RC4

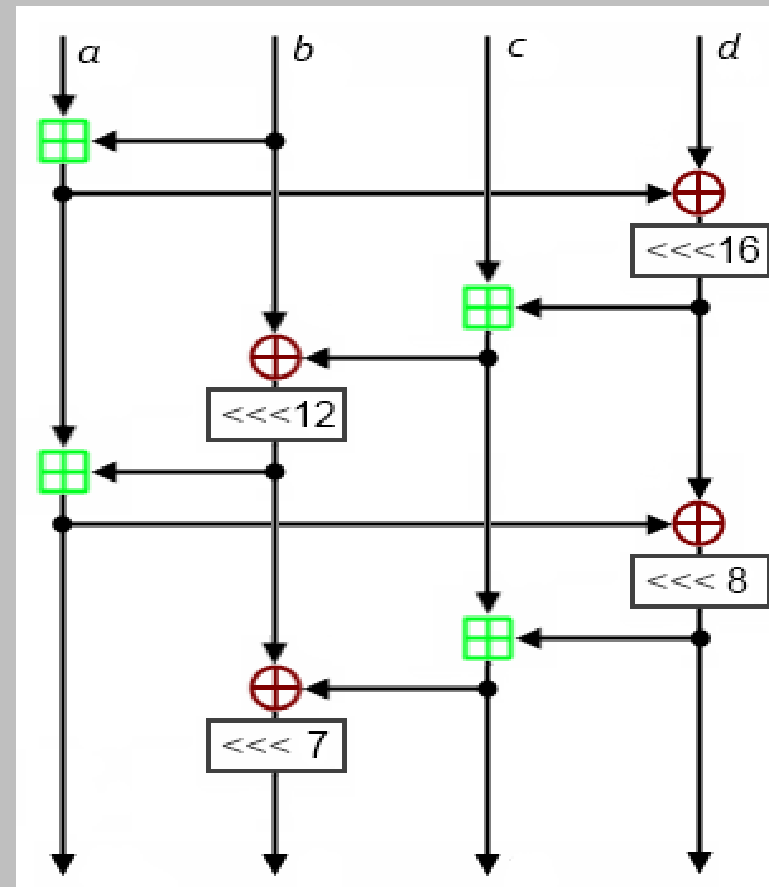
Ron Rivest
1987

RC4: $\text{Byte}^k \longrightarrow \text{Byte}^\infty$
for any $k \in [1..256]$

```
Algorithm RC4(byte string K)
byte i,j     //all arith involving these mod 256
for i ←0 to 255 do S[i] ← i

j ← 0

for i ←0 to 255 do

    j ← j + S[i] + K[i mod |K|]

    S[i] ↔ S[j]


i, j ← 0

repeat

    i ← i + 1

    j ← j + S[i]

    S[i] ↔ S[j]

    output S[(S[i] + S[j]) mod 256]
```

```
Algorithm ChaCha20(key, ctr, non)
                      8    1    3
                4
state ← con | key | ctr | non
s ← state

for i¬1 to 10 do
QR(s[0], s[4],  s[8], s[12])  // col 1
QR(s[1], s[5],  s[9], s[13])  // col 2
QR(s[2], s[6], s[10], s[14])  // col 3
QR(s[3], s[7], s[11], s[15])  // col 4
QR(s[0], s[5], s[10], s[15])  // diag 1
QR(s[1], s[6], s[11], s[12])  // diag 2
QR(s[2], s[7],  s[8], s[13])  // diag 3
QR(s[3], s[4],  s[9], s[14])  // diag 4
od

state += s
return state
```

ChaCha20: $\text{BYTE}^{32} \times \text{BYTE}^{16} \to \text{BYTE}^{64}$



ChaCha20

Dan Bernstein

2008

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

con0	con1	con2	con3
key0	key1	key2	key3
key4	key5	key6	key7
ctr	non0	non1	non2

```
Algorithm QR(a,b,c,d)
a += b; d ^= a; d <<<= 16;
c += d; b ^= c; b <<<= 12;
a += b; d ^= a; d <<<= 8;
c += d; b ^= c; b <<<= 7;
```

ChaCha20

Nice design

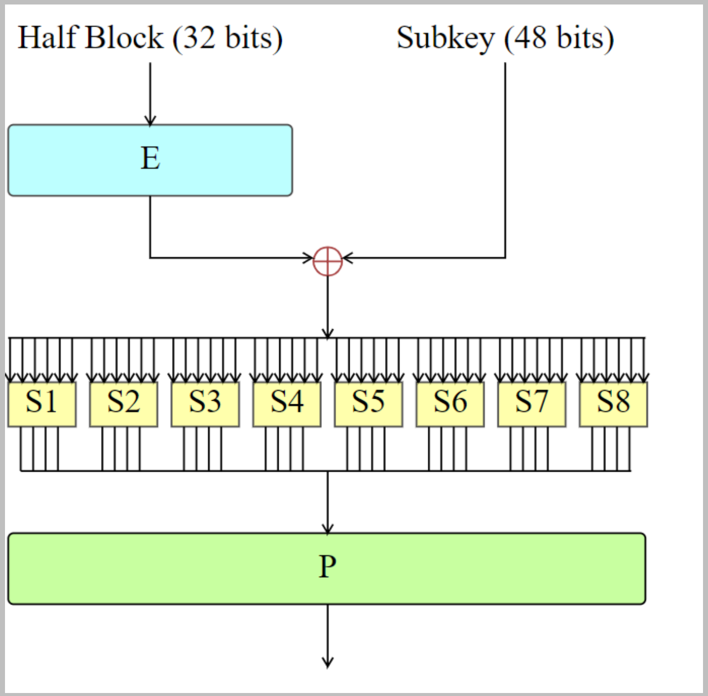
1. Good choice of signature – PRF with 32, 16, 64 byte key, input, output
2. Security has held up very well – no remotely damaging attacks
3. Very fast in SW, with no special HW instructions (eg., 2.3 cpb Sandy Bridge)
4. Spare use of operations – “ARX” (add-rotate-xor are only ops used)
5. Constant time – no tables
6. Open design, no intelligence-agency involvement

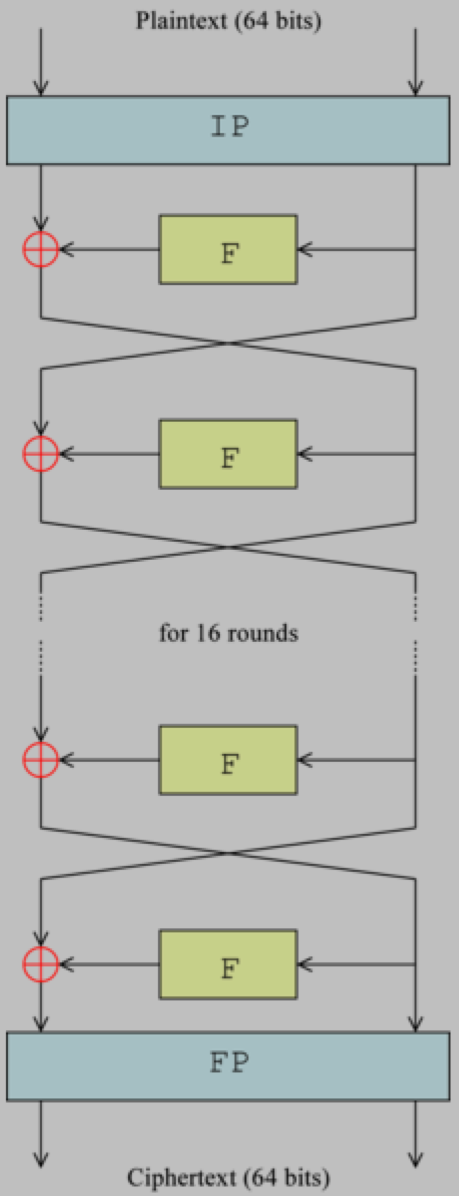
7. No key-setup, no subkeys

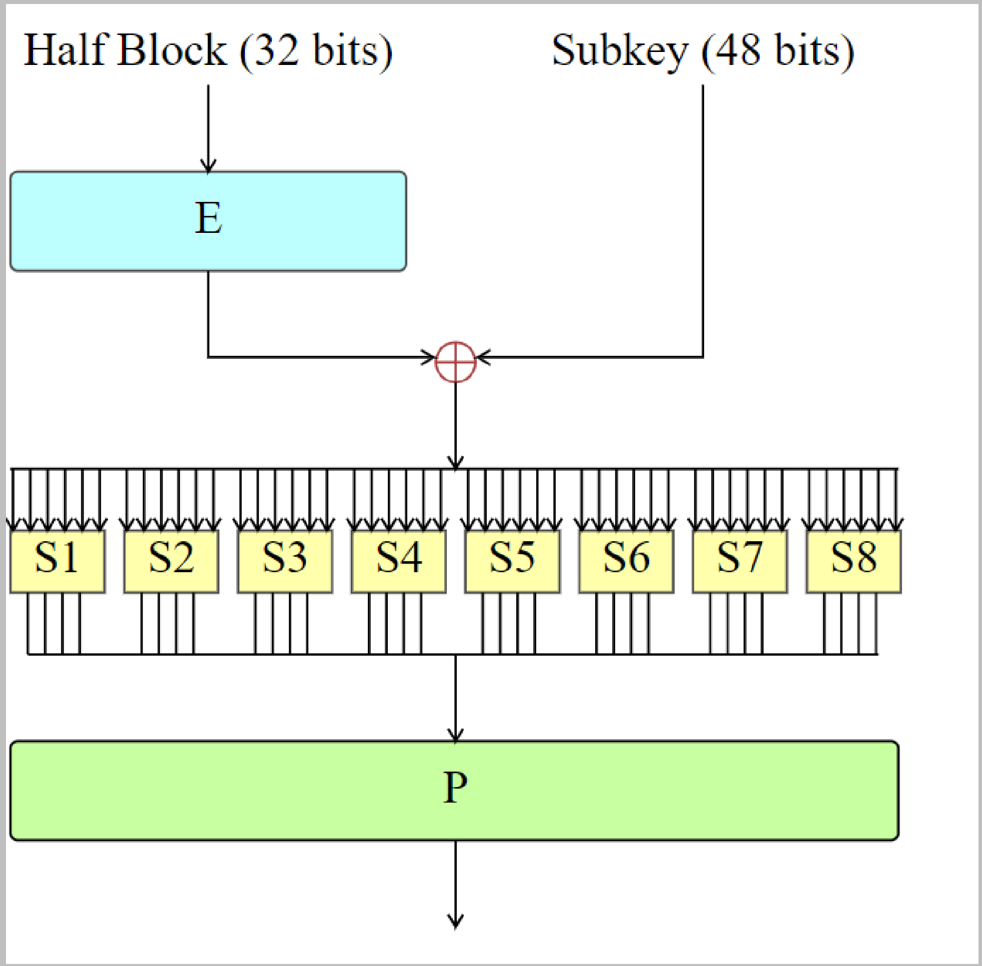
DES

IBM/NSA
1975

DES: $\{0,1\}^{56} \times \{0,1\}^{64} \rightarrow \{0,1\}^{64}$






Half Block (32 bits)
Subkey (48 bits)
E
S1
S2
S3
S4
S5
S6
S7
S8
P

Definition of DES S-Boxes

TABLE 2.6 Definition of DES S-Boxes

	Column Number																
Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Box
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S_2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S_3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S_4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S_5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S_6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S_7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S_8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

DES

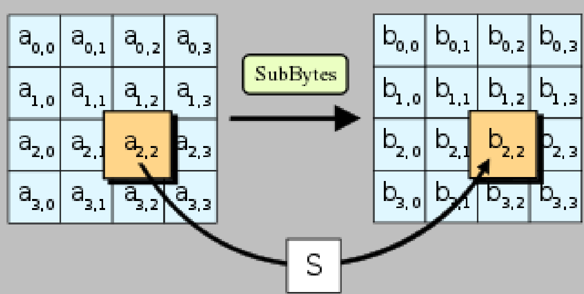
- Historically important but outmoded design
- Politics by way of mathematics

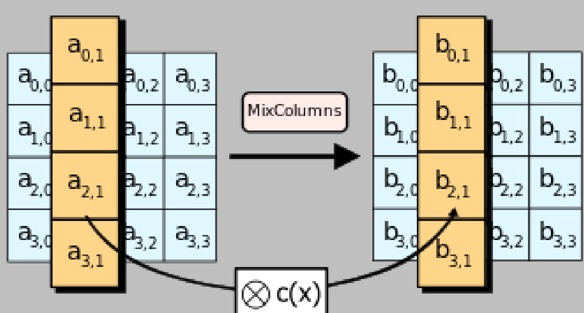
1. Has held up well for its key length
2. But key length is was chosen to permit governmental breaks
3. Other political choices, too: hardware requirement, IP/FP, standardization obstructions
4. Secret, non-competitive process. Design criteria secret (although eventually disclosed by Don Coppersmith, after everything had been figured out)
5. Led to the advances in cryptanalysis, particularly differential and linear cryptanalysis
6. Led to advances in theory, starting with Luby-Rackoff result

AES
Rijndael

Joan Daemen and Vincent Rijmen
1998/2002

DES: $\{0,1\}^{56} \times \{0,1\}^{64} \rightarrow \{0,1\}^{64}$





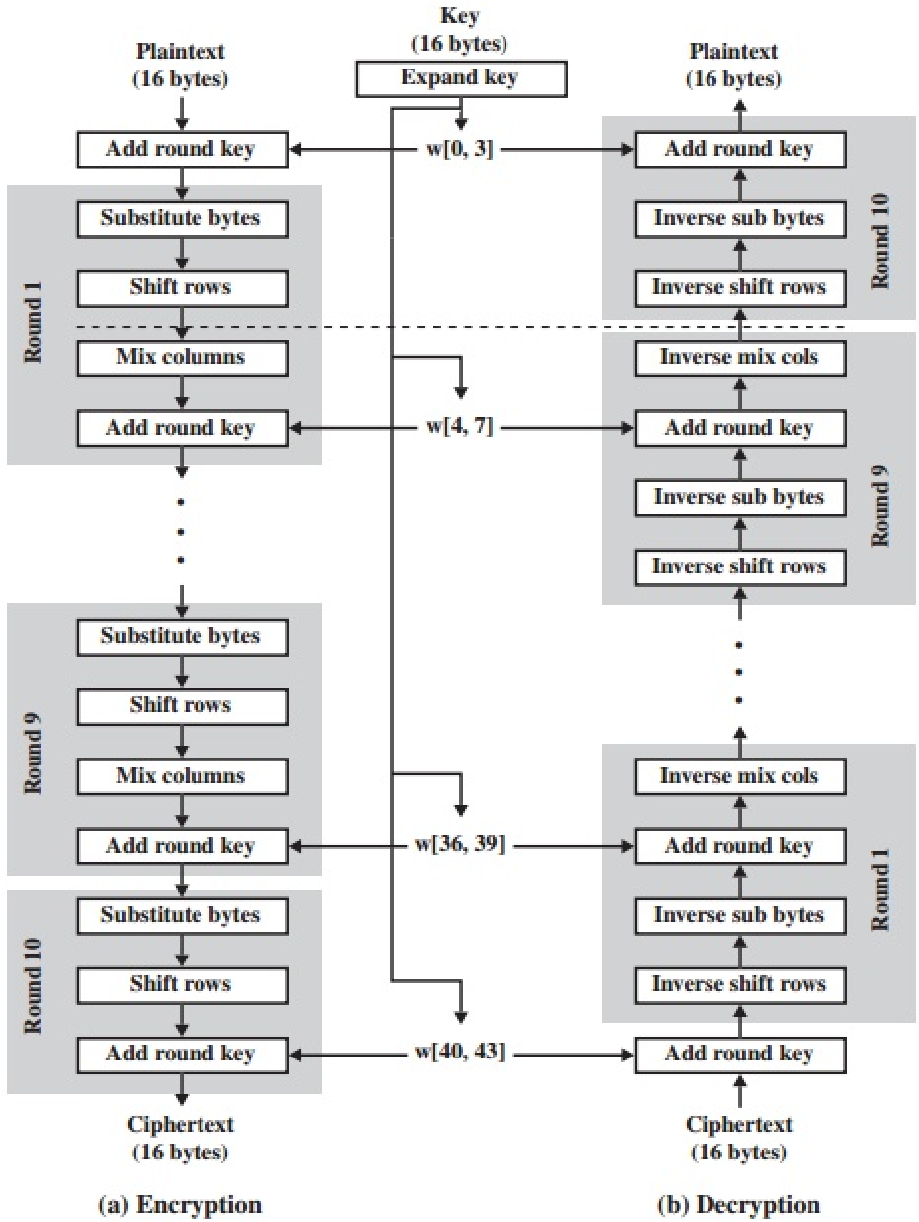


Figure 5.3 AES Encryption and Decryption

AES

Another nice design

1. Good signature
2. Security has held up very well – no remotely damaging attacks
3. Hardware support has emerged on Intel and other platforms, making the algorithm extremely fast (like 0.625 cpb when usage mode permits parallelism)
4. Not great without hardware support
5. Relatively large state and under-considered key setup
6. Open design with minimal intelligence-agency involvement

Switching lemma:
For any adversary A making at most q queries,

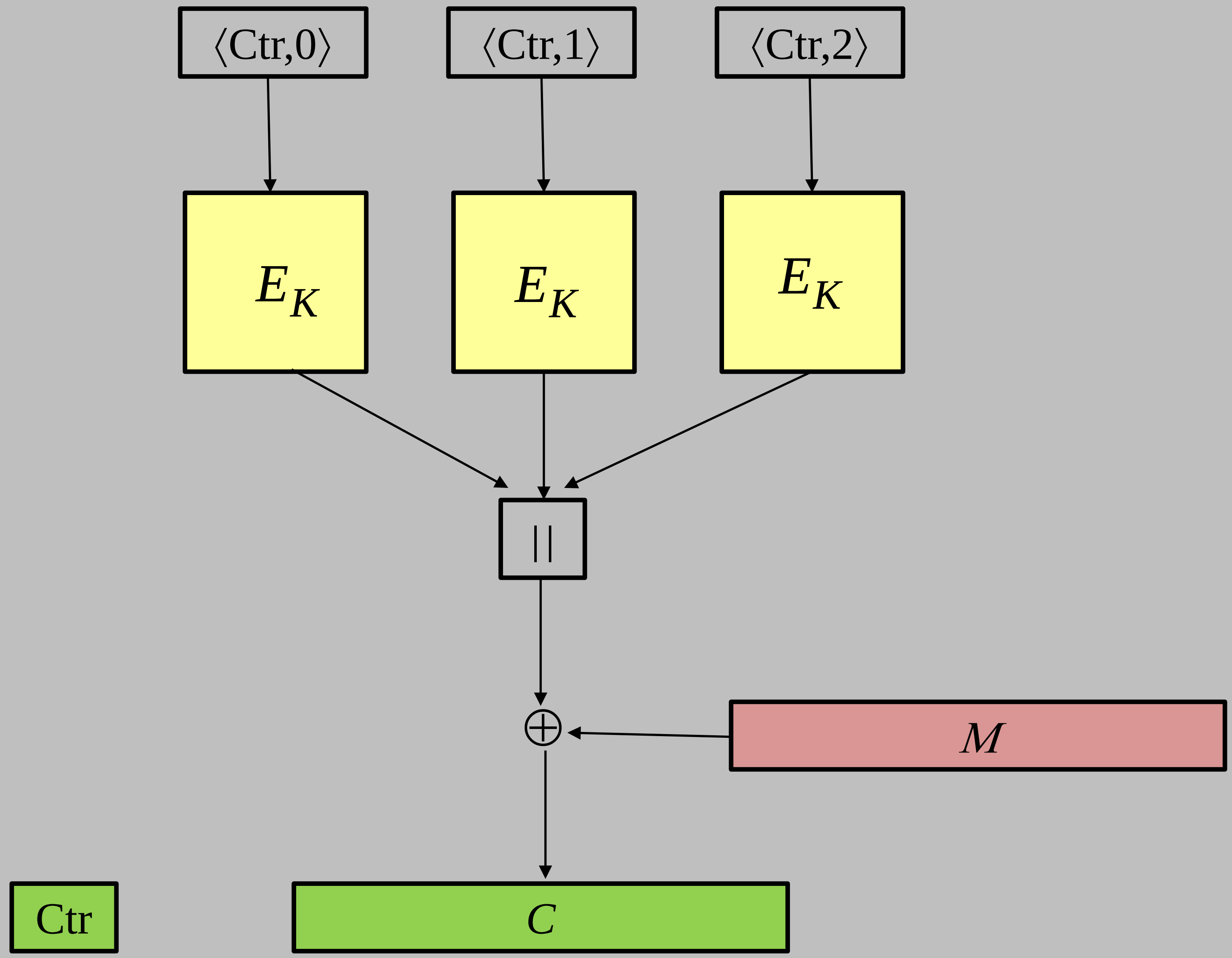
$$\Pr[\pi \twoheadleftarrow \mathrm{Perm}(n)\colon A^{\pi(.)} \Rightarrow 1] - \Pr[\rho \twoheadleftarrow \mathrm{Func}(n,n)\colon A^{\rho(.)} \Rightarrow 1] \leqslant q^2 / 2^{n+1}$$

```
Oracle E(X)
  if X ∈ Dom(f) then return f(X)
  Y ↞ {0,1}^n
  if Y ∈ Ran(f) then bad ← true, Y ↞ {0,1}^n \ Ran(f)
  return Y
```

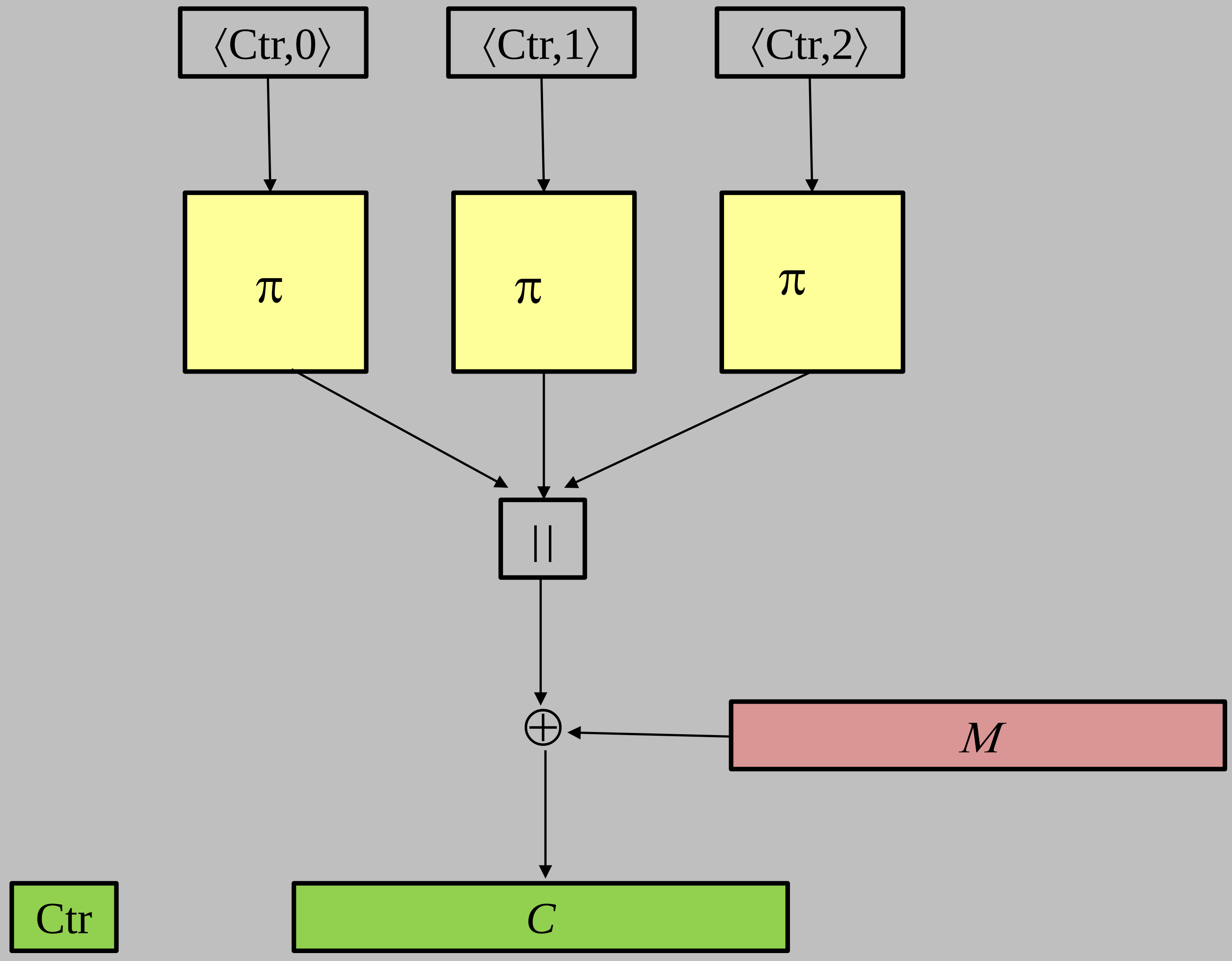
Fundamental lemma of game playing:
If games G and H are identical-until-*bad*, then

$$\mathrm{Adv}^{\mathrm{dist}}_{G,H}(A) = \Pr[A^{G} \Rightarrow 1] - \Pr[A^{H} \Rightarrow 1] \leqslant \Pr[\text{G sets } bad].$$

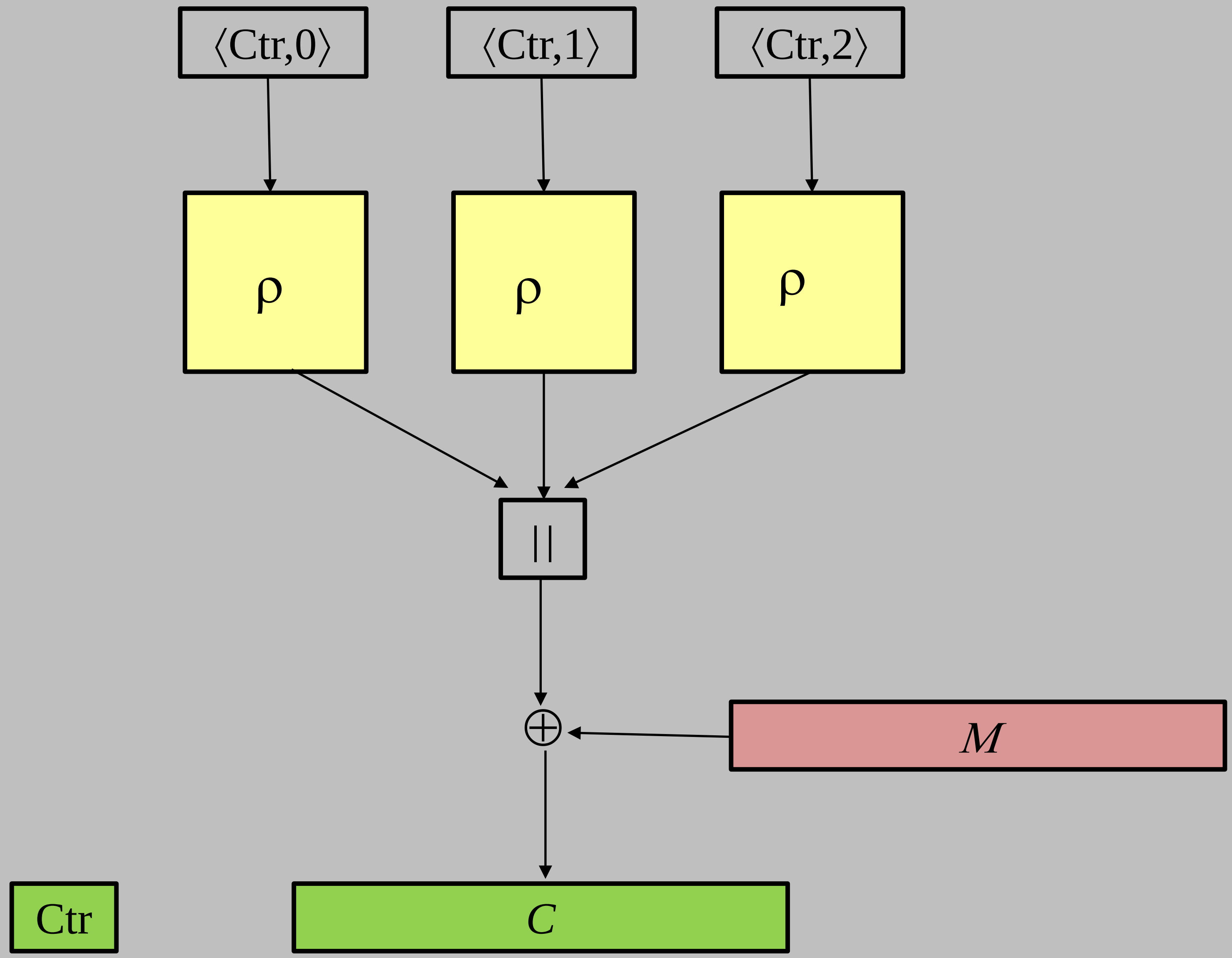
CTR[E]

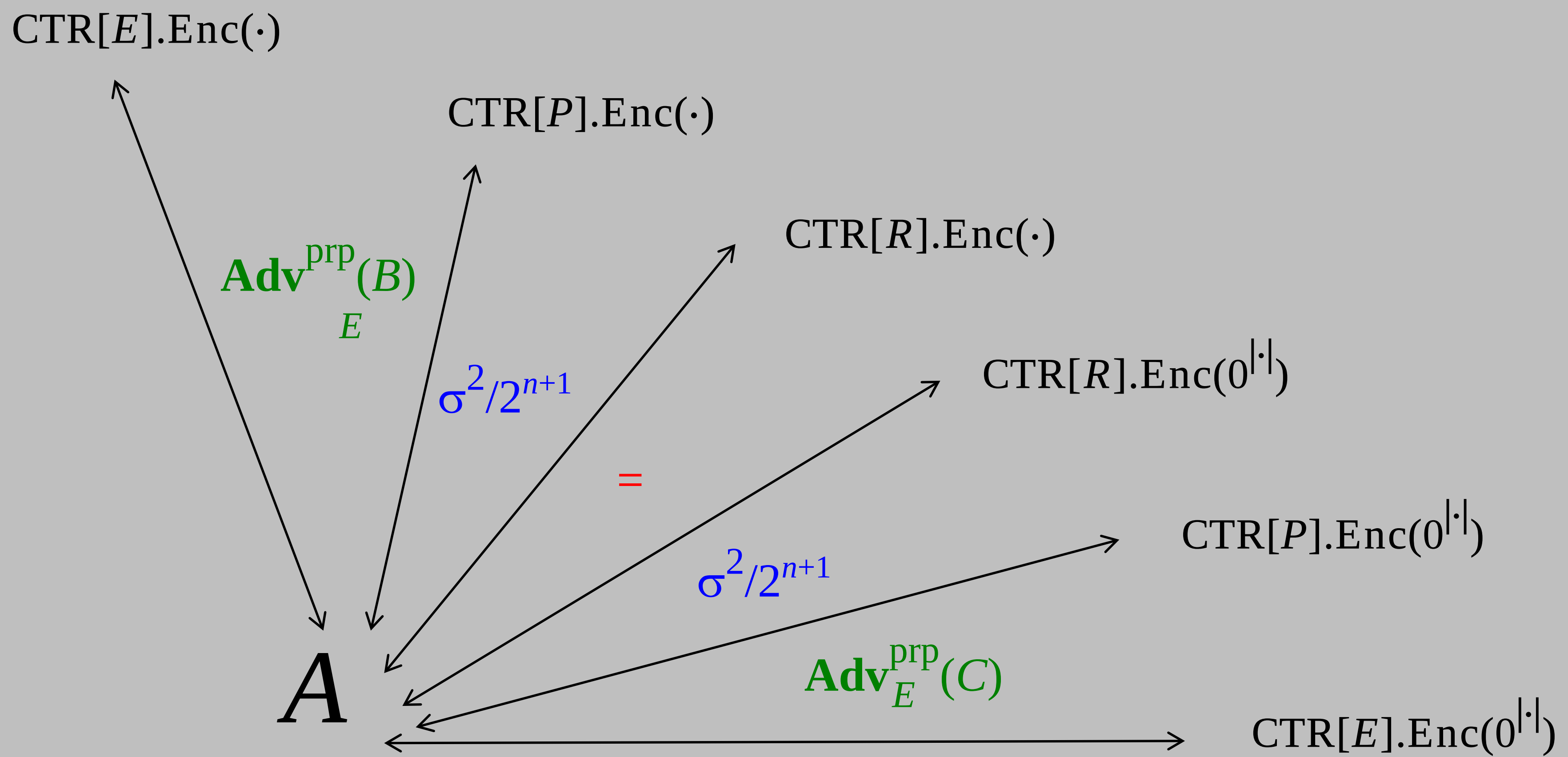


CTR[P]



CTR[R]





Theorem: Let E be an n-bit blockcipher, let Π=CTR[E], and let A be an adversary (for breaking Π) that asks at most σ blocks. Then there's an adversary B that gets advantage

$$\mathbf{Adv}_E^{\mathrm{prp}}(B) \geq 0.5\ \mathbf{Adv}_\Pi^{\mathrm{ind}}(A) - \sigma^2/2^{n+1}$$

Adversary B asks σ queries and run in time approximately that of A.

Define the **prp-advantage** $\mathbf{Adv}_E^{\mathrm{prp}}(A)$ of adversary A attacking E:$\{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ is the number

Question #1

Pr [

Question #2

Not graded / anonymous on a separate piece of paper if you prefer:

How much do you think you **understand** of our class:

Very little | About half | Most things | Almost everything

Any **suggestions** for how I can do better?

What exciting event will happen Friday, Feb 8, in this very class?!

Question #1

Why is it preferred for a PRF/PRP to run in constant time?

Question #2

Consider the PRG $G: \{0,1\}^{100} \rightarrow \{0,1\}^{200}$ defined by

$$G(x) = x \,||\, x$$

An adversary A can do well in breaking G by taking in a 200-bit string $y = y_1 y_2$ (where $|y_1| = |y_2|$) and answering 1 if

Question #1

and answering 0 otherwise.
This adversary gets advantage

Question #2