Authentification

Christophe Kiennert

Sommaire

- Définitions
- Authentification et attaques
- Les modèles d'identités
  - Login / mot de passe : Déclinaisons et alternatives
- Fédération des identités
- Protocoles d'authentification
  - Niveau liaison : PAP, CHAP… mais aussi EAP, RADIUS (vers la sécurité des réseaux sans fil)
  - Niveau TLS
  - Niveau applicatif : Exemple de HTTP Basic / Digest

Sommaire

- Définitions
- Authentification et attaques
- Les modèles d'identités
  - Login / mot de passe : Déclinaisons et alternatives
- Fédération des identités
- Protocoles d'authentification
  - Niveau liaison : PAP, CHAP… mais aussi EAP, RADIUS (vers la sécurité des réseaux sans fil)
  - Niveau TLS
  - Niveau applicatif : Exemple de HTTP Basic / Digest

Services de sécurité

- **C**onfidentialité
- **I**ntégrité
- **A**uthentification
- **Identification**
- **Non répudiation**
- **Horodatage**

L’authentification

- **Authentification** : fournir la preuve certifiant la véracité d’une identité
- Quels types de preuves ?
  - Ce que l’utilisateur connaît (mot de passe, PIN,…)
  - Ce que l’utilisateur possède (générateur d’OTP,…)
  - Ce que l’utilisateur est (biométrie,…)
- L’authentification sous-tend l’existence d’un modèle d’identité numérique
  - Pas de standard universel relatif à l’identité numérique

Définitions

- Types d'authentification :
  - Simple ou mutuelle entre deux parties
  - Faible ou forte (termes répandus mais non normalisés)
- En général, l'authentification est dite « forte » si :
  - Elle fait intervenir deux facteurs différents
  - Elle est basée sur un élément privé et non un secret partagé
- Modèle d'identité le plus en vogue aujourd'hui (et depuis le commencement) : login / mot de passe

Sommaire

- Définitions
- Authentification et attaques
- Les modèles d'identités
  - Login / mot de passe : Déclinaisons et alternatives
- Fédération des identités
- Protocoles d'authentification
  - Niveau liaison : PAP, CHAP… mais aussi EAP, RADIUS (vers la sécurité des réseaux sans fil)
  - Niveau TLS
  - Niveau applicatif : Exemple de HTTP Basic / Digest

Types de protocoles d'authentification

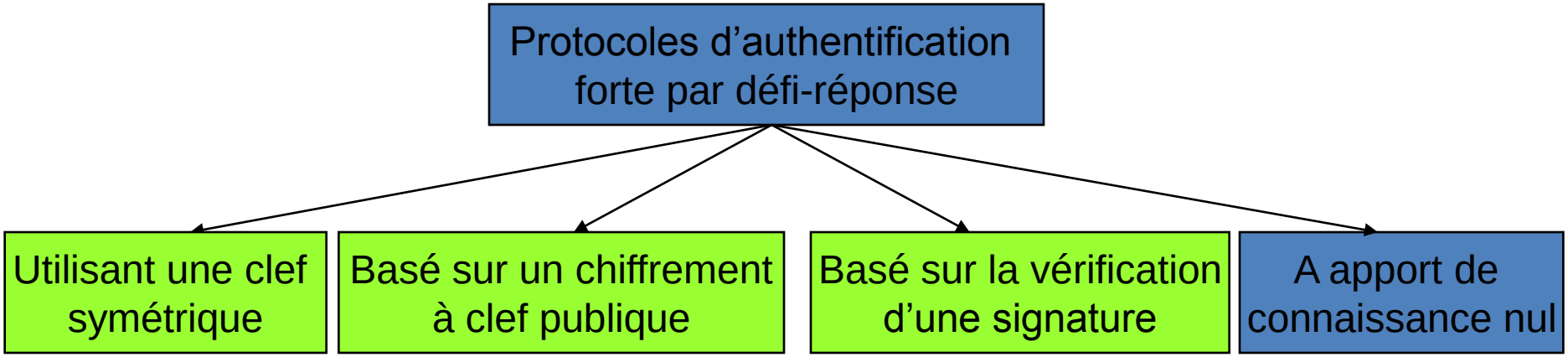
- Protocole à mots de passe
  - mots de passe fixes
    - le prouveur prouve son identité en démontrant au vérificateur qu'il possède un secret en le révélant
  - mots de passe à usage unique (OTP)
- Protocoles de défi-réponse
  - le prouveur montre qu'il sait répondre à un défi sans révéler son secret
    - Défi : nonce, timestamp,…
  - basés sur de la cryptographie symétrique ou asymétrique
- protocoles à apport de connaissance nulle
  - le prouveur démontre la connaissance d'un secret sans révéler d'information sur ce secret ; pour cela le prouveur montre qu'il connaît la solution d'un problème sans révéler la solution

Authentification par défi-réponse

- Authentification par défi réponse: critères de classification
  - Système crypto à clefs publiques ou à clefs secrètes
  - Nombre de messages: 2 passes, 3 passes ...
  - Authentification unilatérale (one-way) ou mutuelle (two-way)
- Avec ou sans utilisation d'une tierce-partie de confiance (TPC)
  - Authentification directe : a et b s'authentifient directement l'un avec l'autre
  - Authentification arbitrée : une *TPC* participe dans les échanges du protocole
    - Avantage : a et b n'ont pas besoin de secret pré-établi
    - Inconvénients : la disponibilité de la TPC est critique et la TPC peut surveiller toute l'activité d'authentification

Authentification par défi-réponse

- Types de protocoles par défi-réponse



Authentification par défi-réponse

- Protocole de distribution de clef (Key establishment protocol)
  - Protocole permettant de distribuer une clef symétrique entre deux parties (ou plus)
  - Protocole de transport de clef (Key transport) : la clef est générée par une des parties puis distribuer à l'autre partie
  - Protocole de mise en accord sur une clef (Key agreement): les deux parties participent à la génération de la clef (exemple : Diffie Hellman)
- Protocole de distribution de clef authentifiée
  - distribuer un secret entre deux parties dont les identités ont été vérifiées

  ➔ Cette clef (symétrique) est souvent appelée clef de session
    - Permet de limiter la quantité de données chiffrées avec une même clef
    - Permet de rendre indépendants des sessions de communications différentes

  ➔ Une clef de session est aussi appelée secret à court terme, par opposition aux secrets à long terme (les clefs symétriques ou privées utilisées pour l'authentification)

Authentification par défi-réponse

- Conventions de notations
  - Kab : *clef secrète* partagée entre a et b (utilisée avec un algorithme symétrique)
  - PKa : *clef publique* de a (utilisée avec un algorithme asymétrique)
  - SKa : *clef privée* de a (utilisée avec un algorithme asymétrique)
  - $CERT_a$: *certificat* de a (émis par une autorité de certification connue)
  - Si m désigne un message
    - $\{m\}_{Kab}$ (resp. $\{m\}_{PKa}$ et $\{m\}_{SKa}$) : m *chiffré* avec Kab (resp. PKa et SKa)
    - $S_a(m)$: *signature* de m par a
    - h(m) : *hash* de m avec la fonction de hachage h
    - $h_K(m)$: *MAC (« message authentication code »)* calculé sur m avec la fonction de hachage paramétrée par la clef secrète K, h_K

Authentification par défi-réponse

- Protocoles avec chiffrement symétrique
  - Supposent l'existence d'une clef secrète pré-partagée K
  - Le prouveur démontre sa connaissance de K en chiffrant un défi (et éventuellement des données)
  - Authentification mutuelle avec nombre aléatoire

    M_1: $b \rightarrow a : r_b$

    M_2: $a \rightarrow b : \{ r_a . r_b . b \}_{Kab}$

    M_3: $b \rightarrow a : \{r_b . r_a\}_{Kab}$

    - Une clef de session peut être dérivée à partir de r_a

    ➔Protocole de transport de clef

Authentification par défi-réponse

- Protocoles basés sur un chiffrement avec une clef publique
  - Le prouveur déchiffre avec sa clef privée un challenge chiffré avec sa clef publique
  - Exemple : Needham Schroeder à clefs publiques (NSPK, 78)

M_1: $a \rightarrow s : a \,.\, b$

M_2: $s \rightarrow a : \{PK_b \,.\, b\}_{SKs}$

M_3: $a \rightarrow b : \{r_a \,.\, a\}_{PKb}$

M_4: $b \rightarrow s : b \,.\, a$

M_5: $s \rightarrow b : \{PK_a \,.\, a\}_{SKs}$

M_6: $b \rightarrow a : \{r_a \,.\, r_b\}_{PKa}$

M_7: $a \rightarrow b : \{r_b\}_{PKb}$

- Une clef de session peut être dérivée à partir de r_a et r_b

➔Protocole de mise en accord sur une clef

Attaques sur les protocoles de défi-réponse

- Modèle de l'attaquant (modèle de Dolev Yao)
  - X peut écouter les messages échangés ;
  - X peut bloquer les messages ;
  - X peut rediriger les messages ;
  - X peut enregistrer les messages ;
  - X peut rejouer les messages ;
  - X peut fabriquer (chiffrer et déchiffrer) de nouveaux messages à partir d'anciens
  - X ne sait pas déchiffrer (sans avoir la clef) dans le temps d'une session (par contre une clef d'une session précédente peut être cassée)
- Convention
  - X/A $\rightarrow$ B : M signifie que X envoie le message M à B en se faisant passer pour A
  - A $\rightarrow$ X/B : M signifie que X intercepte le message M envoyé par A à B, B ne le recevant pas

Attaques sur les protocoles de défi-réponse

- Attaque par **rejeu** : enregistrer un message d'une session pour l'injecter dans une session ultérieure
  - Parade : défi (nonce, timestamp,…)
- Attaque **Man In The Middle (MITM)** : se placer au milieu de la conversation sans que les participants légitimes ne le sachent.
  - Inclut le fait de modifier / bloquer les messages en cours de route
- Attaque par **entrelacement de sessions** : Etablir une session pendant qu'une autre session est en cours.
  - Objectif : Utiliser les messages d'une session pour forger les messages d'une autre session

Exemple d'attaque par rejeu

- Exemple : protocole de Needham-Schroeder à clefs symétriques (NSSK) [NS78]

  M_1, a $\rightarrow$ s : a.b. r_a

  M_2, s $\rightarrow$ a : $\{r_a . b . K_{ab} . \{K_{ab} . a\}_{Ksb}\}_{Ksa}$

  M_3, a $\rightarrow$ b : $\{K_{ab} . a\}_{Ksb}$

  M_4, b $\rightarrow$ a : $\{r_b\}_{Kab}$

  M_5, a $\rightarrow$ b : $\{r_b - 1\}_{Kab}$

- Attaque de Denning-Sacco : l'attaquant x a enregistré une session précédente et a pu "casser" oldKab

  M_3', x/a $\rightarrow$ b : $\{oldK_{ab} . a\}_{Ksb}$ (x rejoue un vieux message)

  M_4', b $\rightarrow$ x/a : $\{r_b\}_{oldKab}$ (x connaît oldKab)

  M_5', x/a $\rightarrow$ b : $\{r_b - 1\}_{oldKab}$

  (b pense partager une clef secrète "fraîche" oldKab avec a)

  x peut communiquer avec b en se faisant passer pour a !

Sommaire

- Définitions
- Authentification et attaques
- Les modèles d'identités
  - Login / mot de passe : Déclinaisons et alternatives
- Fédération des identités
- Protocoles d'authentification
  - Niveau liaison : PAP, CHAP… mais aussi EAP, RADIUS (vers la sécurité des réseaux sans fil)
  - Niveau TLS
  - Niveau applicatif : Exemple de HTTP Basic / Digest

Le modèle login / mot de passe

- Le plus simple à mettre en place…
  - Dans la mouvance de la création des protocoles de l'Internet (HTTP, SMTP, DNS, etc.), tous non sécurisés
- … mais pas le plus sécurisé
  - Attaques par force brute / dictionnaire
  - Exemple *: John The Ripper* pour casser les identifiants Unix.
- … et pas si simple à gérer
  - Tendance à l'obligation de créer un compte sur tout service
  - Très difficile pour l'utilisateur de mémoriser l'ensemble de ses identifiants
  - En revanche, très facile à gérer niveau serveur

Le modèle login / mot de passe

- Authentification par mot de passe en clair
  - Échange en clair sur le réseau
  - Peut être stocké en clair du côté serveur selon les applications
    - Telnet, Ftp, POP, HTTP basic, bases de données (SQL), …
  - Exemple: Protocole PAP (Password Authentication Protocol)

- Authentification par mot de passe avec challenge
  - Le mot de passe ne transite pas durant l'échange
  - Est toujours stocké en clair du côté serveur
  - Exemple: protocole CHAP (Challenge Authentication Protocol)
  - Problème du rejeu résolu par l'utilisation d'un nonce

Le modèle login / mot de passe

- Sécurité du login / mot de passe
  - Mot de passe : secret partagé, architecture symétrique
  - Plusieurs risques intrinsèques :
    1. Peut être récupéré au vol (MITM, Phishing)
    2. Peut être récupéré avant le vol (Trojan, Keyloggers)
    3. Peut être deviné (mots de passe triviaux)
- Solutions possibles :
  1. Authentification préalable du serveur (typiquement TLS)
  2. Stockage du mot de passe dans un environnement de confiance (typiquement carte à puce)
  3. Définir un mot de passe robuste

Le modèle login / mot de passe

- Contre-argumentation des solutions :
  1. Peut-on être sûr que l'utilisateur fera la différence entre un serveur authentifié et non authentifié ?
     - Le problème du phishing n'est pas résolu par TLS, celui du MITM l'est (presque)
  2. Qui va distribuer des cartes à puce aux utilisateurs et leur expliquer comment s'authentifier avec ?
     - Possible dans une communauté restreinte, pas pour le grand public
  3. Comment définit-on la robustesse d'un mot de passe ?
     - On essaye de l'approximer par la notion d'entropie

La notion d'entropie

- En théorie de l'information (Shannon) : quantité qui mesure l'apport d'information d'un événement
  - Un tirage d'un dé à 10 faces non pipé apporte de l'information (3,3 bits)
  - Un tirage d'un dé à 10 faces qui ne sort jamais la face 1 et la face 2 apporte moins d'information (3 bits)
- Par analogie : entropie d'un mot de passe = quantité de bits sur laquelle serait codé ce mot de passe
  - Exemple : 20 bits d'entropie = cassable en 2^{20} essais
- Problème : trop bonne entropie = impossible à retenir…

Les OTP

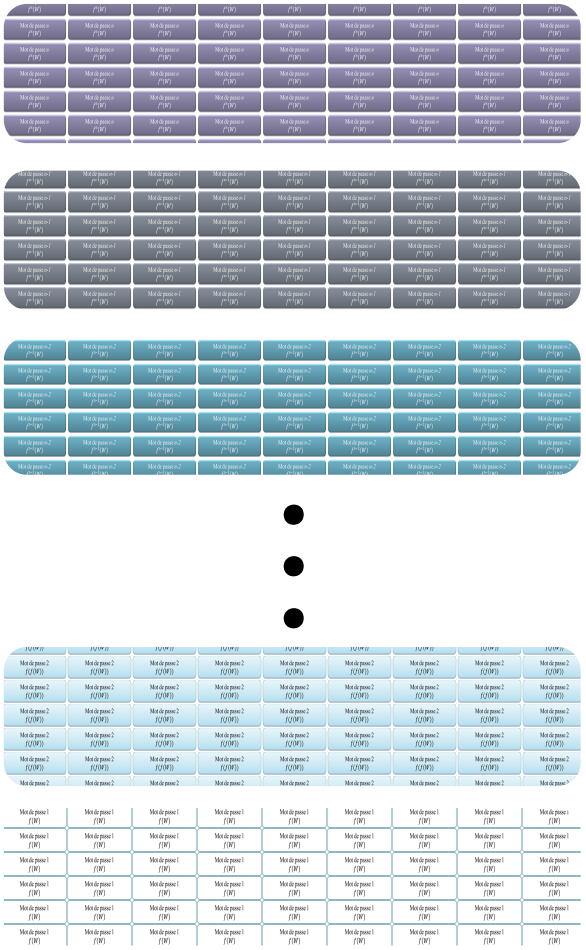
- Mot de passe à usage unique
- Différentes mises en œuvre :
  - Cartes OTP « papier » (Banques,…)
  - Transmission d'un OTP par SMS
  - Générateur Hardware (RSA SecurID)
  - Purement logiciel (S/KEY)
- Rendent inefficaces les attaques par Phishing
  - Mais pas le MITM…
- Ont généralement une fenêtre temporelle de validité

Exemple de calcul d'OTP : S/KEY

- Utilisateur : choisit une graine *s,* jamais transmise
- Fonction *f* à sens unique (par ex. hachage)
- Utilisateur calcule $f(s), f^2(s), \ldots f^n(s)$, puis jette *s*.
- Envoi au serveur de $f^n(s)$ pour référence
- Authentification avec l'itération précédente
- Impossible pour un observateur de remonter les itérations car *f* supposée à sens unique

Le protocole S/KEY

L'utilisateur possède :



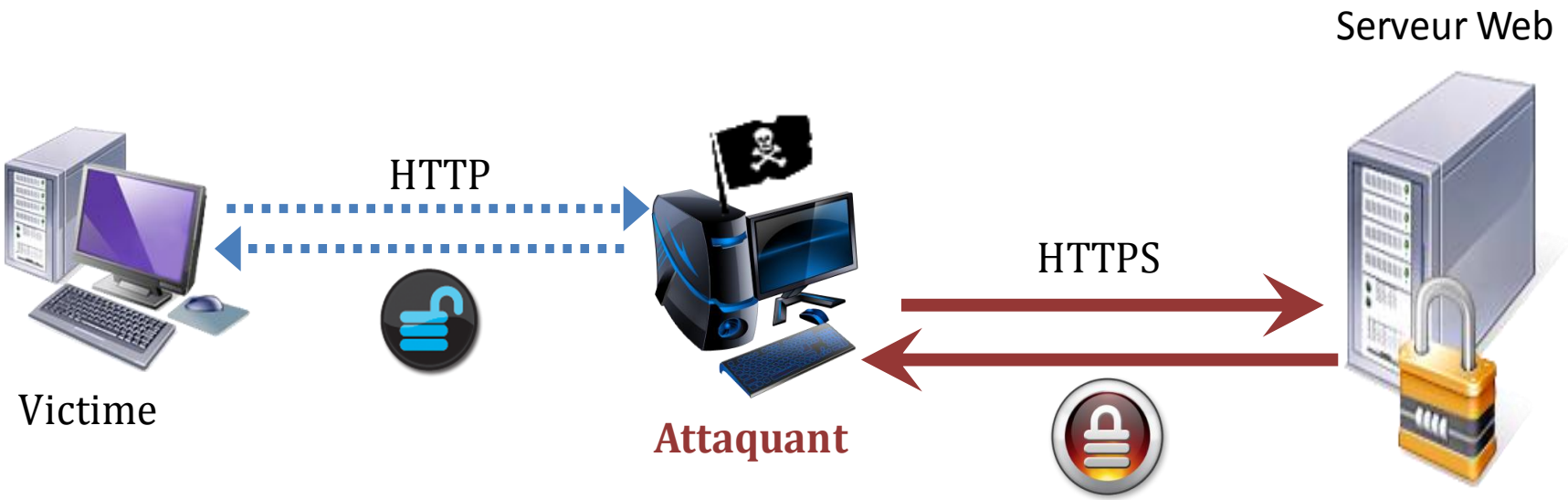
Le serveur connait :

OTP : Limites

- OTP : Bonne dérivation du principe du mot de passe
- Reste un secret partagé sensible au MITM
- Authentification à deux facteurs : TLS + OTP
  - TLS authentifie le serveur
  - OTP authentifie le client postérieurement
- Censé résister à tout, mais HTTPS reste sensible au MITM
  - Attaque sur la partie HTTP de HTTPS
  - Cf. *SSLStrip* (Marlinspike)

Principe de l'attaque sur HTTPS

- On contacte rarement un serveur en saisissant soi-même « https:// » …



La biométrie

- Authentification à partir de ce que la personne est
  - Pas de l'authentification à proprement parler, mais de l'identification
- Large choix de modalités
  - Empreintes digitales, vocales, oculaires, etc.
- Prévention naturelle contre le vol du « mot de passe »
  - Sauf cas de mutilation avérée…
- 2 mises en œuvres possibles :
  - Vérification : login fourni avec la modalité, vérification 1 : 1
  - Identification : seule la modalité est fournie, identification 1 : N

La biométrie

- Avantages :
  - Identification relativement sûre si le lecteur est muni de contre-mesures
    - *Liveness detection*, etc.
  - Rien à mémoriser ou à conserver pour l'utilisateur
- Inconvénients :
  - Coûteux et intrusif
  - Modalités souvent imitables
  - … mais non modifiables : en cas de blessure, la modalité devient inutilisable
  - Peu adapté à l'authentification dans les réseaux informatiques

Le certificat X509 et TLS

- Autre modèle d'identité numérique
  - Pas de notion de secret partagé : modèle asymétrique
- Elément de base du protocole TLS
- Authentification par certificat repose sur
  - La cryptographie asymétrique (Ex : RSA)
  - Diffie-Hellman (éventuellement)
  - La signature numérique PKCS #1
- **A l'heure actuelle** : RSA et Diffie-Hellman sont considérés comme sûrs
- TLS peut aussi fonctionner sur le principe du secret partagé (TLS-PSK)
  - Mais ce secret n'est jamais transmis sur le réseau

Problèmes du certificat X509

- Difficile à gérer
  - Nécessite une infrastructure dédiée (PKI) qui gère la délivrance, les autorités de certification et la révocation
- Très rarement mis en place pour de l'authentification mutuelle
  - Le serveur s'authentifie par son certificat, le client envoie son mot de passe dans le tunnel sécurisé
  - Compliqué de confier un certificat aux utilisateurs
  - Peut être résolu dans des contextes restreints par l'utilisation de cartes à puce

Sommaire

- Définitions
- Authentification et attaques
- Les modèles d'identités
  - Login / mot de passe : Déclinaisons et alternatives
- Fédération des identités
- Protocoles d'authentification
  - Niveau liaison : PAP, CHAP… mais aussi EAP, RADIUS (vers la sécurité des réseaux sans fil)
  - Niveau TLS
  - Niveau applicatif : Exemple de HTTP Basic / Digest

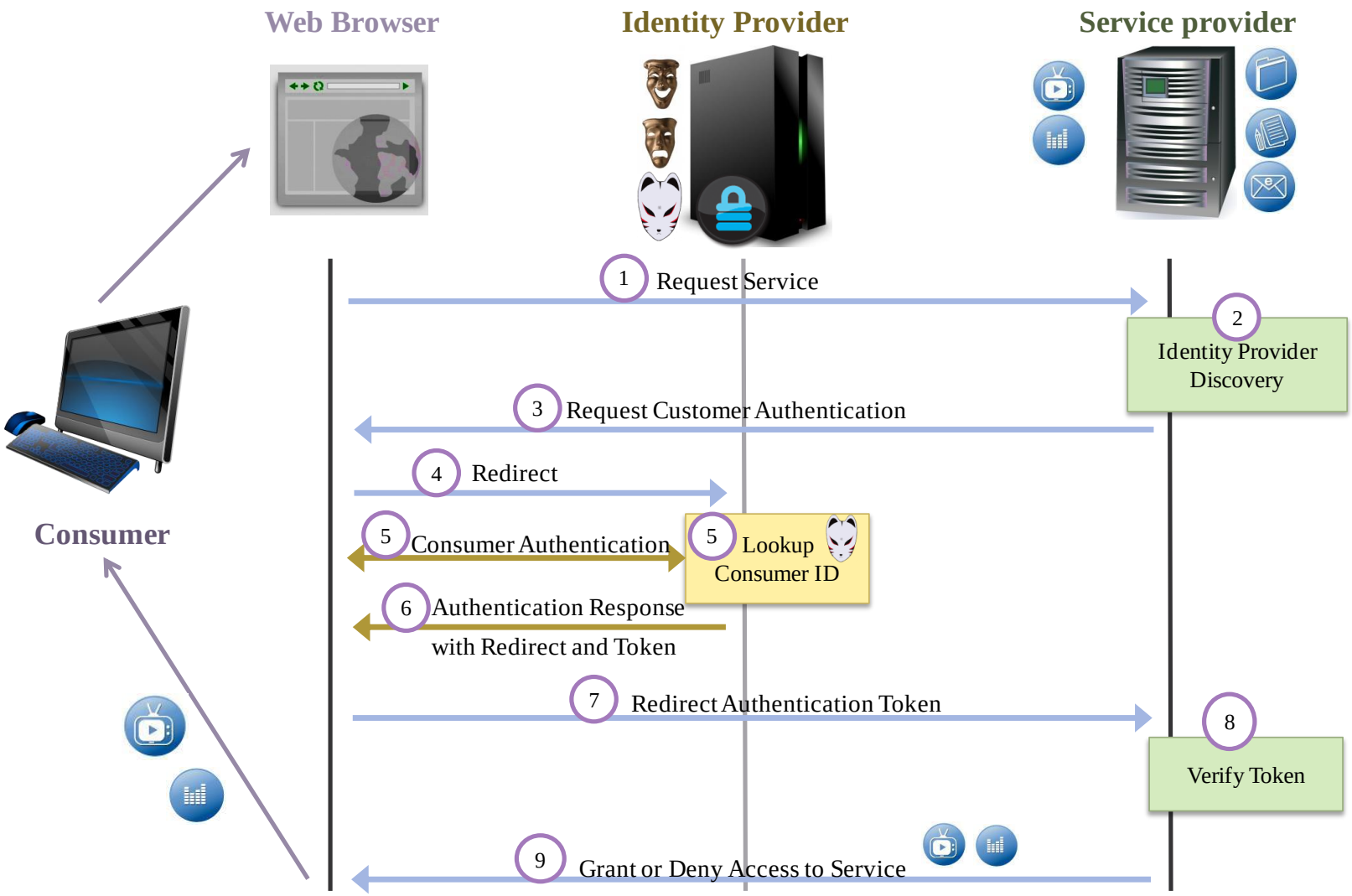
Principe de la fédération d'identité

- **Paradigme fondamental** : authentification unique pour accéder à un ensemble de services
  - Facilite la gestion d'identité pour les utilisateurs
  - Décharge les serveurs de l'authentification des clients
  - Repose sur une technologie assez complexe
  - Principe très sensible à l'usurpation d'identité
- Permet de diffuser des méthodes d'authentification non classiques
- Nombreuses technologies et solutions :
  - SAML (Liberty Alliance), OpenID, etc.

Terminologie

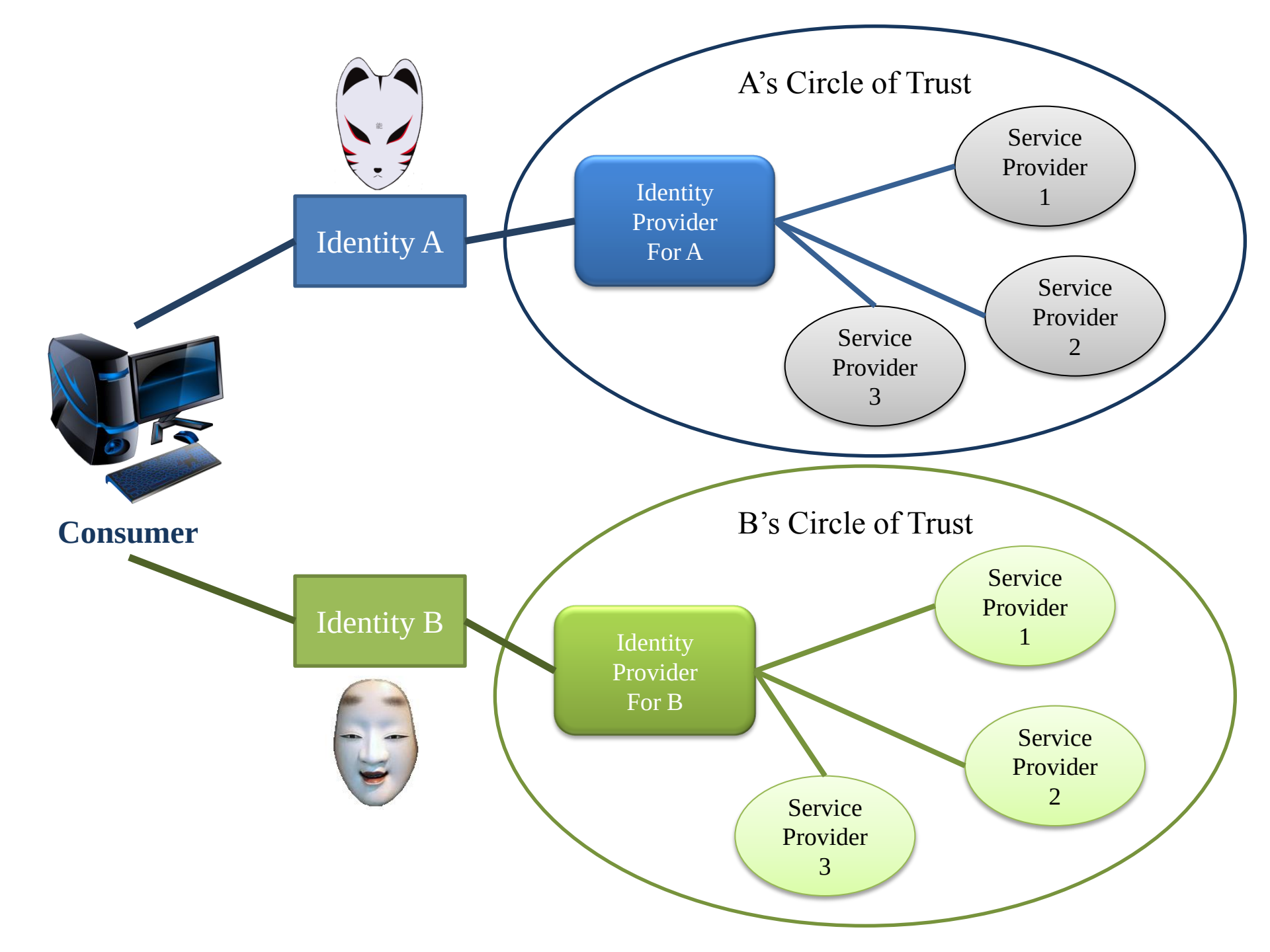
- **Identity Provider** (IdP) : Entité qui fournit l'identité à l'utilisateur et est en charge de l'authentification
- **Service Provider** (SP) : Service dont souhaite profiter l'utilisateur
- **Consumer** : Utilisateur cherchant à faire usage d'un service

Scénario d'authentification



Le problème de l'IdP

- L'IdP est responsable de l'authentification de l'utilisateur
- Le SP doit donc faire confiance à l'IdP
  - Sérieux et honnêteté de l'IdP
  - Robustesse de l'authentification
- Deux approches différentes : avec ou sans cercles de confiance
  - **Cercle de confiance** : ensemble constitué d'un IdP et de plusieurs SP faisant confiance à cet IdP


A's Circle of Trust
Identity A
Identity Provider For A
Service Provider 1
Service Provider 2
Service Provider 3
Consumer
B's Circle of Trust
Identity B
Identity Provider For B
Service Provider 1
Service Provider 2
Service Provider 3

La question du cercle de confiance

- Permet aux SP de ne pas faire confiance à n'importe quel IdP
  - La confiance en tout IdP nuit aux garanties de sécurité
  - C'est la solution retenue par **OpenID** qui choisit de privilégier la simplicité d'utilisation
- Les cercles de confiance complexifient notoirement la gestion de l'identité
  - Mode de fédération difficile d'accès aux utilisateurs finaux (création de communautés fermées)
  - Exemple : **Liberty Alliance** (bâti sur SAML)

Sommaire

- Définitions
- Authentification et attaques
- Les modèles d'identités
  - Login / mot de passe : Déclinaisons et alternatives
- Fédération des identités
- Protocoles d'authentification
  - Niveau liaison : PAP, CHAP… mais aussi EAP, RADIUS (vers la sécurité des réseaux sans fil)
  - Niveau TLS
  - Niveau applicatif : Exemple de HTTP Basic / Digest

Sommaire

- Définitions
- Les modèles d'identités
  - Login / mot de passe : Déclinaisons et alternatives
- Fédération des identités
- Protocoles d'authentification
  - Niveau liaison : PAP, CHAP… mais aussi EAP, RADIUS (vers la sécurité des réseaux sans fil)
  - Niveau TLS
  - Niveau applicatif : Exemple de HTTP Basic / Digest

Authentification niveau 2

- A l'origine : sécurisation du protocole PPP
  - Protocoles simples PAP, CHAP
- Problème : chaque point d'accès doit être un serveur d'authentification
- Evolution : centralisation de l'authentification
  - Protocole et serveurs RADIUS
- Evolution : cadre générique pour l'accès PPP avec très large choix de méthodes d'authentification
  - Protocole EAP
  - Généralisation aux réseaux sans-fil

PAP

- Protocole PAP : Password Authentication Protocol (1992)
  - Impossible de concevoir un protocole plus simple
  - Envoi d'une requête d'authentification avec le mot de passe **en clair**
  - Renvoi d'une acceptation ou d'un rejet par l'hôte distant
- Sécurité : aucune
- Concevable de l'utiliser seulement dans le protocole EAP-TTLS

CHAP

- Protocole CHAP : Challenge Handshake Authentication Protocol (1996)
  - Améliore PAP : pas de transmission du mot de passe en clair
  - Protection contre le rejeu par un challenge (nonce)
- 4 types de messages définis :
  - Challenge
  - Response
  - Accept
  - Failure

CHAP

Client

Host

Challenge : nonce

Response : H(password || nonce)

Success

CHAP - Sécurité

- Résiste à :
  - Ecoutes
  - Rejeu
- Ne résiste pas à :
  - Attaque par dictionnaire après écoute
- Peu pratique à déployer sur tous les points d'accès
- Stockage des mots de passe en clair
- Extension propriétaire : MS-CHAP

MS-CHAP

- Version 1 (1998) : mise en place d'un moyen de ne conserver qu'un hash du mot de passe côté serveur
  - Dérivation de clés DES à partir du hash, pour chiffrer le challenge, que le serveur peut déchiffrer
  - Problème : Faiblesse de la fonction de hash (propriétaire)
- Version 2 (2000) :
  - Correction des faiblesses de la v1 (fonction de hash)
  - Mise en place d'une authentification mutuelle
  - Faiblesses face à une attaque par dictionnaire
    - Dérivation de 3 clés DES : la 3è clé a une longueur de 16 bits au lieu de 56 bits

RADIUS

- Remote Authentication Dial-In User Service (1991, à l'IETF en 1997)
- Objectifs :
  - Approche centralisée de l'authentification
  - Choix du protocole d'authentification
  - Extension des fonctions d'authentification
    - AAA : Authentication, Authorization, Accounting
- Protocole Client / Serveur transporté par UDP
  - Client : Tout point d'accès dans un réseau
  - Serveur : *Daemon* (1 seul serveur par réseau)

RADIUS

- Sécurité :
  - Echanges RADIUS sécurisés par un **secret partagé** entre client et serveur
- Protocole extensible :
  - Messages constitués par une succession d'attributs, au format Type – Longueur – Valeur
    - Ex : Type 1 : User-Name, Type 2 : User-password, Type 3 : CHAP password
  - Ajout possible de nouveaux attributs
  - Large choix de protocoles d'authentification (PAP, CHAP, Kerberos, etc.)

RADIUS

- Attribut de type 2 : **User-Password**
  - MD5(secret, nonce) XOR password
  - Le secret partagé est protégé par MD5
  - Le mot de passe est protégé par le XOR avec un aléa
- **Authentification mutuelle** : utilisation du secret partagé
- Echange de messages en version complète :
  - Client : Access-Request (avec l'identifiant)
  - Serveur : Access-Challenge (auth. mutuelle)
  - Client : Access-Request (données d'authentification)
  - Serveur : Access-Accept

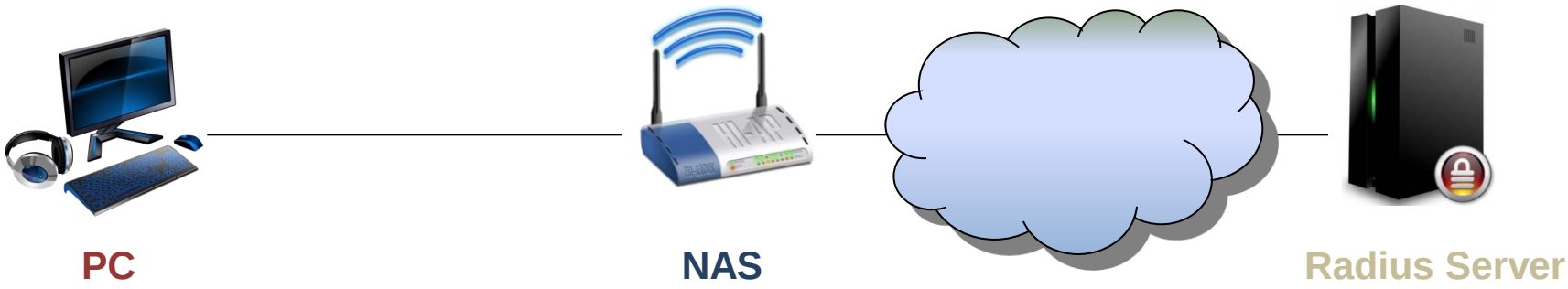
RADIUS

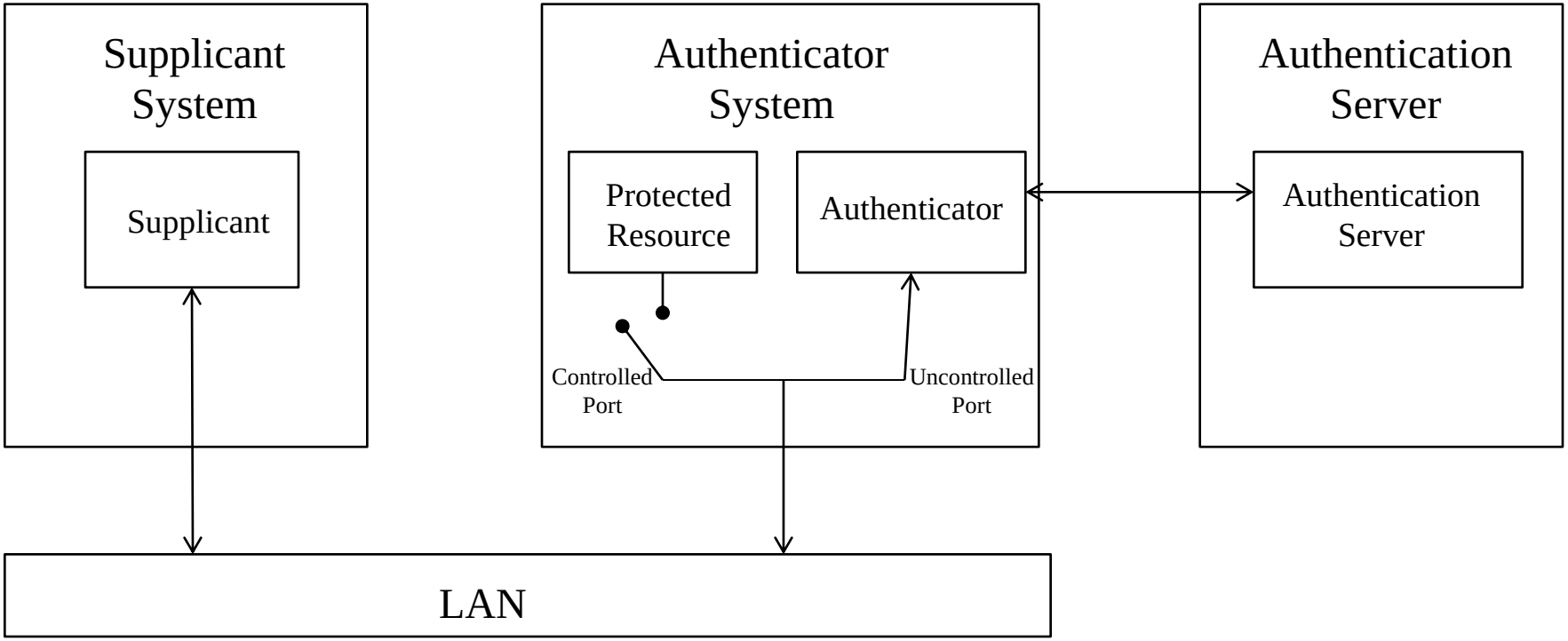
- Très utilisé, notamment :
  - Prestataires de service Internet
  - Intranet
- Options de gestion et d'administration intéressantes (principe du AAA)
  - Facturation sur le temps de connexion par exemple
- RADIUS est un protocole de niveau applicatif, mais activé au niveau liaison
  - Transporté sur UDP

EAP

- Extensible Authentication Protocol (1998)
  - Pas un protocole d'authentification, mais qui encapsule des protocoles d'authentification
  - Des dizaines de méthodes, très variées (cartes à puce, OTP, …)
  - A la base utilisé pour des liaisons PPP
- Objectif : Protocole adapté pour une standardisation de l'architecture d'authentification centralisée
- EAP est utilisé dans l'architecture 802.1X (EAPoL) :
  - **Supplicant** : client désirant accéder à une ressource, qui communique en PPP (pas d'accès au réseau)
  - **NAS** : Point d'accès pouvant communiquer avec un serveur (typiquement un client RADIUS)
  - **Serveur d'authentification** : typiquement serveur RADIUS

EAP : Architecture 802.1X





EAP

- 802.1X définit une architecture de contrôle d'accès par port :
  - Si le supplicant n'est pas authentifié, le NAS ne laisse passer que les échanges avec le serveur d'authentification
  - Sinon, le NAS permet les échanges avec le réseau
- EAP est souvent transporté par RADIUS entre le NAS et le serveur d'authentification (RADIUS)
- Quatre types de messages EAP :
  - EAP-Request / EAP-Response
  - EAP-Success / EAP-Failure
- EAPoL est utilisé abondamment dans les réseaux sans-fil
  - WPA

Supplicant
(client)

Authenticator

Radius Server

EAP-Start

EAP-Request.Identity

EAP-Response.Identity

Acces-Request/EAP

Access-Challenge/EAP

EAP-Request.Type

EAP-Response.Type

Acces-Request/EAP

Unicast Key Computation

Unicast Key Computation

Access-Accept/EAP + Unicast Key

EAP Success

Global Key Computation

EAPoL Key (Global Key)

EAP-TLS

- Créé en 1999 pour les liaisons PPP (2008 pour EAPoL)
- Consiste à porter sous EAP la partie authentification et échange de clés de TLS
- Authentification mutuelle obligatoire
- Limitation corollaire : nécessité d'un certificat client
- Bon niveau de sécurité (à condition de vérifier les certificats)

EAP-TTLS

- **EAP Tunneled-TLS** (2008) : Objectif : se débarrasser de l'obligation du certificat client
- Création d'un tunnel sécurisé par authentification du serveur
- Puis authentification du client par une méthode au choix, par exemple PAP, CHAP, MS-CHAP, EAP-MD5, …
- Sécurité : beaucoup plus sensible que EAP-TLS
  - Dépend beaucoup du choix du protocole d'authentification client
  - … et de la force du mot de passe, le cas échéant !

EAP - Conclusion

- **Cadre** pour transporter tous les protocoles connus au niveau liaison
- **Extensible** : beaucoup de nouvelles propositions pour augmenter les méthodes d'authentification transportées
- Utilisation significative

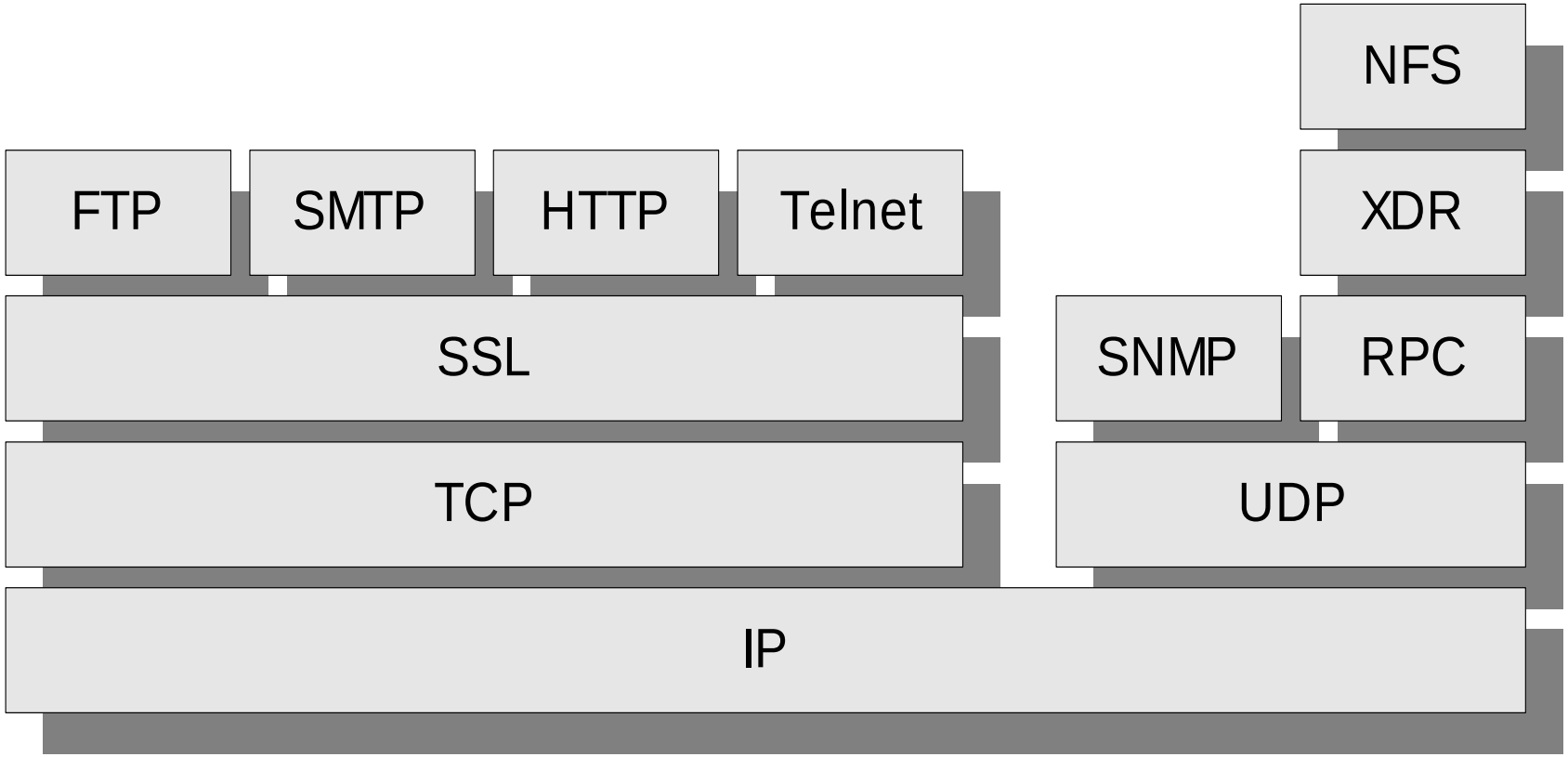
Sommaire

- Définitions
- Authentification et attaques
- Les modèles d'identités
  - Login / mot de passe : Déclinaisons et alternatives
- Fédération des identités
- Protocoles d'authentification
  - Niveau liaison : PAP, CHAP… mais aussi EAP, RADIUS (vers la sécurité des réseaux sans fil)
  - Niveau TLS
  - Niveau applicatif : Exemple de HTTP Basic / Digest

SSL / TLS

- SSL défini par *netscape* et intégré au browser
- Première version de SSL testée en interne Première version de SSL diffusée : V2 (1994)
- Version actuelle V3
- Standard à l ’IETF au sein du groupe Transport Layer Security (TLS)
  - TLS v1.0 = SSL v3.1
  - Version la plus récente : TLS v1.2
- Repose sur l’authentification par certificat X.509

Pile protocolaire



TLS avec les applications usuelles

Protocole sécurisé	Port	Protocole non sécurisé	Application
HTTPS	443	HTTP	Transactions requête-réponse sécurisées
SSMTP	465	SMTP	Messagerie électronique
SNNTP	563	NNTP	News sur le réseau Internet
SSL-LDAP	636	LDAP	Annuaire X.500 allégé
SPOP3	995	POP3	Accès distant à la boîte aux lettres avec rapatriement des messages

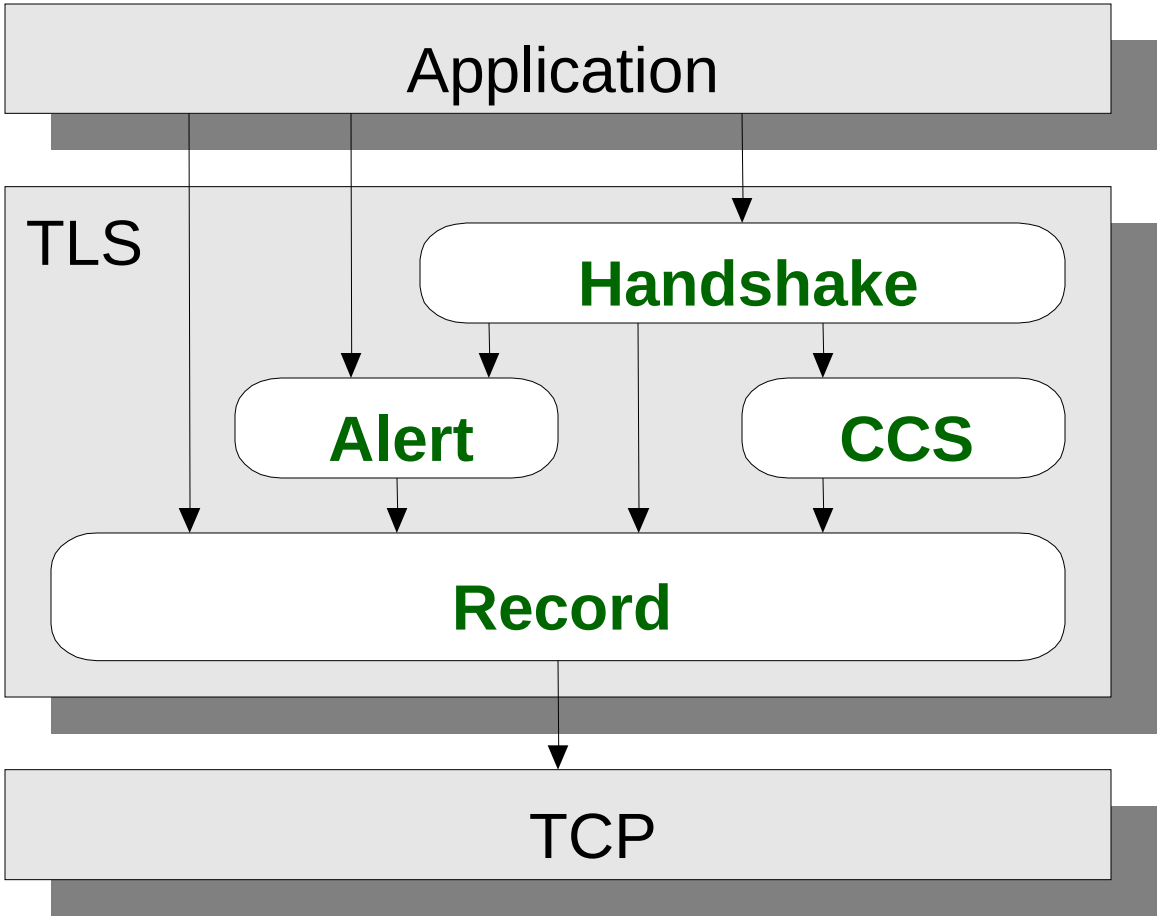
TLS avec les applications usuelles

Protocole sécurisé	Port	Protocole non sécurisé	Application
FTP-DATA	889	FTP	Transfert de fichiers
FTPS	990	FTP	Contrôle du transfert de fichiers
IMAPS	991	IMAP4	Accès distant à la boîte aux lettres avec ou sans rapatriement des messages
TELNETS	992	Telnet	Protocole d'accès distant à un système informatique
IRCS	993	IRC	Protocole de conférence par l'écrit

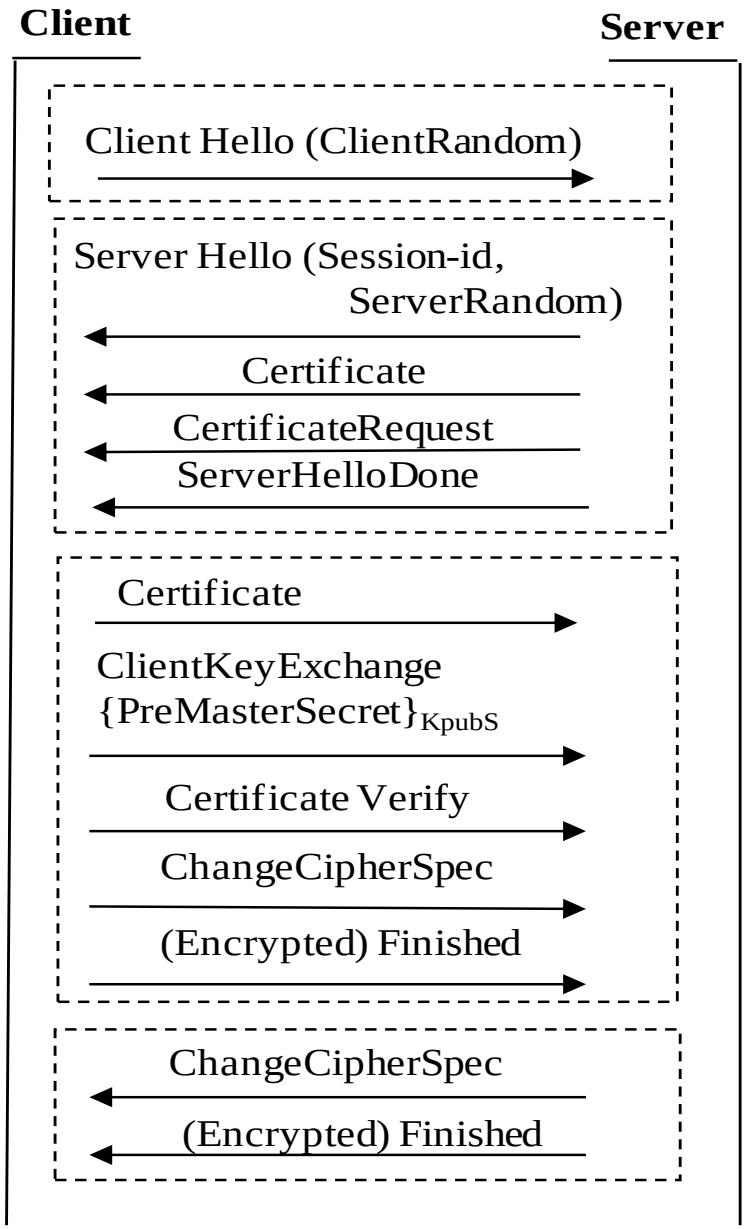
Services assurés par TLS

- Authentification
  - Serveur (obligatoire), client (optionnel)
  - Utilisation de certificat X509 V3
  - A l'établissement de la session.
- Confidentialité
  - Algorithme de chiffrement symétrique négocié, clé générée à l'établissement de la session.
- Intégrité
  - Fonction de hachage avec clé secrète : HMAC(clé secrète, h, Message)
  - HMAC adjoint à chaque fragment de données avant chiffrement
- Non Rejeu
  - Numéro de séquence

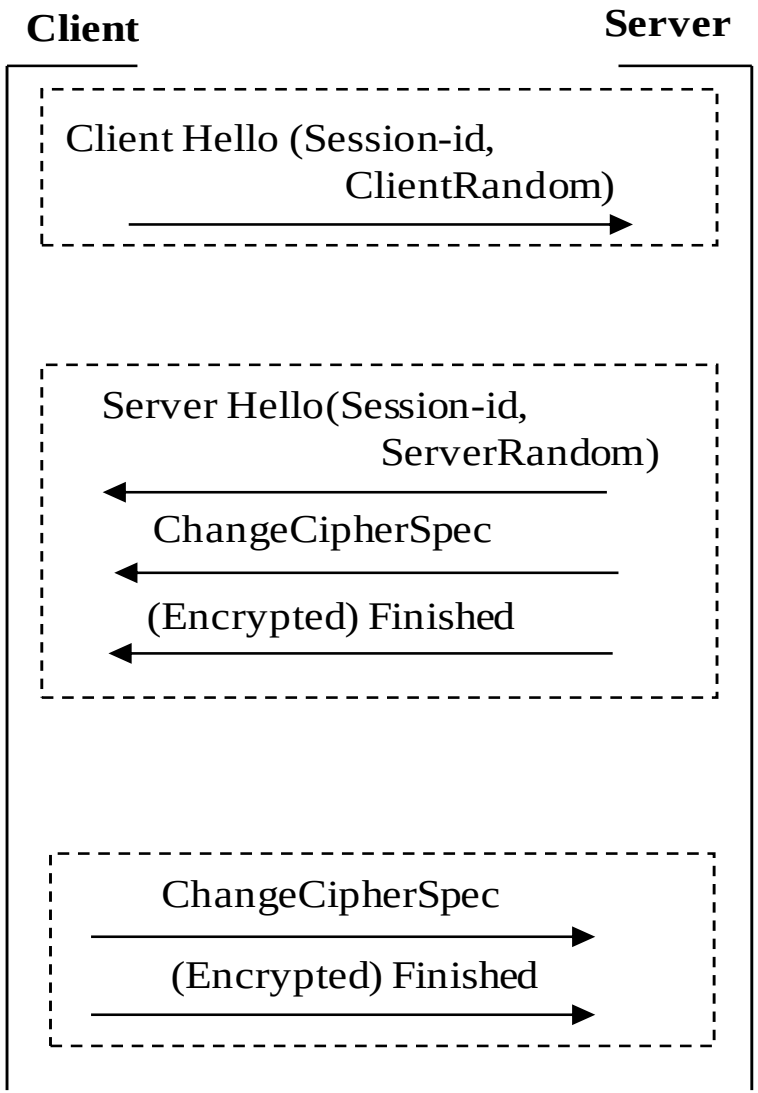
Pile TLS



Session TLS Full



Session TLS Resume



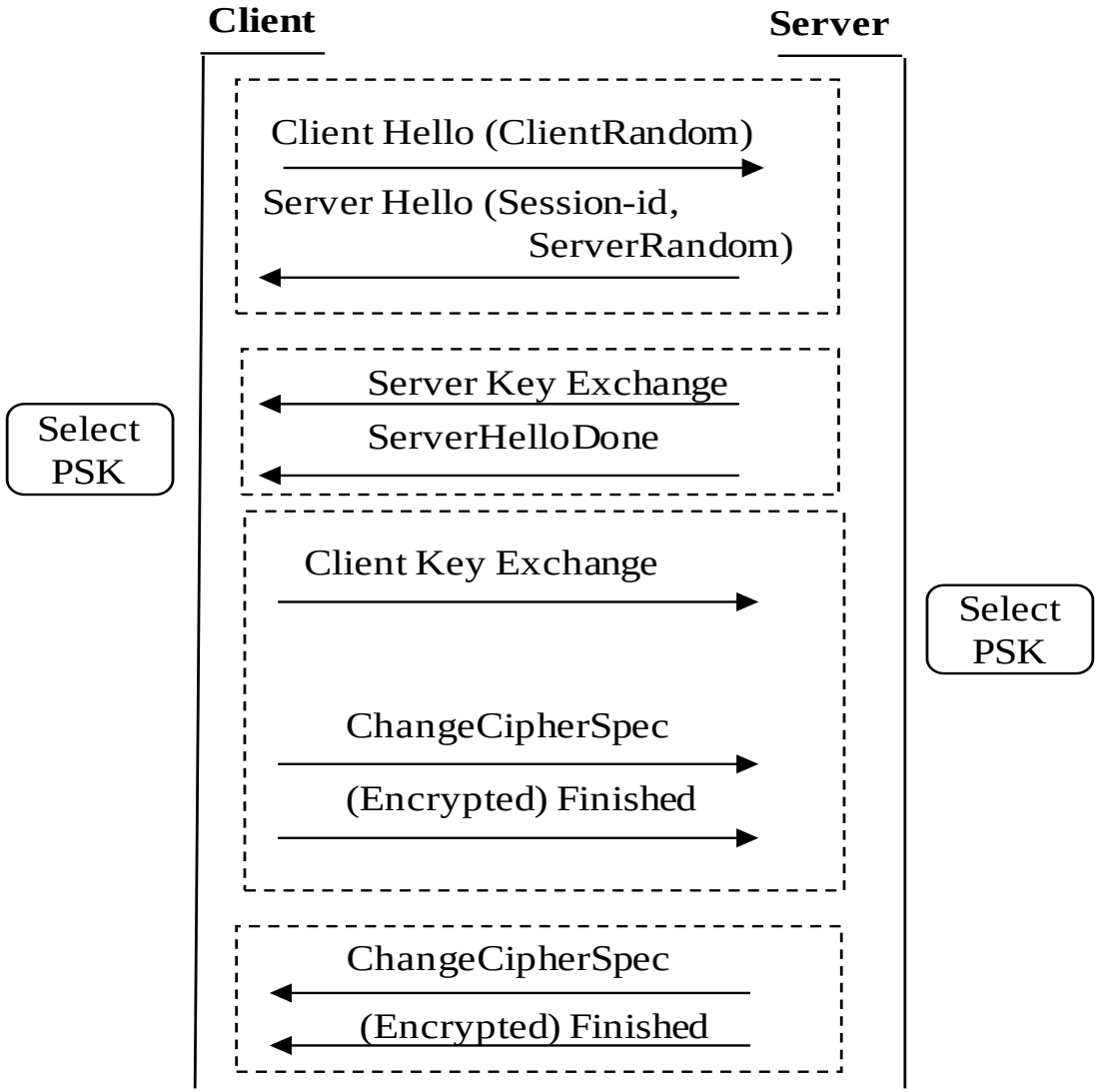
Eléments cryptographiques de TLS (Full)

- **Cryptographie hybride** : mise en accord sur une clé symétrique grâce à la crypto asymétrique
- Génération d'un *Pre Master Secret* par le client (48 octets), envoyé au serveur après chiffrement asymétrique
- Dérivation d'un *Master Secret* par le client et le serveur :
  - `master_secret = PRF(pre_master_secret, "master secret", ClientHello.random + ServerHello.random) [0..47];`
  - Le *Master Secret* est un élément essentiel de la session TLS
- Dérivation de *Key Blocks* (clés temporaires symétriques) à partir du *Master Secret*
  - Confidentialité et intégrité de chaque paquet
  - `key_block = PRF(master_secret, "key expansion", server_random + client_random);`

Eléments cryptographiques de TLS (Resume)

- Session basée sur un Session ID envoyé dans le *Client Hello*
  - La reprise de session peut échouer si la session est trop ancienne ou si le serveur refuse
- Le Master Secret ne change pas
- Les Key Blocks sont renouvelées à chaque paquet
  - Nouveaux *randoms* échangés dans les *Client/Server Hello*
- Si le Master Secret est divulgué, toute la session TLS peut être déchiffrée !

TLS-PSK (pas de certificat)



Analyse de TLS-PSK

- Ne nécessite pas de PKI
- Le secret partagé (PSK) n'est jamais transmis sur le réseau au cours du protocole
- Dérivation du *Master Secret* à partir du PSK et des *randoms*
  - Puis dérivation des Key Blocks
- Le PSK est encore plus sensible que le *Master Secret*, puisqu'il est un secret permanent
  - Stockage sécurisé nécessaire des 2 côtés

Certificats X.509

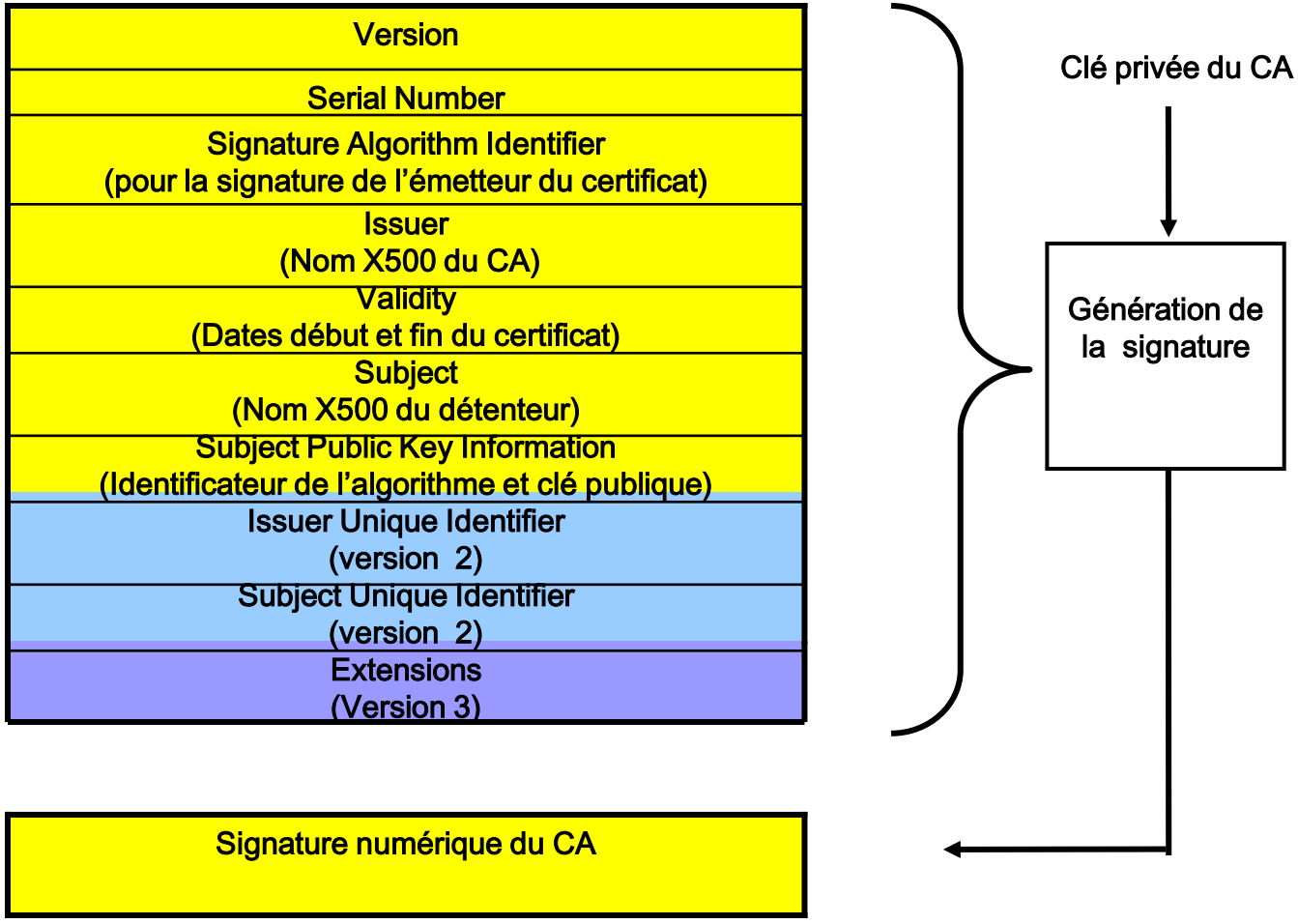
- « Certificat » = relève d'une autorité ou institution
- Le contenu = information « authentique »
- Mise en place d'un état de confiance en présence d'un certificat
- Dictionnaire : « Acte écrit qui rend témoignage de la vérité d 'un fait, d'un droit »
- Présence d'une autorité « reconnu » qui atteste de la véracité du contenu.
- Certificat = document « signé »

Certificats X.509

- Standard:
  - ITU-T X.509(03/2000), ou ISO/IEC 9594-8
    - Certificats de clé publique et d'attribut
  - RFC 3280: (définition de profil fonctionnel basé sur X509)
- Versions successives:
  - 1988 : v1
  - 1993 : v2 = v1 + 2 nouveaux champs
  - 1996 : v3 = v2 + extensions

Format des certificats X.509

- Structure de données permettant de lier différents éléments au moyen d'une signature



Certificats X.509 : format ASN1

Certificat ::= SEQUENCE {

version[0] **Version DEFAULT v1,**

serialNumber **CertificateSerialNumber,**

signature **AlgorithmIdentifier,**

issuer **Name,**

validity **Validity,**

subject **Name,**

subjectPublicKeyInfo **SubjectPublicKeyInfo,**

issuerUniqueIdentifier[1] IMPLICIT UniqueIdentifier OPTIONAL,

-- si ce composant est présent, la version doit être v2 ou v3

subjectUniqueIdentifier[2] IMPLICIT UniqueIdentifier OPTIONAL,

-- si ce composant est présent, la version doit être v2 ou v3

extensions[3] **Extensions OPTIONAL**

-- si ce composant est présent, la version doit être v3 --

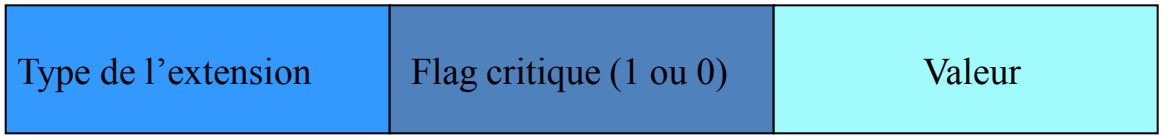
}

Certificats X.509 et extensions

- Le but initial des certificats est de lier:
  - identité et clé publique par la signature d'un tiers de confiance
- Pour couvrir des services plus étendus
  - Nécessaire d'associer d'autres informations à la clé publique
- L'extension consiste à ajouter de nouveaux champs aux certificats
- Extension définie dans ITU-T Rec. X.660 et ISO/IEC 9834-1
- Des extensions sont standardisées
  - possibilité de définir des extensions spécifiques
- Si l'application ne supporte pas une extension critique, elle abandonne le certificat.

Certificats X.509 : format des extensions

- Structure de l'extension



- Type est unique pour chaque extension: le type est un type de base ASN1 Object Identifier (OID)
- Avec un flag critique
  - Si l'application:
    - ne supporte pas cette extension, elle refuse le certificat
    - supporte cette extension et la valeur de l'extension est conforme à l'application elle l'accepte sinon elle le rejette
- Avec un flag non critique
  - Si l'application
    - ne supporte pas cette extension, l'extension est a supporte pas cette extension abandonnée mais le certificat accepté
- La valeur est conforme au type de l'extension

Certificats X.509 : Extensions

- Extensions sur:
  - le nommage de l'objet et du signataire
  - les clés publiques/privées
  - la révocation
  - la politique de certification
  - le rôle
  - Autres …

Certificats X.509 : Key Usage

- **KeyUsage** : usage de la clé publique certifiée

  DigitalSignature

  NonRepudiation

  KeyEncipherment

  DataEncipherment

  keyAgreement

  keyCertSign

  CRLSign

  encipherOnly

  decipherOnly

Vérification d'un certificat serveur dans TLS

- Dans l'ordre :
  - Vérifier la date de validité du certificat
  - Vérifier si le CA est de confiance
    - Eventuellement, vérifier la chaîne des certificats des CA jusqu'au dernier, qui doit être de confiance
    - Les certificats des CA « racine » sont présents par défaut dans les navigateurs, et considérés comme étant de confiance
    - Cette hypothèse est-elle vraiment raisonnable ?
  - Vérifier la signature numérique à l'aide de la clé publique du CA
  - Vérifier que le nom de domaine correspond au DN du certificat (optionnel !)

Conclusion sur TLS

- TLS est un protocole d'authentification robuste
  - Résiste à toutes les attaques classiques, notamment le MITM
  - Confidentialité et intégrité assurées par des clés temporaires renouvelées régulièrement
- Des limitations subsistent toutefois :
  - Attaque sur HTTPS (SSLStrip)
  - Généralement utilisé en authentification simple (serveur)
    - Ne protège pas vraiment du phishing…
  - Repose sur la sécurité des certificats X.509
    - Avez-vous confiance dans les autorités de certification ?

Sommaire

- Définitions
- Authentification et attaques
- Les modèles d'identités
  - Login / mot de passe : Déclinaisons et alternatives
- Fédération des identités
- Protocoles d'authentification
  - Niveau liaison : PAP, CHAP… mais aussi EAP, RADIUS (vers la sécurité des réseaux sans fil)
  - Niveau TLS
  - Niveau applicatif : Exemple de HTTP Basic / Digest

Authentification de niveau applicatif

- L'authentification peut être gérée par les applications elles-mêmes
  - Cas le plus répandu avec le login / mot de passe
  - La requête HTTP transporte (généralement en POST) ces données
  - Transport chiffré seulement si TLS est activé
- Il existe aussi l'authentification avec HTTP
- Login / Mot de passe décliné selon 2 méthodes :
  - HTTP Basic
  - HTTP Digest

Méthodes HTTP

- HTTP Basic
  - Le client envoie une requête HTTP vers le serveur
  - Le serveur lui retourne un message d'erreur (statut 401) et lui demande de s'authentifier avec le mode *basic*
  - Le client réémet sa première requête avec des entêtes supplémentaires incluant son identifiant (login) et son mot de passe, le tout en clair
  - Le serveur lui retourne la ressource si les paramètres fournis ont été validés
  - Sécurité : *no comment*

Méthodes HTTP

- HTTP Digest
  - Le client envoie une requête HTTP vers le serveur
  - Le serveur lui retourne un message d'erreur (statut 401) et lui demande de s'authentifier avec le mode *digest*, et renvoie un nonce
  - Le client réémet sa première requête avec des entêtes supplémentaires incluant son identifiant (login) et le résultat du hash de son mot de passe combiné avec le nonce et éventuellement d'autres paramètres (le mot de passe ne transite pas en clair)
  - Le serveur lui retourne la ressource si les paramètres fournis ont été validés

Méthodes HTTP

- **Réponse du serveur à une requête du client nécessitant l'authentification du client:**

  HTTP/1.1 401 Unauthorized
  WWW-Authenticate: Digest
  realm="testrealm@host.com",
  qop="auth,auth-int",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  opaque="5ccc069c403ebaf9f0171e9517f40e41"

- **Réponse du client au message précédent: Authorization:**

  Digest username="Mufasa",
  realm="testrealm@host.com",
  nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
  uri="/dir/index.html",
  qop=auth,
  nc=00000001,
  cnonce="0a4f113b",
  response="6629fae49393a05397450978507c4ef1",
  opaque="5ccc069c403ebaf9f0171e9517f40e41"

Méthodes HTTP

- Sécurité de HTTP Digest :
  - Pas de mot de passe en clair
  - Prévention du rejeu
  - Prévention contre des attaques à chiffré choisi
  - Mais : vulnérable au MITM