
Lecture 2 – Block ciphers, PRFs/PRPs, AES

TEK4500

06.09.2023

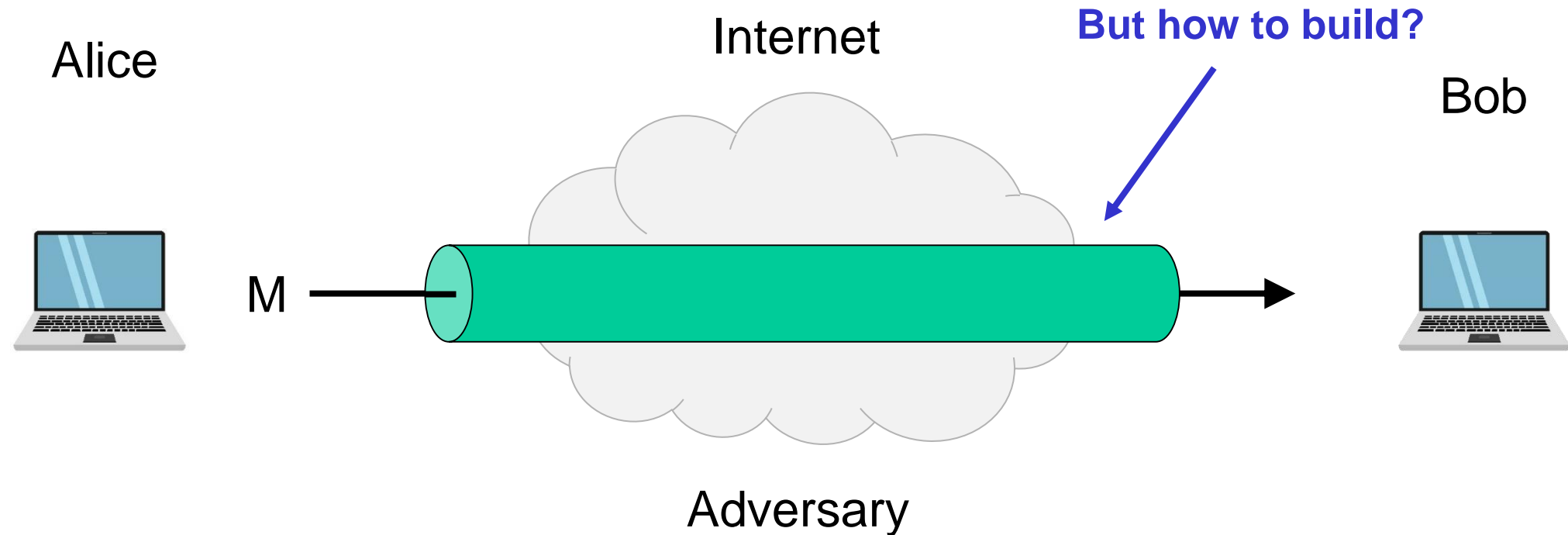
Håkon Jacobsen

hakon.jacobsen@its.uio.no

Announcement

- Recitation hours
 - Today: 14:15-15:00 in Black Box (room 3271 on the 3rd floor of Ole Johan Dahls hus)
 - **Next week (and onwards): 11:15-12:00.**

Ideal solution: secure channels



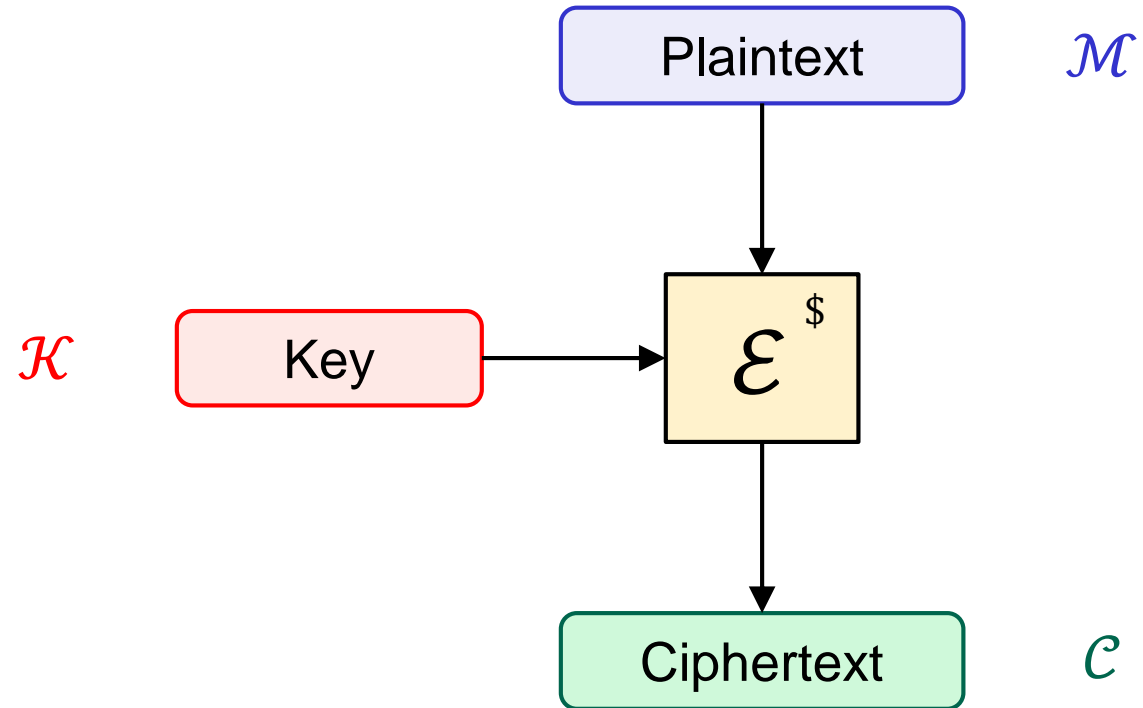
Security goals:

- **Data privacy:** adversary should not be able to read message M ✓
- **Data integrity:** adversary should not be able to modify message M ✓
- **Data authenticity:** message M really originated from Alice ✓

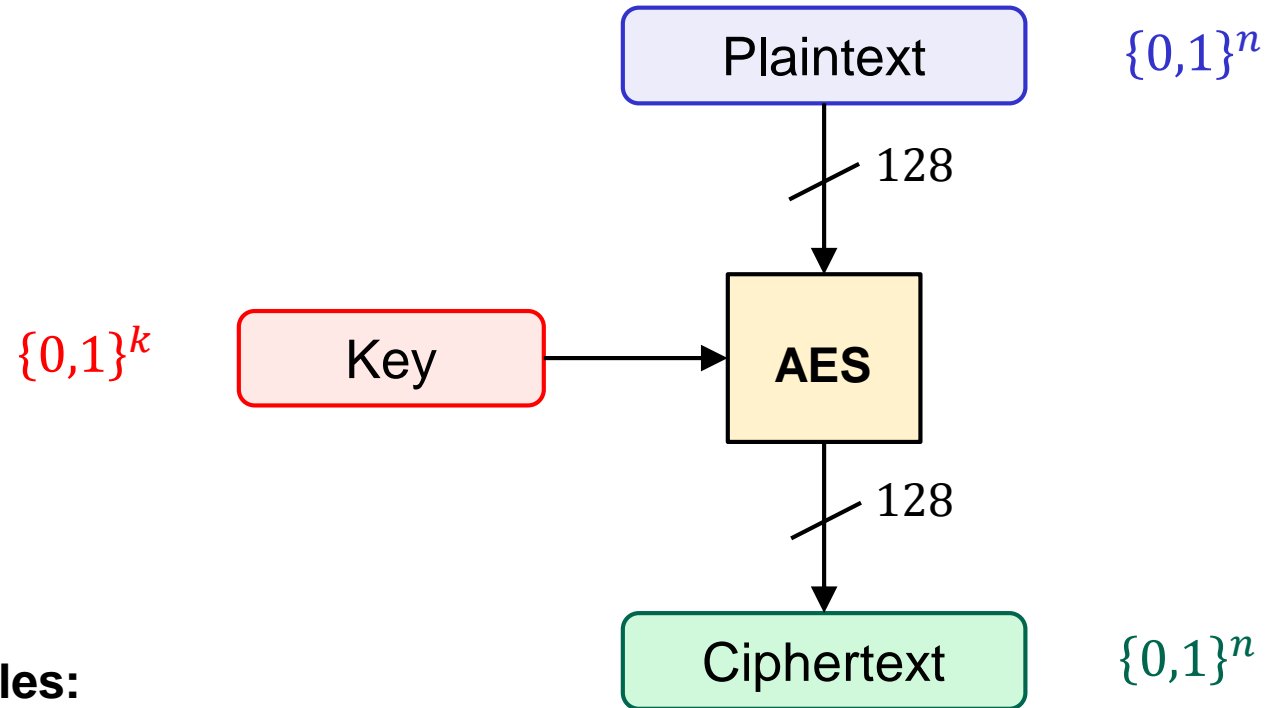
Basic goals of cryptography

	Message privacy	Message integrity / authentication
Symmetric keys	Symmetric encryption	Message authentication codes (MAC)
Asymmetric keys	Asymmetric encryption (a.k.a. public-key encryption)	Digital signatures

Encryption schemes



Block ciphers



Examples:

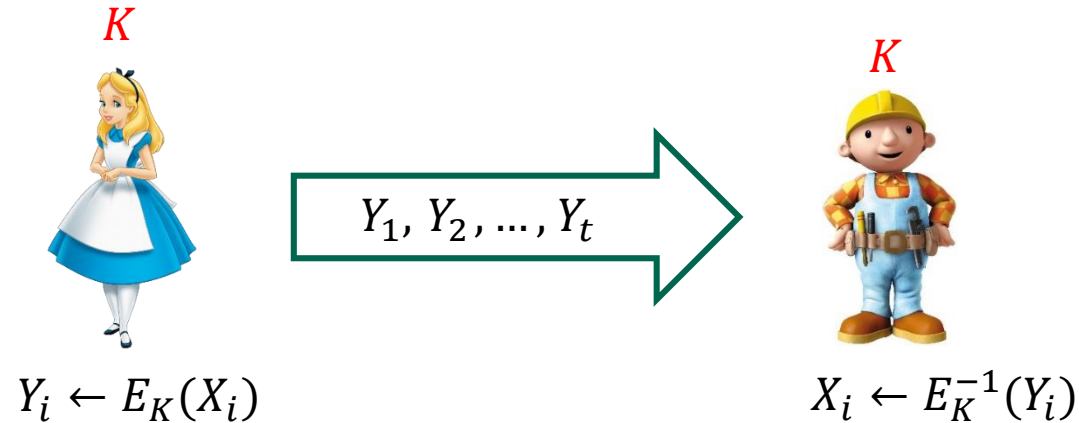
AES-128: $k = 128$, $n = 128$

AES-192: $k = 192$, $n = 128$

AES-256: $k = 256$, $n = 128$

Block cipher applications

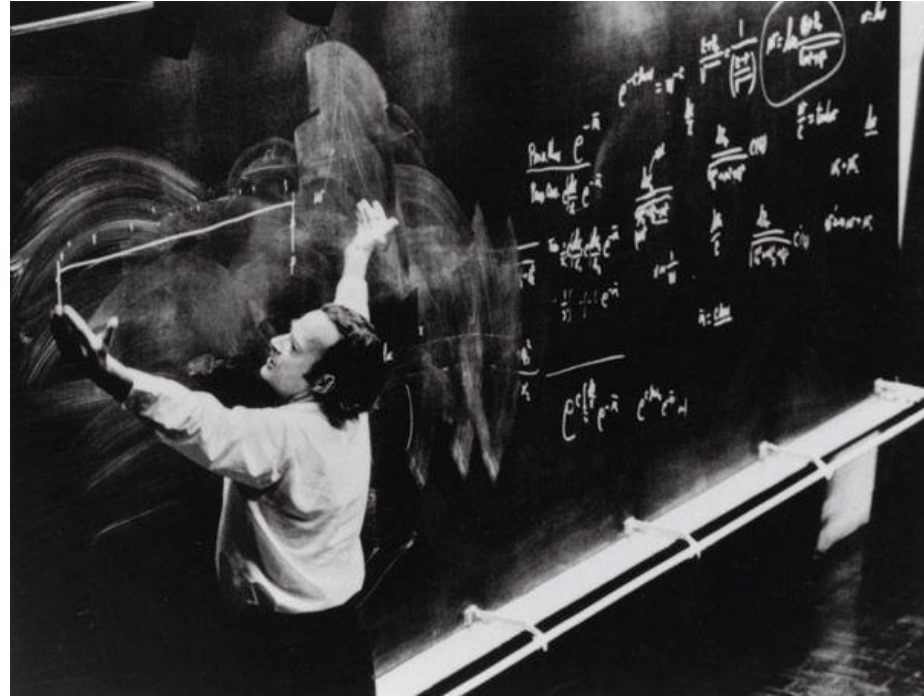
- Encryption of messages of 128 bits (block length)



- **However:** actually want to encrypt messages of *arbitrary* length!
 - Splitting the message into multiple 128 bit blocks (like above) is **not secure!**
 - A **mode-of-operation** is needed (covered later in the course)
- Correct viewpoint: block ciphers are **not** encryption schemes!
 - Block ciphers are **primitives** used to construct other things

Block cipher applications

- The “work horse” of crypto
- Can be used to build:
 - Encryption of arbitrary length messages (including stream ciphers)
 - Message authentication codes
 - Authenticated encryption
 - Hash functions
 - (Cryptographically secure) pseudorandom generators
 - Key derivation functions



Defining block ciphers

Pseudorandom functions (PRFs) and permutations (PRP)

Definition: A pseudorandom function (PRF) is a function

$$F : \{0,1\}^k \times \{0,1\}^{in} \rightarrow \{0,1\}^{out}$$

- k, in, out are called the **key-length**, **input-length**, and **output-length** of F
- Think of a PRF as a *family* of functions:
 - For each $K \in \{0,1\}^k$ we get a function $F_K : \{0,1\}^{in} \rightarrow \{0,1\}^{out}$ defined by $F_K(X) = F(K, X)$

Definition: A pseudorandom permutation (PRP) is a function

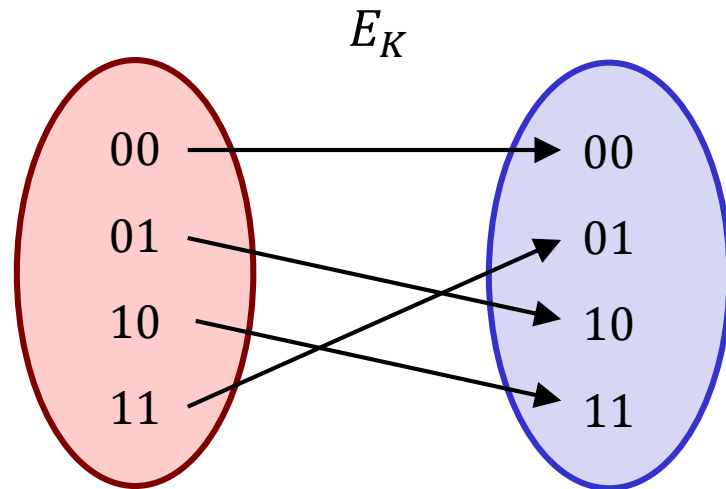
$$E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$$

such that $E_K : \{0,1\}^n \rightarrow \{0,1\}^n$ is a *permutation* for all $K \in \{0,1\}^k$

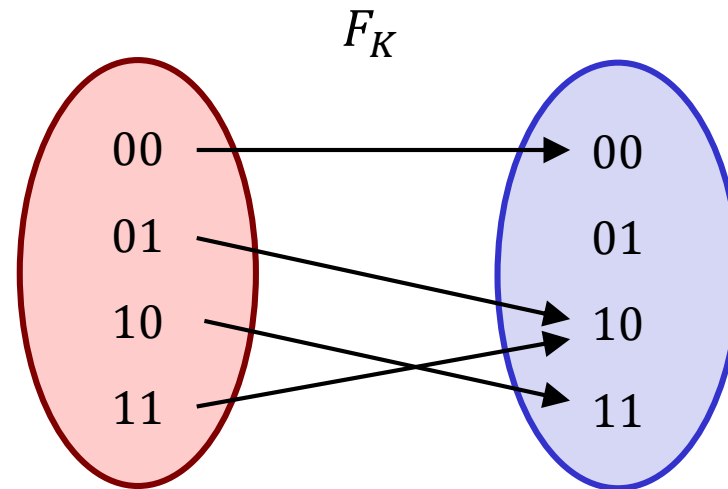
PRP = block cipher

note: all PRPs are PRFs
(but not all PRFs are PRPs!)

Permutations vs. functions

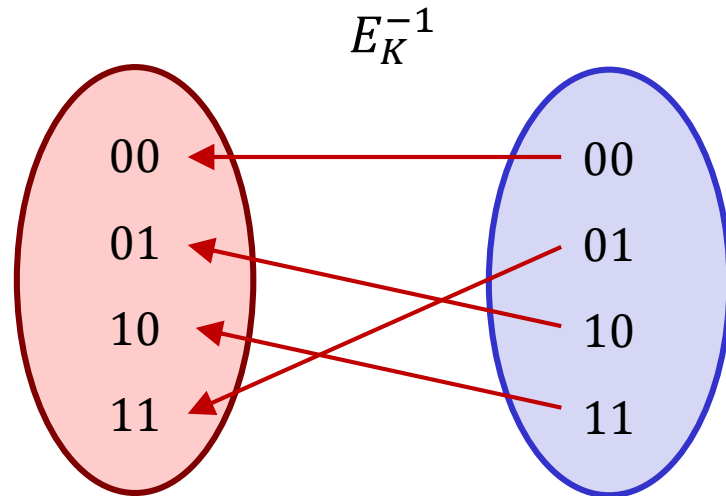


Permutation

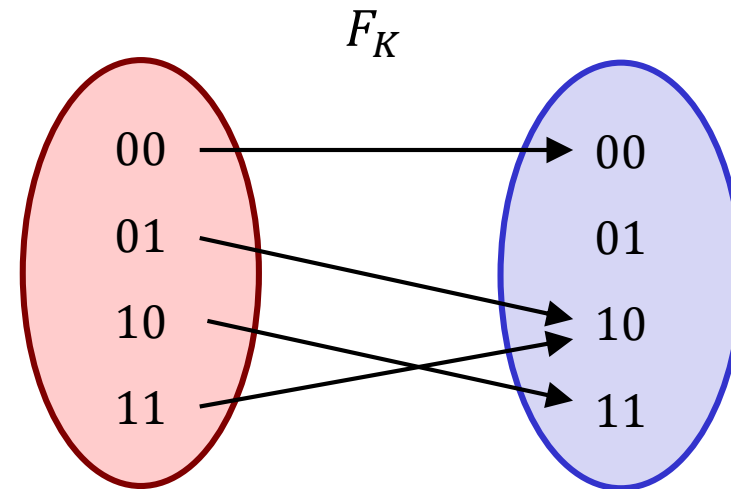


Not a permutation

Permutations vs. functions



Permutation



Not a permutation

Important:

$$E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$$

$$E_K : \{0,1\}^n \rightarrow \{0,1\}^n$$

is **not** a permutation (but a block cipher)
is a permutation

Block cipher security

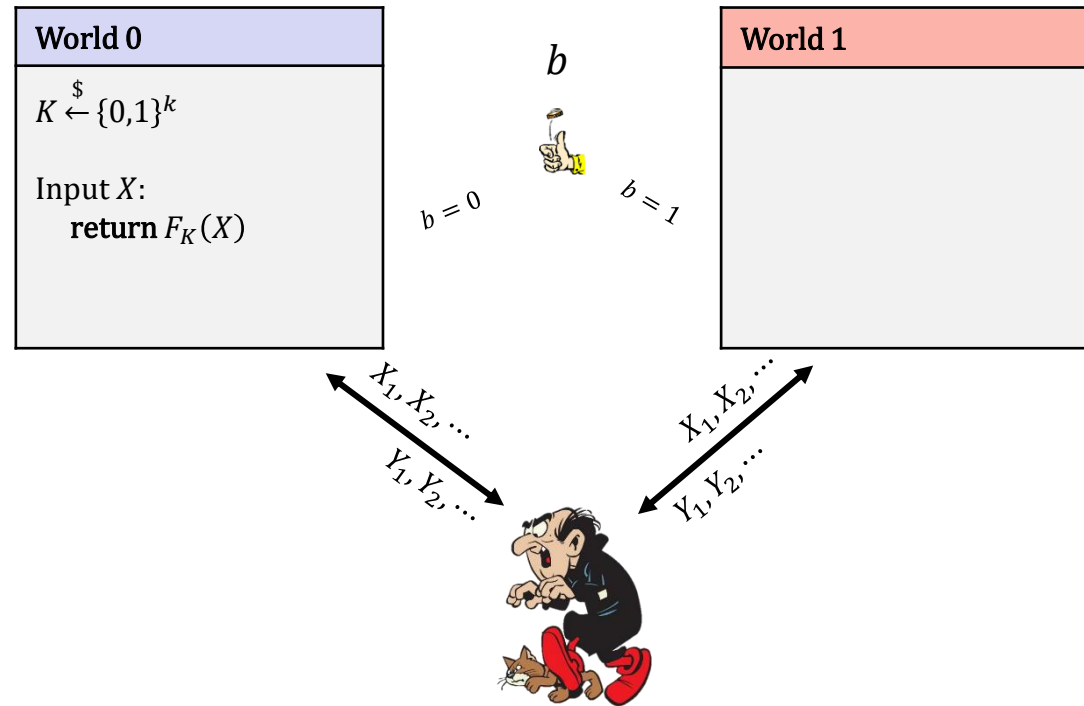
- What should the **security definition** of a block cipher look like?
 - Which security properties should a block cipher satisfy?

- Some suggestions:

- **P1:** Should be hard to obtain K from $E_K(X)$ for secret K
- **P2:** Should be hard to obtain K from $E_K(X_1), E_K(X_2), E_K(X_3) \dots$
- **P3:** Should be hard to obtain X from $E_K(X)$
- **P4:** Should be hard to obtain *any* X_i from $E_K(X_1), E_K(X_2), E_K(X_3) \dots$
- **P5:** Should be hard to learn any *bit* of X from $E_K(X)$
- ~~**P6:** Should be hard to detect repetitions among X_1, X_2, \dots from $E_K(X_1), E_K(X_2), \dots$~~ **Impossible!**
- **P7:** ...

Not good enough!

PRF – security; formal definition



Random functions

$$\tilde{F} : \{0,1\}^{in} \rightarrow \{0,1\}^{out}$$

X	$\tilde{F}(X)$
000 ... 000	101 ... 111
000 ... 001	001 ... 001
000 ... 010	111 ... 100
\vdots	\vdots
111 ... 111	001 ... 001

2^{in}

in out

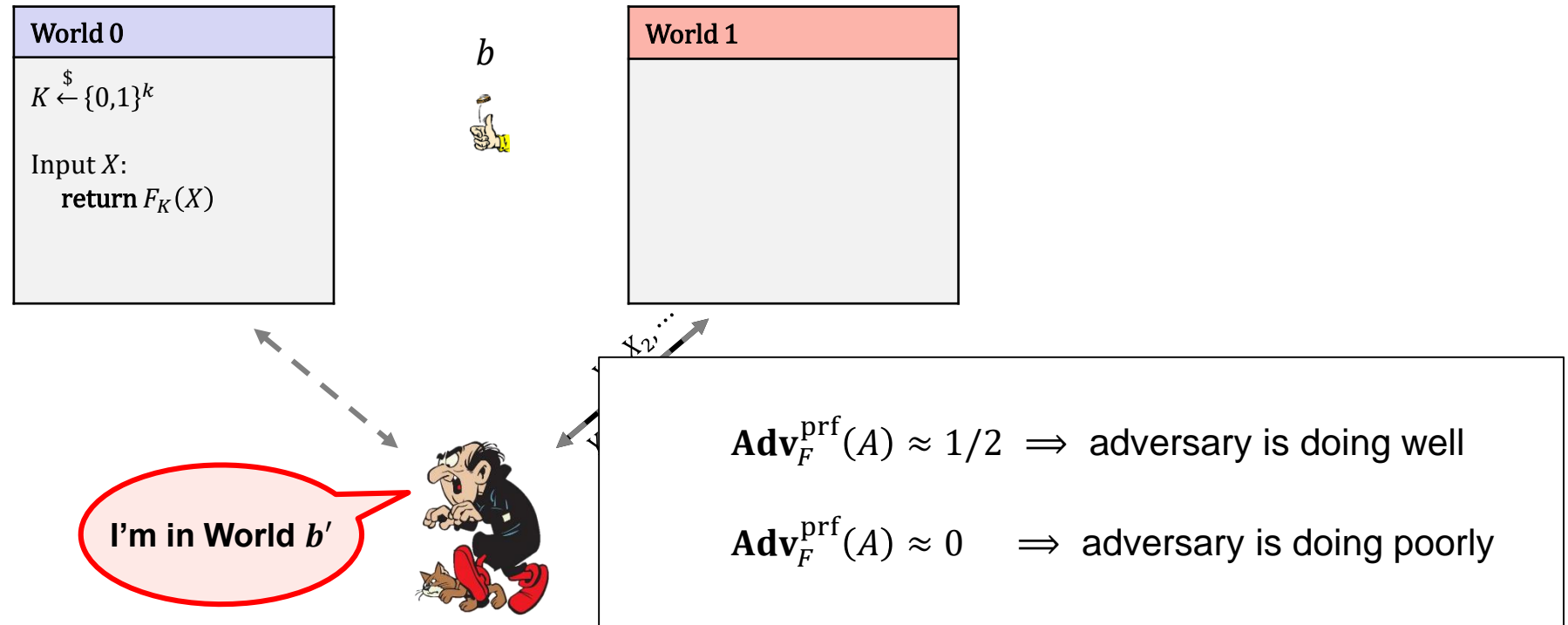
Random functions

$$\tilde{F} : \{0,1\}^{in} \rightarrow \{0,1\}^{out}$$

X	$\tilde{F}(X)$
\vdots	\vdots

1. $T \leftarrow []$
- 2.
3. $\tilde{F}(X):$
4. **if** $T[X] = \perp$:
5. $T[X] \stackrel{\$}{\leftarrow} \{0,1\}^{out}$
6. **return** $T[X]$

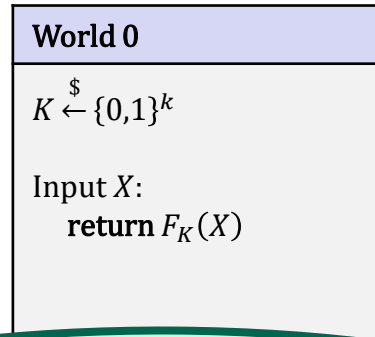
PRF – security; formal definition



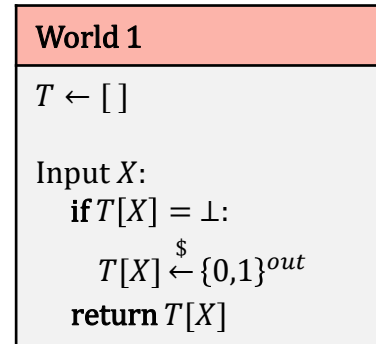
Definition: The **PRF-advantage** of an adversary A is

$$\text{Adv}_F^{\text{prf}}(A) = |\Pr[b' = b] - 1/2|$$

PRF – security; formal definition



b



Intuitive idea: F is a **secure PRF** if $\text{Adv}_F^{\text{prf}}(A)$ is “*small*” for all “*reasonable*” A

$\text{Adv}_F^{\text{prf}}(A) \approx 1 \Rightarrow$ adversary is doing well

$\text{Adv}_F^{\text{prf}}(A) \approx 0 \Rightarrow$ adversary is doing poorly

Definition: The **PRF-advantage** of an adversary A is

$$\text{Adv}_F^{\text{prf}}(A) = |2 \cdot \Pr[b' = b] - 1|$$

Understanding "advantage"

- F is a **secure PRF** if $\text{Adv}_F^{\text{prf}}(A)$ is "*small*" for *all* adversaries A using a "*reasonable*" amount of resources
- PRF security depends on the adversaries':
 - Available resources: running time, number of oracle calls (calls to \tilde{F} / F), memory...
 - *Not* on the adversary strategy! (since we consider *all* possible A)
- What does *small* and *reasonable* mean?
 - **Example: 128-bit** security:

$$\text{Adv}_F^{\text{prf}}(A) \leq \frac{2^{80}}{2^{128}}$$

for all adversaries A that run in time $\leq t \approx 2^{80}$

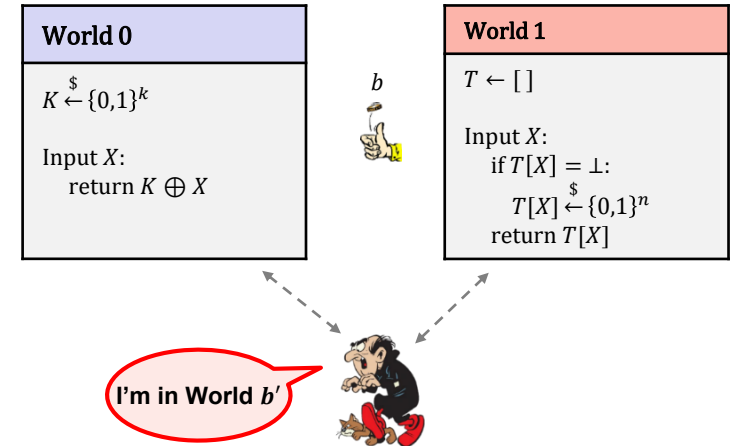
- **Example:** a PRF is *insecure* if we can come up with *one* efficient adversary with good advantage

Example

- Define $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ by $F(K, X) = K \oplus X$
- Claim:** F is *not* a secure PRF

A

1. Query $X = 0^n$



Definition: The **PRF-advantage** of an adversary A is

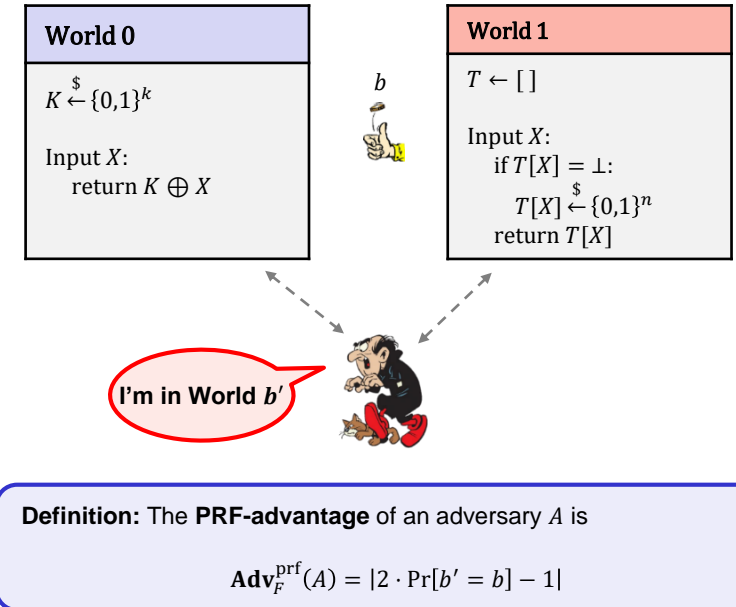
$$\text{Adv}_F^{\text{prf}}(A) = |2 \cdot \Pr[b' = b] - 1|$$

$$\begin{aligned}
 \Pr[b' = b] &= \Pr[b' = 0 \mid b = 0] \cdot \Pr[b = 0] + \Pr[b' = 1 \mid b = 1] \cdot \Pr[b = 1] \\
 &= \Pr[b' = 0 \mid b = 0] \cdot 1/2 + \Pr[b' = 1 \mid b = 1] \cdot 1/2 \\
 &= \Pr[Y \oplus Y' = 1^n \mid b = 0] \cdot 1/2 + \dots \\
 &= 1 \cdot 1/2 + \dots
 \end{aligned}$$

Example

- Define $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ by $F(K, X) = K \oplus X$
- Claim:** F is *not* a secure PRF

A	
1. Query $X = 0^n$	// receive back $Y \leftarrow K \oplus 0^n$ or $Y \stackrel{\$}{\leftarrow} \{0,1\}^n$
2. Query $X' = 1^n$	// receive back $Y' \leftarrow K \oplus 1^n$ or $Y' \stackrel{\$}{\leftarrow} \{0,1\}^n$
3. Output $b' =$	$\begin{cases} 0, & \text{if } Y \oplus Y' = 1^n \\ 1, & \text{otherwise} \end{cases}$

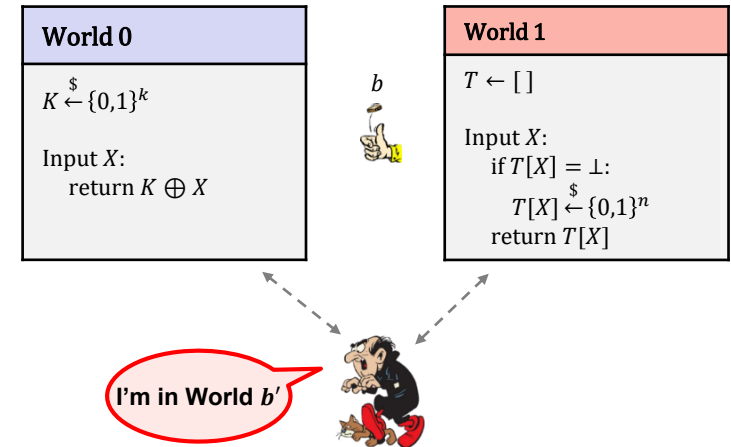


$$\begin{aligned}
 \Pr[b' = b] &= \Pr[b' = 0 \mid b = 0] \cdot \Pr[b = 0] + \Pr[b' = 1 \mid b = 1] \cdot \Pr[b = 1] \\
 &= \Pr[b' = 0 \mid b = 0] \cdot 1/2 + \Pr[b' = 1 \mid b = 1] \cdot 1/2 \\
 &= \Pr[Y \oplus Y' = 1^n \mid b = 0] \cdot 1/2 + (1 - \Pr[b' = 0 \mid b = 1]) \cdot 1/2 \\
 &= 1 \cdot 1/2 + (1 - \Pr[Y \oplus Y' = 1^n \mid b = 1]) \cdot 1/2
 \end{aligned}$$

Example

- Define $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ by $F(K, X) = K \oplus X$
- Claim:** F is *not* a secure PRF

A	
1. Query $X = 0^n$	// receive back $Y \leftarrow K \oplus 0^n$ or $Y \stackrel{\$}{\leftarrow} \{0,1\}^n$
2. Query $X' = 1^n$	// receive back $Y' \leftarrow K \oplus 1^n$ or $Y' \stackrel{\$}{\leftarrow} \{0,1\}^n$
3. Output $b' =$	$\begin{cases} 0, & \text{if } Y \oplus Y' = 1^n \\ 1, & \text{otherwise} \end{cases}$



Definition: The **PRF-advantage** of an adversary A is

$$\text{Adv}_F^{\text{prf}}(A) = |2 \cdot \Pr[b' = b] - 1|$$

$$\begin{aligned}
 \Pr[b' = b] &= \Pr[b' = 0 \mid b = 0] \cdot \Pr[b = 0] + \Pr[b' = 1 \mid b = 1] \cdot \Pr[b = 1] \\
 &= \Pr[b' = 0 \mid b = 0] \cdot 1/2 + \Pr[b' = 1 \mid b = 1] \cdot 1/2 \\
 &= \Pr[Y \oplus Y' = 1^n \mid b = 0] \cdot 1/2 + (1 - \Pr[b' = 0 \mid b = 1]) \cdot 1/2 \\
 &= 1 \cdot 1/2 + (1 - \Pr[Z = 1^n \mid b = 1]) \cdot 1/2 \\
 &= 1/2 + \left(1 - \frac{1}{2^n}\right) \cdot 1/2 = 1 - \frac{1}{2^{n+1}}
 \end{aligned}$$

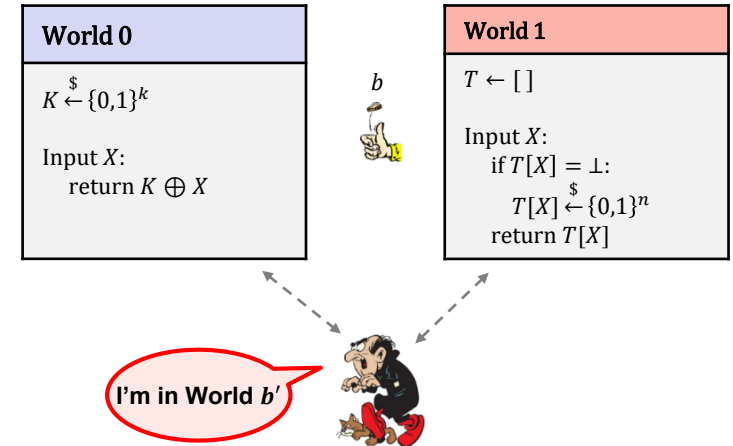
Example

- Define $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ by $F(K, X) = K \oplus X$
- Claim:** F is *not* a secure PRF

A	
1. Query $X = 0^n$	// receive back $Y \leftarrow K \oplus 0^n$ or $Y \stackrel{\$}{\leftarrow} \{0,1\}^n$
2. Query $X' = 1^n$	// receive back $Y' \leftarrow K \oplus 1^n$ or $Y' \stackrel{\$}{\leftarrow} \{0,1\}^n$
3. Output $b' =$	$\begin{cases} 0, & \text{if } Y \oplus Y' = 1^n \\ 1, & \text{otherwise} \end{cases}$

$$\Pr[b' = b] = 1 - \frac{1}{2^{n+1}}$$

$$\text{Adv}_F^{\text{prf}}(A) = |2 \cdot \Pr[b' = b] - 1| = \left| 2 \cdot \left(1 - \frac{1}{2^{n+1}} \right) - 1 \right| = 1 - 2^{-n}$$



Definition: The **PRF-advantage** of an adversary A is

$$\text{Adv}_F^{\text{prf}}(A) = |2 \cdot \Pr[b' = b] - 1|$$

Why is the PRF definition good?

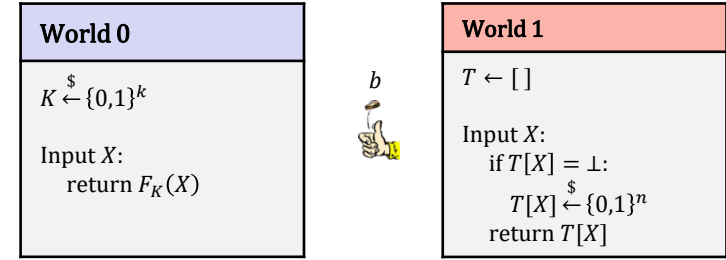
- **P1:** Should be hard to obtain K from $F_K(X)$ for secret K
- **P2:** Should be hard to obtain K from $F_K(X_1), F_K(X_2), F_K(X_3), \dots$
- **P3:** Should be hard to obtain X from $F_K(X)$
- **P4:** Should be hard to obtain *any* X_i from $F_K(X_1), F_K(X_2), F_K(X_3), \dots$
- **P5:** Should be hard to learn any *bit* of X from $F_K(X)$
- **P6:** ...

Logic 101

$A \Rightarrow B$

is equivalent to:

$\text{not } A \Leftarrow \text{not } B$



Definition: The PRF-advantage of an adversary A is

$$\text{Adv}_F^{\text{prf}}(A) = |2 \cdot \Pr[b' = b] - 1|$$

F is PRF secure $\Rightarrow F$ has properties P1, P2, P3, ...

F is **not** PRF secure $\Leftarrow F$ does **not** have one (or more) of properties P1, P2, P3, ...

Why is the PRF definition good?

- **P1:** Should be hard to obtain K from $F_K(X)$ for secret K

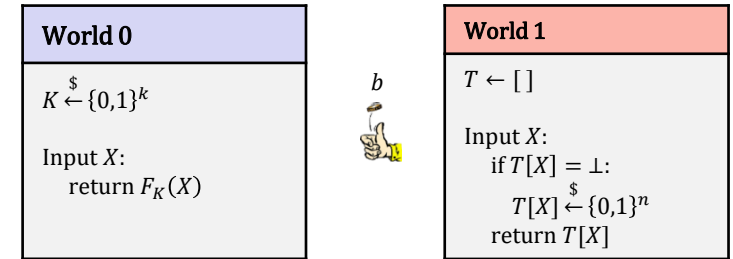


Logic 101

$A \Rightarrow B$

is equivalent to:

not $A \Leftarrow$ not B



Definition: The PRF-advantage of an adversary A is

$$\text{Adv}_F^{\text{prf}}(A) = |2 \cdot \Pr[b' = b] - 1|$$

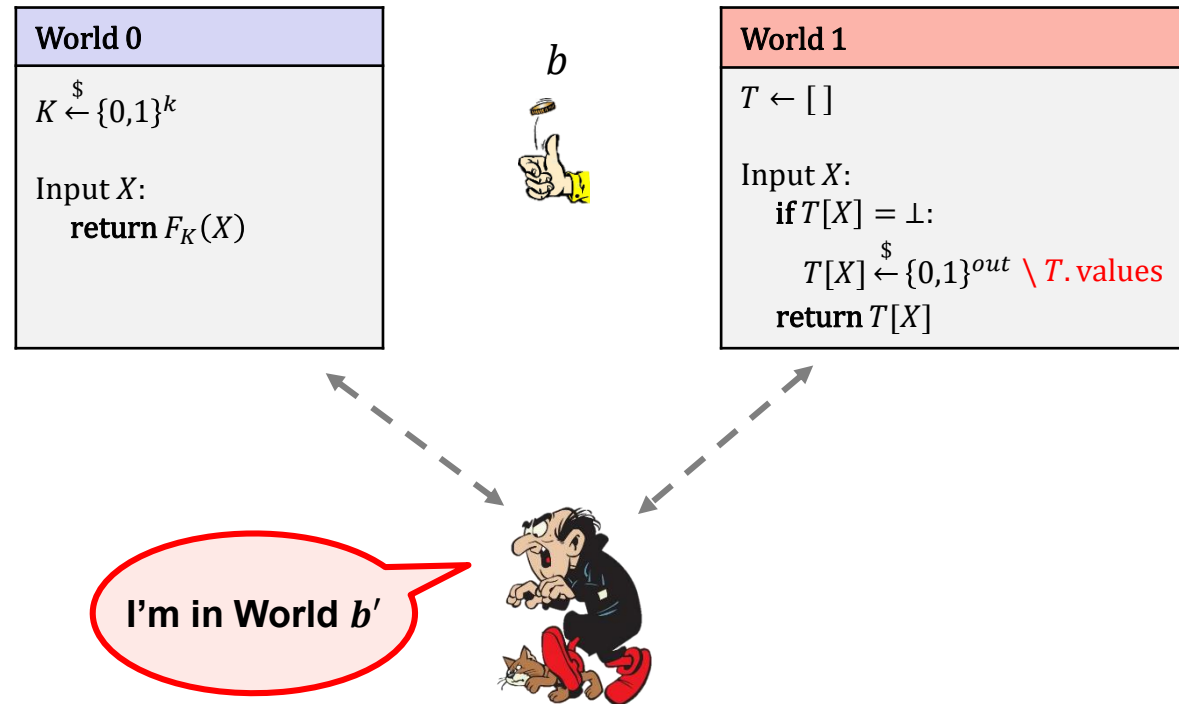
F is PRF secure $\Rightarrow F$ has property P1

F is **not** PRF secure $\Leftarrow F$ does **not** have property P1

$$\Pr[b' = 0 \mid b = 0] = 1$$

$$\Pr[b' = 1 \mid b = 1] = 1 - \frac{1}{2^{\text{out}}}$$

~~PRF~~ PRP – security



Definition: The ~~PRF~~ PRP-advantage of an adversary A is

$$\text{Adv}_F^{\text{prp}}(A) = |2 \cdot \Pr[b' = b] - 1|$$

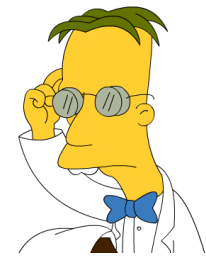
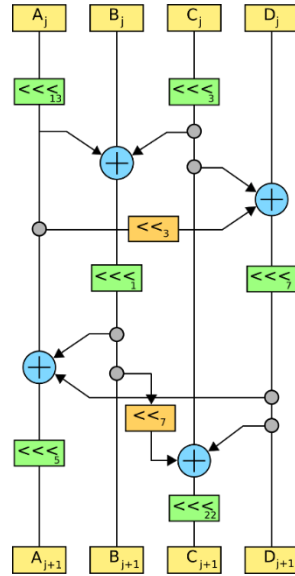
PRP security \Rightarrow PRF security

Theorem: PRP security \Rightarrow PRF security. For all A making at most q oracle queries:

$$\mathbf{Adv}_E^{\text{prf}}(A) \leq \mathbf{Adv}_E^{\text{prp}}(A) + \frac{2q^2}{2^n}$$

Proof sketch:

$$\begin{aligned} \mathbf{Adv}_E^{\text{prf}}(A) &= 2 \cdot \Pr[b' = b] - 1 = 2 \cdot (\Pr[b' = b \wedge \overline{\text{coll}}] + \Pr[b' = b \wedge \text{coll}]) - 1 \\ &\leq 2 \cdot (\Pr[b' = b \wedge \overline{\text{coll}}] + \Pr[\text{coll}]) - 1 \\ &\leq 2 \cdot \Pr[b' = b \wedge \overline{\text{coll}}] + 2 \cdot \frac{q^2}{2^n} - 1 \\ &= \mathbf{Adv}_E^{\text{prp}}(A) + \frac{2q^2}{2^n} \end{aligned}$$



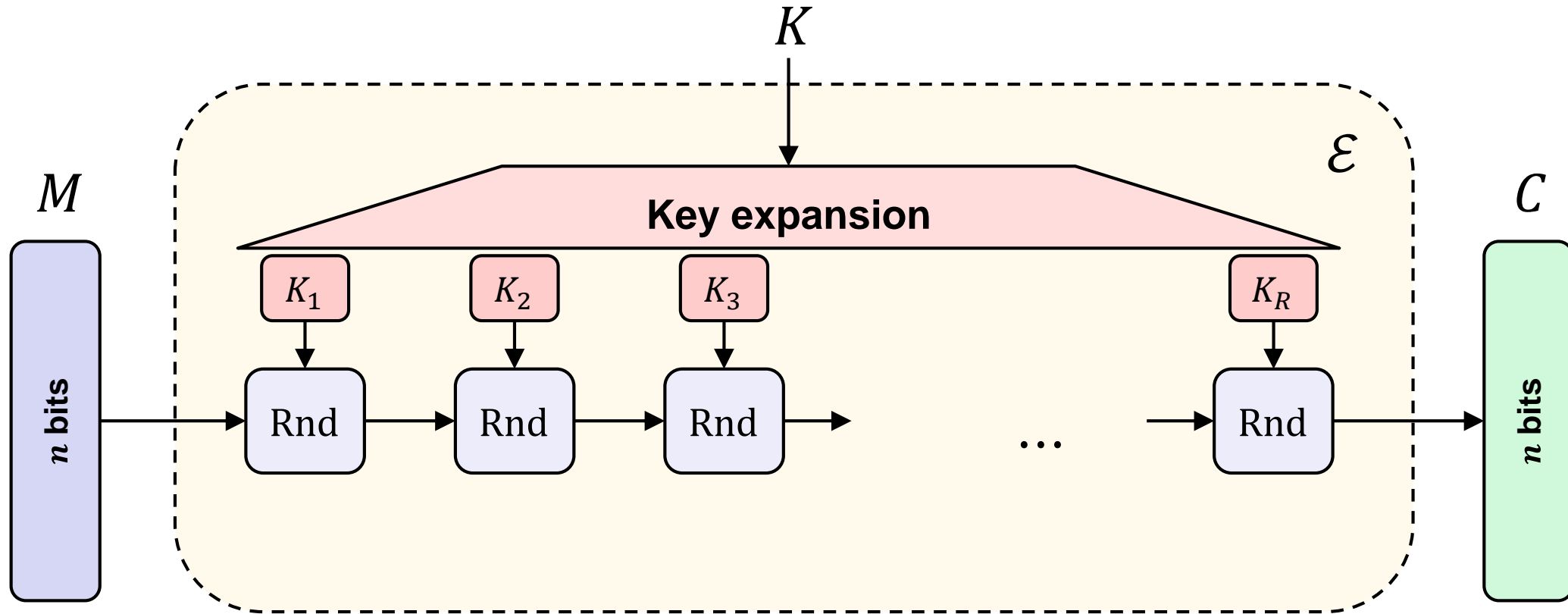
Constructing block ciphers

Principles for designing block ciphers

Claude Shannon, “Communication Theory of Secrecy Systems”(1949):

- **Diffusion:** plaintext spread over large parts of the ciphertext
- **Confusion:** a complex relation between plaintext, key and ciphertext

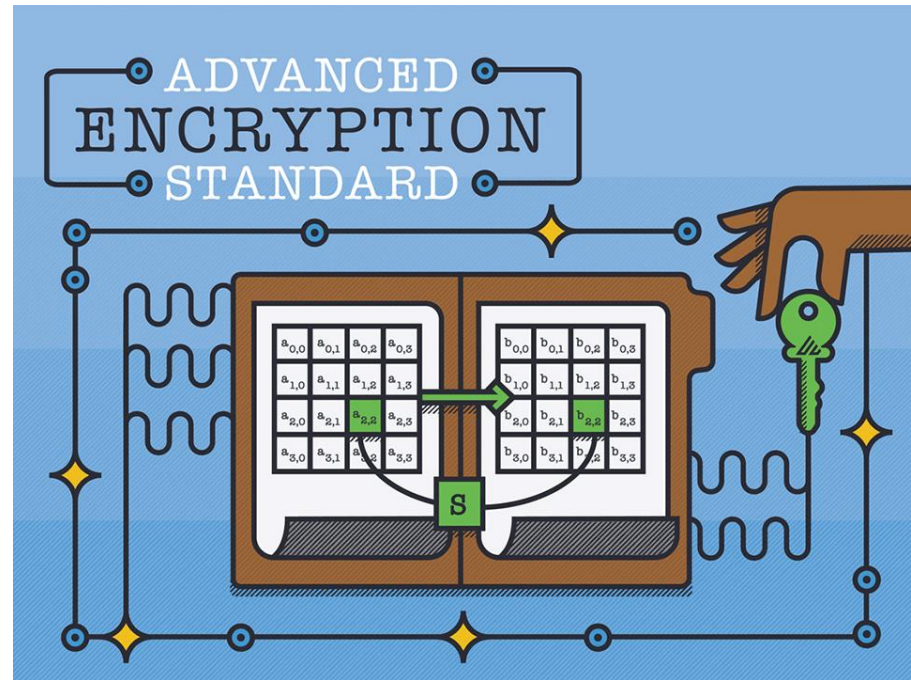
Block ciphers



$\text{Rnd}(K_i, M)$ is called a **round function**

AES-128/192/256

$R = 10/12/14$



Advanced Encryption Standard

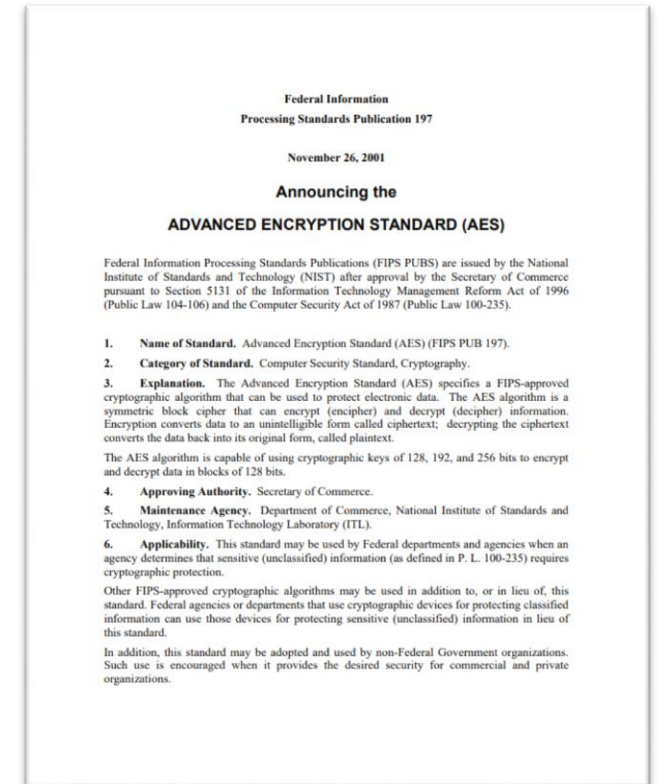
AES

- History
 - Data Encryption Standard (DES) used since the 70's
 - Key length too short (56 bits)
 - Block length too short (64 bits)
 - Too slow in software
 - 1997: NIST hosts a competition to find a new block cipher
 - 2001: Rijndael is announced the winner → renamed AES
- If you remember nothing else:

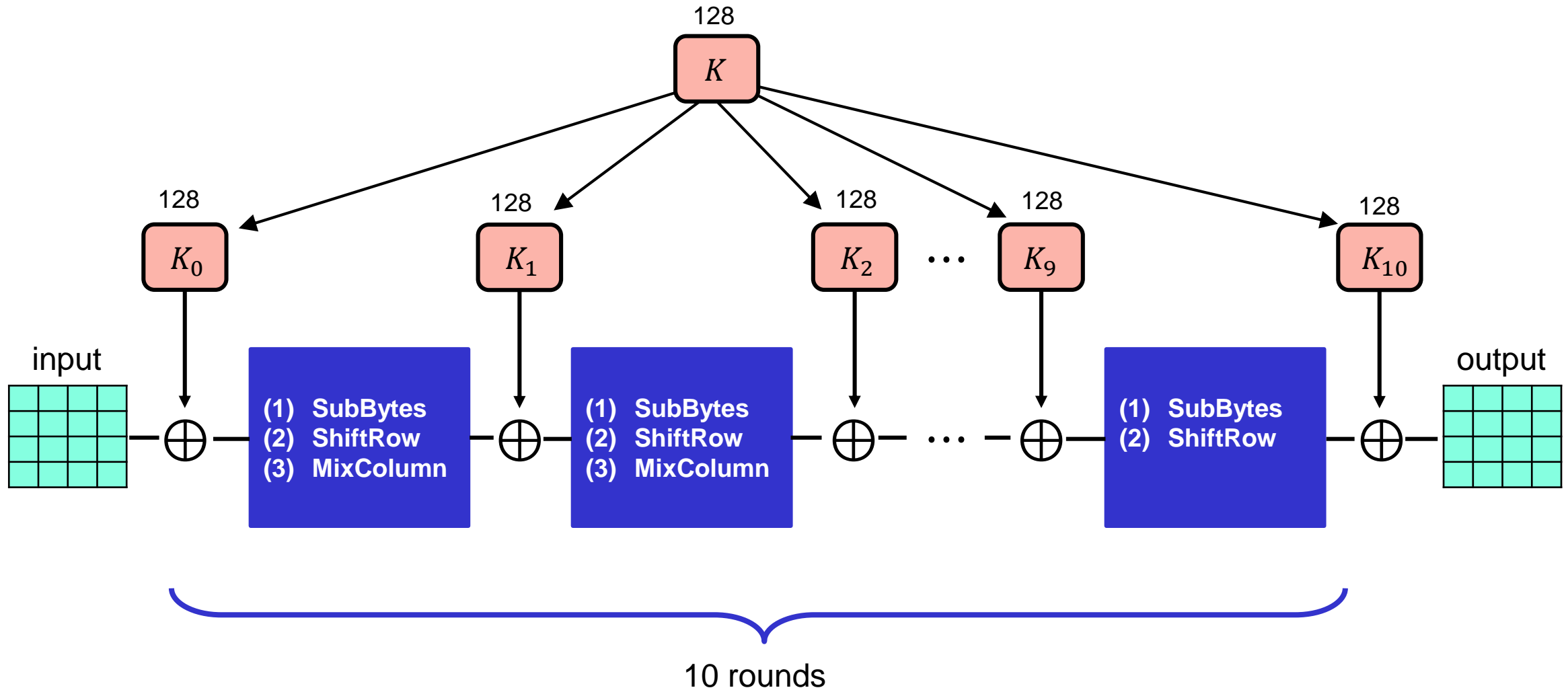
$$\text{AES-128} : \{0,1\}^{128} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$$

$$\text{AES-192} : \{0,1\}^{192} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$$

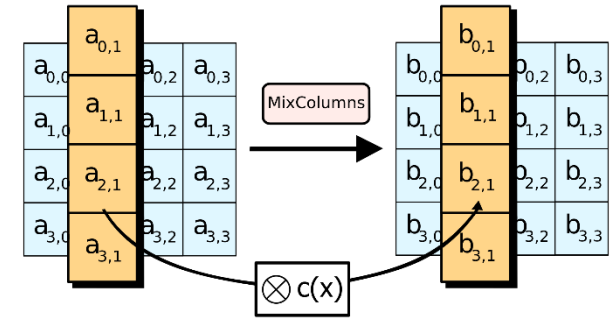
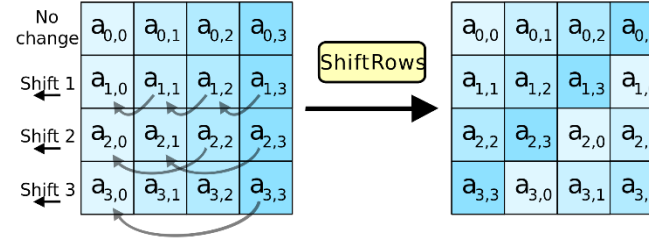
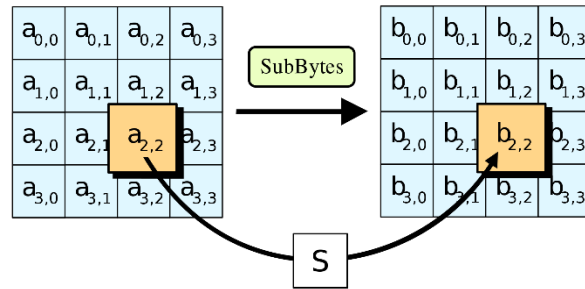
$$\text{AES-256} : \{0,1\}^{256} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$$



AES-128

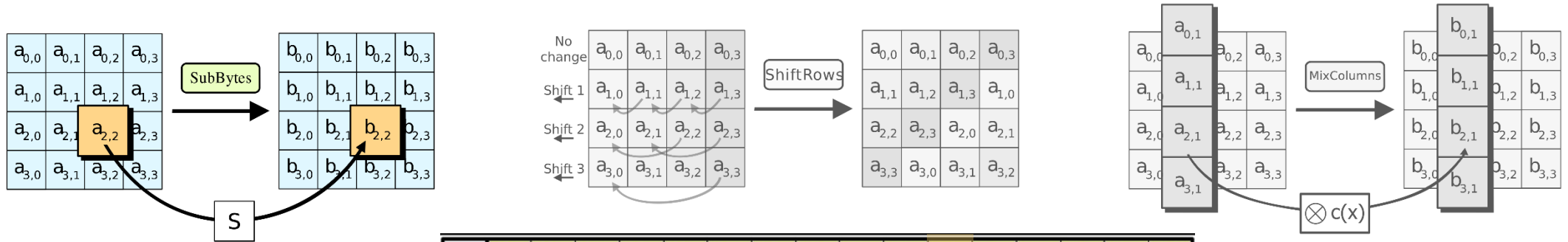


AES round function



- (1) SubBytes
- (2) ShiftRow
- (3) MixColumn

AES round function - SubBytes

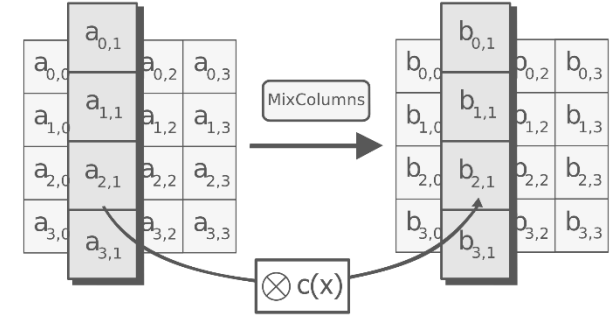
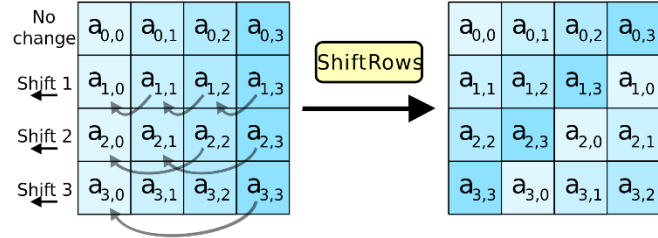
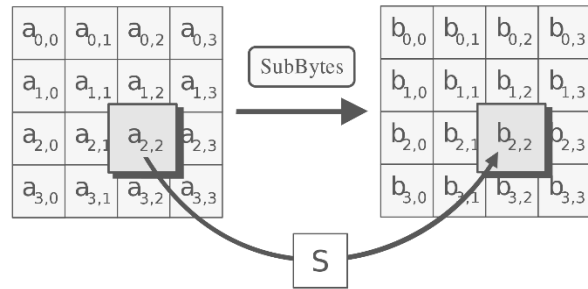


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	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	E7	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

$a_{2,2} = 8A$

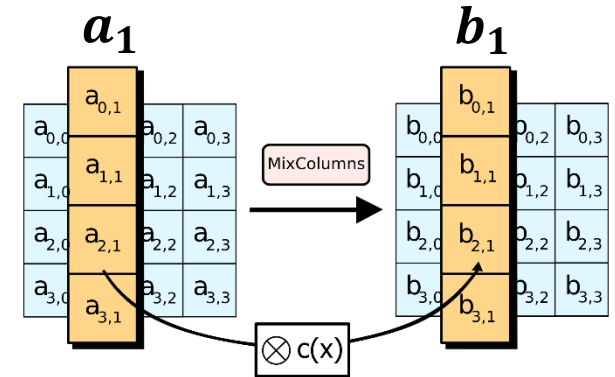
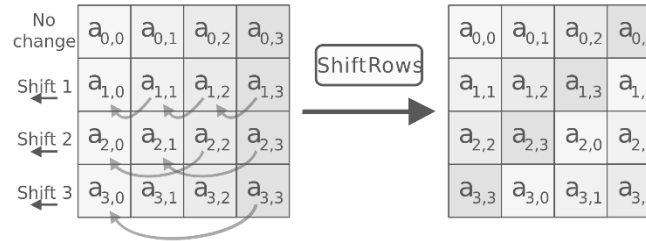
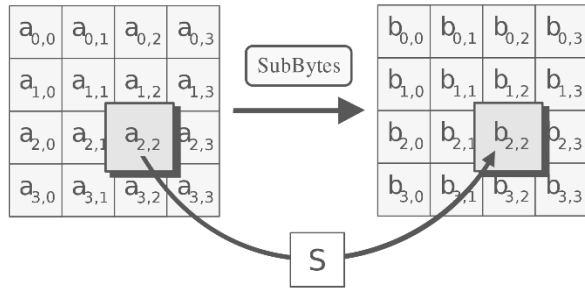
$b_{2,2} = 7E$

AES round function - ShiftRows



- (1) SubBytes
- (2) ShiftRow
- (3) MixColumn

AES round function - MixColumns



- (1) SubBytes
- (2) ShiftRow
- (3) MixColumn

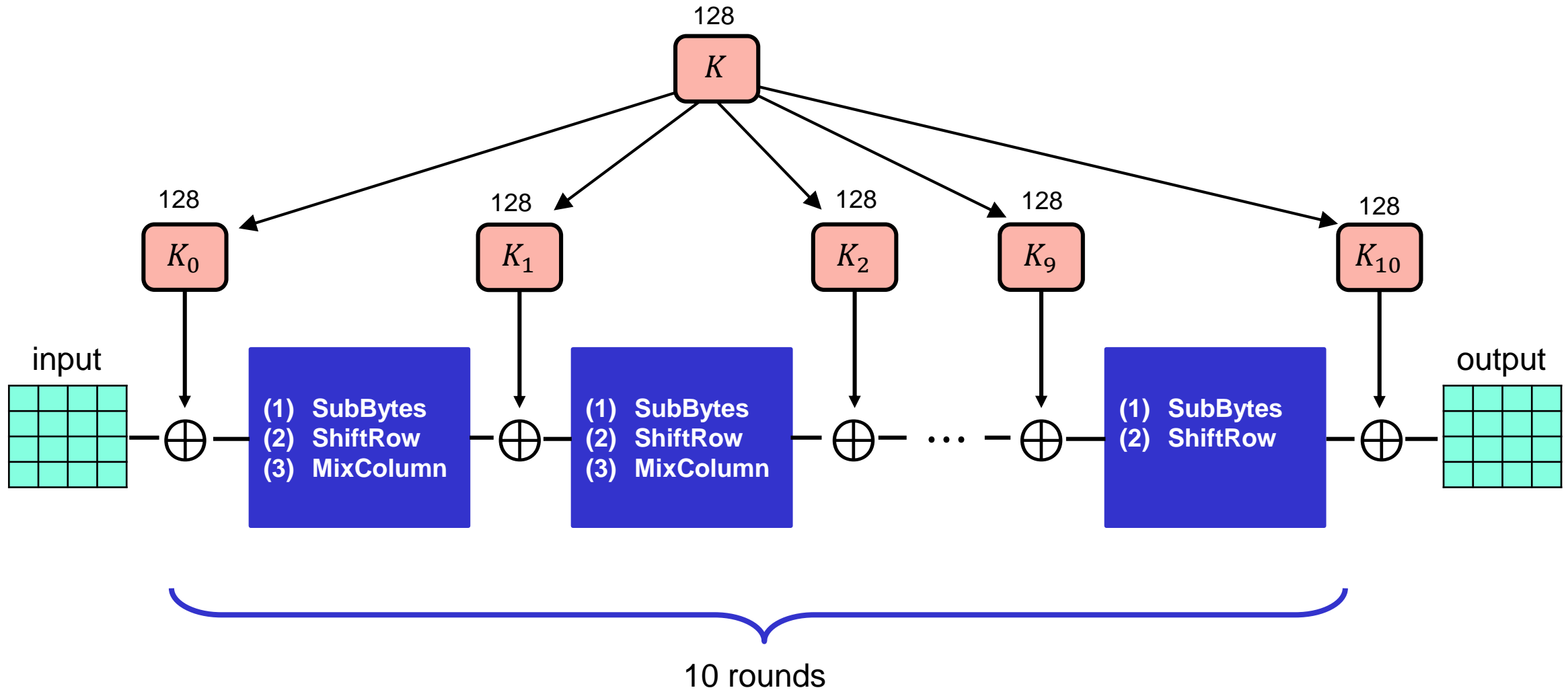
$$M a_1 = b_1$$

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} a_{0,1} \\ a_{1,1} \\ a_{2,1} \\ a_{3,1} \end{pmatrix} = \begin{pmatrix} b_{0,1} \\ b_{1,1} \\ b_{2,1} \\ b_{3,1} \end{pmatrix}$$

$$1 \cdot a_{0,1} + 2 \cdot a_{1,1} + 3 \cdot a_{2,1} + 1 \cdot a_{3,1} = b_{1,1}$$

$$a_{0,1} \oplus 2 * a_{1,1} \oplus 3 * a_{2,1} \oplus a_{3,1} = b_{1,1}$$

AES-128



AES round

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

SubBytes	$b_{i,j} = S[a_{i,j}]$
ShiftRows	$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix}$
MixColumns	$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}$
AddRoundKey	$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$

$$\begin{aligned} \begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} &= \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S[a_{0,j}] \\ S[a_{1,j-1}] \\ S[a_{2,j-2}] \\ S[a_{3,j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix} \\ &= \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \cdot S[a_{0,j}] \right) \oplus \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot S[a_{1,j-1}] \right) \\ &\quad \oplus \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot S[a_{2,j-2}] \right) \oplus \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot S[a_{3,j-3}] \right) \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix} \end{aligned}$$

$T_0[x] = \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \cdot S[x] \right)$	$T_1[x] = \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot S[x] \right)$	$T_2[x] = \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot S[x] \right)$	$T_3[x] = \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot S[x] \right)$
---	---	---	---

AES round

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

SubBytes	$b_{i,j} = S[a_{i,j}]$
ShiftRows	$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix}$
MixColumns	$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}$
AddRoundKey	$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = T_0[a_{0,j}] \oplus T_1[a_{1,j-1}] \oplus T_2[a_{2,j-2}] \oplus T_3[a_{3,j-3}] \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

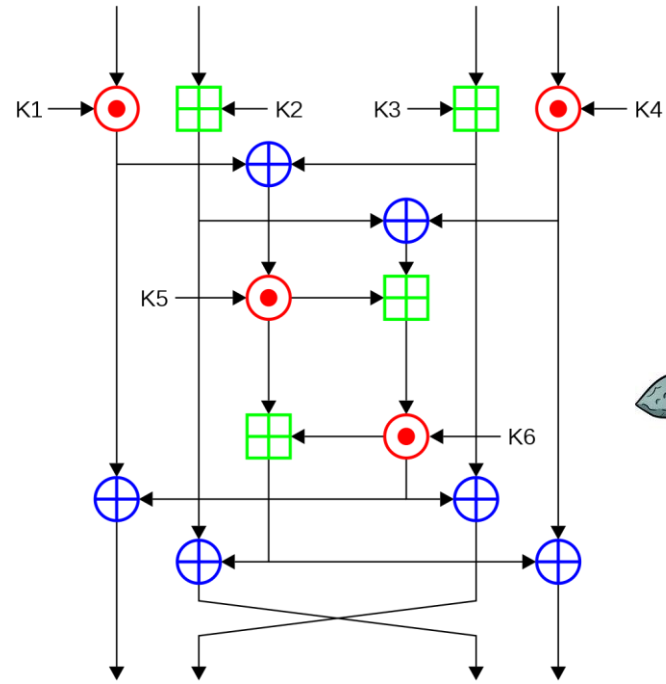
$T_0[x] = \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \cdot S[x] \right)$	$T_1[x] = \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot S[x] \right)$	$T_2[x] = \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot S[x] \right)$	$T_3[x] = \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot S[x] \right)$
---	---	---	---

AES performance

- AES is reasonably efficient in software
 - T-table implementation very fast (but not secure!)
 - Hard to implement fast and constant-time

	Throughput (my laptop)
AES-128 (in software)	0.27 GB/s
AES-128 (w/AES-NI)	3.45 GB/s

- Intel introduced dedicated AES instructions into their CPUs (AES-NI):
 - **aesenc, aesenclast**: do one round of AES in one cycle
 - **aeskeygenassist**: do AES key expansion
 - **aesdec, aesdeclast**: do one round of AES decryption in one cycle
 - **aesimc**: do AES inverser MixColumns
- Now standard in all modern CPUs



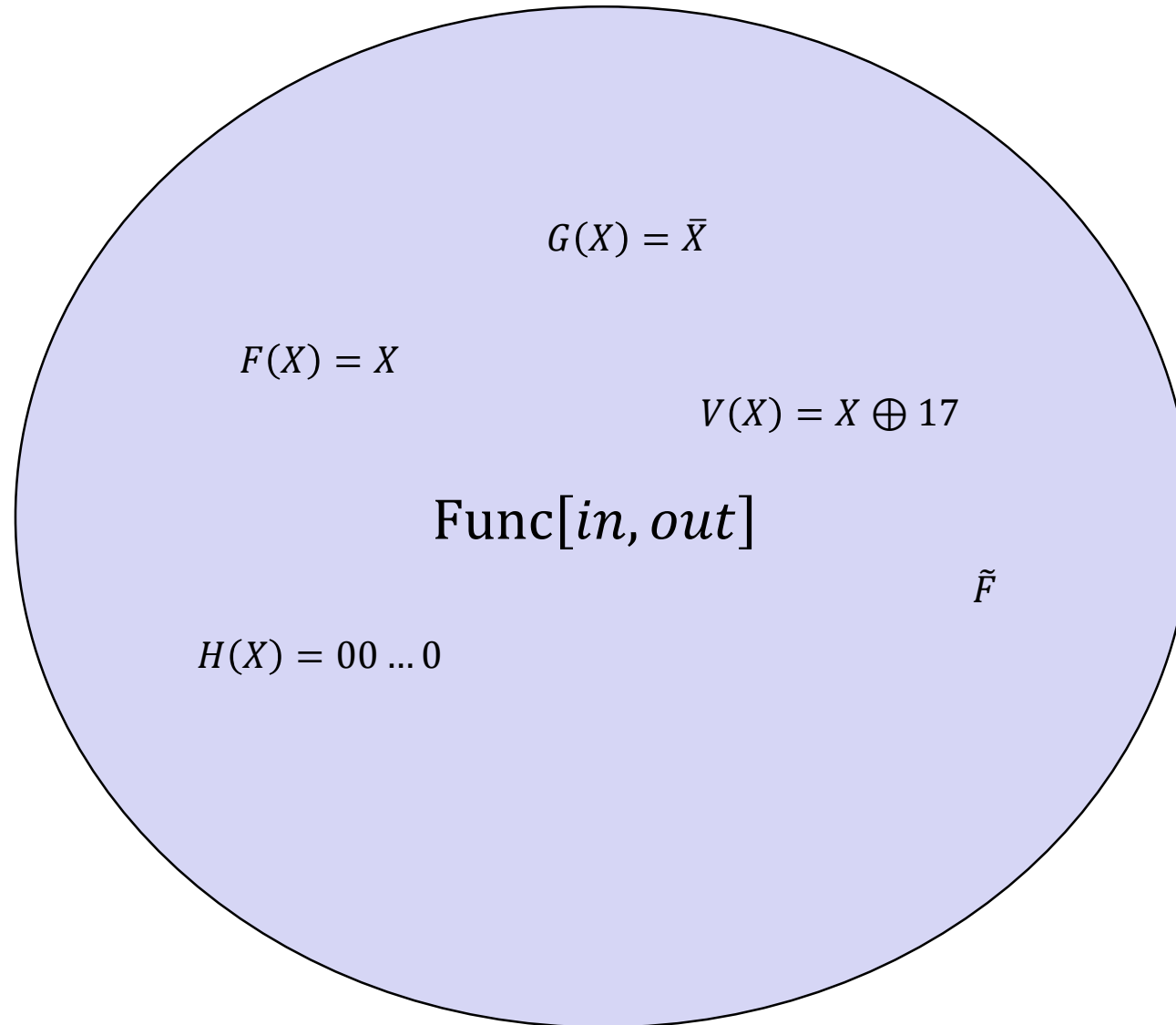
Attacking block ciphers

Attacks on block ciphers

- Brute force attacks: search through every possible key in key space
 - Generic: works for all block ciphers
 - Not practical for large key spaces
- Advanced attacks: try to exploit the concrete details of the block cipher
 - Differential cryptanalysis ('90, but known by the designers of DES + NSA since mid '70)
 - Linear cryptanalysis ('92)
 - AES designed to resist both
- Implementation attacks: vulnerabilities due to implementation characteristics
 - Power draw
 - Timing
 - Cache misses

PRF security – an equivalent view

$|\text{Func}[in, out]| =$



X	$\tilde{F}(X)$
000 ... 000	101 ... 111
000 ... 001	001 ... 001
000 ... 010	111 ... 100
000 ... 011	101 ... 000
\vdots	\vdots
111 ... 111	001 ... 001

2^{in}

out

PRF security – an equivalent view

$$|\text{Func}[in, out]| = \# \text{ bitstrings of length } 2^{in} \cdot out$$

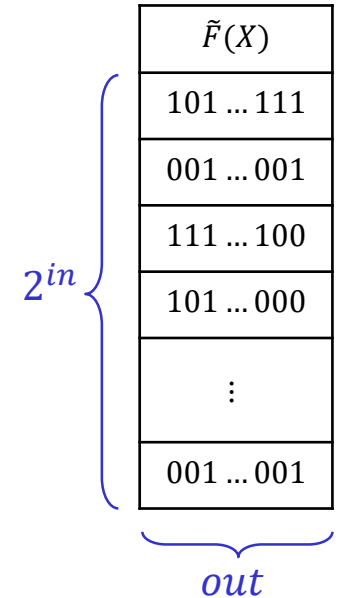
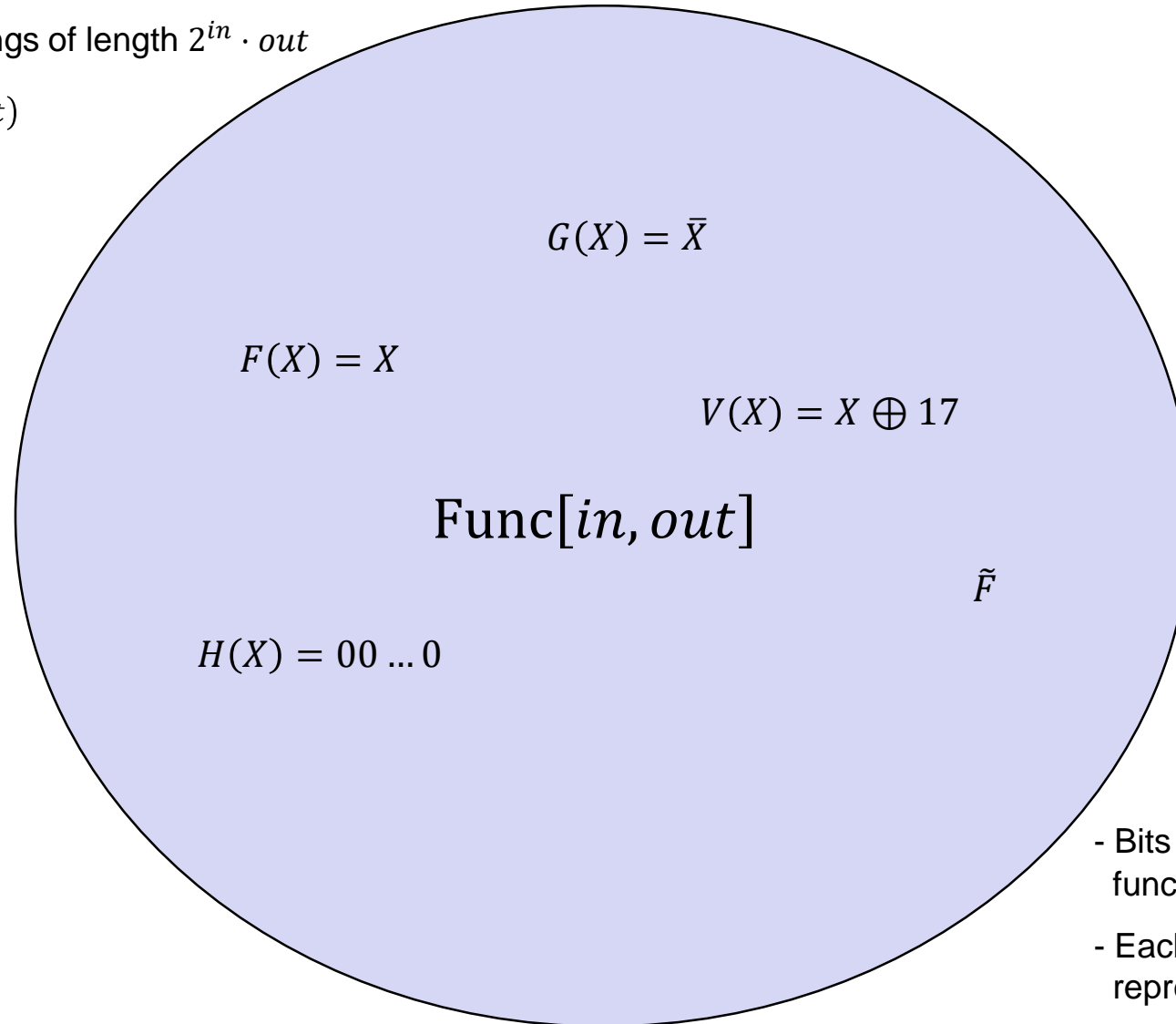
$$= 2^{(2^{in} \cdot out)}$$

Example:

$$|\text{Func}[3,2]| = 2^{2^3 \cdot 2}$$

$$= 2^{16}$$

$$= 65536$$



- Bits needed to specify *one* function: $2^{in} \cdot out$
- Each *distinct* bitstring of length $2^{in} \cdot out$ represents a *unique* function

PRF security – an equivalent view

$$|\text{Func}[in, out]| = \# \text{ bitstrings of length } 2^{in} \cdot out$$

$$= 2^{(2^{in} \cdot out)}$$

Example:

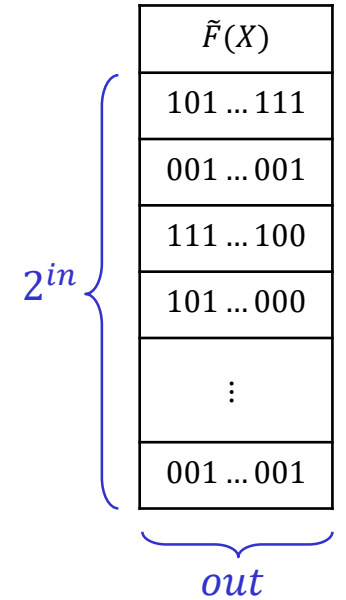
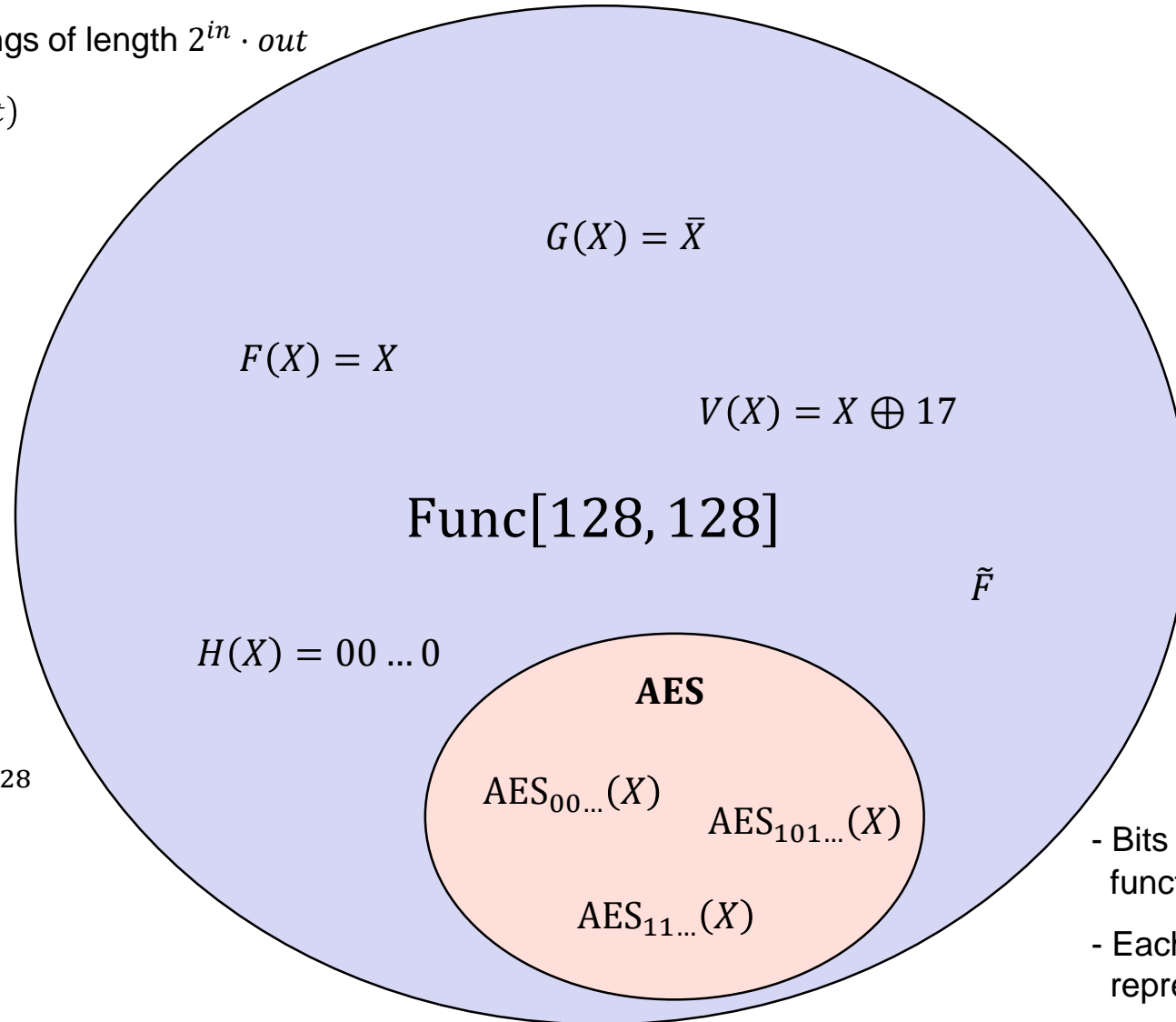
$$|\text{Func}[3,2]| = 2^{2^3 \cdot 2}$$

$$= 2^{16}$$

$$= 65536$$

$$|\text{Func}[128,128]| = 2^{2^{128} \cdot 128}$$

$$|\text{AES}| = 2^{128}$$



- Bits needed to specify *one* function: $2^{in} \cdot out$
- Each *distinct* bitstring of length $2^{in} \cdot out$ represents a *unique* function

PRF security – an equivalent view

$$|\text{Func}[in, out]| = \# \text{ bitstrings of length } 2^{in} \cdot out$$

$$= 2^{(2^{in} \cdot out)}$$

Example:

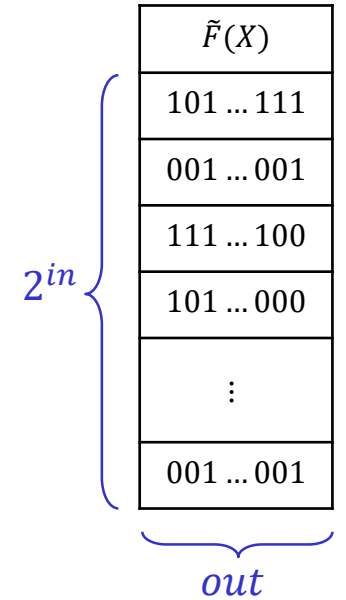
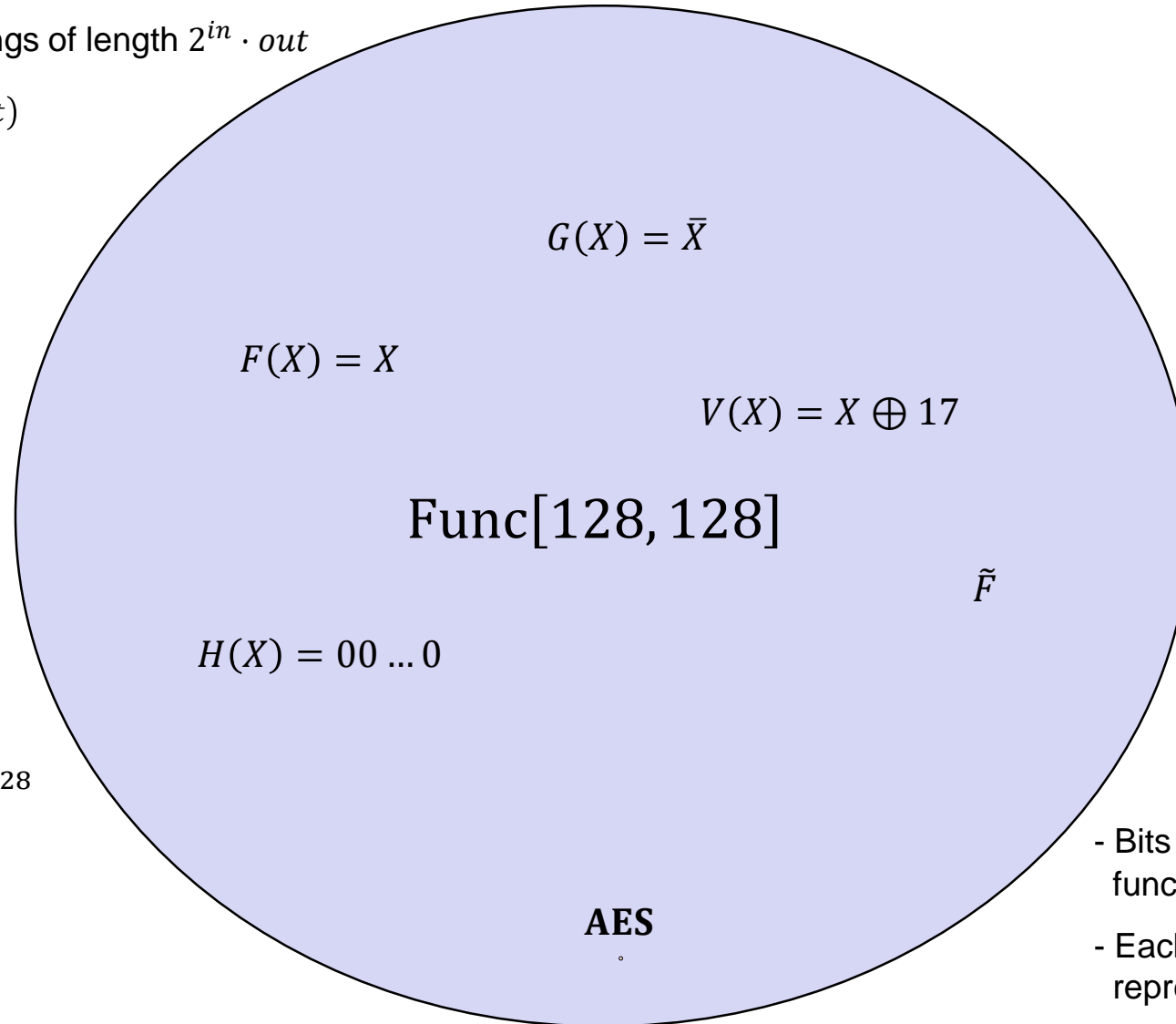
$$|\text{Func}[3,2]| = 2^{2^3 \cdot 2}$$

$$= 2^{16}$$

$$= 65536$$

$$|\text{Func}[128,128]| = 2^{2^{128} \cdot 128}$$

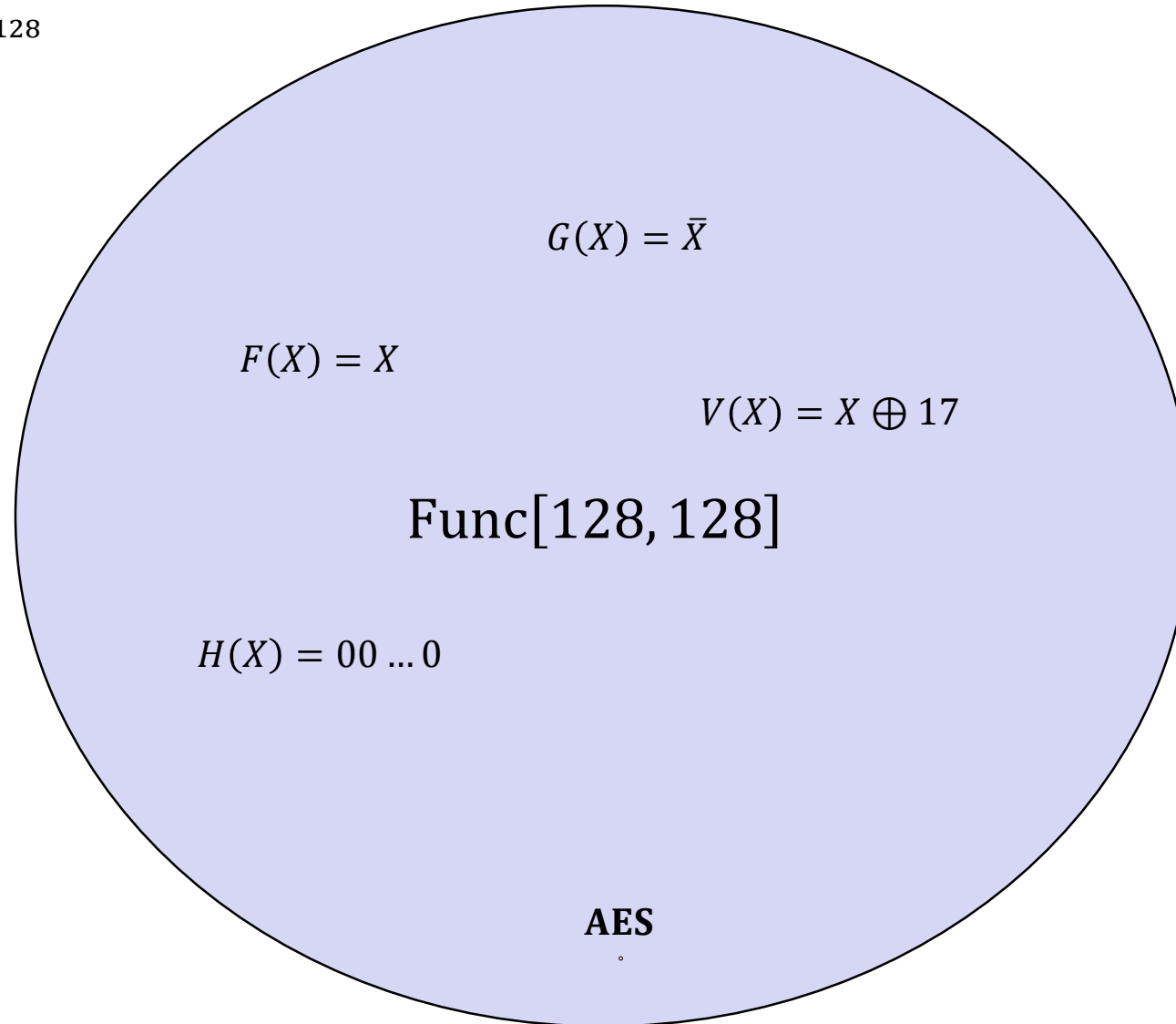
$$|\text{AES}| = 2^{128}$$



- Bits needed to specify *one* function: $2^{in} \cdot out$
- Each *distinct* bitstring of length $2^{in} \cdot out$ represents a *unique* function

Block ciphers – security

$$|\text{Func}[128,128]| = 2^{2^{128} \cdot 128}$$

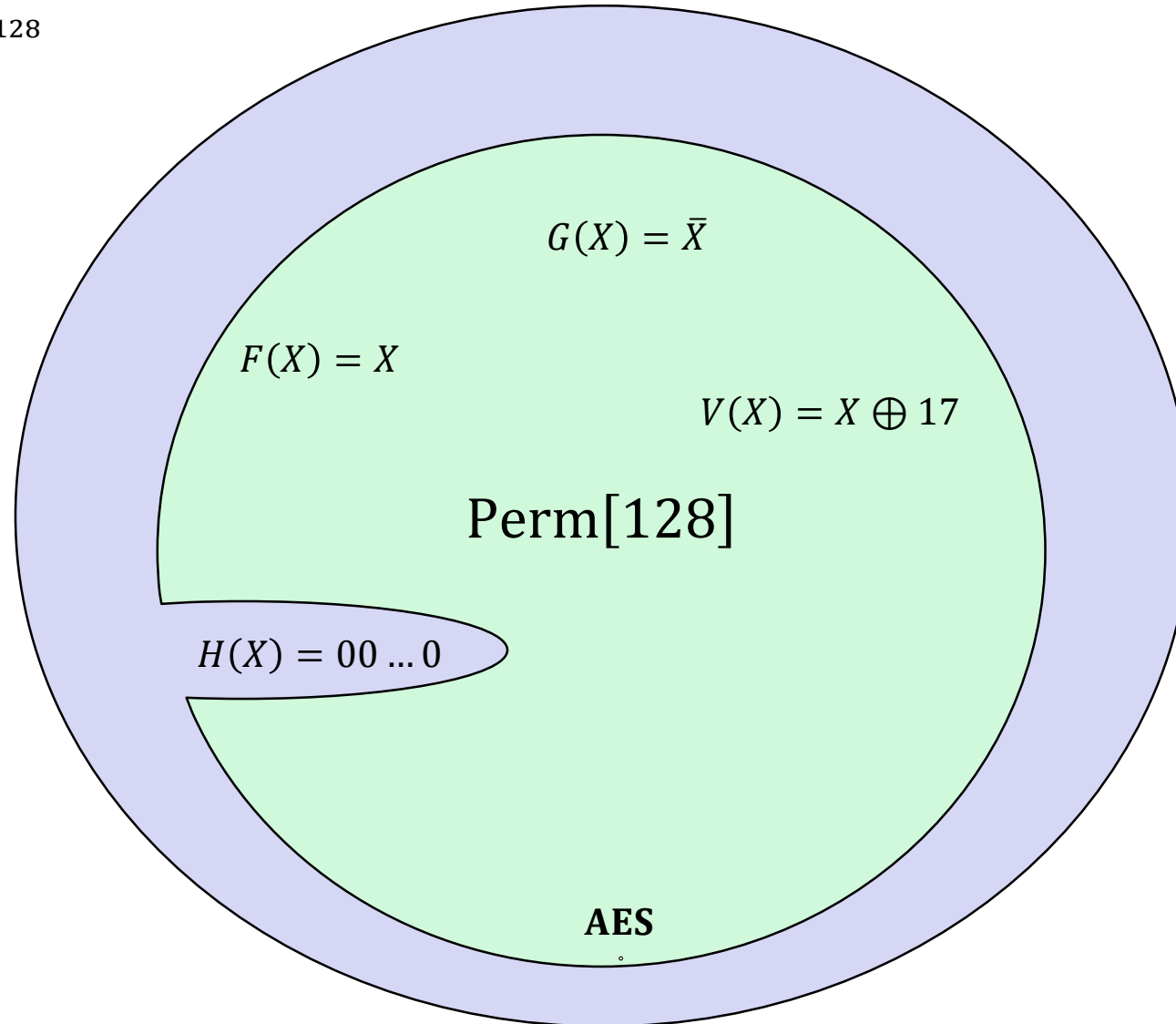


$$|\text{AES}| = 2^{128}$$

Block ciphers – security

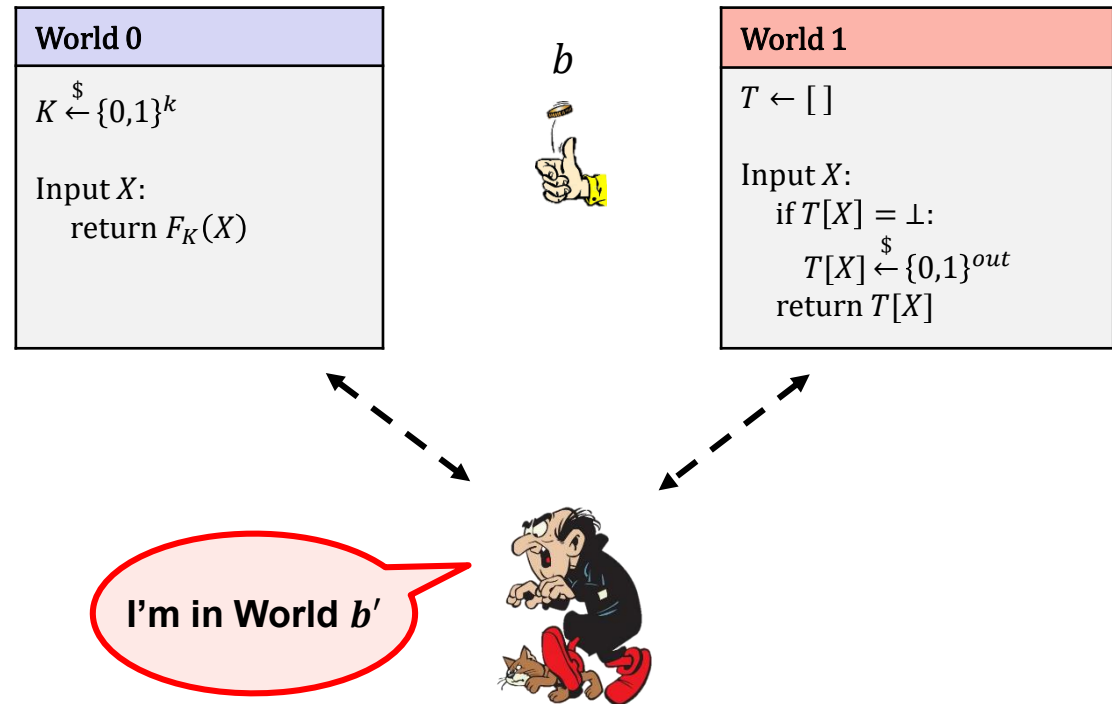
$$|\text{Func}[128,128]| = 2^{2^{128} \cdot 128}$$

$$|\text{Perm}[128]| = 2^{128!}$$




$$|\text{AES}| = 2^{128}$$

PRF – security



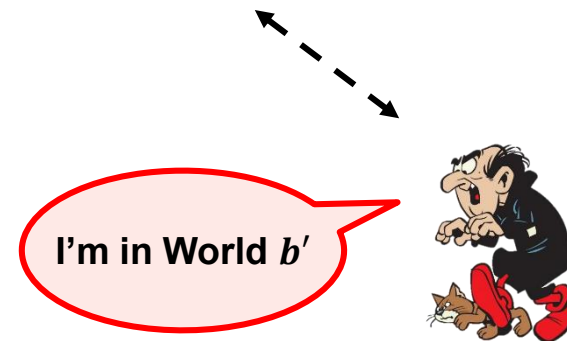
PRF – security; equivalent view

$\text{Exp}_F^{\text{prf}}(A)$	$A =$ 
1. $b \xleftarrow{\$} \{0,1\}$	
2. $K \xleftarrow{\$} \{0,1\}^k$	
3. $F_0 \leftarrow F_K$ // real F	
4. $F_1 \xleftarrow{\$} \text{Func}[in, out]$ // random \tilde{F}	
5. $b' \leftarrow A^{F_b(\cdot)}$	
6. return $b' \stackrel{?}{=} b$	

World 0
$K \xleftarrow{\$} \{0,1\}^k$
Input X : return $F_K(X)$




World 1
$\tilde{F} \xleftarrow{\$} \text{Func}[in, out]$
Input X : return $\tilde{F}(X)$



Definition: The **PRF-advantage** of an adversary A is

$$\text{Adv}_F^{\text{prf}}(A) = \left| 2 \cdot \Pr \left[\text{Exp}_F^{\text{prf}}(A) \Rightarrow \text{true} \right] - 1 \right|$$

PRF PRP – security; equivalent view

Exp_F^{prfprp}(A) A = 

1. $b \xleftarrow{\$} \{0,1\}$
2. $K \xleftarrow{\$} \{0,1\}^k$
3. $F_0 \leftarrow F_K$ **Perm[n]** // real F
4. $F_1 \xleftarrow{\$} \text{Func}[in, out]$ // random \tilde{F}
5. $b' \leftarrow A^{F_b(\cdot)}$
6. **return** $b' \stackrel{?}{=} b$

World 1

$K \xleftarrow{\$} \{0,1\}^k$

Input X :
return $F_K(X)$



World 0

$\tilde{F} \xleftarrow{\$} \text{Func}[in, out]$ **Perm[n]**

Input X :
return $\tilde{F}(X)$

I'm in World b'



Definition: The **PRP-advantage** of an adversary A is

$$\text{Adv}_F^{\text{prp}}(A) = |2 \cdot \Pr[\text{Exp}_F^{\text{prp}}(A) \Rightarrow \text{true}] - 1|$$

Summary

- Block ciphers are very important **primitives** (building blocks) – not encryption schemes!
- Correct abstraction: block ciphers = PRPs
- Right security notion for PRFs/PRPs:
indistinguishability from random function/permutation
- Concrete block cipher design: DES, AES, ...