Information Security and Privacy

David Jelenc

Administrative info

- 3 lab session slots
  - Check official Timetable
- Office hours
  - Slot: TBD and published on Moodle
  - Office: R3.50, third floor, first bridge on the left
  - Welcome at any time, but email first
- Point of contact
  - Forum in Moodle (preferred)
  - david.jelenc@fri.uni-lj.si

Syllabus (1/2)

- Using cryptoprimitives to develop secure applications using cryptographic libraries (JACL)
- Topics (by weeks)
  - Secrecy (stream and block ciphers)
  - Integrity (Message digests, MACs) & authenticated encryption
  - Public-key encryption, steganography,
  - Key exchange protocols, digital signatures, key-derivation
- Prerequisite: Java programming (basics)
- At the end: Midterm 1

Syllabus (2/2)

- Using various tools to secure and securely administrate computer systems (Ubuntu Linux)
- Topics (by weeks)
  - Firewalls with Netfilter/IPtables
  - Firewalls continued, NAT, routing
  - Secure shell with OpenSSH
  - Virtual private networking with IPsec
  - Authentication, authorization and accounting with FreeRADIUS
- Prerequisite: Computer networking & Linux basics
- At the end: Midterm 2

Grading

- **Weekly homework assignments**
  - Needs to be handed in by the end of the week
  - Pass/Fail grading: need at least 7/9 to pass
- **Homework challenges**
  - Optional homework assignments; allows students to receive extra credit
- **Midterms**
  - During the semester, one after the completion of each syllabus topic
  - Format: quiz + programming assignment
    - Quiz [50%]: 10 multiple-choice/closed-form-type questions
    - Programming assignment [50%]:
      - A short seminar-like task, similar to those at lab sessions
      - The solution has to be *defended;* the grading is done in the presence of the student
      - Programming is done on classroom computers, open-book style, Internet disconnected
- **Final lab total**
  - Points = MAX(0.5*MidTerm1 + 0.5*MidTerm2 + ExtraPoints, 100)

Tentative grading dates

- **Dates are tentative, may still change**
- **Midterm 1**
  - Week Nov 25 to Dec 1
- **Midterm 2**
  - Week Jan 6 to Jan 12

Communication Secrecy

Contents

Introduction: providing confidentiality

- We'd like to provide confidential communication
  - Only the intended recipient(s) should be able to read the data

- Two types of encryption and decryption
  - Symmetric ciphers
  - Asymmetric ciphers


I LOVE MATH
IT TORTURES PEOPLE
imgflip.com

Symmetric Ciphers

- A cipher defined over (K, M, C) is a pair of "comp. eff." algorithms ($\boldsymbol{E}$, $\boldsymbol{D}$), where

$$\boldsymbol{E}: K \times M \rightarrow C$$
$$\boldsymbol{D}: K \times C \rightarrow M$$

  s. t. for all k in K and m in M:

$$\boldsymbol{D}(\mathrm{k}, \boldsymbol{E}(\mathrm{k}, \mathrm{m})) = \mathrm{m}$$

- ***E*** is often randomized, ***D*** is always deterministic

Perfect Secrecy

- What is a “secure” cipher?
  - Shannon: Cipher text should reveal “no information” about the plain text
- A cipher (***E***, ***D***) over (*K, M, C*) has **perfect secrecy** if for all m_0, m_1 є *M* ($|m_0|=|m_1|$) and for all c є *C*

$$\Pr[\boldsymbol{E}(k, m_0) = c] \quad = \quad \Pr[\boldsymbol{E}(k, m_1) = c]$$

  where k є *K* is randomly chosen
  - Given cipher text c, one cannot tell whether c is a cryptogram of m_0 or m_1

One Time Pad

- Vernam (1917)
  - $M = C = K = \{0, 1\}^n$
  - $E(k,m) = k \oplus m$
  - $D(k, c) = k \oplus c$
- Features
  - Given a truly random key, OTP has ***perfect secrecy***
  - Key has to be ***random*** and it must be used ***only once***
  - <u>Impractical:</u> Shannon shows that perfect secrecy requires keys to be at least as long as the plain text

Pseudo Random Generator

- Idea: Replace a "random" with a "pseudorandom" key
  $\boldsymbol{G}: \{0, 1\}^s \rightarrow \{0, 1\}^n$ where $n >> s$
- Pseudo Random Generator (PRG) is a function ***G*** that maps *seed space* to *key space*
  - Is "efficiently" computable by a deterministic algorithm
  - Its output (keys) "looks random"
- Stream ciphers
  - $\boldsymbol{E}(k, m) := m \oplus \boldsymbol{G}(k)$
  - $\boldsymbol{D}(k, c) := c \oplus \boldsymbol{G}(k)$
- Examples: RC4, CSS, eStream, Salsa 20
- Can stream ciphers have perfect secrecy, why?

Stream Ciphers: perfect secrecy?

- Stream ciphers cannot have perfect secrecy
  - Keys (seeds) are shorter than messages
- Can stream ciphers ever be secure?
  - Need a new definition of security
  - Security will depend on PRG used

Pseudo Random Generators: defs

- **Statistical test** is an algorithm **A**: $\{0, 1\}^n \rightarrow \{0, 1\}$
  - Returns 1 if it *thinks* the input string is random, 0 otherwise
- **Advantage** of st. test ***A*** against PRG ***G***:

$$\mathrm{Adv}_{\mathrm{PRG}}[A, G] = \left| \Pr_{k \xleftarrow{R} K}[A(G(k)) = 1] - \Pr_{r \xleftarrow{R} \{0,1\}^n}[A(r) = 1] \right|$$

  - If *close* to 0, ***A*** cannot distinguish ***G*** from random
  - Otherwise, ***A*** can distinguish ***G*** from random
- **Def.** A PRG G is secure, if for all eff. stat. tests A: $\mathrm{Adv}_{\mathrm{PRG}}[A, G]$ is negligible.

Negligible? Assume less than 2^{-80}

Pseudo Random Generators: defs

- Def: A PRG is **unpredictable** if given an initial sequence of bits (a prefix), one cannot *efficiently* predict the next bit (with probability higher than ½ + ε)
- **Thm:** A PRG is secure iff. it is unpredictable.
- In practice
  - Unknown if there are provably secure PRG
  - But we have heuristic candidates

Perfect secrecy, threat model

- (Recall) A cipher (***E***, ***D***) over (*K, M, C*) has **perfect secrecy** if for all m_0, m_1 є *M* ($|m_0|=|m_1|$) and for all c є *C*

  $$\Pr[\boldsymbol{E}(k, m_0) = c] \quad = \quad \Pr[\boldsymbol{E}(k, m_1) = c]$$

  where k є *K* is randomly chosen

  - Given cipher text c, one cannot tell whether c is a cryptogram of m_0 or m_1
- **Threat model:** basis for reasoning about security
  - **Adversary's power:** what can she do
  - **Adversary's goal:** what is she trying to achieve

Semantic security: def

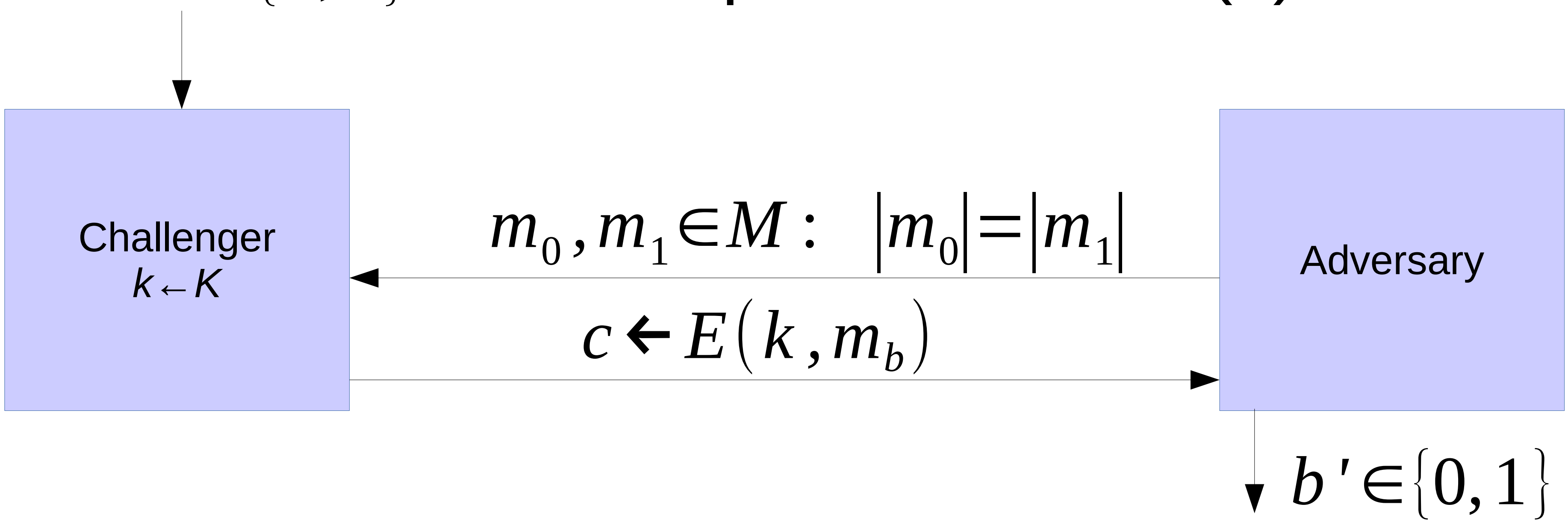
(for one-time key; adv. sees only one CT)

- Adversary's power: **observe <u>one</u> ciphertext**
  - Every message is encrypted with its own key; a particular key is used only once
- Adversary's goal: **learn about the plaintext**

Semantic security: def

(for one-time key; adv. sees only one CT)

- For $b \in \{0,1\}$ define experiments EXP(b) as



- Def: $\zeta = (E, D)$ is **semantically secure** if for all eff. adversaries A $\mathrm{Adv}_{SS}[A, \zeta]$ is negligible.

$$\mathrm{Adv}_{SS}[A, \zeta] := \left| \Pr[\mathrm{EXP}(0) = 1] - \Pr[\mathrm{EXP}(1) = 1] \right|$$

Semantic security

- Informally
  - A cipher has **semantic security** if given only cipher text, an attacker cannot *practically* derive any information about the plain text
- **Thm:** Given a secure PRG, derived stream cipher is semantically secure

Final thoughts

- Two-time pad attack
  - Never use stream-cipher key to encrypt more than one message
    - later we show a secure a multi-message exchange

$$c_1 \leftarrow m_1 \oplus \boldsymbol{G}(k)$$
$$c_2 \leftarrow m_2 \oplus \boldsymbol{G}(k)$$

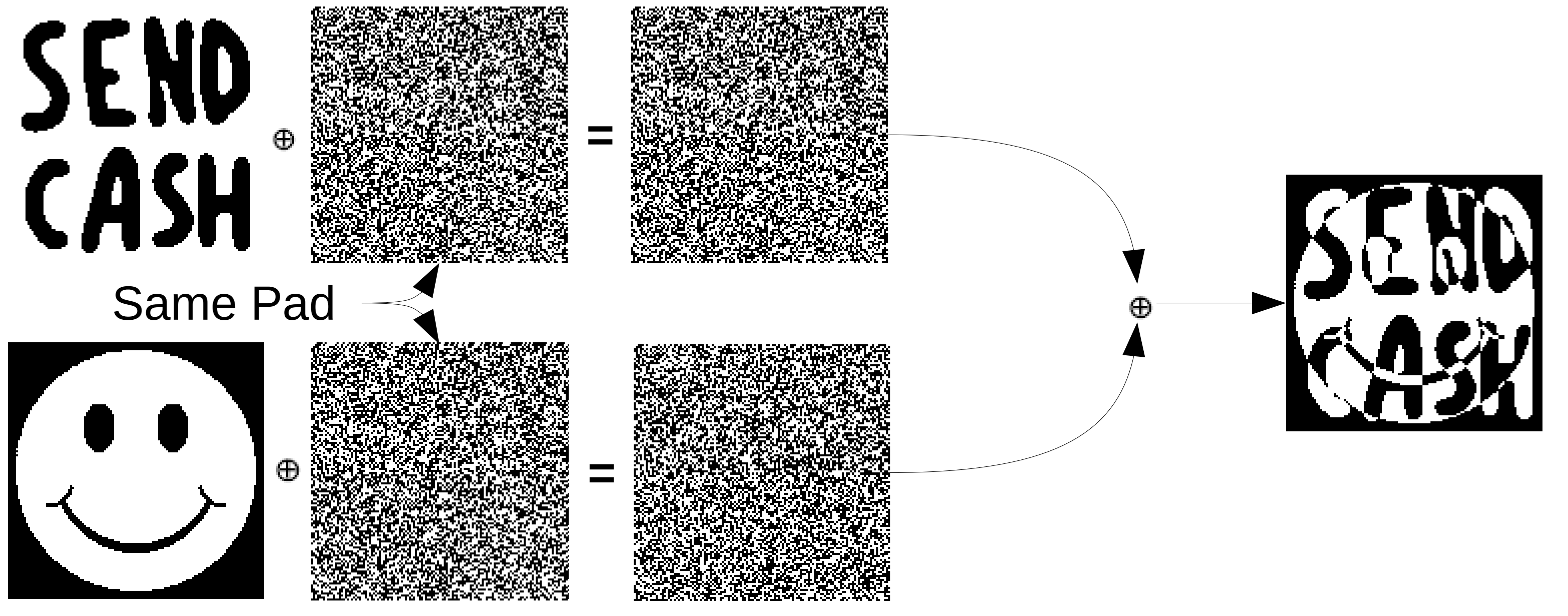
$$m_1 \oplus m_2 \leftarrow c_1 \oplus c_2$$

  - Redundancy in natural languages and in encoding schemes (ASCII, UTF-8, …) to separate $m_1 \oplus m_2 \rightarrow m_1, m_2$
  - http://www.crypto-it.net/eng/attacks/two-time-pad.html

https://crypto.stackexchange.com/questions/59/taking-advantage-of-one-time-pad-key-reuse

Final thoughts

- Two-time pad attack



https://crypto.stackexchange.com/questions/59/taking-advantage-of-one-time-pad-key-reuse

Final thoughts

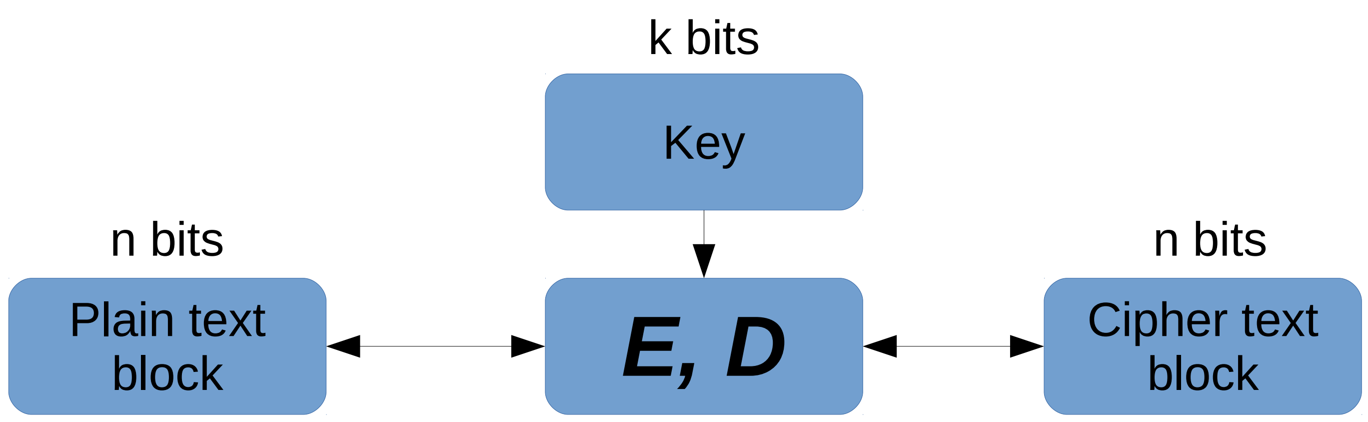
- Malleability
  - Modifications to CT are not detected and have predictable impact on the plain text

```
Encrypt: c  ← m  ⊕  k
 Modify: c’ ← c  ⊕  p
Decrypt: m’ ← c’ ⊕  k
```

  - What is the relation between *m* and *m’*?

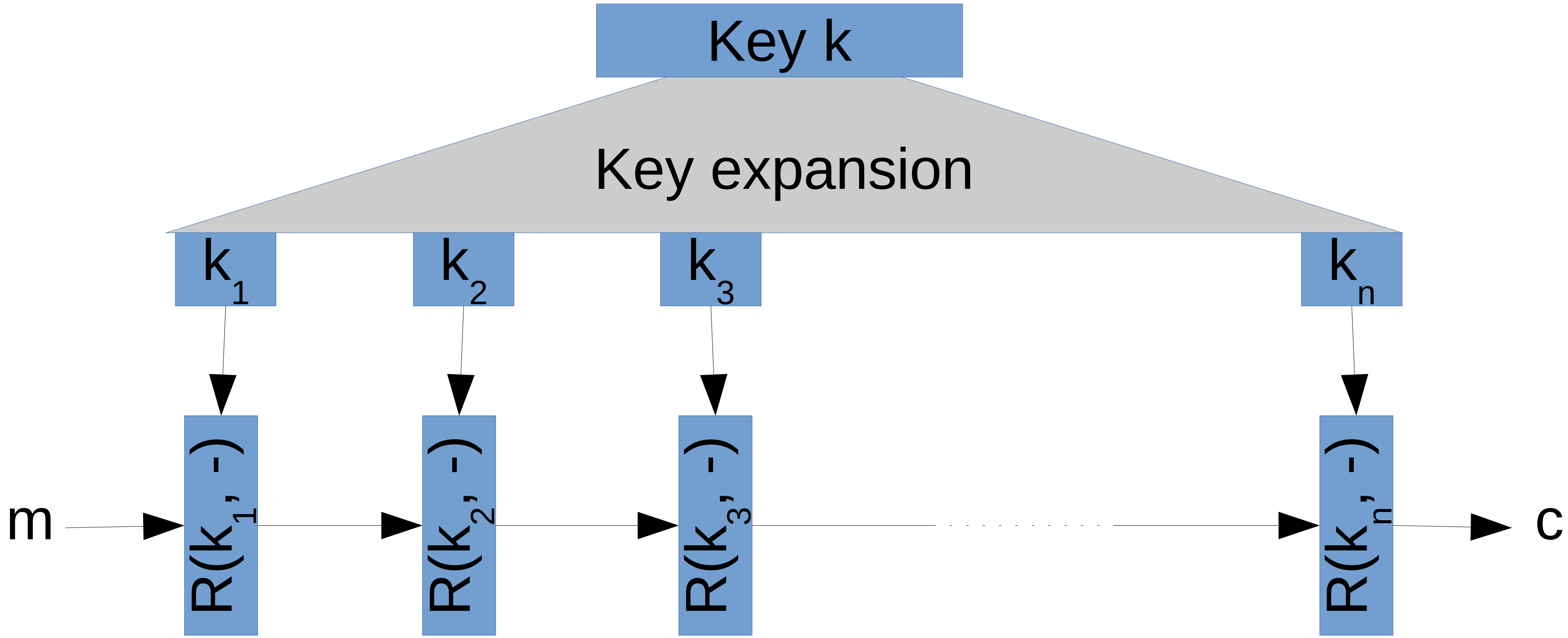
Block Ciphers

- Notable examples
  - 3DES: n = 64 bits, k = 168 bits
  - AES: n = 128 bits, k = 128, 192, 256 bits



Block Ciphers: Built by iteration

- R(k, m) is a round function
  - 3DES (n = 48)
  - AES (n = 10)



Adaptation of: Dan Boneh, Cryptography I, Stanford.

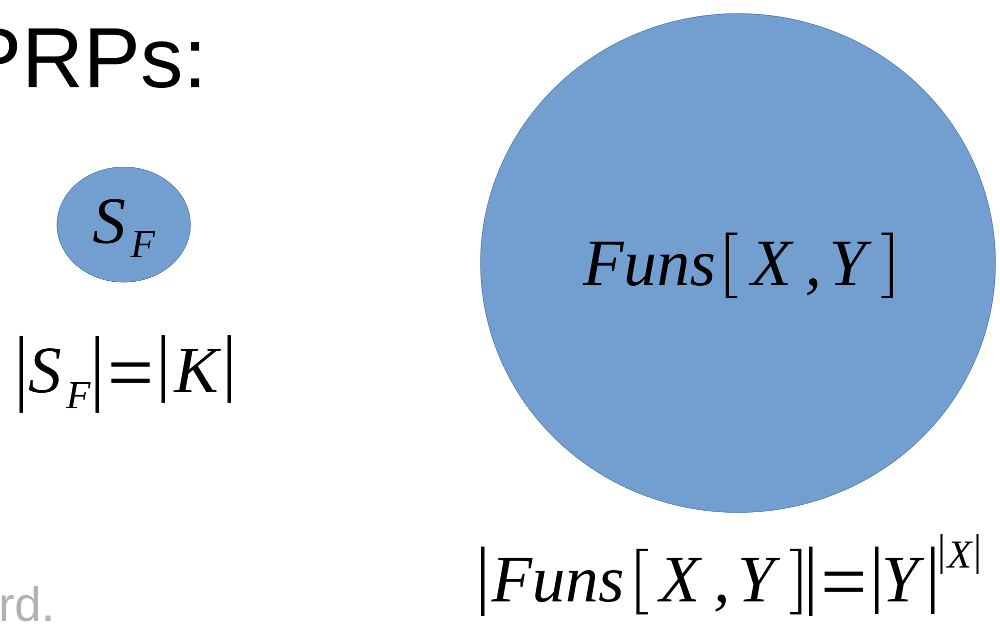
Abstracting BC: PRF and PRP

- Pseudo Random Function (PRF) defined over (K, X, Y):
  $F: K \times X \rightarrow Y$
  - We can evaluate F(k, x) efficiently
- Pseudo Random Permutation (PRP) defined over (K, X):
  $E: K \times X \rightarrow X$
  - We can evaluate E(k, x) efficiently
  - E(k, -) has an inverse
  - We have an efficient inversion algorithm D(k, x)
  - (All PRPs are PRFs.)

Secure PRF

- Let $F: K \times X \to Y$ be a PRF
  - $Funs[X, Y]$ the set of all functions from X to Y
  - $S_F = \{F(k, -) : \forall k \in K\} \quad \subseteq Funs[X, Y]$

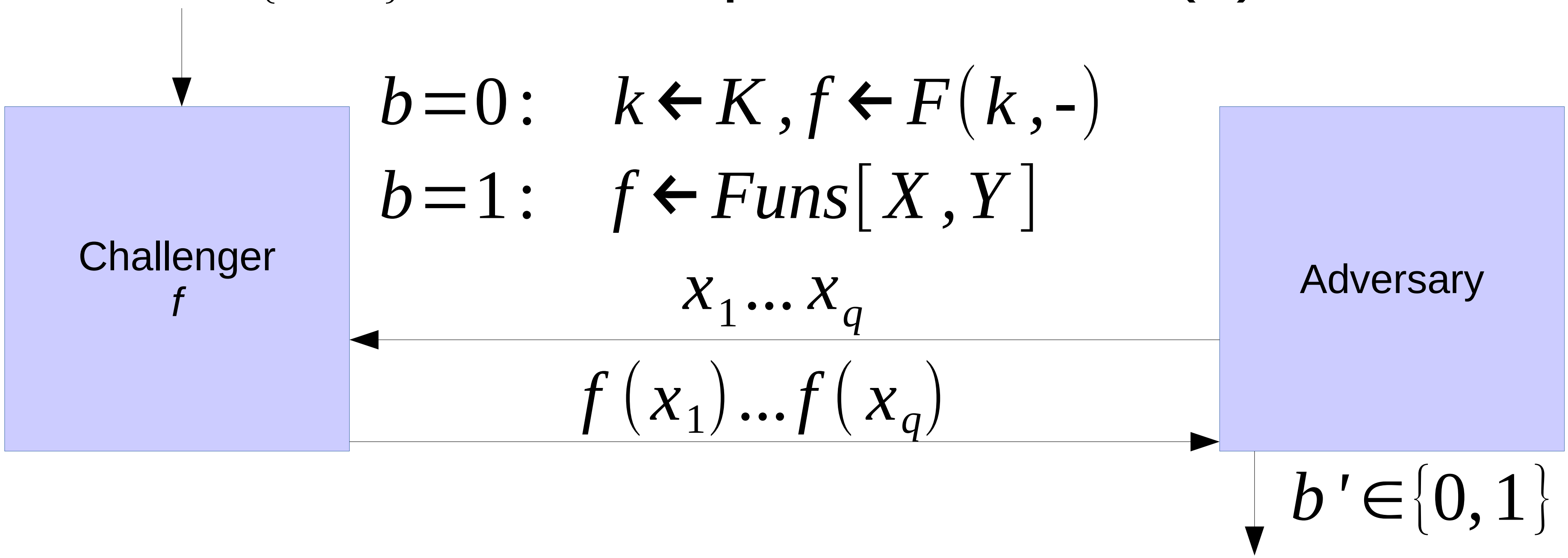
  Intuitively
  - A PRF is secure if a random function in $Funs[X, Y]$ is indistinguishable from a random function in S_F
- Believed to be secure PRPs:
  - AES, 3DES, Blowfish



Adaptation of: Dan Boneh, Cryptography I, Stanford.

Secure PRF (def.)

- For $b \in \{0,1\}$ define experiment EXP(b) as

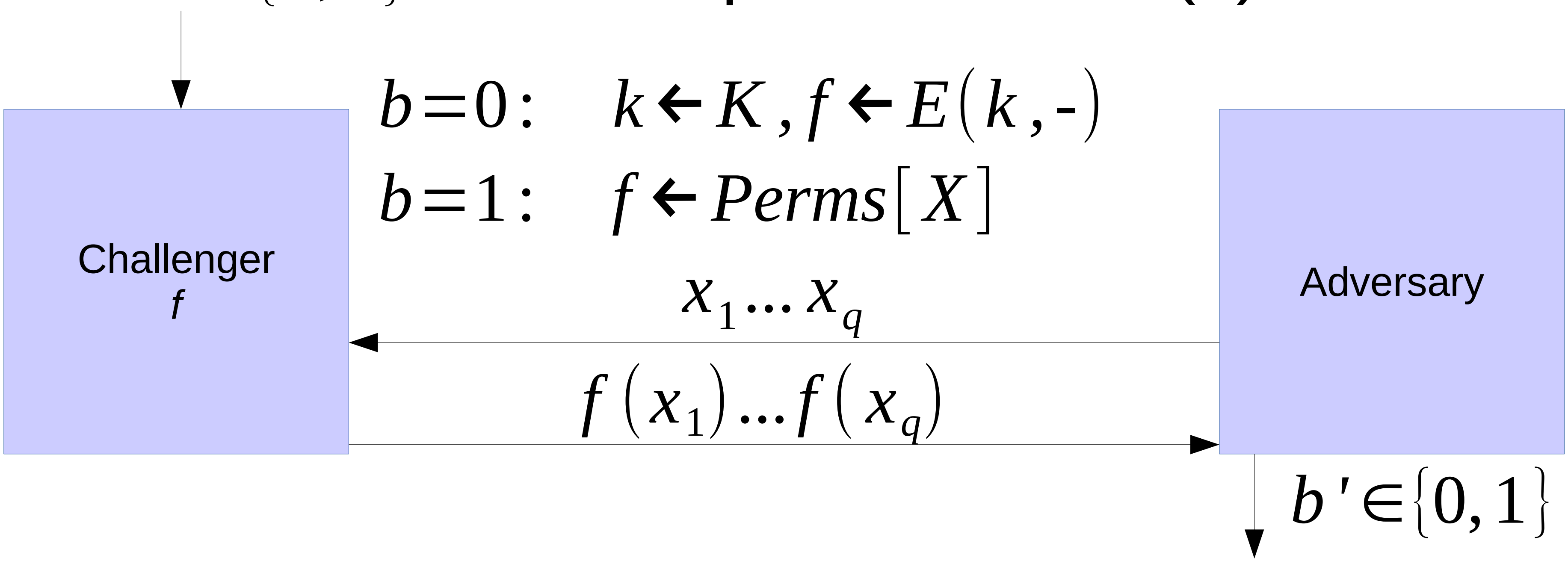


- Def: *F* is a secure PRF if for all eff. adversaries A $\mathrm{Adv}_{\mathrm{PRF}}[A,F]$ is negligible.

$$\mathrm{Adv}_{\mathrm{PRF}}[A,F] := \left|\Pr[\mathrm{EXP}(0)=1] - \Pr[\mathrm{EXP}(1)=1]\right|$$

Secure PRP (def.)

- For $b \in \{0, 1\}$ define experiment EXP(b) as



- Def: *E* is a secure PRP if for all eff. adversaries A $\mathrm{Adv}_{\mathrm{PRP}}[A, E]$ is negligible.

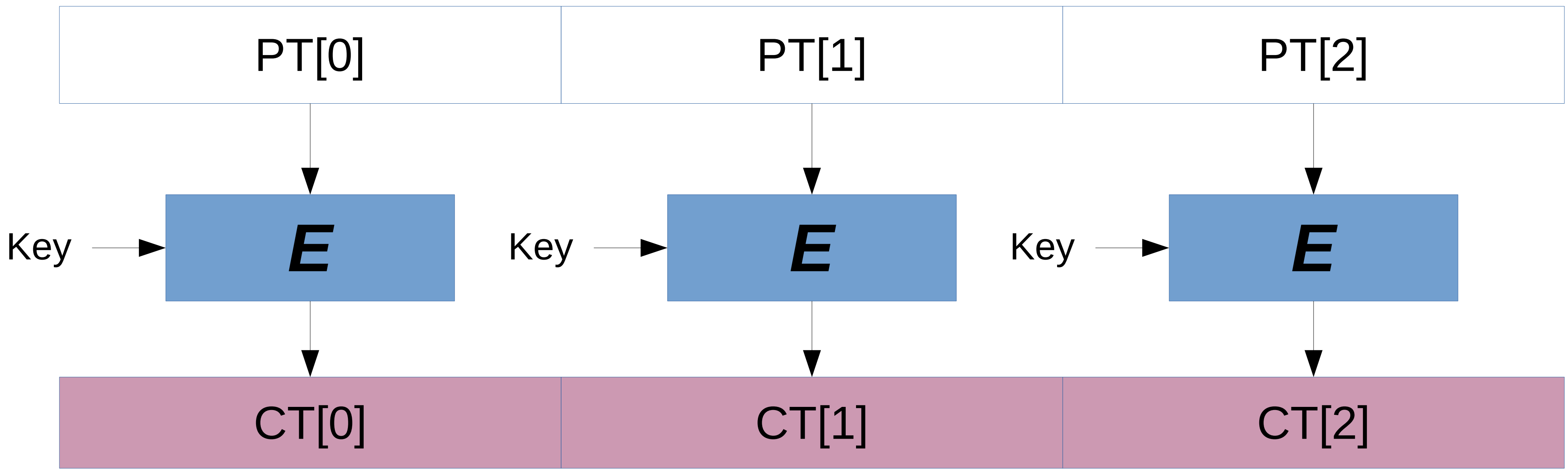
$$\mathrm{Adv}_{\mathrm{PRP}}[A, E] := \left| \Pr[\mathrm{EXP}(0) = 1] - \Pr[\mathrm{EXP}(1) = 1] \right|$$

Block Ciphers: Modes of Operation

- Goal: How do we build a secure encryption from secure PRP (e.g. AES)
  - A PRP encrypts a single data block. How do we encrypt larger data?
- Semantic security (still for one-time key only)
  - Adversary's power: **observe one ciphertext**
  - Adversary's goal: **learn about plaintext**

MO: Electronic Code Book

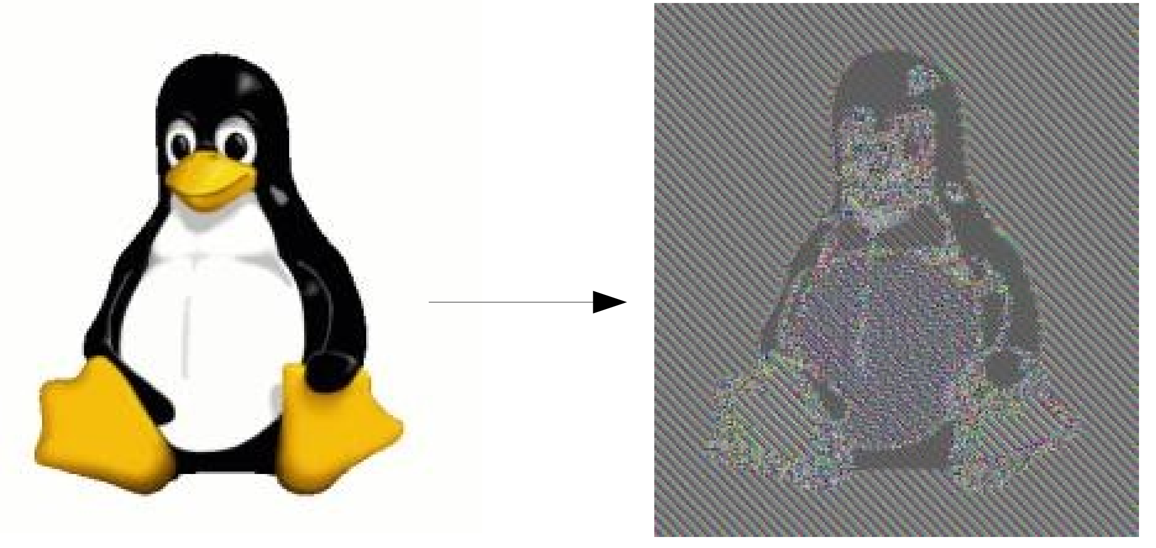
- “Solution” Electronic Code Book (ECB):
  - Split the data into blocks
    - if needed, extend the last block with padding bits
  - Independently encrypt each block



Adaptation of: Dan Boneh, Cryptography I, Stanford.

MO: Electronic Code Book

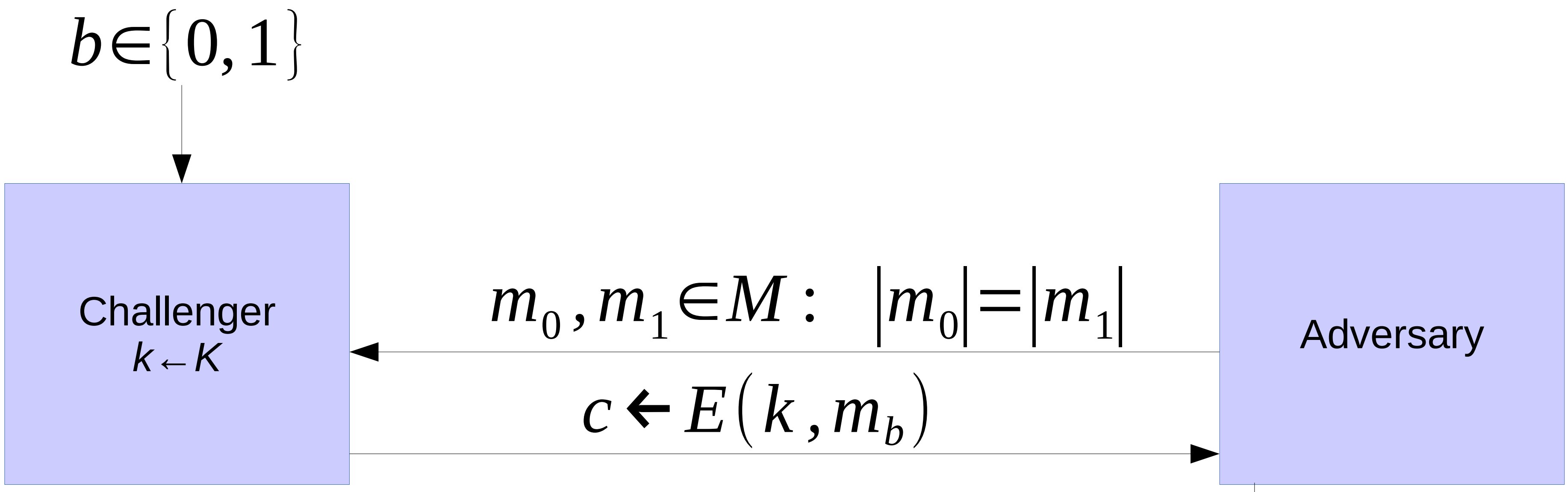
- Problem: If PT[0] == PT[1], then CT[0] == CT[1]
  - If two plaintext blocks are the same, so are the corresponding ciphertexts blocks



How does the Adversary win the semantic security game against ECB?

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Electronic_Codebook_.28ECB.29

ECB is not semantically secure



- What should be m_0, m_1?
- What should the adversary output?

MO: Deterministic counter mode

- Deterministic counter from a pseudorandom function (PRF)

	PT[0]	PT[1]	PT[2]
$\oplus$	PRF(k, 0)	PRF(k, 1)	PRF(k, 2)
	CT[0]	CT[1]	CT[2]

- Creates a stream cipher from a PRF
- Secure (but only for encrypting a single message which may consists of multiple blocks)

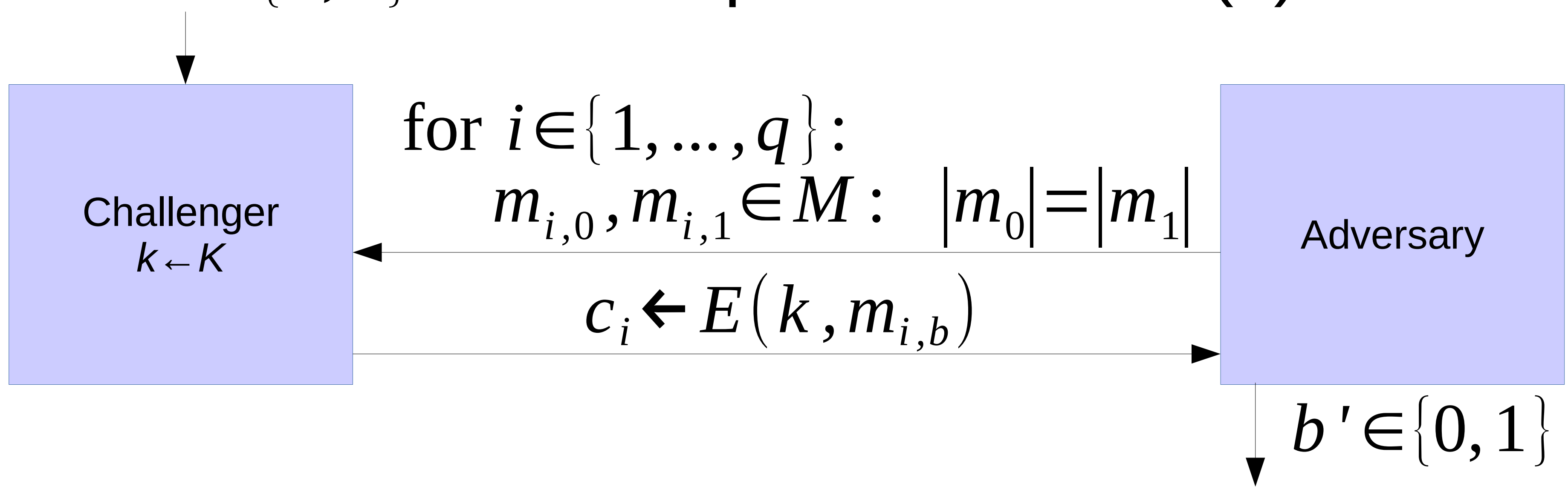
Semantic security for many-time key

- Key is used more than once: adversary sees many CTs encrypted with the same key
- Adversary's power: **chosen-PT attack (CPA)**
  - Can obtain the encryption of any message of her choice
- Adversary's goal: **break semantic security**
  - Learn about the PT from the CT

Semantic security for CPA (def)

(for many-time key)

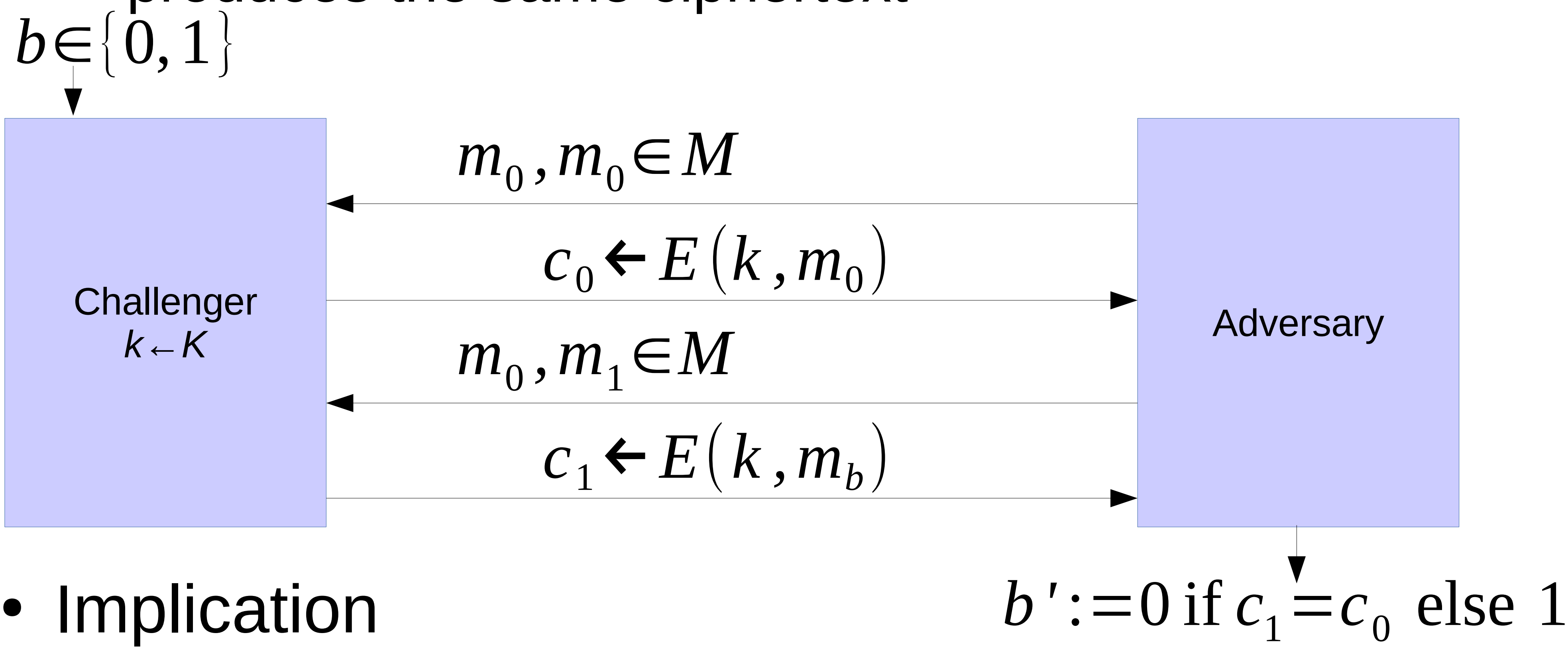
- Let $\zeta=(E,D)$ be a cipher defined over (K,M,C)
- For $b\in\{0,1\}$ define experiments EXP(b) as



- Def: $\zeta=(E,D)$ is **semantically secure under CPA** if for all eff. adversaries A $\mathrm{Adv}_{\mathrm{CPA}}[A,\zeta]$ is negligible. $\mathrm{Adv}_{\mathrm{CPA}}[A,E] := |\Pr[\mathrm{EXP}(0)=1] - \Pr[\mathrm{EXP}(1)=1]|$

Ciphers insecure under CPA

- Suppose a cipher is deterministic
  - Given some message *m*, the cipher always produces the same ciphertext



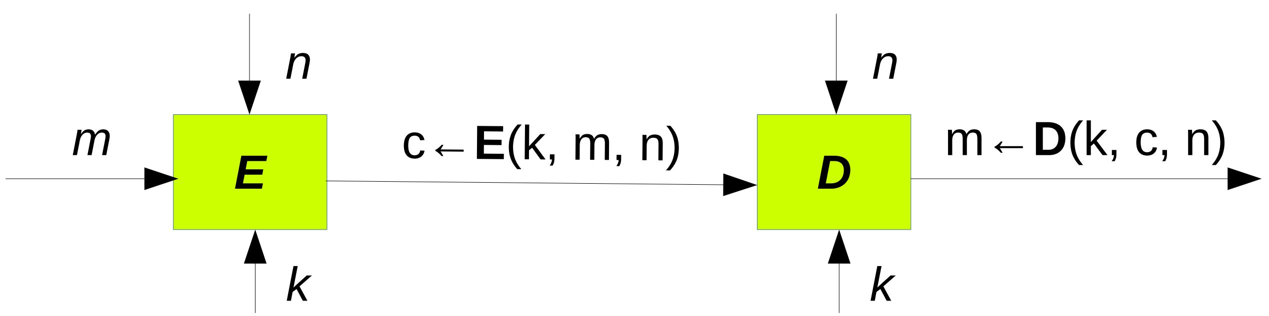
- Implication
  - An attacker can learn that two encrypted elements (files, packets, ...) are the same

Ciphers insecure under CPA

- If a key is to be used multiple times, the encryption should be **non-deterministic**:
  - Encrypting the same PT twice, must produce different CTs
- Solutions
  - Randomized encryption
  - Nonce-based encryption

Non-deterministic encryption

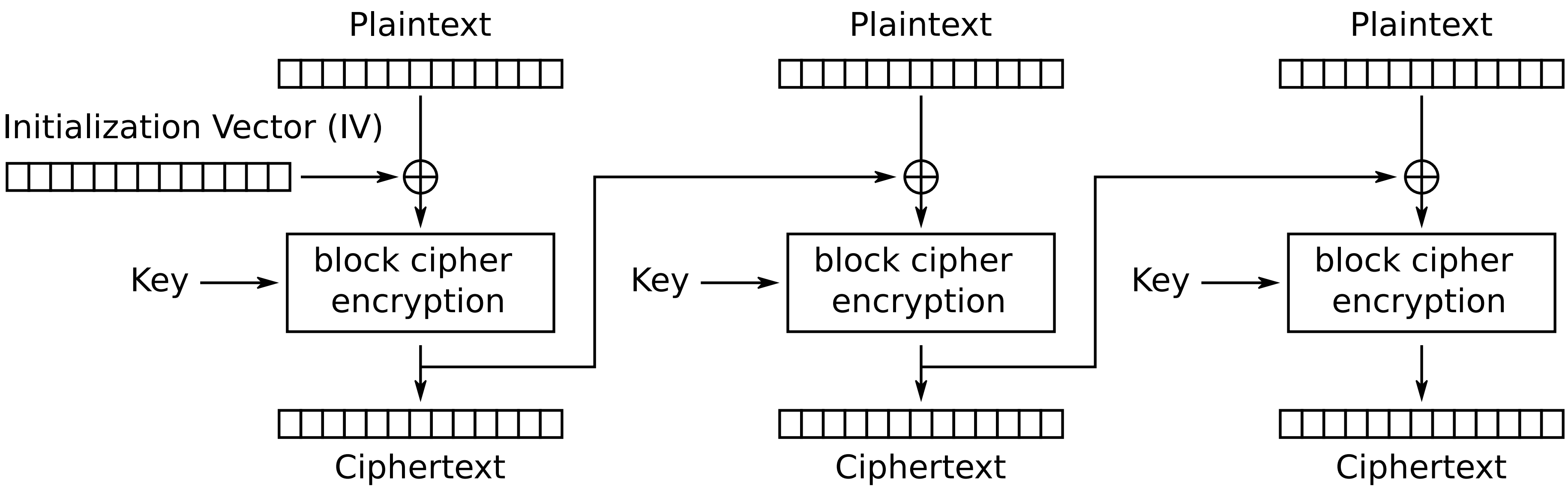
- Nonce ***n***: a value that changes from message to message
  - Pair *(key, n)* **<u>must never repeat</u>**
- Method 1: Nonce is a random value (AES-CBC)
- Method 2: Nonce is a counter (AES-CTR)



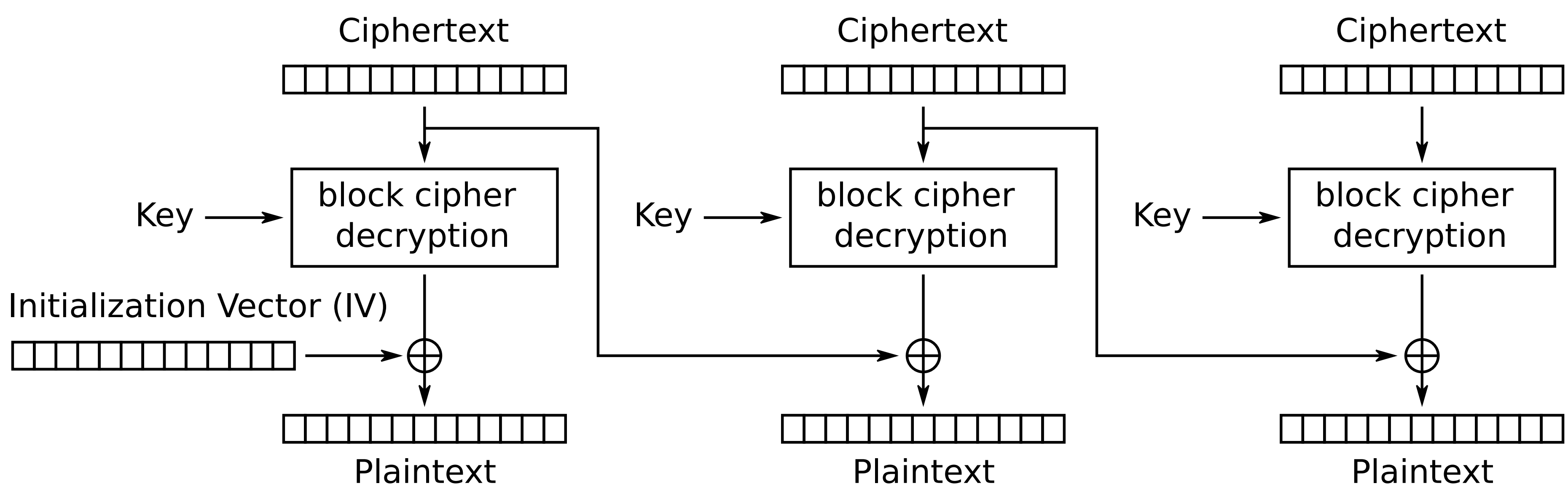
- CPA system should be secure even when the adversary chooses nonces

Modes of Operation: CBC

- Randomize the encryption with an initialization vector (IV)
  - Sent unencrypted
  - Must generate new random IV for every message: pair ***(key, IV)*** must never repeat
  - IV must be unpredictable
- Forces encryption to be sequential
  - Decryption may be parallelized



Cipher Block Chaining (CBC) mode encryption

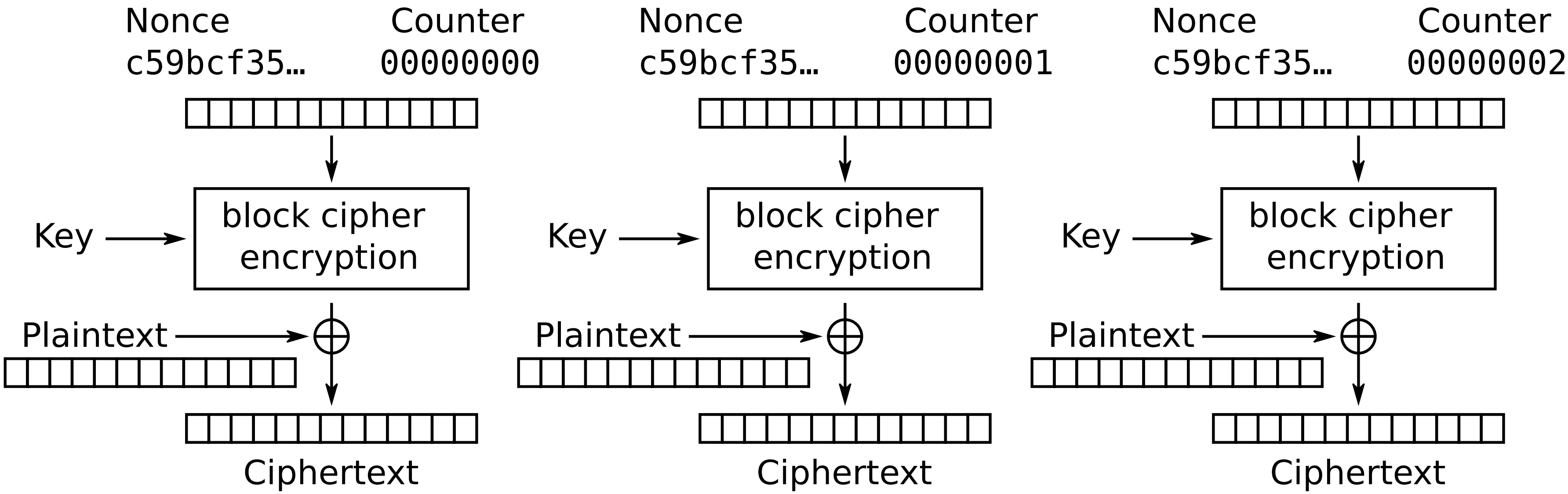


Cipher Block Chaining (CBC) mode decryption

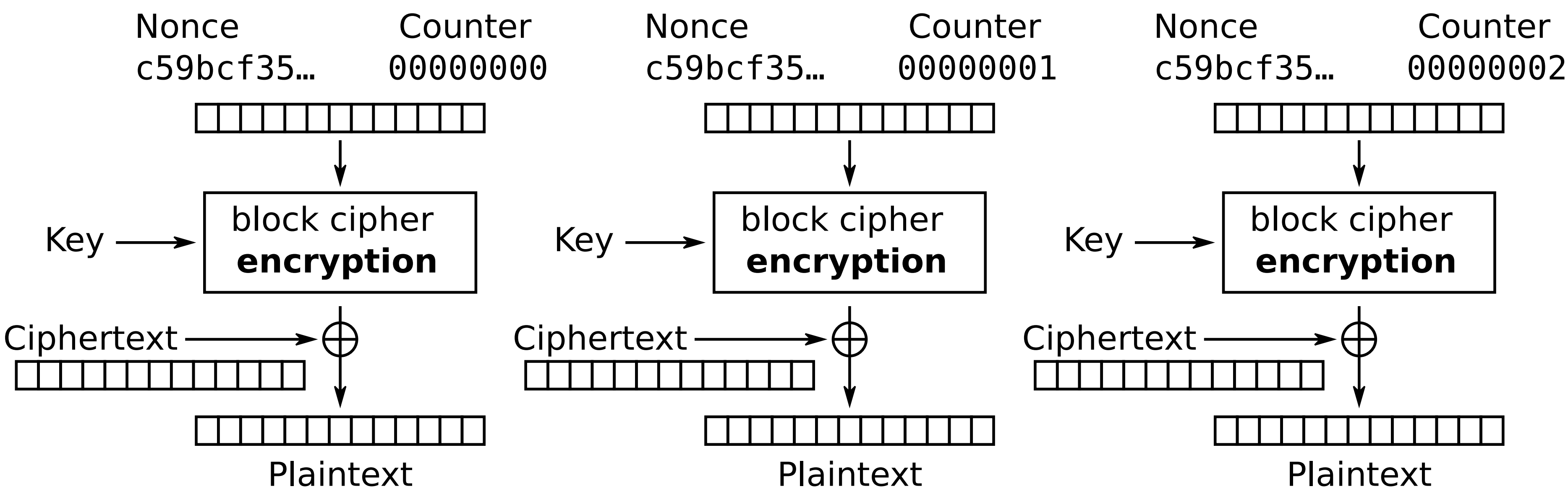
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Counter Mode

- The random element is a counter
  - Or a combination of a random IV and a counter
  - The combination must not repeat for the lifetime of the key
- Encryption and decryption can be done in parallel
- In effect, creates a stream cipher out of a block cipher



Counter (CTR) mode encryption



Counter (CTR) mode decryption

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Summary

- Two security notions
  - Semantic security against one-time CPA
  - Semantic security against many-time CPA
- Only covered secrecy against passive attackers
  - Adversaries can see, but not modify cipher text
  - We’ll cover integrity next week

Power / Goal	One-time key	Many-time key (CPA)
Semantic security	**Stream-ciphers** **Deterministic CTR-mode**	**Rand CBC** **Rand CTR-mode**

Integrity

Contents

Introduction

- Integrity: maintaining accuracy and completeness of data
- Goal
  - Prevent adversary from modifying data
  - More feasible: detect if data has been altered
- Examples
  - Protecting files on disks
  - Assuring installation of correct software
  - Assuring the delivered packet has not been tempered with in traffic

Message Authentication Code

Compute TAG

$$t = S(k, m)$$

Verify TAG

$$1 = V(k, m, t)$$

$MAC\ I = (S, V)$ defined over (K, M, T) is a pair of algs.:

$$S : K \times M \rightarrow T$$

$$V : K \times M \times T \rightarrow \{0,1\}$$

$$|M| \gg |T|$$

such that

$$\forall\, k \in K, m \in M: \;\; V(k, m, S(k, m)) = 1$$

Adaptation of: Dan Boneh, Cryptography I, Stanford.

Is a shared secret required?

- Is all these secrecy required?
- Could we not just simply use
  - MD-5 or
  - SHA-{1,2,3} or
  - CRC?

Secure MAC

- Attacker’s power: **Chosen message attack**
  - For $m_1 ... m_q$ attacker is given $t_i = S(k, m_i)$
- Attacker’s goal: **Existential forgery**
  - Produce a **new** valid (m, t) s. t.

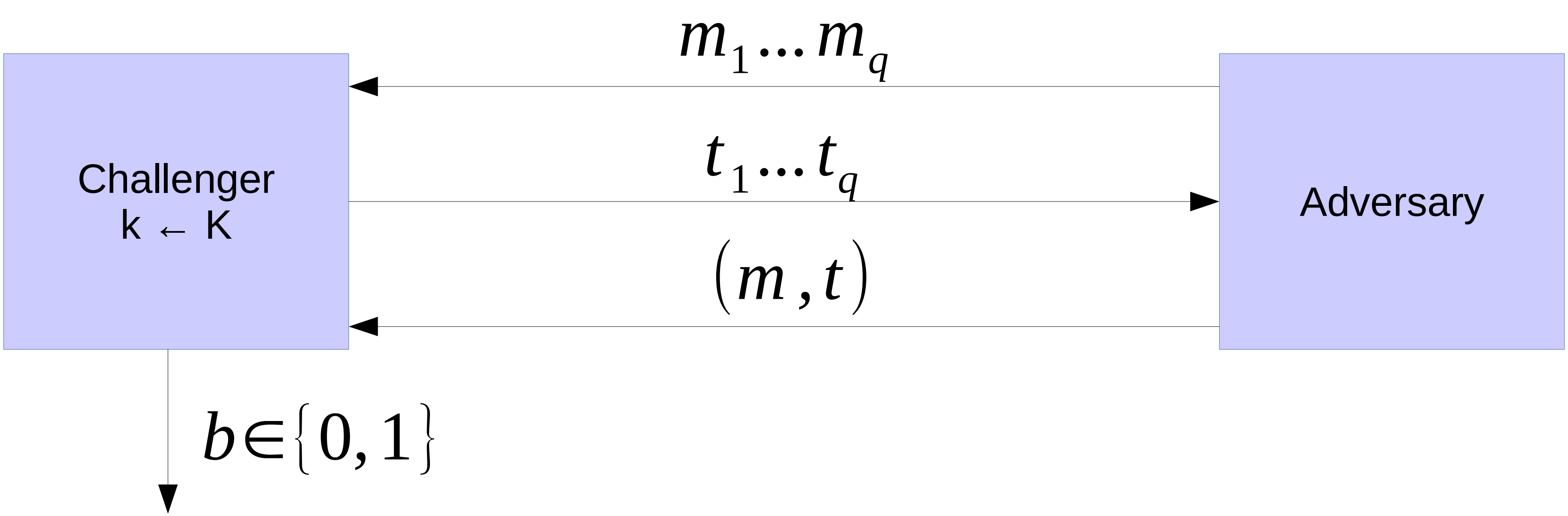
$$(m, t) \notin \{(m_{1,} t_1) ... (m_q, t_q)\}$$

Implications

$\rightarrow$ attacker cannot produce a valid tag for a new message

$\rightarrow$ given (m, t) attacker cannot produce (m, t') for $t \neq t'$

Secure MAC (def)



$$b = 1 \quad \text{if } V(k, m, t) = 1 \text{ and } (m, t) \notin \{(m_1, t_1) \ldots (m_q, t_q)\}$$
$$b = 0 \quad \text{otherwise}$$

$I = (S, V)$ is a **secure MAC** if for all “efficient” adversaries A

$$\text{Adv}_{\text{MAC}}[A, I] = \Pr[\text{Chal. outputs } 1]$$

is “negligible”.

Secure MAC

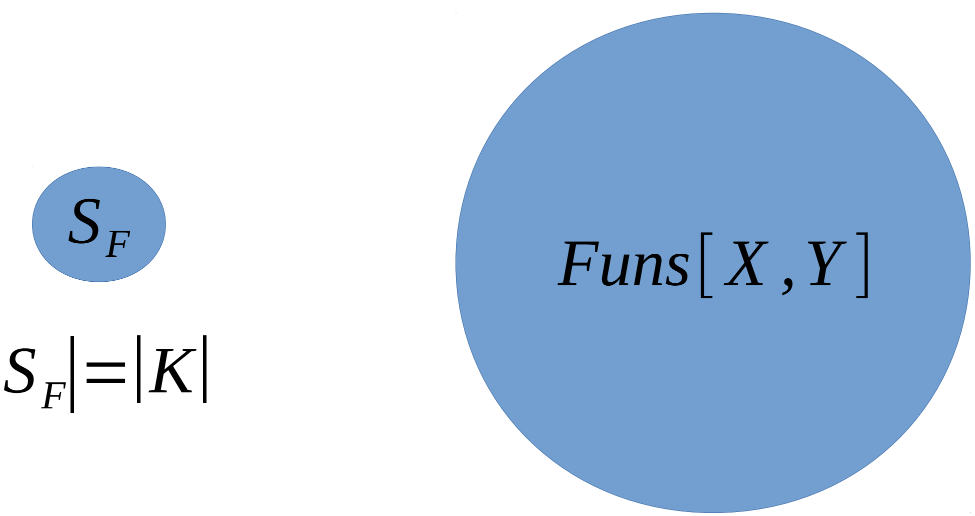
- Negligible?
  - Assume less than 2^{-80}
- Suppose a *S(k, m)* computes 10-bit tags
  - Is such a MAC secure, why?

(Recall) Secure PRF

- Let $F : K \times X \to Y$ be a PRF
  - $Funs[X,Y]$ the set of all functions from X to Y
  - $S_F = \{F(k,-) : \forall k \in K\} \quad \subseteq Funs[X,Y]$

Intuitively

- A PRF is secure if a random function in $Funs[X,Y]$ is indistinguishable from a random function in S_F



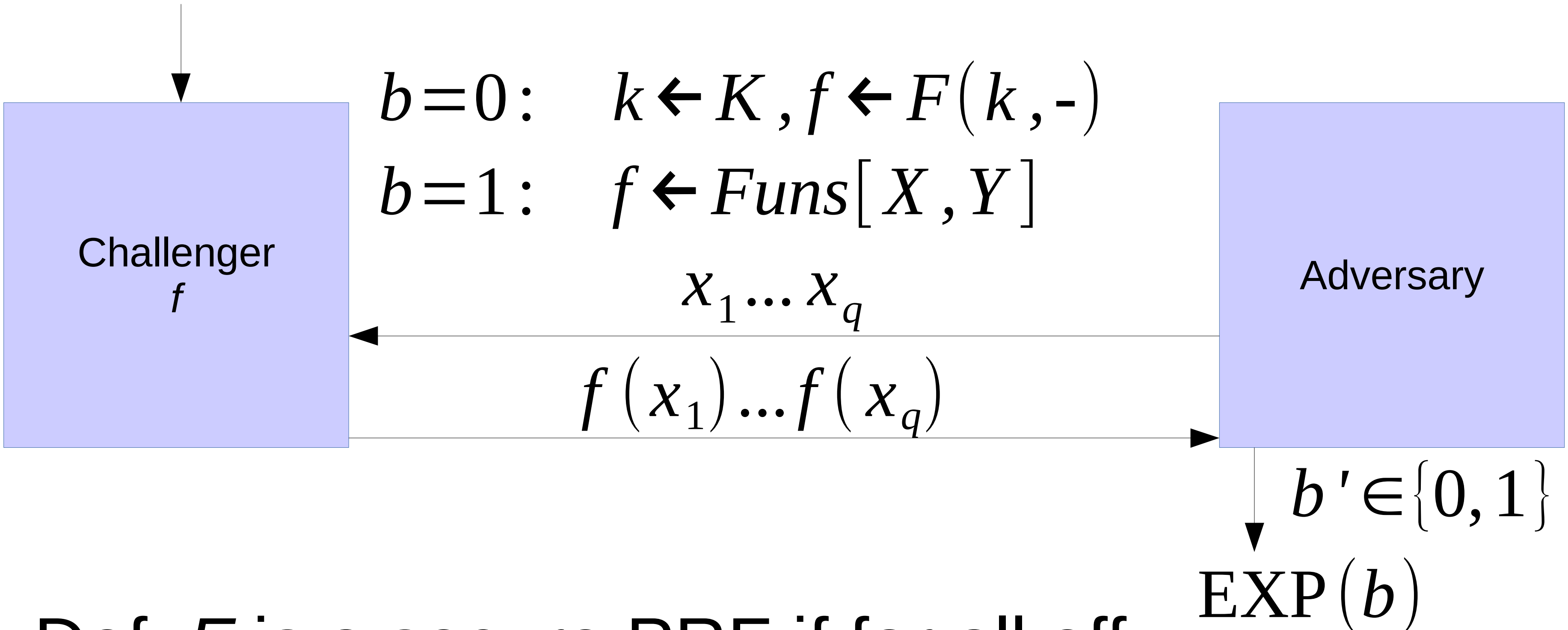
$|S_F| = |K|$

$|Funs[X,Y]| = |Y|^{|X|}$

Adaptation of: Dan Boneh, Cryptography I, Stanford.

(Recall) Secure PRF (def.)

- For $b \in \{0,1\}$ define experiment EXP(b) as



- Def: *F* is a secure PRF if for all eff. adversaries A $\text{Adv}_{\text{PRF}}[A, F]$ is negligible.

$$\text{Adv}_{\text{PRF}}[A, F] := \left| \Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1] \right|$$

Adaptation of: Dan Boneh, Cryptography I, Stanford.

Secure PRF → Secure MAC

- For a PRF $F: K \times X \to Y$ define MAC $I_F = (S, V)$

$$S(k,m) := F(k,m)$$

$$V(k,m,t) := \begin{cases} 1 & t = F(k,m) \\ 0 & \text{otherwise} \end{cases}$$

- **Thm**. If F is a secure PRF and $1/|Y|$ is negligible (i.e. $|Y|$ is sufficiently large), then I_F is a secure MAC.

Truncating MACs based on PRFs

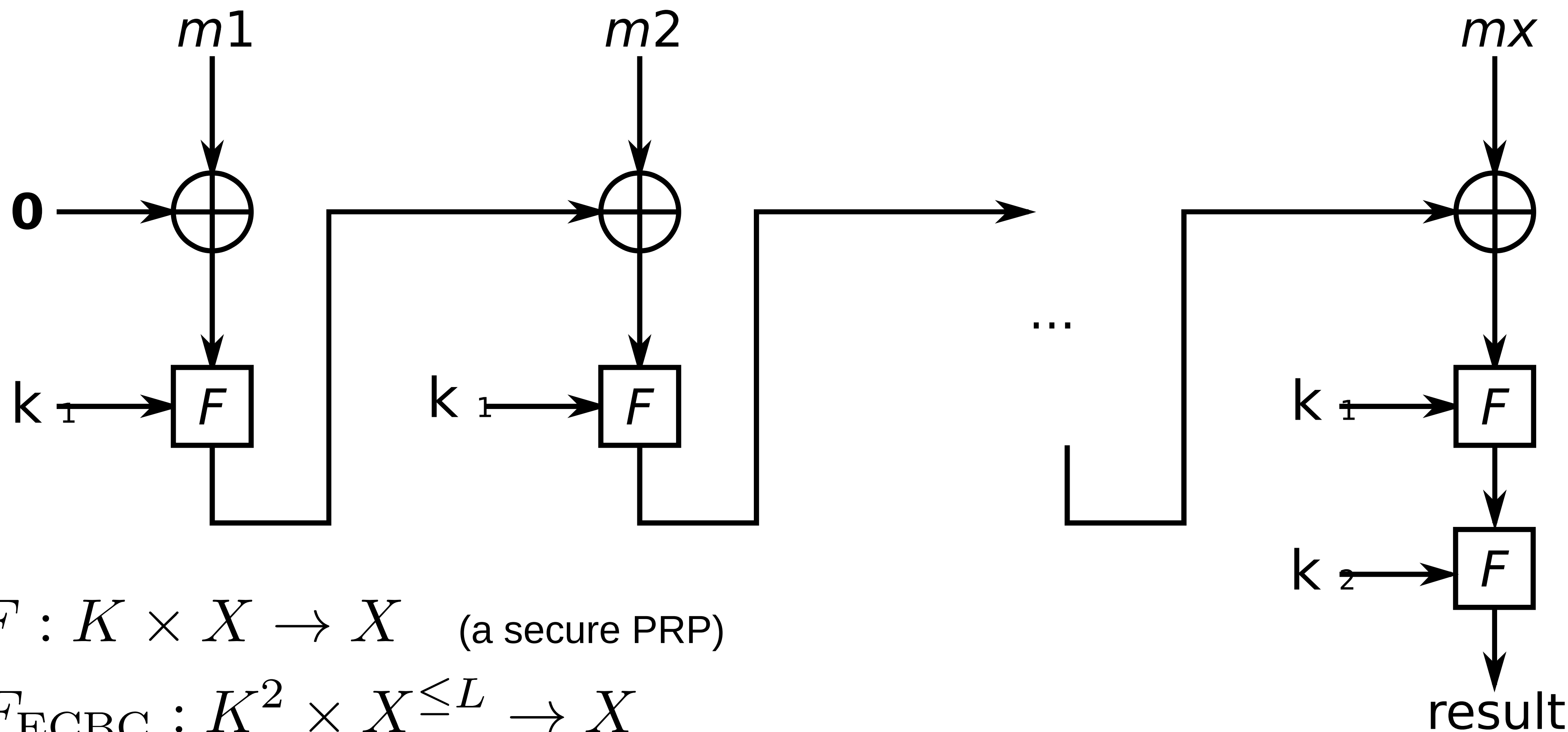
- Lemma: Suppose $F: K \times X \rightarrow \{0,1\}^n$ is a secure PRF. So is $F_t(k,m) := F(k,m)[1 \ldots t]$ for all $1 \leq t \leq n$

- If (S, V) is a MAC based on a secure PRF that outputs *n*-bit tags, then the truncated MAC that outputs *w* bits is also secure.
  - As long as 2^{-w} is still negligible

Examples of secure MAC

- AES (or any secure PRF)
  - A secure MAC for 16-byte (128-bit) messages
- Longer messages?
  - **CBC-MAC**
  - **HMAC**
- Both convert a small-PRF into a big-PRF

ECBC-MAC



$F : K \times X \to X$ (a secure PRP)

$F_{\mathrm{ECBC}} : K^2 \times X^{\leq L} \to X$

$$X^{\leq L} = \bigcup_{i=1}^{L} X^i$$

https://en.wikipedia.org/wiki/CBC-MAC

Hash-MAC (HMAC)

- Built from *collision resistance*
- Let $H : M \to T$ be a hash function $|M| \gg |T|$
- A **collision** for H is a pair $m_0, m_1 \in M$ such that:

$$H(m_0) = H(m_1) \text{ and } m_0 \neq m_1$$

- Function H is **collision resistant** if for all *explicit* "eff." algs. A $\mathrm{Adv}_{\mathrm{CR}}[A, H]$ is negligible.

$$\mathrm{Adv}_{\mathrm{CR}}[A, H] := \Pr[A \text{ outputs collision for } H]$$

- Example: SHA-256

MAC from CR

- Let $I=(S,V)$ be a MAC for short messages over (K,M,T) (e.g. AES)
- Let $H: M^{\text{BIG}} \to M$
- Def: $I^{\text{BIG}}=(S^{\text{BIG}},V^{\text{BIG}})$ over (K,M^{BIG},T) as:

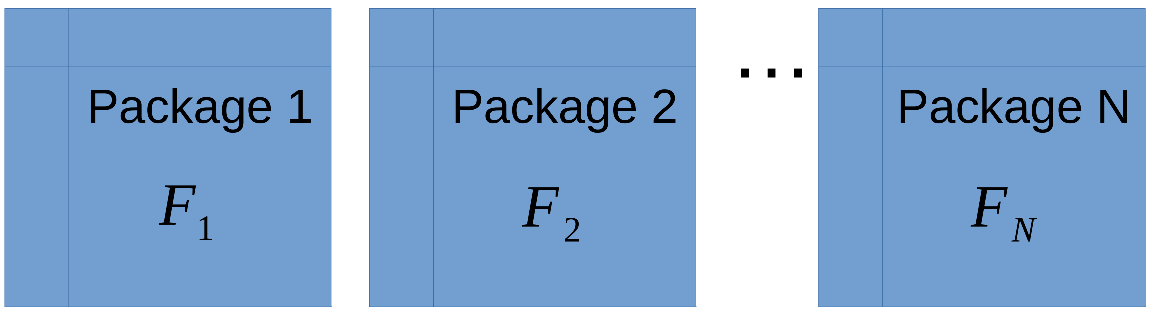
$$S^{\text{BIG}}(k,m):=S(k,H(m))$$

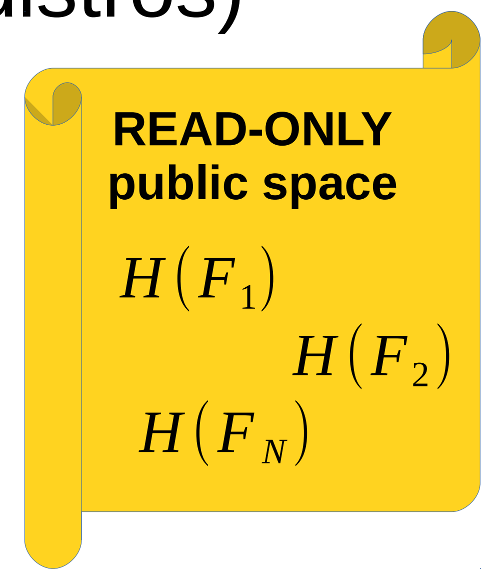
$$V^{\text{BIG}}(k,m,t):=V(k,H(m),t)$$

- **Thm**. If I is a secure MAC and H is collision resistant, then I^{BIG} is a secure MAC.
- Example: $S(k,m):=\text{AES}_{\text{2-block-CBC}}(k,\text{SHA-256}(m))$

Example: Integrity using CR hash

- Protecting software packages (Linux distros)





- User downloads a package and verifies it using hashes in public space
  - If *H* is collision resistant, the attacker cannot modify packages without being detected
- We require <u>no shared secret</u>, but we need a <u>read-only public space</u>

Generic attack on CR

- Let $H : M \rightarrow \{0,1\}^n$ be a hash function $|M| \gg 2^n$
- Generic algorithm to find a collision

1)Chose $\sqrt{2^n} = 2^{\frac{n}{2}}$ random messages: $m_1 \ldots m_{2^{n/2}} \in M$ distinct w.h.p.

2)For $i = 1 \ldots 2^{n/2}$: compute $t_i = H(m_i)$

3)Look for a collision $(t_i = t_j)$. If not found, go to 1.

- How many iterations before we find a collision?

The birthday paradox

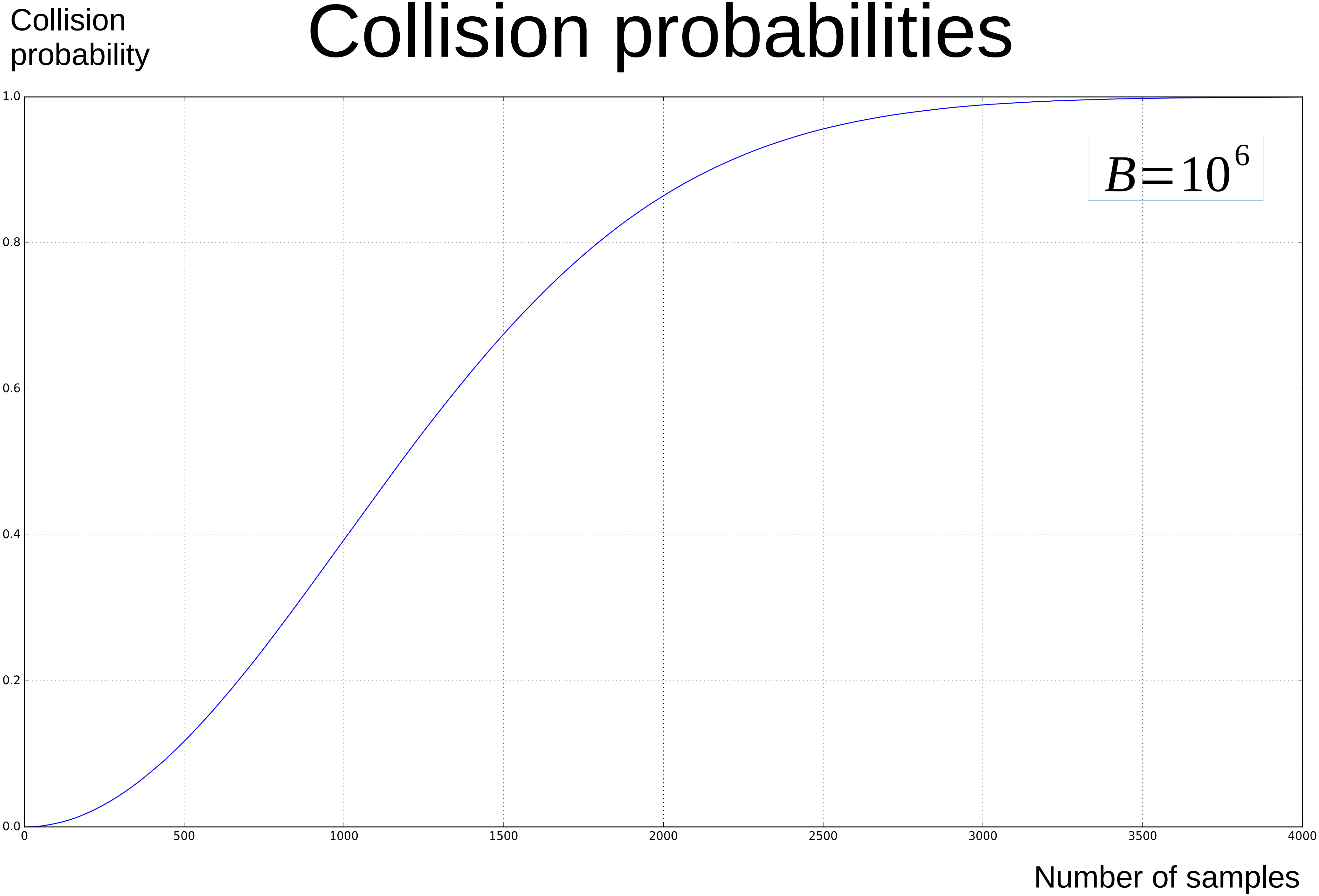
- **Thm.** Let $r_1 \ldots r_n \in [1 \ldots B]$ be independent and identically distributed integers. If we sample $n = 1.2 \times \sqrt{B}$ samples from interval $[1 \ldots B]$ then the probability of finding a collision is

$$\Pr[\exists i \neq j : r_i = r_j] \geq 0.5$$

- Approximation of collision probability given *n* samples with Taylor series

$$p(n) \approx 1 - e^{\frac{-n(n-1)}{2B}}$$

Collision probabilities



Generic attack on CR

- Let $H : M \rightarrow \{0,1\}^n$ be a hash function $|M| \gg 2^n$
- Generic algorithm to find a collision

  1)Chose $\sqrt{2^n} = 2^{\frac{n}{2}}$ random messages: $m_1 \ldots m_{2^{n/2}} \in M$ distinct w.h.p.

  2)For $i = 1 \ldots 2^{n/2}$: compute $t_i = H(m_i)$

  3)Look for a collision $(t_i = t_j)$. If not found, go to 1.

- How many iterations before we find a collision?
  - ~ 2
  - Running time $O(2^{\frac{n}{2}})$

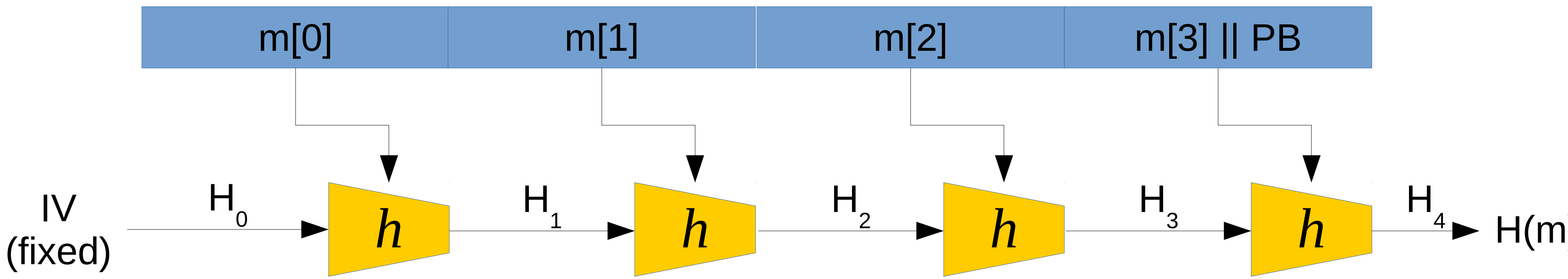
Example CR hash functions

Function	Digest (tag) size [bits]	Generic attack time
MD-5	128	2^{64}
SHA-1*	160	2^{80}
SHA-256	256	2^{128}
SHA-512	512	2^{256}
Whirpool	512	2^{256}

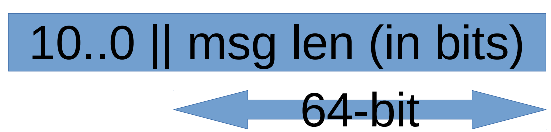
* Found collision by performing $2^{63.1}$ evaluations https://shattered.it

Merkle-Damgard construction

- Goal: given CR function for **short** messages, construct CR function for **long** messages



- CR for short messages (compression function) $h: T \times X \to T$
- CR for long messages $H: X^{\leqslant L} \to T$
- PB: padding block

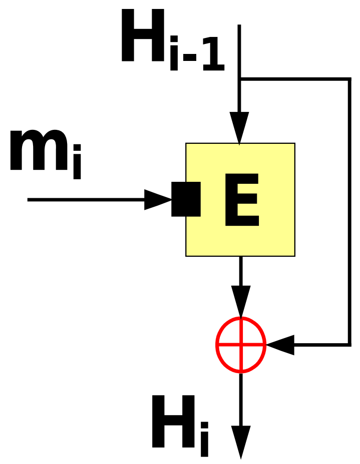


  - If no space for PB, add an extra block
- **Thm.** If *h* is CR, so is *H.*

Compression functions

- Built from block ciphers $E: K \times \{0,1\}^n \rightarrow \{0,1\}^n$
- Several constructions
  - **Davies-Meyer**

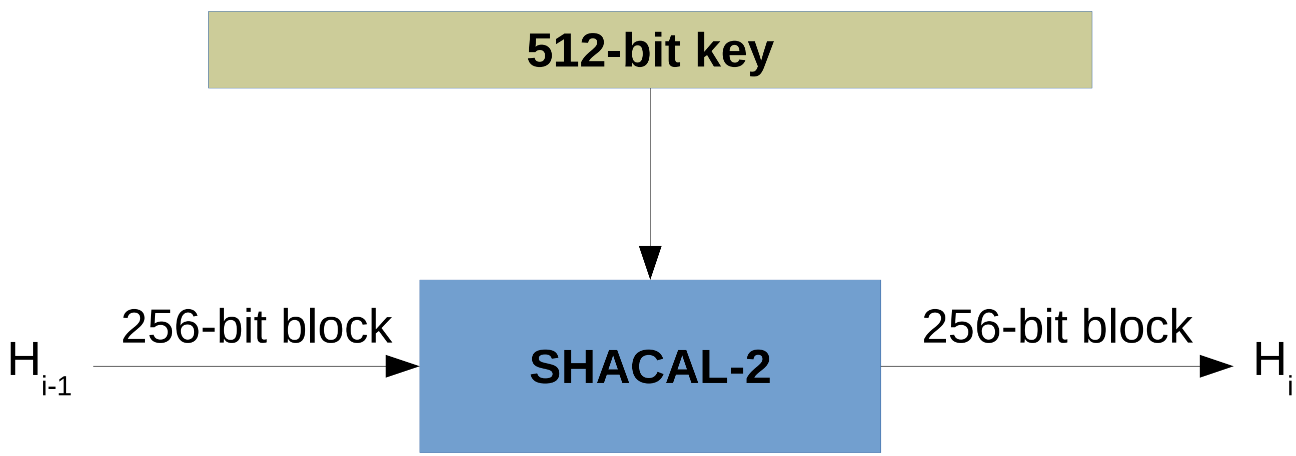
    $$h(H,m) := E(m,H) \oplus H$$
  - Matyas–Meyer–Oseas
  - Miyaguchi–Preneel



https://en.wikipedia.org/wiki/One-way_compression_function

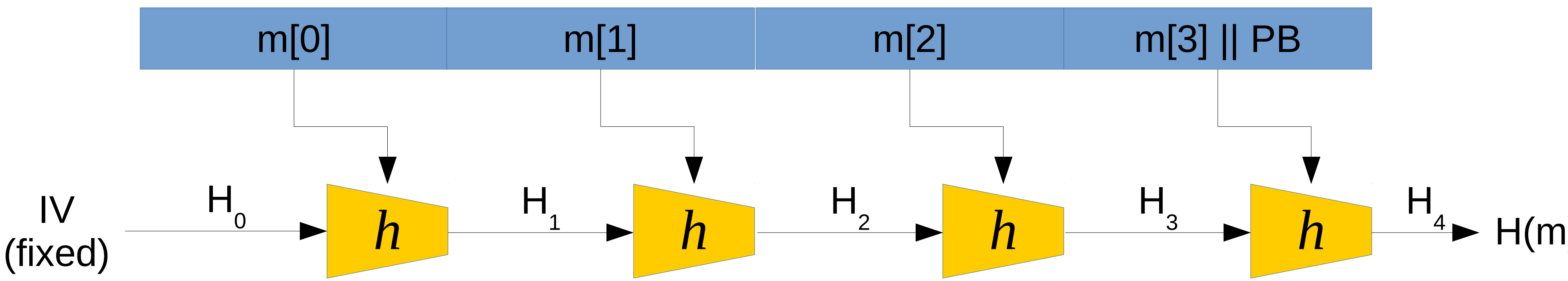
Example: SHA-256

- Merkle-Damgard iterative construction
- Davies-Meyer compression function
  - Block cipher: SHACAL-2



MAC from M-D hash func.

- Can we construct a MAC directly from *H*? (e.g SHA-256)
- Naive attempt $S(k,m) := H(k \| m)$
  - Is it secure?



  - If you knew $H(k \| m)$ could you compute $H(k \| m \| \mathrm{PB} \| w)$ for any w? How?
  - Length-extension attack

Standardized solution: HMAC

- Most commonly used on the Internet
  - https://tools.ietf.org/html/rfc2104
- Given CR hash function *H*, define a MAC as

$$S(k,m) := H(k \oplus \mathrm{opad} \quad \| \quad H(k \oplus \mathrm{ipad} \| m))$$

  - Built from a black-box implementation of SHA-256
  - Assumed to be a secure PRF
  - TLS 1.2 requires support of HMAC-SHA1-96 (TLS 1.3 does not)

Authenticated Encryption

Contents

- Ciphertext integrity
- AE definitions
- Chosen Ciphertext Attack
- Constructions
  - Encrypt-then-MAC
  - Encrypt-and-MAC
  - MAC-then-Encrypt

Authenticated Encryption (AE)

- Everything demonstrated so far provides
  - either integrity
  - or confidentiality (security against eavesdropping)
- CPA security does not provide secrecy against active attacks (where an attacker can tamper with ciphertext)
  - ➔ If you require integrity → **MAC**
  - ➔ If you require integrity and confidentiality → **AE**

AE: Desired properties

– An authenticated encryption system $\zeta = (E, D)$ is a cipher where

as usual $E : K \times M \times N \rightarrow C$

but $D : K \times C \times N \rightarrow M \cup \{\perp\}$ $\perp \notin M$

Nonce

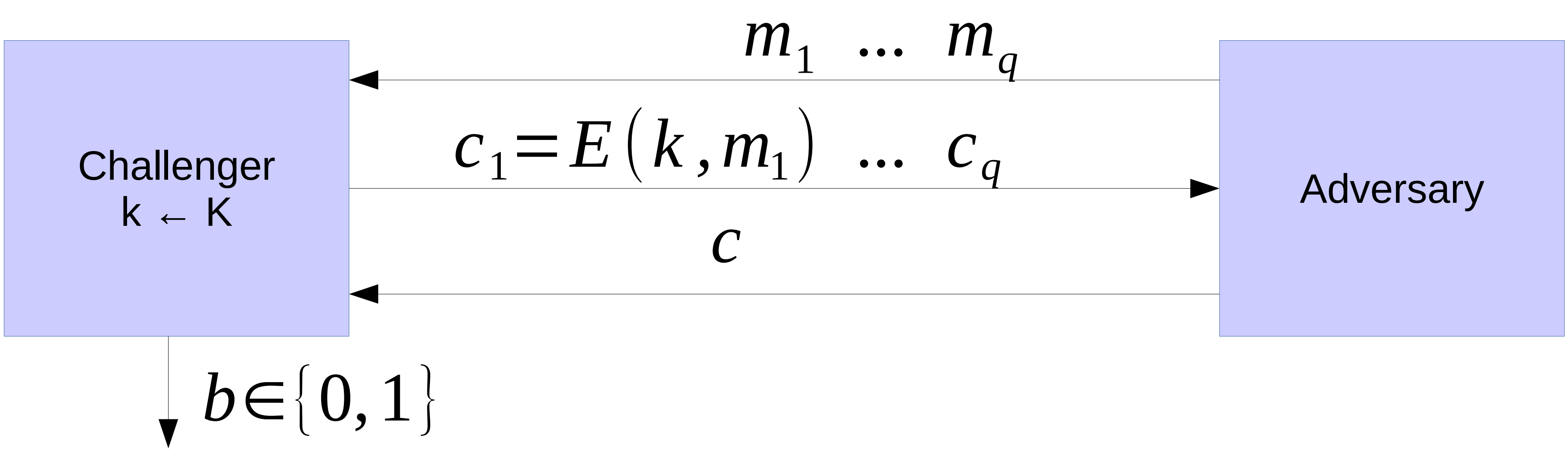
CT is invalid (rejected)

– Security: the system must provide

- **semantic security under CPA**, and
- **ciphertext integrity**
  - an adversary cannot create a new valid CT (such that would decrypt properly)

Ciphertext integrity (def)

Let $\zeta = (E, D)$ be a cipher with message space M



$$b = 1 \quad \text{if } D(k, c) \neq \perp \text{ and } c \notin \{c_1 \ldots c_q\}$$

$$b = 0 \quad \text{otherwise}$$

Def: $\zeta = (E, D)$ has **ciphertext integrity** if for all “efficient” adversaries A: $\text{Adv}_{\text{CI}}[A, \zeta]$ is “negligible”.

$$\text{Adv}_{\text{CI}}[A, \zeta] = \Pr[\text{Chal. outputs } 1]$$

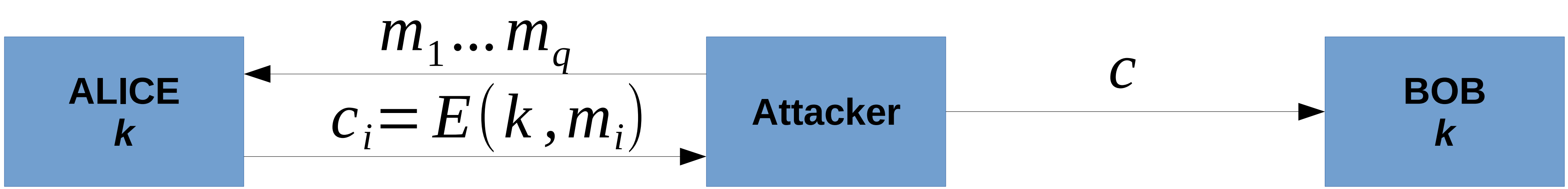
Authenticated Encryption

- Def: A cipher $\zeta = (E, D)$ **provides authenticated encryption (AE)** if it is
  1) semantically secure under CPA, and
  2) has ciphertext integrity.

- Do the following ciphers provide AE:
  - AES-CBC,
  - AES-CTR,
  - RC4?
- Why?

Authenticated Encryption

- Implication 1: Authenticity



  - An attacker cannot create a new valid $c \notin \{c_1 \ldots c_q\}$
  - If message decrypts properly $(D(k,c) \neq \perp)$, it must have come from someone who knows secret key *k*
    - But it could be a replay

- Implication 2: Security against **chosen ciphertext attack (CCA)**

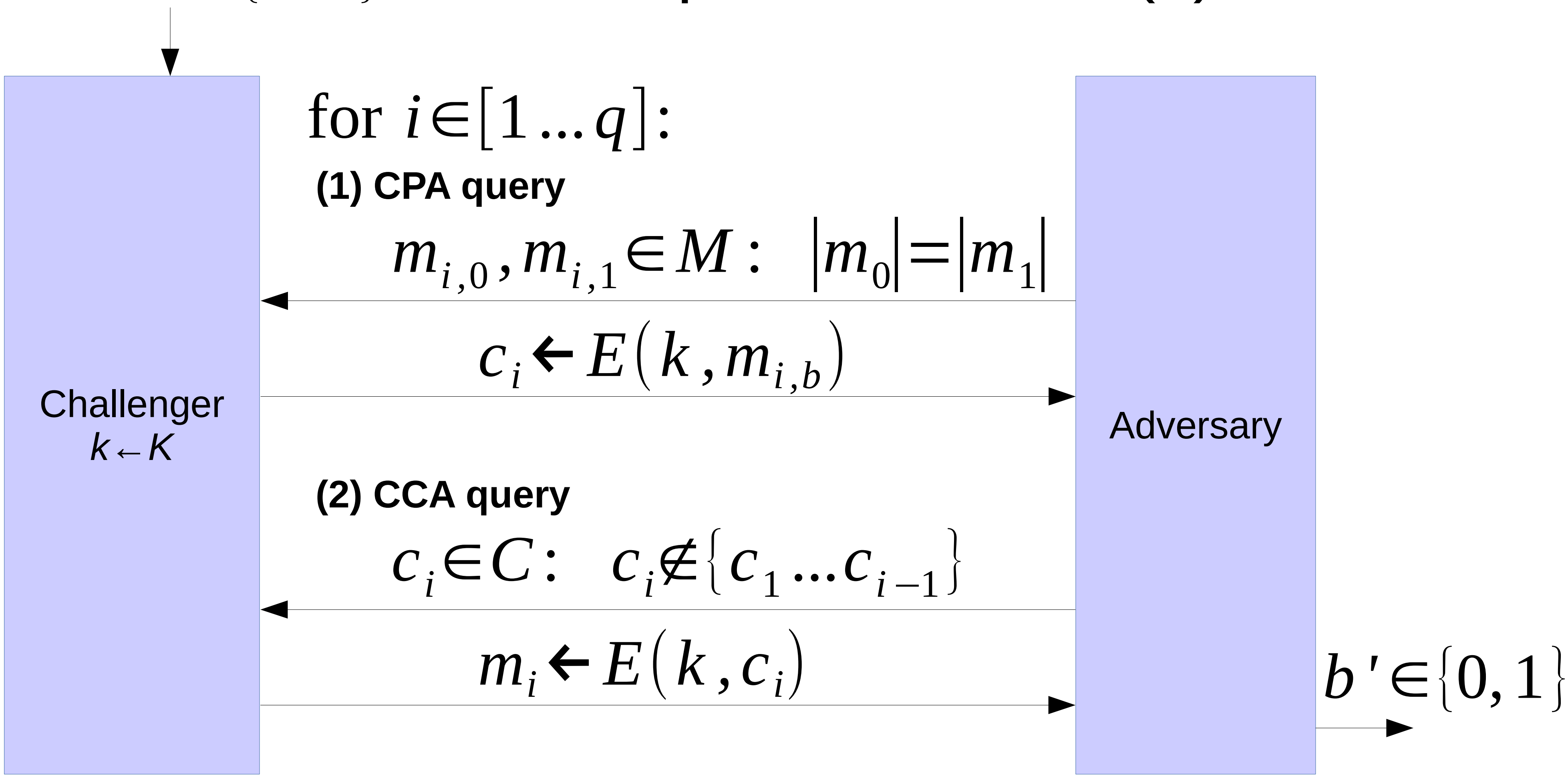
Adaptation of: Dan Boneh, Cryptography I, Stanford.

Chosen ciphertext security

- Adversary’s power: **CPA** and **CCA**
  - Can encrypt any message of her choice
  - Can decrypt any message of her choice *other than some challenge*
  - (still conservative modeling of real life)
- Adversary’s goal: **break semantic security**
  - Learn about the PT from the CT

Chosen ciphertext security (def)

- Let $\zeta=(E,D)$ be a cipher defined over (K,M,C)
- For $b\in\{0,1\}$ define experiments EXP(b) as



Chosen ciphertext security (def)

- Def. Cipher $\zeta=(E,D)$ is CCA secure if for all efficient adversaries A $\mathrm{Adv}_{\mathrm{CCA}}[A,\zeta]$ is negligible.

$$\mathrm{Adv}_{\mathrm{CCA}}[A,\zeta]:=\left|\Pr[\mathrm{EXP}(0)=1]-\Pr[\mathrm{EXP}(1)=1]\right|$$

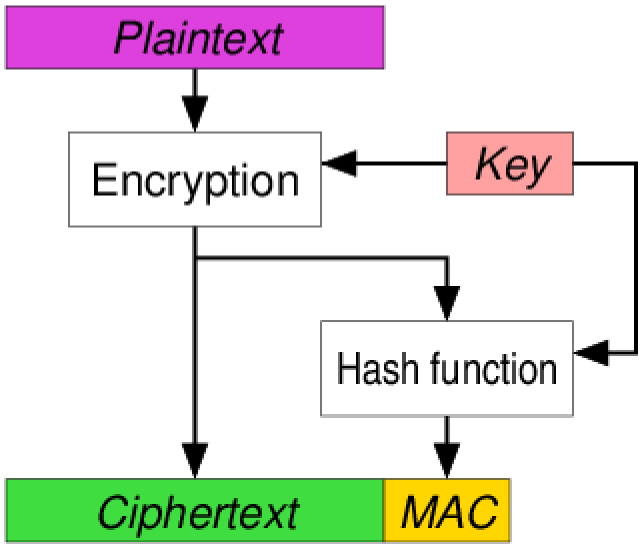
- Thm. A cipher that provides AE is also CCA secure.
- Implication. AE provides confidentiality against an active adversary that can decrypt some ciphertexts.
- Limitations
  - AE does not prevent replay attacks
  - Does not account for side channels attacks (timing)

Ex: AES-CTR is not CCA secure

- Recall
  - AES-CTR is effectively a stream cipher
  - Malleability of stream ciphers

Encrypt then MAC

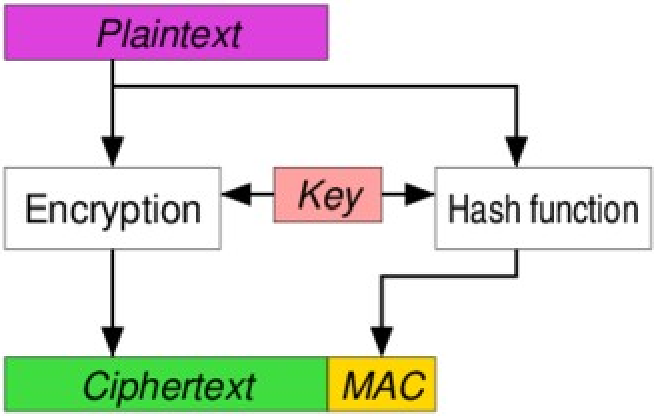
- MAC computed over cipher text
- Used in IPsec, always provides AE
  - Use separate and independent keys



https://en.wikipedia.org/wiki/Authenticated_encryption

Encrypt and MAC

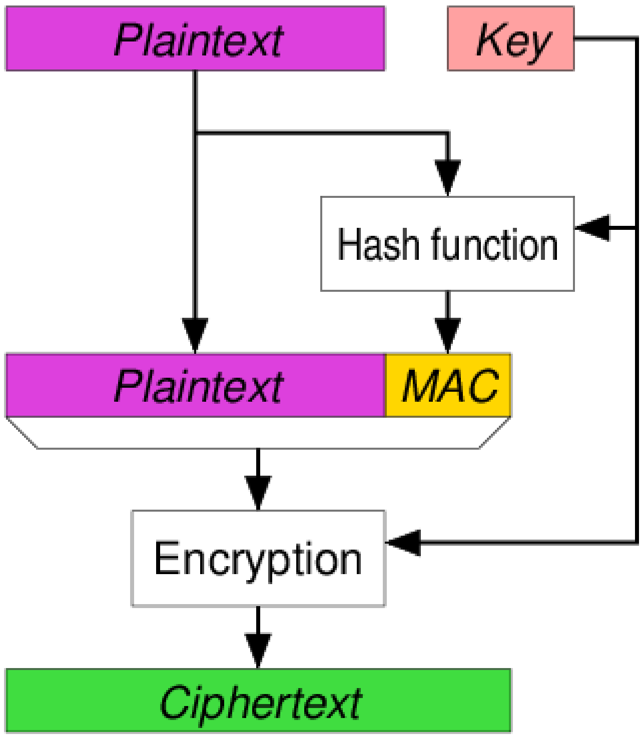
- MAC computed over plain text and sent unencrypted
- Used in SSH
- Use separate and independent keys



https://en.wikipedia.org/wiki/Authenticated_encryption

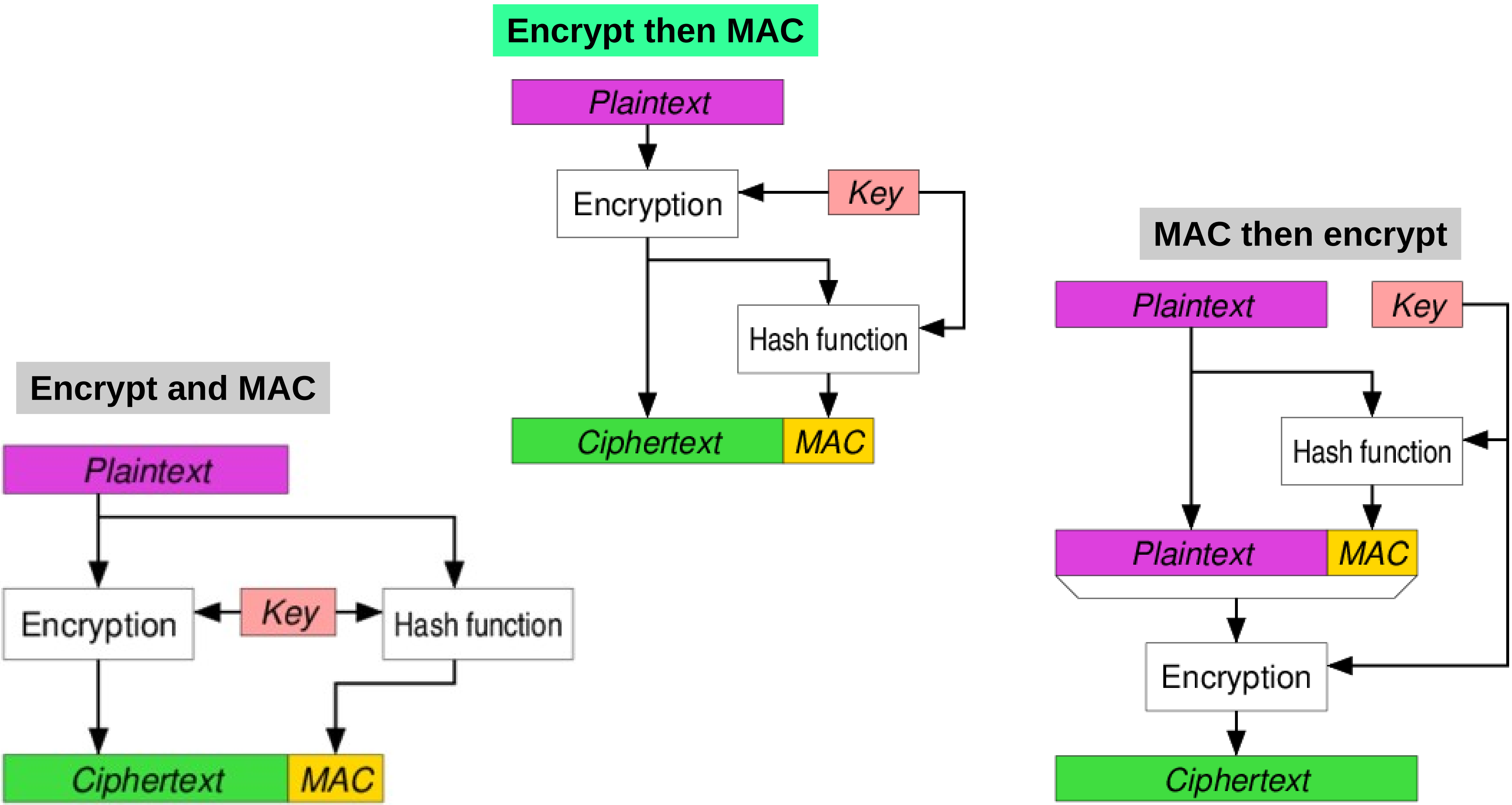
MAC then encrypt

- MAC computed over plain text and then encrypted before sending
- Used in TLS/SSL
- Use separate and independent keys



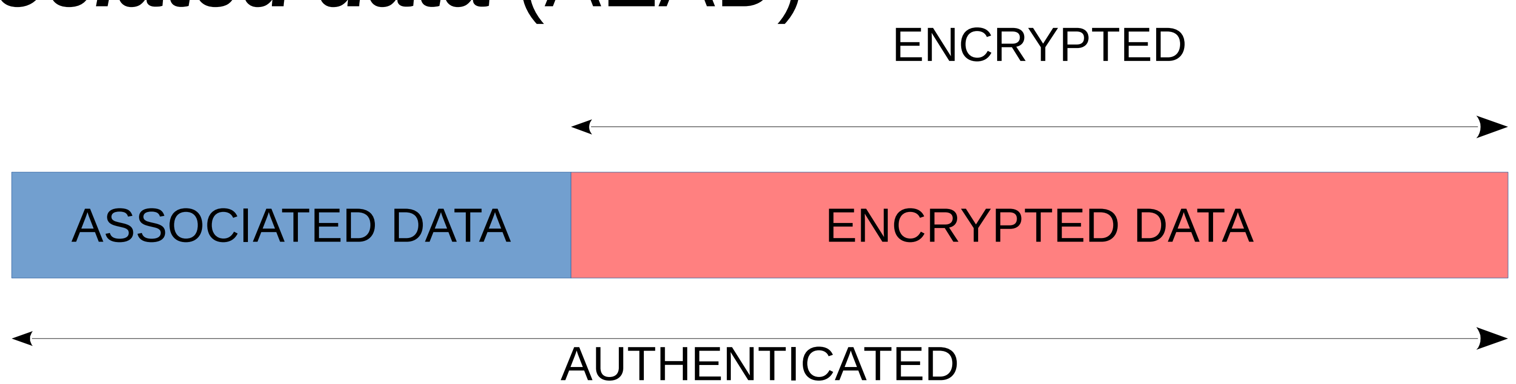
https://en.wikipedia.org/wiki/Authenticated_encryption

Three AE approaches



AE: Standardized solutions

- Galois/Counter Mode (GCM)
  - CTR mode encryption then CW-MAC
  - Made popular by Intel's PCLMULQDQ instruction
- CBC-MAC then CTR mode encryption (CCM)
- EAX
- All support ***authenticated encryption with associated data*** (AEAD)



Authenticated Encryption

Contents

- Ciphertext integrity
- AE definitions
- Chosen Ciphertext Attack
- Constructions
  - Encrypt-then-MAC
  - Encrypt-and-MAC
  - MAC-then-Encrypt

Authenticated Encryption (AE)

- Everything demonstrated so far provides
  - either <u>integrity</u>
  - or <u>confidentiality</u> (security against eavesdropping)
- CPA security does not provide secrecy against active attacks (where an attacker can tamper with ciphertext)
  - ➔ If you require <u>integrity</u> → **MAC**
  - ➔ If you require <u>integrity and confidentiality</u> → **AE**

AE: Desired properties

- An authenticated encryption system $\zeta = (E, D)$ is a cipher where

  as usual $E : K \times M \times N \to C$

  but $D : K \times C \times N \to M \cup \{\perp\}$ $\perp \notin M$

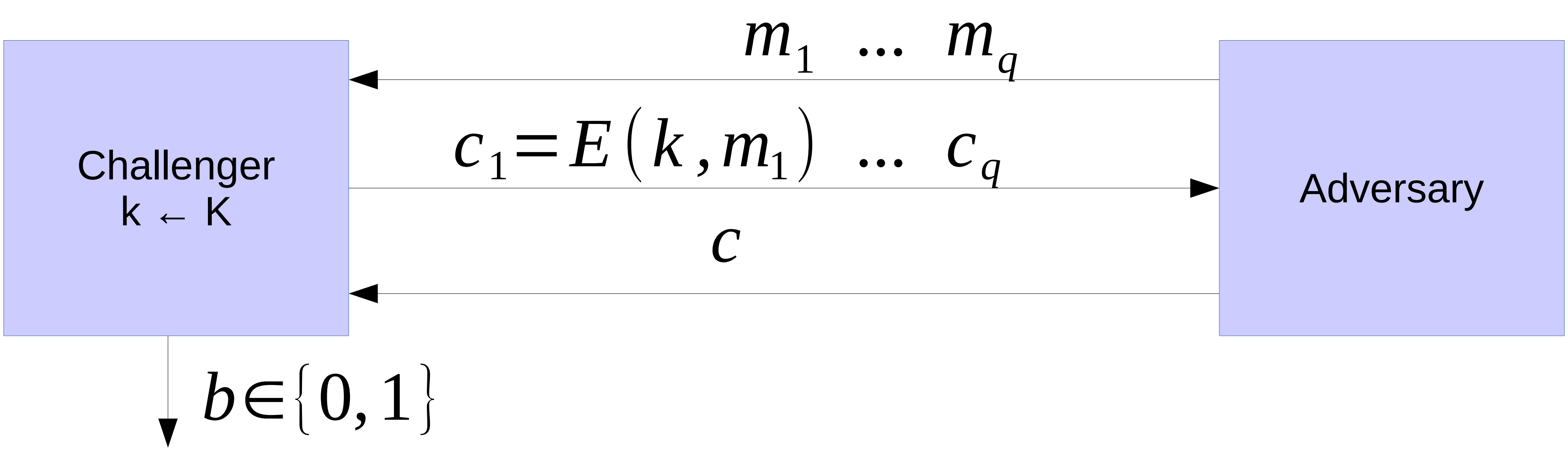
  Nonce

  CT is invalid (rejected)

- Security: the system must provide
  - **semantic security under CPA**, and
  - **ciphertext integrity**
    - an adversary cannot create a new valid CT (such that would decrypt properly)

Ciphertext integrity (def)

Let $\zeta=(E,D)$ be a cipher with message space M



$b=1$ if $D(k,c)\neq\perp$ and $c\notin\{c_1\ldots c_q\}$

$b=0$ otherwise

Def: $\zeta=(E,D)$ has **ciphertext integrity** if for all “efficient” adversaries A: $\mathrm{Adv}_{\mathrm{CI}}[A,\zeta]$ is “negligible”.

$$\mathrm{Adv}_{\mathrm{CI}}[A,\zeta]=\Pr[\text{Chal. outputs } 1]$$

Authenticated Encryption

- Def: A cipher $\zeta = (E, D)$ **provides authenticated encryption (AE)** if it is
  1) semantically secure under CPA, and
  2) has ciphertext integrity.

- Do the following ciphers provide AE:
  - AES-CBC,
  - AES-CTR,
  - RC4?
- Why?

Authenticated Encryption

- Implication 1: Authenticity

  - An attacker cannot create a new valid $c \notin \{c_1 \dots c_q\}$
  - If message decrypts properly $(D(k, c) \neq \perp)$, it must have come from someone who knows secret key *k*
    - But it could be a replay

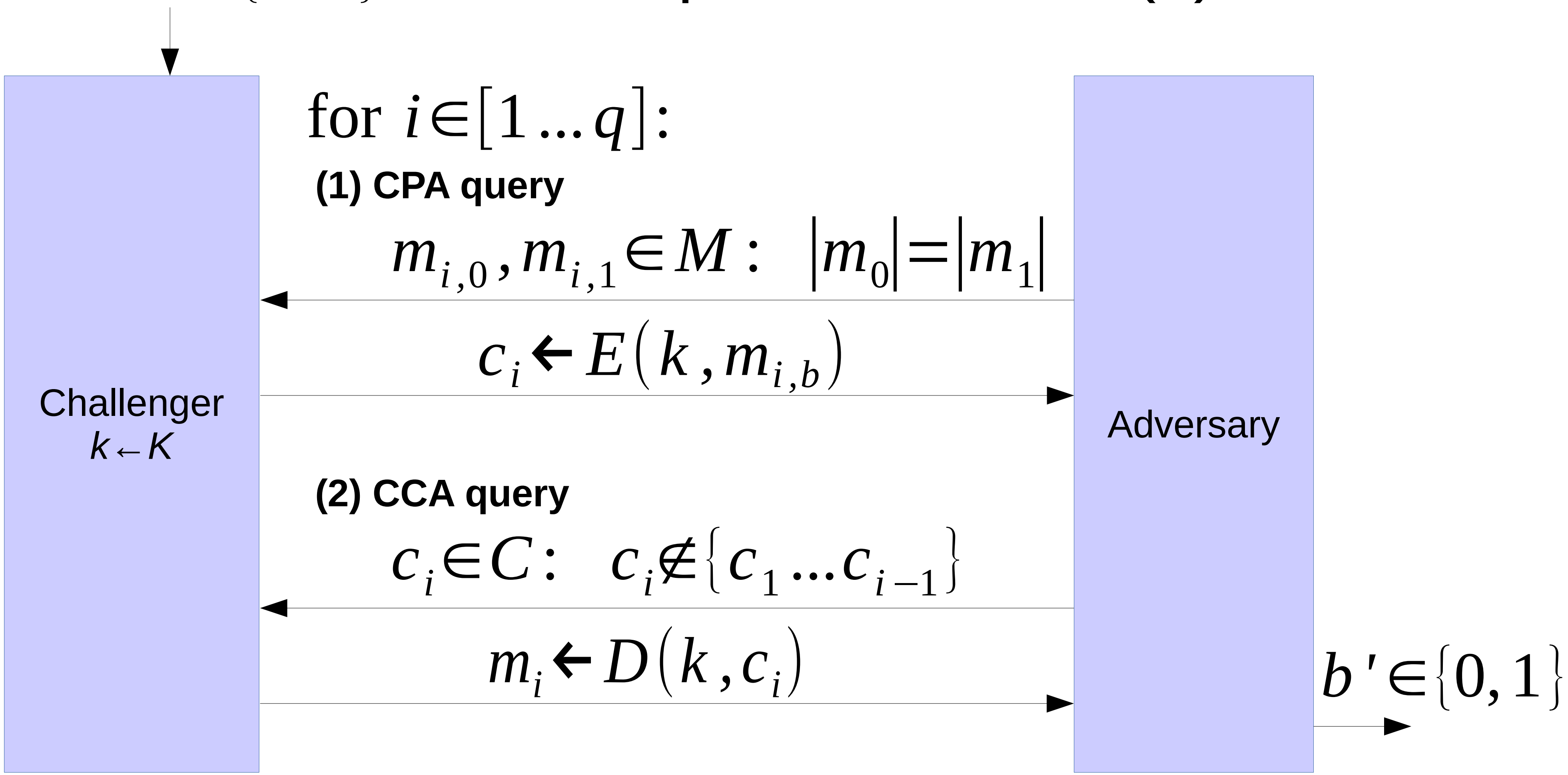
- Implication 2: Security against **chosen ciphertext attack (CCA)**

Chosen ciphertext security

- Adversary’s power: **CPA** and **CCA**
  - Can encrypt any message of her choice
  - Can decrypt any message of her choice *other than some challenge*
  - (still conservative modeling of real life)
- Adversary’s goal: **break semantic security**
  - Learn about the PT from the CT

Chosen ciphertext security (def)

- Let $\zeta = (E, D)$ be a cipher defined over (K, M, C)
- For $b \in \{0, 1\}$ define experiments EXP(b) as



Adaptation of: Dan Boneh, Cryptography I, Stanford.

Chosen ciphertext security (def)

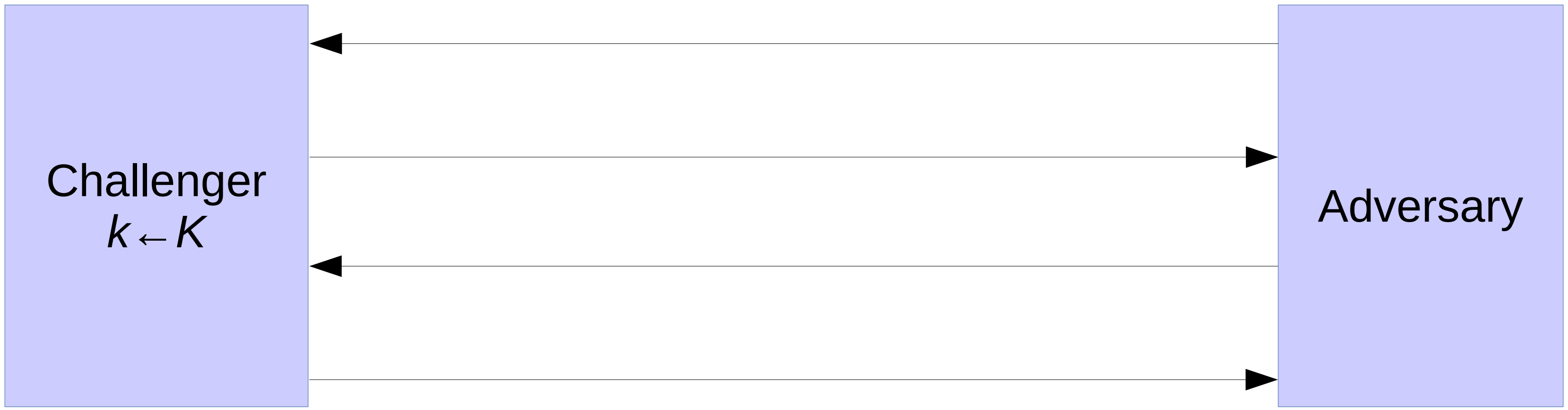
- Def. Cipher $\zeta = (E, D)$ is CCA secure if for all efficient adversaries A $\mathrm{Adv}_{\mathrm{CCA}}[A, \zeta]$ is negligible.

$$\mathrm{Adv}_{\mathrm{CCA}}[A, \zeta] := |\Pr[\mathrm{EXP}(0) = 1] - \Pr[\mathrm{EXP}(1) = 1]|$$

- Thm. A cipher that provides AE is also CCA secure.
- Implication. AE provides confidentiality against an active adversary that can decrypt some ciphertexts.
- Limitations
  - AE does not prevent replay attacks
  - Does not account for side channels attacks (timing)

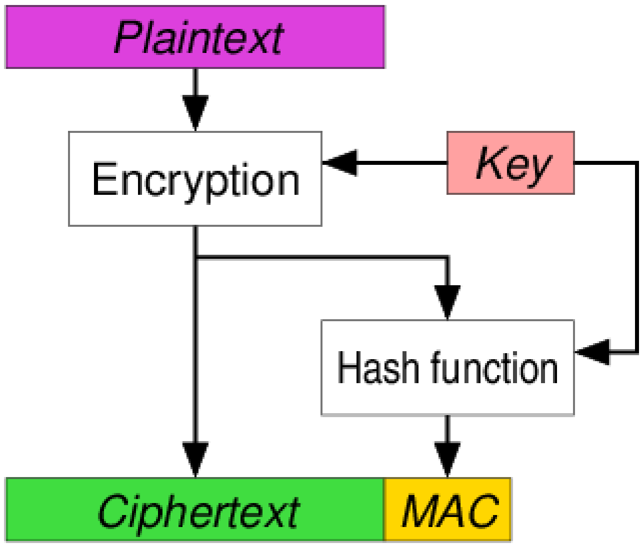
Ex: AES-CTR is not CCA secure

- Recall
  - AES-CTR is effectively a stream cipher
  - Malleability of stream ciphers



Encrypt then MAC

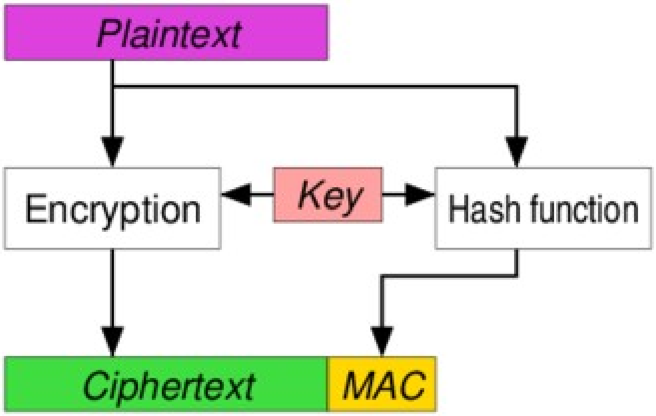
- MAC computed over cipher text
- Used in IPsec, always provides AE
  - Use separate and independent keys



https://en.wikipedia.org/wiki/Authenticated_encryption

Encrypt and MAC

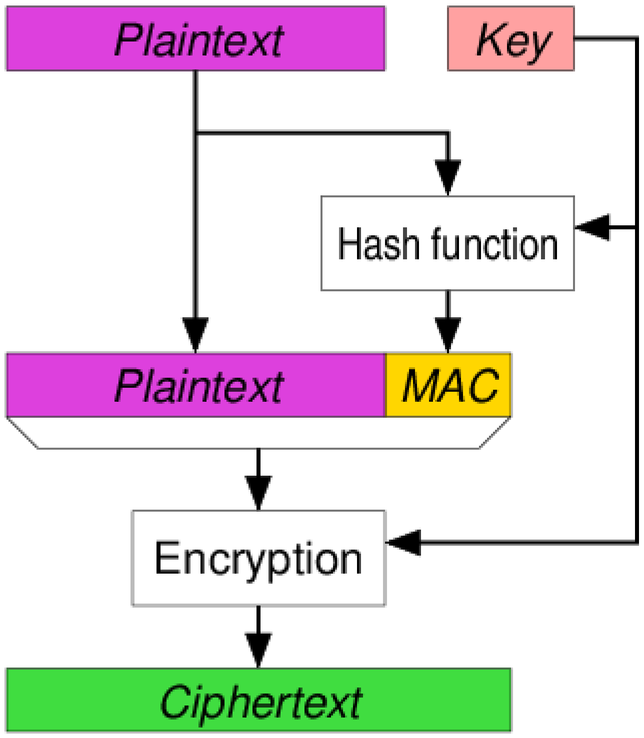
- MAC computed over plain text and sent unencrypted
- Used in SSH
- Use separate and independent keys



https://en.wikipedia.org/wiki/Authenticated_encryption

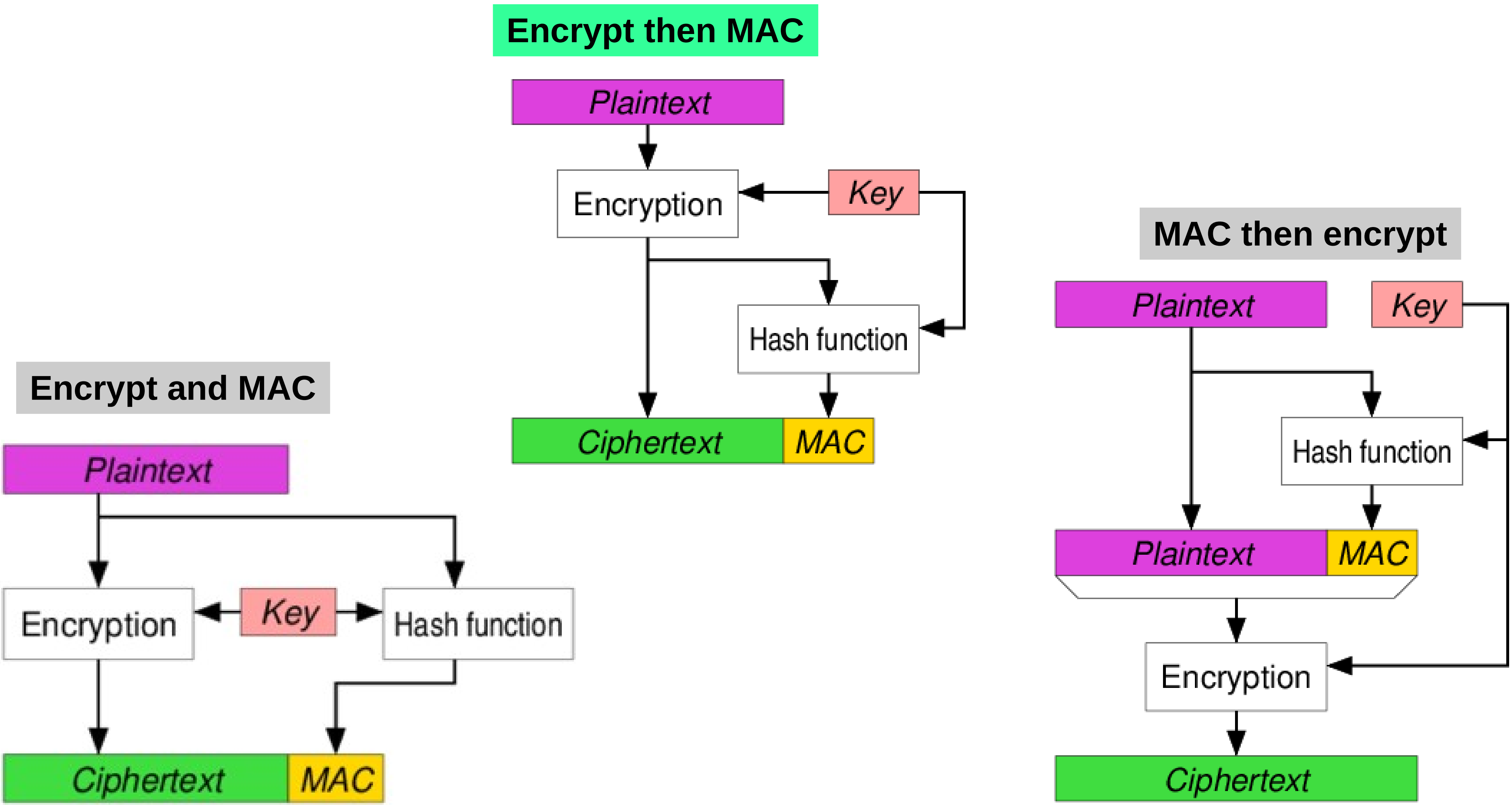
MAC then encrypt

- MAC computed over plain text and then encrypted before sending
- Used in TLS/SSL
- Use separate and independent keys



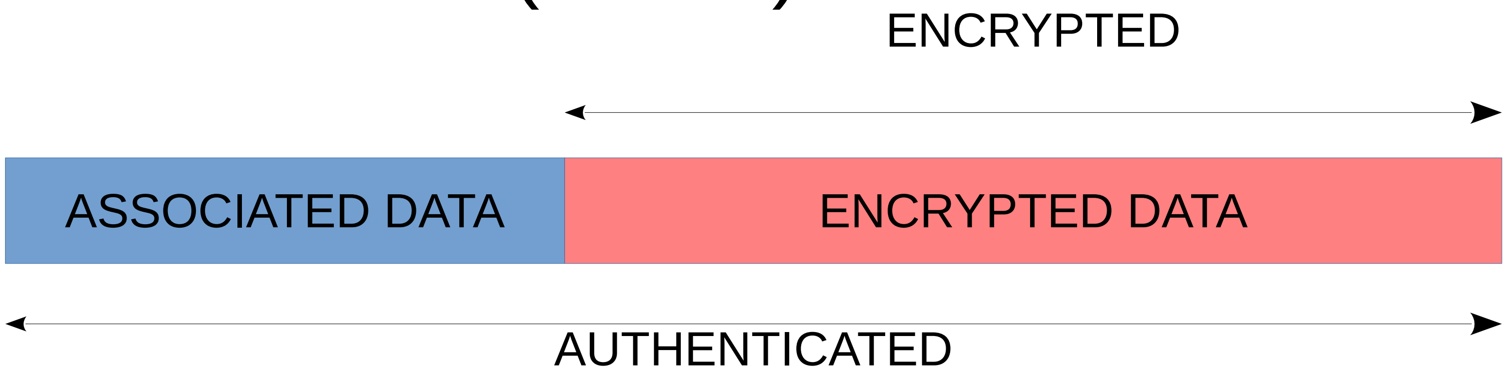
https://en.wikipedia.org/wiki/Authenticated_encryption

Three AE approaches



AE: Standardized solutions

- Galois/Counter Mode (GCM)
  - CTR mode encryption then CW-MAC
  - Made popular by Intel's PCLMULQDQ instruction
- CBC-MAC then CTR mode encryption (CCM)
- EAX
- All support ***authenticated encryption with associated data*** (AEAD)



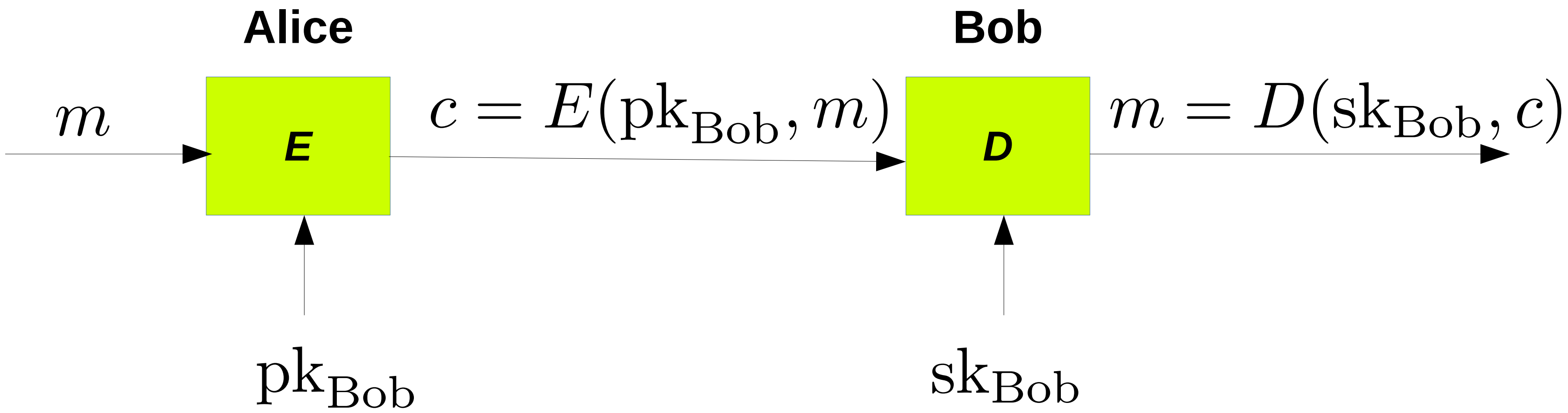
Public key encryption

Index

- Public-key ciphers overview
- Security definitions
  - CPA-security
  - CCA-security
- Trapdoor functions and permutations (TDF, TDP)
  - Encryption schemes from TDF (ISO)
- Example TDP: RSA
  - Definition
  - RSA in practice
  - Security of RSA

Public key encryption

- Each party uses a key pair: $k = (\mathrm{pk}, \mathrm{sk})$
- Public key is given to everyone, secret is kept hidden



Public key encryption: usage

- Communication session set-up
  - A process where Alice and Bob agree upon a shared secret
- Non-interactive applications
  - E.g. email
  - Typically, PKs are long-lived, symmetric keys are ephemeral
  - (But the sender needs to know recipient's PK in advance – need PKI)

Public key encryption: def

Def. A public-key encryption system is triple of algs. (G, E, D)

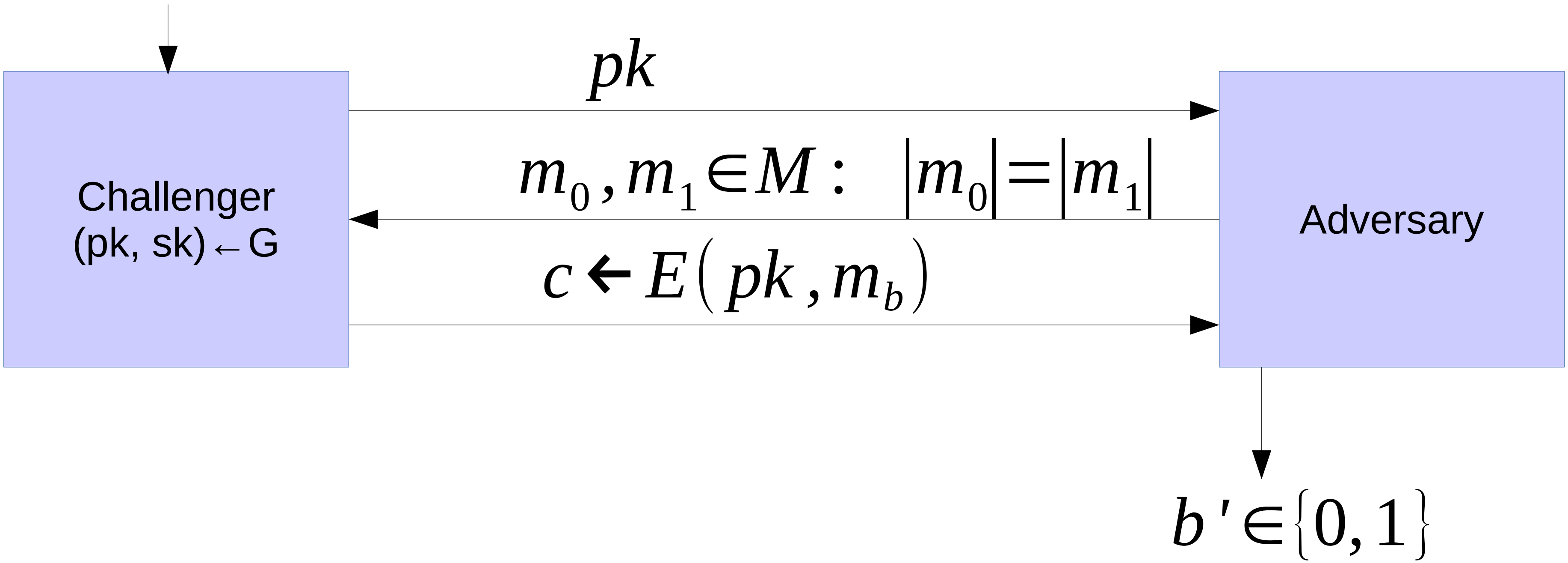
- $G()$ rand. alg. generates key pairs (pk, sk)
- $E(pk, m)$ rand. alg. takes $m \in M$ and returns $c \in C$
- $D(sk, c)$ det. alg. takes $c \in C$ and returns $m \in M$ or $\perp$

such that $\forall (pk, sk)$ output by G:

$$\forall m \in M : D(sk, E(pk, m)) = m$$

Semantic security (def)

Let $\zeta = (G, E, D)$ be a public key encryption system.
For $b \in \{0,1\}$ define experiments EXP(0), EXP(1)



Def: $\zeta = (G, E, D)$ is **semantically secure** (aka IND-CPA) if for all eff. adversaries $A: \mathrm{Adv}_{SS}[A, \zeta]$ is negligible.

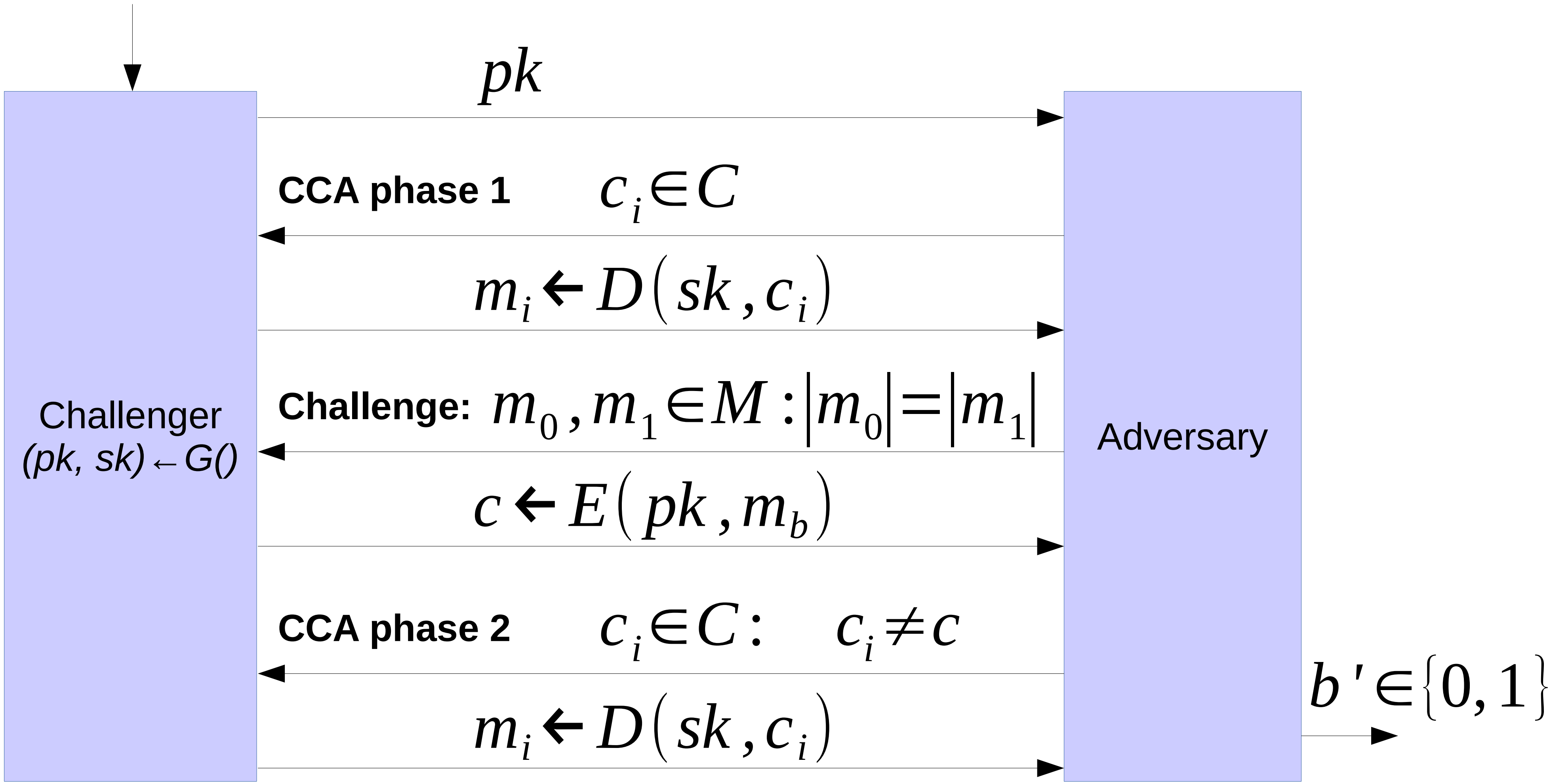
$$\mathrm{Adv}_{SS}[A, \zeta] := \left| \Pr[\mathrm{EXP}(0) = 1] - \Pr[\mathrm{EXP}(1) = 1] \right|$$

Relation to symmetric cipher security

- For symmetric ciphers, we had 2 security definitions
  - <u>One-time security</u> (key used only once) and <u>many-time security</u> (key used many times; CPA)
  - One-time security does not imply many-time security (OTP is broken if used more than once)
- Public key encryption
  - One-time security $\rightarrow$ many-time security (CPA)
    - Because the adversary can encrypt herself (she knows pk)
  - Public key encryption **must be randomized**

(pub-key) Chosen Ciphertext Security (def)

$\zeta = (G, E, D)$ a pub-key enc. over (M, C). For $b \in \{0,1\}$ define experiments EXP(b):



Adaptation of: Dan Boneh, Cryptography I, Stanford.

CCA security

- Def. $\zeta = (G, E, D)$ is CCA secure (aka. IND-CCA) if for all efficient adversaries A: $\mathrm{Adv}_{\mathrm{CCA}}[A, \zeta]$ is negligible.

$$\mathrm{Adv}_{\mathrm{CCA}}[A, \zeta] := \left|\Pr[\mathrm{EXP}(0) = 1] - \Pr[\mathrm{EXP}(1) = 1]\right|$$

- Recall: A secure symmetric cipher provides AE, when it has CPA security and ciphertext integrity
  - Attacker cannot create new ciphertexts (implies CCA security)
- In pub-key setting
  - Attacker knows pk $\rightarrow$ **can** create new ciphertexts
  - Instead: we directly require CCA security
- Next step: Constructing CCA secure pub-key encryption

Trapdoor function (TDF)

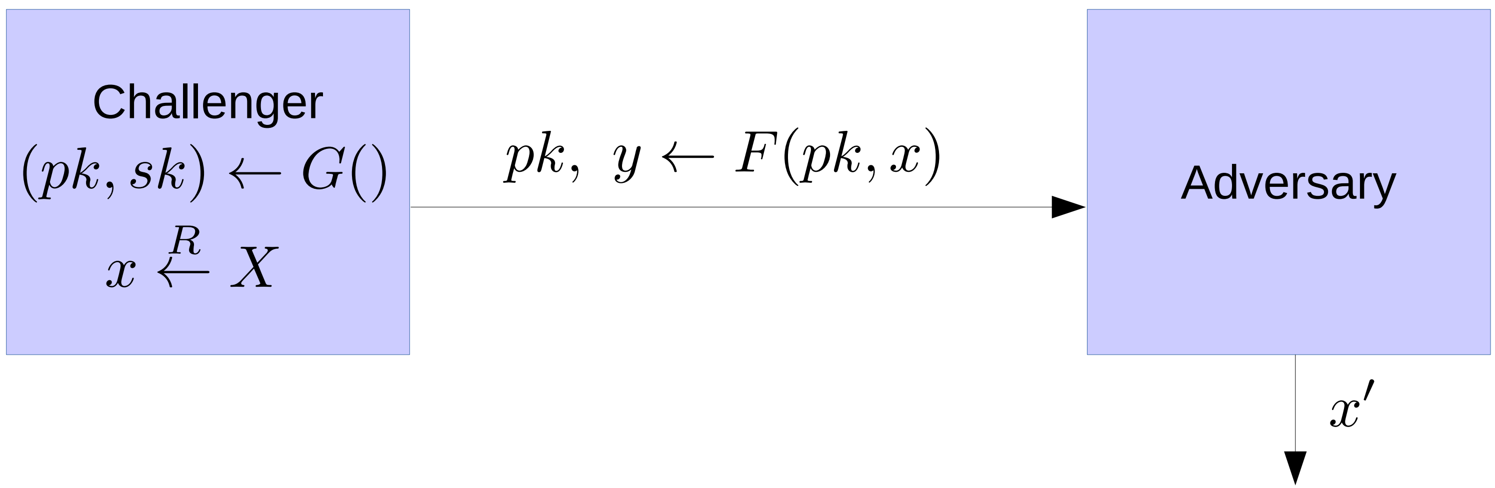
- **Def.** A trapdoor function $X \rightarrow Y$ is a triple of eff. algorithms (G, F, F^{-1})
  - **G()**: rand. alg. for creating (pk, sk)
  - **F(pk, -)**: <u>det. alg.</u> that defines $X \rightarrow Y$
  - **F^{-1}(sk, -)**: det. alg. that defines $Y \rightarrow X$ [inverts F(pk, -)]

For every (pk, sk) returned by **G**

$$F^{-1}[\ sk, F(pk, x)\ ] = x$$

Secure TDFs

- TDF (G, F, F^{-1}) is secure if F(pk, -) is *one-way*
  - It can be evaluated but not inverted without sk



- Def. (G, F, F^{-1}) is a secure TDF if for all eff. algs. A: $\mathrm{Adv}_{\mathrm{OW}}[A, F] := \Pr[x = x']$ is negligible.

Pub-key encryption from TDFs
(ISO 18033-2 standard)

- Building blocks
  - (G, F, F^{-1}) – secure TDF $X \to Y$
  - (E_S, D_S) – symmetric AE cipher over (K, M, C)
  - $H: X \to K$ – a hash function
- Pub-key enc. system **(G, E, D)**
  - Key generation **G**: same as **G** in TDF

E(pk, m):

$x \xleftarrow{R} X$, $\quad y \leftarrow F(pk, x)$
$k \leftarrow H(x)$, $\quad c \leftarrow E_s(k, m)$

return (y, c)

D(sk, (y, c)):

$x \leftarrow F^{-1}(sk, y)$
$k \leftarrow H(x)$, $\quad m \leftarrow D_s(k, c)$

return m

Pub-key encryption from TDFs
(ISO 18033-2 standard)

$F(pk, x)$	$E_S(H(x), m)$

Thm. If **(G, F, F^{-1})** is a secure TDF, if **(E_s, D_s)** provides AE, and if **H: X → K** is a "random oracle", then **(G, E, D)** is CCA^{ro} secure.

An incorrect use of TDF:

E(pk, m) := F(pk, m)

D(sk, c) := F^{-1}(sk, c)

Such construction results in a deterministic encryption scheme: cannot be semantically secure

Trapdoor permutation (TDP)

- TDP is a triple of eff. algorithms (G, F, F^{-1})
  - G(): generates (pk, sk); pk defines a function $X \to X$
  - F(pk, x): evaluates the function at x
  - F^{-1}(sk, y): inverts the function at y using sk

- **Secure** TDP

  The function F(pk, -) is one-way without the sk

Arithmetic modulo composites

Let $N = p \cdot q$ where p, q are primes

$$\mathbb{Z}_N = \{0, 1, ..., N-1\}$$

$$\mathbb{Z}_N^* = \{\text{invertible elements in } Z_N\}$$

Facts $\quad x \in \mathbb{Z}_N$ is invertible $\iff \gcd(x, N) = 1$

$$|\mathbb{Z}_N^*| = \varphi(N) = (p-1)(q-1) = N - p - q + 1$$

Euler's theorem

$$\forall x \in \mathbb{Z}_N^* : x^{\varphi(N)} = 1 \mod N$$

RSA trapdoor permutation

- G():
  - Choose random primes p, q (~1024 bits); $N = p \cdot q$
  - Choose integers e, d such that $e \cdot d = 1 \mod \varphi(N)$
  - Return $pk = (N, e),\ sk = (N, d)$
- F(pk, x): $\mathbb{Z}_N^* \to \mathbb{Z}_N^* : \mathrm{RSA}(x) = x^e \mod N$
- F^{-1}(sk, y):

$$\begin{aligned} y^d &= \mathrm{RSA}(x)^d && \mod N \\ &= x^{ed} && \mod N \\ &= x^{k \cdot \varphi(N) + 1} && \mod N \\ &= (x^{\varphi(N)})^k \cdot x && \mod N \\ &= x \end{aligned}$$

RSA trapdoor permutation

RSA assumption: RSA is one-way permutation

For all eff. algs. A:

$$\Pr[A(N,e,y) = \sqrt[e]{y}] < \text{negligible}$$

$$p, q \leftarrow n\text{-bit primes}$$

$$N = p \cdot q$$

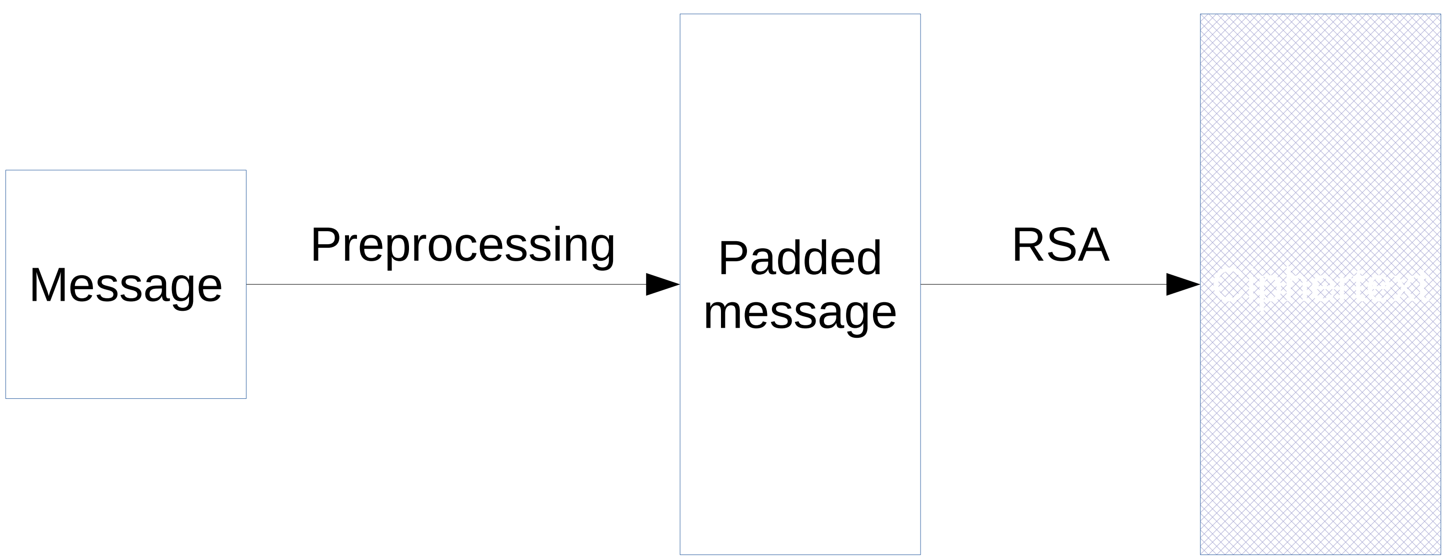
$$y \xleftarrow{R} \mathbb{Z}_N^*$$

Insecure “textbook” RSA

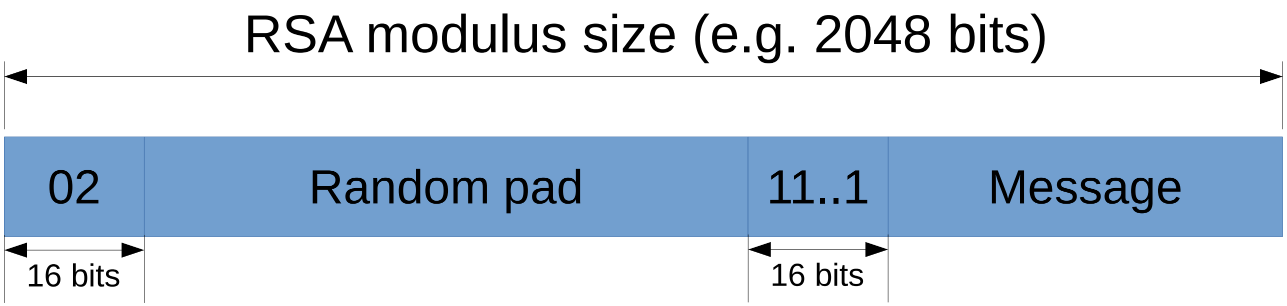
- Encrypting directly with RSA (“textbook” RSA) is insecure
  - $E((N,e),x) := x^e \mod N$
  - $D((N,d),y) := y^d \mod N$
- Problem 1: Ciphertext is **<u>malleable</u>**
  - Given ciphertext $c = E((N,e),m)$ an attacker can create $c' = c \cdot 2^e \mod N$
  - The modified ciphertext c' decrypts to $2m \mod N$
- Problem 2: Encryption is **<u>deterministic</u>**

RSA in practice

- RSA in practice (ISO standard rarely used)
  - Expand the message to the RSA modulus size and add random bits
  - Apply the RSA function



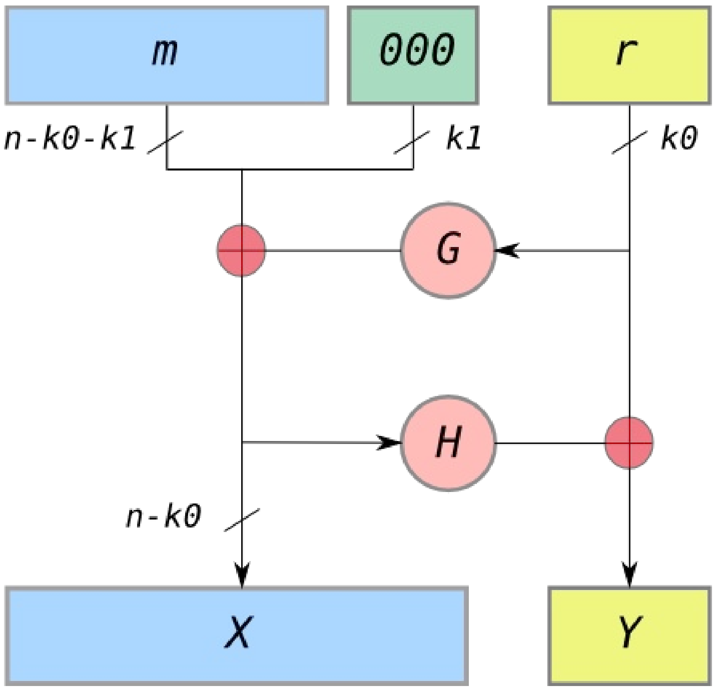
RSA in practice: PKCS1 v1.5



- Resulting value is RSA encrypted
- Widely deployed (HTTPS)
- Attack due to Bleichenbacher (1998)
  - During decryption, the system will signal an error if the decrypted plaintext does not start with 02
  - Enough to completely decrypt the ciphertext
- Solution in RFC 5246
  - set decrypted PT to a random value and *fail later on*
- Generally PKCS1 v1.5 padding should be avoided

RSA in practice: PKCS1: v2.0 (OAEP)

- New preprocessing function: **Optimal asymmetric encryption padding (OAEP)**
- Check pad on decryption
  - Reject CT if invalid
- **Thm.** If RSA is a TDP, then RSA-OAEP is CCA secure if H, G are *random oracles*.
  - In practice we use SHA-256 for H and G



https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding

RSA security (informally)

- To invert RSA one-way function, the attacker must extract x from $c = x^e \mod N$
- How difficult is to compute e’th root modulo N? Currently best known algorithm
  - Step 1: Factor N [difficult]
  - Step 2: Compute e’th roots modulo p and q [easy]
- Shor’s algorithm: a quantum algorithm for integer factorization in polynomial time
  - Unknown if quantum computers can be built

RSA security (informally)

- Security of public key system should be comparable to security of symmetric cipher

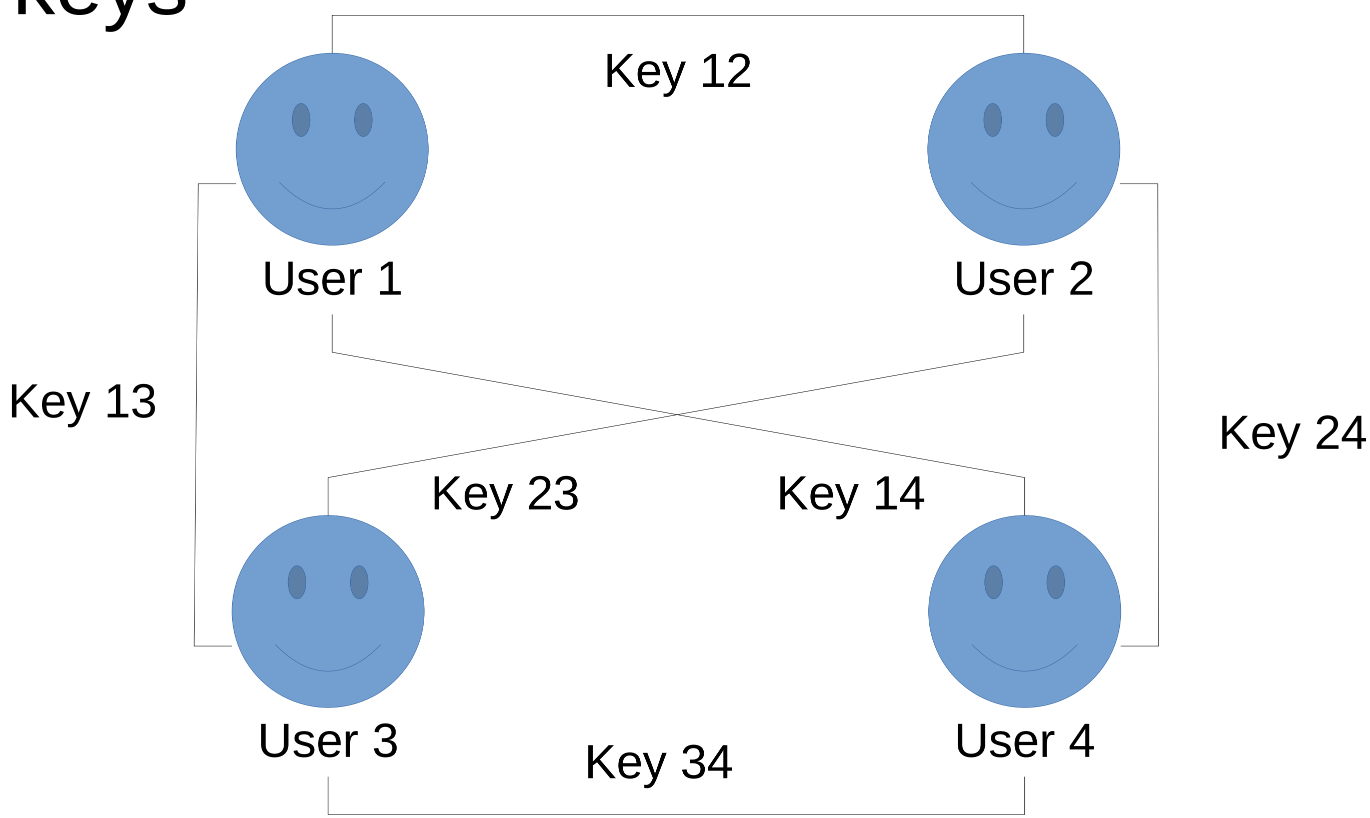
Cipher key size	**RSA modulus size [in modulo primes]**
80	1024
128	3072
256	15360

Key Exchange

Contents

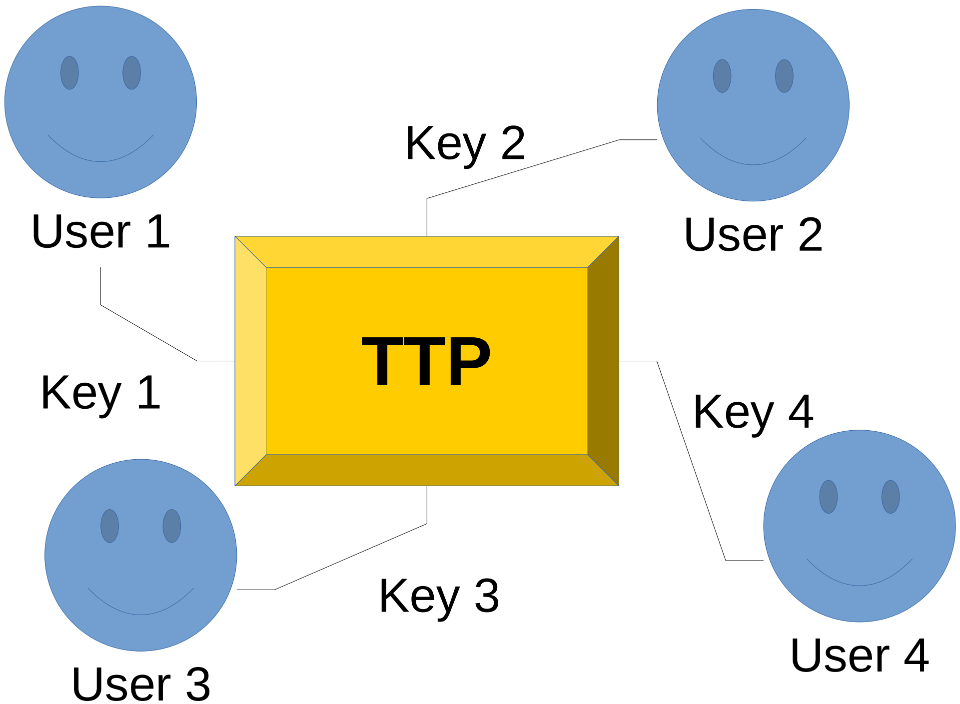
Key management

- Storing mutual secret keys is difficult
- In a universe of *n* users, each user requires *O(n)* keys



On-line Trusted Third Party (TTP)

- Every user has to manage only **a single key**
  - The one used to communicate with TTP
- Upon request, the TTP generates shared secret keys for user sessions



TTP: Generating keys (toy protocol)

- Bob wants a shared secret with Alice

Alice	Bob	TTP
k_A	k_B	
	A key for Alice, please. $\longrightarrow$	$k_{AB} \xleftarrow{R} \{0,1\}^{128}$
		$m_1 = E(k_B, \text{'B, A'} \| k_{AB})$
		$m_2 = E(k_A, \text{'B, A'} \| k_{AB})$
	$\xleftarrow{m_1, m_2}$	Ticket
$\xleftarrow{m_2}$		
k_{AB}	k_{AB}	

(E, D) a CCA secure cipher.

TTP: Security

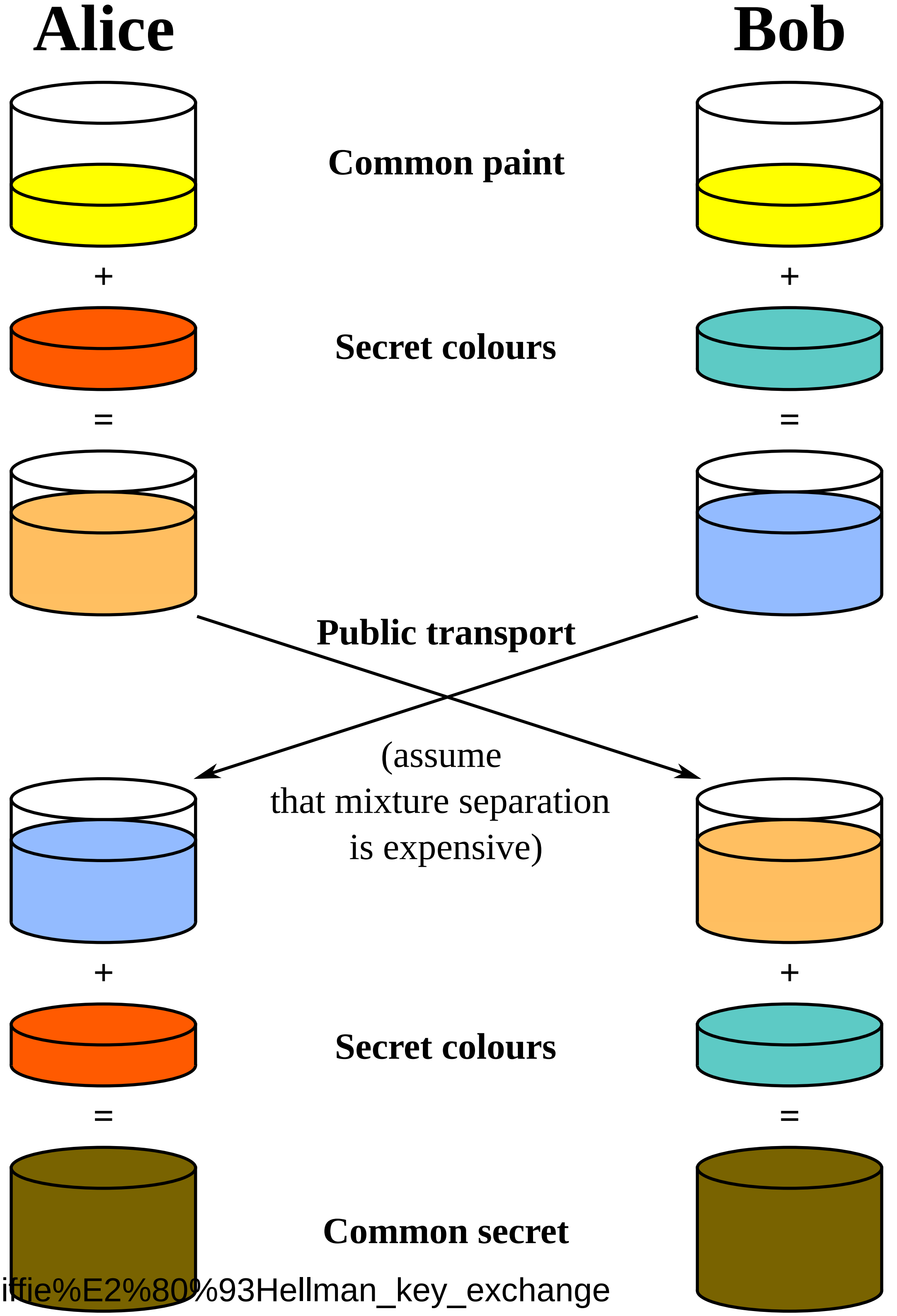
- An eavesdropper sees
  - $m_1 = E(k_B, \text{'B, A'} \| k_{AB})$
  - $m_2 = E(k_A, \text{'B, A'} \| k_{AB})$
- Since (E, D) is CCA secure, she learns nothing about k_{AB}
- Issues
  - TTP needed for all key exchanges
  - TTP knows all user and all session keys
  - Replay attacks possible
- Basis of Kerberos

The main issue

- Can we generate shared keys without an **on-line** TTP?
  - YES!
- Entrance of public-key cryptography
- Two most widely known constructions
  - Diffie-Hellman protocol (1976)
  - RSA crypto system (1977)

Diffie-Hellman protocol

- Stems from hard problems in algebra
- Alice an Bob want to establish a shared secret in the presence of an eavesdropper
- Security against eavesdropping only



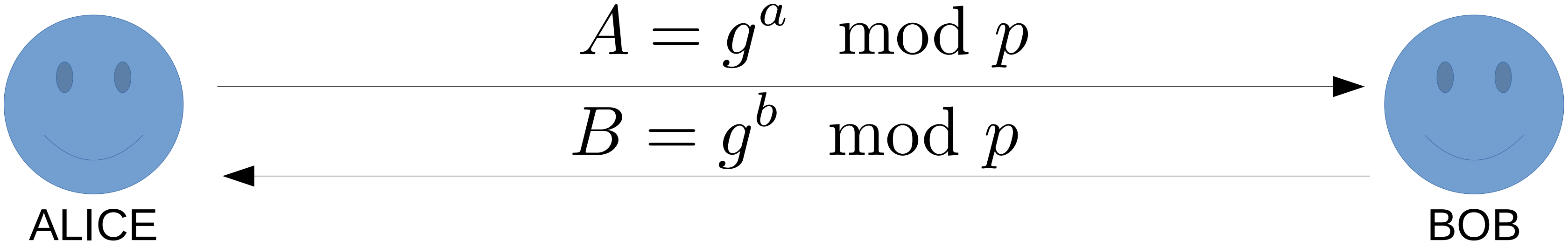
https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

Diffie-Hellman protocol (informally)

- Fix a large prime **p** (600 digits ~ 2kbits long)
- Fix an integer **g** in ***G*** = {1 … p-1} such that **g** is a primitive root modulo **p** (generator)
  - Raising **g** to powers of 0 to p-2 generates all values in {1 … p-1}

Picks a random **a** in {1 … p-1}

Picks a random **b** in {1 … p-1}



$$A = g^a \mod p$$

$$B = g^b \mod p$$

$$k_{AB} = g^{ab} \mod p$$

$$B^a = (g^b)^a = g^{ab} \mod p$$

$$A^b = (g^a)^b = g^{ab} \mod p$$

Adaptation of: Dan Boneh, Cryptography I, Stanford.

$5^0 = 1 \mod 23$

$5^1 = 5 \mod 23$

$5^2 = 2 \mod 23$

$5^3 = 10 \mod 23$

$5^4 = 4 \mod 23$

$5^5 = 20 \mod 23$

$5^6 = 8 \mod 23$

$5^7 = 17 \mod 23$

$5^8 = 16 \mod 23$

$5^9 = 11 \mod 23$

$5^{10} = 9 \mod 23$

$5^{11} = 22 \mod 23$

$5^{12} = 18 \mod 23$

$5^{13} = 21 \mod 23$

$5^{14} = 13 \mod 23$

$5^{15} = 19 \mod 23$

$5^{16} = 3 \mod 23$

$5^{17} = 15 \mod 23$

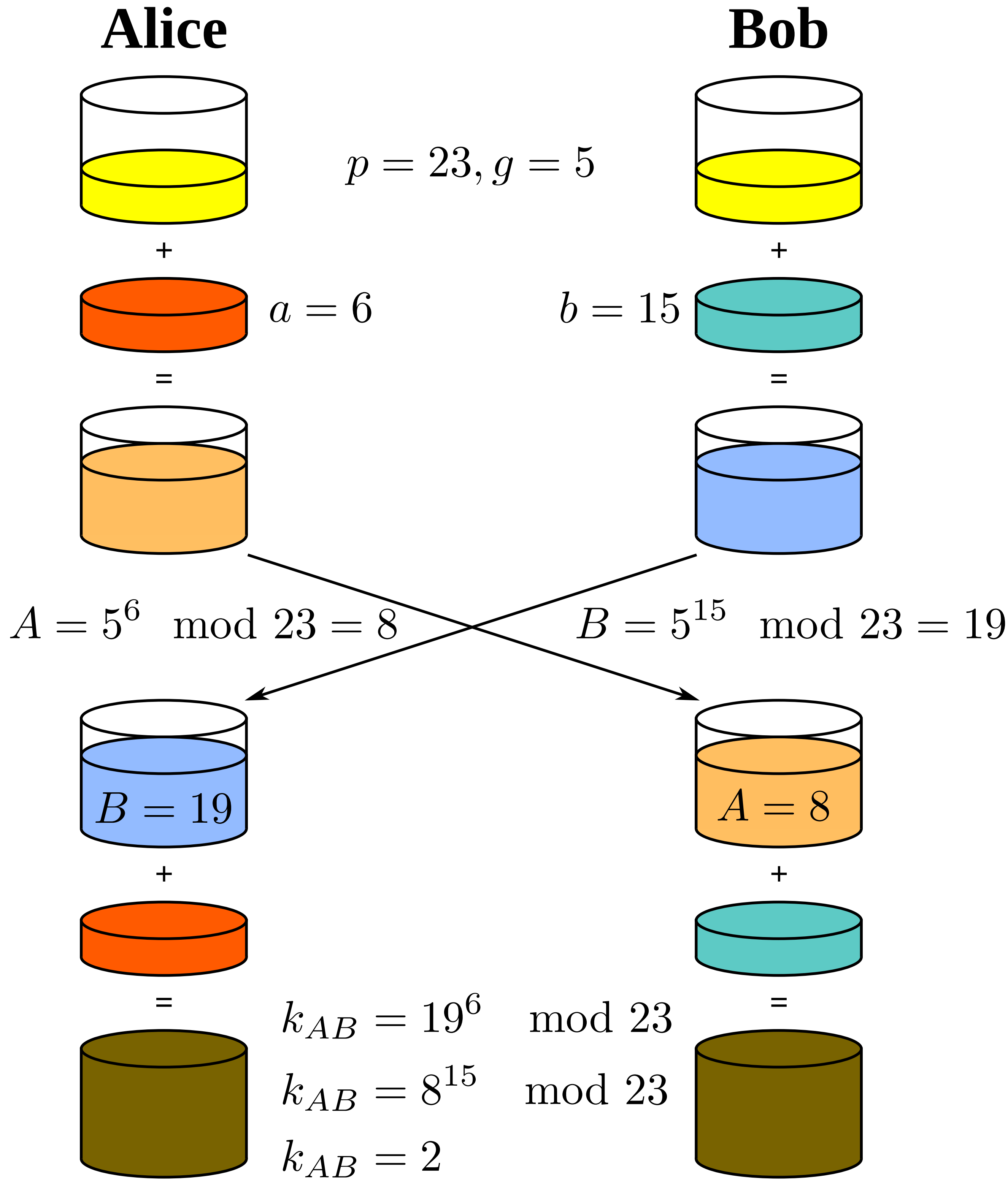
$5^{18} = 6 \mod 23$

$5^{19} = 7 \mod 23$

$5^{20} = 12 \mod 23$

$5^{21} = 14 \mod 23$

$5^{22} = 5^0 = 1 \mod 23$



Security (informally)

- An eavesdropper sees
  - $p, g, A = g^a (mod\ p), B = g^b (mod\ p)$
- Can she derive $g^{ab} (mod\ p)$ herself?

- In general, let's define

$$\mathrm{DH}_g(g^a, g^b) = g^{ab} \qquad (mod\ p)$$

- How difficult is to compute DH function (mod p)?

Security (informally)

- Suppose *p* is *n* bits long
- Best known algorithm (GNFS) computes function DH in $e^{O\left(\sqrt[3]{n}\right)}$
- How difficult is to break DH compared to breaking a symmetric cipher?

Cipher key size	DH modulus size [in modulo primes]	DH modulus size [Elliptic Curve]
80	1024	160
128	3072	256
256	15360	512

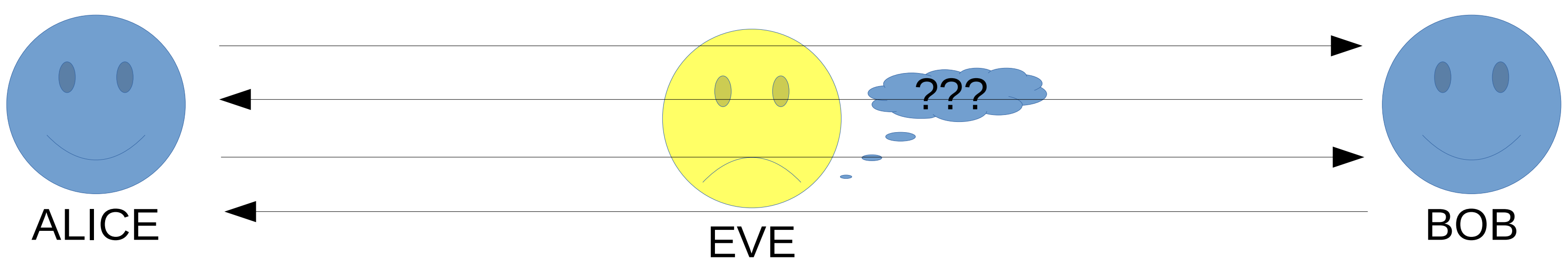
- Slow transition from (mod *p*) to elliptic curves

DH: Open issues

- Remember: security against eavesdropping only
- An active attacker can break the protocol with the man-in-the-middle attack
  - Reason: exchanges are not authenticated

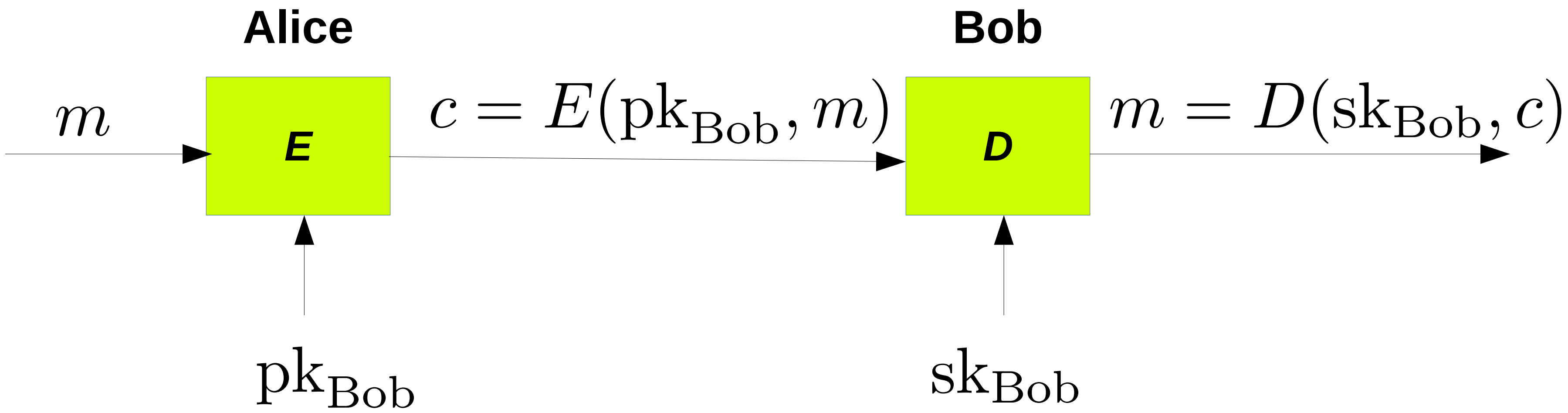
Public key encryption for key exchange

- Alice an Bob want to establish a shared secret in the presence of an eavesdropper
- Security against eavesdropping only

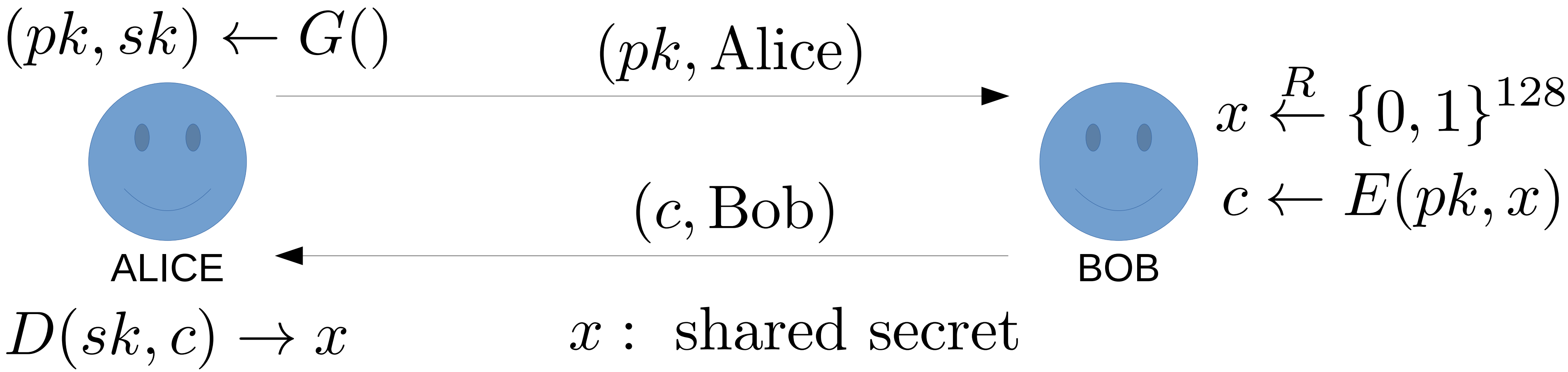


Public key encryption

- Each party uses a key pair: $k = (\mathrm{pk}, \mathrm{sk})$
- Public key is given to everyone, secret is kept hidden



Establishing a shared secret



- Adversary sees $pk,\; E(pk, x)$
- Adversary wants x
- If $\zeta = (G, E, D)$ is sem. secure, the adv. obtains no information about x
- Security **against eavesdropping only**: protocol still vulnerable to man-in-the-middle

Adaptation of: Dan Boneh, Cryptography I, Stanford.

Digital signatures

- Preserving integrity in public-key cryptography
  - "MACs" of public-key cryptography
- Idea: The signer signs a message with her secret key. Anyone can verify the signature using the corresponding public key and thus know:
  - That the message has not been tampered with
  - That the signer indeed signed the message
- Similar to MACs, but digital signatures are
  - **Publicly verifiable**: anyone (with PK) can verify the signature
  - **Non-repudiative**: the signer cannot later deny having signed a particular message

Signature scheme: def.

- **<u>Def:</u>** A signature scheme (G, S, V) is a triple of eff. algs. defined over (M, Z) where:
  - $G()$ is a rand. alg. that generates key pairs (pk, sk)
  - $S(sk, m)$ is an alg. that signs a message $m \in M$ using secret key sk and produces a signature $z \in Z$
  - $V(pk, m, z)$ is a det. alg. that verifies the signature $z \in Z$ of message $m \in M$ using pk and outputs **1** if the signature verifies, or **0** otherwise
  - A signature generated by *S* must always verify by *V*:

    $$\forall (pk, sk), m \in M : \Pr[V(pk, m, S(sk, m)) = 1] = 1$$

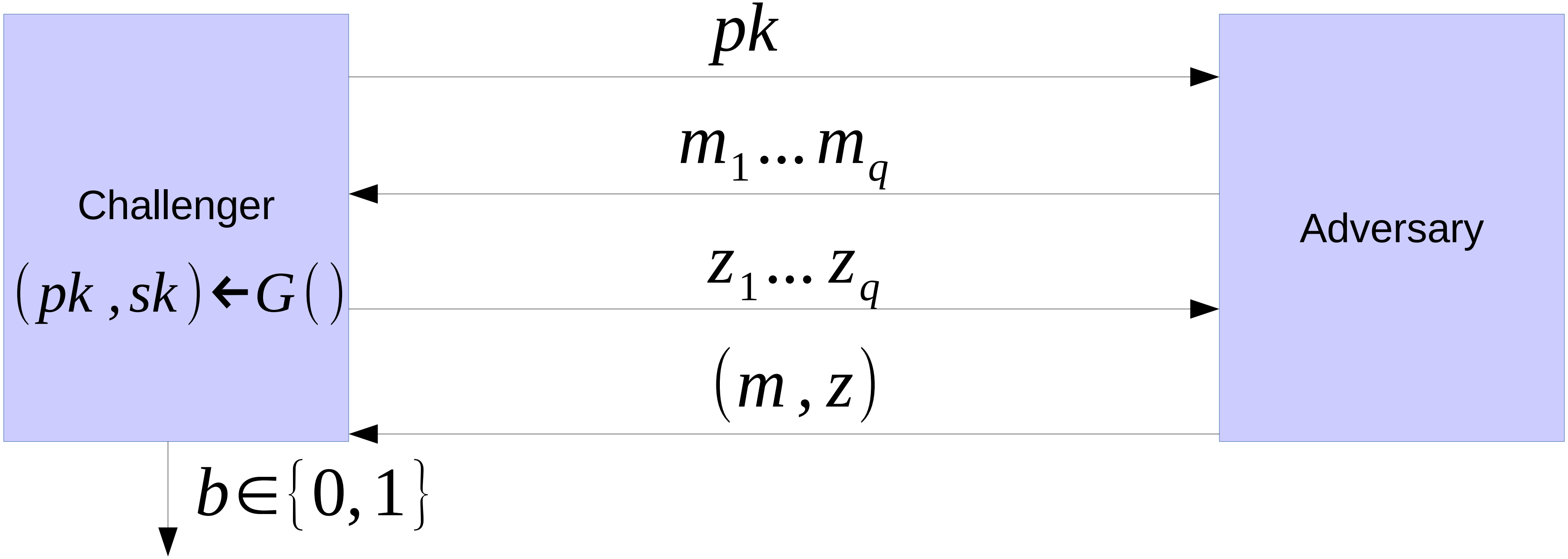
Digital signatures: Threat model

- Attacker's power: **Chosen message attack**
  - For $m_1 \ldots m_q$ attacker is given $z_i = S(sk, m_i)$
- Attacker's goal: **Existential forgery**
  - Produce a **new** valid (m, z) s. t.

$$m \notin \{m_1 \ldots m_q\}$$

$\rightarrow$ *An adversary cannot produce a valid signature for a new message*

Secure digital signature: def.



$b = 1 \quad \text{if } V(pk, m, z) = 1 \text{ and } m \notin \{m_1 \ldots m_q\}$

$b = 0 \quad \text{otherwise}$

A signature scheme (G, S, V) is **secure** if for all "efficient" adversaries A:

$$\mathrm{Adv}_{\mathrm{SIG}}[A, I] = \Pr[\text{Chal. outputs } 1]$$

is "negligible".

Extending the message space

- **Hash-and-sign paradigm**
  - *Constructing a signature scheme for large messages from a signature scheme for small messages (and strengthening security)*
- **Thm.** Let (G, S, V) be a secure signature scheme over (M, Z) and let $H : M' \rightarrow M$ be a collision resistant hash function where $|M'| \gg |M|$. Then (G, S', V') is also secure sig. scheme, where:

$$S'(sk, m) := S(sk, H(m))$$

$$V'(pk, m, z) := V(pk, H(m), z)$$

Signatures from TDP: Full Domain Hash

- Building blocks
  - ***(G, F, F^{-1})*** – Secure trapdoor permutation (TDP)
    - ***F: X→X***
  - ***H: M→X*** – collision resistant hash function
- Full domain (length) hash (FDH)
  - ***G()*** from TDP
  - $S(sk, m) := F^{-1}(sk, H(m))$
  - $V(pk, m, z) := \begin{cases} 1 & H(m) = F(pk, z) \\ 0 & \text{otherwise} \end{cases}$

Signatures from TDP: Full Domain Hash

- **Thm.** Let $\boldsymbol{(G, F, F^{-1})}$ be a secure TDP $\boldsymbol{X} \rightarrow \boldsymbol{X}$ and let $\boldsymbol{H: M} \rightarrow \boldsymbol{X}$ be a collision resistant hash function. Then signature scheme FDH is secure if $\boldsymbol{H}$ is a *random oracle*.
- FDH produces <u>unique signatures</u>: every message has its own signature

Signatures from TDP: Full Domain Hash

- Hashing is required for security; schemes without hashing are insecure. For instance:

$$S(sk, m) := F^{-1}(sk, m) \qquad V(pk, m, z) := F(pk, z) == m$$

  – **Zero-message attack**: create an existential forgery by picking a random signature, and creating a "message" from it

$$z \xleftarrow{R} Z, m \leftarrow F(pk, z)$$

  – **Multiplicative-property attack** (when using RSA)

    - Ask for signatures on two messages $m_{1,} m_2$

$$z_1 \leftarrow S(sk, m_1), z_2 \leftarrow S(sk, m_2)$$

    - Output existential forgery

$$m_3 \leftarrow m_1 \cdot m_2$$
$$z_3 \leftarrow z_1 \cdot z_2$$

Signatures from RSA trapdoor

- ***G()***
  - Choose random primes p, q (~1024 bits); $N = p \cdot q$
  - Choose integers e, d such that $e \cdot d = 1 \mod \varphi(N)$
  - Return $pk = (N, e),\ sk = (N, d)$
- $S((N,d),m) := H(m)^d \mod N$
- $V((N,e),m,z) := \begin{cases} 1 & H(m) = z^e \mod N \\ 0 & \text{otherwise} \end{cases}$
- What about ***H***?

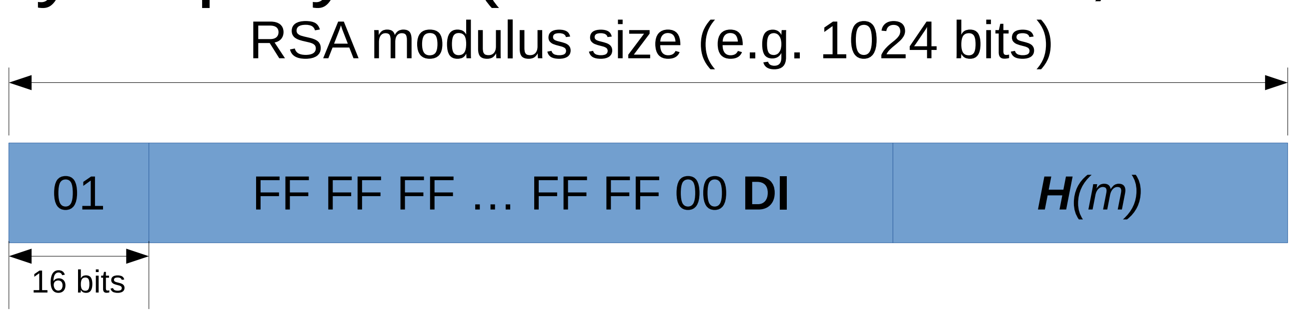
RSA Full Domain Hash

- We require ***H: M*** $\rightarrow \mathbb{Z}_N^*$
  - The output length of ***H*** depends on *N*; could be different for every public key
  - Ideally we want the output length of ***H*** to be fixed
- **Thm.** Let ***H: M***$\rightarrow$***Y*** be a collision resistant hash function where ***Y*** = *{1, …, 2^{n-2}}* and *n* is the number of bits used to represent *N*. Then RSA-FDH is secure sig. scheme if ***H*** is a random oracle.

$\rightarrow$ The bit-length of digests must be of similar length as is the bit-length of the modulus $|Y| \geq N/4$

PKCS1 v1.5 signatures

- Widely deployed (TLS certificates, S/MIME, ...)



- **DI** – digest info encodes the name of the used hash function ***H*** (SHA*, MD*, …)
- The resulting value is then signed by raising it to *d* in *mod N* (recall, *sk = (N, d)*)
- <u>Not FDH</u>, but partial domain hash
  - No security proof; also no known substantial attacks
    - Issue with proving: ***H***(m) maps to a small subset of $\mathbb{Z}_N^*$

Probabilistic Signature Scheme (PSS)

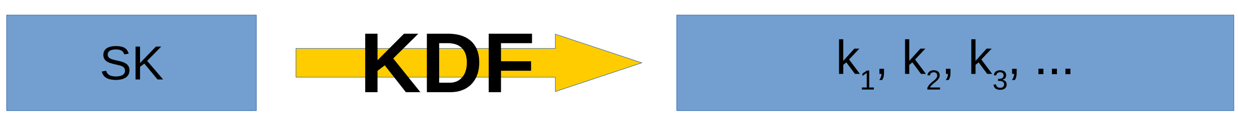
- Randomizes the signature with a public random value *s* called **salt**
- $S((N,d),m,s) := [H(s||m) \;||\; MGF[H(s||m)] \oplus s]^d \mod N$
  - MGF – mask generating function that extends the hash size to the full modulus size
- $V((N,e),m,z,s) := \begin{cases} 1 & H(s||m) \;||\; MGF[H(s||m)] \oplus s = z^e \mod N \\ 0 & \text{otherwise} \end{cases}$
  - Provably secure in random oracle model
  - Part of PKCS1 v2.1

Digital Signature Standard (DSS)

- NIST (FIPS 186)
  - Also called Digital Signature Algorithm (**DSA**)
- Relies on the hardness of Dlog
- No known proof of security
  - But also no serious attacks found
- Has an equivalent in elliptic curves (**ECDSA**)

Deriving many keys from one

- **Scenario**: we obtain a single <u>source key</u> (SK)
  - From a hardware random number generator
  - From a key exchange protocol
- We need many keys to secure the session
  - Unidirectional keys, MAC/encryption keys
- **Goal**: generate many keys from a single SK
  - KDF – key derivation function



Adaptation of: Dan Boneh, Cryptography I, Stanford.

Deriving many keys from one

- Three cases
  1) SK is uniform in key space
  2) SK is non-uniform in key space
  3) SK is a password

$$SK \xrightarrow{\textbf{KDF}} k_1, k_2, k_3, \ldots$$

Key derivation: (1) SK is uniform

- Let PRF F: K × X → $\{0, 1\}^n$
- If source key is **<u>uniform</u>** in K:

$$KDF(sk, ctx, l) := F(sk, ctx||0) \; || \; F(sk, ctx||1) \; || \; ... \; || \; F(sk, ctx||l)$$

- **ctx**: a string unique to every application
  - Assures that two applications derive independent keys even if they sample the same source key

Key derivation: (2) SK is non-uniform

- The KDF can be directly used <u>only when SK is uniform</u>

  → If SK is not uniform, the PRF output may not look random

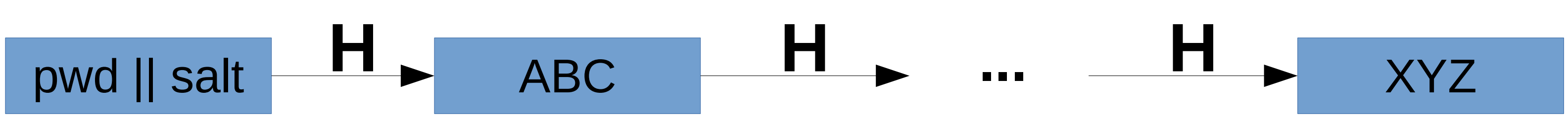
- Reasons for non-uniformity of SK
  - Hardware RNG may be *biased*
  - Key-exchange protocol may produce a key that is *uniform in some subset* of K

Key derivation: (2) SK is non-uniform

- **Extract-then-Expand paradigm**
  - Step 1) Use an **extractor** and **SK** to extract a pseudo-random key **k** that is uniform in key space
    - Use **salt:** a fixed public (non-secret) random string
  - Step 2) expand ***k*** with KDF
- **HKDF** – a KDF from HMAC
  - Step 1) ***k***$\leftarrow$HMAC(salt, SK)
  - Step 2) Expand as you would with uniform keys, but use HMAC for PRF and ***k*** for key
  - https://tools.ietf.org/html/rfc5869

Key derivation: (3) SK is a password

- Particular care needed when deriving keys from passwords
  - HKDF unsuitable here: passwords have low entropy
  - Derived keys will be vulnerable to dictionary attack
- General idea: add **salt** and **slow down hashing**
- **PBKDF** – password-based KDF
  - PKCS #5 v2.0 and https://tools.ietf.org/html/rfc2898
  - Iterate hash function many times



Adaptation of: Dan Boneh, Cryptography I, Stanford.

Final words

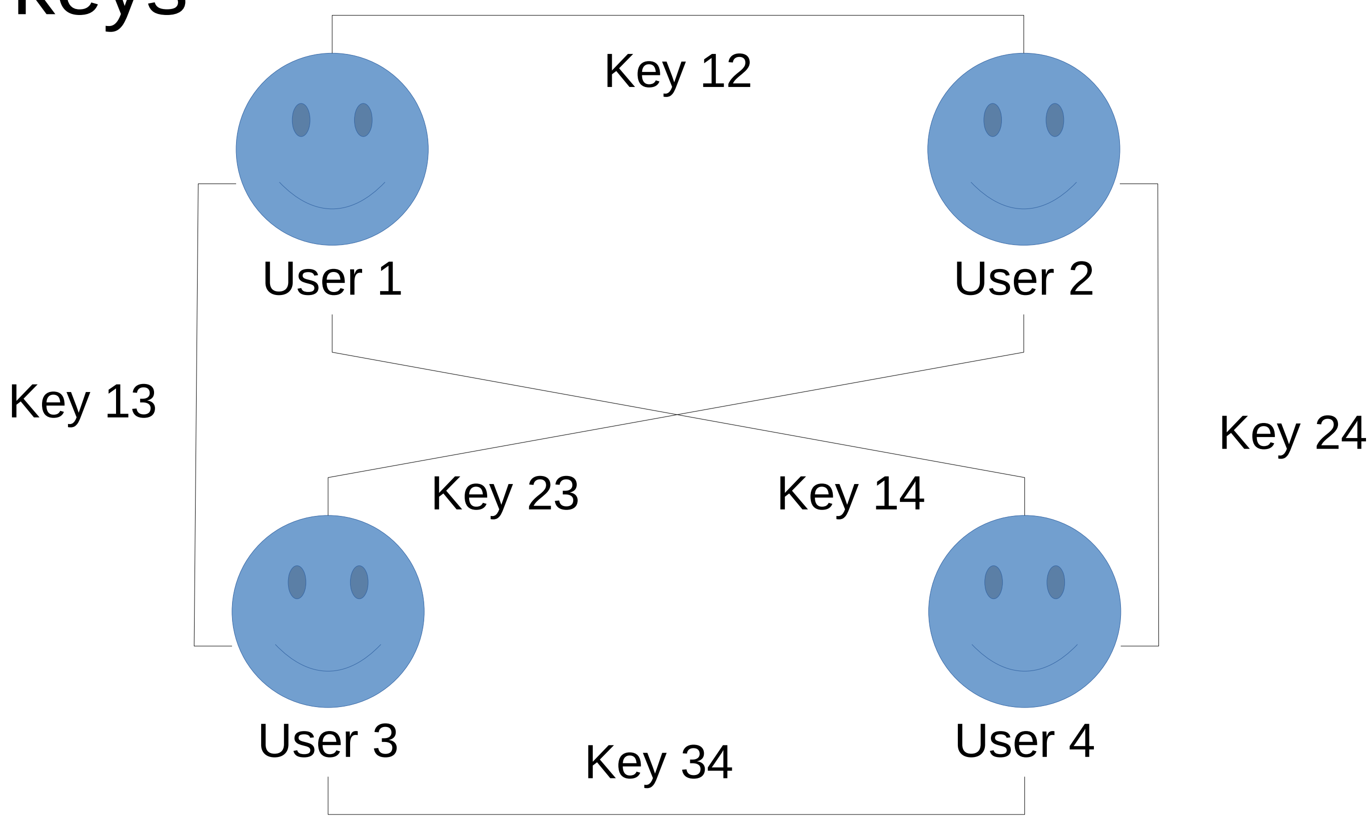
- Cryptography is a powerful tool, but it is too easy to use it incorrectly
  - Systems work, but could be easily attacked
- To reduce the probability of making mistakes
  - Have others review your design and code
  - Never invent your own primitives (ciphers, MACs, modes of operation, …)
  - Avoid implementing your own cryptographic operations
    - E.g. instead of combining AES-CTR and HMAC, prefer AES-GCM

Key Exchange

Contents

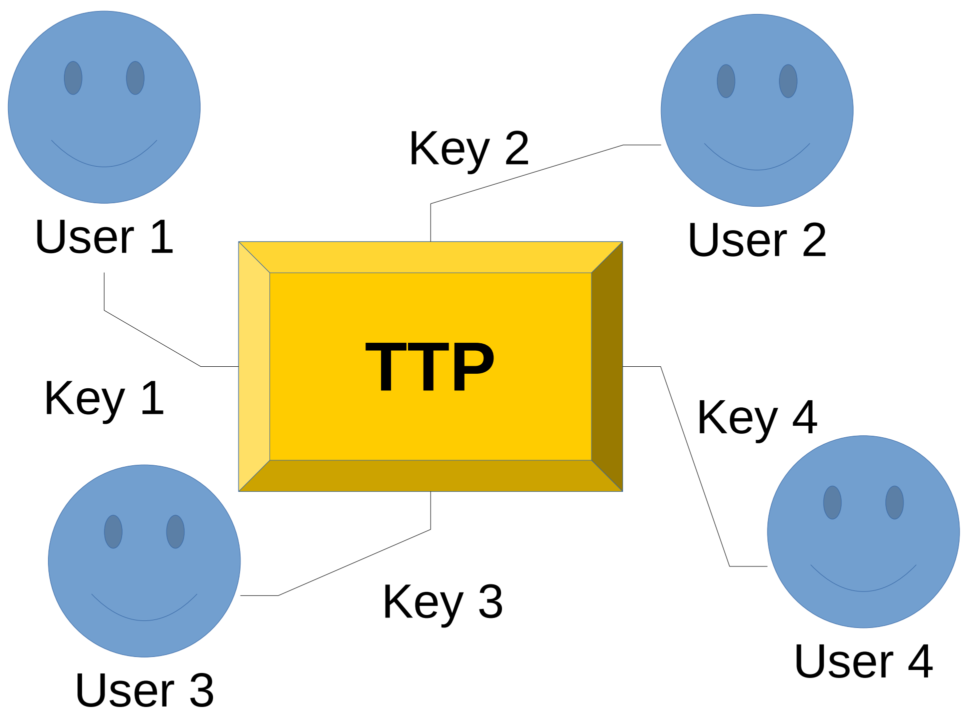
Key management

- Storing mutual secret keys is difficult
- In a universe of *n* users, each user requires *O(n)* keys



On-line Trusted Third Party (TTP)

- Every user has to manage only **a single key**
  - The one used to communicate with TTP
- Upon request, the TTP generates shared secret keys for user sessions



TTP: Generating keys (toy protocol)

- Bob wants a shared secret with Alice

Alice k_A

Bob k_B

TTP

A key for Alice, please.

$$k_{AB} \xleftarrow{R} \{0,1\}^{128}$$

$$m_1 = E(k_B, \text{'B, A'} \| k_{AB})$$

$$m_2 = E(k_A, \text{'B, A'} \| k_{AB})$$

Ticket

m_1, m_2

m_2

k_{AB}

k_{AB}

(E, D) a CCA secure cipher.

TTP: Security

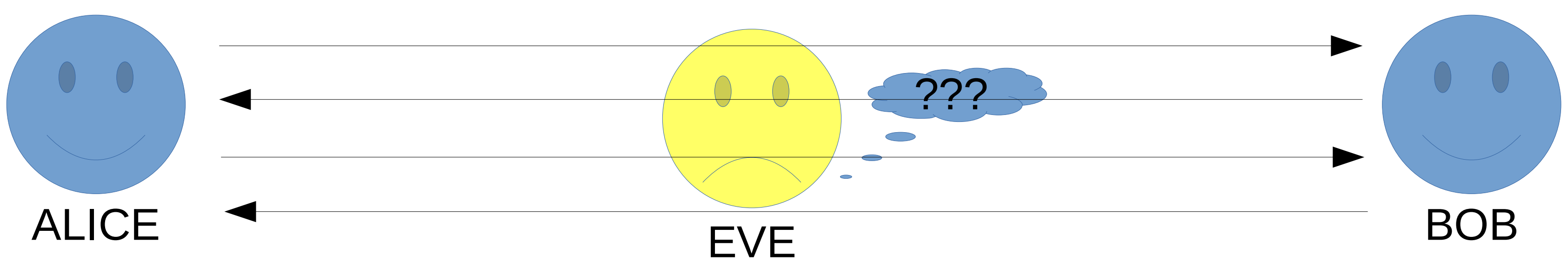
- An eavesdropper sees
  - $m_1 = E\left(k_B, \text{'B, A'} \| k_{AB}\right)$
  - $m_2 = E\left(k_A, \text{'B, A'} \| k_{AB}\right)$
- Since (E, D) is CCA secure, she learns nothing about k_{AB}
- Issues
  - TTP needed for all key exchanges
  - TTP knows all user and all session keys
  - Replay attacks possible
- Basis of Kerberos

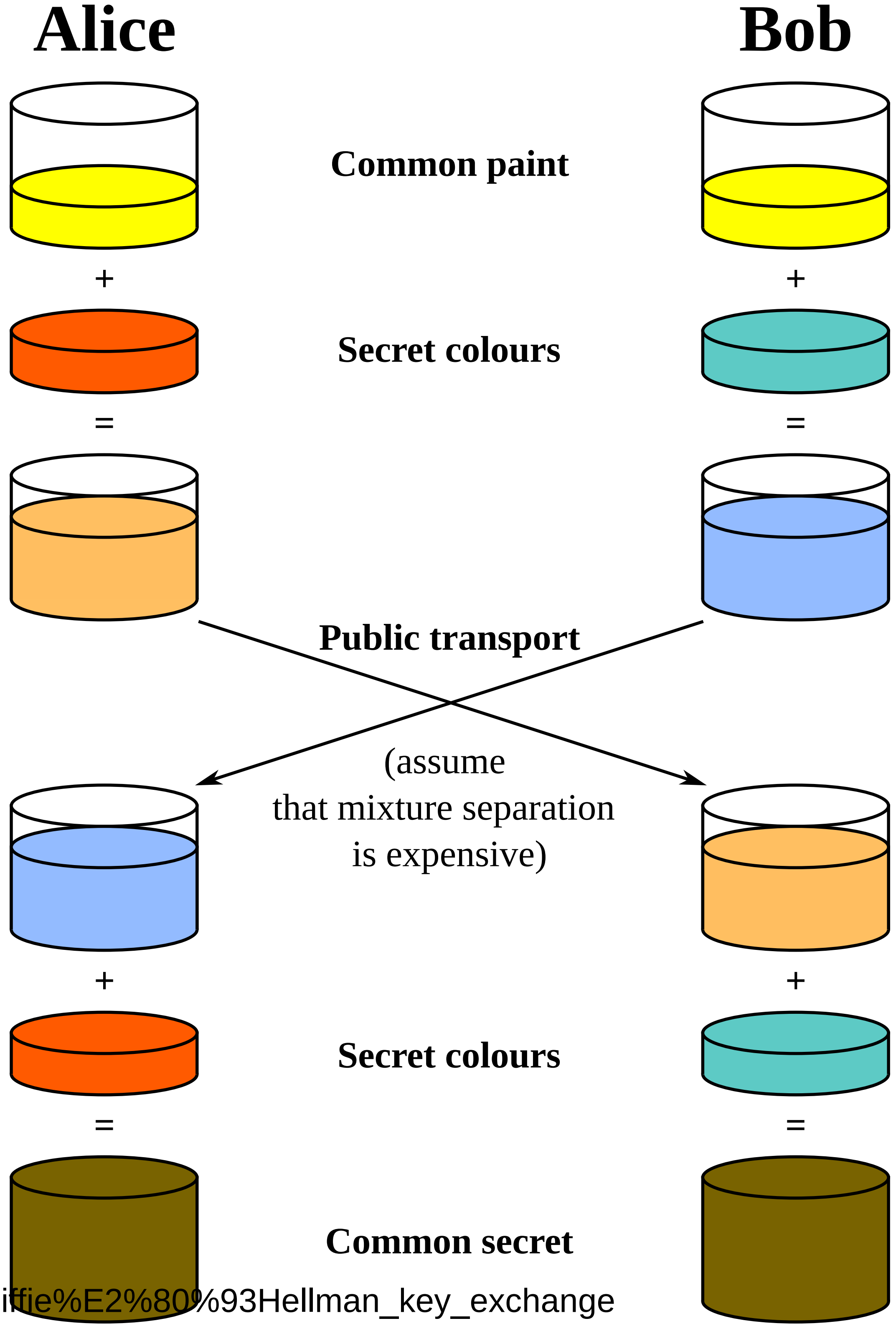
The main issue

- Can we generate shared keys without an **on-line** TTP?
  - YES!
- Entrance of public-key cryptography
- Two most widely known constructions
  - Diffie-Hellman protocol (1976)
  - RSA crypto system (1977)

Diffie-Hellman protocol

- Stems from hard problems in algebra
- Alice an Bob want to establish a shared secret in the presence of an eavesdropper
- Security against eavesdropping only

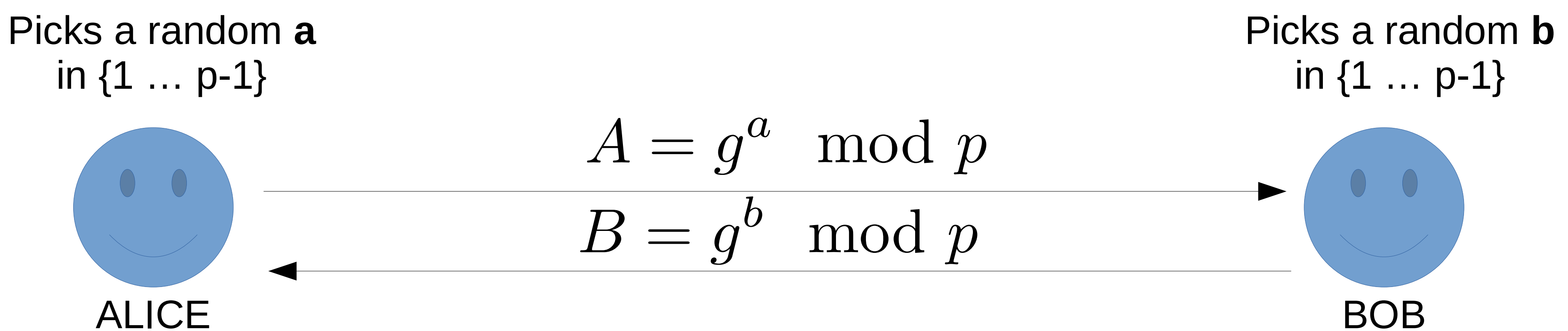




https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

Diffie-Hellman protocol (informally)

- Fix a large prime **p** (600 digits ~ 2kbits long)
- Fix an integer **g** in ***G*** = {1 … p-1} such that **g** is a <u>primitive root</u> modulo **p** (generator)
  - Raising **g** to powers of 0 to p-2 generates all values in {1 … p-1}

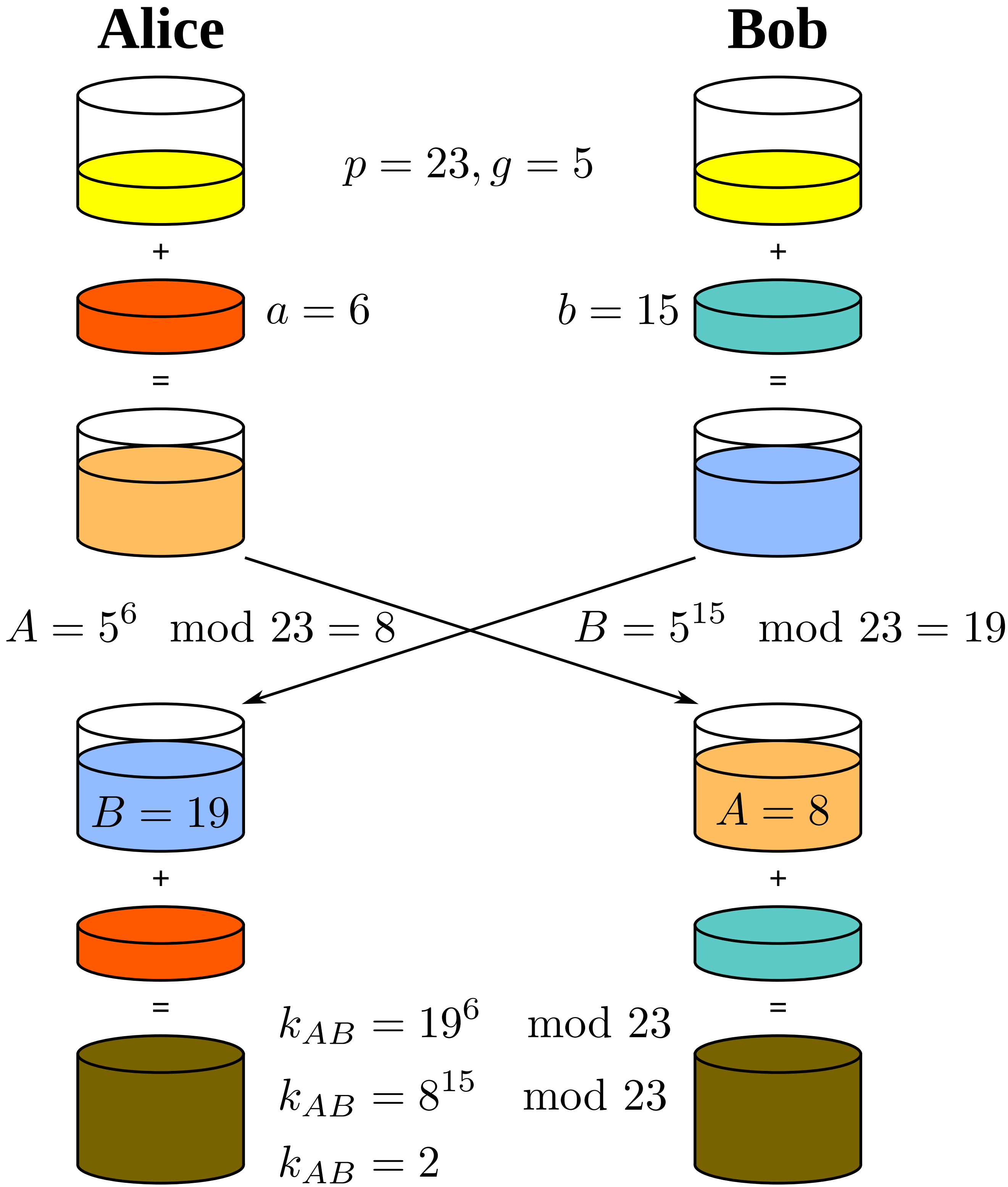


$$k_{AB} = g^{ab} \mod p$$

$$B^a = (g^b)^a = g^{ab} \mod p \qquad A^b = (g^a)^b = g^{ab} \mod p$$

Adaptation of: Dan Boneh, Cryptography I, Stanford.

$5^0 = 1 \mod 23$
$5^1 = 5 \mod 23$
$5^2 = 2 \mod 23$
$5^3 = 10 \mod 23$
$5^4 = 4 \mod 23$
$5^5 = 20 \mod 23$
$5^6 = 8 \mod 23$
$5^7 = 17 \mod 23$
$5^8 = 16 \mod 23$
$5^9 = 11 \mod 23$
$5^{10} = 9 \mod 23$
$5^{11} = 22 \mod 23$
$5^{12} = 18 \mod 23$
$5^{13} = 21 \mod 23$
$5^{14} = 13 \mod 23$
$5^{15} = 19 \mod 23$
$5^{16} = 3 \mod 23$
$5^{17} = 15 \mod 23$
$5^{18} = 6 \mod 23$
$5^{19} = 7 \mod 23$
$5^{20} = 12 \mod 23$
$5^{21} = 14 \mod 23$
$5^{22} = 5^0 = 1 \mod 23$



Security (informally)

- An eavesdropper sees
  - $p, g, A = g^a (mod\ p), B = g^b (mod\ p)$
- Can she derive $g^{ab} (mod\ p)$ herself?

- In general, let's define

$$\mathrm{DH}_g(g^a, g^b) = g^{ab} \qquad (mod\ p)$$

- How difficult is to compute DH function (mod p)?

Security (informally)

- Suppose *p* is *n* bits long
- Best known algorithm (GNFS) computes function DH in $e^{O\left(\sqrt[3]{n}\right)}$
- How difficult is to break DH compared to breaking a symmetric cipher?

Cipher key size	DH modulus size [in modulo primes]	DH modulus size [Elliptic Curve]
80	1024	160
128	3072	256
256	15360	512

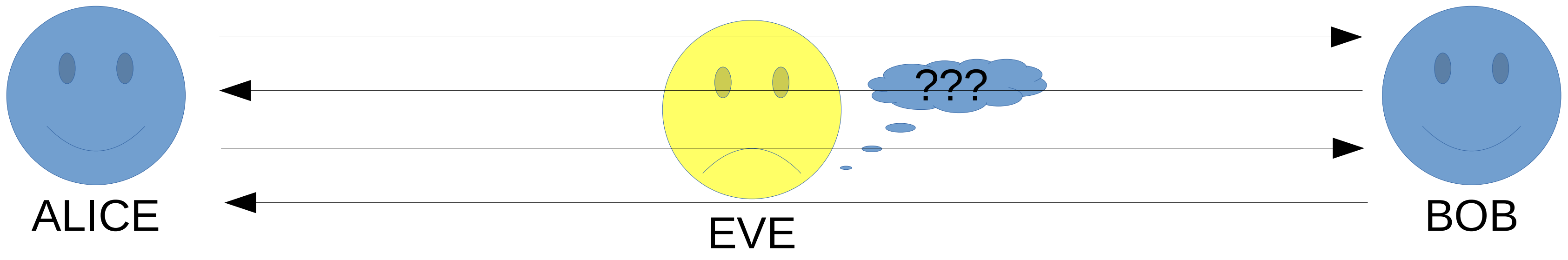
- Slow transition from (mod *p*) to elliptic curves

DH: Open issues

- Remember: security against eavesdropping only
- An active attacker can break the protocol with the man-in-the-middle attack
  - Reason: exchanges <u>are not authenticated</u>

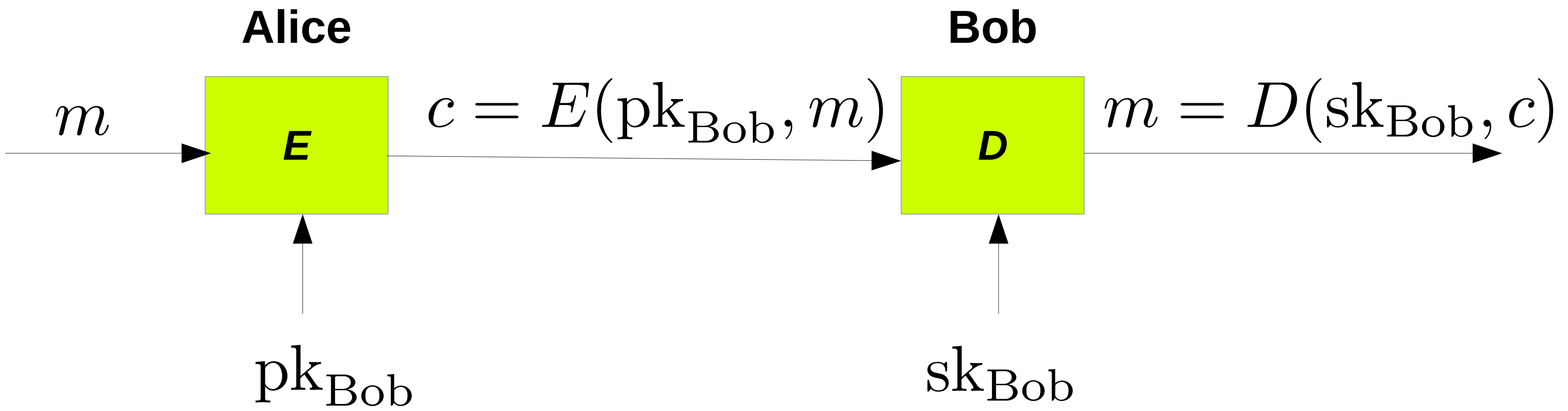
Public key encryption for key exchange

- Alice an Bob want to establish a shared secret in the presence of an eavesdropper
- Security against eavesdropping only



Public key encryption

- Each party uses a key pair: $k = (\mathrm{pk}, \mathrm{sk})$
- Public key is given to everyone, secret is kept hidden



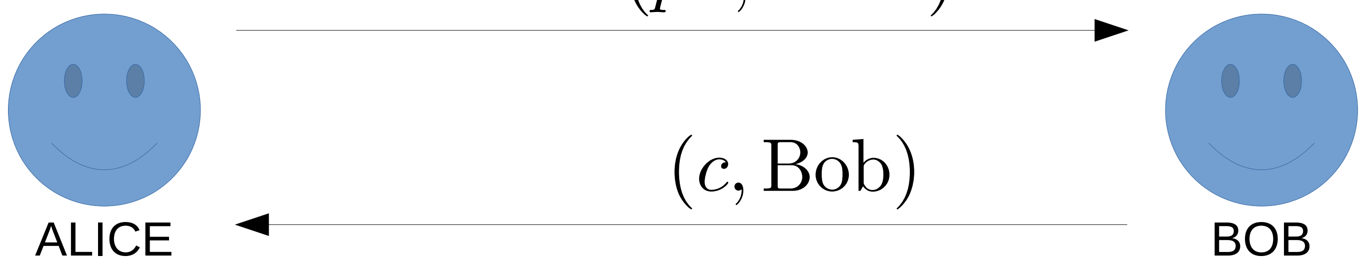
Adaptation of: Dan Boneh, Cryptography I, Stanford.

Establishing a shared secret

$(pk, sk) \leftarrow G()$ $\qquad (pk, \text{Alice})$ $\qquad x \xleftarrow{R} \{0,1\}^{128}$

$c \leftarrow E(pk, x)$

(c, Bob)



$D(sk, c) \rightarrow x$ $\qquad x: \ \text{shared secret}$

- Adversary sees $pk, \ E(pk, x)$
- Adversary wants x
- If $\zeta = (G, E, D)$ is sem. secure, the adv. obtains no information about x
- Security **against eavesdropping only**: protocol still vulnerable to man-in-the-middle

Digital signatures

- Preserving integrity in public-key cryptography
  - “MACs” of public-key cryptography
- Idea: The signer signs a message with her secret key. Anyone can verify the signature using the corresponding public key and thus know:
  - That the message has not been tampered with
  - That the signer indeed signed the message
- Similar to MACs, but digital signatures are
  - **Publicly verifiable**: anyone (with PK) can verify the signature
  - **Non-repudiative**: the signer cannot later deny having signed a particular message

Signature scheme: def.

- **<u>Def:</u>** A signature scheme (G, S, V) is a triple of eff. algs. defined over (M, Z) where:

  - $G()$ is a rand. alg. that generates key pairs (pk, sk)

  - $S(sk, m)$ is an alg. that signs a message $m \in M$ using secret key sk and produces a signature $z \in Z$

  - $V(pk, m, z)$ is a det. alg. that verifies the signature $z \in Z$ of message $m \in M$ using pk and outputs **1** if the signature verifies, or **0** otherwise

  - A signature generated by *S* must always verify by *V*:

$$\forall (pk, sk), m \in M : \Pr[V(pk, m, S(sk, m)) = 1] = 1$$

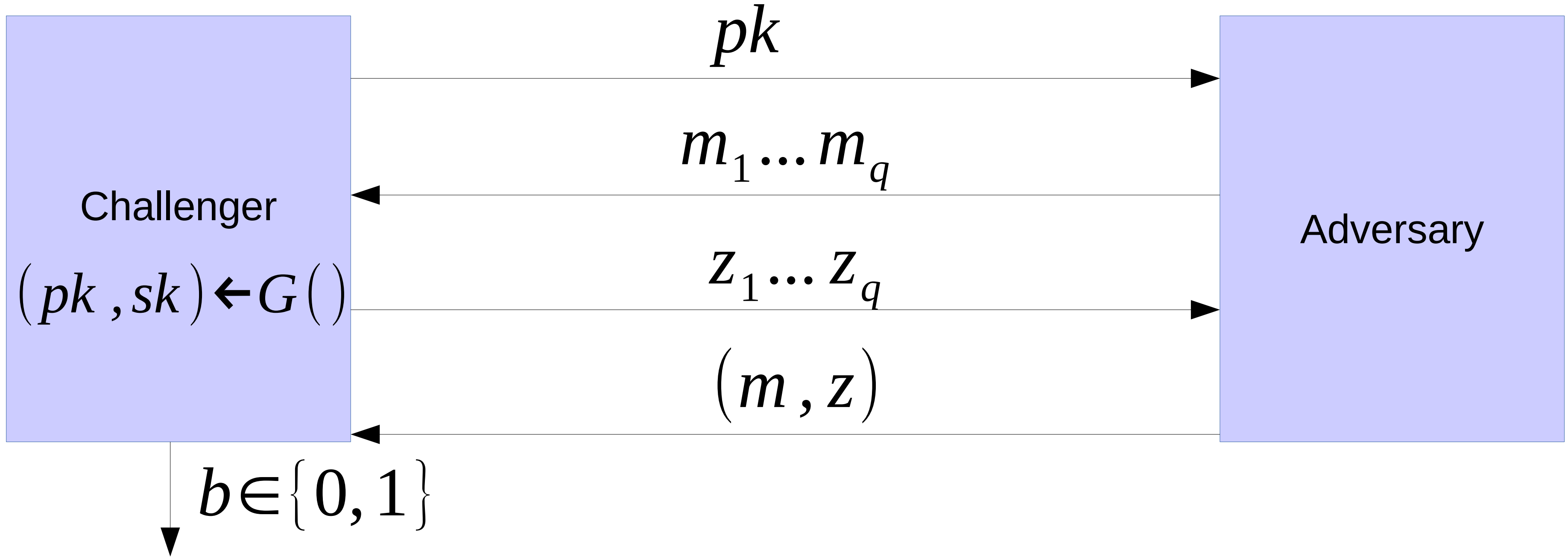
Digital signatures: Threat model

- Attacker's power: **Chosen message attack**
  - For $m_1 \ldots m_q$ attacker is given $z_i = S(sk, m_i)$
- Attacker's goal: **Existential forgery**
  - Produce a **new** valid (m, z) s. t.

$$m \notin \{m_1 \ldots m_q\}$$

$\rightarrow$ *An adversary cannot produce a valid signature for a new message*

Secure digital signature: def.



$$b = 1 \quad \text{if } V(pk, m, z) = 1 \text{ and } m \notin \{m_1 \ldots m_q\}$$
$$b = 0 \quad \text{otherwise}$$

A signature scheme (G, S, V) is **secure** if for all "efficient" adversaries A:

$$\mathrm{Adv}_{\mathrm{SIG}}[A, I] = \Pr[\text{Chal. outputs } 1]$$

is "negligible".

Extending the message space

- **Hash-and-sign paradigm**
  - *Constructing a signature scheme for large messages from a signature scheme for small messages (and strengthening security)*
- **Thm.** Let (G, S, V) be a secure signature scheme over (M, Z) and let $H : M' \to M$ be a collision resistant hash function where $|M'| \gg |M|$. Then (G, S', V') is also secure sig. scheme, where:

$$S'(sk, m) := S(sk, H(m))$$

$$V'(pk, m, z) := V(pk, H(m), z)$$

Signatures from TDP: Full Domain Hash

- Building blocks
  - ***(G, F, F^{-1})*** – Secure trapdoor permutation (TDP)
    - ***F: X→X***
  - ***H: M→X*** – collision resistant hash function
- Full domain (length) hash (FDH)
  - ***G()*** from TDP
  - $S(sk, m) := F^{-1}(sk, H(m))$
  - $V(pk, m, z) := \begin{cases} 1 & H(m) = F(pk, z) \\ 0 & \text{otherwise} \end{cases}$

Signatures from TDP: Full Domain Hash

- **Thm.** Let $\boldsymbol{(G, F, F^{-1})}$ be a secure TDP $\boldsymbol{X} \rightarrow \boldsymbol{X}$ and let $\boldsymbol{H: M} \rightarrow \boldsymbol{X}$ be a collision resistant hash function. Then signature scheme FDH is secure if $\boldsymbol{H}$ is a *random oracle*.
- FDH produces <u>unique signatures</u>: every message has its own signature

Signatures from TDP: Full Domain Hash

- Hashing is required for security; schemes without hashing are insecure. For instance:

$$S(sk, m) := F^{-1}(sk, m) \qquad V(pk, m, z) := F(pk, z) == m$$

  - **Zero-message attack**: create an existential forgery by picking a random signature, and creating a "message" from it

$$z \xleftarrow{R} Z, m \leftarrow F(pk, z)$$

  - **Multiplicative-property attack** (when using RSA)
    - Ask for signatures on two messages $m_{1,} m_2$

$$z_1 \leftarrow S(sk, m_1), z_2 \leftarrow S(sk, m_2)$$

    - Output existential forgery

$$m_3 \leftarrow m_1 \cdot m_2$$
$$z_3 \leftarrow z_1 \cdot z_2$$

Signatures from RSA trapdoor

- ***G()***
  - Choose random primes p, q (~1024 bits); $N = p \cdot q$
  - Choose integers e, d such that $e \cdot d = 1 \mod \varphi(N)$
  - Return $pk = (N, e),\ sk = (N, d)$
- $S((N,d),m) := H(m)^d \mod N$
- $V((N,e),m,z) := \begin{cases} 1 & H(m) = z^e \mod N \\ 0 & \text{otherwise} \end{cases}$
- What about ***H***?

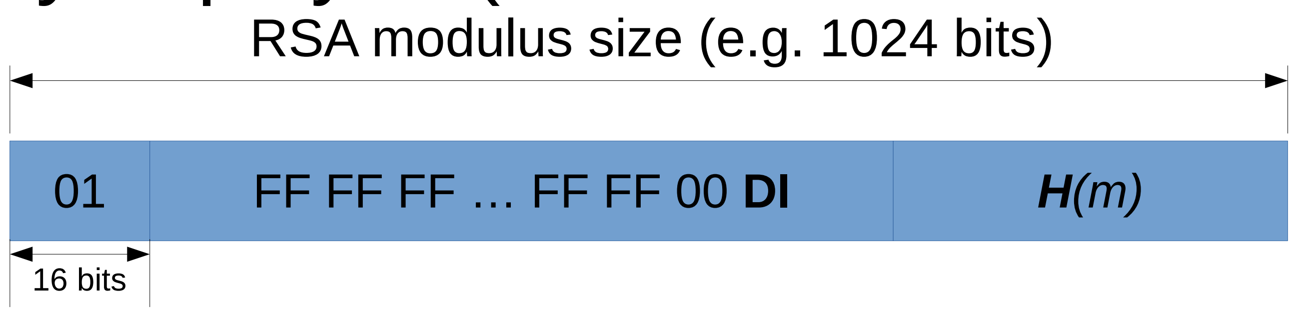
RSA Full Domain Hash

- We require ***H: M*** $\rightarrow \mathbb{Z}_N^*$
  - The output length of ***H*** depends on *N*; could be different for every public key
  - Ideally we want the output length of ***H*** to be fixed
- **Thm.** Let ***H: M*** $\rightarrow$ ***Y*** be a collision resistant hash function where ***Y*** = $\{1, \ldots, 2^{n-2}\}$ and *n* is the number of bits used to represent *N*. Then RSA-FDH is secure sig. scheme if ***H*** is a random oracle.

$\rightarrow$ The bit-length of digests must be of similar length as is the bit-length of the modulus $|Y| \geq N/4$

PKCS1 v1.5 signatures

- Widely deployed (TLS certificates, S/MIME, ...)



- **DI** – digest info encodes the name of the used hash function ***H*** (SHA*, MD*, …)
- The resulting value is then signed by raising it to *d* in *mod N* (recall, *sk = (N, d)*)
- <u>Not FDH</u>, but partial domain hash
  - No security proof; also no known substantial attacks
    - Issue with proving: ***H***(m) maps to a small subset of $\mathbb{Z}_N^*$

Probabilistic Signature Scheme (PSS)

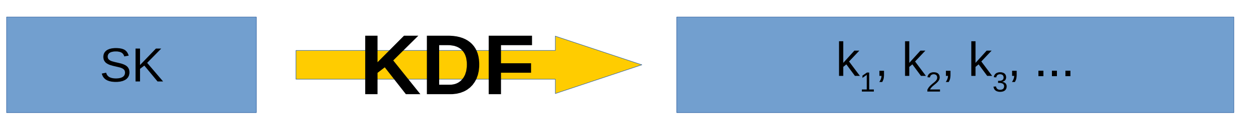
- Randomizes the signature with a public random value *s* called **salt**
- $S((N,d),m,s) := [H(s||m) \;||\; MGF[H(s||m)] \oplus s]^d \mod N$
  - MGF – mask generating function that extends the hash size to the full modulus size
- $V((N,e),m,z,s) := \begin{cases} 1 & H(s||m) \;||\; MGF[H(s||m)] \oplus s = z^e \mod N \\ 0 & \text{otherwise} \end{cases}$
  - Provably secure in random oracle model
  - Part of PKCS1 v2.1

Digital Signature Standard (DSS)

- NIST (FIPS 186)
  - Also called Digital Signature Algorithm (**DSA**)
- Relies on the hardness of Dlog
- No known proof of security
  - But also no serious attacks found
- Has an equivalent in elliptic curves (**ECDSA**)

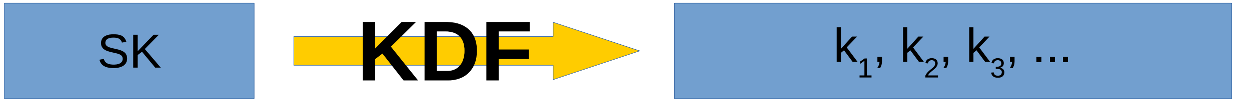
Deriving many keys from one

- **Scenario**: we obtain a single <u>source key</u> (SK)
  - From a hardware random number generator
  - From a key exchange protocol
- We need many keys to secure the session
  - Unidirectional keys, MAC/encryption keys
- **Goal**: generate many keys from a single SK
  - KDF – key derivation function



Deriving many keys from one

- Three cases
  1) SK is uniform in key space
  2) SK is non-uniform in key space
  3) SK is a password



Key derivation: (1) SK is uniform

- Let PRF F: K × X → $\{0, 1\}^n$
- If source key is **uniform** in K:

$$KDF(sk, ctx, l) := F(sk, ctx||0) \; || \; F(sk, ctx||1) \; || \; ... \; || \; F(sk, ctx||l)$$

- **ctx**: a string unique to every application
  - Assures that two applications derive independent keys even if they sample the same source key

Key derivation: (2) SK is non-uniform

- The KDF can be directly used <u>only when SK is uniform</u>

  → If SK is not uniform, the PRF output may not look random

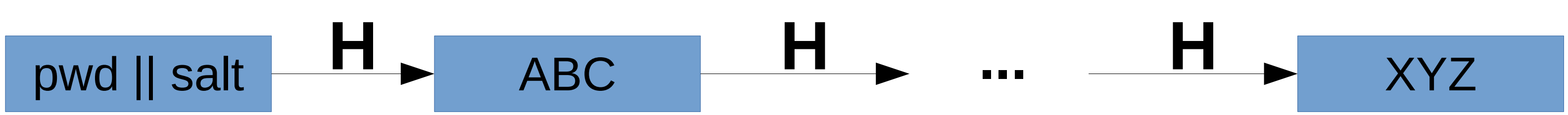
- Reasons for non-uniformity of SK
  - Hardware RNG may be *biased*
  - Key-exchange protocol may produce a key that is *uniform in some subset* of K

Key derivation: (2) SK is non-uniform

- **Extract-then-Expand paradigm**
  - Step 1) Use an **extractor** and **SK** to extract a pseudo-random key **k** that is uniform in key space
    - Use **salt:** a fixed public (non-secret) random string
  - Step 2) expand ***k*** with KDF
- **HKDF** – a KDF from HMAC
  - Step 1) ***k*** ← HMAC(salt, SK)
  - Step 2) Expand as you would with uniform keys, but use HMAC for PRF and ***k*** for key
  - https://tools.ietf.org/html/rfc5869

Key derivation: (3) SK is a password

- Particular care needed when deriving keys from passwords
  - HKDF unsuitable here: passwords have low entropy
  - Derived keys will be vulnerable to dictionary attack
- General idea: add **salt** and **slow down hashing**
- **PBKDF** – password-based KDF
  - PKCS #5 v2.0 and https://tools.ietf.org/html/rfc2898
  - Iterate hash function many times



Adaptation of: Dan Boneh, Cryptography I, Stanford.

Final words

- Cryptography is a powerful tool, but it is too easy to use it incorrectly
  - Systems work, but could be easily attacked
- To reduce the probability of making mistakes
  - Have others review your design and code
  - Never invent your own primitives (ciphers, MACs, modes of operation, …)
  - Avoid implementing your own cryptographic operations
    - E.g. instead of combining AES-CTR and HMAC, prefer AES-GCM