

# Comprendre Secure Shell Packet Exchange

## Table des matières

---

[Introduction](#)

[Conditions préalables](#)

[Exigences](#)

[Composants utilisés](#)

[Protocole SSH](#)

[Échange SSH](#)

[Informations connexes](#)

---

## Introduction

Ce document décrit l'échange au niveau paquet pendant la négociation Secure Shell (SSH).

## Conditions préalables

### Exigences

Cisco recommande que vous ayez une connaissance des concepts de sécurité de base :

- Authentification
- Confidentialité
- Intégrité
- Méthodes d'échange de clés

### Composants utilisés

Ce document n'est pas limité à une version matérielle spécifique.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration.

## Protocole SSH

Le protocole SSH est une méthode de connexion à distance sécurisée d'un ordinateur à un autre. Les applications SSH sont basées sur une architecture client-serveur, connectant une instance de client SSH à un serveur SSH.

# Échange SSH

1. La première étape de SSH est appelée Identification String Exchange.

a. Le client construit un paquet et l'envoie au serveur contenant :

- Version du protocole SSH
- Version du logiciel

```
323 5.946818 10.65.54.8 10.106.51.72 SSHv2 82 Client: Protocol (SSH-2.0-PuTTY_Release_0.76)
> Frame 323: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
> Ethernet II, Src: Cisco_3c:7a:00 (00:05:9a:3c:7a:00), Dst: Cimsys_33:44:55 (00:11:22:33:44:55)
> Internet Protocol Version 4, Src: 10.65.54.8, Dst: 10.106.51.72
> Transmission Control Protocol, Src Port: 56127, Dst Port: 22, Seq: 1, Ack: 1, Len: 28
v SSH Protocol
  Protocol: SSH-2.0-PuTTY_Release_0.76
```

La version du protocole client est SSH2.0 et la version du logiciel est Putty\_0.76.

b. Le serveur répond avec son propre échange de chaîne d'identification, y compris sa version de protocole SSH et sa version logicielle.

```
326 6.016955 10.106.51.72 10.65.54.8 SSHv2 73 Server: Protocol (SSH-2.0-Cisco-1.25)
> Frame 326: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
> Ethernet II, Src: Cimsys_33:44:55 (00:11:22:33:44:55), Dst: Cisco_3c:7a:00 (00:05:9a:3c:7a:00)
> Internet Protocol Version 4, Src: 10.106.51.72, Dst: 10.65.54.8
> Transmission Control Protocol, Src Port: 22, Dst Port: 56127, Seq: 1, Ack: 29, Len: 19
v SSH Protocol
  Protocol: SSH-2.0-Cisco-1.25
```

La version du protocole du serveur est SSH2.0 et la version du logiciel est Cisco1.25

2. Étape suivante **Algorithm Negotiation**. Dans cette étape, le client et le serveur négocient les algorithmes suivants :

- Échange De Clés
- Chiffrement
- HMAC (Hash-based Message Authentication Code)
- Compression

1. Le client envoie un message d'initialisation d'échange de clés au serveur, en spécifiant les algorithmes qu'il prend en charge. Les algorithmes sont répertoriés par ordre de préférence.

```
329 6.021990 10.65.54.8 10.106.51.72 SSHv2 238 Client: Key Exchange Init
> Frame 329: 238 bytes on wire (1904 bits), 238 bytes captured (1904 bits) on interface 0
> Ethernet II, Src: Cisco_3c:7a:00 (00:05:9a:3c:7a:00), Dst: Cimsys_33:44:55 (00:11:22:33:44:55)
> Internet Protocol Version 4, Src: 10.65.54.8, Dst: 10.106.51.72
> Transmission Control Protocol, Src Port: 56127, Dst Port: 22, Seq: 1101, Ack: 20, Len: 184
> [3 Reassembled TCP Segments (1256 bytes): #327(536), #328(536), #329(184)]
v SSH Protocol
  SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
    Packet Length: 1252
    Padding Length: 11
  Key Exchange
    Message Code: Key Exchange Init (20)
    Algorithms
```

Init d'échange de clés

```

Algorithms
Cookie: 47a96215afc92003180b60342970a105
kex_algorithms length: 315
kex_algorithms string [truncated]: curve448-sha512,curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,dif
server_host_key_algorithms length: 123
server_host_key_algorithms string: rsa-sha2-512,rsa-sha2-256,ssh-rsa,ssh-ed448,ssh-ed25519,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,ssh-dss
encryption_algorithms_client_to_server length: 189
encryption_algorithms_client_to_server string: aes256-ctr,aes256-cbc,rijndael-cbc@lysator.liu.se,aes192-ctr,aes192-cbc,aes128-ctr,aes128-cbc,chacha20-poly1305
encryption_algorithms_server_to_client length: 189
encryption_algorithms_server_to_client string: aes256-ctr,aes256-cbc,rijndael-cbc@lysator.liu.se,aes192-ctr,aes192-cbc,aes128-ctr,aes128-cbc,chacha20-poly1305
mac_algorithms_client_to_server length: 155
mac_algorithms_client_to_server string: hmac-sha2-256,hmac-sha1,hmac-sha1-96,hmac-md5,hmac-sha2-256-etm@openssh.com,hmac-sha1-etm@openssh.com,hmac-sha1-96-etm
mac_algorithms_server_to_client length: 155
mac_algorithms_server_to_client string: hmac-sha2-256,hmac-sha1,hmac-sha1-96,hmac-md5,hmac-sha2-256-etm@openssh.com,hmac-sha1-etm@openssh.com,hmac-sha1-96-etm
compression_algorithms_client_to_server length: 26
compression_algorithms_client_to_server string: none,zlib,zlib@openssh.com
compression_algorithms_server_to_client length: 26
compression_algorithms_server_to_client string: none,zlib,zlib@openssh.com

```

Algorithmes pris en charge par le client

b. Le serveur répond avec son propre message d'initialisation d'échange de clés, en répertoriant les algorithmes qu'il prend en charge.

c. Puisque ces messages sont échangés simultanément, les deux parties comparent leurs listes d'algorithmes. S'il existe une correspondance dans les algorithmes pris en charge par les deux côtés, ils passent à l'étape suivante. S'il n'y a pas de correspondance exacte, le serveur sélectionne le premier algorithme de la liste du client qu'il prend également en charge.

d. Si le client et le serveur ne parviennent pas à s'entendre sur un algorithme commun, l'échange de clés échoue.

```

334 6.093250 10.106.51.72 10.65.54.8 SSHv2 366 Server: Key Exchange Init
> Frame 334: 366 bytes on wire (2928 bits), 366 bytes captured (2928 bits) on interface 0
> Ethernet II, Src: Cimsys_33:44:55 (00:11:22:33:44:55), Dst: Cisco_3c:7a:00 (00:05:9a:3c:7a:00)
> Internet Protocol Version 4, Src: 10.106.51.72, Dst: 10.65.54.8
> Transmission Control Protocol, Src Port: 22, Dst Port: 56127, Seq: 20, Ack: 1285, Len: 312
SSH Protocol
  SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
    Packet Length: 308
    Padding Length: 4
    Key Exchange
      Message Code: Key Exchange Init (20)
      Algorithms

```

Init Exchange de clé serveur

3. Ensuite, les deux parties entrent en phase de génération du secret partagé à l'aide de l'échange de clés DH et d'authentification du serveur **Key Exchange**:

a. Le client génère une paire de clés **Public** and **Private** et envoie la clé publique DH dans le paquet **DH Group Exchange Init**. Cette paire de clés est utilisée pour le calcul des clés secrètes.

```

337 6.201114 10.65.54.8 10.106.51.72 SSHv2 326 Client: Diffie-Hellman Group Exchange Init
> Frame 337: 326 bytes on wire (2608 bits), 326 bytes captured (2608 bits) on interface 0
> Ethernet II, Src: Cisco_3c:7a:00 (00:05:9a:3c:7a:00), Dst: Cimsys_33:44:55 (00:11:22:33:44:55)
> Internet Protocol Version 4, Src: 10.65.54.8, Dst: 10.106.51.72
> Transmission Control Protocol, Src Port: 56127, Dst Port: 22, Seq: 1309, Ack: 612, Len: 272
SSH Protocol
  SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
    Packet Length: 268
    Padding Length: 6
    Key Exchange
      Message Code: Diffie-Hellman Group Exchange Init (32)
      Multi Precision Integer Length: 256
      DH client e: 1405ab00ff368031363467ad6653967d5a64eac4734e5dc6.
      Padding String: 5c81f2cffc95

```

Client DH Public Key & Diffie-Hellman Group Exchange Init

b. Le serveur génère sa propre **Public and Private** paire de clés. Il utilise la clé publique du client et sa propre paire de clés pour calculer le secret partagé.

c. Le serveur calcule également un hachage **Exchange** avec les entrées suivantes :

- Chaîne D'identification Des Clients
- Chaîne D'identification Du Serveur
- Charge utile du client KEXINIT
- Charge utile du serveur KEXINIT
- Serveurs Clé publique à partir des clés d'hôte ( Paire de clés RSA )
- Clé publique DH des clients
- Serveurs DH Clé publique
- Clé secrète partagée

d. Après avoir calculé le hachage, le serveur le signe avec sa clé privée RSA.

e. Le serveur construit un message **DH\_Exchange\_Reply** qui inclut :

- RSA-Clé publique du serveur (pour aider le client à authentifier le serveur)
- DH-Clé publique du serveur (pour le calcul du secret partagé)
- HASH (pour authentifier le serveur et prouver que le serveur a généré le secret partagé, car la clé secrète fait partie du calcul de hachage)

```
343 6.330017 10.106.51.72 10.65.54.8 SSHv2 350 Server: Diffie-Hellman Group Exchange Reply
Internet Protocol Version 4, Src: 10.106.51.72, Dst: 10.65.54.8
Transmission Control Protocol, Src Port: 22, Dst Port: 56127, Seq: 1148, Ack: 1581, Len: 296
[2 Reassembled TCP Segments (832 bytes): #342(536), #343(296)]
SSH Protocol
  SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
    Packet Length: 828
    Padding Length: 8
  Key Exchange
    Message Code: Diffie-Hellman Group Exchange Reply (33)
  KEX host key (type: ssh-rsa)
    Host key length: 279
    Host key type length: 7
    Host key type: ssh-rsa
    Multi Precision Integer Length: 3
    RSA public exponent (e): 010001
    Multi Precision Integer Length: 257
    RSA modulus (N): 0098c7d23c9ababd730f07b5c2aee1e4e51bac67970aa5af...
    Multi Precision Integer Length: 256
    DH server f: 3a17a0995531f12d629a48ab6f25715bc181ea3deb6c6793...
    KEX H signature length: 271
    KEX H signature: 000000077373682d72736100000100691d2c896761bc7481...
    Padding String: 0000000000000000
```

Server DH Public Key & Diffie-Hellman Group Exchange Réponse

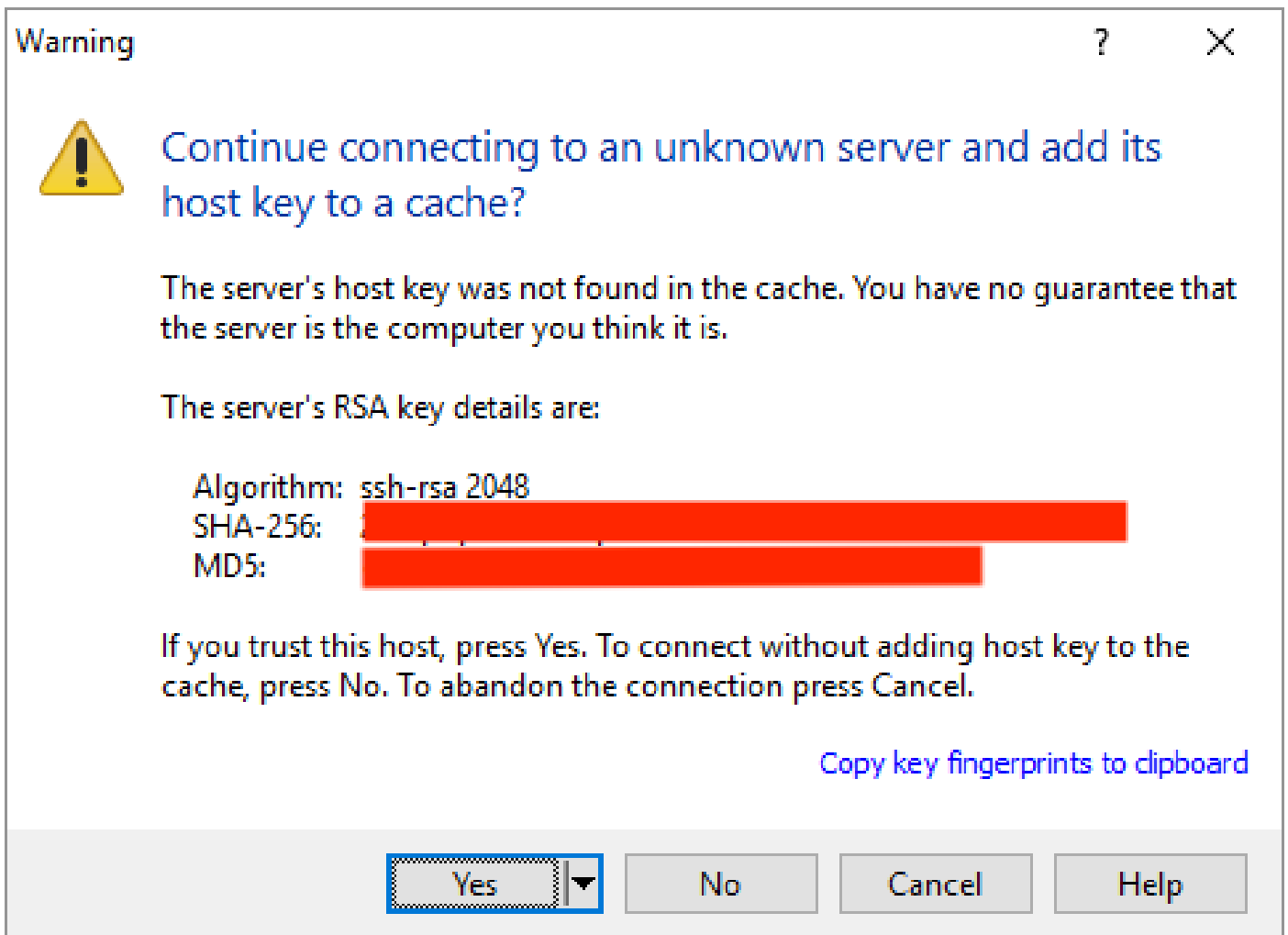
f. Après réception de **DH\_Exchange\_Reply**, le client calcule le hachage de la même manière et le compare au hachage reçu, en le décryptant à l'aide de la clé publique RSA du serveur.

g. Avant de déchiffrer le HASH reçu, le client doit vérifier la clé publique du serveur. Cette vérification s'effectue au moyen d'un certificat numérique signé par une autorité de certification (AC). Si le certificat n'existe pas, c'est au client de décider d'accepter ou non la clé publique du serveur.



Remarque : lorsque vous accédez pour la première fois à SSH sur un périphérique qui n'utilise pas de certificat numérique, vous pouvez rencontrer une fenêtre contextuelle vous demandant d'accepter manuellement la clé publique du serveur. Pour éviter de voir cette fenêtre contextuelle chaque fois que vous vous connectez, vous pouvez choisir d'ajouter la clé d'hôte du serveur à votre cache.

---



Clé RSA du serveur

4. Puisque le secret partagé est maintenant généré, les deux terminaux l'utilisent pour dériver ces clés :

- Clés de chiffrement
- IV Keys (Clés IV) : nombres aléatoires utilisés comme entrée dans des algorithmes symétriques pour améliorer la sécurité.
- Clés d'intégrité

La fin de l'échange de clés est signalée par l'échange du `NEW KEYS` message, qui informe chaque partie que tous les messages futurs seront chiffrés et protégés à l'aide de ces nouvelles clés .

```
346 6.330368 10.106.51.72 10.65.54.8 SSHv2 70 Server: New Keys
347 6.365552 10.65.54.8 10.106.51.72 SSHv2 70 Client: New Keys

> Frame 346: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: Cimsys_33:44:55 (00:11:22:33:44:55), Dst: Cisco_3c:7a:00 (00:05:9a:3c:7a:00)
> Internet Protocol Version 4, Src: 10.106.51.72, Dst: 10.65.54.8
> Transmission Control Protocol, Src Port: 22, Dst Port: 56127, Seq: 1444, Ack: 1581, Len: 16
✓ SSH Protocol
  ✓ SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
    Packet Length: 12
    Padding Length: 10
  ✓ Key Exchange
    Message Code: New Keys (21)
    Padding String: 000000000000000000000000
```

Nouvelles clés client et serveur

5. La dernière étape est la demande de service. Le client envoie un paquet de demande de

service SSH au serveur pour lancer l'authentification de l'utilisateur. Le serveur répond avec un message d'acceptation de service SSH, invitant le client à se connecter. Cet échange se produit sur le canal sécurisé établi.

## Informations connexes

- <https://www.cisco.com/c/en/us/support/docs/security-vpn/secure-shell-ssh/4145-ssh.html>
- <https://datatracker.ietf.org/doc/html/rfc4253>
- [Assistance technique de Cisco et téléchargements](#)

À propos de cette traduction

Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.