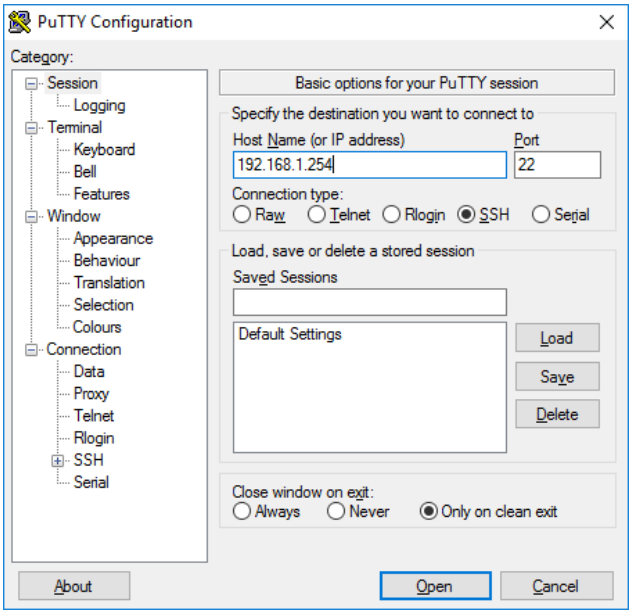
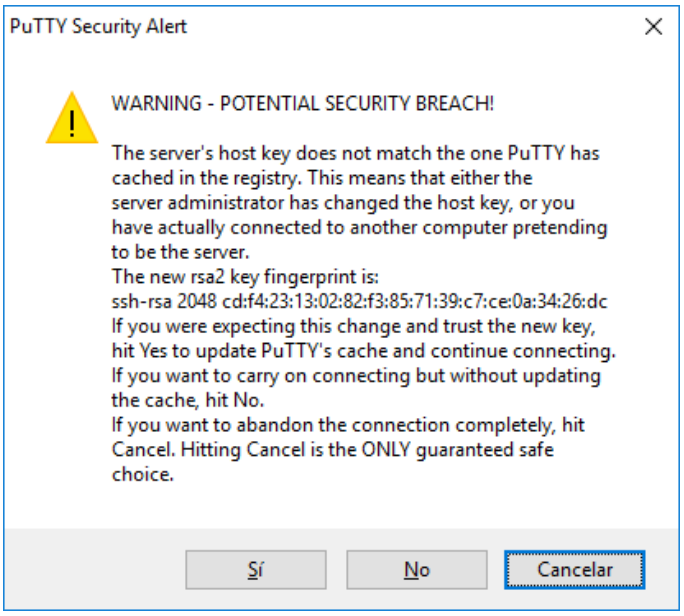
Acceso dende Microsoft Windows

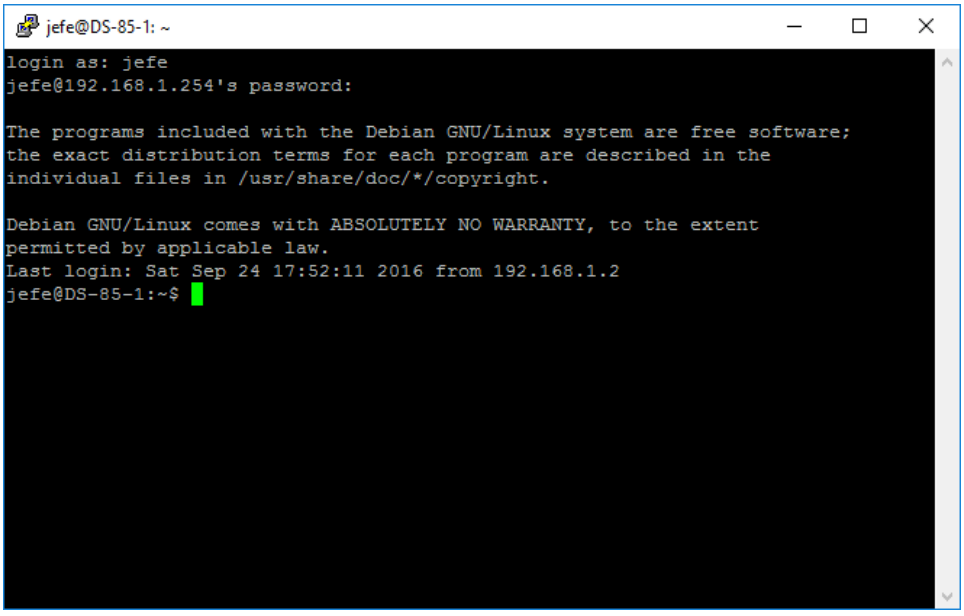
Putty é un dos clientes ssh para Microsoft Windows máis coñecidos. Na súa páxina web podemos descargar o programa ***putty.exe*** e outras utilidades para traballar con ssh, entre as que destacan:

- PuTTY: cliente ssh/telnet.
- PSCP: cliente scp (secure-copy).
- PSFTP: cliente sftp (ftp seguro).
- PuTTYgen: utilidade para xerar as claves de tipo RSA e DSA.

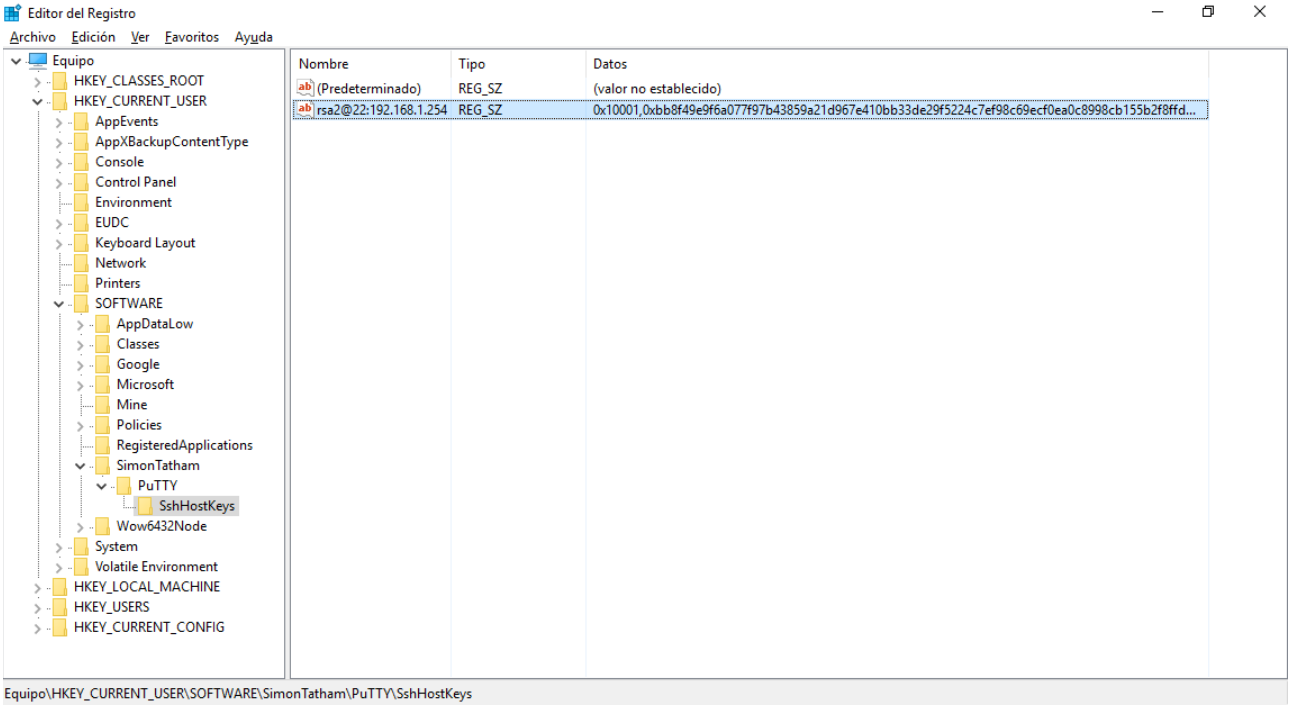
Igual que o cliente ssh en Linux, ao acceder por primeira vez con putty aparece a mensaxe relativa á clave pública do servidor:







O cliente putty garda as claves públicas dos servidores aos que se conecta nunha entrada do rexistro:



Igual que antes, cambiar as claves no servidor provocará un erro de autenticación. No caso do cliente putty, aparecerá unha mensaxe de erro e a opción de actualizar a clave pública do servidor. Tamén é posible eliminar a clave directamente no rexistro.

Outra ferramenta que podemos empregar para a conexión dende Windows (ou Linux) é a APP de `Chrome Secure Shell App`.

2.- Escenario B: ataques de contrasinal

O sistema de autenticación por contrasinal, a pesar de ser o máis empregado polos usuarios, non é o máis seguro.

- Empregando unha distribución de pentesting como Kali Linux, e as ferramentas medusa e hydra para lanzar un ataque de dicionario sobre o servizo ssh correndo nun equipo Linux, pode comprobarse que o uso deste tipo de ferramentas permite automatizar este tipo de ataques e ser
- Está claro que un ataque destas características deixa unha pegada bastante clara. Sen embargo, esta pegada tan clara pode pasar desapercibida no caso de que ninguén revise os logs do sistema. Se entramos na máquina atacada e revisamos o log /var/log/auth.log veremos que está cheo de liñas avisando de erros de autenticación:

```
$ less /var/log/auth.log
Sep 24 18:05:44 DS-85-1 sshd[767]: Failed password for jefe from 192.168.1.2 port 35898 ssh2
Sep 24 18:05:44 DS-85-1 sshd[767]: Connection closed by 192.168.1.2 [preauth]
Sep 24 18:05:44 DS-85-1 sshd[767]: PAM 2 more authentication failures; logname= uid=0 euid=0
tty=ssh ruser= rhost=192.168.1.2  user=jefe
Sep 24 18:05:47 DS-85-1 sshd[769]: pam_unix(sshd:auth): authentication failure; logname=
uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.2  user=jefe
Sep 24 18:05:49 DS-85-1 sshd[769]: Failed password for jefe from 192.168.1.2 port 35899 ssh2
Sep 24 18:05:52 DS-85-1 sshd[769]: Failed password for jefe from 192.168.1.2 port 35899 ssh2
Sep 24 18:05:55 DS-85-1 sshd[769]: Failed password for jefe from 192.168.1.2 port 35899 ssh2
Sep 24 18:05:55 DS-85-1 sshd[769]: Connection closed by 192.168.1.2 [preauth]
Sep 24 18:05:55 DS-85-1 sshd[769]: PAM 2 more authentication failures; logname= uid=0 euid=0
tty=ssh ruser= rhost=192.168.1.2  user=jefe
Sep 24 18:05:58 DS-85-1 sshd[771]: pam_unix(sshd:auth): authentication failure; logname=
uid=0 euid=0 tty=ssh ruser= rhost=192.168.1.2  user=jefe
Sep 24 18:06:00 DS-85-1 sshd[771]: Failed password for jefe from 192.168.1.2 port 35900 ssh2
Sep 24 18:06:02 DS-85-1 sshd[771]: Failed password for jefe from 192.168.1.2 port 35900 ssh2
Sep 24 18:06:05 DS-85-1 sshd[771]: Failed password for jefe from 192.168.1.2 port 35900 ssh2
```

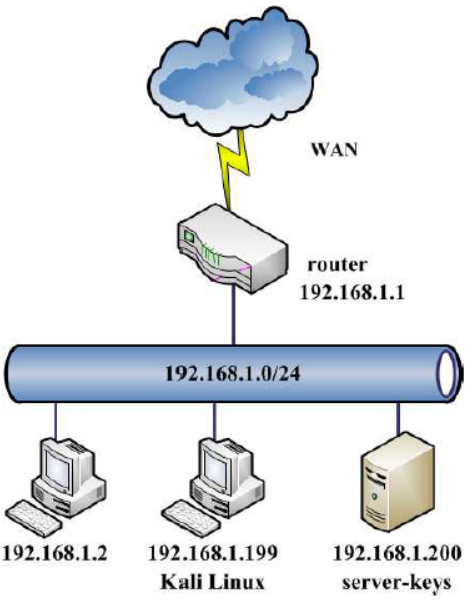
- Chegados este momento, un pode pensar que o escenario está ben, incluso que é espectacular; pero non deixa de ser iso, un espectáculo programado, onde todo está orquestado para funcionar como debe. Ademais, no mundo real:
  - **A miña organización usa *firewalls*.** Usa *firewalls*, con servidores separados en redes DMZ, regras verificadas dende o interior e o exterior da organización e limitando o acceso unicamente aos servizos que realmente necesitan. Isto pode levar a unha sensación de total seguridade e invulnerabilidade. Sensación errónea; xa que, se nun *firewall* permito o acceso ao servizo ssh, pasarán todas as comunicacións deste tipo. Pasarán tanto os intentos lexítimos de autenticación como as probas dun ataque de dicionario; ben é certo, que é posible limitar o número de conexións simultáneas, o número de conexións abertas nun determinado período de tempo e os equipos dende onde se pode conectar. Sen embargo, estas medidas ralentizarían o ataque, non o impedirían; xa que, os *Network Level Firewall* non analizan o contido dos paquetes a nivel de aplicación. Para iso, é preciso un *firewall* de nivel de aplicación e non sempre é posible usalos; por exemplo, para revisar os contidos de paquetes dunha conexión cifrada hai que descifrala previamente, o cal ás veces non é posible por motivos técnicos e/ou legais.
  - **A miña organización non usa conta de root.** Efectivamente, distribucións como Ubuntu teñen a conta de root desactivada e os traballos de administración fanse con `sudo`.

    Desactivar a conta de root e usar unha conta de superusuario de nome pouco habitual é un paso para protexernos, pero hai que pensar no seguinte:
    - A seguridade dunha conta protexida por contrasinal non está no nome de usuario, está no contrasinal. Usar nomes 'raros' facilita en certa maneira a seguridade (seguridade por escuridade), mentres ese nome non sexa coñecido.
    - Os ordenadores non son os únicos sistemas nunha organización e a gran maioría de dispositivos de internetworking (switches, routers, puntos de acceso wifi, ...), cámaras IP, impresoras de rede, ..., soen ter unha única conta de administración coñecida (figura nos manuais dos aparellos). Neste tipo de aparellos non só é coñecido o nome de usuario, tamén é o contrasinal; e para máis complicacións, na maioría dos casos non se pode cambiar o nome de usuario nin desactivalo.

3.- Escenario C: autenticación por claves

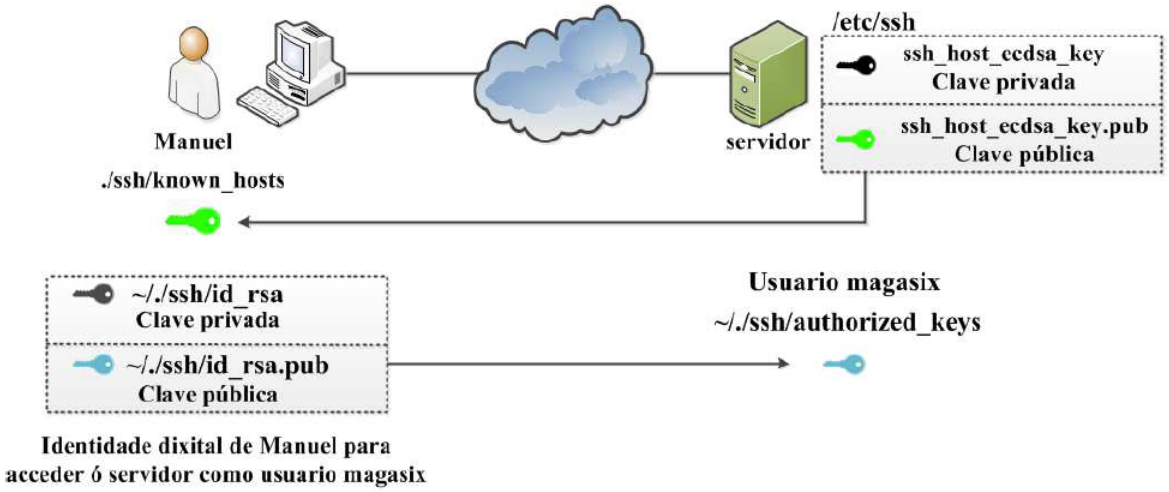
Este escenario ten como principal obxectivo explicar e implementar un sistema de autenticación por claves, que usando a criptografía de clave simétrica permita que os usuarios se autentiquen no sistema a través dunha identidade dixital.

Ao longo deste escenario crearemos varias identidades dixitais e configuraremos o servidor `ssh` para permitir unicamente o acceso mediante claves criptográficas (identidades dixitais). Unha vez configurado o sistema de autenticación, debería verse como un ataque de dicionario lanzado con medusa fracasaría **(esta parte da práctica non a imos a realizar)**.



1.2. Autenticación por claves

SSH permite autenticar aos usuarios facendo uso de identidades dixitais (formadas cunha parella de claves privada-pública) e os algoritmos criptográficos de clave simétrica. **O procedemento a seguir é crear a identidade dixital e copiar a clave pública no arquivo de claves autorizadas no *host* remoto.** Resúmese na seguinte imaxe.



Crear a identidade dixital

En primeiro lugar, o usuario creará a súa identidade dixital xerando a parella de claves privada-pública. Como xa sabemos, a clave privada debe manterse segura a toda costa e a pública pode distribuírse. A clave privada pode protexerse mediante o uso dunha *passphrase*; é dicir, que cando se use haberá que introducir a *passphrase*. Sinalar que cando se indica a *passphrase*, o que se está a escribir non é visible. No caso de esquecer a *passphrase* a clave privada non se poderá usar ao non poderse descifrar; xa que, o mesmo cifrado que fai seguro o sistema ssh fai irrecuperable a clave. A solución sería abandonar a clave, xerar unha nova e instalar a nova pública nos equipos remotos.

Está claro que protexer as claves privadas cunha *passphrase* é o máis seguro; sen embargo, o uso de claves privadas sen *passphrase* tamén ten a súa utilidade. Por exemplo, as claves privadas sen *passphrase* permiten o seu uso sen necesidade da intervención manual do usuario, o que é axeitado na execución automática de *scripts*.

Para crear a parella de claves empregarase o comando `ssh-keygen`, sendo as opcións máis salientables:

- `-t <tipo_clave>` permite especificar o tipo de clave (RSA, DSA, ECDSA, etc.).
- `-b <tamaño_clave>` permite especificar o tamaño de clave. Lembrar que estas claves serven unicamente para autenticar ao usuario; polo tanto, incrementar o seu tamaño non significará incrementar a carga de CPU cando se envíen datos por ssh.

- `-C "<comentario>"` permite engadir un comentario á clave para poder distinguila doutras.
- `-f <nome_ficheiro>` permite especificar a ruta e nome dos ficheiros das claves (é posible ter varias identidades dixitais).

A continuación creamos unha parella de claves no equipo local **192.168.1.2**, protexendo a clave privada cunha *passphrase*:

```
$ ssh-keygen -t rsa -b 4096 -C "Identidade jefe - ASO"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jefe/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jefe/.ssh/id_rsa.
Your public key has been saved in /home/jefe/.ssh/id_rsa.pub.
The key fingerprint is:
47:6f:a6:d9:53:af:c7:f9:ba:d3:2e:5f:52:e0:59:8b Identidade jefe - ASO
The key's randomart image is:
+---[RSA 4096]----+
|                 |
|                 |
|          .   . .|
|         . . ..+.|
|        S . +E+..|
|         . * . o |
|          o o .o+|
|             .o+=|
|              +O=|
+-----------------+
```

Se facemos un listado da carpeta `$HOME/.ssh` veremos que aparecen dous novos arquivos, `id_rsa` e `id_rsa.pub`. Esta parella de claves criptográficas formarán a identidade dixital do usuario. Fixarse nos diferentes permisos da clave privada `id_rsa` e a clave pública `id_rsa.pub`.

```
~/.ssh$ ls -lahF
total 16K
drwx------ 2 jefe jefe 4,0K sep 24 18:37 ./
drwxr-xr-x 3 jefe jefe 4,0K sep 24 18:37 ../
-rw------- 1 jefe jefe 3,3K sep 24 18:37 id_rsa
-rw-r--r-- 1 jefe jefe  747 sep 24 18:37 id_rsa.pub
```

Autorizar o acceso á identidade dixital

Para poder usar esta identidade dixital hai que autorizar o seu uso no equipo remoto; para facer isto, a clave pública debe copiarse no arquivo de claves autorizadas (`authorized_keys`) na conta que se teña no equipo remoto onde se quere acceder (conta **jefe** do equipo servidor **192.168.1.200**). Ollo, é moi importante salientar que é a clave pública unicamente a que debe copiarse; non a clave privada. Pode engadirse manualmente ao final do arquivo ou, se está permitido o acceso por `ssh` ao equipo, usar o comando **`ssh-copy-id` que se executará no equipo local 192.168.1.2:**

```
$ ssh-copy-id jefe@192.168.1.200
The authenticity of host '192.168.1.200 (192.168.1.200)' can't be established.
ECDSA key fingerprint is 28:39:b0:98:a1:08:ae:78:c4:50:bd:e1:aa:c4:b5:37.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
to install the new keys
jefe@192.168.1.200's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'jefe@192.168.1.200'"
and check to make sure that only the key(s) you wanted were added.
```

No equipo servidor 192.168.1.200, dentro da carpeta .ssh, que se atopa no directorio *home* do usuario **jefe** aparece o arquivo *authorized_keys* coas claves públicas das identidades dixitais autorizadas para acceder por `ssh` como usuario **jefe**:

```
$ ls -lahF .ssh/
total 24K
drwx------ 2 jefe jefe 4,0K sep 24 20:01 ./
drwxr-xr-x 3 jefe jefe 4,0K sep 24 18:37 ../
-rw------- 1 jefe jefe  747 sep 24 20:01 authorized_keys
-rw------- 1 jefe jefe 3,3K sep 24 19:57 id_rsa
-rw-r--r-- 1 jefe jefe  747 sep 24 19:57 id_rsa.pub
-rw-r--r-- 1 jefe jefe  222 sep 24 20:00 known_hosts
$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCzoHtJFjxLosaXj1/EsGPE8dw9CHXNHoBwTF5sHD2TOinxHpAha...
```

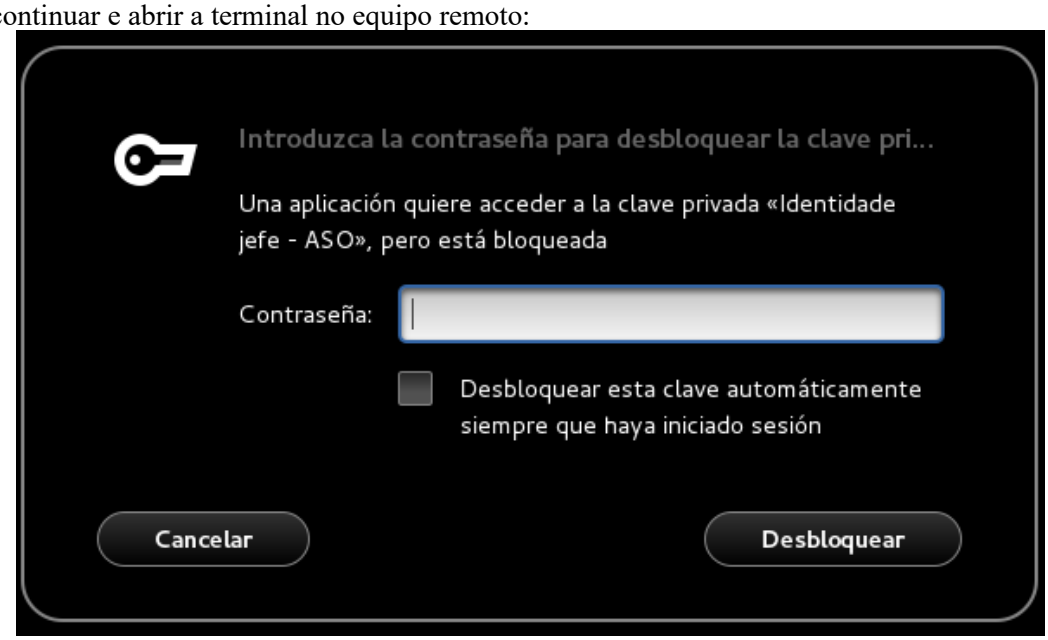
Unha vez copiada a clave pública é posible acceder usando esa identidade dixital sen necesidade de usar contrasinais:

```
$ ssh jefe@192.168.1.200
Enter passphrase for key '/home/jefe/.ssh/id_rsa':

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Sep 24 19:55:44 2016
```

Como a clave privada está protexida por unha *passphrase*, o usuario será preguntado pola mesma antes de continuar e abrir a terminal no equipo remoto:



Normalmente os sistemas Linux proporcionan axentes `ssh` que gardan en memoria as claves para non ter que teclear a *passphrase* cada vez que se use a clave (durante a sesión do usuario ou por un tempo configurado).

Saber que non estamos limitados a ter unha única identidade dixital.

A autenticación por claves é válida non só para obter a liña de comandos senón para traballar con `ssh` en modo comando e con `scp`.

```
$ ssh jefe@192.168.1.200 /sbin/ifconfig -a
eth0      Link encap:Ethernet  HWaddr 08:00:27:2c:c0:28
          inet addr:192.168.1.200  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe2c:c028/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1057 errors:0 dropped:0 overruns:0 frame:0
          TX packets:837 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:115444 (112.7 KiB)  TX bytes:100557 (98.2 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:391 errors:0 dropped:0 overruns:0 frame:0
          TX packets:391 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:48427 (47.2 KiB)  TX bytes:48427 (47.2 KiB)
```

Para ver un exemplo de utilización do comando `scp`, podemos baixar, por exemplo, o instalador de `webmin` dende o cliente, para logo envialo por `scp` ao servidor:

```
$ cd /tmp
$ wget http://prdownloads.sourceforge.net/webadmin/webmin-1.810.tar.gz
...
$ scp webmin-1.810.tar.gz jefe@192.168.1.200:
webmin-1.810.tar.gz                                   100%   27MB  27.3MB/s   00:01
```

Configurando o cliente `ssh`

Pode suceder que teñamos que acceder a múltiples equipos por `ssh`, posiblemente empregando diferentes usuarios, diferentes identidades dixitais, diferentes formas de autenticación, co servizo `ssh` correndo en diferentes portos,... Para facilitar as cousas existen clientes `ssh` como PacManager en Linux ou *Putty* en Windows que permiten crear perfís para cada conexión. Tamén é posible configurar estes perfís na nosa conta de usuario a través do arquivo ~/.ssh/config.

Vexamos un exemplo:

```
$ nano .ssh/config
Host jefe_DS1
        User jefe
        Hostname 192.168.1.200
        Port 22
        PubkeyAuthentication yes
        IdentityFile ~/.ssh/id_rsa
Host admin1_DS1
        User admin1
        Hostname 192.168.1.200
        Port 22
        PubkeyAuthentication yes
        IdentityFile ~/.ssh/id_rsa_admin1
Host admin2_DS1
        User admin2
        Hostname 192.168.1.200
        Port 22
        PubkeyAuthentication no
Host FW
        User jefe
        Hostname 192.168.1.254
        Port 25000
        PubkeyAuthentication no
```

Creamos así catro perfís coas seguintes características:

- **Perfil jefe_DS1:** para acceder ao equipo 192.168.1.200 como usuario jefe empregando autenticación por claves e a identidade dixital id_rsa.
- **Perfil admin1_DS1:** para acceder ao equipo 192.168.1.200 como usuario admin1 empregando autenticación por claves e a identidade dixital id_rsa_admin1.
- **Perfil admin2_DS1:** para acceder ao equipo 192.168.1.200 como usuario admin2 empregando autenticación por contrasinal.
- **Perfil FW:** para acceder ao equipo 192.168.1.254 polo porto tcp/25000 como usuario **jefe** empregando autenticación por contrasinal.

Dende o equipo de traballo e para conectarnos ao equipo remoto empregando o perfil, escribimos, por exemplo:

```
$ ssh jefe_DS1
```

Có comando `man ssh_config` pódese acceder á documentación de axuda dos arquivos de configuración do cliente openssh.

Práctica realizada coa axuda de Manuel González Regal – IES Xulián Magariños – Negreira.