­

**M11 – Seguretat Informàtica – UF1**

Pràctica 2 – SEGURETAT LÒGICA: FUNCIÓ HASH CRIPTOGRÀFICA

Nil Massó

**Table of Contents**

[Pràctica 2 – SEGURETAT LÒGICA: FUNCIÓ HASH CRIPTOGRÀFICA 3](#__RefHeading___Toc5320_2677517219)

[Pràctica 4](#__RefHeading___Toc5322_2677517219)

[md5sum 4](#__RefHeading___Toc5324_2677517219)

[sha256sum 5](#__RefHeading___Toc5326_2677517219)

[Exercici 1 (mostra les captures de pantalla i explicacions del procés) (1,5 punts) 5](#__RefHeading___Toc5328_2677517219)

[Exercici 2 (mostra les captures de pantalla i explicacions del procés) (1,5 punts) 6](#__RefHeading___Toc5330_2677517219)

[hashlib for Python 7](#__RefHeading___Toc5332_2677517219)

[Exercici 3 (mostra les captures de pantalla i justifica correctament el procés) (4 punts) 7](#__RefHeading___Toc5334_2677517219)

[Exercici 4: Xifratge i signatura digital (en parelles) (3 punts) 9](#__RefHeading___Toc5336_2677517219)

• Contesteu cada exercici raonant I posant-hi captures de pantalla quan es demani o considereu que es necessari.

• A l’hora d’avaluar i qualificar el treball es tindran en compte els aspectes estètics, de correctesa lingüística (sintàctica i ortogràfica) a més del que s’hagi comentat al cicle formatiu sobre la redacció de documentació tècnica i manuals.

• El mòdul professional pertany a uns estudis orientats al món laboral, cosa que fa que un cop complerts els requisits mínims la nota resultant serà condicionada per la quantitat i qualitat del treball individual realitzat per cada alumne.

# Pràctica 2 – SEGURETAT LÒGICA: FUNCIÓ HASH CRIPTOGRÀFICA

Una **funció de hash criptogràfica** és una classe especial de funció matemàtica també anomenada ***funció digest o funció resum***, que té certes propietats que el fan adequat per al seu ús en la criptografia. Aquest algoritme matemàtic:

• **mapeja dades de mida arbitraria a una cadena de bits d'una mida fixa** (funció resum) i

• està dissenyat per a ser també una funció d'un sol sentit, és a dir, una funció que és **impossible d'invertir**.

Les dades d'entrada es diu sovint el **missatge**, i la sortida (el valor de resum o hash) és sovint anomenat el **resum del missatge** o simplement el producte de digestió.

La funció hash criptogràfica ideal té quatre propietats principals:

• **Ràpid** de calcular el valor hash per a qualsevol missatge donat.

• **No** és factible per **generar el missatge a partir del seu valor hash**.

• Un **petit canvi** en un missatge ha **de canviar el valor del resum** de manera tan extensiva que el nou valor re-sum no pot aparèixer correlacionat amb l'antic hash.

• **No** és factible trobar **dos missatges diferents amb el mateix valor hash**.

La funció hash criptogràfica tenen **moltes aplicacions** de seguretat de la informació, sobretot en:

• Per a signatures digitals.

• Per a codis d'autenticació de missatges.

• Per altres formes d'autenticació.

• Per a les dades d'índex en les taules hash.

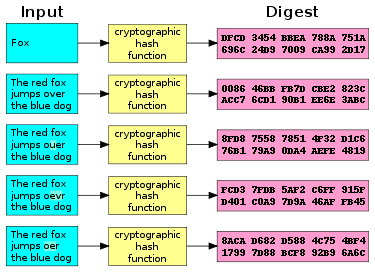
• Per *fingerprinting* (empremta dactilar, eina per tal de defensar els drets d'autor i combatre la pirateria).

• Per detectar dades duplicats.

• Per identificar de forma exclusiva els arxius.

• Per sumes de comprovació per detectar corrupció de dades accidental.

Figura 1: A una funció *hash* un petit canvi en l'entrada (a la paraula "over") canvia dràsticament la sortida (digest). Aquest és l'anomenat efecte allau.



# Pràctica

Aquesta activitat es pot fer des d'una màquina real o des d'una màquina virtual amb sistema operatiu ***Linux*** **Ubuntu Desktop** i accés a *Internet*.

## md5sum

Recordar que l'algoritme MD5 **ja no es considera segur**. Per tant, mentre que *md5sum* és molt adequat per a la identificació d'arxius coneguts en situacions que no estan relacionats amb la seguretat, *md5sum* no ha de ser invocat si hi ha una possibilitat que els arxius han estat intencionalment i maliciosament manipulat, en aquest últim cas, es recomana l'ús d'una eina de hash més recents, com ***sha256sum***.

## sha256sum

***sha1sum*** és un programa informàtic que calcula i verifica hash ***SHA-1***. S'acostuma a fer servir per a verificar la integritat dels arxius. Es troba instal·lat per defecte en la majoria dels sistemes operatius basats en Unix. Les variants inclouen ***shasum, sha224sum, sha256sum, sha384sum i sha512sum***, que fan servir una funció específica ***SHA-2*** de hash, i sha3sum (que permet ***SHA-3*** a través de ***SHA3-512***). També hi ha versions per a Microsoft Windows i la distribució Acti-vePerl inclou una implementació de Perl de shasum. En FreeBSD aquesta utilitat es diu sha512 i amb característiques addicionals.

Les variants ***SHA-1 es consideren vulnerables als atacs de col·lisió***, i els usuaris han d'utilitzar per exemple una variant SHA-2 com ara ***sha256sum*** en el seu lloc si es fa servir amb el propòsit de prevenir la manipulació d'arxius per part d'un adversari.

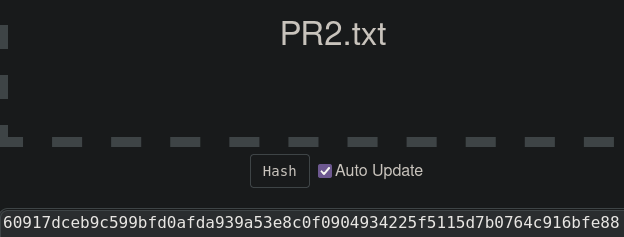
## Exercici 1 (mostra les captures de pantalla i explicacions del procés) (1,5 punts)

Selecciona qualsevol fitxer (un de text per exemple) i fes el seu resum (digest) mitjançant el programari sha256sum. Fes dues operatives:

1.1. Des de la línia d'ordres invoca el programa i aconsegueix el resum d’aquest fitxer.



1.2. Des d'una calculadora en línia *Checksum sha256* fes la mateixa operació. Pots trobar una al següent enllaç: <https://emn178.github.io/online-tools/sha256_checksum.html>



1.3. Compara el resum obtingut des de la línia d'ordres i la calculadora web.

Dona el mateix resultat ja que el metode per a crear el hash a estat el mateix

1.4. Què passa si modifiquem el contingut del fitxer?

Que el hash canviara completament

## Exercici 2 (mostra les captures de pantalla i explicacions del procés) (1,5 punts)

Ara provarem una altra aplicació del Hash: **identificar de forma exclusiva els arxius**.

Fes tot els passos per a assegurar la integritat de les dades i l'autenticitat de la descàrrega d'una ISO d'Ubuntu.

Tots els passos que s'han de fer es poden veure documentats en el següent enllaç: <http://www.ubuntu.com/download/how-to-verify>

Descarreguem la iso juntament amb el seu checksum i les signatures.

Primer executem aquesta comanda per verificar que els nostres arxius estan firmats per ubuntu:

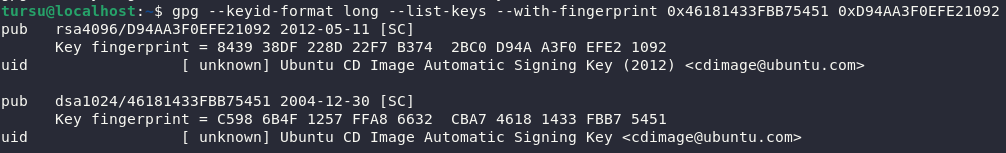
gpg --keyid-format long --verify SHA256SUMS.gpg SHA256SUMS

En cas de no estar ja en un ubuntu haurem de descarregar les claus publiques de ubuntu per verificarho, ho fem amb:

gpg --keyid-format long --keyserver hkp://keyserver.ubuntu.com --recv-keys 0x46181433FBB75451 0xD94AA3F0EFE21092

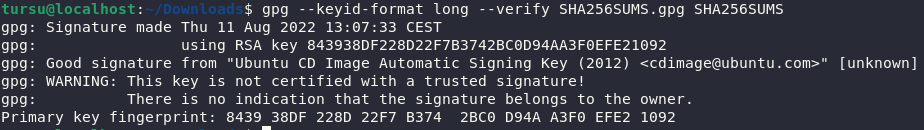
Ara s’hauran afegit clauer del nostre linux

Verifiquem que la emprempta de les claus sigui de ubuntu amb la seguen comanda:

gpg --keyid-format long --list-keys --with-fingerprint 0x46181433FBB75451 0xD94AA3F0EFE21092

Ara ja podem verificar que el checksum sigui legitim, ho fem amb:

gpg --keyid-format long --verify SHA256SUMS.gpg SHA256SUMS

Mirem que ens digui que la signatura es valida

Ara anem al directori on tinguem la ISO en el nostre cas Downloads:

cd ~/Downloads

I executem la comanda per a verificar el hash de la ISO



## hashlib for Python

***hashlib*** és una llibreria per a Python que es permet utilitzar diferents funcions de hash.

El següent codi codifica la paraula ‘hello’ utilitzant les funcions sha512, sha256 i sha1:

*import hashlib*

*paraula=”hello”*

*paraula8=paraula.encode(‘utf8’)*

*hexhash = hashlib.sha512(paraula8).hexdigest()*

*print (hexhash)*

*hexhash = hashlib.sha256(paraula8).hexdigest()*

*print (hexhash)*

*hexhash = hashlib.sha1(paraula8).hexdigest()*

*print (hexhash)*

Encara que les funcions de Hash intenten ser indesxifrables, amb diccionaris de paraules habituals, podem arribar a trobar ràpidament la paraula encriptada.

## Exercici 3 (mostra les captures de pantalla i justifica correctament el procés) (4 punts)

4.1. Amb Python, obté el codi SHA256 de les següents paraules:

• hello

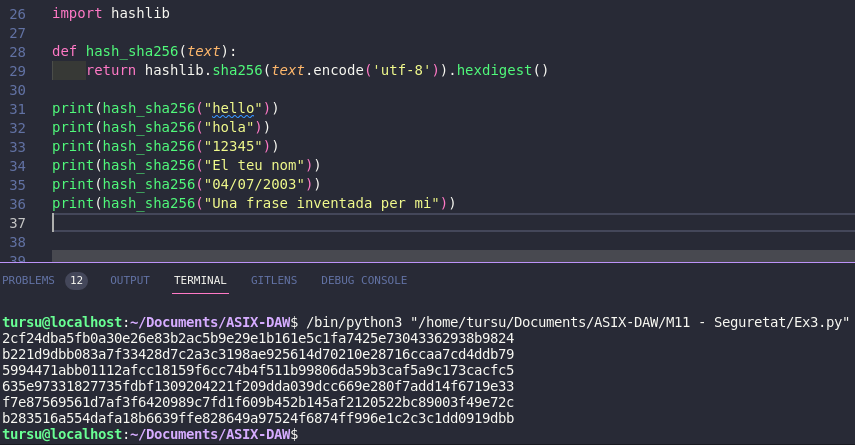
• hola

• 12345

• El teu nom

• La teva data de naixement en format de 6 dígits (2 per al dia, 2 per al mes, 2 per a l’any)

• Una frase



4.2. Utilitza el següent codi Python <https://github.com/Starwarsfan2099/Python-Hash-Cracker> per provar la força de les contrasenyes anteriors. Quan temps tarda en desxifrar cadascuna? N’hi ha alguna que no trobi?

Amb les que troba tarda manys de un segon, les que no estan a la llista de paraules no les pot crackejar

4.3. En el cas de la data de naixement, prova a rebentar-la mitjançant la opció numèrica del codi Python anterior. L’aconsegueix trobar? Quan temps ha necessitat?

4.4. En lloc del diccionari Wordlist.txt que ve per defecte, busca altres diccionaris que s’acostumin a utilitzar per rebentar contrasenyes. Fes una llista dels més habituals i prova almenys un. Quina diferència hi ha entre ells?

## Exercici 4: Xifratge i signatura digital (en parelles) (3 punts)

5.1. Utilitza l’eina GnuPG i la seva eina Kleopatra per a xifrar i desxifrar fitxers. Ho podeu provar amb [Linux](https://apps.kde.org/kleopatra/) o [Windows](https://gpg4win.org/download.html). **Realitza i documenta (tna el procés com el resultat) les tasques següents**:

• Crea’t una parella de claus (ja es creen les dues: pública i privada)

• Puja la teva clau al servidor i baixa't la del teu company. Si no et funciona la interfície del Kleopatra, ves directament a <https://keys.openpgp.org/>

• El servidor et servirà per intercanviar les claus. També pots fer servir altres mètodes (emmagatzemament extraible, correu electrònic,...)

• Xifra i signa un fitxer de text. **Ja saps quina clau has de fer servir.**

• Envia fitxers signats i xifrats al teu company.

• Desxifra els fitxers rebuts del teu company i comprova'n la signatura. **Ja saps quina clau has de fer servir**