

# Resilient Networking

*Disclaimer: Some parts have been inspired by Dan Boneh/Mark Manulis*

Module 2b – Background on Crypto (Winter Term 2021)

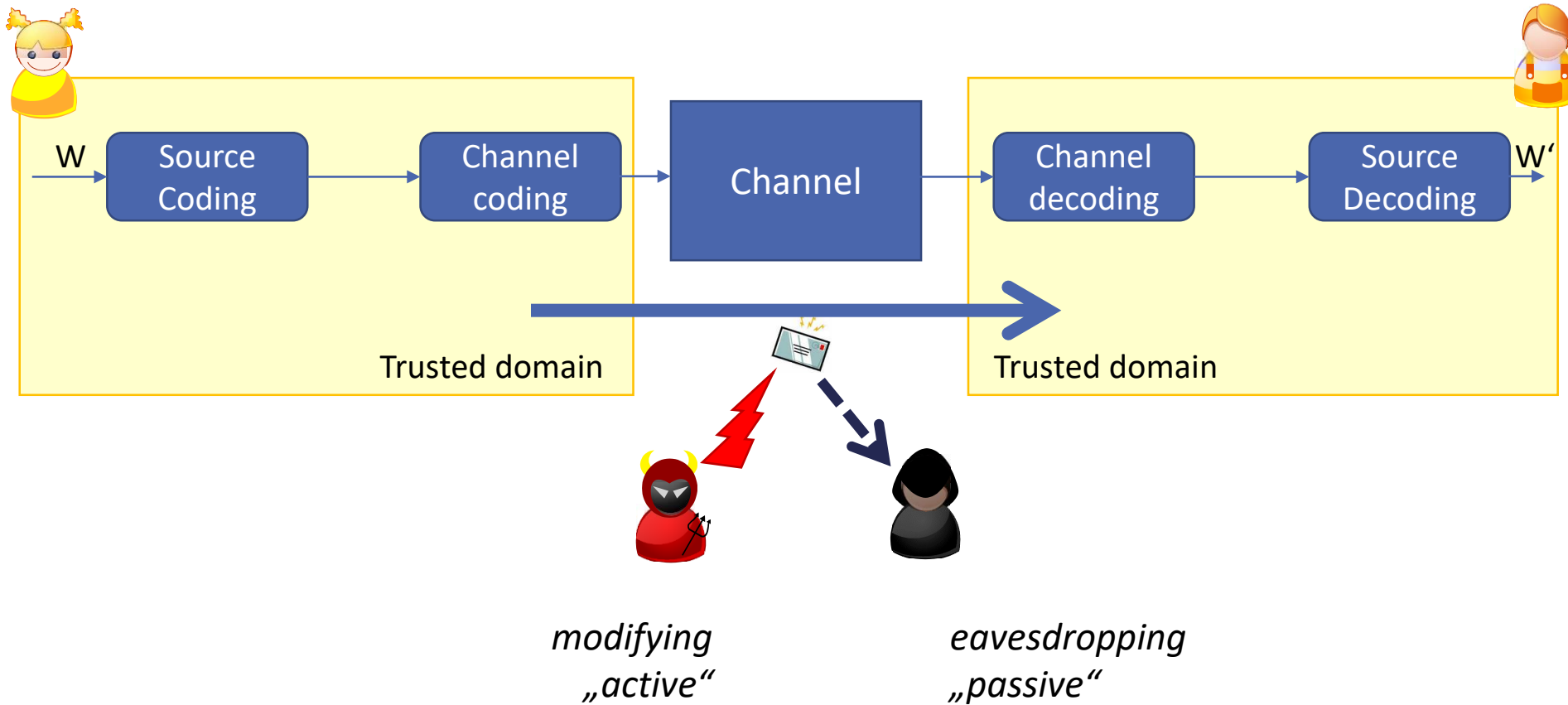
Thorsten Strufe



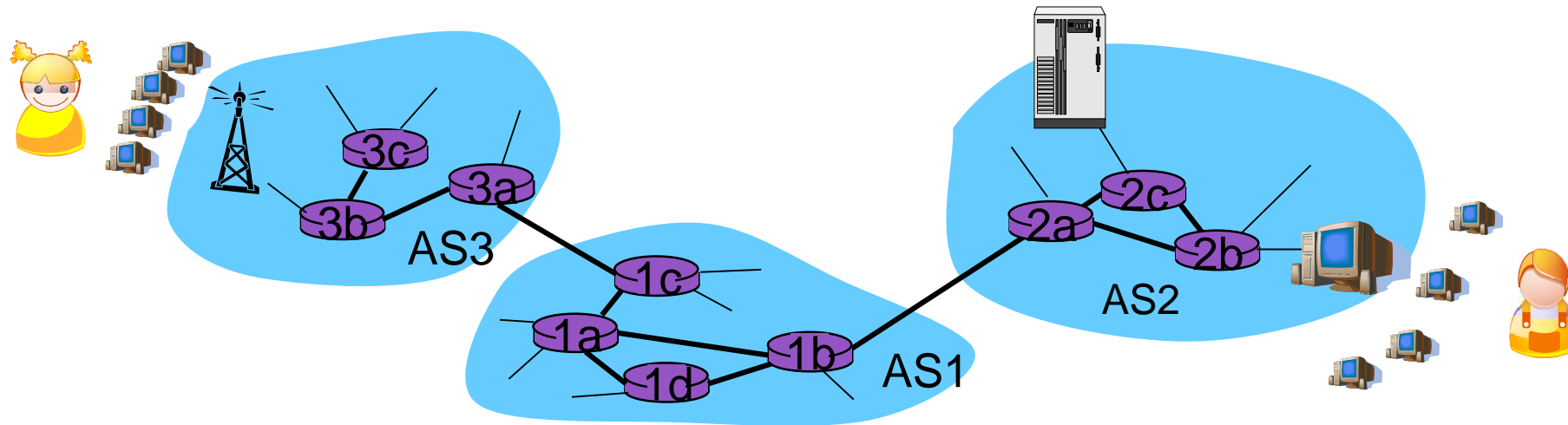
Competence Center for Applied Security Technology



# Security in Telecommunication



# (Inter)networked Communication



# More Specific: The Dolev - Yao Adversary



- Mallory has full control over the communication channel
  - Intercept/eavesdrop on messages (passive)
  - Relay messages
  - Suppress message delivery
  - Replay messages
  - Manipulate messages
  - Exchange messages
  - Forge messages
- But:
  - Mallory *can't* break (secure) cryptographic primitives!

# And what we (crypto) wants to achieve (CIA)

- **Confidentiality:**
  - Data transmitted or stored should only be revealed to the intended audience
  - **Confidentiality of entities** is also referred to as **anonymity**
- **(Data) Integrity:**
  - It should be possible to detect any modification of data
  - This requires to be able to *identify* the creator of some data
- **Availability:**
  - Services should be available and function correctly
- **Accountability:**
  - *It should be possible to identify the entity responsible for any communication event*
- **Controlled Access:**
  - *Only authorized entities should be able to access certain services or information*

# ...By means of... *Security Services:*

## ■ *Authentication*

- Ensure that an entity has in fact the identity it claims to have

## ■ *Integrity (authenticated modification detection)*

- Ensure that data created by specific entity isn't modified ***without detection***

## ■ *Confidentiality (content hiding)*

- Ensure the secrecy of protected data

## ■ *Access Control*

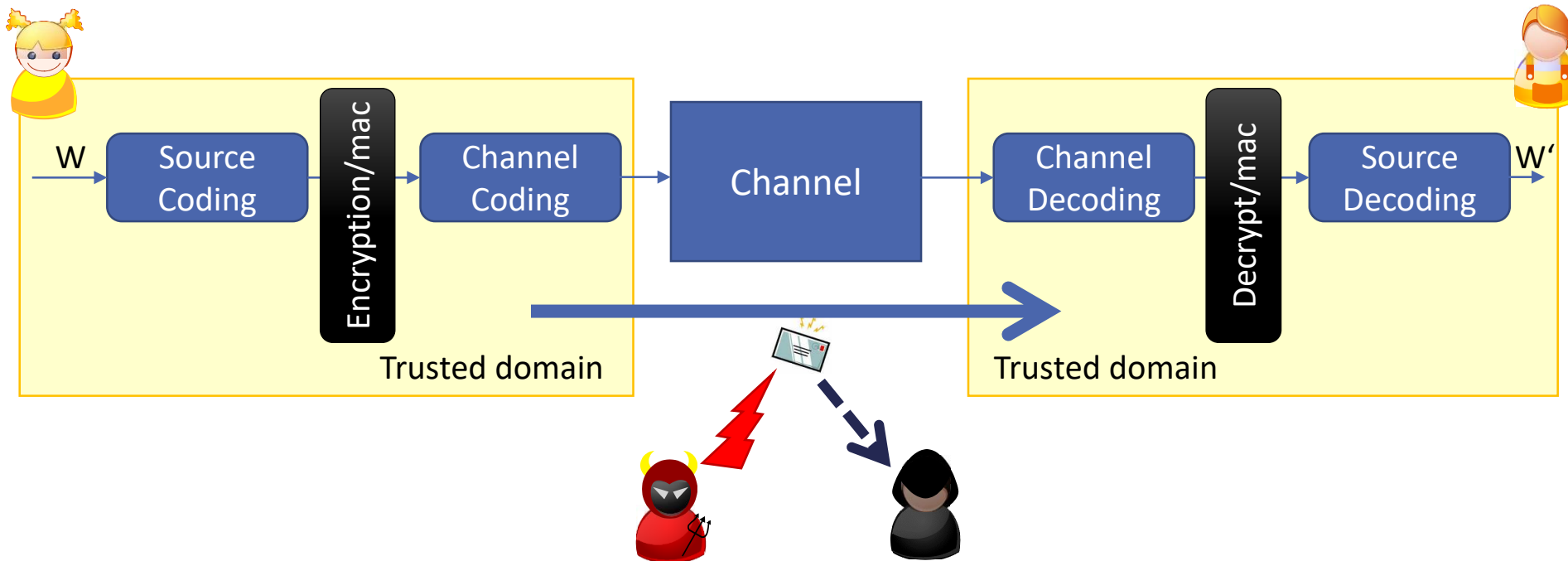
- Ensure that each entity accesses only services and information it is entitled to

## ■ *Non Repudiation*

- Prevent entities participating in a communication exchange from later falsely denying that the exchange occurred

# Extending the Channel Model:

- Crypto needs to transform payload, or add information, that cannot be interpreted nor predicted (iow: *“looks like random noise”*)



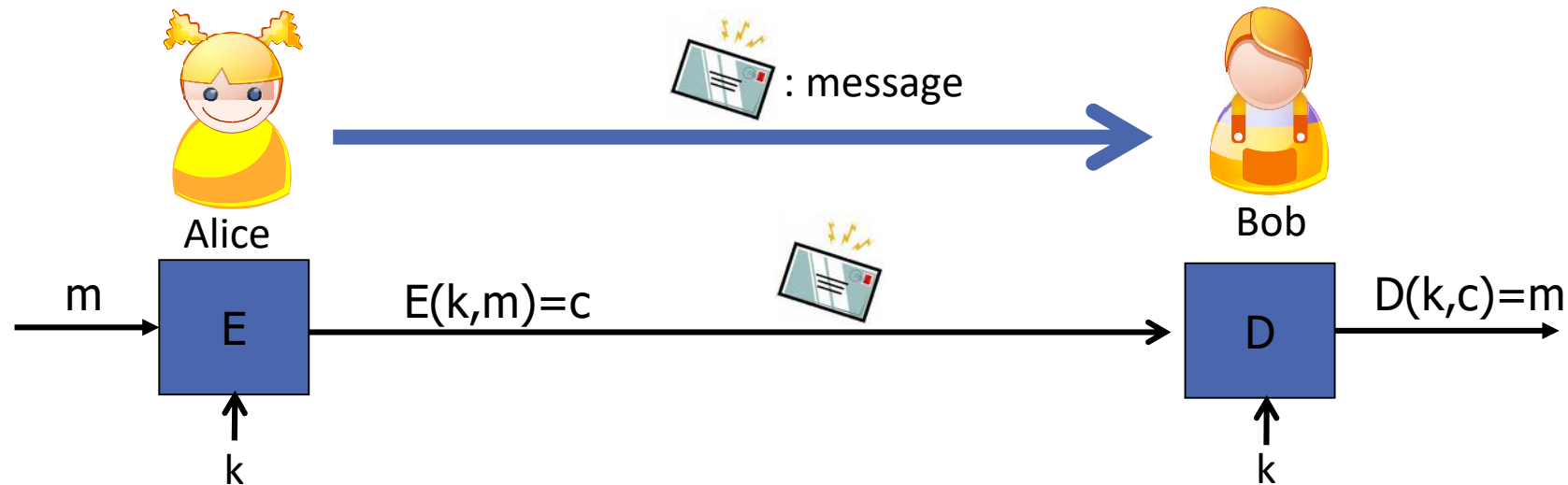
# Confidential Communication with Integrity

# Security Services for Confidentiality/Integrity

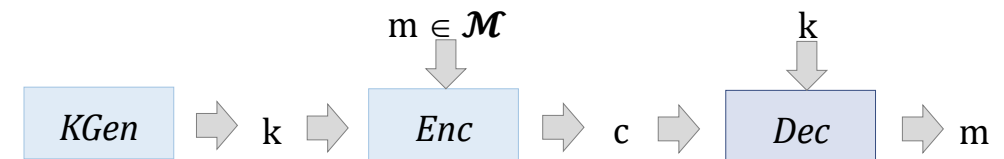
- We need algorithms (and protocols)
- To achieve confidentiality:
  - Conventional encryption
- To guarantee integrity:
  - Signing, „Message Authentication Codes“(MAC)
- Cryptographic algorithms for both...

# Confidential Communication

- Alice sends Bob a private (any!) message...



- $m$ : message (plaintext)  $\in M$  (message space, sometimes  $P$ )
- $k$ : key  $\in K$  (key space)
- $c$ : ciphertext  $\in C$  (ciphertext space)



- A cipher is a triple of algorithms:  $E, D, keygen$  (random/deterministic?; sometimes:  $Enc, Dec$ )

- **Correctness:** for all  $k \in \mathcal{K}, m \in \mathcal{M}$  :  $Dec(k, Enc(k, m)) = m$

# Communication with Integrity

- Defining similar sets and spaces:

M:	Space of possible messages	$(\{0,1\}^*)$
T:	Space of possible „tags“	(e.g. $\{0,1\}^{160}$ )
K:	Space of possible keys	(e.g. $\{0,1\}^{128}$ )

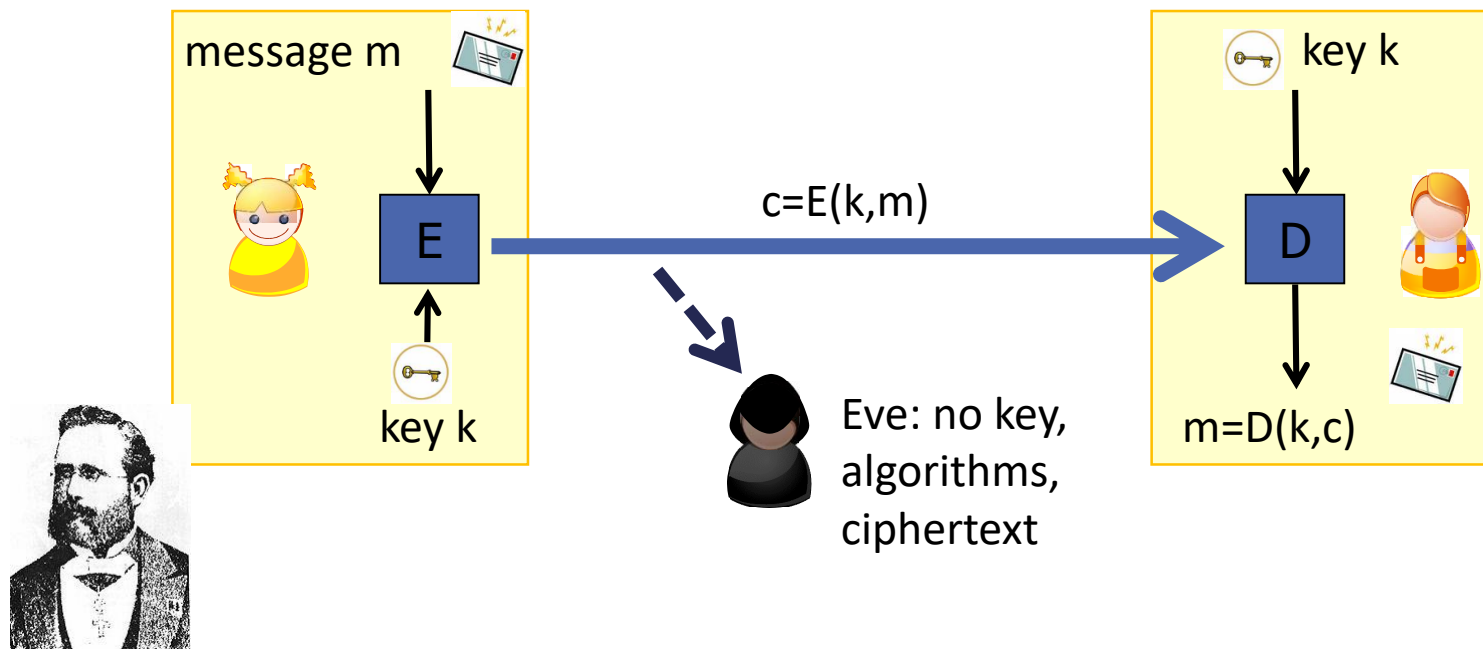
- Algorithms

- KeyGen()  $\rightarrow k$  *randomized*
- $S(k,m) \rightarrow t$       with       $m \in \{0,1\}^n, t \in \{0,1\}^t$       ( $n \gg t$ )
- $V(k,m,t) \rightarrow \{0,1\}$  *both: deterministic*



# Kerckhoffs' principle (*how to make this secure?*)

- KGen, Enc, und Dec will be lost to the adversary
  - Publish all algorithms right away!
  - Security must depend only on secrecy of the key (secret, unpredictable)



*“The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.”*

# So what does it mean „secure“?

- What can the adversary observe, what does she know?
- What is it she's not allowed to learn in addition, achieve?
  
- Confidentiality:
  - Cannot learn the plaintext of the message (??)
  - Cannot extract the key (??)
  - → *Shouldn't learn anything they don't already know!*
  
- Integrity:
  - Shouldn't be able to modify a message (flip „0“ to 1) (??)
  - → *Must not be able to generate a valid tuple of message and tag!*

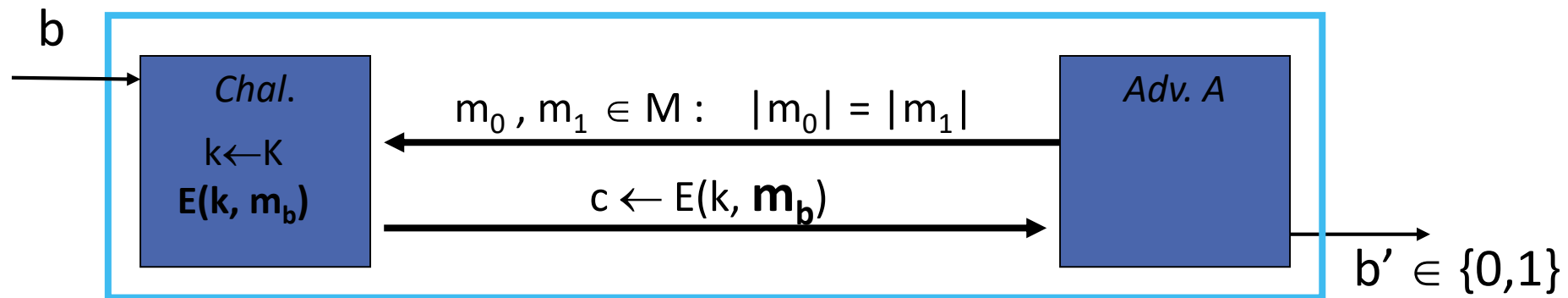
# Formalizing Security

# Formalizing Security / Worst-Case

- Discussion/Analysis requires to formalize „security“ (Conf./Int.)
- To be sure: Analyse Worst-Case!
  - Inputs for which the adversary would learn the most/could most easily tamper  
→ If adversary doesn't achieve anything in this case, we're safe!
- Assuming, algorithms/protocols are weaker for some inputs
  - Adversary would choose these, if she was challenged to break the system!
- Rationale of our analysis
  - To model Worst-Case, we play a game and let the adversary choose their inputs arbitrarily
  - *OK, but what does she need to learn or to generate to win the game?*

# Formalising Confidentiality as a Game

- The confidentiality game with two parties:
  - Challenger (C) throws a coin ( $b=\{0,1\}$ )
  - Adversary (A) communicates with (C) and has to guess  $b$ :

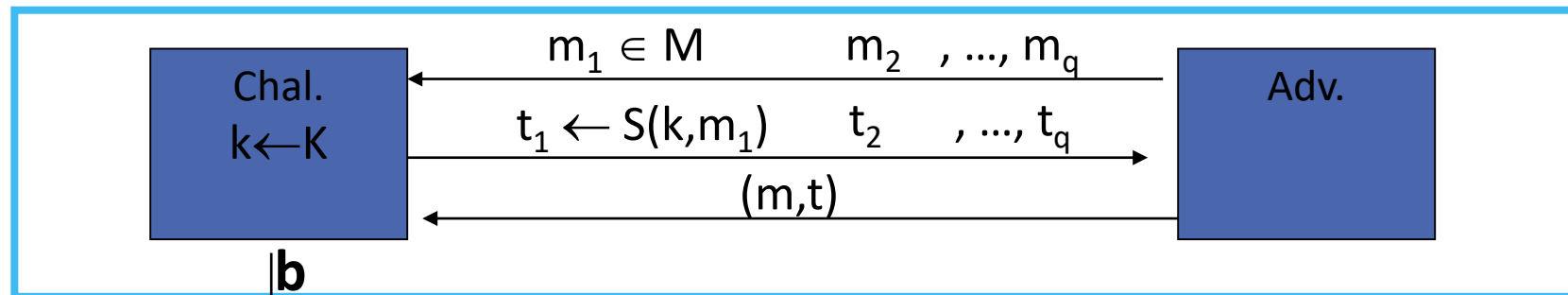


- What is the frequency of the adversary to win the game beyond chance?

$$Adv_{conf}[A, E] := |\Pr[b' = b] - \Pr[b' \neq b]| \in [0,1] \quad (\text{we demand this be: } \leq \epsilon)$$

# Formalising Integrity as a Game

- The integrity game with 2 parties,  $I = (S, V)$ :
- C produces valid tuple(s)  $(m, t)$  upon request
- Adversary communicates with (C), has to generate **one** valid tuple:



$$\begin{cases}
 \mathbf{b}=1 & \text{if } V(k, m, t) = '1' \text{ and } (m, t) \notin \{(m_1, t_1), \dots, (m_q, t_q)\} \\
 \mathbf{b}=0 & \text{otherwise}
 \end{cases}$$

- What is the frequency of the adversary to win the game?

$$\text{Adv}_{\text{MAC}}[A, I] = \Pr[\text{Chal. outputs } 1] \quad (\text{we demand this be: } \leq \epsilon)$$

# Concrete Examples

Let's try...

# Intermezzo: XOR

- XOR of two strings in  $\{0,1\}^n$  is their bitwise addition mod 2:

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

$$\begin{array}{cccccccc}
 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
 \hline
 0 & 1 & 1 & 0 & & & & 
 \end{array} \oplus$$

- **Theorem:**

- Let  $Y$  be random variable over  $\{0,1\}^n$
- Let  $X$  be an independent random variable with uniform distribution over  $\{0,1\}^n$
- Then  $Z := Y \oplus X$  has uniform distribution over  $\{0,1\}^n$ :
- (Proof on the blackboard)

# Try more: let's define two ciphers

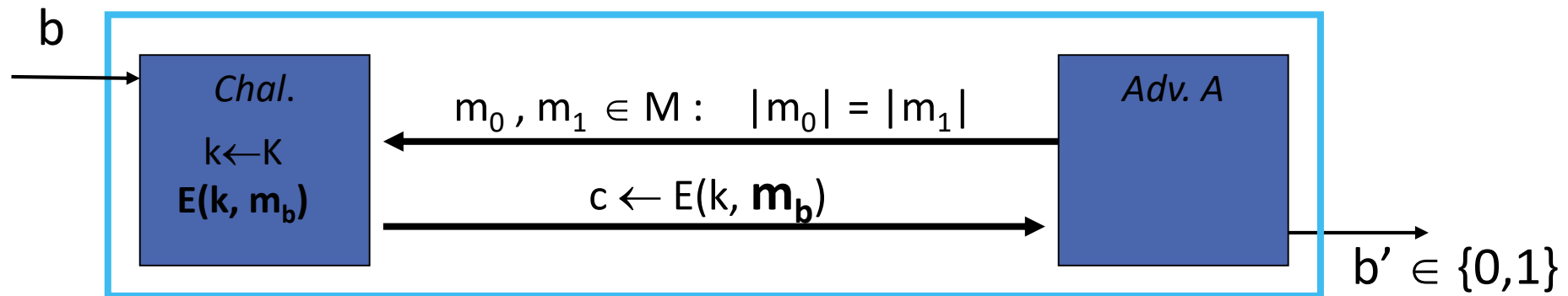
Let  $m = m_0 \dots m_{n-1}$  and

$k = k_0 \dots k_{l-1}$  a string of uniform distributed random bits, with  $l=n$

- $\text{Enc}_1(k,m) := c_0 \dots c_{n-1}$  with  $c_i = m_i \oplus m_{i+1}$  and  $m_n = m_0$
- $\text{Enc}_2(k,m) := c_0 \dots c_{n-1}$  with  $c_i = m_i \oplus k_i$

# Analysis of Enc<sub>1</sub>

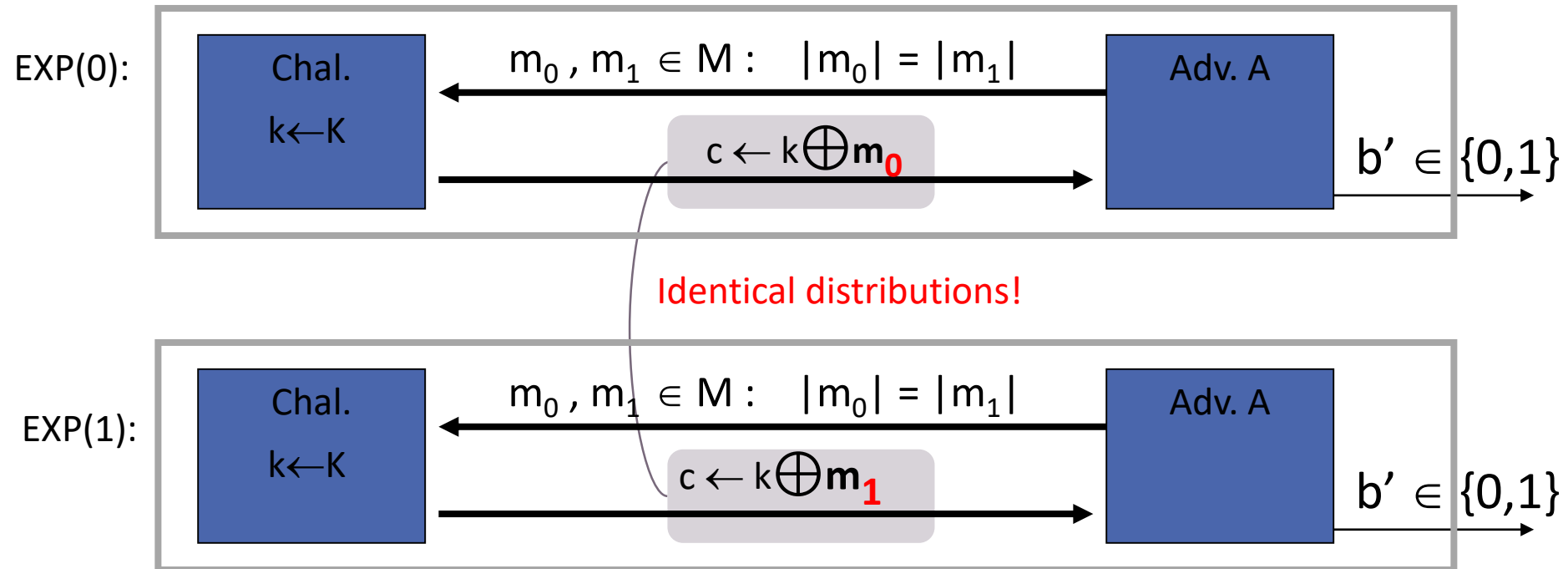
- $\text{Enc}_1(k, m) := c_i = m_i \oplus m_{i+1}$
- which tuple of messages is the adversary going to choose?



$$Adv_{conf}[A, \text{Enc}_1] := |\Pr[b' = b] - \Pr[b' \neq b]| \in [0, 1]$$



# Analysis of Enc<sub>2</sub> (One Time Pad)



For all A:  $\text{Adv}_{\text{Conf}}[A, \text{Enc}_2] = \left| \Pr[A(k \oplus m_0) = 1] - \Pr[A(k \oplus m_1) = 1] \right| = \blacksquare$

# The One Time Pad (Vernam Cipher)



Gilbert Vernam  
(1890-1960)

- Fundamental concept:
  - Key: string, as long as the message
  - Choose key bits truly random (no discernable pattern)
- $Enc(k, m) = c_0 \dots c_{n-1}$  with  $c_i = f(k_i, m_i)$  („+ mod |alphabet|“)

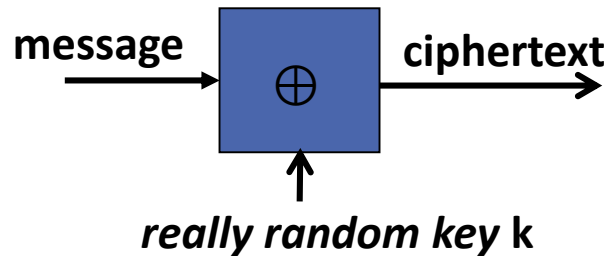
A	T	T	A	C	K	T	H	E	C	I	T	Y	A	T	T	W	E	L	V	E	(+ mod 26)
P	S	P	I	U	H	G	D	S	P	H	G	D	S	P	I	W	E	E	W	O	
P	L	I	I	W	R	Z	K	W	R	P	Z	B	S	I	B	S	I	P	R	S	
Y	H	P	R	S	R	G	F	F	D	D	X	N	S	Q	I	S	P	W	N	F	(+ mod 26)
R	E	T	R	E	A	T	F	R	O	M	C	O	A	S	T	A	T	T	E	N	

- Adversary knows length, cipher text, no plain text – learns nothing!
- *What are the problems for application?*

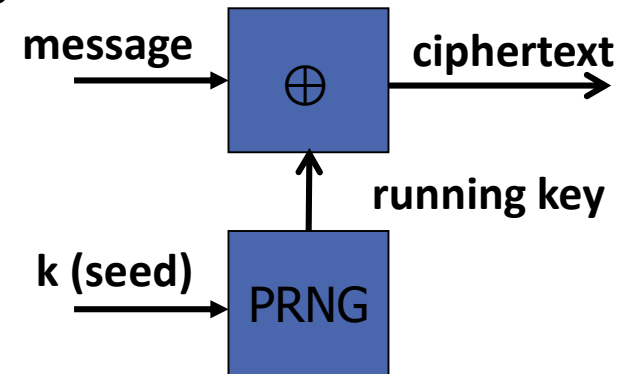
# A useful approach

# A practical cipher $\text{Enc}_3$

- OTP:



- $\text{Enc}_3$ :

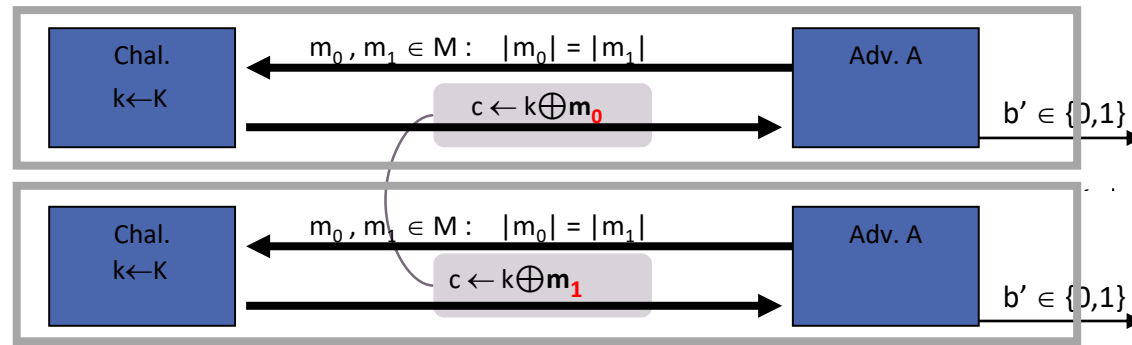


- Idea: Use „pseudo random“ sequence as key bits
- PRNG is a function  $G: \{0,1\}^s \rightarrow \{0,1\}^l$   $l \gg s$
- Det. algorithm from Seed- to key space (seemingly random)

- $k' = \text{PRNG}(k)$

- $\text{Enc}_3(k,m) := c_0 \dots c_{n-1}$  with  $c_i = m_i \oplus k'_i$

# Enc<sub>2</sub> vs Enc<sub>3</sub>



- Let PRNG be a „secure“ (unpredictable) PRNG,
- $\rightarrow$  Distributions in EXP(0) and EXP(1) are (almost) identical
- Some assumptions regarding the adversary (Adv. model #3, resources)
- Adv unlimited in time (theor.) vs. „efficient“ (PPT) adversary
- 1. Tests Enc<sub>3</sub> with all  $2^s$  possible keys and determines  $b' = b$
- 2. Cannot test exponentially many keys (limited to poly. time)
- We call Enc<sub>2</sub> **“information theoretically secure”** (perfect secrecy)
- We call Enc<sub>3</sub> (best case) **“semantically secure”**

# Information Theoretic Security

Shannon (1949):

- „CT should not reveal **any** information about PT“

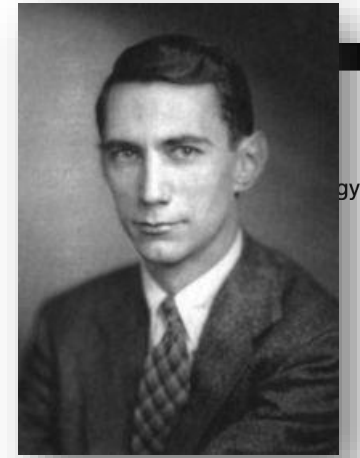
Def: A cipher (E,D) over  $(\mathcal{K}, \mathcal{M}, \mathcal{C})$  has **perfect secrecy** if

$$\begin{aligned} &\forall m_0, m_1 \in \mathcal{M} && (\text{with } \text{len}(m_0) = \text{len}(m_1) ) \\ &\forall c \in \mathcal{C} && \text{and } k \xleftarrow{R} \mathcal{K}: \\ &&& \Pr[E(k, m_0) = c] = \Pr[E(k, m_1) = c] \end{aligned}$$

*So being an attacker, what do I learn?*

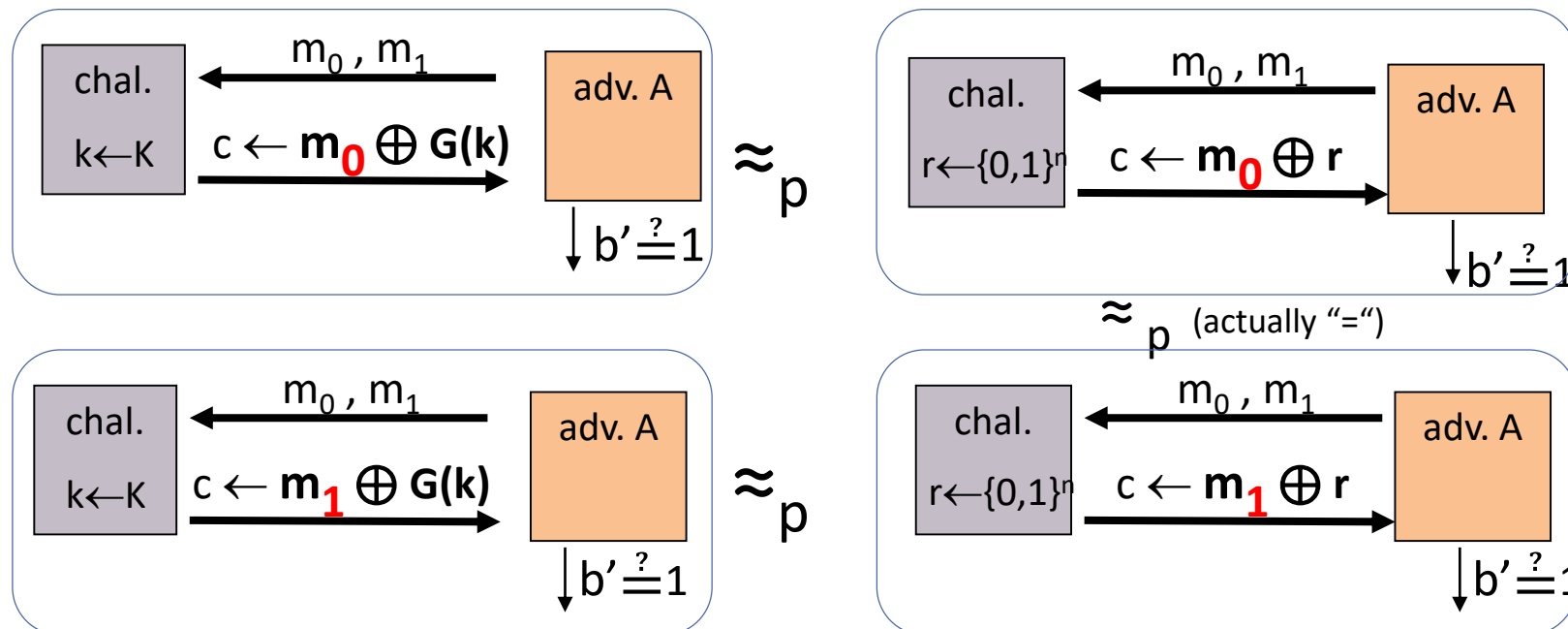
No CT attack can tell if msg is  $m_0, m_1$  (or any other message)

→ No CT only attacks



# Does our stream cipher have semantic security?

- Assume our PRNG cannot be distinguished from „real randomness“ by an efficient adversary.
- We've shown:
  - OTP (XOR real random key) is secure
- Intuition for our new proof:



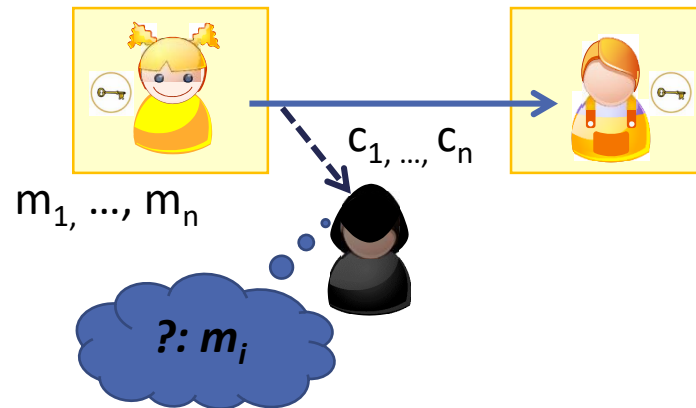
# Variations of the Game...

Defining different „security notions“

# Security Notions, Variations of the Game

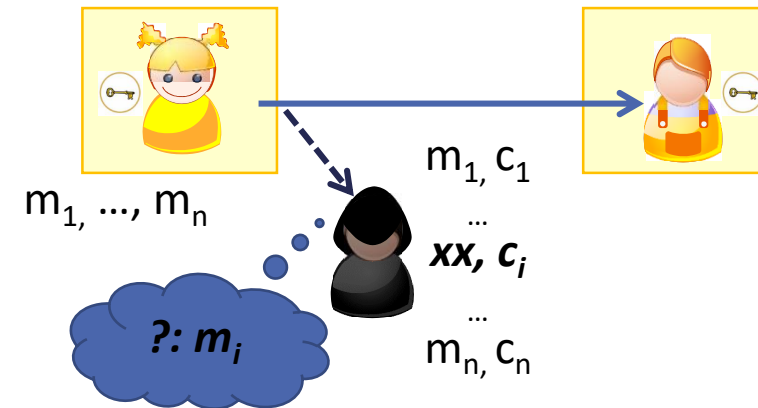
- **Ciphertext-only attack:**

- despite concealed key
- using ciphertext only
- learn about plaintext (or key)



- **Known-plaintext attack:**

- despite concealed key
- Knowing some plaintexts
- Learn about plaintext (or key)

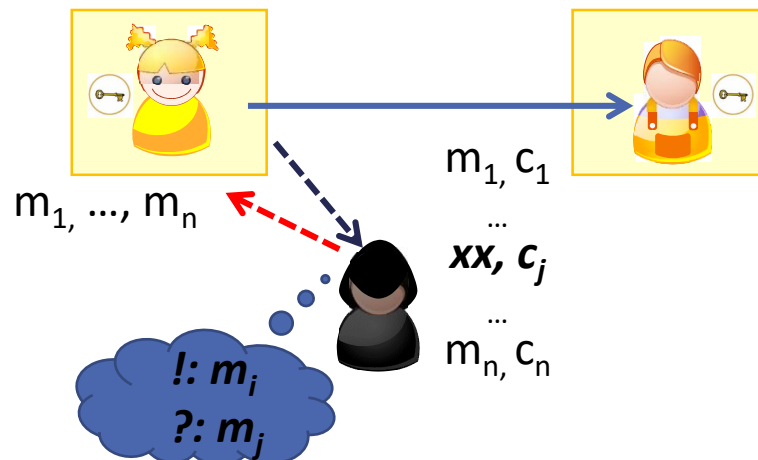


- **Weakest Adversary!**

# Security Notions ctd.

## ■ **Chosen-plaintext attack:**

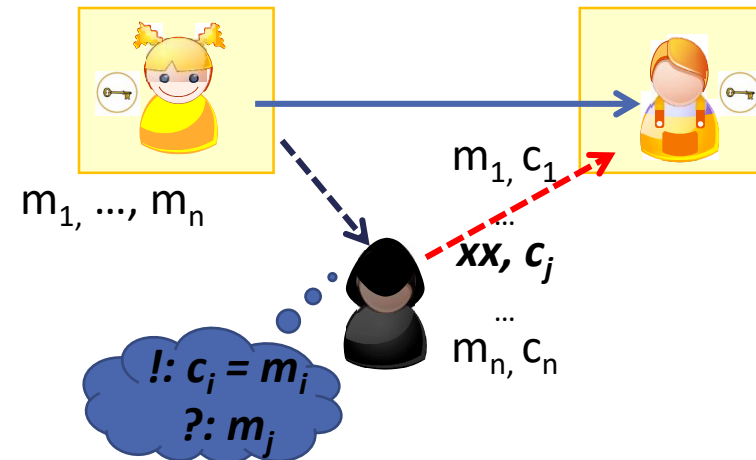
- despite concealed key
- asking Alice to encrypt  $m_a$
- learn about  $m_j$  (or key)



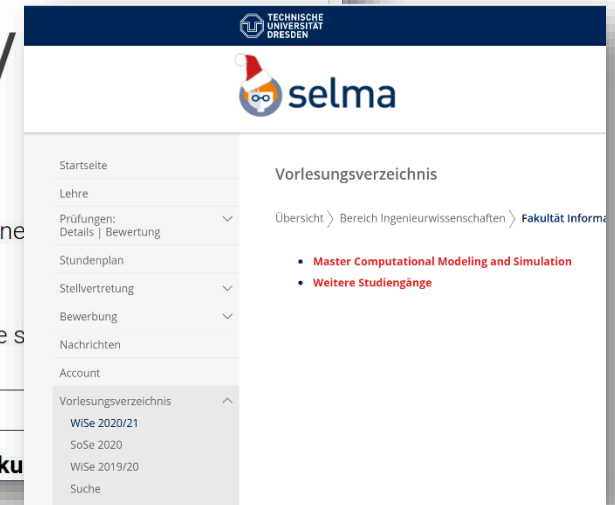
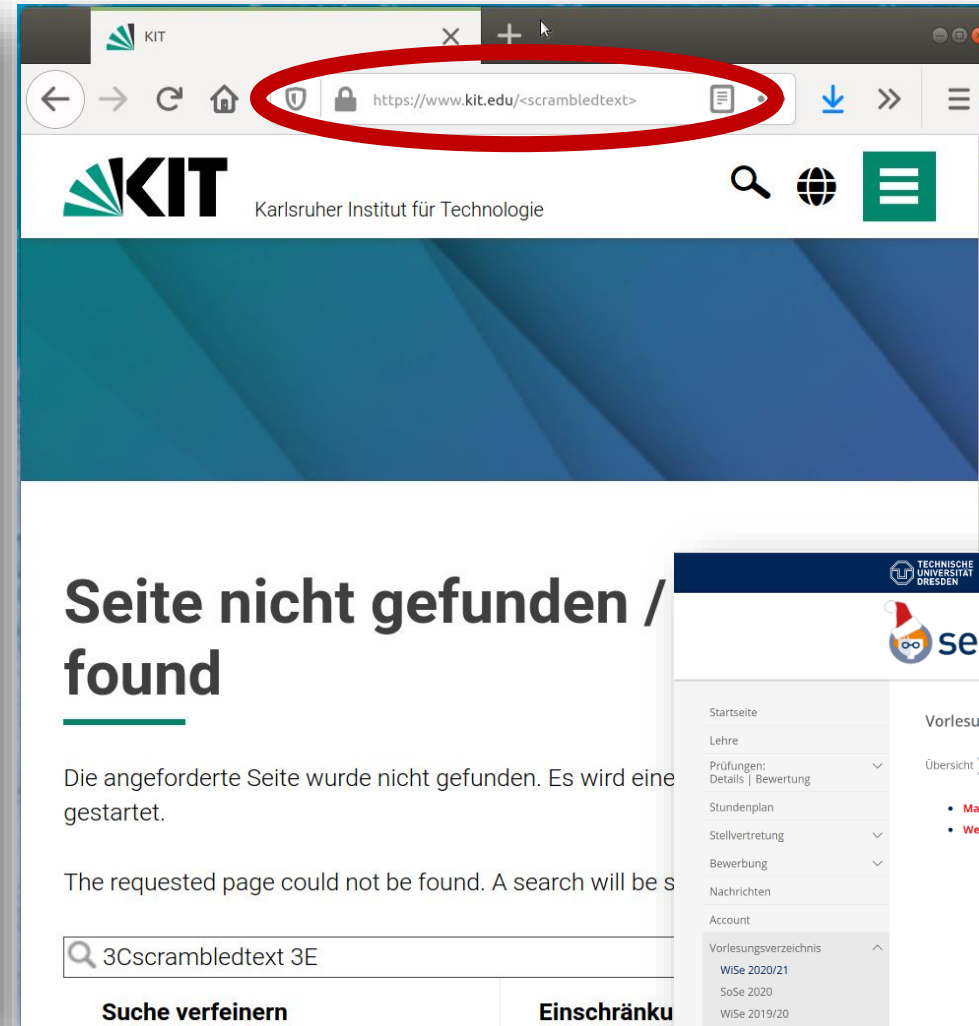
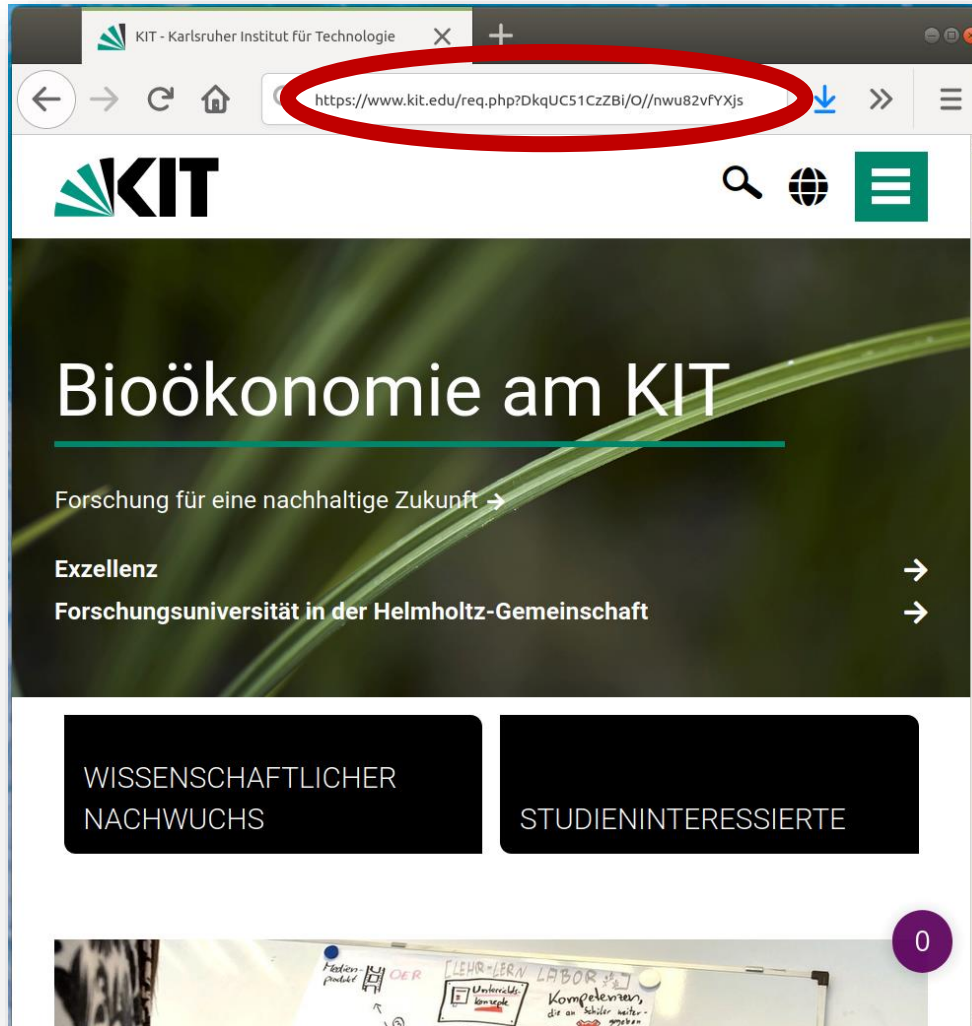
Asymmetric crypto: non-modifying („passive“) attack...

## ■ **Chosen-ciphertext attack:**

- despite concealed key
- asking Bob to decrypt  $c_a$
- learn about  $m_j$  (or key)



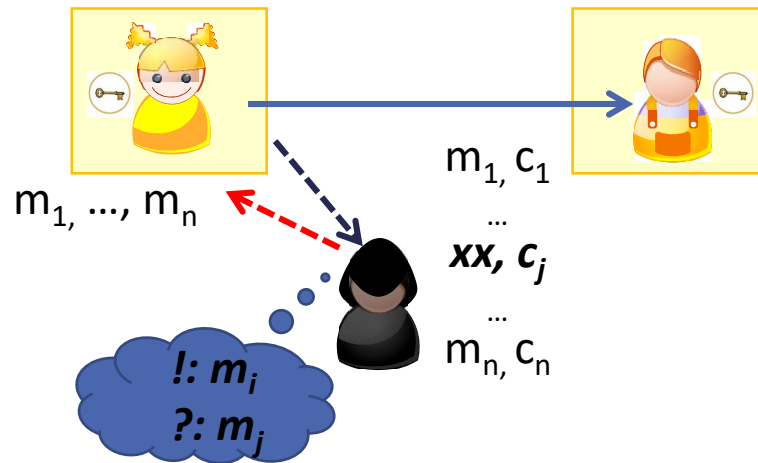
# CCA Oracles in Reality...



# Security Notions ctd.

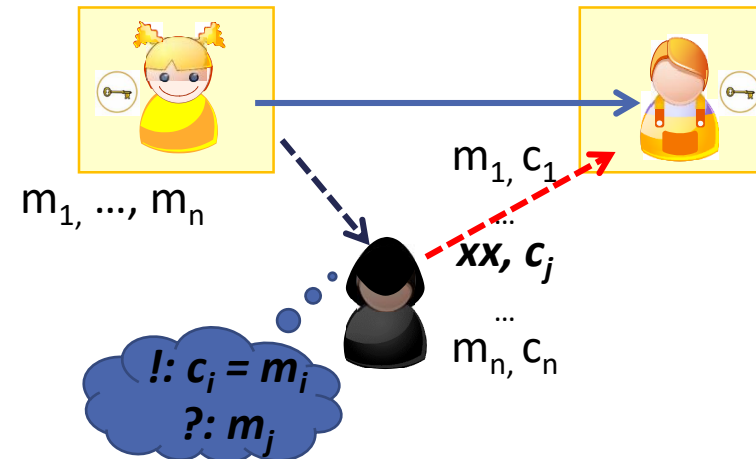
## ■ **Chosen-plaintext attack:**

- despite concealed key
- asking Alice to encrypt  $m_a$
- learn about  $m_j$  (or key)



## ■ **Chosen-ciphertext attack:**

- despite concealed key
- asking Bob to decrypt  $c_a$
- learn about  $m_j$  (or key)



- **Strong adversary, realistic!**

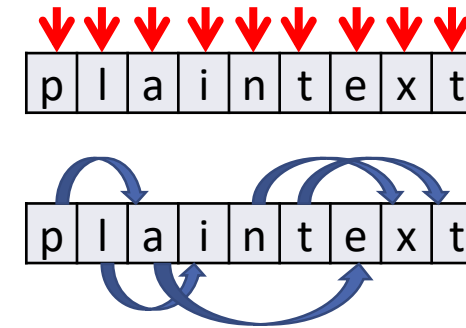
Asymmetric crypto: non-modifying („passive“) attack...

# Constructing Ciphers

# Constructing Ciphers

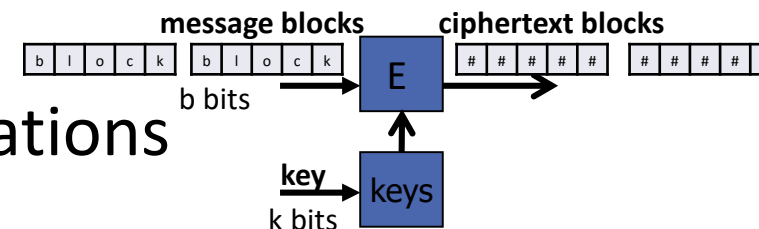
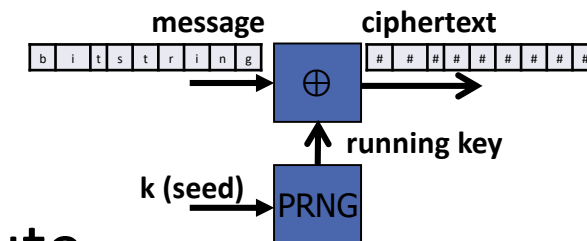
## ■ Operations for Encryption

- *Substitution*      exchange symbols for others
- *Transposition*    permute symbols systematically



## ■ Processing the plain text

- *Stream ciphers*    Generate key bits and substitute
- *Block ciphers*     Pseudo random permutations



# Constructing a Pseudo Random Permutation

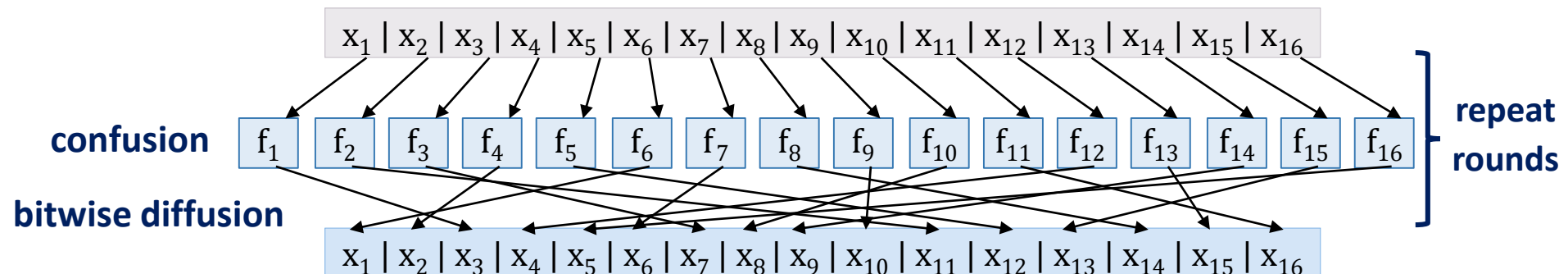


## Original idea of the confusion – diffusion paradigm (Shannon, 1949):

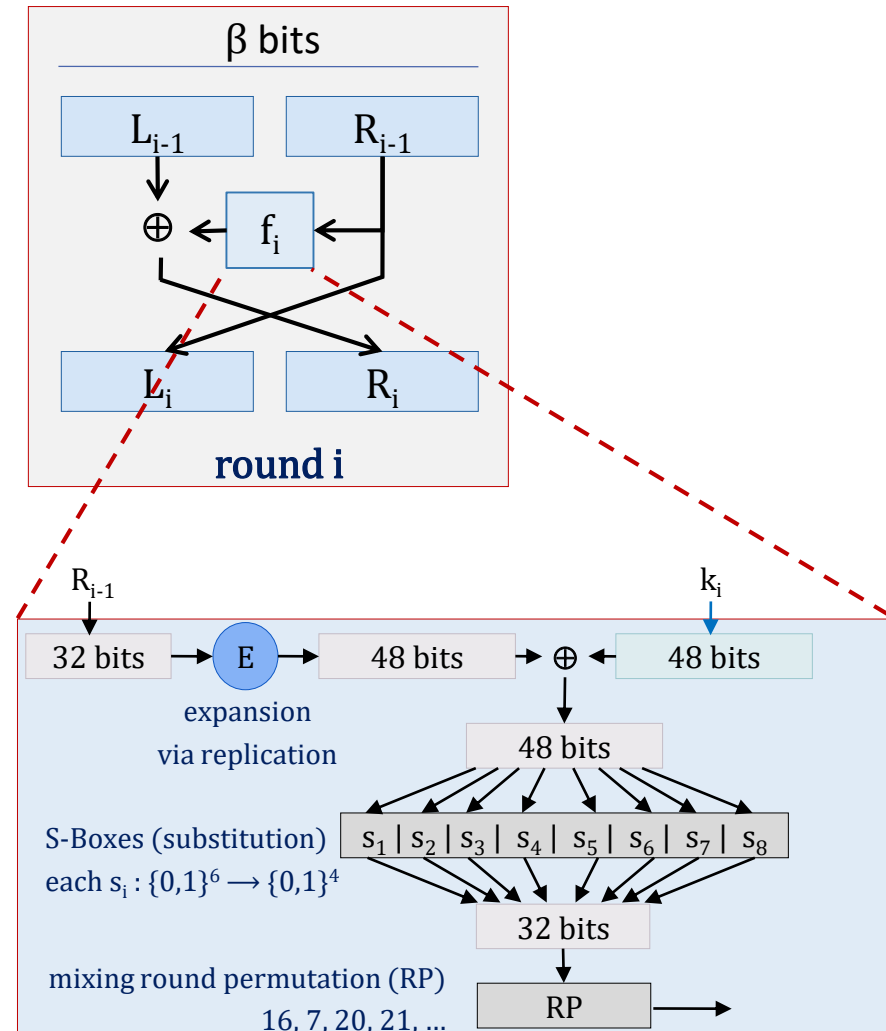
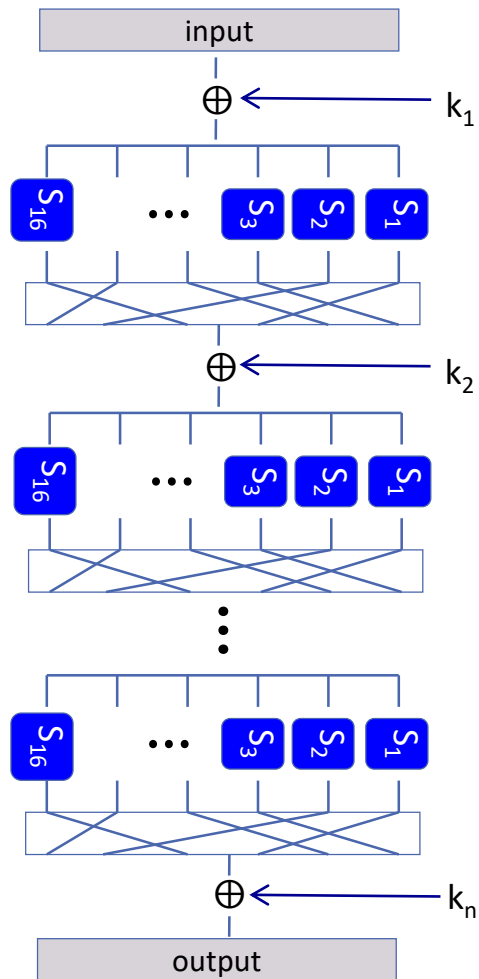
- Construct a random-looking permutation  $F$  with large block size using random-looking permutations  $\{f_i\}$  with smaller block sizes.
- Create product cipher with confusion step (hide relation between CT and  $k$ ) and diffusion step (distribute redundancy of PT)
- With **avalanche effect**: Bit-flip in input should change half of the output bits, every output bit should depend on all input bits.

## Construction:

- To construct  $F_k : \{0,1\}^{128} \rightarrow \{0,1\}^{128}$ :
  - Combine  $f_1, \dots, f_{16}$  random-looking permutations  $f_i : \{0,1\}^8 \rightarrow \{0,1\}^8$  (s-boxes),
  - Perform subsequent bitwise transposition
  - Rinse and repeat (generate key per round,  $\oplus$  vs round input)



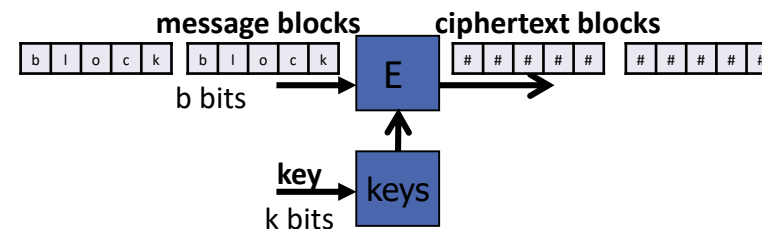
# Examples of PRPs: AES and Feistel Networks (DES)



But PRPs aren't ciphers...

# From PRPs to Block Ciphers (Mode of Operation)

- AES/DES are PRPs with fixed input length (64 or 128 bits)
- Messages in reality sometimes are slightly longer...

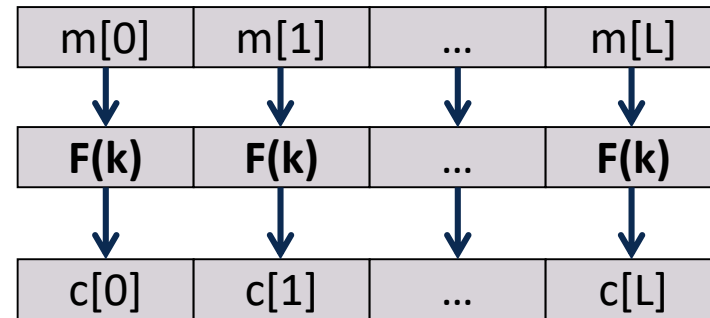


- ***How can we encrypt longer messages using PRPs?***

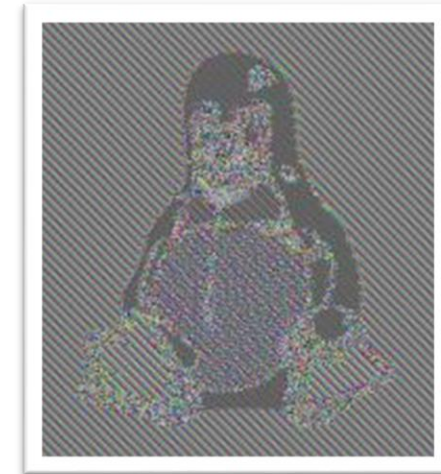
# Electronic Code Book Mode

**FAIL**

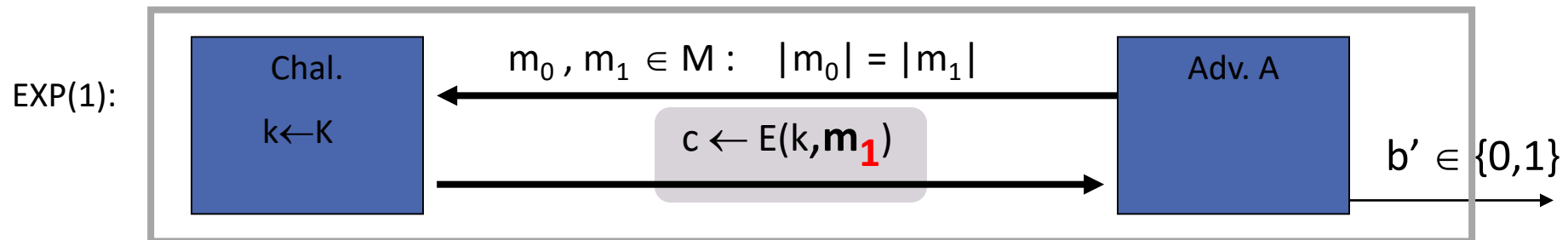
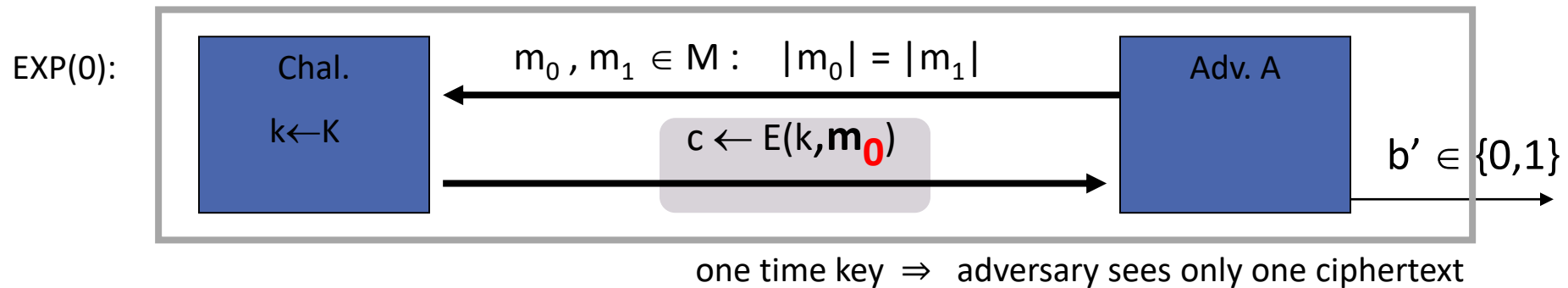
- Independently encrypt every block with a PRP:



- ECB encryption is *deterministic*
- ⇒ identical PT blocks → same CT blocks
- Is this “secure” (how)?*

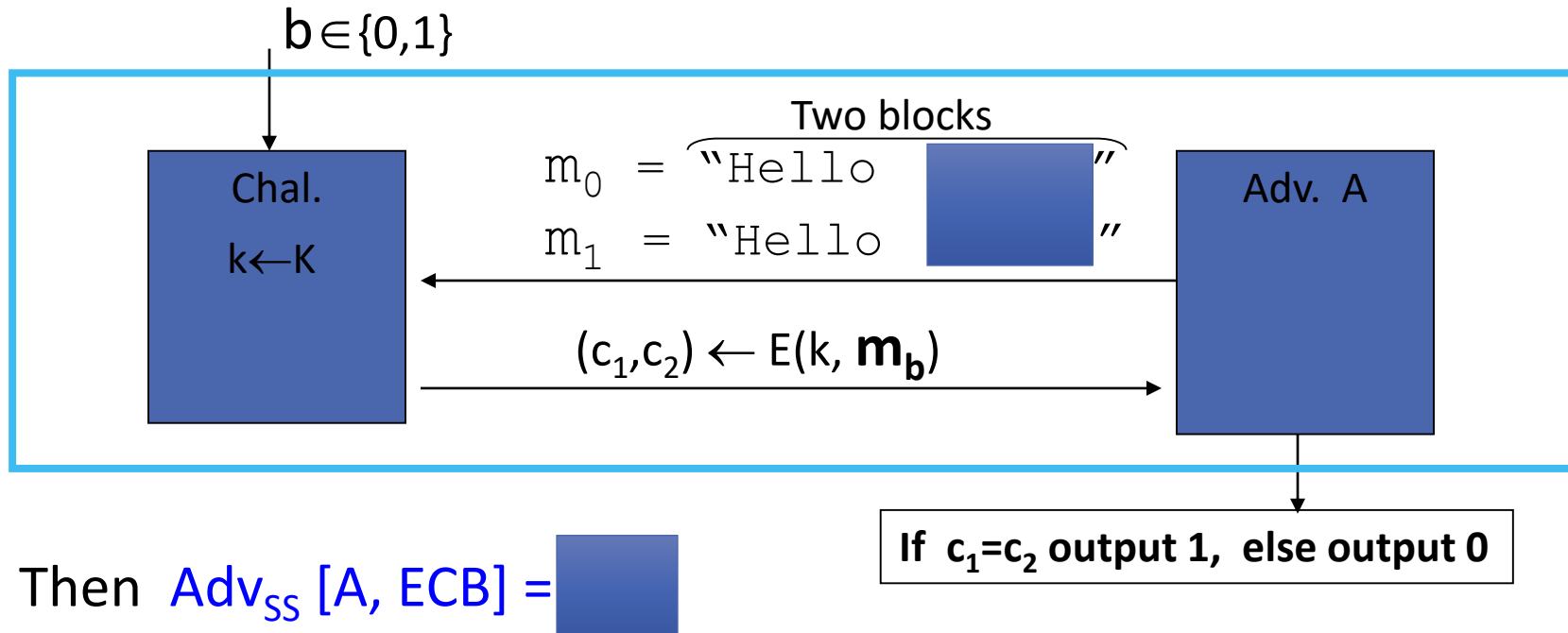


# Semantic Security (One Time Key)



$$\text{Adv}_{\text{SS}}[A, \text{ECB}] = \left| \Pr[\mathbf{EXP}(0)=1] - \Pr[\mathbf{EXP}(1)=1] \right| \text{ should be "neg."}$$

# ECB not semantically secure



- ECB is not semantically secure for messages of more than one block.

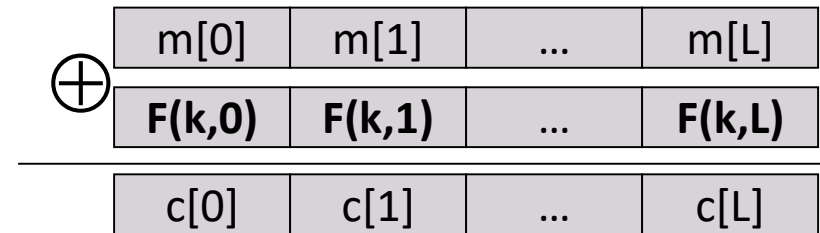
# Secure Modes of Operation

- ECB with deterministic PRP is insecure:
  - Two identical PT yield same CT if same key is used
  - Two identical PT blocks yield identical CT blocks
  
- So what can we do?
  - One-time key (internal):
    - Encrypt every block differentially
  - Many-time key (external):
    - Randomize, add random value to the input

# Counter Modes and Chaining

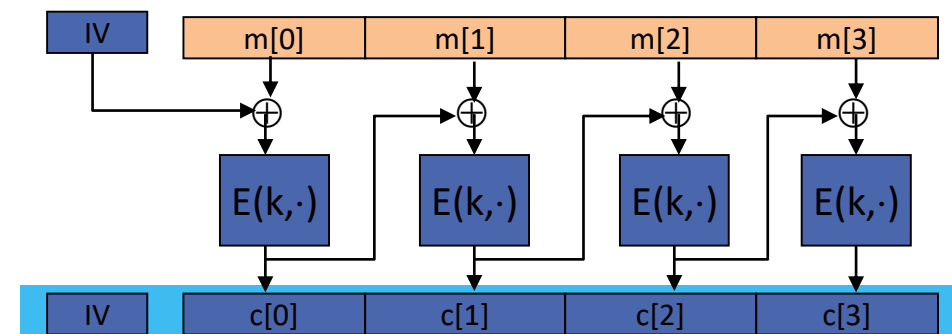
## (1) Encrypt each block differently:

- integrate (changing) value in the encryption of each block
  - Nonces:  $c_i = E(k, n_i, m_i) = E(k, (n_i, m_i))$  or  $E((k, n_i), m_i)$ ?
  - ...and transmit all  $n_i$ ?
  - Counters to the rescue!



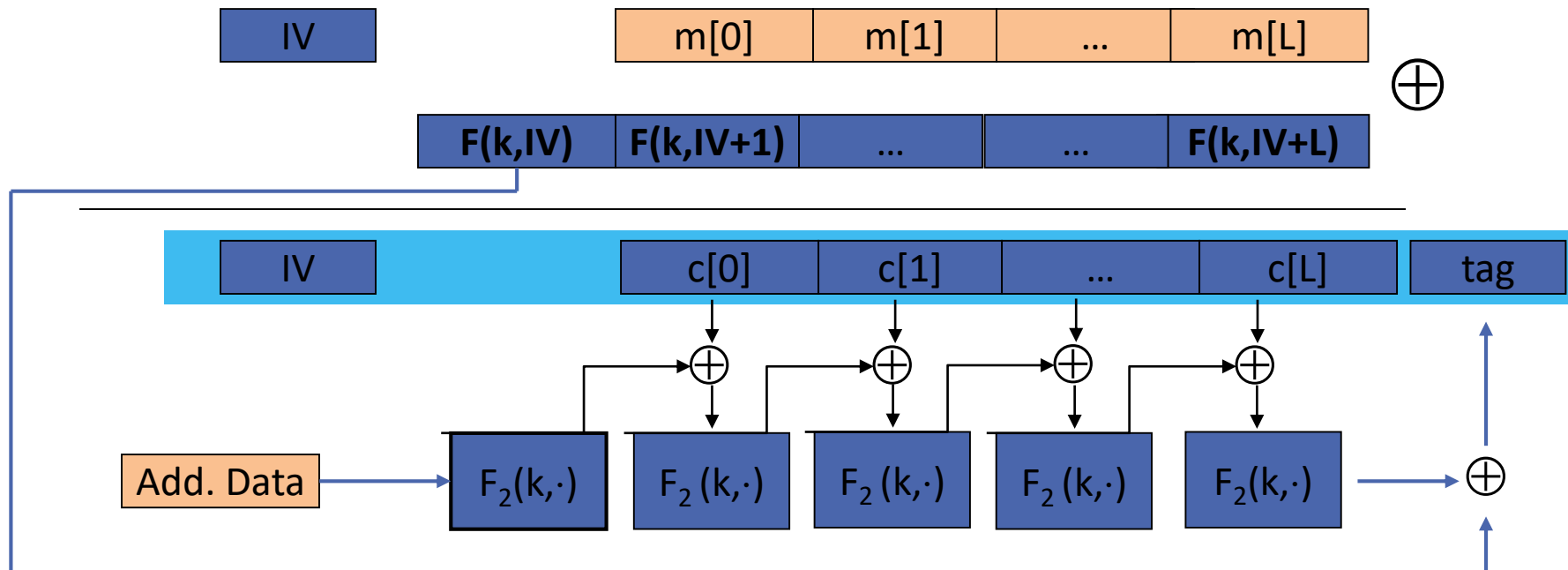
## (2) Many-time key (random initialization vector):

- integrate (changing) value in the encryption of each message!
  - May we don't need indepent randomness *for each block*?
  - Choose random IV
  - Chain the encryption of the blocks



# Galois Counter-Mode

- Anticipating future module, we may want „authenticated encryption“
- May be also unencrypted (header) data shouldn't be manipulated („AEAD“)
- Idea: extend process by parallel authentication function



# Intermediate Summary

- Attackers and adversary model
- Threats
- Security Objectives (CIA)
- Security services
- Definitions and Formalizing Security as a game
- Perfect security/ semantic security
- One-Time-Pad, Stream cipher
- Block-Cipher and modes of operation

# Next stop: Integrity

# Constructing an integrity service

- Aim: Detect unauthorized manipulation (*recall: we've got nothing to hide!*)
- Map message of arbitrary length to a tag of (short) fixed length
- Tag authenticates sender and proves absence of modification
- → „Message Authentication Code“ (MAC)

- Algorithms

- $\text{KeyGen}() \rightarrow k \in K$

- $S(k,m) \rightarrow t$

- $V(k,m,t) \rightarrow \{0,1\}$

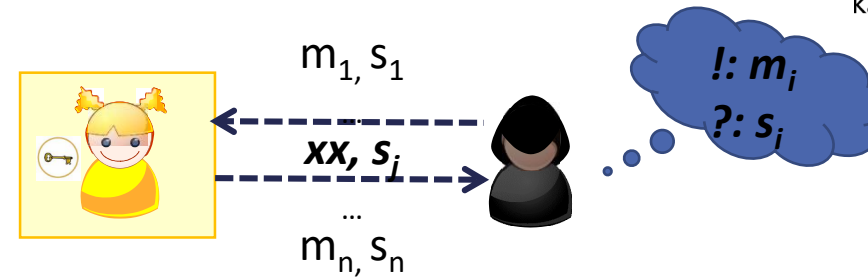


with  $m \in \{0,1\}^n$ ,  $t \in \{0,1\}^t$  ( $n \gg t$ )

# Security Notions: Existential Forgery

- **Chosen Message Attack:**

- given  $s_1, s_2, \dots, s_n$  for chosen  $m_i$



- (Variations: known verification key / known signature attacks)

- **Existential Forgery:**

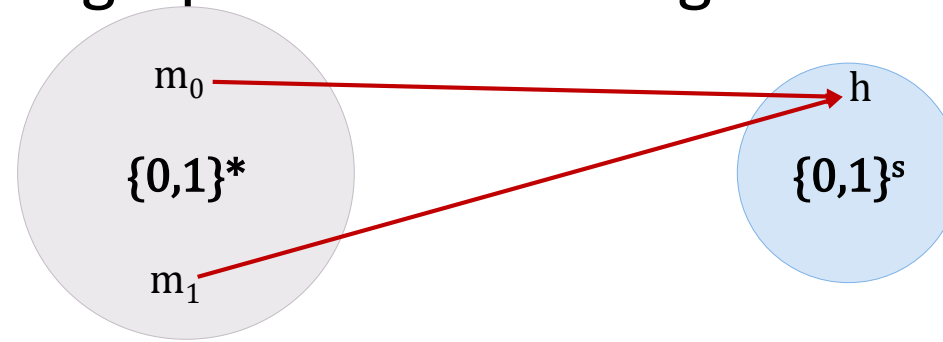
- Produce arbitrary, valid tuple  $(m, t)$  (arbitrary message even gibberish)
- $\Rightarrow$  adversary cannot produce valid tag for new message
- $\Rightarrow$  can't even produce  $(m, t')$ ,  $(m', t)$  for  $(m, t)$  and  $t' \neq t$  or  $m \neq m'$

- *General notions:*

- *Exist. forgery < selective forgery < universal forgery < total break*

# Constructing MACs

- Ideas:
  - Authenticating the sender
  - Map message to fingerprint of fixed length



→ secret

→ Hash functions

- Secret remains secret (no encryption involved)
- Smallest modification of  $m$  unpredictable in  $t$

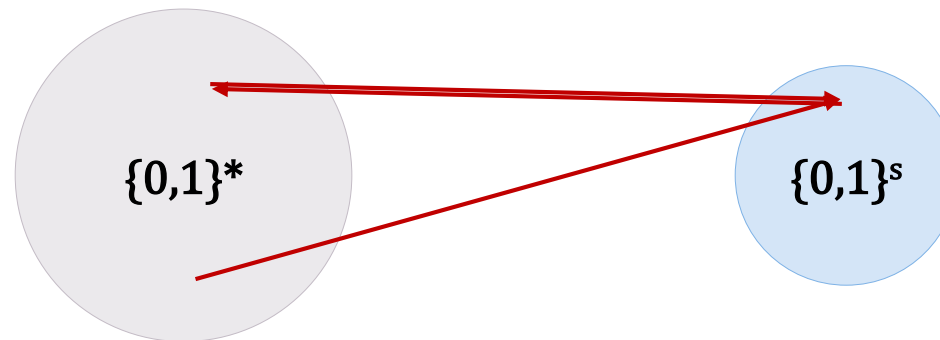
→ one-way functions

→ „Chaos“

# What were those fingerprints?

# Requirements for *secure* Hash/MACs

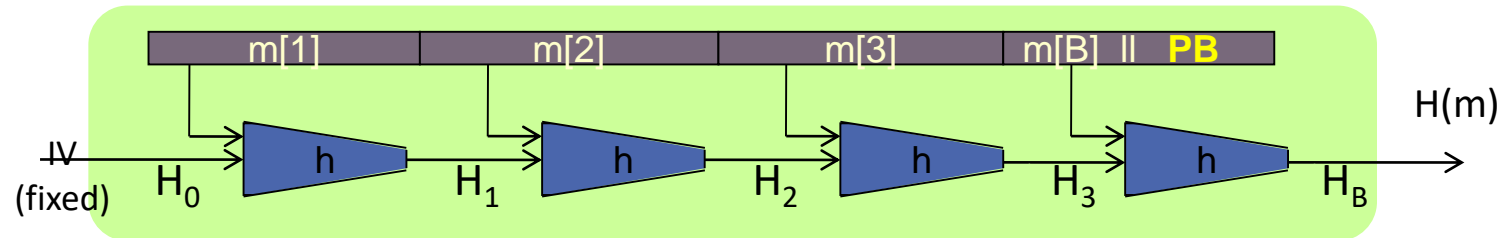
1. One-way property (preimage resistance)
2. Strong collision resistance (collision resistance)
3. Weak collision resistance (2nd preimage resistance)



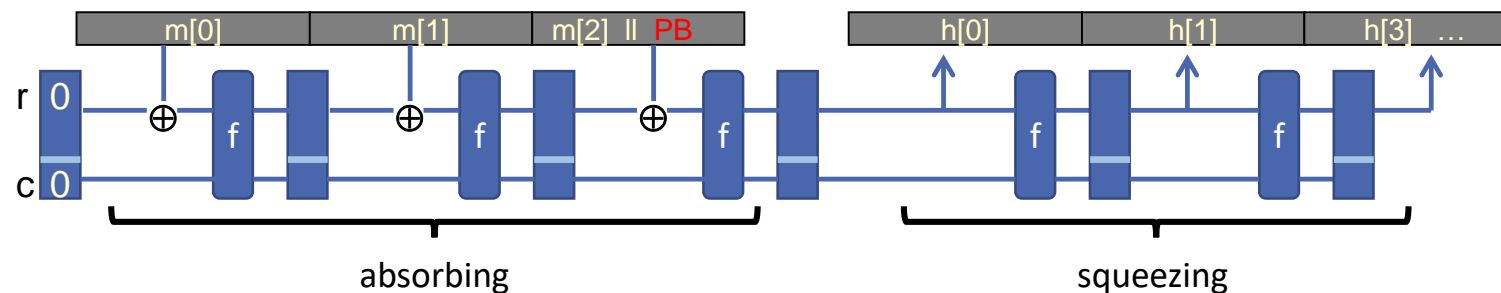
- *Why don't we design **collision free** hash functions?*

# Two examples for secure Hash functions

- Merkle-Damgård: Cascade of PRP without encryption (MD5/SHA1..)



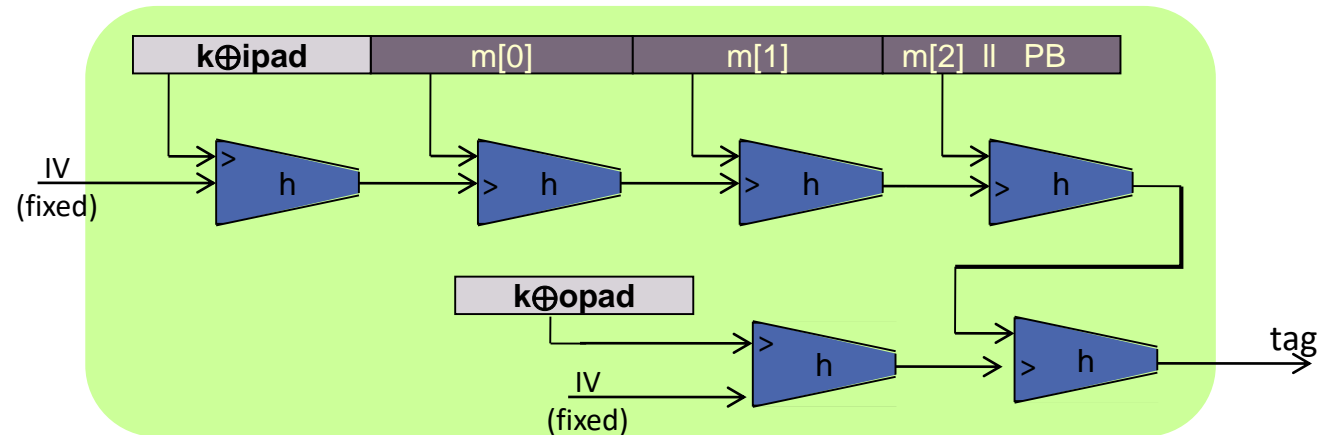
- SHA-3: Ingesting the entire message into pseudorandom state



# Finally: Ensuring Integrity!

# Constructing MACs from Hash functions

- Security requirements of hash functions (one-way/collision resistance)
- Security requirements of MACs (no Existential Forgery)
  
- MAC:= Enc(k,h(m))?
- HMAC:

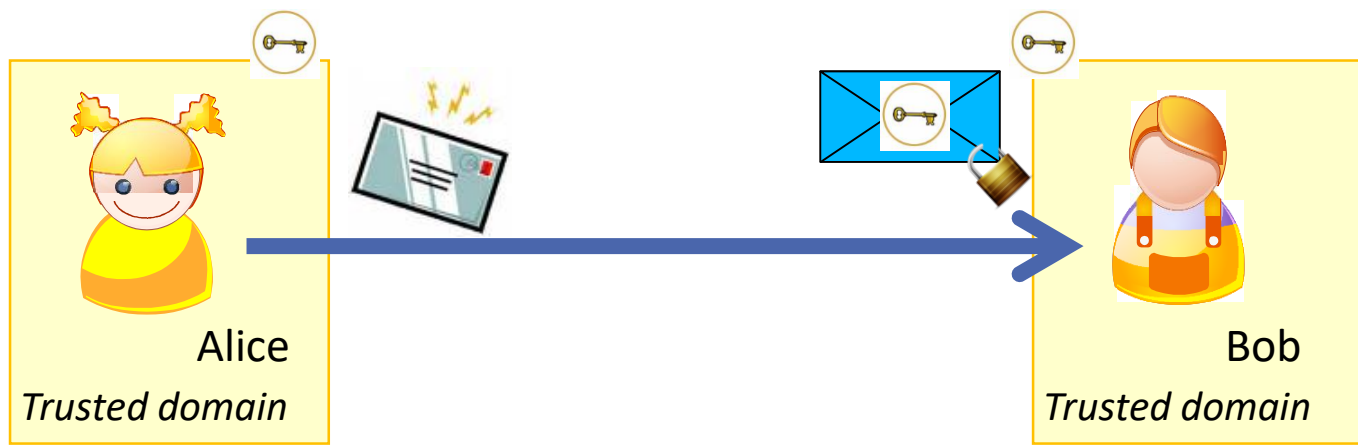
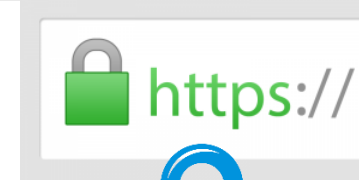


# Intermediate Summary

- Threats for communication and networks
  
- Security objectives
  - Confidentiality
  - Integrity
  - Availability
  
- Definition and Design of Security Services
  
- Game-based analysis of encryption and signing
- Perfect secrecy and semantic security
  
- Design of stream ciphers (OTP) and block ciphers/modes of operation
- Design of hash functions and Message Authentication Codes (MACs)

# Now where do we get those keys?

# Key Distribution/Agreement



## Alternatives:

- Alice/Bob agree mutually
  - In advance
  - When establishing their connection
- Alice and Bob delegate trust into a third entity („TTP“, „KDC“)

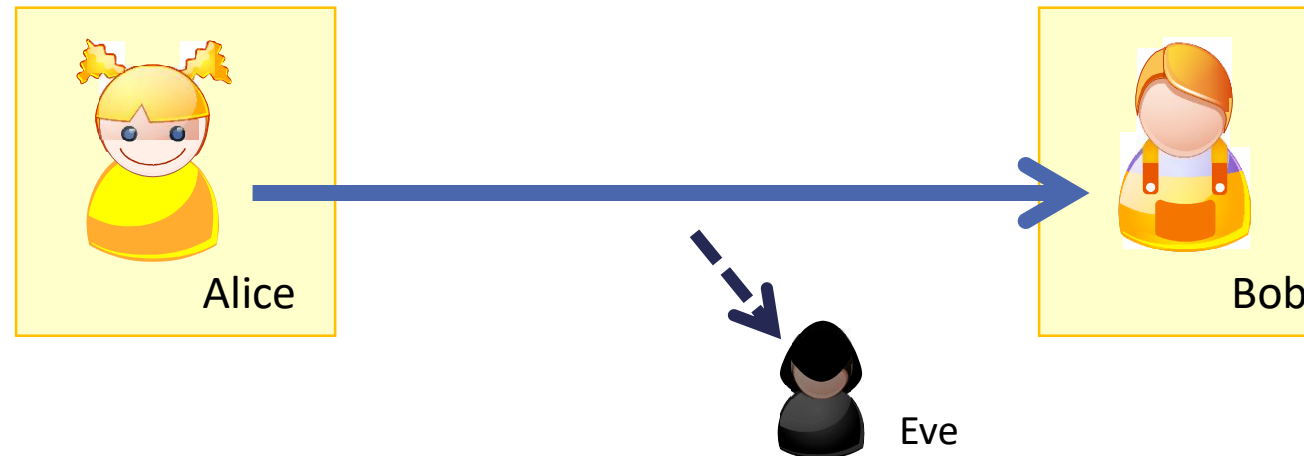


# First Attempt: A Passive Adversary



Eve can:

- Eavesdrop on the channel



- Can Alice and Bob securely exchange a key?

# Key Agreement

## **Aim:**

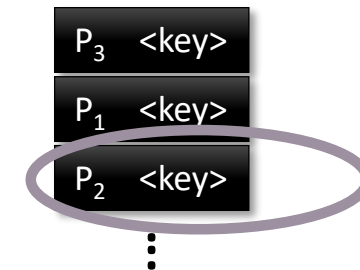
- Given open and public communication
- Output: A key secret to Alice and Bob

## Initial Idea (Merkle, '74):

- Alice creates  $2^{32}$  Puzzles (with an Index  $P_i$  and key)
- Alice shuffles the puzzles and sends them to Bob
- Bob chooses one at random and solves it
- Bob informs Alice of the index  $P_j$ , both know the respective key.



Ralph Merkle, Martin Hellman, Whitfield Diffie



How much effort does Eve have to invest?



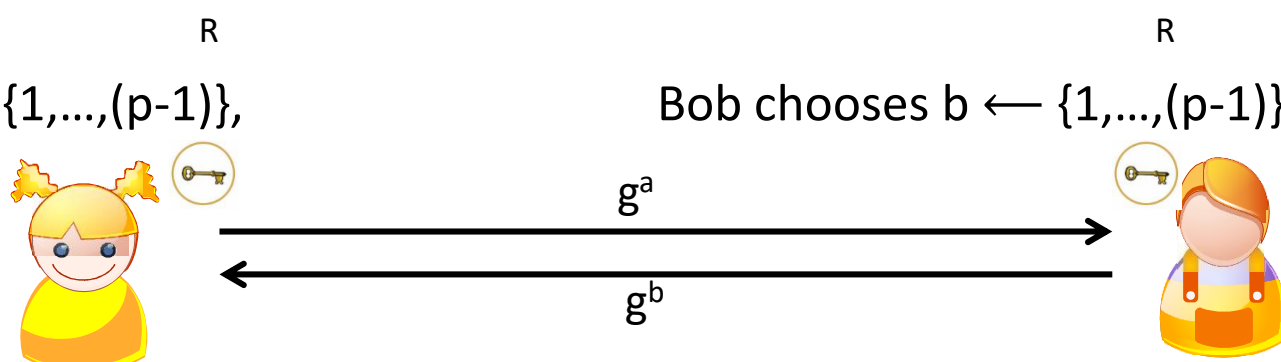
# Polynomial Advantage: Diffie-Hellman(-Merkle)

Can we make Eve's life more difficult?

Observations:

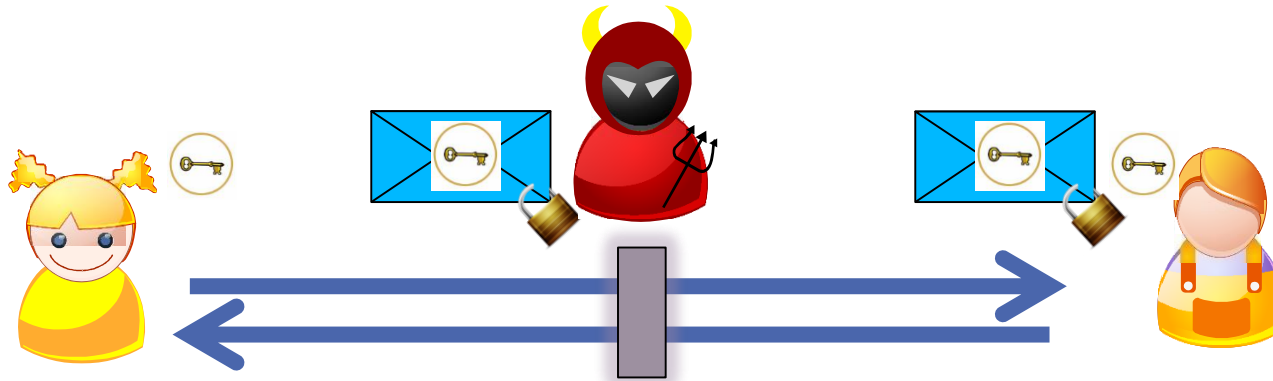
1. Discrete Logarithm (some inversions) hard to solve
2. Power Law:  $(g^x)^y = g^{xy} = g^{yx} = (g^y)^x$

Idea:

- Choose cyclic group  $\mathbb{Z}_p^*$ , generated over  $g$ , and  $\varphi(p) = p-1$
  - Alice chooses  $a \leftarrow \{1, \dots, (p-1)\}$ , Bob chooses  $b \leftarrow \{1, \dots, (p-1)\}$
- 

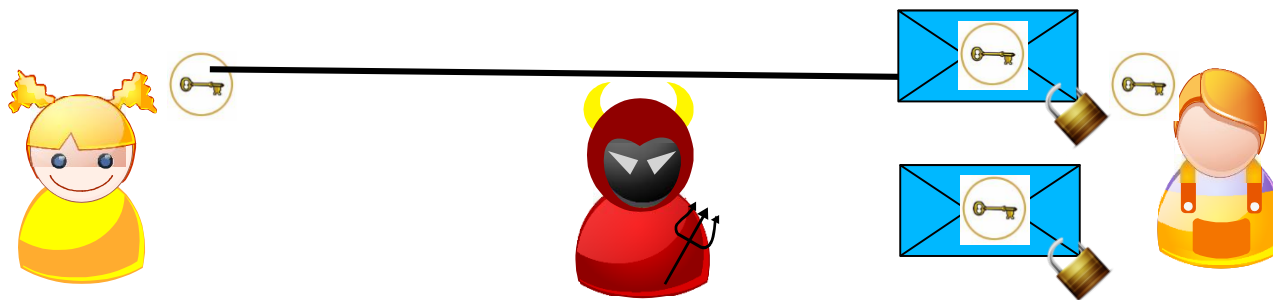
$g^a$  (top arrow) and  $g^b$  (bottom arrow)
- Alice calculates:  $(g^b)^a \bmod p = g^{ab} \bmod p =$  Bob calculates:

# Two Specific Attacks



Identify/  
authenticate  
parties pro-  
actively or with  
external help

- Man-in-the-middle

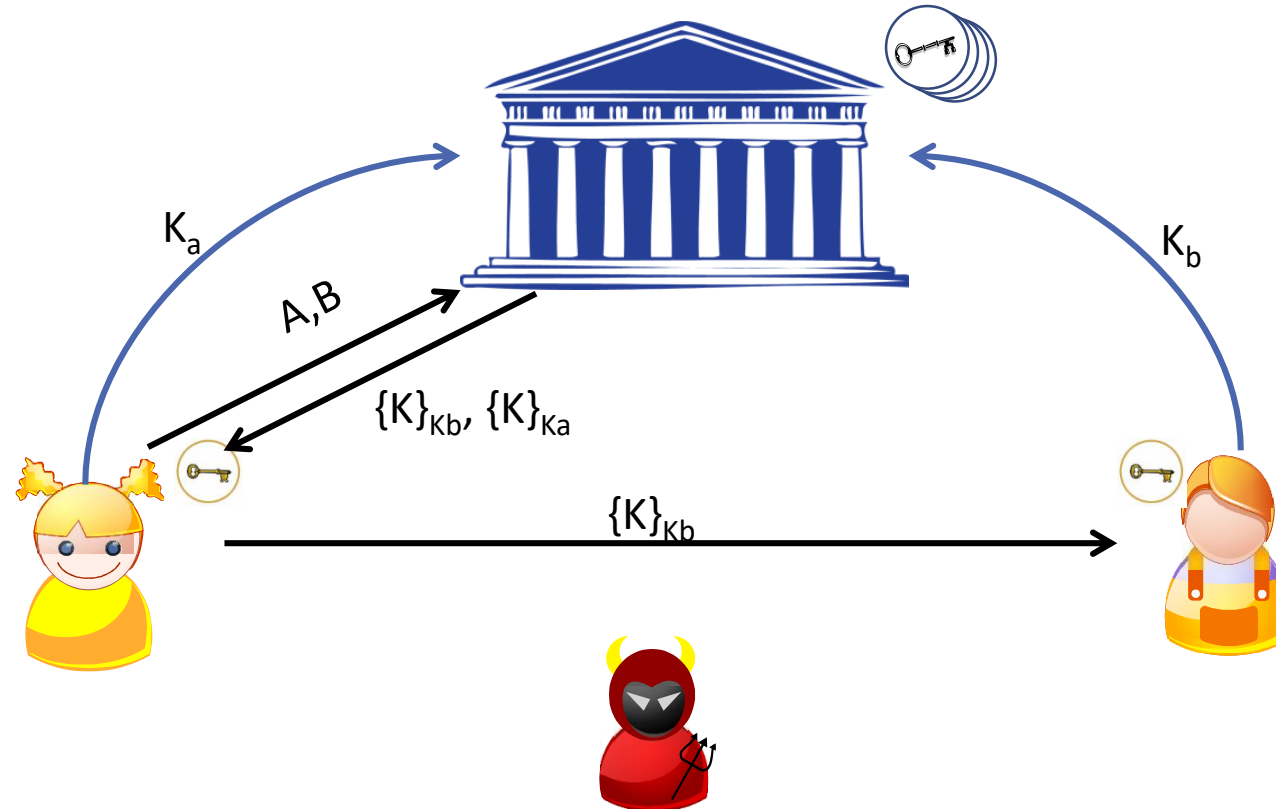


Ensure  
freshness!

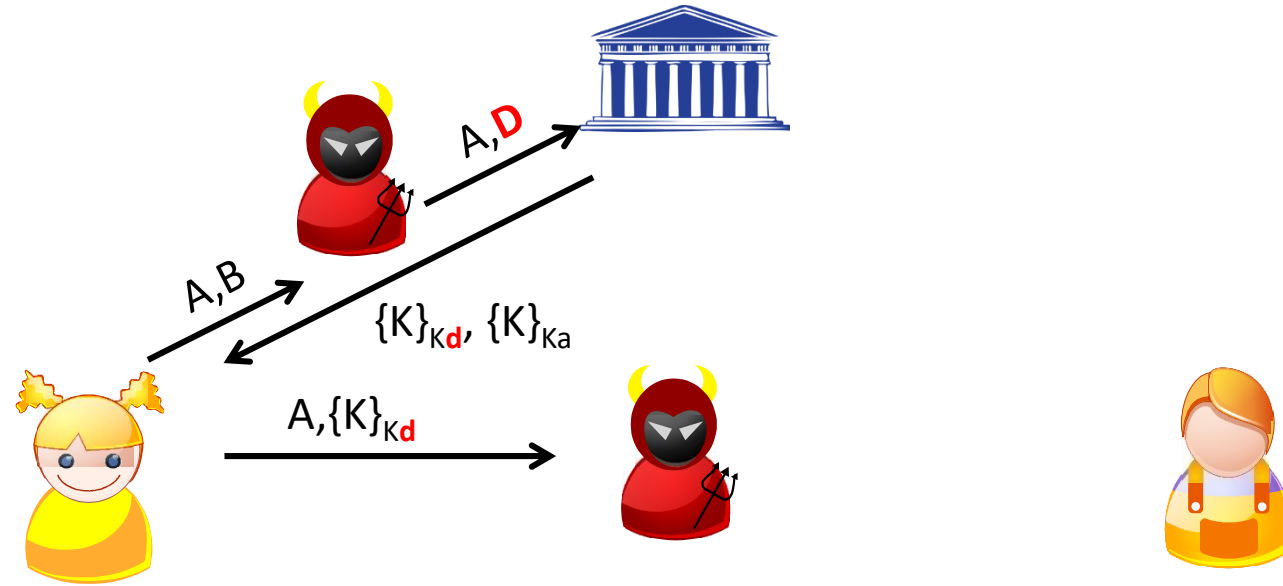
- Replay

# Key distribution with an arbiter

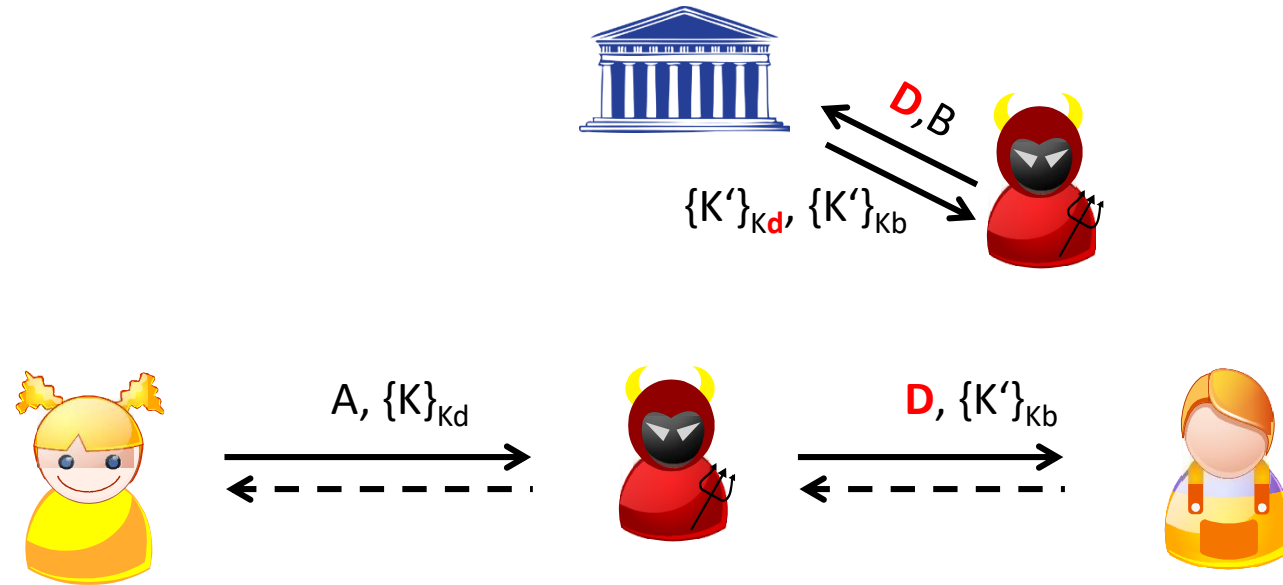
# A simple (insecure) Protocol



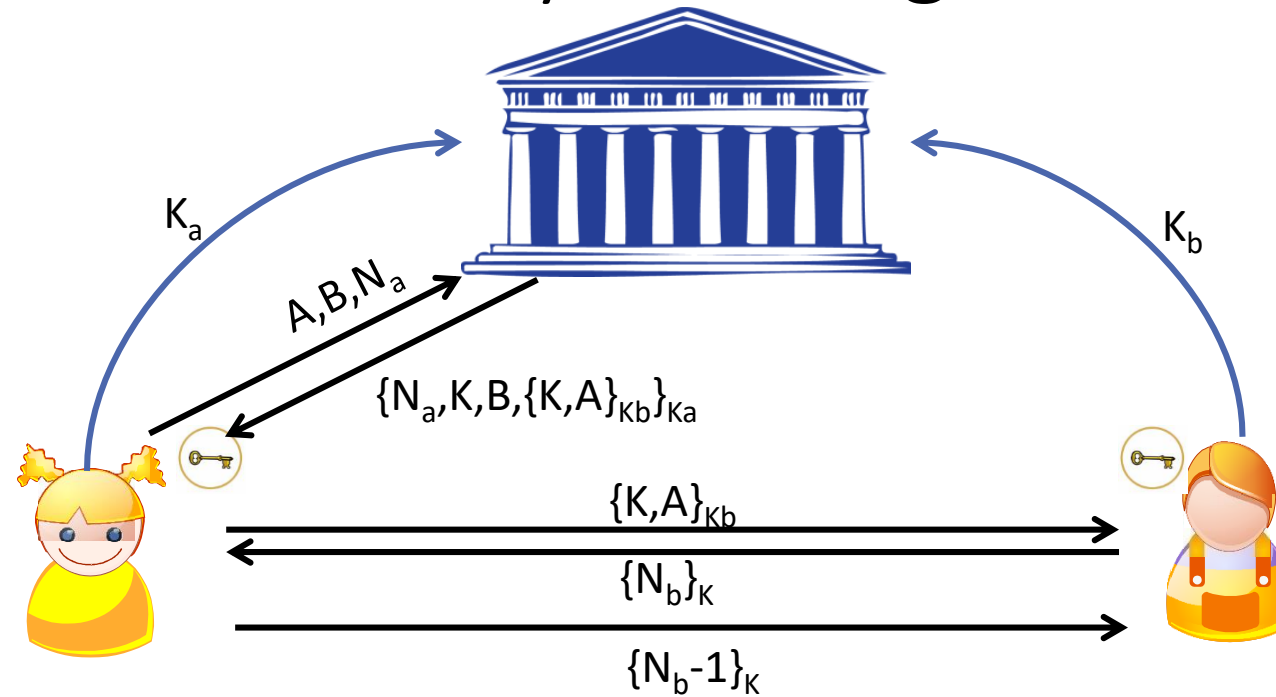
# Man in the Middle - 1



# Man in the Middle - 2



# Schroeder-Needham Key Exchange



## Observation:

- Prevent *MitM* through *Identification*
  - Alice and Bob are authenticated through knowledge of key  $K$  (need  $K_a$  or  $K_b$ )
- Prevent *Replay* by „Nonces“
- *Extension: Timestamps for Freshness, in case that Mallory guesses an old key*

# Applying this to encryption...

# Asymmetry (RSA) – Math Background

- Given randomly chosen, large primes  $p$  and  $q$ :
  - Multiplying  $n = p \cdot q$  is simple
  - Factoring  $n$  to prime factors  $p$  and  $q$  is hard
  
- Given cyclic multiplicative groups: Multiplication is trivial, Division hard...
  
- Knowing prime factors using ext. Eucl. algorithm, simply calculate the mult. inverse:

- For cyclic mult. group  $Z_n^*$  and  $e$  (coprime to  $n$ ):

$$e^{-1} : \quad \gcd(e, n) = 1: d \cdot e + k \cdot n$$

$$k \cdot n \equiv 0 \pmod n$$

$$e \cdot e^{-1} \equiv 1 \pmod n$$

$$\Rightarrow \quad e^{-1} = d$$

# RSA – Key Generation

- Each participant
  - Chooses two independent, large random primes  $p, q$
  - Calculates  $N = p \cdot q$  and  $\varphi(N) = N - p - q + 1 = (p-1)(q-1)$
  - Chooses random  $e$ , with  $2 < e < \varphi(N)$ ,  $\gcd(e, \varphi(N)) = 1$
  - And calculates  $d$  such that  $e \cdot d = 1 \pmod{\varphi(N)}$
- Subsequently:
  - Store  $(p, q, d)$  (as secret key  $sk$ )
  - Publish  $(N, e)$  (as public key  $pk$ )

# RSA – Encryption and Decryption

- **Permute („Encryption“)**

- Given  $pk = (N, e)$ :

- $\text{RSA}(pk, m): \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$  ;  $c = \text{RSA}(e, m) = m^e \bmod N$  (in  $\mathbb{Z}_N$ )

- **Invert (“Decryption“)**

- Given  $sk = (p, q, d)$ :

- $m = \text{RSA}^{-1}(pk, c) = c^{1/e} = c^d = \text{RSA}(d, c) \bmod N$  (in  $\mathbb{Z}_N$ )

- $c^d = \text{RSA}(m)^d = m^{ed} = m^{k\varphi(N)+1} = (m^{\varphi(N)})^k \cdot m = m$  ( $c \in \mathbb{Z}_N^*$ )

- **Bonus: „Signing“ a message**

- Given  $sk = (p, q, d)$ :

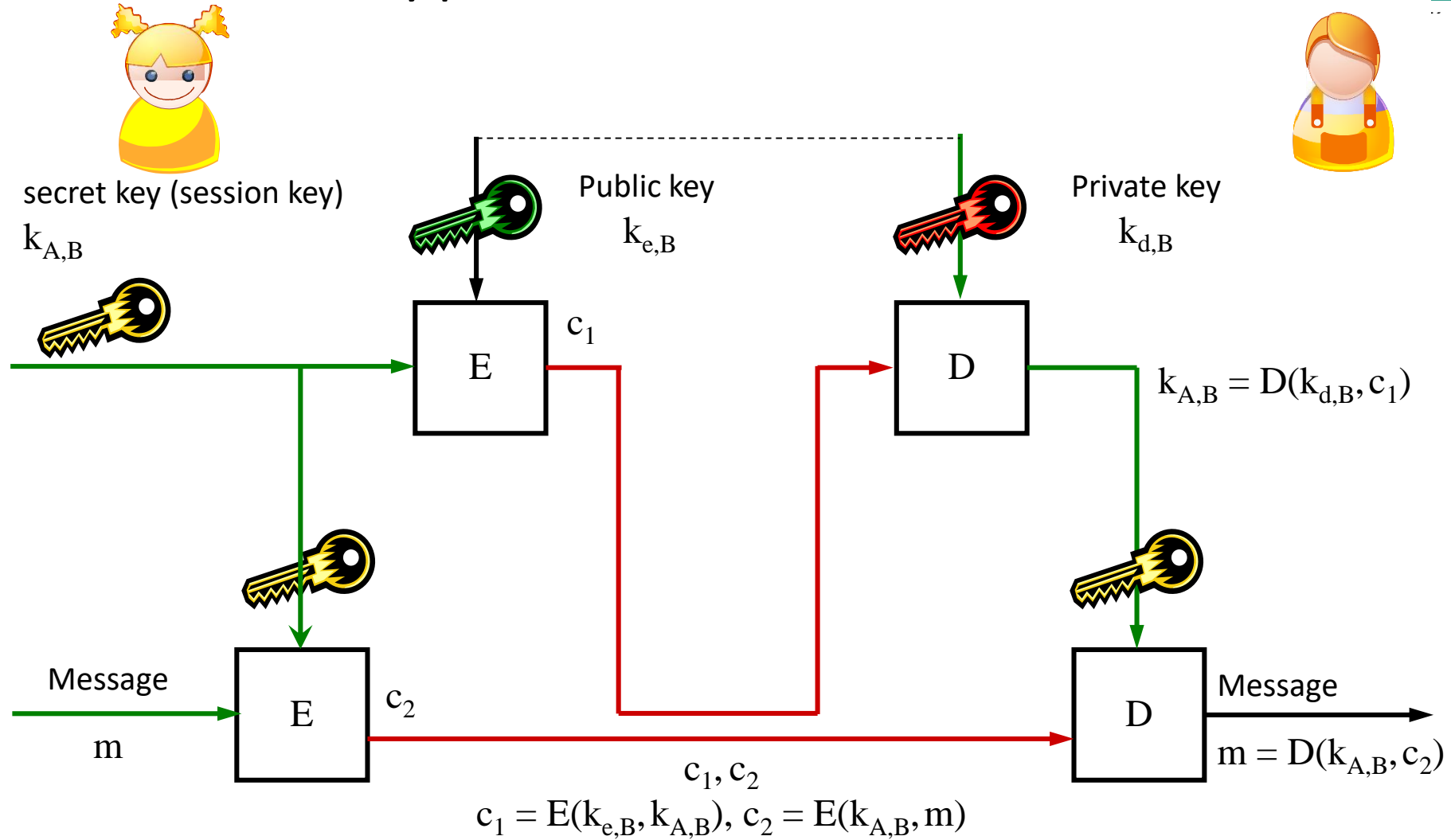
- $\text{tag} = \text{RSA}^{-1}(pk, h(m)) = \text{RSA}(d, h(m))$

# Factoring (prime decomposition)

- Theorem: all integers  $> 1$  are either prime or a product of primes.
- **Factoring**:
- Consider set of integers  $\mathbb{Z}_{(2)}(n) = \{ N=pq, \text{ where } p, q \text{ are } n\text{-bit primes} \}$
- Task: Find the prime factors ( $p$  and  $q$ ) of a random  $N$  in  $\mathbb{Z}_{(2)}(n)$
- Best known algorithm (NFS):  $\exp(\tilde{O}(\sqrt[3]{n}))$  for  $n$ -bit integers
- **Current world record**: RSA-768 (232 digits) (200 machine years)
- *Consumed enough energy to heat to boiling point 2 olympic pools...*
- *(Breaking RSA-2380 equivalent to evaporating all water on earth)*

Lenstra, Kleinjung, Thomé

# Hybrid / ISO Encryption



# Summary

- Adversaries, Attacker Models, and Threats
- Security Objectives (CIA) and Security Services
- Defining and formalizing security as a game
- Perfect secrecy / semantic security
- One-Time-Pad, Stream-Cipher
- Block-Cipher and modes of operation
- Hash functions and MACs
- Key Agreement direct and with arbiter
- Asymmetric encryption and signatures