
Lecture 2 – Block ciphers, PRFs/PRPs, AES

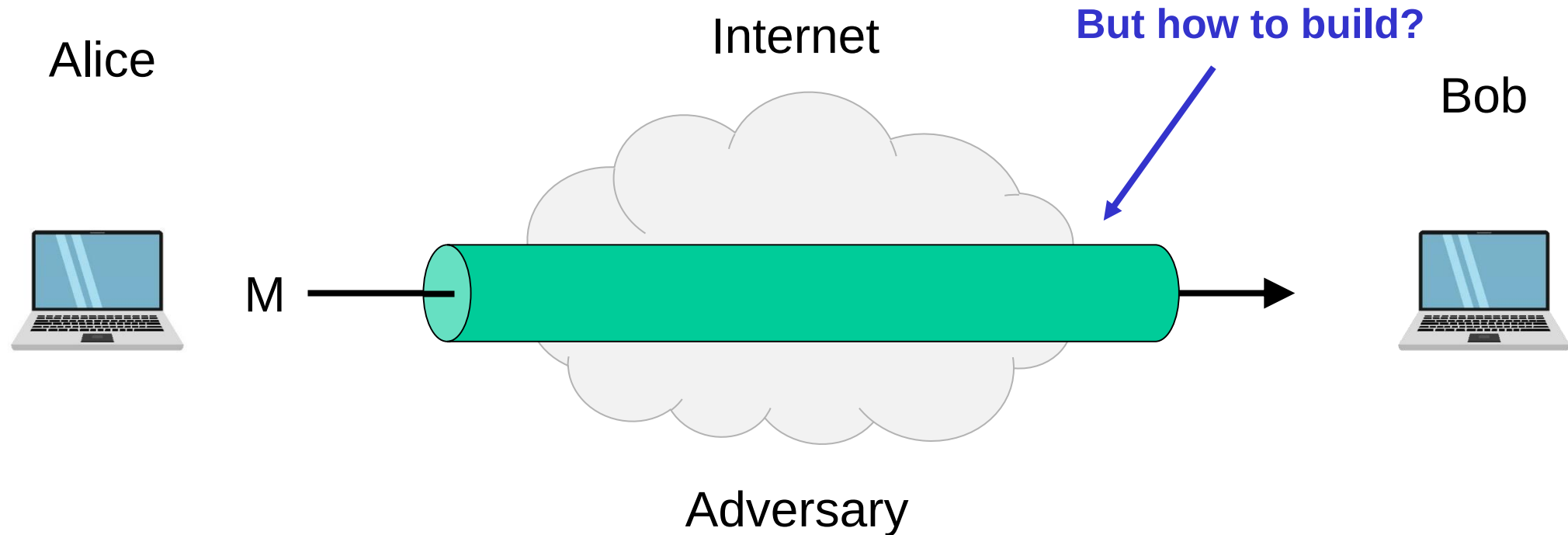
TEK4500

04.09.2024

Håkon Jacobsen

hakon.jacobsen@its.uio.no

Ideal solution: secure channels



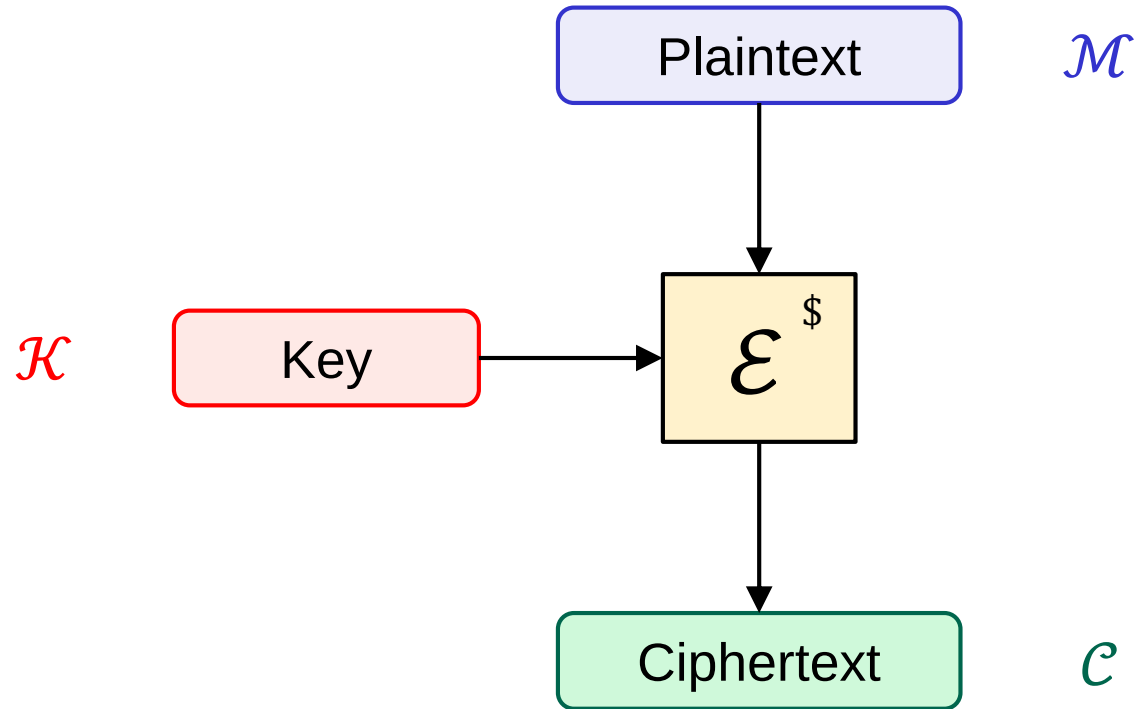
Security goals:

- **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 ✓

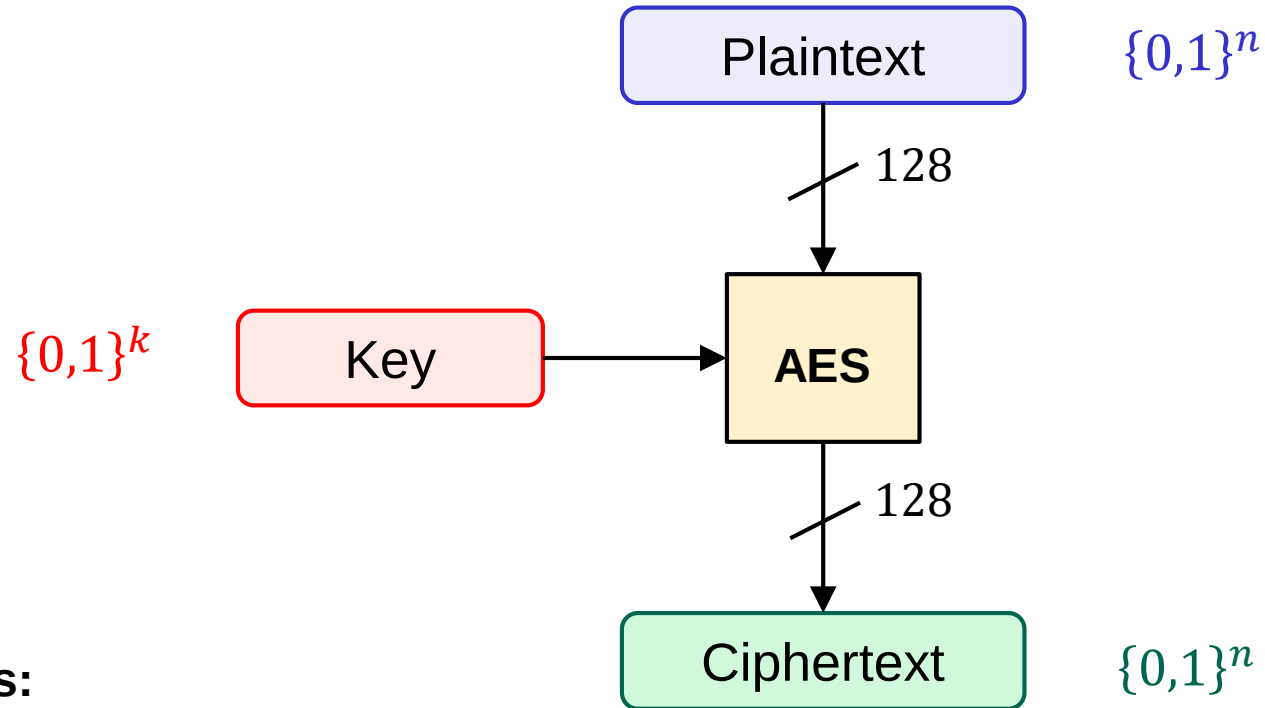
Basic goals of cryptography

	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:

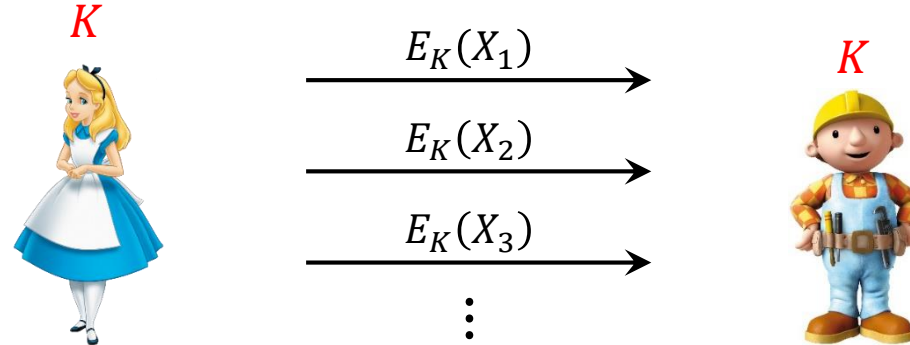
AES-128: $k = 128$, $n = 128$

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

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

Block cipher applications

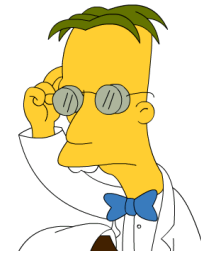
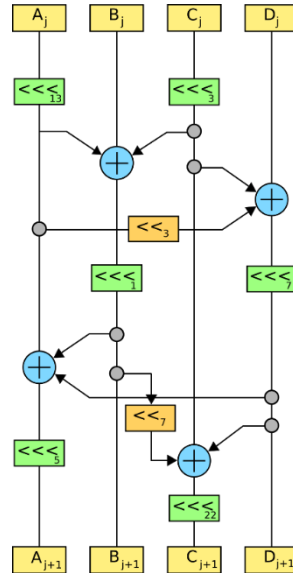
- Encryption of messages of 128 bits (block length)



- **However:** we 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
 - Pseudorandom generators (PRNGs)
 - Key derivation functions
 - One-time tokens



Constructing block ciphers

Principles for designing block ciphers

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

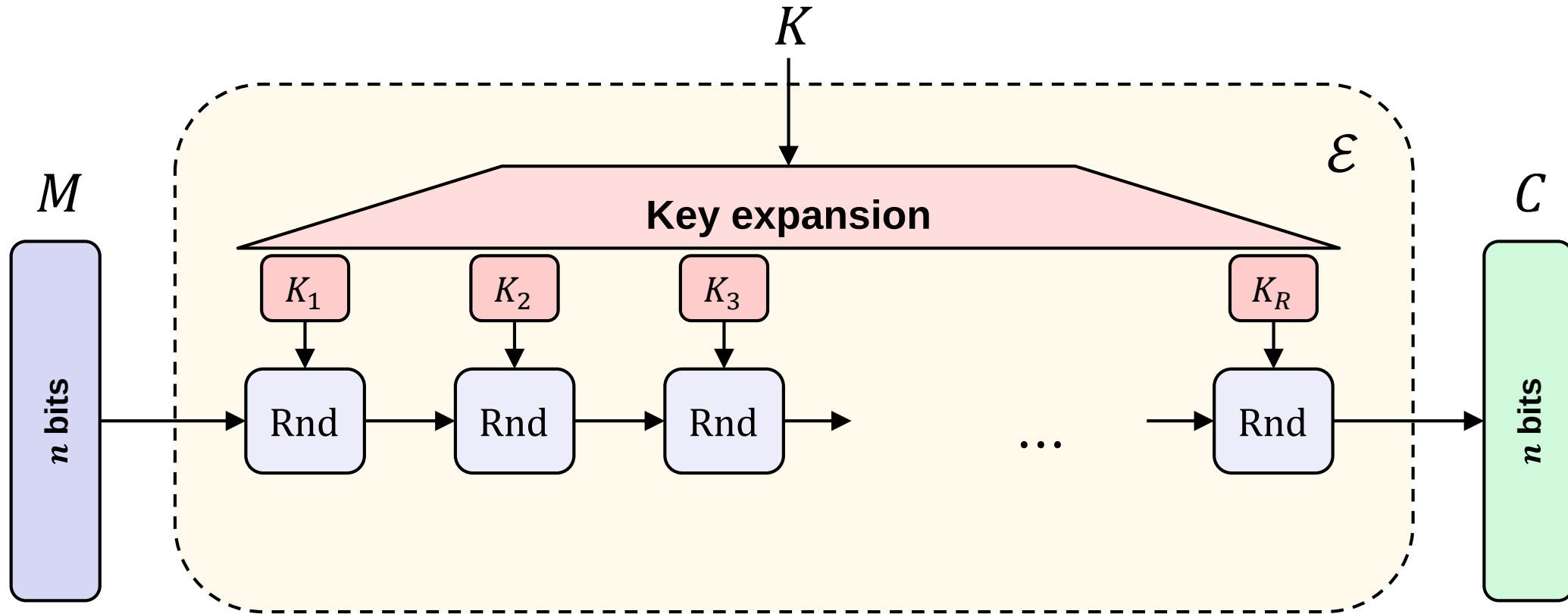
- **Diffusion:** plaintext spread over large parts of the ciphertext
- **Confusion:** a complex relation between plaintext, key and ciphertext

Example:

$$F(K, M) = K \oplus M \quad \text{(one-time pad)}$$

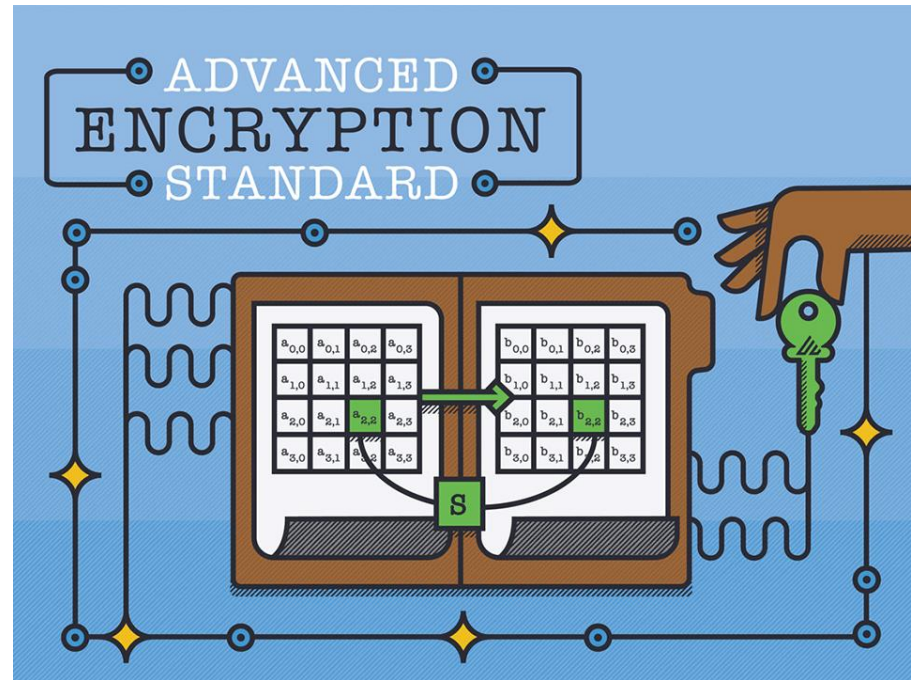
Is this a good block cipher?

Block ciphers



Rnd = **round function** (iterated R times)

AES-128/192/256: $R = 10/12/14$



Advanced Encryption Standard

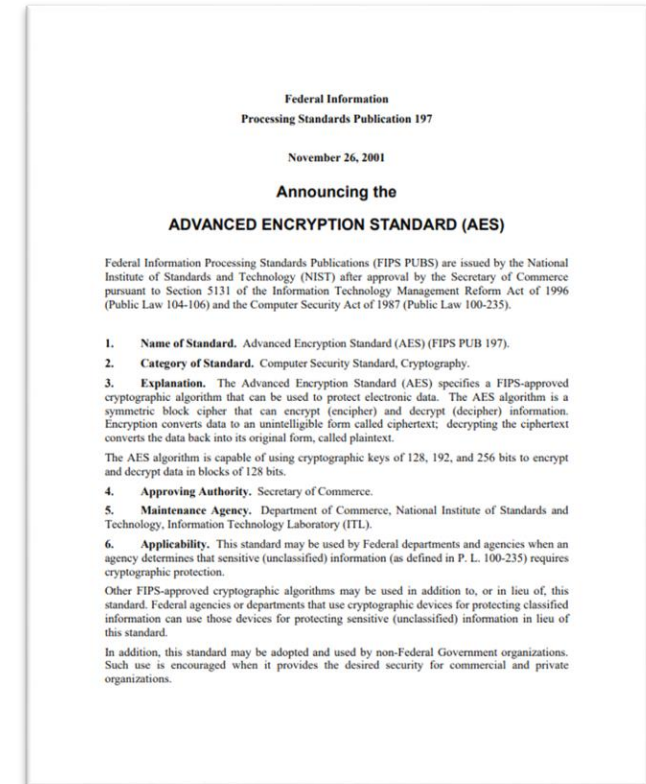
AES – history

- 1970-1990's: Data Encryption Standard (DES)
 - 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:

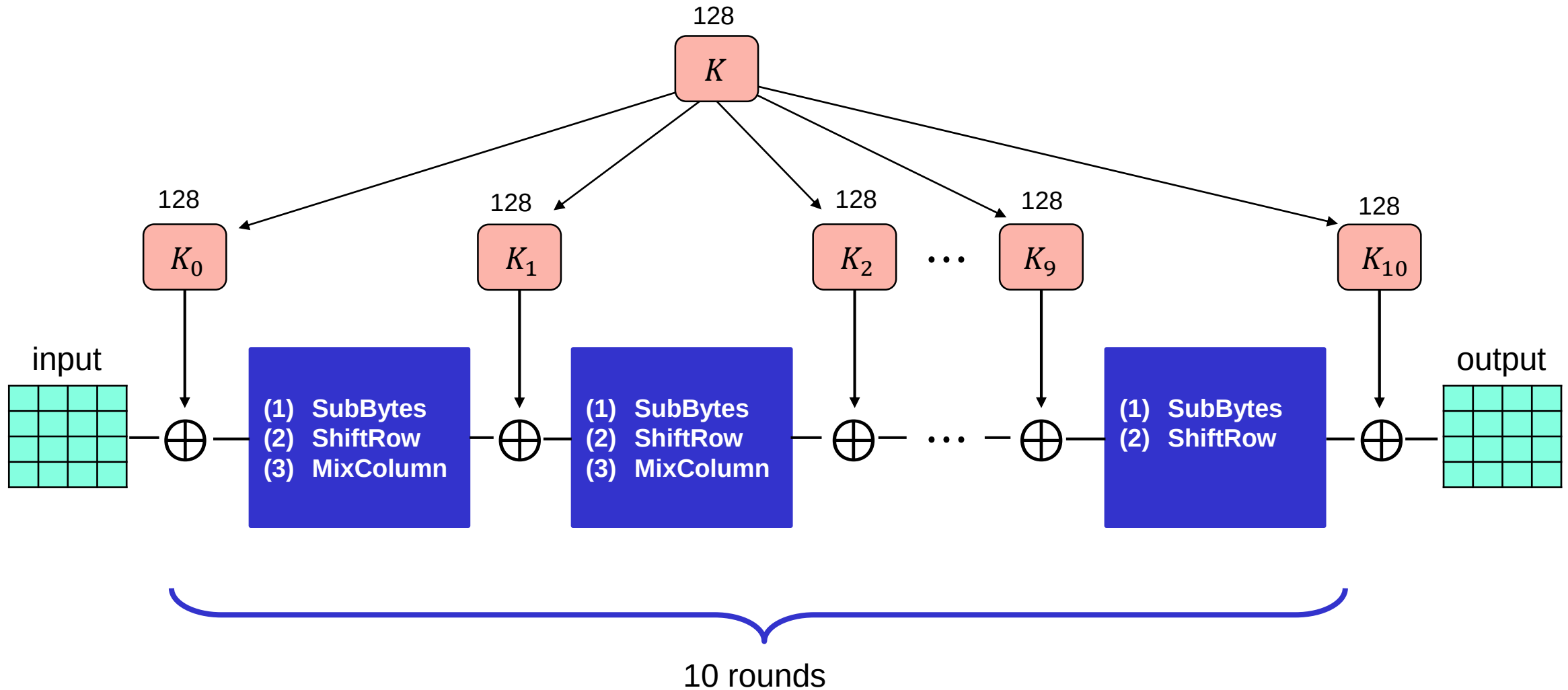
$$\text{AES-128} : \{0,1\}^{128} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$$

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

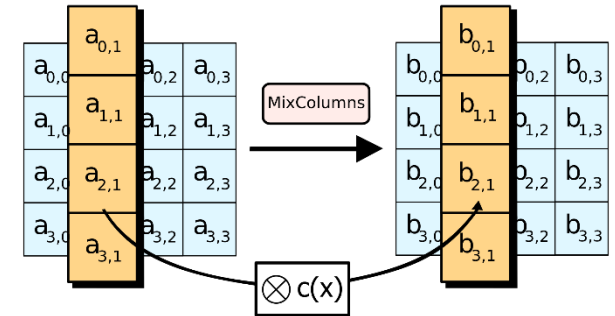
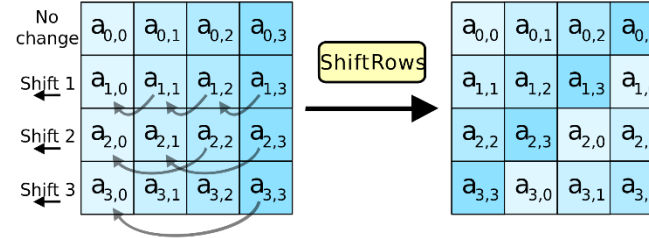
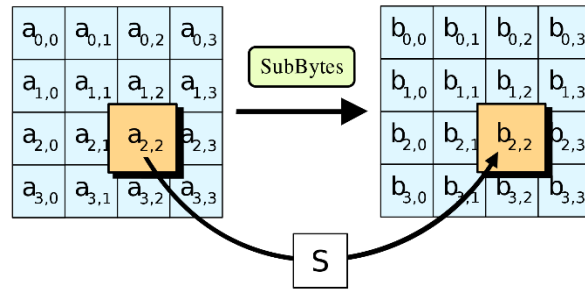
$$\text{AES-256} : \{0,1\}^{256} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$$



AES-128

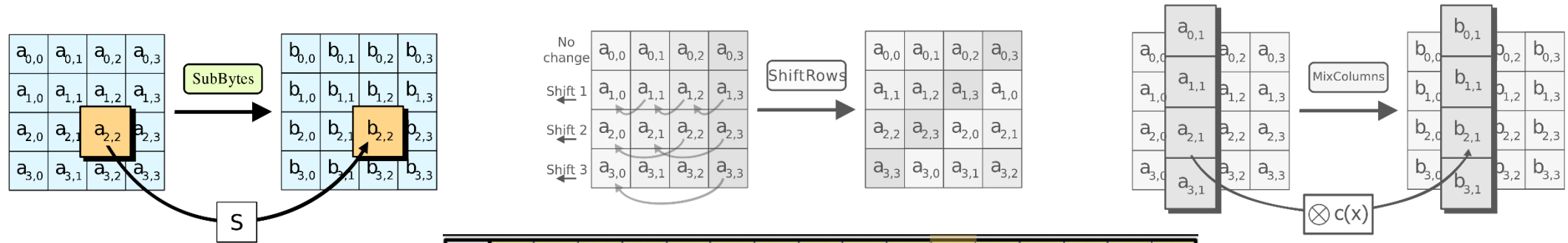


AES round function



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

AES round function - SubBytes

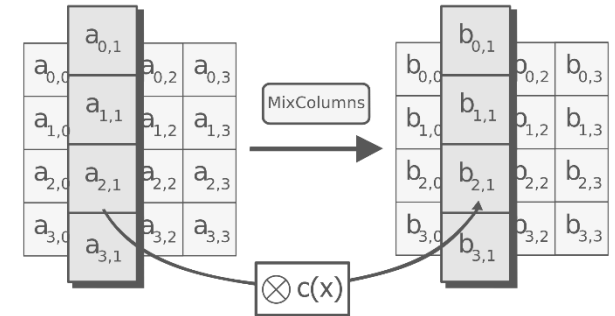
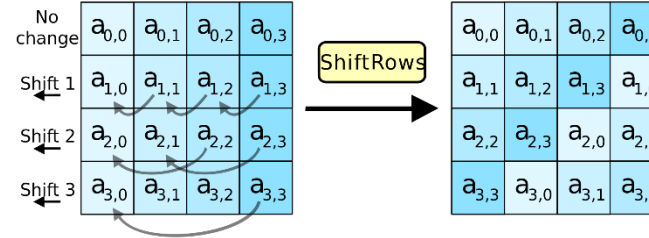
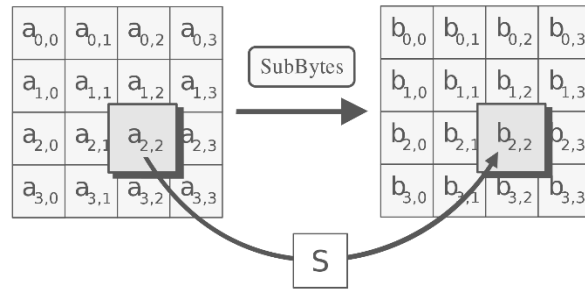


$$a_{2,2} = 8A$$

	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

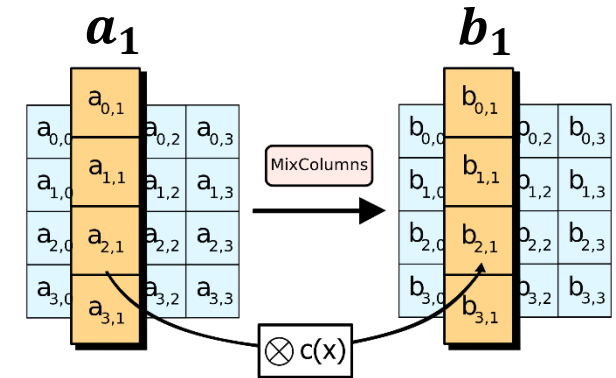
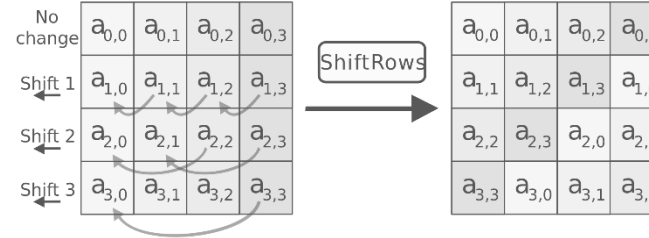
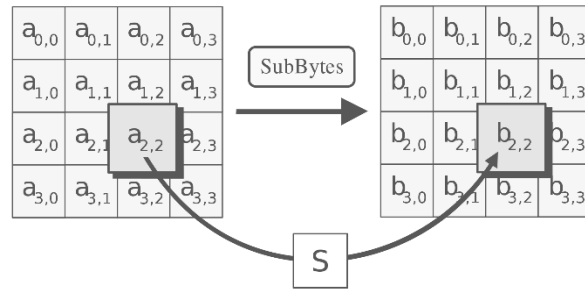
$$b_{2,2} = 7E$$

AES round function - ShiftRows



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

AES round function - MixColumns



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

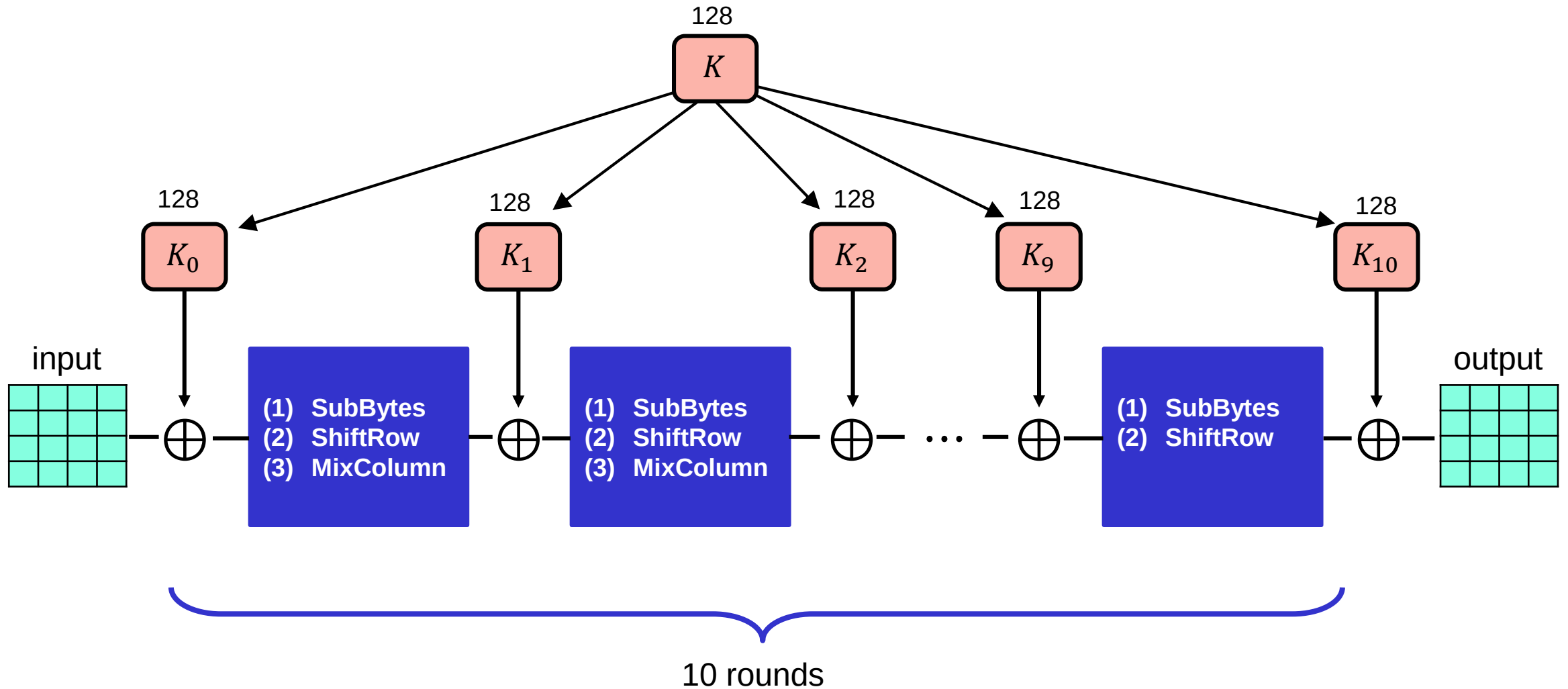
$$Ma_1 = b_1$$

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

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

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

AES-128

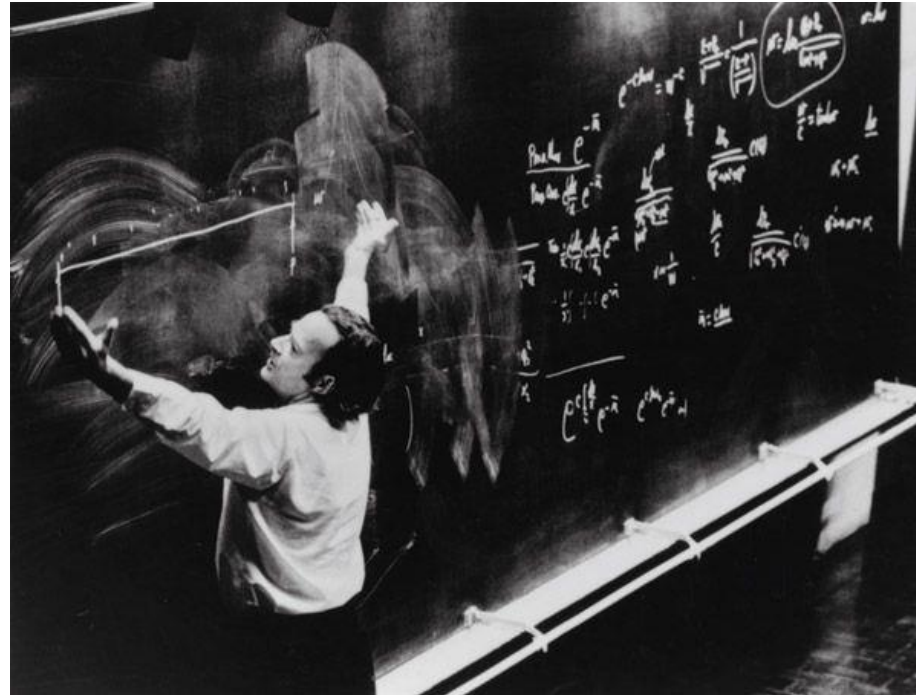


AES performance

- AES is reasonably efficient in software
 - Table-based implementation very fast (but not secure!)
 - Difficult to implement AES both fast *and* securely (in SW)

	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 inversed MixColumns
- Now standard in all modern CPUs



Defining block cipher security

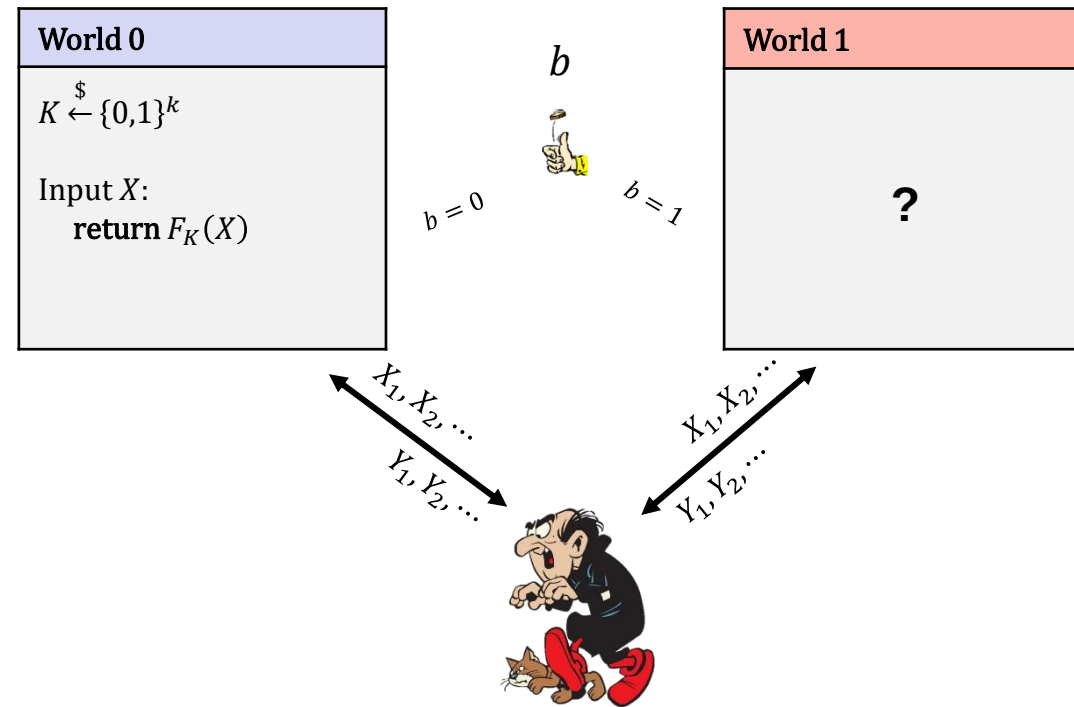
Pseudorandom functions (PRFs)

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 choice of $K \in \{0,1\}^k$ we get a "new" function $F_K : \{0,1\}^{in} \rightarrow \{0,1\}^{out}$ defined by $F_K(X) \stackrel{\text{def}}{=} F(K, X)$

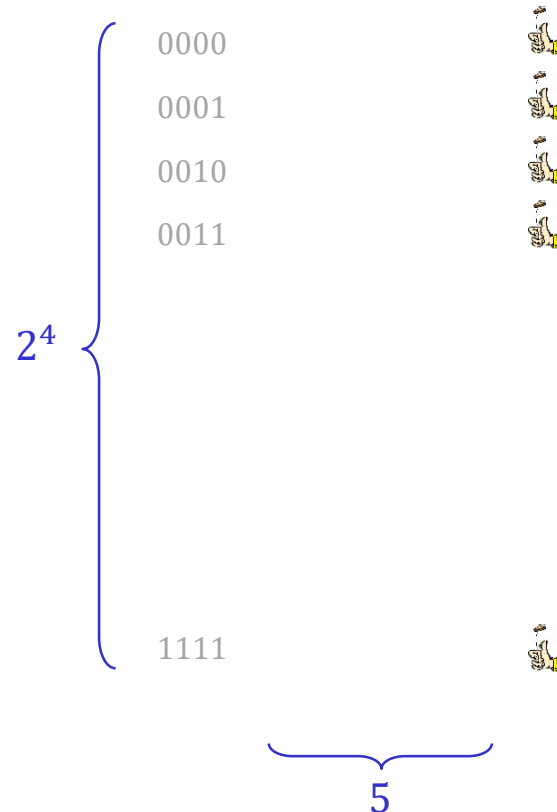
PRF – security



Random functions

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

Example:
 $in = 4, out = 5$



Notation:

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

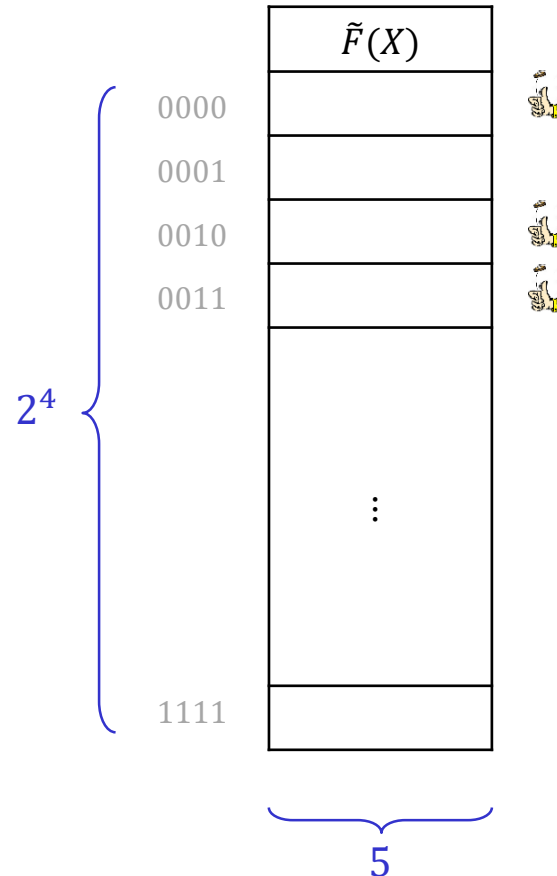
"pick a random function from in bits to out bits"

(by filling out the table with random values)

Random functions – lazy sampling

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

Example:
 $in = 4, out = 5$



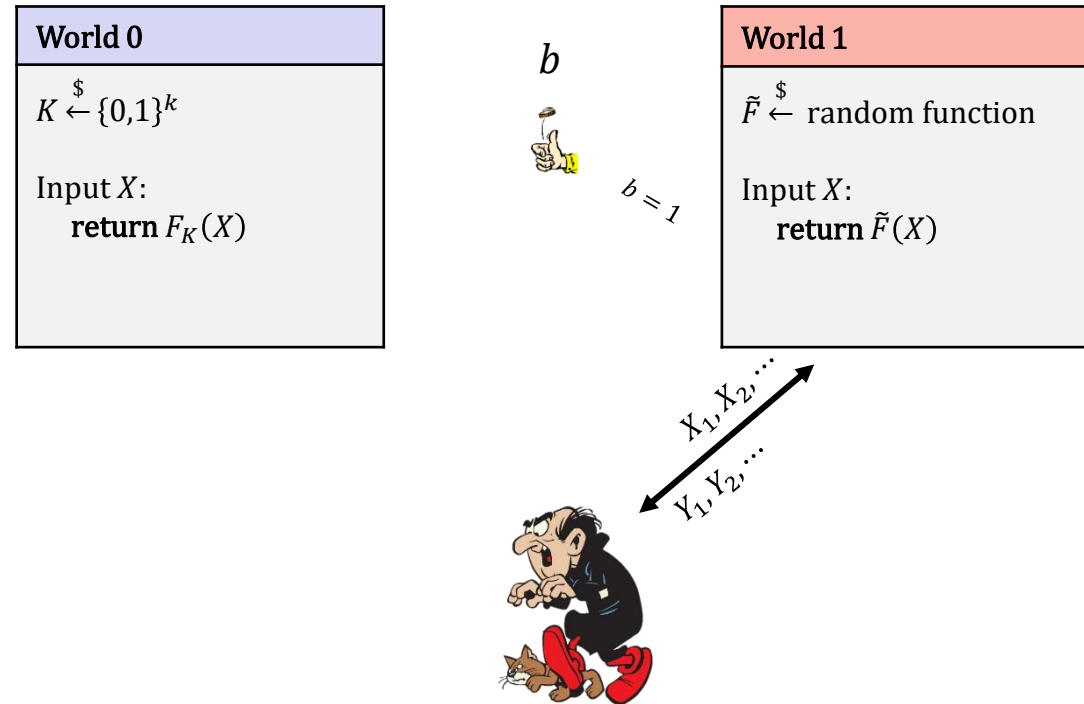
Notation:

$$\tilde{F} \stackrel{\$}{\leftarrow} \text{Func}[in, out]$$

"pick a random function from in bits to out bits"

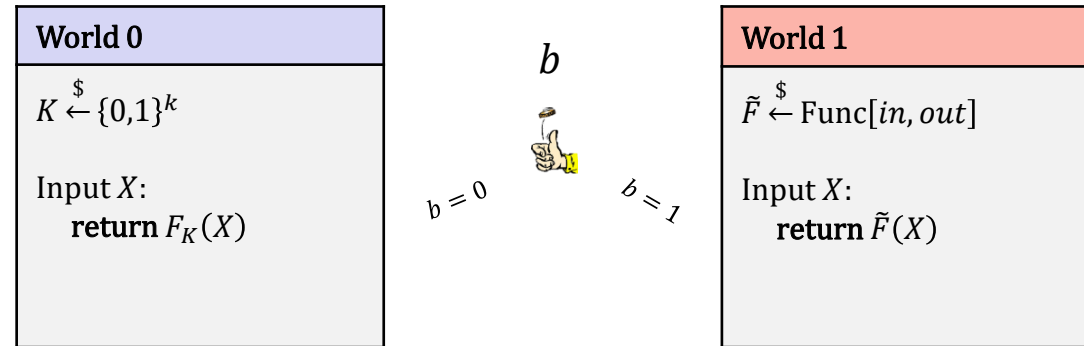
(by filling out the table with random values)

PRF – security



PRF – security

$\text{Exp}_F^{\text{prf}}(A)$	$A =$ 
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 $b' \stackrel{?}{=} b$	



$\text{Adv}_F^{\text{prf}}(A) \approx 1/2 \Rightarrow$ 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

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

PRF – security

$\text{Exp}_F^{\text{prf}}(A)$	$A =$ 
$b \xleftarrow{\\$} \{0,1\}$ $K \xleftarrow{\\$} \{0,1\}^k$ $F_0 \leftarrow F_K$ // real F $F_1 \xleftarrow{\\$} \text{Func}[in, out]$ b'	

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

b

 $b=0$ $b=1$

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

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

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

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

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

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

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 adversary's:
 - Available resources: running time, memory usage, number of oracle calls (calls to F or $\tilde{F}$), ...
 - *Not* on the adversary's strategy! (since we consider *all* possible A)
- What does *small* and *reasonable* mean?
 - **Example:** **128-bit** security:

$$\text{Adv}_F^{\text{prf}}(A) \leq \frac{2^{80}}{2^{128}} = 2^{-48} \approx 0.0000000000000001$$

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

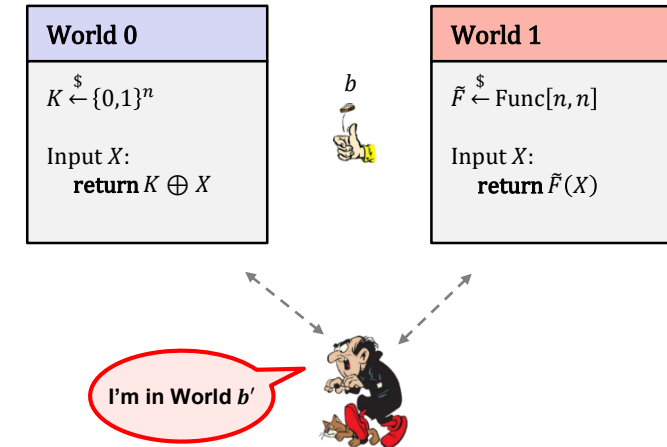
- **Example:** a PRF is *insecure* if we can come up with *one* efficient adversary with good advantage

Using the PRF definition

- Let $F(K, X) = K \oplus X$ (i.e., the OTP)
- Claim:** F is *not* a secure PRF

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

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



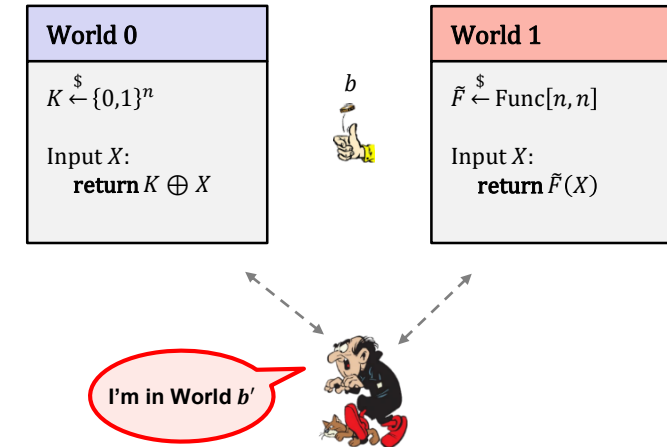
Definition: The **PRF-advantage** of an adversary A is

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

Using the PRF definition

- Let $F(K, X) = K \oplus X$ (i.e., the OTP)
- Claim:** F is *not* a secure PRF

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



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

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

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

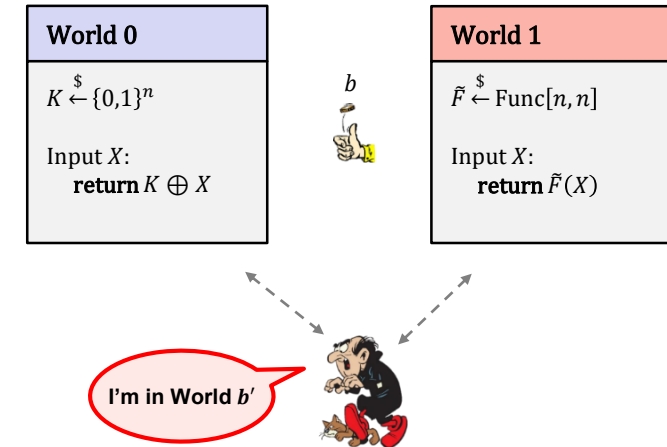
Using the PRF definition

- Let $F(K, X) = K \oplus X$ (i.e., the OTP)
- Claim:** F is *not* a secure PRF

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

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

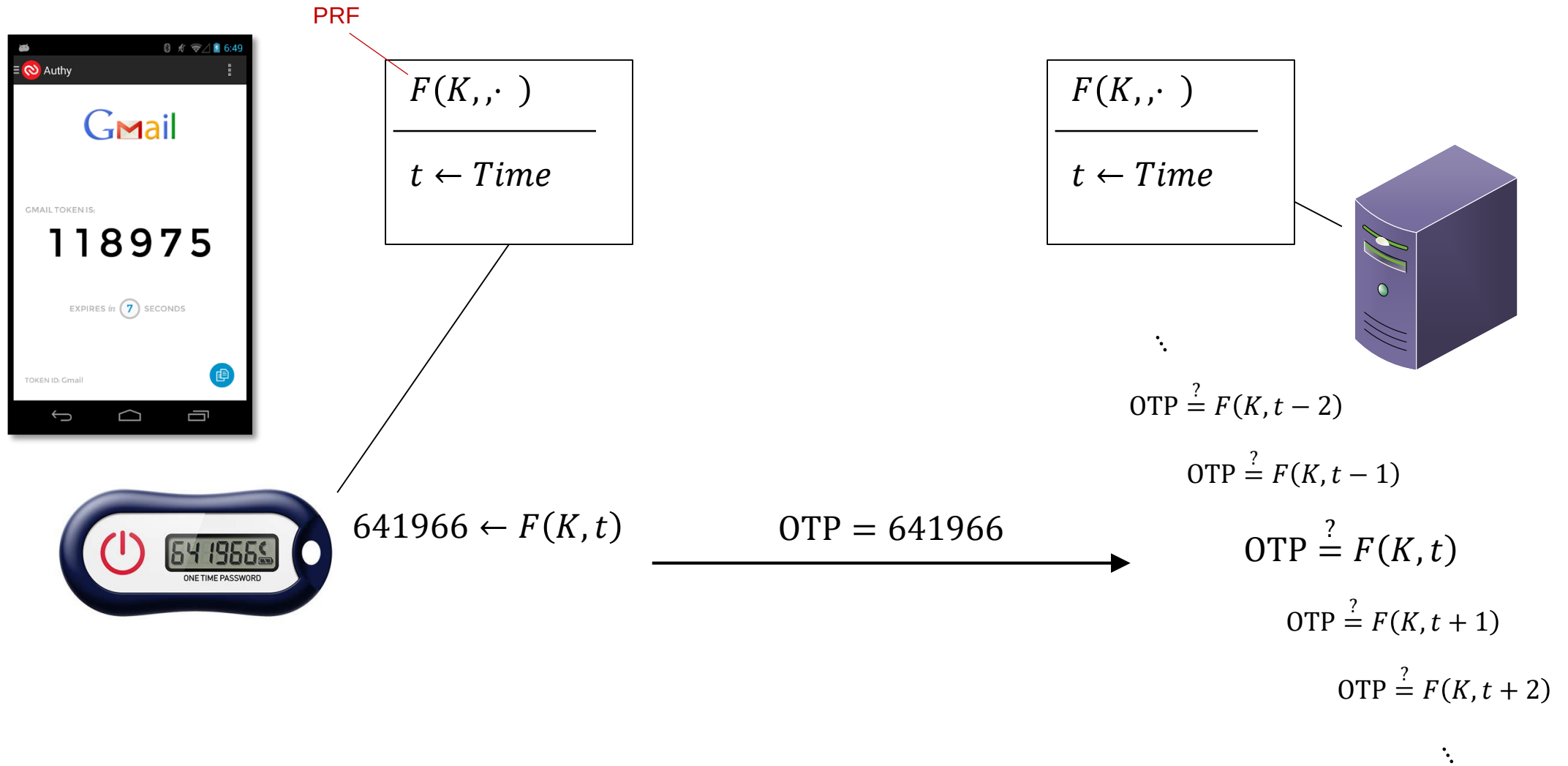
$$\mathbf{Adv}_F^{\text{prf}}(A) = 2 \cdot \left| \Pr[b' = b] - \frac{1}{2} \right| = 2 \cdot \left| \left(1 - \frac{1}{2^{n+1}} \right) - \frac{1}{2} \right| = 1 - 2^{-n} \approx 1$$



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

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

PRF applications: one-time tokens



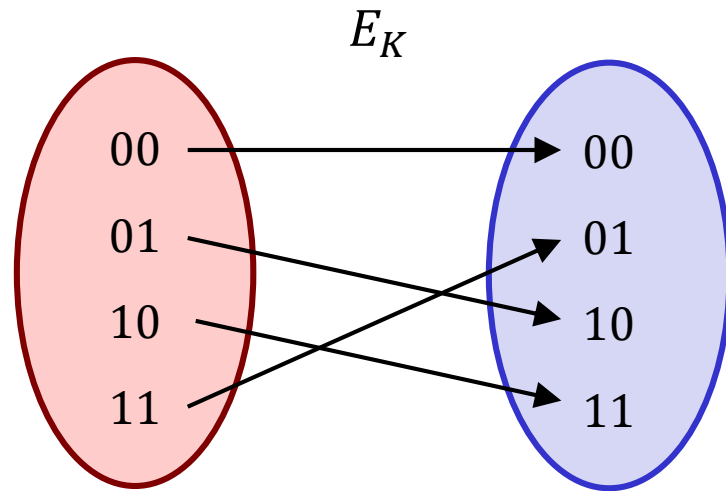
Block ciphers, a.k.a. pseudorandom *permutations* (PRP)

Definition: A pseudorandom function (PRF) is a function

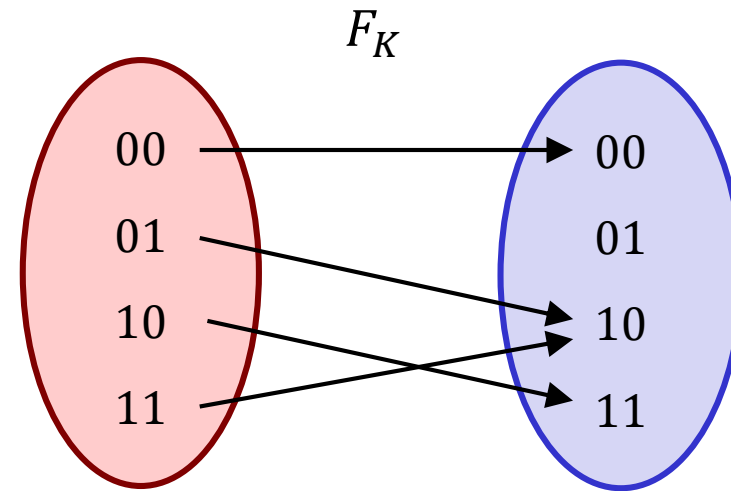
$$F : \{0,1\}^k \times \{0,1\}^{in} \rightarrow \{0,1\}^{out}$$

- **Definition:** A block cipher, or pseudorandom *permutation* (PRP), is a PRF where:
Special case:
 - $in = out = n$
 - $F_K : \{0,1\}^n \rightarrow \{0,1\}^n$ is *invertible* for every $K \in \{0,1\}^k$

Permutations vs. functions

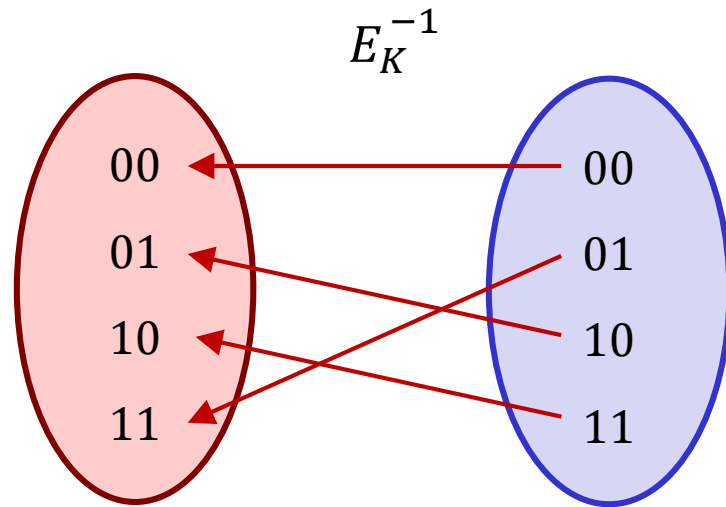


Permutation

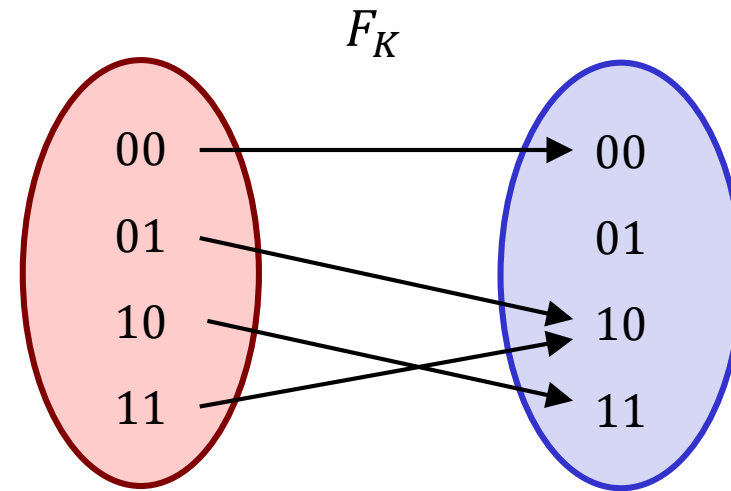


Not a permutation

Permutations vs. functions



Permutation



Not a permutation

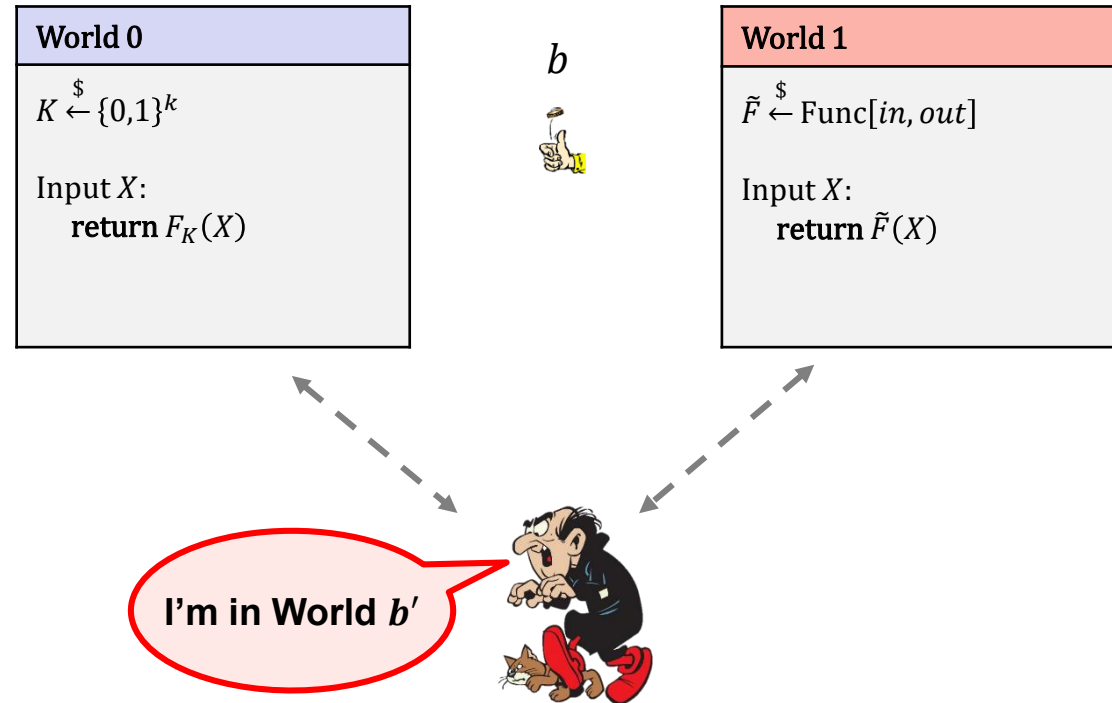
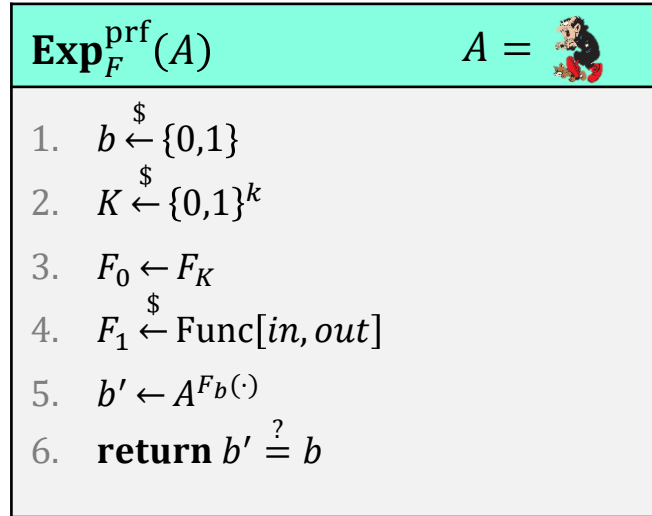
Important:

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

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

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

PRF – security

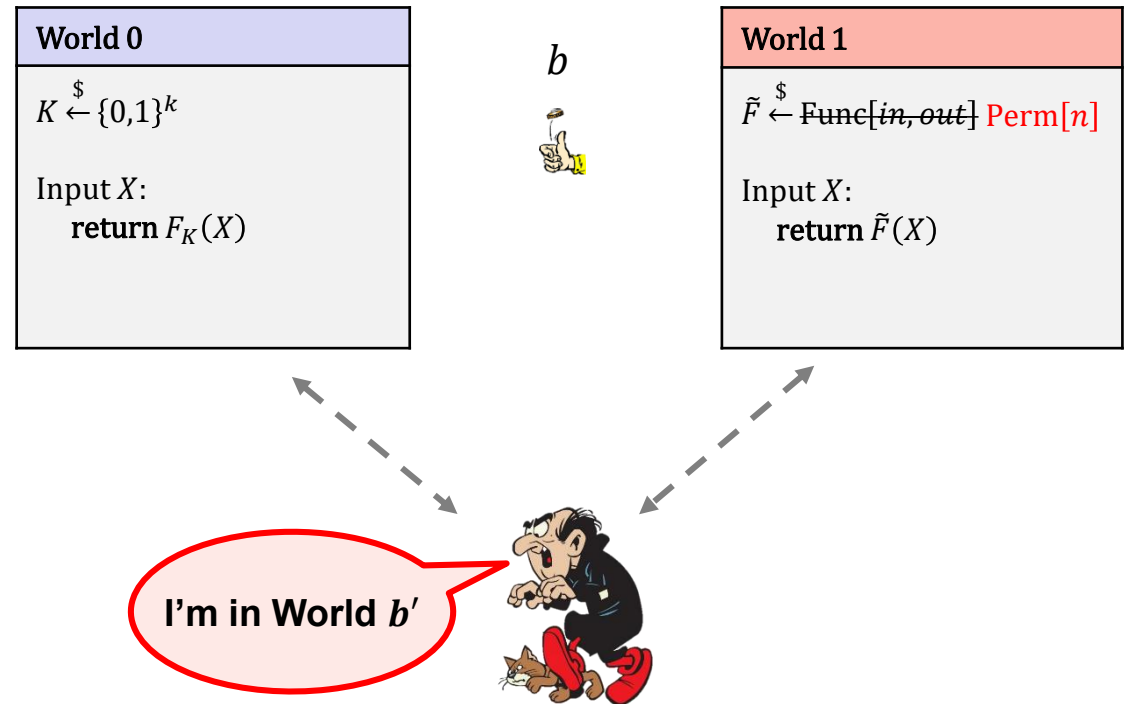


Definition: The **PRF-advantage** of an adversary A is

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

PRF PRP – security

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



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

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

PRP security $\Rightarrow$ PRF security

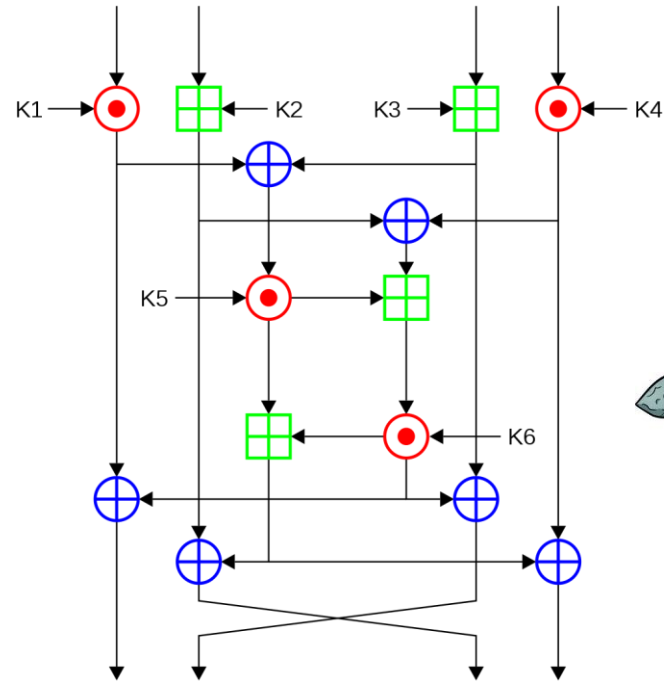
PRP/PRF Switching Lemma: Let E be a PRP. For all A making exactly q oracle queries:

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

Proof sketch:

all outputs from PRF oracle are *distinct*

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



Attacking block ciphers

Attacks on block ciphers

- **Brute force attacks:** search through every possible key in the key space
 - Generic: works for all block ciphers
 - Not practical for large key spaces
- **Specialized 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

Summary

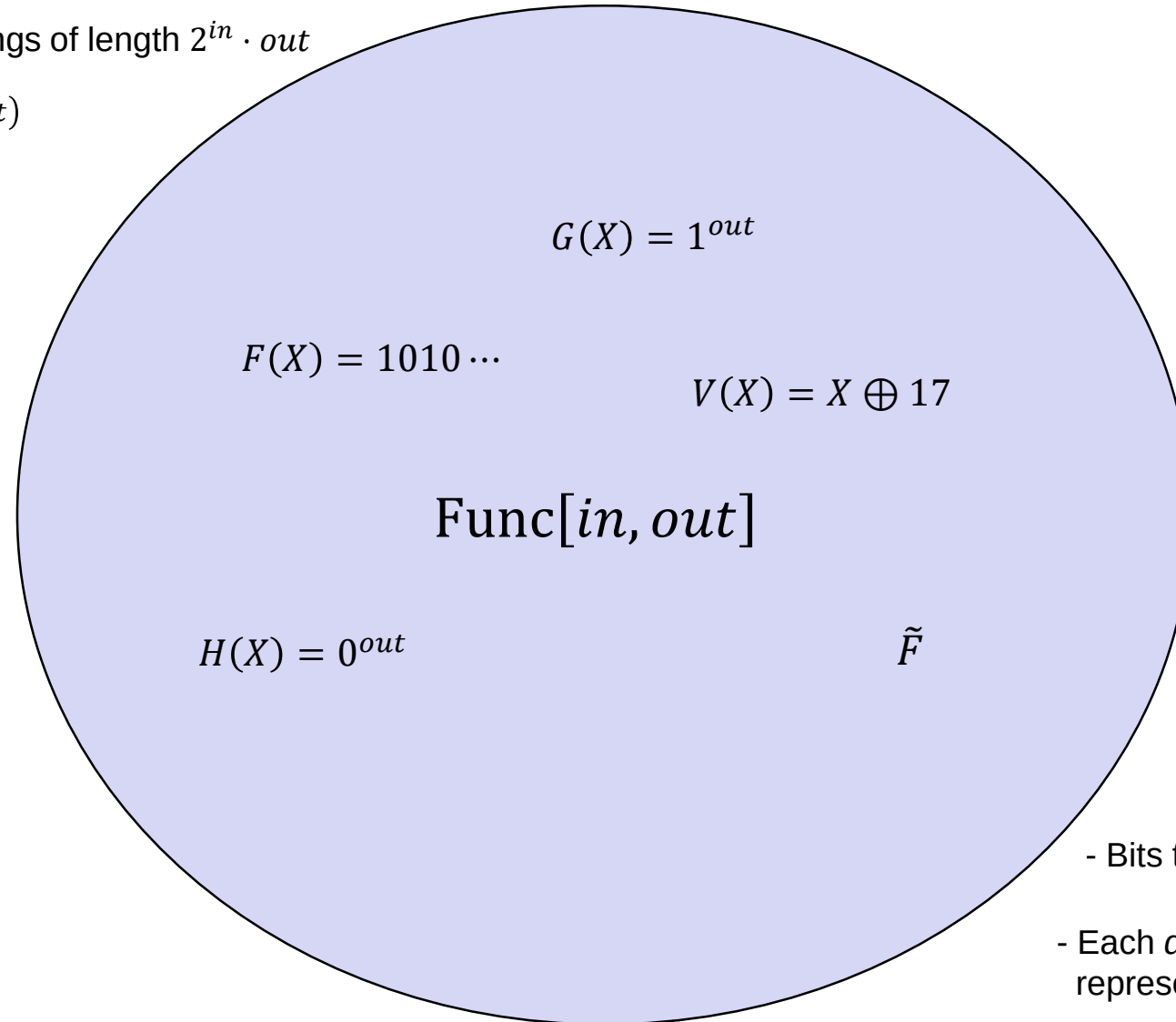
- Block ciphers are very important **primitives** (building blocks) – not encryption schemes!
- Correct abstraction: block ciphers = PRPs
- Right security notion for PRFs/PRPs:
indistinguishability from random function/permutation
- Concrete block cipher designs: DES, AES, ...

PRF security – an equivalent view

$$\begin{aligned} |\text{Func}[in, out]| &= \# \text{ bitstrings of length } 2^{in} \cdot out \\ &= 2^{(2^{in} \cdot out)} \end{aligned}$$

Example:

$$\begin{aligned} |\text{Func}[3, 2]| &= 2^{2^3 \cdot 2} \\ &= 2^{16} \\ &= 65536 \end{aligned}$$



	$\tilde{F}(X)$
2^{in}	101 ... 111
	001 ... 001
	111 ... 100
	101 ... 000
	$\vdots$
	001 ... 001
	out

- Bits 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:

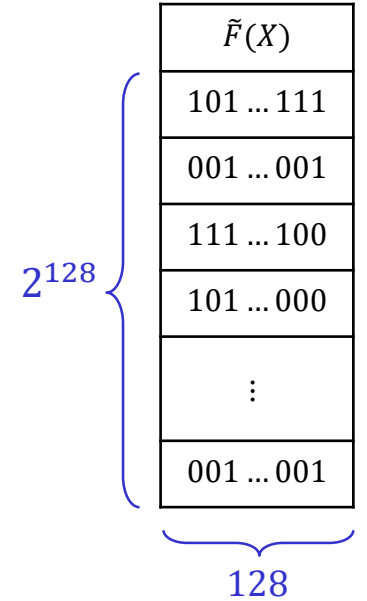
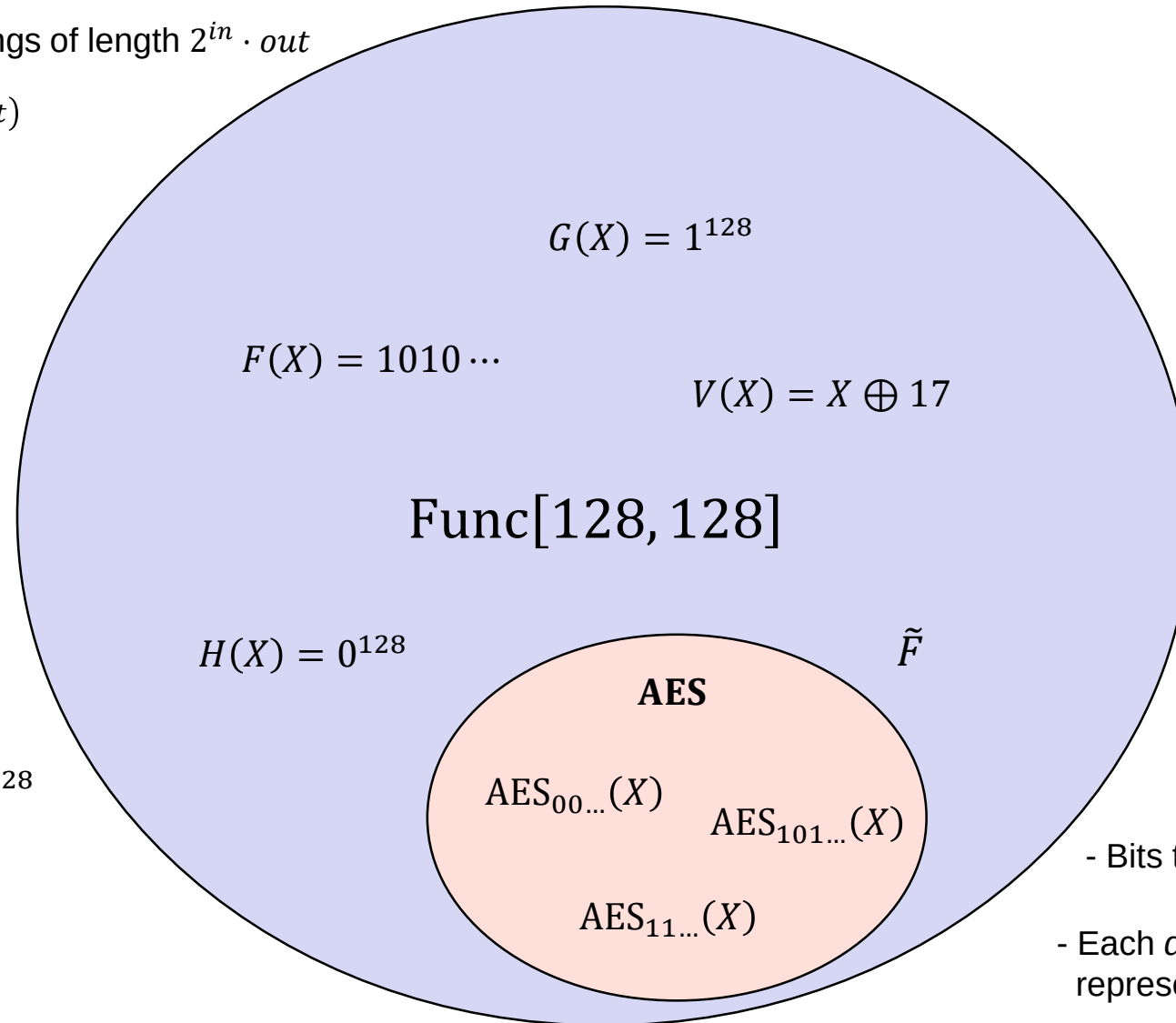
$$|\text{Func}[3, 2]| = 2^{2^3 \cdot 2}$$

$$= 2^{16}$$

$$= 65536$$

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

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



- Bits 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:

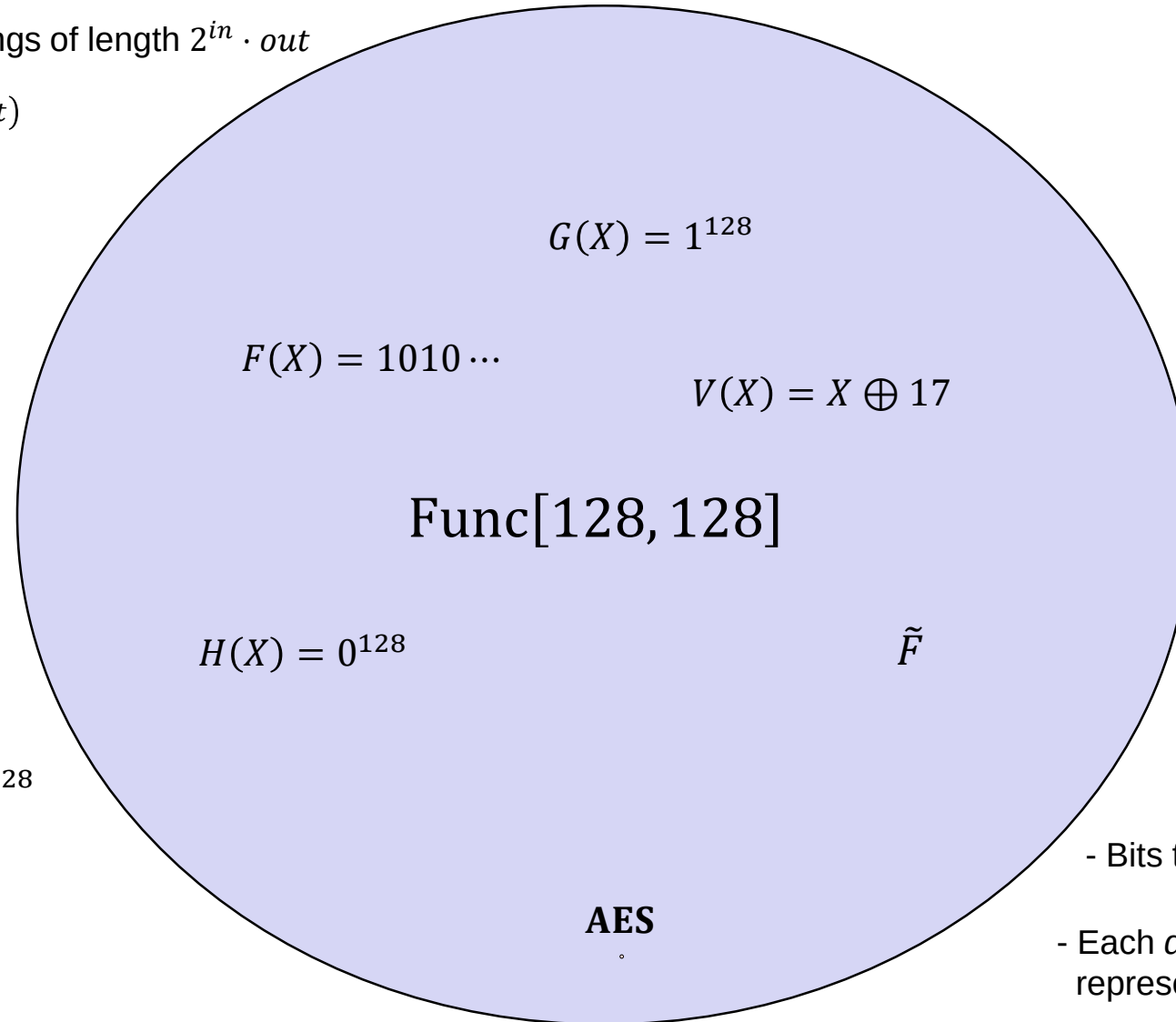
$$|\text{Func}[3,2]| = 2^{2^3 \cdot 2}$$

$$= 2^{16}$$

$$= 65536$$

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

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



2^{128} {

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

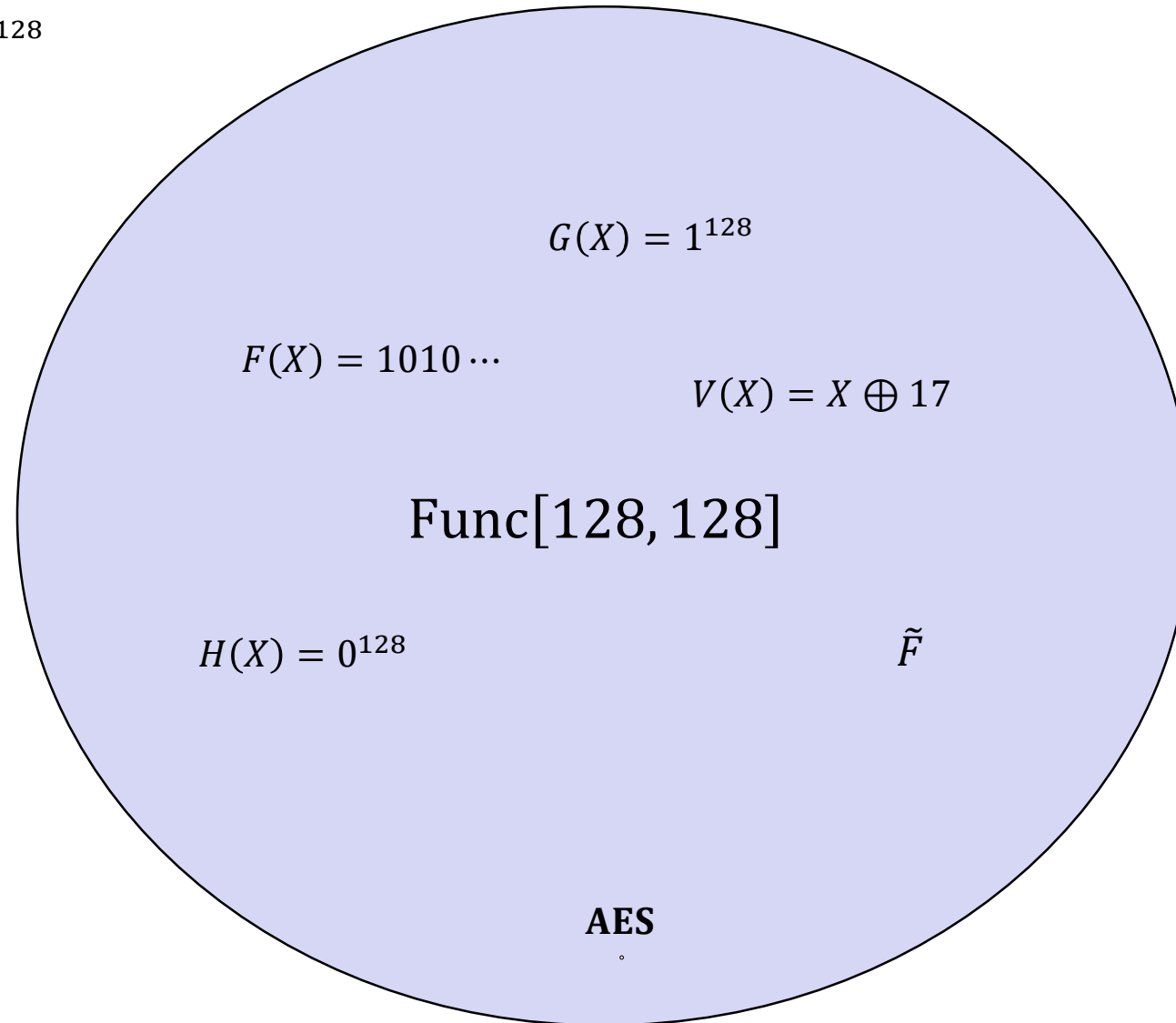
128

- Bits to specify *one* function: $2^{in} \cdot out$

- Each *distinct* bitstring of length $2^{in} \cdot out$ represents a *unique* function

Block ciphers – security

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

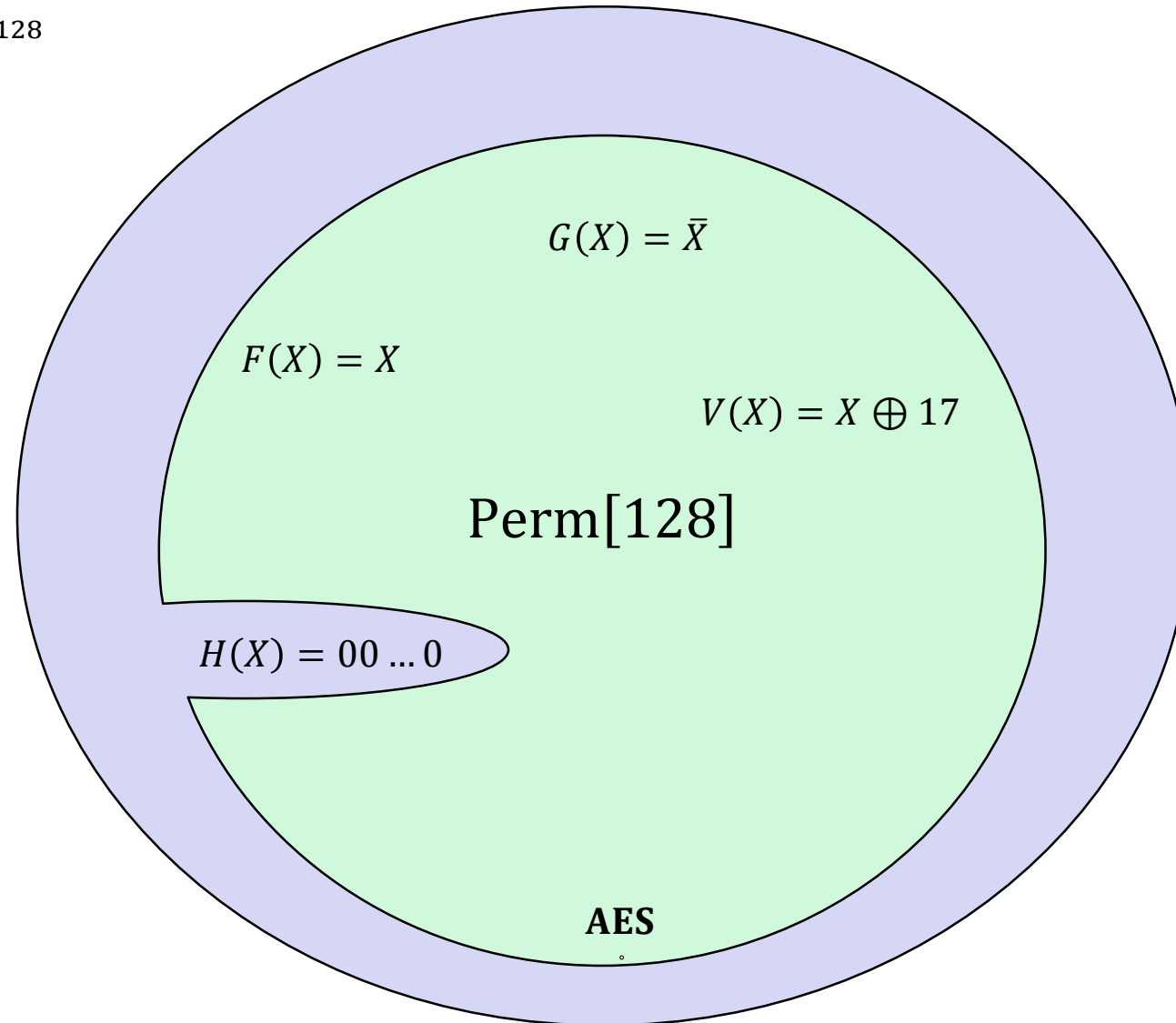


$$|\text{AES}| = 2^{128}$$

Block ciphers – security

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

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



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