Intro to IT Security

CS306C—Fall 2022

Prof. Antonio R. Nicolosi

Antonio.Nicolosi@stevens.edu

STEVENS INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY

Introduction to IT Security

Cryptography and IT Security

- Cryptography provides valuable tools for IT security
- Superficial knowledge of Crypto often leads to real-worl attacks
  - Insufficient entropy for key generation in early Debian GNU/Linux (2008)
  - Early implementations of SSL (1995)
  - RSA key generation in embedded devices (2012)
  - Improper use of SHA as an authentication code

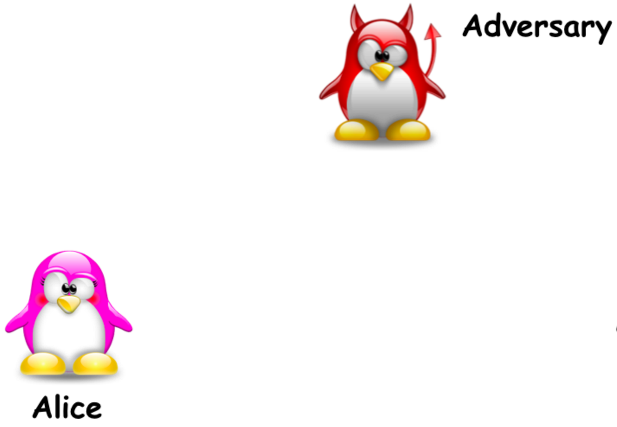
Motivation

- Most problems stemmed from misunderstanding of protection afforded by each cryptographic tool
- $\Rightarrow$ Need rudimentary knowledge of common protocols to use crypto effectively

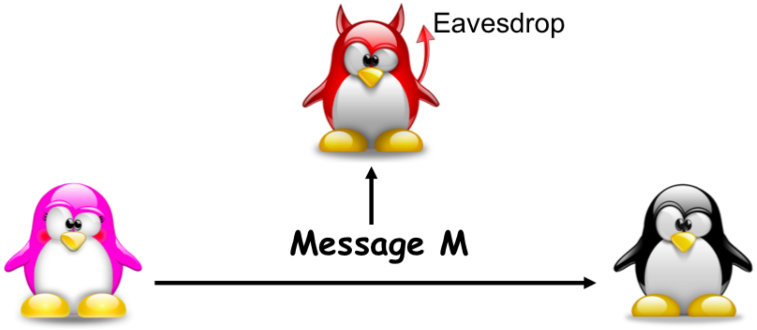
What is Cryptography?

- “*Cryptography*”: From Greek “*Crypto*” (“hidden” or “secret”) and “*graphy*” (“writing”)
- *Cryptography* is a branch of both mathematics and computer science and is related closely to information theory, computer security and engineering
- In short: theory and practice of *controlling* information
- E.g.: encryption scheme (our initial focus)
- Other examplesL signature schemes, hashing / fingerprinting, commitments, authentication codes...

Cryptographic Concerns

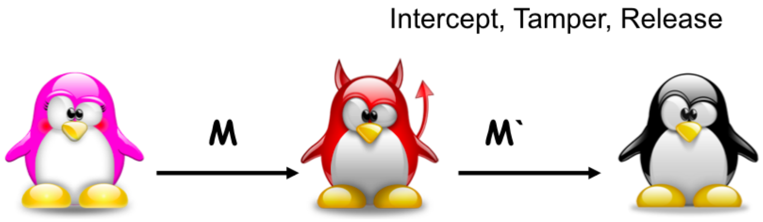


Crypto Requirements: Data Secrecy



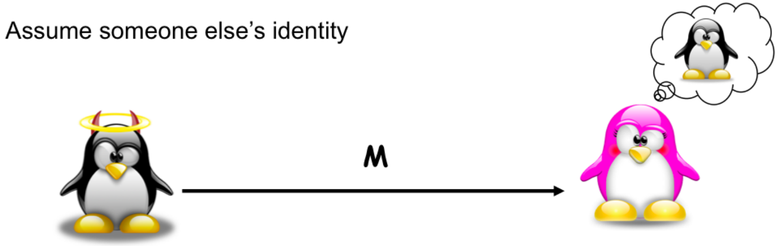
- Protect against unauthorized disclosure of the msg
  - If **A** sends a msg to **B**, no one else should understand its content

Crypto Requirements: Data Integrity



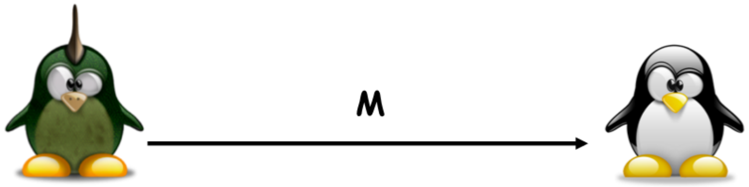
- The receiver should be able to check whether the msg was modified during transmission
  - No one should be able to tamper with the msg, without the recipient noticing the alteration

Crypto Requirements: Data Origin



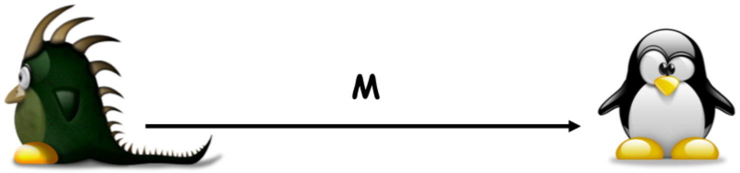
- The receiver of a msg should be able to verify its origin
  - No one else should be able to send a msg to **A** pretending to be **B**

Crypto Requirements: Non-Repudiation



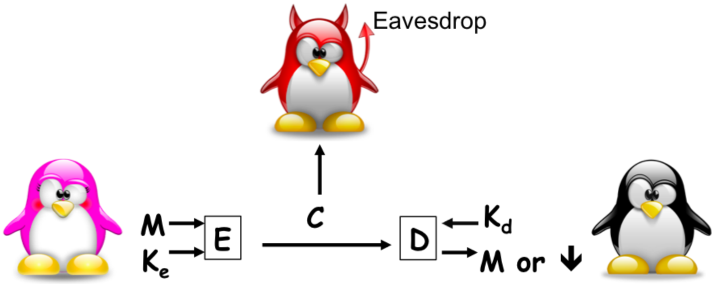
- The sender should not be able to later deny responsibility for msgs sent

Crypto Requirements: Non-Repudiation



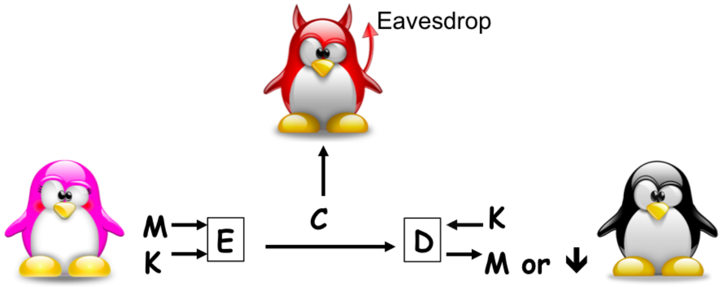
- The sender should not be able to later deny responsibility for msgs sent

Achieving Data Secrecy: Encryption



- Notation
  - **M**: msg or plaintext
  - **C**: encrypted msg or ciphertext
  - **E**: encryption algorithm
  - **D**: decryption algorithm
  - K_e: encryption key
  - K_d: decryption key

Symmetric Encryption

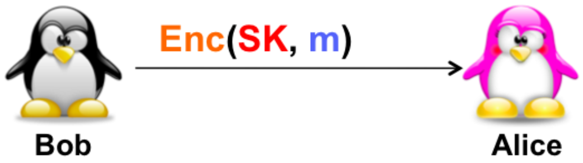


- **A** and **B** share the same secret information
  - $K = K_e = K_d$: secret

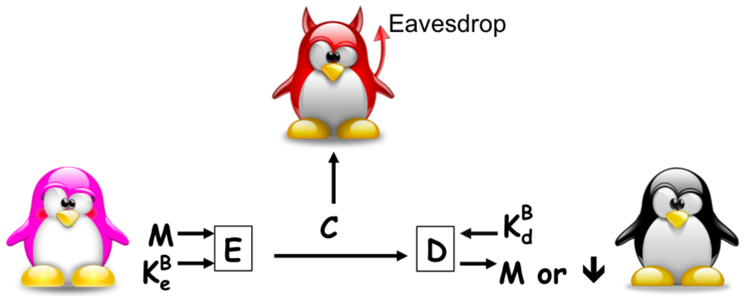
Symmetric Encryption

- Defined by three algorithms:

- Gen(λ) → (SK)	outputs secret key SK
- Enc(SK, m) → c	encrypt m using secret key SK
- Dec(SK, c) → m	decrypt c using SK



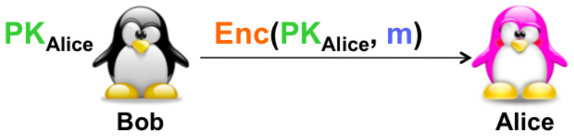
Asymmetric Encryption



- **A** and **B** each have their own pair of (Public, Secret) information
  - K_e^A: public; K_d^A: secret
  - K_e^B: public; K_d^B: secret

Asymmetric Encryption

- Defined by three algorithms:
  - $\text{Gen}(\lambda) \rightarrow (\text{SK}, \text{PK})$ outputs secret key SK and public key PK
  - $\text{Enc}(\text{PK}, m) \rightarrow c$ encrypt m using public key PK
  - $\text{Dec}(\text{SK}, c) \rightarrow m$ decrypt c using SK


PK_{Alice}
$\text{Enc}(\text{PK}_{\text{Alice}}, m)$
Bob
Alice

Kerckhoff’s principle

- **E** and **D** are public
- Motivations
  - Modularity: if key is leaked, easier to change the key than the entire algorithm
  - Reverse engineering of the executable of **E** and **D**
  - Standardization

Defining Security

In designing cryptographic systems we need to understand exactly what security properties are guaranteed and how the systems can be made secure against potential attacks

The Definition, Intuitively

We would like for our cryptosystems to behave like a completely opaque, locked box. What would some of the properties be?

The Definition, Intuitively

We would like for our cryptosystems to behave like a completely opaque, locked box. What would some of the properties be?

1. The locked box reveals no information about the contents (beyond perhaps some rough size estimates)
2. Only the key can open it

This should still be the case even if an adversary

- Has seen the contents of many other boxes
- Knows exactly how the lock works (but does not have the key)

Perfect Secrecy

- Informal definition:

  A system is said to have perfect security (a.k.a. *information theoretic security*) if any given ciphertext contains no information about the plaintext. Equivalently, a given ciphertext is equally likely to have been produced by any message in the plaintext set.

- Note that even an adversary with infinite computing power can learn nothing about the plaintext given the ciphertext.

Perfect Secrecy

- Formal definition given by Shannon in ’49: Provides a mathematical model to prove impossibility of breaking an encryption scheme

  If a cryptosystem has perfect security, then it must be that the entropy of the key space is larger than that of the message space

- Essentially, cryptosystems with perfect security require a key that is as long as the message, and that the same key is never used twice
- Somewhat paradoxical: in order to securely exchange an n-bit message, one must first securely exchange an n-bit key
  - This is especially problematic in the public key model, since the encryption key is assumed to be used for many messages without any synchronization among the senders

Example: One-Time Pad (OTP)

- **A** and **B** share a secret key K
  - K: random bit sequence, $|K| = |M|$
- Based on the bit-wise XOR function
  - $E(M, K) : C = M \oplus K$
  - $D(C, K) : M = C \oplus K$

Advantages of OTP

- **E** and **D** are the same algorithm
- Bitwise XOR is a ‘cheap’ operation
- Perfectly secure, as long as...
  - the key is truly random
  - the key has the same length as the plaintext
  - the key is used only once

Disadvantages of OTP

- Hard/expensive to generate long random bit-strings
- How does **A** communicate the key to **B**?
- XOR alone does not provide data integrity
  - $M = 10111$, $K = 00110$, $C = 10111 \oplus 00110 = 10001$
  - If **B** receives $\bar{C} = 11001$, then $\bar{C} \oplus K = 11111 \neq M$
- Need to generate a new key for each msg
  - $M_1 = 00000$, $K = 00110$, $C_1 = 00110$
  - $M_2 = 11111$, $K = 00110$, $C_2 = 11001$
  - Then $C_1 \oplus C_2 = 11111 = M_1 \oplus M_2$

Perfect Secrecy: Too Strong?

- In light of Shannon’s theorem, it seems that perhaps our goal was too strong to be of any use. In what follows we describe an alternative that admits many instantiations that are quite practical

Computational Approach

- For practical reasons, rather then considering schemes that are impossible to break, we consider systems which are computational difficulty to break
  - Attacks might exist as long as cost to mount them is prohibitive (in term of computing time and money)
- But what should breaking an encryption system mean, and what types of attacks should be defended against?
- Will define an attack model by its *goal* and *means*

Attacker’s Goals

- Recover the key?
  - What if attacker can learn the plaitext without the key?
- Recover plaitext from ciphertext?
  - What if attacker can learn some bits or some function of the plaitext?
- Correlate plaitext with ciphertext?
  - What if attacker sees two identical ciphertexts and infers that the corresponding plaintexts are identical?
    - Encryption must be randomized!
- Preventing easier attack goals $\Longrightarrow$ Better security

Attacker’s Goals (cont’d)

- For stronger security, rule out plaintext-ciphertext correlations
- How to formalize “correlation”?
  - Probability theory (Shannon security)
    - Back to OTP limitations
  - Modern approach: Indistinguishability (Goldwasser-Micali)
    - *Even the mother couldn’t tell her twins apart when they’re masqueraded*

Indistinguishability

- Informal definition:

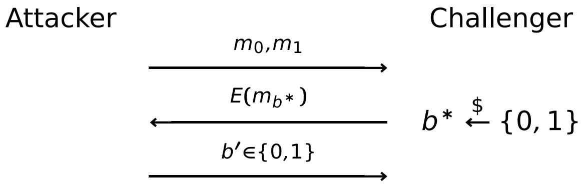
  For all probability distributions on the message space, whatever an adversary can compute in polynomial time about the plaintext given the ciphertext, they should also be able to compute without the ciphertext.

- Simply put, indistinguishability is the computational analog of perfect security: *with respect to what is efficiently computable*, the ciphertext yields no information about the plaintext

Indistinguishability

- Formalized as *challenge game*
  1. The challenger picks the encryption key
  2. Attacker chooses two different plaintexts $M_0, M_1, |M_0| = |M_1|$
  3. Challenger encrypts either M_0 or M_1
  4. Attacker try to guess which message is "inside" the ciphertext
- Remark: can always correctly guess with probability $\frac{1}{2}$
- How much better attacker can do?
  - The scheme is secure if no *efficient* attacker can do *substantially better* than random guessing

Indistinguishability Attack Game



The attacker "wins" if $b' = b^*$. If k is the length of the key, denote Attacker's probability of success by $Adv(k)$. We say that the scheme is secure against indistinguishability if for *any* probabilistic polynomial time Attacker and for *every* constant c,

$$Adv(k) < 1/2 + \frac{1}{k^c}$$

for sufficiently large k.

Attacker's Means

- Kerckhoffs Principle: Details of encryption and decryption algorithms are known
- But attacker may have access to additional information
- The broader the type of info available to the attacker, the stronger the attack model
  - Ciphertext-only attack
  - Known-plaintext attack
  - Chosen-plaintext attack (CPA)
  - Chosen-ciphertext attack (CCA)

Attacker's Means (cont'd)

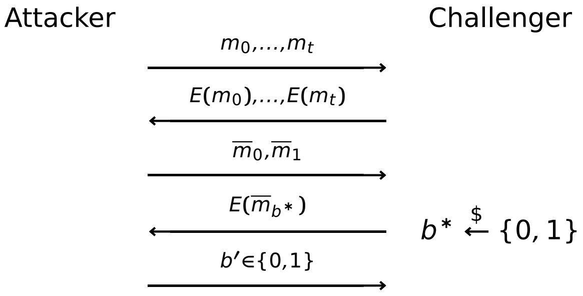
- Ciphertext-only attack
  - Attacker can observe ciphertexts for plaintexts she *does not know*
- Known-plaintext attack
  - Attacker can observe ciphertexts for plaintexts she *knows*
- Chosen-plaintext attack (CPA)
  - Attacker can observe ciphertexts for plaintexts she *chooses*
- Chosen-ciphertext attack (CCA)
  - As a chosen-plaintext attack, but additionally attacker can observe *plaintexts* for *ciphertexts* she *chooses*

Chosen-Plaintext Attack Game

1. The challenger picks the encryption key
2. Attacker chooses as many plaintexts as she wants and learns a corresponding ciphertext
3. Attacker chooses two different plaintexts M_0, M_1, $|M_0| = |M_1|$
   - Note: Even allowed to pick plaintexts for which she previously learned a ciphertext

   $\Rightarrow$ Secure encryption ought to be randomize
4. Challenger encrypts either M_0 or M_1 and gives the resulting ciphertext C^* to the attacker
5. Attacker can again choose as many plaintexts as she wants and learns a corresponding ciphertext
6. Attacker try to guess which message is “inside” the ciphertext

Chosen-Plaintext Attack Game



Chosen-Ciphertext Attack Game

1. The challenger picks the encryption key
2. Attacker chooses as many plaintexts as she wants and learns a corresponding ciphertext
3. Attacker chooses as many ciphertexts as she wants and learns the corresponding plaintext
4. Attacker chooses two different plaintexts M_0, M_1, $|M_0| = |M_1|$
   - Note: Even allowed to pick plaintexts for which she previously learned a ciphertext

   $\Rightarrow$ Secure encryption ought to be randomize
5. Challenger encrypts either M_0 or M_1 and gives the resulting ciphertext C^* to the attacker
6. Attacker can again choose as many plaintexts as she wants and learns a corresponding ciphertext
7. Attacker can again choose as many ciphertexts as she wants and learns a corresponding plaintexts
   - Clearly, the attacker cannot ask to see the decryption of C^*
8. Attacker try to guess which message is "inside" the ciphertext

Combining Goals and Means

Goal → ↓ **Means**	key recovery	plaintext recovery	(plaintext-ciphertext) correlation
ciphertext only			
known plaintext			
CPA			
CCA			√