https://brown-csci1660.github.io

CS1660: Intro to Computer Systems Security
Spring 2026

Lecture 3: Confidentiality

Instructor: **Nikos Triandopoulos**

January 29, 2025

BROWN

CS1660: Announcements

- Course updates
  - Please make sure you complete Homework 0 and Project 0
  - Please make sure you have access to Ed Discussion and Gradescope
  - Project 1 “Cryptography” is going out today; due in 3 weeks

Last class

- Introduction to Computer Security
  - Motivation
  - Basic security concepts
- Cryptography
  - Secret communication
    - Symmetric-key encryption & classical ciphers
    - Perfect secrecy & the One-Time Pad

⇨ **Completed**

⇨ **Current**

⇨ **Upcoming**

Today

- Cryptography
  - Secret communication → **Confidentiality**
    - Symmetric-key encryption & classical ciphers
    - Perfect secrecy & the One-Time Pad
  - Encryption in practice
    - Computational security, pseudo-randomness
    - Stream & block ciphers, modes of operations for encryption, DES & AES
    - Introduction to modern cryptography → **Intro to Crypto**

3.0 Symmetric-key encryption

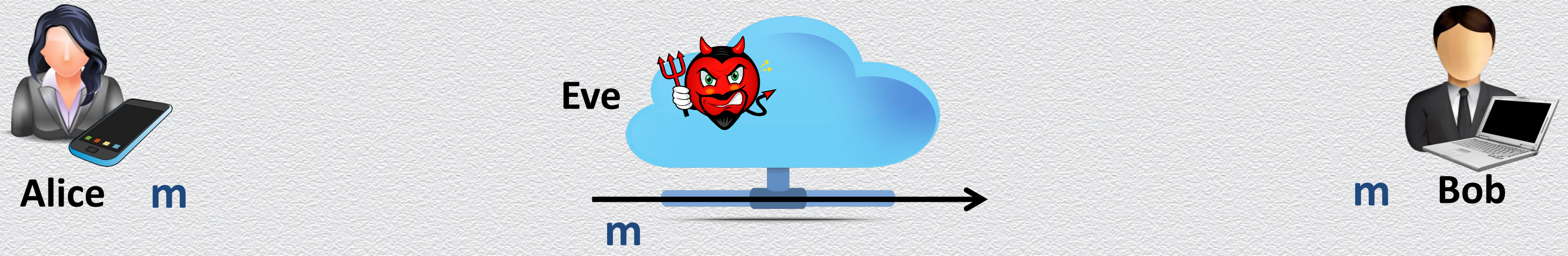
Problem setting: Secret communication

Two parties wish to communicate over a channel

- Alice (sender/source) wants to send a message m to Bob (recipient/destination)

Underlying channel is unprotected

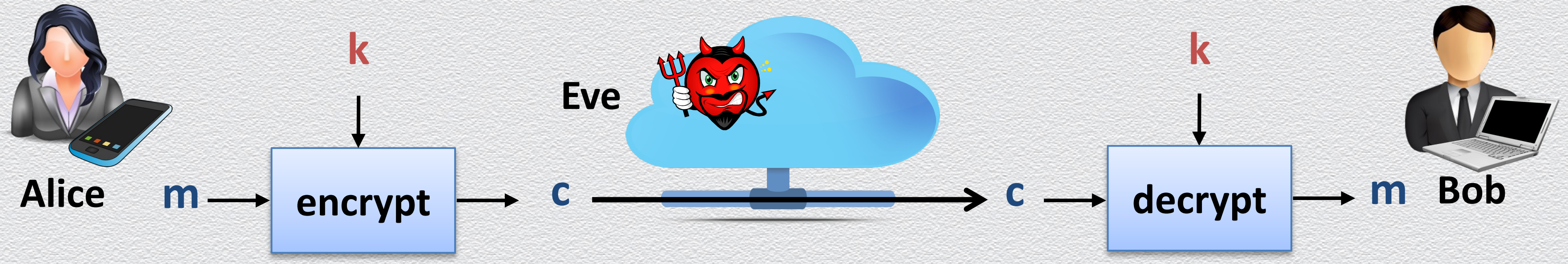
- Eve (attacker/adversary) can eavesdrop any sent messages
- e.g., packet sniffing over networked or wireless communications



Solution concept: Symmetric-key encryption

Main idea

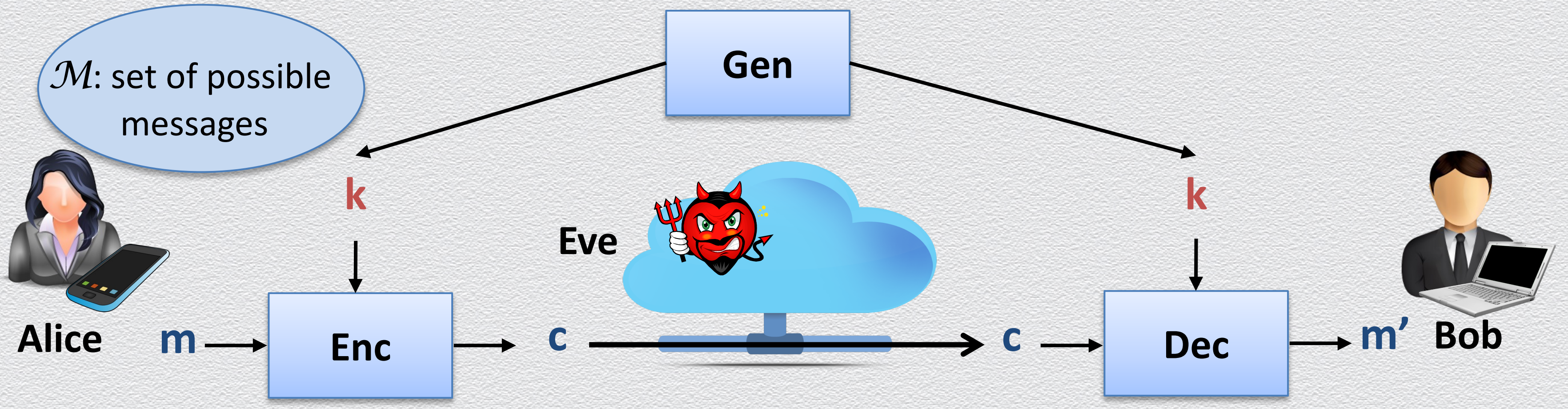
- secretly transform message so that it is **unintelligible** while in transit
  - Alice **encrypts** her message m to **ciphertext c**, which is sent instead of **plaintext m**
  - Bob **decrypts** received message c to original message m
  - Eve can intercept c but "**cannot learn**" m from c
  - Alice and Bob share a **secret key k** that is used for both message transformations



Security tool: Symmetric-key encryption scheme

Abstract cryptographic primitive, **a.k.a. cipher**, defined by

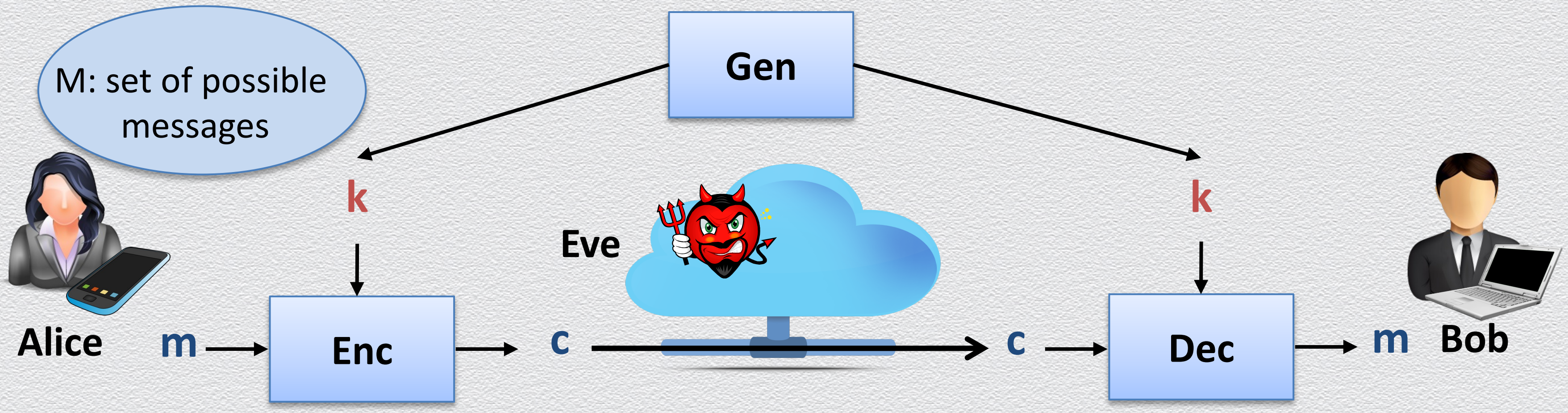
- a **message space** $\mathcal{M}$; and
- a triplet of algorithms **(Gen, Enc, Dec)**
  - Gen is randomized algorithm, Enc may be raldomized, whereas Dec is deterministic
  - Gen outputs a uniformly random key k (from some key space $\mathcal{K}$)



Desired properties for symmetric-key encryption scheme

By design, any symmetric-key encryption scheme should satisfy the following

- **efficiency**: key generation & message transformations "are fast"
- **correctness**: for all m and k, it holds that Dec(Enc(m, k) , k) = m
- **security**: one "cannot learn" plaintext m from ciphertext c



(Auguste) Kerckhoff's principle (1883)

"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."

Reasoning

- due to security & correctness, Alice & Bob must share some secret info
- if no shared key captures this secret info, it must be captured by Enc, Dec
- but keeping Enc, Dec secret is problematic
    - harder to keep secret an algorithm than a short key (e.g., after user revocation)
    - harder to change an algorithm than a short key (e.g., after secret info is exposed)
    - riskier to rely on custom/ad-hoc schemes than publicly scrutinized/standardized ones

(Auguste) Kerckhoff's principle (1883)

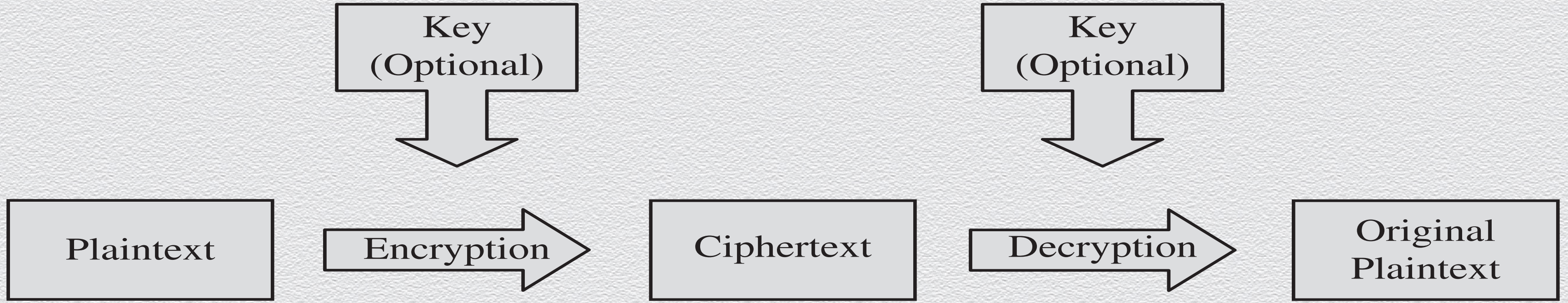
"The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience."

General good-hygiene principle (beyond encryption)

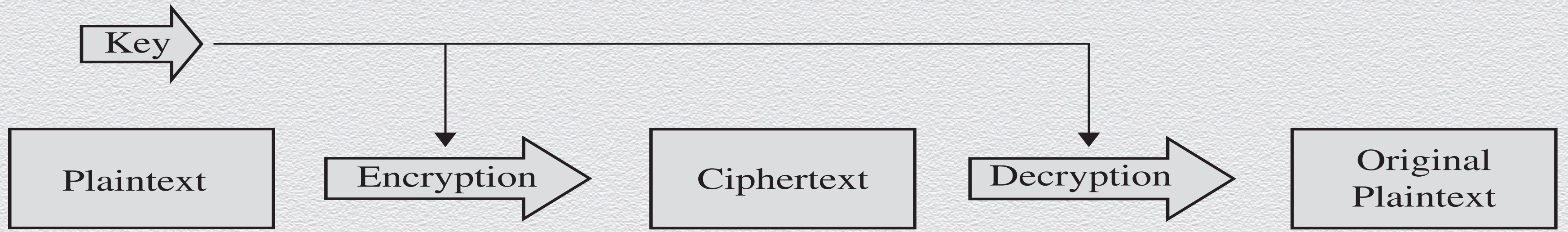
- Security relies solely on keeping secret keys
- System architecture and algorithms are publicly available
- Claude Shannon (1949): *"one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them"*
- Opposite of "security by obscurity" practice

Symmetric-key encryption

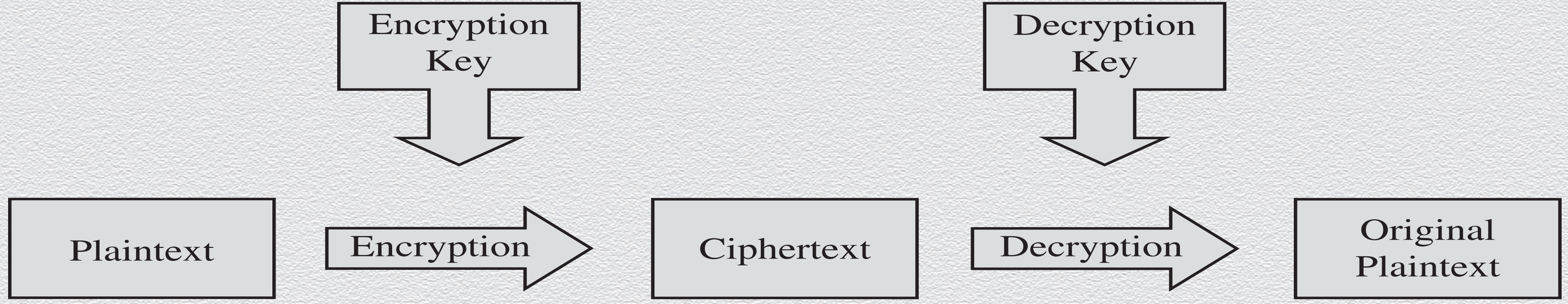
- Also referred to as simply "symmetric encryption"



Symmetric Vs. Asymmetric encryption



(a) Symmetric Cryptosystem



(b) Asymmetric Cryptosystem

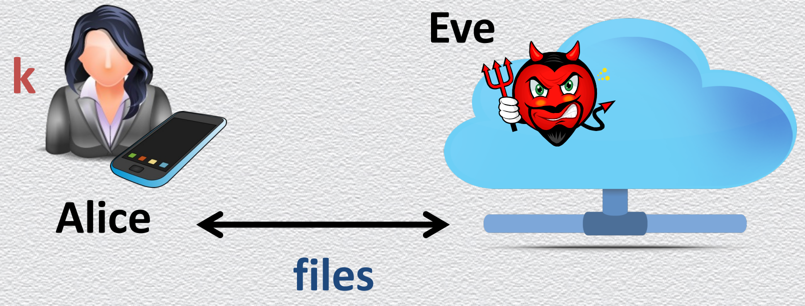
Main application areas

Secure communication

- **encrypt messages** sent among parties
- assumption
  - Alice and Bob **securely generate, distribute & store shared key k**
  - attacker does not learn key k

Secure storage

- **encrypt files** outsourced to the cloud
- assumption
  - Alice **securely generates & stores key k**
  - attacker does not learn key k



Brute-force attack

Generic attack

- given a captured ciphertext c and known key space $\mathcal{K}$, Dec
- strategy is an **exhaustive search**
  - for all possible keys k in $\mathcal{K}$
    - determine if Dec (c,k) is a likely plaintext m
- **requires some knowledge on the message space $\mathcal{M}$**
  - i.e., structure of the plaintext (e.g., PDF file or email message)

Countermeasure

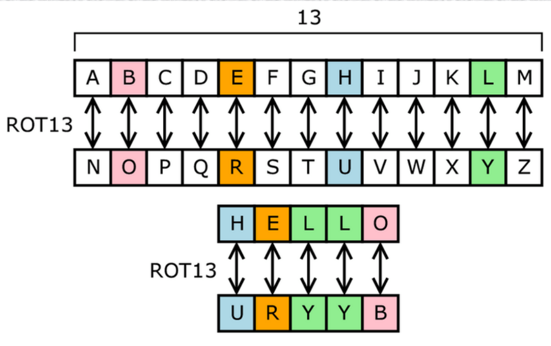
- key should be a **random** value from a **sufficiently large** key space $\mathcal{K}$ to make exhaustive search attacks **infeasible**

3.1 Classical ciphers

Substitution ciphers

Large class of ciphers: each letter is **uniquely** replaced by another

- key is a (random) permutation over the alphabet characters
- there are 26! ≈ 4×10^{26} possible substitution ciphers
- huge key space (larger than the # of starts in universe)
- e.g., one popular substitution "cipher" for some Internet posts is ROT13



- historically
  - all classical ciphers are of this type

Classical ciphers – general structure

Class of ciphers based on letter substitution

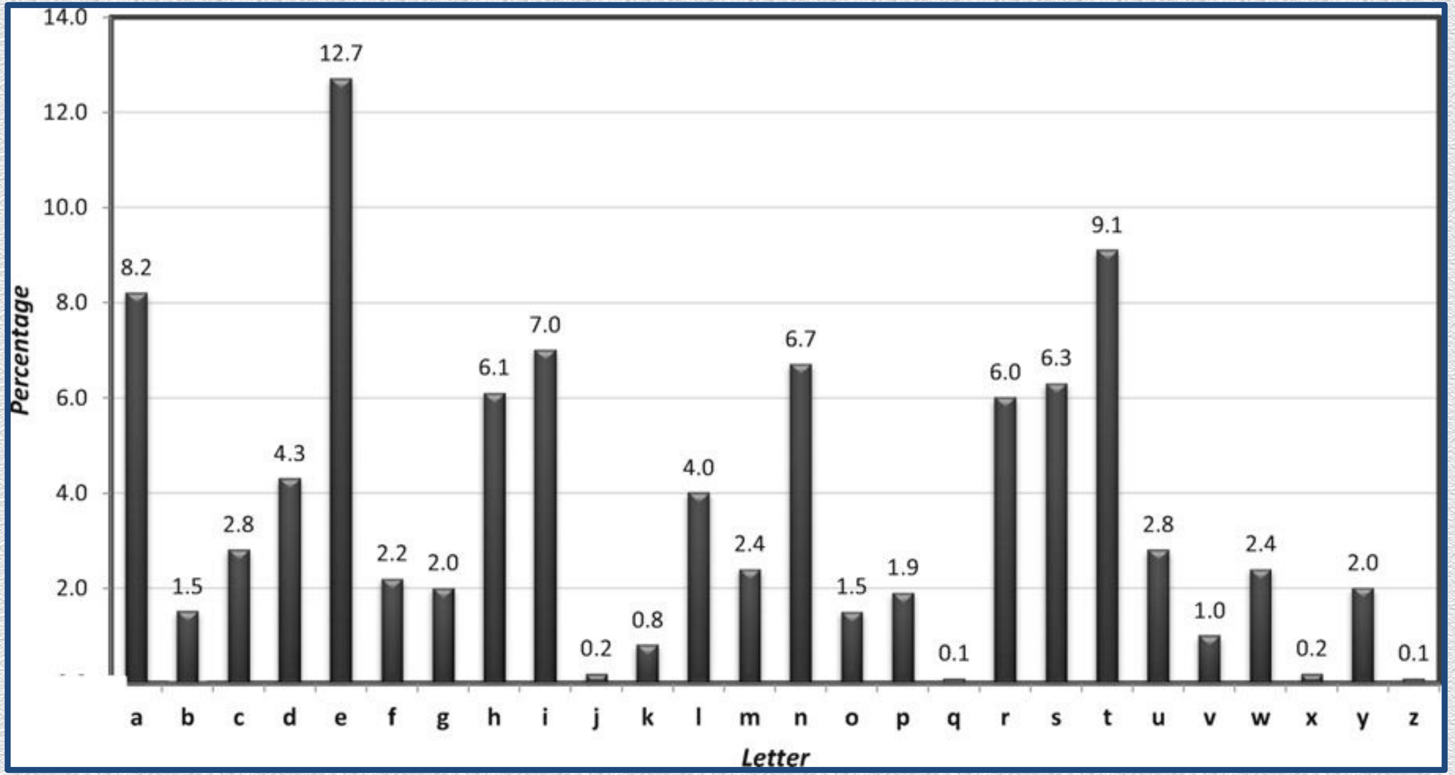
- message space $\mathcal{M}$ is "**valid words**" from a given alphabet
  - e.g., English text without spaces, punctuation or numerals
  - characters can be represented as numbers in [0:25]
- based on a predetermined **1-1** character mapping
  - map each (plaintext) character into another **unique** (ciphertext) character
  - typically defined as a "**shift**" of each plaintext character by a **fixed** per alphabet character number of positions in a canonical ordering of the characters in the alphabet
- encryption: character shifting occurs with "**wrap-around**" (using mod 26 addition)
- decryption: **undo shifting** of characters with "wrap-around" (using mod 26 subtraction)

Limitations of substitution ciphers

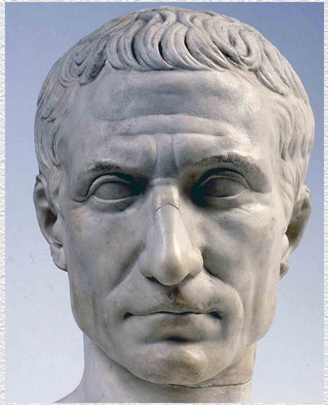
Generally, susceptible to **frequency (and other statistical) analysis**

- letters in a natural language, like English, are not uniformly distributed
- cryptographic attacks against substitution ciphers are possible
  - e.g., by exploiting knowledge of letter frequencies, including pairs and triples
    - most frequent letters in English: e, t, o, a, n, i, ...
    - most frequent digrams: th, in, er, re, an, ...
    - most frequent trigrams: the, ing, and, ion, ...
  - Attack framework first described in a 9th century book by al-Kindi

Letter frequency in (sufficiently large) English text



Classical ciphers – examples



(Julius) Caesar's cipher

- shift each character in the message by 3 positions
  - I.e., 3 instead of 13 positions as in ROT-13
- cryptanalysis
  - **no secret key is used** – based on “security by obscurity”
  - thus the code is trivially insecure once knows Enc (or Dec)

Classical ciphers – examples (II)

Shift cipher

- **keyed extension** of Caesar's cipher
- randomly set key k in [0:25]
  - shift each character in the message by k positions
- cryptanalysis
  - **brute-force attacks** are effective given that
    - **key space is small** (26 possibilities or, actually, 25 as 0 should be avoided)
    - message space M is **restricted to "valid words"**
      - e.g., corresponding to valid English text

Alternative attack against "shift cipher"

- brute-force attack + inspection if English "make sense" is quite **manual**
- a better **automated** attack is based on statistics
  - if character i (in [0:25]) in the alphabet has frequency p_i (in [0..1]), then
    - from known statistics, we know that $\mathbf{\Sigma_i\, p_i^2 \approx 0.065}$, so
    - since character i (in plaintext) is mapped to character i + k (in ciphertext)
      - if $\mathbf{L_j = \Sigma_i\, p_i\, q_{i+j}}$, then we expect that $\mathbf{L_k \approx 0.065}$ (q_i: frequency of character i in ciphertext)
- thus, a brute-force attack can **test** all possible keys w.r.t. the **above criterion**
  - the search space **remains the same**
  - yet, the condition to finish the search **becomes much simpler**: Choose j so that $\mathbf{L_j \approx 0.065}$

Classical ciphers – examples (III)

Mono-alphabetic substitution cipher

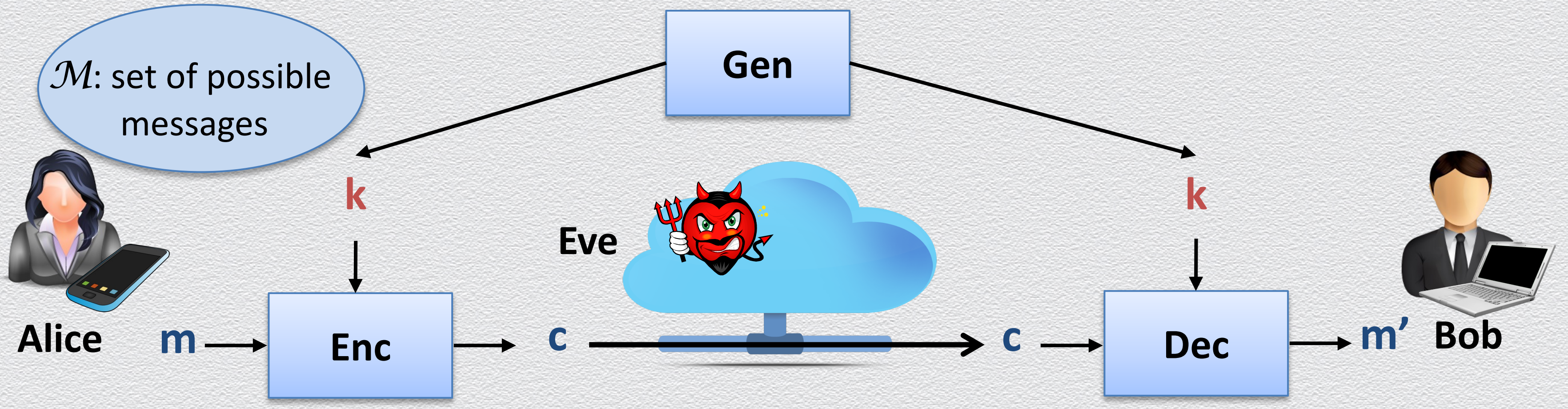
- **generalization** of shift cipher
- key space defines **permutation** on alphabet
  - use a **1-1 mapping between characters** in the alphabet to produce ciphertext
  - i.e., shift each **distinct** character in the plaintext (by some appropriate number of positions defined by the key) to get a **distinct** character in the ciphertext
- cryptanalysis
  - key space is large (of the order of 26! or ~2^{88}) but cipher is vulnerable to attacks
  - character mapping is **fixed** by key so **plaintext & ciphertext exhibit same statistics**

3.2 Perfect secrecy

Security tool: Symmetric-key encryption scheme

Abstract cryptographic primitive, **a.k.a. cipher**, defined by

- a **message space** $\mathcal{M}$; and
- a triplet of algorithms **(Gen, Enc, Dec)**
  - Gen is randomized algorithm, Enc may be raldomized, whereas Dec is deterministic
  - Gen outputs a uniformly random key k (from some key space $\mathcal{K}$)



Probabilistic formulation

Desired properties

- Efficiency
- Correctness
- Security

Our setting so far is a random experiment

- a message m is chosen according to $\mathcal{D}_{\mathcal{M}}$
- a key k is chosen according to $\mathcal{D}_{\mathcal{K}}$
- $\mathrm{Enc}_k(m) \rightarrow c$ is given to the adversary

Perfect correctness

For any k ∈ $\mathcal{K}$, m ∈ $\mathcal{M}$ and any ciphertext c output of $Enc_k(m)$,

it holds that

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

Perfect security

Defining security for an encryption scheme is not trivial

- what we mean by "Eve "cannot learn" m (from c)" ?

Attempt 1: Protect the key k!

- Security means that

  the adversary should **not** be able to **compute the key k**

- Intuition
  - it'd better be the case that the key is protected!...

  necessary condition

- Problem
  - this definition fails to exclude clearly insecure schemes
  - e.g., the key is never used, such as when $Enc_k(m) := m$

  but not sufficient condition!

Attempt 2: Don't learn m!

- Security means that

  the adversary should **not** be able to **compute the message m**

- Intuition
  - it'd better be the case that the message m is not learned...
- Problem
  - this definition fails to exclude clearly undesirable schemes
  - e.g., those that protect m partially, i.e., they reveal the least significant bit of m

Attempt 3: Learn nothing!

- Security means that

  the adversary should **not** be able to **learn any information about m**

- Intuition
  - it seems close to what we should aim for perfect secrecy…
- Problem
  - this definition ignores the adversary's prior knowledge on $\mathcal{M}$
  - e.g., distribution $\mathcal{D}_{\mathcal{M}}$ may be known or estimated
    - m is a valid text message, or one of "`attack`", "`no attack`" is to be sent

Attempt 4: Learn nothing more!

- Security means that

  the adversary should **not** be able to **learn any <u>additional</u> information on m**

- How can we formalize this?

Alice **m**

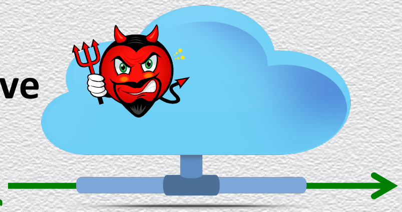
$\mathbf{Enc_k(m) \rightarrow c}$

Eve

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

Eve's view remains the same!

Eve



c

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

Two equivalent views of perfect secrecy

a posteriori = a priori ~ **C is independent of M**

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

$$\mathbf{Pr[\ M = m\ |\ C = c\ ] = Pr[\ M = m\ ]}$$

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

$$\mathbf{Pr[\ Enc_K(m) = c\ ] = Pr[\ Enc_K(m') = c\ ]}$$

random experiment
$\mathcal{D}_{\mathcal{M}} \rightarrow$ **m = M**
$\mathcal{D}_{\mathcal{K}} \rightarrow$ k = K
$\mathrm{Enc_k(m)} \rightarrow$ c = C

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

Eve's view remains the same!

c

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

Perfect secrecy (or information-theoretic security)

Definition 1

A symmetric-key encryption scheme (Gen, Enc, Dec) with message space $\mathcal{M}$,
is **perfectly secret** if for every $\mathcal{D}_{\mathcal{M}}$, every message m ∈ $\mathcal{M}$ and every ciphertext c ∈ C
for which Pr [C = c] > 0, it holds that

$$\Pr[\, M = m \mid C = c \,] = \Pr[\, M = m \,]$$

- Intuitively
  - the *a posteriori* probability that any given message m was actually sent is the **same** as the *a priori* probability that m would have been sent
  - observing the ciphertext reveals **nothing (new)** about the underlying plaintext

Alternative view of perfect secrecy

Definition 2

A symmetric-key encryption scheme (Gen, Enc, Dec) with message space $\mathcal{M}$, is **perfectly secret** if for every messages m, m' ∈ $\mathcal{M}$ and every c ∈ $\mathcal{C}$, it holds that

$$\Pr[\,\mathrm{Enc}_K(m) = c\,] = \Pr[\,\mathrm{Enc}_K(m') = c\,]$$

- Intuitively
  - the probability distribution $\mathcal{D}_C$ **does not depend** on the plaintext
  - i.e., M and C are **independent** random variables
  - the ciphertext contains "**no information**" about the plaintext
  - "**impossible to distinguish**" an encryption of **m** from an encryption of **m'**

3.3 The One-Time Pad

The one-time pad: A perfect cipher

A type of "<u>substitution</u>" cipher that is "<u>absolutely unbreakable</u>"

- invented in 1917 Gilbert Vernam and Joseph Mauborgne
- "substitution" cipher
  - **individually** replace plaintext characters with **shifted** ciphertext characters
  - **independently** shift each message character in a **random** manner
    - to encrypt a plaintext of length n, use n uniformly random keys $k_1, \ldots, k_n$
- "absolutely unbreakable"
  - **perfectly secure** (when used correctly)
  - based on message-symbol specific **independently random** shifts

The one-time pad (OTP) cipher

Fix n to be any positive integer; set $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$

- **Gen**: choose n bits uniformly at random (each bit independently w/ prob. .5)
  - Gen $\rightarrow \{0,1\}^n$
- **Enc**: given a key and a message of equal lengths, compute the bit-wise XOR
  - $\text{Enc}(k, m) = \text{Enc}_k(m) \rightarrow k \oplus m$ (i.e., mask the message with the key)
- **Dec**: compute the bit-wise XOR of the key and the ciphertext
  - $\text{Dec}(k, c) = \text{Dec}_k(c) := k \oplus c$
- Correctness
  - trivially, $k \oplus c = k \oplus k \oplus m = 0 \oplus m = m$

OTP is perfectly secure (using Definition 2)

For all n-bit long messages m_1 and m_2 and ciphertexts c, it holds that

$$\Pr[\, E_K(m_1) = c \,] \quad = \quad \Pr[\, E_K(m_2) = c \,],$$

where probabilities are measured over the possible keys chosen by Gen.

Proof

- events "$Enc_K(m_1) = c$", "$m_1 \oplus K = c$" and "$K = m_1 \oplus c$" are equal-probable
- K is chosen at random, irrespectively of m_1 and m_2, with probability 2^{-n}
- thus, the ciphertext does not reveal anything about the plaintext

OTP characteristics

A "substitution" cipher

- encrypt an n-symbol m using n uniformly random "shift keys" $k_1, k_2, \ldots, k_n$

2 equivalent views

- $\mathcal{K} = \mathcal{M} = \mathcal{C}$
- "shift" method

	view 1		view 2
$\mathcal{K} = \mathcal{M} = \mathcal{C}$	$\{0,1\}^n$	or	G, (G,+) is a group
"shift" method	bit-wise XOR ($m \oplus k$)		addition/subtraction (m +/- k)

Perfect secrecy

- since each shift is random, every ciphertext is equally likely for any plaintext

Limitations (on efficiency)

- "shift keys" (1) are **as long as messages** & (2) **can be used only once**

Perfect, but impractical

Despite its perfect security, OTP has 2 notable weaknesses

- the key has to be **as long as** the plaintext
  - limited applicability
  - key-management problem
- the key **cannot be reused** (thus, the "one-time" pad)
  - if reused, perfect security is not satisfied
    - e.g., reusing a key once, leaks the XOR of two plaintext messages
    - this type of leakage can be devastating against secrecy

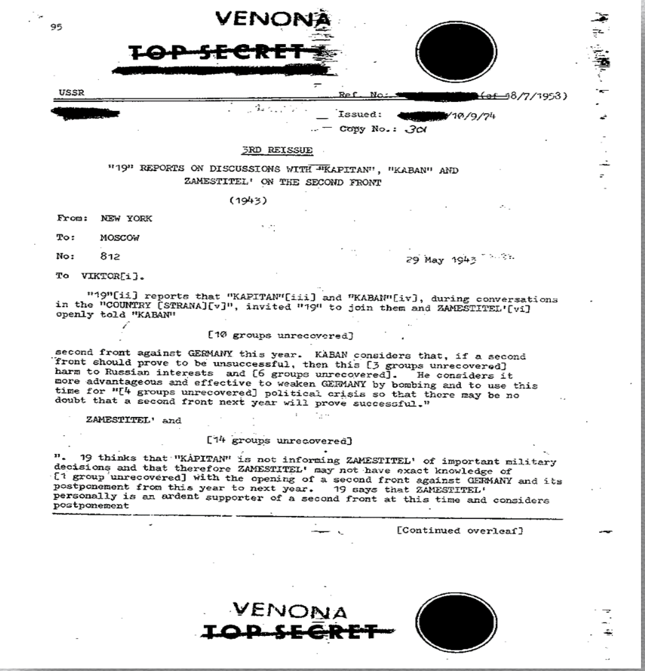
These weakness are detrimental to secure communication

- securely distributing fresh long keys is as hard as securely exchanging messages…

Importance of OTP weaknesses

Inherent trade-off between efficiency/practicality Vs. perfect secrecy

- historically, OTP has been used efficiently & insecurely
  - repeated use of one-time pads compromised communications during the cold war
    - NSA decrypted Soviet messages that were transmitted in the 1940s
    - that was possible because the Soviets reused the keys in the one-time pad scheme
- modern approaches resemble OTP encryption
  - efficiency via use of pseudorandom OTP keys
  - “almost perfect” secrecy



95

VENONA

TOP SECRET [redacted]

[redacted]

USSR

Ref. No.: [redacted](of 18/7/1953)

[redacted]

Issued: [redacted]/10/9/74

Copy No.: 301

3RD REISSUE

"19" REPORTS ON DISCUSSIONS WITH "KAPITAN", "KABAN" AND ZAMESTITEL' ON THE SECOND FRONT

(1943)

From: NEW YORK

To: MOSCOW

No: 812

29 May 1943

To VIKTOR[i].

"19"[ii] reports that "KAPITAN"[iii] and "KABAN"[iv], during conversations in the "COUNTRY [STRANA][v]", invited "19" to join them and ZAMESTITEL'[vi] openly told "KABAN"

[10 groups unrecovered]

second front against GERMANY this year. KABAN considers that, if a second front should prove to be unsuccessful, then this [3 groups unrecovered] harm to Russian interests and [6 groups unrecovered]. He considers it more advantageous and effective to weaken GERMANY by bombing and to use this time for "[4 groups unrecovered] political crisis so that there may be no doubt that a second front next year will prove successful."

ZAMESTITEL' and

[14 groups unrecovered]

". 19 thinks that "KAPITAN" is not informing ZAMESTITEL' of important military decisions and that therefore ZAMESTITEL' may not have exact knowledge of [1 group unrecovered] with the opening of a second front against GERMANY and its postponement from this year to next year. 19 says that ZAMESTITEL' personally is an ardent supporter of a second front at this time and considers postponement

[Continued overleaf]

VENONA

TOP SECRET

[redacted]

3.4 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

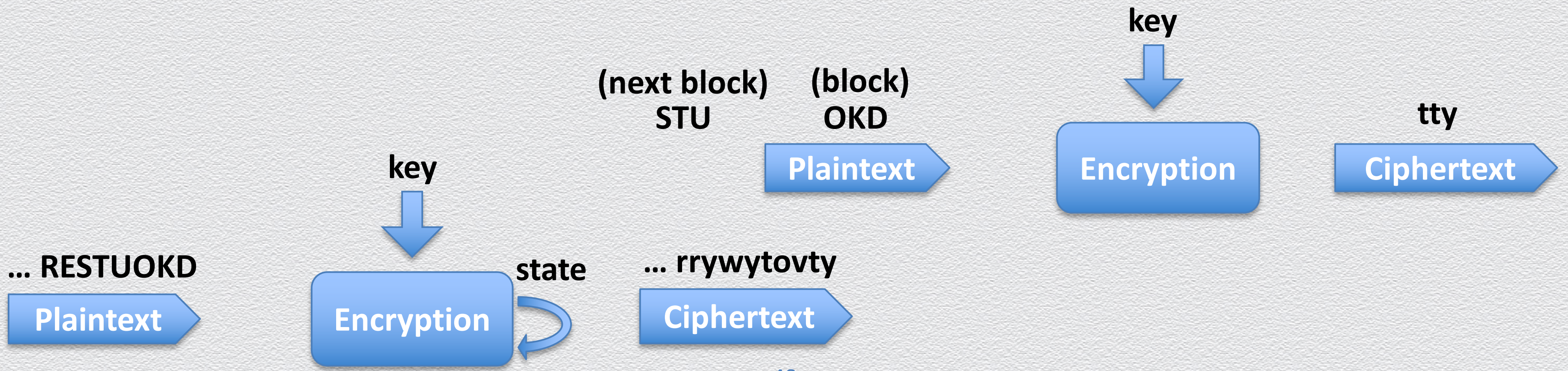
Uses a **short** key to encrypt **long** symbol **streams** into a **pseudorandom** ciphertext

- 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)**



3.5 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 => 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 $10^{80} \Rightarrow 2^{266}$

3.6 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)?

Recall: 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

- Recall: 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

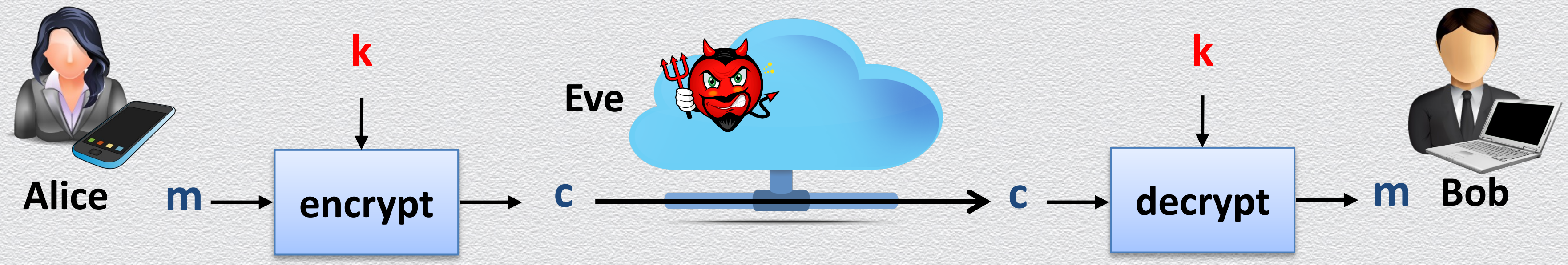
Alice/Bob may transform the transmitted/received message and share info

Example: Formal definitions (2)

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

symmetric-key encryption scheme

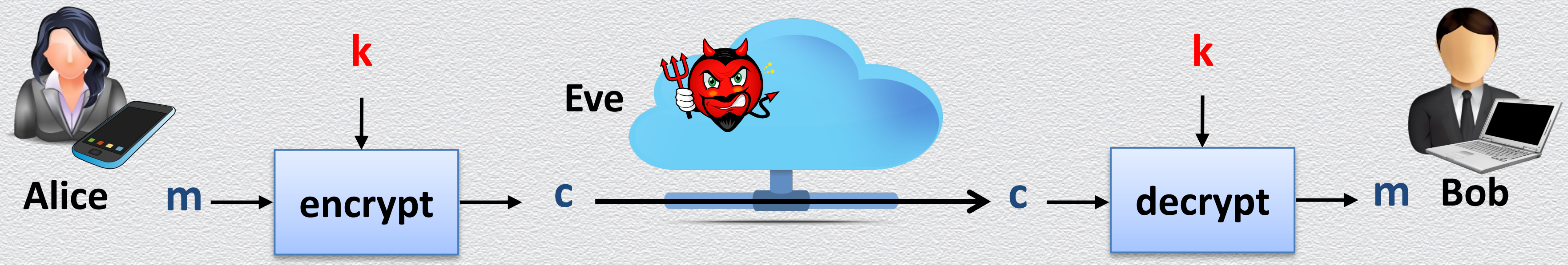
- Alice and Bob share and use a key k
- Alice encrypts plaintext m to ciphertext c and sends c instead of m
- Bob decrypts received c to get a message m’



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

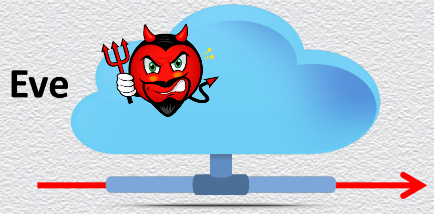
- for any k ∈ $\mathcal{K}$, m ∈ $\mathcal{M}$ and any ciphertext c output of $\text{Enc}_k(m)$, it holds that

$$\textbf{Pr[Dec}_k\textbf{ (c) = m] = 1}$$

Perfect security (or information-theoretic security)

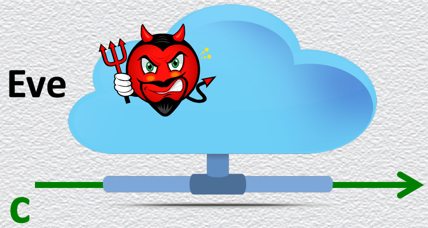
- the adversary should be able to learn **no additional** information on m

random experiment
$\mathcal{D}_{\mathcal{M}} \rightarrow$ **m**
$\mathcal{D}_{\mathcal{K}} \rightarrow$ k
$\textbf{Enc}_k\textbf{(m)} \rightarrow$ **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

$$\textbf{Pr[M = m | C = c] = Pr[M = m]}$$

2) C is independent of M

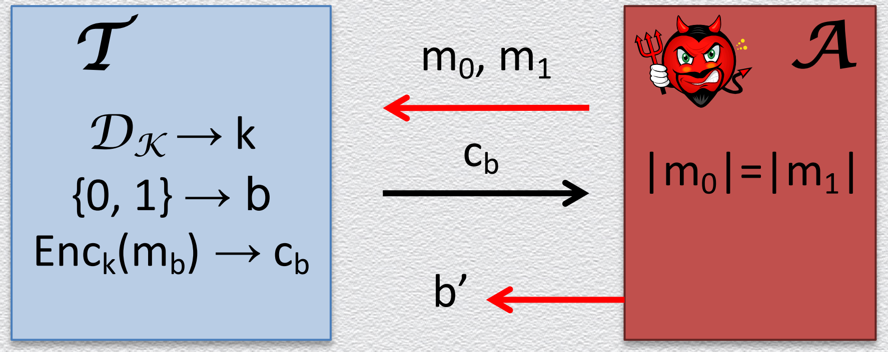
For every m, m' ∈ $\mathcal{M}$ and c ∈ $\mathcal{C}$, it holds that

$$\Pr[\, Enc_K(m) = c\,] = \Pr[\, Enc_K(m') = c\,]$$

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

PPT

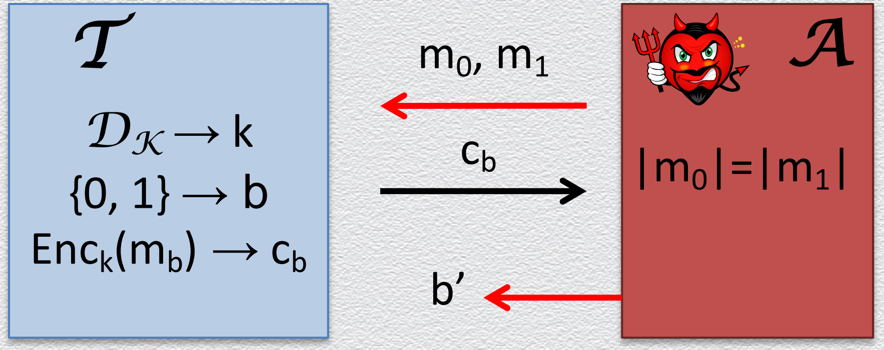
3) indistinguishability

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

PPT

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

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?

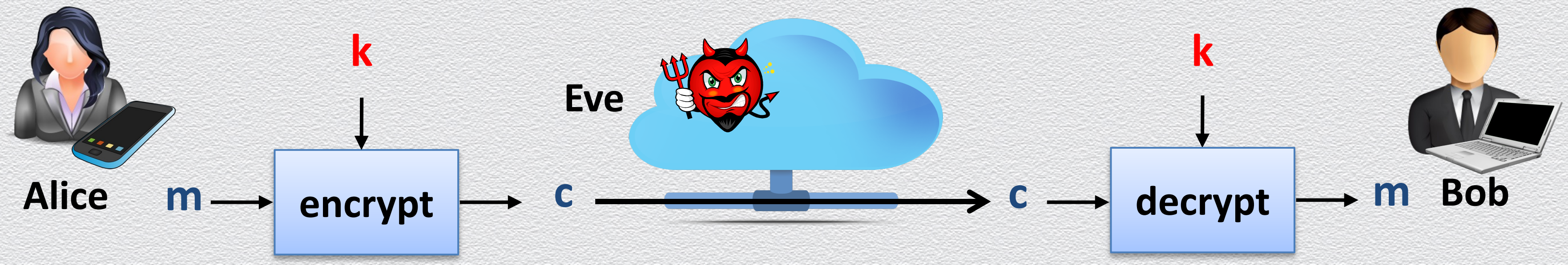
- **basis** for proofs of security
  - security holds under specific assumptions
- **comparison** among possible solutions
  - relations among different assumptions
    - stronger/weaker (i.e., less/more plausible to hold), “A implies B” or “A and B are equivalent”
    - refutable Vs. non-refutable
- **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

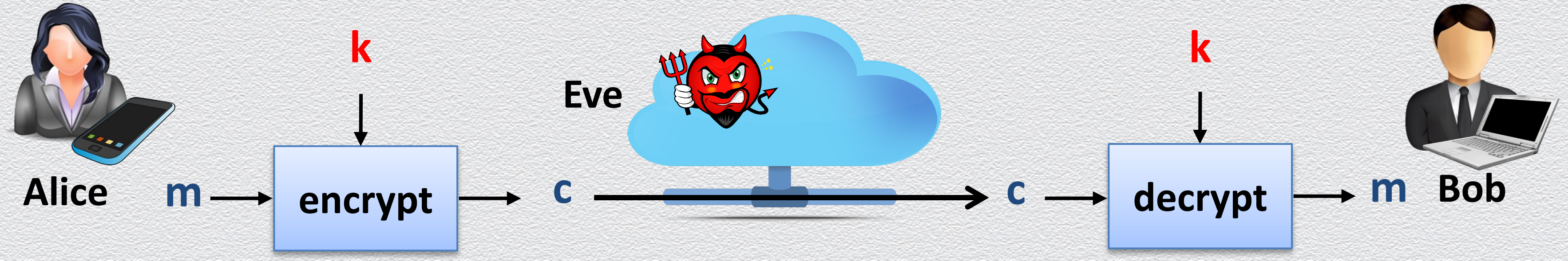
⇒ Eve doesn't know/learn the secret k (shared by Alice and Bob)



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)

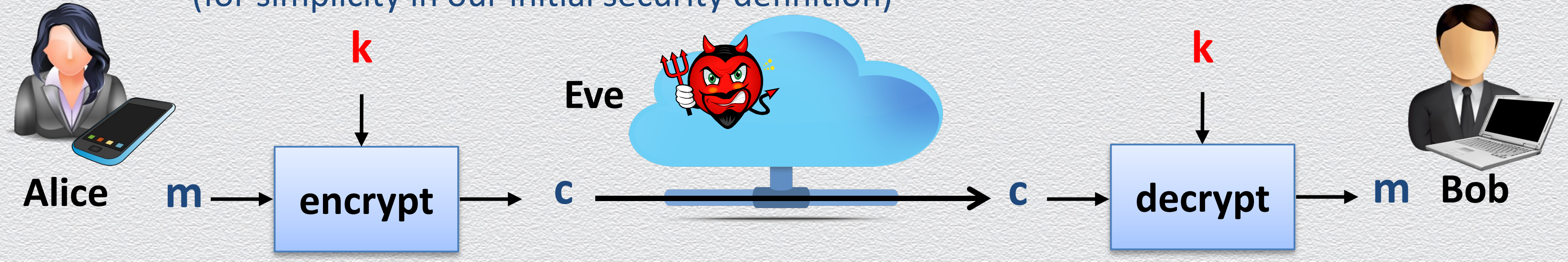


Example: Precise assumptions (3)

- **computing setting**
  - **system set up**, initial state, **key distribution**, **randomness**… ⇒ key k is generated randomly using the uniform distribution
  - means of **communication** (e.g., channels, rounds, messages…)
  - timing assumptions (e.g., synchronicity, epochs, …)

⇒ 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

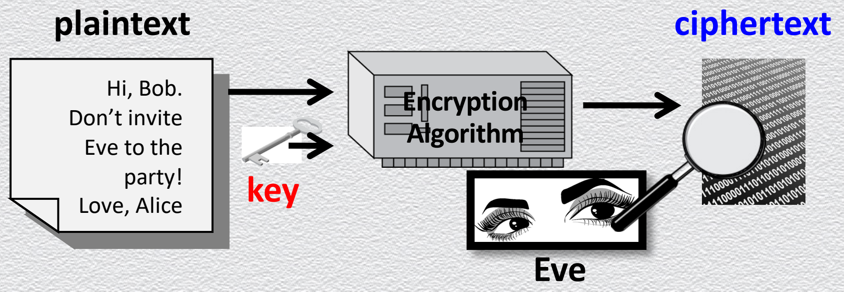


Possible eavesdropping attacks (I)

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

(a)

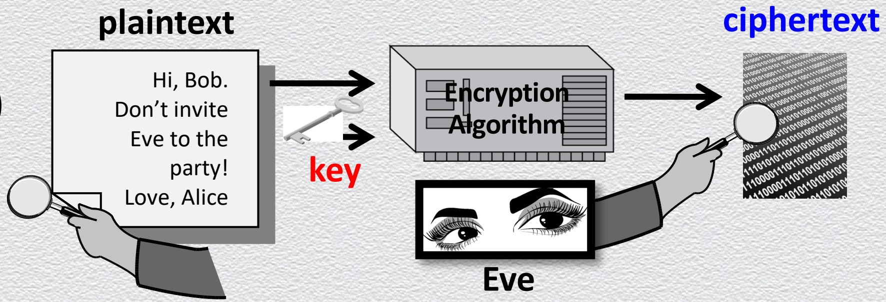


Possible eavesdropping attacks (II)

An attacker may possess a

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

(b)

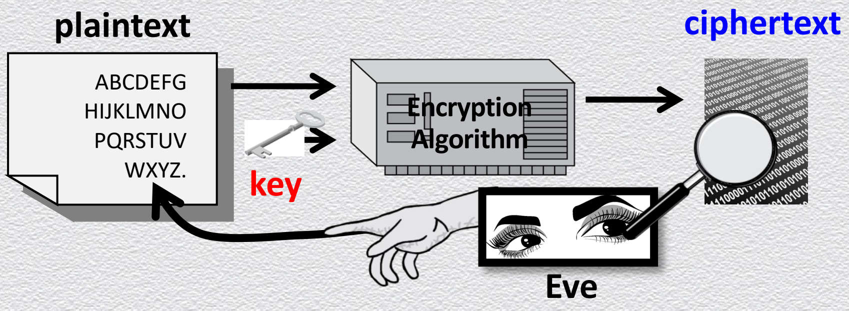


Possible eavesdropping attacks (III)

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)

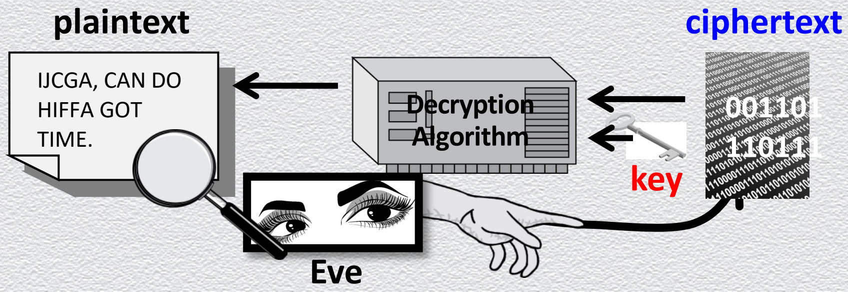


Possible eavesdropping attacks (IV)

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

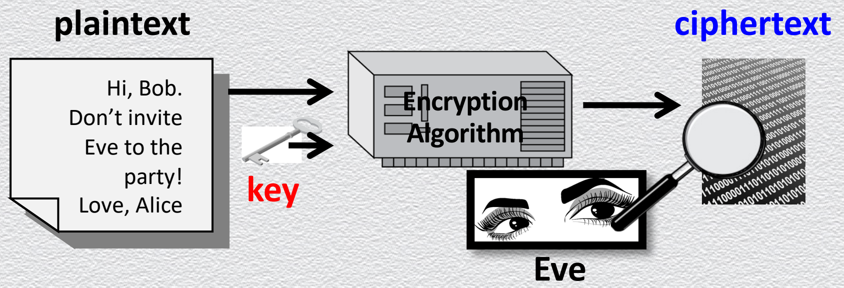
(d)



Main security properties against eavesdropping

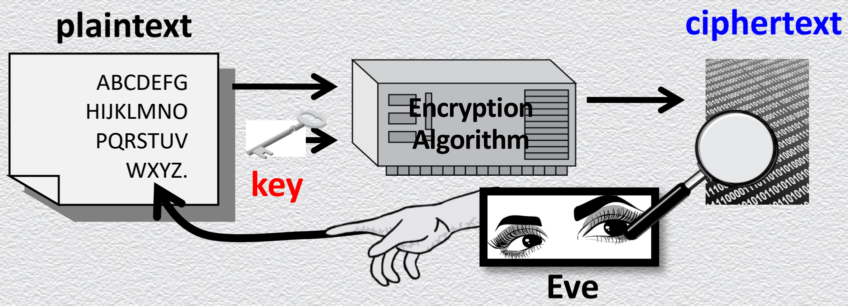
“plain” security

- protects against ciphertext-only attacks
  - EAV-attack



“advanced” security

- protects against chosen plaintext attacks
  - CPA-attack



Game-based computational EAV-security

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

Π $\mathcal{T}$

$\text{Gen}(1^n) \rightarrow k$

$\{0, 1\} \rightarrow b$

$\text{Enc}_k(m_b) \rightarrow c_b$

m_0, m_1

c_b

b'

$\mathcal{A}$

chooses m_0, m_1

s.t. $|m_0| = |m_1|$

We say that (Enc,Dec) is **EAV-secure** if any PPT adversary $\mathcal{A}$ guesses b correctly with probability at most $0.5 + \varepsilon(n)$, 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 $\Pi = \{\mathcal{M}, (\text{Gen}, \text{Enc}, \text{Dec})\}$

Π $\mathcal{T}$

$\text{Gen}(1^n) \rightarrow k$

$\{0, 1\} \rightarrow b$

$\text{Enc}_k(m_b) \rightarrow c_b$

m_0, m_1 ←

c_b →

b' ←

$\mathcal{A}$

chooses $\mathbf{m_0, m_1}$

s.t. $|\mathbf{m_0}| = |\mathbf{m_1}|$

Alternatively:

"**is semantically secure**"

We say that (Enc,Dec) is **EAV-secure** if any PPT adversary $\mathcal{A}$ guesses b correctly with probability at most 0.5 + $\varepsilon(n)$, 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 $\Pi = \{\mathcal{M}, (\text{Gen}, \text{Enc}, \text{Dec})\}$

Π $\mathcal{T}$

$\text{Gen}(1^n) \rightarrow k$

$\{0, 1\} \rightarrow b$

$\text{Enc}_k(m_b) \rightarrow c_b$

m_i

c_i

m_0, m_1

c_b

b’

$\mathcal{A}^{\text{Enc}(k,)}$

chooses $\mathbf{m_0, m_1}$

s.t. $|\mathbf{m_0}| = |\mathbf{m_1}|$

We say that (Enc,Dec) is **CPA-secure** if any PPT adversary $\mathcal{A}$ guesses b correctly with probability at most $0.5 + \varepsilon(n)$, 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 probabilistic encryption – can you see why?**
- EAV-security for multiple messages implies probabilistic 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