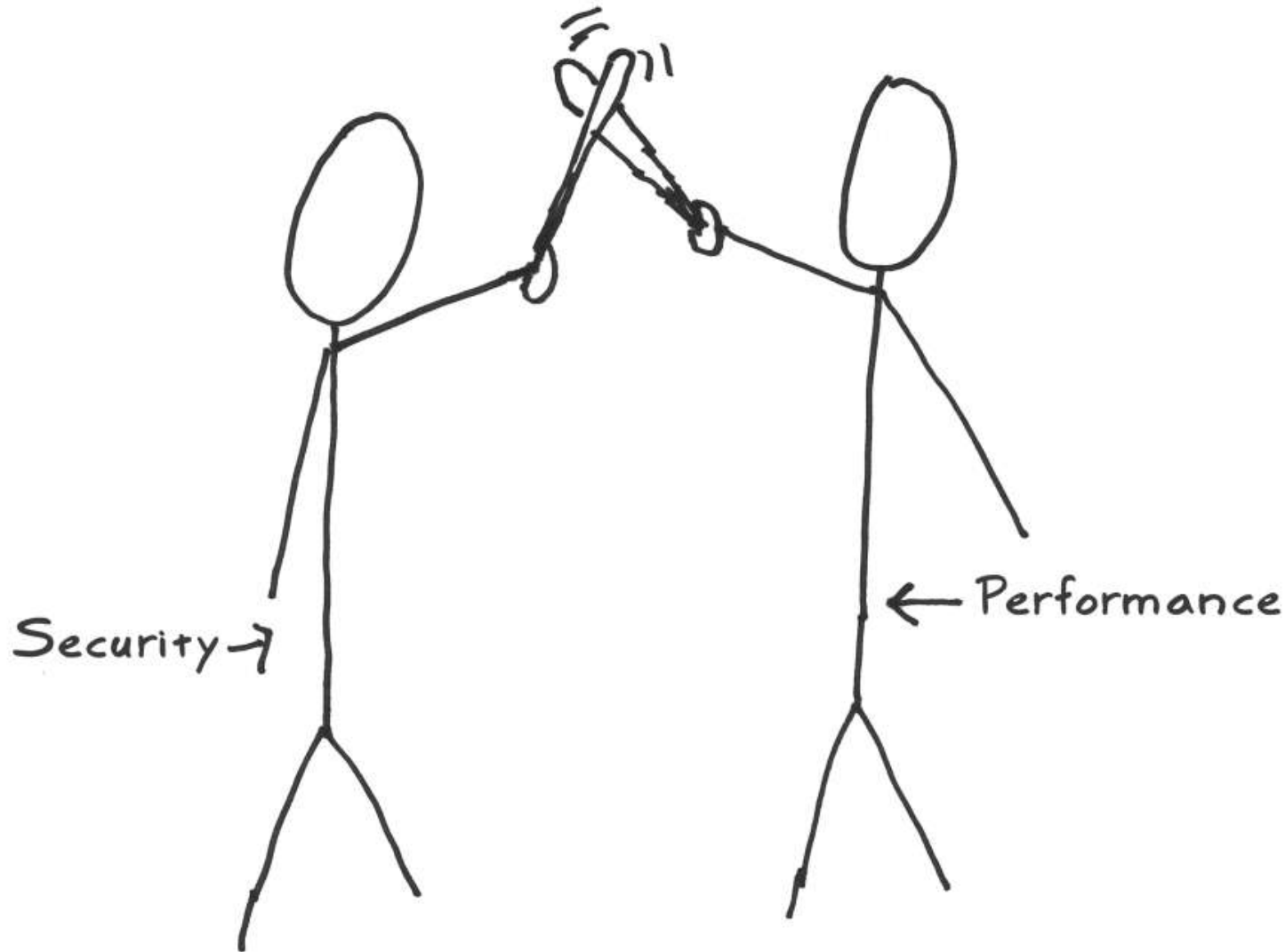


*The diffusion it would provide wouldn't go to the next round.

...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!

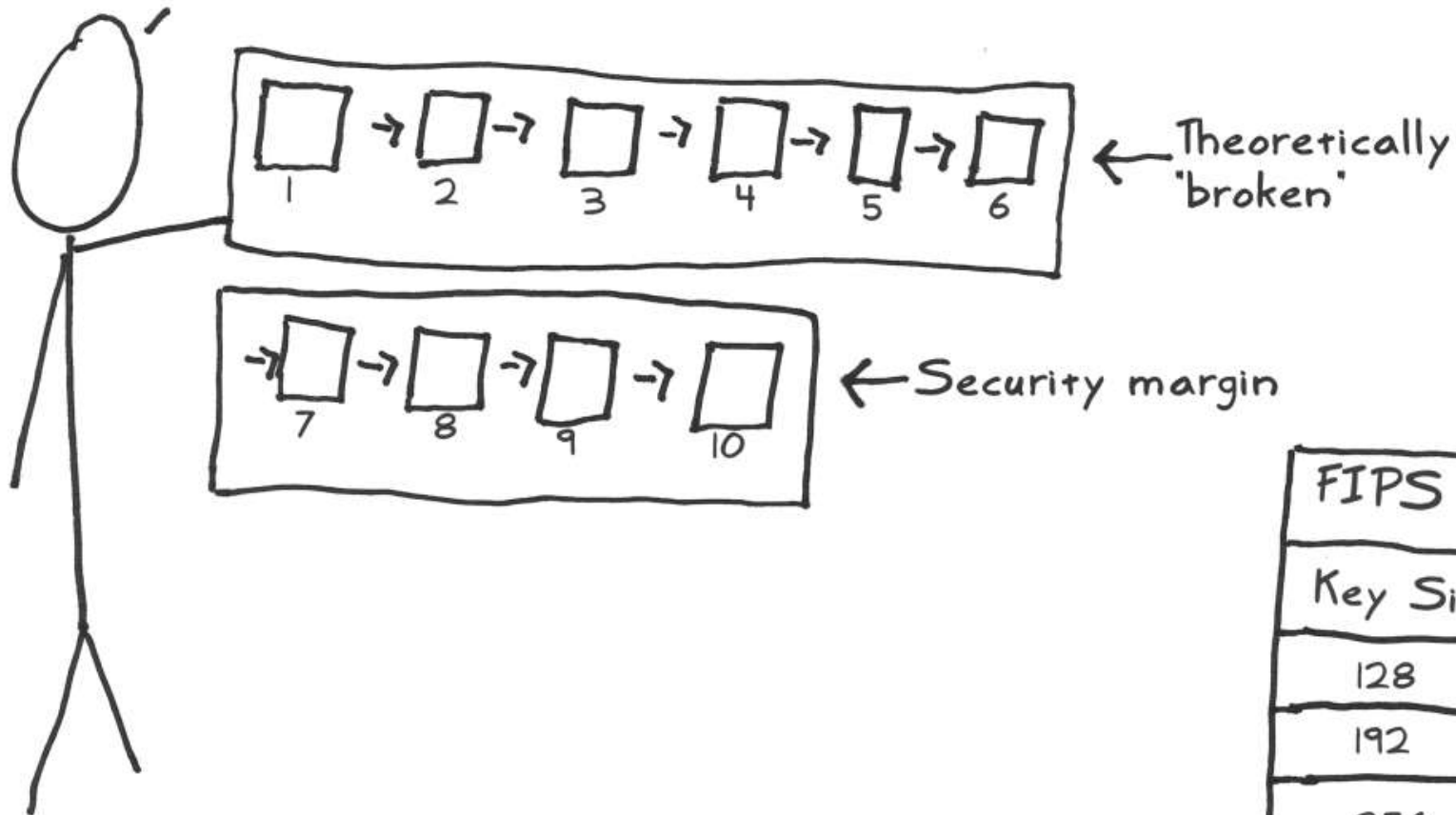


Determining the number of rounds always involves several tradeoffs.



'Security always comes at a cost to performance' - Vincent Rijmen

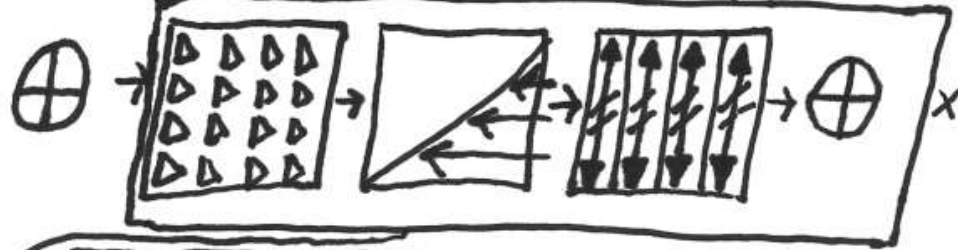
When I was being developed, a clever guy was able to find a shortcut path through 6 rounds. That's not good! If you look carefully, you'll see that each bit of a round's output depends on every bit from two rounds ago. To increase this diffusion 'avalanche,' I added 4 extra rounds. This is my 'security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



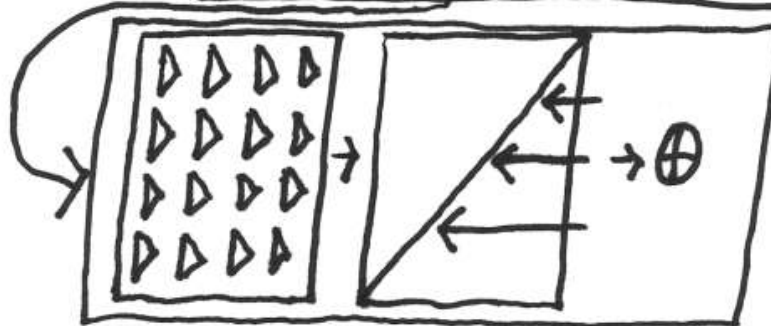
FIPS 197 Spec	
Key Size	Rounds
128	10
192	12
256	14

So in pictures, we have this:

Intermediate Round ↴

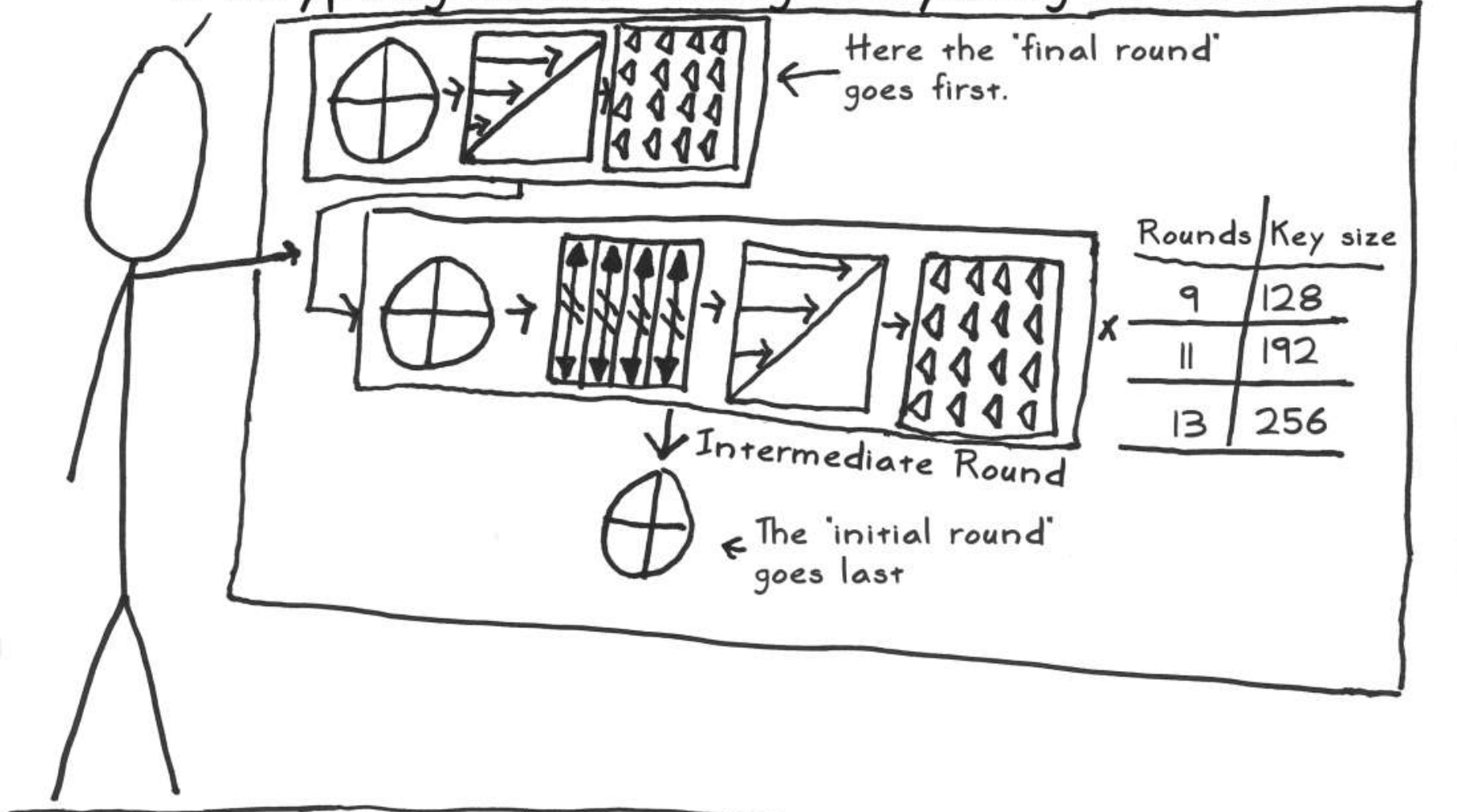


Rounds	Key Size
9	128
11	192
13	256



Final Round ↴

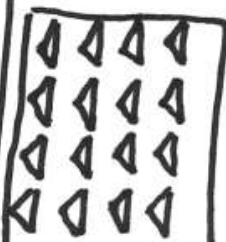
Decrypting means doing everything in reverse



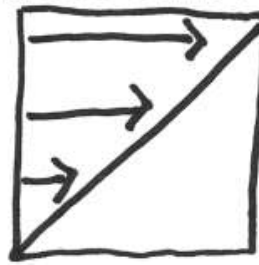
Add Round Key Inverse



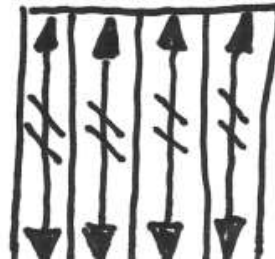
Inverse Substitute Bytes



Inverse Shift Rows

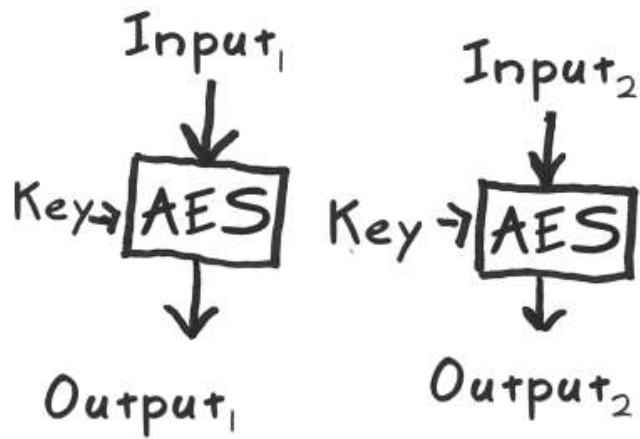


Inverse Mix Columns



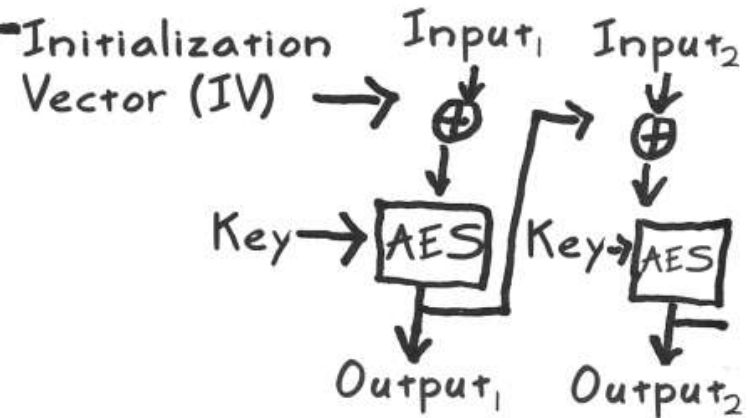
One last tidbit: I shouldn't be used as-is, but rather as a building block to a decent "mode."

Electronic Codebook Mode (ECB)



BAD!

Cipher-block Chaining (CBC)

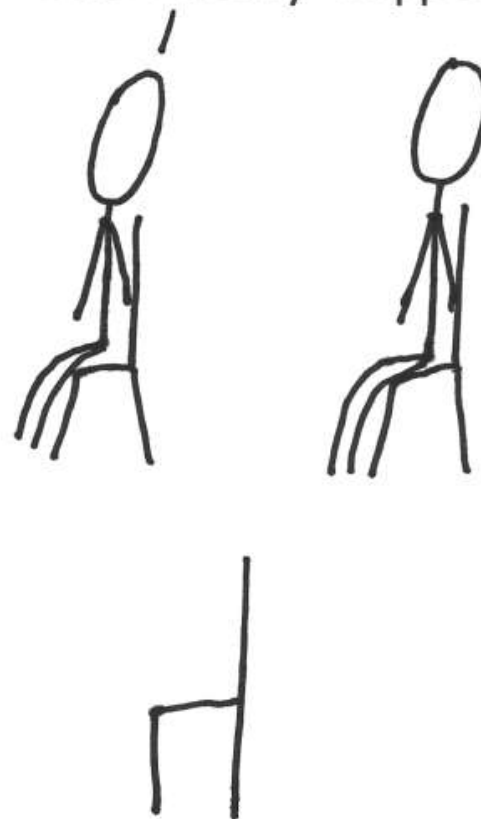


Better

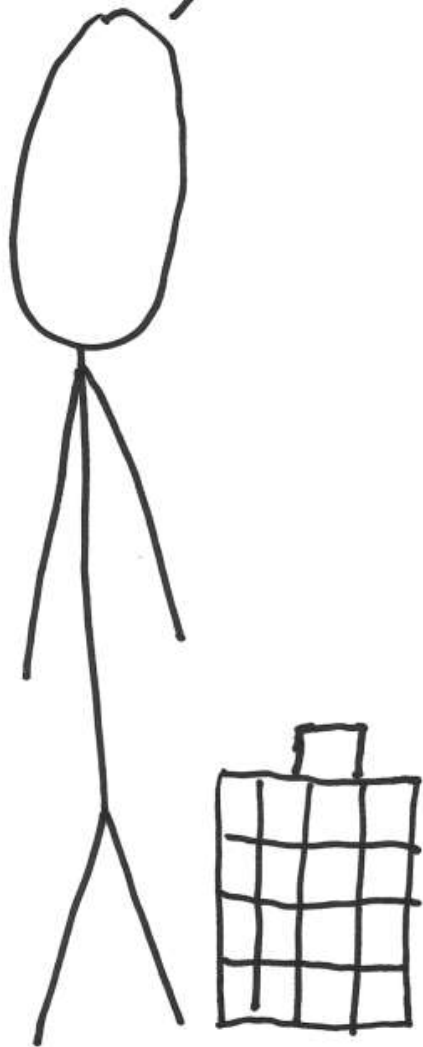
Make sense? Did that
answer your question?



Almost...except you just
waved your hands and
used weird analogies.
What really happens?



Another great question! It's
not hard, but... it involves
a little... math.



I'm game.
Bring it on!!



Math is hard!
Let's go shopping!

