Sécurité des Echanges

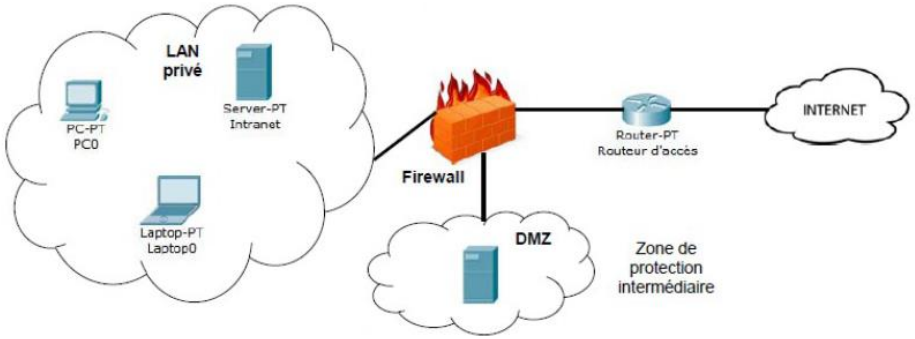
P. Laurençot

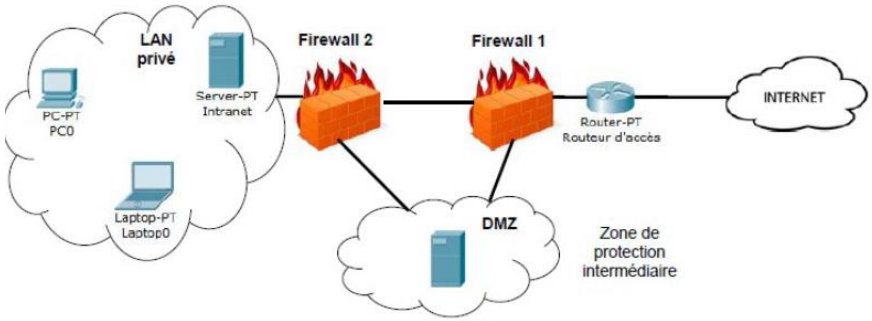
Plan d'ensemble

- Rappel
  - Architecture d'un réseau
  - Clé symétrique /asymétrique
    - Diffie –Hellman
  - Hachage
- IGC (PKI)
- Ipsec
  - Ikev1
  - Ikev2
- Kerberos

Architecture (1)

- DMZ: est un sous-réseau séparé du réseau local et isolé de celui-ci et d'internet par un parefeu. Ce sous-réseau contient des machines étant susceptibles d'être accédées depuis internet. (wikipédia)
  - Possibilité d'attaques…
  - Plus ou moins de protection suivant l'architecture utilisée

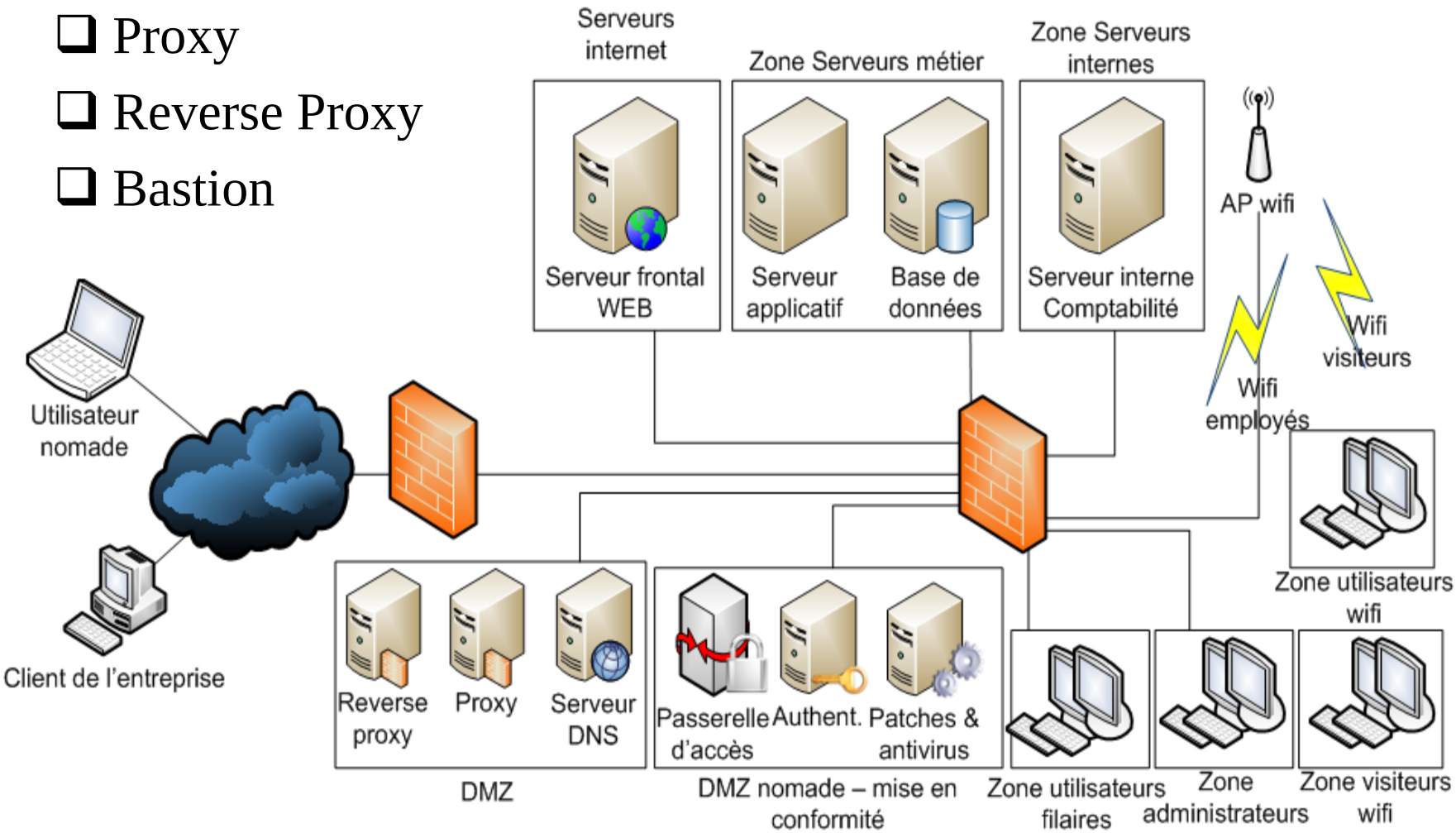




- Utilisation adresse IP privée
  - Mise en place de NAT

Architecture (2)

- VLAN
- Proxy
- Reverse Proxy
- Bastion



Le chiffrement

- *Objectif*
  - Assurer l'authentification, la confidentialité et l'intégrité des données échangées
  - La non-répudiation est aussi abordée
  - Vocabulaire : chiffrer, déchiffrer, décrypter
  - Principe de Kershoffs

    La sécurité d'un système de chiffrement n'est pas fondée sur le secret de la procédure qu'il suit mais sur un paramètre utilisé lors de sa mise en œuvre, la **clé.**

Clé symétrique (1)

- Même clé pour le chiffrement /déchiffrement
  - Utilisation du terme : clé secrète
- En général, émetteur et récepteur possèdent la clé
  - Pour n équipement, nb = n*(n-1)/2
- Avantage
  - Chiffrement très sûr (dépend de la longueur de la clé)
  - Rapide
- Désavantage
  - ***Distribution de la clé***
  - Pas de non-répudiation (le destinataire peut avoir lui-même signé son message, pas de preuve)
  - Difficile à mettre en œuvre pour grand groupe

Clé symétrique (2)

- Quelques exemples
  - DES (Data Encryption Standard)
    - Clé sur 56 bits
    - 5 modes (ECB, CBC, CFB, OFB, CTR)
      - Electronic Code Book: chiffre 16 fois un bloc de 64 bits
      - Cipher Block Chaining : utilisation d'un iv (vecteur d'initialisation) xor (iv, texte), puis ECB, puis xor (bloc chiffré, bloc suivant), ECB, etc pb si 1 bit corrompu dans le transfert
      - Cipher Feedback : ECB (iv), puis xor (clé chiffré, texte clair), puis ECB(texte chiffré), puis xor (texte chiffré, texte clair),…
      - Output Feedback : ECB(iv), puis xor(clé chiffré, texte clair), puis ECB (iv chiffré), etc..
      - Pb : longueur de clé insuffisante actuellement, calcul long
  - Triple DES - Clé de 168 bits ou 112 bits
    - E (K1, E(K2, E (K3,P))) ou E(K1, E(K2, E(K1,P)))

Clé symétrique (3)

- IDEA (International Data Encryption Algorithm)
  - Utilisation 1 clé de 128 bits
  - Transformé en 52 clés de 16 bits, puis ECB ou CBC ou CFB
  - Utilisé dans PGP (Pretty Good Privacy)
- AES (Advanced Encryption Standard) / Rijndael
  - Clé 128 bits (10 rounds), 192 (12 rounds), 256 (14 rounds)
  - Utilisation matrice, xor, etc..;
  - Le plus sûr actuellement
- RC2, Skipjack, rc4, rc5, blowfish, twofish,…

Algorithme de Diffie-Hellman

- Permet de faire transiter un nombre secret, sur un canal non sécurisé
  - A et B choisissent deux nombres, qui passent en clair sur le réseau
    - P : grand nombre premier
    - G : grand nombre (appelé générateur)
  - A choisit un nombre secret x, B un nombre secret y
  - A calcule : pA = G^x mod P, et envoie pA sur le réseau
  - B calcule : pB = G^y mod P, et envoie pB sur le réseau
  - A calcule Ka= pB^x mod P, et B calcule Kb= pA ^y mod P

du fait de preuve mathématique sur les groupes, Ka = Kb

⟹ on prend cette clé

Impossible de calculer x ou y même si on a P, G, pA, pB

Pb inversion d'exponentiation très long

Clé asymétrique

- Clé de chiffrement ≠ clé de déchiffrement
- Création simultanée de la bi-clé
  - Une clé publique (diffusion à tous)
  - Une clé privée (personnel)
  - Impossible de déterminer l'une à partir de l'autre
- Avantage
  - Chiffrement très sûr (dépend de la longueur de la clé)
  - Pas besoin d'utiliser un secret
  - Facile à révoquer, et non répudiation si utilisation clé privée ok
- Désavantage
  - ***Peu performant***
  - Qui se trouve de l'autre côté ?

RSA (1)

- Rivest, Shamir, Adleman : 1978
  - Basé sur le petit théorème de Fermat ($a^{(p-1)} \equiv 1 \bmod p$)
  - Et sur la décomposition en nombre premier
  - Utilisé par de nombreux logiciels (web)
- Principe
  - P et q , 2 nombres premiers grands
  - n= pq
  - On choisit e, nombre premier avec (p-1)(q-1) et e < n
  - Clé publique : (n, e)
  - Calcul de la clé privée : d, inverse de e modulo (p-1)(q-1)
    ed = 1 mod (p-1)(q-1) (algorithme d'Euclide étendu)
  - On a alors : $C = M^e \bmod n$ et $M = C^d \bmod n$
- Autre : courbe elliptique, El Gamal algorithm,…

RSA (2)

- Exemple
  - Alice prend p=3 et q=11 -> n=33
  - (p-1)(q-1)= 20 -> e=3 clé publique (33,3)
  - Message z= 26

    C = (26)^3 mod 33 = 20
  - Clé privée : 3*7 = 21 -> 3*7=1 (mod 20)
    - Clé privée : (33,7)
    - M = 20^7 mod (33) -> 26

RSA (3)

- Utilisation de grandes clés
- Attaque par brute force
  - Objectif : trouvé les nombres p et q
  - Très très long
    (faisable si clé < 800 bits)
  - Clé RSA : min 1024 bits ,
    mais 2048 bits
- Pb ordinateur quantique
  - Algorithme de Shor

Module (2048 bits) : n

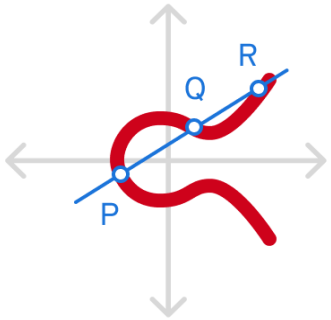
a6 66 91 a4 25 e8 e0 fa 65 db c3 46 96 b6 c7 f9
9a 34 7d ec 8e d6 93 81 54 dd b8 f9 85 aa 00 bd
a1 90 97 ec 72 05 1d 37 23 9c 52 f5 50 60 01 8c
5b 0c aa d3 ca 50 8f 76 11 36 b8 e4 36 e2 da d0
bb f2 5a 84 b9 bf c9 73 6d fc a6 23 bd 37 d3 3d
9f ee 36 e4 05 a1 79 0d 6a 4b 70 99 65 ab 42 0d
70 ae e5 17 b7 8f e5 b7 76 9a 15 9f 80 94 bd 67
25 e7 4c c5 92 86 af 0f b6 0c 5e 5e 9e f8 35 fa
3d 9a c5 a8 99 4b 27 22 f4 8a 15 7e 30 97 d1 c3
1a 18 f5 20 56 e8 a7 0c 04 cb bd fa 15 f4 c2 bd
d4 15 88 88 cc ed af ee 21 6c 80 7a 09 2b 60 66
39 aa 9f 76 ad 56 7a 3f f3 e0 53 12 89 7d 08 db
1e 4c 52 05 be 05 0a 8e a1 c0 a9 19 de 8a 23 0e
5a a4 ef 8e f3 07 06 ab 84 e1 c2 4d bd 92 ff e4
5f 12 d3 c8 5d 59 a6 3b 3e b2 6d ff 0b 75 2b a5
46 4b dc 9a 7a 1e b8 f4 a0 61 a7 62 49 f5 78 11

Exposant (24 bits) : e

65537

ECC

- Cryptographie à Courbe Elliptique (ECDH)
  - Remplace RSA car algorithme et puissance de calcul aide à la factorisation de nombre premier….
  - Equation $y^2 = x^3 + ax + b$ (a et b choisi)
  - Basé sur la problématique du logarithme discret sur courbe elliptique
    - Plus difficile à casser que rsa
    - Plus robuste, même pour les ordinateurs quantiques
  - Clé plus courte
  - Choix public de a et b, puis d'un point P et d'un point Q= n.P ➔ secret n
    - Impossible de trouver n connaissant a,b,P,et Q.



Hachage

- Création d'une empreinte numérique
  - Unique pour un objet
  - Fonction à sens unique
  - Obtention : hash ou condensat ou empreinte sur n bits
  - Problème collision
- Fonctions
  - Md5, sha-1, sha2-256,Sha2-384, sha3-384, sha2-512, …
  - HMAC (Keyed-hash message authentication code)
    - Chiffrement clé secrète + hachage

PKI

- Public Key Infrastructure
- IGC (Infrastructure de Gestion de Clé)
  - Problème :
    - Certifier à qui appartient la clé publique
    - Peut-on faire confiance à celui qui envoie la clé
  - Autorité de certification (Tiers de confiance)
    - Utilisation de certificat (X509)
    - Utilisation d'une autorité d'enregistrement
      - Autorité de validation (mettre à jour les certificats révoqués)
    - Reconnu de confiance par tous
    - Possède une bi-clé

Certificat x509 (1)

- Structure
  - Version
  - Numéro de série (venant de l'ac)
  - Algorithme de signature du certificat
  - *Issuer name* (nom de l'émetteur du certificat)
    - Contient le Distinguished name de l'émetteur
    - Forme X.501 : CN common name, O organisation, OU organisation unit, C country, E email….
  - Période de validité
  - *Distinguished name* du demandeur (celui à qui appartient la clé publique) ou *subject*
  - Clé publique + algorithme à utiliser
  - Signature de l'AC

(hachage de toutes les informations précédentes, puis clé privée)

Certificat X509 (2)

- Extensions -> x509v3
  - BasicConstraints : CA: True permet de signer d'autre certificat
  - Keyusage : utilisation du certificat
    - digitalSignature, nonRepudiation, dataEncipherment, …
  - ExtendedKeyusage : précise le cadre d'utilisation
    - serverAuth, clientAuth, emailProtection,…
  - AuthorityInfoAccess (serveur OCSP),….

- Codage
  - Utilisation d'ASN.1 -> format DER (extension .cer) ou .pem (bin64)
  - Possibilité d'échange de clés privées (si ac a créé la bi-clé)
    - PKCS#12

Certification

- Un certificat peut autoriser une entreprise à certifier d'autres entreprises

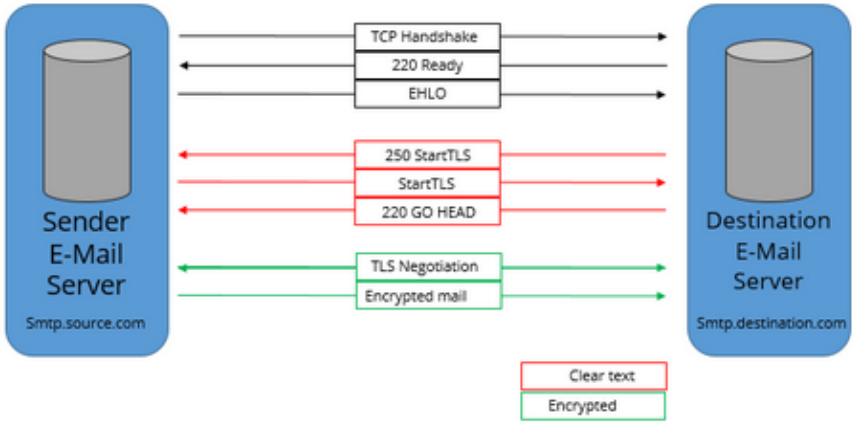
- Vérification d'un certificat
  - Obtention de la clé publique du issuer
    - Directement mis dans des logiciels (web,…)
  - Hachage du certificat puis vérification via la signature
  - Contrôle de la validité
  - Vérification de la révocation (CRL : Certificate Revocation List)

- Avantage
  - Tout le monde peut vérifier la clé publique d'un autre utilisateur
  - Seule l'AC peut modifier un certificat sans que cela soit détecté
  - Tous les certificats sont échangeables

Et le mail ?

- Application très utilisée, mais pas de sécurité dans les échanges de mail standard
- Utilisation possible de S/MIME (chiffrement des pièces jointes)
- Utilisation de pgp pour chiffrer le contenu de ces mails (permet la signature et l'authentification)

- Possibilité d'utiliser un serveur mail utilisant le chiffrement
  - Au niveau smtp, ajout d'une commande StartTLS



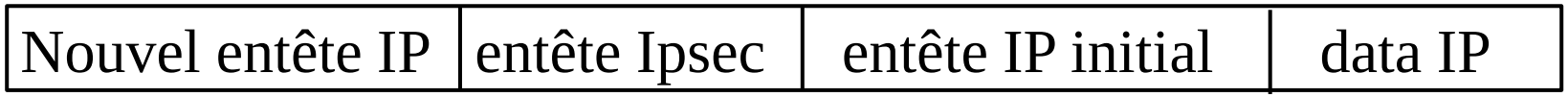
Ipsec (1)

- Internet Protocol Security
- Caractéristiques
  - Ensemble de mécanismes destinés à protéger le trafic au niveau IP (v4 ou v6)
  - Services offerts:
    - Confidentialité, intégrités des paquets
    - Authentification de l'émetteur
    - Anti-rejeux
    - Non répudiation
  - Utilisé au niveau de la couche 3, donc indépendance applicative
  - Ipsec est transparent pour les utilisateurs
  - Normalisé par l'IETF, nombreuses RFC 2410, 2402, 2406, 2408

Ipsec (2)

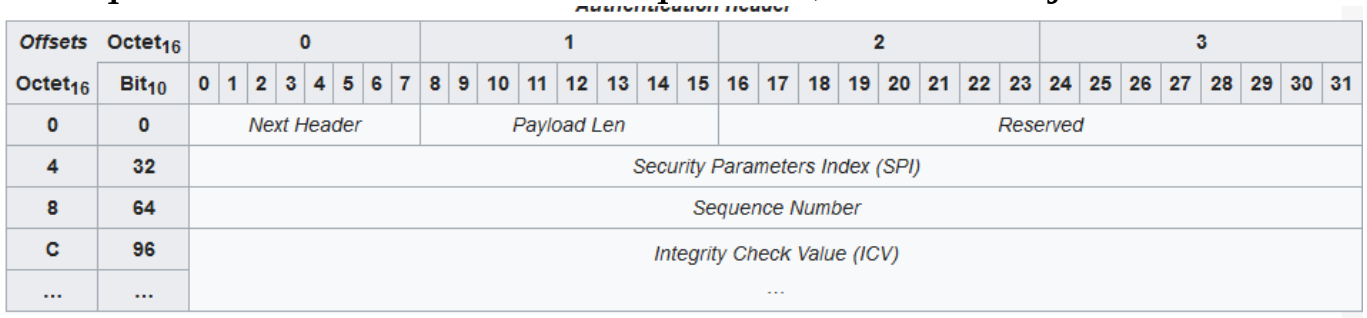
- 2 modes d'exploitation
  - Mode transport :
    - Insertion d'un entête Ipsec entre l'entête IP et les données IP
    - Protège uniquement les données
    - Utilisation pour les contacts point à point

  - Mode Tunnel
    - Encapsulation du paquet IP à l'intérieur d'un nouveau paquet IP
    - Utilisation pour les interconnexions de sites (via VPN)
    - Utilisation pour le nomadisme



Ipsec (3)

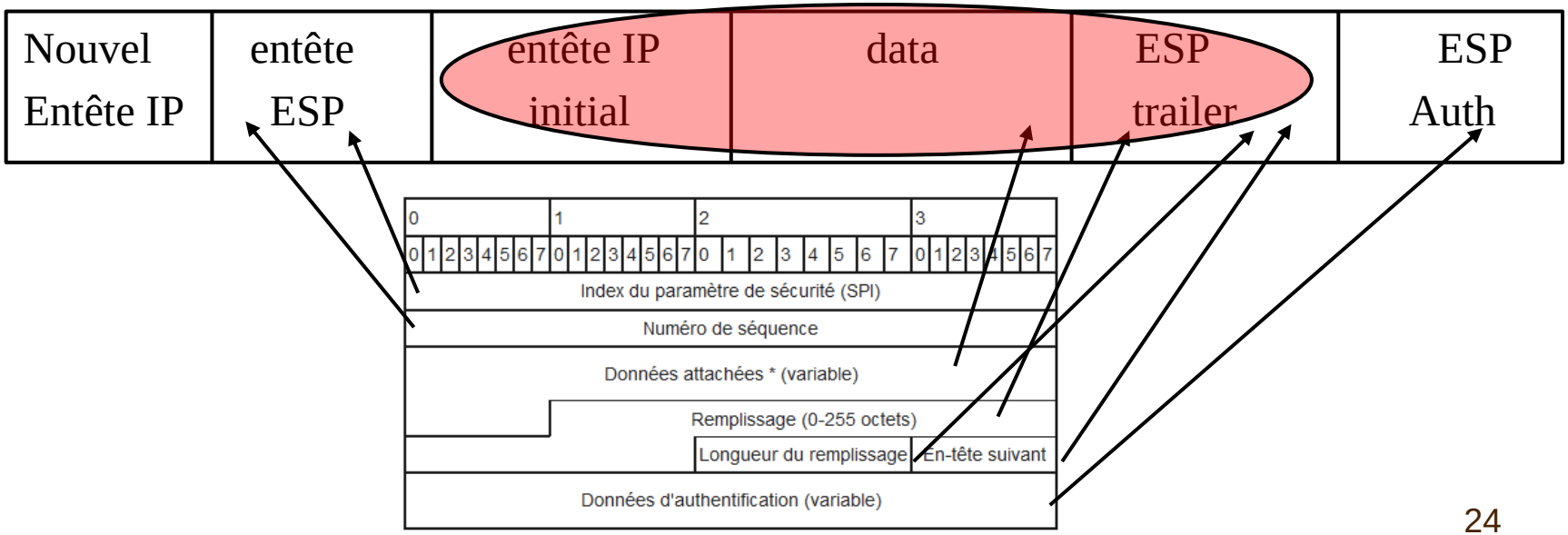
- Extension de sécurité
  - Indépendant du mode (transport ou tunnel)
  - Association de Sécurité (SA)
    - SPI : Security Parameter Identifier (32 bits)
    - Adresse de destination
    - Types d'entête (tunnel ou transport), paramètres de chiffrement, durée de vie, paramètres d'authentification, ….
- AH (Authentication Header), RFC 4302 et 4305
  - Authentification, Intégrité et anti-rejeu
  - Pas du tout confidentialité
  - Sequence number : +1 à chaque fois, évite le rejeu



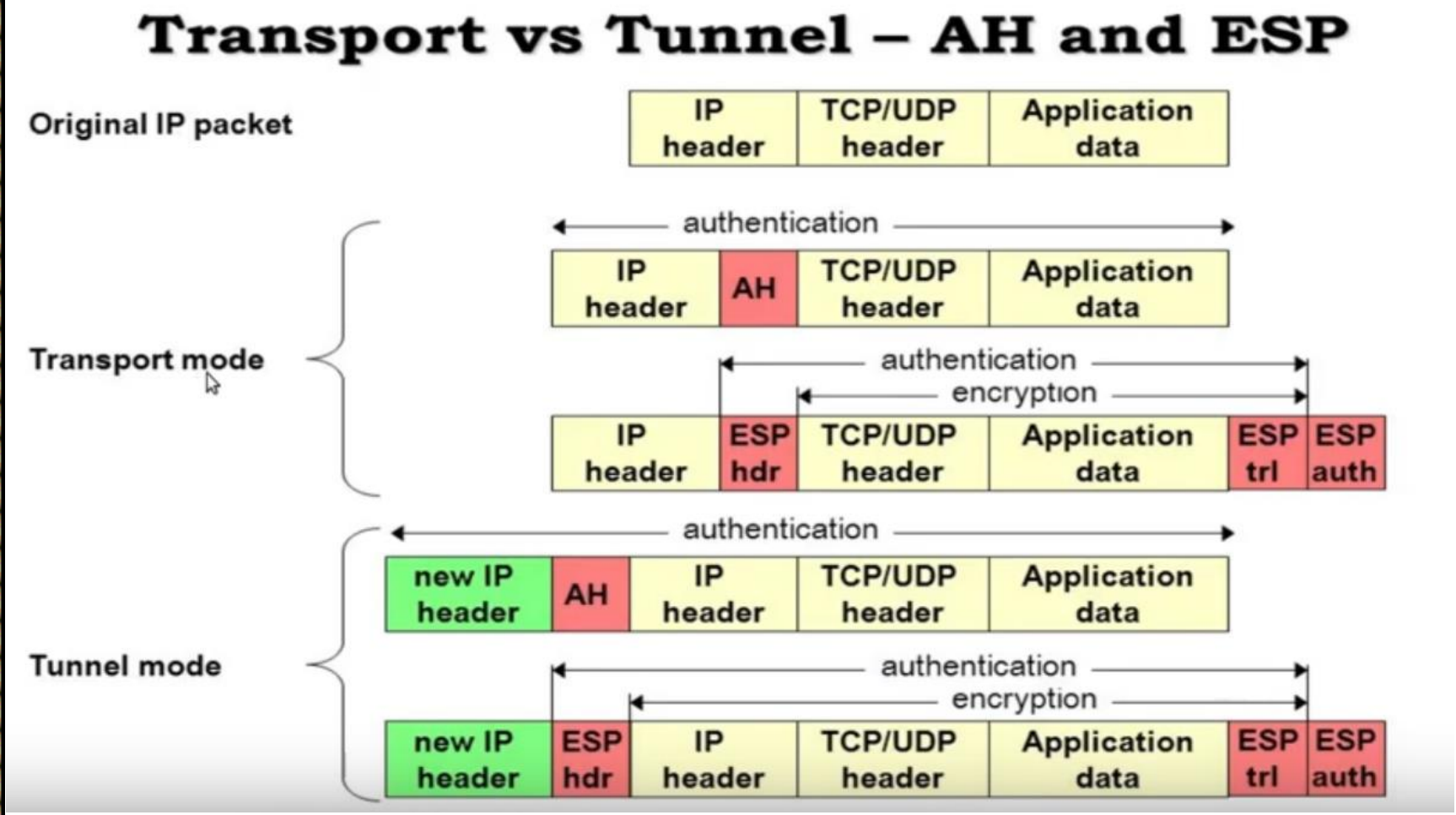
Offsets	Octet$_{16}$	0								1								2								3							
Octet$_{16}$	Bit$_{10}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Next Header								Payload Len								Reserved															
4	32	Security Parameters Index (SPI)																															
8	64	Sequence Number																															
C	96	Integrity Check Value (ICV)																															
...	...	...																															

Ipsec (4)

- AH (suite)
  - Signature de tout le paquet IP initial (sauf TTL)
  - Problème : NAT
- ESP (Encapsulating Security Payload), RFC 2406
  - Authentification, Intégrité et anti-rejeu (optionnel)
  - Confidentialité (données si transport, tout autrement)
  - Sequence number : +1 à chaque fois, évite le rejeu



Trames IPsec



Ipsec (5)

- ESP
  - Besoin : algorithme chiffrement symétrique (AES)
  - Option : hachage : SHA2, SHA3… ou HMAC
    - donc utilisation de clé

- Gestion des clés
  - Manuel
    - Problème usure des clés
    - Pour AES, changement « obligatoire » tous les 2^{38} Go (maths…)
  - Automatique
    - IKEv1 ou IKEv2 (RFC 2409 et 7296)
    - Internet Key Exchange

IKE v1 (1)

- IKEv1
  - Prérequis : avoir un moyen de s'authentifier
    - Clé partagée (pre-shared key)
    - Gestion clé publique/clé privée
    - Pki (certificat)
- 2 phases :
  - Phase 1:
    - Objectif : authentification, échange de clé (Diffie-Hellman)
    - Méthode agressive
      - Échange 1: A-> B g^x, cert, signature+ proposition SA
      - Echange 2 : B-> A g^y , cert, signature+ acceptation SA
      - Echange 3 : A->B, ACK

      La signature permet l'authentification, mais les certificats passent en clair.

IKE v1 (2)

- 2 phases :
  - Phase 1:
    - Méthode main (6 échanges)
      - 2 échanges pour préciser quelles SA utilisées
      - 4 échanges pour mettre en place Diffie-Hellman et l'authentification chiffré (2 messages en clair, 2 derniers chiffrés) -> protection de l'identité des endpoints
  - Phase 2 : mise en place Ipsec SA (quick mode)
    - Méthode : 2 échanges pour échanger 2 nombres aléatoires, (puis 1 ack) puis création d'une clé à partir de Diffie-hellman, et de ces deux nombres.
    - Mise en place PFS (Perfect Forward Secrety) optionnel
- Utilisation clé obtenue pour le chiffrement

IKE v2

- RFC 7296
- La phase 1 est replacé par IKE_SA
  - Message IKE_SA_INIT
    - Propose des algorithmes cryptographiques, échange de graines (2) et diffie-Hellman
    - Tout passe en clair
    - Calcul d'une clé à partir des graines, de DH, et de l'algo choisie
      clé : SKEYSEED
  - Message IKE_AUTH
    - Objectif : authentification des correspondants
    - Chacun peut s'authentifier comme il veut (pre-shared key, EAP, certificat,…)
    - Création d'un Child_SA, avec la clé de DH du IKE_SA_INIT
      - Par la suite, CREATE_CHILD_SA permet de créer d'autres communications, à partir d'un nouveau DH
      - Permet le PFS (Perfect Forward Secrety)

IKE v2 - NAT

- Même problème que pour ike V1 avec le NAT
  - Solution : RFC 3947, 3948
  - NAT-T : NAT traversal

Encapsulation des paquets Ipsec dans UDP (niveau 4)

on passe sur le port 4500 en UDP, après le IKE_SA_INIT

SPD /SAD

- IP sec maintient 2 bases de données
  - SAD : Security Association Database
    - Contient les informations pour chaque SA (algorithme utilisé, SPI, clé, N° séquence, durée de vie des clés,…)
  - SPD : Security Policy Database
    - Spécifie quels services sont offert au trafic entrant/sortant selon @IP, port, …
      - Possible : *none*, *discard*, *ipsec*
    - Décide si ipsec est appliqué au trafic sortant
      - Cherche une entrée dans le SPD
      - Détermine sa SA
      - Fait le traitement si nécessaire

Kerberos (1)

- Développé au MIT, dans les années 80
- Version actuelle : kerberos V5 (non compatible avec V4)

- Basé sur clés symétriques, ticket
- Authentification, confidentialité, intégrité, anti-rejeux

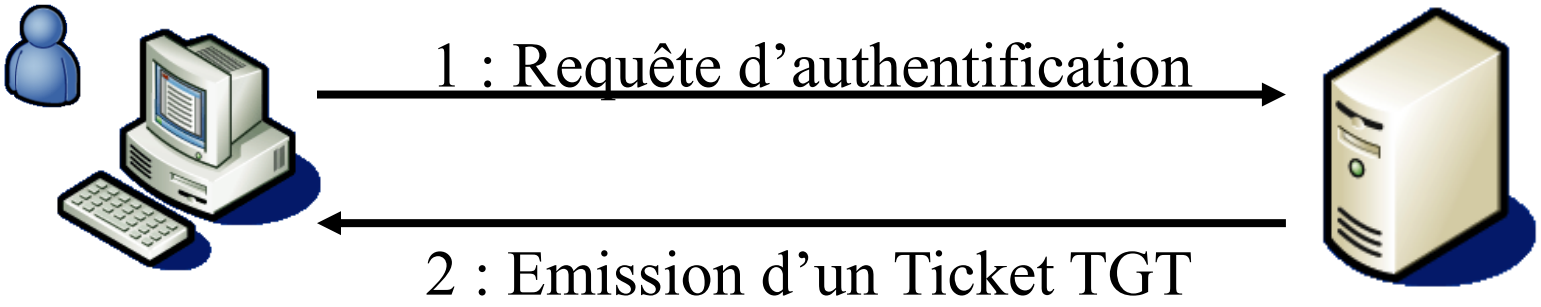
- Architecture 3 tiers
  - Un client
  - Un serveur (qui veut être contacté par le client)
  - Une autorité approuvée (un serveur de confiance)
    - KDC (Key Distribution Center)

- Toutes les machines doivent être dans un *royaume*

Kerberos (2)

- Tickets
  - 2 types
    - TGT (Ticket Granting Ticket) fourni par un TGS (Ticket Granting Service)
    - ST (Service Ticket)
- Le KDC
  - Stocke toutes les clés secrètes entre lui et le reste des équipements du royaume
  - en général, occupe la fonction de TGS

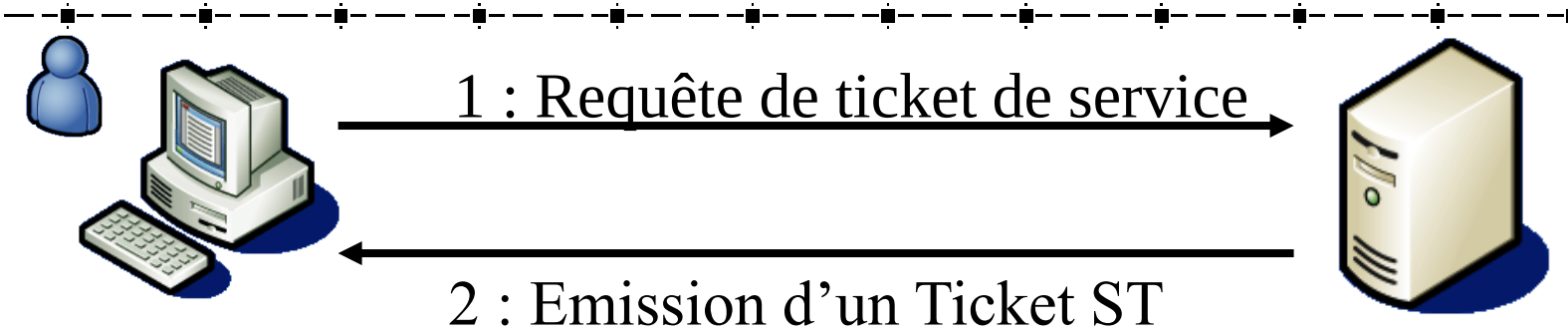
Kerberos (3)



- Requête initiale en clair demandant l'accès à un serveur au KDC
- Le TGT est chiffré avec la clé secrète du client

- Le TGT contient une clé de session, un ticket et le temps
  - Ticket : - clé de session chiffré avec clé secrète du serveur
    - le temps

Kerberos (4)



- Le message ST contient :
  - Le ticket reçu
    - Lisible que par le serveur
  - Un message chiffré avec la clé de session

- A l'ISIMA, utilisation de nfs/kerberos
  - kinit en cas de problème

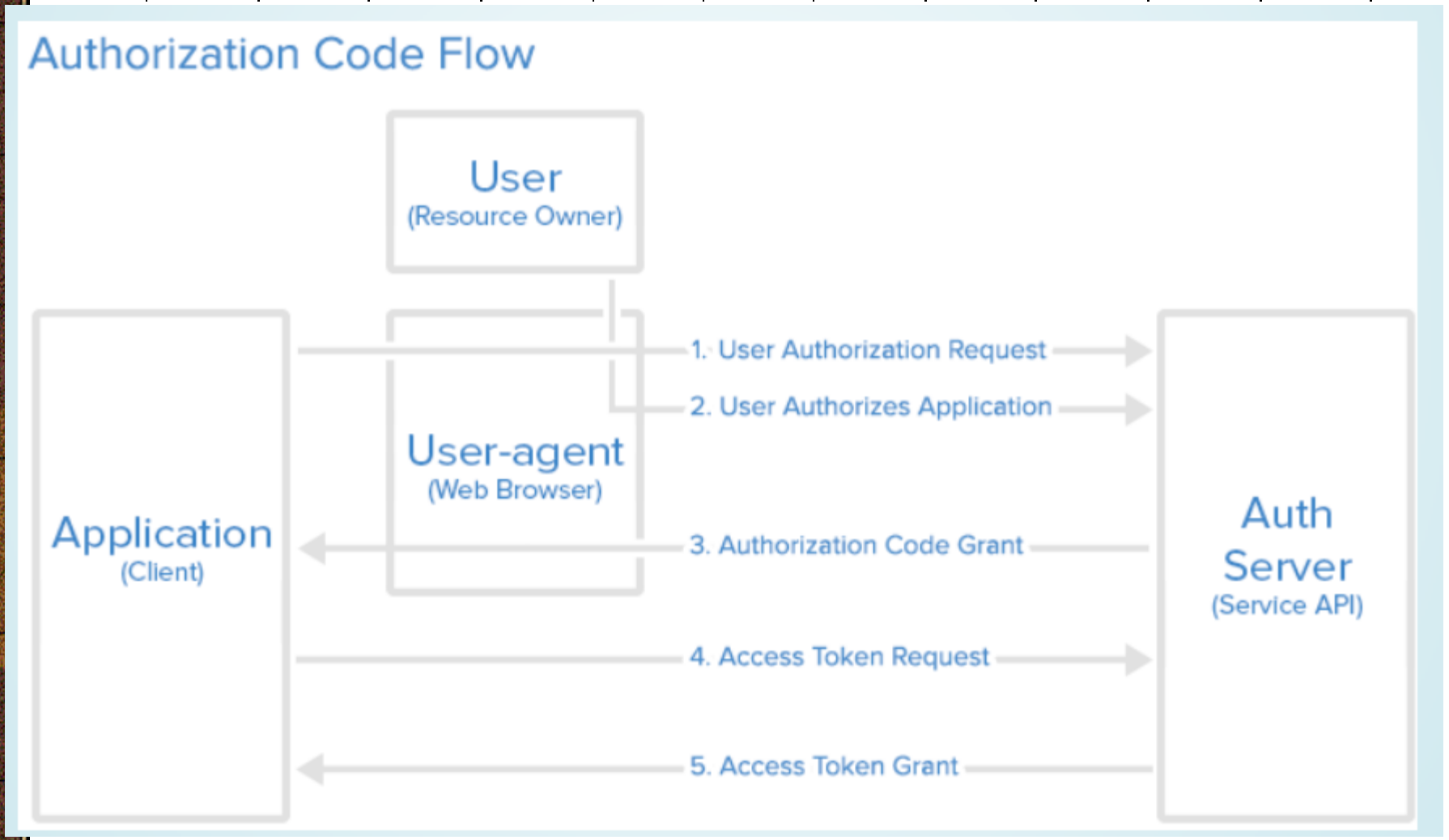
Oauth 2.0 (1)

- Utilisation d’accès délégués
  - **OAuth** est un protocole libre qui permet d'autoriser un site web, un logiciel ou une application (dite « consommateur ») à utiliser l'API sécurisée d'un autre site web (dit « fournisseur ») pour le compte d'un utilisateur.

TERMINOLOGIES

- **Resource Owner:** The User
- **Resource Server:** The API
- **Client:** The third party application
- **Authorization Server:** The server authorizing the client app to access the resources of the resource owner

Oauth 2.0 (2)


Authorization Code Flow
User
(Resource Owner)
Application
(Client)
User-agent
(Web Browser)
Auth
Server
(Service API)
1. User Authorization Request
2. User Authorizes Application
3. Authorization Code Grant
4. Access Token Request
5. Access Token Grant