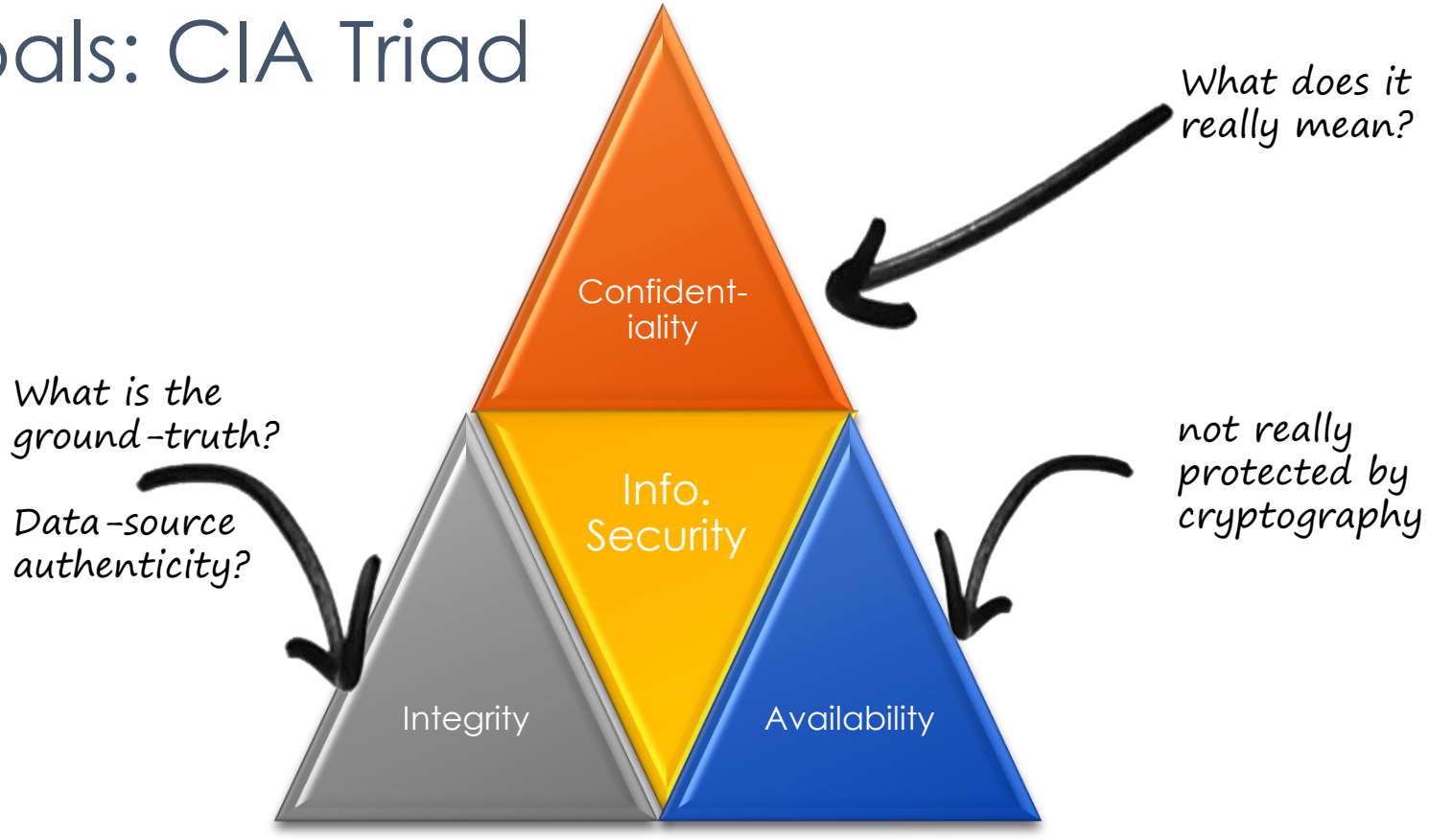
IERG4150
Intro. to Cryptography

Sherman Chow
Chinese University of Hong Kong
Fall 2022
Lecture 1: Introduction

Goals: CIA Triad



Caesar Cipher

- Consider the 26 alphabets of English
- Encoded them as a number in [0, 25]
- E(*m*) → *m* + *k* mod 26
- D(*c*) → *c* – *k* mod 26
- salad -> wepeh (*k* = 4)

- Review concepts:
  - Encoding (is not encryption)
  - Modular arithmetic

- Vulnerable to Frequency Analysis
  - w/ knowledge of plaintext distribution

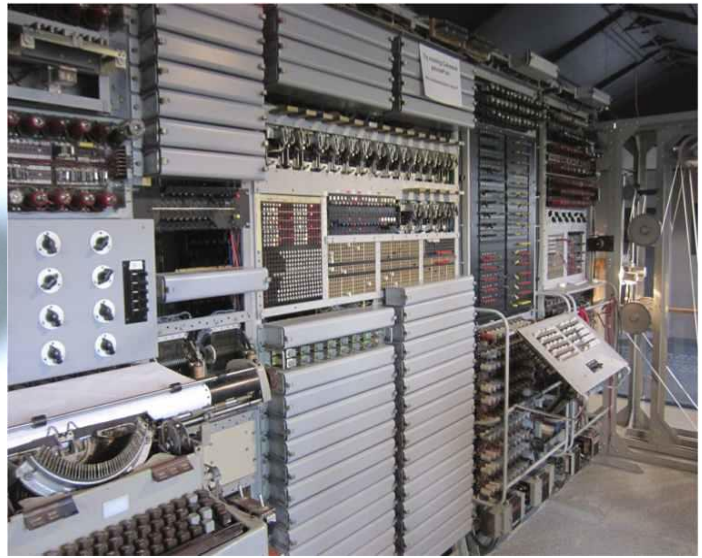
Vigenère Cipher

- Variants of Caeser Cipher
- Idea: not always map a plaintext to the same ciphertext
- Plaintext: AttackAtDawn (case insensitive)
- Key: Lemon
- Key "Sequence": LEMONLEMONLE
- Ciphertext: LXFOPVEFRNHR

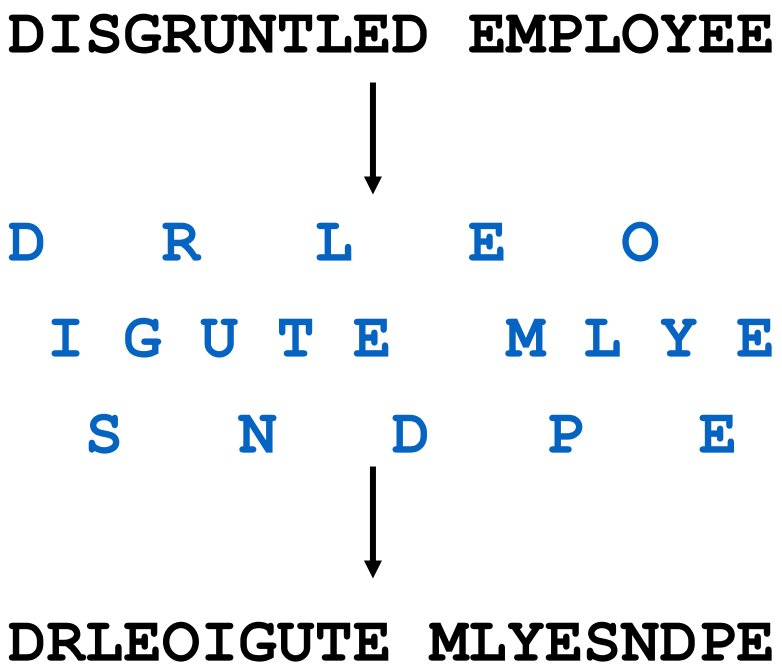
- How to attack?

Enigma

- Caeser and Vigenère Ciphers are both polyalphabetic
- Based on Substitution
- So does Enigma



“Rail-Fence” Cipher via Transposition



What (Modern) Cryptography is?

- not a magic spell that solves all security problems
- providing solutions to *cleanly defined* problems
  - often *abstract away* important but messy real-world concerns
- "Cryptographic guarantees"/"Provable security":
  - What happens (or what cannot happen) in the presence of certain well-defined classes of attacks
  - What if the model is too restrictive (in defining the attacks)?
  - What if the "real-world" attackers don't follow the "rules"?
  - Disappointing/Underwhelming?

Defining Security

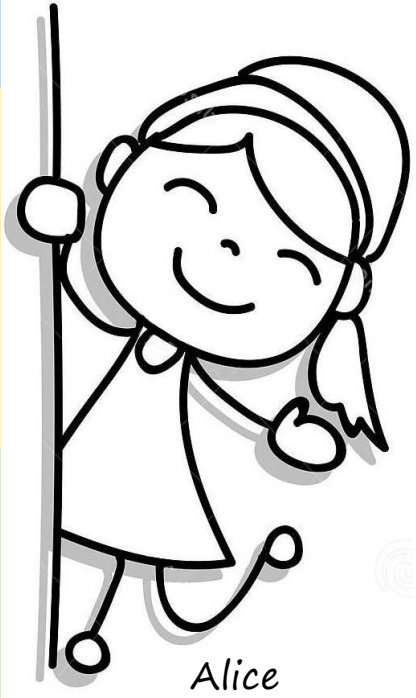
- Making the nebulous concept of "security" concrete
- Breaking the vicious circle of "cat-and-mouse" games

- We will try to model the attacker as "powerful" as possible

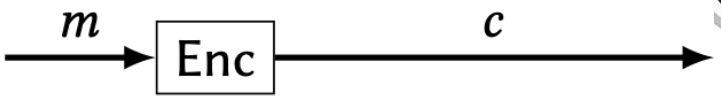
- Always keep in mind: we define (*i.e.*, limit) our problems

"To define is to limit."
—Oscar Wilde

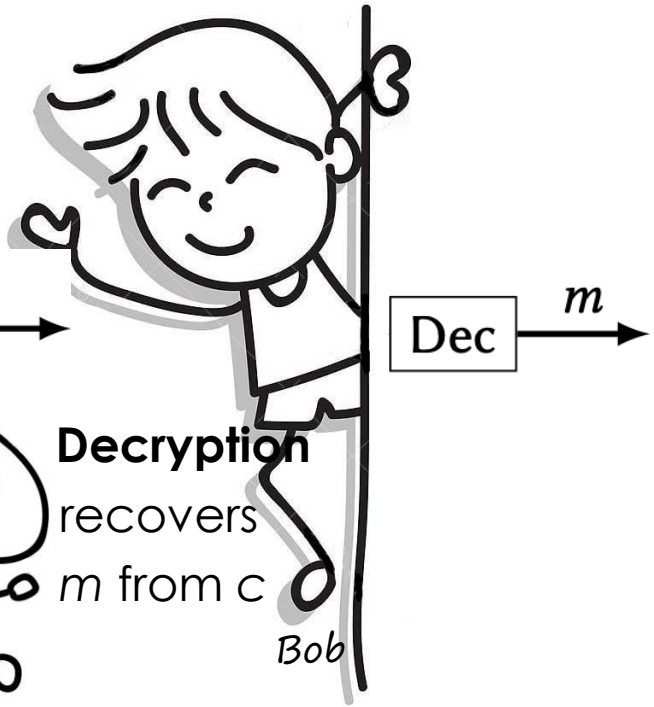
“Private” (Confidential) Communication



- **Plaintext**: *m*
- **Ciphertext**: *c*
- **Encryption** turns *m* into *c*



- Eavesdropper can (passively) observe the communication



Decryption recovers *m* from *c*

Secret, or secrecy of the algorithms?

- We want Bob to be able to decrypt c
- but Eve to not be able to decrypt c
- Suppose Eve has unbounded computational power
- Argue that both sender and receiver must share a secret not known to the adversary [Exercise]
- Hide the details of the Enc() and Dec() algorithms secret?
  - how crypto was done throughout most of the last 2000 years
  - but it has major drawbacks!

Kerckhoffs' Principle

"Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi."

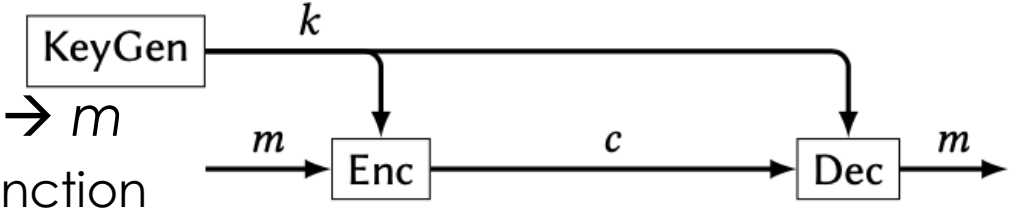
- A system designer wants the system to be widely used.
- It is hard to keep a secret (*e.g.*, reverse engineering).
- If details of Enc() and Dec() are leaked, what can we do?
- Invent a new encryption system!
  - Inventing even a good one is already hard enough!
- [The method] must not be required to be secret, and it must be able to fall into the enemy's hands without causing inconvenience.
- Bottom line: Design your system to be secure even if the attacker has complete knowledge of all its algorithms.
  - vs. security by obscurity

Confidentiality

- Prevent the disclosure of info. to unauthorized party
- Encryption: use a "key" to turn a *plaintext* into a *ciphertext*
- Without the "secret key", the ciphertext is not "useful"
- What constitutes an encryption?
  - Framework / A suite of algorithms

What constitutes an encryption scheme?

- Key generation algorithm (KeyGen)
  - Input: security parameter λ (lambda)
  - Output: a key k
- $\mathrm{Enc}_k(m) \rightarrow c$, $\mathrm{Dec}_k(c) \rightarrow m$
  - *i.e.*, they are key-ed function
  - All these algorithms are supposed to be public



- A crypto scheme/construction is a collection of algorithms
- *Symmetric*-key encryption = (KeyGen, Enc, Dec)

Syntax forms the basis of Security

- We call the inputs/outputs (*i.e.*, the "function signature") of the various algorithms the *syntax* of the scheme.

- KeyGen is a *probabilistic/randomized* algorithm
- Knowing the details (*i.e.*, source code) of a randomized algorithm does *not* mean you know the *specific* output it gave when it was executed

- Encoding/decoding methods is *not* encryption [Why?]
  - What is "b25seSBuZXJkcyB3aWxsIHJlYWQgdGhpcw=="?

What are outside our model's protection?

- The fact that Alice is sending something to Bob
  - We only want to hide the *contents* of that message
  - Steganography hides the existence of a communication channel
- How c reliably gets from Alice to Bob
- We aren't considering an attacker that tampers with c (causing Bob to receive and decrypt a different value)
  - We will consider such attacks later though

What it takes in the "real world"?

- How Alice and Bob actually obtain a common secret key
- How they can keep them secret while (keep) using it
- How to uniformly sample random (bit-)strings?
  - No randomness, no cryptography
  - Obtaining uniformly random bits from *deterministic* computers is extremely non-trivial

"Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin."
— John von Neumann

Probabilistic Polynomial Time (PPT) Algo.

- $y = A(x)$
- Input x is of *size/length n*
  - We write $|x| = n$
- A PPT algorithm has $O(n^c)$ run-time, *c* being a constant
  - We say a PPT algorithm is an "efficient" algorithm
- Probabilistic: allows "flipping a coin" to make it randomized
- $y \leftarrow A(x)$
- *y* denotes the random variable corresponds to A's output
- Or $y = A(x; r)$, where *r* denotes *A*'s "*coin tossing*"
  - *r*'s length is also polynomial in *n*
  - when we had the need to specify the randomness explicitly

Negligible Function

- A function $v(n)$ is called negligible, denoted **negl**(n), if:
- $(\forall c > 0)\ (\exists n')\ (\forall n \geq n')\ [v(n) \leq 1/n^c]$
- Less than the inverse of any polynomial for large enough n

- Prob. of breaking a secure system should be negligible in n

- Let **poly**(n) denote some polynomial function in n
- We have $\text{poly}(n) \cdot \text{negl}(n) = \text{negl}(n)$ (abusing notations)

Security Parameter (& some notations)

- We want a "set" of cryptosystems parameterized by n
- Algo.'s run by all parties take commonly agreed input n
- They run in time polynomial in their input length n
- Notations:
  - poly(n): runtime of all parties are sufficiently fast, e.g., n^3
  - negl(n): e.g., $1/2^n$ is in negl(n)
  - 1^n denotes n "copies" of 1's, i.e., 1^n is in $\{0, 1\}^n$
- Security parameter of the system is 1^n (with length n bits)
  - but not n (length of n is log(n) bits)

Tasks of Crypto. Study [*]

- Identification of the problem / application scenario
- Identification of the primitive which may be useful
  - Do not re-invent the wheel
  - Extending existing primitives
  - Relation between primitives (one implies another?)
- Definition of Functional Requirements
  - A suite of algorithms / protocols, their input & output behavior / interfaces
  - System model: what entities are involved, which entity executes which algorithm/protocols
- Definition of Security requirements
  - Relation of security notions (one implies another?)
- Construction of the schemes
- Analysis of the proposed construction
  - Security Proof: Provable Security!
  - Efficiency (Order Analysis and/or Experiment on Prototype Implementation)