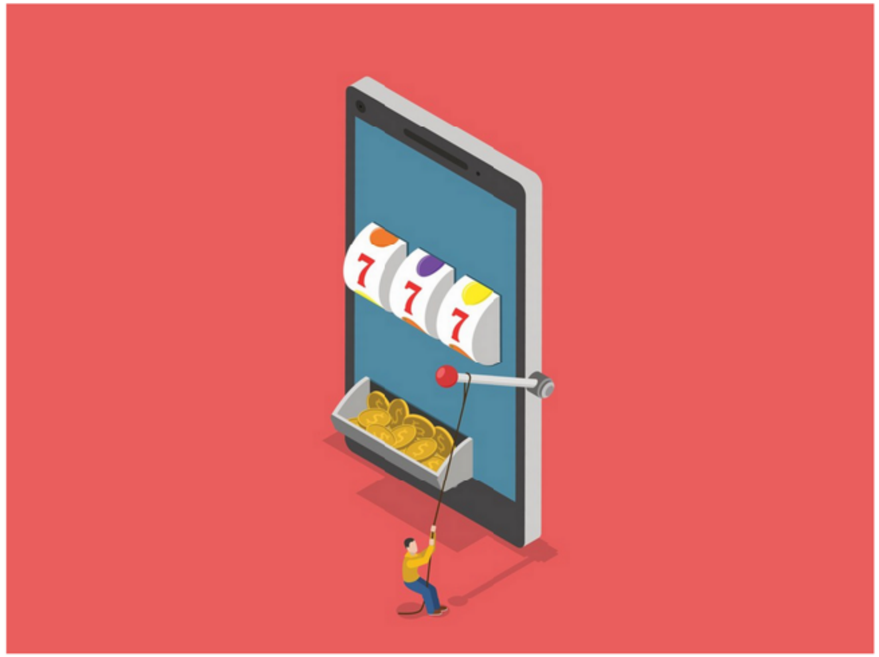
Lecture 3: Block ciphers

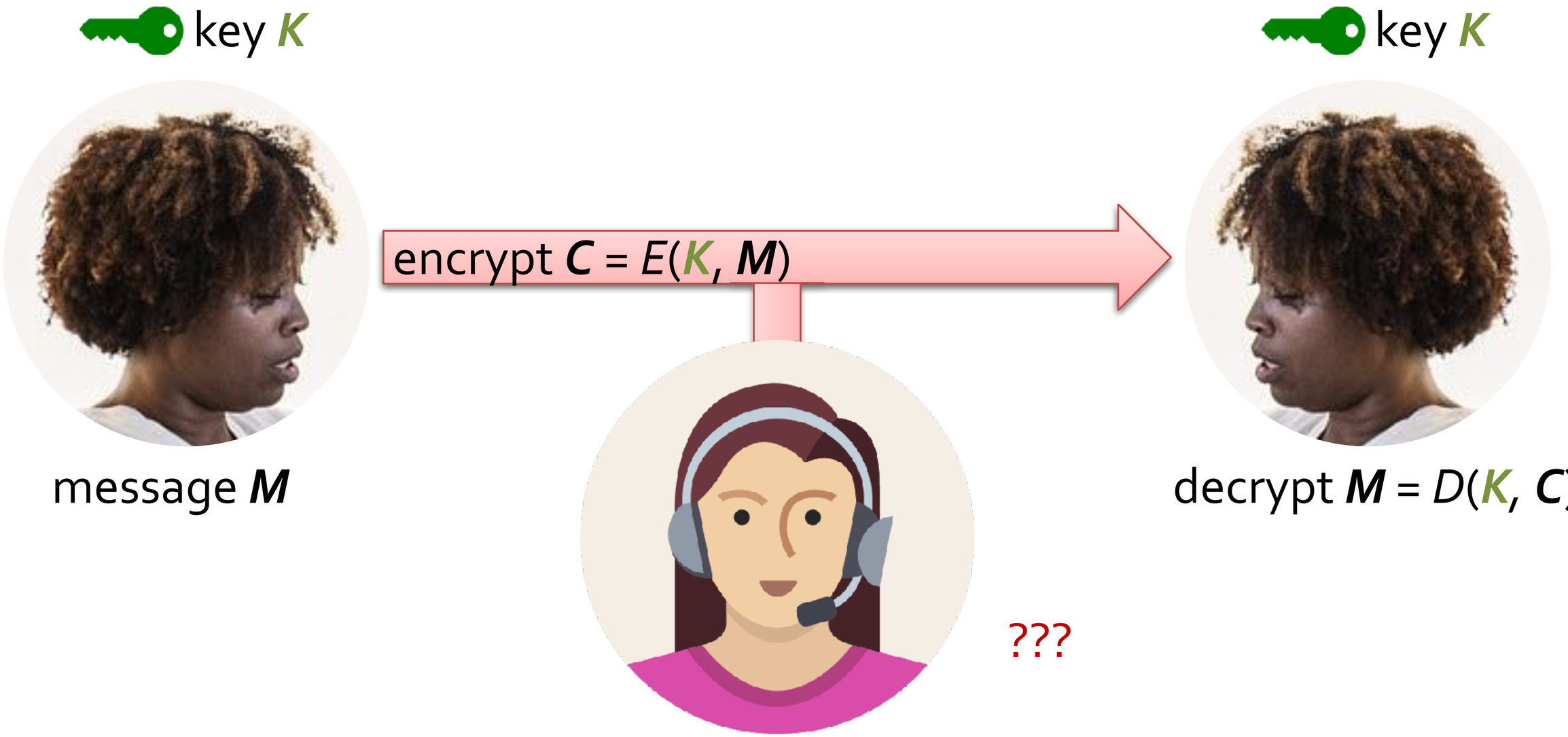
Posted on piazza.com

- Week 2 reading
- Lab 2 (due on Monday 2/4 at 11pm)
- Lab 1 answers

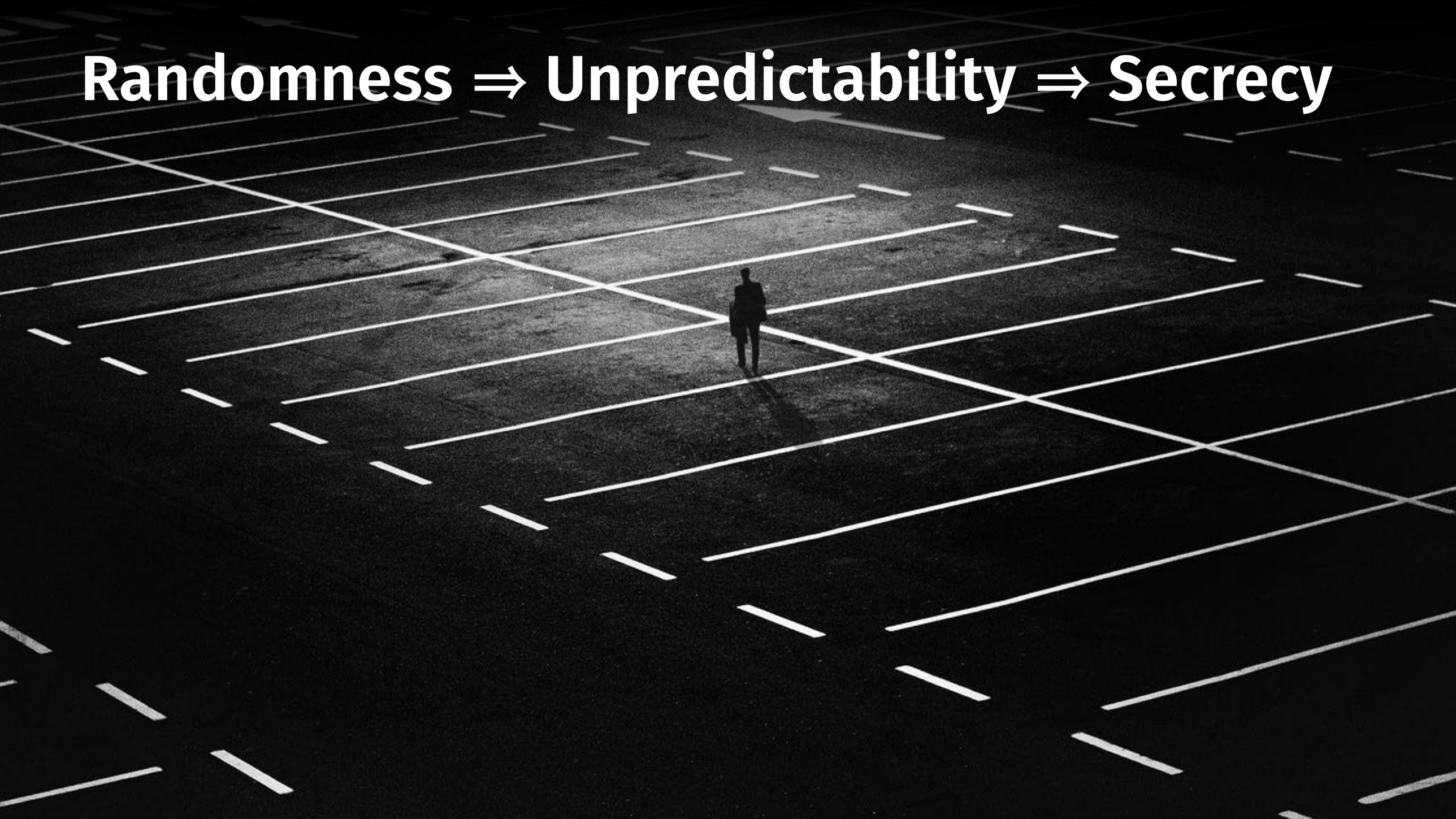
RUSSIANS ENGINEER A BRILLIANT SLOT MACHINE CHEAT—AND CASINOS HAVE NO FIX



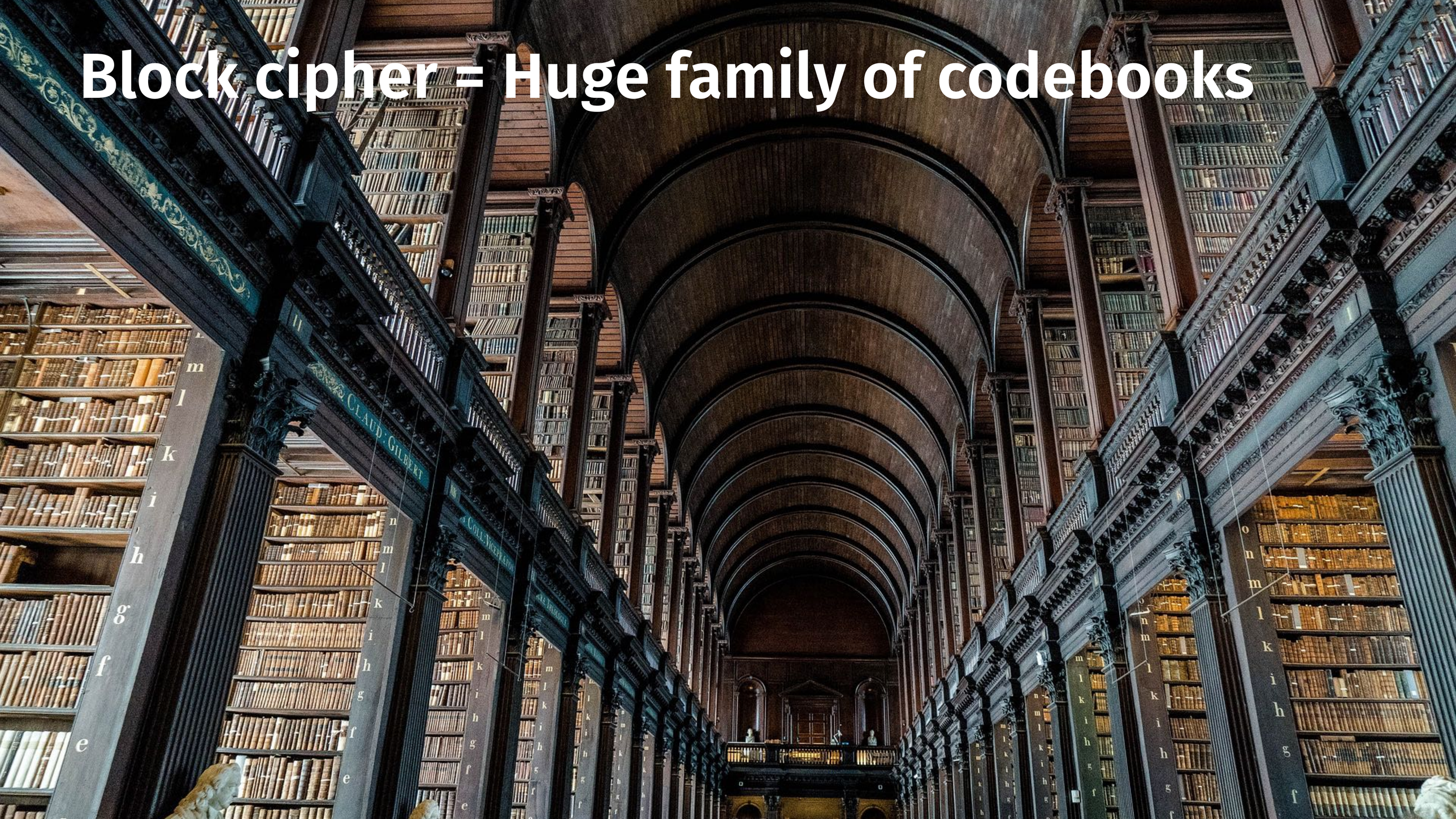
Part 1: Protecting data at rest



Randomness ⇒ Unpredictability ⇒ Secrecy



Block cipher = Huge family of codebooks



Structure of 1 codebook

- Codebook is a *random-looking* function $B : \{0,1\}^{in} \rightarrow \{0,1\}^{out}$
- So far we have considered input length == output length
  - As a result, can insist that B is invertible
  - Will explore other options starting on Thursday

X	Y
aba	nrq
abs	mbk
ace	ybd
act	wxv
add	jen
ado	hhg
aft	uxv
age	zmx
ago	dgs
aha	ase
aid	ktf
⋮	⋮
zip	cyu
zoo	dux

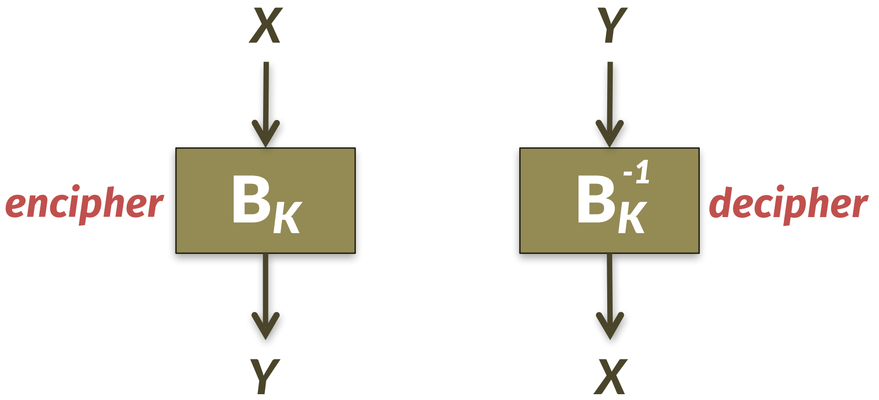
Today’s plan

1. Formally state the guarantees we want from a block cipher
2. Design a block cipher from a single, public, “perfect” codebook
3. Instantiate a “good enough” approximation of a perfect codebook

Block cipher

- Family of permutations, indexed by a secret key
- Design goals

1. **Simple** - built from native CPU operations like XOR, cyclic shifts, and table lookups
2. **Makes no sense** - unpredictable
3. **Simple to see why it makes no sense** - we have simple, convincing arguments that the cipher is unpredictable (remember Schneier's law!)



Security game

- Let Π = truly random + secret permutation
- B_K is *strongly pseudorandom* if every resource-bounded adversary can only distinguish the real cipher Π from with very small probability ε

Block cipher details

Parameters

- μ = block length = log(length of a book)
- λ = key length = log(# books in library)

Algorithms

- **KeyGen:** Randomly choose a key *K* of length λ, often uniformly from $\{0,1\}^\lambda$
- **Encipher:** Given input $X \in \{0,1\}^\mu$, outputs $B_K(X) \rightarrow Y$, where $Y \in \{0,1\}^\mu$ too
- **Decipher:** Given $Y \in \{0,1\}^\mu$, outputs $B_K^{-1}(Y) \rightarrow X$, where $X \in \{0,1\}^\mu$ too

Assume for now that there is a "good" method to generate a random key. Will explore later in the course:

- How to generate random numbers
- Crypto designs that withstand not-so-great sources of randomness

Block cipher details

Parameters

- μ = block length = log(length of a book)
- λ = key length = log(# books in library)

Algorithms

- **KeyGen:** Randomly choose a key *K* of length λ, often uniformly from $\{0,1\}^\lambda$
- **Encipher:** Given input $X \in \{0,1\}^\mu$, outputs $B_K(X) \to Y$, where $Y \in \{0,1\}^\mu$ too
- **Decipher:** Given $Y \in \{0,1\}^\mu$, outputs $B_K^{-1}(Y) \to X$, where $X \in \{0,1\}^\mu$ too

Guarantees

- **Performance:** All 3 algorithms are efficiently computable
- **Correctness:** For every $K \in \{0,1\}^\lambda$ and $X \in \{0,1\}^\mu$, it holds that $B_K^{-1}(B_K(X)) = X$
- **(q, t, ε)-strong pseudorandomness:** For every adversary *A* that makes ≤ q queries and executes in time ≤ t,

$$|\Pr[A^{B_K, B_K^{-1}} = 1] - \Pr[A^{\Pi, \Pi^{-1}} = 1]| < \varepsilon$$

over the choices of key $K \in \{0,1\}^\lambda$ and permutation $\Pi : \{0,1\}^\mu \to \{0,1\}^\mu$

Block cipher details

Guarantees

- **Performance:** All 3 algorithms are efficiently computable
- **Correctness:** For every K $\in \{0,1\}^\lambda$ and X $\in \{0,1\}^\mu$, it holds that $B_K^{-1}(B_K(X)) = X$
- **(q, t, ε)-strong pseudorandomness:** For every adversary *A* that makes ≤ q queries and executes in time ≤ t,

  $$|\Pr[A^{B_K, B_K^{-1}} = 1] - \Pr[A^{\Pi, \Pi^{-1}} = 1]| < \varepsilon$$

  over the choices of key K $\in \{0,1\}^\lambda$ and permutation $\Pi : \{0,1\}^\mu \to \{0,1\}^\mu$

Notational shorthand for this claim

$$A^{B_K, B_K^{-1}} \approx_{(q,t,\varepsilon)} A^{\Pi, \Pi^{-1}}$$

Pseudorandomness → Claude Shannon's goals

- **Confusion:** uncertainty *within* each entry of a codebook
- **Diffusion:** uncertainty *between* entries of a codebook

Source: Claude Shannon's (many) papers
- Communication Theory of Secrecy Systems
- A Mathematical Theory of Communication
- A Mathematical Theory of Cryptography

Confusion: Uncertainty within a row

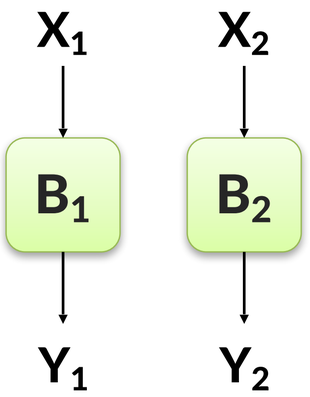
- Uncertainty of $K \rightarrow$ cannot predict Y given X or vice-versa
- Tough even to **correlate** X and Y
- Ideal: Prob[correlation] so small that attacker is better off with a brute force attack

```
> from Cryptodome.Cipher import AES
> key = 16 * '\x00'
> B = AES.new(key, AES.MODE_ECB)
> B.encrypt('abcdefghijklmnop')
'c3af71addfe4fcac6941286a76ddedc2'
```

Diffusion: Uncertainty between rows

- 1 bit ΔX → huge ΔY
- *Partial knowledge* of input doesn't help to learn output
- Ideal goal is **avalanching**: each bit of output depends on all input bits
- Note: confusion ↛ diffusion
  - Combine 2 functions
  - Can be confusing but not diffusing

```
> from Cryptodome.Cipher import AES
> key = 16 * '\x00'
> B = AES.new(key, AES.MODE_ECB)
> B.encrypt('abcdefghijklmnop')
'c3af71addfe4fcac6941286a76ddedc2'
> B.encrypt('abcdefghijklmnoq')
'b5c180bcf80baae8ac0de2673370450c'
```

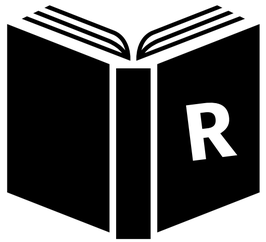


Today's plan

1. Formally state the guarantees we want from a block cipher
2. Design a block cipher from a single, public, "perfect" codebook
3. Instantiate a "good enough" approximation of a perfect codebook

A crypto “Manhattan project”

- Imagine society spends an enormous effort to make a single codebook R and its inverse (so Alice can decipher her original message later)
- Can Alice use this codebook to protect her messages from Eve?
- Intuitively: no!
  - Eve can use the codebook too
  - Codebook is too large for Alice to carry around
  - Codebook’s input + output lengths may not suffice to encode Alice’s message
- Actually: yes! We can address all of these concerns



+

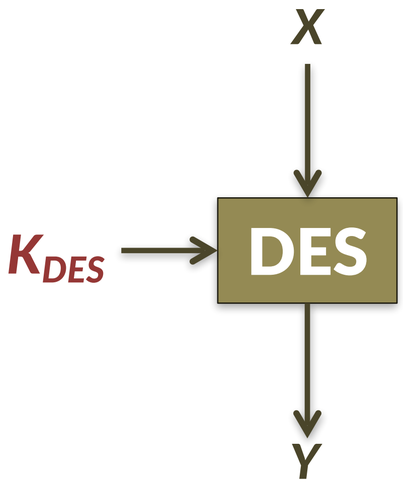
⇒

⊕	0	1
0	0	1
1	1	0

Flashback: The Data Encryption Standard (DES)

History

- 1972: NIST* seeks standard mechanism to protect US federal gov “sensitive but unclassified” info
- 1st request: Rejected all submissions
- 2nd request: accepted the *Lucifer* cipher by Horst Feistel & others at IBM



Lengths of DES components

- Block length: 8 bytes
- Key length: 7 bytes

NSA changes: ▲ cryptanalytic strength

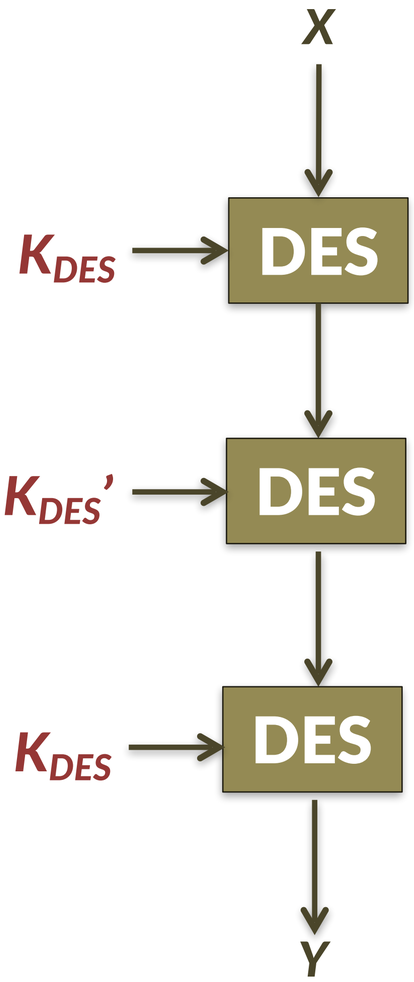
▼ key length 8 → 7 bytes

Crypto war 1: key length

Question: how to increase key length without making a new standard?

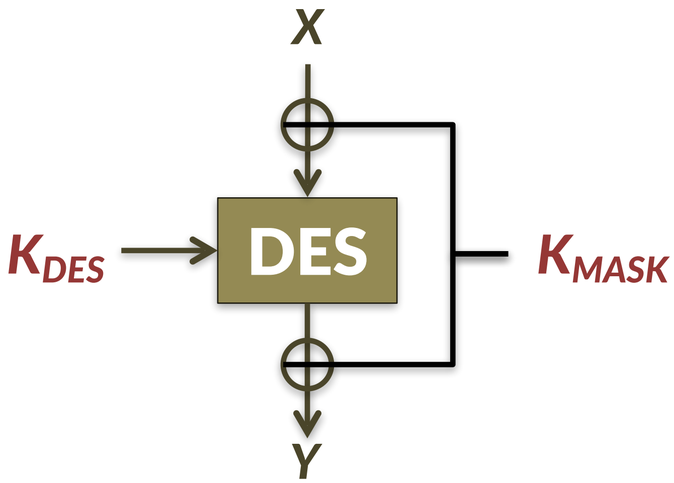
Solutions

- 2DES: Run DES twice
- 3DES: Run DES three times



Crypto war 1: key length

Question: how to increase key length without making a new standard?



Solutions

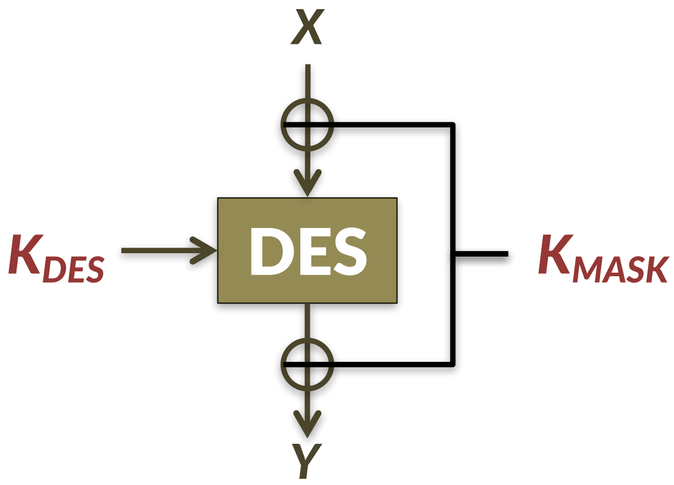
- 2DES: Run DES twice
- 3DES: Run DES three times
- DESX (Rivest 84): mask input + output
  Resulting key = 15 bytes

Benefits of DESX

- Fast: 1 block cipher call, quick re-keying
- Available: RSA Security had in their BSAFE software
  since the late 1980s (before the rise of open source crypto software)

Random permutation → block cipher

Question: What is the simplest possible construction of a block cipher that has a formal proof of security?



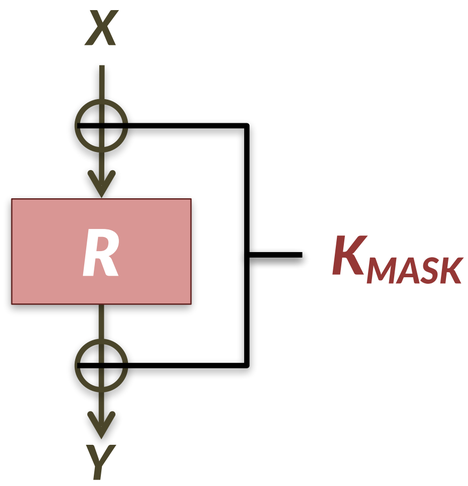
Random permutation → block cipher

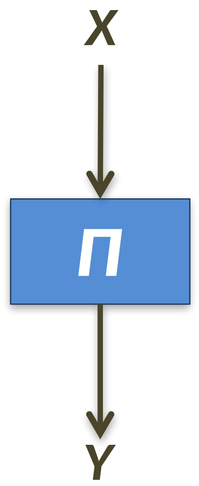
Question: What is the simplest possible construction of a block cipher that has a formal proof of security?

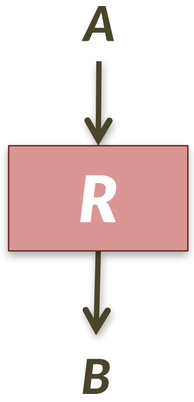
Even & Mansour 91: Rivest's idea applies to any "public, random-looking permutation"

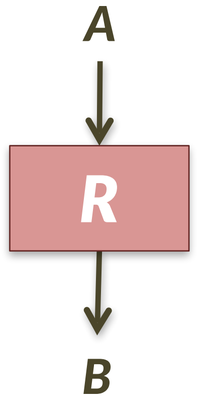
Theorems

1. Resulting block cipher is *strongly pseudorandom* …even if *R* is public
2. Construction is *minimal* in the sense that nothing can be removed







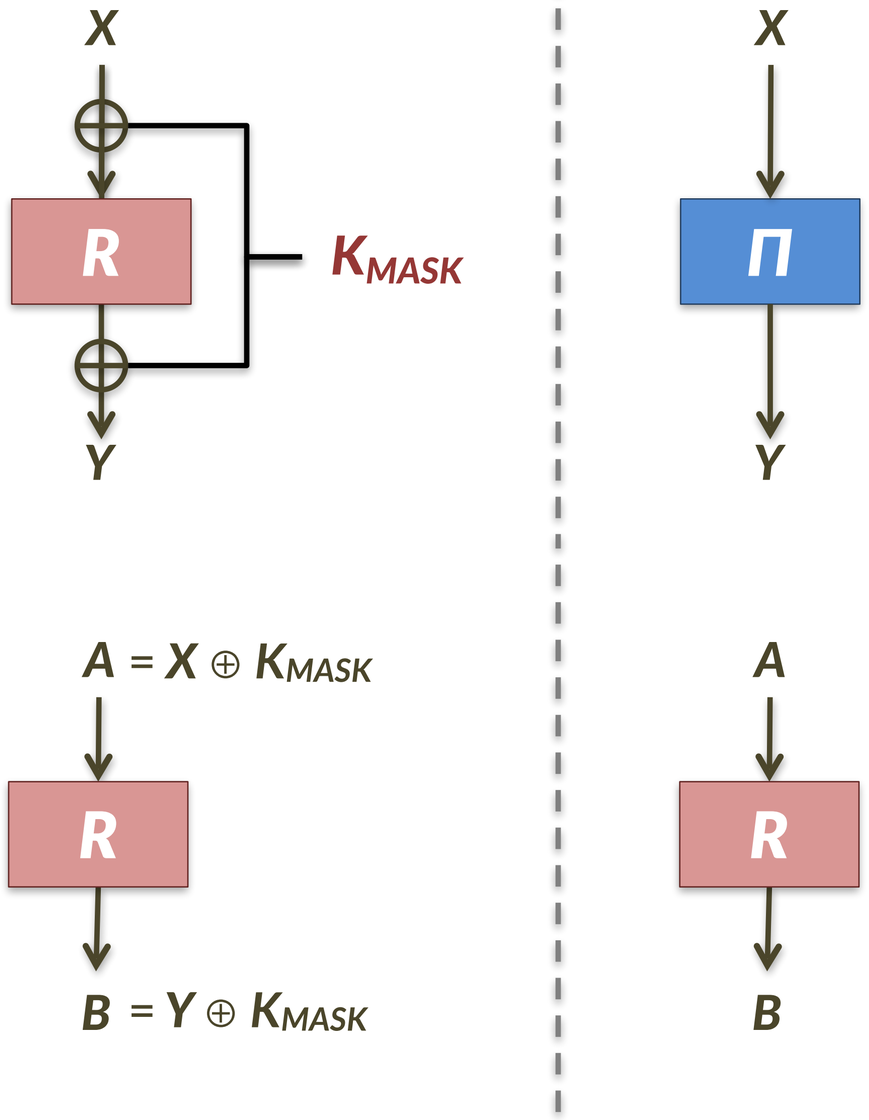


Proof of pseudorandomness

Thm. Construction is *strongly pseudorandom*.

Proof.

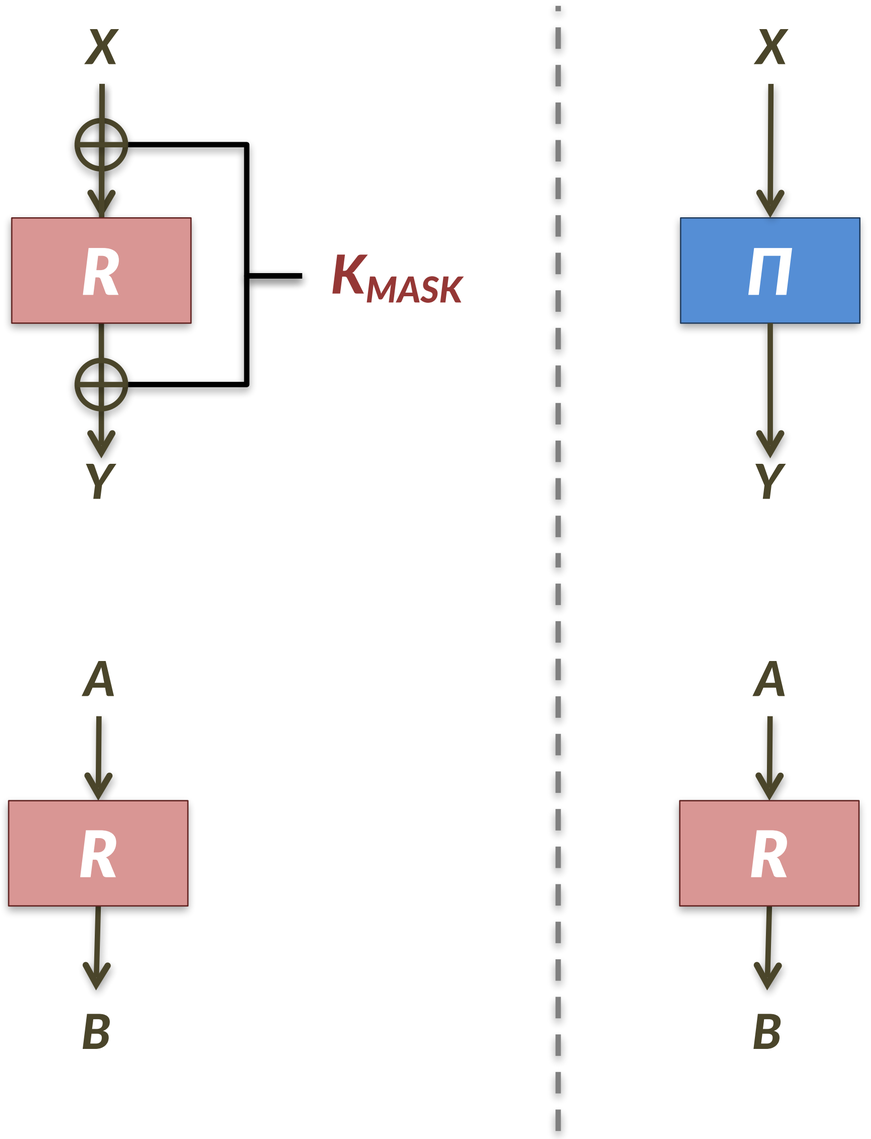
- Before the adversary  makes any queries, all choices of K_{MASK} are equally likely
- To reduce the set of possible K_{MASK}, adversary must find collisions between Π and R, which are unlikely
- For each (X, Y) and (A, B) pair, label the keys $(X \oplus A)$ and $(Y \oplus B)$ as *bad*; q queries yield only $2q^2$ bad keys
- All good keys are equally likely: they all fail to cause collisions anywhere
- Same argument applies to the inverse direction



Proof of minimality

Thm. Construction is *minimal* in the sense that nothing can be removed.

Proof.

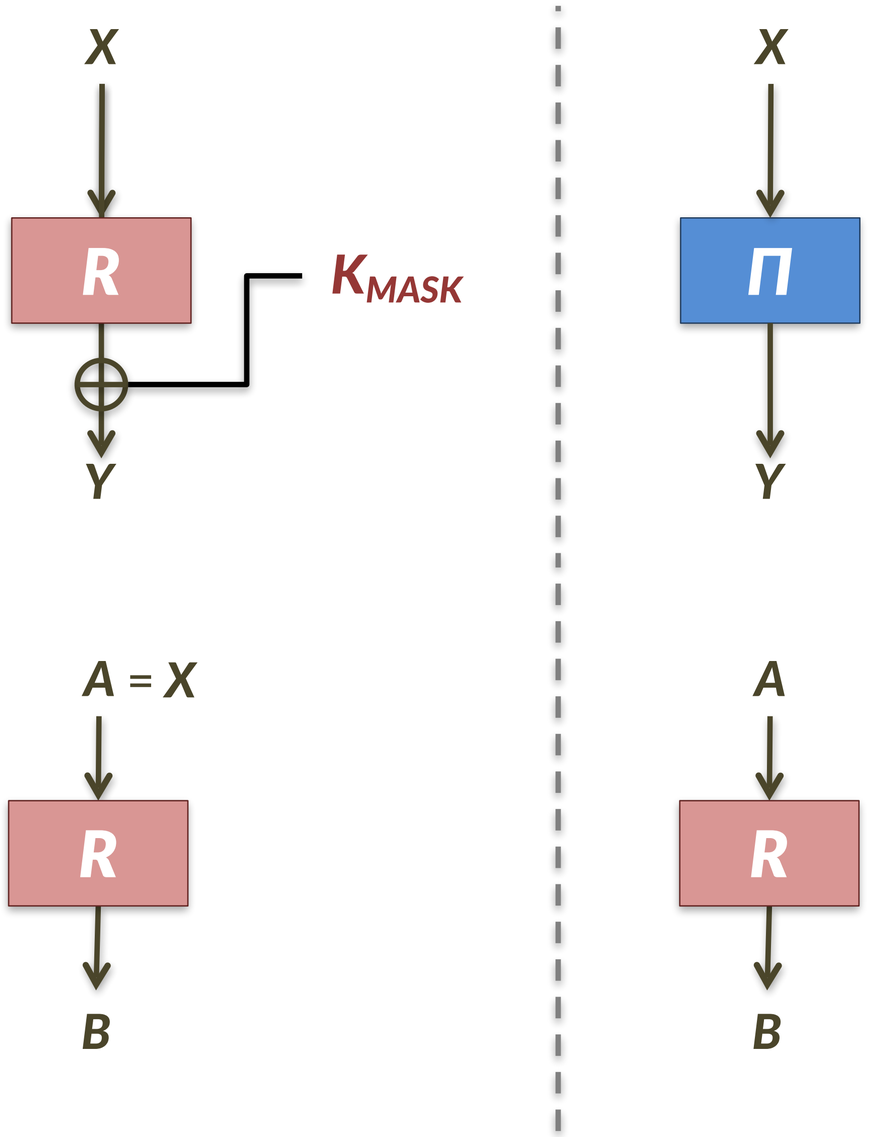


Proof of minimality

Thm. Construction is *minimal* in the sense that nothing can be removed.

Proof.

- Removing either ⊕ allows adversary to learn the key with one X/Y pair and one query to ***R***.

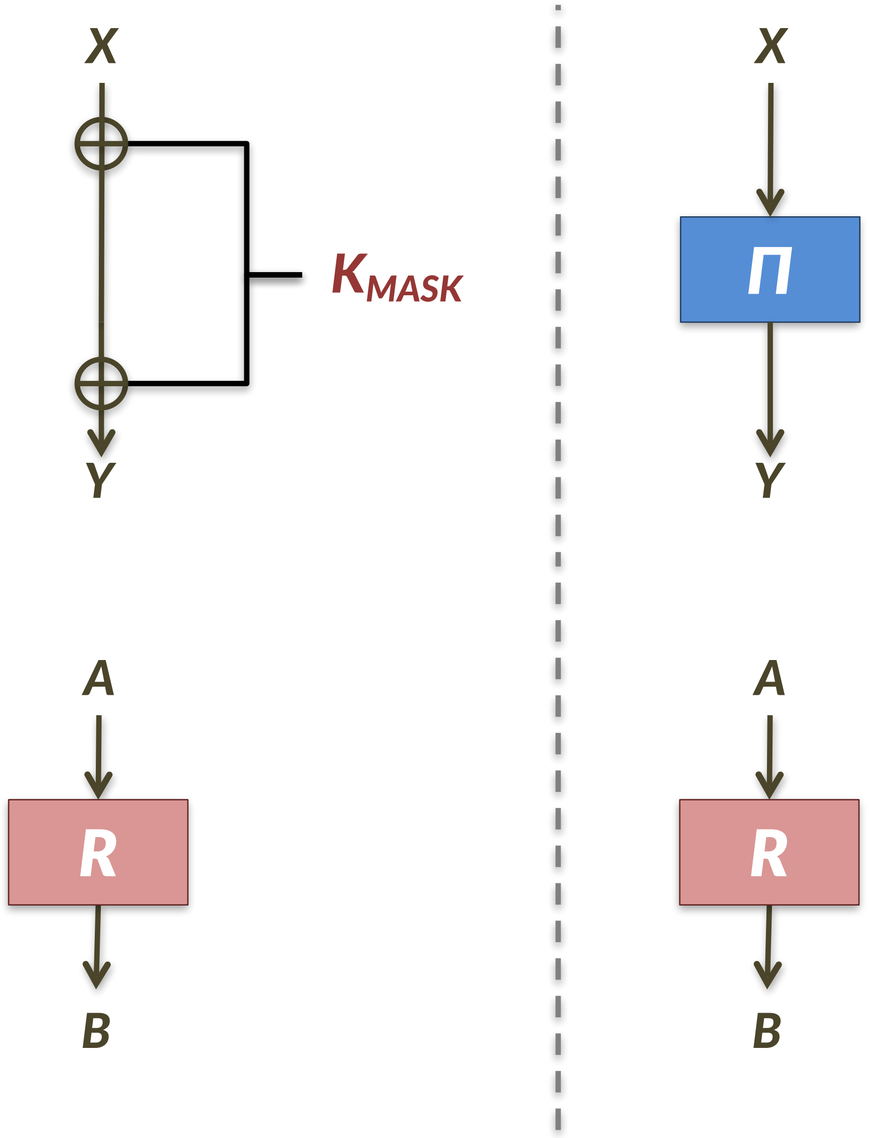


Proof of minimality

Thm. Construction is *minimal* in the sense that nothing can be removed.

Proof.

- Removing either ⊕ allows adversary to learn the key with one X/Y pair and one query to ***R***.
- Removing ***R*** leaves the identity function.



+

⇒

⊕	0	1
0	0	1
1	1	0

Today's plan

1. Formally state the guarantees we want from a block cipher
2. Design a block cipher from a single, public, "perfect" codebook
3. Instantiate a "good enough" approximation of a perfect codebook

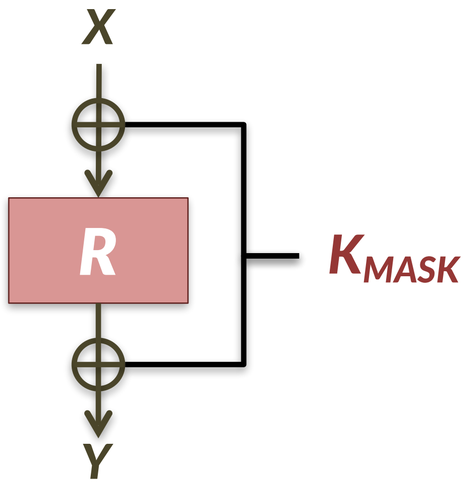
"If an adversary *A* has not **explicitly** queried a [perfect codebook] ***R*** on some point *X*, then the value of ***R***(*X*) is **completely random**... at least as far as *A* is concerned."

–*Jon Katz and Yehuda Lindell*, Introduction to Modern Cryptography

Great, let's build a block cipher!

Problems?

1. Hmm, how do we go about building a single random permutation *R*?
2. Isn't the truth table for *R* huge?



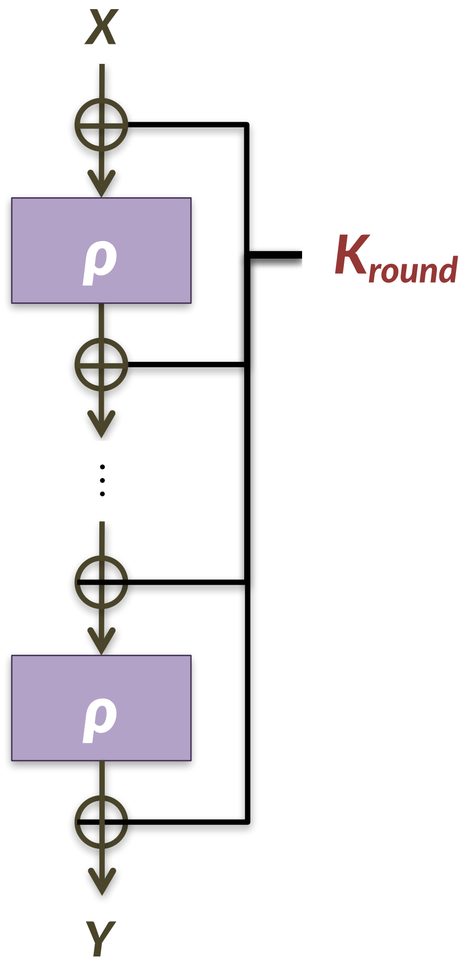
Great, let's build a block cipher!

Problems?

1. Hmm, how do we go about building a single random permutation *R*?
2. Isn't the truth table for *R* huge?

Solution to 1: multiple rounds

- Let's make life easier: what if we make *ρ* that is *somewhat* random?
- Then we can use the 3DES trick



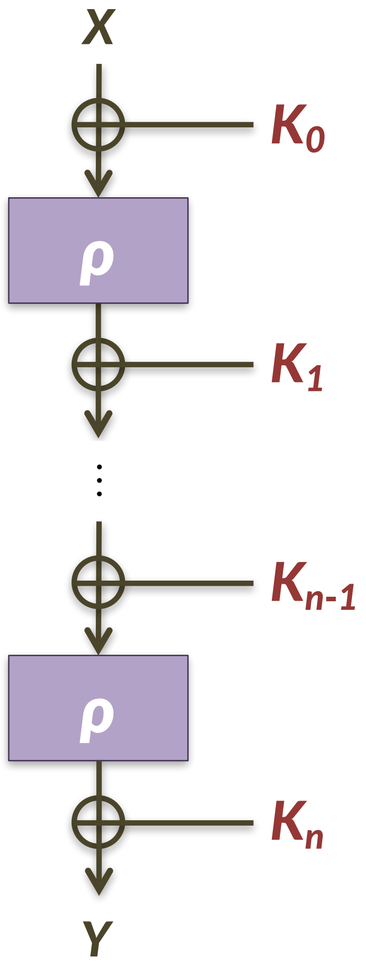
Great, let's build a block cipher!

Problems?

1. Hmm, how do we go about building a single random permutation *R*?
2. Isn't the truth table for *R* huge?

Solution to 1: multiple rounds

- Let's make life easier: what if we make *ρ* that is *somewhat* random?
- Then we can use the 3DES trick
- (Nitpicky detail: each round needs a different key to thwart *slide attacks*)



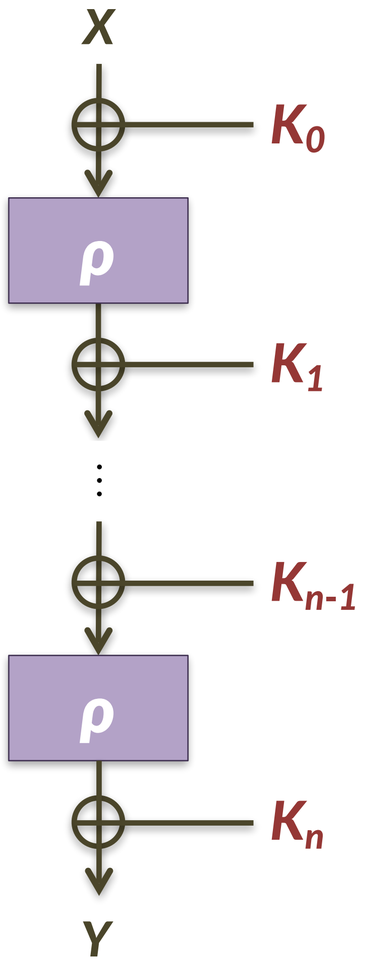
Great, let's build a block cipher!

Problems?

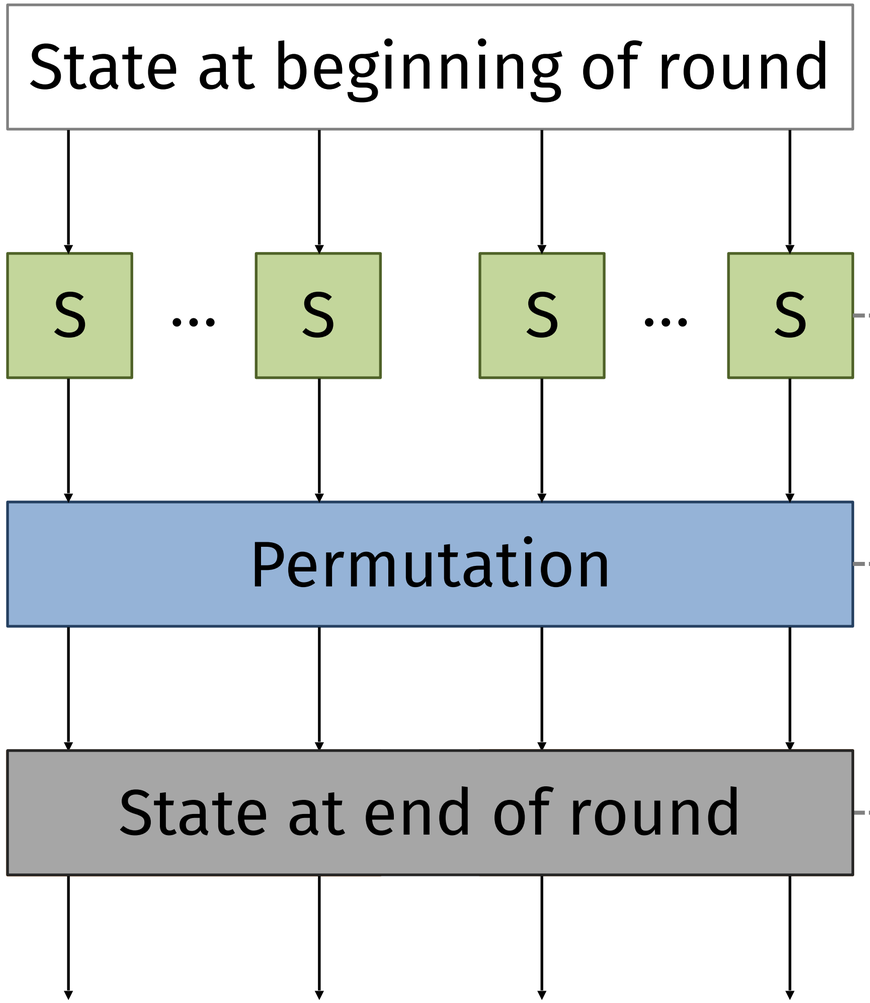
1. Hmm, how do we go about building a single random permutation *R*?
2. Isn't the truth table for *R* huge?

Solution to 2: simple round function ρ

- Linear functions are very simple!
- Err, perhaps too simple; we could then solve for the key
- We need non-linearity somewhere
- But let's keep its truth table small



Designing ρ: The substitution-permutation model



1 Substitution box (S-box)

Provides *confusion*, but at a cost

2 Linear permutation

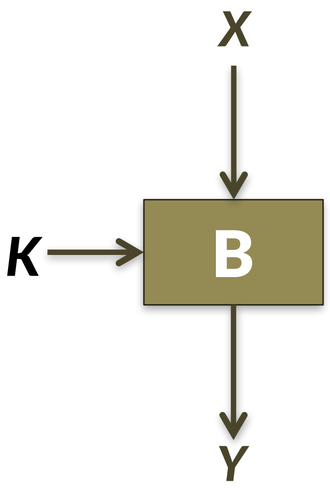
Provides global *diffusion*

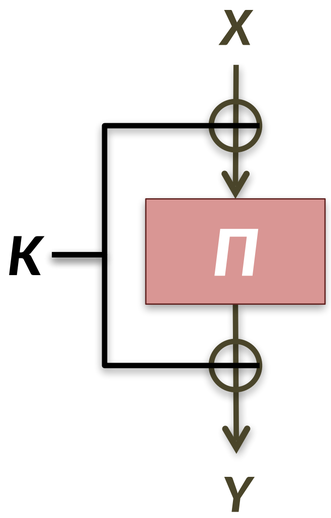
3 AES-specific middle step

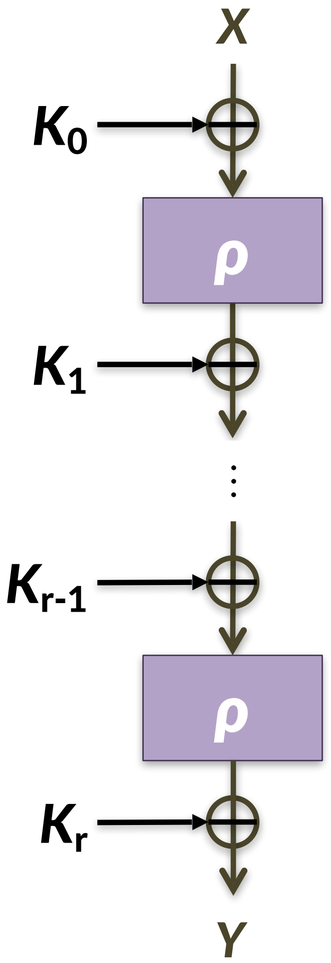
A linear operation that (somehow) provides both *diffusion* and *confusion*

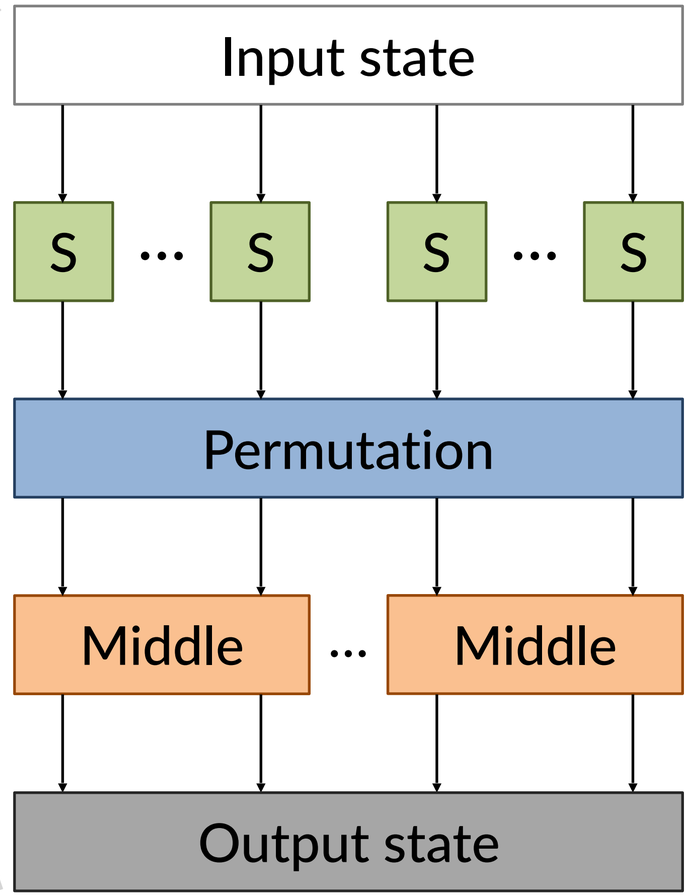
Block cipher design

Block cipher ← Key alternation ← Iterated rounds ← Substitution-Permutation









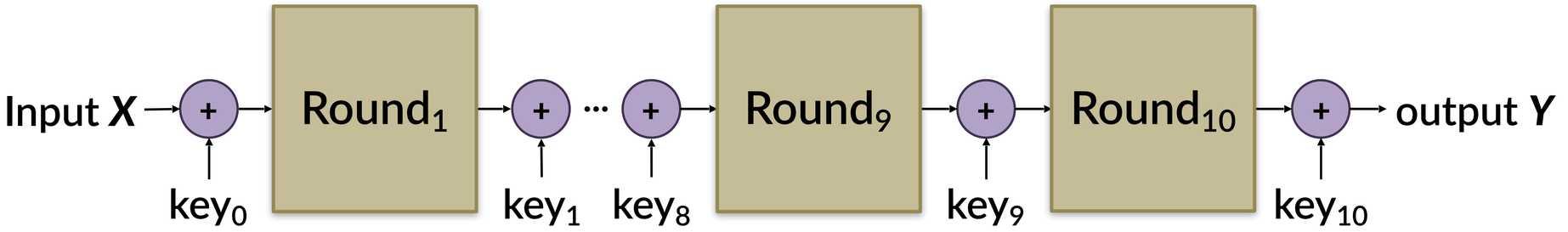
Advanced Encryption Standard (AES) Competition

- NIST competition held 1997–2000
- Required good performance for
  - 8-bit smartcard
  - 32-bit software
  - Dedicated hardware
- Well-run competition
  - Many candidates: 15 initially, 5 finalists
  - Included 3 conferences
- Winner: Rijndael
  - Authors: Joan Daemen, Vincent Rijmen

"Algorithms will be judged on the extent to which their output is indistinguishable from a random permutation on the input block."

	Rijndael	Serpent	Twofish	MARS	RC6
General security	2	3	3	3	2
Simplicity to implement	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

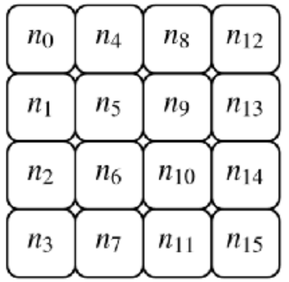
Rijndael, aka AES



Key alternating structure

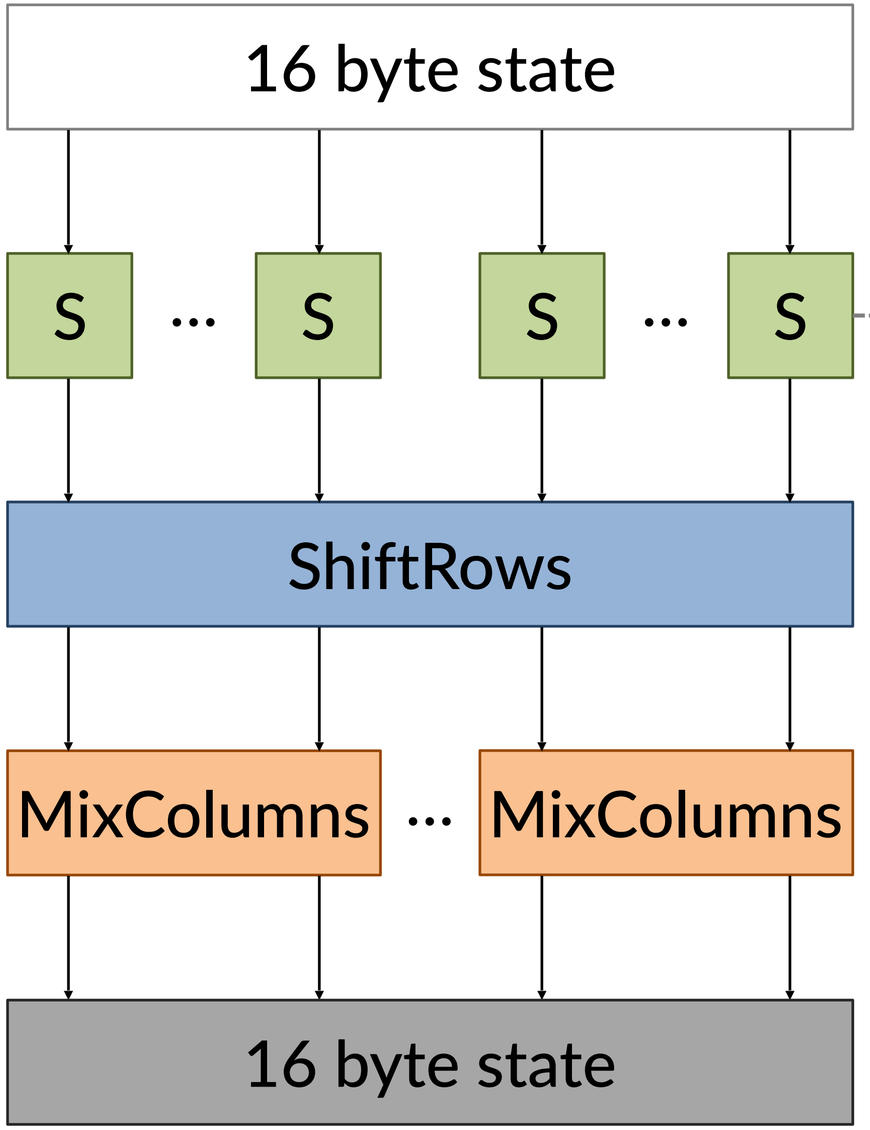
- 128, 192, or 256 bit initial key
- Expand into $r+1$ round keys, each of which is 128 bits long
- Invertible key schedule: given key_i, can compute key_{i-1} or key_{i+1}

Iterated round structure



- 16 bytes of state
- Total of 10 to 14 rounds
- 3 invertible operations per round
  - Final round is slightly different
- Only S-box is nonlinear

AES components



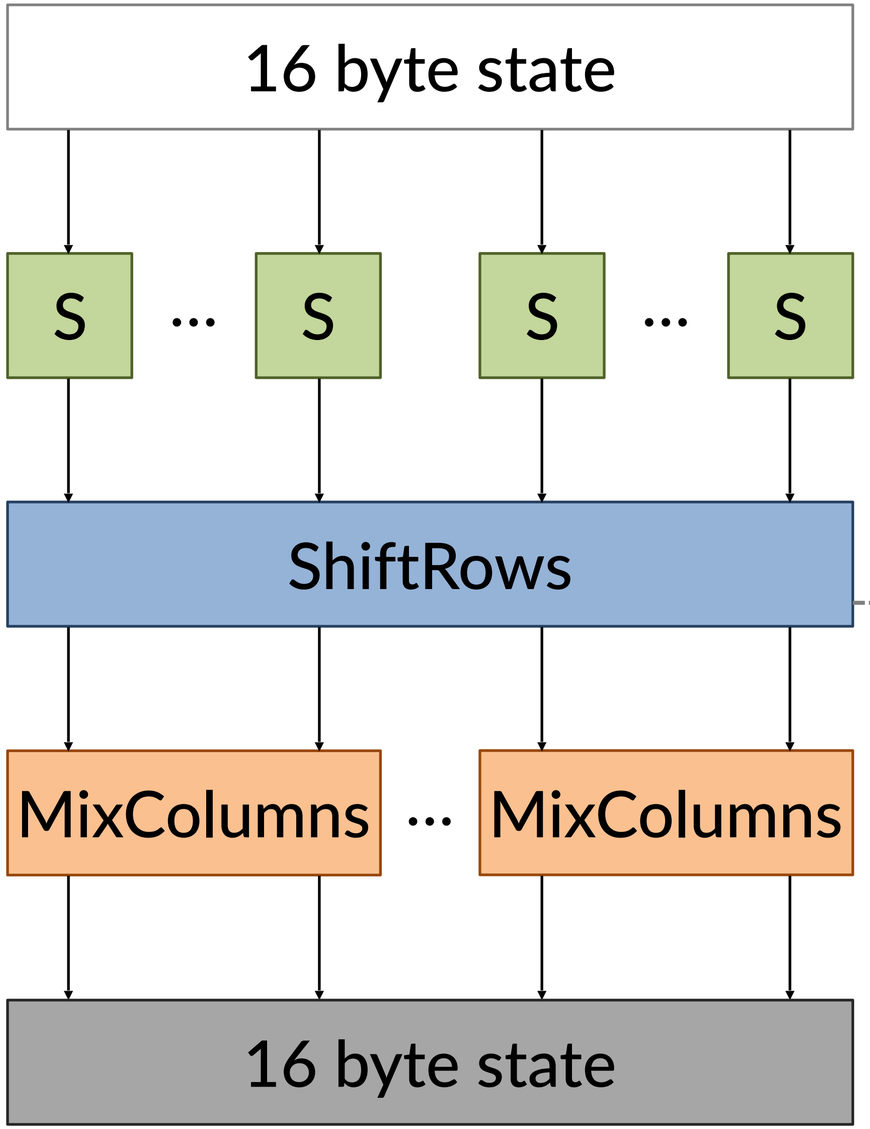
1 SubBytes

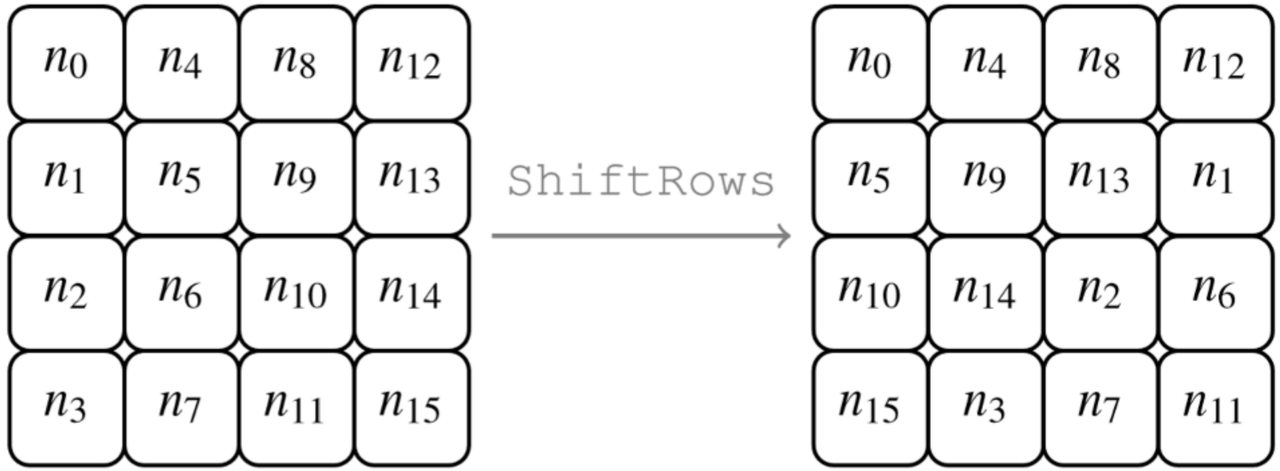
Table lookup, one byte at a time

$S[\cdot]$

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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

AES components

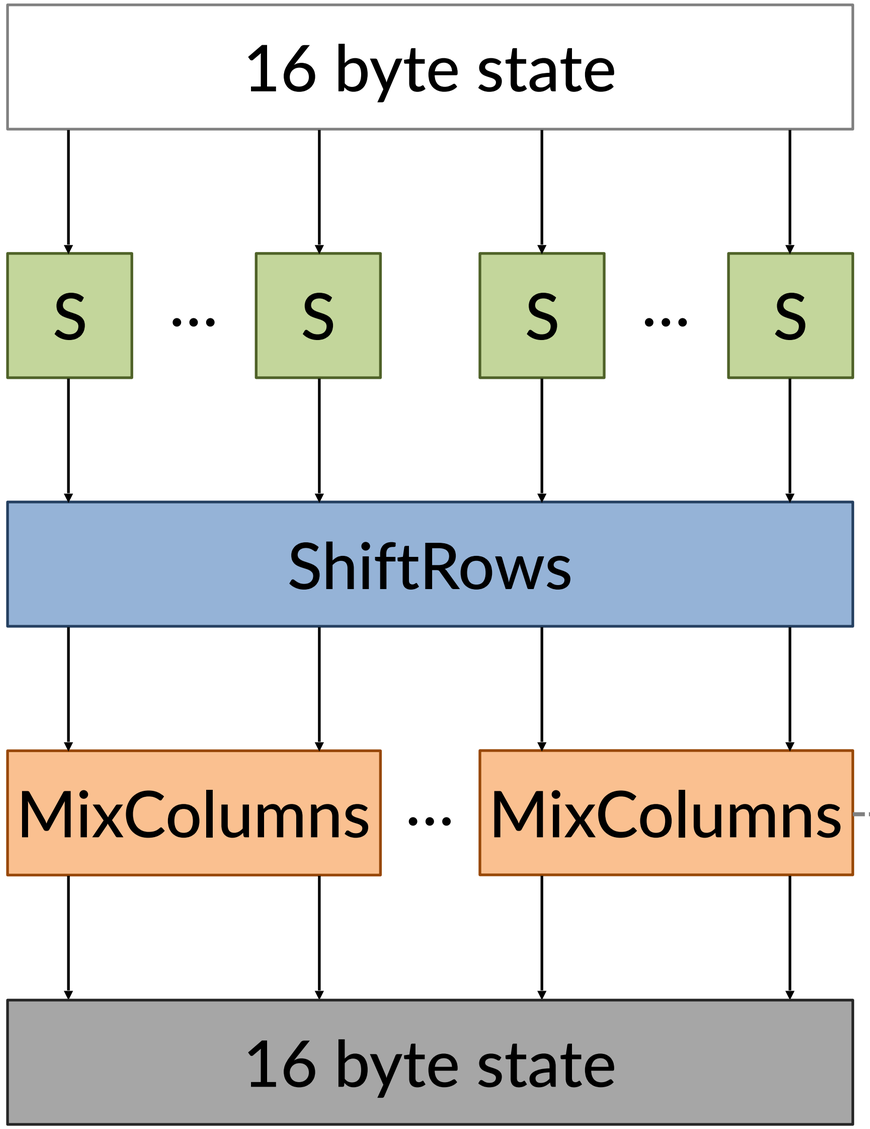




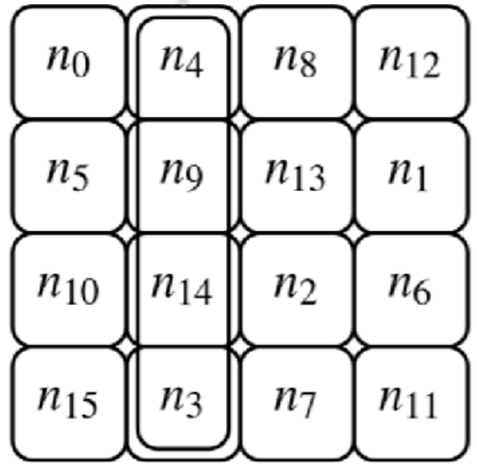
2 ShiftRows

Byte-wise transposition

AES components



$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$



3 MixColumns

Matrix multiplication in GF(256)

Speed of AES

- **Intel's AES-NI:** Intel CPUs include 6 instructions to perform AES quickly
- **Apple Secure Enclave:** "Every iOS device has a dedicated AES-256 crypto engine built into the DMA path between the flash storage and main system memory, making file encryption highly efficient."
- **AMD's Secure Memory Enc**

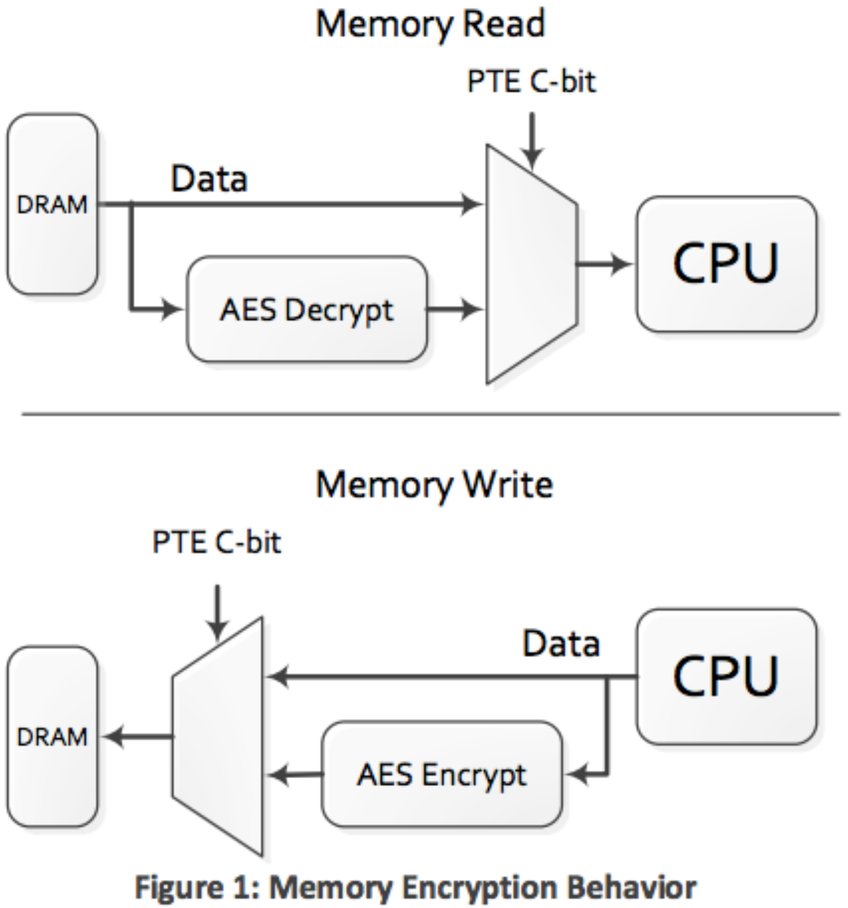


Figure 1: Memory Encryption Behavior

Sources:
- www.apple.com/business/docs/iOS_Security_Guide.pdf
- amd-dev.wpengine.netdna-cdn.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf

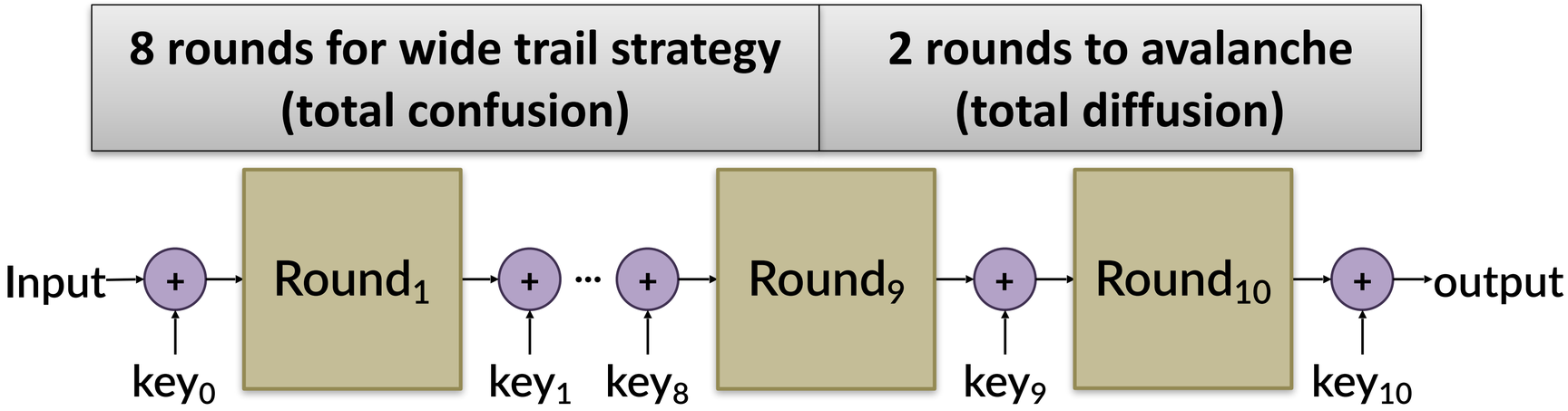
Claude Shannon's two security goals for block ciphers

Confusion

Ideal: Prob[correlation] so small that attacker is better off with a brute force attack (takes 8 rounds in AES)

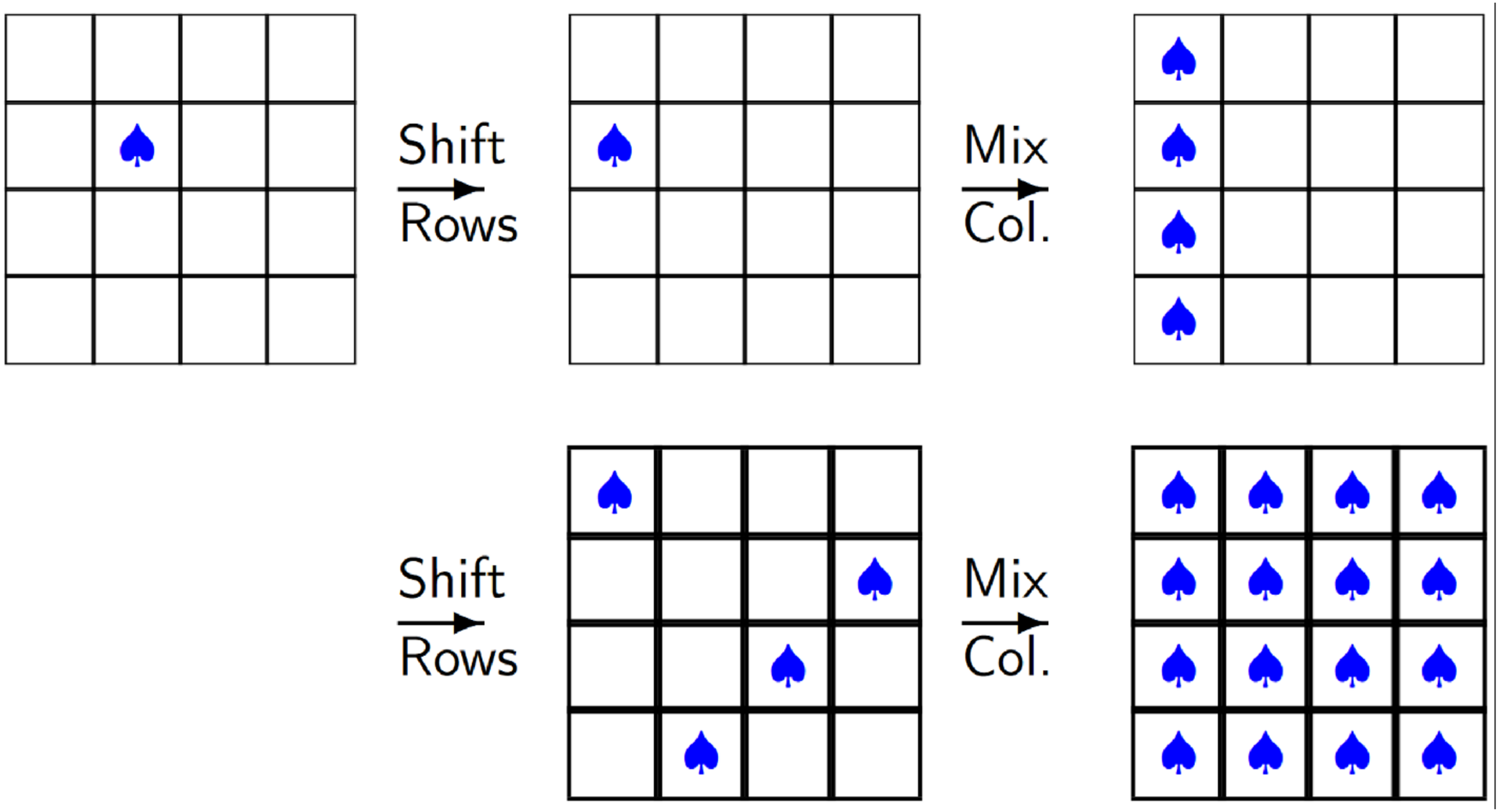
Diffusion

Ideal goal is avalanching: each bit of output depends on all input bits (takes 2 rounds in AES)



Security margin = (# rounds in a cipher) **–** (# rounds we can break)

Avalanching in AES



Source: slide 17 of summerschool-croatia.cs.ru.nl/2014/slides/Advanced Encryption Standard.pdf

Why do we think that AES is pseudorandom?