SSH PK Authentication and Auto login configuration for Chassis Management Controller

Dell technical white paper
Dell Engineering

May 2016

Author:

Elangovan G

Revisions

Date	Description
August 2013	Initial release
May 2016	Copyright statement updated

Table of contents

Executive summary

Public key cryptography, or asymmetric cryptography, is any cryptographic system that uses two kinds of keys: public keys that may be disseminated widely, while private keys are known only to the owner. In a public key encryption system, any person can encrypt a message using the public key of the receiver, but such a message can be decrypted only with the receiver's private key. For this to work it must be computationally easy for a user to generate a public and private key-pair to be used for encryption and decryption. The strength of a public key cryptography system relies on the degree of difficulty (computational impracticality) for a properly generated private key to be determined from its corresponding public key. Security then depends only on keeping the private key private, and the public key may be published without compromising security.

Public key cryptography systems often rely on cryptographic algorithms based on mathematical problems that currently admit no efficient solution—particularly those inherent in certain integer factorization, discrete logarithm, and elliptic curve relationships. Public key algorithms, unlike symmetric key algorithms, do not require a secure channel for the initial exchange of one (or more) secret keys between the parties.

Because of the computational complexity of asymmetric encryption, it is usually used only for small blocks of data, typically the transfer of a symmetric encryption key (e.g. a session key). This symmetric key is then used to encrypt the rest of the potentially long message sequence. The symmetric encryption/decryption is based on simpler algorithms and is much faster.

Message authentication involves hashing the message to produce a "digest," and encrypting the digest with the private key to produce a digital signature. Thereafter anyone can verify this signature by computing the hash of the message, decrypting the signature with the signer's public key, and comparing the computed digest with the decrypted digest. Equality between the digests confirms the message is unmodified since it was signed, and that the signer, and no one else, intentionally performed the signature operation — presuming the signer's private key has remained secret. The security of such procedure depends on a hash algorithm of such quality that it is computationally impossible to alter or find a substitute message that produces the same digest - but studies have shown that even with the MD5 and SHA-1 algorithms, producing an altered or substitute message is not impossible.The current hashing standard for encryption is SHA-2. The message itself can also be used in place of the digest.

Public key algorithms are fundamental security ingredients in cryptosystems, applications and protocols. They underpin various Internet standards, such as Transport Layer Security (TLS), S/MIME, PGP, and GPG. Some public key algorithms provide key distribution and secrecy (e.g., Diffie–Hellman key exchange), some provide digital signatures (e.g., Digital Signature Algorithm), and some provide both (e.g., RSA).

Public key cryptography finds application in, among others, the information technology security discipline. Information security (IS) is concerned with all aspects of protecting electronic information assets against security threats.[6] Public key cryptography is used as a method of assuring the confidentiality, authenticity and non-repudiability of electronic communications and data storage.

1 Introduction

The system management consoles like Chassis Management Controller (CMC) and iDRAC providing a support for double factor authentication method for Auto login which is known as "SSH PK Authenication".

This is accomplished using Public and Private keys which can be generated though client tools like "Putty Key Generator".

The same Public and Private keys can be re-used across same or different platforms.

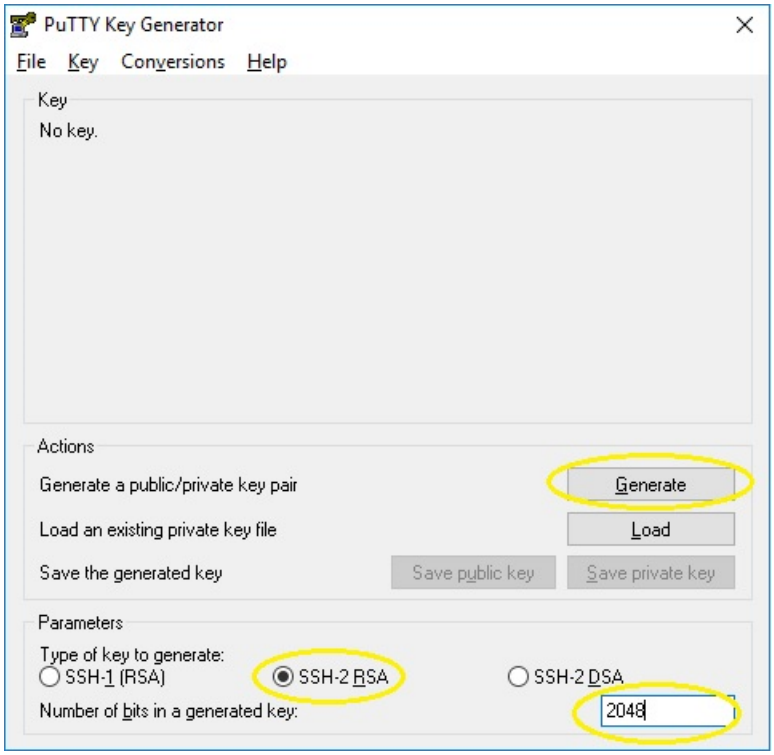
This document provides step by step procedure to Setup, Configure and Use "**SSH PK Authentication**" for CMC.

2 Generate Public and Private key using Putty Generate tool

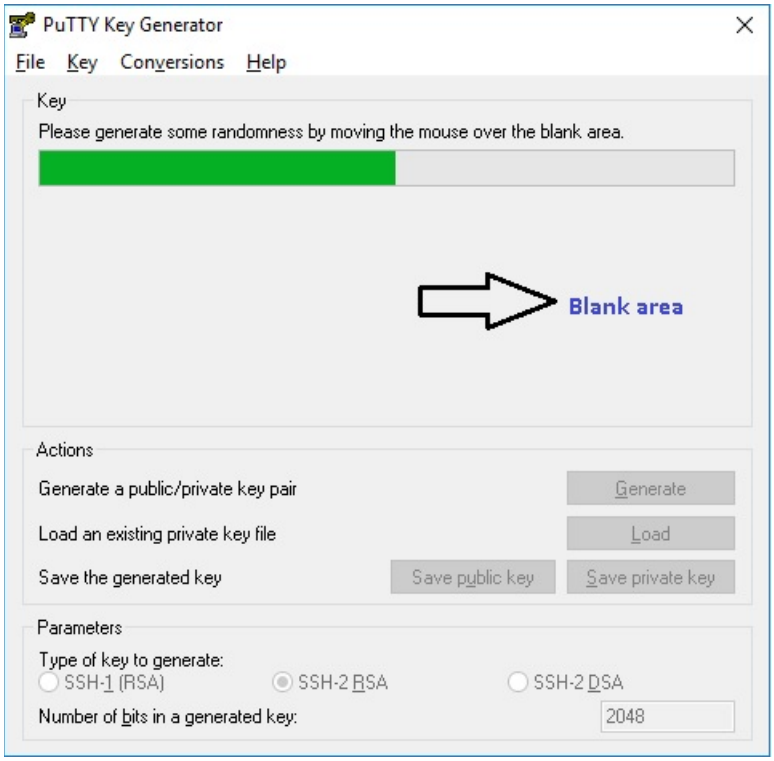
To generate public and private key using the "**Putty Key Generator"** tool, perform the following steps.

1) Select Type of key to generate as "**SSH-2 RSA**" and Valid key size is 2048 and above.
2) Open Putty and click the Generate button to generate apublic and private key set.

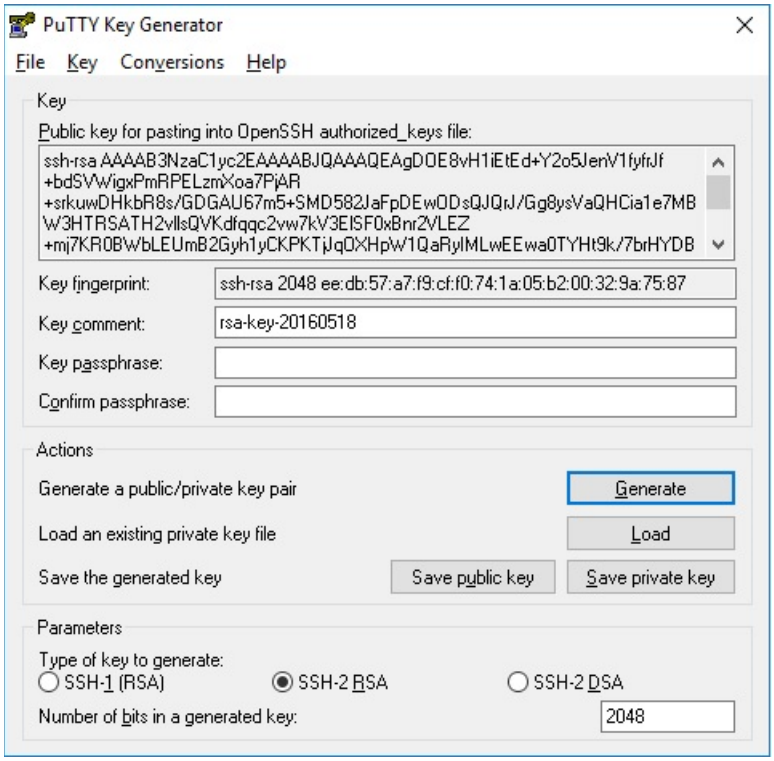
Note: SSH-2 DSA algoritham is not supported for CMC.



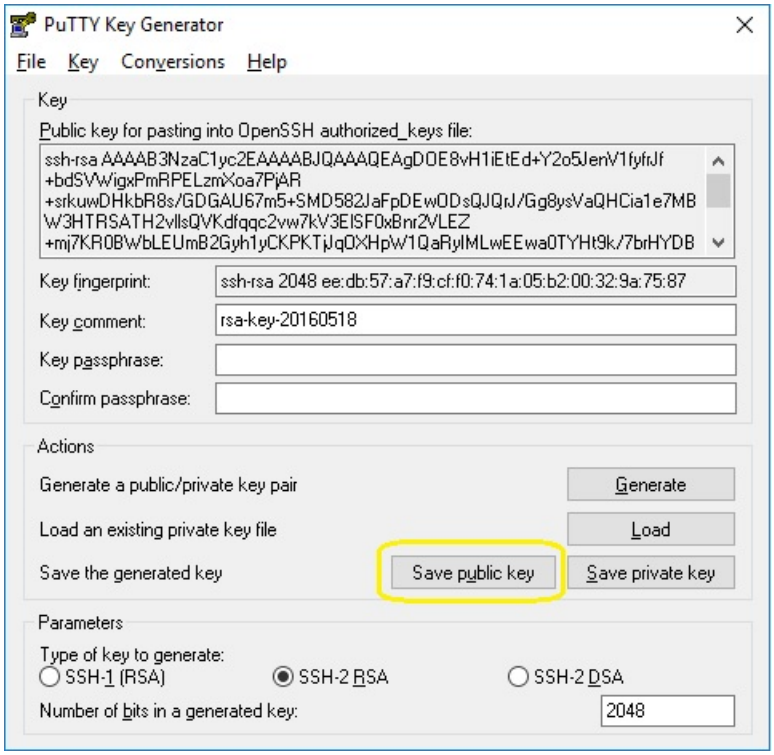
3) Move the cursor in the blank area to generate the key set.



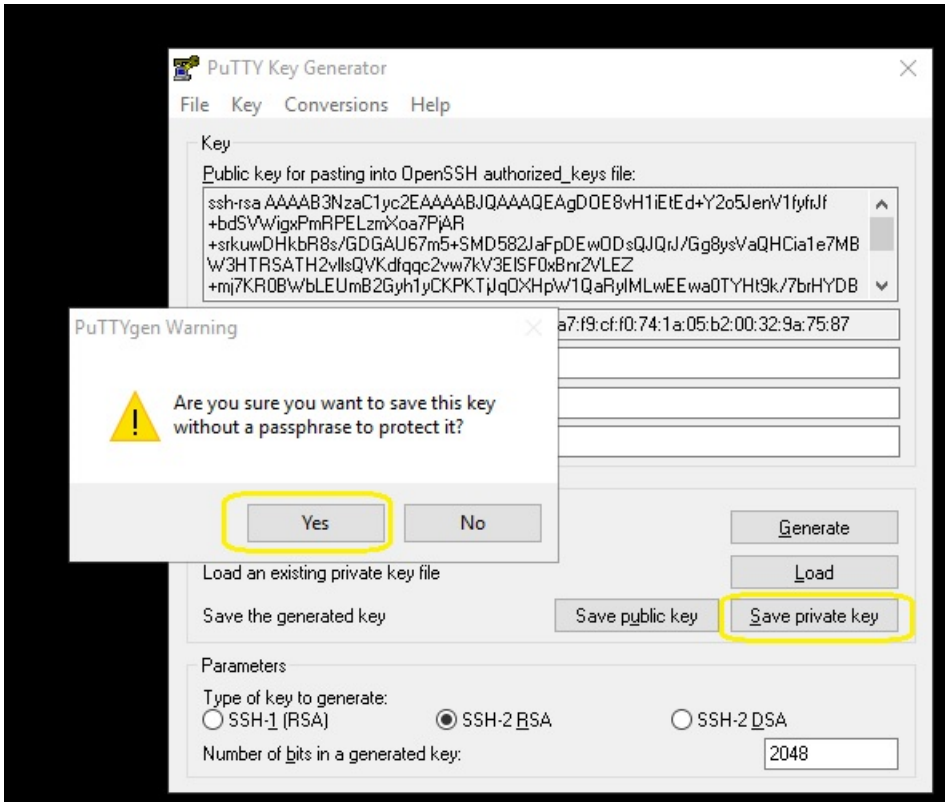
4) The public and private key are generated.



5) Click the "**Save public key**" button to save the puplic key.



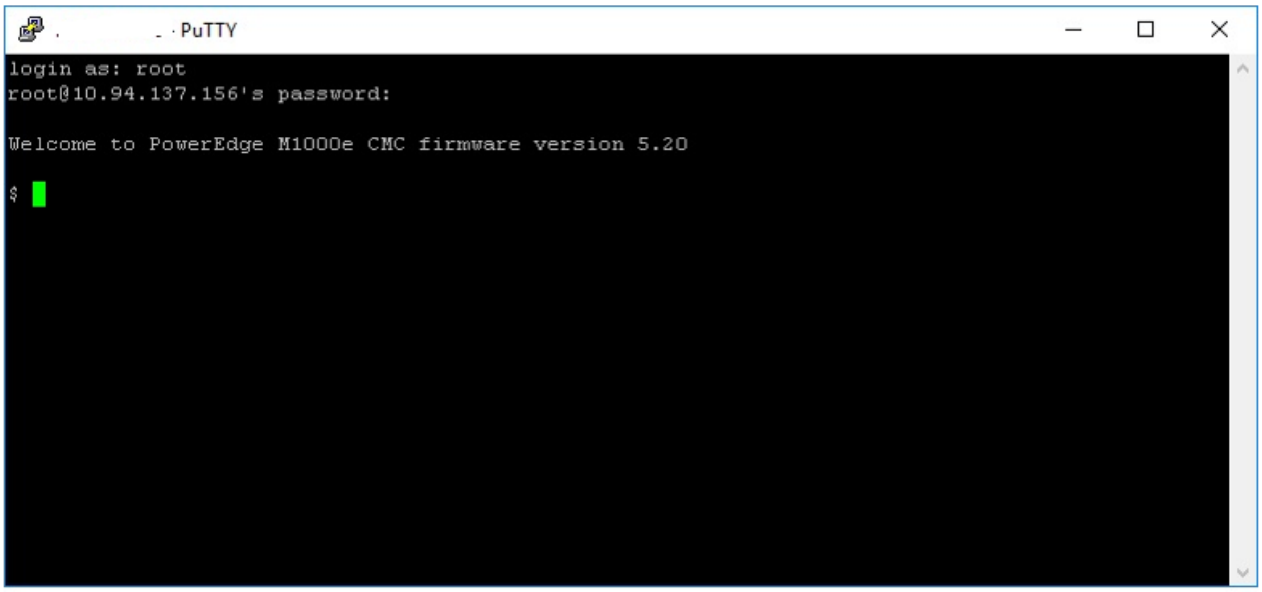
6) Save the Private key by clicking "**Save private key**" button.



3 Uploading public key to CMC

Login to CMC and upload the SSH PK Authentication public key, which is generated using Putty Generate tool, using the following method.

1) Login to CMC using admin credentials (the default credentials are root/calvin)



2) Verify the sshpk authentication command help details using the command, "**racadm sshpkauth**".

```
PuTTY

$ ? sshpkauth

sshpkauth -- manage PK Authentication keys and accounts

Usage:

racadm sshpkauth -i svcacct -k <key_index> -p <privilege> -t <PK_key_text>
racadm sshpkauth -i svcacct -k <key_index> -p <privilege> -f <PK_key_file>
racadm sshpkauth -v -i svcacct -k all|<key_index>
racadm sshpkauth -d -i svcacct -k all|<key_index>

--------------------------------------------------------------------------------

Valid Options:

-i : index for the user, it must be svcacct for CMC
-k : <key_index> : index from 1 to 6 (or all for -v / -d options) to assign the PK key being uploaded
-p : <privilege> : level to give to user for this PK key
-t : key text for PK key
-f : file containing key text to upload
    NOTE: This option is only supported on the remote interface(s).
-v : view privilege and key text
-d : delete key and privilege for index provided


--------------------------------------------------------------------------------

Usage Examples:

- View all keys:
  racadm sshpkauth -i svcacct -k all -v

- Delete all keys:
  racadm sshpkauth -i svcacct -k all -d

- Upload key at index 2 using text option:
  racadm sshpkauth -i svcacct -k 2 -p 0xfff -t "key text"

- Upload key at index 1 using file upload option:
  racadm sshpkauth -i svcacct -k 1 -p 0xfff -f idrsa.pub

$
```

3) Verify the already uploaded public key details using the command, "**racadm sshpkauth -i svcacct -k all -v**".

```
- PuTTY
$ racadm sshpkauth -i svcacct -k all -v
Key 1=UNDEFINED
Key 2=UNDEFINED
Key 3=UNDEFINED
Key 4=UNDEFINED
Key 5=UNDEFINED
Key 6=UNDEFINED
Privilege 1=0x0
Privilege 2=0x0
Privilege 3=0x0
Privilege 4=0x0
Privilege 5=0x0
Privilege 6=0x0
$
```

4) Upload the generated puplic key text to CMC using the command, "racadm sshpkauth -i svcacct -k 1 -p 0xfff -t "ssh-rsa.AAAAB3NzaC1yc2EAAAABJQAAAIEA7ki1GVLtX3/jf76WJTohOlKjivNzHqcRgn5sm6xK+h+edOVs4m+J38hNgdumI9Ee5g3mddNSaHadHrVXjoE4OujqIK8EwjzO6umjTIS2NIC7aI8+Zo5InPjyK0Xi7Px233AQDbN9yX+5eCU38wvvd9RR6vRNy1UWSXX5MABAgWM= rsa-key-20160419""

```
$ racadm sshpkauth -i svcacct -k 1 -p 0xfff -t "ssh-rsa AAAAB3NzaC1yc2EAAAABJQAA
AIEA7ki1GVLtX3/jf76WJTohOlKjivNzHqcRgn5sm6xK+h+edOVs4m+J38hNgdumI9Ee5g3mddNSaHad
HrVXjoE4OujqIK8EwjzO6umjTIS2NIC7aI8+Zo5InPjyK0Xi7Px233AQDbN9yX+5eCU38wvvd9RR6vRN
y1UWSXX5MABAgWM= rsa-key-20160419"
Key successfully uploaded
$
```

Note: Once the key is uploaded successfully, the command output is displayed as "**Key successfully uploaded**".

Or upload the public key file using the command, "racadm -r xxx.xxx.xxx.xxx -u root -p password@123 sshpkauth -i svcacct -k 1 -p 0xfff -f Putty-Public".

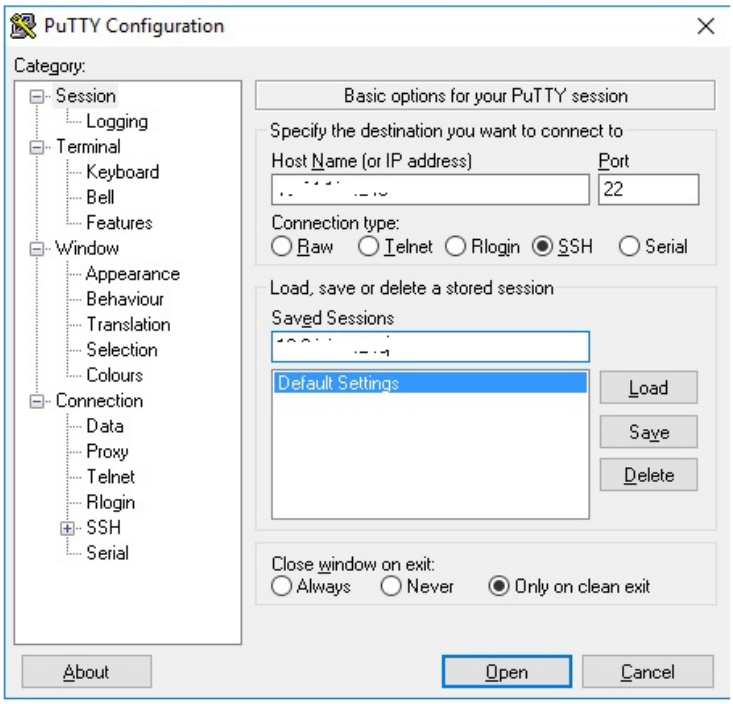
```
C:\Share\Putty-Key>racadm -r [redacted] -u root -p password@123 sshpkauth -i svcacct -k 1 -p 0xfff -f Putty-Public
Security Alert: Certificate is invalid - Certificate is not signed by Trusted Third Party
Continuing execution. Use -S option for racadm to stop execution on certificate-related errors.
PK SSH Authentication Key file successfully uploaded to the RAC
```

5) Verify the uploaded public key using the command,"**racadm sshpkauth -i svcacct -k all -v**".

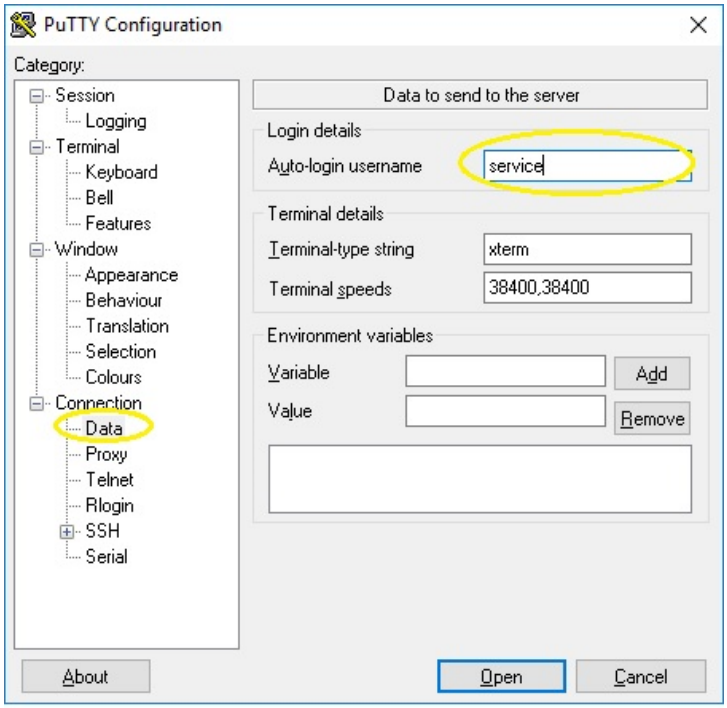
```
$ racadm sshpkauth -i svcacct -k all -v
Key 1=ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAIEA7ki1GVLtX3/jf76WJTohOlKjivNzHqcRgn5sm6
xK+h+edOVs4m+J38hNgdumI9Ee5g3mddNSaHadHrVXjoE4OujqIK8EwjzO6umjTIS2NIC7aI8+Zo5InP
jyKOXi7Px233AQDbN9yX+5eCU38wvvd9RR6vRNy1UWSXX5MABAgWM= rsa-key-20160419
Key 2=UNDEFINED
Key 3=UNDEFINED
Key 4=UNDEFINED
Key 5=UNDEFINED
Key 6=UNDEFINED
Privilege 1=0xfff
Privilege 2=0x0
Privilege 3=0x0
Privilege 4=0x0
Privilege 5=0x0
Privilege 6=0x0
$
```

4 Configure and uploading private key to Putty

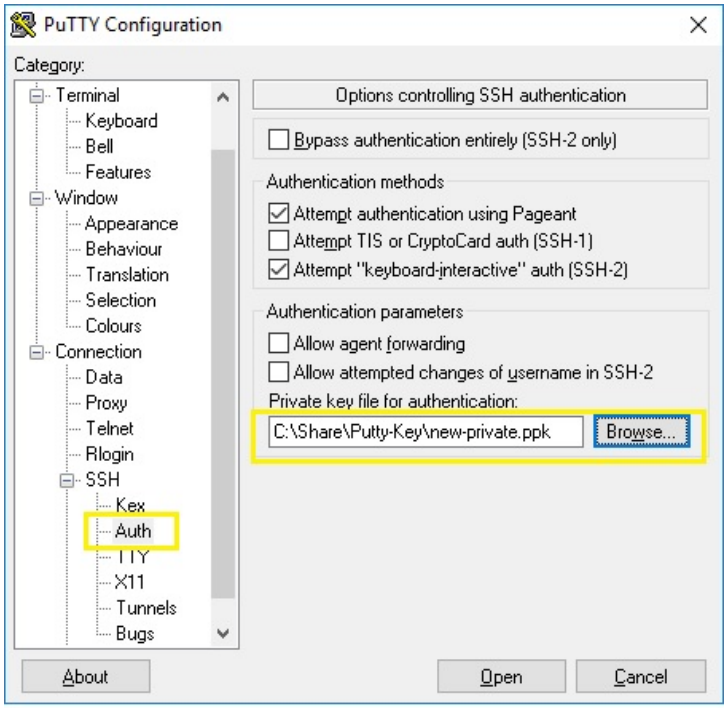
1) Open the Putty and provide the **Host Name or IP address** for the CMC in the Host Name (or IP address) and Saved Sessions text boxes.



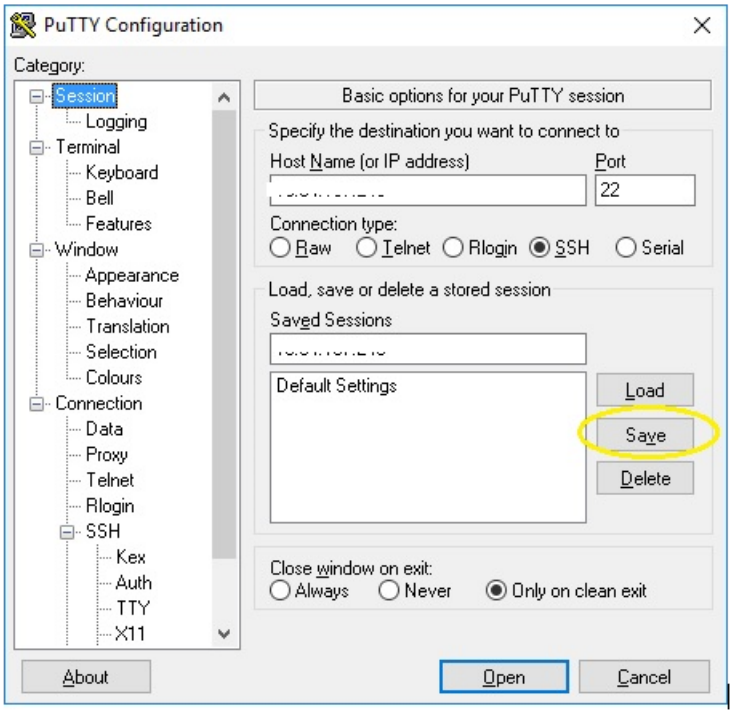
2) Navigate the Putty configuration to Session →Terminal → Window → Connection → Data and provide the Login details. Use the Auto-login username, "**service**".



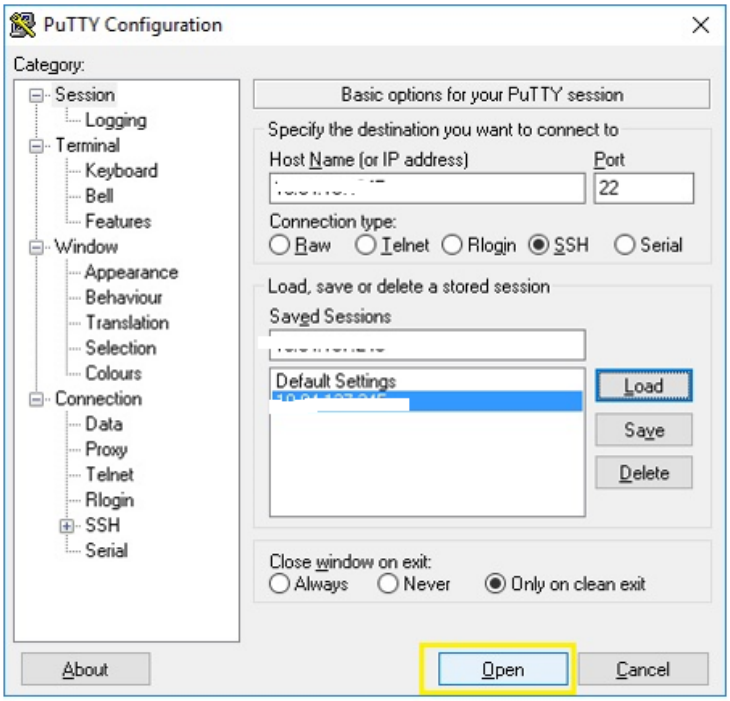
3) Upload the Private key file by navigate Putty configuration to Session → Terminal →Window →Connection → SSH → Auth and click the browse button for provide the Private key(***.ppk**) file location



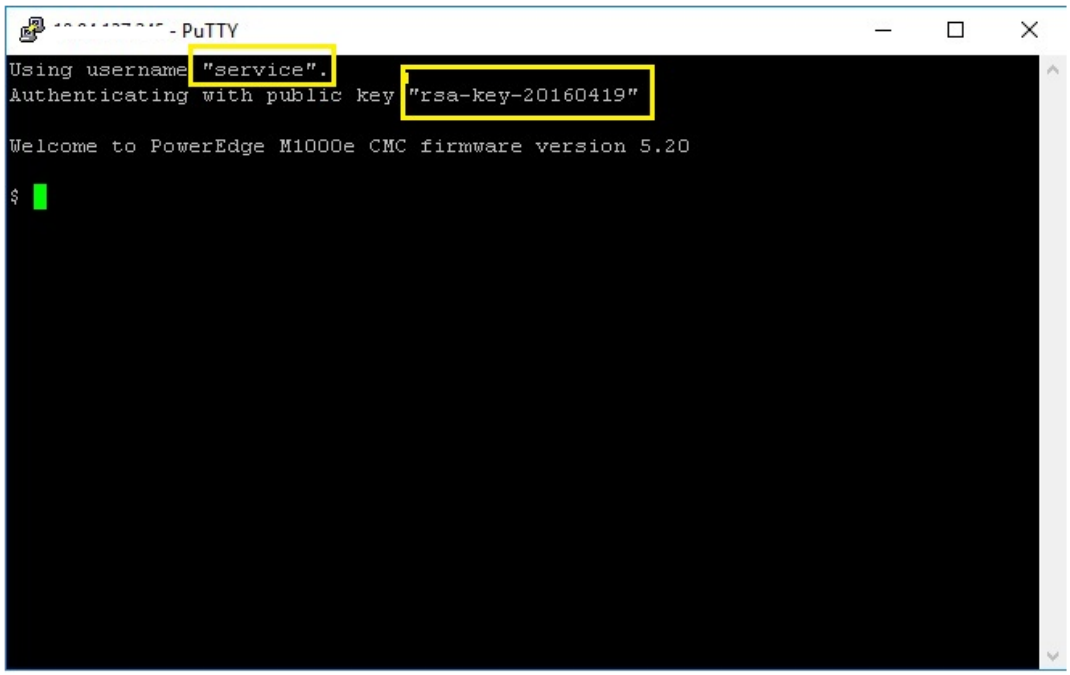
4) Navigate to Session and click the save button to save the CMC session in Putty.



5) Click the Open button to open the CMC SSH PK Authentication session.



6) The SSH PK Authentication session for CMC is opened successfully without prompting for username and password.



7) Check the Open session for CMC using the command, "**racadm getssninfo**".