Algoritmi di Crittografia

Corso di Laurea Magistrale in Informatica

A.A. 2018/2019

Algoritmi di Crittografia

1. Administrivia
2. Introductory concepts
   - Encryption, decryption, and other introductory notions
   - Classical ciphers
   - Security, goals and attack models

Course material, grading, office, ...

- 6 credits, 42 class hours, 150 hrs students' workload
- Textbook: *Jean-Philippe Aumasson, Serious Cryptography: A Practical Introduction to Modern Encryption, No Starch Press*
- Additional material (slides, code, problems, ...) available at: https://github.com/leoncini/Algoritmi-di-crittografia
- Office hours: Tuesday, 2.30 - 4.00 pm
- Grading: oral exam also with practical exercises

Algoritmi di Crittografia

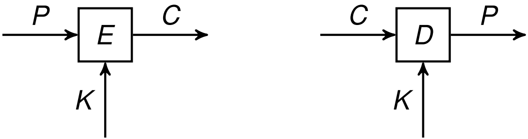
1 Administrivia

2 Introductory concepts
- Encryption, decryption, and other introductory notions
- Classical ciphers
- Security, goals and attack models

A little terminology

- *Plaintext* is our piece of “source” data (the one we would like to keep secret)
- *Encryption* is the process that makes data (plaintexts) incomprehensible
- Encrypted data are referred to as *ciphertexts*
- *Decryption* is the process of turning ciphertexts back to plaintexts
- Encryption and decryption are the two component algorithms of a *cipher*
- A cipher also makes use of a piece of secret information, known as the *key*
- When the key to decrypt is the same as (or can be easily computed from) the one used to encrypt we speak of *symmetric encryption*

Encryption and decryption



(Symmetric) Encryption and decryption schemes: P and C are the plaintext and ciphertext, respectively, while K is the secret key

A fundamental principle

In an encryption scheme, security must reside in the secrecy of the key only (and not of the algorithms)

A. Kerckhoff, *La Cryptographie Militaire*, 1883.

Try to guess some reasons for why this is reasonable

Characters on the stage...

- *Cryptographer* is a person who manages to design cryptographic algorithms and protocols
- *Cryptoanalyist* is a person who try to design algorithms to break ciphers and other cryptographic protocols
- *Alice* and *Bob*, sometimes contracted simply to $\mathcal{A}$ and $\mathcal{B}$, are names traditionally used in cryptography to denote characters that want to communicate confidential messages over an *insecure channel* C
- *Eve* (or simply $\mathcal{E}$) is the malicious adversary that can read all the stuff that transits over C
- Other characters will be introduced ahead in these notes

Algoritmi di Crittografia

1 Administrivia

2 Introductory concepts
- Encryption, decryption, and other introductory notions
- Classical ciphers
- Security, goals and attack models

Classical ciphers: Caesar cipher

- A *mono-alphabetic* cipher that works by replacing each character of the plaintext with a different character at "fixed distance"
- The distance is the secret key k
- Example: let $\{\mathtt{A},\mathtt{B},\ldots,\mathtt{Z}\}$ be the input alphabet, and let $k = 3$; then encryption replaces `A` with `D`, `B` with `E`, and so on
- For the purpose of substitution, `A` is assumed to follow `Z`; hence `X` is replaced by `A`, `Y` by `B`, and `Z` by `C`
- Clearly, other alphabets are possible
- The number of allowable keys is limited to the size of the alphabet
- To decrypt a ciphertext, simply replace each character with the one at key distance (but in the other direction)

Encryption with Caesar cipher

```
CHARSET = "␣'.0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"

def encrypt(plaintext, key):
    key = key%len(CHARSET)
    ciphertext = ''
    for p in plaintext:
        if p in CHARSET:
            i = CHARSET.find(p)
            j = (i+key)%len(CHARSET)
            c = CHARSET[j]
            ciphertext += c
        else:
            raise NotSupportedSymbol
    return ciphertext
```

Challenge 1

- Let $\Sigma = \{␣'.,:;!?()\mathtt{a}-\mathtt{z}\mathtt{à}\mathtt{è}\mathtt{é}\mathtt{ì}\mathtt{ò}\mathtt{ù}\backslash\mathtt{n}\}$
- Unknown key k (any integer value)
- Decrypt: $C =$
  `:òhìv..rxxzùhtzw'r,ùhtùòhzéhtzw'r'zùhuzhtv.r'v`
- Plaintext language is Italian...

Decryption with Caesar cipher

```
def decrypt(ciphertext, key):
    return encrypt(ciphertext,-key)
```

Challenge 1: solution

```
# Brute force approch, i.e., try all possible keys

for k in range(len(CHARSET)):
    print(decrypt(C,k))
```

- See file `challenge1.py`

Frequency analysis

- For the Caesar cipher, brute force is clearly adequate
- However, for this and other *mono-alphabetic* ciphers, we can adopt a strategy that is more efficient (in general), namely *frequency analysis*
- With mono alphabetic substitutions, the hidden identity of the plaintext symbols can be revealed by analyzing the frequencies of the symbols in the ciphertext
- Suppose we know the plaintext is written in English
- Suppose also we observe that the “strange” symbol @ is the one that appears most often in the cipher text.
- Then it is possible (and even likely) that the secret key maps the letter *e* to @, since *e* is the letter that occurs most often in English texts

Challenge 2

- Decrypt a (relatively) long file written in Italian
- The text is encrypted using the Caesar cipher...
- ... but use frequency analysis!
- The reference for the italian language is a very long (classical) text
- See file `challenge2.py`

The cipher of Capt. Kidd

- Monoalphabetic (substitution) cipher, not easy as Ceaser's but still insecure
- Appears in a short story by Edgar Allan Poe, named *The gold bug*

  “53‡‡†305))6∗;4826)4‡.)4‡);806∗;48†8
  ¶60))85;;]8∗;:‡∗8†83(88)5∗†;46(;88∗96
  ∗?;8)∗‡(;485);5∗†2:∗‡(;4956∗2(5∗−4)8
  ¶8∗;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡
  1;48†85;4)485†528806∗81(‡9;48;(88;4
  (‡?34;48)4‡;161;:188;‡?;”

- See file `challenge3.py`

Classical ciphers: Vigenère cipher

- Combines Caesar with different shifts at different positions
- Shift at position i depends on the j-th character of a secret key k, where $j = i \bmod |k|$.
- Say $k = \texttt{crypto}$ and let $P = \texttt{algorithms}$ be the plaintext; then (assuming $\Sigma = \{\mathrm{a}, \mathrm{b}, \ldots, \mathrm{z}\}$ for simplicity):

key:	c	r	y	p	t	o	c	r	y	p
p:	a	l	g	o	r	i	t	h	m	s
c:	c	c	e	d	k	w	v	y	k	h

Python code for the Vigenère cipher

```
def encrypt(plaintext,key,offset=lambda x: x):
    n, k = len(CHARSET), len(key)
    ciphertext = ''
    for i, p in enumerate(plaintext):
        if p in CHARSET:
            j = i%k
            p_index = CHARSET.find(p)
            key_index = CHARSET.find(key[j])
            c = CHARSET[(p_index+offset(key_index))%n]
            ciphertext += c
        else: raise NotSupportedSymbol
    return ciphertext

def decrypt(ciphertext, key):
    return encrypt(ciphertext,key,lambda x: -x)
```

Challenge 4

- Let $\Sigma = \{\texttt{\textvisiblespace abcdefghijklmnopqrstuvwxyz}\}$
- Unknown key k (a character string over Σ)
- Decrypt: $C = \texttt{fxbxzktwrdvfcsfxbxqwplbn}$
- Plaintext language is English...

Dictionary attack to Vigenere ciphers

- Assumes the plaintext language is known (say, English)
- Brute force attack using English words as possible keys
- Can be improved by a clever guess of the key length
- For instance, there are less than 12K seven-character words in the american English collection in the *wamerican* Debian package
- Compare this with the $27^7 \approx 10^{10}$ seven-character strings over Σ above
- Uses a “language recognition” tool

Challenge 4: solution

```
from langdetect import detect_langs
for word in dictionary:
    if len(word) == keylen:
        key = word.lower()
        p = decrypt(ciphertext, key)
        languages = detect_langs(p)
        if len(languages)==1 and \
           languages[0].lang == 'en' and\
           languages[0].prob>=prob:
            tokens = p.split()
            for tok in tokens:
                if tok in dictionary:
                    plaintexts.append((p,key))
                    break
```

Challenge 4: solution

- In the ciphertext `fxbxzktwrdvfcsfxbxqwplbn` we see the four characters `fxbx` repeated at distance 14
- This suggests to try key lengths that divide 14 (say 2, 7 and 14)

```
./vigenereDictAttack.py fxbxzktwrdvfcsfxbxqwplbn 7

('computational␣complexity', 'ciphers')
('cftuof␣t␣vsvyzcftufrwike', 'cricket')
('uicsldokceqswnuicscpk␣no', 'lozenge')
('ucphqfbkxrfxyaucphhry␣ha', 'lumpier')
('ow␣vnfaeqbtuy␣ow␣verxual', 'rabbles')
```

(In)Security of Vigenère cipher

- Not so easy to break (as the above example might suggest) provided that:
  1. messages are short, compared to the key length
  2. keys are character sequences picked at random
  3. keys are not reused
- Requirements 1 and 3 above make it difficult (if not impossible) to apply frequency-analysis, even if the plaintext language is known
- Requirements 2 makes it impossible to apply classical dictionary attack
- However, Vigenère ciphers miss some important properties that are required to modern encryption algorithms, that must resist to high-speed computer attacks
- No more adopted in any significant application

Algoritmi di Crittografia

1. Administrivia

2. Introductory concepts
   - Encryption, decryption, and other introductory notions
   - Classical ciphers
   - Security, goals and attack models

General principles

- Ciphers are characterized by two main “ingredients”: a *permutation* and a *mode of operation*
- As is well-known, a permutation is an invertible function
- In classical ciphers, like Caesar and Vigenère’s, the permutations apply to single letters
- In modern ciphers, permutations usually apply to groups of m bits, for some positive integer m
- The mode of operation is the algorithm that incorporates the “logic” by which the permutations are applied to the plaintext

Good (secure) permutations

- Secure permutations must possess at least the following three fundamental properties
  1. The permutation depends on the key only, which is the sole piece of information that must kept (and is considered) secret
  2. Ideally, the permutation should appear as "randomly generated". Knowing that a letter (or bit sequence) x maps to y tells nothing about what $z \neq x$ maps to, except that cannot be y
  3. Different keys lead to different permutations.
- Concerning 2, recall that Vigenère permutations are completely determined by a single letter: if the letter at position i of the key is (say) `h`, then we know that `a` maps to `h`, `b` maps to `i`, and so on
- To fully understand why property 3 is important, we will offer an example concerning the Vigenère cipher

Keys and permutations

- In Vigenère, we know that each letter determines a different substitution in the plaintext
- Hence a key that is k letters long is just one element in a set of 26^k possible keys
- Suppose, as an extreme case, to consider only two substitutions
- E.g., letters `a-m` determine a shift of 3 and the letters `n-z` determine a shift of 7
- In this case the number of length-k keys is still 26^k but the set of combined permutations has only 2^k elements
- When trying to break the code, it is not important to recover the exact key since keys with same permutation “behave the same”
- Hence the search space may be significantly smaller

Mode of operation

- A secure permutation is by no means a “sufficient” condition for security
- For instance, if we apply the same (secure) permutation to all the letters, the ciphertext preserves all the “linguistic” properties of the plaintext
- In this case, if the plaintext is sufficiently long, frequency analysis is (certainly) sufficient but not even necessary to decrypt the message
- As an example, try to recover the English plaintext over $\Sigma = \{\texttt{␣abcdefghijklmnopqrstuvwxyz}\}$ from the following ciphertext:

```
#!-/&?\-/<@/%$|%<(>/#?/#!-/@-%$-%)
```

Mode of operation

- A good mode applies the permutations in a way that the ciphertext as a whole appears random (and not just the single permutations)
- Not only that! A good mode must prevent an attacker from learning anything on the ciphertexts (see the security models ahead)
- A good mode also endows the cipher with another important property, which is crucial to prevent some types of attacks, namely *diffusion*

A "quantitative" explanation for the insecurity of classical ciphers

- Classical cipher are ... classical, which implies they were invented before the advent of digital computers
- In turn, this means encryption and decryption were performed by hand ...
- ... which in turn imposed one of the two constraints below:
  1. only the "simplest" permutations could be employed, namely ones that can be "constructed" by using simple rules (recall the shifts of Caesar and Vigenère ciphers)
  2. if a number of complex (i.e., random looking) permutations were required, then they had to be "pre-computed" and stored in, e.g., look-up table
- Essentially, there was a time-space tradeoff but none of the alternatives was viable

A secure but impractical cipher: one-time pad

- This cipher requires: (1) that different messages use different keys; (2) that each key is randomly selected and has the same length as (or is longer than) the plaintext
- ... which immediately explains why the cipher is impractical
- Applies to bit sequences and is based on the exclusive or operation $\oplus$:

$$x \oplus y = \begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{if } x = y \end{cases}$$

  for $x, y \in \{0, 1\}$.
- Key property (for decryption):

$$(x \oplus y) \oplus x = x \oplus (y \oplus x) = y$$

- Key property (for security): for given y, if x is chosen uniformly at random in $\{0, 1\}$, then $\mathrm{prob}(x \oplus y = 1) = \mathrm{prob}(x \oplus y = 0) = \frac{1}{2}$

Example

- Exclusive or applies bitwise
- Encryption

key:	0	0	0	1	0	1	1	0	1	0
p:	1	0	1	1	0	0	1	0	0	1
c:	1	0	1	0	0	1	0	0	1	1

- Decryption:

key:	0	0	0	1	0	1	1	0	1	0
c:	1	0	1	0	0	1	0	0	1	1
p:	1	0	1	1	0	0	1	0	0	1

Security

- One-time pad is secure because (if the key is chosen uniformly at random) all the $0/1$ sequences are equiprobable
- However, if a key is used repeatedly, an attacker may learn a lot about the plaintexts
- Let

$$P_1 \oplus K = C_1, \qquad \text{and} \qquad P_2 \oplus K = C_2$$

  then

$$\begin{aligned} C_1 \oplus C_2 &= (P_1 \oplus K) \oplus (P_2 \oplus K) \\ &= (P_1 \oplus P_2) \oplus (K \oplus K) \\ &= (P_1 \oplus P_2) \oplus 0 \\ &= P_1 \oplus P_2 \end{aligned}$$

- Neither P_1 nor P_2 are directly exposed, but the knowledge of their xor allows to recover one if the other is known

Attack models and security goal

- What is an attack model?
  1. Tells what an attacker can (and cannot) do
  2. Provides a formal ground to assess the degree of success of a cipher to achieve a given *security goal*
  3. Helps users to consciously choose the cipher that best suits his/her needs
- Attack models and security goals provide the formal background for the work of cryptographers and crypto-analysts
- We consider here *black-box* models, so called because the attacker has access only to the input and output of the cipher and not the internals

Attack models

- *Ciphertext only* model (*COA*). This is what is popularly intended when talking of cipher (in)security
- Under this model, the ciphertext is the only piece of information that transits over the channel and that Eve can see
- The goal for $\mathcal{E}$ is just to recover the plaintext
- It is the hardest model for the attacker
- The downside is that security results under the ciphertext only model are the weakest possible
- *Known-Plaintext* model (*KPA*). Under this model, $\mathcal{E}$ can see plaintext/ciphertext pairs
- Her goal is to recover the key (recall the plaintext is known) in order to decrypt future messages

Attack models

- KPA is not as unreasonable a model as it might seem at first
- Actually, there are application settings where Eve can collect at least parts of plaintexts and the corresponding ciphertexts
- KPA is clearly a more powerful model than COA for the attacker and, consequently, security results under KPA are stronger
- *Chosen-plaintext* model (*CPA*). This is still stronger than KPA since here Eve can select plaintexts and see the corresponding ciphertexts
- As in KPA, Eve's goal is to recover the key. Here, however, Eve plays an *active* role
- Like KPA, CPA is not unreasonable. A realistic example is when Alice (the sender) will forward to Bob encrypted messages she receives by (a disguised) Eve in plain text
- CPA can always be done with public-key encryption schemes

Attack models

- *Chosen-ciphertext* model (*CCA*). Under this model, $\mathcal{E}$ can chose the plaintext/ciphertext and see the corresponding ciphertext/plaintext
- The goal is once more to recover the key, in general with the goal of redistribute/sell it, such in certain video-protection systems
- It is clearly the most powerful model for the attacker, and any strong encryption scheme should have no difficulty resisting to CCA
- There are clearly attacks for other (actually, any) types of cryptographic functions
- We will consider some of these when appropriate

Security goals

- Security goals go together with attack models
- With respect to encryption, the two main security goals are *indistinguishability* and *non-malleability*
- Indistinguishability for encryption schemes: the adversary cannot distinguish (w.h.p.) a pair of ciphertexts based on the message they encrypt
- A scheme is *IND-CPA* secure if exhibits indistinguishability under chosen plaintext attacks
- IND-CPA is regarded as the minimal level of security that must be provided by any encryption scheme

The IND-CPA game

- The adversary $\mathcal{E}$ is given a ciphertext C and she is told that C corresponds to one of two possible plaintexts, P_1 and P_2
- Although E may submit encryption queries, if the system is IND-CPA secure she is not able to tell whether $C = (P_1, K)$ or $C = (P_2, K)$ with a probability of success significantly greater than $1/2$.
- It is plain that, if the encryption scheme is deterministic, then all $\mathcal{E}$ has to do is simply to perform one query to get the correct answer
- Messages do change: from $C = E(K, M)$ to $C = \langle E(K, M, R), R \rangle$, where R is a sequence of random bits
- Yes, $\mathcal{E}$ is allowed to see the random bits, for otherwise ... (guess the implications)
- We shall see how to ensure IND-CPA security later

Non-malleability

- Has to do with *message integrity*, i.e., you must be sure that the message han not been forged
- Given a ciphertext C, corresponding to a plaintext P, $\mathcal{E}$ without the key is unable to produce a ciphertext C' whose corresponding plaintext is related to P in a significant way
- One-time pad is IND-CPA secure but not NM-CPA secure: given $C = D(P, K)$, it is easy to see that $C' = C \oplus 1 = D(P \oplus 1, K)$
- Some not-obvious implications:

  NM-CPA $\Rightarrow$ IND-CPA
  NM-CCA $\Leftrightarrow$ IND-CCA

Other attack models

- In gray-box models the box is ... gray! That means you can partially see inside (or aside)
- You can, e.g., measure execution time of a cipher, detect (or even induce) exceptions to occur, measure power-consumption, etc
- For some applications (i.e., when encryption is performed by a dedicated device you bought or rented) you can even “open” the encrypting device and perform mechanical/physical activities