Licence de ce document

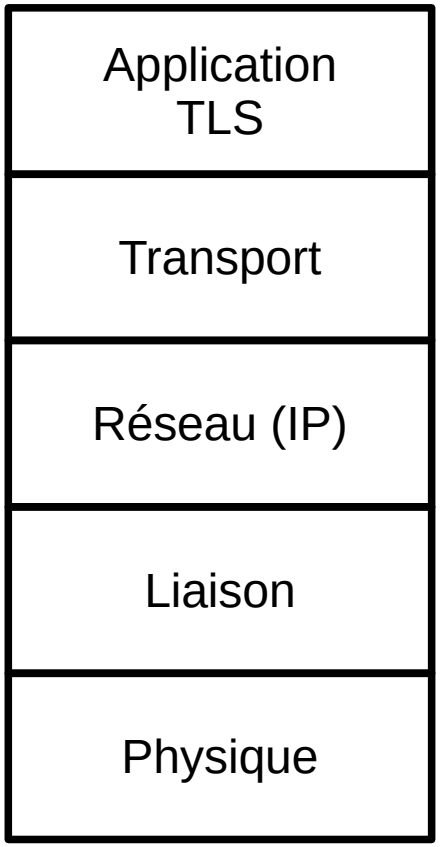
- licence créative common BY-SA
- cf http://creativecommons.org/licenses/by-sa/4.0/
- pour réaliser ce document, j'ai utilisé le travail d'autres personnes que je remercie et qui sont citées :
  - soit directement si leur travail est utilisé tel quel
  - soit en bibliographie si leur travail m'a permis d'améliorer ma compréhension du domaine sans être utilisé directement
- un retour normal est de permettre à quiconque d'utiliser les éléments de ce document librement :
  - à condition de me citer
  - et de respecter les mêmes conditions d'utilisation
- parmi les sources utilisées, remerciements particuliers à
  - S. Bortzmeyer pour son blog (et ses tee-shirts :-)) : http://www.bortzmeyer.org/
  - V. Bernat pour son article sur PFS

TLS/SSL, HTTPS

- 1994-1996 : développement de SSL par Netscape
  - 1994, SSL 1.0 (non diffusé)
  - Février 1995, SSL 2.0 (http://www.frameip.com/rfc/draftxxx.php)
  - novembre 1996, SSL 3.0 (http://www.frameip.com/rfc/draft302.php)
- 1995, création de Thawte et de Vérisign
- Janvier 1999, RFC 2246, TLS 1.0 (~ SSL V3.1)
- Avril 2006, RFC 4346, TLS 1.1
- Aout 2008 : RFC 5246, TLS 1.2

TLS et couches

- modèle TCP/IP : entre la couche application et la couche transport
  - transparent pour l'utilisateur : protocoles de couche application non modifiés
  - doit être géré spécifiquement par les applications (navigateurs, ...)
- ipsec, les vpn ssl sont plus transparents pour les applications



TLS : services fournis

- modèle client/serveur
- confidentialité via un tunnel chiffré entre client et serveur
- Authentification du serveur (à partir de ssl v2)
- intégrité et identification de la source des données
- Authentification du client (optionnelle)

TLS : ce qu'il ne fait pas

- TLS ; c'est de l'informatique, pas de la magie. Il ne protège pas
  - si un programme espion tourne sur votre poste de travail
  - si le serveur est compromis
  - si les données envoyées sont compromises plus tard par l'incompétence de la société qui les gèrent
  - si vous ignorez les alertes sur le fait que le certificat du site distant est obsolète ou ne correspond pas au site ou est autosigné
  - exemple de compromission de données : Sony en 2011, 77 millions de comptes concernées (http://www.pcinpact.com/news/63279-sony-psn-qriocity-vol-donnees-bancaires-personnelles.htm)
  - 12/2013, 40 millions d'informations bancaires volées à la chaîne de grande distribution Target (USA) http://www.zdnet.fr/actualites/donnees-privees-le-piratage-de-target-encore-plus-grave-qu-annonce-39796580.htm et http://www.zdnet.fr/actualites/donnees-privees-target-confirme-le-vol-des-codes-pin-chiffres-39796624.htm

TLS

- https
- des protocoles ont une version « s »
  - pops, imaps, smtps, ldaps, ...
  - pb : affecter un port avec la version non « s » et un port à la version « s »
- Start TLS :
  - sur le port standard,
  - la commande start TLS démarre tls
  - exemples : pop+startTLS, imap+startTLS, smtp+startTLS, …
- identification et chiffrement sans TLS
  - exemples : dnssec, ssh

TLS : ports dédiés ou StartTLS

- certains protocoles ont un No de port dédié pour la version TLS :
  - smtp (25) et ssmtp(445)
  - http (80) et https (443)
  - pop3 (110) et pop3s (995)
  - ldap (389) et ldaps (636)
  - ...
- Pb : ca fait 2 ports par protocole

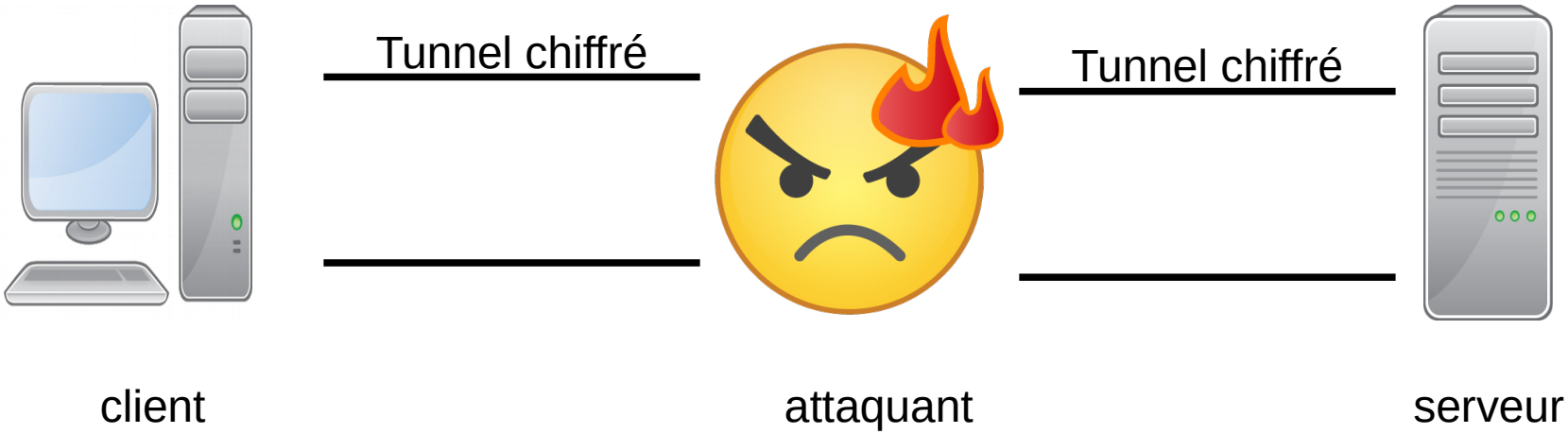
TLS : ports dédiés ou StartTLS

- Start TLS :
  - sur le port standard,
  - la commande start TLS démarre tls
  - exemples : pop+startTLS, imap+startTLS, smtp+startTLS, …
- on peut aussi avoir identification et chiffrement sans TLS
  - exemples : dnssec, ssh

Authentification du serveur

- via des certificats X509 (rfc5280, https://tools.ietf.org/html/rfc5280) ;
- le serveur a un couple clé privée/clef publique. La clef privée est connue seulement du serveur ;
- un certificat associe une identité à une clef publique ;
- fournie la certitude (ahem!) que c'est bien la clef publique du serveur
- indispensable pour garantir :
  - qu'on se connecte sur le bon serveur et pas sur un serveur pirate (hameçonnage)
  - pour éviter les attaques « Man In the Middle » (MiM)

authentification serveur : MiM



Sans authentification du serveur, un attaquant peut se faire passer pour le serveur et en même temps, se connecter au serveur

- la clef publique qu'il envoie au client est la sienne (et pas celle du serveur)
- il espionne tout le trafic
- il peut modifier les données transmises

Les donnée sont chiffrées entre le client et l'attaquant et entre l'attaquant et le serveur

authentifier un serveur

- Crytpo asymétrique (à clef publique) :
  - le serveur a une clef privée connue de lui seul
  - la clef publique peut être donnée à tous
  - pour s'authentifier, le serveur démontre qu'il a la clef privée correspondant à la clef publique
- Problème :
  - la clef publique est-elle bien celle du serveur ?
  - si un attaquant nous fait croire qu'une clef publique est celle de du serveur, il peut se faire passer pour lui

Certifier une identité

Carte d'identité

Pascal PETIT
validité : 22/04/2011

Certifier une identité

certifier une identité

- les papiers d'identité permettent de certifier une identité
- la confiance repose sur :
  - l'émetteur du papier d'identité
  - les procédures qu'il suit (carte d'identité vs carte d'étudiant vs carte navigo vs bout de papier auto-imprimé)
  - l'impossibilité de créer de faux papiers
- de nombreux états, de nombreuses autorités de confiance
- peut-on faire confiance à un état si on a maille à partir avec ses services secrêts ?

Certificat X509

- l'association d'une clef publique et d'une identité
- certifiée directement ou indirectement par une autorité de certification
- autorités de certifications reconnues :
  - de base dans le navigateur (nombreuses)
  - ou ajoutées manuellement
- la confiance repose :
  - sur la solidité des protocoles de crypto (être capable de faire un faux certificat ~ faire des faux papiers)
  - le sérieux et la fiabilité d'autorités de certifications
- avis personnel : ça ne marche pas bien

Alternatives

- PGP et WeBofTrust
  - système décentralisé
  - ràf
- ssh :
  - des clefs publiques
  - transmises à la première connexion (ou par l'administrateur du poste) ou via le dns
- DANE :
  - certificats placés dans le dns
  - avantage : pour chaque identité, une autorité de certification unique

Exemples réels passés de problèmes avec les certificats

- les autorités ajoutées aux navigateurs et l'espionnage des entreprises
- idem mais via un vrai/faux certificat : quand l'état français joue avec le feu
- tunisie : quand une autorité de certification nationale aide les services secrets
- Diginotar : piratage d'une autorité de certification
- NSA:ràf

Man In the Middle

- objectif : déchiffrer le trafic chiffré pour
  - détecter les usages non autorisés (trop de facebook, ...)
  - passer les fichiers récupérés sur gmail & Co à l'antivirus
- en cas de Man In The Middle, le navigateur va protester :
  - si le certificat n'est pas celui du site visité
    - comparaison de l'URL et de l’identifié du certificat
  - si le certificat est signé par une autorité non reconnue par le navigateur
- solution :
  - inclure cette autorité de certification dans les navigateurs des postes de l'entreprise
- impact :
  - l'entreprise peut lire tout le trafic chiffré de ses utilisateurs
- solution clef en main proposée par les coupes feux récents style « palo alto networks ».

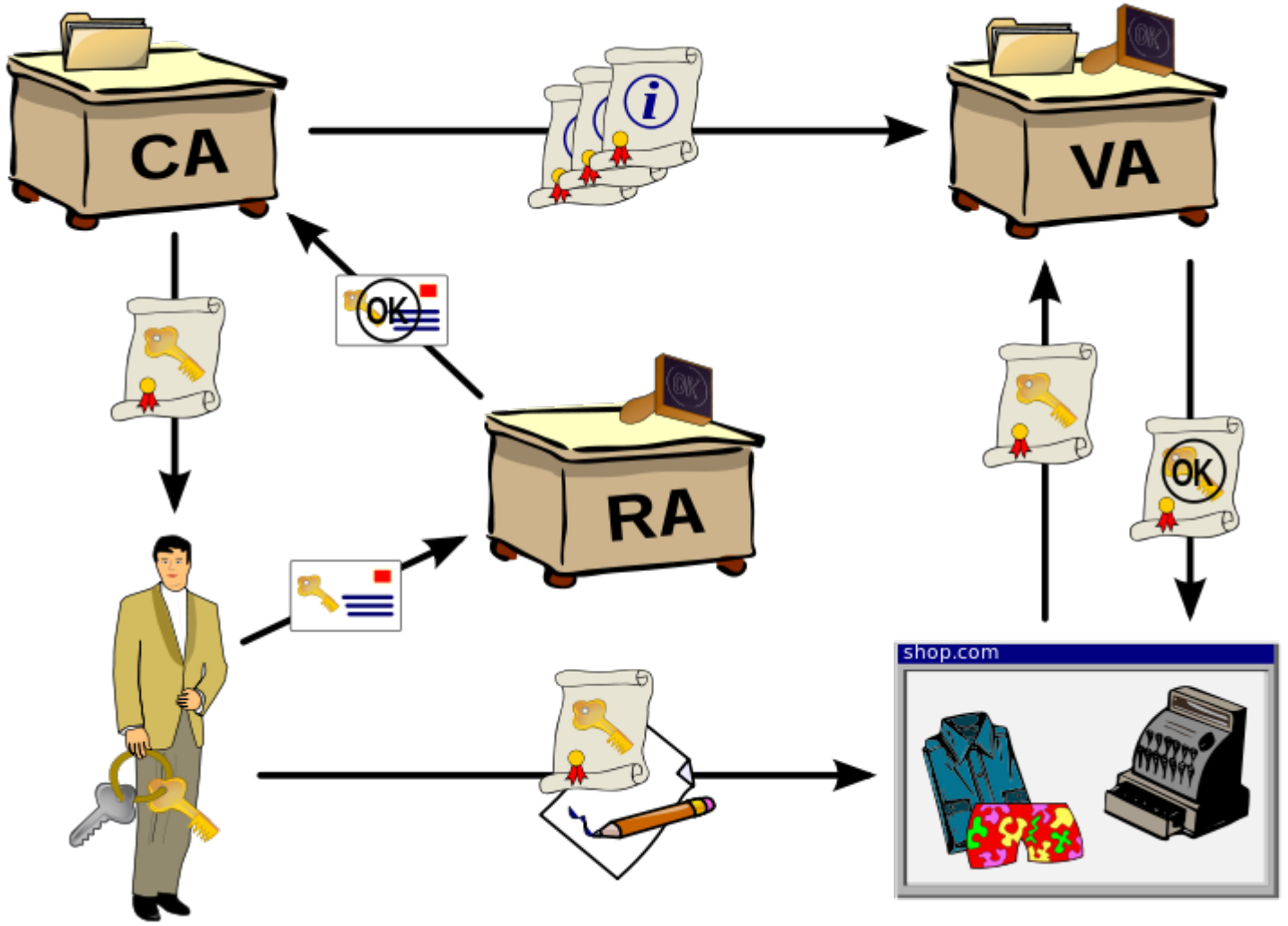
Man In the Middle

- plus simple : s'appuyer sur des certificats d'autorités reconnues
  - pas besoin d'inclure d'autorité de certification dans les navigateurs des postes de l'entreprise
  - espionnage des postes personnels, mobiles, ...
- décembre 2013 :
  - google a détecté qu'une autorité de certification dépendant de l'état Français a créé un certificat valide pour les sites google
  - cette autorité *était* intégrée à tous les navigateurs
  - cf http://reflets.info/mitm-de-google-par-l-anssi-la-theorie-du-doigt-qui-pointait-la-lune/

Man In the Middle

- dans le monde, 1800 entités capables d'émettre des certificats pour n'importe quel domaine et reconnues par les navigateurs
- cf IMC 2013, « Analysis of the HTTPS Certificate Ecosystem », Z. Durumeric, J. Kasten, M. Bailey, J.A. Halderman (University of Michigan) , http://conferences.sigcomm.org/imc/2013/papers/imc257-durumericAemb.pdf

infrastructure de gestion de clefs



certificat : procédure

- générer un couple de clefs :
  - une clef privée
  - une clef publique
- générer une CSR (Certificat Signing Request)
- la transmettre à l'Autorité d'Enregistrement (RA) qui la signera
- NE JAMAIS LAISSER la RA générer pour vous le couple de clefs et vous les retourner
- certificats autosignés : ràf

certificats X509

- différents types de certificats
  - démarche : DV/OV/EV
  - classe 1 :
  - classe 2 :
  - EV (extended validation certificat) :
  - extended validation certificat https://en.wikipedia.org/wiki/Extended_Validation_Certificate
  - voir @vincib
  - cerficat EV (cadenas, couleur et nom de la société mais rien de plus qu'un certificat standard)
  - https://en.wikipedia.org/wiki/Certificate_policy
  - https://en.wikipedia.org/wiki/Certification_Practice_Statement
  - verisign CPS : https://www.verisign.com/repository/CPS/
  - CACERT CP et CPS : http://www.cacert.org/policy/CertificationPracticeStatement.php
  - https://www.globalsign.com/ssl-information-center/types-of-ssl-certificate.html

révocation : ce qui ne marche pas

- En cas de compromission de la clef privée, un certificat doit être révoqué
- les CA doit maintenir et mettre à disposition des CRL (listes de révocation de certificat) pour que les navigateurs en tiennent compte
- Mises en œuvres possibles :
  - le navigateur récupére régulièrement les CRL de toutes les CA : lourd, non fait.
  - le navigateur récupère la CRL ad hoc au moment où il valide le certificat :
    - lent
    - pas toujours possible
  - OSCP : récupérer l'information sur le certificat que l'on souhaite valider :
    - plus léger que récupérer toute la CRL
    - reste lent (median à 300ms, moyen à 1s)
    - les CA n'ont pas mis l'infrastructure en place pour gérer un grand nombre de requêtes => DOS possible
    - fournit des informations sur les sites consultés à la CA
    - pas toujours possible (ex. : portail captif)

Révocation : exemple pratique

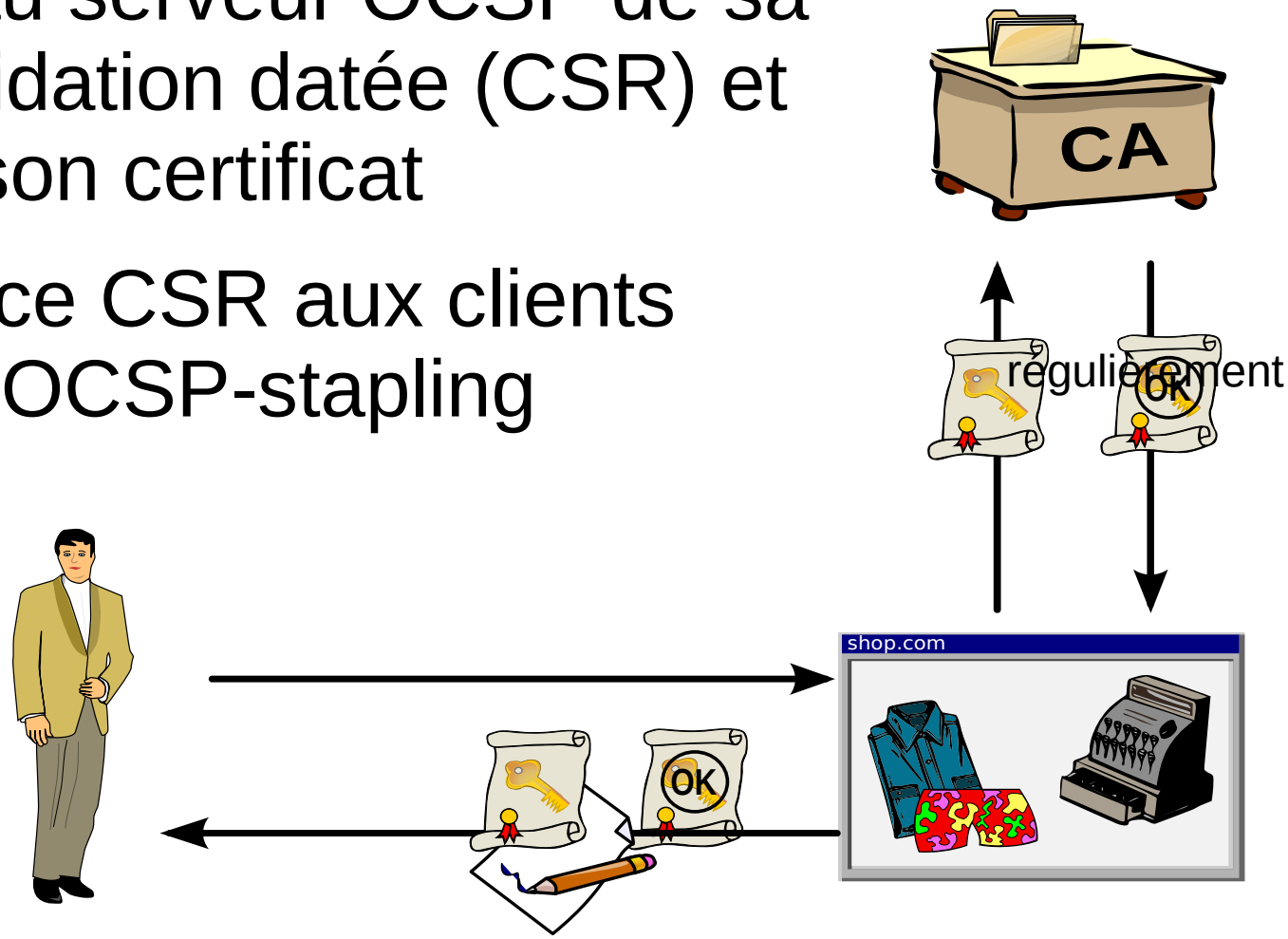
- tiré de « How certificate revocation (doesn’t) work in practice »
  http://news.netcraft.com/archives/2013/05/13/how-certificate-revocation-doesnt-work-in-practice.html
- le 30/04/2013, un cerficat a été révoqué
- l'un des sites WeB concerné (e-commerce, gros trafic) a continué à utiliser l'ancien certificat
- comportement des navigateurs :
  - pas d'alerte pour : chrome, firefox, safari mobile
  - alerte pour ie, opéra
  - CRL valide 6 mois donc pas d'alerte pour ie et opéra si consultation du site il y a moins de 6 mois

révocation : ce qui marche

- mettre à jour la CRL lors de la mise à jour du navigateur : fait
- Chrome CRLSETS : mise à jour des CRL sans mise à jour de Chrome (pas de redémarrage)
- Firefox travaille sur un mécanisme équivalent
- OCSP Stapling :
  - la validité récente du certificat est fournie par le site contacté lui-même

OCSP Stapling

- le possesseur du certificat demande au serveur OCSP de sa CA une validation datée (CSR) et signée de son certificat
- il transmet ce CSR aux clients supportant OCSP-stapling



OCSP Stapling

- avantages :
  - les clients ne contactent plus individuellement la CA
    - charge (moins de requête pour la CA)
    - vie privée (la CA n'est plus contactée par les clients)
  - l'information est obtenue même sans accès réseau à la CA (portail captif par ex.)
  - bonnes performances
- mécanisme récent
  - apache 2.3.3+, nginx 1.3.7+, IIS7.5+
  - firefox 26 (12/2013), chrome12+windows, ie7.5+

révocation bibliographie

- CRL
- OCSP
  - pourquoi OCSP est une mauvaise idée : http://security.stackexchange.com/questions/4052/why-isnt-ocsp-required-by-default-in-browsers
  - https://fr.wikipedia.org/wiki/Online_Certificate_Status_Protocol
- OCSP Stapling
  - https://en.wikipedia.org/wiki/OCSP_stapling
  - firefox : https://blog.mozilla.org/security/2013/07/29/ocsp-stapling-in-firefox/
  - firefox : https://wiki.mozilla.org/CA:ImprovingRevocation#OCSP_Stapling
  -
- sur chrome :
  - http://www.scribd.com/doc/163768316/Chrome-Group-Policy

révocation des certificats

- les URL de récup des CRL sont dans les certificats
  - openssl s_client -connect www.cesnet.cz:443 | openssl x509 -noout -text |grep crl
  - URI:http://crl.globalsign.net/educational.crl
- un article intéressant « How certificate revocation (doesn’t) work in practice » http://news.netcraft.com/archives/2013/05/13/how-certificate-revocation-doesnt-work-in-practice.html
- Revocation checking and Chrome's CRL (05 Feb 2012) https://www.imperialviolet.org/2012/02/05/crlsets.html
-

TLS en pratique

- les fichiers classiques
- les outils
- la configuration des navigateurs & Co pour durcir

TLS fichiers classiques

- pem(Privacy Enhanced Message)
- key : clef privée protégée ou non par un mot de passe
- csr (Certificate Signing Request) : clef publique prête à être signée. C'est un csr que l'on transmet à l'autorité d'enregistrement pour obtenir un certificat (clef publique signée)

TLS openssl, s_client

- debug de serveur à la « telnet » :
  - openssl peut monter la connnexion ssl et permettre ensuite de taper des commandes
  - 2 modes :
    - ssl direct
    - starttls
  - à savoir :
    - un R (r majuscule) en début de ligne provoque une renégociation (et le R est ignoré)
    - un Q (q majuscule) en début de ligne provoque la fin de la connexion

TLS openssl, s_client, starttls, smtp

openssl s_client -host ns-quad.ibisc.univ-evry.fr -port 25 -CAfile /etc/ssl/certs/ca-certificates.crt -starttls smtp

[tout un tas d'information SSL]

250 DSN

ehlo toto.shayol.org

250-newns.evry2.ibisc.univ-evry.fr

...

là, on est dans la session smtp et on peut la continuer à la main (RCPT TO :, MAIL FROM :, DATA, ...)

TLS openssl s_client avec d'autres outils : gnutls-cli

- gnutls : gnutls-cli www.ibisc.univ-evry.fr -p 443
  - pour starttls, c'est moins automatique que s_client :
    - on utilise l'option -s ; on lance à la main le starttls de l'hôte distant
    - on fait Ctrl-D pour indiquer à gnutls-cli qu'il doit reprendre le contrôle et lancer tls

```
gnutls-cli mx1.toto.net -p 25 -s
Resolving 'mx1.toto.net'...
Connecting to '1.2.3.4:25'...
- Simple Client Mode:
220 mx1.toto.net ESMTP Postfix
ehlo smtp.shayol.org
250-mx1.galacsys.net
...
250-STARTTLS
starttls
220 2.0.0 Ready to start TLS
*** Starting TLS handshake
...
```

TLS openssl s_client avec d'autres outils : ncat

- ncat --ssl www.ibisc.univ-evry.fr 443

TLS : openssl

- création couple+csr : openssl req -new -config openssl.conf
- autosignature : openssl x509 -req -days 365 -in test.csr privkey.pem server.key -out test.crt
- affichage : openssl x509 -in test.crt -text
- déprotection de la clef
-

TLS et navigateurs

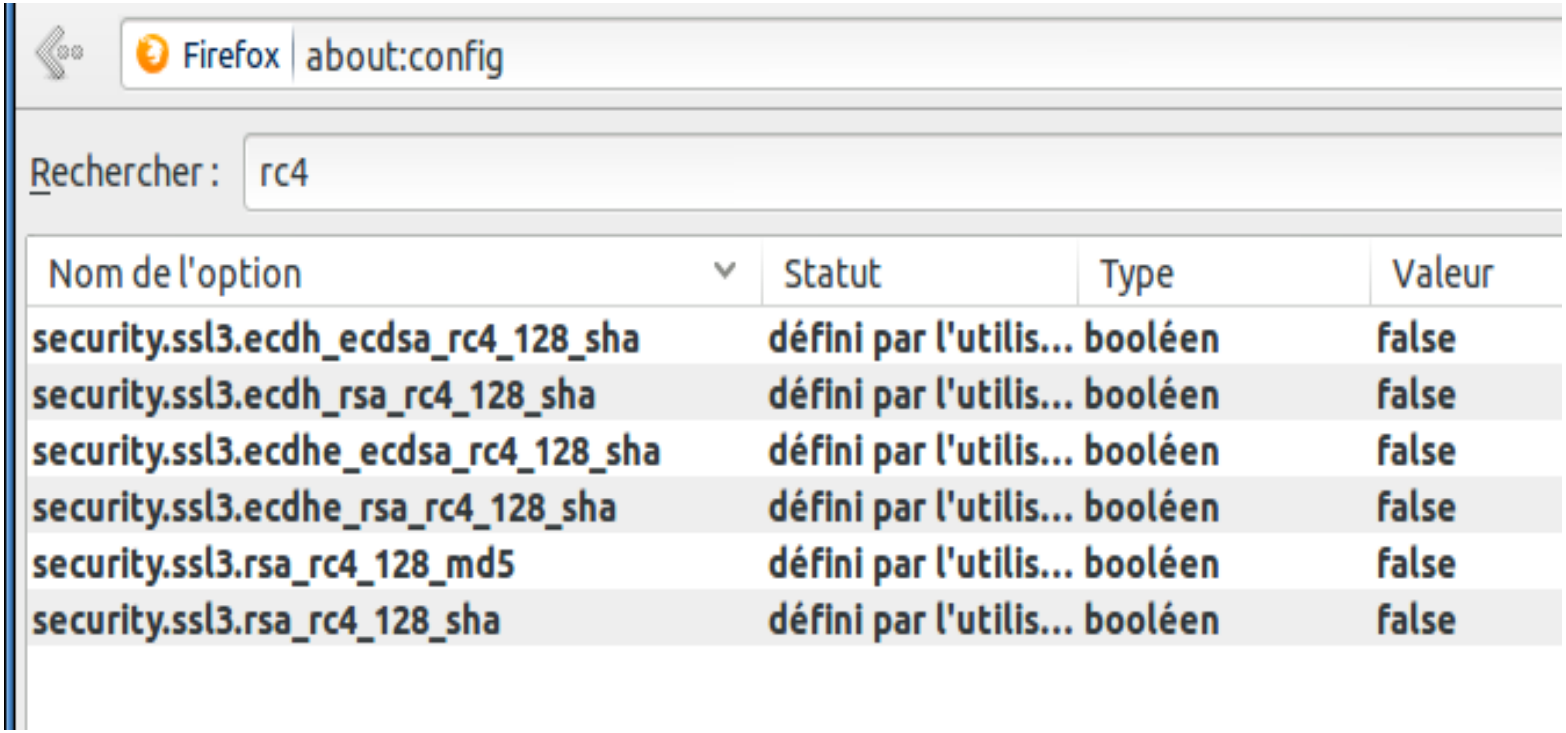
- httpseverywhere : greffon permettant de forcer https (https://www.eff.org/https-everywhere)
- Certificate Patrol : être averti des changements de certificats, d'autorités

TLS et firefox : version de ssl

- via about:config
- version de tls supportée
  - security.tls.version.min
  - security.tls.version.max
  - 0=ssl3 ; 1=tls1.0;2=tls1.1 (firefox 19+), 3=tls1.2 (firefox 24+), …
- conseil :
  - security.tls.version.max=3
  - security.tls.version.min=0 (certains sites ne supportent que ssl v3)
- source : http://kb.mozillazine.org/Security.tls.version.*

TLS et firefox : algorithmes choisis

- clefs security.ssl3.* à activer/désactiver un par un
- Exemple : désactiver rc4



TLS et firefox : algorithmes choisis

- contraintes opposées
  - avoir un niveau minimal de chiffrement
  - rester compatible avec certains sites ne supportant pas certains chiffrements
- solution possible : 2 navigateurs
  - un réglé strictement
  - l'autre plus laxiste
- test : https://cc.dcsec.uni-hannover.de/
  - liste des algorithmes supportés
  - version de tls

TLS et firefox : proposition

- voir les soucis de timing attack et le choix de la taille de la clef (128 ou 256 bits)
- voir ràf
- voir aussi le problème de backdoor sur les courbes elliptiques

TLS et firefox : résultat

SSL Cipher Suite Details of Your Browser

This websites gives you information on the SSL cipher suites your browser supports for securing HTTPS connections.

Cipher Suites Supported by Your Browser (ordered by preference):

Spec	Cipher Suite Name	Key Size	Description
(c0,0a)	ECDHE-ECDSA-AES256-SHA	256 Bit	Key exchange: ECDH, encryption: AES, MAC: SHA1.
(c0,14)	ECDHE-RSA-AES256-SHA	256 Bit	Key exchange: ECDH, encryption: AES, MAC: SHA1.
(00,88)	DHE-RSA-CAMELLIA256-SHA	256 Bit	Key exchange: DH, encryption: Camellia, MAC: SHA1.
(00,87)	DHE-DSS-CAMELLIA256-SHA	256 Bit	Key exchange: DH, encryption: Camellia, MAC: SHA1.
(00,39)	DHE-RSA-AES256-SHA	256 Bit	Key exchange: DH, encryption: AES, MAC: SHA1.
(00,38)	DHE-DSS-AES256-SHA	256 Bit	Key exchange: DH, encryption: AES, MAC: SHA1.
(c0,0f)	ECDH-RSA-AES256-SHA	256 Bit	Key exchange: ECDH, encryption: AES, MAC: SHA1.
(c0,05)	ECDH-ECDSA-AES256-SHA	256 Bit	Key exchange: ECDH, encryption: AES, MAC: SHA1.
(00,84)	RSA-CAMELLIA256-SHA	256 Bit	Key exchange: RSA, encryption: Camellia, MAC: SHA1.
(00,35)	RSA-AES256-SHA	256 Bit	Key exchange: RSA, encryption: AES, MAC: SHA1.

Further information:

User-Agent:	Mozilla/5.0 (Windows NT 6.1; rv:6.0) Gecko/20100101 Firefox/6.0
Preferred SSL/TLS version:	TLSv1
SNI information:	cc.dcsec.uni-hannover.de
SSL stack current time:	Wed, 25 Dec 2013 23:58:20

This connection uses TLSv1.2 with DHE-RSA-CAMELLIA256-SHA and a 256 Bit key for encryption.

TLS en pratique du coté serveur

- un test en ligne : https://www.ssllabs.com/ssltest/index.html
- mozilla : Security/Server Side TLS https://wiki.mozilla.org/Security/Server_Side_TLS
- Mozilla : « Navigating the TLS landscape » https://blog.mozilla.org/security/2013/11/12/navigating-tls/
- Qualys SSL labs « SSL/TLS Deployment Best Practices » https://www.ssllabs.com/downloads/SSL_TLS_Deployment_Best_Practices_1.3.pdf

TLS et la NSA : PFS

- de base, TLS utilise le certificat pour :
  - authentifier le serveur
  - chiffrer un secret partagé, clef du chiffrement symétrique
- conséquence
  - posséder la clef privée du serveur => déchiffrer le trafic
    - obtenir les éléments permettant de le faire
    - en temps réel ou via un enregistrement de tout le trafic
  - la NSA (& Co) peut obtenir cette clef privée légalement
- source :
  - [VB2011] « SSL/TLS & Perfect Forward Secrecy » de V. Bernat, http://vincent.bernat.im/fr/blog/2011-ssl-perfect-forward-secrecy.html

TLS et la NSA : PFS

- handshake tls avec AES128-SHA ou ~
- phase1 : client transmet aléa
- phase 2 : le serveur transmet sa clef publique au client (+aléa)
- phase 3 : un secret est transmis au serveur, chiffré avec sa clef publique
- secret maître :
  - valeurs aléatoires des phases 1 et 2
  - secret phase 3

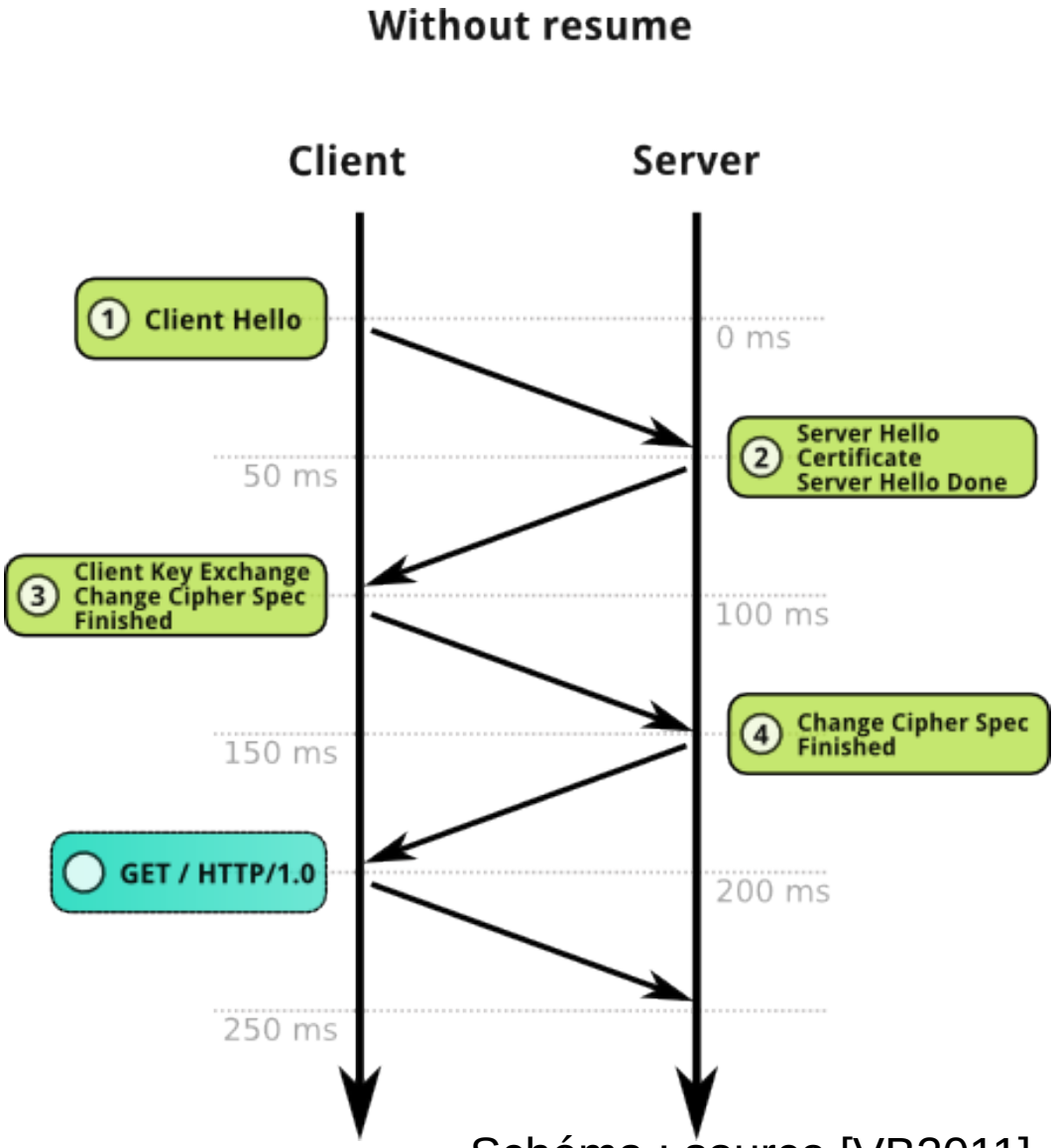


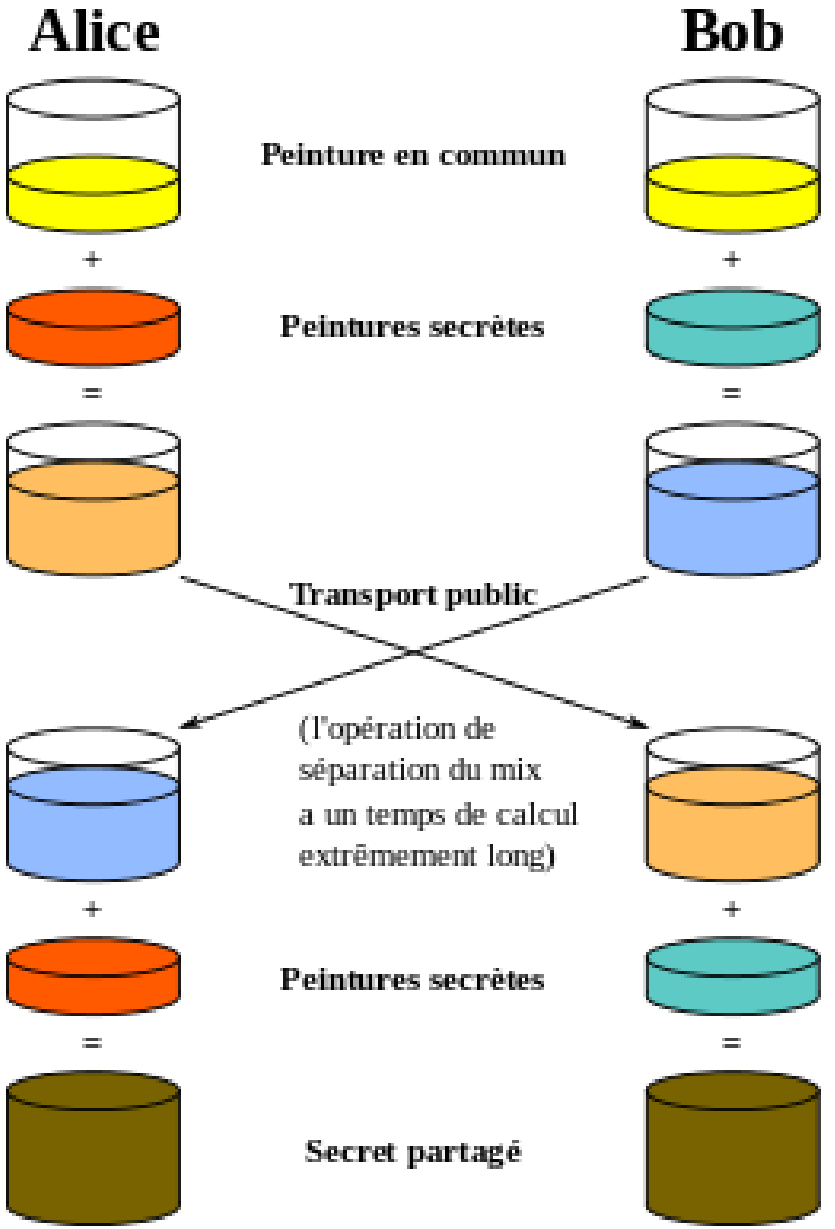
Schéma : source [VB2011]

Solution : Diffie Hellman

- se mettre d'accord sur une donnée secrète commune
- en échangeant des messages public
- l'attaquant peut voir les messages publics mais ne peut en déduire le secret commun

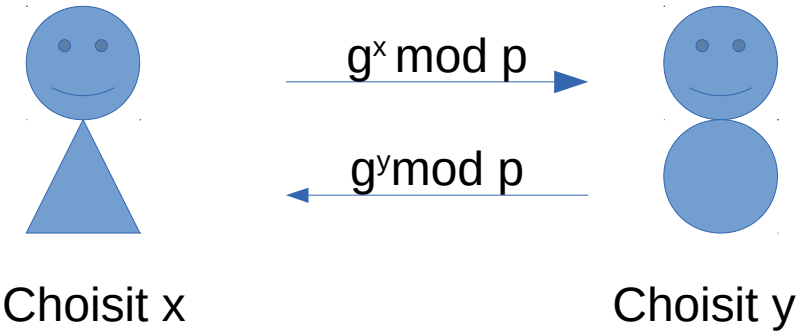
Diffie Hellman

- principe :
  - Alice a une donnée secrète (couleur orange)
  - Bob a une donnée secrète (couleur bleue)
  - transmise mélangée à une donnée publique (couleur jaune)
  - le secret commun est :
    - Alice : orange+(jaune+bleu)
    - Bob : bleu + (jaune+orange)
  - principe : impossible de séparer des couleurs mélangées



source : wikimedia

Diffie Hellman



- Valeurs publiques:
  - p un grand nombre premier
  - G un groupe multiplicatif de cardinal p-1
  - g un générateur de G
- Secret d'Alice: x, secret de Bob: y
- Secret commun: $K=g^{xy}=(g^x)^y=(g^y)^x$
- Attaque: trouver x connaissant g^x
  - Problème du logarithme discret

Diffie Hellman et l'homme du milieu (MiM)

- Algorithme attaquable par MiM
  - Oscar l'attaquant intercepte le g^x d'Alice et transmet à Bob un $g^{x'}$ avec x' choisi par lui.
  - idem avec Bob
  - Oscar a ainsi un secret partagé avec Bob et un secret partagé avec Alice
  - Alice et Bob croient dialoguer ensemble alors qu'ils dialoguent avec Oscar
- Solution :
  - signer les messages échangés
  - ex. DH-DSA : Diffie Hellman et DSA (Digital

TLS et NSA : Solution PFS

- Diffie Helmann :
  - connaître la clef privée ne donne pas la clef secrète de l'algorithme symétrique
  - il faut une attaque coûteuse en temps processeur pour l'avoir (est-ce faisable?)
- DHE : Diffie Helmann Ephemeral
  - x et y sont différents pour chaque connexion
  - il faut attaquer chaque connexion (arg ! pan dans l'oeil des espions !)
  - faire des calculs de puissance sur de grands nombres est coûteux en temps CPU (division par 5 du nombre de connexion max cf http://vincent.bernat.im/fr/blog/2011-ssl-perfect-forward-secrecy.html)

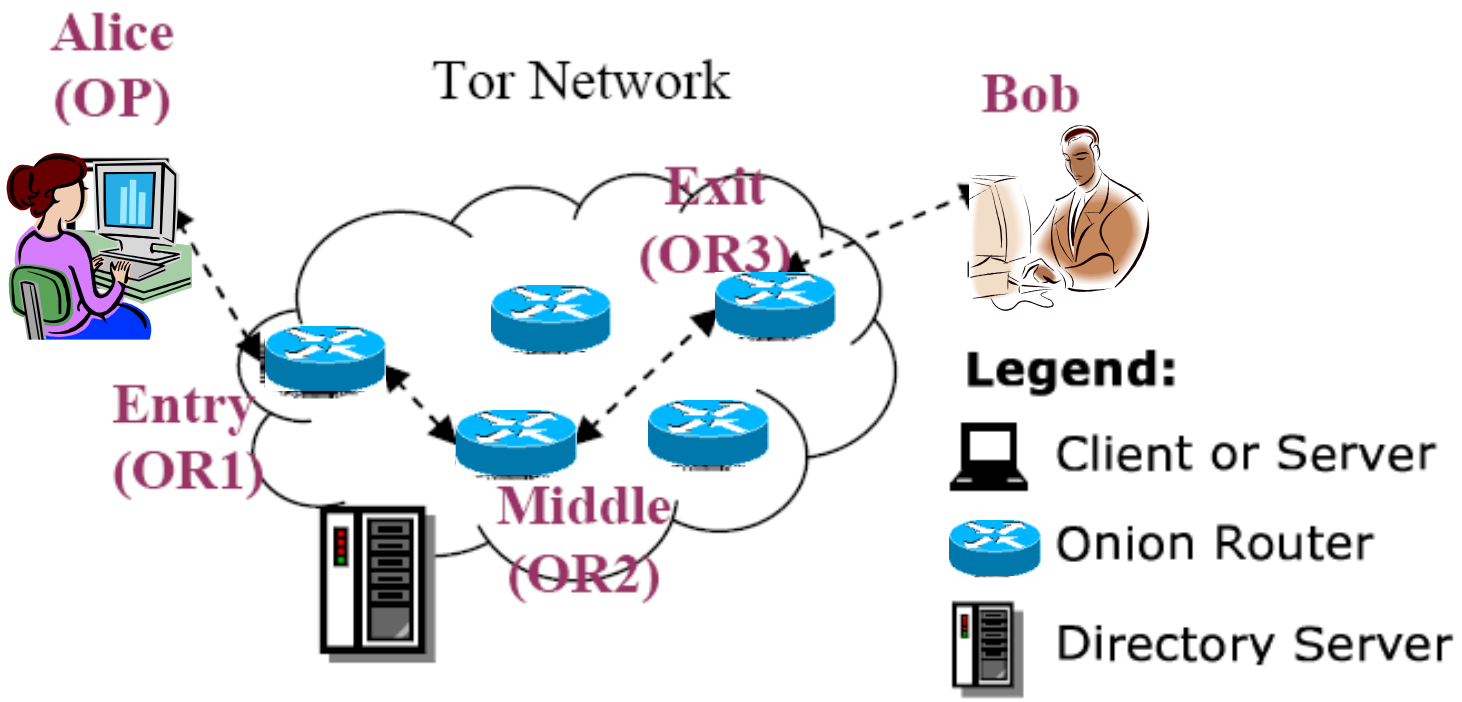
TLS et NSA : Solution PFS

- DHE : Diffie Helmann Ephemeral
  - coûteux (CPU, temps)
- ECDHE :
  - Diffie Helmann adapté aux courbes elliptiques
  - RFC 4492, RFC 7027
  - courbes elliptiques :
    - à résistance égale :
      - clef plus petites
      - peu gourmand en cpu
    - certaines auraient été volontairement affaiblies par la NSA

TOR : The Oignon Router

- Objectifs :
  - transmettre de manière anonyme des flux TCP
    - chiffre les requêtes entre vous et TOR ;
    - camoufler votre IP au service destination ;
    - seul le routeur de sortie peut avoir accès à votre requête mais ne sait d'où elle provient ;
    - seul le routeur d'entrée peut savoir quelle est la machine source de la requête.
- Moyens :
  - chiffrement en couche
  - un ensemble de routeurs camouflant l'ip source des connexion TCP qu'ils transmettent : proxy

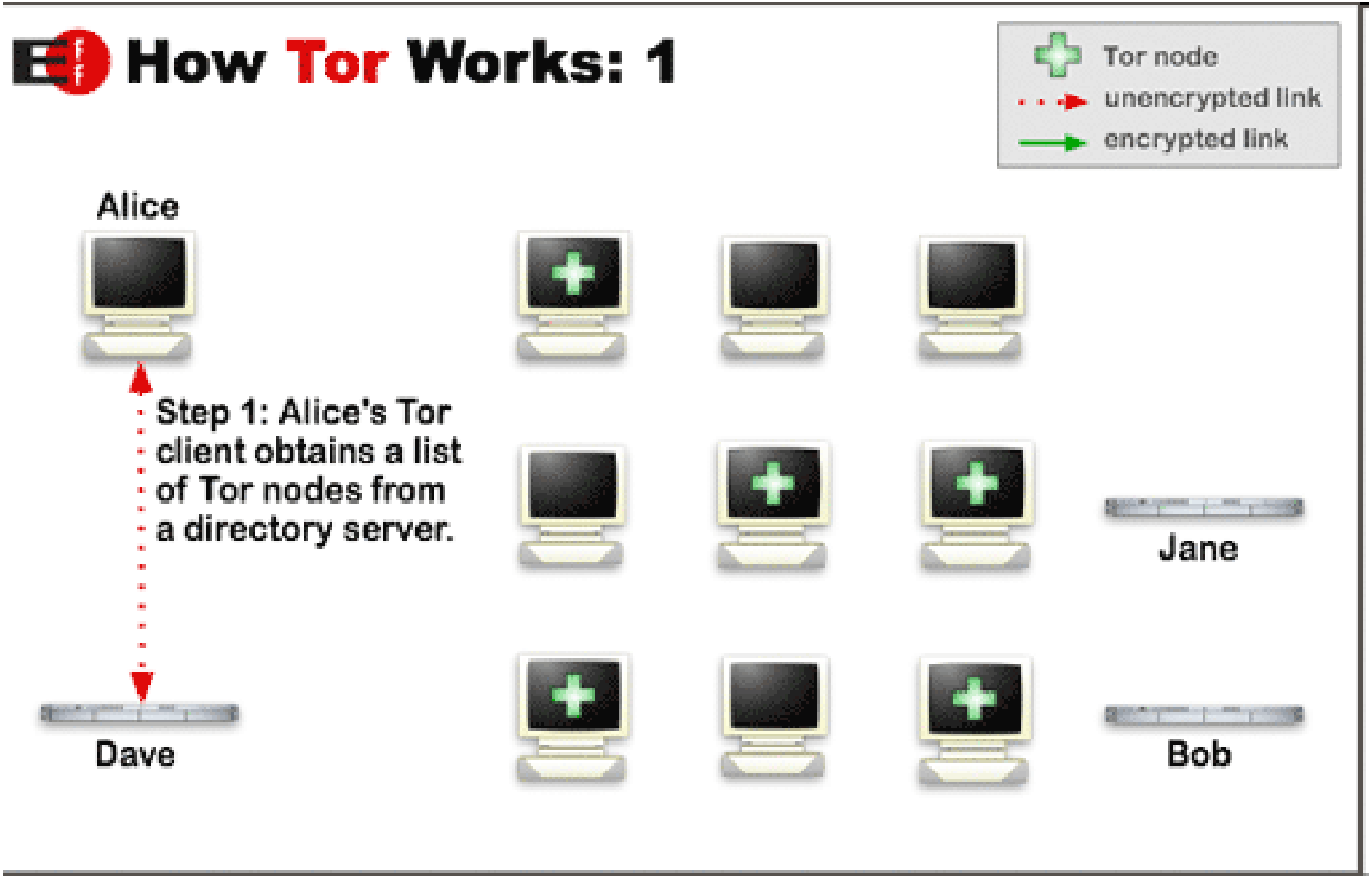
Composants de Tor



- **Client**: L'utilisateur du réseau TOR, envoie une requête au serveur Bob
- **Server**: une application TCP cible sur le serveur Bob
- **Tor (onion) router**: un proxy spécial qui relaie la requête
- **Directory server**: serveur fournissant des informations sur les routeurs TOR

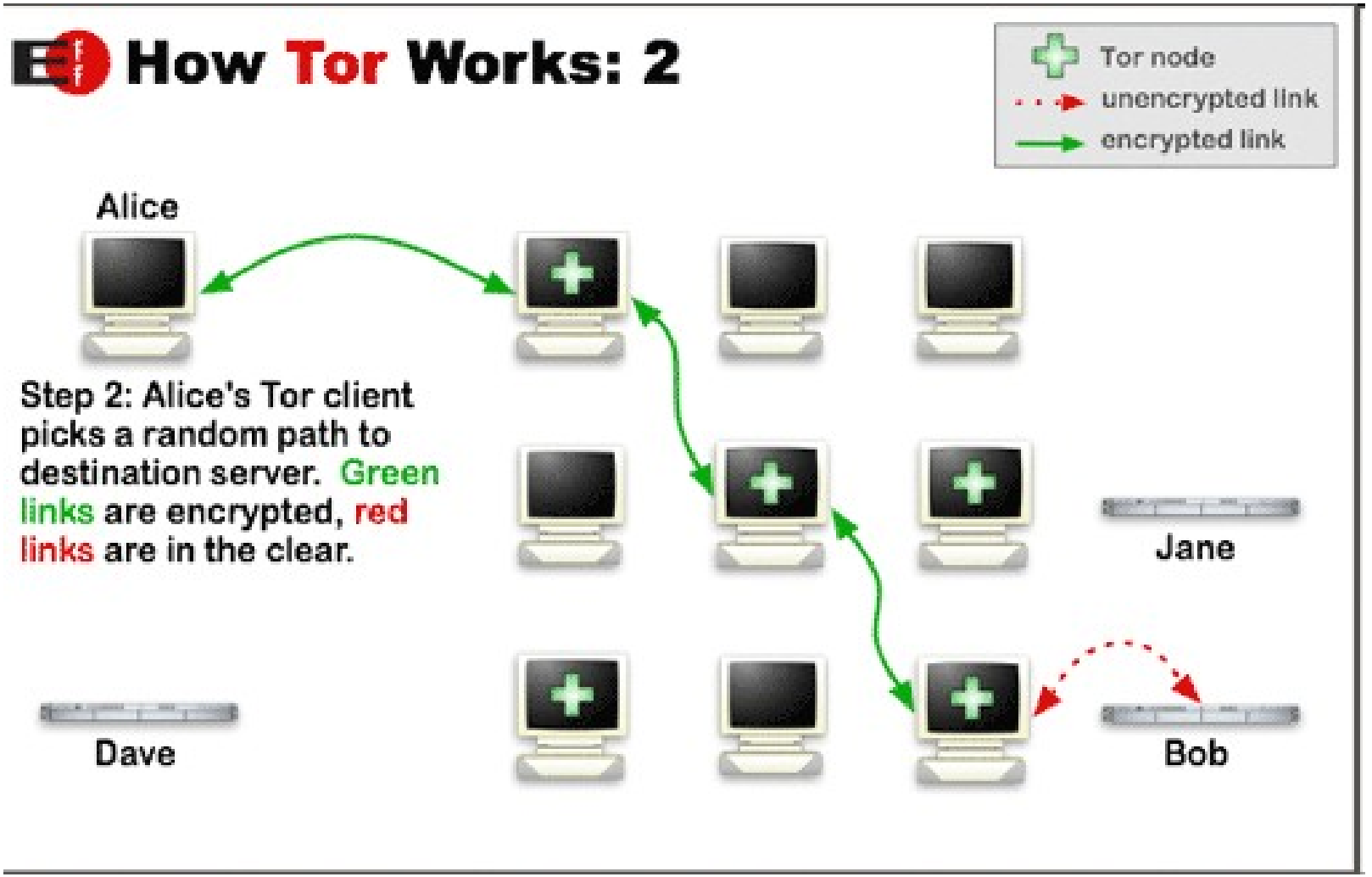
source:
http://www.cse.unr.edu/~mgunes/cpe401/cpe401sp11/student/tor.ppt

How does Tor work?



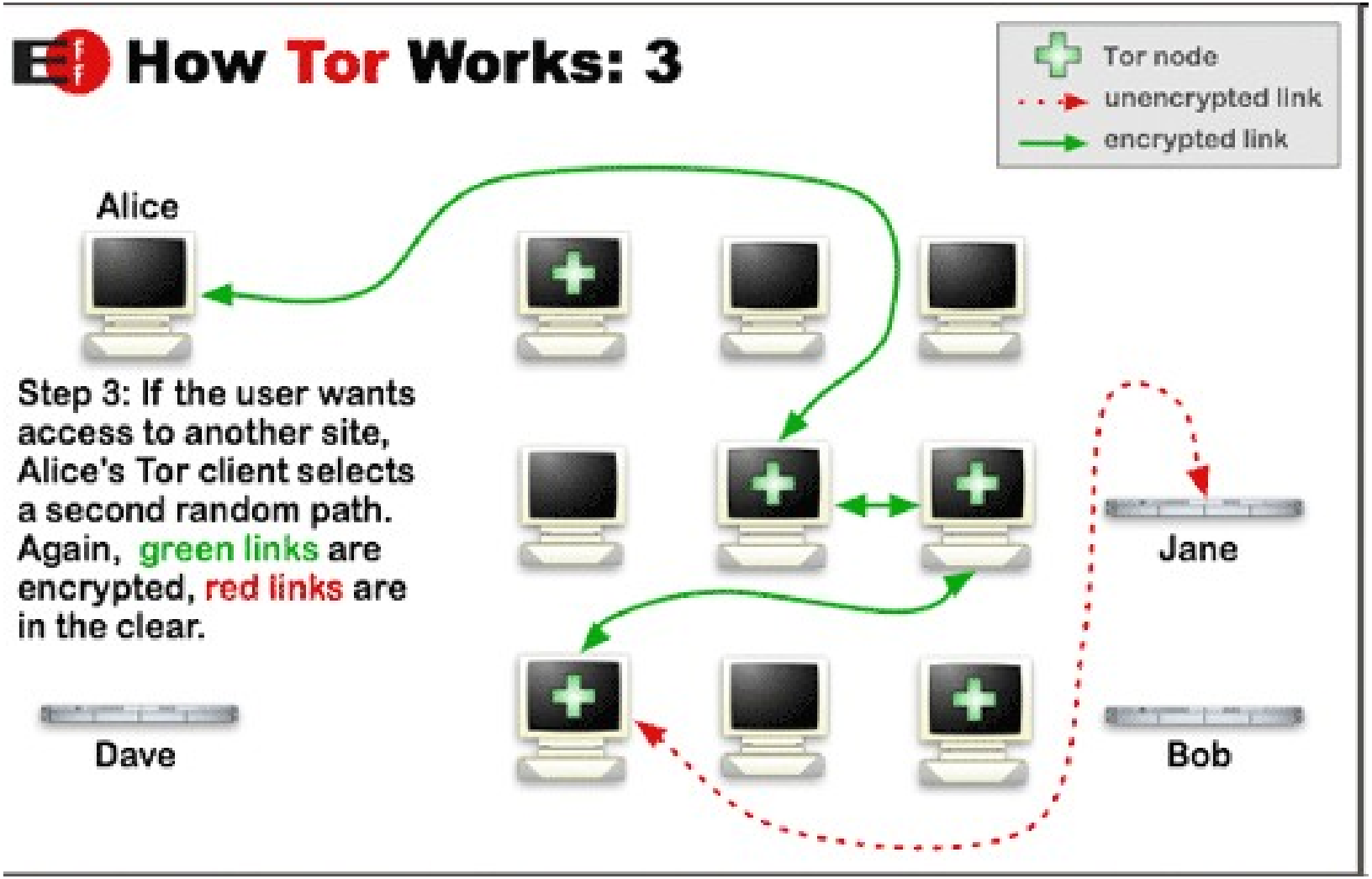
source:
http://www.cse.unr.edu/~mgunes/cpe401/cpe401sp11/student/tor.ppt

How does Tor work?



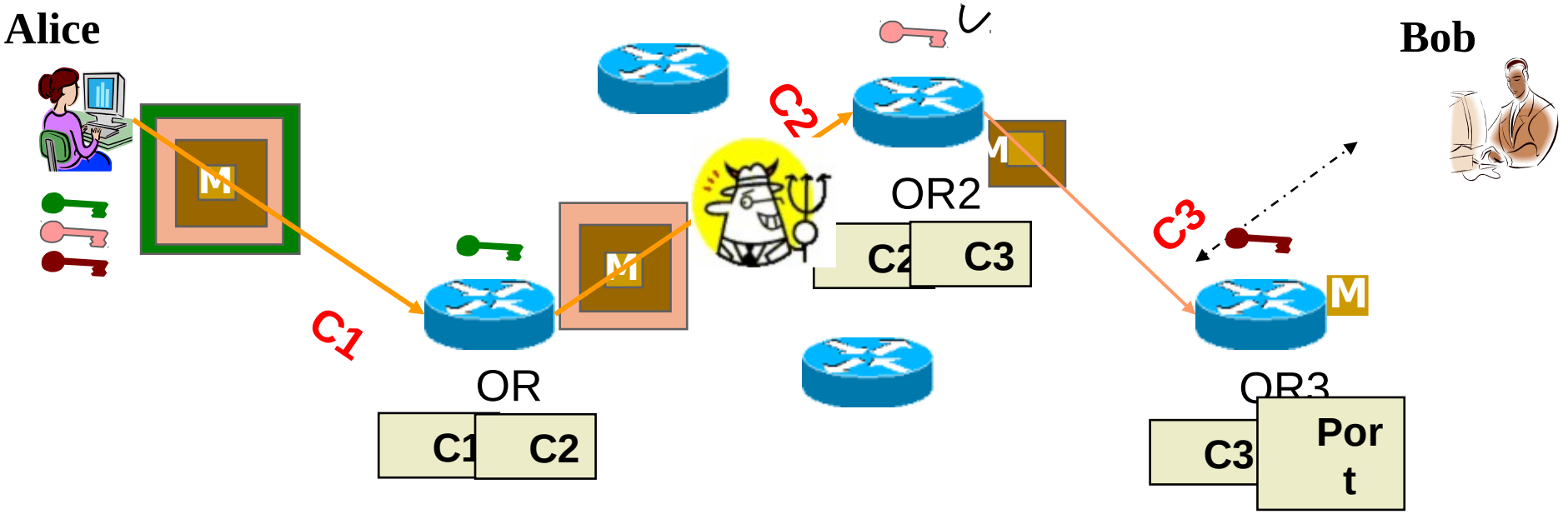
source:
http://www.cse.unr.edu/~mgunes/cpe401/cpe401sp11/student/tor.ppt

How does Tor work?



source:
http://www.cse.unr.edu/~mgunes/cpe401/cpe401sp11/student/tor.ppt

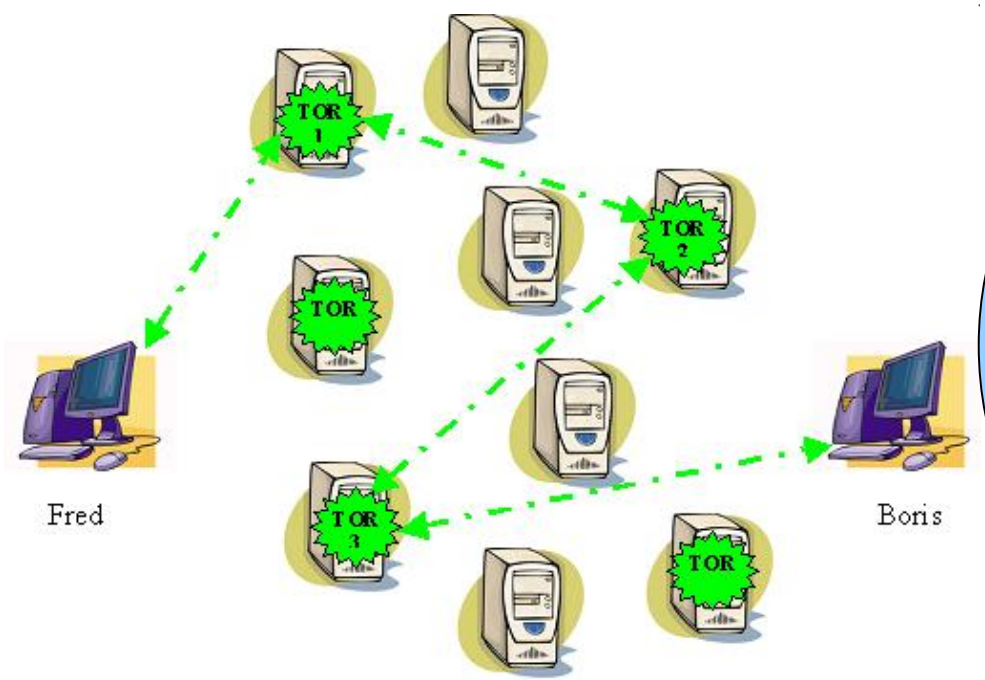
How Tor Works? --- Onion Routing

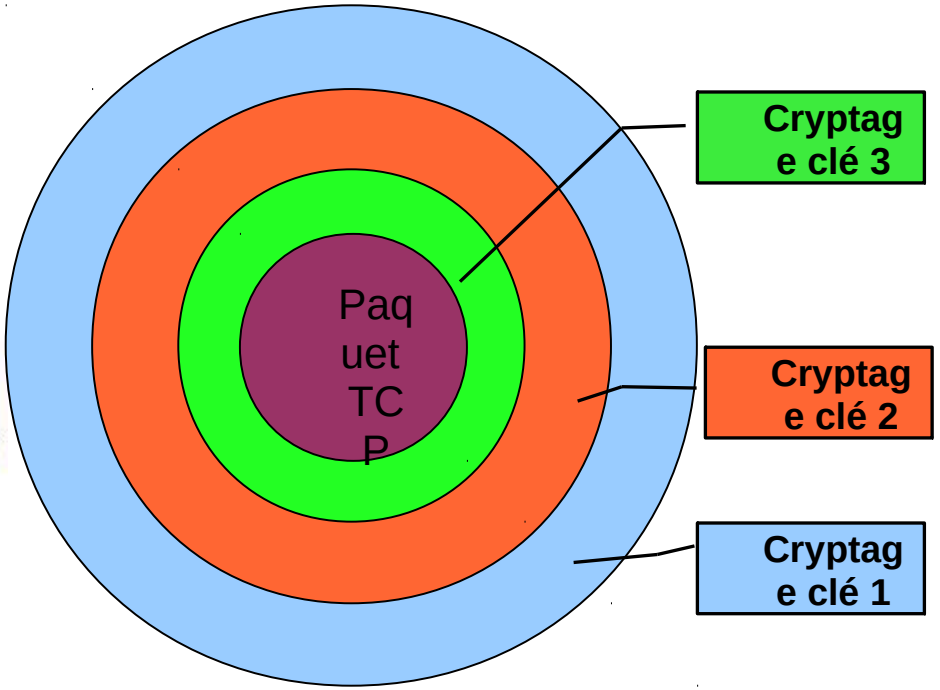


- un circuit est créé pas à pas
- chiffrement en couches d'oignons
  - Alice négocie une clef avec chaque routeur via diffie-hellman
  - chaque routeur ne connaît que son prédécesseur et son successeur
  - quand un routeur déchiffre le message, il y trouve :
    - un paquet chiffré et l'adresse du routeur suivant
  - seul le routeur de sortie peut voir le message mais il ne sait pas d'où il vient

source:
http://www.cse.unr.edu/~mgunes/cpe401/cpe401sp11/student/tor.ppt

TOR : the oignon routers





source des images :
http://wapiti.enic.fr/commun/ens/peda/options/ST/RIO/pub/exposes/exposesrio2007/Bouillin-Morvan/internet_anonymat.ppt

Tor : sécurité

- votre ip source est cachée à la destination
- seul le routeur de sortie voit vos données
- la destination et la requête sont cachées entre vous et le premier routeur TOR
- ne protège pas si l'attaquant :
  - peut écouter le réseau en tout point
- MAIS
  - outils (navigateur) ou protocoles (ftp) envoient dans les requêtes beaucoup d'informations à votre insu (cf http://www.cnil.fr/vos-libertes/vos-traces)
  - votre dns local peut connaître les sites consultés
- Solution :
  - privoxy : un proxy qui nettoie les requêtes WeB
  - Mon conseil : une clef bootable dédié : tails (cf tails.boum.org)

ressources :

- tails, pour faire une clef bootable utilisant tor
  - anonymat garanti
  - protège de tous les malwares du pc hôtes
  - protèges donc des keyloggers & Co
  - un clavier virtuel pour se protéger des keyloggers matériels
  - https://tails.boum.org/
- version portables (utilisables sans installation) de logiciels courants :
  - chez framasoft : http://framakey.org/Portables/Index
  - firefox portable : http://framakey.org/Portables/FramafoxPortable
  - attention : ces solutions ne protègent pas des keyloggers et ~