Laboratorio de seguridad informática

Uso de Kleopatra con RSA

José Benito Camiña

Equipo: Los Vatos de Azcapo

Diego Monroy Fraustro	A01165792
Eduardo Azuri Gaytán Martínez	A01165988
Carlos Alejandro Reyna González	A01165824
Diego Galíndez Barreda	A01370815

Descripción de la actividad:

La actividad consiste en dos partes: primero se genera las llaves pública y privada con el método de RSA.
Después de esto, se importó la llave pública que envió el profesor, de forma que se pudieran encriptar archivos que solo él pueda leer.
Finalmente, se encriptó un archivo con dicha llave al igual que con la nuestra para poder tener acceso a este.

Generando las llaves:

Se generaron las llaves utilizando el comando `gpg --gen-key`[1] (Figura 1), el cual se encarga de pedir información necesaria para generar una "identidad" de la persona, de forma que se pueda relacionar su llave pública con ella.

Esto lo hace usando el nombre, correo y un comentario opcional, lo cual permite que tu llave pública sea fácil de localizar por otros al ser puesta en un servidor público. Se utilizó, además, un tamaño de 2048 para el algoritmo de RSA y se pidió que no expire.

Encriptando el archivo:

Se generó un archivo .docx conteniendo el nombre del equipo (no el hostname). Este fue encriptado con el comando `gpg --encrypt --sign --armor -r <mail> <file_name>`[2] (Figura 2) el cual recibe el correo de quien recibe el archivo, tras importar su llave.

Para importar las llaves se usó el comando `gpg --import <public key file>`[1] este nos permite importar la información de una llave pública desde un archivo, en este caso el .asc que nos envió el profesor con su llave pública (Figura 3).

Para poder comprobar que la encripción funcionó, también incluimos nuestra identidad, simplemente para poder desencriptar el archivo y asegurar que el proceso funcionó como debería (Figura 4).

Finalmente se hizo un "backup" de la llave pública para poder compartirla libremente. Esto se hizo con el comando `gpg --export --armor <mail> > <output_file>`[1](Figura 5).

Evidencias:

```
C:\Users\Diego\Documents\Seguridad>gpg --gen-key
gpg (GnuPG) 2.0.29; Copyright (C) 2015 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: diego
Email address: dgalindez1@gmail.com
Comment: Comment :)
Invalid character in comment
Comment: Comment
You selected this USER-ID:
    "diego (Comment) <dgalindez1@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: C:/Users/Diego/AppData/Roaming/gnupg/trustdb.gpg: trustdb created
gpg: key 27B55786 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
```

```
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   2048R/27B55786 2015-09-30
      Key fingerprint = C867 0E6D 8553 6070 BF11  9F1A 4DA9 BC33 27B5 5786
uid       [ultimate] diego (Comment) <dgalindez1@gmail.com>
sub   2048R/9DB82BDD 2015-09-30
```

Figura 1 - Generando la identidad y las llaves de RSA

```
C:\Users\Diego\Documents\Seguridad\RSA>gpg --encrypt --sign --armor -r jb.camina
@itesm.mx -r dgalindez1@gmail.com "RSA TOP-SECRET.docx"

You need a passphrase to unlock the secret key for
user: "diego (Comment) <dgalindez1@gmail.com>"
2048-bit RSA key, ID 27B55786, created 2015-09-30

gpg: 22BA9866: There is no assurance this key belongs to the named user

pub  2048R/22BA9866 2015-09-30 José Camiña <jb.camina@itesm.mx>
 Primary key fingerprint: 019B D5FE 5765 DFE5 FFB9  F1E2 8A74 BA44 FB03 3BAE
      Subkey fingerprint: 8504 4B00 3C93 1419 4EE8  E1A7 FF2D D68B 22BA 9866

It is NOT certain that the key belongs to the person named
in the user ID.  If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
File `RSA TOP-SECRET.docx.asc' exists. Overwrite? (y/N) y
```

Figura 2 - Cifrado de un documento con el correo relacionado a la llave pública importada

```
C:\Users\Diego\Documents\Seguridad\RSA>gpg --import 019BD5FE5765DFE5FFB9F1E28A74
BA44FB033BAE.asc
gpg: key FB033BAE: "José Camiña <jb.camina@itesm.mx>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
```

Figura 3 - Importado de una llave pública desde un archivo

```
C:\Users\Diego\Documents\Seguridad\RSA>gpg "RSA TOP-SECRET.docx.asc"

You need a passphrase to unlock the secret key for
user: "diego (Comment) <dgalindez1@gmail.com>"
2048-bit RSA key, ID 9DB82BDD, created 2015-09-30 (main key ID 27B55786)

gpg: encrypted with 2048-bit RSA key, ID 22BA9866, created 2015-09-30
      "José Camiña <jb.camina@itesm.mx>"
gpg: encrypted with 2048-bit RSA key, ID 9DB82BDD, created 2015-09-30
      "diego (Comment) <dgalindez1@gmail.com>"
gpg: Signature made  using RSA key ID 27B55786
gpg: Good signature from "diego (Comment) <dgalindez1@gmail.com>" [ultimate]
```

Figura 4 - Descifrado del documento

```
C:\Users\Diego\Documents\Seguridad>gpg --export --armor dgalindez1@gmail.com > v
atosDeAzcapo-pubkey.asc
```

Figura 5 - Exportando la llave pública

Conclusiones:

El uso de métodos de encriptación de llave pública y privada, como RSA, facilita mucho el poder compartir información privada de forma que no sea fácil desencriptar. También es útil porque permite verificar la identidad de quienes reciben los archivos, ya que tienen que usar su llave privada que está atada a su llave pública, la que fue usada para encriptar.

Este procedimiento resulta muy eficiente al contar con herramientas como GPG que hacen esto de forma automática, ya que quitan la complejidad del algoritmo y nos otorgan la misma fiabilidad.

Referencias:

1. Creating GPG Keys. (2014, April 30). Retrieved September 30, 2015, from https://fedoraproject.org/wiki/Creating_GPG_Keys
2. Ellingwood, J. (2013, October 4). How To Use GPG to Encrypt and Sign Messages on an Ubuntu 12.04 VPS. Retrieved September 30, 2015, from https://www.digitalocean.com/community/tutorials/how-to-use-gpg-to-encrypt-and-sign-messages-on-an-ubuntu-12-04-vps