Laboratorio 3.1 – Crittografia, hashing e firma dei file con GPG

1. Panoramica del laboratorio

1.1 Descrizione del laboratorio

In questo laboratorio imparerai a utilizzare **GNU Privacy Guard (GPG)** per proteggere i file con tecniche crittografiche. GPG è uno strumento open source ampiamente utilizzato per **la crittografia, l'hashing e la firma digitale**.

Imparerai come:

- Crittografare i file per proteggere la riservatezza.
- Generare e controllare gli hash per garantire l'integrità dei file.
- Firmare digitalmente i file per provarne l'autenticità e prevenirne la manomissione.

Queste competenze trovano applicazione diretta nella condivisione sicura dei file, nei backup e persino nella distribuzione di software.

1.2 Obiettivi di apprendimento

Dopo aver completato questo laboratorio, sarai in grado di:

- Installa e configura GPG sul tuo sistema.
- Genera e gestisci coppie di chiavi pubbliche/private.
- Crittografa i file per te e per gli altri.
- Utilizza l'hashing per verificare l'integrità dei file.
- Firma digitale dei file e convalida delle firme.
- Applica un flusso di lavoro completo di **Crittografia → Hash → Firma → Verifica**.

1.3 Prerequisiti

Prima di iniziare questo laboratorio, è necessario disporre di:

- Un computer con sistema operativo **Linux, macOS o Windows**.
- Accesso al terminale o alla riga di comando.

- Una conoscenza di base dell'uso dei comandi della shell.
- (Solo Windows) Il programma di installazione **Gpg4win**.
- (Facoltativo) Un editor di testo per la creazione di file di prova.

1.4 Tempo stimato per il completamento

Attività	**Tempo stimato**
Installazione e configurazione di GPG	45 minuti
Crittografia e decrittografia dei file	45 minuti
Hashing e verifica dell'integrità dei file	30 minuti
Firma e verifica dell'autenticità dei file	45 minuti
Sfida: flusso di lavoro Crittografia → Hash → Firma	60 minuti
Riflessione e documentazione	15 minuti
Tempo totale stimato	**~4 ore**

Punto di controllo
Prima di proseguire, assicurati di:

- Comprendi che questo laboratorio richiederà circa **4 ore**.
- Di disporre di un computer con accesso alla riga di comando.
- Sei pronto per installare GPG (o confermare che sia già installato).

2. Per iniziare

2.1 Che cos'è GPG?

GNU Privacy Guard (GPG) è un'implementazione gratuita e open source dello **standard OpenPGP**. Fornisce strumenti crittografici per:

- **Crittografia** → Rendere i file illeggibili senza la chiave privata corretta.
- **Hashing** → Generare un'impronta digitale per rilevare eventuali modifiche.
- **Firma** → Allegare una firma digitale che ne attesti la paternità e l'autenticità. GPG è

ampiamente utilizzato da sviluppatori, amministratori di sistema e professionisti della

sicurezza
, dalla protezione delle e-mail personali alla verifica dell'autenticità dei pacchetti software.

2.2 Perché utilizzare GPG per la crittografia, l'hashing e la firma?

GPG è potente perché combina **riservatezza, integrità e autenticità** in un unico strumento:

- **Riservatezza** → Crittografa i file in modo che solo il destinatario previsto possa leggerli.
- **Integrità** → Rileva la corruzione accidentale o la manomissione dolosa con gli hash.
- **Autenticità** → Verifica dell'identità del mittente tramite firme digitali.

Esempi reali:

- I download di software sono spesso accompagnati da **hash e firme**.
- I team utilizzano GPG per scambiare in modo sicuro file sensibili.
- I backup possono essere crittografati per impedire l'esposizione in caso di furto.

2.3 Lista di controllo per la configurazione del laboratorio

Prima di iniziare, assicurati di avere:

GPG installato

- Linux: preinstallato su molti sistemi; altrimenti:
  ```
  sudo apt install gnupg
  ```

- macOS:
  `brew install gnupg`
- Windows: Scaricare e installare **Gpg4win** dal sito ufficiale.

File di prova pronti

- Crea un semplice file di testo (ad esempio, secret.txt) per la crittografia, la firma e l'hashing.

Coppia di chiavi disponibile

- Generarne una se necessario:
  `gpg --full-generate-key`

Accesso al terminale/alla riga di comando

- Tutti i comandi di questo laboratorio saranno eseguiti dal terminale.

Punto di controllo

Prima di proseguire:

- GPG è installato sul tuo sistema?
- Hai generato una coppia di chiavi con gpg --full-generate-key?
- Hai almeno un file di testo pronto per il test?

3. Installazione e configurazione di GPG

3.1 Installazione di GPG

GPG è gratuito e disponibile su tutte le principali piattaforme:

- **Linux**
    - Spesso preinstallato. Se manca, installalo con:

      ```
      sudo apt update
      sudo apt install gnupg
      ```

- **macOS**
    - Installare tramite Homebrew:

      ```
      brew install gnupg
      ```

- **Windows**
    - Scarica **Gpg4win** da: https://gpg4win.org
    - Esegui il programma di installazione e segui la procedura guidata.

Suggerimento: conferma l'installazione con:

```
gpg --version
```

3.2 Generazione della prima coppia di chiavi

GPG utilizza una **coppia di chiavi pubblica/privata**:

- **La chiave pubblica** viene condivisa con altri utenti in modo che possano crittografare i file per te.
- **La chiave privata** viene mantenuta segreta e utilizzata per

decrittografare e firmare i file. Per generare una coppia di chiavi:

```
gpg --full-generate-key
```

Passaggi:

1. Scegliere **RSA e RSA** (impostazione predefinita).
2. Scegliere la dimensione della chiave → **40G6 bit** consigliati.
3. Imposta la scadenza → **2 anni** (può essere rinnovata in seguito).

4. Inserisci il tuo nome e indirizzo e-mail (fungono da ID chiave).
5. Imposta una passphrase complessa.

3.3 Porting e gestione delle chiavi

Una volta generate le chiavi, è possibile gestirle:

- **Elenca le tue chiavi**
  `gpg --list-keys`
- **Esportare la chiave pubblica (per condividerla)**
  `gpg --armor --exportyour@email.com > publickey.asc`
- **Esporta la tua chiave privata (solo come backup – conservala in un luogo sicuro!)**
  `gpg --armor --export-secret-keysyour@email.com > privatekey.asc`

Non condividere mai la tua chiave privata. Distribuisci solo **la** tua **chiave pubblica**.

3.4 Riepilogo: Nozioni fondamentali sulla configurazione di GPG

Attività	**Comando/Azione**
Installa GPG	sudo apt install gnupg (Linux) brew install gnupg (macOS) Gpg4win (Windows)
Genera coppia di chiavi	`gpg --full-generate-key`
Elenca le chiavi	`gpg --list-keys`
Esporta chiave pubblica	`gpg --armor --exportyou@example.com > publickey.asc`
Esporta chiave privata	`gpg --armor --export-secret-keys \ you@example.com > privatekey.asc`

Punto di controllo
Prima di proseguire:

- Hai installato GPG correttamente?
- Hai generato una coppia di chiavi con gpg --full-generate-key?
- È in grado di elencare le sue chiavi ed esportare la sua chiave pubblica?

4. Crittografia di un file

4.1 Crittografia per un destinatario

Per inviare un file in modo sicuro, è necessaria la **chiave pubblica del destinatario**. Una volta importata in GPG, eseguire:

```
gpg -e -rrecipient@example.com secret.txt
```

- -e → Crittografa
- -r → Email del destinatario (ID chiave)
- secret.txt → File da crittografare

Questo comando produce un file crittografato denominato

secret.txt.gpg. Solo il destinatario in possesso della chiave

privata può decrittografarlo.

4.2 Crittografia per sé stessi

Se desideri proteggere i tuoi file, basta crittografarli al **tuo indirizzo e-mail**:

```
gpg -e -ryou@example.com notes.txt
```

Questo è utile per **i backup** o i documenti sensibili che desideri che solo tu possa leggere.

4.3 Decrittografia di un file

Per decrittografare un file crittografato, eseguire:

```
gpg -d secret.txt.gpg
```

- Se il file è stato crittografato per te, GPG ti chiederà **la passphrase della** tua **chiave privata**.
- È possibile esportare il testo in chiaro in un file:
  `gpg -d secret.txt.gpg > secret_decrypted.txt`

Verifica sempre la crittografia decrittografando i tuoi file per assicurarti che le tue chiavi e la tua passphrase funzionino correttamente.

4.4 Riepilogo: crittografia dei file

Attività	Comando
Crittografa per il destinatario	`gpg -e -rrecipient@example.com file.txt`
Crittografare per sé stessi	`gpg -e -ryou@example.com file.txt`
Decrittografare un file	`gpg -d file.txt.gpg`

Punto di controllo

Prima di proseguire:

- Hai crittografato correttamente un file di prova?
- Hai decriptato il file riportandolo alla sua forma originale?
- Hai compreso il ruolo delle **chiavi pubbliche e private** nella crittografia?

5. Hash e verifica dei file

5.1 Generazione di un hash SHA-256

Una **funzione hash** produce un'impronta digitale unica di un file. Anche la più piccola modifica nel file crea un hash completamente diverso.

Per generare un hash SHA-256:

```
sha256sum importante.zip
```

Esempio di output:

```
d2d2c3f3b4a5e6c7d8f9a0b1c2d3e4f56789abcdeffedcba9876543210fedcba important.zip
```

Gli hash vengono comunemente pubblicati insieme ai download di software per consentire agli utenti di verificare i file.

5.2 Verifica dell'integrità dei file

Successivamente, è possibile ricalcolare l'hash e confrontarlo:

```
sha256sum important.zip
```

- Se l'hash è lo stesso → il file è invariato.
- Se è diverso → il file è stato modificato o danneggiato.

5.3 Perché l'hashing è importante

L'hashing viene utilizzato in molti scenari reali:

- **Integrità del software** → Verifica che i download non siano `stati` manomessi.
- **Backup** → Assicurarsi che i file archiviati rimangano invariati.
- **Analisi forense digitale** → Rileva se le prove sono state alterate.
- **Protocolli di sicurezza blockchain** → Gli hash sono alla base delle firme digitali e della prova di lavoro.

5.4 Riepilogo: Hashing dei file

Attività	Comando
Generare hash SHA-256	`sha256sum nomefile`
Verifica hash	Confronta i risultati in momenti diversi

Punto di controllo

Prima di proseguire:

- Hai generato un hash SHA-256 per uno dei tuoi file?
- Hai verificato che l'hash sia rimasto coerente quando il file non è stato modificato?
- Capisci come gli hash rilevano le manomissioni o i danneggiamenti?

6. Firma digitale dei file

6.1 Creazione di una firma allegata

Una firma allegata associa la firma al file stesso. Per firmare un file:

```
gpg --sign report.txt
```

- Questo crea un nuovo file report.txt.gpg.
- Esso contiene sia il file originale che la firma.

Utilizza le firme allegate quando desideri condividere il file e la sua prova di autenticità insieme.

6.2 Creazione di una firma separata

A volte non si desidera modificare il file stesso, ma è possibile creare una **firma separata**.

```
gpg --detach-sign report.txt
```

- Questo comando genera report.txt.sig (o .asc).
- Il file della firma è separato dal file originale.

Le firme separate sono comunemente utilizzate nella distribuzione di software (ad esempio, file .tar.gz con un .sig per la verifica).

6.3 Verifica di una firma

Per verificare che la firma di un file sia valida:

- Per una firma allegata:

  ```
  gpg --verify report.txt.gpg
  ```

- Per una firma separata:

  ```
  gpg --verify report.txt.sig report.txt
  ```

Se il file è stato modificato o viene utilizzata una chiave pubblica errata, GPG segnalerà che la firma non è valida.

6.4 Riepilogo: firma dei file e autenticità

Attività	Comando
Creare una firma allegata	`gpg --sign file.txt`
Creare una firma separata	`gpg --detach-sign file.txt`
Verifica della firma allegata	`gpg --verify file.txt.gpg`
Verifica firma separata	`gpg --verify file.txt.sig file.txt`

Punto di controllo
Prima di proseguire:

- Hai firmato un file con una firma allegata?
- Hai creato anche una firma separata?
- Hai verificato entrambe le firme con successo?

7. Sfida di laboratorio

Ora tocca a te applicare tutte e tre le tecniche crittografiche insieme: **crittografia, hashing e firma**.

7.1 Attività: crittografare → eseguire l'hash → firmare un file

1. **Crea un file di prova**

   ```
   echo "Questo è il mio file segreto di laboratorio." > labfile.txt
   ```

2. **Crittografalo utilizzando la tua chiave pubblica**

   ```
   # Produce labfile.txt.gpg:
   gpg -e -ryou@example.com labfile.txt
   ```

3. **Genera un hash del file originale**

   ```
   sha256sum labfile.txt > labfile.txt.sha256
   ```

4. **Firma il file con la tua chiave privata**

   ```
   # Produce labfile.txt.sig. gpg
   --detach-sign labfile.txt
   ```

5. **Verifica la firma e l'hash**

   ```
   gpg --verify labfile.txt.sig labfile.txt
   sha256sum -c labfile.txt.sha256
   ```

7.2 Istruzioni dettagliate

Passaggio	Comando	Risultato previsto
Crea file	`echo "testo" > labfile.txt`	File creato
Crittografare il file	`gpg -e -ryou@example.com labfile.txt`	labfile.txt.gpg generato
File hash	`sha256sum labfile.txt > labfile.txt.sha256`	Checksum SHA-256 salvato

Passaggio	Comando	Risultato previsto
Firma file	`gpg --detach-sign labfile.txt`	labfile.txt.sig generato
Verifica dei file	`gpg --verify labfile.txt.sig labfile.txt`	Messaggio firma valida
Verifica hash	`sha256sum -c labfile.txt.sha256`	Controllo integrità superato

7.3 Domande di riflessione

Scrivi le tue risposte nel tuo quaderno di laboratorio o nella tua consegna:

1. Qual è la differenza tra crittografia e firma?
2. Perché qualcuno potrebbe pubblicare **l'hash** di un file oltre alla **firma**?
3. Quale fase protegge **la riservatezza**?
4. Quale fase protegge **l'integrità**?
5. Quale fase protegge **l'autenticità**?
6. Come spiegheresti questo flusso di lavoro a un collega non esperto di tecnologia?

Sfida completata!
Hai applicato con successo l'intero ciclo **Crittografa → Hash → Firma → Verifica**. Questo è esattamente il modo in cui viene effettuata la distribuzione sicura dei file nel mondo reale.

8. Conclusioni e cosa c'è di nuovo

8.1 Punti chiave

In questo laboratorio hai messo in pratica tecniche crittografiche reali utilizzando **GPG** e strumenti correlati. Hai:

Installato e configurato GPG sul tuo sistema.
Generato una coppia di chiavi pubblica/privata.
Criptato file per te stesso o per un destinatario.
Verificato l'integrità dei file con **hash SHA-256**.
Hai firmato digitalmente i file (allegati e separati).
Hai verificato l'autenticità e l'integrità dei file
firmati.
Completato un flusso di lavoro completo **di crittografia → hash → firma → verifica**.

Queste sono le **stesse tecniche utilizzate nell'industria** per la distribuzione di software, i backup, le comunicazioni sicure e la digital forensics.

8.2 Competenze acquisite

Completando questo laboratorio, ora sei in grado di:

- Spiegare la differenza tra **crittografia, hash e firma**.
- Proteggi la riservatezza dei file con la crittografia.
- Rileva manomissioni o corruzioni con gli hash.
- Dimostra la paternità e l'autenticità con le firme digitali.
- Applica protezioni crittografiche combinate in un flusso di lavoro pratico.

Queste competenze costituiscono la base per comprendere il funzionamento dei protocolli di comunicazione sicuri e dei sistemi di fiducia.

8.3 Ne:t Lab: HTTPS e certificati nella pratica

Successivamente, vedrai come questi concetti si estendono oltre i file **nell'ambiente web**.

Nel **Lab 3.2: HTTPS e certificati nella pratica**, potrai:

- Ispezionare le connessioni HTTPS in un browser.
- Comprendere come **i certificati** stabiliscono la fiducia online.
- Esplorare il sistema **delle autorità di certificazione (CA)**.
- Scopri come l'hashing e la firma proteggono i siti web, proprio come hanno fatto con i tuoi file.

Passaggi successivi

- Salva i tuoi file .gpg, .sig e .sha256 come artefatti di laboratorio.
- Documenta le tue risposte di riflessione.
- Rivedi i comandi in modo da poterli ripetere con sicurezza.
- Preparati per **il Laboratorio 3.2 - HTTPS e certificati nella pratica.**

Ora sei passato dalla **crittografia dei file personali** alla preparazione per **la crittografia su scala web e i modelli di fiducia**.

6. Appendi:

6.1 Riferimento GPG Connand

Attività	Comando
Genera nuova coppia di chiavi	gpg --full-generate-
key Elenca le chiavi	gpg --list-keys
Esporta chiave pubblica	gpg --armor --exportyou@example.com > publickey.asc
Esporta chiave privata (solo backup)	gpg --armor --export-secret-keysyou@example.com >
solo)	privatekey.asc
Crittografare il file per il destinatario	gpg -e -
rrecipient@example.com file.txt Crittografare il file per sé stessi	gpg -e -
ryou@example.com file.txt	
Decrittografare il file	gpg -d file.txt.gpg
Firmare il file (allegato)	gpg --sign file.txt
Firma il file (separato)	gpg --detach-sign file.txt
Verifica firma allegata	gpg --verify file.txt.gpg
Verifica firma separata	gpg --verify file.txt.sig file.txt

6.2 Comandi di hashing Connon

Attività	Comando
Genera hash SHA-256	sha256sum nomefile
Verifica hash (rispetto al file)	sha256sum -c filename.sha256

6.3 Suggerimenti per la risoluzione dei problemi

Problema	Possibile causa	Soluzione
GPG non trovato	Non installato	Installare tramite il gestore dei pacchetti o Gpg4win
Generazione chiave non riuscita processi in background	Entropia debole (Linux)	Muovi il mouse, digita o esegui per generare casualità
Errore "Nessuna chiave pubblica" durante la crittografia	Chiave pubblica del destinatario non importata	Importare la chiave con gpg --import key.asc
Passphrase errata durante la decrittografia	Chiave privata o password errate	Riprovare con la passphrase corretta
Hash non corrispondente	File modificato o danneggiato	Scaricare nuovamente il file originale, confrontare di nuovo

6.4 Ulteriori risorse

- **Documentazione ufficiale GnuPG**: https://gnupg.org/documentation/
- **Gpg4win (Windows Installer)**: https://gpg4win.org
- **Tutorial SHA256SUM (Linux/Unix)**: https://linux.die.net/man/1/sha256sum
- **Crittografia pratica con GPG (libro/guida)**: https://www.debian.org/doc/manuals/securing-debian-manual/ch-crypto.en.html#gpg