Title

Nathan Liccardo [1]

nathan.liccardo@ulb.ac.be

Abstract

Dans le cadre de mon mini-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 en lui-même. 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.

Introduction

Le système Tor est un réseau permettant aux personnes l'utilisant d'échanger des messages ou des informations de manières anonyme. Dans le cadre de ce travail, je vais tenter de montrer comment fonctionne ce réseau. Pour ce faire, j'utiliserai une implémentation personnelle permettant d'illustrer les différents concepts définissant ce système. Au travers de mon implémentation, je parcourerai les notions de noeuds, clients et serveurs mais également le type de cryptage appliqué sur les messages transmis au travers de ce réseau. Il est donc important de différencier les deux parties majeures définissant la conception du réseau Tor à savoir la partie architecture (noeuds, serveurs et clients) et la partie sécurité (Cryptographie). Lors du détail de ce second point on abordera donc bien les notions de routage en oignon et de cryptographie symétrique et asymétrique. Ce principe est en effet le coeur de la sécurité du réseau Tor et permet d'obtenir une sécurité très élevée. Comme dit précédemment, la sécurité de Tor est très évoluée mais reste tout de même franchissable en certains points.
L'intérêt pour Tor m'a été porté du à son importance dans le milieu de l'informatique mais également pour son aspect social. En effet, nous vivons dans une époque ou la sécurité et la protection des données est un point extrêmement important pour bon nombre de personnes. L'utilisation de Tor fait couramment l'objet d'une exposition médiatique importante et pas forcément positive. C'est pourquoi il me paraissait intéressant de me pencher sur l'architecture et la technologie présente au sein de ce réseau.

Le système Tor

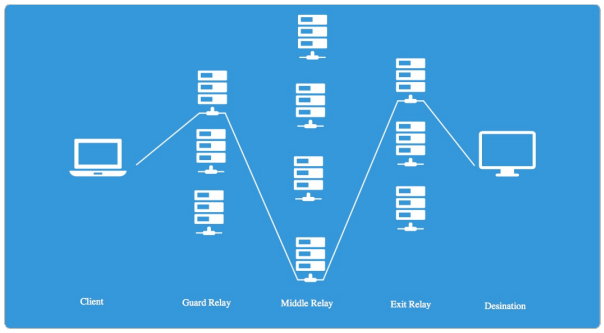
Le réseau Tor est une plateforme permettant une anonymisation totale des messages ou informations transférés entre deux clients au travers de ce réseau. Par souci de simplification, dans la suite de ce rapport je parlerai uniquement de paquets lorsque je souhaiterai évoquer un transfert d'informations. En effet, Tor permet d'échanger tout types d'informations (messages, site web, ...) mais ces informations sont toutes considérées de la même manière. On peut donc voir ce réseau comme un intermédiaire permettant de chiffrer et de protéger la vie privée des individus l'utilisant. Tor fonctionne sur base de trois éléments distincts à savoir : les clients, les noeuds et les serveurs.

Client

Les clients constituent l'une des couches les plus simple du système. Ce sont eux qui vont se connecter au réseau afin de bénéficier du plus d'anonymat possible lors de l'échange de paquets avec un autre client ou un service internet quelconque. Ces derniers peuvent être considéré comme des électrons libres externes au réseau. En effet, leur implication au sein de ce dernier est minime, ils ne seront donc à aucun moment considéré comme éléments interne de l'infrastructure Tor.

Noeuds

L'un des points les plus importants du réseau Tor réside bel et bien dans son utilisation de noeuds. Ce sont eux qui permettent d'établir une ou plusieurs sécurités supplémentaires permettant à l'utilisateur de rester inconnu lors de son utilisation d'internet. Dans de nombreuses documentations, les noeuds sont vus comme des proxy permettant à l'utilisateur de passer par plusieurs ponts avant d'obtenir le résultat de se requête. Au sein du réseau Tor, il existe donc trois types de noeuds différents qui seront vu dans la suite de ce documents. Ces trois noeuds sont appelés Guard, Bridge et Exit. Il représentent chacun une fonctionnalité du réseau spécifique.



Guard Ce premier type de noeud représente la porte ouverte du circuit. Il est effet le lien direct entre l'utilisateur et le réseau Tor. Ce type de noeud est présent sur une liste publique et est accessible librement.

Bridge Ce seconde type de noeud constitue la deuxième étape du circuit. Ils sont quant à eux absolument non référencés et uniquement présent ou accessibles à partir du réseau Tor en lui-même.

Exit Ce dernier type de noeud est l'un des maillons les plus important de la chaine. Il s'agit en effet ici de noeuds de confiance ayant également leur IP publique. Lorsque des messages sont échangés, seul ce dernier noeud est visible par l'interlocuteur se trouvant à l'autre bout de la ligne. Il est à noter que le noeud de sortie peut être le plus vulnérable étant donné son exposition.

Serveurs

Afin de détailler de manière complète l'architecture du réseau, il nous reste un élément majeur à mettre en évidence. Pour que les clients puisse joindre ce dernier, il faut qu'ils puissent trouver un lien entre eux et les noeuds d'entrées. Ce lien est effectué par ce que l'on appelle les « directory authorities ». Ces derniers sont au nombre de dix et sont des noeuds comprenant une liste identique des noeuds accessibles. L'ensemble des serveurs sont codés « en dur » au sein du programme Tor. Ils sont donc connus de tous systèmes utilisant le programme. La mise à mal d'une grande partie de ces serveurs pourrait causer un grand problème au sein du réseau. il s'agit donc de l'un des points sensible de Tor.

La sécurité

Nous avons pour le moment détaillé l'architecture dite générale du réseau Tor. Cette architecture peut être vue comme une succession de ponts ou de relais permettant de perdre les requêtes des différents utilisateurs au sein d'un sous-réseau. Cependant, l'ensemble des éléments vus jusqu'à présent n'introduisent en aucun cas une quelconque sécurité concernant les données de l'utilisateur qui sont, en l'état actuel des choses toujours accessible par un individu effectuant certaines recherches plus approfondies. Dans cette seconde partie nous verrons donc l'ensemble des techniques utilisées par Tor afin de pouvoir échanger des données utilisateur au sein du réseau tout en empêchant une consultation de ces dernières et donc un espionnage des requêtes ou des paquets envoyés/reçus par l'utilisateur. Afin de protéger ses données, Tor utilise donc un chiffrement qui permet de rendre incompréhensible les paquets transitant par un noeud quelconque. En pratique, le réseau applique une sécurité supplémentaire pour chaque noeud traversé lors de l'utilisation de ce dernier. Etant donné qu'un paquet transite par trois noeuds distincts, trois couches de sécurité sont donc appliquées sur le paquet. A chaque interception de paquet par un noeud, ce dernier enlève la sécurité appliquée sur ce dernier et envoie le paquet à l'adresse « découverte » en enlevant cette dernière. Cette technique vaut à Tor l'utilisation de l'onion routing. Dans cette nouvelle section, nous verrons donc quel sont les deux techniques de cryptographie existante et pourquoi ces dernières sont toute deux utilisées au sein du réseau Tor.

Techniques de chiffrement

Il existe donc deux types de protection des données. Ces dernières émanent toutes deux de techniques de cryptographie. La principale différence entre ces deux techniques résident donc bel et bien dans leurs performances. En effet, le chiffrement dit symétrique est beaucoup plus performant que son « rival » à savoir le chiffrement asymétrique.

Chiffrement symétrique

Dans ce premier cas, on utilise des techniques reposant sur l'échange ou le partage d'une et une seule clé secrète. On établit donc que les deux interlocuteurs disposent d'une clé permettant de chiffrer et de déchiffrer les messages. Cette méthode dispose d'un atout non négligeable à savoir sa rapidité d'exécution contrairement aux techniques que nous verront dans la suite de ce document. Néanmoins, le désavantage majeur de cette solution est le partage de la clé secrète. Il faut en effet que les utilisateurs souhaitant communiquer arrivent à se partager la clé de manière totalement sécurisée. Il est intéressant de noter que ce type de sécurité est la plus ancienne des méthodes de sécurisation de données.

Chiffrement asymétrique

Dans un second temps, on s'intéressera à une deuxième technique de chiffrement à savoir la méthode asymétrique. Ce second procédé permet de palier certains problèmes introduits par le chiffrement symétrique. Cependant, la technique asymétrique est extrêmement plus lente (plusieurs ordres de grandeurs) comparé à la méthode symétrique. C'est pourquoi, ces deux procédés seront utilisés de manière différentes et combinés afin d'obtenir des rapidités optimales d'exécution. Il a précédemment été annoncé que cette nouvelle technique de cryptage permettait de palier aux problèmes introduits par la méthode symétrique. En effet, lors de l'utilisation de cette dernière, il est obligatoire que les deux utilisateurs disposent de la clé secrète ayant été préalablement distribuée de manière totalement sécurisée. C'est précisément ce point de faiblesse que la méthode asymétrique vient résoudre. En utilisant ce nouveau procédé, on introduira deux types de clés à savoir : la clé privée (gardée secrète par la personne devant recevoir les messages chiffrés) et la clé publique (distribuée de manière publique et utilisée par chaque personne souhaitant communiquer).

RSA A l'heure actuelle, l'une des méthode de chiffrement asymétrique les plus connues et répandue est la technique RSA. C'est également cette méthode qui est utilisée dans le programme Tor afin de mettre en contact deux noeuds indépendants. Afin d'obtenir une explication claire et simplifiée de la manière dont fonctionne le système RSA, je vais commencer par donner les grandes étapes nécessaires afin de mettre en place ce système :

- Choix de deux entiers premiers **p** et **q** de grande taille
- Recherche d'une valeur **n** première avec **(p-1)*(q-1)**
- Publication de la clé **(RSA,n,e)**

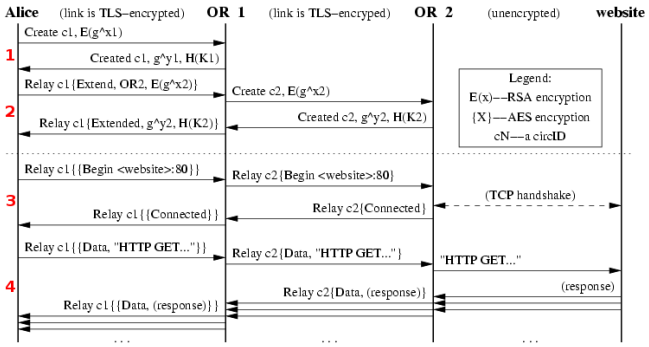
Une fois ces données obtenues, on doit savoir comment chiffrer et déchiffrer le message. Pour ce faire on utilise les deux formules suivantes :

- Chiffrement : $C = B^e \ mod(n)$
- Déchiffrement : $B = C^d \ mod(n)$

Pour le déchiffrement on aura besoin d'une valeur en plus à savoir **d**. Cette valeur est obtenue de la manière suivante : **e * d mod ((p - 1) * (q - 1)) = 1**.

Utilisation dans le réseau Tor

Dans les réseau Tor, on a une combinaison des système symétriques et asymétriques. En effet, étant donné la grandeur du réseau, il est pratiquement impossible de communiquer de manière sécurisée l'ensemble des clés privées. Pour palier à ce problème, Tor utilise dans un premier temps un système asymétrique afin d'établir une connexion avec un certains noeud et de lui partager une clé symétrique qui sera utilisée dans le reste des échanges. On peut visualiser le système de la façon suivante (les noeuds suivants sont sélectionnés par chaque noeud courant) :



Dans le cadre de cet exemple, on voit que deux algorithmes différents de chiffrement ont été utilisés à savoir :

- RSA : chiffrement asymétrique
- AES : chiffrement symétrique

Le schéma ci-dessus permet d'exprimer de manière non formelle le fonctionnement du réseau Tor et de comprendre comment fonctionne de façon exacte le système.

Implémentation

Afin de comprendre au mieux le réseau Tor et de visualiser de manière efficaces, les différents points abordés dans la section précédente, il était intéressant d'implémenter un réseau (sous forme locale dans un premier temps) Tor exécutant les fonctionnalités principales de ce dernier. L'architecture appliquée a donc été calquée sur celle utilisée dans le réseau décrit précédemment à savoir : clients, noeuds, serveurs. Chacune des ces parties a été développée au sein d'un projet indépendant afin de préserver l'aspect de division présent au sein de la réalité.

Serveurs

Le premier point de mon implémentation repose sur la création du serveur. Dans le système présenté ci-dessous, un seul serveur sera mis en route ce qui contrairement au système Tor complet sera moins robuste confronté à de multiples attaques. Ce serveur fonctionnera donc d'une façon assez similaire au « directory authorities ». Afin d'obtenir une représentation assez similaire à la réelle, le serveur enregistre l'ensemble des noeuds connectés au réseau et étant accessibles pour l'ensemble des clients. Il existe donc deux types de connexions au serveur principal à savoir une connexion en tant que noeud (enregistrement de données) et une connexion en tant que client (récupération des données). Le serveur permettant des connexions simultanées, une fois que l'un des clients se connecte, on procédera à la création d'un Thread permettant de récupérer les informations nécessaires (ou de les envoyer) tout en gardant la possibilité de se connecter au serveur.

Nodes Le serveur possède en son sein une représentation générale d'un noeud. Cette représentation générale permet de transféré au sein des messages l'objet noeud en tant que

tel et donc par la même occasion de simplifier les échanges. Cette représentation est extrêmement simple et n'est composée que des informations générale d'un noeud à savoir : son adresse IP, son port de connexion ainsi que sa clé RSA publique.

Réception La première fonction proposée par le serveur est la connexion d'un noeud. Dans ce premier cas, le serveur va donc récupérer les informations directement auprès du noeud récemment connecté afin de pouvoir l'ajouter à son annuaire. Lors de l'ajout de ce dernier, le serveur crée donc un objet du type précédent en y ajoutant l'adresse, le port et pour finir la clé RSA publique.

Envoi La seconde fonction proposée par le serveur, une fois que plusieurs noeuds ont été mis en place, est de mettre à disposition des clients les noeuds connectés. Pour ce faire, le serveur va simplement envoyé la liste des ≪ objets ≫ noeuds connectés afin que l'utilisateur puisse s'en servir.

Les clients

Dans le point précédent, nous avons déjà abordé de manière superficielle les clients ainsi que leurs interactions avec le serveur. Nous pouvons à présent détailler la manière dont ils fonctionnent exactement ainsi que leur architecture. Lorsque le client démarre, il commence par vouloir se connecter au réseau Tor, pour se faire, il devra dans un premier temps récupérer la liste des noeuds auxquels il peut se connecter. C'est pourquoi, dans un premier temps, le client se connectera au serveur (unique dans notre cas) et récupérera l'ensemble des noeuds enregistré sur ce dernier. A ce stade du développement, le client possède donc une liste de noeuds auxquels il peut se connecter. Pour se faire, il utilisera les clés RSA publiques présentes dans chaque structure de noeud. Etant donné que notre système est un cas simplifié de la vie réelle, une simplification a été apportée au mode de fonctionnement. En effet, afin de faire fonctionner le système, chaque utilisateur devra connaître l'adresse IP ainsi que le port sur lequel il pourra joindre son interlocuteur. Pour finir, il est à noter que chaque utilisateur choisira le chemin emprunté par son message (succession de trois noeuds distincts).

Nodes De manière similaire au serveur, chaque client dispose d'une structure de noeud interne et identique. Cette structure a été implémentée afin de pouvoir communiquer de manière simple les informations concernant les noeuds entre le client et le serveur.

Réception et envoi Le client, après s'être connecté au serveur afin de récupérer les noeuds, créera deux Thread permettant d'exécuter de manière parallèle la réception et l'envoi de message. En effet, ce dernier doit être capable d'envoyer un message (création du chemin, connexion aux noeuds, chiffrement du message) tout en restant capable de recevoir un message et d'afficher ce dernier.

Chiffrement Nous verrons dans la suite de ce document comment sont appliquées les mesures de sécurités sur les messages. Il est tout de même intéressant à ce stade-ci de préciser que l'ensemble des ces opérations seront faites au sein du client. Les messages partiront donc de manière sécurisée.

Les noeuds

Afin de cloturer la partie architecture de ce projet, il est maintenant nécessaire de se pencher sur la réalisation des noeuds intermédiaires. Ces derniers devront donc être capable, dans un premier temps, de se connecter au serveur afin d'envoyer les informations nécessaire à la connexion. Une fois cette première étape réalisée, une clé privée RSA aura été générée et pourra donc être utilisée. Le noeud passe donc d'un statut actif (connexion au serveur) à un statut passif (attente de connexion de clients ou noeuds relais). Une fois qu'un message est reçu par le noeud, ce dernier devra dans un premier temps l'analyser afin d'en faire ressortir les informations nécessaires. Les informations majeures dont le noeud aura besoin seront directement liées au noeud suivant. Il doit en effet connaître l'adresse de connexion de la prochaine passerelle ou du client final.

Connexion serveur On observera donc dans un premier temps une connexion auprès du serveur principal afin d'enregistrer le nouveau noeud au sein de l'annuaire publique. Afin de distinguer un client d'un noeud, ce dernier enverra une valeur spécifique (et propre au noeud) lui permettant de s'identifier comme tel auprès du serveur. Une fois cette étape faite, le serveur en déduira qu'il doit récupérer les informations de connexion.

Attente de message Une fois la connexion faite, le processus principal du noeud créera deux thread permettant l'écoute et l'envoi de message. Dans un premier temps, le thread d'écoute permettra donc de recevoir un message chiffré sur le port correcte du noeud. Ce dernier sera analysé par la partie cryptographie (c.f. fin du document) qui renverra un nouveau message comportant l'adresse du noeud suivant. Une fois cette étape effectuée, on passera donc au second thread à savoir le thread d'envoi.

Envoi du message Dans ce second thread, les messages seront envoyés au noeud ou client suivant. Il est tout de même intéressant de relever qu'un élément ≪ node ≫ a également été implémenté au sein de ce dernier élément afin de pouvoir utilisé les données de connexion à la passerelle suivante de la manière la plus simple possible.

Sécurité

Jusqu'à présent, l'aspect sécurité a été survolé volontairement au sein de chaque explications de l'architecture du programme. Pour pouvoir comprendre le mode de fonctionnement il est intéressant de s'attarder de manière plus générale à la façon dont cette dernière est gérée. La volonté première lors de la réalisation de ce projet était de mettre en évidence l'utilisation de la structure en oignon mais également l'utilisation des clés symétriques et asymétriques. Cette dernière partie expliquera donc la manière dont sont gérés ces trois techniques au sein du réseau. J'aborderai donc dans un premier temps la notion de routage en oignon ainsi que l'application de ce dernier sur les messages transmis. Une fois que ce premier point aura été abordé, je passerai donc au point concernant le chiffrement.

Routage en oignon Comme expliqué précédemment, le système Tor repose sur un échange de paquets sous forme de couches. Pour ce faire, l'ensemble des paquets sont encapsulé dans des paquets plus grands et ainsi de suite. Pour ce faire, lorsque le client souhaite envoyer un message à un autre client, il va devoir dans un premier temps sélectionner un circuit valide au sein de la liste reçue et dans un second temps encapsuler son message. La création de l'≪ oignon ≫ se fait donc étape par étape. L'étage le plus bas sera constitué du message accompagné de l'adresse du destinataire. Une fois ce premier étage constitué, on appliquera la sécurité correspondant au dernier noeud pour le chemin sélectionné. On arrive donc à un nouveau message n'ayant, déjà à ce stade-ci, plus aucune signification. Ce procédé sera donc appliqué pour chaque noeud sélectionné en cascade. Afin de défaire la sécurité au fur et à mesure, chaque noeud enlèvera sa couche de protection ce qui permettra au final au destinataire de recevoir le message en clair. Afin d'obtenir une explication complète, voici la méthode détaillée du chiffrement :

— Chiffrement pour le dernier noeud et ajout de son adresse (message 1)
— Chiffrement pour le noeud du milieu et ajout de son adresse (message 2)
— Chiffrement pour le noeud un et ajout de son adresse (message 3)

Chiffrement asymétrique Afin de pouvoir contacter dans un premier temps chaque noeud, le client utilisera les clés asymétrique grâce auxquels il enverra la clé symétrique. Le chemin est donc parcouru dans un premier temps afin d'établir la liaison entre le client et tous les noeuds utilisés lors de son chemin. Cette étape est faite avant d'envoyer le premier message. Il est à noter que dans le réseau Tor, chaque chemin change environ toutes les cinq minutes ce qui permet d'augmenter la sécurité du système.

Chiffrement symétrique Dans un second temps, une fois que les connexion sont établies, les messages seront transmis via une clé symétrique. Ce procédé permet donc de gagner un temps considérable lors du déchiffrage du message. Dans certains points précédents on parlait de la rapidité d'exécution entre les deux systèmes. On voit donc maintenant pourquoi il est intéressant de combiner ces deux derniers ainsi que d'en exploiter les atouts indépendants.

Aspect social

Tor est un sujet récurent et fortement présent dans les médias. Sa popularité est assez souvent liée à des histoires douteuse ou ayant un lien direct avec la justice. Ma volonté première au sein de ce document était donc de montrer les capacités mais également la technologie mise en place derrière ce réseau. Tor est en effet l'un des réseau les plus sécurisé à l'heure actuelle qui soit disponible pour le grand publique. Ce dernier permet également à certains défenseur des droits de l'homme, de la liberté d'expression ou de tout autre forme de liberté de pouvoir s'exprimer mais également de pouvoir communiquer à travers une plateforme ne subissant aucune forme de censure.

Conclusion

L'ensemble de 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 de part sa destination. En plus de ce point important, chaque message transitant par le réseau est équipé d'une sécurité de haut niveau ne permettant pas à une personne qui souhaiterais accéder au contenu échanger de le faire. Comme dit précédemment, l'utilisation d'un tel système permet donc bien de donner accès à la liberté d'expression et d'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 supprimerais, comme nous l'avons vu toute forme de sécurité et de confidentialité. Dans les temps actuels, 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 ressortir du document présent que le réseau expliqué est un système extrêmement complexe et basé sur une association de techniques parfois opposées mais pouvant être associées afin d'optimiser et d'accroitre la sécurité proposée.

Remerciements

Ce travail a été supervisé par monsieur François Gérard. Je tient à le remercier pour son aide durant la rédaction de

ce mémoire.

Sources

https://blog.imirhil.fr/2015/09/22/installer-noeud-tor.html

http://igm.univ-mlv.fr/~dr/XPOSE2009/TOR/fonctionnement.html

https://www.torproject.org/docs/

http://jordan-wright.com/blog/2015/02/28/how-tor-works-part-one/

https://fr.wikipedia.org/wiki/Cryptographie_symetrique

http://www.nymphomath.ch/crypto/rsa/index.html