https://brown-csci1660.github.io

CS1660: Intro to Computer Systems Security
Spring 2026

Lecture 4: Confidentiality II

Instructor: **Nikos Triandopoulos**

February 3, 2026

BROWN

CS1660: Announcements

- Course updates
  - Project 1 “Cryptography” is out
  - Reminder
    - Enrolling to advanced version (via 1620 or 2660) is only for educational purposes
    - Only for this term: It’s unrelated to “Capstone”

Last class

- Cryptography
  - Secret communication
    - Symmetric-key encryption & classical ciphers
    - Perfect secrecy & the One-Time Pad

Today

- Cryptography
  - Encryption in practice
    - Computational security, pseudo-randomness
    - Stream & block ciphers, modes of operations for encryption, DES & AES
    - Introduction to modern cryptography

4.1 Symmetric encryption, revisited: OTP with pseudorandomness

Big picture

Secret communication

- We learned what it means for a cipher to be perfectly secure
- We learned that the simple OTP cipher achieves this property
  - XOR (**mask**) message (**once**) with the secret key (**random pad**)
  - …but it cannot be used in practice!
- We learned how we can fix this problem
  - just use OTP with a freshly-generated "**random looking**" pads
  - **mask** each message **once** with a **pseudorandom pad**

Big picture (cont.)

Secret communication

- But there is no free lunch…
  - if we **mask** each message **once** with a **pseudorandom pad**, we must lose **perfect** secrecy!
  - because "**random looking**" pads are not **random**…
- But not perfect won't be imperfect – it will be close to perfect
  - for all practical purposes
    - "**random looking**" pads will be **as random as** **truly random** ones
    - **OTP + pseudo-randomness** will be **as secure as** (standard) **OTP**

Perfect secrecy & randomness

Role of randomness in encryption is **integral**

- in a perfectly secret cipher, the ciphertext **doesn't depend** on the message
  - the ciphertext appears to be **truly random**
  - the uniform key-selection distribution **is imposed also onto** produced ciphertexts
    - e.g., c = k XOR m (for uniform k and any distribution over m)

When security is computational, randomness is **relaxed** to "pseudorandomness"

- the ciphertext appears to be "**pseudorandom**"
  - it **cannot be efficiently distinguished** from truly random

Symmetric encryption as “OPT with pseudorandomness”

Stream cipher

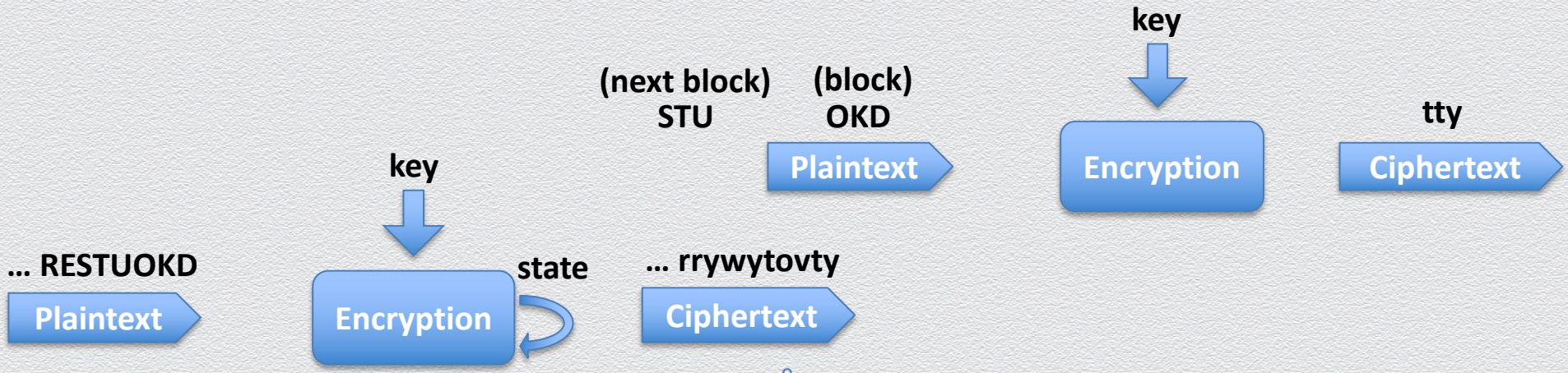
Uses a **short** key to encrypt **long** symbol **streams** into a **pseudorandom** ciphertext

- based on abstract crypto primitive of **pseudorandom generator (PRG)**

Block cipher

Uses a **short** key to encrypt **blocks** of symbols into **pseudorandom** ciphertext blocks

- based on abstract crypto primitive of **pseudorandom function (PRF)**



4.2 Computational security

The big picture: OPT is perfect but impractical!

We formally defined and constructed the perfectly secure OTP cipher

- This scheme has some major drawbacks
  - it employs a very large key which can be used only once!
- Such limitations are unavoidable and make OTP not practical
  - why?

Now, what?

Our approach: Relax perfectness for cipher security

Initial model

- **Perfect secrecy** (or security) guarantees that
  - the ciphertext leaks (absolutely) **no extra information** about the plaintext
  - (unconditionally) to adversaries of **unlimited computational power**

Refined model

- **Computational security** guarantees a relaxed notion of security, namely that
  - the ciphertext leaks **a tiny amount of extra information** about the plaintext
  - to adversaries with **bounded computational power**

Computational security

General concept in Cryptography

Computational security of a cryptographic scheme guarantees that

- (1) the scheme can be broken only with **a tiny likelihood**
- (2) by adversaries with **bounded computational power**

In contrast to **perfect** or **information-theoretic** or **unconditional security**

- which is typically harder, more costly or, often impossible, to achieve

Computational security (cont.)

General concept in Cryptography

- *de facto* model for security in most settings
  - based on an underlying hardness (computational) assumption
  - integral part of modern cryptography
  - still allowing for rigorous mathematical proof of security
- **Asymptotic** description of results

"A scheme is **computationally secure** if
any **efficient** attacker succeeds in breaking it
with at most **negligible** probability"

Computational security (cont.)

General concept in Cryptography

- entails two relaxations
  - security is guaranteed against **efficient** adversaries
    - if an attacker invests in **sufficiently large resources**, it may break security
    - goal: make required resources larger than those available to any realistic attacker!
  - security is guaranteed in a **probabilistic** manner
    - with some **small probability**, an attacker may break security
    - goal: make attack probability sufficiently small so that it can be practically ignored!

Security relaxation for encryption

Perfect security: |k| = 128 bits, M, $Enc_K(M)$ are independent, **unconditionally**

- no extra information is leaked to any attacker

Computational security: M, $Enc_K(M)$ are independent, **for all practical purposes**

- no extra information is leaked but a tiny amount
  - e.g., with prob. 2^{-128} (or much less than the likelihood of being hit by lighting)
- to computationally bounded attackers
  - e.g., who cannot count to 2^{128} (or invest work of more than one century)
- attacker's best strategy remains **ineffective**
  - random guess a secret key or exhaustive search over key space (brute-force attack)

Towards a rigorous definition of computational security

Concrete approach

- "A scheme is (t,ε)-secure if any attacker A, running for time at most t, succeeds in breaking the scheme with probability at most ε"

Asymptotic approach

- "A scheme is secure if any efficient attacker A succeeds in breaking the scheme with at most negligible probability"

Examples

- almost optimal security guarantees
  - if key length n, the number of possible keys is 2^n
  - attacker running for time t succeeds w/ prob. at most ~ $t/2^n$ (brute-force attack)
- if n = 60, security is enough for attackers running a desktop computer
  - 4 GHz (4×10^9 cycles/sec), checking all 2^{60} keys require about 9 years
  - if n = 80, a supercomputer would still need ~2 years
- today's recommended security parameter is at least n = 128
  - large difference between 2^{80} and 2^{128}; e.g., #seconds since Big Bang is ~2^{58}
  - a once-in-100-years event corresponds to probability 2^{-30} of happening at a particular sec
  - if within 1 year of computation attack is successful w/ prob. $1/2^{60}$
    then it is more likely that Alice and Bob are hit by lighting

Examples: Big Numbers in the real world

- Odds for all 5 numbers + Powerball
  - $292 \times 10^6 \Rightarrow 2^{38}$
- The Age of the Universe in Seconds
  - $4.3 \times 10^{17} \Rightarrow 2^{58}$
- # of cycles in a century of a 4 GHz CPU $\Rightarrow 2^{64}$
- # of arrangements of a Rubik's cube $4.3 \times 10^{19} \Rightarrow 2^{65}$
- Atoms in the Earth $1.33 \times 10^{50} \Rightarrow 2^{166}$
- Electrons in the universe $10^{80} \Rightarrow 2^{266}$

4.3 Introduction to modern cryptography

Cryptography / cryptology

- Etymology
  - two parts: “crypto” + “graphy” / “logy”
  - original meaning: κρυπτός + γράφω / λόγος (**in Greek**)
  - English translation: secret + write / speech, logic
  - meaning: secret writing / the study of secrets
- Historically developed/studied for secrecy in communications
  - i.e., message encryption in the symmetric-key setting
  - main application area: use by military and governments

Classical cryptography Vs. modern cryptography

antiquity – ~70s

- *"the art or writing and solving codes"*
- approach
  - ad-hoc design
  - trial & error methods
  - empirically evaluated

~80s – today

- *"the study of **mathematical techniques** for **securing** digital information, systems, and distributed computations again **adversarial attacks**"*
- approach
  - systematic development & analysis
  - formal notions of security / adversary
  - rigorous proofs of security (or insecurity)

Example: Classical Vs. modern cryptography for encryption

antiquity – ~70s

*"the **art** of **writing** and **solving codes**"*

- **ad-hoc study**
  - vulnerabilities/insecurity of
    - Caesar's cipher
    - shift cipher
    - mono-alphabetic substitution cipher

~80s – today

*"the study of **mathematical techniques** for **securing** information, systems, and distributed computations against **adversarial attacks**"*

- **rigorous study**
  - **problem statement**: secret communication over insecure channel
  - **abstract solution concept**: symmetric encryption, Kerckhoff's principle, perfect secrecy
  - **concrete solution & analysis**: OTP cipher, proof of security

Example: Differences of specific ciphers

Caesar's/shift/mono-alphabetic cipher

- substitution ciphers
  - Caesar's cipher
    - **shift is always 3**
  - shift cipher
    - **shift is unknown but the same for all characters**
  - mono-alphabetic substitution/Vigènere cipher
    - **shift is unknown but the same for all/many character occurrences**

The one-time pad

- also, a substitution cipher
  - **shift is unknown and independent for each character occurrence**

Approach in modern cryptography

Formal treatment

- **fundamental notions** underlying the **design & evaluation** of crypto primitives

Systematic process

- A) **formal definitions** (what it means for a crypto primitive to be “secure”?)
- B) **precise assumptions** (which forms of attacks are allowed – and which aren’t?)
- C) **provable security** (why a candidate instantiation is indeed secure – or not)?

Secure against what?

- “Security” has no meaning per se…
- The security of a system, application, or protocol is always relative to
  - A set of desired properties
  - An adversary with specific capabilities
- Difficult to define general rules for security
  - Adapt best practices, heuristics based on the system we are considering!

Example: Physical safes

TL-15 ($3,000)
15 minutes with common tools

TL-30 ($4,500)
30 minutes with common tools

TRTL-30 ($10,000)
30 minutes with common tools and a cutting torch

TXTL-60 (>$50,000)
60 minutes with common tools, a cutting torch, and up to 4 oz of explosives

The 3 pillars in Cryptography

- We have already been familiar with:
  - A) formal definitions
  - B) precise assumptions
  - C) provable security

- Let’s remind ourselves…

The 3 pillars in Cryptography

- We have already been familiar with:
  - **A) formal definitions**
  - B) precise assumptions
  - C) provable security

- Let’s remind ourselves…

A) Formal definitions

abstract but rigorous description of security problem

- **computing setting** (to be considered)
  - involved parties, communication model, core functionality
- **underlying cryptographic scheme** (to be designed)
  - e.g., symmetric-key encryption scheme
- **desired properties** (to be achieved)
  - security related
  - non-security related
    - e.g., correctness, efficiency, etc.

Why formal definitions are important?

- **successful project management**
  - good design requires clear/specific security goals
    - helps to avoid critical omissions or over engineering
- **provable security**
  - rigorous evaluation requires a security definition
    - helps to separate secure from insecure solutions
- **qualitative analysis/modular design**
  - thorough comparison requires an exact reference
    - helps to secure complex computing systems

Example: Problem at hand

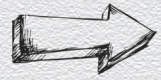
abstract but rigorous description of security problem (to be solved)

secret communication

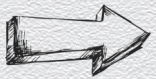
Insecure channel

Example: Formal definitions (1)

- computing setting (to be considered)
  - e.g., involved parties, communication model, core functionality



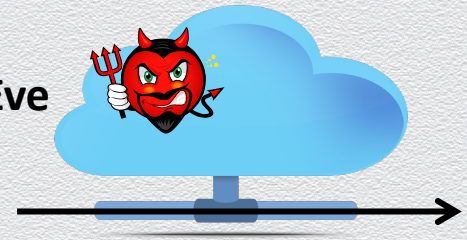
Alice, Bob, Eve



Alice wants to send a message m to Bob; Eve can eavesdrop sent messages

Alice/Bob may transform the transmitted/received message and share info

Alice m



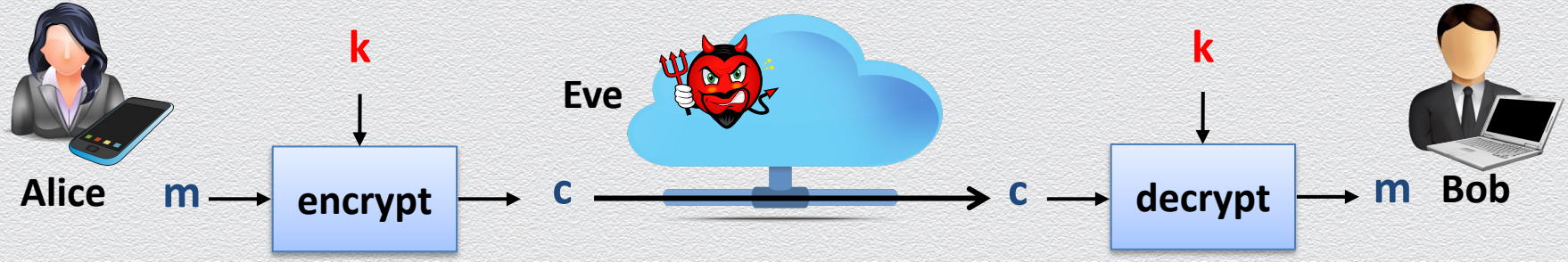
Bob

Example: Formal definitions (2)

- **underlying cryptographic scheme** (to be designed)

⇒ symmetric-key encryption scheme

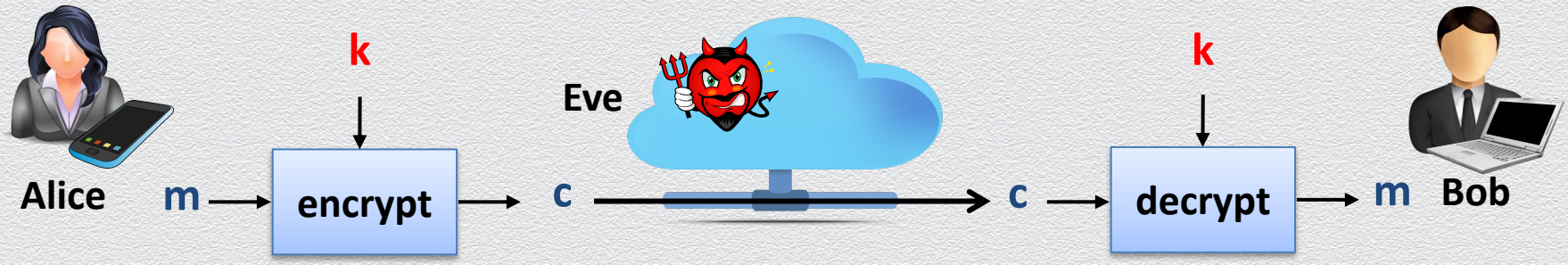
- Alice and Bob share and use a key k
- Alice encrypts plaintext m to ciphertext c and sends c instead of m
- Bob decrypts received c to get a message m'



Example: Formal definitions (3)

- **desired properties** (to be achieved)
  - **security (informal)** ⇨ Eve "cannot learn" m (from c)
  - **correctness (informal)**

⇨ If Alice encrypts m to c, then Bobs decrypts c to (the original message) m



Example: Probabilistic view of symmetric encryption

A symmetric-key encryption scheme is defined by

- a **message space** $\mathcal{M}$, $|\mathcal{M}| > 1$, and a triple (**Gen**, **Enc**, **Dec**)
- **Gen**: probabilistic key-generation algorithm, defines **key space** $\mathcal{K}$
  - $\text{Gen}(1^n) \rightarrow k \in \mathcal{K}$ (security parameter n)
- **Enc**: probabilistic encryption algorithm, defines **ciphertext space** $\mathcal{C}$
  - Enc: $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$, $\text{Enc}(k, m) = \text{Enc}_k(m) \rightarrow c \in \mathcal{C}$
- **Dec**: deterministic encryption algorithm
  - Dec: $\mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$, $\text{Dec}(k, c) = \text{Dec}_k(c) := m \in \mathcal{M}$ or $\perp$

Example: Formal definitions (4)

Perfect correctness

- for any k ∈ $\mathcal{K}$, m ∈ $\mathcal{M}$ and any ciphertext c output of $Enc_k(m)$, it holds that

$$\mathbf{Pr[\ Dec_k\ (c) = m\ ] = 1}$$

Perfect security (or information-theoretic security)

- the adversary should be able to learn **no additional** information on m

random experiment

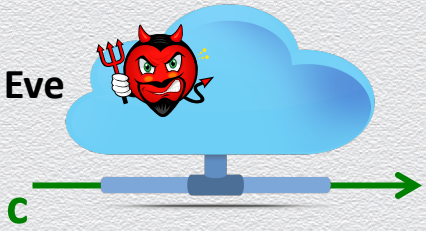
$\mathcal{D}_{\mathcal{M}} \rightarrow$ **m**

$\mathcal{D}_{\mathcal{K}} \rightarrow$ k

$Enc_k(m) \rightarrow$ c

m = { attack w/ prob. 0.8 ; no attack w/ prob. 0.2 }

Eve's view remains the same!



m = { attack w/ prob. 0.8 ; no attack w/ prob. 0.2 }

Example: Equivalent definitions of perfect security

1) a posteriori = a priori

For every $\mathcal{D}_{\mathcal{M}}$, m ∈ $\mathcal{M}$ and c ∈ $\mathcal{C}$, for which Pr [C = c] > 0, it holds that

Pr[M = m | C = c] = Pr[M = m]

2) C is independent of M

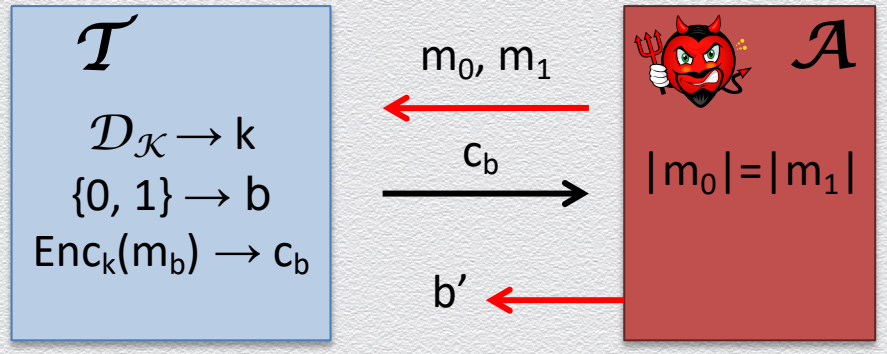
For every m, m' ∈ $\mathcal{M}$ and c ∈ $\mathcal{C}$, it holds that

Pr[Enc_K(m) = c] = Pr[Enc_K(m') = c]

3) indistinguishability

For every $\mathcal{A}$, it holds that

Pr[b' = b] = 1/2



From perfect to computational EAV-security

- **perfect** security: M, $Enc_K(M)$ are independent
  - absolutely **no information is leaked** about the plaintext
  - to adversaries that **unlimited computational power**
- **computational** security: for all **practical** purposes, M, $Enc_K(M)$ are independent
  - **a tiny amount of information is leaked** about the plaintext (e.g., w/ prob. 2^{-60})
  - to adversaries with **bounded computational power** (e.g., attacker invests 200ys)
- attacker's **best strategy** remains **ineffective**
  - **random guess** on secret key; or
  - **exhaustive search** over key space (**brute force attack**)

Relaxing indistinguishability

Relax the definition of perfect secrecy – that is based on indistinguishability

- require that $\mathbf{m_0, m_1}$ are chosen by a **PPT adversary**
- require that no **PPT adversary** can distinguish $\mathbf{Enc_k(m_0)}$ from $\mathbf{Enc_k(m_1)}$

non-negligibly better than guessing

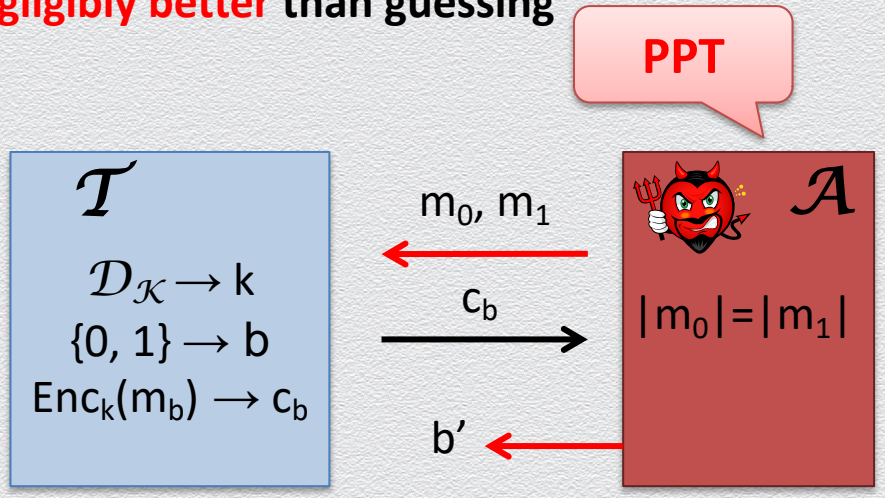
3) indistinguishability

For every $\mathcal{A}$, it holds that

PPT

$$\Pr[\, b' = b \,] = 1/2 + \text{negl}$$

negl



The 3 pillars in Cryptography

- We have already been familiar with:
  - A) formal definitions
  - **B) precise assumptions**
  - C) provable security

- Let’s remind ourselves…

B) Why precise assumptions are important?

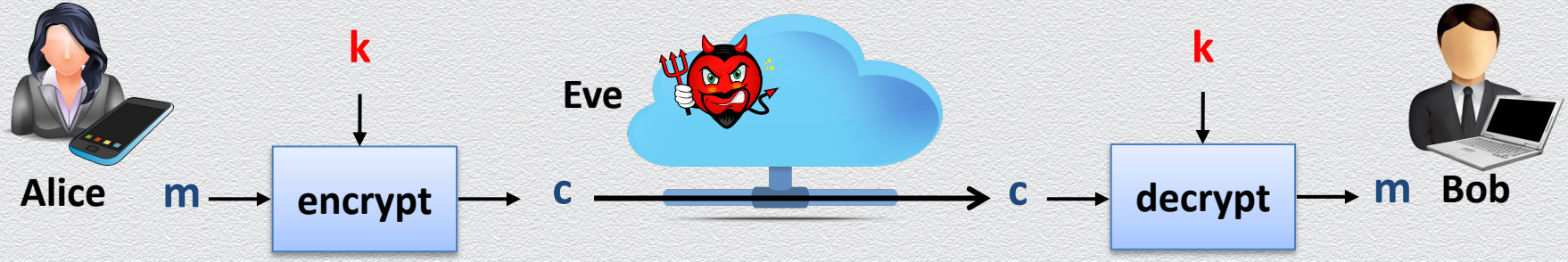
- **basis** for proofs of security
  - security holds under specific assumptions
- **comparison** among possible solutions
  - relations among different assumptions
    - stronger/weaker (i.e., less/more plausible to hold), “A implies B” or “A and B are equivalent”
    - refutable Vs. non-refutable
- **flexibility** (in design & analysis)
  - **validation** – to gain confidence or refute
  - **modularity** – to choose among concrete schemes that satisfy the same assumptions
  - **characterization** – to identify simplest/minimal/necessary assumptions

Example: Precise assumptions (1)

- **adversary**
  - type of attacks – a.k.a. **threat model** → eavesdropping
  - **capabilities** (e.g., a priori knowledge, access to information, party corruptions)
  - **limitations** (e.g., bounded memory, passive Vs. active)

→ Eve may know the a priori distribution of messages sent by Alice

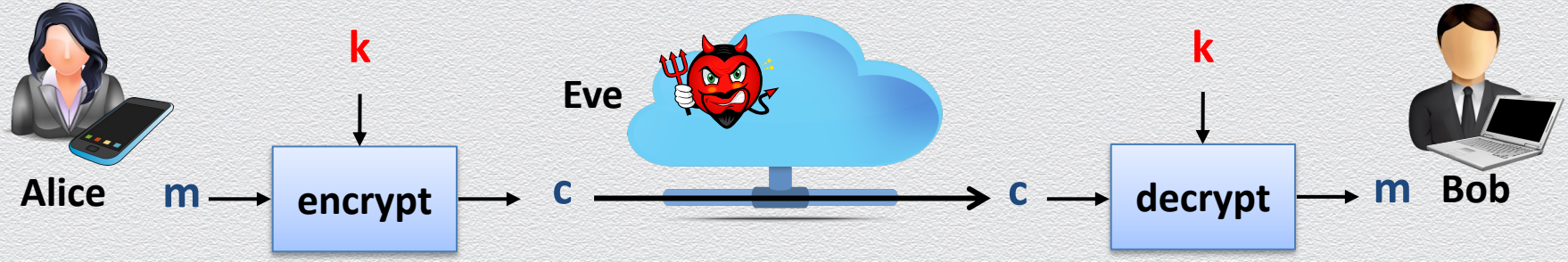
→ Eve doesn't know/learn the secret k (shared by Alice and Bob)



Example: Precise assumptions (2)

- **computational assumptions** (about hardness of certain tasks)
  - e.g., factoring of large composite numbers is hard

no computational assumptions
– a.k.a. perfect secrecy (or information-theoretic security)



Example: Precise assumptions (3)

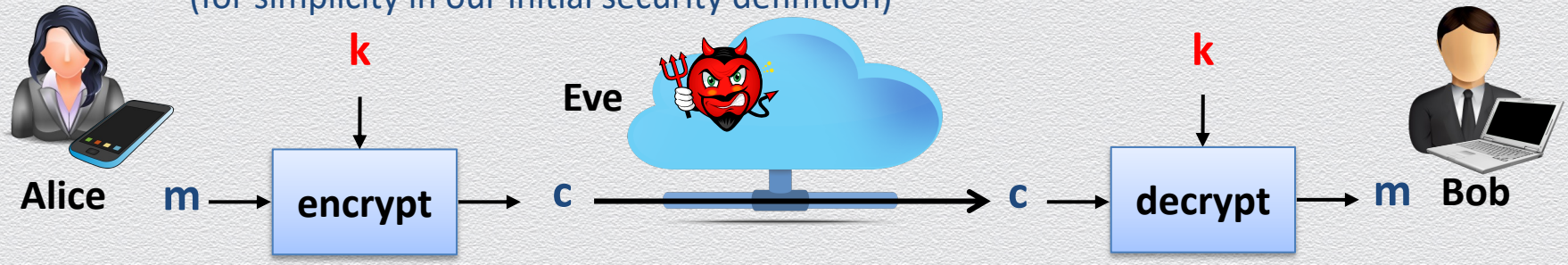
- **computing setting**
  - **system set up**, initial state, **key distribution**, **randomness**…
  - means of **communication** (e.g., channels, rounds, messages…)
  - timing assumptions (e.g., synchronicity, epochs, …)

⇒ key k is generated randomly using the uniform distribution

⇒ key k is securely distributed to and securely stored at Alice and Bob

⇒ one message m is only communicated (for simplicity in our initial security definition)

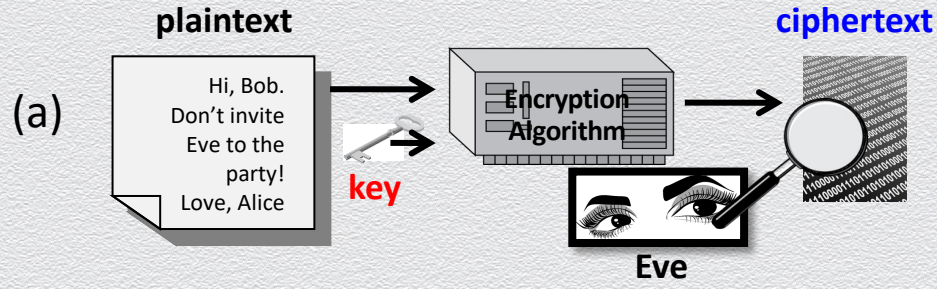
⇒ k, m are chosen independently



Possible eavesdropping attacks (I)

An attacker may possess a

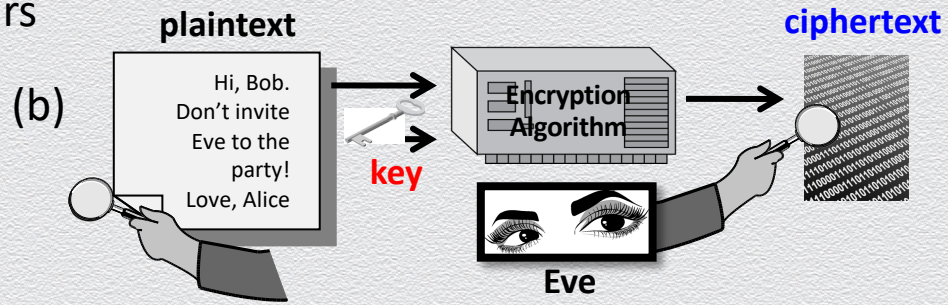
- (a) collection of ciphertexts
  - ciphertext only attack
  - this will be the **default attack type** when we will next define the concept of perfect security



Possible eavesdropping attacks (II)

An attacker may possess a

- (a) collection of ciphertexts
  - ciphertext only attack
- (b) collection of plaintext/ciphertext pairs
  - known plaintext attack

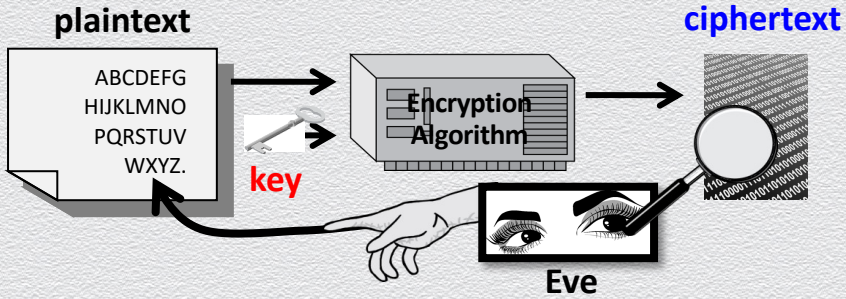


Possible eavesdropping attacks (III)

An attacker may possess a

- (a) collection of ciphertexts
  - ciphertext only attack
- (b) collection of plaintext/ciphertext pairs
  - known plaintext attack
- (c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker
  - chosen plaintext attack

(c)

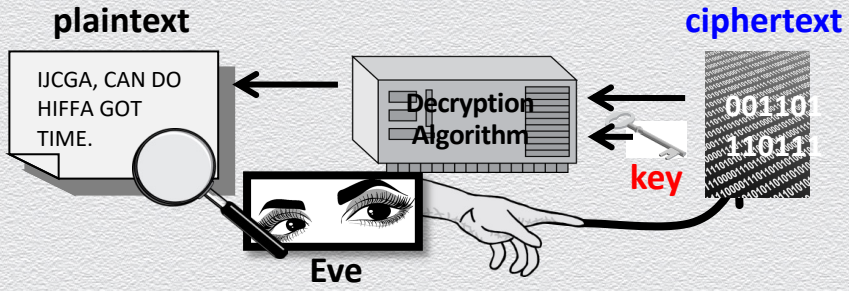


Possible eavesdropping attacks (IV)

An attacker may possess a

- (a) collection of ciphertexts
  - ciphertext only attack
- (b) collection of plaintext/ciphertext pairs
  - known plaintext attack
- (c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker
  - chosen plaintext attack
- (d) collection of plaintext/ciphertext pairs for (plaintexts and) ciphertexts selected by the attacker
  - chosen ciphertext attack

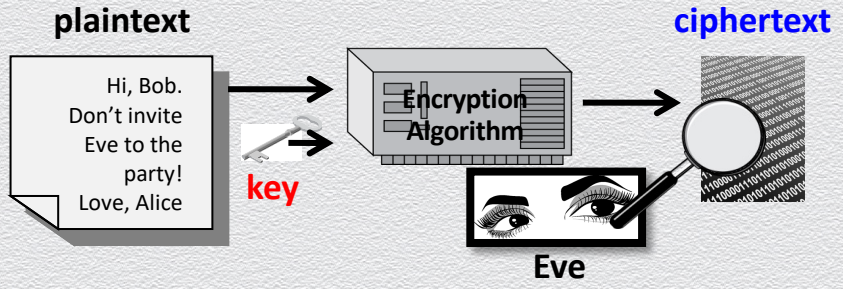
(d)



Main security properties against eavesdropping

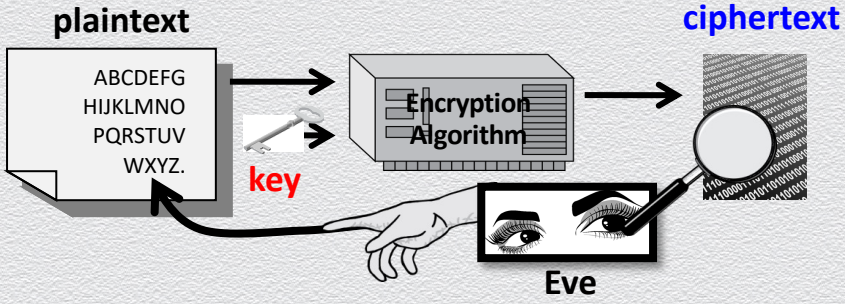
“plain” security

- protects against ciphertext-only attacks
  - EAV-attack



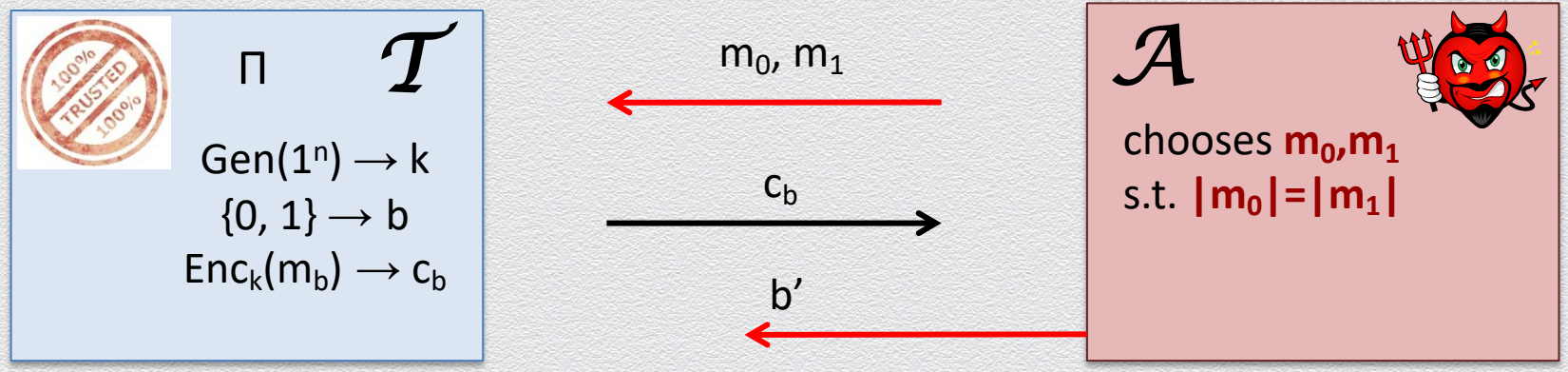
“advanced” security

- protects against chosen plaintext attacks
  - CPA-attack



Game-based computational EAV-security

encryption scheme Π = {$\mathcal{M}$, (Gen, Enc, Dec)}

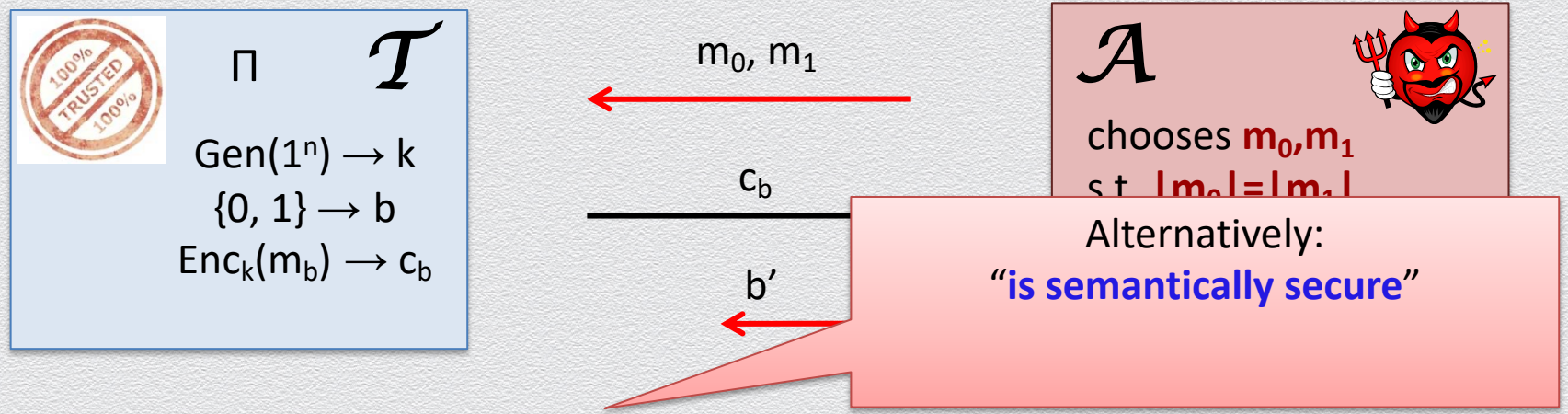


We say that (Enc,Dec) is **EAV-secure** if any PPT adversary $\mathcal{A}$ guesses b correctly with probability at most 0.5 + ε(n), where ε is a negligible function

I.e., no PPT $\mathcal{A}$ computes b correctly non-negligibly better than randomly guessing

Game-based computational EAV-security

encryption scheme Π = {$\mathcal{M}$, (Gen, Enc, Dec)}

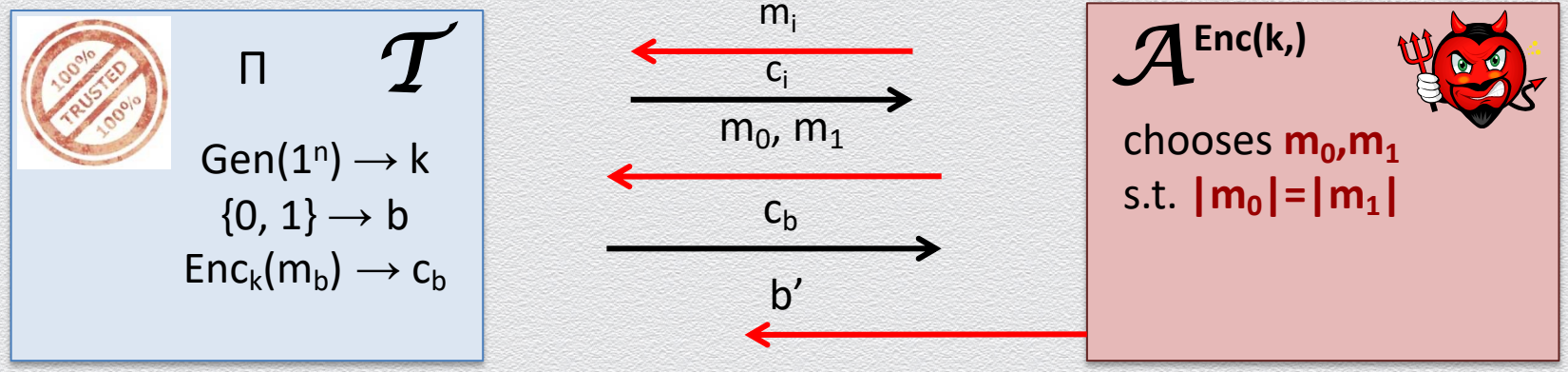


We say that (Enc,Dec) is **EAV-secure** if any PPT adversary $\mathcal{A}$ guesses b correctly with probability at most 0.5 + ε(n), where ε is a negligible function

I.e., no PPT $\mathcal{A}$ computes b correctly non-negligibly better than randomly guessing

Game-based computational CPA-security

encryption scheme Π = {$\mathcal{M}$, (Gen, Enc, Dec)}



We say that (Enc,Dec) is **CPA-secure** if any PPT adversary $\mathcal{A}$ guesses b correctly with probability at most 0.5 + ε(n), where ε is a negligible function

I.e., no PPT $\mathcal{A}$ computes b correctly non-negligibly better than randomly guessing, **even when it learns the encryptions of messages of its choice**

On CPA security

Facts

- Any encryption scheme that is CPA-secure is also CPA-secure for multiple encryptions
- **CPA security implies randomized encryption – can you see why?**
- EAV-security for multiple messages implies randomized encryption

The 3 pillars in Cryptography

- We have already been familiar with:
  - A) formal definitions
  - B) precise assumptions
  - **C) provable security**

- Let’s remind ourselves…

C) Provably security

Security

- subject to certain **assumptions**, a scheme is proved to be **secure** according to a specific **definition**, against a specific **adversary**
  - in practice the scheme may break if
    - some assumptions do not hold or the attacker is more powerful

Insecurity

- a scheme is proved to be **insecure** with respect to a specific **definition**
  - it suffices to find a **counterexample attack**

Why provable security is important?

Typical performance

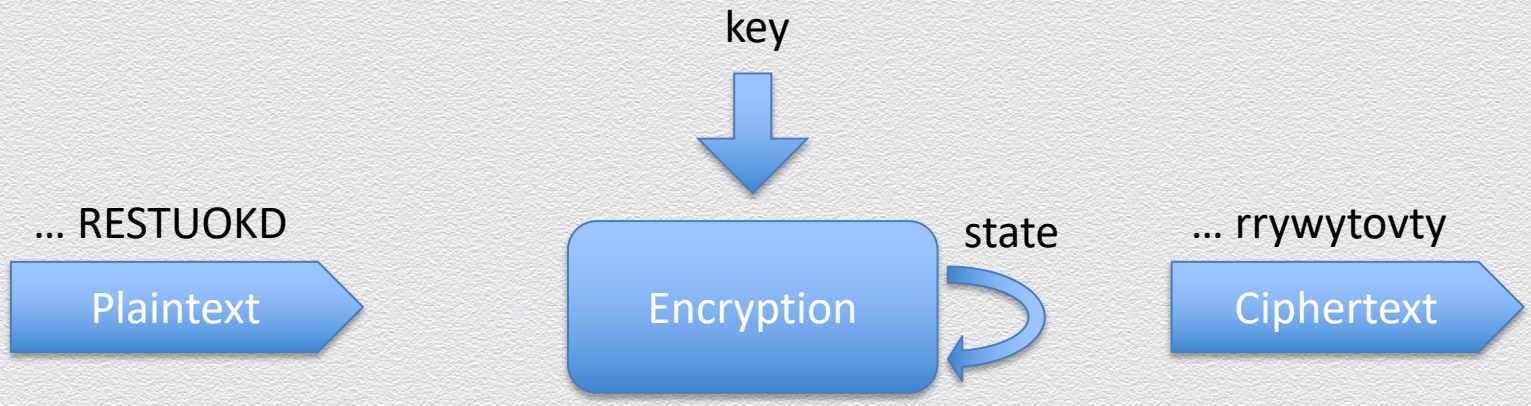
- in some areas of computer science
  formal proofs may not be essential
  - simulate hard-to-analyze algorithm to experimentally study its performance on "typical" inputs
- in practice, **typical/average case** occurs

Worst case performance

- in cryptography and secure protocol design
  formal proofs are essential
  - "experimental" security analysis is not possible
  - the notion of a "typical" adversary makes little sense and is unrealistic
  - in practice, **worst case attacks will occur**
    - an adversary will use any means in its power to break a scheme

4.4 Pseudorandom generators (or stream ciphers)

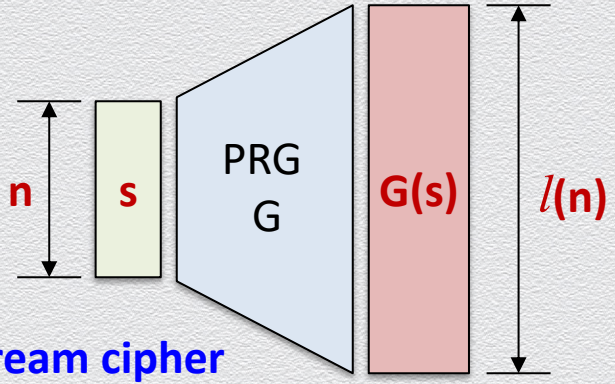
Stream ciphers



Pseudorandom generators (PRGs)

Deterministic algorithm G that

on input a **seed** $s \in \{0,1\}^t$, outputs $G(s) \in \{0,1\}^{l(t)}$



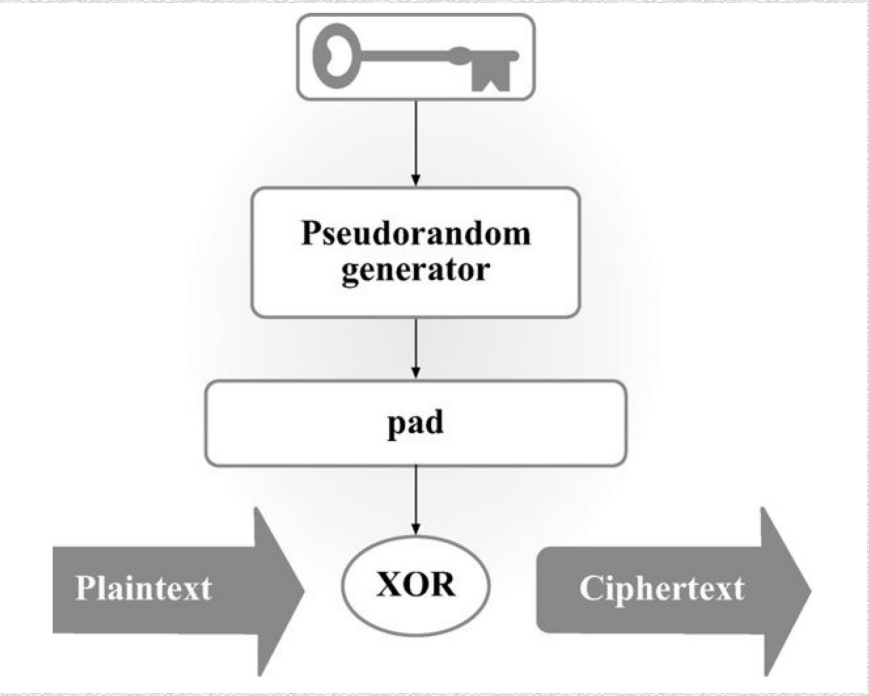
a.k.a. **stream cipher**

G is a PRG if:

- **expansion**
  - for polynomial *l*, it holds that for any n, *l*(n) > n
  - models the process of **extracting** randomness from a short random string
- **pseudorandomness**
  - **no efficient statistical test can tell apart** G(s) from a **truly random** string

Generic PRG-based symmetric encryption

- **Fixed-length** message encryption



encryption scheme is plain-secure
as long as the underlying PRG is secure

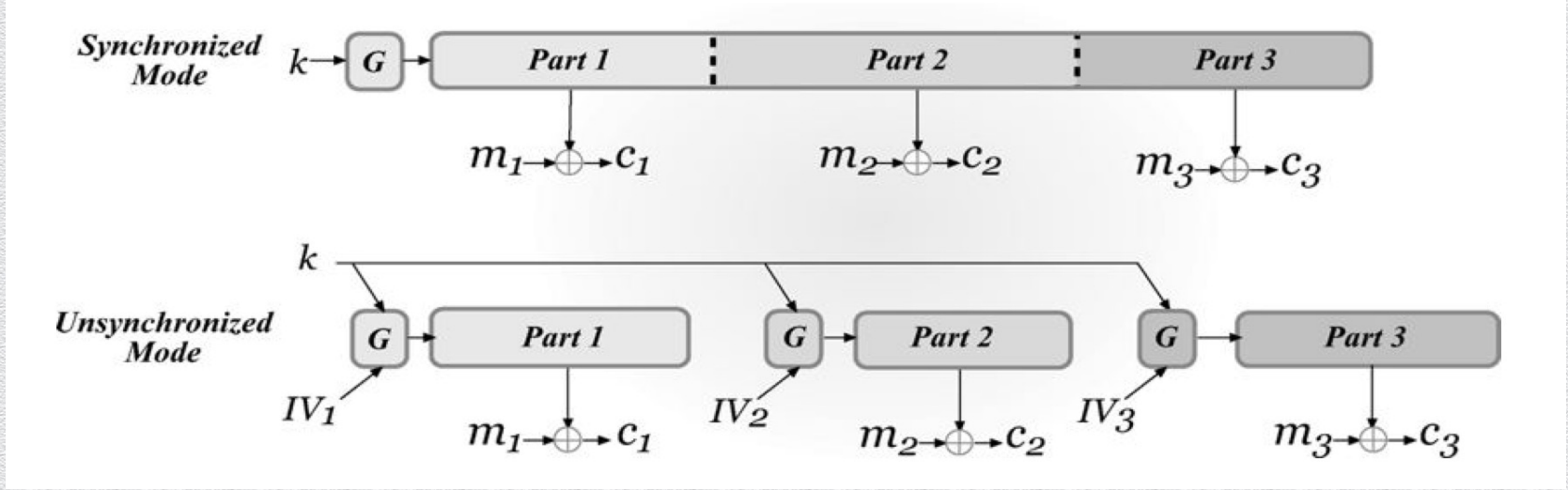
Generic PRG-based symmetric encryption (cont.)

- **Bounded- or arbitrary-length** message encryption
  - specified by a mode of operation for using an underlying stateful stream cipher, repeatedly, to encrypt/decrypt a stream of symbols

Stream ciphers: Modes of operations

- **Bounded- or arbitrary-length** message encryption

on-the-fly computation of new pseudorandom bits, no IV needed, plain-secure



random IV used for every new message is sent along with ciphertext, advanced-secure