La sécurité des réseaux

9e cours—2014

Louis Salvail

Échanges de clés authentifiés

- Supposons qu'Obélix et Astérix, qui possèdent des clés publiques certifiées PK_O et PK_A, veulent établir une communication sûre entre eux.
- Ceci est habituellement réalisé en leur permettant d'établir une clé de session K (clé de courte vie) qui est utilisée par la suite, via les techniques à clés secrètes (symétriques).
- La méthode utilisée doit garantir qu'Astérix et Obélix s'entendent sur K et que chacun soit certain de l'identité de l'autre.
- Donc, K doit n'être connue que d'Astérix et d'Obélix.

Conditions

Voici les 3 conditions que doivent satisfaire les échanges de clés authentifiés :

À la fin, Astérix est convaincu qu'Obélix est présent!

1. Si Astérix et Obélix concluent que le protocole a été appliqué avec succès, alors ils ont communiqué entre eux.

Il y a entente!

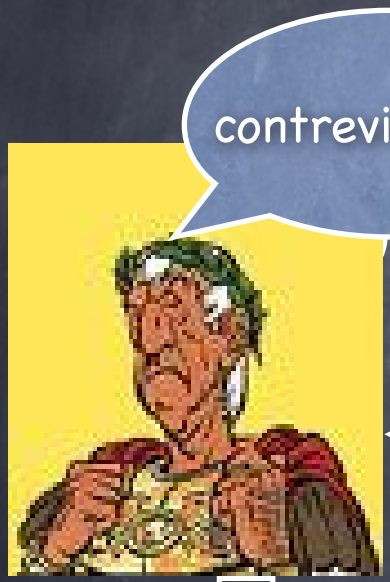
2. À moins que les méthodes cryptographiques ne soient cassées, César ne peut forcer une situation où Astérix (Obélix) croit qu'il partage K avec Obélix (Astérix) mais en fait la partage avec César.

Pas de redite admise!

3. Si le protocole est appliqué avec succès, alors la clé K est nouvelle, aléatoire et indépendante de tout le reste.

Mauvais exemple

1. Si Astérix ou Obélix conclut que le protocole a été appliqué avec succès alors les deux participants sont présents. ✗



contrevient à 3.

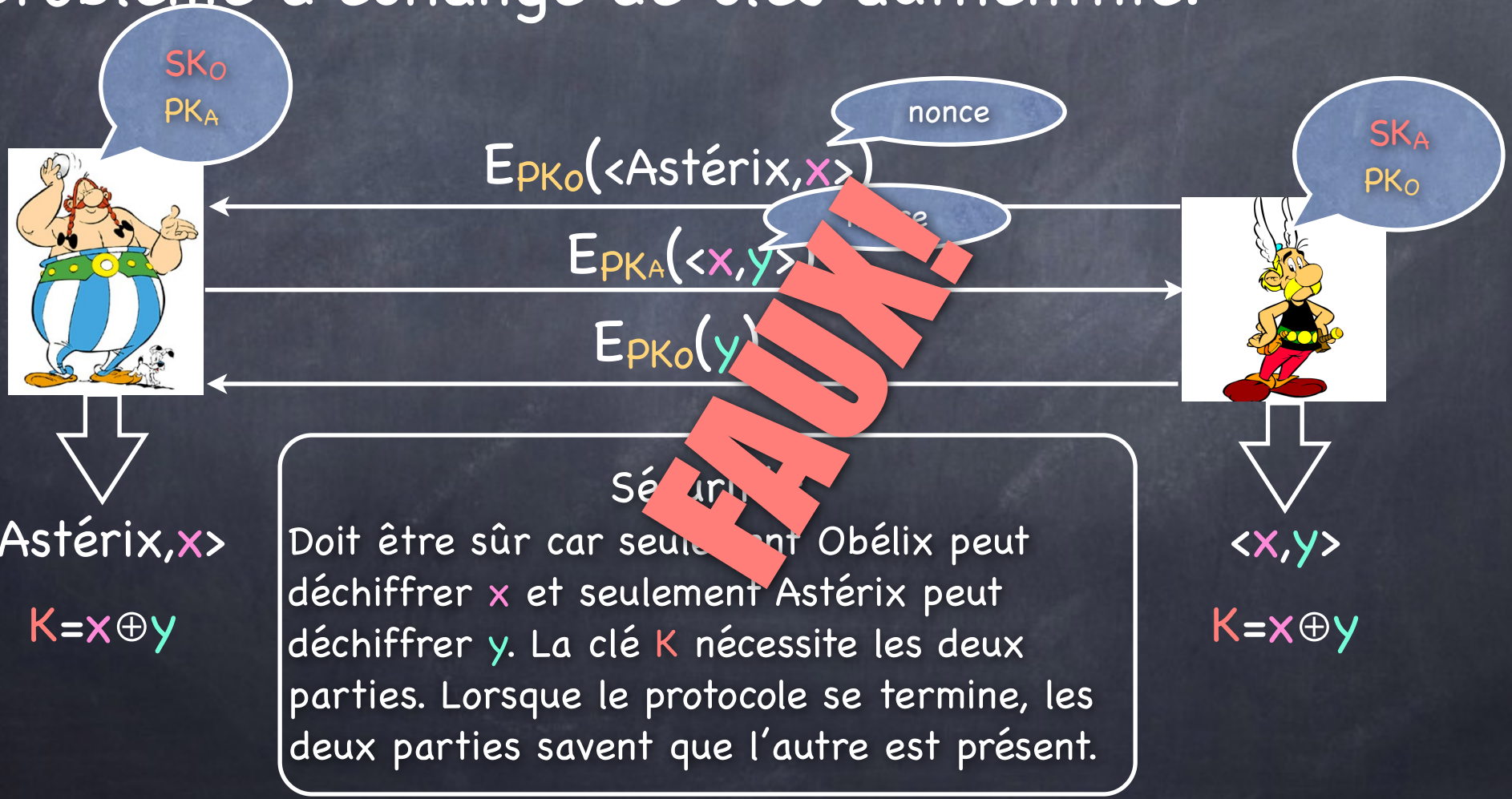
$E_{PK}(\langle Astérix, K \rangle)$

3. Si le protocole est appliqué avec succès alors la clé K est nouvelle, aléatoire et indépendante de tout le reste. ✗

2. À moins que les méthodes cryptographiques soient cassées, César ne peut forcer une situation où Obélix croit qu'il partage K avec Astérix, mais en fait la partage avec César. ✗

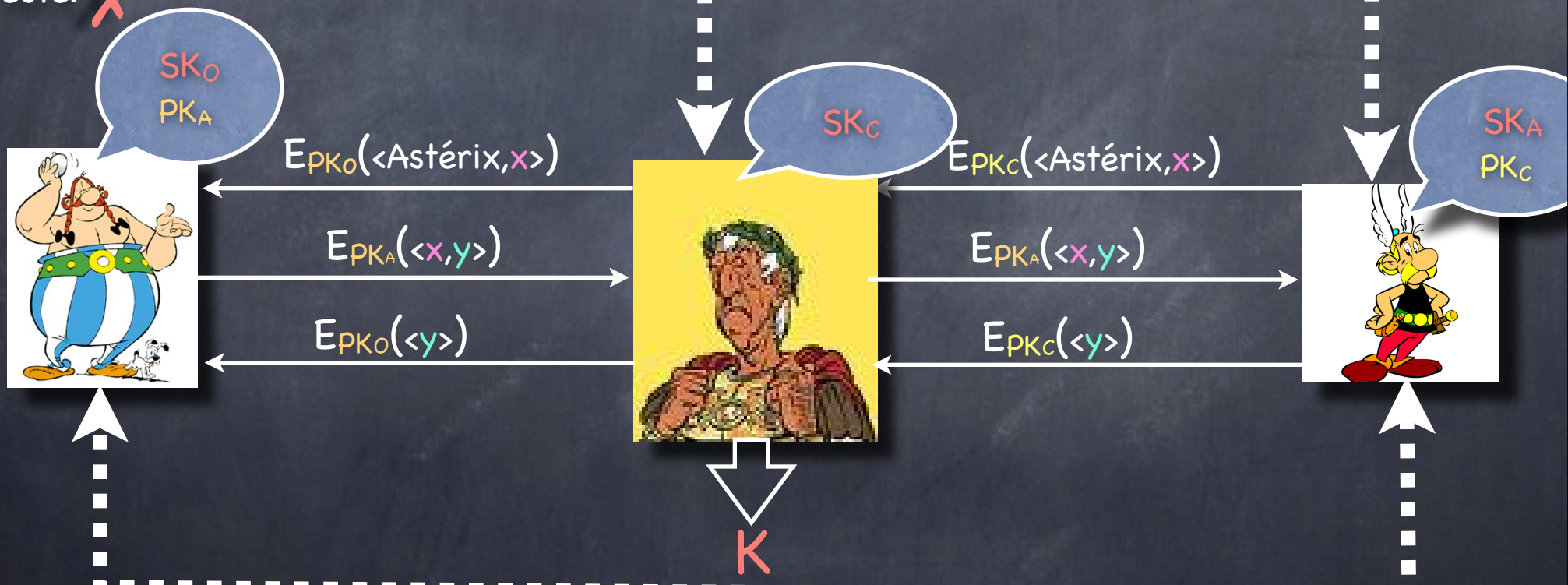
Le protocole de Needham et Schroeder

Voici un exemple d'un protocole proposé en 1978 pour le problème d'échange de clés authentifié.



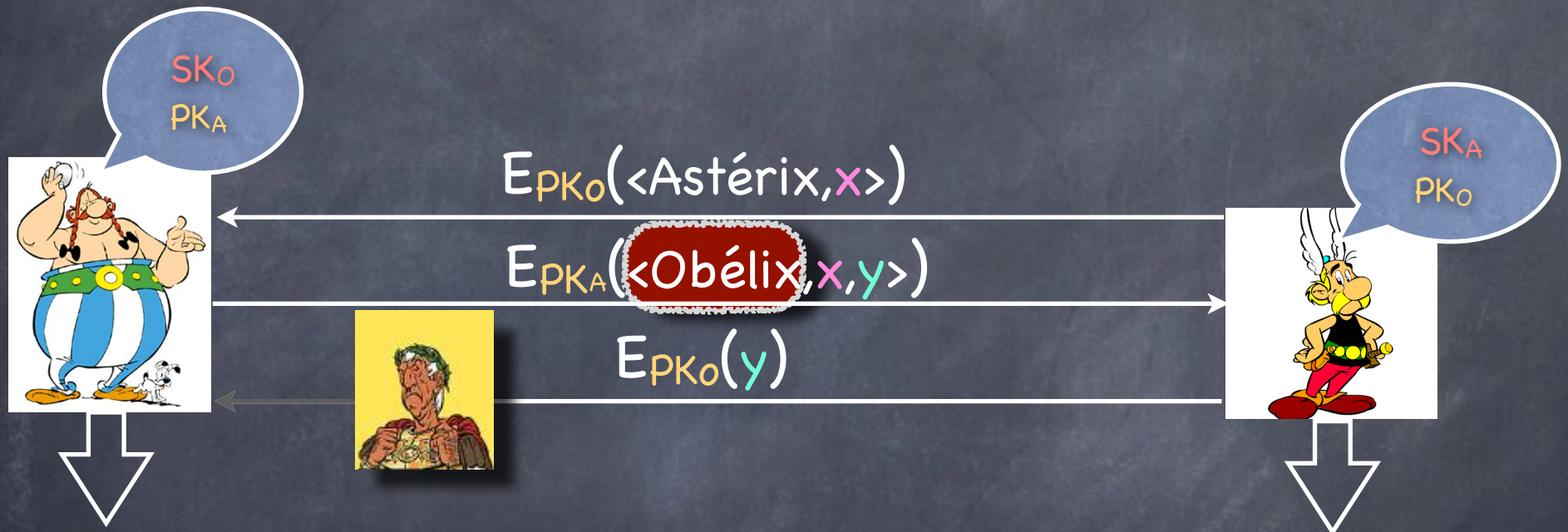
Casser Needham-Schroeder

2. À moins que les méthodes cryptographiques soient cassées, César ne peut forcer une situation où Obélix croit qu'il partage K avec Astérix mais en fait la partage avec César. ✗
3. Si le protocole est appliqué avec succès, alors la clé K est nouvelle, aléatoire et indépendante de tout le reste. ✗



Réparer Needham-Schroeder

Needham-Schroeder-Lowe

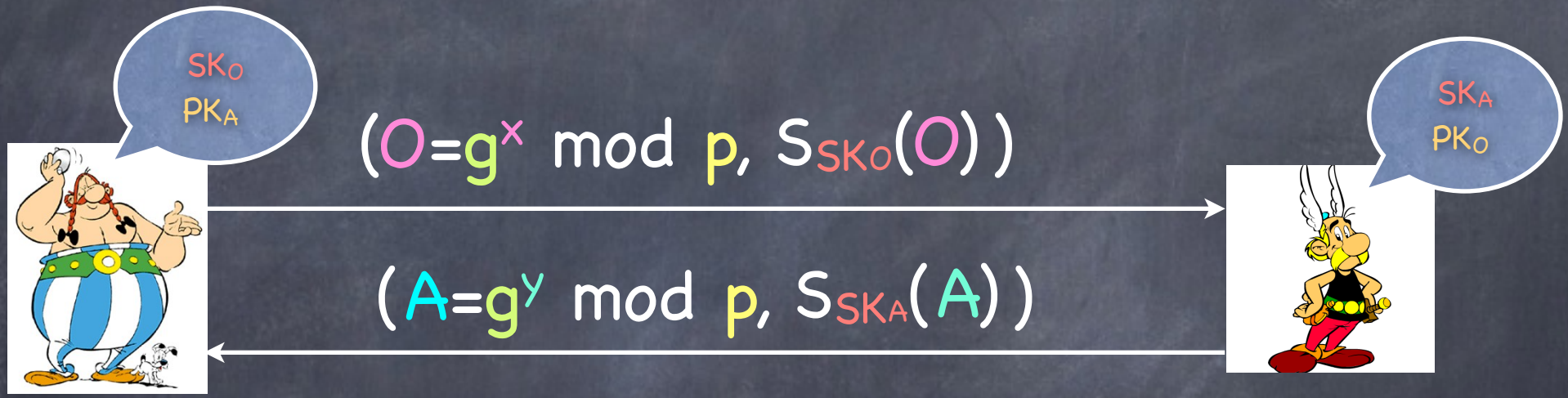


Notez que César peut tout de même stopper la dernière transmission de telle façon qu'Astérix croit qu'il partage une clé avec Obélix tandis qu'Obélix n'en croit rien.

Ceci est impossible à éviter!!!!!

Réparer Diffie-Hellman

- Supposons qu'Obélix et Astérix ont des clés publiques certifiées pour un système de signatures numériques.



$x \in Z_{p-1}$

$K=A^x \bmod p$

Si la signature est vérifiée avec succès

$y \in Z_{p-1}$

$O^y \bmod p=K$

Si la signature est vérifiée avec succès

SSL

- La solution la plus utilisée aujourd'hui pour l'échange de clés authentifié est le «Secure Socket Layer» (SSL) [Netscape].
- L'approche est différente de Needham-Schroeder-Lowe car elle utilise un système de signatures numériques en plus du chiffrement. Essentiellement l'idée derrière le protocole de Diffie-Hellman authentifié vu précédemment.
- La dernière version de SSL semble être sûre contre l'attaque de l'homme du milieu.
- SSL est en voie d'être remplacé par TLS (Transport Security Layer). Les différences avec SSL sont minimes.

TCP/IP

- Pour comprendre SSL, il est utile de voir sur quoi il repose.
- IP (Internet Protocol) est le nom donné au protocole qui déploie le trafic sur l'Internet. Le trafic est transmis par petits paquets qui contiennent les adresses IP de l'expéditeur et du destinataire.
- TCP (Transmission Control Protocol) repose au-dessus de IP. Il est orienté connexion, ce qui signifie que certains types de paquets sont transmis et acceptés dans le but d'ouvrir une connexion. Les paquets qui suivent peuvent alors être reconnus comme appartenant à une connexion établie précédemment.

IP et TCP

- Le protocole IP est responsable du transport de paquets d'une adresse internet à une autre. Chacun des petits paquets sait où il va et d'où il vient (les adresses IP). Deux paquets envoyés par le protocole IP ne sont pas nécessairement reçus dans le même ordre.
- Le protocole TCP utilise le protocole IP pour établir des connexions. Ceci permet de réordonner les paquets dans l'ordre d'expédition. S'il y a perte d'un paquet, celui-ci sera demandé à nouveau.
- TCP/IP est responsable de la communication à travers une connexion entre deux adresses IP.
- L'architecture de TCP/IP est par couches («layers»).

La pile TCP/IP

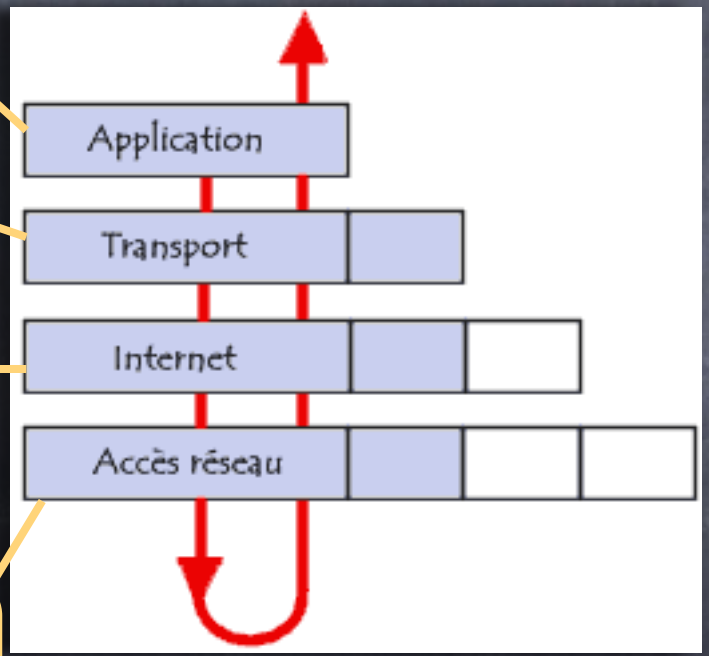
- L'ensemble des couches TCP/IP est appelé pile TCP/IP. La pile TCP/IP contient plusieurs couches et protocoles.

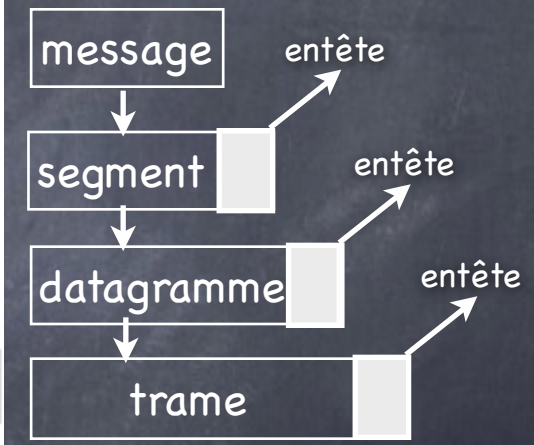
Ensemble de services réseau faisant l'interface entre le réseau et le système d'exploitation. HTTP, FTP, SMTP.

Permet d'identifier les programmes sur lesquels les données transportées roulent. Réalise les protocoles TCP et UDP.

Responsable de l'intégration des adresses IP. La transport de données vers des machines distantes. Forme des datagrammes qui contiennent les données ainsi que les adresses IP...

Première couche de la pile. Responsable de la transmission des données via un réseau physique arbitraire. Achemine les données, coordonne la transmission, formate, conversion analogique/numérique, contrôle d'erreurs. Entre machines voisines.

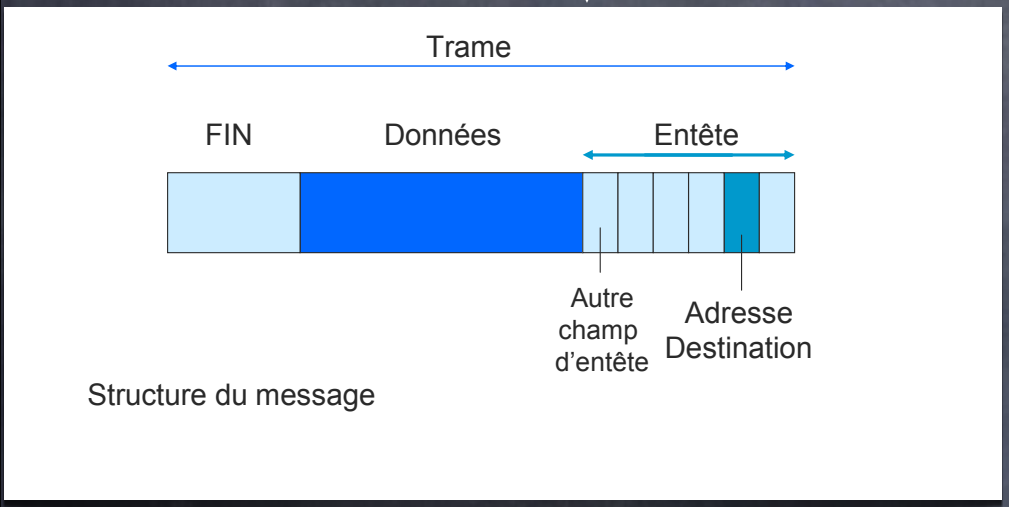




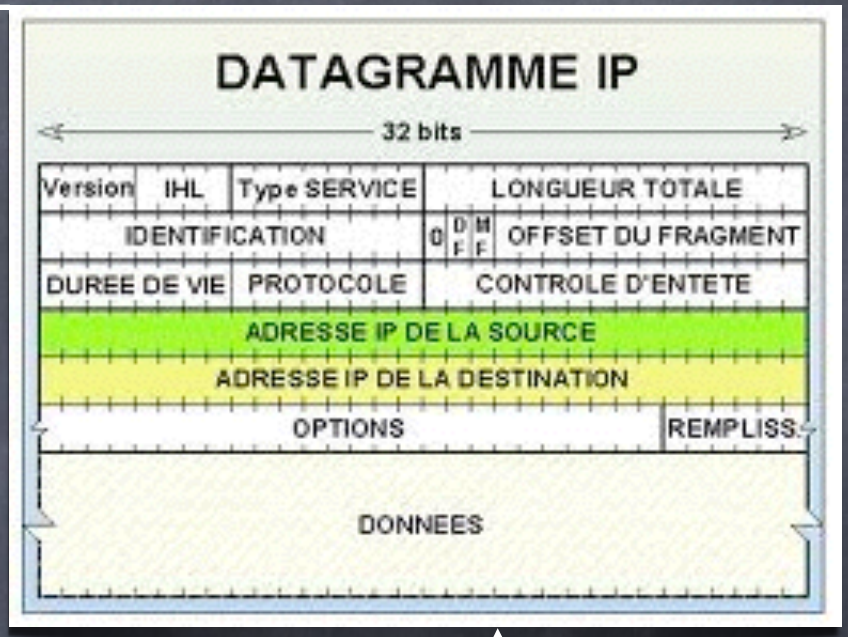
Trames-Datagrammes-Segments

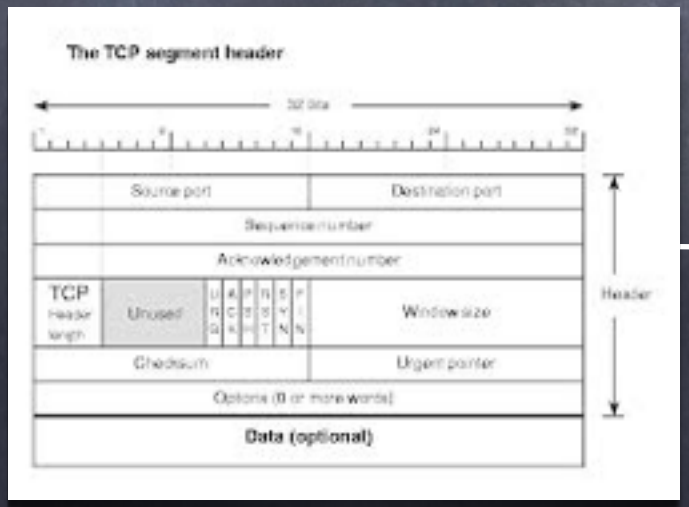
2

Fragmentation (en paquets d'environ 1500 octets)



Structure du message





1

La couche SSL

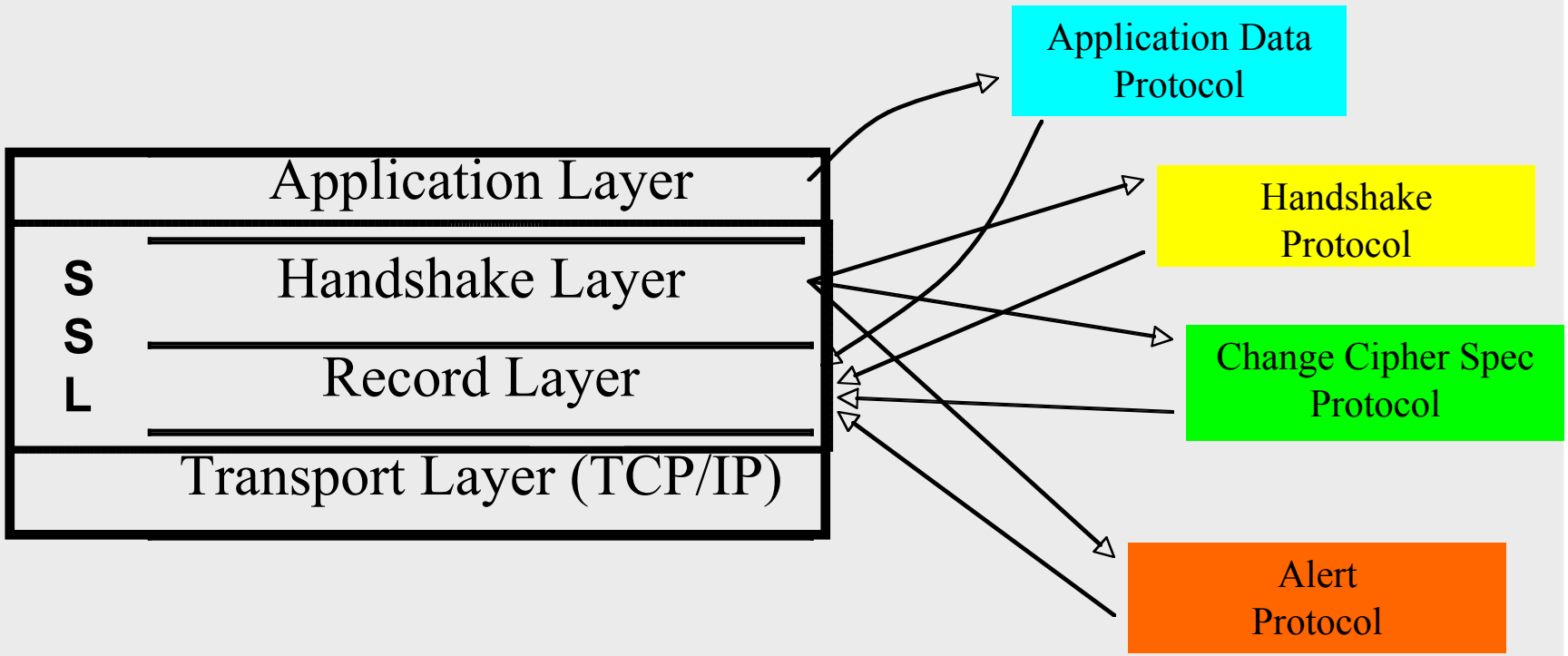
- La couche SSL est localisée entre la couche de transport et la couche d'application de la pile TCP/IP.
- Les données transmises par SSL sont incluses dans les paquets TCP/IP pour être acheminées au bon endroit par la connexion TCP.
- Cependant, les données ne peuvent être manipulées que par des clients et serveurs compatibles avec SSL.
- SSL est toujours exécuté entre un client et un serveur. Il demande que les deux entités aient des clés publiques certifiées avec la clé privée correspondante.
- Chaque entité doit pouvoir vérifier les certificats. Il doit exister une chaîne qui connecte les deux entités.
- SSL peut aussi être exécuté d'un seul côté, où seul le serveur détient un certificat.

Les protocoles SSL

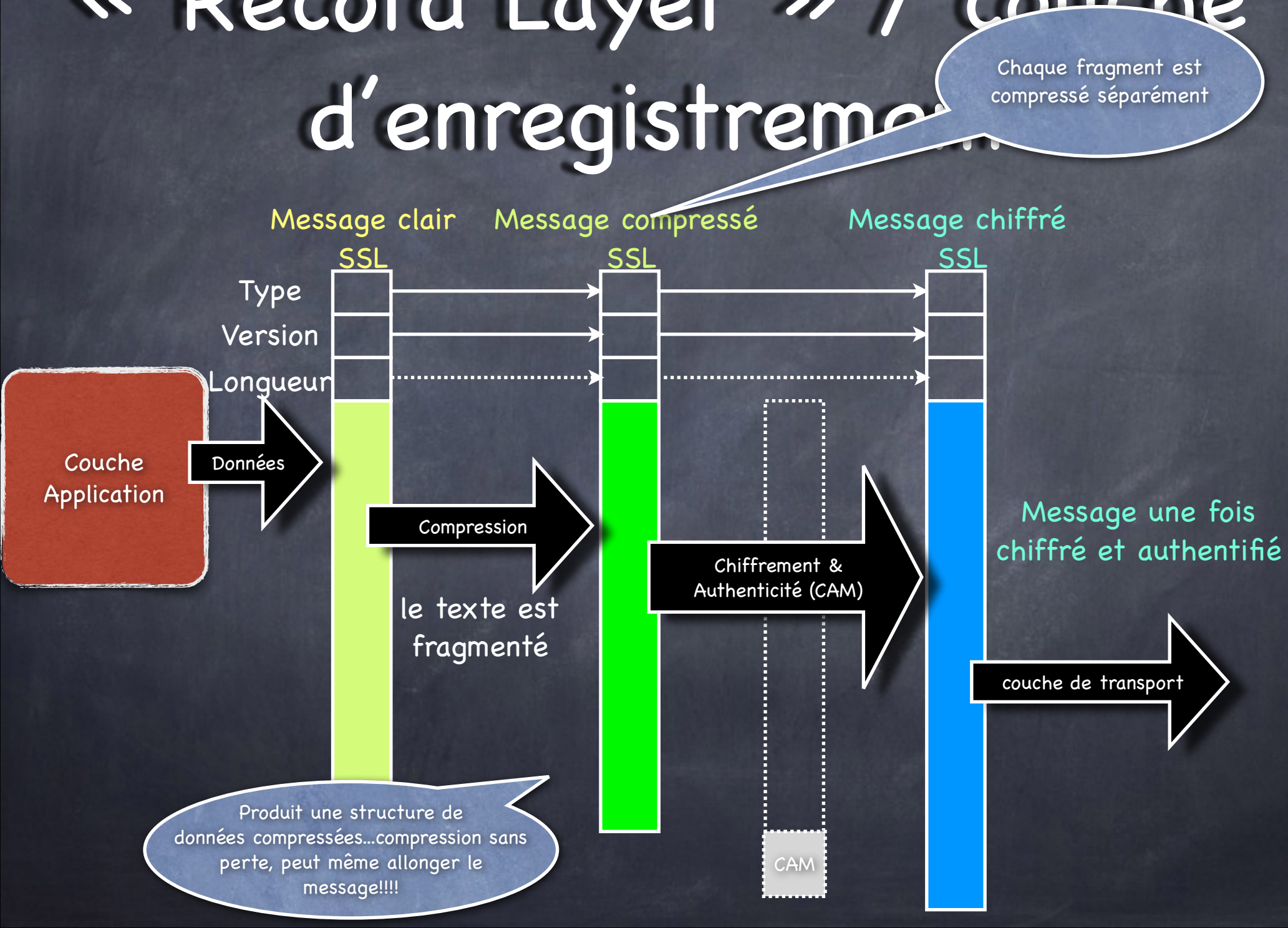
SSL est composé des protocoles suivants :

- Record Protocol : Responsable du transport des données brutes. Il communique avec la couche transport. Il chiffre et authentifie (calcule un CAM) l'information en utilisant le cipher_spec. Si le cipher_spec est vide, alors l'information n'est ni chiffrée ni authentifiée.
- Handshake Protocol : Responsable de l'échange de clés authentifiées. Le but de ce protocole est d'amener le client et le serveur qui ne partagent pas de clés à s'entendre sur un cipher_spec, choisi parmi ceux voulus par le client, avant d'exécuter l'échange de clés authentifiées.
- ChangeCipher Protocol : Un message ou une entité annonce à l'autre un changement de cipher_spec tel que négocié précédemment.
- Alert Protocol : Un message indiquant une erreur.

L'anatomie de SSL



« Record Layer » / couche d'enregistrement...



Calcul du CAM

- Noter que SSL chiffre le message avec son CAM.
- Le CAM calculé par SSL dans la phase de chiffrement & authentification utilise HMAC avec un choix de deux fonctions de hachage, SHA1 ou MD5 :
  - MAC = h(SK⊕pad2 || h(SK⊕pad1 || nb_seq || lg || texte))
  - pad1= 0x36 répété 40 fois pour h=SHA1 et 48 fois pour h=MD5.
  - pad2=0x5c comme pour pad1.

Protocole d'établissement de liaison (« handshake »)

Serveur

client allo

serveur allo

certificat

échange de clé

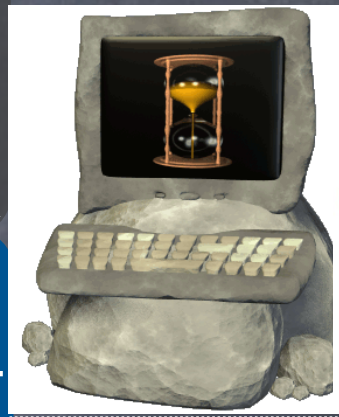
fin allo

certificat

échange de clé

fini

fini



serveur allo :
-sélection de version SSL
-sélection d'algos
-un nonce

certificat :
-Transmet certificat X509
-Demande certificat client (rare)

échange de clé I :
Message complémentaire pour l'échange de clé

fini :
-change cipher_spec

fini :
-change cipher_spec

client allo :
-la version SSL maximale
-liste d'algos supportés
-un nonce

certificat
-certificat éventuel client
-signature de la discussion

échange de clé II :
clé pré-master chiffrée

Échange de clé SSL

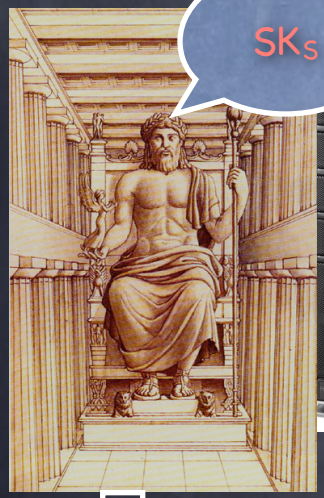
Voici comment l'échange de clé fonctionne (grosso modo) en éliminant les détails. Les opérations sont distribuées en plusieurs phases :

$Cert_O$ OK?
Signature OK?
déchiffre pms

$F(n_o,n_s,pms)=MD5(pms||SHA1(\text{'A'}||pms||n_o||n_s))\ ||\ MD5(pms||SHA1(\text{'BB'}||pms||n_o||n_s))\ ||\ MD5(pms||SHA1(\text{'CCC'}||pms||n_o||n_s))$

$Cert_S$ OK?

Serveur



SK_S

'allo', n_o

n_s, $Cert_S(PK_S)$

optionnel!

$Cert_O(PK_O)$,$E_{PK_S}(pms)$,$S_{SK_O}(n_o,n_s, E_{PK_S}(pms))$

'fini'+CAM_{pms}(conversation)

'fini'+CAM_{pms}(conversation)

SK_O

$ms=F(n_o,n_s,pms)$

Longueur appropriée pour le chiffre spécifié

En fait, deux clés : S->O et O->S.

Longueur appropriée pour le CAM spécifié

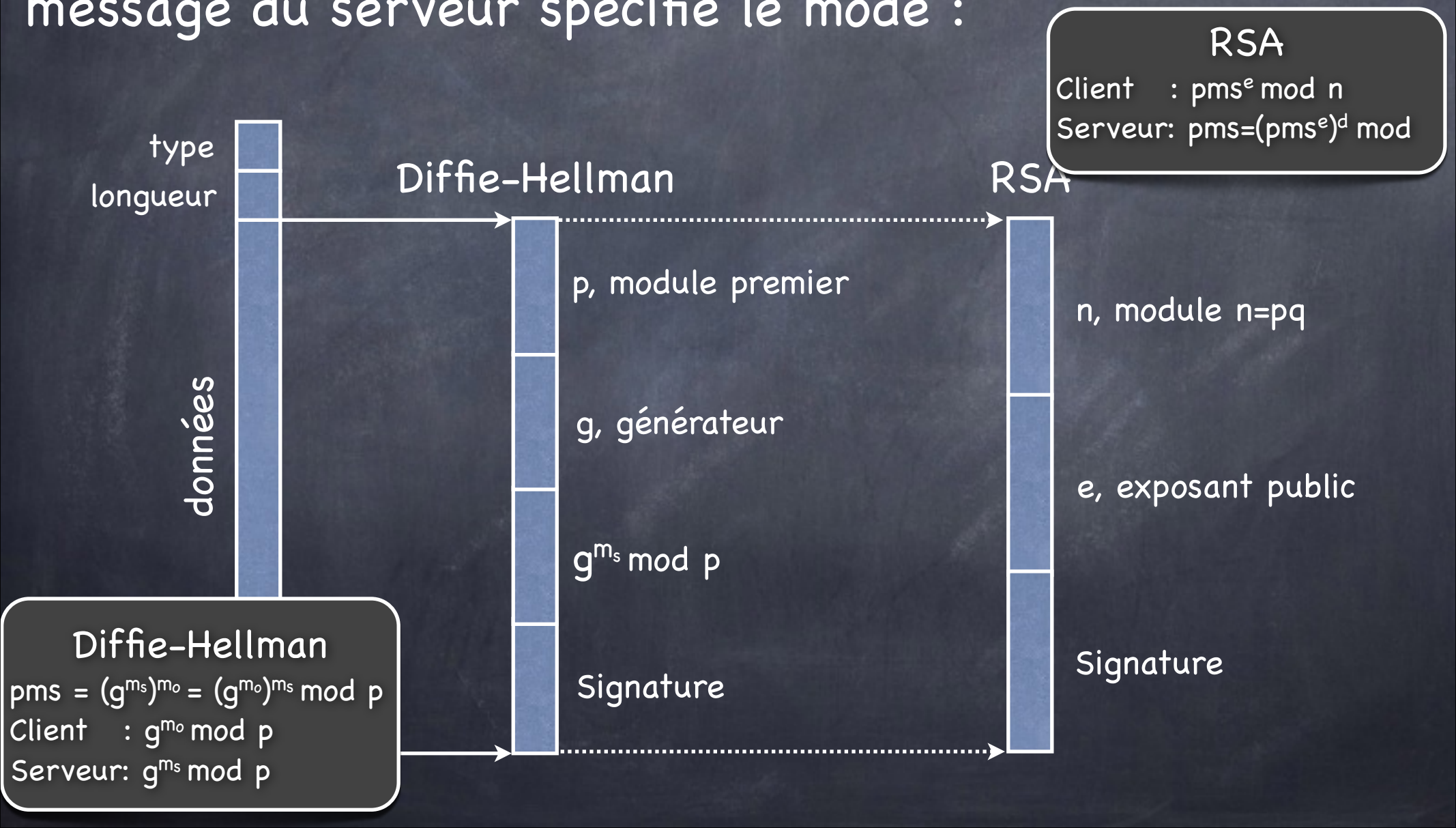
$ms=F(n_o,n_s,pms)$

SK chiffrement	SK CAM

ms

Échange de clé : serveur

Diffie-Hellman et RSA peuvent être utilisés par SSL. Le message du serveur spécifie le mode :



Sécurité de l'échange de clé contre l'homme du milieu

- Rien ne montre que l'échange de clé SSL est sûr. En particulier, est-ce que l'homme du milieu peut compromettre l'échange comme c'est le cas pour Needham & Schroeder?
- La preuve de sécurité n'est pas chose facile. Nous n'allons que montrer l'intuition derrière le protocole.
- Voyons deux cas :
  - S' veut personnifier S en jouant l'homme du milieu avec un client C.
  - C' veut personnifier C en jouant l'homme du milieu avec un serveur S.

L'homme du milieu voulant personnifier le serveur

César (S') voudrait utiliser S pour terminer une session avec Obélix en lui faisant croire qu'il est S. César doit authentifier les transmissions avec Obélix. Mais S ne transmet jamais pms, même chiffré. Il est donc réduit à retransmettre la même chose que S à Obélix. César ne peut réussir en se comportant ainsi, il n'est qu'un canal entre S et Obélix!

Aidons César : supposons qu'Obélix n'a pas de clé publique pour signer! (possible)

Server

E_{PK_S}(pms)

'fini'+CAM_{pms}(conversation)

'fini'+CAM_{pms}(conversation)

E_{PK_S}(pms),S_{SK_O}(n_O,n_S,E_{PK_S}(pms))

'fini'+CAM_{pms}(conversation)

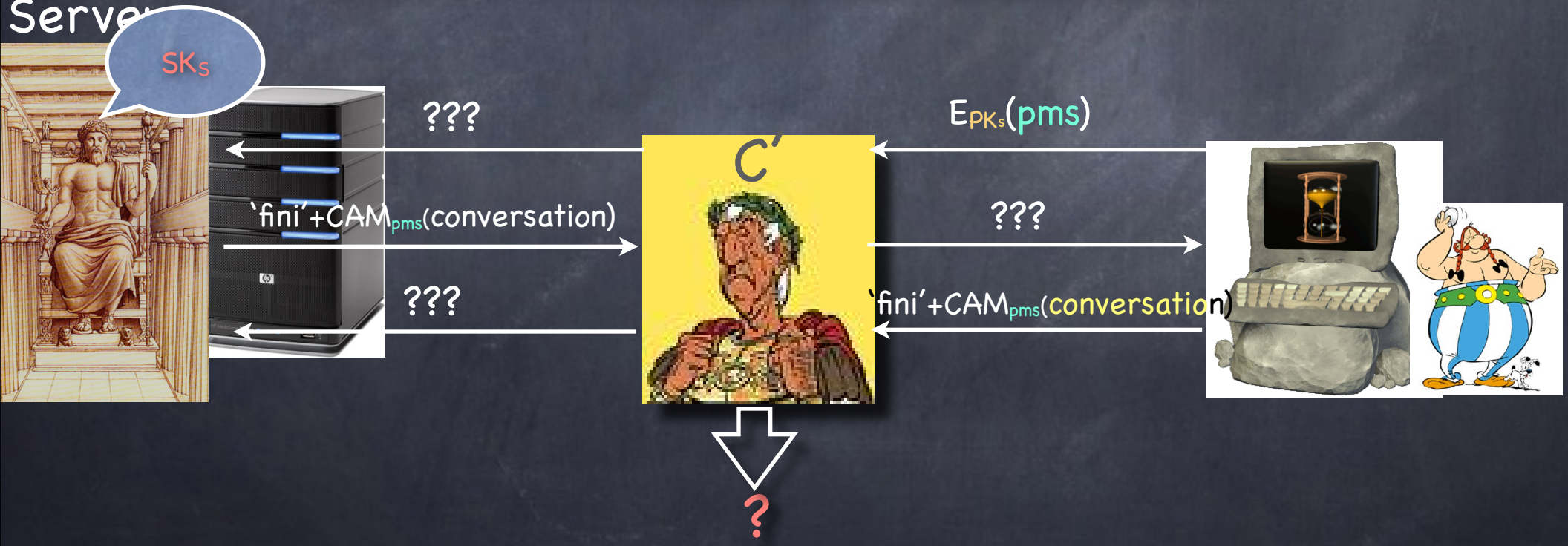
'fini'+CAM_{pms}(conversation)

Nous pouvons aussi montrer que César ne réussira pas plus s'il débute une session avec S jouant le client. Dans ce cas César choisira pms ce qui ne peut l'aider contre Obélix.

?

L'homme du milieu voulant personnifier le client

César (C') voudrait personnifier Obélix auprès de S. César doit authentifier les transmissions avec Obélix y compris le chiffrement de pms choisi par celui-ci. Une fois de plus, César ne sera qu'un canal entre S et Obélix, car il ne sait pas produire un message final valide sans connaître pms.



SSL sans authentification du client

- Comme nous venons de le voir, un client SSL n'est pas obligé de s'authentifier. SSL permet des sessions où seulement le serveur s'authentifie au client mais pas l'inverse.
- Dans bien des situations, c'est la meilleure chose possible, car les clients n'ont pas nécessairement de clés publiques certifiées.
- Dans cette situation, le client sait qu'il parle au bon serveur, mais le serveur ne sait rien de l'identité du client.
- L'identification du client peut alors se faire (comme c'est le cas dans la pratique) en demandant un mot de passe.

L'authentification du client par mot de passe

- Supposons qu'un serveur WEB détient une liste d'utilisateurs + mots de passe mais les utilisateurs n'ont pas de clés publiques certifiées. Seul le serveur possède une telle clé publique.
- Le serveur veut authentifier un utilisateur qui se connecte.
- Comment faire?
  - Ouvrir une session SSL,
  - Demander à l'utilisateur de fournir son mot de passe.

Limites de l'approche par mot de passe

- Le client ne peut s'authentifier qu'au serveur qui connaît un mot de passe pour celui-ci.
- La sécurité d'une telle session est plus difficile à établir, puisque la gestion des mots de passe n'est pas intégrée au protocole mais arrive plus tard.
- De tels systèmes sont aussi sûrs que le mot de passe. Donc, il semble qu'un système qui établirait l'identité d'un client basé sur un mot de passe dès le départ serait au moins aussi bon :
  - Plus précisément, nous pourrions avoir un protocole d'établissement de liaison qui ne puisse être compromis plus rapidement que le temps nécessaire pour deviner le mot de passe en ligne.
  - Rien de mieux ne peut être obtenu.
  - Un tel échange de clés est dit « échange de clés authentifié par mots de passe ».

Échange de clés authentifié par mot de passe

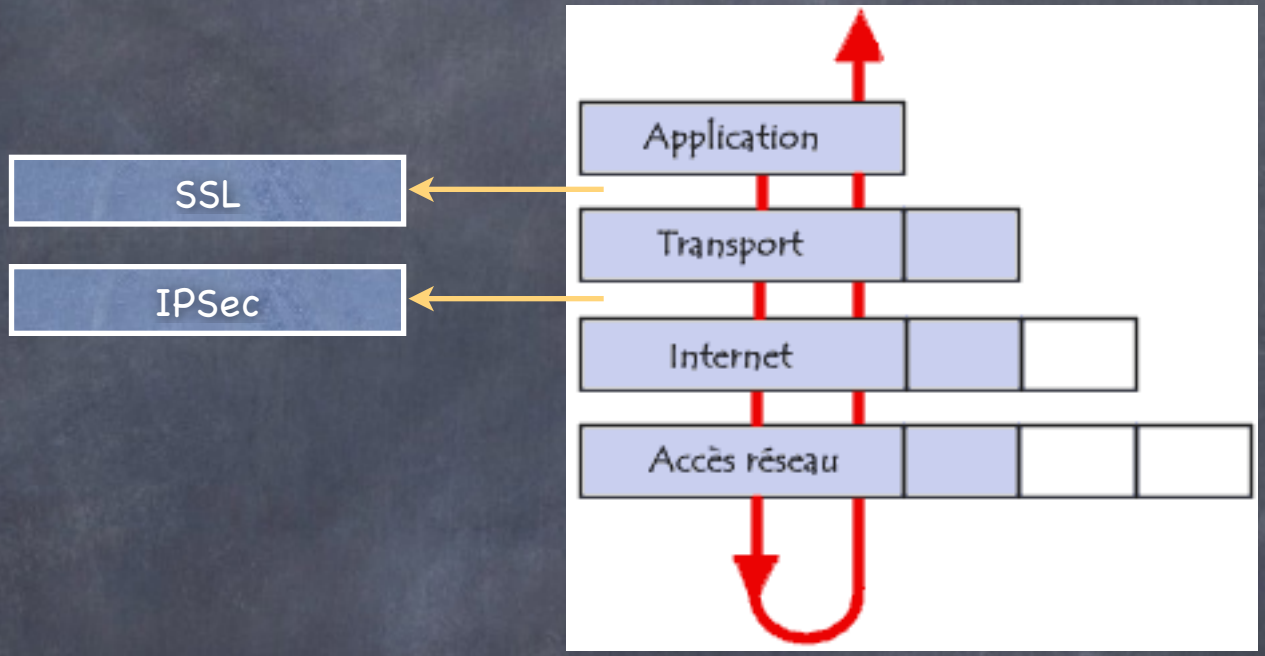
- Ceci semble facile. Les deux parties connaissent un secret pw :
  - Nous n'avons qu'à dériver une clé secrète directement à partir de pw ou même utiliser pw directement pour le chiffrement et le calcul de CAM...
  - L'utilisation à long terme sur beaucoup de données cause des problèmes de sécurité.
  - Les mots de passe doivent être assez courts pour être faciles à mémoriser. Le nombre de mots de passe possibles est donc beaucoup plus petit que le nombre de possibilités pour la clé secrète d'un chiffrement symétrique.
  - L'adversaire, ayant intercepté un cryptogramme, peut donc tenter le déchiffrement en essayant les mots de passe possibles hors ligne...tandis que deviner le mot de passe en ligne peut résulter en une désactivation du compte après quelques essais infructueux.
- Ce problème difficile a des solutions connues (p.ex. Bellare et al. Eurocrypt'2000) . Quelques-uns (sans preuve formelle de sécurité) sont considérés pour standardisation par l'Internet Engineering Task Force et IEEE. Ne sont pas utilisés par les produits commerciaux...

TLS

- SSL (1994) est remplacé par le Transport Layer Security (1999).
- TLS est très semblable à SSL, les différences sont minimes.
- Règle des problèmes de sécurité qui sont apparus au sujet des fonctions de hachage cryptographiques SHA1 et MD5. Les fonctions HMAC-SHA et HMAC-MD5 sont maintenant utilisées. TLS peut aussi fonctionner comme SSLv3 si nécessaire.
- TLS offre aussi Kerberos.
- TLS-PSK utilise des clés déjà partagées pour l'échange de clés au lieu d'une infrastructure à clé publique (pas de mot de passe). Permet l'utilisation de TLS dans des environnements où la crypto à clé publique n'est pas possible.
- SSL se limitait à des clés symétriques de 40 bits pour se conformer à la loi américaine sur l'exportation de technologie cryptographique. Après des recours en justice et la reconnaissance que la cryptographie étrangère utilise aussi des clés plus longues, les autorités ont assoupli des aspects de la loi.

IPSec

- IPSec est un ensemble de protocoles qui accomplit des fonctions semblables à SSL mais situé à un autre endroit de la pile TCP/IP.



- IPSec est de plus bas niveau que SSL/TLS. Le résultat est ce qui est appelé une association de sécurité entre deux adresses IP.
- Étant de plus bas niveau implique que même les informations de contrôle, les numéros de séquences utilisés pour gérer la connexion TCP, sont chiffrés.

Tunnels IPSec

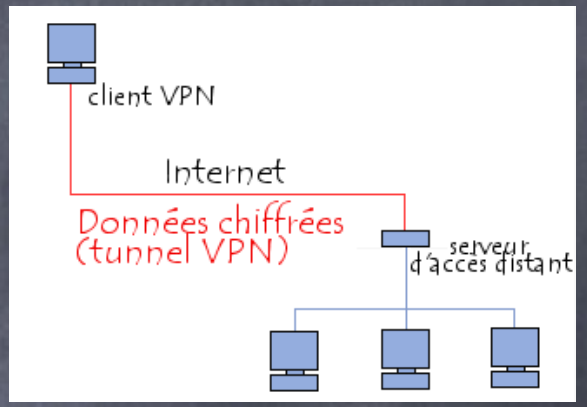
- Un mode d'échange de données IPSec est appelé tunnel. Un tunnel permet de chiffrer les entêtes des paquets IP. SSL ne peut faire ceci, car ces entêtes sont inclus plus bas dans la pile TCP/IP.
  - Le résultat est qu'un adversaire observant un lien ne peut déterminer la quantité de données qui transite entre deux adresses IP.
- IPSec utilise un protocole d'échange de clés appelé IKE (Internet Key Exchange). Ceci désigne plusieurs méthodes possibles. Des certificats à clés publiques sont la plupart du temps nécessaires comme pour SSL/TLS.
  - Le mécanisme principal est basé sur Diffie-Hellman protégé contre l'homme du milieu.

SSL versus IPSec

- IPSec demande qu'un client soit installé sur chaque unité pour permettre le chiffrement. Sa configuration est plus difficile que pour SSL (transparent).
- SSL est inclus dans les navigateurs, ce qui ne nécessite pas l'installation de clients.
- SSL fonctionne plutôt pour des applications Web, ce qui n'est pas le cas d'IPSec.
  - IPSec est utilisé pour les communications site à site qui ne sont pas de type client-serveur.
- Certains routeurs NAT (Network Adress Translation) ont de la difficulté avec IPSec. SSL n'a pas ce problème.
- Certains fournisseurs d'accès Internet demandent d'être payés pour des communications IPSec. C'est impossible à faire avec SSL.
- Si IPSecv6 devenait supporté par les applications, alors SSL pourrait devenir inutile.

Applications

- Secure HTTP est le nom donné à l'utilisation de SSL ou TLS pour établir une connexion sûre entre un serveur WEB et son client. Le résultat est un tunnel entre le client et le serveur, par lequel chaque transmission est chiffrée sans que l'application ait à s'en soucier :
  - https://www.abc.def.com



- VPN (Virtual Private Network) permet d'établir des connexions sûres entre un client distant et un réseau local. Durant la phase d'initialisation entre le client et la passerelle VPN, celui-ci se voit attribuer une adresse IP comme s'il était membre du réseau local.
  - Une fois initialisées, toutes les communications sont chiffrées et authentifiées. Pour ce faire, un échange de clé est nécessaire. Des VPN le réalisent avec leurs propres méthodes tandis que d'autres utilisent IPSec ou SSL/TLS. VPN permet de travailler comme si le client faisait partie du réseau local. Des employés peuvent travailler à la maison avec les mêmes privilèges qu'au bureau.

Applications de haut niveau

- Les protocoles que nous avons rencontrés sont de bas niveau. Ils ne sont pas bien adaptés aux humains!
  - SSL ne peut être utilisé pour signer des courriels ou documents, car SSL ne reconnaît pas le concept de document.
- La cryptographie appliquée à des objets de plus haut niveau est offerte par d'autres solutions.
  - MIME : un standard pour la transmission des courriels. S/MIME est un standard avec des champs supplémentaires pour la signature et le chiffrement de courriels.
  - XML : un standard qui propose une extension de HTML qui, en particulier, permet de chiffrer et de signer des pages Web.
  - PGP (Pretty Good Privacy) : permet de signer et de chiffrer des courriels. Au lieu de baser la sécurité sur des CA, il utilise un concept de toile de confiance («web of trust»). Votre ami(e) peut vous envoyer une clé publique d'une troisième personne qui déclare que ceux-ci font confiance à cette clé. Vous pouvez décider de faire confiance à cette clé si suffisamment de vos amis l'ont recommandée.

Petits exercices (I)

- L'unité de cryptographique IBM 4753 contient une clé maîtresse MK unique à chaque unité et rangée de façon sûre à l'intérieur. Cette clé a 112 bits, pour son utilisation avec 3DES à deux clés.

- IBM 4753 doit aussi avoir d'autres clés en plus de MK (pour signer, chiffrer, authentifier,...). Celles-ci sont rangées chiffrées selon MK sur une unité externe pas nécessairement sûre.

- L'architecture de l'unité interdit d'utiliser ses clés pour d'autres applications que celles prévues. À chaque clé est associé un vecteur de contrôle VC. Il s'agit d'une chaîne de bits spécifiant les applications qui peuvent utiliser cette clé.

- Il est très important que l'unité ne puisse être amenée à utiliser une clé de la mauvaise façon. VC doit donc être aussi rangé avec la clé K. Une solution naturelle aurait été d'ajouter un CAM pour ($E_{MK}(K)$, VC). IBM n'a pas voulu utiliser d'espace supplémentaire. IBM voulait une solution où VC est utilisé pour les chiffrements et déchiffrements, de sorte que s'il est modifié, alors le résultat est inutile lorsque la clé est déchiffrée dans le 4753. Voici des solutions possibles :

a) $E_{MK}(K) \oplus VC$, VC

b) $E_{MK}(K \oplus VC)$, VC

c) $E_{MK \oplus VC}(K)$, VC

Peut être sûr, mais la distance entre la clé déchiffrée et K est connue lorsque VC est changé.

C'est le choix d'IBM

Petits exercices (II)

- Le NIP de votre carte bancaire (lorsqu'entré depuis un terminal) est transmis chiffré au serveur central pour vérification. Le serveur a, sur son disque, une base de données avec les (numcarte, NIPchiffré) où les NIP sont tous chiffrés avec la même clé connue seulement d'une unité sûre. Cette unité déchiffre de façon interne le NIP de l'utilisateur et déchiffre l'entrée de la base de données associée au numcarte de l'utilisateur.
- Il y a 2 si
  - ISO-0 où NIPchiffré = E_K(nip)
  - ISO-1 où NIPchiffré = E_K(nip+numcarte).
- Comparer les attaques possibles par des employées ayant accès au serveur et pouvant modifier les données. Comment faire ceci de façon plus intelligente?

Considérez :
(n1, E_K(n1+nip))
(n2, E_K(n2+n1)) ---» (n1, E_K(n2+n1))

Ceci est meilleur (mais loin d'être bon) mais le ou exclusif ne doit pas être calculé mod 10 mais plutôt mod 26 par exemple! Pourquoi?

Mauvais client-employé N choisit NIP=-N...les clients de la banque sont floués