Introduction

1. Protocole FTP
2. Generalite
3. Historique
4. Modele FTP
5. Fonctionnement du FTP
6. Presentation des donnees
7. Etablissement du canal
8. Mode de transmission
9. Recuperation des erreurs et retransmission
10. Fonction de transfert des fichiers
11. Les commandes FTP
12. Les reponses FTP
13. Installation et configuration du FTP
14. Installation
15. Configuration
16. Les limites du FTP
17. Protocole SSH
18. Generalite sur le SSH
19. Methode de chiffrement
20. Chiffrement symetrique
21. Chiffrement asymetrique
22. Fonctionnement du SSH
23. Installation et configuration du ssh
24. Sécurisation du transfert des fichiers avec le SSH
25. Client
26. MySecureShell
27. Protocole FTP
28. Historique

La première proposition de mécanisme de transfert de fichiers a été faite 1971 et avait été développée pour une application sur les hôtes du MIT (*Massachusetts Institute of Technology*). MIT-Project MAC, en juin 1971 proposait un protocole de niveau utilisateur pour le transfert de fichiers entre ordinateurs (y compris des terminaux IMPs). Une révision de celui-ci redonnait un état du FTP pour évolution ultérieure, tandis que la BBN en décembre 71 suggérait encore d'autres modifications. L'usage d'une transaction "Set Data Type" a été proposée par MIT-Project MAC en Janvier 1982. Le File Transfer Protocol était désormais défini comme un protocole de transfert de fichiers entre des hôtes d'un ARPANET, et dont la fonction première était définie comme le transfert efficace et fiable entre des hôtes pour profiter de l'utilisation d'une capacité de stockage de données distante. Certains correctifs à certaines erreurs, certains points, et notions au protocole ont été ajoute dans les années 72.

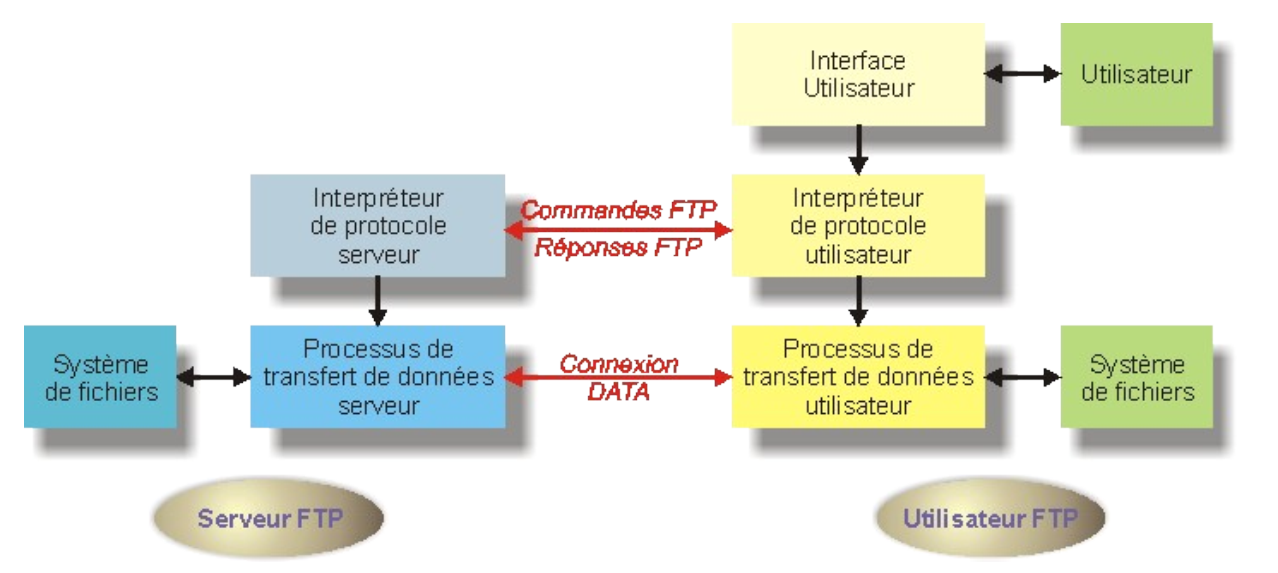
Depuis Juillet 1973, des changements considérables sont intervenus, mais la structure globale est restée la même. BBN a été publié comme une nouvelle spécification "officielle" pour refléter certains changements. Cependant, de nombreuses implémentations basées sur l'ancienne spécification n'étaient pas remises à jour. En 1974, les BBN et UCSB ont apporté de nouveaux commentaires à propos de FTP. Motivée par le passage du NCP (Network Communication Protocol) à TCP comme protocole sous-jacent, un phœnix est né de nouveau à partir de tous les efforts ci-dessus en 1980 comme une nouvelle spécification de FTP basée sur le protocole réseau TCP. Cette édition de la spécification FTP est écrite pour corriger quelques erreurs par la suite, tout en étendant les explications de certaines fonctionnalités du protocole, et enfin en ajoutant la définition de quelques commandes supplémentaires.

1. Principe et fonctionnement du FTP
2. Principe de base du FTP

Les objectifs de FTP sont multiples parmi lesquels on :

* Promouvoir le partage de fichiers (programmes informatiques et/ou données)
* Faciliter l'utilisation indirecte ou implicite (via des programmes) d'ordinateurs distants
* Prémunir l'utilisateur contre les variations de formats de stockage de données entre les différents hôtes
* Transférer les données d'une façon efficace et fiable. FTP, bien que directement utilisable par un utilisateur depuis un terminal, est néanmoins conçu essentiellement pour être utilisé par des programmes.

FTP, bien que directement utilisable par un utilisateur depuis un terminal, est néanmoins conçu essentiellement pour être utilisé par des programmes. Son principe de fonctionnement est le suivant :

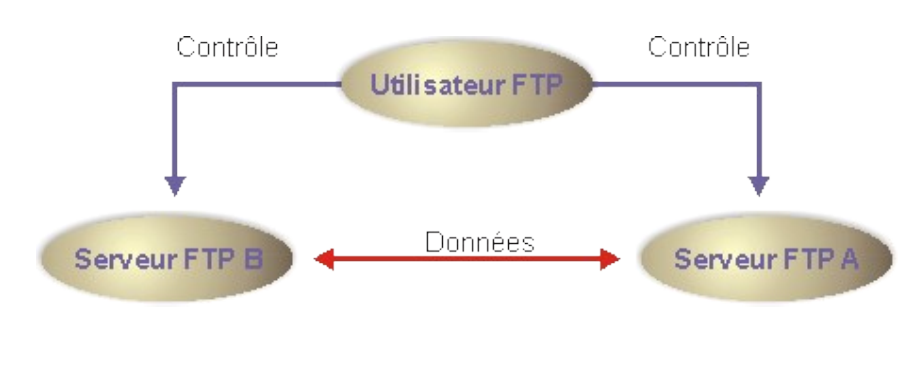
Le client ouvre une session FTP sur un serveur. Il existe une grande quantité de serveurs FTP  
publics. Un serveur FTP requiert une identification du client. Il existe souvent un compte  
"anonyme", qui donne accès en lecture seule dans la partie publique du serveur, mais il existe  
également des parties privées où les clients disposant d'un compte peuvent accéder en écriture sur certains répertoires de l'arborescence. C'est le cas, par exemple, pour les mises à jour de pages web personnelles. 

*Fig1: Fonctionnement du protocole FTP.*

La première chose que l'on constate, c'est que, contrairement à d'autres protocoles comme HTTP,  
nous allons ici utiliser deux canaux distincts :

* L'un pour l'échange des commandes du protocole ;
* l'autre pour le transfert des données elles-mêmes.

Le client FTP (partie de droite), par l'intermédiaire de l'interface utilisateur, va cacher les diverses commandes du protocole FTP par des manipulations plus conviviales, en proposant à l'utilisateur une vision des choses similaire à un gestionnaire de fichiers. Avec des clicks et des  
"glisser/déposer" l'utilisateur exploitera FTP sans en connaître la multitude de commandes. Mais  
sachez qu'avec beaucoup de courage et de connaissance du protocole, vous pourriez utiliser Telnet pour faire du FTP.

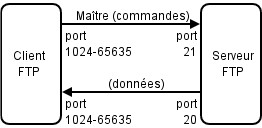


*Fig2 : Transfert de fichiers entre serveurs FTP distant.*

Un utilisateur pourra exploiter FTP pour transférer depuis son poste de travail des fichiers d'un serveur distant à un autre serveur distant, sans que les données ne transitent par sa machine, ce qui est fort intéressant si l'on travaille depuis une connexion RTC pour passer des données d'une machine à une autre, ces dernières étant, elles, connectées par des liens à haut débit. Cependant, cette opération  
ne sera possible que si les serveurs FTP l'acceptent, ce qui n'est pas souvent le cas, pour des raisons de  
sécurité. Après avoir longuement tourné autour du pot (et avoir relu plusieurs fois la RFC 9593), j'ai  
finalement pensé que le meilleur moyen pour comprendre FTP n'était certainement pas la lecture de  
cette RFC mais plutôt l'expérimentation. Nous allons donc mettre en œuvre FTP, voir comment ça se passe et vérifier seulement après que c'est bien conforme à ce qui est dit dans les Livres. Les manipulations sont faites depuis un poste client Windows connecté à un LAN, lui-même connecté à l'Internet par une passerelle NAT GNU/Linux. Un sniffeur est placé sur le poste Windows lui-même, il aurait pu l'être sur la passerelle.

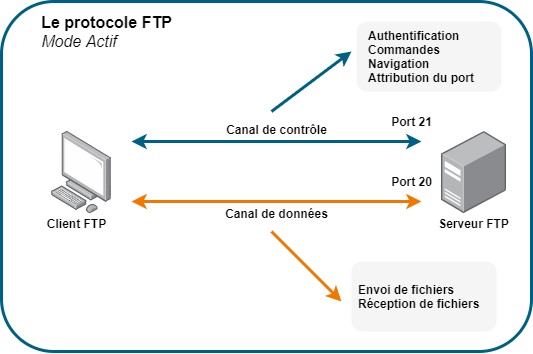
1. Fonctionnement du SSH

Comme le montre les figures précédentes, le protocole FTP obéit à un modèle client-serveur, c'est-à-dire qu'une des deux parties, le **client**, envoie des requêtes a la deuxième partie le ***serveur*** qui a son tour lui envoi des réponses.



En pratique, le serveur est un ordinateur (physique ou virtuel) sur lequel fonctionne un logiciel lui-même appelé serveur FTP, qui rend public une arborescence de fichiers similaire à un système de fichiers UNIX. Pour accéder à un serveur FTP, on utilise un logiciel client FTP (possédant ou non une interface graphique) à l’instar de Filezilla. Le protocole FTP supporte deux manières de fonctionner, à peine différentes, mais la différence est d'importance, surtout lorsque l'on a à traverser un firewall par filtrage de paquets.

* **Mode actif** : c'est le client FTP qui détermine le port de connexion à utiliser pour  
  Permettre le transfert des données. Ainsi, pour que l'échange des données puisse se  
  faire, le serveur FTP initialisera la connexion de son port de données (port 20) vers  
  le port spécifié par le client. Le client devra alors configurer son pare-feu pour  
  autoriser les nouvelles connexions entrantes afin que l'échange des données se  
  fasse.
* **Mode passif** : le serveur FTP détermine lui-même le port de connexion à utiliser  
  pour permettre le transfert des données (data connexion) et le communique au  
  client. En cas de présence d'un pare-feu devant le serveur, celui-ci devra être  
  configuré pour autoriser la connexion de données. L'avantage de ce mode, est que  
  le serveur FTP n'initialise aucune connexion. Dans les nouvelles implémentations,  
  le client initialise et communique directement par le port 21 du serveur ; cela  
  permet de simplifier les configurations des pare-feu serveur.



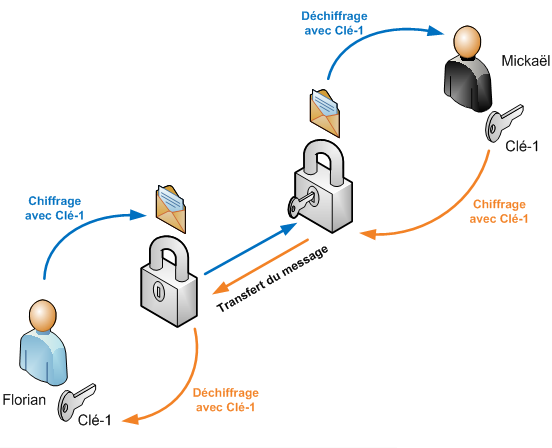
1. Installation et configuration d’un serveur FTP
2. Installation du serveur FTP
3. Configuration du serveur
4. Protocole ssh
5. Généralités sur ssh

**SSH** (Secure Shell) est un protocole de communication sécurisé. Apres que l’université de technologie d’Helsinki ait été victime d’une attaque par renfilage de mot de passe (cela consiste à créer un faux réseau wifi se faisant passer pour un vrai réseau wifi public, pour écouter les communications afin de récupérer le contenu transmis). La première version de SSH a été développée en 1995 par TATUN YLONEN, un professeur Finlandais qui, à l’époque souhaitait sécuriser les connexions distantes vers un serveur Unix. Dans cette version, le serveur écoute sur le port 22 et attend que le client initie la connexion. Ce client génère une clé qui va être envoyée au serveur et qui va permettre le chiffrement des données. Grâce à ce protocole, on peut ainsi accéder à des machines à distance et interagir avec elles en ligne de commande de manière sécurisée. Cette version est cependant obsolète à cause des problèmes de sécurité qui subsistent dans la **vérification de l’intégrité des données**, C’est ainsi qu’est née la version 2.0 de ssh, en Janvier 2006 a **l’IETF** (Internet Engineering Task Force). Grace à cette nouvelle version, de nouveaux algorithmes et des services de transfert de fichiers (SFTP, le Tunneling, le port forwarding, l’auhtentification via des clés privées sécurisées et bientôt l’utilisation des certificats x.509) ont été intégrés.

**2) Méthodes de chiffrement**

**a) Chiffrement symétrique**

Son principe est le suivant : si **A** veut envoyer un message confidentiel à **B**, **A** et **B** doivent d'abord posséder une même clé secrète. **A** crypte le message avec la clé secrète et l'envoie à **B** sur un canal qui n'est pas forcément sécurisé. **B** décrypte le message grâce à la clé secrète. Toute autre personne en possession de la clé secrète peut décrypter le message.



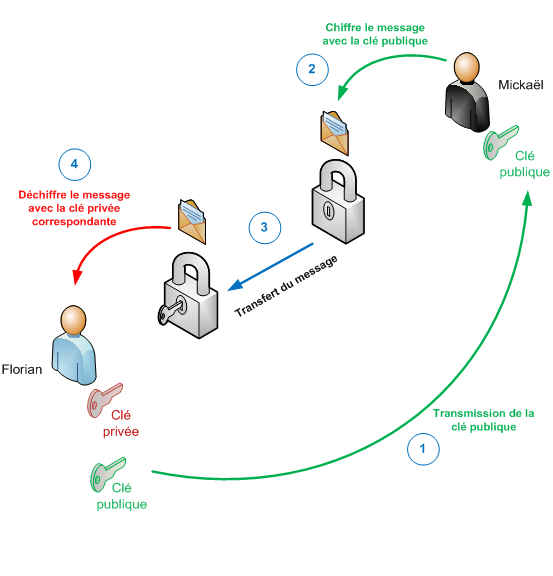
A travers ce schéma on voit bien que Florian chiffre un message avec la clé « **Clé-1** » puis le transfert à Mickaël qui le déchiffre avec la clé « **Clé-1** » qui est la même que celle dont dispose Florian. Il en est de même dans le sens inverse c'est-à-dire de Mickaël vers Florian. Ce qui montre bien que lors de l’utilisation d’une clé symétrique, la même clé permet de chiffrer et déchiffrer.

**Avantage et inconvénient**

L’avantage d’utiliser ce type de chiffrement est la simplicité de fonctionnement ce qui rend le [système](https://www.it-connect.fr/cours-tutoriels/administration-systemes/windows-server/systeme/) rapide et performant. Toutefois, si vous perdez ou distribuez malencontreusement votre clé, la personne qui l’a récupère pourra chiffrer des messages mais surtout, et c’est plus inquiétant, déchiffrer les messages chiffrés à partir de cette clé.

1. **Chiffrement asymétrique**

SSH utilise la cryptographie asymétrique RSA ou DSA. En cryptographie asymétrique, chaque personne dispose d'un couple de clé : une clé publique et une clé privée. La clé publique peut être librement publiée tandis que la clé privée doit rester secrète. La connaissance de la clé publique ne permet pas d'en déduire la clé privée. Si la personne **A** veut envoyer un message confidentiel à la personne **B**, **A** crypte le message avec la clé publique de **B** et l'envoie à **B** sur un canal qui n'est pas forcément sécurisé. Seul **B** pourra décrypter le message en utilisant sa clé privée.

****

A travers ce schéma on voit que Florian possède une paire de clés asymétriques dont une clé privée et une clé publique, ce sera lui qui recevra les messages et qui sera les seuls à pouvoir les déchiffrer. Dans un premier temps, Florian transmet sa clé publique à Mickaël pour qu’il puisse lui transmette des messages chiffrés. Ensuite, Mickaël chiffre un message avec la clé publique reçu puis transfert le message à Florian. Pour finir, Florian déchiffre le message grâce à sa clé privée qui est la seule à pouvoir déchiffrer le message.

**Avantage et inconvénient**

L’avantage de ce type de chiffrement c’est qu’on peut distribuer la clé publique sans risquer que les messages soient déchiffrés avec, étant donné que seul la clé privée permet de déchiffrer ses messages. L’inconvénient c’est que ça fonctionne en mode unidirectionnel uniquement à moins d’avoir deux paires de clés, et que le mode de fonctionnement est plus complexe que l’utilisation de clés symétriques et donc moins performant. De plus, il faut bien prendre conscience qu’il est impossible de retrouver une clé privée à partir d’une clé publique.

* + 1. **Fonctionnement du ssh**

Un serveur SSH dispose d'un couple de clés RSA stocké dans le répertoire **/etc/ssh/** et généré lors de l'installation du serveur. Le fichier **ssh\_host\_rsa\_key**contient la clé **privée** et a les permissions **600**. Le fichier **ssh\_host\_rsa\_key.pub** contient la clé **publique** et a les permissions **644**. Nous allons suivre par étapes l'établissement d'une connexion SSH :

1. Le serveur envoie sa clé publique au client.
2. Le client génère une clé secrète et l'envoie au serveur, en cryptant l'échange avec la clé publique du serveur (cryptographique asymétrique). Le serveur décrypte la clé secrète en utilisant sa clé privée, ce qui prouve qu'il est bien le vrai serveur.
3. Pour le prouver au client, il crypte un message standard avec la clé secrète et l'envoie au client. Si le client retrouve le message standard en utilisant la clé secrète, il a la preuve que le serveur est bien le vrai serveur.
4. Une fois la clé secrète échangée, le client et le serveur peuvent alors établir un canal sécurisé grâce à la clé secrète commune (cryptographie symétrique).
5. Une fois que le canal sécurisé est en place, le client va pouvoir envoyer au serveur le login et le mot de passe de l'utilisateur pour vérification. Le canal sécurisé reste en place jusqu'à ce que l'utilisateur se déloge.

La seule contrainte est de s'assurer que la clé publique présentée par le serveur est bien sa clé publique... sinon le client risque de se connecter à un faux serveur qui aurait pris l'adresse IP du vrai serveur (ou toute autre magouille). Une bonne méthode est par exemple de demander à l'administrateur du serveur quelle est le *fingerprint* de la clé publique du serveur avant de s'y connecter pour la première fois.  **NB : Le** **fingerprint** d'une clé publique est une chaîne de 32 caractères hexadécimaux unique pour chaque clé ; il s'obtient grâce à la commande **ssh-keygen -l**.

**4) Installation et configuration du ssh**

**4.1) Installation sur linux**

## -Mise a jour du gestionnaire de paquets :

## Sudo apt update