Sockets en IPv6

Généralité

En langage C

Utilisation de ssl

Programmation IPv6

- Non compatibilité des programmes
  - Mémoire 4 octets pour IPv4, 16 octets pour IPv6

- Pas de changement pour les langages qui utilisent des couches d'abstraction et qui ne référencient pas les adresses IPv4 directement (Java)

- Pour les autres, les changements sont minimisés….

- Les API restent identiques :
  - socket() utilise AF_INET6
  - bind() connect() accept()
  - send() recv()

Programmation IPv6 en C (1)

- Quelques changements !!
  - **struct sockaddr_in6 {**
    u_char sin6_family; ← domaine
    u_int16m_t sin6_port; ← n° port
    struct in6_addr sin6_addr; ← @IP
    u_int32m_t sin6_flowinfo; … } ← id de flux
  - Pour les conversions, utilisation de *struct sockaddr_storage*
    - Permet le mappage du sockaddr_in
    - Permet le mappage du sockkadr_in6
  - Affectation de l'adresse IP
    - Si serveur, l'adresse d'écoute vaut in6addr_any
      d'où
      memcpy((void *)&adr6.sin6_addr, (void *)&in6addr_any,
      sizeof (in6addr_any));

Programmation IPv6 en C (2)

- Disparition de gethostbyname et de gethostbyaddr
  - **Remplacement par getaddrinfo() et getnameinfo()**

```
int getaddrinfo(
        const char *nodename,          // nom d'une machine
        const char *servname,          // nom du service ou n° port
        const struct addrinfo *hints,  // filtre
        struct addrinfo **res);        // liste de résultats possibles


struct addrinfo {
        int ai_flags;                  // AI_PASSIVE, AI_NUMERICHOST,…
        int ai_family;                 // AF_INET, AF_INET6, AF_UNSPEC….
        int ai_socktype;               // SOCK_...
        int ai_protocol;               // 0 ou IPPROTO_xx pour ipv4 et ipv6
        size_t ai_addrlen;             // taille de l'adresse binaire ai_addr
        char * ai_canonname;           //le FQDN
        struct sockaddr * ai_addr;     // l'adresse binaire
        struct addrinfo *ai_next;      // liste chainée sur la structure
                                       ( cf man getaddrinfo() )
```

Programmation IPv6 en C (3)

getnameinfo -> traduction adresse IP vers nom

```
int getnameinfo(
        const struct sockaddr *sa,          // l'adresse IP connue
        socklen_t salen,                    // sa taille
        char *host, size_t hostlen,         // résultat pour le nom
        char *serv, size_t servlen, int flags);      // résultat pour le service
```

- Autres fonctions possibles :
  - inet_ntop et inet_pton qui permettent de convertir une adresse binaire en texte et vice –versa

```
        char * inet_ntop (int af,    // Af_INET ou AF_INET6
                          const void *src,        //adresse binaire
                          char *dst,              //adresse du résultat
                          size_t size)            // taille du tampon
```

  - getifaddrs (struct ifaddrs *) -> permet de récupérer les interfaces réseaux

Sécurisation d’un réseau

Généralité

Sécurité d'un système appartenant à un réseau

Techniques

Généralité

- 4 mots clés et 1 concepts
  - Authentification
  - Confidentialité
  - Intégrité
  - Disponibilité
  - Non-répudiation

Est-ce que TCP/IP respecte ces critères ?

NON

- Aucune vérification sur l'adresse IP source, ni sur le chemin parcouru
- Buffer de réception de dimension finie…

Objectifs, moyens, techniques

- *Objectifs de la sécurité*
  - Sécurisation des systèmes
  - Sécurisation des accès
    - pour protéger les systèmes
    - pour protéger les informations transmises
  - Sécurisation des échanges

- *Moyens*
  - définir une politique de sécurité (PSSI)
    - identifier les entités
    - définir quelle entité a le droit de faire quoi
    - définir ce qui est interdit
  - installer la politique définie
  - surveiller les systèmes
  - participer à la surveillance des réseaux

Techniques pour la sécurité

- chiffrement
- protocoles sécurisés
- restreindre les accès
- Anti-virus, patches, durcissement config,
- etc...

Pb : les failles de sécurité proviennent à 80% de l'intérieur de l'entreprise

Sécurité sur un système isolé

Objectifs

- éviter que des entités effectuent des actions non autorisées
  - éviter les processus interdits
  - éviter que des processus autorisés n'effectuent des opérations interdites

Définir une politique de sécurité et la valider

- qu'est ce qui est autorisé / interdit à chaque entité ?

Installer la politique définie

- identifier qui tente de faire quoi
- spécifier les autorisations et les interdictions
- dans un système d'exploitation qui le permet

attribuer des comptes
droits sur les programmes
droits sur les données

Vérifier la politique installée

- journaliser
- alerter

niveau de sécurité d'un système
Common Criteria (CC)
voir ANSSI `www.ssi.gouv.fr`

Sécurité sur un système appartenant à un réseau

Rappel : un système ne peut être attaqué que s'il a un processus serveur en attente de demande (*port en état listen* → commande netstat –an).

- *Configuration des services*
  - quels services sont nécessaires ? pour quels clients ?
  - droits des services (UID, GID, ...)
  - choix des implémentations

- *Contrôle des services actifs*
  - à partir de la configuration du système
  - en examinant les services en attente, de l'intérieur (**netstat**) ou de l'extérieur du système (**nmap**)

- *Contrôle des demandes de service*
  - filtrage des demandes avant de les livrer aux entités serveurs
  - journalisation des demandes
  - alertes en cas de demandes interdites ou de tentatives malveillantes

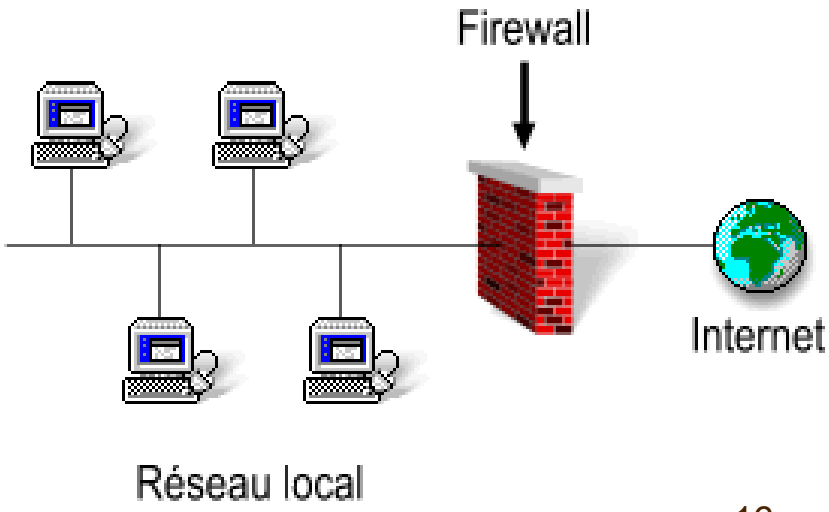
CERT *Computer Emergency Response Team* `www.cert.org`

Le Pare-feu (1)

- La parefeu (firewall)
  - permet de protéger le réseau interne de l'extérieur
  - utilise des règles définies par la politique de sécurité
  - point de passage obligatoire des données
  - Architecture généralement logicielle

Rôle principal : **FILTRAGE**

- au niveau IP
- au niveau Transport
- quelque fois au niveau applicatif : proxy



Le pare-feu (2)

- Restriction
  - Ne gère pas les communications sur le réseau interne
  - Ne protège pas contre les virus
  - Ne peut voir que le trafic qui passe par lui
  - Ne peut se configurer tout seul !!!

- Avantage
  - Journalisation du trafic

Actuellement, politique de tout interdire et de n'ouvrir que les services utiles.

Sécurité d'un réseau (1)

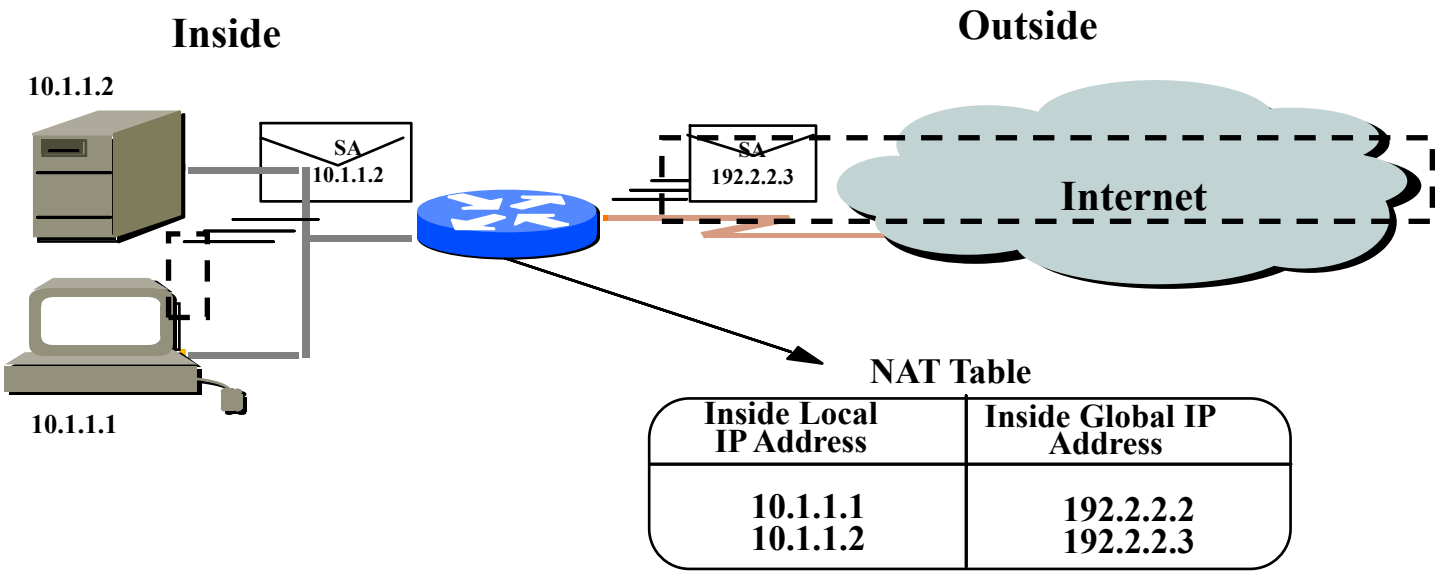
- Au niveau Réseau (pour un réseau isolé par un équipement)
  - identification du réseau et des systèmes qui le composent
    - **adresses privées (RFC 1918)**
      - **Classe « A » : 10.0.0.0/8**
      - **Classe « B » : 172.16.0.0 à 172.31.0.0/16**
      - **Classe « C » : 192.168.0.0 à 192.168.255.0/24**
  - protection par un équipement actif (routeur, pare-feu,…)
    - règles de filtrage sur la source, la destination ou les deux (fonction pare-feu)
    - traduction d'adresses (**IP-masquerading)**
      - **Network Address Translation (statique ou dynamique),**
      - **Port Address Translation,**
        - Attention, il peut être nécessaire de réécrire le message du fait du changement d '@IP.
        - Pas de chiffrement au niveau IP

Sécurité d'un réseau (2)

Le NAT statique = association de n adresses avec n adresses.

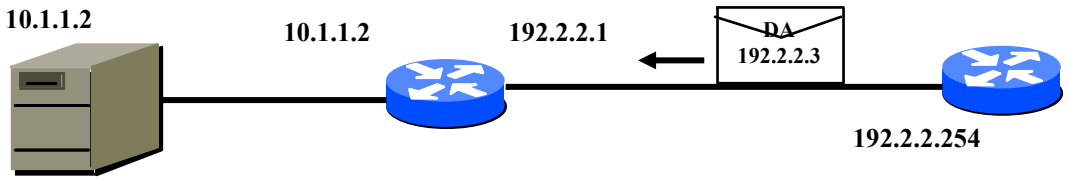
C'est à dire qu'à une adresse IP interne, on associe une adresse IP externe.

Rôle du système: remplacer l'adresse de la station du réseau par une adresse externe (publique).



Sécurité d'un réseau (3)

Problème : comment retrouver le chemin de retour ?



NAT Table

Inside Local IP Address	Inside Global IP Address
10.1.1.1	192.2.2.2
10.1.1.2	192.2.2.3

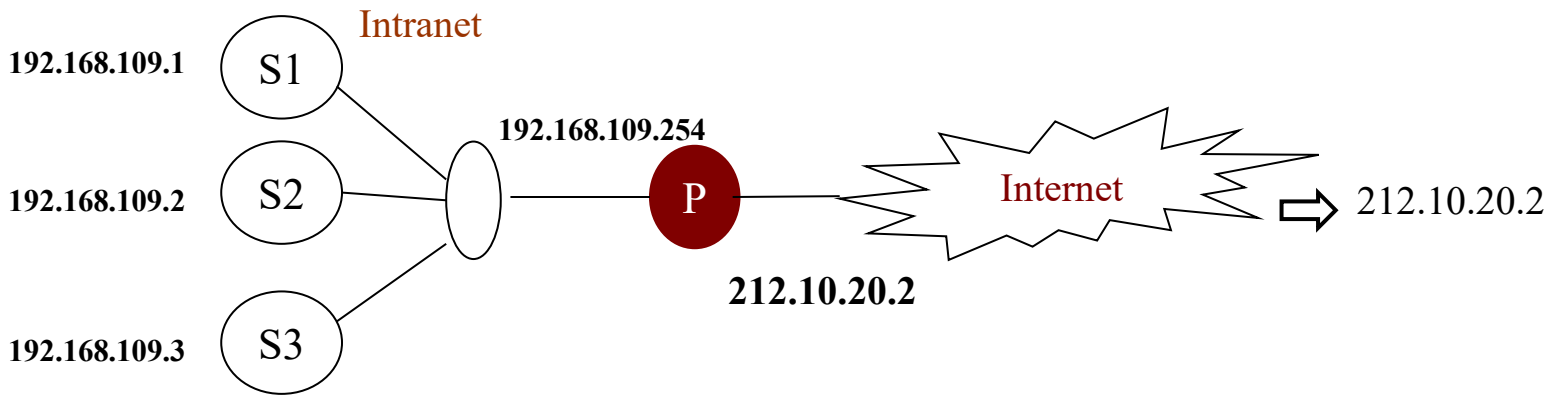
Requête ARP sur le 192.2.2.3 …
Résultat ???

Solution : le Proxy-arp

Le « routeur » renvoie son adresse MAC pour toutes les adresses contenues dans la table NAT.

En général, les pare-feux ou les routeurs ont cette fonctionnalité

Sécurité d'un réseau (4)



1 seule adresse est disponible pour envoyer des paquets IP vers Internet (ex: adsl)

Pb: Si plusieurs stations appartiennent au réseau local, comment peuvent-elles envoyer des PDU-IP vers l'internet et comment les différencier ?

=> Utilisation du NAT dynamique

Sécurité d'un réseau (5)

2 cas possibles :

- **1 seule adresse IP de sortie -> PAT**
- **n adresses de sortie pour m ordinateurs (m > n)**
  -> IP masquerading ou PAT

Fonctionnement :

- *En Masquerading*, traduction automatique de l'adresse IP de la station émettrice avec une adresse IP possible de la Passerelle (routeur, proxy)
- *En PAT*, traduction automatique de l'adresse IP de la station émettrice avec l'adresse IP de la Passerelle (routeur, proxy) et translation du port.

La translation du port permet de différencier les stations qui utilisent la même adresse IP Passerelle.

Application possible : mini-réseau derrière un routeur ADSL
➔ pb : adresse IP statique/dynamique

Sécurité d'un réseau (6)

- Sonde IPS /IDS
  - IDS **I**ntrusion **D**etection **S**ystem
  - IPS **I**ntrusion **P**revention **S**ystem

➢ Chargés d'analyser le trafic réseau pour y **détecter des tentatives d'intrusion** :

- soit en analysant le comportement des flux réseaux ;
- soit en se basant sur une base de signatures identifiant des données malveillantes (principe similaire à celui des anti-virus).

➢En cas de détection d'une intrusion :

- Les **IDS alertent** les administrateurs, libre à eux d'intervenir ou non ;
- Les **IPS bloquent** les flux réseau concernés.

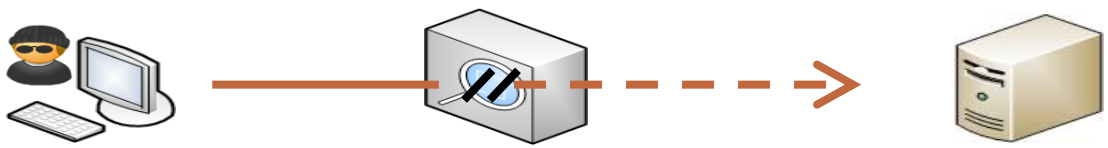
Sécurité d'un réseau (7)

➢Les IDS/IPS demandent une configuration fine et maintenue :

- Ils sont en effet connus pour présenter de nombreux faux-positifs (i.e. ils détectent à tort une tentative d'intrusion) → couplage possible avec SIEM
- De plus, les IDS/IPS basés sur des signatures ne peuvent détecter que les intrusions dont les caractéristiques *techniques sont déjà connues et référencées.*
- *Mise en place du Deep learning !!!*

➢Un IDS peut être soit en coupure du flux réseaux, soit **positionné en écoute.**

➢Un IPS **doit forcément** être en **coupure du flux** de façon à pourvoir bloquer le trafic lorsque cela est nécessaire.



Sécurité d'un ensemble de réseaux

- Contexte
  - une organisation possède plusieurs réseaux distants les uns des autres
    - interconnexion par un médium "privé" : cher, mais sûr
    - interconnexion par l'Internet public : économique, mais non sûr

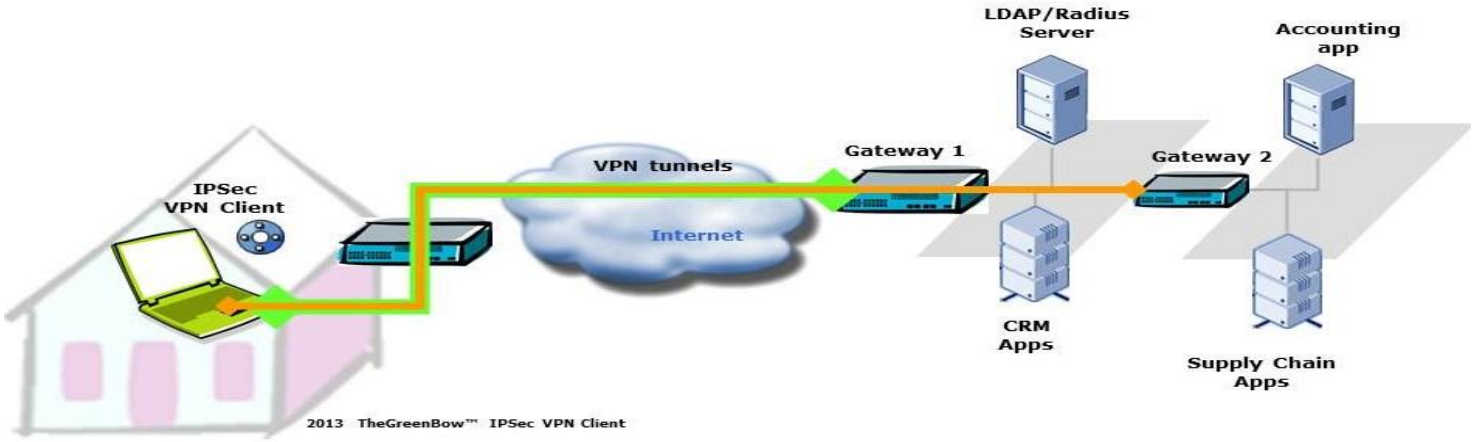
- une solution : le *tunnelling*
  - les PDU sont transportées encapsulées entre 2 réseaux grâce à un Service
    "On intègre la machine distante dans le réseau de l'entreprise"
  - les extrémités sont identifiées

VPN

Virtual Private Network

Un VPN est une connexion sécurisée et chiffrée entre deux réseaux ou entre un utilisateur individuel et un réseau.

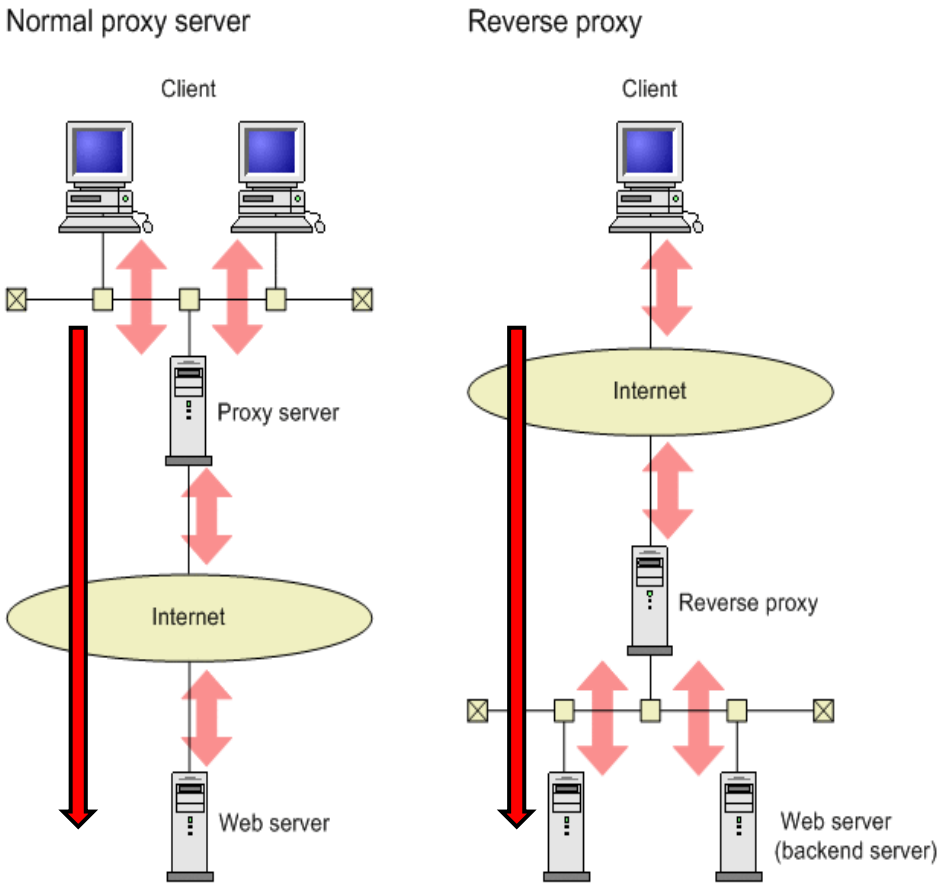
- Nécessite un serveur VPN
  - Le client VPN se connecte sur le serveur VPN et reçoit une nouvelle adresse IP pour communiquer avec ce serveur
  - Les communications passent par ce serveur avant de retourner sur Internet



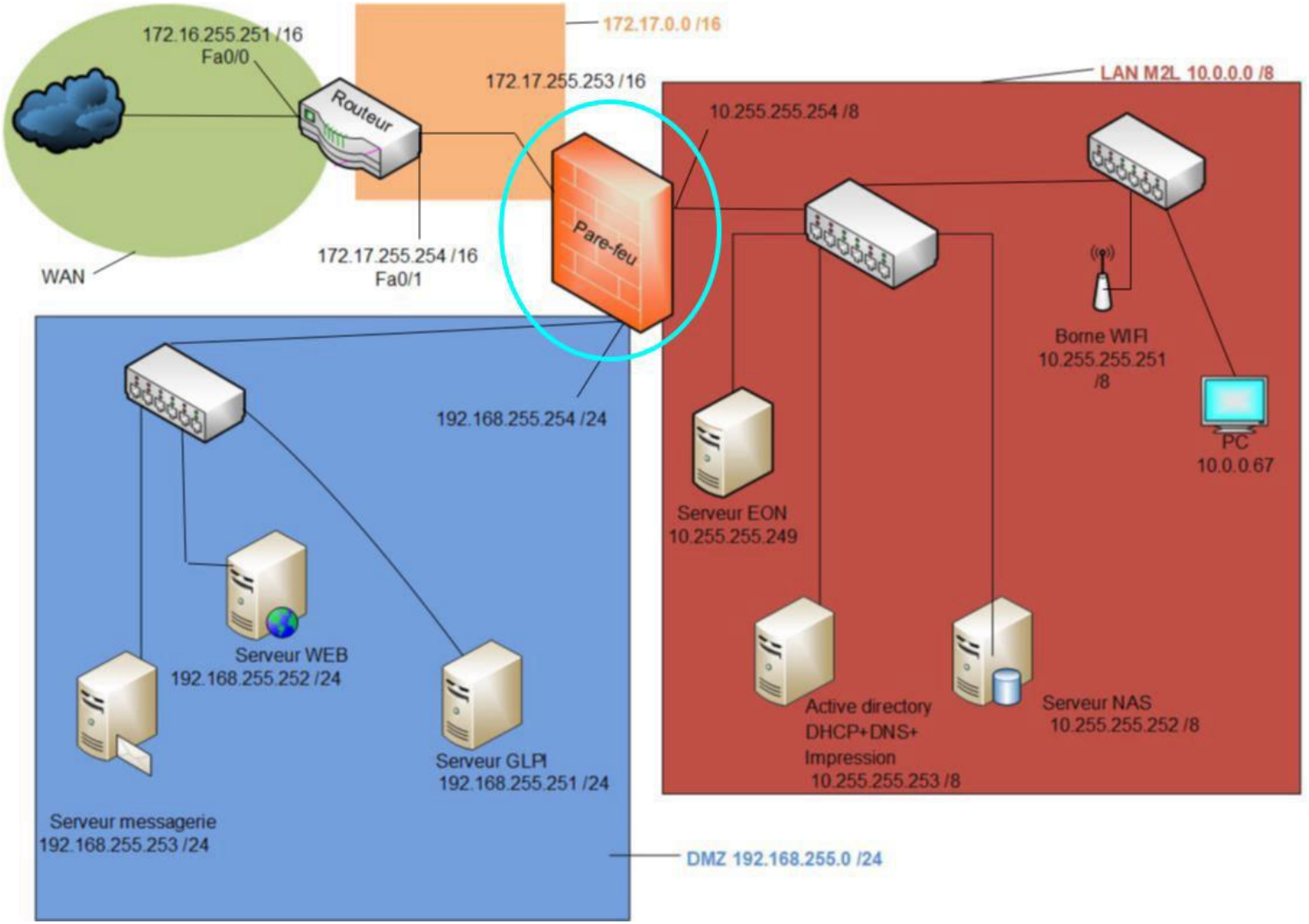
Proxy et reverse proxy

- Proxy : *composant logiciel servant d'intermédiaire entre la source et la destination*
  - Filtrage
  - Cache
  - Etc…

Le reverse proxy permet de faire aussi:
- load balancer



Architecture (avec DMZ)



Chiffrement : principes

Définitions

- chiffrement $M \xrightarrow{E_k} C$
- déchiffrement $C \xrightarrow{D_{k'}} M$

E	algorithme de chiffrement
D	algorithme de déchiffrement
k	clé de chiffrement
k'	clé de déchiffrement

Propriétés (souhaitées) d'un cryptosystème

- $D_{k'}(E_k(M)) = M$ où les clés k et k' sont associées
- $D_{k'}$ et E_k dépendent totalement ou partiellement d'informations secrètes
- les algorithmes doivent être économiques : processeur, mémoire, taille de code
- le secret doit reposer sur les clés plutôt que sur les algorithmes
  - *algorithme public - > qualité meilleure*
- le calcul de k' doit être très difficile, même si on connaît C et M
- $D_{b'}(E_a(M))$ doit être une information non valide

Chiffrement : cryptosystèmes

A chiffre symétrique (k = k')

- économique
- problème de la gestion des clés

- **DES** *(Data Encryption Standard –fin en 2001)*
- **Triple-DES**
- **IDEA** *(International Data Encryption Algorithm)*
- **AES** *(Advanced Encryption Standard)*

A chiffre asymétrique (k ≠ k')

- $D_{k'}(E_k(M)) = E_k(D_{k'}(M)) = M$
- peu économique

- **DH** *(Diffie-Hellman)*
- **RSA** *(Rivest, Shamir, Adleman)*

$E_k(M) = M^e$ modulo n k = {e,n}
$D_{k'}(C) = C^d$ modulo n k' = {d,n}

n = p*q
p et q premiers entre eux
e premier avec (p-1)*(q-1)
d*e = 1 modulo ((p-1)*(q-1))

Hachage ou signature

- pour créer des empreintes d'information (*digest*)
- algorithmes analogues à ceux du chiffrement
- pas de déchiffrement possible

MD5 *(Message Digest)*
SHA-256 *(Secure Hash Algorithm)*

Chiffrement : asymétrique

3 éléments essentiels :

Nécessité de deux clés -> créer en même temps par la même personne

- Clé privée : comme son nom l'indique, cette clé est personnelle et connue de son seul propriétaire
- Clé publique : clé distribuée à tout le monde permettant de chiffrer un message

clé privée (clé publique (M)) = clé publique (clé privée ((M)) = M

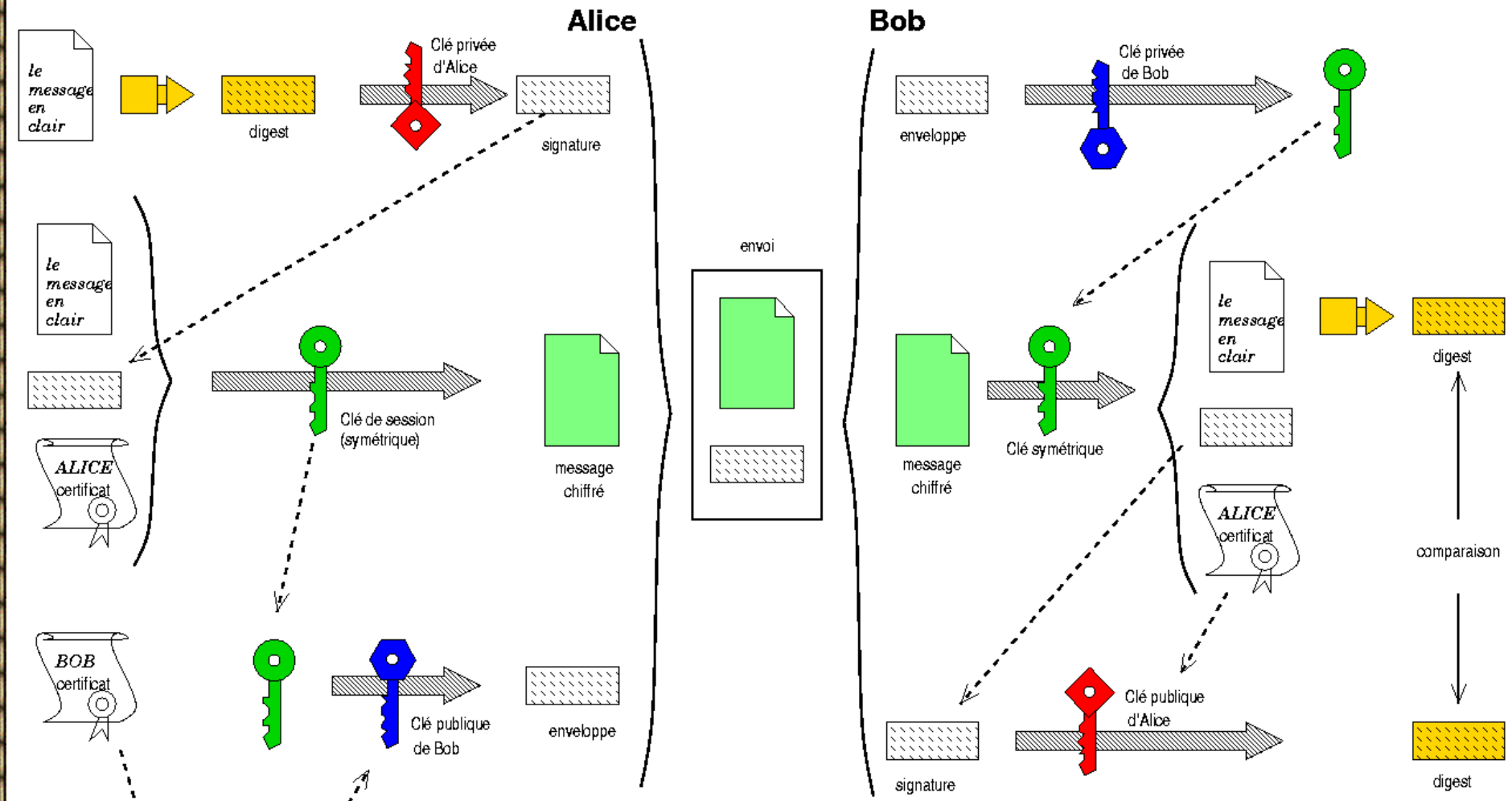
- Certificat : donner par une autorité de certification

-> permet de s'assurer qu'une clé publique appartient bien au propriétaire annoncé.

Chiffrement : utilisations

- chiffrement avec une clé secrète partagée entre 2 entités — authentification, confidentialité, intégrité, non répudiation
  - seules les 2 entités peuvent déchiffrer
- chiffrement avec la clé publique d'une entité — confidentialité
  - seule l'entité possédant la clé privée associée à la clé de chiffrement peut déchiffrer
- chiffrement avec la clé privée d'une entité — authentification, intégrité, non répudiation
  - tout le monde peut déchiffrer avec la clé publique associée
  - si la tentative de déchiffrement produit un résultat valide
    - preuve de la source
    - l'information est intègre
    - seule l'entité a pu chiffrer
- distribution des clés — domaine de confiance
  - certificats
    - preuve de possession
    - contient la clé publique
    - délivré par une autorité de certification

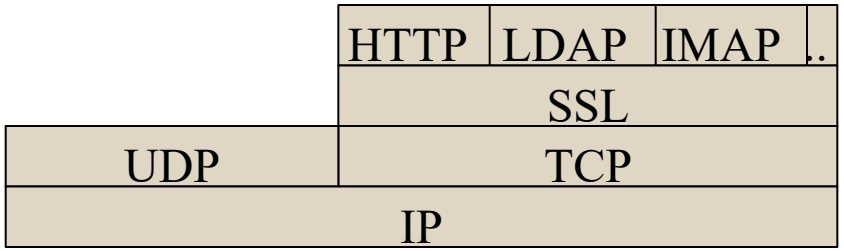
Chiffrement : exemple résumé



SSL /TLS (1)

- SSL (Secure Socket Layer) assure :
  - l'authentification du serveur
  - la confidentialité des données
  - l'intégrité des données
  - (optionnel) l'authentification du client

- Transparent pour l'utilisateur
  - Chiffrement seulement des données
  - Se situe entre la couche TCP et la couche applicative



SSL /TLS (2)

- SSLv1 -> juillet 1994 par Netscape (jamais utilisé)

- SSLv2 -> fin 1994, intégré à Netscape Navigator
  en mars 1995 → apparition du **https**

- SSLv3 -> novembre 1995

- TLS (Transport Layer Security)
  -> normalisé par l'IETF
  -> RFC 2246, en 1999
  -> basé sur SSLv3, mais avec de petits changements, donc incompatibilité

  actuellement TLS v1.3 possible

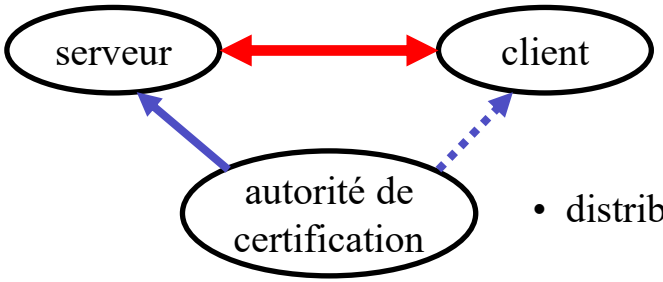
SSL /TLS(3)

- SSL est composé de deux étapes
  - SSL Handshake
    - Échange des informations pour le chiffrement (longueur de clé, protocoles utilisés, etc..)
    - Utilisation de certificats et de clés publiques pour échanger une clé de session symétrique
  - SSL Record
    - Échange des données chiffrées par la clé de session
    - Impossible à décrypter même si les échanges précédents ont été récupérés

Pré-requis serveur : Certificat + clé publique/privée

SSL/TLS (4)

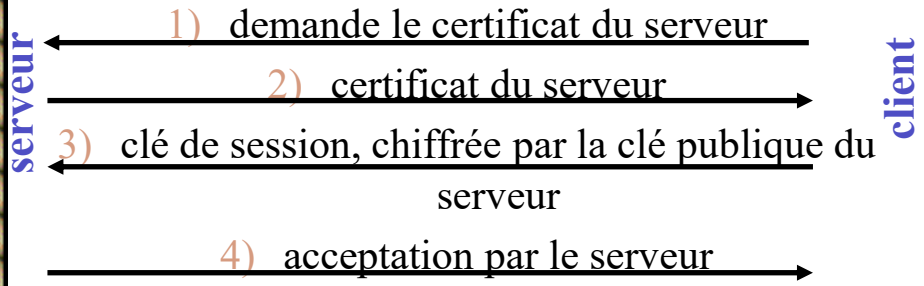
- fournit des services
- possède un certificat



- achète des services
- possède ou non un certificat

- distribue des certificats

établissement de connexion ssl



transmission d'informations

- non chiffrée
- chiffrée avec la clé publique du serveur
- chiffrée avec la clé de session

- compression éventuelle
- Hachage du message envoyé

problèmes

- le client n'est certifié qu'optionnellement ⇒ risque pour le serveur
- si le client est certifié, il ne peut pas être anonyme ⇒ risque pour le client
- pas de contrôle de validité des certificats entre délivrance et fin de validité !!
- autorités de certification…

Sécurité Wi-FI

Protection par défaut : le WEP

- Utilise un chiffrement pour les données

- l'implémentation WEP (Wired Equivalent Privacy) (clé sur 40 bits / 104bits) donnée par les utilisateurs auquel est rajouté un vecteur d'initialisation (24 bits).

Fonctionnement : chiffrement RC4 en utilisant la clé + vecteurs d'initialisation (IV)

message envoyé = (M.c(M)) xor RC4(IV . K)

c(M) = cheksum de M et K = clé

le RC4 donne des séquences pseudo-aléatoires

Le vecteur d'initialisation change à chaque trame envoyée, on lui rajoute 1

(assez facilement crackable si on connaît le 1er octet de M et IV)

Pb : faiblesse d'implémentation dans IV commencent à 0 puis incrémentés de 1 à chaque envoi, vecteurs faibles ….

Actuellement, quelques dizaines de minutes pour cracker clé WEP

Ne pas utiliser le WEP

Sécurité actuelle , norme 802.11i = Wifi Protected Acces (WPA) ou WPA2.

(faille découverte en octobre 2017)

Protection des logiciels/ matériels

- **But : prendre la main sur un ordinateur**
  - Avoir accès à l'ordinateur
    - Connexion à internet
    - Avoir des ports ouverts (présence de programmes serveurs)
    - Utilisation de port USB….
  - Infiltration
    - Faille du 0-day
    - Phishing
    - Virus, malware, worm, rootkit …
    - Sql Injection, xss (cross-site scriping),…
  - Mise en attente du virus jusqu'à son activation via un Command and Control
  - Contre-mesure : sonde IDS et IPS
  - Quelques virus :Tchernobyl, stuxnet, zeus, spyeye, Wannacry,…

Aide

- ANSSI

- Lutte contre le téléchargement illégal : HADOPI

Haute Autorité pour la Diffusion des œuvres et la protection des droits sur Internet.

Comment cela fonctionne-t-il ?

Dissous au 1er janvier 2022 → repris par l'ARCOM (Autorité de régulation de la communication audiovisuel et numérique)

Différent web

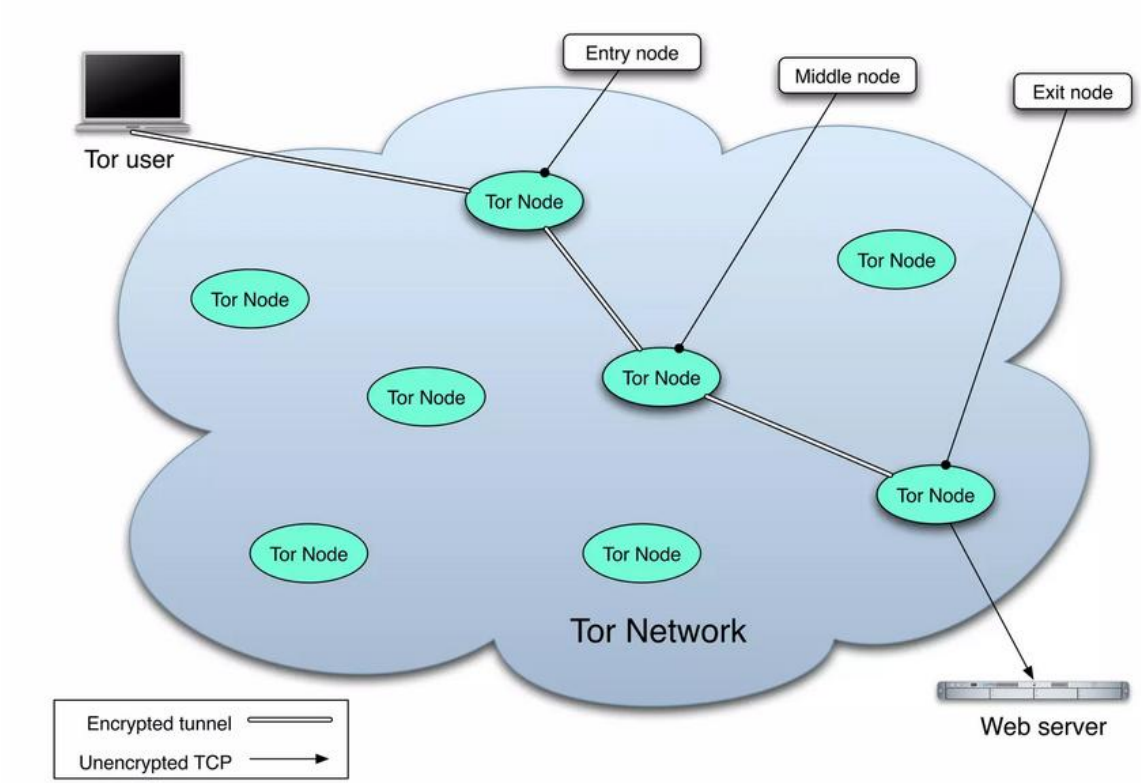
TOR

TOR (1)

- Tor
  - Logiciel libre
    - Permet de se connecter à des machines sur Internet via des relais d'une manière anonyme
  - Objectif
    - Ne pas pouvoir être tracé
    - Accéder à des informations en laissant moins de traces
    - Pour éviter les censures
    - Etc…
  - Moyen
    - Chiffrement asymétrique
    - Passage par minimum 3 relais

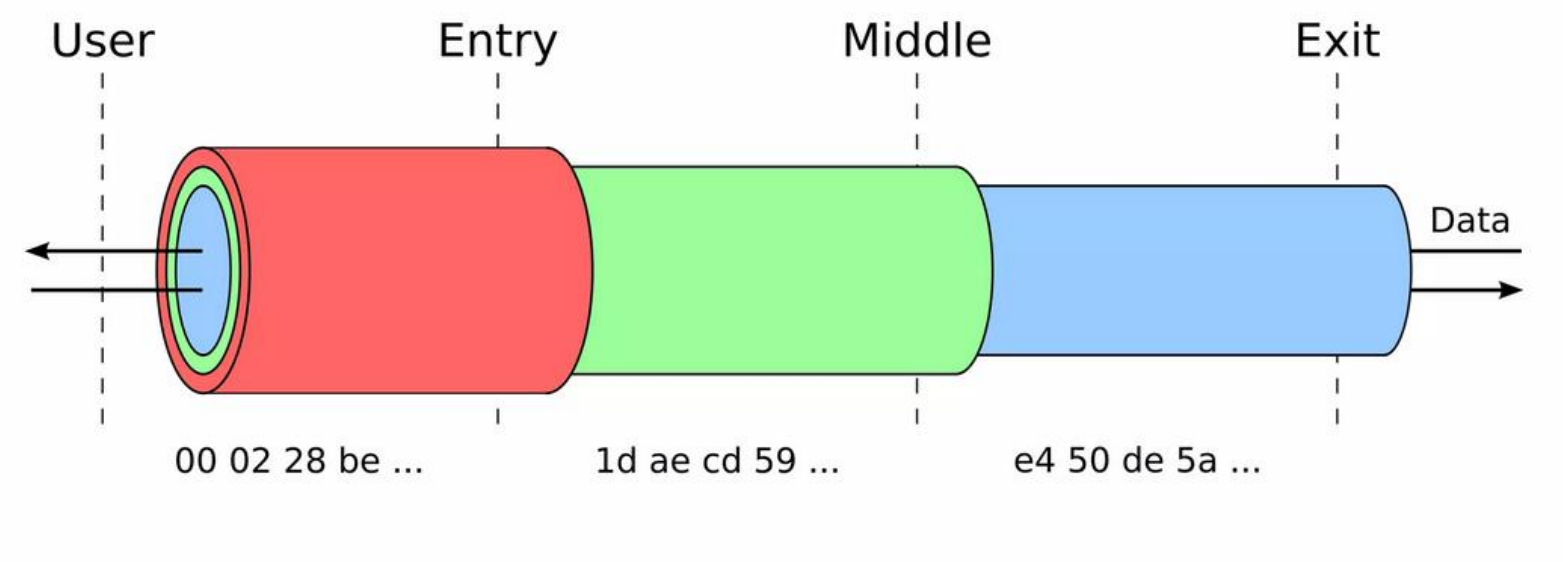
TOR (2)

- L'utilisateur demande 3 clés (AES) pour chacun des relais
- Encapsulation des 3 messages
- La sortie n'est pas chiffrée !!!



TOR (3)

- Autre appellation : routage en oignon



Une adresse onion se compose de 56 lettres et chiffres, suivis de ".onion".

Ex : ***s4k4ceiapwwgcm3mkb6e4diqecpo7kvdnfr5gg7sph7jjppqkvwwqtyd.onion***
torchdeedp3i2jigzjdmfpn5ttjhthh5wbmda2rr3jvqjg5p77c54dqd.onion