

DES and AES

Diego F. Aranha

Institute of Computing
UNICAMP

Introduction

Objectives:

- Visit practical constructions of block ciphers.
- Apply concepts discussed in previous classes.

Introduction

Objectives:

- Visit practical constructions of block ciphers.
- Apply concepts discussed in previous classes.

Hidden intentions:

- Observe how standardization processes can influence cryptographic design.

Feistel network

Definition

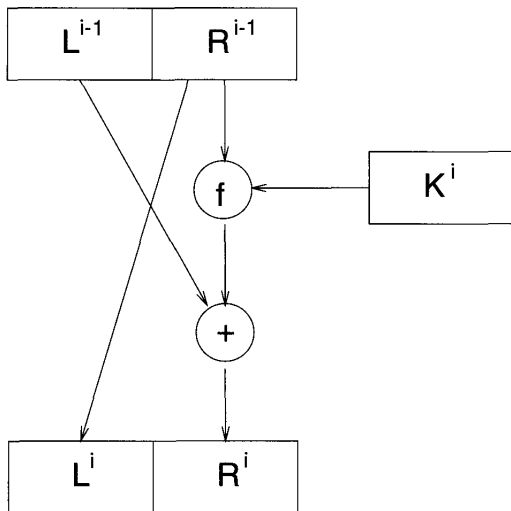
A **Feistel network** is a special case of iterated cipher where $g : \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$ has the form $g(L^{i-1}R^{i-1}, K^i) = (L^iR^i)$, where

$$L^i = R^{i-1}, R^i = L^{i-1} \oplus f(R^{i-1}, K^i).$$

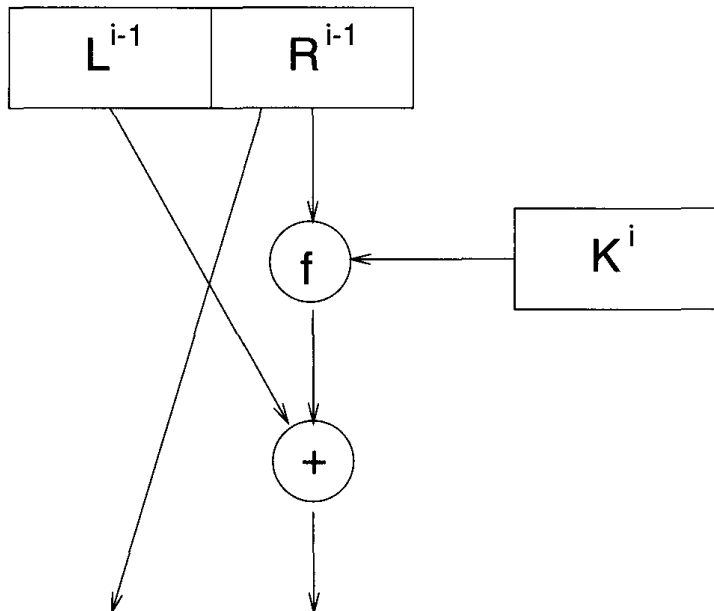
Characteristics:

- Cipher state w^i split into two halves L^i e R^i of the same size.
- There are no restrictions to function f , because g is invertible by definition.

Feistel network



Feistel network



Feistel network

Encryption algorithm

Input: $x, \pi, f_i, \langle K^1, K^2, \dots, K^r \rangle$.

- 1 $L^0 \parallel R^0 \leftarrow x$
- 2 **for** $i \leftarrow 1$ **to** r **do**
 - 2.1 $L^i \leftarrow R^{i-1}$
 - 2.2 $R^i \leftarrow L^{i-1} \oplus f_i(K^i, R^{i-1})$ (compression!)
- 3 **return** $y \leftarrow L^r \parallel R^r$

Problem: if f_i is not invertible, how to decrypt?

Feistel network

Encryption algorithm

Input: $x, \pi, f_i, \langle K^1, K^2, \dots, K^r \rangle$.

- 1 $L^0 \parallel R^0 \leftarrow x$
- 2 **for** $i \leftarrow 1$ **to** r **do**
 - 2.1 $L^i \leftarrow R^{i-1}$
 - 2.2 $R^i \leftarrow L^{i-1} \oplus f_i(K^i, R^{i-1})$ (compression!)
- 3 **return** $y \leftarrow L^r \parallel R^r$

Problem: if f_i is not invertible, how to decrypt?

Solution: Round i can be inverted:

Feistel network

Encryption algorithm

Input: $x, \pi, f_i, \langle K^1, K^2, \dots, K^r \rangle$.

- 1 $L^0 \parallel R^0 \leftarrow x$
- 2 **for** $i \leftarrow 1$ **to** r **do**
 - 2.1 $L^i \leftarrow R^{i-1}$
 - 2.2 $R^i \leftarrow L^{i-1} \oplus f_i(K^i, R^{i-1})$ (compression!)
- 3 **return** $y \leftarrow L^r \parallel R^r$

Problem: if f_i is not invertible, how to decrypt?

Solution: Round i can be inverted:

$$R^{i-1} = L^i, L^{i-1} = R^i \oplus f_i(K^i, R^{i-1})$$

Data Encryption Standard (IBM, 1975)

History:

- Standard defined jointly by IBM and NSA (*National Security Agency*).
- Based on the *Lucifer* block cipher, designed by Feistel.
- It was proposed to have a lifetime of 10-15 years, but this was much longer in practice.

Important: NSA interference in the standardization effort?

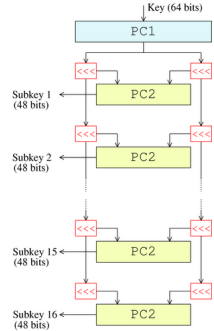
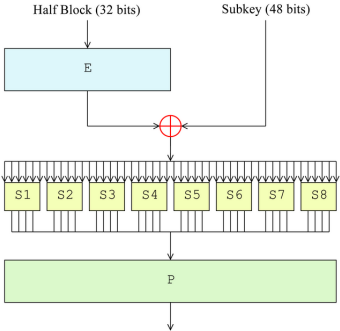
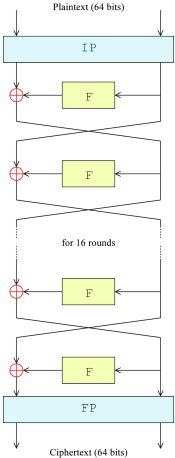
Data Encryption Standard (IBM, 1975)

Features:

- $\mathcal{M} = (\mathbb{Z}_2)^{64}, \mathcal{K} = (\mathbb{Z}_2)^{56}$.
- 8-bit parity in the key.
- Permutation $L^0R^0 = \mathbf{IP}(x)$ is applied at the beginning.
- Inverse permutation $y = \mathbf{IP}^{-1}(R^{16}L^{16})$ is applied at the end.
- $Nr = 16, lm = 64$.
- Function f has format $\{0, 1\}^{32} \times \{0, 1\}^{48} \rightarrow \{0, 1\}^{32}$.
- Subkeys $(K^1, K^2, \dots, K^{16})$ are permutations of the bits from K .

Question: Why is the purpose of \mathbf{IP} e \mathbf{IP}^{-1} ?

Data Encryption Standard (IBM, 1975)



Data Encryption Standard (IBM, 1975)

Analysis:

- Only the substitution boxes are non-linear.
- They are speculated to be vulnerable or to store a backdoor.
- They were actually chosen to resist differential cryptanalysis.
- 20 years later, researchers from academia independently discovered the attack.
- Any other problem?

Data Encryption Standard (IBM, 1975)

Analysis:

- Only the substitution boxes are non-linear.
- They are speculated to be vulnerable or to store a backdoor.
- They were actually chosen to resist differential cryptanalysis.
- 20 years later, researchers from academia independently discovered the attack.
- **Small key space!**

Data Encryption Standard (IBM, 1975)

Cryptanalysis:

- 1977: special-purpose machine costing 20 million was capable of brute-forcing the key space in a single day.
- CRYPTO 1993: special machine costing 100,000 capable of exhaustive search in $\frac{7}{2}$ of a day of a million in 1.5 day.
- 1994: Linear cryptanalysis needs 2^{43} pairs (x, y) .
- 1998: EFF builds *DES cracker* costing 250,000 and capable of exhaustive search in 56 hours.

Data Encryption Standard (IBM, 1975)

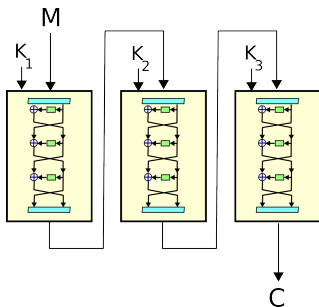
NSA wanted a secure cipher to everyone, but not *too* secure:

- Substitution boxes were chosen to improve resistance against differential cryptanalysis.
- Key length was reduced from 64 to 56 bits.

Conclusion: Never trust cryptographic standards to intelligence agencies (see DUAL_EC_DRBG and TLS export-grade cipher suites!)

Warning: Avoid DES at all costs, use AES!

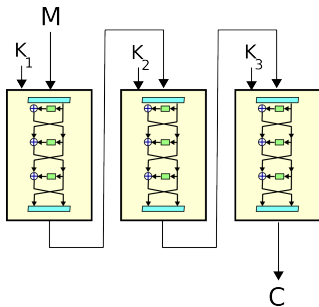
Triple DES



Encryption and decryption:

- $e_K(x) = e_{K_3}(d_{K_2}(e_{K_1}(x)))$
- $d_K(y) = d_{K_1}(e_{K_2}(d_{K_3}(y)))$

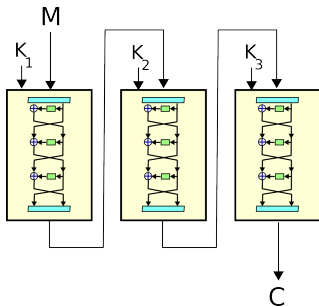
Triple DES



Variants:

- $K_1 = K_2 = K_3$ (56 bits of security).
- $K_1 = K_3 \neq K_2$ (112 bits of security).
- $K_1 \neq K_2 \neq K_3$ (168 bits of security).

Triple DES



Variants:

- $K_1 = K_2 = K_3$ (56 bits of security).
- $K_1 = K_3 \neq K_2$ (80 bits of security).
- $K_1 \neq K_2 \neq K_3$ (112 bits of security).

Meet-in-the-middle attack

Definition

It is a known-plaintext attack what exploits the naive intuition that double encryption with different keys is equivalent to encrypting with a key two times longer.

Assumptions:

- Encryption function is $e_K(x) = e_{K_1}(e_{K_2}(x))$.
- Decryption function is $d_K(y) = d_{K_2}(d_{K_1}(y))$.

Meet-in-the-middle attack

Algorithm

Input: Plaintext and ciphertext pair (x, y) .

Output: Key K .

- 1 Attacker computes encryptions $y' = e_{K_1}(x)$ for all keys K_1 .
- 2 Attacker stores all y' in a table.
- 3 Attacker computes decryptions $x' = d_{K_2}(y)$ for all keys K_2 .
- 4 If $x' = y'$, attacker finds correct key $K = (K_1, K_2)$.

Important: What is the complexity?

Meet-in-the-middle attack

Algorithm

Input: Plaintext and ciphertext pair (x, y) .

Output: Key K .

- 1 Attacker computes encryptions $y' = e_{K_1}(x)$ for all keys K_1 .
- 2 Attacker stores all y' in a table.
- 3 Attacker computes decryptions $x' = d_{K_2}(y)$ for all keys K_2 .
- 4 If $x' = y'$, attacker finds correct key $K = (K_1, K_2)$.

Important: 2^{n+1} encryptions and storage of 2^n ciphertexts!

Advanced Encryption Standard (2001, NIST)

History:

- Public challenge.
- 21 submissions, 15 accepted.
- 5 finalists: *MARS*, *RC6*, *Rijndael*, *Serpent* e *Twofish*.
- Cipher *Rijndael* selected as the standard.

Criteria:

- Security.
- Computational cost in software and hardware.
- Simplicity and flexibility in the design.

Advanced Encryption Standard (2001, NIST)

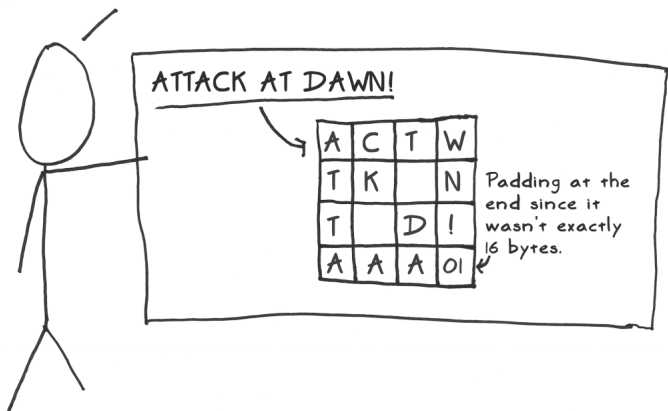
Features:

- Three security levels: 128, 192 e 256 bits.
- $\mathcal{M} = (\mathbb{Z}_2)^{128}$, $\mathcal{K} = (\mathbb{Z}_2)^{128}, (\mathbb{Z}_2)^{192}, (\mathbb{Z}_2)^{256}$.
- $Nr = 10, 12, 14$, respectively, $lm = 128$.
- Follows the SPN paradigm.

Curiosity: Implemented as native instruction in modern Intel processors!

Advanced Encryption Standard (2001, NIST)

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

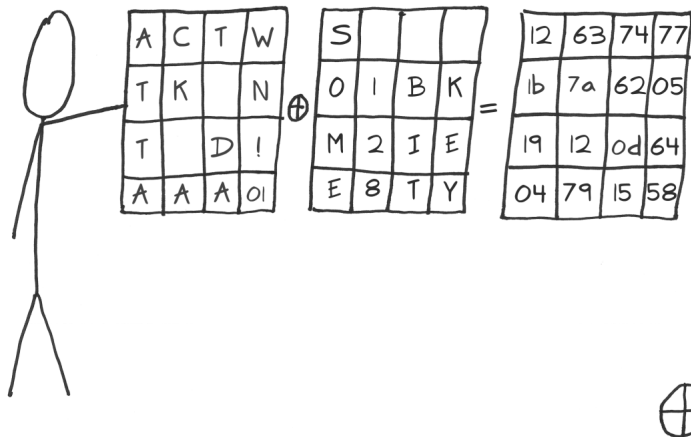


* This is the 'state matrix' that I carry with me at all times.

Credit: Jeff Moser

Advanced Encryption Standard (2001, NIST)

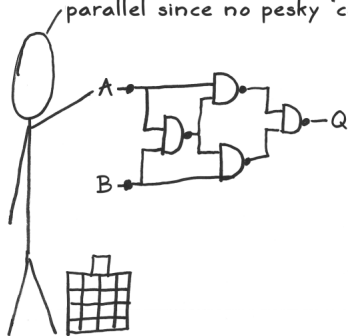
The initial round has me xor each input byte with the corresponding byte of the first round key.



Advanced Encryption Standard (2001, NIST)

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.

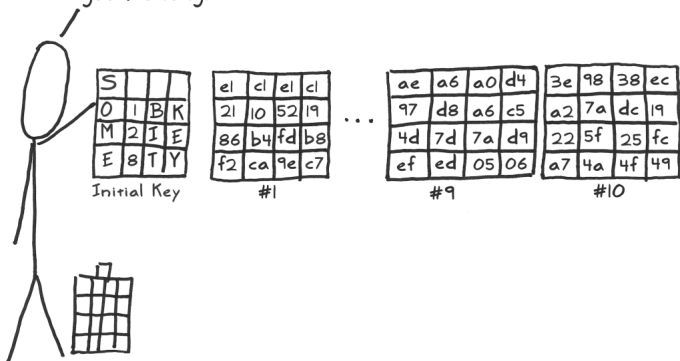


AES ♥ ⊕

Advanced Encryption Standard (2001, NIST)

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.

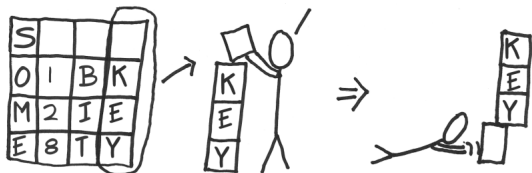


* By far, most complaints against AES's design focus on this simplicity.

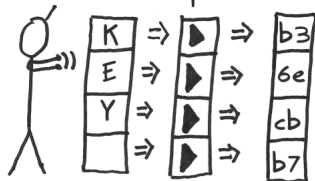
Advanced Encryption Standard (2001, NIST)

Key Expansion: Part 2a

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



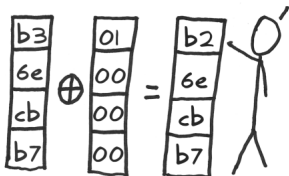
- ② Next, I run each byte through a substitution box that will map it to something else:



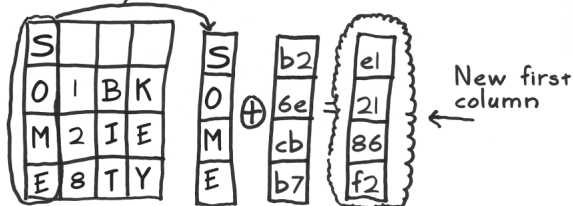
Advanced Encryption Standard (2001, NIST)

Key Expansion: Part 2b

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



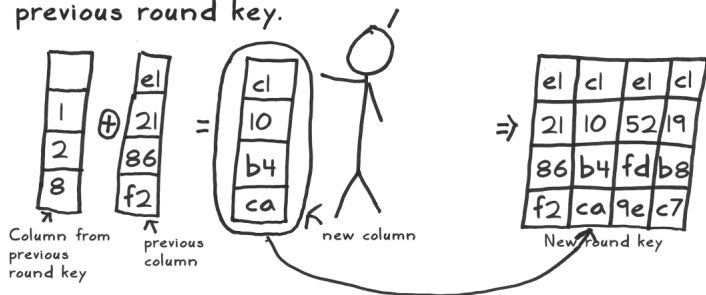
- ④ Finally, I xor it with the first column of the previous round key:



Advanced Encryption Standard (2001, NIST)

Key Expansion: Part 3

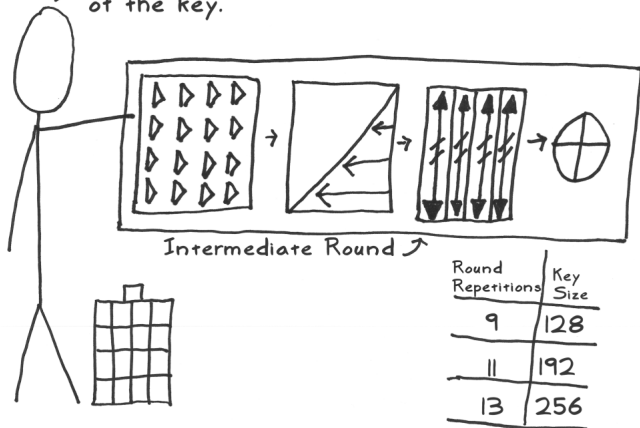
The other columns are super-easy,* I just xor the previous column with the same column of the previous round key.



* Note that 256 bit keys are slightly more complicated.

Advanced Encryption Standard (2001, NIST)

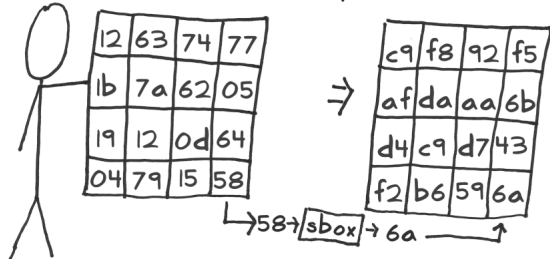
Next, I start the intermediate rounds. A round is just a series of steps I repeat several times. The number of repetitions depends on the size of the key.



Advanced Encryption Standard (2001, NIST)

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'

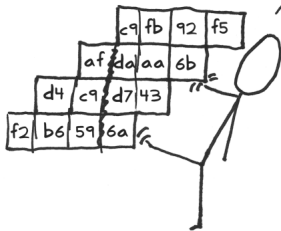


Advanced Encryption Standard (2001, NIST)

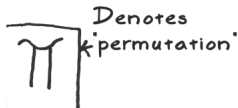
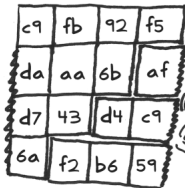
Applying Diffusion, Part I: Shift Rows

Next I shift the rows to the left

Hiiii yaah!

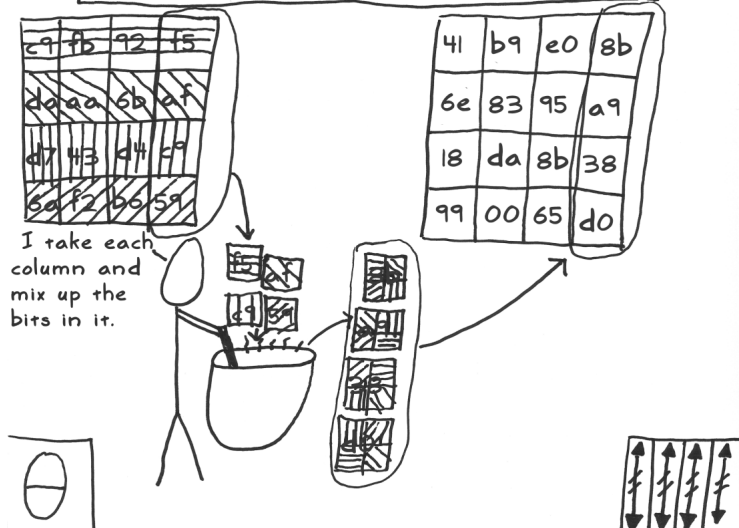


...and then wrap them around the other side



Advanced Encryption Standard (2001, NIST)

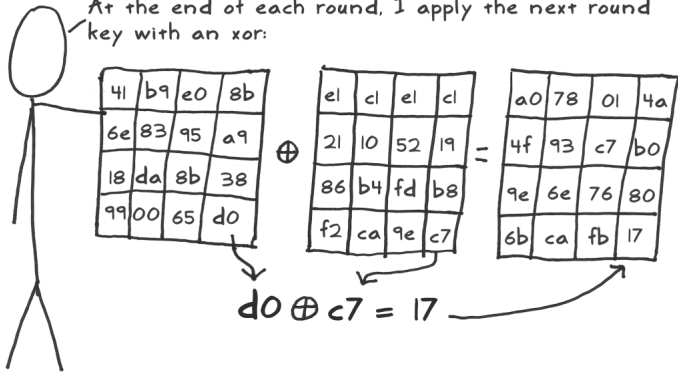
Applying Diffusion, Part 2: Mix Columns



Advanced Encryption Standard (2001, NIST)

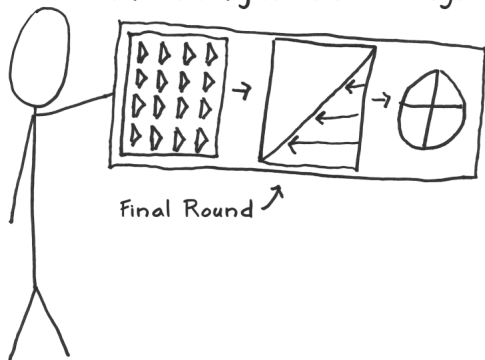
Applying Key Secrecy: Add Round Key

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



Advanced Encryption Standard (2001, NIST)

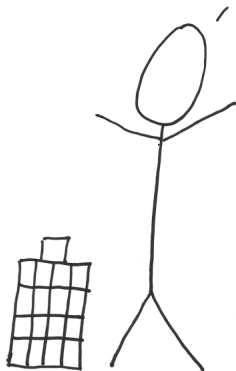
In the final round, I skip the 'Mix Columns' step since it wouldn't increase security* and would just slow things down:



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

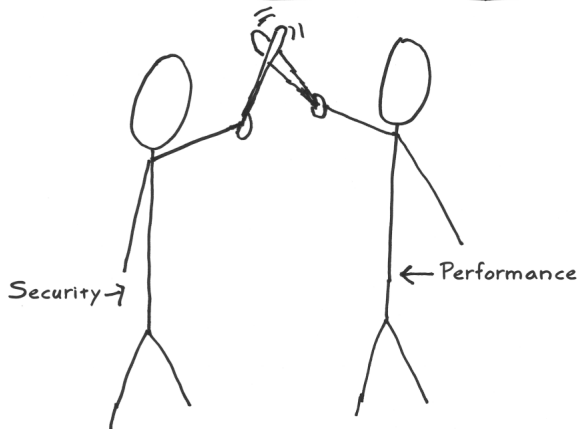
Advanced Encryption Standard (2001, NIST)

...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!



Advanced Encryption Standard (2001, NIST)

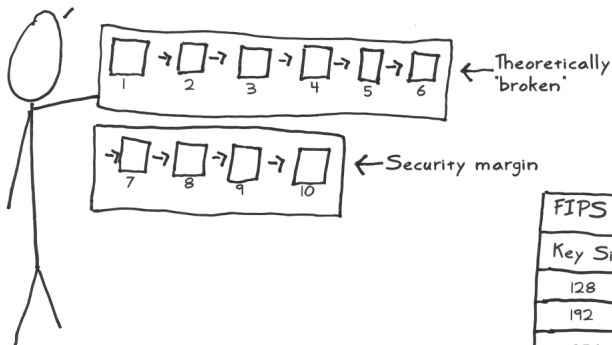
Determining the number of rounds always involves several tradeoffs.



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

Advanced Encryption Standard (2001, NIST)

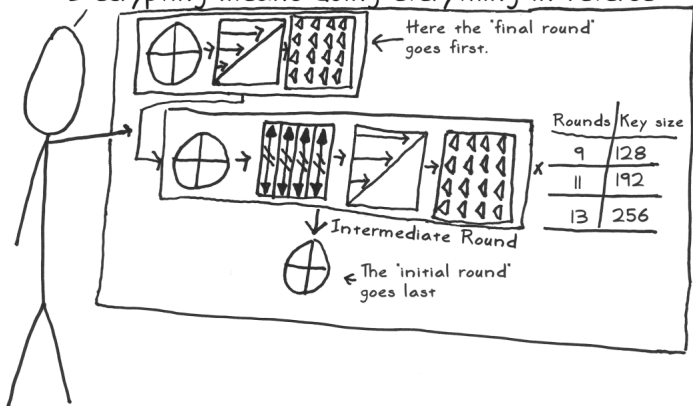
When I was being developed, a clever guy was able to find a shortcut path through 6 rounds. That's not good! If you look carefully, you'll see that each bit of a round's output depends on every bit from two rounds ago. To increase this diffusion 'avalanche,' I added 4 extra rounds. This is my 'security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



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

Advanced Encryption Standard (2001, NIST)

Decrypting means doing everything in reverse



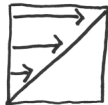
Add Round Key Inverse



Inverse Substitute Bytes



Inverse Shift Rows

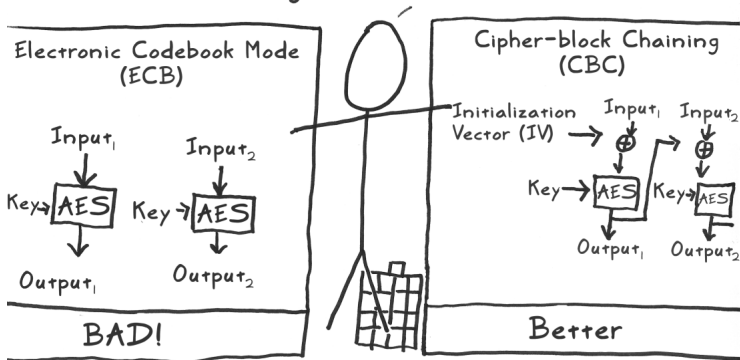


Inverse Mix Columns



Advanced Encryption Standard (2001, NIST)

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



Advanced Encryption Standard (2001, NIST)

