GPG: GNU Privacy Guard // PGP – Pretty Good Privacy
RUBEN APUNTES
PGP --> Pretty Good Privacity (Software de xifrat de privacitat / seguretat).

GNU --> Va crear GPG (Software lliure).

Els següents individus (Alice i Bob), és volen communicar de manera segura, com ho fan?

Mitjançant **criptografía simètrica (secret compartit).**

- **Info:** La velocitat de *transformació* **és + ràpida** que la **'asmètrica'**, el cost computacional és - **petit** i les claus són + **petites**.
- El receptor com sap el **'secret'** pot rebre, l'inconvenient, és que el **'secret' és comparti**, llavors apareix...

Criptografía asimètrica (Cada usuari té una **clau privada** i una **clau pública**).

- **Info:** La velocitat de *transformació* és **+ lenta**, el cost computacional és **+ gran** i les claus són **+ grans**.
- Inconvenient: No té rendiment per poder processar en fluxe contínu el **tràfic** '**in-motion**'.

Encara això, **LES DOS CLAUS SÓN IGUAL DE SEGURES, S'UTILITZEN PER PROPÒSITS DIFERENTS.**

L'objectiu de la **clau privada** és que **NOMÉS** la pot tenir **EL PROPIETARI**.

L'objectiu de la clau pública és **PROPAGARLA**, quan més persones la coneguin, millor.

Les dues claus formen un '**joc**' **conjunt**, el que una **fà**, l'altra ho **desfà**.

Com pot fer l'Alice per enviar-li un missatge secret a Bob fent servir 'criptografía asimètrica'?

L'Alice enviarà el **missatge** + **xifra amb la clau pública** d'en Bob, i aquest ho rebrà amb la seva pròpia **clau privada** (aquesta només la te ell perquè és la SEVA clau privada)

Xifrar / desxifrar

Quan en Bob **Desxifri** + possi la seva **clau privada** llavors obtindrà el missatge original.

Signar:

- **Autoría** --> Comprovar que **l'autor** del document en qüestió és qui diu ser.
- **Integritat** --> Comprovar que el document no ha sigut **modificat**.
- **No repudi** --> No pots dir que **no** has sigut tú.

GPGs --> **criptografía simètrica, criptografía asimètrica, xifrar / desxifrar, signar**.

Com fa l'Alice per enviarun document signat al Bob? (Híbrid)

Agafa el missatge, i a aquest li aplica una **transformació** amb la seva **clau privada** i ho envía (perquè vol demostrar que és seu). → **Hash de la Clave Privada Emisor Cifrada → Envía mensaje.**

El Bob agafa el **missatge signat (CALCULA HASH)**, i li aplica la **clau pública (DESCIFRA)** de **l'Alice** (quan fa això pot extreure el missatge original). → **Obtiene el resumen original → Compara HASH.**

Criptografía híbrida --> El client i el servidor fan servir **'asimètric' per establir una comunicació** i per acordar un '**secret**' **compartit**, llavors el '**tub**' de **transmissió** de dades serà '**simètric**'

Hash / Funció Resum --> **MD5, SHA2**. Donat un **contingut**, genera un **resum** (aquest, és únic, si canviem alguna cosa, el HASH serà totalment diferent) (256 o 512 bits --> l'actual).

La probilitat de que el HASH generat sigui igual a un altre, és pràcticament imposible (99%).

S'utlitza molt en les operacions de **signatura**.

Donat un document (**missatge**), és genera el **HASH** del **missatge**, és **signa** amb la **clau privada** (és **transforma**) i **s'envía**. De manera que, l'Alice i en Bob, en lloc de **signar el missatge**, el missatge s'envía tal qual i és **signa el HASH**.
El **destinatari** rep el **missatge** i del **missatge** que rep, **calcula el HASH**. (Quan aplica la **clau pública** d'Alice sobre el missatge, obté el **HASH**, ademés és **demostra** que **és de l'Alice** si el **HASH** és el **mateix** que ha enviat **l'Alice)**.
PKI --> **Publick Key Infrastructure (model de seguretat)**.

GPG funciona en el model 'Web Of Trust' (Xarxa de confiança).

Web Of Trust --> Cada usuari ha de decidir en qui confía. **AARON**

Cripto Clave Simétrica: Más rapido, barato, ***compartido*** – Symmetric cipher – Misma clave para encriptar y desencriptar – Clave compartida (Problema secreto). Buenos flujos contínuos para envío (TCP). DES, 3DES, Aes..

Cripto Clave Asimétrica: Mayor coste (Lento) – Public Key cipher – **C Priv / C Pub** – Privada **secreto** / Pública **spread**. Para **CIFRAR** con la **C Púb** del **Recep**. Para **DESCIF** con la **C Priv** suya (**dest**). Para **FIRMAR** → **C Priv Emisor** y **RECEP** verifica la **FIRMA** con la **C Pub** del **Emisor**. RSA, DSA, Elgamal... No tiene flujo contínuo...

Cripto Clave Híbrida: ***Simétrica y asimétrica***. *Establece comunicación* (**ASIMÉTRICA**) – Envío de datos (***SIMÉTRICA***) → SECRETO COMPARTIDO (COMUNICACIÓN CIFRADA SIMÉTRICA) → **Dialogo / flujo contínuo TCP**.

El **cliente** y el **SERVER** usan **ASIMÉTRICO** para establecer un **SECRETO COMPARTIDO**. --> El tubo de transmisión de datos es **SIMÉTRICO**.

Se negocia una **clave principal de SESIÓN** y luego se negocia una **CLAVE** en que se **ENVIAN** los datos.

Tipos de Claves: PRIVADA → Secreto (RSA..) L'objectiu de la clau privada és que **NOMÉS** la pot tenir EL **PROPIETARI**./ **Pública** → Certificados. L'objectiu de la clau pública és **PROPAGARLA**, quan més persones la coneguin, millor. / **Certificados** → Avalada por entidad + sello. (Autoridad, Integridad + Verificación de legitimidad)

Formato ficheros: PEM – Formato ASCII Base64 (Cabe, pies) – DER → Binario.

Creación de Claves y Certificados: Claves → Certificados autofir // Petición a entidad certificación CA (Aval)

Modelo de confianza (Web of Trust): Emisor y receptor → Nivel de confianza // Depende del receptor si confía mucho o poco en algún emisor. El emisor debe avalar un certificado válido por alguna entidad CA. Web of TRUST → El usuario es responsable de confiar o no con otros Certificados. Si varias personas confían en una persona → Confías automáticamente. PKI → Public Key Infraestructure – Confianza emisora o CA.

Elemde confianza: CA selfsigned – CA top – CA Delegada – Cadena de confianza – **Llavero de claves** → Todas las claves se van acumulando (C Pub y C Priv) – HASH → Fingerprints → Resumen de contraseña (MD5).

Firma Híbrida: 1. Genera **HASH** – **2**. **Cifra C Priv Emisor** (**Resumen**). - **3**. Recep **calcula el HASH** y verifica. **4.** El recep **DESCIFRA** el **HASH** → **Descifra** con **C Pub** del **EMISOR** y obtiene el **RESUMEN ORIGINAL**. **5.** Los dos **HASH** deben ser **IGUALES.** * Garantiza la autenticación + integridad. → Sólo HASH.

Cifrado Híbrido: 1. Mensaje se cifra con una **clave SIMÉTRICA** - **2.** Se **Encripta la C Simétrica** con la **C Púb** del **Recep**. 3. La **clave Simétrica** → Pasa a ser **SECRETO COMPARTIDO. 4. Se envía el mensaje cifrado (simétrico) y la clave de cifrado (asimétrico). 5.** El recep **Desencripta** con su **C Privada. 6.** Una vez sabe cuál es el '**SECRETO COMPARTIDO**' puede pasar a **Descifrar** el mensaje usando esta **Clave Simétrica. →** Cripto Simétrica → más rápido – Asimétrica → Envío del **Secreto Compartido** , la clave que se usa realmente para **ENCRIPTAR → SE LLAMAN CLAVES DE SESIÓN.**

Ejemplo Simétrica: Emi → Cifra: Msg + C Priv Emi// Rece → Descifra: Msg + C Priv Emi (Share)

Ejemplo Asimétrica: Emi → Cifra: Msg + C Pub Rece // Rece → Descifra: Msg + C Priv Rece

COMANDOS GPG:

Gpg –gen-key	**Genera claves GPG**	gpg>sign
gpg-full-generate-key	**Genera claves GPG**	
Gpg –gen-revoke	**Borra claves GPG**	
Gpg –export [UID]	**Exporta claves GPG**	
Gpg –list-keys	**Muestra claves**	
Gpg –fingerprint	**Muestra fingerprint**	
Gpg –list-secret-keys	**Muestra Cl Privadas**	
Gpg –delete-key [UID]	**Borra claves**	
Gpg –delete-secret-key	**Borra Cl Privadas**	
Gpg –edit-key [UID]	**Edita**	
Gpg -u UID	**Selecciona clave**	
Gpg -r // --recipient	**CIFRA un fichero w clave (dest)**	
Gpg -e [FICHERO]	**Encripta fichero**	
Gpg -er [UID] [Fichero]	**Encripta fichero de alguna llave Pr**	
Gpg –armor // -a	**ASCII .pem**	
Gpg -a -er	**Con armor y recipient**	
Gpg -d // --decrypt	**Decripta salida**	
Gpg -d -o [Fichero]	**Decripta salida fichero**	
Gpg -s [Fichero]	**Firma con clave**	
Gpg –clearsign [Fichero]	**Firma formato ASCII**	
Gpg -b [Fichero]	**Detach sign. Binarios**	
Gpg -u [Remi] -r [Dest] –armor –sign –encrypt [fichero.pem]	**Encriptar con armor .pem y firma**	