

# Mini-Tor : Le réseau Tor démystifié

Nathan Liccardo

nathan.liccardo@ulb.ac.be

## Abstract

Dans le cadre de mon mémoire de fin de bachelier, j'ai décidé d'aborder le réseau d'anonymisation de connexions et communications Tor. Afin de réaliser mon travail, il m'a été demandé, dans un premier temps, de comprendre l'architecture de ce système afin de pouvoir, dans un second temps, implémenter le système de façon schématique et moins complexe que le réseau Tor réel. On retrouvera donc dans l'ensemble de ce document à la fois la partie théorique de mon travail reprenant les concepts sur lesquels repose le réseau mais également les détails de mon implémentation permettant de retrouver et d'illustrer ces concepts. Je tiens à remercier Monsieur François Gérard pour son aide précieuse lors de la rédaction et l'implémentation de ce travail.

## Introduction

Selon la page officielle du projet Tor <sup>?</sup>, ce dernier a pour objectif principal d'améliorer la vie privée des utilisateurs ainsi que la sécurité de ces derniers sur internet. Tor serait également, un outil efficace contre la censure permettant aux différentes personnes utilisant le réseau de pouvoir avoir accès aux contenus bloqués par différentes formes de régimes autoritaires. Tout au long de ce document, je détaillerai les différentes techniques informatiques utilisées afin de pouvoir créer un réseau tel que Tor. Afin d'explicitier le mieux possible les différents concepts abordés, un réseau Tor simplifié a également été développé. Ce travail sera mis à disposition en annexe de ce document. Tor utilise au sein de son réseau différentes techniques de chiffrement et déchiffrement des messages. L'ensemble des termes directement liés à la sécurité ainsi que les techniques de chiffrement seront donc détaillés dans un premier temps afin de pouvoir ensuite expliciter le fonctionnement de Tor. Mon intérêt pour le réseau Tor est né de part ses techniques informatiques évoluées (que nous verrons dans la suite du document) mais également de part son aspect socio-politique important. Tor reste, à l'heure actuelle, un outil majeur pour la défense de la liberté d'expression dans certains pays.

Avant d'aborder le coeur du sujet et les différents aspects technique de Tor, il est intéressant de se pencher

un instant sur l'histoire de ce réseau. Si l'on se base sur la page officielle des membres du réseau Tor <sup>?</sup>, on trouve deux noms importants à savoir Roger Dingledine et Nick Mathewson. Ces deux informaticiens (maintenant président et vice-président du projet) sont donc en effet à l'origine du réseau Tor. Il est important de noter que le routage en oignon, à savoir l'envoi de message enveloppé sous plusieurs couches distinctes, a été développé dans les années 1990 mais que le réseau (qui se base complètement sur cette technique) n'est sorti qu'en 2002 dans sa version alpha <sup>?</sup> et c'est en 2004 que « The Second-Generation Onion Router » est présenté. Enfin, ce n'est qu'en 2006 que « The Tor Project » est créé <sup>?</sup>. Ce dernier a pour objectif de maintenir et mettre à jour continuellement le réseau Tor <sup>?</sup>.

Comme dit précédemment, Tor a pour objectif d'améliorer la vie privée et la liberté des utilisateurs du réseau. Si l'on se réfère une fois de plus à <sup>?</sup>, on constate que l'anonymat <sup>1</sup> total ne peut être atteint. Le réseau Tor représente donc un outil mis à disposition de manière libre et permettant aux utilisateurs de sécuriser <sup>2</sup> le transport de leurs données d'un point A à un point B. On se rend donc compte que Tor ne permet en aucun d'assurer une anonymisation de l'utilisateur. Pour effectuer cette tâche, le groupe Tor Project conseille donc d'utiliser ce que l'on appelle un « protocol-specific support software » qui permettra à l'utilisateur de masquer les données permettant de l'identifier. The Tor Project a également conçu un navigateur web permettant d'utiliser Tor tout en masquant et bloquant les possibilités d'identification de l'utilisateur.

Dans la prochaine section, nous aborderons donc les techniques de chiffrement des données afin de pouvoir les faire transiter au sein du réseau de manière sécurisée. Suivra ensuite une seconde section permettant d'explicitier le fonctionnement du réseau Tor ainsi que l'utilisation du chiffrement des données au sein de ce dernier.

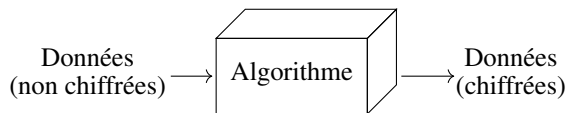
---

1. Anonymat : Etat de quelque chose, quelqu'un qui est anonyme à savoir, dont l'auteur ou le nom est inconnu

2. Sécuriser : Rendre quelque chose plus sûr, fiabiliser

## Chiffrement

La question de la sécurisation des données au sein du réseau Tor est un élément important si l'on souhaite comprendre correctement ce dernier. Il existe, à l'heure actuelle, deux techniques permettant de chiffrer des données à savoir : le chiffrement asymétrique et symétrique. Dans la suite de ce document, lorsque nous parlerons de sécurisation des données, cela reviendra à appliquer l'une de ces deux techniques de chiffrement sur un paquet de données. Dans cette section, je commencerai donc par détailler le principe du chiffrement asymétrique. Ce dernier sera également accompagné d'une application concrète (RSA ?). J'évoquerai donc dans un second temps le chiffrement symétrique en accompagnant également ce dernier d'une application concrète (AES ?). Dans le cas des deux techniques de sécurisation, il est important de comprendre que l'on applique une transformation sur les données. Les fonctions de chiffrement prennent donc une valeur en entrée et ressortent une valeur complètement différente en sortie. Les algorithmes mathématiques appliqués sur les données afin d'obtenir une valeur de sortie différente se basent sur l'utilisation de clés de chiffrement. Ce sont ces dernières (clés) qui varient en fonction du type de chiffrement appliqué (asymétrique ou symétrique). Voici une illustration permettant de schématiser de manière simple le fonctionnement d'une fonction de chiffrement (que l'on utilise la technique symétrique ou asymétrique) :



Les clés évoquées précédemment sont donc utilisées par l'algorithme afin de pouvoir modifier les données fournies en entrée de ce dernier. On parle donc bien de paramètre fourni par l'utilisateur de l'algorithme de chiffrement afin que ce dernier fournisse une sortie définie par la clé. Pour finir, nous utiliserons dans le cadre de ce rapport des clés composées de différents caractères. Il est à noter que ces dernières peuvent être de toute forme à partir du moment où l'algorithme permettant de chiffrer les données est capable d'utiliser cette dernière (câblage électrique, chaîne de caractères, chaîne binaire, ...)

### Asymétrique

Le chiffrement asymétrique est donc la première approche que nous aborderons. Ce dernier est défini par l'utilisation de deux clés de chiffrement et déchiffrement (action inverse du chiffrement)<sup>3</sup> différentes et se base sur les fonctions à sens unique. Soit  $n \in \mathbb{N}$  on pose  $A_n = \{1, 2, \dots, n\}$  et  $B_n$  deux ensembles finis. La fonction :

3. Action de déchiffrer à savoir lire ou comprendre un texte écrit peu lisiblement ou codé.

$$f_n : A_n \rightarrow B_n$$

est définie comme à sens unique si :

- Il est facile de calculer  $f_n(x)$  pour un  $n$  très grand ( $x \in A_n$ ).  $\rightarrow f_n$  facile à calculer.
- Il est difficile pour  $y \in f_n(A_n)$  de trouver un  $x$  tel que  $f(x) = y$ .  $\rightarrow f_n$  difficile à inverser.

Les algorithmes de chiffrement asymétriques sont donc directement inspirés de ce principe. Si l'on applique ce principe sur une phrase à chiffrer, il est aisé d'appliquer la transformation dans un sens mais extrêmement difficile de retrouver le message originel. Il existe cependant bien deux solutions permettant de retrouver le message : effectuer l'ensemble des solutions (généralement exponentiel) ou posséder une clé permettant le déchiffrement. On déduit donc effectivement les deux clés citées précédemment :

- Publique : permet de chiffrer le message (facile)
- Privée : permet de déchiffrer le message (difficile)

Comme dit précédemment, l'utilisation de ce type d'algorithme reste néanmoins extrêmement lent même si l'on possède la clé dite privée. Ils restent cependant très utilisés grâce à leurs propriétés distribuées. L'algorithme le plus connu des systèmes asymétriques est connu sous le nom de RSA ?. Il a été développé en 1978 par Ron Rivest, Adi Shamir et Leonard Adleman ?. Ce dernier porte donc comme nom les initiales des ses trois créateurs. RSA se définit donc par l'utilisation de deux clés différentes pour le chiffrement et le déchiffrement des données. Les étapes nécessaires à la génération des clés sont les suivantes :

1. Générer deux nombres premiers  $p$  et  $q$
2. Calculer  $N = p * q$  et  $\varphi(N) = (p - 1) * (q - 1)$
3. Choisir un entier  $e$ ,  $1 < e < \varphi(N)$  t.q  $\text{pgcd}(e, \varphi(N)) = 1$
4. Calculer  $d$  t.q  $e * d \equiv 1 \pmod{\varphi(N)}$
5. La clé publique :  $(N, e)$  ; La clé privée :  $(d)$

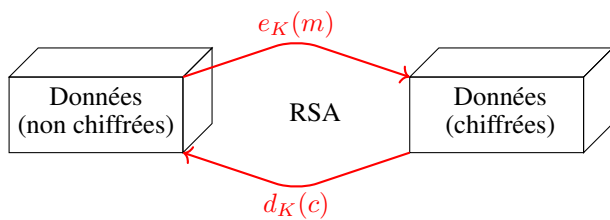
On peut donc à présent calculer les fonctions permettant d'obtenir un message chiffré mais également d'effectuer l'opération inverse (déchiffrement). La fonction de chiffrement  $e_K$  publique est donc définie par :

$$e_K(m) := m^e \pmod{N}$$

La fonction de déchiffrement quand à elle est définie par :

$$d_K(c) := c^d \pmod{N}$$

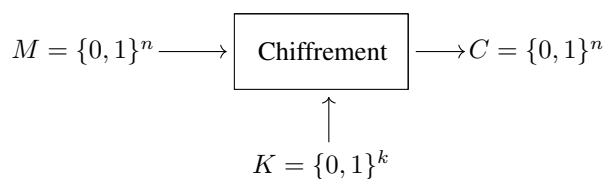
Il est à noter que les fonctions de chiffrement et déchiffrement sont commutatives. Dans le cas où le message est chiffré à partir de la fonction  $e_k(m)$  (resp.  $d_K(c)$ ) il peut alors être déchiffré par  $d_K(c)$  (resp.  $e_K(m)$ ).



RSA est fondé sur l'utilisation de grands nombres ainsi que la difficulté de factoriser ces derniers<sup>4</sup>. Il n'existe cependant à l'heure actuelle aucune preuve mathématique permettant de prouver l'équivalence entre le problème de factorisation et le déchiffrement aisé. On se base donc principalement sur des intuitions mathématiques. L'utilisation de grands nombres signifie utiliser des clés d'au moins 1024 bits.

## Symétrique

Il sera maintenant question du second type de chiffrement à savoir symétrique. Ce dernier se base sur un principe complètement différent du chiffrement asymétrique. En effet, la clé de chiffrement et déchiffrement utilisée dans ce cas-ci est unique et identique. Le chiffrement symétrique se divise également en deux grandes catégories à savoir le chiffrement par bloc (que nous verrons par la suite) et le chiffement par flot (que nous n'aborderons pas ici)<sup>5</sup>. Avant d'entamer les notions de chiffement par bloc, il est important de souligner la différence majeure entre les fonctions de chiffement symétriques et asymétriques. Il a été dit précédemment que le fonctionnement de ces dernières était extrêmement lent. Cela n'est pas le cas pour les systèmes symétriques qui s'appuient sur des fonctions inversibles et efficaces<sup>6</sup>. Un algorithme de chiffement symétrique se définit donc comme la transformation d'un message  $M$  avec une clé secrète  $K$ . Le résultat est donc un message chiffré  $C$ .



Comme dit précédemment, les algorithmes à chiffement symétrique fonctionnent généralement par blocs. On voit alors que le message est traité par morceaux de données de tailles équivalentes<sup>4</sup>. On effectue donc un premier traitement durant lequel on découpe le message afin d'obtenir une succession de blocs pouvant être traités par la suite par l'algorithme. Comme dit dans<sup>5</sup>, il existe deux principes fondamentaux qui sont utilisés dans le chiffement symétrique par blocs. Le premier étant la confusion<sup>5</sup> et le

second la diffusion<sup>6</sup>. Ces deux notions seront donc à la base des algorithmes de chiffement et seront appliqués sur les blocs cités précédemment. Il faut également prendre en compte un dernier point à savoir l'inversibilité de la fonction de chiffement symétrique. Etant donné que la méthode prend comme principale propriété le fait de pouvoir retomber sur le message à partir de la clé de chiffement, on dit que les fonctions de chiffement symétrique sont bijectives. Afin d'illustrer au mieux le fonctionnement d'un chiffement symétrique, nous utiliserons dans ce rapport le système AES (Advanced Encryption Security) qui a été développé dans les années 2000<sup>7</sup>. Concrètement, AES est basé sur un algorithme itératif. En fonction de la taille de la clé (128 ou 256 bits), l'algorithme verra son nombre de tours modifié (resp. 10 ou 14 tours). A chaque tour, on a donc dans un premier temps une confusion qui est assurée par une substitution non-linéaire suivie d'une diffusion quand à elle assurée par permutation linéaire. Voici donc le détail des différentes étapes nécessaires.

**Etape 1.** On commence donc par placer l'ensemble des données (sous forme d'octets) dans une matrice 4\*4 (128 bits au total). Cette étape constitue donc une étape préliminaire qui ne sera pas effectuée lors des différentes itérations. On a donc l'ensemble des données représentées sous la forme suivante :

**Etape 2.**

## Architecture

### Introduction

Comme dit précédemment, chaque message parcourant le réseau Tor est encapsulé dans une structure sous forme d'oignon. Le message encapsulé passe également par une série de proxy avant d'atteindre le destinataire. Afin de comprendre un tel mécanisme, il est intéressant de se pencher sur le fonctionnement des différents éléments constituant le réseau. En premier lieu, nous aborderons donc les clients (qui se connectent au réseau), nous détaillerons ensuite les noeuds du système (qui permettent de « noyer » le message) et pour finir nous expliquerons la façon dont les noeuds sont distribués via les serveurs Tor. Nous émettrons l'hypothèse, pour cette section, que les messages échangés sont sécurisés. L'aspect cryptographique de Tor fera, en effet, l'objet d'une section dédiée dans la suite de ce travail afin de pouvoir approfondir certains aspects importants du réseau.

### Client

Il est intéressant, dans un premier temps, d'aborder la notion de clients. Ce sont eux qui se connectent au réseau afin d'effectuer des requêtes auprès de serveurs web ou

4. On découpe le message en blocs de tailles équivalentes

5. Masquer toute relation linéaire entre le chiffré et le message en clair

6. Cacher la redondance en répartissant l'influence d'un bit de clé sur tout le chiffré

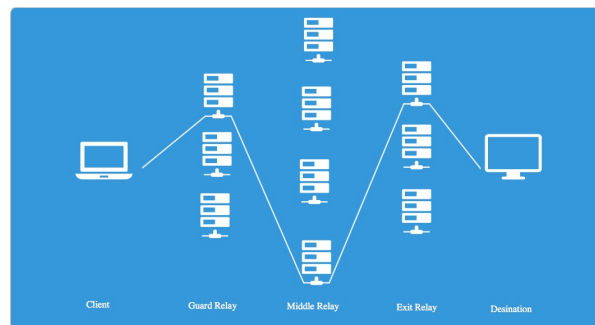
d'échanger des messages de manière anonyme. L'implication des clients reste minimale. En effet, ils se connectent au réseau mais n'effectuent aucune modification sur ce dernier. Il est important de signaler également que ce sont eux qui doivent être protégés. C'est donc pour les clients que le projet Tor a été créé et c'est pour cette raison que l'association Tor Project a mis en place plusieurs outils permettant une simplification d'utilisation. Cependant, il faut noter que les clients peuvent également configurer eux-même les outils Tor. Pour cette raison, le réseau ne bloque pas l'accès via une application unique mais permet bel et bien l'utilisation à toutes les personnes ayant configuré leur système afin d'accéder au réseau. Lorsqu'un client utilise le réseau, il a un certain nombre de tâches à effectuer. Dans un premier temps, il est utile de sélectionner un chemin à parcourir avant d'arriver à la destination. Dans mon implémentation personnelle, ce chemin est constitué de trois noeuds ce qui reflète bien le schéma réel. Une fois le chemin obtenu, le client peut donc transmettre des messages au sein du réseau. Pour ce faire, il lui est utile de formater ces messages sous la forme d'un oignon (utilisation de l'oignon routing). Une fois le message empaqueté, il peut être distribué sur le réseau. A partir de ce moment, la mission du client est terminée et le réseau Tor s'occupe du reste. Pour conclure la partie concernant le client, voici donc les 4 étapes majeures de ce dernier :

- Récupération de la liste des noeuds publics
- Création d'un chemin possible pour les messages
- Empaquetage du message sous le bon format
- Envoi du message au premier noeud du circuit

## Noeuds

L'un des points les plus importants du réseau Tor réside bel et bien dans son architecture interne à savoir les noeuds. Ces derniers constituent un ensemble de proxy (notation souvent utilisée dans diverses sources) améliorés afin d'obtenir une confidentialité maximale aux clients. Lorsque ces derniers se connectent au réseau et constituent un chemin, ils choisissent de préférence des noeuds situés dans différents pays. Le circuit sera alors constitué d'un noeud d'entrée (Entry Guard Node), suivi d'un noeud intermédiaire et non public (Bridge Node) et finira avec un noeud de sortie (Exit Node). Dans un schéma réel, les noeuds de sortie constituent un point de faiblesse puisqu'ils sont en lien direct avec l'extérieur (serveurs, ...). Lorsque la connexion n'est pas sécurisée, le noeud de sortie sera capable de lire et d'écouter l'ensemble des informations transmises. Dans mon implémentation personnelle, la notion de noeuds d'entrée ou de sortie n'existe pas. Afin de simplifier le programme, un seul type de noeud a été mis en place. Le client choisit dans une liste fournie quels noeuds seront empruntés (3 différents). Les noeuds ont donc un rôle majeur au sein du réseau. Ce sont eux qui vont recevoir les messages, les traiter et les renvoyer à l'endroit souhaité. Lorsque l'on parle de traitement, on sous-entend l'action d'enlever la couche in-

utile au message reçu afin d'obtenir une couche inférieure de l'oignon (nouveau message). Une fois ce niveau inférieur découvert, le noeud connaît le destinataire suivant (noeud ou client). Afin de visualiser de manière simple le trajet d'un message au sein du réseau, voici une illustration de la propagation à l'aide des noeuds :



La gestion des messages par les différents noeuds représente un point crucial au sein du réseau Tor. Cela représente également l'une des failles principales. En effet, l'ensemble des noeuds sont mis à disposition par des utilisateurs lambda ou par des associations créées afin de lutter en faveur de la liberté d'expression. Chaque noeud est vérifié par l'association Tor project mais leurs compétences ont également des limites. Il existe, en effet, un certain nombre de failles qui permettent à certains utilisateurs de créer des noeuds espions qui seront référencés comme corrects par le système. L'attaque la plus connue qui a touché le réseau Tor se nomme : Man In The Middle (l'Homme du Milieu). Cette faille consiste à filtrer et analyser le trafic passant par le noeud créé à cette fin. Afin de réduire les risques d'attaques et d'espionnages, il existe une hiérarchie au niveau des noeuds. Cette hiérarchie fait évoluer le noeud au fur et à mesure du temps en filtrant les éventuels noeuds espions. Ce système n'est évidemment pas infaillible mais il permet de réduire la masse de noeuds infectés. Il est à noter que la certification du noeud peut prendre, en moyenne, une septantaine de jours. Pour finir, voici donc les différentes étapes nécessaires à la création d'un noeud :

- Réception du message
- Traitement (+ Timer)
- Envoi à l'étape suivante

## Serveurs

Pour obtenir un aperçu complet de l'architecture du réseau Tor, il nous reste à aborder un dernier point à savoir les serveurs. Ces derniers sont indispensables au bon fonctionnement du réseau puisqu'ils permettent d'enregistrer l'ensemble des noeuds actifs accessibles mais également le statut de ces derniers. Les serveurs Tor sont appelés Directory Authorities et permettent en effet d'apporter un certain contrôle à ce dernier. Dans la pratique, ces serveurs sont au nombre de neuf et sont maintenus par l'association

Tor Project. Ces derniers sont codés en « dur » au sein du réseau et représentent également l'un des points critiques du système. La fonction première de ces serveurs est, dans un premier temps, de maintenir une liste des noeuds accessibles et, dans un second temps, de distribuer cette liste afin de pouvoir obtenir un noeud d'entrée dans le réseau. Au sein de mon implémentation, les Directory Authorities ont été représentées par un unique serveur sur lequel les noeuds s'enregistrent. La complexité et la taille de mon implémentation étant réduite par rapport au projet Tor réel, un unique serveur était suffisant. Une fois qu'un nombre suffisant de noeuds est enregistré (au minimum trois) sur le serveur, les clients peuvent alors obtenir la liste de ces noeuds afin de pouvoir créer un circuit personnel au sein du réseau. Le serveur est donc, dans notre cas, une partie de l'implémentation extrêmement simple. La complexité réelle de ces derniers dans le cas de Tor est bel et bien réelle. Ils doivent notamment faire face à des attaques récurrentes mais également assurer que les noeuds enregistrés sont corrects et légitimes. Le nombre élevés de serveurs est principalement lié à ces deux raisons. Une liste identique est effectivement maintenue par l'ensemble de ces serveurs ce qui permet d'assurer un service permanent du réseau. Pour résumer, voici donc les grands points qui ont été abordés et implémentés afin d'offrir les services des directory authorities :

- Connexion des noeuds
- Enregistrement des noeuds sur le serveur
- Distribution de la liste publique

## Fonctionnement

### Introduction

Nous avons jusqu'à présent détaillé l'architecture interne du réseau Tor. Il est maintenant intéressant de s'attarder sur les mécanismes internes de fonctionnement du réseau. Ces derniers regroupent le mode de transmission des messages, la façon dont sont sécurisées les données ainsi que la création d'un circuit au sein du réseau. L'ensemble de ces aspects seront donc détaillés dans cette nouvelle section. Nous aurons, à la fin de cette seconde partie, détaillé l'ensemble des éléments nécessaires à l'implémentation d'un réseau Tor simplifié à savoir : l'architecture et le fonctionnement de ce dernier. Il est utile de préciser une seconde fois que le réseau implémenté a été volontairement simplifié afin de mettre en lumière les concepts généraux définissant Tor. Afin d'obtenir une structure cohérente, la suite de cette section sera organisée de façon similaire à l'initialisation d'un client au sein de mon projet. Pour ce faire, nous commencerons donc par détailler la création d'un circuit, suivra ensuite l'échange de clés entre le client et les différents noeuds. Nous finirons donc par expliquer l'envoi et la réception de messages entre clients dans le cadre d'une messagerie instantanée.

### Circuits

La création d'un circuit au sein du réseau constitue la première étape du client. Ce dernier commence par récupérer auprès du serveur (l'un des directory authorities) la liste des noeuds accessibles. Une fois cette liste obtenue, le client est donc en mesure de se créer un chemin au sein de l'infrastructure. Dans le cas de Tor, les noeuds sont choisis dans un ordre spécifique. Le client commence par le noeud de sortie (exit node). Une fois ce dernier obtenu, le système choisit alors un noeud d'entrée et de relais. L'ensemble de ces opérations sont effectuées de manière sécurisée par le client mais également de manière transparente pour l'utilisateur lambda. Cependant, il est important de souligner que le choix des noeuds est un des aspects majeurs du réseau. Dans le cas où ces derniers seraient mal choisis, cela pourrait mettre en péril la sécurité de l'utilisateur ainsi que la confidentialité de ses communications. C'est pour cela que le choix des noeuds est effectué dans un ordre spécifique mais également, que les noeuds choisis doivent avoir le moins de liens possibles entre eux. Pour ce faire, Tor privilégie les noeuds se situant à une distance élevée (dans le meilleur des cas le pays doit être différent). Comme dit dans la section précédente, les noeuds de sorties sont un point critique de l'implémentation. On remarque donc également que ces derniers sont choisis en premier et de manière individuelle. Cela met donc bien en évidence les propriétés évoquées précédemment. Pour conclure, il est à noter que les circuits sont des éléments temporaires à intervalle de temps spécifique. Cette mesure de sécurité permet d'éviter qu'un chemin ne soit découvert et mis sous écoute.

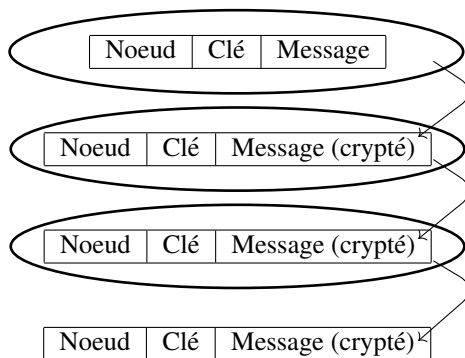
### Messages

L'ensemble des messages échangés sur le réseau Tor sont protégés. Pour ce faire, il faut appliquer un format spécifique à ces derniers qui sera lisible dans l'ensemble des cas par les différents noeuds. Dans le cadre de mon implémentation, ce format a été simplifié. Cependant, l'idée générale est exactement la même dans le projet Tor. Les messages circulants au sein du réseau peuvent donc être vus comme des cellules disposant elles-même de sous-cellules de tailles fixes. Chaque message dispose donc de plusieurs éléments à savoir un message chiffré (ou non dans le cas où l'on se trouve sur le dernier noeud), une destination mais également une éventuelle clé (détaillé dans la section suivante). Afin d'être le plus compréhensible possible, voici, une représentation schématique d'un message :

Noeud	Clé	Message (crypté)
-------	-----	------------------

Comme dit précédemment, la caractéristique principale du routage en oignon est l'encapsulation des messages. Pour ce faire, chaque noeud doit être capable d'effectuer un traitement sur les messages reçus afin de pouvoir obtenir le message suivant. Lorsqu'un noeud reçoit donc un message de la

forme précédente, il en extrait la valeur de la cellule message et la déchiffre à l'aide d'une clé spécifique. Le nouveau message obtenu est du même format que le message précédent et peut être injecté à nouveau sur le réseau vers le noeud suivant. Par souci de facilité, mon implémentation ne prend pas en compte les différents types de noeuds. Cela implique que les noeuds de sorties n'existent pas. Lorsque le message atteint son dernier point relais, celui-ci est redirigé vers un dernier noeud fictif à savoir le client. L'ensemble de ces étapes doivent être effectuées par les différents noeuds. Cependant, afin de pouvoir procéder ainsi, il faut, au préalable, que le client ait envoyé le message correctement. Pour ce faire, il va devoir procéder à la création du message quatre fois. En récupérant à chaque itération le message précédent. Il commencera donc à chaque fois par initialiser le message arrivant au dernier noeud. Voici à nouveau un schéma représentant les 4 itérations :



## Echanges de clés

L'ensemble des messages échangés au sein du réseau Tor doivent être protégés de toutes formes d'écoutes. Pour ce faire, un certain nombre de clés doivent être échangées afin de sécuriser les données transitant sur le réseau. Tor propose deux niveaux de sécurité à savoir : une sécurisation du message ainsi qu'une sécurisation du transport du message. Le procédé mis en place dans le cadre mon implémentation diffère légèrement de l'implémentation réelle de Tor. Cependant l'idée générale qui sera développée dans cette section est belle et bien identique. La sécurisation des données échangées passe, dans le cadre de Tor, par le système TLS. Ce protocole permet de sécuriser les échanges entre deux entités. Il permet également de rendre la lecture des données impossible. TLS (Transport Layer Security) prend en charge un grand nombre de fonctions permettant d'assurer l'intégrité des données mais également la cohérence des informations fournies par le serveur ou les noeuds de passage. Etant donné la complexité et le nombre de fonctionnalités proposées par TLS, le projet réalisé n'a pas été doté de ce dernier. Nous considérerons donc que les données échangées sont correctes et que les noeuds existants disposent des certificats adéquats. Lorsqu'un client initialise

une connexion, il doit, dans un premier temps, échanger avec les différents noeuds des clés permettant de sécuriser et chiffrer les messages échangés. Il existe deux méthodes de cryptage permettant de sécuriser des données :

1. Les clés asymétriques
2. Les clés symétriques

Dans le cadre de ce projet, nous utiliserons les deux ce qui est également le cas dans le projet Tor. Il existe cependant deux grandes différences entre ces dernières. Les clés asymétriques permettent de livrer une clé publique à l'ensemble des personnes souhaitant chiffrer leurs messages. Une fois le message transformé, il est impossible de le retrouver avec la même clé, il est donc nécessaire d'avoir une autre clé, privée cette fois, permettant de retomber sur le message initial. Ce procédé a été extrêmement généralisé grâce à l'utilisation du système RSA. L'avantage indiscutable de ce procédé est bel et bien le fait de pouvoir livrer la clé de manière publique sans risquer de compromettre la sécurité. On pourrait alors s'arrêter là en estimant que la mission est remplie. Ce serait une erreur majeure étant donné la lenteur de ce procédé. C'est donc à ce moment là que l'on fait intervenir les clés symétriques. Ces clés ont pour avantage, face au système asymétrique, leur rapidité bien plus élevée mais doivent cependant rester secrètes durant les transmissions. Etant donné les avantages et inconvénients développés ci-dessus, on peut donc en déduire qu'une combinaison des deux techniques pourrait être extrêmement avantageuse. On va donc utiliser les forces des deux systèmes afin de créer un système résultant parfaitement robuste. Pour ce faire, les clés asymétriques sont donc utilisées dans un premier temps. Ces dernières permettent d'initialiser les échanges entre le client et les différents noeuds choisit. Une fois la connexion établie, les communications seront alors protégées à l'aide des clés symétriques afin de pouvoir augmenter la rapidité du système.

**Asymétriques** Lorsque les noeuds s'initialisent, ils envoient alors leurs informations au(x) serveur(s). Au sein de ces informations réside, l'adresse et le port de connexion mais également la clé publique. Lorsqu'un client souhaite initialiser un circuit, il va alors se connecter en cascade aux différents noeuds. Lors de la première connexion, le client crée un premier message sous la forme d'un oignon. A chaque couche (3) il insère une clé symétrique générée. Chaque clé symétrique est donc protégée par la clé asymétrique correspondant au noeud adéquat. Pour visualiser de manière correcte la façon dont ces clés sont imbriquées, il suffit de reprendre le schéma utilisé précédemment. Lorsqu'un noeud reçoit le premier message d'une connexion, il va donc enregistrer la clé symétrique correspondante afin de pouvoir échanger de manière plus rapide. Une dernière précision concernant l'implémentation réelle du système peut être apportée. Précédemment, il a

été dit que l'implémentation réalisée ne connaissait pas les noeuds de sortie, ceci est uniquement partiellement vrai. En effet, chaque noeud, lorsqu'il reçoit sa clé symétrique, vérifie s'il est utile d'envoyer le message une étape plus loin. Dans le cas où ce n'est pas nécessaire, le message est mis à null. Le noeud est donc capable de savoir s'il est oui ou non le dernier maillon de la chaîne.

**Symétriques** Une fois les clés échangées, le système peut donc utiliser la technique des clés symétriques. Ce procédé, comme dit précédemment, est beaucoup plus rapide mais également plus complexe à la mise en oeuvre. Pour des raisons de sécurité, le chemin emprunté par les messages est changé à un intervalle de temps régulier. Ce changement nécessite à nouveau l'échange de clés symétriques. Dans le cadre de mon projet, les clés utilisées sont des clés AES. Le projet Tor, utilise quant à lui, le procédé de Diffie-Hellman ce qui ne modifie pas grandement la manière de les utiliser. Cependant, la méthode utilisée afin d'échanger des clés Diffie-Hellman introduit certaines nuances non utilisées par mon implémentation. Afin d'utiliser ce procédé, il faut dans un premier temps définir un nombre premier  $p$  et un nombre aléatoire  $g$  inférieur à  $p$ . Une fois cette première étape effectuée, le client peut envoyer une clé  $C$  au serveur obtenu de la manière suivante avec un nombre  $c$  aléatoire :  $C = g^c \bmod p$ . Une fois que le serveur a fait de même, ils peuvent tous deux calculer la clé secrète de la façon suivante (exemple pour le client) :  $S^c \bmod p$  (en sachant que  $S$  est la clé envoyée par le serveur).

## Implémentation

Comme expliqué à plusieurs reprises dans ce rapport, un projet a réellement été implémenté afin de comprendre concrètement les techniques utilisées dans le réseau Tor. L'architecture de ce projet reprend réellement l'idée du réseau Tor et permet de se faire une idée réelle de la manière dont les concepteurs ont imaginé ce projet. L'implémentation fournie en complément consistait en la réalisation d'une messagerie instantanée entre deux utilisateurs. Afin de pouvoir réaliser de la meilleure façon possible les tests, le projet s'exécute à l'heure actuelle en local. Cependant, il est tout à fait possible de changer les adresses de connexion afin de pouvoir l'exécuter sur plusieurs périphériques. Afin de pouvoir utiliser le réseau, il faut lancer au minimum (et dans l'ordre) un serveur principal permettant de récupérer les informations de connexion, trois noeuds (ou plus) permettant de jouer le rôle de passerelles et deux clients. Il est évident que le projet peut être amélioré ou complexifié, toutefois il représente à l'heure actuelle une schématisation idéale du réseau et peut être utilisé comme illustration des différents concepts évoqués ci-dessus. L'ensemble du projet a été réalisé en Java et peut donc être porté sur toute machine ayant le système installé. Le choix du langage, bien que lourd et pas forcément

optimisé, a été encouragé par l'ensemble des possibilités offertes. En effet, Java est un langage offrant un certain nombre de facilités lors de l'échange de messages. A titre d'exemple, le transfert de messages est optimisé puisqu'il est possible à travers un socket de faire passer tout type d'objet. On est donc capable de transmettre des informations formatées et de les récupérer facilement. Il est évident que d'autres langages auraient pu être tout aussi compétitifs.

## Aspect social

Tor est un sujet récurrent et omniprésent dans les médias. Sa popularité est souvent liée à des histoires ayant un lien direct avec la justice. Ma volonté première était de montrer les capacités ainsi que les technologies qui ont été mises en oeuvre pour développer ce réseau. Tor est l'un des réseaux les plus sécurisés disponible pour le grand public. Ce dernier permet également à certains défenseurs des droits de l'homme, de la liberté d'expression ou de toute autre forme de liberté de pouvoir s'exprimer mais également de pouvoir communiquer à travers une plateforme ne subissant aucune forme de censure. Il joue donc encore à l'heure actuelle un rôle important dans la politique de certains pays. Plusieurs groupes ou associations ont donc pour but de maintenir en place ce réseau. Parmi eux, on compte notamment le groupe d'activistes Anonymous ou encore l'association No Ignorance.

## Conclusion

Ce document permet donc de mettre en lumière les techniques utilisées au sein de Tor afin de sécuriser de protéger les données utilisateur. Pour ce faire, l'utilisation de noeuds intermédiaires est indispensable et permet de bloquer certains traqueurs visant à suivre un paquet selon sa destination. Parallèlement à cet aspect, chaque message transitant par le réseau est équipé d'une sécurité de haut niveau ne permettant pas à une personne qui souhaiterait accéder au contenu d'échanger de le faire. Comme dit précédemment, l'utilisation d'un tel système permet donc de garantir l'accès à la liberté d'expression et l'utilisation d'internet de manière absolue. Cependant, le réseau possède bel et bien certaines failles. En effet, comme nous l'avons vu, chaque information transitant via le réseau est chiffré à l'exception des données arrivant auprès du serveur destinataire qui lui est normalement capable de comprendre les requêtes. C'est pourquoi, une identification auprès d'un site internet ou autre supprimerait, toute garantie de sécurité et de confidentialité. Dans le contexte actuel, l'anonymat proposé par ce réseau est un point extrêmement important et c'est probablement ce point qui lui vaut son succès croissant. Nous pouvons donc conclure que le réseau détaillé dans ce document, est un système extrêmement complexe et basé sur une association de techniques parfois opposées mais pouvant être associées afin d'optimiser et d'accroître la sécurité.

## References

- The Tor Project. Overview, Avril 2017. <https://www.torproject.org/about/overview.html.en>.
- The Tor Project. People, Avril 2017. <https://www.torproject.org/about/corepeople.html.en>.
- Roger Digledine. First onion proxy, Septembre 2002. <http://archives.seul.org/or/dev/Sep-2002/msg00019.html>.
- Wikipedia. Tor, Avril 2017. [https://en.wikipedia.org/wiki/Tor\\_\(anonymity\\_network\)](https://en.wikipedia.org/wiki/Tor_(anonymity_network)).
- Pierre Rouchon. Cryptographie et theorie des nombres, Octobre 2009. <http://cas.ensmp.fr/~rouchon/MinesCrypto/CryptoPR09.pdf>.
- J Nechvatal, E Barker, L Bassham, W Burr, M Dwor-kin, J Foti, and E Roback. Report on the development of the advanced encryption standard (aes), Octobre 2000. <http://csrc.nist.gov/archive/aes/round2/r2report.pdf>.
- Christophe Grenier. Techniques de cryptanalyst de rsa, Janvier 2009. [ftp://ftp.irisa.fr/local/caps/DEPOTS/BIBLIO2009/Grenier\\_Christophe.pdf](ftp://ftp.irisa.fr/local/caps/DEPOTS/BIBLIO2009/Grenier_Christophe.pdf).
- A.Shamir R.L.Rivest and L.Adleman. A method for obtaining digital signatures and public-key cryptosystems, Fevrier 1978. <https://people.csail.mit.edu/rivest/Rsapaper.pdf>.
- Houda FERRADI. Introduction a la cryptographie (cours 3) : Chiffrement par bloc (des), Fevrier 2016. <http://www.di.ens.fr/~ferradi/coursDES.pdf>.
- Houda FERRADI. Introduction a la cryptographie (cours 4) : Chiffrement par bloc (aes), Fevrier 2016. <http://www.di.ens.fr/~ferradi/coursAES.pdf>.