SSH Public Key Authentication on Cisco IOS

Lesson Contents

PKI (Public Key Authentication) is an authentication method that uses a key pair for authentication instead of a password. Two keys are generated:

- Public key
- Private key

Anyone (or any device) that has the public key is able to **encrypt data that can only be decrypted by the private key**. This means you can share the public key with anyone you want, and they will be able to send you encrypted messages. The private key **has to be protected**...make sure it doesn't leave your computer.

In this lesson, we will generate a public and private key on a Windows and Linux computer. We will then add the public key to a Cisco IOS router and use it for SSH authentication. The router will send us encrypted messages, that only we can decrypt because we have the private key. This proves that we are the user that we claim we are, which allows access to the router.

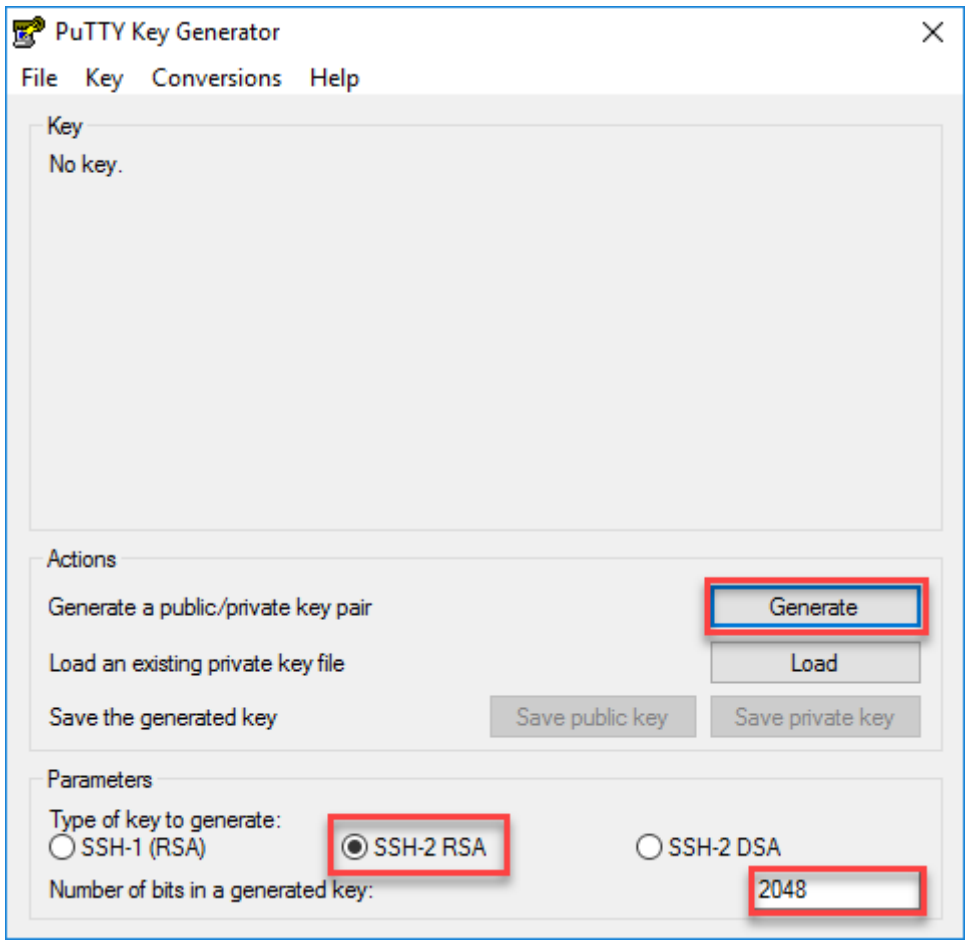
1. Configuration

First, we have to generate an RSA public / private keypair. I will show you how to do this on Windows and Linux.

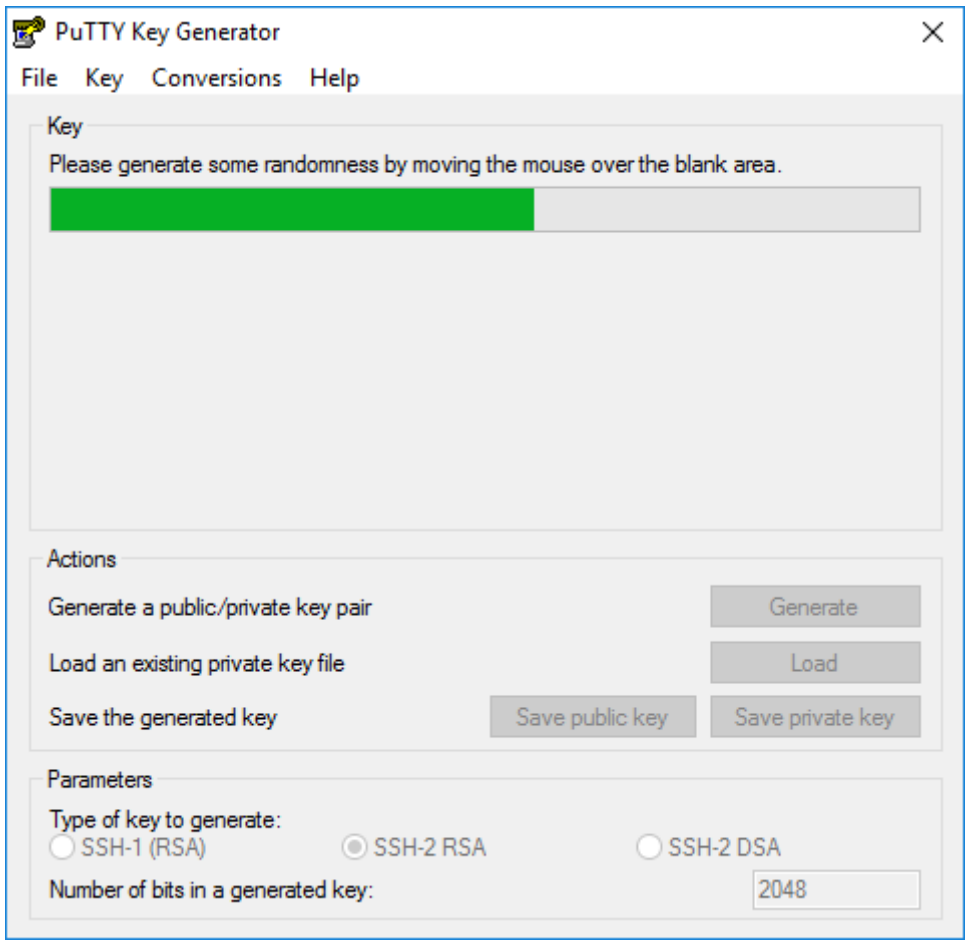
1.1. Windows

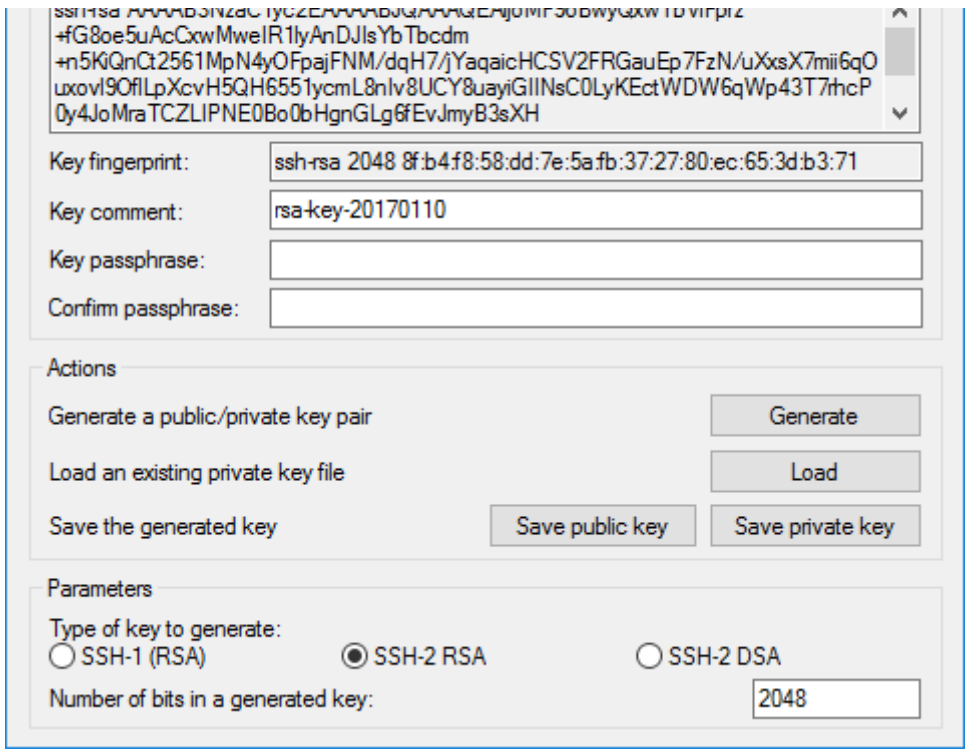
PuTTY is the most common choice as a SSH client on Windows so that's what we will use. Putty itself can't generate any RSA keys but we can do this with PuTTYgen (PuTTY Key Generator).

Once you start it, you will see the main screen:



The default settings are fine, we will generate a 2048 bit RSA keypair. Hit the generate button and you will see this:

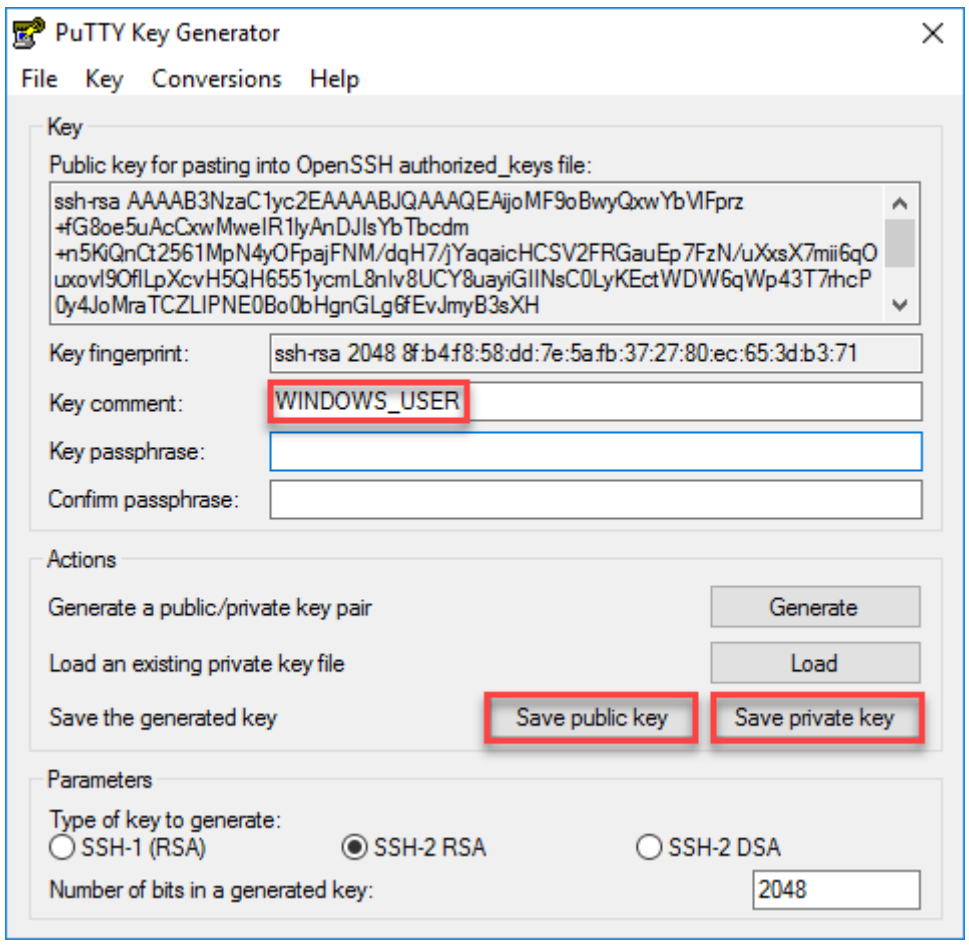




Once you close the application, your keys are gone so make sure to save them. Optionally, you can add a key comment.

You have to protect your private key carefully, it should never leave your computer. It is advised to set a key passphrase to protect it, to keep it simple, I won't do it in this lesson.

Hit the **save public key** and **save private key** buttons:



I will use the following filenames:

- public key: windows_user.pub
- private key: windows_user.ppk

That's all we have to do. Before we can test this, we have to configure our Cisco IOS router (or switch) first.

1.2. Linux

Most Linux distributions come with SSH, I will use Ubuntu for this example.

First, we have to generate a 2048 bit RSA keypair:

```
$ ssh-keygen -b 2048 -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Created directory
'https://cdn.networklessons.com/home/ubuntu/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
39:97:0c:ab:33:ea:bb:8b:e3:9f:4f:db:9a:fe:cf:fe ubuntu@HOST1
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|        .        |
|         = .     |
|        S +      |
|       . o       |
|      =          |
|  .. + * .       |
| .o+O**ooo+.E    |
+-----------------+
```

The filenames are:

- public key: id_rsa.pub
- private key: id_rsa

You can see them here:

```
$ ls -lh /home/ubuntu/.ssh
total 8,0K
-rw------- 1 ubuntu ubuntu 1,7K jan 10 19:41 id_rsa
-rw-r--r-- 1 ubuntu ubuntu  394 jan 10 19:41 id_rsa.pub
```

That’s all we have to do for now. Time to configure the Cisco IOS router / switch.

And a domain name:

```
R1(config)#ip domain-name NETWORKLESSONS.LOCAL
```

Let's generate a 2048 bit RSA key pair:

```
R1(config)#crypto key generate rsa modulus 2048
The name for the keys will be: R1.NETWORKLESSONS.LOCAL

% The key modulus size is 2048 bits
% Generating 2048 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 24 seconds)

%SSH-5-ENABLED: SSH 1.99 has been enabled
```

And enable SSH version 2:

```
R1(config)#ip ssh version 2
```

And configure the VTY lines to accept only SSH and local authentication:

```
R1(config)#line vty 0 4
R1(config-line)#transport input ssh
R1(config-line)#login local
```

Optionally, you can configure the router to disable SSH password authentication:

```
R1(config)#no ip ssh server authenticate user password
R1(config)#no ip ssh server authenticate user keyboard
```

Now we can import the public keys from our windows and Linux users.

1.3.1. Windows

You can open the public key file (windows_user.pub) in your favorite text editor. It will look like this:


```
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "WINDOWS_USER"
AAAAB3NzaC1yc2EAAAABJQAAAQEAijoMF9oBwyQxwYbVlFprz+fG8oe5uAcCxwMw
eIR1lyAnDJIsYbTbcdm+n5KiQnCt2561MpN4yOFpajFNM/dqH7/jYaqaicHCSV2F
RGauEp7FzN/uXxsX7mii6qOuxovl9OflLpXcvH5QH6551ycmL8nIv8UCY8uayiGI
INsC0LyKEctWDW6qWp43T7rhcP0y4JoMraTCZLIPNE0Bo0bHgnGLg6fEvJmyB3sX
H+7BaxHdYKg2OcIgVqYzclWhDwxj32kqd1BCq089iBMrb4QppDU2eM/t22iK29mn
eqOGTiCkxB80ix+KULT9okmqkj3TbhCpunTfuPCCRNrjqndBsw==
---- END SSH2 PUBLIC KEY ----
```

```
INsC0LyKEctWDW6qWp43T7rhcP0y4JoMraTCZLIPNE0Bo0bHgnGLg6fEvJmyB3sX
H+7BaxHdYKg2OcIgVqYzclWhDwxj32kqd1BCq089iBMrb4QppDU2eM/t22iK29mn
eqOGTiCkxB80ix+KULT9okmqkj3TbhCpunTfuPCCRNrjqndBsw==
```

We can add the public key for a username we choose. I'll call this user "WINDOWS_USER". Once you enter the **key-string** command, you can keep adding lines until you type **exit**:

```
R1(config)#ip ssh pubkey-chain
R1(conf-ssh-pubkey)#username WINDOWS_USER
R1(conf-ssh-pubkey-user)#key-string
R1(conf-ssh-pubkey-
data)#AAAAB3NzaC1yc2EAAAABJQAAAQEAijoMF9oBwyQxwYbVlFprz+fG8oe5uAcC
xwMw
R1(conf-ssh-pubkey-
data)#eIR1lyAnDJIsYbTbcdm+n5KiQnCt2561MpN4yOFpajFNM/dqH7/jYaqaicHC
SV2F
R1(conf-ssh-pubkey-
data)#RGauEp7FzN/uXxsX7mii6qOuxovl9OflLpXcvH5QH6551ycmL8nIv8UCY8ua
yiGI
R1(conf-ssh-pubkey-
data)#INsC0LyKEctWDW6qWp43T7rhcP0y4JoMraTCZLIPNE0Bo0bHgnGLg6fEvJmy
B3sX
R1(conf-ssh-pubkey-
data)#H+7BaxHdYKg2OcIgVqYzclWhDwxj32kqd1BCq089iBMrb4QppDU2eM/t22iK
29mn
R1(conf-ssh-pubkey-
data)#eqOGTiCkxB80ix+KULT9okmqkj3TbhCpunTfuPCCRNrjqndBsw==
R1(conf-ssh-pubkey-data)#exit
R1(conf-ssh-pubkey-user)#exit
R1(conf-ssh-pubkey)#exit
```

Our router now knows the public key of our windows users.

> It is possible to add multiple public keys for a single username. This allows you to have multiple users access the router with the same username but with different keypairs.

1.4. Linux

Let's do the same thing for our Linux user. Let's take a look at the public key:

```
ge+Rw7zn+O0i1Ib95djzNfVdHq+174mchGx3zV6l/6EXvc7G7MyXj89ffLdXIp/Xy/
wdWkc1P9Ei8feFBVLTWijXiilbYWwdLhrk7L2EQv5x ubuntu@HOST
```

The key is printed on a single line, that's fine but Cisco IOS only supports a maximum of 254 characters on a single line so you won't be able to paste this in one go. There's a useful Linux command you can use to break the public key in multiple parts:

```
$ fold -b -w 72 /home/ubuntu/.ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC80DsOF4nkk15V0V2U7r4Q2MyAwIbgQX/7
rqdUyNCTulliYZWdxnQHaI0WpvcEHQTrSXCauFOBqUrLZglI2VExOgu0TmmWCajW/v
np8J5b
ArzwIk83ct35IHFozPtl3Rj79U58HwMlJ2JhBTkyTrZYRmsP+r9VF7pYMVcuKgFS+g
Dvhbux
M8DNLmS1+eHDw9DNHYBA+dIaEIC+ozxDV7kF6wKOx59E/Ni2/dT9TJ5Qge+Rw7zn+O
0i1Ib9
5djzNfVdHq+174mchGx3zV6l/6EXvc7G7MyXj89ffLdXIp/Xy/wdWkc1P9Ei8feFBV
LTWijX
iilbYWwdLhrk7L2EQv5x ubuntu@HOST1
```

We can remove the "ssh-rsa" part at the beginning and the comment at the end. This is just the public key:

```
AAAAB3NzaC1yc2EAAAADAQABAAABAQC80DsOF4nkk15V0V2U7r4Q2MyAwIbgQX/7
rqdUyNCTulliYZWdxnQHaI0WpvcEHQTrSXCauFOBqUrLZglI2VExOgu0TmmWCajW/v
np8J5b
ArzwIk83ct35IHFozPtl3Rj79U58HwMlJ2JhBTkyTrZYRmsP+r9VF7pYMVcuKgFS+g
Dvhbux
M8DNLmS1+eHDw9DNHYBA+dIaEIC+ozxDV7kF6wKOx59E/Ni2/dT9TJ5Qge+Rw7zn+O
0i1Ib9
5djzNfVdHq+174mchGx3zV6l/6EXvc7G7MyXj89ffLdXIp/Xy/wdWkc1P9Ei8feFBV
LTWijX
iilbYWwdLhrk7L2EQv5x
```

Let's add it to the router, I will use the username "LINUX_USER":

```
R1(conf-ssh-pubkey-
data)#rqdUyNCTulliYZWdxnQHaI0WpvcEHQTrSXCauFOBqUrLZglI2VExOgu0TmmW
CajW/vnp8J5b
R1(conf-ssh-pubkey-
data)#ArzwIk83ct35IHFozPtl3Rj79U58HwMlJ2JhBTkyTrZYRmsP+r9VF7pYMVcu
KgFS+gDvhbux
R1(conf-ssh-pubkey-
data)#M8DNLmS1+eHDw9DNHYBA+dIaEIC+ozxDV7kF6wKOx59E/Ni2/dT9TJ5Qge+R
w7zn+O0i1Ib9
R1(conf-ssh-pubkey-
data)#5djzNfVdHq+174mchGx3zV6l/6EXvc7G7MyXj89ffLdXIp/Xy/wdWkc1P9Ei
8feFBVLTWijX
R1(conf-ssh-pubkey-data)#iilbYWwdLhrk7L2EQv5x
R1(conf-ssh-pubkey-data)#exit
R1(conf-ssh-pubkey-user)#exit
R1(conf-ssh-pubkey)#exit
```

Our work is finished, let's figure out if it works or not.

2. Verification

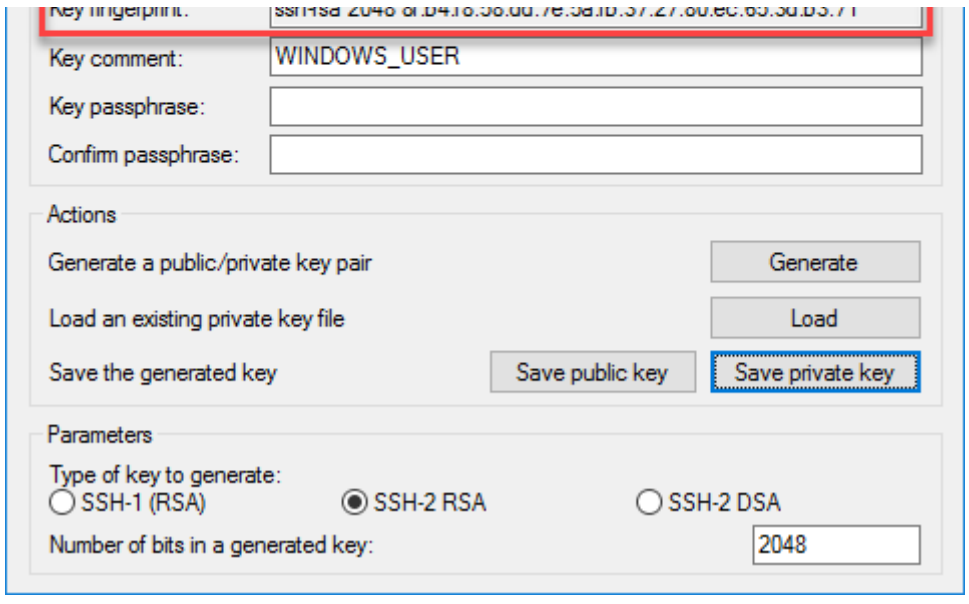
We will try our Windows user first, the the Linux user.

2.1. Windows

Once you added the public key to the router, Cisco IOS will calculate a key hash:

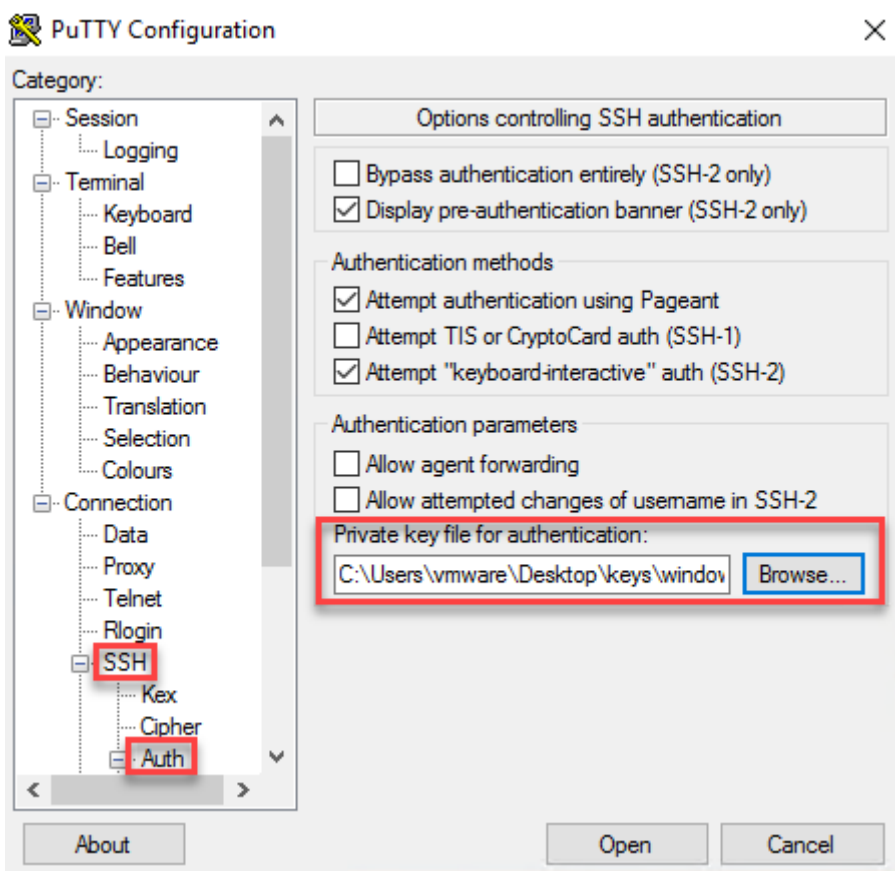
```
R1#show running-config | begin pubkey
ip ssh pubkey-chain
  username WINDOWS_USER
   key-hash ssh-rsa 8FB4F858DD7E5AFB372780EC653DB371
  quit
```

We can verify if it's the same, the PuTTY Key Generator also shows it:

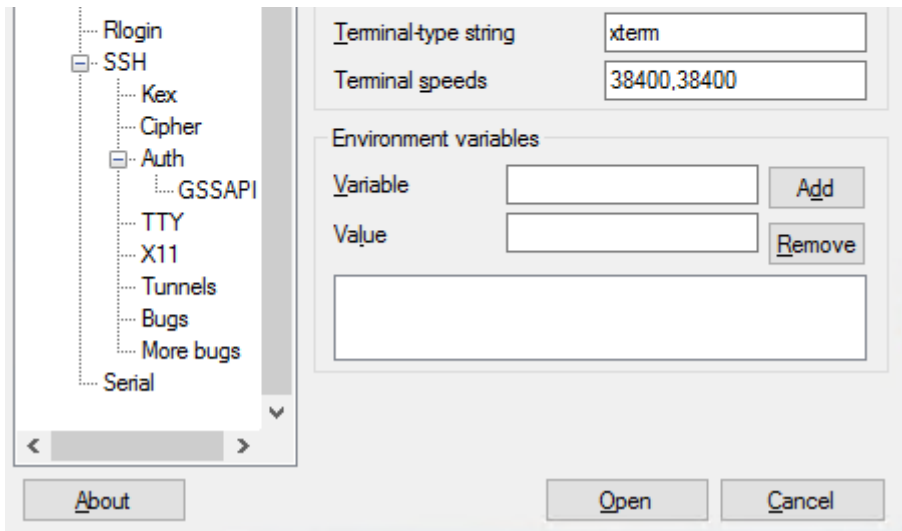


The key hash (fingerprint) matches so at least we know our router has the correct public key. Time to configure PuTTY. Open PuTTY and look for the Connection > SSH setting.

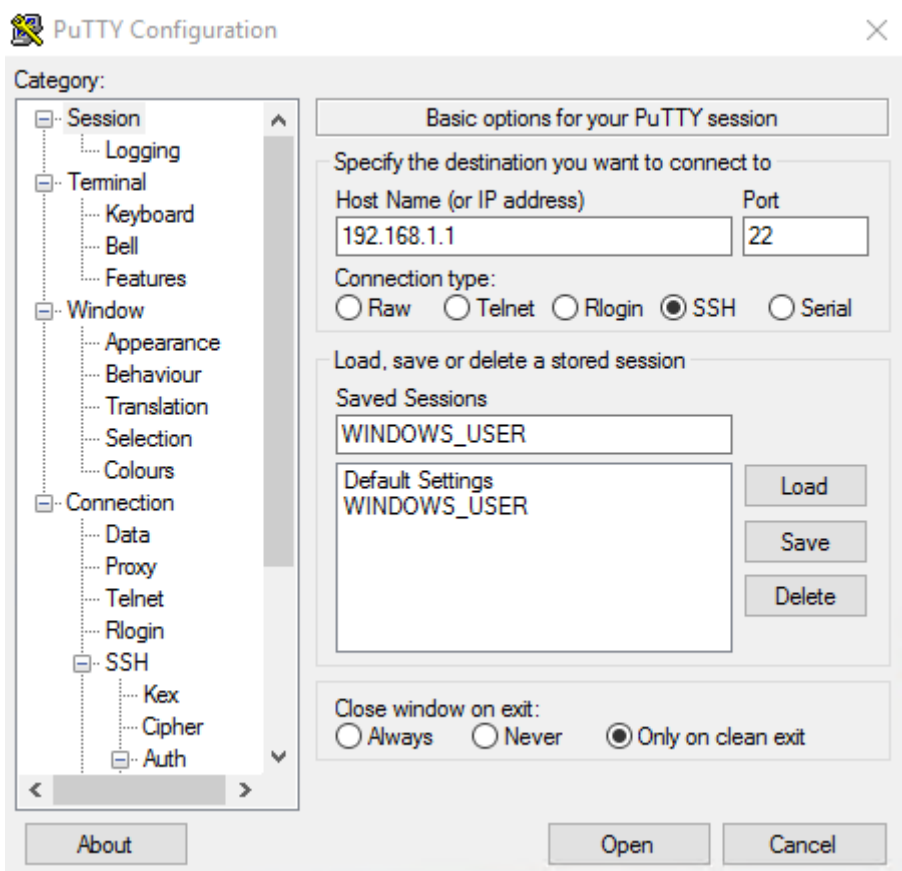
Click on the browse button and select your private key file (windows_user.ppk):



Now go to the Connection > Data setting, add the username here:



Go to the main screen and if you don't want to lose these settings, save your session.



Hit the Open button and you will see this:

```
Using username "WINDOWS_USER".
Authenticating with public key "WINDOWS_USER"

R1>
```

Great! We now have access to the router.

2.2. Linux

Let's see if our Linux user is able to connect. Let's verify the public key fingerprint / hash first:

```
   key-hash ssh-rsa 39970CAB33EABB8BE39F4FDB9AFECFFE
  quit
```

You can see the fingerprint on Linux with the following command:

```
$ ssh-keygen -l -f /home/ubuntu/.ssh/id_rsa.pub
2048 39:97:0c:ab:33:ea:bb:8b:e3:9f:4f:db:9a:fe:cf:fe ubuntu@HOST1
(RSA)
```

The key fingerprints match, let's see if we can connect:

```
$ ssh LINUX_USER@192.168.1.1
R1>
```

There we go, we are now connected!

Configurations

Want to take a look for yourself? Here you will find the final configuration of each device.

R1

```
hostname R1
!
ip cef
!
ip domain name NETWORKLESSONS.LOCAL
!
ip ssh pubkey-chain
  username WINDOWS_USER
   key-hash ssh-rsa 8FB4F858DD7E5AFB372780EC653DB371
  quit
  username LINUX_USER
   key-hash ssh-rsa 39970CAB33EABB8BE39F4FDB9AFECFFE
  quit
!
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
!
end
```

windows_user.ppk

```
RGauEp7FzN/uXxsX7mii6qOuxovl9OflLpXcvH5QH6551ycmL8nIv8UCY8uayiGI
INsC0LyKEctWDW6qWp43T7rhcP0y4JoMraTCZLIPNE0Bo0bHgnGLg6fEvJmyB3sX
H+7BaxHdYKg2OcIgVqYzclWhDwxj32kqd1BCq089iBMrb4QppDU2eM/t22iK29mn
eqOGTiCkxB80ix+KULT9okmqkj3TbhCpunTfuPCCRNrjqndBsw==
Private-Lines: 14
AAABAFmpHJnZhJpA/a32mGA6pswL9qsnmw+V2Em84SuUMJlg9rwLycLWLXpIDPHi
ksNQAhHqZ7Iad9w/L57T3wzzgxuDYw7rM876ONWkch6HytulQghZ5ro7FnEhRxS9
MxkAzIMuon8VGY8F0xt4F64nO11EneQsw0emqZfeZiMmRwPu7JFgb29D5vdUNmdq
nhbR9WecApW1dvURbZ2Zie8euivzmPnoQSjaRqO1u25TOr3JwefnEk0GnC5+nw4i
/cbLyLTf1gIZ99L50rVEbKaB9+AUu//0ujXTI3MoGBz1w9+CNdvujTshUnvasJR+
kjCT5Ix3U0ekDUy4PwEI+IAPXq0AAACBAPsbfbPAwSWJX8jcU2gy/K6j84Y3a08W
BQKAW1O4FOEzF3IMYvyZqP1B8NYyTRWdTNmRj4zoE6Eym8wsyzkI9okS2IMH1kO8
Nkdsnf5Mo4GmLuMVyaCMTqAV6y0qXnQnFOGsFmIgnumBTF9ibQ01WVFdswfuR7jS
cZSS3+8SyPBRAAAAgQCM64OhK8+Avpyak/hwMCd/POtXz2xb6jIuYPpOmo0x9kKN
oKljrqAyiSdqo4wGk9ijiZRtGu75BwcyD78gyPEFYGI9gZH2NIJ77quwTNhJ2eZc
OG0DkXRpdGX/zkou2SMFZPkOBK1BpczOBZl7/KssYnsBaGWSZY2MkGJ1fOv0wwAA
AIB/w7ZxhuNhugm853HITN4zk1iaanbEsbEspE5tM568PletZPCCKYdL2GE6CS9l
y/ItVCraZ+4w7i4kbtYfI+WFcpDMU0g5NM3wF5ggXp30pL2WJHKLzTtonpbmKEGx
mOOvlLNVlxhit8w0TXgXoNXik2UGIg3DryMWo+HSLeLjsA==
Private-MAC: 2c0a896321c46de890909cd1697869a18690ae7c
```


windows_user.pub


```
---- BEGIN SSH2 PUBLIC KEY ----
Comment: "WINDOWS_USER"
AAAAB3NzaC1yc2EAAAABJQAAAQEAijoMF9oBwyQxwYbVlFprz+fG8oe5uAcCxwMw
eIR1lyAnDJIsYbTbcdm+n5KiQnCt2561MpN4yOFpajFNM/dqH7/jYaqaicHCSV2F
RGauEp7FzN/uXxsX7mii6qOuxovl9OflLpXcvH5QH6551ycmL8nIv8UCY8uayiGI
INsC0LyKEctWDW6qWp43T7rhcP0y4JoMraTCZLIPNE0Bo0bHgnGLg6fEvJmyB3sX
H+7BaxHdYKg2OcIgVqYzclWhDwxj32kqd1BCq089iBMrb4QppDU2eM/t22iK29mn
eqOGTiCkxB80ix+KULT9okmqkj3TbhCpunTfuPCCRNrjqndBsw==
---- END SSH2 PUBLIC KEY ----
```


id_rsa

```
NT/RIvH3hQVS01oo14opW2FsHS4a5Oy9hEL+cQIDAQABAoIBAAM4kilVYXbSxMM2
BSCrSfbDnAl+P07ID0Td3+DkZBqBDVFOVN0SK5yMvgtq4b/iNtlEsXUSO+p0VuBN
Z08k1eu/D8ANrmNu2sCALQc+oZTv4QU/sJZDkWHUmdqt1IQSV4DE2Ifph4Y0u8rn
6effy+CJov3syyLKetVSUihpcwhVT+k4tMNHJAVTLmlbP5opqL1m6B10uI9QaoHV
uXmenMW0N6V8zy+46rbzbqFdyFUX3SEeRHhmPQZOJ6+T90KFt35mk6Nnkn442OUT
68ErQXf0LwUYt7XtokrFDS2kGghy28wiyJ3cAnUGikrFp8OGVy75bwQjOs0rLE7r
ppBps7kCgYEA+17zfFq4PTwO2MS/MhaeozvkD+saKd+xqi/2wUAa6/XasF28rGlg
Kk7haO9mkJBzttQpQxqzfiSgzf8L9097BcZsKNOO1UOA9is4cwJzvso1JrbIqO0q
pRMvSPmJiNtLdrfaDruQ0/b/5vJqqVhapWDpzAIe7vmqEV+cA2BiuT8CgYEAwEpc
q2uSHVXFY4TFOpLArY2kQRUcGrXL36i5diARqboy0R0/AHmVtjjoT3pXPS12lfCT
Ph1AhyPxUgP36jqiM8xDO8h0a5Q900hF9ewhfbuGJMsMmTauY3NhIqS/LNepOSW2
FnBhhWgrAhNNmhj4e2CAxHzsopms6lume9fLLE8CgYEAuqs8bbCA+RhMfjU9NtkN
XXLwXdHdUBNKQHP17nTIiUm96RLzNaXbQA/r3J1LsTdUdwT+z1JY0Ogqck9gd3uS
hCthzr+sZjsG3cgi12WOxrQq3GEIikn1TTj9+Fc3B2ayl6rYR/CKEJ5wUvTauH9g
cUeS12kLx7On09cIaIl/RskCgYAxGItypSOIY+bjEvo8i02wwlJm35nuY+5q66x2
sZdw636gD8SPPXvxK7R83nK5xwrZG7SsjlF0b8PkueipoFDbwtKDyBmZgh8K/BAI
y0J91MxaRpGv6Ns7vzDU5JV/QI0Pb0Z/kjAEHOWmQQF2T9vZvHkEMhVFKtGQgNgQ
FLfmVwKBgQCc8dniIRy2PLa6N7hZuEGnAYwX1Wx+BizF4UxDrSoYu1T5TJizjqaY
s2uFvtTb6GmAEYg+n9GICifYG5Yv8SWPCaBHehKetIdDsChlrLzYq2LJRt1/n7Bp
dnRXg8G9zQILjOWji1rk4UYSLrz7PidYUjs2jcMrqJd7vPp7oW2Nlg==
-----END RSA PRIVATE KEY-----
```


id_rsa.pub


```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC80DsOF4nkk15V0V2U7r4Q2MyAwIbgQX/7rq
dUyNCTulliYZWdxnQHaI0WpvcEHQTrSXCauFOBqUrLZglI2VExOgu0TmmWCajW/vnp
8J5bArzwIk83ct35IHFozPtl3Rj79U58HwMlJ2JhBTkyTrZYRmsP+r9VF7pYMVcuKg
FS+gDvhbuxM8DNLmS1+eHDw9DNHYBA+dIaEIC+ozxDV7kF6wKOx59E/Ni2/dT9TJ5Q
ge+Rw7zn+O0i1Ib95djzNfVdHq+174mchGx3zV6l/6EXvc7G7MyXj89ffLdXIp/Xy/
wdWkc1P9Ei8feFBVLTWijXiilbYWwdLhrk7L2EQv5x ubuntu@HOST1
```


3. Conclusion

You have now learned how to configure your Cisco IOS router or switch to accept SSH public key authentication using Windows and Linux users.

« **Previous Lesson**
OpenSSL Certification Authority (CA) on Ubuntu Server

Next Lesson »
NAT Extendable on Cisco IOS

Tags: Certificate, Network Management, SSH

Forum Replies

lagapides

Hello Edgar

Thanks for pointing that out, I'll give a shout out to Rene to see if it can be located somewhere more accessible.

Thanks again!

Laz

syncope988

Hi Rene and staff,
i was doing a basic lab ipv6 with GNS3 and SSH came in front of the scene (no matter ipv4 or ipv6 in this post)
This is the lab

https://cdn-forum.networklessons.com/uploads/default/original/2X/1/1ed1247d4d05d2ff35f2f39a51941064d60694ae.png

and opening SSH session from HostA to R1 led me to review crypto keys with cisco (just to remind)
R1 configuration is

https://cdn-forum.networklessons.com/uploads/default/original/2X/3/3cddb9fa219925dc3053076c7c5c323d6698405d.png

Host A is toolbox
https://cdn-forum.networklessons.com/uploads/default/origina

... Continue reading in our forum

lagapides

Hello Dominique

The key data in the router and that found within the known_hosts file do appear different, and this is simply because of the method of encoding. Within the router, the key data is displayed in Hexadecimal while in the known_hosts file, in what is known as Base64 encoding which represents binary data in ASCII (and this is why you see all the letters of the alphabet as well as many symbols). This Ubuntu man page includes a description of the format of the known_hosts file format.

I hope this has been helpful!

Laz

cohenaa1

Hi!

This is a great walkthrough but I have an older 3560 in my lab and the `ip ssh pubkey-chain` command doesn't exist. Is there a different method to accomplish this on older switches? Here's the image it's running: