Gestion des clés, protocoles sécurisés

Contenu

Echange de clés

Tous les chiffres nécessitent l'échange sécurisé de clés.
Evident pour les clés secrètes ; pour les clés publiques, il faut contrer une attaque de "l'homme du milieu". Comment faire ?

Quelques solutions

1. En se fixant un rendez-vous
2. En envoyant la clé par courrier spécial
3. En utilisant une clé commune aux deux parties pour chiffrer une clé supplémentaire.

Deux premiers cas : contact physique nécessaire.
Pas toujours possible : dans un conflit quand des unités d'une armée sont séparées physiquement par l'ennemi.

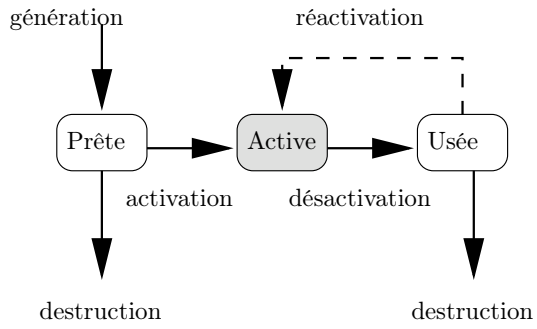
Techniques de gestion de clés [1]

Reposent sur :

- mécanismes de chiffrement
- utilisation des clés
- politique de sécurité

permettent d'éviter :

- la modification
- la destruction malintentionnée
- le rejeu
- la divulgation (disclosure)



Pendant toute la durée de vie des clés, il faut assurer :

génération sûre	suppression	révocation	certification
enregistrement	distribution	destruction	installation

Modèles pour l'établissement de clés

Procédé qui permet de rendre disponible une clé à une ou plusieurs entités.
Recouvre :

- mise en accord
- transport de clés (publiques, secrètes)
- mise à jour des clés
- dérivation des clés

Mise en accord

Imaginer une solution basée sur la complexité des problèmes.
Facile à calculer à ses utilisateurs, difficile à casser aux autres.

Il faut donc trouver une fonction à sens unique (sans trappe).

Un bon candidat est le log discret.

Mise en accord de Diffie Hellman

On choisit p premier et α, $1 < \alpha < p$ un générateur de $\mathbb{Z}_p^\star$.
Chaque utilisateur U

- choisit secrètement une valeur aléatoire X_U, $1 < X_U < p$ et la conserve secrète
- publie (ou transmet) $Y_U = \alpha^{X_U} \mod p$

A et B construisent une clé commune, connue d'eux seuls.

- A calcule $K = (Y_B)^{X_A} \mod p$
- B calcule $K = (Y_A)^{X_B} \mod p$

A et B ont alors une clé commune K :

$$\begin{aligned}(Y_B)^{X_A} &\equiv (\alpha^{X_B})^{X_A} \equiv \alpha^{X_B X_A} \equiv \\ &\equiv \alpha^{X_A X_B} \equiv (\alpha^{X_A})^{X_B} \equiv (Y_A)^{X_B} \mod p\end{aligned}$$

Sécurité

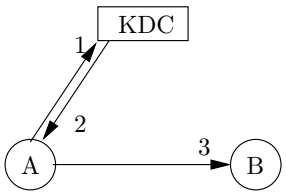
1. l'information privée est sûre : calculer X_A à partir de $Y_A = \alpha^{X_A} \mod p$, est le problème du log discret
2. Est-il possible de retrouver la clé partagée par A et B à partir de Y_A et Y_B sans calculer leur log discret ? C'est un problème aussi difficile que celui du log discret.
3. Quand la clé commune est changée à chaque session (en changeant les valeurs aléatoires), le protocole assure la propriété de **confidentialité persistante** (forward secrecy) qui garantit que si on a obtenu une des valeurs aléatoires de l'échange, on ne sera pas en mesure de déchiffrer des échanges passés enregistrés. Le seul qui peut l'être est celui qui utilise cette clé commune.

Transport de clés secrètes

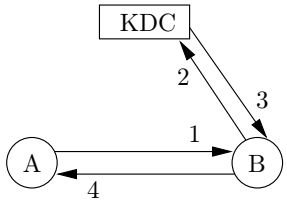
Procédé qui permet de transférer une clé secrète créée –ou obtenue s'il s'agit d'un tiers de confiance– par une entité à une autre entité.
En utilisant du chiffrement, asymétrique ou symétrique.
ISO/IEC 11770-2 et 3 définissent 18 mécanismes, dont 5 sont point à point, les autres utilisant des tiers de confiance comme centre de distribution de clés. En résumé : distribution

- au sein d'un domaine
- entre des domaines

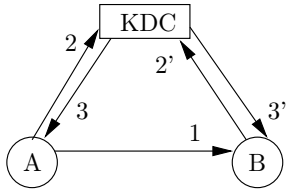
Modèles de distribution de clés



Modèle pull



Modèle push



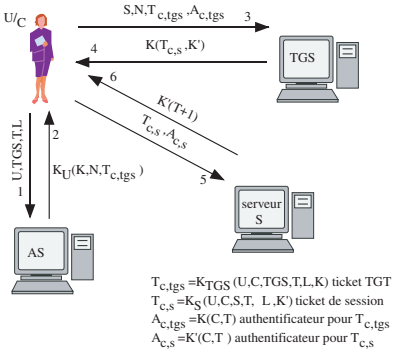
Modèle mixte

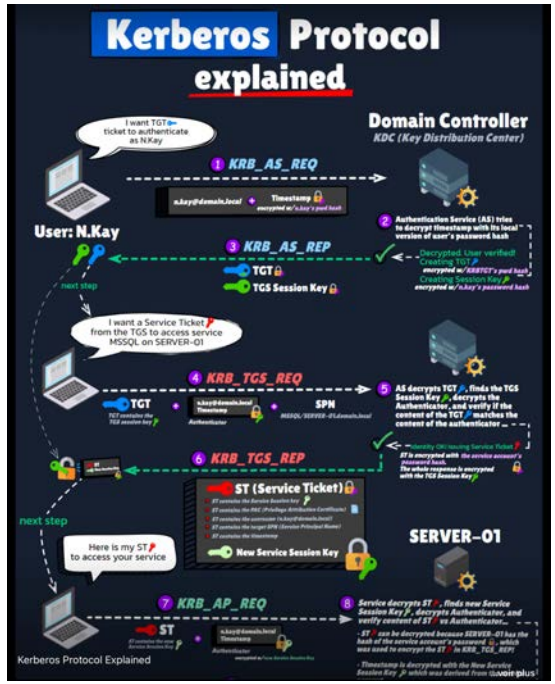
Exemple de Kerberos

Permettre à un client travaillant pour un utilisateur de faire la preuve de son identité à un service ou à un serveur d'application sans que ces données transitent par le réseau.
Requiert un tiers de confiance qui sert de **centre de distribution de clé** (KDC) pour le domaine ; il est composé de :

- serveur d'authentification (AS)
- distributeur de tickets (TGS)

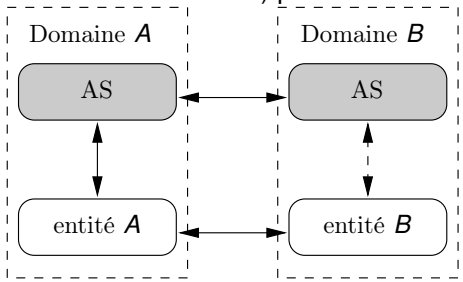
qui sont sécurisés





Etablissement de clés entre deux domaines

Un CDC (ou **autorité de sécurité**) par domaine.



Si A et B ont une confiance mutuelle ou si chaque entité a confiance en l'AS de l'autre domaine, tout se passe comme s'il n'y en avait qu'un.
Réalisations asymétriques ou symétriques.

Mise à jour des clés

Faire évoluer la clé session après session : **mise à jour des clés**.
Procédé qui permet de partager des clés construites au préalable en les faisant évoluer au moyen de paramètres obtenus pour la session
On définit une nouvelle clé de session K à partir :

- d'une clé partagée K_{AB}
- d'un paramètre F (aléa, estampille, numéro de séquence)
- d'une fonction de mise à jour de clé f

Fonctionnement en deux étapes :

1. l'initiateur A choisit paramètre de dérivation F transmis à B
2. A et B calculent la nouvelle clé K par f t.q.

$$K = f(K_{AB}, F)$$

Exemple de fonction f : fonction de hachage cryptographique h à la concaténation de données : $K = h(K_{AB}; F)$. Voir, `https://fr.wikipedia.org/wiki/PBKDF2`

Id & auth

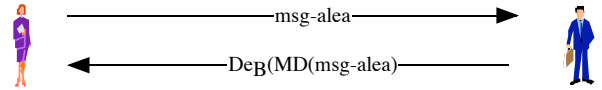
identification : affirmation de l'identité d'une entité au moyen de son identifiant. (Je suis Bruno)
authentification : procédé de vérification de l'identité d'une entité. (Je suis Bruno et en voici la preuve)

L'identification permet de connaitre l'identité d'une entité et l'authentification de vérifier cette identité prétendue.

Service d'authentification vérifie l'identité à différents niveaux :

- applicatif : `http, ftp`
- transport : `ssl, ssh`
- réseau : `ipsec`
- transmission : `pap, chap` (qui utilisent `md5`)

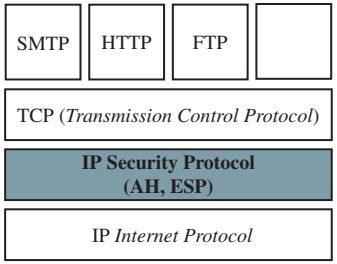
Exemple d'authentification « asymétrique »



- Sans calcul h(msg-aléa), KPA possible : (m/$\{m\}_{SK}$) connus, ...
- **Hyp :** Alice connaît la clé publique de Bob au préalable.

Sécurité IP

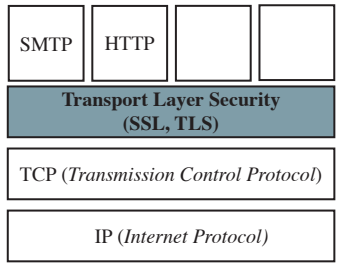
Ajout de mécanismes cryptographiques aux protocoles standards de l'infrastructure Internet (IPSec). L'architecture de sécurité au niveau IP fournit des mécanismes de sécurité (de la RFC1825) qui implémentent les services d'authentification, d'intégrité, de contrôle d'accès et confidentialité.



Sécurité de TCP

Protocoles pour sécuriser TCP :

- **Secure Socket Layer** utilisé par Netscape
- ~~**Private Communication Technology**~~ de Microsoft (arrêt avec SSL3)
- **Transport Layer Security** standard IETF



Et, plus récemment, `https://www.libressl.org` ou `https://boringssl.googlesource.com/boringssl/`

Protocole vs Implementation

Ne pas confondre le protocole avec son implémentation ! TLS (qui remplace maintenant SSL) est implémenté par de nombreuses libraires. Voir à ce sujet `https://en.wikipedia.org/wiki/Comparison_of_TLS_implementations`. Les plus classiques :

- OpenSSL
- LibreSSL
- BoringSSL
- GnuTLS

Et voir aussi les recommandations sur Crypto Best Practice

SSL & TLS

SSL fournit authentification, compression, intégrité, chiffrement. Supporte plusieurs mécanismes d'authentification et de chiffrement et protège tout protocole applicatif.
SSL est devenu (par l'IETF) un standard en tant que TLS qui repose sur SSL3.0. Deux couches le composent :

- **Rencontre** ou Handshake Protocol
- **Communication** ou Record Protocol

qui fournissent les services suivants :

- **confidentialité de la connexion** par AES, IDEA, RC2, RC4,... ainsi que du chiffrement en flot
- **intégrité de la connexion** par un MAC utilisant une clé en valeur initiale (~~MD5~~, SHA-1 ou ultérieur)

Identification–handshake

C'est le moyen pour Alice de vérifier l'identité de Bob.
pk_B la clé publique de Bob et sk_B sa clé privée.

$A \to B$	$r =$ un message aléatoire
$B \to A$	$c = \{r\}_{sk_B}$

Signer un message aléatoire r fourni par un tiers et le réexpédier peut s'avérer dangereux.
On pourrait utiliser une fonction de hachage h afin que Bob signe $h(r)$. Mais le danger persiste.

Identification–handshake

Mieux vaut que Bob signe un message de son cru

$A \to B$	*"Bonjour, est-ce Bob ?"*
$B \to A$	$m =$ *"Alice, je suis bien Bob"*
	$c = \{h(m)\}_{sk_B}$

Authentification–handshake

Alice n'a pas forcément déjà connaissance de la clé publique de Bob. Comment informer sûrement quelqu'un de sa clé publique ?

$A \to B$	*"Bonjour"*
$B \to A$	*"Bonjour, je suis Bob. Voici ma clé publique"* pk_B
$A \to B$	*"Prouve-le."*
$B \to A$	$m =$ *"Alice, c'est bien Bob"*
	$c = \{h(m)\}_{sk_B}$

N'importe qui peut se faire passer pour Bob aux yeux d'Alice, par une attaque MIM.

Transmettre un certificat–handshake

Certificat garantit la relation entre une identité et clé publique.

$A \to B$	*"Bonjour"*
$B \to A$	*"Bonjour, je suis Bob. Voici mon certificat"* $cert_B$
$A \to B$	*"Prouve-le."*
$B \to A$	$m =$ *"Alice, c'est bien Bob"*
	$c = \{h(m)\}_{sk_B}$

Eve pourrait se substituer à Bob dans les premiers échanges, mais échouera au dernier.

Echanger un secret–handshake

La communication par clés publiques est coûteuse, une fois finie la phase d'authentification, on échange une clé symétrique.

$A \to B$	*"Bonjour"*
$B \to A$	*"Bonjour, je suis Bob. Voici mon certificat"* $cert_B$
$A \to B$	*"Prouve-le".*
$B \to A$	$m =$ *"Alice, c'est bien Bob"*
	$c = \{h(m)\}_{sk_B}$
$A \to B$	*"Ok Bob, voici notre secret :"*
	$s = \{secret\}_{pk_B}$
$B \to A$	$m' = \{\text{message de Bob}\}_{secret}$

Attaque MIM

L'homme du milieu Melchior peut s'interposer dans les 5 premiers échanges. Arrivé au sixième, il peut brouiller le message de Bob, quitte à ne pas envoyer un message très intelligible à Alice :

$B \to M$	$m' = \{\text{message de Bob}\}_{secret}$
$M \to A$	altération de m'

Alice n'a aucune certitude quant à l'existence de Melchior, même si elle trouve suspect le dernier message de Bob.

SSL–handshake

Pour éviter cette incertitude, on utilise un MAC :

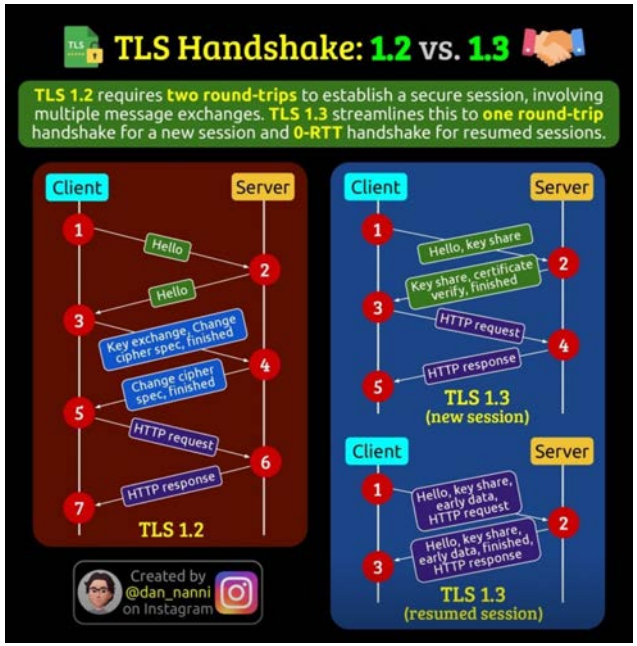
$M = h(\text{un message de Bob}, secret)$

$A \to B$	*"Bonjour"*
$B \to A$	*"Bonjour, je suis Bob. Voici mon certificat"* $cert_B$
$A \to B$	*"Prouve-le".*
$B \to A$	$m =$ *"Alice, c'est bien Bob"*
	$c = \{h(m)\}_{sk_B}$
$A \to B$	*"Ok Bob, voici notre secret :"*
	$s = \{secret\}_{pk_B}$
$B \to A$	$m' = \{\text{message de Bob}\}_{secret}$
	$M = h(\text{message de Bob}, secret)$

Melchior peut pertuber ce qu'il veut, M aura au moins l'avantage d'en avertir le destinataire.

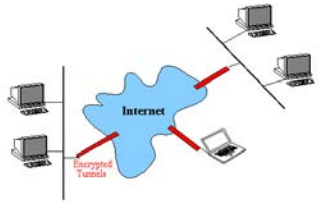
La communication

Ce protocole permet de transmettre un message de taille arbitraire. Il le découpe en blocs, le comprime éventuellement, ajoute un MAC, chiffre et transmet le résultat en ajoutant un numéro de séquence pour détecter s'il manque des messages ou si certains ont été altérés.



VPN

- Réseau privé qui utilise Internet pour connecter :
  - des pairs distants
  - des sites distants
- VPN utilise des connexions virtuelles routées au travers d'Internet vers l'entité distante



Un VPN :

- Etend la connectivité géographique
- améliore la sécurité
- réduit les coûts par rapport à un WAN dédié
- simplifie la topologie réseau

VPN – modes de fonctionnement

passent par l'interface `tun/tap` ; différence au niveau de la couche OSI

- **Mode routé (routed) :** relie des machines distantes. Travaille au niveau de la couche 3 (réseau), ie au niveau d'IP. utilise interface `tun`. Etablit route spécifique entre adresses réseau différentes. Pas de broadcast. Fonctionne en point à point.
- **Mode pont (bridge) :** relie des réseaux distants. Travaille au niveau de la couche 2 (liaison) par protocole dédié `PPTP`, `EoIP`, `IPSec`. utilise interface `tap`. ifaces VPN et LAN liées entre elles en une seule entité ; adresse VPN donnée dynamiquement au client. Routage entre réseaux fait par tables de routage au niveau du serveur VPN. Permet le broadcast et assure transparence complète.

TUN/TAP

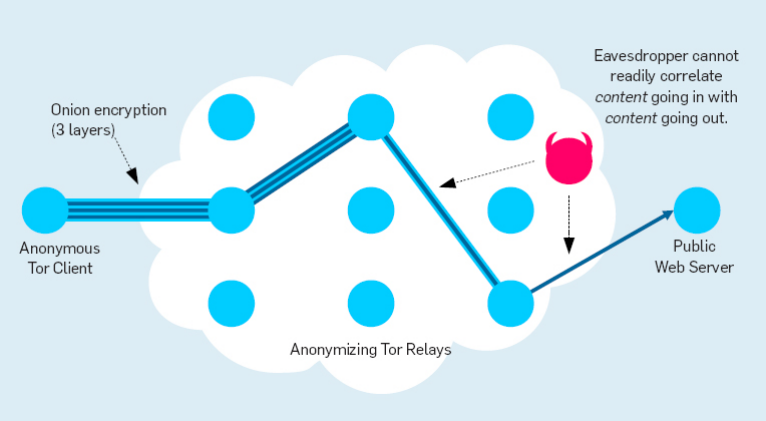
TUN/TAP représentent des périphériques réseaux dans un réseau virtuel.

- **TUN** ; ou network tunnel simule un périphérique de lien réseau et agit sur les paquets de la couche 3 (IP)
- **TAP** : ou network tap simule un périphérique de lien réseau et agit sur les paquets de la couche 2 (liaison des données), comme les trames ethernet

L'interface `tun` faire du routage et l'interface `tap` pour réaliser un pont réseau (bridge).

Les paquets envoyés par l'OS au périphérique `tun/tap` sont gérés par un programme de l'espace utilisateur qui peut aussi réaliser l'opération inverse, i.e. injecter les paquets vers l'OS.

Onion routing/TOR



Onion routing/TOR

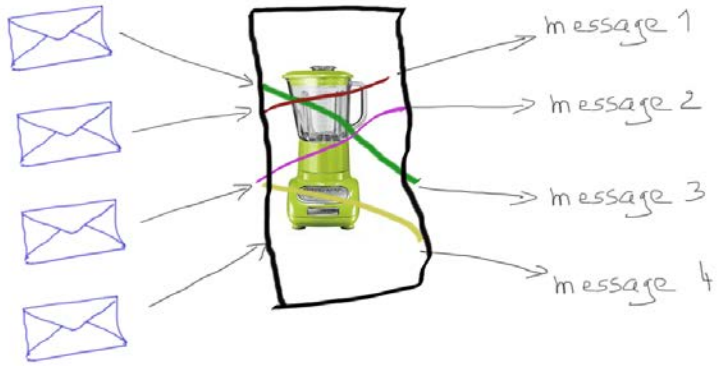
Sur Internet (public), les en-têtes des paquets identifient les parties qui communiquent.
Même en chiffrant la charge utile des paquets, les en-têtes sont accessibles et ne masquent pas l'information de routage.
Cette information fait partie de la vie privée :

- qui communique avec qui ?
- quels sites web regardez-vous ?
- où travaillez-vous, où habitez-vous ?
- vous achats, vos médecins, vos loisirs ?

Deux approches pour anonymiser les communications :

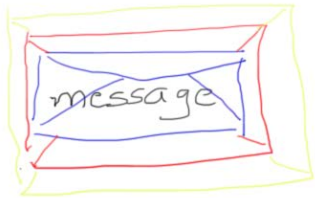
- mixees / mélangeur
- proxies / serveur mandataire

Mélangeur



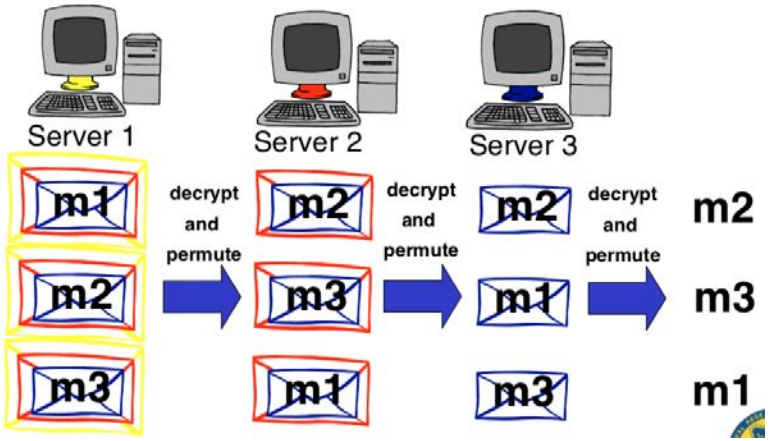
Déchiffre et permute les entrées.
Propriété principale : un adversaire ne peut pas associer un chiffré (une enveloppe) à un des messages.

Chiffrement d'un message



Avec Chiffré=$\{\{\{\text{message}\}_{PK3}\}_{PK2}\}_{PK1}$

Mélangeur de Chaum de base



Un seul serveur honnête préserve l'anonymat.

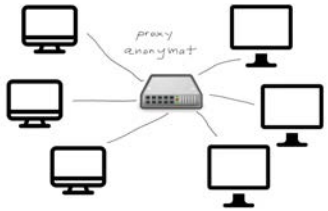
Inconvénients des mélangeurs...

Tout ce qui demande une interaction rapide :

- navigation web, connexion distante, chat, etc.
- mélangeurs introduits pour le mail et d'autres applications de latence élevée
- chaque enveloppe autour du message demande beaucoup de cryptographie à clé publique

Serveur mandataire d'anonymat

- liens semblent venir du proxy, pas de l'émetteur
- approprié pour communications à faible latence
- chiffrement symétrique
- **avantage** : simple ; permet d'anonymiser bcp de trafic
- **inconvénient** : un seul point de panne, compromission, attaque



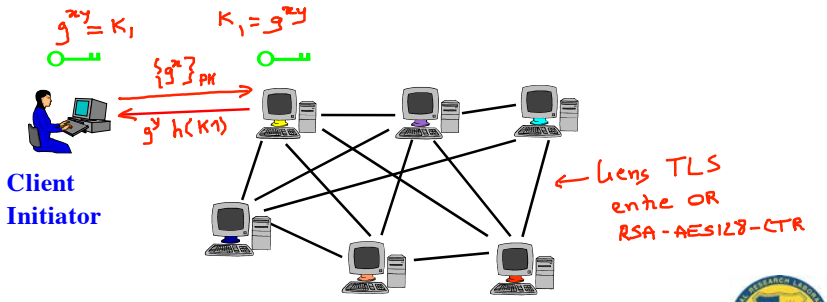
Routage en oignon

But : Construire une infrastructure robuste à l'analyse de trafic.

- Combiner les avantages des mélangeurs et des serveurs mandataires
- Utiliser PKC (coûteuse) pour établir des routes
- Utiliser crypto symétrique pour échanger des données
  - analogue à des proxies basés sur TLS
- Confiance distribuée par les mélangeurs
- Pas mal de travaux, implémentation, POC
  - ISDN mixes
  - Crowds, JAP webmixes, Freedom net
  - Tarzan, Morphmix
- https://www.torproject.org/docs/onion-services.html.en
- https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt
- https://www.onion-router.net/Publications/tor-design.pdf

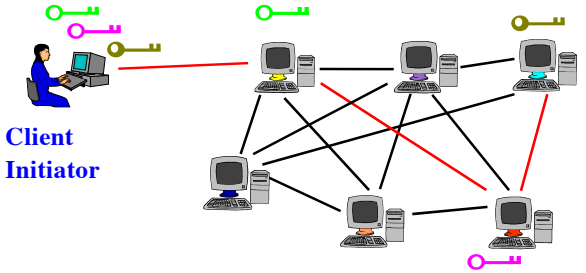
Tor Circuit Setup

- Client Proxy establishes session key and circuit w/ Onion Router 1



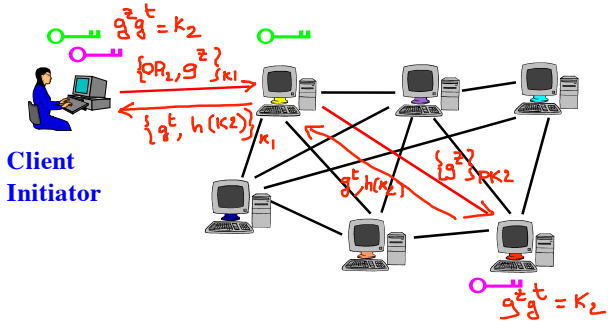
Tor Circuit Setup

- Client Proxy establishes session key and circuit w/ Onion Router 1
- Proxy tunnels through that circuit to extend to **Onion Router 2**
- Etc



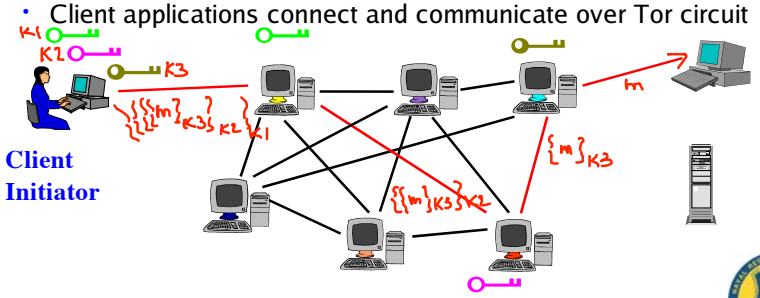
Tor Circuit Setup

- Client Proxy establishes session key and circuit w/ Onion Router 1
- Proxy tunnels through that circuit to extend to **Onion Router 2**



Tor Circuit Usage

- Client Proxy establishes session key and circuit w/ Onion Router 1
- Proxy tunnels through that circuit to extend to **Onion Router 2**
- Etc
- Client applications connect and communicate over Tor circuit



Bibliographie


W. Fumy.
Key management techniques.
In State of the art in applied cryptography, number 1528 in LNCS, pages 209–223. Springer Verlag, 1997.