

Cryptographie et tunnels

La cryptographie, c'est la science du chiffrement et du déchiffrement d'informations, dans le but de les rendre illisibles par des non initiés. Cependant, le chiffrement est toujours déchiffrable et la durée de vie de la confidentialité dépendra de l'investissement sur la force du chiffrement.

En informatique, c'est la base de tout échange sécurisé. La notion de sécurité inclut non seulement l'aspect confidentiel, mais également l'authentification des partenaires.

C'est une science très hautement mathématique et nous n'aborderons pas du tout cet aspect ici, c'est vraiment une affaire de spécialistes, et je n'en suis pas un.

Nous verrons plutôt comment mettre en oeuvre de tels procédés et dans quel cadre les utiliser, car la cryptographie n'est finalement qu'une fonctionnalité. Elle peut être appliquée à peu près à tous les niveaux du modèle OSI, principalement au niveau IP (avec par exemple IPsec) et au niveau des applications. A peu près tous les protocoles de haut niveau (HTTP, FTP, POP, IMAP...) peuvent la mettre en oeuvre. Il y a également SSH, le shell distant sécurisé, qui permet, entre autres, de disposer d'un terminal distant, un peu comme telnet, mais avec plus de sécurité.

Une application des plus intéressantes dans le domaine des réseaux est certainement la réalisation d'un tunnel sécurisé. Un tel procédé permet l'établissement d'un "réseau privé virtuel" (VPN) à travers un réseau de faible sécurité, comme l'internet, en créant une liaison point à point entre deux hôtes. La cryptographie intervient pour sécuriser cette liaison, pour la rendre "étanche".

Nous allons mettre en place un tel tunnel avec OpenVPN, et ce tunnel, nous allons le sécuriser au moyen d'OpenSSL.

Plan du chapitre

Avant propos.....	4
Position du problème.....	4
Rendre un message incompréhensible par le non initié.....	4
Juste un algorithme secret.....	4
Un algorithme et une clé.....	5
Cacher un message dans un autre message.....	5
Un peu de vocabulaire.....	7
Un peu de philosophie.....	7
Les clés du chiffrement.....	9
Confidentialité.....	9
Contrôle d'intégrité et authentification.....	9
Intégrité.....	9
Authentification.....	10
Non répudiation.....	10
Les clés, leurs types et leurs utilités.....	10
Chiffrement symétrique.....	11
Chiffrement asymétrique.....	11
Principes de base.....	11
Authentification.....	12
Confidentialité.....	13
Les deux.....	14
Un échange authentifié et confidentiel.....	15
Le tiers de confiance.....	16
Les certificats.....	17
Arbres et réseaux de confiance.....	20
Le réseau de connaissances.....	20
Les PKI.....	20
Bref.....	20
Méthodes.....	22
A quel niveau chiffrer.....	22
Mise en application.....	22
OpenVPN simple.....	24
La plate-forme de tests.....	24
Démarrage du serveur.....	25
Démarrage du client.....	27
Contrôle du tunnel.....	30
Un petit coup de sniffeur.....	31
Premières conclusions.....	32
OpenSSL.....	33
Un petit peu de sadisme.....	33
La stratégie.....	33
Un certificat autosigné.....	34
Un jeu de clés.....	34
AARON contresigne.....	35
Au tour de CYCLOPE.....	38

Création de la clé privée et de la demande de certificat.....	38
Et pour cacher les données dans le tunnel ?.....	40
Tunnel sécurisé.....	41
La dernière ligne droite.....	41
Sur AARON.....	41
Sur CYCLOPE.....	43
La touche finale.....	46
Applications.....	48
Un tunnel sécurisé entre les deux hôtes.....	48
Connectés directement à l'Internet.....	48
Ou derrière un pare-feu.....	48
Interconnecter les deux réseaux privés.....	49

Avant propos

Position du problème

Rendre un message incompréhensible par le non initié

Transmettre de l'information de telle manière que seuls les initiés puissent l'utiliser n'est pas un concept nouveau, il est même la base de la communication.

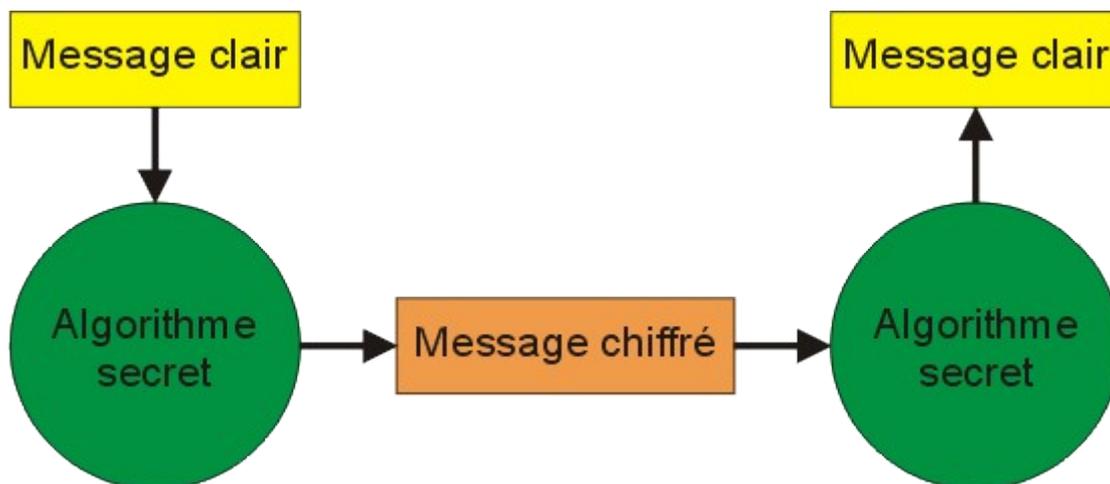
Un internaute qui ne connaît pas le français sera bien incapable de comprendre quoi que ce soit à la lecture de ces pages. Dans l'exemple, cependant, rien n'interdit à cet internaute d'apprendre le français pour en déchiffrer le contenu. Tout le monde peut apprendre le français, il n'y a aucun secret dans cette langue (juste pas mal de difficultés, mais pas de secret).

Dans de nombreux cas, cette "protection" ne sera donc pas satisfaisante.

Pour qu'un langage devienne un moyen de transfert de données confidentielles, il faut que l'apprentissage de ce langage soit impossible. Impossible n'étant pas français, il faudra donc se contenter d'en rendre l'apprentissage le plus difficile possible.

Juste un algorithme secret

Il peut s'avérer nécessaire de créer un langage spécifique, confidentiel, dont l'apprentissage reste difficile, idéalement impossible, pour le non initié. Dans un tel cas, nous disposons d'un algorithme de chiffrement qui assure à lui seul la confidentialité du message. Aussi longtemps que cet algorithme ne sera partagé que par les seules personnes autorisées, le secret restera inviolé.

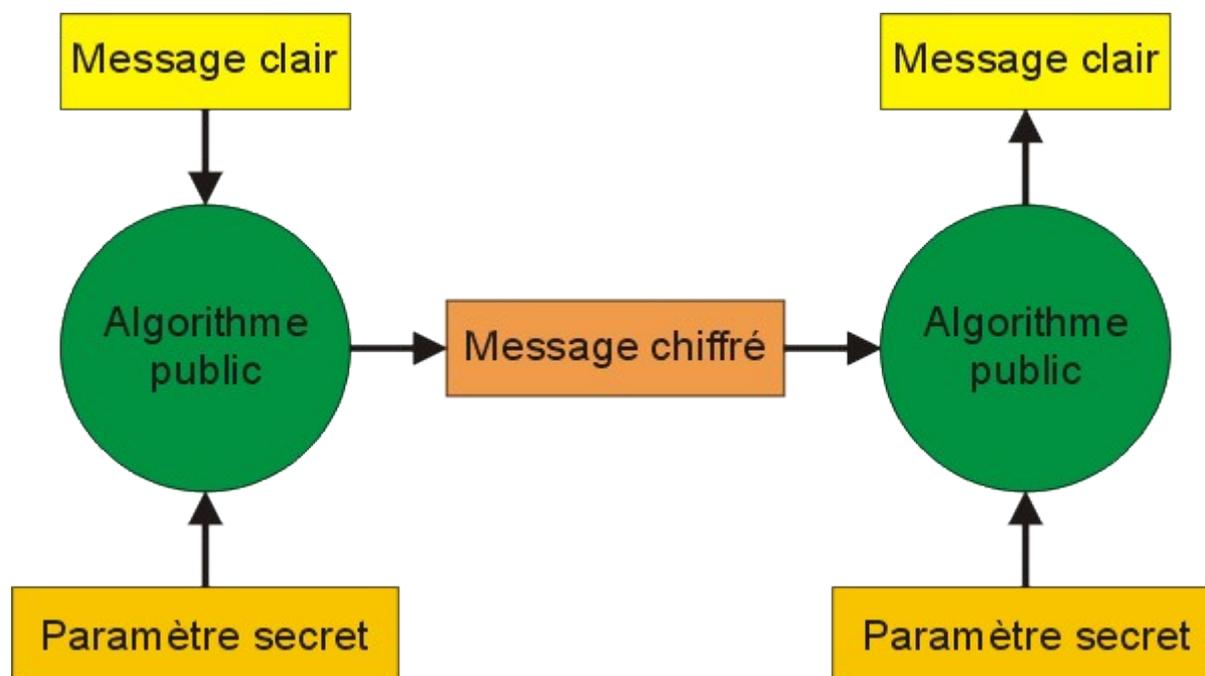


Un tel procédé, cependant, n'est pas considéré comme sûr. Que quelqu'un réussisse à reconstituer l'algorithme et il n'y aura plus de secret.

La réussite du procédé repose sur le fait qu'un nombre minimal de personnes sont dans le secret. Il se trouve que pour qu'un système soit réellement efficace, il faut qu'un maximum de personnes puisse en étudier le mécanisme, pour découvrir toutes les failles qu'il peut contenir. Il y a donc ici une incohérence fondamentale.

Un algorithme et une clé

Il est largement préférable d'utiliser un algorithme public, que tout le monde peut analyser et utiliser, mais qui exploitera un paramètre de chiffrement qui, lui, ne sera pas partagé. Si l'on ne connaît pas la valeur de ce paramètre, même en disposant de l'algorithme, il sera impossible de déchiffrer le message. Le paramètre secret s'appelle une clé de chiffrement.



Ce principe, qui peut éventuellement adopter des combinaisons de clés, comme nous le verrons plus loin, reste à l'heure actuelle le procédé le plus sûr. Ici, pour déchiffrer le message, il "suffira" de trouver la bonne clé, l'algorithme étant public. La difficulté avec laquelle une clé non connue pourra être retrouvée sera seule garante du secret.

En ce qui concerne l'algorithme, comme chacun peut l'utiliser et l'analyser, s'il possède une faille, elle sera facilement et rapidement découverte et corrigée.

Cacher un message dans un autre message

Il peut être amusant, utile, nécessaire, voire les trois à la fois, de cacher un message dans un autre. Un message chiffré, on voit qu'il est chiffré et on essaye de le déchiffrer. Un message caché, on ne le voit pas et donc on ne cherche pas à le retrouver si l'on ne sait pas qu'il existe.

La science de la dissimulation d'un message dans un autre s'appelle la stéganographie.

Bien que cette approche n'entre pas dans le cadre de cet exposé, il est intéressant de savoir que ça existe.

Un exemple classique s'il en est : une correspondance écrite entre George Sand à Alfred de Musset.

George écrit ceci à Alfred :

Je suis très émue de vous dire que j'ai bien compris, l'autre jour, que vous avez toujours une envie folle de me faire danser. Je garde un souvenir de votre baiser et je voudrais que ce soit là une preuve que je puisse être aimée par vous. Je suis prête à vous montrer mon Affection toute désintéressée et sans calcul. Si vous voulez me voir ainsi dévoiler, sans aucun artifice mon âme toute nue, daignez donc me faire une visite Et nous causerons en amis et en chemin. Je vous prouverai que je suis la femme sincère capable de vous offrir l'affection la plus profonde et la plus étroite Amitié, en un mot, la meilleure amie que vous puissiez rêver. Puisque votre âme est libre, alors que l'abandon où je vis est bien long, bien dur et bien souvent pénible, ami très cher, j'ai le coeur gros, accourez vite et venez me le fait oublier. À l'amour, je veux me soumettre.

Comme c'est beau, comme c'est joliment dit, comme c'est poétique (normal, c'est George Sand, quand même) !

Bon, maintenant, lisez le texte, mais seulement une ligne sur deux, en sautant les lignes paires. Vous constaterez que le message devient tout de suite nettement plus direct...

Et, pour continuer la démonstration, lisez donc la réponse que fait Alfred de Musset :

Quand je vous jure, hélas, un éternel hommage
Voulez-vous qu'un instant je change de langage
Que ne puis-je, avec vous, goûter le vrai bonheur
Je vous aime, ô ma belle, et ma plume en délire
Couche sur le papier ce que je n'ose dire
Avec soin, de mes vers, lisez le premier mot
Vous saurez quel remède apporter à mes maux.

Lisez bien, l'astuce stéganographique est fournie avec.

Georges Sand, qui n'est pas de celles qui abandonnent en chemin, conclut de la sorte :

Cette grande faveur que votre ardeur réclame
Nuit peut-être à l'honneur mais répond à ma flamme.

A quoi bon perdre son temps en attentes inutiles. Plus vite c'est fait, plus vite on pourra le refaire...

Il est possible d'inventer une foule de façons de dissimuler un message dans un autre

Avec l'information numérique, il devient possible de cacher à peu près n'importe quoi dans n'importe quoi. Souvent dans des images.

Il est aussi possible, au cas où, d'utiliser une méthode de chiffrement sur le message caché.

Un peu de vocabulaire

Ce qui suit est directement inspiré de l'ouvrage :

Cryptographie Appliquée

De Bruce Schneier

Traduit en français par Laurent Viennot

Publié chez Vuibert.

Une sorte de bible pour qui veut approfondir sérieusement le sujet.

- Un message en **texte clair** est un message que tout le monde peut interpréter, soit directement, soit avec un outil de traduction (un dictionnaire et une grammaire par exemple),
- la transformation d'un message pour le rendre incompréhensible, même avec un outil de traduction est appelée **chiffrement** (encryption, éventuellement),
- le résultat du chiffrement donne un texte chiffré (ou cryptogramme),
- l'action inverse s'appelle le **déchiffrement**, elle permet de restituer le texte en clair,
- l'art de chiffrer et de déchiffrer s'appelle la **cryptographie**, les spécialistes en la matière sont des **cryptographes**,
- ceux qui s'amuse à essayer (et parfois même à arriver) à déchiffrer un message sans en connaître la ou les clés sont les **cryptanalystes**, ils font de la **cryptanalyse**,
- la branche mathématique sur laquelle s'appuient la cryptographie et la cryptanalyse s'appelle la **cryptologie** et les spécialistes de cette chose sont les **cryptologues**.

Un peu de philosophie

Comme il est illusoire de penser que l'on pourra mettre un jour au point un procédé de chiffrement qui ne sera jamais "cassable" par un cryptanalyste, il faut se résoudre à considérer qu'un système de chiffrement est forcément vulnérable et donc l'utiliser dans un domaine où il conservera son maximum d'efficacité.

Ce domaine d'efficacité peut s'évaluer en considérant quelques critères :

- les efforts déployés pour casser un chiffrement seront proportionnels à l'intérêt qu'il y a à obtenir les données déchiffrées (obtenir le moyen de déchiffrer des transactions bancaires peut être plus motivant que de percer le secret d'un e-mail que M. X envoie à Mlle. Y),
- pour casser un chiffrement, il faut de la puissance de calcul (puissance=effort/temps). Il faut donc utiliser un chiffrement pas plus longtemps que le temps nécessaire à le casser, avec les moyens de calculs supposés pouvoir être mis en oeuvre pas les attaquants (les moyens incluent non seulement le potentiel de calcul, mais également les algorithmes de recherche),
- plus le volume de données chiffrées avec la même méthode est important, plus il fournit aux attaquants du matériel de travail, il faut donc ne pas dépasser un volume critique avec le

même chiffrement.

- une information n'a généralement pas besoin de rester indéfiniment secrète. "Demain, on débarque sur les plages de l'Atlantique". Après demain, cette information n'aura plus besoin de rester secrète. Il suffit donc de trouver un procédé de chiffrement qui puisse résister 24h.

Il faut comprendre de tout ceci que la sécurité introduite par un procédé de chiffrement reste relative. Elle n'est fonction que de l'efficacité du procédé en rapport à l'intérêt qu'il y a à le casser.

Les clés du chiffrement

Résumons nous.

En cryptographie numérique, chiffrer une information consiste à modifier la suite d'octets qui constituent cette information au moyen d'un algorithme mathématique.

L'algorithme étant normalisé, il peut être connu de tout le monde. Ainsi, pour que le secret soit assuré, il faudra que cet algorithme utilise un paramètre supplémentaire appelé une clé. Une information chiffrée avec un algorithme connu restera déchiffrable par les seuls possesseurs de la clé appropriée, même si l'algorithme est connu de tous. Les clés de chiffrement sont donc les éléments essentiels dans la garantie du secret souhaité.

Une fois le procédé mis en place, nous pouvons en attendre quelques services.

Confidentialité

L'usage auquel on pense en premier est naturellement la confidentialité des données. Le premier désir est bien que les messages ne soient lisibles que par les seules personnes autorisées.

C'est bien, mais c'est loin de suffire, pour assurer une totale relation de confiance.

Contrôle d'intégrité et authentification

Intégrité

Dans certains cas, il peut être nécessaire d'assurer simplement que les données sont intègres, c'est à dire qu'elles n'ont pas été au passage falsifiées par un intrus. Ces données restent "claires", au sens où elles ne sont pas secrètes.

Un exemple simple : je propose en téléchargement un fichier contenant une application informatique. Je voudrais éviter qu'un intrus ne puisse, par un moyen quelconque, dénaturer mon application et y introduisant, par exemple, un "root kit".

Mon code est sous licence GPL, les sources sont disponibles, je ne veux pas rendre mon application secrète, je veux juste en assurer l'intégrité du code.

Je vais donc réaliser un "résumé concis" de mon fichier, typiquement une somme MD5¹ (ou SHA²). Ce résumé est suffisamment précis pour qu'il puisse mettre en évidence toute modification

1 MD5

Message Digest 5. Fonction définie dans la RFC 1321. Il s'agit d'un hachage unidirectionnel, permettant d'identifier un message, car deux messages produiront deux hachages différents, et il est impossible de retrouver le message à partir de son hachage.

2 SHA

Secure Hash Algorithm. Algorithme de hachage Sécurisé, c'est-à-dire qu'il calcule l'empreinte d'une suite d'octets. L'entrée peut être de taille quelconque, la sortie fait toujours 20 octets. Les caractéristiques de SHA (comme tous les algorithmes de hachage) sont :

- l'irréversibilité : connaissant le haché d'un message, il est impossible de reconstituer le message,
- l'absence de collision : il est impossible de trouver deux messages différents produisant le même haché,
- en outre, la modification d'un seul bit du message d'entrée produit un haché qui aura en moyenne la moitié des octets différents .

ultérieure de mon fichier. Bien sûr, nous parlons ici de mathématiques. Un résumé MD5 d'un fichier contenant l'image ISO d'un DVD ROM d'installation d'une distribution linux, par exemple, serait de la forme : 5025c41edf87b679f036377013234d9b (MD5sum du DVD d'installation de la fedora core 2).

Ce résumé concis, ou empreinte, dispose de caractéristiques fondamentales :

- l'empreinte d'un message est complètement significative de ce message. Il n'y a pratiquement aucune chance que deux messages différents puissent avoir la même empreinte,
- la modification d'un seul bit dans le fichier original va considérablement modifier son empreinte,
- le procédé est irréversible, c'est à dire qu'il est impossible de reconstituer le message à partir de son empreinte.

Bien. Mais sans précautions supplémentaires, ça ne va pas suffire, parce que celui qui arrivera à corrompre mon fichier en téléchargement, pourra sans doute modifier aussi l'empreinte, pour qu'elle soit celle du fichier corrompu.

Une empreinte, si elle utilise bien un procédé de chiffrement, ne dispose pas d'un secret. l'algorithme utilisé (MD5, SHA...) est public et s'il n'est pas possible de reconstituer un message à partir de son empreinte, il est en revanche tout à fait possible pour quiconque de recalculer une empreinte après modification de l'information.

Typiquement, une somme MD5, sans précautions particulières, ne servira qu'à vérifier que le fichier n'a pas été corrompu lors du téléchargement, mais elle n'apportera pas la preuve de l'authenticité du fichier en téléchargement.

Il faudra donc trouver en plus, un moyen pour certifier l'authenticité de cette empreinte.

Authentification

Il s'agit d'apporter par la cryptographie la preuve que le message est bien authentique. Compte tenu de ce que nous venons de voir, il suffira dans la plupart des cas de pouvoir assurer que l'auteur de l'empreinte du message est bien celui qu'il prétend être. Il s'agit en quelque sorte d'une lettre manuscrite, non raturée et signée.

Non répudiation

C'est le corollaire direct de l'authentification. Celui qui a rédigé une lettre manuscrite, non raturée et signée de sa main, ne pourra en aucun cas prétendre par la suite qu'il n'en est pas l'auteur. Il en va de même pour l'authentification numérique.

Les clés, leurs types et leurs utilités

Il y a deux types de clés :

- les clés symétriques, on chiffre et déchiffre avec la même clé.
- les clés asymétriques, avec une clé publique et une clé privée, ce qui est chiffré avec l'une ne peut être déchiffré qu'avec l'autre.
Les deux clés sont uniques et sont liées l'une à l'autre, mais si la clé privée reste

confidentielle, la clé publique, elle, peut être copiée à volonté.

Il s'agit en fait d'un abus de langage. Les clés ne sont pas symétriques ni asymétriques, ce sont les procédés de chiffrement qui le sont :

Chiffrement symétrique

C'est le plus facile à comprendre, c'est aussi la méthode de chiffrement la plus facile à réaliser et qui consomme le moins de ressources de calcul et de bande passante.

Les deux hôtes qui doivent échanger des données confidentielles (secrètes) disposent tous les deux d'une clé identique. L'émetteur chiffre les données avec, puis les envoie au récepteur. Ce dernier déchiffre avec la même clé pour récupérer des données lisibles.

Cette méthode assure la confidentialité des données, celui qui intercepterait la communication ne pourra pas lire les données échangées tant qu'il n'aura pas pu se procurer la clé. Il n'y a aucune authentification de faite sur l'émetteur comme sur le récepteur, sauf si deux personnes seulement disposent de la clé.

Le principal souci avec cette méthode, c'est qu'il faut s'échanger la clé et lors de cet échange, sans précautions particulières, n'importe quoi peut se produire.

Chiffrement asymétrique

Cette méthode permet de faire aussi bien **l'authentification** que la **confidentialité** des données. Il est évidemment possible de combiner les deux.

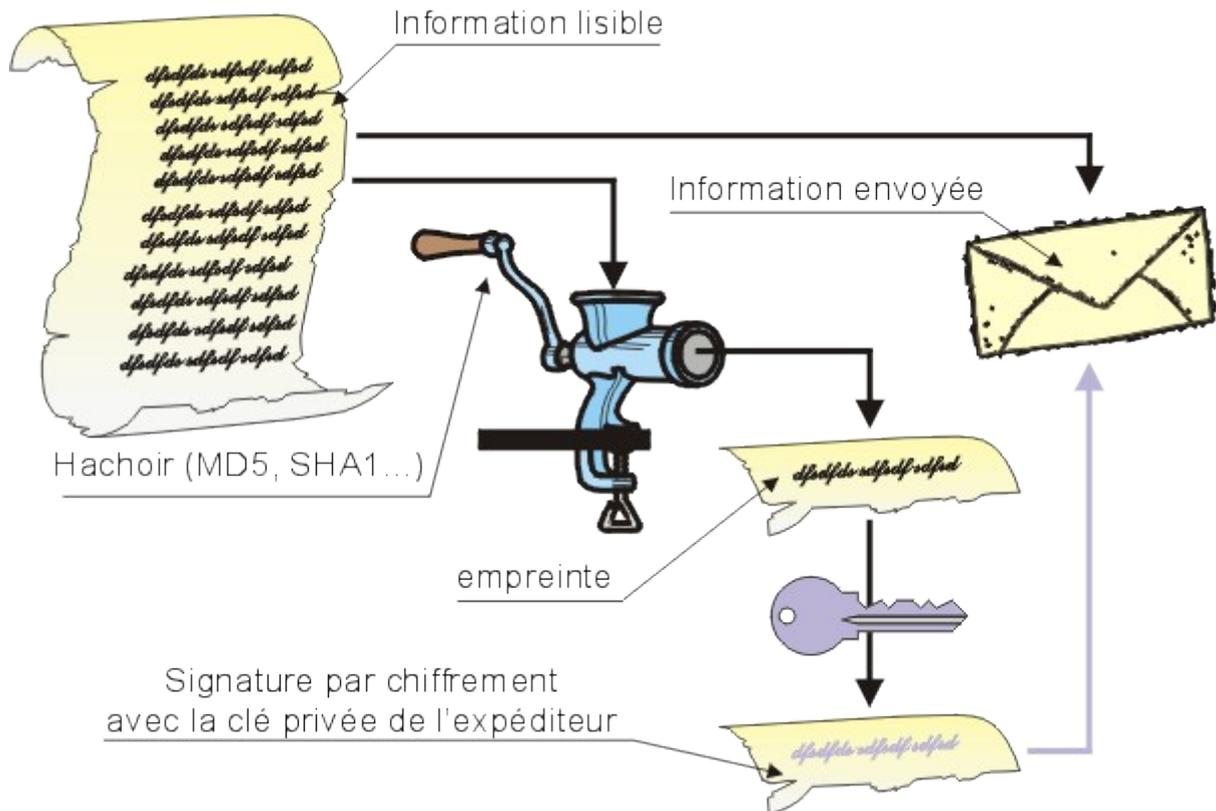
Principes de base

Là, c'est un peu plus complexe et il vaut mieux bien suivre pour ne pas se perdre :

- chaque personne dispose d'un jeu de clés comportant :
 - une clé privée : elle est unique et confidentielle, elle appartient exclusivement à l'hôte concerné, il ne la distribue à personne, aucun double de cette clé ne doit être créé,
 - une clé publique : elle est unique également, mais tout le monde peut s'en procurer une copie, il suffit d'aller la chercher chez un dépositaire, dit "tiers de confiance". Il s'agit en quelque sorte d'un concierge qui garde ces clés publiques et certifie qu'elles appartiennent bien à la personne indiquée,
- ce qui est chiffré avec une clé publique ne peut être déchiffré qu'avec la clé privée correspondante,
- ce qui est chiffré avec la clé privée ne peut être déchiffré qu'avec la clé publique correspondante.

Tout ceci peut à première vue paraître absurde, puisqu'il y a à chaque fois une clé qui peut être récupérée par n'importe qui. Oui mais...

Authentification



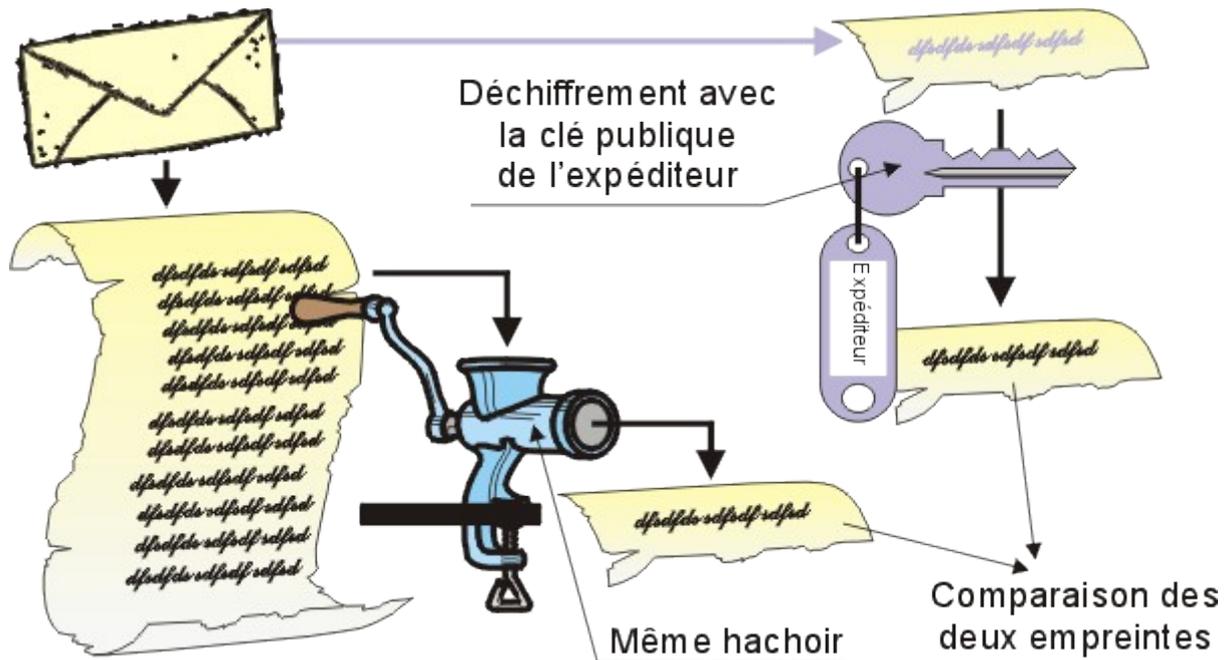
Je réalise une empreinte avec un algorithme non réversible (MD5, SHA...). Cette empreinte ne permet pas de reconstruire le message original, mais elle représente "à coup sûr" un résumé de mon message original. Un simple bit modifié dans le message donnerait une empreinte complètement différente.

Je chiffre cette empreinte avec **ma** clé privée et l'envoie, avec le message en clair, au destinataire.

Mon message n'est pas confidentiel, il est envoyé en clair. Mais :

- L'empreinte réalisée est intimement attachée au contenu de mon message, si le message est intercepté et modifié, l'empreinte devra être recalculée,
- comme l'empreinte a été signée avec ma clé privée, je suis le seul à pouvoir le faire, donc toute modification ultérieure à l'envoi ne pourra pas être signée avec la bonne clé (aussi longtemps que ma clé privée restera secrète).

Lorsque le destinataire recevra le message signé :



- le destinataire recalcule l'empreinte du message reçu,
- il déchiffre l'empreinte signée avec la clé publique de l'expéditeur,
- il compare les deux empreintes. Si elles sont identiques, c'est que le message :
 - est bien conforme à l'envoi, puisque les empreintes sont identiques,
 - est bien envoyé par l'expéditeur propriétaire de la clé publique avec laquelle l'empreinte signée a été déchiffrée.

Voilà résolu le problème de la signature (authentifiée). Ce type de signature engage l'expéditeur. Il ne peut nier avoir envoyé cette information, puisqu'il l'a signée sans falsification possible, pour autant que l'on soit sûr de l'authenticité de la clé publique.

Confidentialité

Je chiffre **mon** information avec **la clé publique du destinataire**, et je lui envoie cette information. N'importe qui peut faire de même, puisque le chiffrement se fait avec une clé publique. N'importe qui peut récupérer la clé publique de n'importe qui chez le concierge.

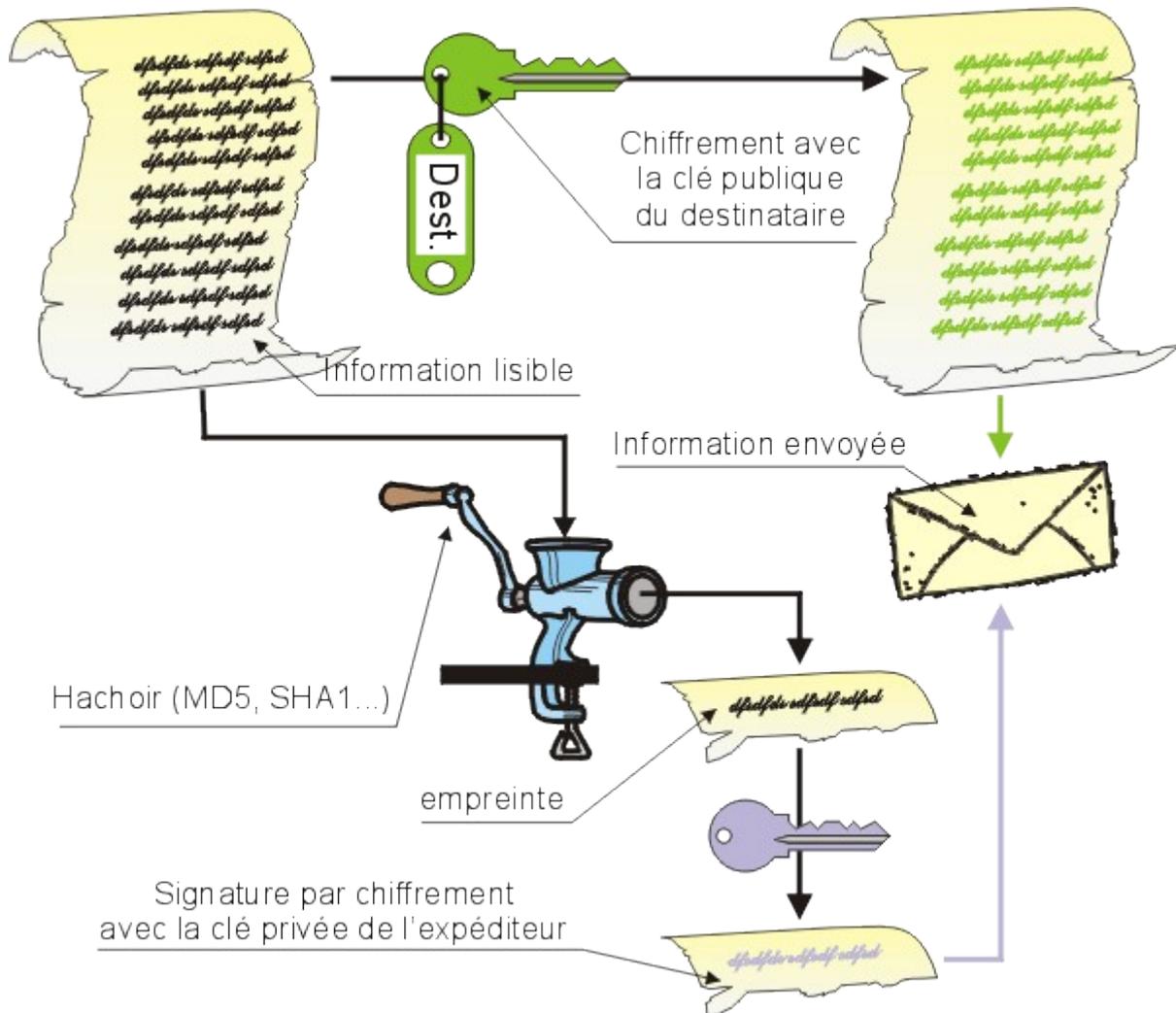
Donc, le message n'est pas authentifié, mais :

- comme j'ai chiffré avec **la clé publique du destinataire**, le message sera illisible pour qui ne détient pas **la clé privée correspondante**,
- comme seul le bon destinataire, normalement, détient cette clé privée, il sera le seul à pouvoir déchiffrer le message.

Je n'ai pas pu m'authentifier auprès du destinataire, mais j'ai pu lui envoyer un message confidentiel, puisqu'il est le seul à pouvoir le déchiffrer.

Les deux

Vous avez compris le principe ? Alors, faire les deux devient simple :



- Je réalise une empreinte de mon message et je la chiffre avec ma clé privée pour l'authentifier,
- je chiffre le message lui-même avec la clé publique du destinataire pour le rendre confidentiel,
- j'envoie le tout au destinataire.

Le destinataire déchiffre le message avec sa clé privée, en calculera localement l'empreinte, puis déchiffre l'empreinte envoyée avec ma clé publique :

- comme il est le seul à pouvoir déchiffrer avec sa clé privée, le message est bien confidentiel,
- comme je suis le seul à pouvoir chiffrer l'empreinte avec ma clé privée, le message est bien authentique.

Bon. Ça roule, mais vous comprenez bien que l'opération est lourde :

- il faut chiffrer et déchiffrer deux fois au lieu d'une, comme on le ferait avec une clé

symétrique,

- le chiffrement/déchiffrement avec des clés différentes est une opération mathématique plus lourde, plus coûteuse en ressources CPU.

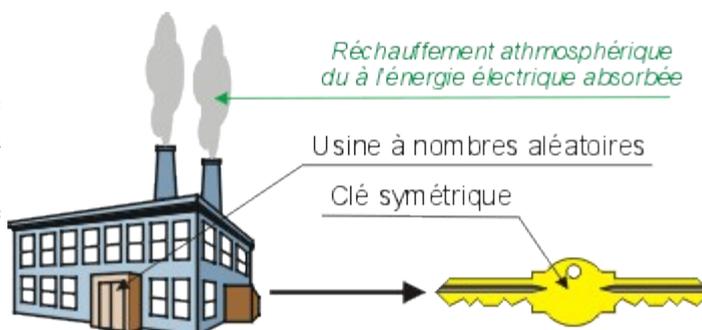
Rappelons que le principal problème du chiffrement symétrique, est qu'il faut s'échanger l'unique clé à un moment donné et que, lors de cet échange, quelqu'un pourrait l'intercepter.

Ce problème va être résolu magistralement de la façon suivante :

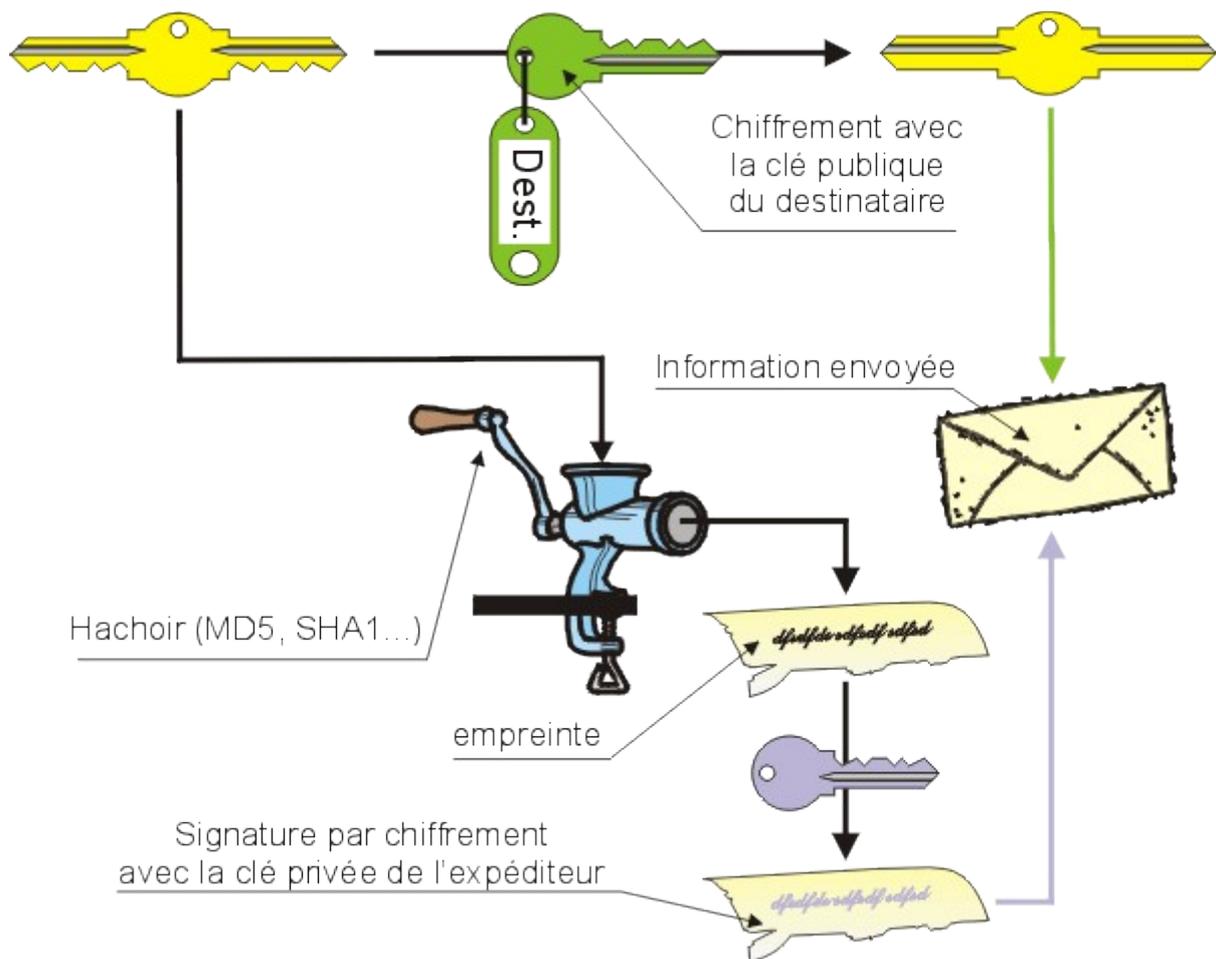
Un échange authentifié et confidentiel

Dans un échange d'informations (tunnel VPN par exemple), l'idéal, pour économiser les ressources CPU et augmenter le débit de l'échange, serait de trouver un moyen sûr d'échanger entre les partenaires une clé symétrique. Avec ce que nous savons, c'est relativement facile à réaliser.

Dans un premier temps, je fabrique une clé symétrique, à usage limité dans le temps, que nous appellerons une clé de session,



Je vais envoyer à mon interlocuteur cette "clé symétrique", que je vais authentifier et rendre confidentielle par les méthodes vues plus haut.



Et voilà le travail. Nous sommes deux à disposer de la même clé, transmise par une voie sécurisée.

La suite de l'échange pourra être chiffrée et déchiffrée avec cette clé de façon symétrique. Mais comme ça ne suffit pas, cette clé aura une durée de vie assez courte, quitte à devoir en échanger une nouvelle en cours de dialogue.

En effet, un intrus qui suivrait la discussion pourrait, au moyen d'outils faits exprès pour, finir par découvrir la clé symétrique, parce qu'il dispose d'un volume suffisant de données chiffrées, qu'il a eu le temps de trouver la clé grâce à la puissance de ses outils de cryptanalyse. Il faudra donc donner à la clé une durée de vie inférieure au temps nécessaire estimé pour la découverte de la clé.

Plus la clé sera "compliquée", plus ce temps sera long, mais plus les temps de chiffrement et de déchiffrement seront longs eux aussi. Tout est donc ici affaire de compromis.

Le tiers de confiance

Vous l'avez compris, cette magnifique mécanique ne fonctionnera qu'à une condition : le "concierge" (CA³) qui détient les clés publiques doit être digne de confiance, faute de quoi, n'importe quoi peut se produire...

Que pourrait-il arriver si une clé publique n'appartenait pas réellement à la personne indiquée ? Tout

³ CA
Certificate Authority. Tierce Partie de Confiance en français.

simplement une personne pourrait se faire passer pour une autre. Une fausse carte d'identité, en quelque sorte.

Tout repose donc sur la confiance que l'on peut mettre dans la personne qui détient les clés publiques.

Généralement, il s'agit d'un organisme de réputation sérieuse, à qui l'on confiera sa clé publique, et dont on détient la clé publique de façon sûre. La clé publique de l'organisme permettra de vérifier l'authenticité du dit organisme.

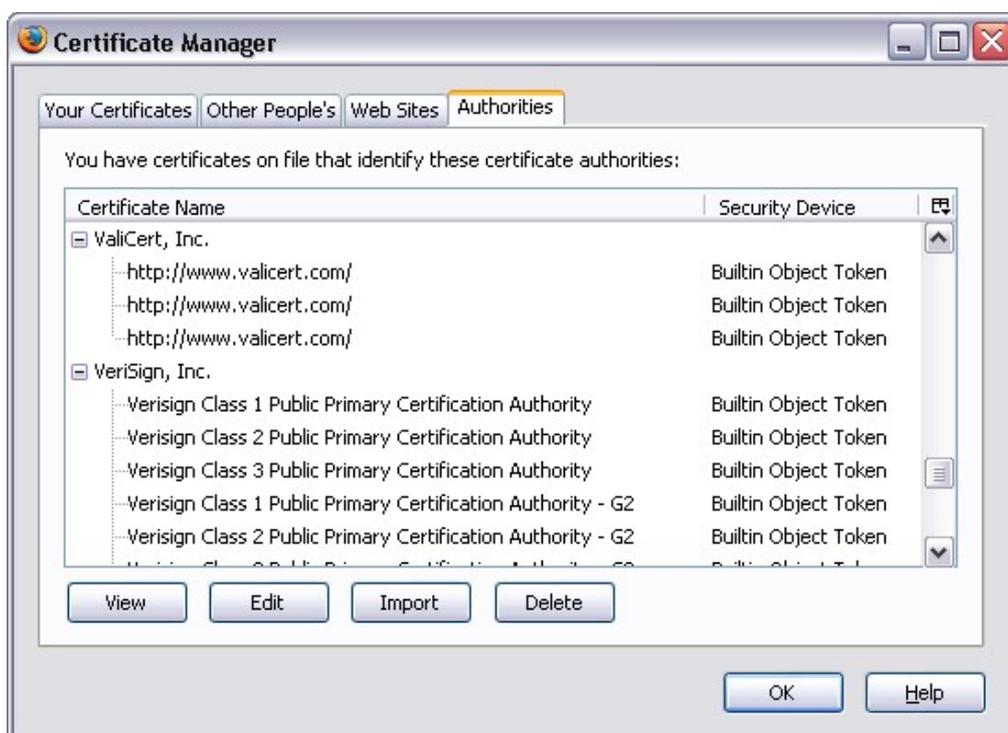
Les certificats

Il existe dans le monde plusieurs organismes (CA) de ce type. Leurs services, est-il nécessaire de le dire, ne sont pas gratuits, loin de là.

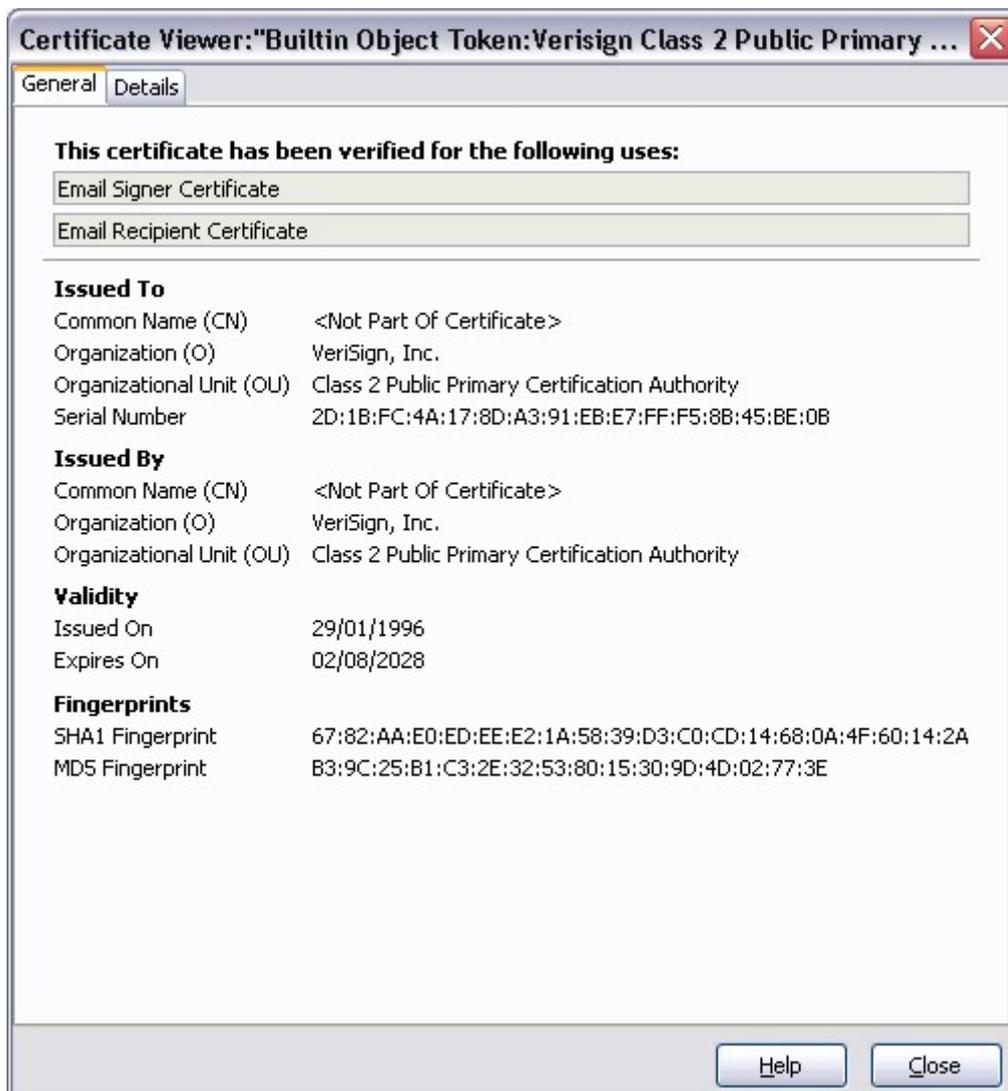
Lorsque je m'adresse à un tel organisme pour récupérer une clé publique, il me l'enverra avec un certificat. En gros, il s'agit pour ledit organisme de chiffrer la clé publique demandée, ainsi que quelques informations supplémentaires sur le propriétaire de cette clé, avec sa propre clé privée, pour authentifier son envoi. Il me reste à disposer de la clé publique du tiers de confiance, par un moyen sûr. Les certificats sont à la norme X.509.

Si je suis sûr de la clé publique du tiers et si j'ai confiance en lui, alors je pourrai mettre aussi ma confiance dans les clés publiques qu'il me distribuera.

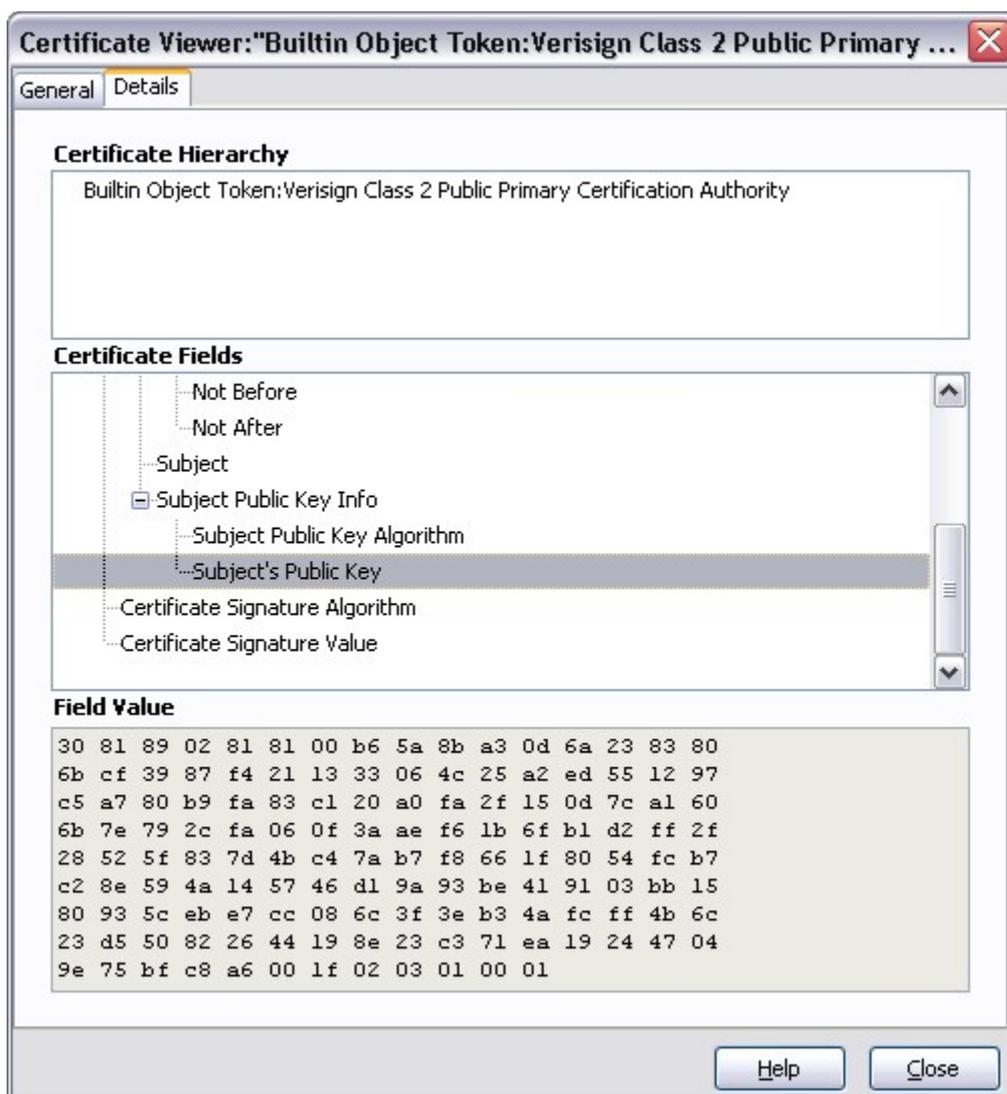
Votre navigateur a installé, peut-être à votre insu, des certificats de divers organismes de certification (exemple avec Mozilla Firefox) :



Voici un résumé du contenu d'un certificat :



Et bien entendu, le certificat contient une clé publique :



Comme une clé s'use (le volume de données chiffrées avec devient trop important, ce qui risque d'augmenter significativement les chances de découverte de cette clé), il faudra prévoir une durée de vie à cette clé et donc au certificat.

Comme une clé peut se perdre ou être volée, il faudra aussi prévoir un système "d'opposition". Une liste de révocation, qui permet de signaler les certificats non encore expirés, mais qui ne sont plus dignes de confiance pour une raison quelconque. La CA se doit donc de tenir à jour une liste de révocation, et, lorsqu'un utilisateur doit user d'un certificat qui est déjà en sa possession, il devrait commencer par vérifier si celui-ci n'a pas été révoqué entre temps.

Comme nous sommes dans un monde imparfait (d'ailleurs, si nous étions dans un monde parfait, les clés ne seraient d'aucune utilité), il peut se faire que l'usage de certains types de clés particulièrement "solides" soit réglementé. Il peut se faire qu'un exemplaire de la clé privée doive être mis sous séquestre, à disposition éventuelle des services de renseignements. Des organismes spécialisés dans cette mise sous séquestre de clés privées existent à cet effet.

Arbres et réseaux de confiance

Normalement, si deux interlocuteurs disposent chacun d'un certificat, mais chez des CA différentes, ils ne peuvent à priori se faire mutuellement confiance, sauf si par un procédé quelconque les deux CA affichent une confiance mutuelle. Entrer ici dans les détails nous mènerait vraiment trop loin, mais sachons tout de même que :

- les autorités de certification sont hiérarchisées, il y a des autorités racines et des autorités intermédiaires, ce qui aboutit à une structure arborescente. Si deux autorités intermédiaires sont certifiées par une même autorité hiérarchiquement supérieure, elles appartiennent au même arbre et donc la confiance est mutuelle, nous avons ici un arbre de confiance,
- deux arbres différents peuvent à un niveau quelconque (mais supérieur à celui des partenaires impliqués), établir entre eux un lien de confiance mutuelle, ce qui donne naissance à des réseaux de confiance.

Le réseau de connaissances

Une autre méthode consiste à utiliser un réseau de connaissances. J'ai, en principe, confiance dans mes amis et les amis de mes amis sont mes amis, donc, je peu faire confiance à une clé publique qui est contresignée par un ami dont je suis sûr. Ça peut aller loin, c'est le principe de l'homme qui a vu l'homme qui a vu l'homme qui a vu l'homme qui a vu Dieu.

C'est aussi le principe adopté en messagerie électronique par des procédés de signature comme GnuPG.

Les PKI

Les Public Key Infrastructures regroupent tout ce qui est nécessaire à la gestion des clés publiques et des certificats :

- récupération des demandes de certificats,
- réalisation des certificats,
- tenue de la liste de révocation,
- distribution des certificats aux demandeurs...

Lorsqu'une organisation désire mettre en place une telle infrastructure, sans avoir recours à des entreprises spécialisées, il lui est possible de le faire.

Bien entendu, les certificats produits n'auront de valeur qu'au sein de cette organisation. Il existe des projets Open Source qui proposent des outils de gestion d'infrastructure de clés publiques. Nous n'irons pas aussi loin dans cet exposé.

Bref...

Le principe est donc assez simple, il faut au départ disposer de clés publiques appartenant à des gens dont on est sûr. Si ces gens dont on est sûr me certifient comme étant authentiques des clés de gens que je ne connais pas, ces clés publiques seront à leur tour réputées sûres.

Vous voyez la limite d'un tel système ? Non ? Alors, imaginez que dans la chaîne, il y ait une clé

falsifiée qui passe pour authentique, à la suite d'une malversation quelconque. Alors, toutes les clés certifiées par cette clé falsifiée pourront être ou ne pas être authentiques...

Il est donc assez illusoire d'imaginer ce système comme parfaitement sûr. Fort heureusement, dans la plupart des cas, un seul administrateur certifiera les clés dont nous aurons besoin pour, par exemple, créer un tunnel sécurisé ou simplement une authentification entre deux noeuds d'un même réseau.

Méthodes

A quel niveau chiffrer

Dans la pile des protocoles réseau, il est courant d'utiliser le chiffrement :

- dans le noyau du système, au niveau IP, avec IPsec. Ainsi, tout ce qui passera sur le réseau sera automatiquement chiffré suivant les règles établies sans que les applications n'aient à s'en soucier,
- dans l'espace utilisateur, au niveau des applications elles-mêmes, qui choisiront ou non de chiffrer, avec par exemple, SSL (Secure Socket Layer). HTTPS en est une illustration, comme IMAPS, POPS ou encore SSH.

SSL, développé au départ par Netscape a été repris en OpenSource sous le nom de TLS : Transport Layer Security. Protocole de sécurisation de la couche transport, défini par la RFC 2246. La version 1.0 de TLS est en fait SSL v3 (définition donnée par "le jargon français"⁴)

Mise en application

Comme nous n'allons bien entendu pas nous contenter de théories, nous allons mettre en oeuvre SSL, qui est donc un procédé de chiffrement dans l'espace utilisateur, à travers un système de tunnels connu : OpenVPN.

Ailleurs dans ce site⁵, il est étudié un autre tunnel : GRE, qui n'est pas sécurisé et qui travaille en mode noyau, mais un tunnel reste un tunnel. Nous retrouverons donc des concepts communs.

OpenVPN est un système puissant, qui offre beaucoup de possibilités, qui vont du simple tunnel, n'apportant rien de plus qu'un tunnel GRE, jusqu'à un tunnel très sécurisé. OpenVPN propose :

- la compression des données, qui est une forme de chiffrement de données, dans la mesure où elle nécessite une décompression (déchiffrement) pour retrouver les données lisibles. Il n'y a pas de clé, c'est juste un algorithme (standardisé), donc ce n'est pas un moyen pour rendre le tunnel sécurisé, bien que ça y participe. La fonction principale est de réduire, pour une information donnée, le nombre de bits à faire circuler dans le tunnel, augmentant ainsi le débit apparent,
- l'authentification (signature) des données qui circulent,
- la confidentialité des données au moyen d'une clé de session, chiffrement symétrique au moyen d'une clé qui sera échangée entre les partenaires.

Toutes ces fonctionnalités peuvent être mises en oeuvre, si nécessaire, mais ce n'est pas une obligation, ce qui permet d'adapter le tunnel à ses besoins du moment, en conservant le maximum de bande passante.

Contrairement à d'autres concurrents, nous allons ici faire passer une connexion TCP dans un tunnel réalisé sur UDP. L'avantage est que le support du tunnel est plus souple qu'une connexion TCP, qui nécessite l'établissement complet d'une nouvelle connexion en cas de rupture, ce qui oblige par voie

4 Jargon français : <http://www.linux-france.org/prj/jargonf/T/TLS.html>

5 VPN : <http://christian.caleca.free.fr/vpn/>

de conséquence à rétablir également la connexion TCP dans le tunnel.

OpenVpn fonctionne sur le modèle "client/serveur" :

- l'un des bouts du tunnel sera serveur, à savoir qu'il restera en permanence à l'écoute sur un port UDP, il est bien sûr fort intéressant de disposer d'une adresse IP fixe sur ce bout du tunnel,
- l'autre bout du tunnel sera un client, il décidera quand il veut établir le tunnel et n'aura pas besoin de disposer d'une adresse IP fixe. Mieux, même si son IP change alors que le tunnel est établi, OpenVpn saura gérer cette situation.

OpenVpn, du fait qu'il utilise un seul port UDP, pourra facilement passer à travers un routeur/pare-feu, si ce routeur sait faire du "port forwarding". Ainsi il sera possible d'établir un tunnel entre deux machines situées toutes deux dans des réseaux distants, connectés à l'internet par des routeurs/pare-feux, sans rien avoir à installer sur ces routeurs, il suffira juste de "forwarder" le port UDP utilisé pour le tunnel.

Avec un peu d'astuce, il sera possible de relier par un tunnel OpenVpn :

- deux machines entre elles et seulement ces deux machines, qu'elles soient derrière un routeur ou non,
- une machine unique à un réseau distant,
- deux réseaux distants.

OpenVPN simple

La plate-forme de tests

Deux machines disposent d'une connexion internet.

L'une s'appelle AARON, elle dispose d'une adresse IP publique fixe : 82.127.57.95.

L'autre s'appelle CYCLOPE, et dispose d'une adresse IP dynamique : 80.8.135.67, au moment de ce premier test.

Les deux machines sont des Debian Sarge (testing), avec un kernel 2.6. En effet, OpenVPN ne figure pas dans l'actuelle version stable (Woody). D'autres distributions, comme Mandrake ou Fedora Core feraient aussi parfaitement l'affaire. Si vous utilisez un kernel 2.4, il vous faudra installer le support de TUN/TAP, qui permet à l'espace utilisateur d'accéder aux interfaces réseaux, ce support est nativement inclus dans 2.6.

```
aaron:~# apt-get install openvpn
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  openvpn
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 166kB of archives.
After unpacking 463kB of additional disk space will be used.
Get:1 ftp://ftp.fr.debian.org testing/main openvpn 1.6.0-3 [166kB]
Fetched 166kB in 12s (13.1kB/s)
Preconfiguring packages ...
Selecting previously deselected package openvpn.
(Reading database ... 21897 files and directories currently installed.)
Unpacking openvpn (from ../openvpn_1.6.0-3_i386.deb) ...
Setting up openvpn (1.6.0-3) ...
Stopping openvpn:..
Starting openvpn:..
```

Le script d'installation vous pose deux questions :

- la première est relative à la création du "device" virtuel nécessaire pour TUN, répondez "yes",
- la seconde n'a d'intérêt que si vous faites une mise à jour d'OpenVPN, à travers un tunnel OpenVPN. A mon avis, il vaut mieux, chaque fois que c'est possible, éviter de se mettre dans des situations aussi hasardeuses. Pour ce genre d'opérations, SSH fera parfaitement l'affaire.

Tant qu'on y est, vérifions la présence de la librairie de compression LZO, qui vas nous permettre d'optimiser le débit du tunnel :

```
aaron:~# dpkg -l | grep lzo
ii liblzol 1.08-1 A real-time data compression library
aaron:~#
```

Elle y est. Sinon, un "apt-get install liblzol" y remédiera.

Comme nous n'avons pour l'instant aucune configuration d'OpenVPN, bien que l'installation ait indiqué :

```
Starting openvpn:..
```

Rien n'a démarré. Pour l'instant, nous faisons des choses simples, nous allons monter un tunnel "à la main", juste pour voir.

OpenVPN travaille en "client/serveur". Autrement dit, l'une des extrémités sera en écoute permanente l'autre établira le tunnel au moment du besoin.

Démarrage du serveur

Sur AARON, qui dispose d'une adresse IP fixe, nous démarrons le serveur :

```
aaron:~# openvpn --port 8147 --dev tun1 --ifconfig 192.168.25.1 192.168.25.2 \  
--comp-lzo --verb 5
```

Quelques mots d'explication sur cette ligne de commande :

--port 8147, c'est le port qui sera utilisé pour supporter le tunnel,

--dev tun1, l'interface réseau virtuelle qui constitue en quelque sorte le bout du tunnel côté serveur,

--ifconfig 192.168.25.1 192.168.25.2, va permettre d'attribuer les adresses IP à chaque bout du tunnel :

192.168.25.1 côté local,

192.168.25.2 côté distant

--comp-lzo pour indiquer que l'on utilise la compression en temps réel LZO

--verb 5, c'est le niveau de bavardage que l'on souhaite pour OpenVPN. Le niveau 5 est relativement bavard, comme l'indique la suite :

```
Sat Jun 26 14:58:37 2004 0[0]: Current Parameter Settings:  
Sat Jun 26 14:58:37 2004 1[0]: config = '[UNDEF]'  
Sat Jun 26 14:58:37 2004 2[0]: persist_config = DISABLED  
Sat Jun 26 14:58:37 2004 3[0]: persist_mode = 1  
Sat Jun 26 14:58:37 2004 4[0]: show_ciphers = DISABLED  
Sat Jun 26 14:58:37 2004 5[0]: show_digests = DISABLED  
Sat Jun 26 14:58:37 2004 6[0]: genkey = DISABLED  
Sat Jun 26 14:58:37 2004 7[0]: askpass = DISABLED  
Sat Jun 26 14:58:37 2004 8[0]: show_tls_ciphers = DISABLED  
Sat Jun 26 14:58:37 2004 9[0]: proto = 0  
Sat Jun 26 14:58:37 2004 10[0]: local = '[UNDEF]'  
Sat Jun 26 14:58:37 2004 11[0]: remote = '[UNDEF]'  
Sat Jun 26 14:58:37 2004 12[0]: local_port = 8147  
Sat Jun 26 14:58:37 2004 13[0]: remote_port = 8147  
Sat Jun 26 14:58:37 2004 14[0]: remote_float = DISABLED  
Sat Jun 26 14:58:37 2004 15[0]: ipchange = '[UNDEF]'  
Sat Jun 26 14:58:37 2004 16[0]: bind_local = ENABLED  
Sat Jun 26 14:58:37 2004 17[0]: dev = 'tun1'  
Sat Jun 26 14:58:37 2004 18[0]: dev_type = '[UNDEF]'  
Sat Jun 26 14:58:37 2004 19[0]: dev_node = '[UNDEF]'  
Sat Jun 26 14:58:37 2004 20[0]: tun_ipv6 = DISABLED  
Sat Jun 26 14:58:37 2004 21[0]: ifconfig_local = '192.168.25.1'  
Sat Jun 26 14:58:37 2004 22[0]: ifconfig_remote_netmask = '192.168.25.2'  
Sat Jun 26 14:58:37 2004 23[0]: ifconfig_noexec = DISABLED  
Sat Jun 26 14:58:37 2004 24[0]: ifconfig_nowarn = DISABLED  
Sat Jun 26 14:58:37 2004 25[0]: shaper = 0  
Sat Jun 26 14:58:37 2004 26[0]: tun_mtu = 1300  
Sat Jun 26 14:58:37 2004 27[0]: tun_mtu_defined = DISABLED  
Sat Jun 26 14:58:37 2004 28[0]: link_mtu = 1300  
Sat Jun 26 14:58:37 2004 29[0]: link_mtu_defined = ENABLED  
Sat Jun 26 14:58:37 2004 30[0]: tun_mtu_extra = 0  
Sat Jun 26 14:58:37 2004 31[0]: tun_mtu_extra_defined = DISABLED  
Sat Jun 26 14:58:37 2004 32[0]: fragment = 0  
Sat Jun 26 14:58:37 2004 33[0]: mtu_discover_type = -1  
Sat Jun 26 14:58:37 2004 34[0]: mtu_test = 0
```

```
Sat Jun 26 14:58:37 2004 35[0]: mlock = DISABLED
Sat Jun 26 14:58:37 2004 36[0]: inactivity_timeout = 0
Sat Jun 26 14:58:37 2004 37[0]: ping_send_timeout = 0
Sat Jun 26 14:58:37 2004 38[0]: ping_rec_timeout = 0
Sat Jun 26 14:58:37 2004 39[0]: ping_rec_timeout_action = 0
Sat Jun 26 14:58:37 2004 40[0]: ping_timer_remote = DISABLED
Sat Jun 26 14:58:37 2004 41[0]: persist_tun = DISABLED
Sat Jun 26 14:58:37 2004 42[0]: persist_local_ip = DISABLED
Sat Jun 26 14:58:37 2004 43[0]: persist_remote_ip = DISABLED
Sat Jun 26 14:58:37 2004 44[0]: persist_key = DISABLED
Sat Jun 26 14:58:37 2004 45[0]: mssfix_defined = DISABLED
Sat Jun 26 14:58:37 2004 46[0]: mssfix = 0
Sat Jun 26 14:58:37 2004 47[0]: passtos = DISABLED
Sat Jun 26 14:58:37 2004 48[0]: resolve_retry_seconds = 0
Sat Jun 26 14:58:37 2004 49[0]: connect_retry_seconds = 5
Sat Jun 26 14:58:37 2004 50[0]: username = '[UNDEF]'
Sat Jun 26 14:58:37 2004 51[0]: groupname = '[UNDEF]'
Sat Jun 26 14:58:37 2004 52[0]: chroot_dir = '[UNDEF]'
Sat Jun 26 14:58:37 2004 53[0]: cd_dir = '[UNDEF]'
Sat Jun 26 14:58:37 2004 54[0]: writepid = '[UNDEF]'
Sat Jun 26 14:58:37 2004 55[0]: up_script = '[UNDEF]'
Sat Jun 26 14:58:37 2004 56[0]: down_script = '[UNDEF]'
Sat Jun 26 14:58:37 2004 57[0]: up_restart = DISABLED
Sat Jun 26 14:58:37 2004 58[0]: daemon = DISABLED
Sat Jun 26 14:58:37 2004 59[0]: inetd = 0
Sat Jun 26 14:58:37 2004 60[0]: log = DISABLED
Sat Jun 26 14:58:37 2004 61[0]: nice = 0
Sat Jun 26 14:58:37 2004 62[0]: verbosity = 5
Sat Jun 26 14:58:37 2004 63[0]: mute = 0
Sat Jun 26 14:58:37 2004 64[0]: gremlin = DISABLED
Sat Jun 26 14:58:37 2004 65[0]: occ = ENABLED
Sat Jun 26 14:58:37 2004 66[0]: http_proxy_server = '[UNDEF]'
Sat Jun 26 14:58:37 2004 67[0]: http_proxy_port = 0
Sat Jun 26 14:58:37 2004 68[0]: http_proxy_auth_method = '[UNDEF]'
Sat Jun 26 14:58:37 2004 69[0]: http_proxy_auth_file = '[UNDEF]'
Sat Jun 26 14:58:37 2004 70[0]: http_proxy_retry = DISABLED
Sat Jun 26 14:58:37 2004 71[0]: socks_proxy_server = '[UNDEF]'
Sat Jun 26 14:58:37 2004 72[0]: socks_proxy_port = 0
Sat Jun 26 14:58:37 2004 73[0]: socks_proxy_retry = DISABLED
Sat Jun 26 14:58:37 2004 74[0]: comp_lzo = ENABLED
Sat Jun 26 14:58:37 2004 75[0]: comp_lzo_adaptive = ENABLED
Sat Jun 26 14:58:37 2004 76[0]: route_script = '[UNDEF]'
Sat Jun 26 14:58:37 2004 77[0]: route_default_gateway = '[UNDEF]'
Sat Jun 26 14:58:37 2004 78[0]: route_noexec = DISABLED
Sat Jun 26 14:58:37 2004 79[0]: route_delay = 0
Sat Jun 26 14:58:37 2004 80[0]: route_delay_defined = DISABLED
Sat Jun 26 14:58:37 2004 81[0]: shared_secret_file = '[UNDEF]'
Sat Jun 26 14:58:37 2004 82[0]: key_direction = 0
Sat Jun 26 14:58:37 2004 83[0]: ciphername_defined = ENABLED
Sat Jun 26 14:58:37 2004 84[0]: ciphername = 'BF-CBC'
Sat Jun 26 14:58:37 2004 85[0]: authname_defined = ENABLED
Sat Jun 26 14:58:37 2004 86[0]: authname = 'SHA1'
Sat Jun 26 14:58:37 2004 87[0]: keysize = 0
Sat Jun 26 14:58:37 2004 88[0]: replay = ENABLED
Sat Jun 26 14:58:37 2004 89[0]: replay_window = 64
Sat Jun 26 14:58:37 2004 90[0]: replay_time = 15
Sat Jun 26 14:58:37 2004 91[0]: packet_id_file = '[UNDEF]'
Sat Jun 26 14:58:37 2004 92[0]: use_iv = ENABLED
Sat Jun 26 14:58:37 2004 93[0]: test_crypto = DISABLED
Sat Jun 26 14:58:37 2004 94[0]: tls_server = DISABLED
Sat Jun 26 14:58:37 2004 95[0]: tls_client = DISABLED
Sat Jun 26 14:58:37 2004 96[0]: key_method = 1
Sat Jun 26 14:58:37 2004 97[0]: ca_file = '[UNDEF]'
Sat Jun 26 14:58:37 2004 98[0]: dh_file = '[UNDEF]'
Sat Jun 26 14:58:37 2004 99[0]: cert_file = '[UNDEF]'
Sat Jun 26 14:58:37 2004 100[0]: priv_key_file = '[UNDEF]'
Sat Jun 26 14:58:37 2004 101[0]: cipher_list = '[UNDEF]'
Sat Jun 26 14:58:37 2004 102[0]: tls_verify = '[UNDEF]'
Sat Jun 26 14:58:37 2004 103[0]: tls_remote = '[UNDEF]'
Sat Jun 26 14:58:37 2004 104[0]: crl_file = '[UNDEF]'
Sat Jun 26 14:58:37 2004 105[0]: tls_timeout = 2
Sat Jun 26 14:58:37 2004 106[0]: renegotiate_bytes = 0
Sat Jun 26 14:58:37 2004 107[0]: renegotiate_packets = 0
Sat Jun 26 14:58:37 2004 108[0]: renegotiate_seconds = 3600
Sat Jun 26 14:58:37 2004 109[0]: handshake_window = 60
Sat Jun 26 14:58:37 2004 110[0]: transition_window = 3600
Sat Jun 26 14:58:37 2004 111[0]: single_session = DISABLED
```

```

Sat Jun 26 14:58:37 2004 112[0]:  tls_auth_file = '[UNDEF]'
Sat Jun 26 14:58:37 2004 113[0]:  OpenVPN 1.6.0 i386-pc-linux-gnu [SSL] [LZO] [PTHREAD] built on Jun
10 2004
Sat Jun 26 14:58:37 2004 114[0]:  ***** WARNING *****: all encryption and authentication features
disabled
Sat Jun 26 14:58:37 2004 115[0]:  -- all data will be tunnelled as cleartext
Sat Jun 26 14:58:37 2004 116[0]:  LZO compression initialized
Sat Jun 26 14:58:38 2004 117[0]:  TUN/TAP device tun1 opened
Sat Jun 26 14:58:38 2004 118[0]:  /sbin/ifconfig tun1 192.168.25.1 pointopoint 192.168.25.2 mtu 1299
Sat Jun 26 14:58:38 2004 119[0]:  Data Channel MTU parms [ L:1300 D:1300 EF:1 EB:19 ET:0 EL:0 ]
Sat Jun 26 14:58:38 2004 120[0]:  Local Options String: 'V3,dev-type tun,link-mtu 1300,tun-mtu
1299,proto UDPv4,
ifconfig 192.168.25.2 192.168.25.1,comp-lzo'
Sat Jun 26 14:58:38 2004 121[0]:  Expected Remote Options String: 'V3,dev-type tun,link-mtu 1300,tun-
mtu 1299,
proto UDPv4,ifconfig 192.168.25.1 192.168.25.2,comp-lzo'
Sat Jun 26 14:58:38 2004 122[0]:  Local Options hash (VER=V3): '904e90d5'
Sat Jun 26 14:58:38 2004 123[0]:  Expected Remote Options hash (VER=V3): '18cff9b7'
Sat Jun 26 14:58:38 2004 124[0]:  PTHREAD support initialized
Sat Jun 26 14:58:38 2004 125[0]:  UDPv4 link local (bound): [undef]:8147
Sat Jun 26 14:58:38 2004 126[0]:  UDPv4 link remote: [undef]

```

Tout ceci n'a pour but que de montrer que nous sommes loin d'utiliser tous les paramètres proposés par OpenVPN. Le but est tout de même d'arriver le plus rapidement possible à une solution sécurisée, plutôt que d'explorer toutes les ressources d'OpenVPN.

Ce qui est surligné montre les options définies dans le démarrage d'OpenVPN.

Vérifications :

```

aaron:~# ifconfig
...
ppp0      Link encap:Point-to-Point Protocol
          inet addr:82.127.57.95  P-t-P:193.253.160.3  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1492  Metric:1
          RX packets:342756 errors:0 dropped:0 overruns:0 frame:0
          TX packets:290200 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:426707207 (406.9 MiB)  TX bytes:26657415 (25.4 MiB)

tun1     Link encap:Point-to-Point Protocol
          inet addr:192.168.25.1  P-t-P:192.168.25.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1299  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

aaron:~#

```

Nous avons, en plus de ppp0 qui est la connexion à l'internet, une interface tun1 qui apparaît elle aussi comme une liaison point à point entre 192.168.25.1 (local) et 192.168.25.2 (distant)

```

Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
192.168.25.2    0.0.0.0         255.255.255.255  UH      0      0      0 tun1
...
0.0.0.0         193.253.160.3  0.0.0.0        UG      0      0      0 ppp0
aaron:~#

```

et nous avons bien la route vers 192.168.25.2 qui passe par tun1.

Démarrage du client

Sur CYCLOPE, nous allons faire quelque chose de très similaire :

```

cyclope:~# openvpn --remote 82.127.57.95 --port 8147 --dev tun1 --ifconfig 192.168.25.2 192.168.25.1
\
--comp-lzo --verb 5

```

Notez qu'ici, comme nous sommes client, nous indiquons en plus l'adresse IP distante qui supporte le tunnel (--remote 82.127.57.95).

```
Sat Jun 26 15:22:47 2004 0[0]: Current Parameter Settings:
Sat Jun 26 15:22:47 2004 1[0]: config = '[UNDEF]'
Sat Jun 26 15:22:47 2004 2[0]: persist_config = DISABLED
Sat Jun 26 15:22:47 2004 3[0]: persist_mode = 1
Sat Jun 26 15:22:47 2004 4[0]: show_ciphers = DISABLED
Sat Jun 26 15:22:47 2004 5[0]: show_digests = DISABLED
Sat Jun 26 15:22:47 2004 6[0]: genkey = DISABLED
Sat Jun 26 15:22:47 2004 7[0]: askpass = DISABLED
Sat Jun 26 15:22:47 2004 8[0]: show_tls_ciphers = DISABLED
Sat Jun 26 15:22:47 2004 9[0]: proto = 0
Sat Jun 26 15:22:47 2004 10[0]: local = '[UNDEF]'
Sat Jun 26 15:22:47 2004 11[0]: remote = '82.127.57.95'
Sat Jun 26 15:22:47 2004 12[0]: local_port = 8147
Sat Jun 26 15:22:47 2004 13[0]: remote_port = 8147
Sat Jun 26 15:22:47 2004 14[0]: remote_float = DISABLED
Sat Jun 26 15:22:47 2004 15[0]: ipchange = '[UNDEF]'
Sat Jun 26 15:22:47 2004 16[0]: bind_local = ENABLED
Sat Jun 26 15:22:47 2004 17[0]: dev = 'tun1'
Sat Jun 26 15:22:47 2004 18[0]: dev_type = '[UNDEF]'
Sat Jun 26 15:22:47 2004 19[0]: dev_node = '[UNDEF]'
Sat Jun 26 15:22:47 2004 20[0]: tun_ipv6 = DISABLED
Sat Jun 26 15:22:47 2004 21[0]: ifconfig_local = '192.168.25.2'
Sat Jun 26 15:22:47 2004 22[0]: ifconfig_remote_netmask = '192.168.25.1'
Sat Jun 26 15:22:47 2004 23[0]: ifconfig_noexec = DISABLED
Sat Jun 26 15:22:47 2004 24[0]: ifconfig_nowarn = DISABLED
Sat Jun 26 15:22:47 2004 25[0]: shaper = 0
Sat Jun 26 15:22:47 2004 26[0]: tun_mtu = 1300
Sat Jun 26 15:22:47 2004 27[0]: tun_mtu_defined = DISABLED
Sat Jun 26 15:22:47 2004 28[0]: link_mtu = 1300
Sat Jun 26 15:22:47 2004 29[0]: link_mtu_defined = ENABLED
Sat Jun 26 15:22:47 2004 30[0]: tun_mtu_extra = 0
Sat Jun 26 15:22:47 2004 31[0]: tun_mtu_extra_defined = DISABLED
Sat Jun 26 15:22:47 2004 32[0]: fragment = 0
Sat Jun 26 15:22:47 2004 33[0]: mtu_discover_type = -1
Sat Jun 26 15:22:47 2004 34[0]: mtu_test = 0
Sat Jun 26 15:22:47 2004 35[0]: mlock = DISABLED
Sat Jun 26 15:22:47 2004 36[0]: inactivity_timeout = 0
Sat Jun 26 15:22:47 2004 37[0]: ping_send_timeout = 0
Sat Jun 26 15:22:47 2004 38[0]: ping_rec_timeout = 0
Sat Jun 26 15:22:47 2004 39[0]: ping_rec_timeout_action = 0
Sat Jun 26 15:22:47 2004 40[0]: ping_timer_remote = DISABLED
Sat Jun 26 15:22:47 2004 41[0]: persist_tun = DISABLED
Sat Jun 26 15:22:47 2004 42[0]: persist_local_ip = DISABLED
Sat Jun 26 15:22:47 2004 43[0]: persist_remote_ip = DISABLED
Sat Jun 26 15:22:47 2004 44[0]: persist_key = DISABLED
Sat Jun 26 15:22:47 2004 45[0]: mssfix_defined = DISABLED
Sat Jun 26 15:22:47 2004 46[0]: mssfix = 0
Sat Jun 26 15:22:47 2004 47[0]: passtos = DISABLED
Sat Jun 26 15:22:47 2004 48[0]: resolve_retry_seconds = 0
Sat Jun 26 15:22:47 2004 49[0]: connect_retry_seconds = 5
Sat Jun 26 15:22:47 2004 50[0]: username = '[UNDEF]'
Sat Jun 26 15:22:47 2004 51[0]: groupname = '[UNDEF]'
Sat Jun 26 15:22:47 2004 52[0]: chroot_dir = '[UNDEF]'
Sat Jun 26 15:22:47 2004 53[0]: cd_dir = '[UNDEF]'
Sat Jun 26 15:22:47 2004 54[0]: writepid = '[UNDEF]'
Sat Jun 26 15:22:47 2004 55[0]: up_script = '[UNDEF]'
Sat Jun 26 15:22:47 2004 56[0]: down_script = '[UNDEF]'
Sat Jun 26 15:22:47 2004 57[0]: up_restart = DISABLED
Sat Jun 26 15:22:47 2004 58[0]: daemon = DISABLED
Sat Jun 26 15:22:47 2004 59[0]: inetd = 0
Sat Jun 26 15:22:47 2004 60[0]: log = DISABLED
Sat Jun 26 15:22:47 2004 61[0]: nice = 0
Sat Jun 26 15:22:47 2004 62[0]: verbosity = 5
Sat Jun 26 15:22:47 2004 63[0]: mute = 0
Sat Jun 26 15:22:47 2004 64[0]: gremlin = DISABLED
Sat Jun 26 15:22:47 2004 65[0]: occ = ENABLED
Sat Jun 26 15:22:47 2004 66[0]: http_proxy_server = '[UNDEF]'
Sat Jun 26 15:22:47 2004 67[0]: http_proxy_port = 0
Sat Jun 26 15:22:47 2004 68[0]: http_proxy_auth_method = '[UNDEF]'
Sat Jun 26 15:22:47 2004 69[0]: http_proxy_auth_file = '[UNDEF]'
Sat Jun 26 15:22:47 2004 70[0]: http_proxy_retry = DISABLED
Sat Jun 26 15:22:47 2004 71[0]: socks_proxy_server = '[UNDEF]'
Sat Jun 26 15:22:47 2004 72[0]: socks_proxy_port = 0
```

```

Sat Jun 26 15:22:47 2004 73[0]: socks_proxy_retry = DISABLED
Sat Jun 26 15:22:47 2004 74[0]: comp_lzo = ENABLED
Sat Jun 26 15:22:47 2004 75[0]: comp_lzo_adaptive = ENABLED
Sat Jun 26 15:22:47 2004 76[0]: route_script = '[UNDEF]'
Sat Jun 26 15:22:47 2004 77[0]: route_default_gateway = '[UNDEF]'
Sat Jun 26 15:22:47 2004 78[0]: route_noexec = DISABLED
Sat Jun 26 15:22:47 2004 79[0]: route_delay = 0
Sat Jun 26 15:22:47 2004 80[0]: route_delay_defined = DISABLED
Sat Jun 26 15:22:47 2004 81[0]: shared_secret_file = '[UNDEF]'
Sat Jun 26 15:22:47 2004 82[0]: key_direction = 0
Sat Jun 26 15:22:47 2004 83[0]: ciphertype_defined = ENABLED
Sat Jun 26 15:22:47 2004 84[0]: ciphertype = 'BF-CBC'
Sat Jun 26 15:22:47 2004 85[0]: authname_defined = ENABLED
Sat Jun 26 15:22:47 2004 86[0]: authname = 'SHA1'
Sat Jun 26 15:22:47 2004 87[0]: keysize = 0
Sat Jun 26 15:22:47 2004 88[0]: replay = ENABLED
Sat Jun 26 15:22:47 2004 89[0]: replay_window = 64
Sat Jun 26 15:22:47 2004 90[0]: replay_time = 15
Sat Jun 26 15:22:47 2004 91[0]: packet_id_file = '[UNDEF]'
Sat Jun 26 15:22:47 2004 92[0]: use_iv = ENABLED
Sat Jun 26 15:22:47 2004 93[0]: test_crypto = DISABLED
Sat Jun 26 15:22:47 2004 94[0]: tls_server = DISABLED
Sat Jun 26 15:22:47 2004 95[0]: tls_client = DISABLED
Sat Jun 26 15:22:47 2004 96[0]: key_method = 1
Sat Jun 26 15:22:47 2004 97[0]: ca_file = '[UNDEF]'
Sat Jun 26 15:22:47 2004 98[0]: dh_file = '[UNDEF]'
Sat Jun 26 15:22:47 2004 99[0]: cert_file = '[UNDEF]'
Sat Jun 26 15:22:47 2004 100[0]: priv_key_file = '[UNDEF]'
Sat Jun 26 15:22:47 2004 101[0]: cipher_list = '[UNDEF]'
Sat Jun 26 15:22:47 2004 102[0]: tls_verify = '[UNDEF]'
Sat Jun 26 15:22:47 2004 103[0]: tls_remote = '[UNDEF]'
Sat Jun 26 15:22:47 2004 104[0]: crl_file = '[UNDEF]'
Sat Jun 26 15:22:47 2004 105[0]: tls_timeout = 2
Sat Jun 26 15:22:47 2004 106[0]: renegotiate_bytes = 0
Sat Jun 26 15:22:47 2004 107[0]: renegotiate_packets = 0
Sat Jun 26 15:22:47 2004 108[0]: renegotiate_seconds = 3600
Sat Jun 26 15:22:47 2004 109[0]: handshake_window = 60
Sat Jun 26 15:22:47 2004 110[0]: transition_window = 3600
Sat Jun 26 15:22:47 2004 111[0]: single_session = DISABLED
Sat Jun 26 15:22:47 2004 112[0]: tls_auth_file = '[UNDEF]'
Sat Jun 26 15:22:47 2004 113[0]: OpenVPN 1.6.0 i386-pc-linux-gnu [SSL] [LZO] [PTHREAD] built on Jun
10 2004
Sat Jun 26 15:22:47 2004 114[0]: ***** WARNING *****: all encryption and authentication features
disabled
-- all data will be tunneled as cleartext
Sat Jun 26 15:22:47 2004 115[0]: LZO compression initialized
Sat Jun 26 15:22:47 2004 116[0]: TUN/TAP device tun1 opened
Sat Jun 26 15:22:47 2004 117[0]: /sbin/ifconfig tun1 192.168.25.2 pointopoint 192.168.25.1 mtu 1299
Sat Jun 26 15:22:47 2004 118[0]: Data Channel MTU parms [ L:1300 D:1300 EF:1 EB:19 ET:0 EL:0 ]
Sat Jun 26 15:22:47 2004 119[0]: Local Options String: 'V3,dev-type tun,link-mtu 1300,tun-mtu
1299,proto UDPv4,
ifconfig 192.168.25.1 192.168.25.2,comp-lzo'
Sat Jun 26 15:22:47 2004 120[0]: Expected Remote Options String: 'V3,dev-type tun,link-mtu 1300,tun-
mtu 1299,
proto UDPv4, ifconfig 192.168.25.2 192.168.25.1,comp-lzo'
Sat Jun 26 15:22:47 2004 121[0]: Local Options hash (VER=V3): '18cff9b7'
Sat Jun 26 15:22:47 2004 122[0]: Expected Remote Options hash (VER=V3): '904e90d5'
Sat Jun 26 15:22:47 2004 123[0]: PTHREAD support initialized
Sat Jun 26 15:22:47 2004 124[0]: UDPv4 link local (bound): [undef]:8147
Sat Jun 26 15:22:47 2004 125[0]: UDPv4 link remote: 82.127.57.95:8147

```

Vérification des interfaces virtuelles :

```

cyclope:~# ifconfig
...
ppp0      Link encap:Point-to-Point Protocol
          inet addr:80.8.135.67 P-t-P:80.8.128.1 Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1492 Metric:1
          RX packets:5197 errors:0 dropped:0 overruns:0 frame:0
          TX packets:133 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:295907 (288.9 KiB) TX bytes:9499 (9.2 KiB)

tun1     Link encap:Point-to-Point Protocol
          inet addr:192.168.25.2 P-t-P:192.168.25.1 Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1299 Metric:1

```

```
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:10
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)
```

Vérification des routes :

```
cyclope:~# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.25.1 0.0.0.0 255.255.255.255 UH 0 0 0 tun1
...
0.0.0.0 80.8.128.1 0.0.0.0 UG 0 0 0 ppp0
```

Contrôle du tunnel

Depuis CYCLOPE (192.168.25.2), un petit ping sur AARON (192.168.25.1) :

```
cyclope:~# ping -c 4 192.168.25.1
PING 192.168.25.1 (192.168.25.1): 56 data bytes

--- 192.168.25.1 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
cyclope:~#
```

Ah ! Ça ne fonctionne pas...

Et c'est bon signe !

Si ça fonctionnait, ça voudrait dire que les deux machines sont connectées à l'internet sans firewall, ce qui serait très **mal!**

Réfléchissons. Nous avons sur les deux hôtes des règles IPTables du genre :

```
iptables -P INPUT DROP
iptables -A INPUT -i ppp0 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Donc, les paquets "NEW" n'entrent pas, c'est normal. Ajoutons ceci de chaque côté :

```
iptables -A INPUT -i ppp0 -p UDP --dport 8147 -j ACCEPT
```

Rappelons-nous en effet qu'OpenVPN utilise UDP et que nous avons établi le tunnel sur le port 8147.

Deuxième essai :

```
cyclope:~# ping -c 4 192.168.25.1
PING 192.168.25.1 (192.168.25.1): 56 data bytes

--- 192.168.25.1 ping statistics ---
4 packets transmitted, 0 packets received, 100% packet loss
cyclope:~#
```

Ça, c'est ce qui arrive quand on ne réfléchit pas assez... On a dit quelque chose au firewall, à propos de tun1 ? Non ? Alors, c'est normal que ça ne fonctionne toujours pas (iptables -P INPUT DROP).

```
iptables -A INPUT -i tun1 -j ACCEPT
iptables -A OUTPUT -o tun1 -j ACCEPT
```

Ceci afin d'éviter les ennuis, mais par la suite, ce sera peut-être une bonne chose d'affiner un peu plus ces règles de filtrage.

Troisième essai :

```
cyclope:~# ping -c 4 192.168.25.1
PING 192.168.25.1 (192.168.25.1): 56 data bytes
64 bytes from 192.168.25.1: icmp_seq=0 ttl=64 time=89.0 ms
64 bytes from 192.168.25.1: icmp_seq=1 ttl=64 time=65.3 ms
64 bytes from 192.168.25.1: icmp_seq=2 ttl=64 time=71.4 ms
64 bytes from 192.168.25.1: icmp_seq=3 ttl=64 time=74.9 ms

--- 192.168.25.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 65.3/75.1/89.0 ms
cyclope:~#
```

Bon. On y est arrivé, et tout ça pour pas grand chose, à part que l'on a vérifié que le tunnel fonctionne.

Attention tout de même que ça pourrait encore ne pas fonctionner, en fonction des règles en vigueur sur FORWARD.

Mais réfléchissons encore un peu...

Lorsque nous avons établi le tunnel, en lançant OpenVPN de chaque côté, nous n'avons rien établi du tout, puisque les firewalls ne laissaient pas passer. Pourtant, ça a fonctionné quand même, après modification des règles, ce qui prouve qu'OpenVPN est très efficace sur des liaisons difficiles.

Un petit coup de sniffeur

Nous sommes sur CYCLOPE. On sniffe le ping sur tun1 :

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.25.2	192.168.25.1	ICMP	Echo (ping) request
2	0.077503	192.168.25.1	192.168.25.2	ICMP	Echo (ping) reply
3	1.007802	192.168.25.2	192.168.25.1	ICMP	Echo (ping) request
4	1.095914	192.168.25.1	192.168.25.2	ICMP	Echo (ping) reply
5	2.018634	192.168.25.2	192.168.25.1	ICMP	Echo (ping) request
6	2.083968	192.168.25.1	192.168.25.2	ICMP	Echo (ping) reply
7	3.019537	192.168.25.2	192.168.25.1	ICMP	Echo (ping) request
8	3.087613	192.168.25.1	192.168.25.2	ICMP	Echo (ping) reply

Pas besoin d'entrer dans les détails, nous voyons bien ICMP qui circule entre 192.168.25.1 et 192.168.25.2.

Puis on le resniffe sur ppp0 :

No.	Time	Source	Destination	Protocol	Info
1	0.000000	80.8.135.67	82.127.57.95	UDP	Source port: 8147 Destination port: 8147
2	0.067128	82.127.57.95	80.8.135.67	UDP	Source port: 8147 Destination port: 8147
3	1.011132	80.8.135.67	82.127.57.95	UDP	Source port: 8147 Destination port: 8147
4	1.074716	82.127.57.95	80.8.135.67	UDP	Source port: 8147 Destination port: 8147
5	2.027369	80.8.135.67	82.127.57.95	UDP	Source port: 8147 Destination port: 8147
6	2.096456	82.127.57.95	80.8.135.67	UDP	Source port: 8147 Destination port: 8147
7	3.041653	80.8.135.67	82.127.57.95	UDP	Source port: 8147 Destination port: 8147
8	3.105374	82.127.57.95	80.8.135.67	UDP	Source port: 8147 Destination port: 8147

A ce niveau, nous ne voyons que de l'UDP, bien sûr. Si nous regardons en détail l'une des trames :

```
Frame 1 (129 bytes on wire, 129 bytes captured)
  Arrival Time: Jun 26, 2004 16:22:50.261813000
  Time delta from previous packet: 0.000000000 seconds
  Time since reference or first frame: 0.000000000 seconds
  Frame Number: 1
  Packet Length: 129 bytes
  Capture Length: 129 bytes
Linux cooked capture
  Packet type: Sent by us (4)
  Link-layer address type: 512
```

```

Link-layer address length: 0
Source: <MISSING>
Protocol: IP (0x0800)
Internet Protocol, Src Addr: 80.8.135.67, Dst Addr: 82.127.57.95
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  0000 00.. = Differentiated Services Codepoint: Default (0x00)
  .... ..0. = ECN-Capable Transport (ECT): 0
  .... ..0 = ECN-CE: 0
Total Length: 113
Identification: 0x0200 (512)
Flags: 0x04 (Don't Fragment)
  0... = Reserved bit: Not set
  .1.. = Don't fragment: Set
  ..0. = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: UDP (0x11)
Header checksum: 0xd552 (correct)
Source: 80.8.135.67 (80.8.135.67)
Destination: 82.127.57.95 (82.127.57.95)
User Datagram Protocol, Src Port: 8147 (8147), Dst Port: 8147 (8147)
Source port: 8147 (8147)
Destination port: 8147 (8147)
Length: 93
Checksum: 0x6263 (correct)
Data (85 bytes)
0000  fa 45 00 00 54 00 00 40 00 40 01 87 55 c0 a8 19  .E..T..@..U...
0010  02 c0 a8 19 01 08 00 5c 4c ee 0a 00 00 40 dd 86  .....L....@..
0020  ba 00 03 fb 0a 08 09 0a 0b 0c 0d 0e 0f 10 11 12  .....
0030  13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22  .....!"
0040  23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32  #S%&'()*+,-./012
0050  33 34 35 36 37 34567

```

Et que nous savons décoder les données transportées, nous trouverons le paquet ICMP compressé par LZO. Un simple sniff ne suffira déjà pas à lire simplement les données qui circulent.

Premières conclusions.

Nous avons réussi à monter un tunnel tout simple, qui relie point à point deux hôtes distants, tous deux connectés à l'internet.

A l'intérieur de ce tunnel, tout se passe comme si les deux hôtes étaient reliés par une liaison série, comme par exemple avec PPP :

```

tun1 Link encap:Point-to-Point Protocol
inet addr:192.168.25.2 P-t-P:192.168.25.1 Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1258 Metric:1
RX packets:146939 errors:0 dropped:0 overruns:0 frame:0
TX packets:115942 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:10
RX bytes:22750260 (21.6 MiB) TX bytes:12818254 (12.2 MiB)

```

Nous n'avons pas réuni deux réseaux, juste deux machines. Mais si ces machines sont des routeurs, en réfléchissant (encore) un peu, nous trouverons bien des règles de routages intelligentes qui permettront aux réseaux qui sont derrière ces routeurs de communiquer entre eux.

Il n'y a pas d'authentification, il n'y a pas de confidentialité, il y a juste une compression des données.

Bien sûr, nous allons faire mieux, en mettant en oeuvre SSL.

OpenSSL

Un petit peu de sadisme

Il existe avec OpenVPN une solution extrêmement simple pour sécuriser un tunnel, en utilisant une clé symétrique (et statique) pré-partagée.

Il suffit d'utiliser la commande :

```
cyclope:~# openvpn --genkey --secret shared.key
```

Vous obtenez ainsi un fichier ASCII nommé shared.key, qui a cette allure :

```
cyclope:~# cat shared.key
#
# 2048 bit OpenVPN static key
#
-----BEGIN OpenVPN Static key V1-----
37e871e611c48178a684b1aac3c42d1a
4682db3c5703056643382b6ae43da0d4
afe7ada90f2506d291cc26848afb05e7
1f1ec55y8b2df1d2b73ey8842baf5a23
edc6d3b28f017c8c78d545e1cd5b40af
f9a76d537941c6e5ab4eabc866160511
10f7774686cye882b603673288b56713
a48a7859959cea87b87273019751b867
c0b5dae293d5259a2170f36ecf19d94c
9c5c39a7fa7df03827bc0fa3b140eb76
129f5d7f8f14f7bea2cbyf13cfc2ef4b
752426d77ab677065bcd8af84878b477
c47aef5628aaa8bb6e706e93252ey576
5aae05bc0f58626811d020f8f6106c97
cf69114ab0504473yc2adcdd9b1008e6
315d3073c6d5e2feef24e138af86e397
-----END OpenVPN Static key V1-----
```

Il vous suffit alors de transporter une copie de cette clé à l'autre bout de tunnel par un moyen sûr ("scp", par exemple, qui utilise lui-même un système cryptographique, le même que SSH), et de bâtir le tunnel de manière à ce qu'il utilise cette clé pour le chiffrement des données.

C'est simple, ça fonctionne et comme c'est vous qui avez généré la clé et qui l'avez transportée à l'autre bout, vous êtes par le fait sûr de l'authenticité.

Oui mais voilà, c'est trop simple. Nous ne ferons donc pas comme ça, nous utiliserons plutôt un système de certificats avec SSL.

La stratégie

Nous allons créer une autorité de certification rien que pour nous, sur AARON. Cette autorité permettra de gérer les clés et certificats nécessaires aux clients qui utiliseront le tunnel.

Pour l'instant, tout se passe donc sur AARON. Nous allons utiliser openssl comme une recette de cuisine, pour ne pas trop compliquer cet exposé.

D'abord, il faut s'assurer que la librairie est installée :

```
aaron:~# dpkg -l openssl
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
```

```

||| Name                Version                Description
+++-----
ii openssl              0.9.7d-3              Secure Socket Layer (SSL) binary and related cryptog
aaron:~#

```

Elle y est. Sinon :

```
aaron:~# apt-get install openssl.
```

Un certificat autosigné

Comme AARON sera l'autorité, il va se signer lui-même son certificat. Théoriquement, ça n'a pas de valeur, mais ici, c'est nous le tiers de confiance et nous savons qui nous sommes.

```
aaron:~# openssl req -nodes -new -x509 -keyout aaron-ca.key -out aaron-ca.crt
```

- aaron-ca.key sera la clé privée de l'autorité de certification, celle qui servira à contresigner les clés publiques des demandeurs de certificats,
- aaron-ca.crt sera le certificat de l'autorité, celui qui contient la clé publique de l'autorité de certification. Ce certificat sera construit selon la norme x509.

```

Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'aaron-ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:France
Locality Name (eg, city) []:Marseille
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MyOrg
Organizational Unit Name (eg, section) []:admin
Common Name (eg, YOUR name) []:Supervisor
Email Address []:supervisor@yahoo.fr
aaron:~#

```

A l'issue de cette opération, nous retrouvons bien aaron-ca.key et aaron-ca.crt dans notre répertoire courant.

Répetons-le encore une fois, parce que ce n'est pas si simple à bien comprendre :

- aaron-ca.key est une clé privée qui va servir à certifier les clés publiques qui seront confiées à AARON, autorité de certification,
- aaron-ca.crt est le certificat de l'autorité. Il contient la clé publique qui permet à qui en a besoin d'authentifier les certificats qui seront émis par AARON.

Un jeu de clés

Maintenant, il faut se construire une clé privée et une demande de certificat pour AARON :

```
aaron:~# openssl req -nodes -new -keyout aaron.key -out aaron.csr
```

aaron.key sera la clé privée et aaron.csr sera en fait un certificat à faire contresigner par l'autorité, par AARON lui-même, donc.

```

Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'aaron.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:France
Locality Name (eg, city) []:Marseille
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MyOrg
Organizational Unit Name (eg, section) []:admin
Common Name (eg, YOUR name) []:Supervisor
Email Address []:supervisor@yahoo.fr

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:lageducapitaine
An optional company name []:
aaron:~#

```

Voilà. Nous disposons maintenant d'une clé privée (aaron.key) et d'un candidat à la certification (aaron.csr) pour AARON.

aaron.csr n'est pas encore un certificat, il n'est qu'une demande de certification qui devra être approuvée par l'autorité (c'est à dire par AARON lui même, dans ce cas précis).

AARON contresigne

AARON, en bon maître absolu, va maintenant approuver son propre certificat. Seulement ici, les recettes de cuisine ne vont pas fonctionner sans précautions préalables. Il nous faut donc mettre un peu les mains dans le cambouis d'openssl.

openssl s'appuie sur un fichier de configuration qui, dans la Debian, se trouve sous /etc/ssl/openssl.cnf. Il y a dans ce fichier une foule de paramètres, qui seront pris par défaut pour diverses opérations si ces paramètres ne sont pas outrepassés par la ligne de commande. Il y a tellement de paramètres qu'il n'est pas humainement possible de se passer de ce fichier de configuration.

Dans l'opération qui va suivre, nous aurons besoin de vérifier cette configuration par défaut et d'adapter un peu notre système en fonction.

La partie utile pour la suite est la suivante :

```

[ CA_default ]
dir                = ./demoCA                # Where everything is kept
certs              = $dir/certs              # Where the issued certs are kept
crl_dir            = $dir/crl                # Where the issued crl are kept
database           = $dir/index.txt         # database index file.
#unique_subject    = no                     # Set to 'no' to allow creation of
#                  # several certificates with same subject.
new_certs_dir      = $dir/newcerts          # default place for new certs.

certificate        = $dir/cacert.pem        # The CA certificate
serial             = $dir/serial            # The current serial number
#crlnumber         = $dir/crlnumber         # the current crl number
#                  # must be commented out to leave a V1 CRL
...

```

Nous pourrions modifier la configuration par défaut, mais autant s'y conformer, quitte à faire les

choses plus proprement par la suite.

Continuons les recettes :

```
aaron:~# mkdir demoCA
cd demoCA
aaron:~/demoCA# touch index.txt
aaron:~/demoCA# mkdir newcerts
aaron:~/demoCA# echo "01" > serial
```

Je vous laisse comprendre ce que nous venons de faire. Passons maintenant à la signature :

```
aaron:~# openssl ca -cert aaron-ca.crt -keyfile aaron-ca.key -out aaron.crt -in aaron.csr
```

au moyen de son certificat d'autorité (aaron-ca.crt) et de sa clé privée d'autorité (aaron-ca.key), nous allons approuver la demande de certification aaron.csr, obtenant ainsi un certificat approuvé aaron.crt

```
Using configuration from /usr/lib/ssl/openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 1 (0x1)
  Validity
    Not Before: Jun 27 11:15:37 2004 GMT
    Not After : Jun 27 11:15:37 2005 GMT
  Subject:
    countryName           = FR
    stateOrProvinceName  = France
    organizationName      = MyOrg
    organizationalUnitName = admin
    commonName            = Supervisor
    emailAddress          = supervisor@yahoo.fr
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
    Netscape Comment:
      OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
      C9:97:59:5A:B1:0D:6C:61:CC:D1:90:79:43:D3:F4:3E:0F:07:38:4C
    X509v3 Authority Key Identifier:
      keyid:AC:F2:A8:DD:56:10:80:AC:C3:F5:6B:F6:29:72:B7:D2:F3:BF:10:86
      DirName:/C=FR/ST=France/L=Marseille/O=MyOrg/OU=admin/CN=Supervisor/emailAddress=supervisor@yahoo.fr
      serial:00

Certificate is to be certified until Jun 27 11:15:37 2005 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]:y
Write out database with 1 new entries
Data Base Updated
aaron:~#
```

Et voilà. Par curiosité, allons voir dans ./demoCA :

```
aaron:~/demoCA# ls -l
total 17
-rw-r--r--  1 root    root          119 Jun 27 13:15 index.txt
-rw-r--r--  1 root    root           21 Jun 27 13:15 index.txt.attr
-rw-r--r--  1 root    root           0 Jun 27 13:15 index.txt.old
drwxr-xr-x  2 root    root           72 Jun 27 13:15 newcerts
-rw-r--r--  1 root    root           3 Jun 27 13:15 serial
-rw-r--r--  1 root    root           3 Jun 27 13:15 serial.old
```

il y a des choses en plus. Voyons tout ça :

```
aaron:~/demoCA# cat index.txt
V          050627111537Z          01          unknown /C=FR/ST=France/O=MyOrg/OU=admin/CN=Supervisor
                                          /emailAddress=supervisor@yahoo.fr
```

```
aaron:~/demoCA# cat index.txt.attr
unique_subject = yes

aaron:~/demoCA# cat serial
02
```

et dans newcerts :

```
aaron:~/demoCA# cd newcerts
aaron:~/demoCA/newcerts# ls -l
total 4
-rw-r--r--  1 root    root          3668 Jun 27 13:15 01.pem
```

Voyons ce qu'il y a de beau dans ce 01.pem :

```
aaron:~/demoCA/newcerts# cat 01.pem
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=FR, ST=France, L=Marseille, O=MyOrg, OU=admin,
CN=Supervisor/emailAddress=supervisor@yahoo.fr
  Validity
    Not Before: Jun 27 11:15:37 2004 GMT
    Not After : Jun 27 11:15:37 2005 GMT
  Subject: C=FR, ST=France, O=MyOrg, OU=admin, CN=Supervisor/emailAddress=supervisor@yahoo.fr
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
      Modulus (1024 bit):
        00:a4:46:38:f2:cd:f3:83:15:a6:2b:7c:41:bd:13:
        d7:8a:71:7d:66:bb:25:81:26:41:59:7f:9f:ee:94:
        f9:9b:fc:fc:4e:88:c4:04:1b:b7:c8:4a:cb:f9:e2:
        73:4a:bf:e4:d7:0b:48:fe:50:33:43:03:8b:62:91:
        b2:3f:33:4f:99:60:b3:e6:b4:f5:d7:e6:37:be:e2:
        90:16:42:e1:35:f2:19:6c:09:55:e4:c9:8c:a7:9d:
        89:99:55:a0:70:9e:53:ff:c0:f4:97:25:f7:5c:b1:
        0e:e0:30:8a:05:81:fa:d8:1a:5b:30:d2:5a:b0:54:
        57:f0:f2:61:a1:fc:c3:ec:57
      Exponent: 65537 (0x10001)
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
    Netscape Comment:
      OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
      C9:97:59:5A:B1:0D:6C:61:CC:D1:90:79:43:D3:F4:3E:0F:07:38:4C
    X509v3 Authority Key Identifier:
      keyid:AC:F2:A8:DD:56:10:80:AC:C3:F5:6B:F6:29:72:B7:D2:F3:BF:10:86
      DirName:/C=FR/ST=France/L=Marseille/O=MyOrg/OU=admin/CN=Supervisor/emailAddress=supervisor@yahoo.fr
      serial:00

  Signature Algorithm: md5WithRSAEncryption
    34:2d:bb:5e:cc:eb:c9:77:46:f0:b3:c2:93:9b:95:bc:a3:e1:
    a2:9a:a7:f4:ac:9b:cc:8f:76:cd:54:7e:ec:c2:50:5d:94:04:
    65:e1:ab:ec:46:b8:0d:60:59:df:56:de:38:6d:d4:89:db:0a:
    2c:f3:76:b5:0f:4c:ff:25:8b:26:af:f1:a6:ff:22:89:a5:4b:
    59:3d:02:e1:35:bd:96:3d:12:4c:ee:bc:b2:d1:25:f6:c7:50:
    1d:cc:85:32:ca:c7:10:6a:7a:45:51:86:47:01:b1:80:4b:6e:
    a2:ef:73:79:07:35:75:ec:7b:f0:1e:c7:3a:08:9a:ad:ad:e7:
    29:5e
-----BEGIN CERTIFICATE-----
MIIDtDCCAx2gAwIBAgIBATANBgkqhkiG9w0BAQQFADCBljELMAkGA1UEBhMCRLIx
DzANBgNVBAsTBkZyYW5jZTETMBAGA1UEBjMjTWFyc2VpbGx1MQwwCgYDVQKKEwNF
TUUxDjAMBGNVBAStBWFkbWluMQ8wDQYDVQDEwZDQxUxZDQDEwZDQxUxZDQDEw
CQEWJGNocm1zdGhhbi5jYWx1Y2FAZW11LWVuc2VpZ251bWVudC5mcjAeFw0wNDAA
Mjc0MTE1MzdaFw0wNTA2Mjc0MTE1MzdaMIGCMQswCQYDVQKGEwJGUjEPMA0GA1UE
CBMGRnJhbmNlMQwwCgYDVQKKEwNFTEUxDjAMBGNVBAStBWFkbWluMQ8wDQYDVQKQ
EwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQ
xUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQx
UxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxU
xZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUx
ZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZ
DQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZD
QDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQ
DEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDE
wZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEw
ZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZ
DQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
QxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZDQxUxZDQDEwZD
```

```
XLEO4DCKBYH62BpbMNJasFRX8PJhofzD7FcCAwEAAaOCASiWggEeMAkGA1UdEwQC
MAAwLAYJYIZIAyb4QgENBB8WHU9wZW5TU0wgR2VuZXJhdGVkIEN1cnRpZmljYXRl
MB0GA1UdDgQWBBTJl1lasQ1sYczRkHlD0/Q+Dwc4TDCBwwYDVR0jBIG7MIG4gBSs
8qjdVhCarMP1a/Ypcrfs878QhqGBnKSBmTCBljELMAkGA1UEBhMCRLIxDzANBgNV
BAGTBkZyYW5jZTEZSMBAGA1UEBxMjTWFyc2VpbGx1MQwwCgYDVQQKEwNFTUUXDjAM
BgNVBAsTBWVkbWluMQ8wDQYDVQQDEwZDQUxFOExMzAxZG9wZDQwODQwODQwODQw
cmlzdGhbi5jYWx1Y2FAZW1lLWVuc2VpZ251bWVudC5mcoIBADANBgkqhkiG9w0B
AQQFAAOBgQAOLbtezOvJd0bws8KTm5W8o+Gimqf0rJvMj3bNVH7sw1BdlARl4avs
RrgNYFnfVt44bdSJ2wos83a1D0z/JYsmr/Gm/yKJpUtZPQLhNb2WPRJM7ryy0SX2
xlAdzIUyyscQanpFUYZHAbsGAS26i73N5BzV17HvwHsc6CJqtrecpXg==
-----END CERTIFICATE-----
aaron:~/demoCA/newcerts#
```

Tout ça est sûrement bien intéressant, il faut le garder sous le coude...

Ce qui saute aux yeux immédiatement, c'est que ce certificat n'est valable qu'un an, à partir de la date de création :

```
Validity
Not Before: Jun 27 11:15:37 2004 GMT
Not After : Jun 27 11:15:37 2005 GMT
```

Ce sont des options calculées à partir des paramètres par défaut. Bien entendu, nous aurions pu forcer deux autres dates dans la ligne de commande.

Au tour de CYCLOPE

Il faut maintenant créer pour CYCLOPE une clé privée et un certificat. CYCLOPE va créer localement sa clé privée, sa demande de certificat, envoyer à AARON sa demande de certificat pour le faire approuver et AARON va lui renvoyer son certificat.

Création de la clé privée et de la demande de certificat

```
cyclope:~# openssl req -nodes -new -keyout cyclope.key -out cyclope.csr
```

Et c'est reparti...

```
cyclope:~# openssl req -nodes -new -keyout cyclope.key -out cyclope.csr
Generating a 1024 bit RSA private key
.....+++++
...+++++
writing new private key to 'cyclope.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:France
Locality Name (eg, city) []:Marseille
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MyOrg
Organizational Unit Name (eg, section) []:admin
Common Name (eg, YOUR name) []:CHRIS
Email Address []:chris@cyclope.maison.mrs

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:lageducapitaine
An optional company name []:
cyclope:~#
```

Nous envoyons le fichier cyclope.csr à AARON, nous allons le faire avec scp, ça ira plus vite que

par la poste :

```
cyclope:~# scp cyclope.csr 82.127.57.95:/root/
The authenticity of host '82.127.57.95 (82.127.57.95)' can't be established.
RSA key fingerprint is 60:fa:5b:26:4e:f1:5a:7d:96:f5:c9:da:48:62:78:1a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '82.127.57.95' (RSA) to the list of known hosts.
Password:
cyclope.csr                               100% 745      0.7KB/s   00:00
cyclope:~#
```

Sur AARON, nous validons le certificat :

```
aaron:~# openssl ca -cert aaron-ca.crt -keyfile aaron-ca.key -out cyclope.crt -in cyclope.csr
Using configuration from /usr/lib/ssl/openssl.cnf
DEBUG[load_index]: unique_subject = "yes"
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 2 (0x2)
  Validity
    Not Before: Jun 27 14:21:46 2004 GMT
    Not After : Jun 27 14:21:46 2005 GMT
  Subject:
    countryName           = FR
    stateOrProvinceName  = France
    organizationName      = MyOrg
    organizationalUnitName = admin
    commonName            = CHRIS
    emailAddress          = chris@cyclope.maison.mrs
  X509v3 extensions:
    X509v3 Basic Constraints:
      CA:FALSE
    Netscape Comment:
      OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
      A0:45:F9:E4:25:0C:B1:07:3B:D3:1F:EE:73:3B:1A:C2:46:32:70:14
    X509v3 Authority Key Identifier:
      keyid:AC:F2:A8:DD:56:10:80:AC:C3:F5:6B:F6:29:72:B7:D2:F3:BF:10:86
      DirName:/C=FR/ST=France/L=Marseille/O=MyOrg/OU=admin/CN=Supervisor/emailAddress=supervisor@yahoo.fr
      serial:00

Certificate is to be certified until Jun 27 14:21:46 2005 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]:y
Write out database with 1 new entries
Data Base Updated
aaron:~#
```

que nous renvoyons à cyclope :

```
aaron:~# scp cyclope.crt 80.8.135.15:/root/
The authenticity of host '80.8.135.15 (80.8.135.15)' can't be established.
RSA key fingerprint is 49:08:1b:f3:5e:cb:3c:97:67:b9:06:47:8b:8c:36:e5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '80.8.135.15' (RSA) to the list of known hosts.
Password:
cyclope.crt                               100% 3635     3.6KB/s   00:00
aaron:~#
```

Et nous devons aussi lui envoyer le certificat de l'autorité aaron-ca.crt :

```
aaron:~# scp aaron-ca.crt 80.8.135.15:/root/
Password:
aaron-ca.crt 100% 1310 1.3KB/s 00:00
aaron:~#
```

Bien. Nous sommes presque au bout :

- AARON possède :
 - son propre certificat d'autorité aaron-ca.crt
 - le certificat aaron.crt qui correspond à sa clé privée aaron.key
 - le certificat de CYCLOPE, cyclope.crt, puisque c'est lui qui l'a validé,
- CYCLOPE possède :
 - sa clé privée, cyclope.key
 - le certificat qui lui correspond,
 - le certificat de l'autorité de certification, aaron-ca.crt.

Et pour cacher les données dans le tunnel ?

Nous pourrions nous arrêter là, si nous voulions juste nous contenter d'authentifier les extrémités du tunnel, sans chiffrer les données qui vont passer dedans, mais, au point où nous en sommes, autant donner à AARON, aussi serveur OpenVPN, le matériel pour créer une clé de session pour un chiffrement symétrique des données dans le tunnel, pour les rendre confidentielles.

Sur AARON :

```
aaron:~# openssl dhparam -out dh1024.pem 1024
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
.....
```

Et huit minutes plus tard, nous avons le fichier :

```
aaron:~# cat dh1024.pem
-----BEGIN DH PARAMETERS-----
MIGHAoGBANcTIW0XcvuNDSgK+KiS0rmo14zHNxnMK3IHjvWCqjJL8F0+nwFKvLq1
FBggAoL1Si+4iuVJoZU3T4H/tsGlsayvWF114PNVuaenER2bhIDeFNGx1A/WbYmK
1JNOVpyfwC/ilfv1L/8bpzrjGsgl4GIy8sKzJt+PXRcV61fcJIM7AgEC
-----END DH PARAMETERS-----
```

Celui là, on ne pourra pas dire qu'il a été écrit à la légère.

Nous avons maintenant tout ce qu'il faut pour réaliser ce tunnel "parfaitement" sécurisé, c'est à dire confidentiel et dont les extrémités sont mutuellement authentifiées.

Le tunnel est dit "étanche" parce que ce qui y circule dedans ne peut être vu de dehors (confidentialité) et personne ne peut y introduire de données frauduleuses (authentification des extrémités).

Tunnel sécurisé

La dernière ligne droite

Nous démarrons le tunnel, comme précédemment, mais avec les options nécessaires à l'étanchéité (un tunnel sans fuites parce les données seront chiffrées et que les extrémités seront authentifiées) :

Sur AARON

```
aaron:~# openvpn --port 8147 \  
> --dev tun1 \  
> --ifconfig 192.168.25.1 192.168.25.2 \  
> --tls-server \  
> --dh dh1024.pem \  
> --ca aaron-ca.crt \  
> --cert aaron.crt \  
> --key aaron.key \  
> --reneg-sec 21600 \  
> --comp-lzo \  
> --verb 5
```

Pour de plus amples renseignements sur les paramètres d'OpenVPN, consultez le "man", dont il existe une traduction française [ici](#)⁶.

Démarrage d'OpenVPN sur AARON :

```
Sun Jun 27 17:16:06 2004 0[0]: Current Parameter Settings:  
Sun Jun 27 17:16:06 2004 1[0]: config = '[UNDEF]'  
Sun Jun 27 17:16:06 2004 2[0]: persist_config = DISABLED  
Sun Jun 27 17:16:06 2004 3[0]: persist_mode = 1  
Sun Jun 27 17:16:06 2004 4[0]: show_ciphers = DISABLED  
Sun Jun 27 17:16:06 2004 5[0]: show_digests = DISABLED  
Sun Jun 27 17:16:06 2004 6[0]: genkey = DISABLED  
Sun Jun 27 17:16:06 2004 7[0]: askpass = DISABLED  
Sun Jun 27 17:16:06 2004 8[0]: show_tls_ciphers = DISABLED  
Sun Jun 27 17:16:06 2004 9[0]: proto = 0  
Sun Jun 27 17:16:06 2004 10[0]: local = '[UNDEF]'  
Sun Jun 27 17:16:06 2004 11[0]: remote = '[UNDEF]'  
Sun Jun 27 17:16:06 2004 12[0]: local_port = 8147  
Sun Jun 27 17:16:06 2004 13[0]: remote_port = 8147  
Sun Jun 27 17:16:06 2004 14[0]: remote_float = DISABLED  
Sun Jun 27 17:16:06 2004 15[0]: ipchange = '[UNDEF]'  
Sun Jun 27 17:16:06 2004 16[0]: bind_local = ENABLED  
Sun Jun 27 17:16:06 2004 17[0]: dev = 'tun1'  
Sun Jun 27 17:16:06 2004 18[0]: dev_type = '[UNDEF]'  
Sun Jun 27 17:16:06 2004 19[0]: dev_node = '[UNDEF]'  
Sun Jun 27 17:16:06 2004 20[0]: tun_ipv6 = DISABLED  
Sun Jun 27 17:16:06 2004 21[0]: ifconfig_local = '192.168.25.1'  
Sun Jun 27 17:16:06 2004 22[0]: ifconfig_remote_netmask = '192.168.25.2'  
Sun Jun 27 17:16:06 2004 23[0]: ifconfig_noexec = DISABLED  
Sun Jun 27 17:16:06 2004 24[0]: ifconfig_nowarn = DISABLED  
Sun Jun 27 17:16:06 2004 25[0]: shaper = 0  
Sun Jun 27 17:16:06 2004 26[0]: tun_mtu = 1300  
Sun Jun 27 17:16:06 2004 27[0]: tun_mtu_defined = DISABLED  
Sun Jun 27 17:16:06 2004 28[0]: link_mtu = 1300  
Sun Jun 27 17:16:06 2004 29[0]: link_mtu_defined = ENABLED  
Sun Jun 27 17:16:06 2004 30[0]: tun_mtu_extra = 0  
Sun Jun 27 17:16:06 2004 31[0]: tun_mtu_extra_defined = DISABLED  
Sun Jun 27 17:16:06 2004 32[0]: fragment = 0  
Sun Jun 27 17:16:06 2004 33[0]: mtu_discover_type = -1  
Sun Jun 27 17:16:06 2004 34[0]: mtu_test = 0  
Sun Jun 27 17:16:06 2004 35[0]: mlock = DISABLED  
Sun Jun 27 17:16:06 2004 36[0]: inactivity_timeout = 0  
Sun Jun 27 17:16:06 2004 37[0]: ping_send_timeout = 0
```

⁶ Manpage d'OpenVPN en français : <http://lehmann.free.fr/openvpn/OpenVPNMan/>

```
Sun Jun 27 17:16:06 2004 38[0]: ping_rec_timeout = 0
Sun Jun 27 17:16:06 2004 39[0]: ping_rec_timeout_action = 0
Sun Jun 27 17:16:06 2004 40[0]: ping_timer_remote = DISABLED
Sun Jun 27 17:16:06 2004 41[0]: persist_tun = DISABLED
Sun Jun 27 17:16:06 2004 42[0]: persist_local_ip = DISABLED
Sun Jun 27 17:16:06 2004 43[0]: persist_remote_ip = DISABLED
Sun Jun 27 17:16:06 2004 44[0]: persist_key = DISABLED
Sun Jun 27 17:16:06 2004 45[0]: mssfix_defined = DISABLED
Sun Jun 27 17:16:06 2004 46[0]: mssfix = 0
Sun Jun 27 17:16:06 2004 47[0]: passtos = DISABLED
Sun Jun 27 17:16:06 2004 48[0]: resolve_retry_seconds = 0
Sun Jun 27 17:16:06 2004 49[0]: connect_retry_seconds = 5
Sun Jun 27 17:16:06 2004 50[0]: username = '[UNDEF]'
Sun Jun 27 17:16:06 2004 51[0]: groupname = '[UNDEF]'
Sun Jun 27 17:16:06 2004 52[0]: chroot_dir = '[UNDEF]'
Sun Jun 27 17:16:06 2004 53[0]: cd_dir = '[UNDEF]'
Sun Jun 27 17:16:06 2004 54[0]: writepid = '[UNDEF]'
Sun Jun 27 17:16:06 2004 55[0]: up_script = '[UNDEF]'
Sun Jun 27 17:16:06 2004 56[0]: down_script = '[UNDEF]'
Sun Jun 27 17:16:06 2004 57[0]: up_restart = DISABLED
Sun Jun 27 17:16:06 2004 58[0]: daemon = DISABLED
Sun Jun 27 17:16:06 2004 59[0]: inetd = 0
Sun Jun 27 17:16:06 2004 60[0]: log = DISABLED
Sun Jun 27 17:16:06 2004 61[0]: nice = 0
Sun Jun 27 17:16:06 2004 62[0]: verbosity = 5
Sun Jun 27 17:16:06 2004 63[0]: mute = 0
Sun Jun 27 17:16:06 2004 64[0]: gremlin = DISABLED
Sun Jun 27 17:16:06 2004 65[0]: occ = ENABLED
Sun Jun 27 17:16:06 2004 66[0]: http_proxy_server = '[UNDEF]'
Sun Jun 27 17:16:06 2004 67[0]: http_proxy_port = 0
Sun Jun 27 17:16:06 2004 68[0]: http_proxy_auth_method = '[UNDEF]'
Sun Jun 27 17:16:06 2004 69[0]: http_proxy_auth_file = '[UNDEF]'
Sun Jun 27 17:16:06 2004 70[0]: http_proxy_retry = DISABLED
Sun Jun 27 17:16:06 2004 71[0]: socks_proxy_server = '[UNDEF]'
Sun Jun 27 17:16:06 2004 72[0]: socks_proxy_port = 0
Sun Jun 27 17:16:06 2004 73[0]: socks_proxy_retry = DISABLED
Sun Jun 27 17:16:06 2004 74[0]: comp_lzo = ENABLED
Sun Jun 27 17:16:06 2004 75[0]: comp_lzo_adaptive = ENABLED
Sun Jun 27 17:16:06 2004 76[0]: route_script = '[UNDEF]'
Sun Jun 27 17:16:06 2004 77[0]: route_default_gateway = '[UNDEF]'
Sun Jun 27 17:16:06 2004 78[0]: route_noexec = DISABLED
Sun Jun 27 17:16:06 2004 79[0]: route_delay = 0
Sun Jun 27 17:16:06 2004 80[0]: route_delay_defined = DISABLED
Sun Jun 27 17:16:06 2004 81[0]: shared_secret_file = '[UNDEF]'
Sun Jun 27 17:16:06 2004 82[0]: key_direction = 0
Sun Jun 27 17:16:06 2004 83[0]: ciphername_defined = ENABLED
Sun Jun 27 17:16:06 2004 84[0]: ciphername = 'BF-CBC'
Sun Jun 27 17:16:06 2004 85[0]: authname_defined = ENABLED
Sun Jun 27 17:16:06 2004 86[0]: authname = 'SHA1'
Sun Jun 27 17:16:06 2004 87[0]: keysize = 0
Sun Jun 27 17:16:06 2004 88[0]: replay = ENABLED
Sun Jun 27 17:16:06 2004 89[0]: replay_window = 64
Sun Jun 27 17:16:06 2004 90[0]: replay_time = 15
Sun Jun 27 17:16:06 2004 91[0]: packet_id_file = '[UNDEF]'
Sun Jun 27 17:16:06 2004 92[0]: use_iv = ENABLED
Sun Jun 27 17:16:06 2004 93[0]: test_crypto = DISABLED
Sun Jun 27 17:16:06 2004 94[0]: tls_server = ENABLED
Sun Jun 27 17:16:06 2004 95[0]: tls_client = DISABLED
Sun Jun 27 17:16:06 2004 96[0]: key_method = 1
Sun Jun 27 17:16:06 2004 97[0]: ca_file = 'aaron-ca.crt'
Sun Jun 27 17:16:06 2004 98[0]: dh_file = 'dh1024.pem'
Sun Jun 27 17:16:06 2004 99[0]: cert_file = 'aaron.crt'
Sun Jun 27 17:16:06 2004 100[0]: priv_key_file = 'aaron.key'
Sun Jun 27 17:16:06 2004 101[0]: cipher_list = '[UNDEF]'
Sun Jun 27 17:16:06 2004 102[0]: tls_verify = '[UNDEF]'
Sun Jun 27 17:16:06 2004 103[0]: tls_remote = '[UNDEF]'
Sun Jun 27 17:16:06 2004 104[0]: crl_file = '[UNDEF]'
Sun Jun 27 17:16:06 2004 105[0]: tls_timeout = 2
Sun Jun 27 17:16:06 2004 106[0]: renegotiate_bytes = 0
Sun Jun 27 17:16:06 2004 107[0]: renegotiate_packets = 0
Sun Jun 27 17:16:06 2004 108[0]: renegotiate_seconds = 21600
Sun Jun 27 17:16:06 2004 109[0]: handshake_window = 60
Sun Jun 27 17:16:06 2004 110[0]: transition_window = 3600
Sun Jun 27 17:16:06 2004 111[0]: single_session = DISABLED
Sun Jun 27 17:16:06 2004 112[0]: tls_auth_file = '[UNDEF]'
Sun Jun 27 17:16:06 2004 113[0]: OpenVPN 1.6.0 i386-pc-linux-gnu [SSL] [LZO] [PTHREAD] built on Jun
10 2004
```

```

Sun Jun 27 17:16:07 2004 114[0]: Diffie-Hellman initialized with 1024 bit key
Sun Jun 27 17:16:07 2004 115[0]: WARNING: file 'aaron.key' is group or others accessible
### Une clé privée doit être PRIVÉE, donc il faut corriger ça...
Sun Jun 27 17:16:07 2004 116[0]: LZO compression initialized
Sun Jun 27 17:16:07 2004 117[0]: Control Channel MTU parms [ L:1300 D:138 EF:38 EB:0 ET:0 EL:0 ]
Sun Jun 27 17:16:07 2004 118[0]: TUN/TAP device tun1 opened
Sun Jun 27 17:16:07 2004 119[0]: /sbin/ifconfig tun1 192.168.25.1 pointopoint 192.168.25.2 mtu 1258
Sun Jun 27 17:16:07 2004 120[0]: Data Channel MTU parms [ L:1300 D:1300 EF:42 EB:19 ET:0 EL:0 ]
Sun Jun 27 17:16:07 2004 121[0]: Local Options String: 'V3,dev-type tun,link-mtu 1300,tun-mtu 1258,
proto UDPv4,ifconfig 192.168.25.2 192.168.25.1,
comp-lzo,cipher BF-CBC,auth SHA1,keysize 128,tls-server'
Sun Jun 27 17:16:07 2004 122[0]: Expected Remote Options String: 'V3,dev-type tun,link-mtu 1300,
tun-mtu 1258,proto UDPv4,ifconfig 192.168.25.1 192.168.25.2,
comp-lzo,cipher BF-CBC,auth SHA1,keysize 128,tls-client'
Sun Jun 27 17:16:07 2004 123[0]: Local Options hash (VER=V3): '562c6d4b'
Sun Jun 27 17:16:07 2004 124[0]: Expected Remote Options hash (VER=V3): '65854add'
Sun Jun 27 17:16:07 2004 125[0]: PTHREAD support initialized
Sun Jun 27 17:16:07 2004 126[0]: UDPv4 link local (bound): [undef]:8147
Sun Jun 27 17:16:07 2004 127[0]: UDPv4 link remote: [undef]

```

Pas d'insultes, c'est bon signe. Dans la fin du log, observez les lignes qui explicitent le mode de chiffrement choisi.

Sur CYCLOPE

```

cyclope:~# openvpn \
> --remote 82.127.57.95 \
> --port 8147 \
> --dev tun1 \
> --ifconfig 192.168.25.2 192.168.25.1 \
> --tls-client \
> --ca aaron-ca.crt \
> --cert cyclope.crt \
> --key cyclope.key \
> --reneg-sec 21600 \
> --comp-lzo \
> --verb 5

```

Nous retrouvons ici le même principe que pour AARON, hormis les paramètres :

- --remote, nous avons déjà vu ça dans le test d'OpenVPN sans chiffrement,
- --ca qui est bien entendu la même autorité utilisée aux deux bouts du tunnel.

```

Sun Jun 27 17:27:19 2004 0[0]: Current Parameter Settings:
Sun Jun 27 17:27:19 2004 1[0]: config = '[UNDEF]'
Sun Jun 27 17:27:19 2004 2[0]: persist_config = DISABLED
Sun Jun 27 17:27:19 2004 3[0]: persist_mode = 1
Sun Jun 27 17:27:19 2004 4[0]: show_ciphers = DISABLED
Sun Jun 27 17:27:19 2004 5[0]: show_digests = DISABLED
Sun Jun 27 17:27:19 2004 6[0]: genkey = DISABLED
Sun Jun 27 17:27:19 2004 7[0]: askpass = DISABLED
Sun Jun 27 17:27:19 2004 8[0]: show_tls_ciphers = DISABLED
Sun Jun 27 17:27:19 2004 9[0]: proto = 0
Sun Jun 27 17:27:19 2004 10[0]: local = '[UNDEF]'
Sun Jun 27 17:27:19 2004 11[0]: remote = '82.127.57.95'
Sun Jun 27 17:27:19 2004 12[0]: local_port = 8147
Sun Jun 27 17:27:19 2004 13[0]: remote_port = 8147
Sun Jun 27 17:27:19 2004 14[0]: remote_float = DISABLED
Sun Jun 27 17:27:19 2004 15[0]: ipchange = '[UNDEF]'
Sun Jun 27 17:27:19 2004 16[0]: bind_local = ENABLED
Sun Jun 27 17:27:19 2004 17[0]: dev = 'tun1'
Sun Jun 27 17:27:19 2004 18[0]: dev_type = '[UNDEF]'
Sun Jun 27 17:27:19 2004 19[0]: dev_node = '[UNDEF]'
Sun Jun 27 17:27:19 2004 20[0]: tun_ipv6 = DISABLED
Sun Jun 27 17:27:19 2004 21[0]: ifconfig_local = '192.168.25.2'
Sun Jun 27 17:27:19 2004 22[0]: ifconfig_remote_netmask = '192.168.25.1'
Sun Jun 27 17:27:19 2004 23[0]: ifconfig_noexec = DISABLED
Sun Jun 27 17:27:19 2004 24[0]: ifconfig_nowarn = DISABLED
Sun Jun 27 17:27:19 2004 25[0]: shaper = 0
Sun Jun 27 17:27:19 2004 26[0]: tun_mtu = 1300
Sun Jun 27 17:27:19 2004 27[0]: tun_mtu_defined = DISABLED

```

```
Sun Jun 27 17:27:19 2004 28[0]: link_mtu = 1300
Sun Jun 27 17:27:19 2004 29[0]: link_mtu_defined = ENABLED
Sun Jun 27 17:27:19 2004 30[0]: tun_mtu_extra = 0
Sun Jun 27 17:27:19 2004 31[0]: tun_mtu_extra_defined = DISABLED
Sun Jun 27 17:27:19 2004 32[0]: fragment = 0
Sun Jun 27 17:27:19 2004 33[0]: mtu_discover_type = -1
Sun Jun 27 17:27:19 2004 34[0]: mtu_test = 0
Sun Jun 27 17:27:19 2004 35[0]: mlock = DISABLED
Sun Jun 27 17:27:19 2004 36[0]: inactivity_timeout = 0
Sun Jun 27 17:27:19 2004 37[0]: ping_send_timeout = 0
Sun Jun 27 17:27:19 2004 38[0]: ping_rec_timeout = 0
Sun Jun 27 17:27:19 2004 39[0]: ping_rec_timeout_action = 0
Sun Jun 27 17:27:19 2004 40[0]: ping_timer_remote = DISABLED
Sun Jun 27 17:27:19 2004 41[0]: persist_tun = DISABLED
Sun Jun 27 17:27:19 2004 42[0]: persist_local_ip = DISABLED
Sun Jun 27 17:27:19 2004 43[0]: persist_remote_ip = DISABLED
Sun Jun 27 17:27:19 2004 44[0]: persist_key = DISABLED
Sun Jun 27 17:27:19 2004 45[0]: mssfix_defined = DISABLED
Sun Jun 27 17:27:19 2004 46[0]: mssfix = 0
Sun Jun 27 17:27:19 2004 47[0]: passtos = DISABLED
Sun Jun 27 17:27:19 2004 48[0]: resolve_retry_seconds = 0
Sun Jun 27 17:27:19 2004 49[0]: connect_retry_seconds = 5
Sun Jun 27 17:27:19 2004 50[0]: username = '[UNDEF]'
Sun Jun 27 17:27:19 2004 51[0]: groupname = '[UNDEF]'
Sun Jun 27 17:27:19 2004 52[0]: chroot_dir = '[UNDEF]'
Sun Jun 27 17:27:19 2004 53[0]: cd_dir = '[UNDEF]'
Sun Jun 27 17:27:19 2004 54[0]: writepid = '[UNDEF]'
Sun Jun 27 17:27:19 2004 55[0]: up_script = '[UNDEF]'
Sun Jun 27 17:27:19 2004 56[0]: down_script = '[UNDEF]'
Sun Jun 27 17:27:19 2004 57[0]: up_restart = DISABLED
Sun Jun 27 17:27:19 2004 58[0]: daemon = DISABLED
Sun Jun 27 17:27:19 2004 59[0]: inetd = 0
Sun Jun 27 17:27:19 2004 60[0]: log = DISABLED
Sun Jun 27 17:27:19 2004 61[0]: nice = 0
Sun Jun 27 17:27:19 2004 62[0]: verbosity = 5
Sun Jun 27 17:27:19 2004 63[0]: mute = 0
Sun Jun 27 17:27:19 2004 64[0]: gremlin = DISABLED
Sun Jun 27 17:27:19 2004 65[0]: occ = ENABLED
Sun Jun 27 17:27:19 2004 66[0]: http_proxy_server = '[UNDEF]'
Sun Jun 27 17:27:19 2004 67[0]: http_proxy_port = 0
Sun Jun 27 17:27:19 2004 68[0]: http_proxy_auth_method = '[UNDEF]'
Sun Jun 27 17:27:19 2004 69[0]: http_proxy_auth_file = '[UNDEF]'
Sun Jun 27 17:27:19 2004 70[0]: http_proxy_retry = DISABLED
Sun Jun 27 17:27:19 2004 71[0]: socks_proxy_server = '[UNDEF]'
Sun Jun 27 17:27:19 2004 72[0]: socks_proxy_port = 0
Sun Jun 27 17:27:19 2004 73[0]: socks_proxy_retry = DISABLED
Sun Jun 27 17:27:19 2004 74[0]: comp_lzo = ENABLED
Sun Jun 27 17:27:19 2004 75[0]: comp_lzo_adaptive = ENABLED
Sun Jun 27 17:27:19 2004 76[0]: route_script = '[UNDEF]'
Sun Jun 27 17:27:19 2004 77[0]: route_default_gateway = '[UNDEF]'
Sun Jun 27 17:27:19 2004 78[0]: route_noexec = DISABLED
Sun Jun 27 17:27:19 2004 79[0]: route_delay = 0
Sun Jun 27 17:27:19 2004 80[0]: route_delay_defined = DISABLED
Sun Jun 27 17:27:19 2004 81[0]: shared_secret_file = '[UNDEF]'
Sun Jun 27 17:27:19 2004 82[0]: key_direction = 0
Sun Jun 27 17:27:19 2004 83[0]: ciphername_defined = ENABLED
Sun Jun 27 17:27:19 2004 84[0]: ciphername = 'BF-CBC'
Sun Jun 27 17:27:19 2004 85[0]: authname_defined = ENABLED
Sun Jun 27 17:27:19 2004 86[0]: authname = 'SHA1'
Sun Jun 27 17:27:19 2004 87[0]: keysize = 0
Sun Jun 27 17:27:19 2004 88[0]: replay = ENABLED
Sun Jun 27 17:27:19 2004 89[0]: replay_window = 64
Sun Jun 27 17:27:19 2004 90[0]: replay_time = 15
Sun Jun 27 17:27:19 2004 91[0]: packet_id_file = '[UNDEF]'
Sun Jun 27 17:27:19 2004 92[0]: use_iv = ENABLED
Sun Jun 27 17:27:19 2004 93[0]: test_crypto = DISABLED
Sun Jun 27 17:27:19 2004 94[0]: tls_server = DISABLED
Sun Jun 27 17:27:19 2004 95[0]: tls_client = ENABLED
Sun Jun 27 17:27:19 2004 96[0]: key_method = 1
Sun Jun 27 17:27:19 2004 97[0]: ca_file = 'aaron-ca.crt'
Sun Jun 27 17:27:19 2004 98[0]: dh_file = '[UNDEF]'
Sun Jun 27 17:27:19 2004 99[0]: cert_file = 'cyclope.crt'
Sun Jun 27 17:27:19 2004 100[0]: priv_key_file = 'cyclope.key'
Sun Jun 27 17:27:19 2004 101[0]: cipher_list = '[UNDEF]'
Sun Jun 27 17:27:19 2004 102[0]: tls_verify = '[UNDEF]'
Sun Jun 27 17:27:19 2004 103[0]: tls_remote = '[UNDEF]'
Sun Jun 27 17:27:19 2004 104[0]: crl_file = '[UNDEF]'
```



```
Sun Jun 27 17:22:29 2004 134[1]: Data Channel Encrypt: Using 160 bit message hash 'SHA1' for HMAC authentication
WWRRRR
Sun Jun 27 17:22:29 2004 135[1]: Control Channel: TLSv1, cipher TLSv1/SSLv3 DHE-RSA-AES256-SHA, 1024 bit RSA
Sun Jun 27 17:22:29 2004 136[1]: Peer Connection Initiated with 80.8.135.15:8147
```

Donc, tout semble parfait.

Nous pourrions re-vérifier les routes et les interfaces sur les deux machines, mais nous savons que ça fonctionne, donc gagnons du temps :

```
cyclope:~# ping -c 4 192.168.25.1
PING 192.168.25.1 (192.168.25.1): 56 data bytes
64 bytes from 192.168.25.1: icmp_seq=0 ttl=64 time=69.8 ms
64 bytes from 192.168.25.1: icmp_seq=1 ttl=64 time=70.0 ms
64 bytes from 192.168.25.1: icmp_seq=2 ttl=64 time=74.7 ms
64 bytes from 192.168.25.1: icmp_seq=3 ttl=64 time=69.1 ms

--- 192.168.25.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 69.1/70.9/74.7 ms
```

Le tunnel fonctionne, les extrémités sont authentifiées et les données sont compressées et chiffrées dans le tunnel.

Avec tout ça, nous devrions être à peu près tranquilles.

La touche finale

La ligne de commande nécessaire au démarrage d'OpenVPN n'est pas très pratique. De plus, comme nous l'avons fait, une console sera bloquée à chaque bout du tunnel, tout au long de son existence. C'est souhaitable pendant la période de test, ça ne l'est pas en production.

Fort heureusement, du moins dans la distribution Debian, il est possible de démarrer un tunnel OpenVPN au moyen du SystemV. Un script, `/etc/init.d/openvpn` est fourni dans ce but.

Ce script va démarrer, arrêter, redémarrer, recharger un ou plusieurs tunnels dont les paramètres sont définis dans autant de fichiers placés dans `/etc/openvpn`.

Par exemple, sur CYCLOPE, nous pouvons créer un fichier :

```
/etc/openvpn/MyOrg.conf
```

Qui contiendrait les choses suivantes :

```
cyclope:~# cat /etc/openvpn/MyOrg.conf
remote 82.127.57.95
port 8147
dev tun1
ifconfig 192.168.25.2 192.168.25.1
tls-client
ca /root/aaron-ca.crt
cert /root/cyclope.crt
key /root/cyclope.key
reneg-sec 21600
comp-lzo
verb 3
```

Il suffit de reprendre chacune des options ligne par ligne en supprimant les "--" et en indiquant les chemins complets des certificats. Dans l'exemple, il faudra bien entendu que OpenVPN soit démarré par root.

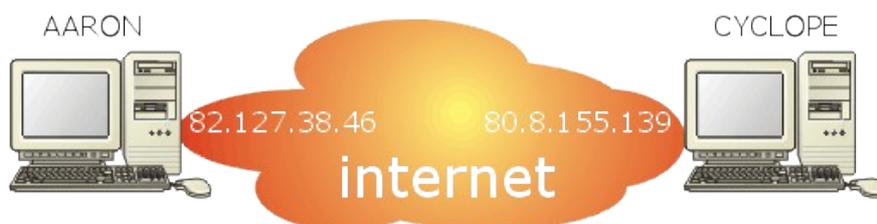
En production, limitez le flot verbal d'openVPN (verb de 1 à 4 maximum), parce que le baratin va se retrouver dans `/var/log/syslog`.

Il est possible de créer autant de tunnels que nécessaire, en prenant soin de changer à chaque fois le port UDP. Les autres paramètres (remote, tun...) dépendront de la configuration de vos divers tunnels.

Un fichier de configuration par tunnel est nécessaire. Lors de la commande `"/etc/init.d/openvpn start"` tous les fichiers présents dans `/etc/openvpn/` seront lus et donneront naissance à une extrémité de tunnel.

Applications

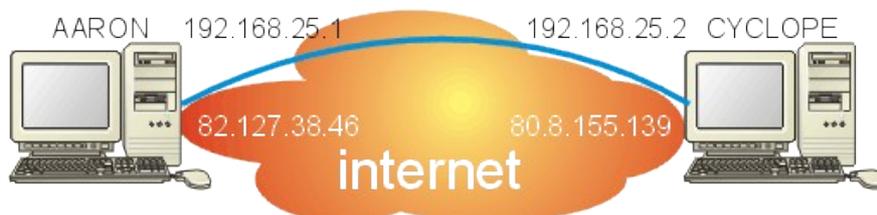
A partir d'une configuration très classique comme celle-ci, où deux hôtes sont connectés à l'Internet :



Il est possible de réaliser pas mal de choses...

Un tunnel sécurisé entre les deux hôtes...

Connectés directement à l'Internet

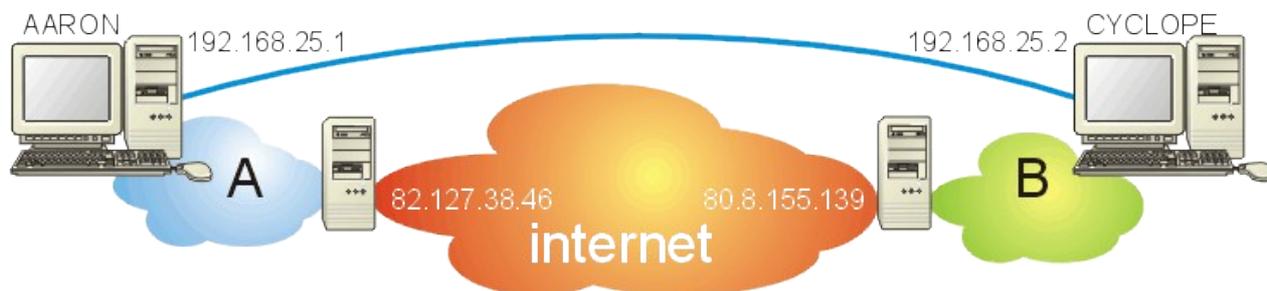


C'est le cas le plus simple, que nous avons vu dans la démonstration, et qui pourra probablement intéresser pas mal de monde, surtout si je vous dis qu'OpenVPN existe également pour Windows.

Notez que pour réaliser un tunnel entre deux systèmes Windows, il existe une solution intégrée, à base de PPTP, protocole Microsoft, qui réalise elle aussi un tunnel point à point. Mais avec OpenVPN, vous pourrez réaliser un tunnel entre plate-formes différentes.

Ou derrière un pare-feu

Nous avons vu qu'OpenVPN crée un tunnel sur UDP, en utilisant un seul port. Il devient alors très simple, si les pare-feux savent faire du "port forwarding", de prolonger le tunnel dans les réseaux privés, jusqu'aux hôtes que l'on souhaite relier :



Dans cette illustration, et en gardant les conventions de la manipulation précédente, il suffira de modifier quelques détails :

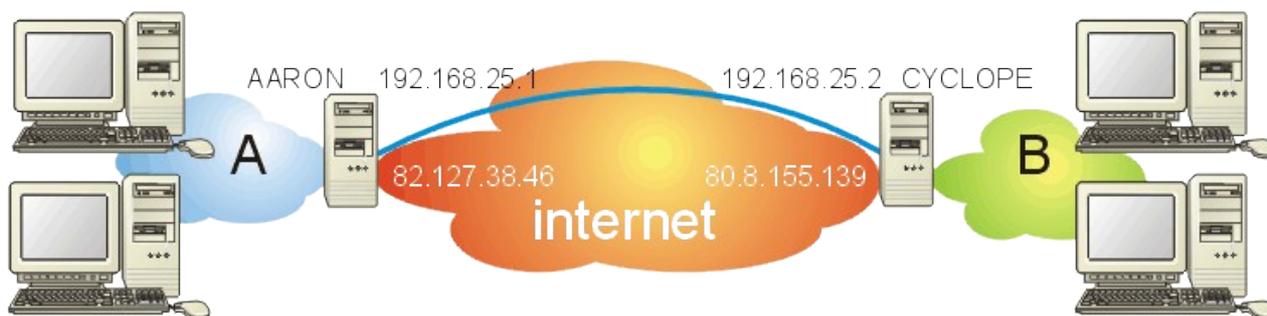
- le serveur OpenVPN est toujours sur AARON,
- le client OpenVPN est toujours sur CYCLOPE,
- les deux hôtes utilisent **toujours** les adresses IP publiques de leurs passerelles respectives,
- la passerelle du réseau A devra rediriger le trafic entrant port 8147 sur le port 8147 d'AARON,
- la passerelle du réseau B devra rediriger le trafic entrant port 8147 sur le port 8147 de CYCLOPE.

Si les deux passerelles sont de type Linux, avec un kernel 2.4 ou supérieur, une règle de type :

```
iptables -t NAT -A PREROUTING -i ppp0 -p udp --dport 8147 -j DNAT --to-destination xx.xx.xx.xx
```

ppp0 peut-être autre chose, suivant le type de connexion à l'internet, xx.xx.xx.xx étant l'adresse IP de l'extrémité du tunnel dans le réseau considéré.

Interconnecter les deux réseaux privés



Si AARON et CYCLOPE sont les routeurs/pare-feux des réseaux A et B, à partir du moment où l'on a établi un tunnel entre ces deux machines, il n'y a aucune raison de ne pas pouvoir établir une route entre les réseaux A et B à travers ce tunnel. Alors, tous les hôtes de A pourront communiquer avec tous les hôtes de B et réciproquement, nous ramenant à une configuration vue avec le tunnel GRE, mais avec plus de sécurité (et moins de bande passante).

Imaginons :

- que le réseau A soit 192.168.1.0, avec le masque 255.255.255.0,
- que le réseau B soit 192.168.0.0, avec le masque 255.255.255.0.

Il suffira alors d'écrire quelques routes bien senties sur les deux passerelles :

- sur AARON :

```
route add -net 192.168.0.0 gw 192.168.25.2 netmask 255.255.255.0
```

- sur CYCLOPE :

```
route add -net 192.168.1.0 gw 192.168.25.1 netmask 255.255.255.0
```

Bien entendu, il ne faudra pas oublier le pare-feu, en lui permettant le "forward" entre le tunnel et le réseau privé sur chacune des passerelles, et nos deux réseaux A et B seront interconnectés.

OpenVPN permet d'ailleurs d'effectuer automatiquement l'ajout de la route dans ses paramètres de configuration, par l'option "--route".

Par exemple, pour le démarrage du serveur sur AARON :

```
openvpn --port 8147 \  
> --dev tun1 \  
> --ifconfig 192.168.25.1 192.168.25.2 \  
> --route 192.168.0.0 255.255.255.0 192.168.25.2 \  
> --tls-server \  
> --dh dh1024.pem \  
> --ca aaron-ca.crt \  
> --cert aaron.crt \  
> --key aaron.key \  
> --reneg-sec 21600 \  
> --comp-lzo \  
> --verb 5
```

Et vous retrouverez automatiquement la bonne route dans la table de routage :

```
aaron:~# route -n  
Kernel IP routing table  
Destination Gateway Genmask Flags Metric Ref Use Iface  
192.168.25.2 0.0.0.0 255.255.255.255 UH 0 0 0 tun1  
...  
192.168.0.0 192.168.25.2 255.255.255.0 UG 0 0 0 tun1  
...  
0.0.0.0 193.253.160.3 0.0.0.0 UG 0 0 0 ppp0  
aaron:~#
```

je vous laisse trouver l'équivalent sur CYCLOPE.