A Secure Peer-to-Peer Backup System

Houssem Jarraya
Institut TELECOM
TELECOM & Management SudParis
9 rue Charles Fourier
91011 EVRY
33 1 60 76 40 40
Houssem.Jarraya@int-edu.eu

Maryline Laurent-Maknavicius
Institut TELECOM
TELECOM & Management SudParis
CNRS Samovar UMR 5157,
9 rue Charles Fourier
91011 EVRY
33 1 60 76 40 40
Maryline.Maknavicius@it-sudparis.eu

ABSTRACT

A peer-to-peer (P2P) backup system enables users to save a copy of their data in the free disk space of other peers in order to restore them later in case of hard disk crash, or handling errors…
In this paper, we present a new peer-to-peer backup system implemented in the « DisPairSe » project and we concentrate on the security of such a system. Indeed, we provide mechanisms to ensure the integrity and confidentiality of data stored in the P2P network while respecting the anonymity of users. We also propose a solution to encourage peers to contribute to the backup system, to banish the selfish behaviour of a few peers and to give access to authorized nodes only. These solutions are based on the definition of a third trust party in charge of controlling the rights and the behaviour of peers.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *distributed applications, distributed databases.*

General Terms

Security.

Keywords

Security, peer-to-peer, data confidentiality, data integrity, anonymity, selfish peers

1. INTRODUCTION

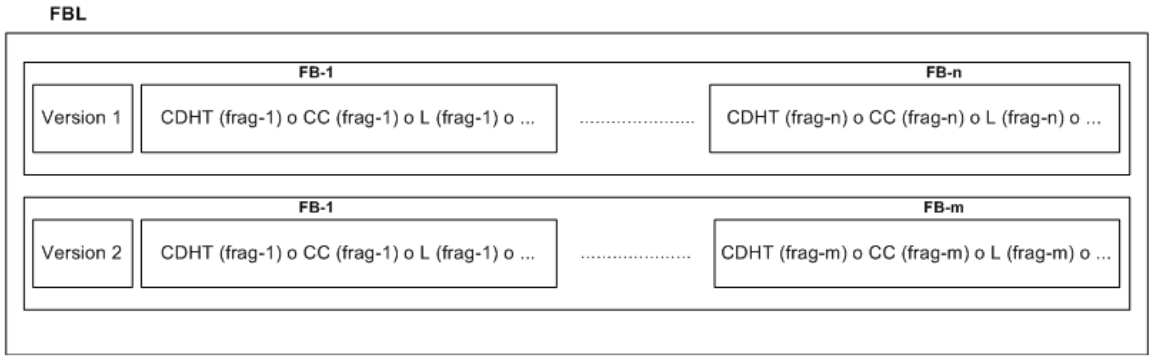
Si les systèmes de partage de fichiers Bittorrent, eMule,…ont fait leur succès, les réseaux pair à pair (P2P, de l'anglais "peer-to-peer") interviennent dans plusieurs autres domaines comme le calcul distribué, le travail collaboratif… En tant que système collaboratif, le système de sauvegarde distribuée repose sur la coopération des pairs à mettre à disposition leurs espaces disques. Un pair peut ainsi sauvegarder une copie de ses données dans les espaces disques d'autres pairs, l'objectif étant pour ce pair de restaurer ses données si besoin était, par exemple dans le cas de la perte de ses données locales. L'intérêt d'un tel service de sauvegarde distribuée réside dans la capacité de stockage apportée par le grand nombre de pairs dans un réseau P2P et au pourcentage important d'espace disponible dans chaque machine. Il n'est donc plus nécessaire de réserver des ressources de stockage de façon individualisée pour chaque utilisateur. Il en résulte une gestion optimisée des ressources de stockage.

De nos jours, plusieurs systèmes de sauvegarde distribuée ont été élaborés, plus ou moins différents sur certains aspects de leur architecture. Cependant, peu ont atteint le stade du prototype et aucun ne l'a dépassé pour être exploité en conditions réelles. L'objectif du projet DisPairSe [1] réalisé sur l'année 2007 est d'étudier un service de sauvegarde réparti entre utilisateurs, aussi bien au niveau système qu'économique, en considérant également les aspects réseaux et sécurité et l'usage d'IPv6. Du point de vue la sécurité, nous nous sommes intéressés aux enjeux classiques, à savoir la protection des données sauvegardées contre tout accès abusif et tentative de corruption, et ce, en intégrant les services de confidentialité et d'intégrité des données. Au-delà de ces enjeux, nous avons aussi apporté des solutions pour garantir l'anonymat des utilisateurs, contrôler le bon fonctionnement du système, inciter les nœuds à contribuer au service et contrôler l'accès au système P2P.

En vue d'une répartition homogène de la charge entre pairs et d'une plus grande rapidité de fonctionnement du service de sauvegarde, le projet DisPairSe a adopté le protocole Pastry [5] pour assurer le routage des requêtes de sauvegarde et de recherche des données. En effet, ce protocole se base sur une table de hachage distribuée (DHT) et ne nécessite que O(log(N)) sauts pour l'acheminement d'une requête, où N est le nombre de nœuds dans le réseau P2P.

Cet article est organisé de la façon suivante. La section 2 positionne les travaux effectués dans le projet DisPairSe par rapport aux autres solutions de sauvegarde aujourd'hui définies. La section 3 présente les principes de Pastry avec la notion de table DHT. La section 4 définit la structure d'un fichier sauvegardé par un pair dans le réseau P2P. La section 5 caractérise les traitements de sécurité appliqués sur les éléments d'information afin d'assurer leur intégrité, leur confidentialité et le respect de l'anonymat de leur propriétaire. Ces traitements sont majoritairement issus de la solution pStore [2], avec une proposition d'amélioration. La section 6 présente l'architecture réseau utile au support de la sécurité en précisant le rôle central joué par le fournisseur de service dans le processus de contrôle. La section 7 décrit le principe de fonctionnement du système

NOTERE 2008, June 23-27, 2008, Lyon, France.



CDHT: clé DHT du fragment; **CC**: clé de chiffrement du fragment; **L**: la longueur du fragment; **o** : opération de concaténation

Figure 1. Structure d'un fragment FBL

DisPairSe, c'est-à dire les processus suivis par un pair pour sauvegarder, modifier, récupérer ou supprimer un fichier du réseau P2P. La section 8 explique les propriétés et fonctions de notre approche, avec en particulier, la technique de contrôle d'accès au service de sauvegarde et le maintien de l'autonomie de ce service malgré le caractère centralisé du contrôle. Enfin quelques perspectives viennent conclure cet article.

2. NOTRE POSITIONNEMENT PAR RAPPORT A L'EXISTANT

Les systèmes de sauvegarde/restauration (P2P ou non) actuels se divisent principalement en deux catégories :

- Les systèmes payants permettent de garder une copie des données importantes dans un serveur distant tel que : Connected TLM. Le fait de communiquer avec une seule entité, a priori de confiance, rend ces systèmes fiables mais en contrepartie, ils sont très chers, surtout en cas de sauvegarde de données de grande taille.
- Les systèmes gratuits utilisant une infrastructure P2P tels que : pStore [2], PeerStore [3], Pastiche [4]. Malgré la gratuité de ces systèmes, aucune contrainte n'est fixée sur le volume d'espace disque dédié à un utilisateur. L'absence de relations de confiance entre pairs pose des problèmes de sécurité, de contrôle d'accès, et rend possibles les comportements égoïstes.

Dans le projet DisPairSe, nous avons défini un nouveau système de sauvegarde distribué qui prend les avantages de chacune de ces catégories tout en évitant leurs problèmes.

3. PRINCIPE DU PROTOCOLE PASTRY

Pastry est un protocole de routage et de localisation dans un réseau P2P basé sur une table de hachage distribuée. Ce protocole attribue pour chaque pair un identifiant unique (généralement sur 128 bits) qui est le résultat d'une fonction de hachage appliquée par exemples sur son adresse IP ou son nom de machine. De même, pour chaque ressource sauvegardée dans le réseau P2P, se trouve associée une clé unique de même longueur que les identifiants des pairs. Cette clé désignée dans l'article par « clé DHT » ou « CDHT » est calculée en appliquant une fonction de hachage sur un élément de la ressource, comme par exemples son nom ou son contenu. Par la suite, chaque ressource est sauvegardée dans le pair dont l'identifiant est le plus proche de sa clé, d'après une fonction de distance fixée.

Les deux opérations de base définies dans Pastry [5], comme dans les autres protocoles P2P utilisant une DHT, sont :

- store (key, value) : permet de sauvegarder la ressource « value » dans le pair dont l'identifiant est le plus proche numériquement de sa clé « key ».
- lookup (key) : permet de récupérer la ressource associée à la clé « key ». Cette requête sera routée vers le pair dont l'identifiant est le plus proche numériquement de la clé « key ». Soit ce dernier héberge lui-même la ressource demandée, et la transmet directement au nœud-client. Soit il connaît les paramètres de localisation (l'adresse IP et le numéro de port) de son hébergeur et redirige le nœud-client vers cet hébergeur.

4. STRUCTURE D'UN FICHIER DE SAUVEGARDE DANS LE RESEAU P2P

Avant de présenter les mécanismes de sécurité que nous avons adoptés pour protéger les données d'un utilisateur, nous définissons dans cette section la structure d'un fichier sauvegardé dans le réseau P2P.

Tout d'abord, chaque fichier sauvegardé dans le réseau est fragmenté. Cette fragmentation a pour but de réduire la duplication inutile des données dans l'espace (c'est-à-dire entre fichiers appartenant à différents utilisateurs) et dans le temps (entre différentes versions d'un même fichier). Elle permet, aussi, d'assurer une répartition homogène de la charge de stockage entre pairs du réseau P2P. En effet, la création de fragments de même longueur et l'utilisation d'une bonne fonction de hachage pour le calcul des clés DHT garantissent une distribution uniforme des données entre les différents pairs.

Comme dans le système pStore [2], chaque fichier est formé de deux types de fragments : un ensemble de fragments FB (File Block) contenant les informations réelles du fichier et un fragment de description FBL (File Block List). Ce dernier fragment, contient la liste des différents fragments FB de chaque version du fichier (voir Figure 1). Pour chaque fragment FB, nous précisons sa clé DHT, sa clé de chiffrement, sa longueur et son offset par rapport au début du fichier. Ainsi, à partir du fragment FBL, l'utilisateur peut récupérer, lors d'une phase de restauration, n'importe quelle version de son fichier.

5. SECURISATION DES DONNEES

Dans cette section, nous présentons les mécanismes de sécurité appliqués sur chaque élément de FB et FBL, afin d'en assurer son intégrité et sa confidentialité tout en respectant l'anonymat des utilisateurs.

Ne permettre la vérification de l'intégrité d'un fichier que par son propriétaire et lors de la phase de restauration ne suffit pas à garantir un service P2P de bonne qualité. En effet, il sera toujours

possible qu'un nœud pair malicieux participant au routage P2P modifie un fichier en cours de sauvegarde. Ainsi, la copie de secours obtenue et sauvegardée dans le réseau P2P sera incohérente et donc inutilisable. Sur l'exemple de la figure 2, la donnée (X = 6) a été modifiée par le nœud (N3), et malgré cela, elle a été sauvegardée dans le nœud (N5). Ce type de comportement malveillant affecte la fiabilité du système et induit, pour le système P2P, une perte de ressources (espace disque, bande passante, …).

Pour résoudre ce problème, il est primordial que les pairs de sauvegarde vérifient l'intégrité des données issues d'autres pairs et ce, sans aucune connaissance particulière sur leur identité. De cette façon, sera évitée la sauvegarde de données incohérentes dans le réseau P2P.

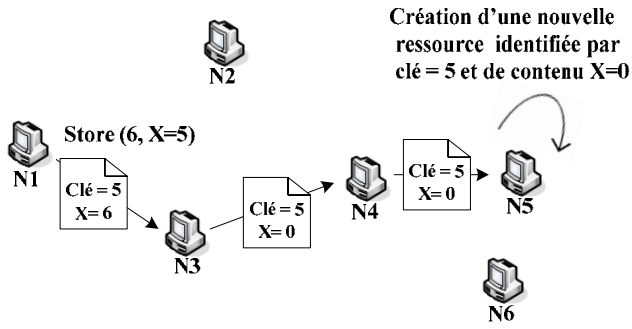


Figure 2. Exemple de comportement malveillant : modification du contenu d'une ressource

5.1 Protection des fragments FB

5.1.1 Confidentialité

Pour assurer la confidentialité des fragments FB, nous préconisons d'utiliser la technique du chiffrement convergent « Convergent Encryption » [6]. Cette technique consiste à chiffrer un bloc de données par le condensé de son contenu (voir partie gauche de la Figure 3). Elle est utilisée par plusieurs systèmes de sauvegarde distribués tels que pStore [2], ABS [7],… .

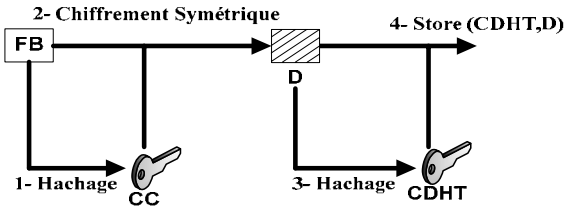


Figure 3. Traitement de sécurité appliqué par un nœud client sur un fragment FB

L'application de la technique *« Convergent Encryption »* sur les fragments de données (FB) offre les propriétés suivantes :

- Optimisation de l'utilisation de l'espace disque du système : des fragments FB identiques appartenant à des utilisateurs différents ou des fichiers différents aboutissent à des fragments chiffrés identiques (à cause de la génération de la même clé de chiffrement) et donc ils seront sauvegardés dans le même nœud et une seul fois.

- Utilisation d'une clé de chiffrement dépendante du contenu et donc ne pouvant être connue que des propriétaires des fragments. Ainsi, il n'y a aucun moyen, pour les nœuds malicieux, de déterminer la clé de chiffrement d'un fragment FB appartenant aux autres nœuds.

La clé DHT (CDHT) est calculée par hachage du fragment chiffré et l'identité entre deux fragments FB sera déterminée par la comparaison de leurs clés (condensés). Nous pouvons remarquer que des phénomènes de collision peuvent se produire, c'est-à-dire où deux fragments de contenu différent aboutissent au calcul de la même clé DHT et donc ils seront considérés comme identiques. Ce risque de collision est considéré comme négligeable (plus faible que la probabilité d'avoir une erreur hardware [10], i.e. disque dur) dans la littérature. En effet, la méthode de comparaison par condensé est utilisée dans plusieurs applications comme l'algorithme de synchronisation des fichiers à distance rsyn [8], le système de fichier distribué LBFS [9], les systèmes de sauvegarde distribuée Pastiche [4] et pStore [2]…

5.1.2 Intégrité

L'intégrité d'un fragment FB peut être simplement contrôlée par tout nœud pair en vérifiant que la clé DHT d'un fragment FB est le condensé de son contenu chiffré D. Un lien très fort unit en effet le fragment à sa clé DHT, comme le montre la figure 3. Cependant, ce contrôle ne suffit pas à parer toute altération des fragments FB. Un nœud malveillant peut en effet modifier à la fois le contenu D et la clé DHT associée en respectant toujours le mécanisme de la figure 3. Malgré le contrôle de l'intégrité de la requête de sauvegarde, la ressource D sera quand même sauvegardée (voir Figure 4). Nous avons résolu la détection de cette attaque en intégrité par l'ajout d'une entité de contrôle supplémentaire tiers de confiance, comme le décrit la section 8.1.

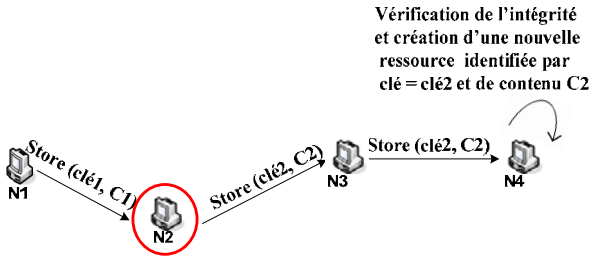


Figure 4. Echec du contrôle d'intégrité simple face à l'attaque par modification à la fois du contenu et de la clé DHT associée (avec clé1 = H (C1) , clé2 = H (C2) et H pour une fonctions de hachage)

5.2 Protection des fragments FBL

Lors de la phase de restauration d'un fichier, il est clair que le fragment FBL est le plus important : il sert de point d'entrée utile à toute restauration d'un fichier dans son entier. Aussi faut-il être prudent dans le calcul de sa clé DHT (pour éviter les collisions) et

faut-il garantir sa confidentialité et son intégrité (pour éviter une restauration d'un fichier par un tiers non autorisé ou l'échec d'une restauration future).

Contrairement à un fragment FB, le contenu d'un fragment FBL est modifié après chaque mise à jour du fichier. Il n'est donc pas souhaitable que la clé DHT soit calculée de la même façon qu'à la figure 3, sinon une mise à jour aboutirait à chaque fois à un jeu de clés différent - clé DHT / clé de chiffrement - et surtout l'utilisateur aurait à charge de conserver ces clés (en vue d'une restauration éventuelle future). Pour résoudre ce problème, nous avons adopté la solution proposée dans le système pStore qui consiste à déterminer la clé DHT d'un fragment FBL à partir de son nom et de son chemin au lieu de son contenu.

5.2.1 Calcul de la clé DHT du fragment FBL

Contrairement aux fragments FB, le risque de collision entre les fragments FBL n'est plus négligeable si nous calculons leurs clés DHT uniquement à partir du nom et du chemin de leur fichier. En effet, pour des fichiers différents appartenant à différents utilisateurs et répondant aux mêmes caractéristiques - même nom et même chemin - des fragments FBL de contenus différents leur sont associés mais la même clé DHT leur correspond. Pour éviter ces collisions, il suffit d'introduire dans le calcul de la clé DHT d'un fragment FB une information propre à l'utilisateur afin de pouvoir lui associer un espace de nom privé. Dans le système pStore, cette information correspond à la clé privée de l'utilisateur, mais de façon générale, elle peut correspondre à n'importe quelle information propre et secrète (mot de passe, f (clé privée), f (mot de passe), …). Dans la suite de cet article, cette information secrète sera désignée par le « secret » de l'utilisateur.

Dans le projet DisPairSe, le calcul de la clé DHT d'un fragment FBL se fait comme suit :

CDHTFBL = H (secret o pathname o filename)

Formule 1. Calcul de la clé DHT d'un fragment FBL

5.2.2 Confidentialité

Dans le système pStore, un utilisateur chiffre ses différents fragments FBL avec la même clé de chiffrement. Cette méthode n'est pas robuste car la découverte de sa clé de chiffrement, par un nœud malicieux, entraîne le déchiffrement de ses différents fragments FBL. De plus, s'il veut partager l'un de ses fichiers avec d'autres pairs, il doit leur fournir la clé de chiffrement du fragment FBL associé au fichier à partager, mais l'obtention de cette clé entraîne le déchiffrement de tous ses autres fragments FBL.

Afin d'attribuer une clé de chiffrement différente à chaque fragment FBL, nous avons utilisé la formule suivante :

CCFBL = H (f (secret) o pathname o filename)

Formule 2. Calcul de la clé de chiffrement d'un fragment FBL (où f est une fonction appliquée sur le secret de l'utilisateur)

Ainsi, le nom et le chemin du fichier sont suffisants pour permettre à l'utilisateur de déterminer la clé DHT et la clé de chiffrement du fragment FBL et par la suite de pouvoir restaurer tout le fichier.

5.2.3 Intégrité

De la même façon que pour le fragment FB, il faut que l'intégrité du fragment FBL puisse être vérifiée par son propriétaire et son nœud de sauvegarde. Dans le système pStore, chaque utilisateur signe son fragment FBL en utilisant sa clé privée et inclut sa clé publique dans le fragment FBL à sauvegarder. Par l'adjonction de cette signature et de sa clé publique, l'utilisateur rend possible la vérification de l'intégrité de son fragment FBL par n'importe quel nœud P2P. Cependant, cette solution souffre de deux inconvénients majeurs : elle nécessite la mise en place d'une infrastructure de gestion de clés publiques et elle n'assure pas le respect de l'anonymat des utilisateurs puisque l'identité du propriétaire d'un fragment peut être connue du fait la divulgation nécessaire de sa clé publique associée.

Dans le projet DisPairSe, nous avons résolu le double problème de vérification de l'intégrité par deux solutions techniques distinctes :

- Chaque utilisateur signe ses fragments FBL en suivant le processus décrit à la figure 5 : il ajoute au fragment FBL son condensé et puis il chiffre tout le message (fragment en clair et son condensé) en utilisant sa clé de chiffrement. Avec ce mécanisme, seul le possesseur de la clé de chiffrement est en mesure de vérifier l'intégrité du fragment FBL. Ainsi, cette méthode ne permet qu'au propriétaire du fichier de vérifier l'intégrité de ses fragments FBL.

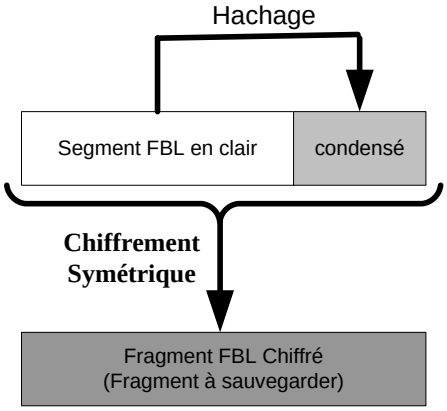


Figure 5. Intégrité du fragment FBL

- La vérification de l'intégrité d'un FBL par un nœud de sauvegarde n'est possible que par l'intervention d'une entité de confiance appelée par la suite « Fournisseur de Service » ou SP (Service Provider). C'est cette entité qui se charge de vérifier si les fragments FBL ont bien été sauvegardés dans le réseau P2P en suivant le processus décrit à la figure 6. Plus précisément, l'utilisateur lance la sauvegarde de son fragment FBL (étape 1) et communique simultanément au SP la clé DHT, la longueur et le condensé du fragment FBL (étape 2). Le nœud de sauvegarde (Nœud j) conserve le fragment FBL dans une mémoire cache en attendant que la vérification de l'intégrité soit terminée et il envoie la clé DHT, la longueur et le condensé qu'il a lui-même calculé au SP (étape 3). Par comparaison des deux condensés (étape 4), le SP peut vérifier l'intégrité du FBL reçu par le nœud j : si cohérence il y a, le nœud j en est informé et termine la sauvegarde normalement, sinon, SP demande au nœud client i (propriétaire du fragment) de le retransmettre.

Le rôle de l'entité SP de confiance apparaît plus clairement par la suite, en particulier à la section 6. Il ne se limite pas au simple contrôle d'intégrité, mais s'étend à des fonctions de contrôle plus complexes.

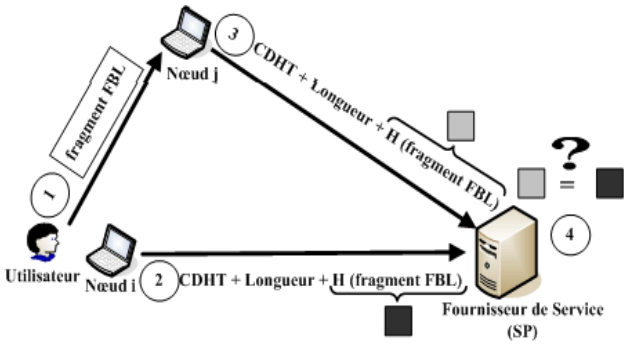


Figure 6. Vérification de l'intégrité d'un fragment FBL par son nœud de sauvegarde dans DisPairSe

6. ARCHITECTURE GLOBALE

Dans le projet DisPairSe, la sécurisation du système P2P de sauvegarde nous a amenés à définir une entité de confiance, que nous avons appelée SP pour Service Provider (voir section 5.2.3). Ce SP ayant un rôle critique dans le service de sauvegarde, son accès doit être protégé par une authentification préalable d'un pair (sous couvert d'un utilisateur). Nous avons ainsi adopté l'architecture de la figure 7 qui distingue les entités de réseau suivantes :

- Le serveur d'authentification (AS) authentifie les pairs et renseigne le NAS sur les droits d'accès de ces pairs au SP. Les échanges entre le NAS et l'AS font classiquement appel aux protocoles AAA (Authentication, Authorization, Accounting) de type RADIUS.
- Le Network Access Server (NAS) est positionné en frontal et doit bloquer toute demande de connexion issue des pairs vers le SP tant que les pairs ne se sont pas authentifiés auprès de l'AS. Le NAS relaie donc tous les échanges utiles à l'authentification entre les pairs et l'AS. Une fois l'authentification réussie et l'autorisation obtenue par l'AS pour qu'un pair entre en contact avec le SP, il relaie le trafic entre le pair autorisé et SP. Notons que les pairs peuvent être éloignés du NAS de plusieurs routeurs. De ce fait, on privilégiera un protocole en support à l'authentification qui opère au-dessus de la couche réseau (couche 3 du modèle OSI). Le protocole PANA [14] mentionné sur la figure 7 est parfaitement adapté à ce besoin.
- Le fournisseur de service (SP) est une entité de confiance à la base du service P2P de sauvegarde. SP a globalement pour rôle de contrôler le bon fonctionnement du système (détection de comportements malveillants, vérification de l'intégrité des fragments FBL en cours de sauvegarde,…).

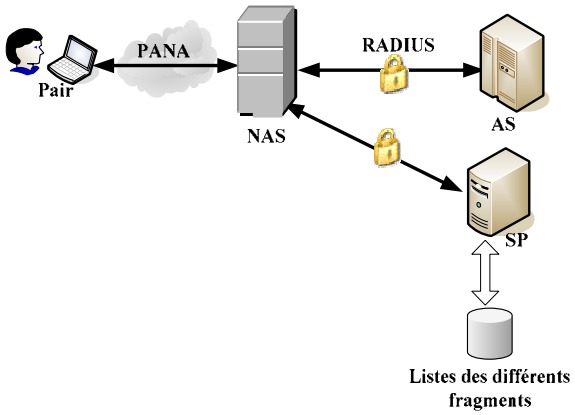


Figure 7. Architecture réseau retenue dans DisPairSe

Du fait du rôle important assuré par le SP, il est utile de donner plus précisément les fonctions centrales assurées par ses soins :

- Contrôle d'intégrité des fragments FBL (voir section 5.2.3) : SP vérifie qu'un fragment FBL est intègre avant d'autoriser un nœud de sauvegarde à procéder à la sauvegarde du fragment.
- Comptabilisation des ressources consommées par chaque pair : SP maintient à jour une liste des fragments sauvegardés dans le système, à partir de laquelle il est possible de connaître les ressources consommées par un pair, mais aussi les ressources consommées par d'autres pairs sur son propre espace disque mis volontairement à disposition. Cette fonction de comptabilisation a un rôle double. (1) Elle permet d'une part de vérifier qu'un nœud remplit bien son rôle de nœud de sauvegarde. (2) Elle sert de base au mécanisme d'incitation des nœuds à contribuer.

- Détection du comportement égoïste des pairs. Des erreurs de sauvegarde répétées, une forte indisponibilité d'un nœud, une forte dissymétrie entre ressources consommées par un pair et ressources consommées par d'autres pairs sur son disque, un écart conséquent constaté entre les ressources théoriquement mises à disposition par un nœud et le taux d'occupation de son disque… ce sont autant d'éléments qui servent à la détection des comportements égoïstes.
- Incitation des pairs à contribuer au service de sauvegarde : l'incitation peut être mise en pratique de nombreuses façons. Une incitation répressive par exemple serait de menacer d'exclusion un nœud pair s'il ne participe pas suffisamment au service. Une incitation financière serait de définir un coût de l'octet sauvegardé et un coût de l'octet mis à disposition. De la sorte, un nœud serait crédité s'il offre globalement plus d'espace disque qu'il n'en consomme chez les autres ; un nœud serait débité dans le cas contraire. Des explications précises sont apportées à la section 8.1.
- Contrôle d'accès au service P2P de sauvegarde : bien que le service P2P de sauvegarde soit anonyme et soit donc accessible a priori à tous, l'accès au service de sauvegarde est indirectement contrôlé. En effet, chaque opération de sauvegarde, suppression … de fichiers est contrôlée par le SP. Un utilisateur qui tenterait de forcer le système P2P à sauvegarder des fichiers verrait donc sa sauvegarde échouer. Le contrôle exercé par le SP pourrait laisser croire que l'indisponibilité du SP entrainerait l'indisponibilité du service de sauvegarde. Il n'en est rien, car le réseau P2P conserve une grande part d'autonomie. La solution retenue est donc particulièrement intéressante car elle assure à la fois un contrôle sécurisé des opérations liées à la sauvegarde, et un niveau d'autonomie du service de sauvegarde suffisant pour pallier les indisponibilités du SP. Des explications plus complètes sont fournies dans les sections 8.2 et 8.3.
- Test de disponibilité des fragments : par la connaissance du nœud de sauvegarde et du condensé de chaque fragment sauvegardé dans le réseau P2P (voir section 7.1), le SP peut vérifier périodiquement la disponibilité et l'intégrité d'un sous ensemble aléatoire de fragments et peut faire diminuer le prix d'un octet fourni par des nœuds malicieux (nœuds qui suppriment ou modifient un fragment de données après l'avoir déclaré au SP). Ainsi, chaque nœud se trouve obligé de correctement sauvegarder les fragments des autres pairs s'il veut augmenter le prix de son espace disque.

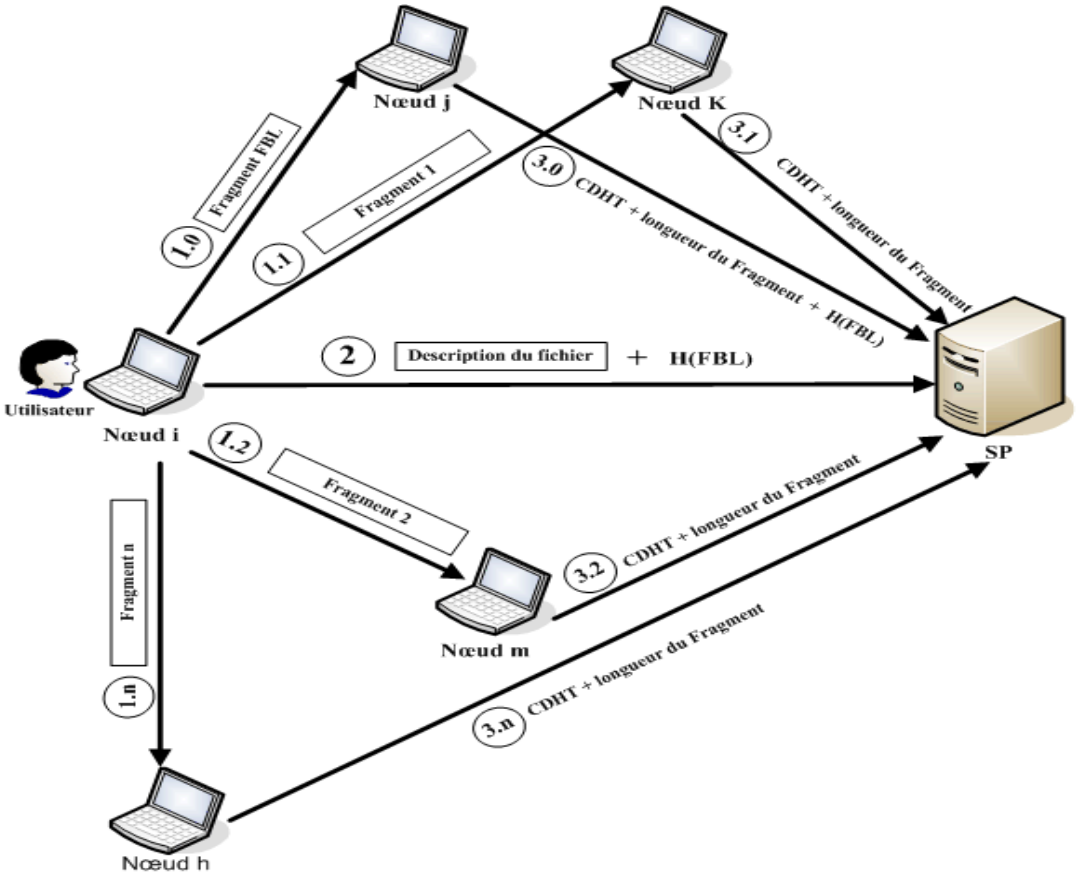


Figure 8. Sauvegarde d'un nouveau fichier

De plus, le projet DisPairSe adopte la méthode « erasure coding » qui consiste à transformer une donnée de n blocs en une donnée de m blocs (tel que m > n). L'idée consiste à permettre la restauration de la donnée originale même si quelques uns de ces blocs sont perdus. Appliqué au contexte P2P, un fichier peut donc être restauré dans son entier même en cas de perte de quelques fragments FB.

La définition d'une telle architecture avec des capacités de comptabilisation des ressources et des outils de contrôle sécurisés est un premier pas vers l'intégration d'un système de sauvegarde distribuée dans la chaîne des valeurs.

7. FONCTIONNEMENT DE DISPAIRSE

Cette section présente les différentes opérations de la solution de sauvegarde retenue dans le projet DisPairSe. Pour plus de détails, le lecteur intéressé pourra se reporter au rapport de recherche [13].

7.1 Sauvegarde d'un fichier

L'opération de base fournie par un système de sauvegarde distribuée DisPairSe est la sauvegarde d'un fichier dans le réseau P2P. Dans la plupart des systèmes élaborés, un pair fragmente le nouveau fichier à sauvegarder en plusieurs fragments de donnée, applique les traitements de sécurité sur chaque fragment de donnée, construit le fragment de description, applique les traitements de sécurité sur le fragment de description et enfin sauvegarde l'ensemble de ces fragments dans le réseau P2P. Bien que ce processus crée et protège l'ensemble des fragments nécessaires à la restauration du fichier, il ne permet ni de garantir que les fragments créés ont bien été sauvegardés dans le réseau P2P, ni de contrôler le comportement de chacun des nœuds, ni d'offrir les informations nécessaires pour comptabiliser les ressources consommées ou fournies par les pairs.

Pour résoudre ces problèmes, nous avons ajouté au processus classique de sauvegarde d'un fichier les deux derniers échanges 2 et 3 faisant intervenir le SP (voir Figure 8), comme précédemment décrit à la section 5. Ainsi, le nouveau processus a lieu en trois étapes :

- Etape 1 : L'utilisateur crée et sauvegarde les différents fragments dans le système (fragment FBL et l'ensemble des fragments FB).
- Etape 2 : L'utilisateur envoie au SP une description du nouveau fichier qui contient la liste des clés DHT et la longueur des différents fragments créés (ces informations sont nécessaires à la comptabilisation) ainsi que le condensé du fragment FBL.
- Etape 3 : Chaque nœud envoie au SP une description des nouveaux fragments acquis. S'il s'agit d'un fragment FB, le nœud envoie sa clé DHT et sa longueur et s'il s'agit d'un fragment FBL, il envoie en plus son condensé.

Pour ne pas surcharger le SP, les différents échanges entre un pair et le SP sont périodiques. Par exemple toutes les deux heures, chaque nœud envoie au SP une description des nouveaux fichiers créés et une description des nouveaux fragments acquis.

7.2 Contrôle exercé par SP lors de la sauvegarde

En recevant les déclarations du propriétaire du fichier et celles des nœuds de sauvegarde, le SP qui est une entité de confiance, aura toutes les informations nécessaires pour contrôler le bon fonctionnement du système de sauvegarde. En effet, si nous considérons la liste L1 des clés DHT déclarées par les propriétaires des fichiers et la liste L2 des clés DHT acquittées par les nœuds de sauvegarde, le SP peut faire face aux cinq cas suivants :

- **Cas 1 :** si une clé DHT i appartient à L1 et à L2 et avec des longueurs de fragments identiques ➔ le fragment dont la clé DHT est i a bien été sauvegardé dans le système : l'espace disque consommé par son propriétaire et l'espace disque fourni par son nœud de sauvegarde augmentent.
- **Cas 2 :** si une clé DHT i appartient à L1 et n'appartient pas à L2, on en déduit que le fragment dont la clé DHT est i n'a pas été sauvegardé dans le système (il a été supprimé ou modifié par un nœud malveillant) : le SP demande à son propriétaire de le retransmettre.
- **Cas 3 :** si une clé DHT j appartient à L2 et n'appartient pas à L1, on en déduit soit qu'un pair n'a pas déclaré ce fragment pour le SP, soit que le nœud de sauvegarde n'a pas acquis réellement ce fragment : le SP envoie une requête pour supprimer le fragment associé à cette clé DHT.
- **Cas 4 :** si une même clé DHT appartient à la fois à la liste L1 et L2 mais avec des longueurs de fragments différents ➔ soit le propriétaire du fragment, soit le nœud final est malveillant : le SP peut vérifier la longueur du fragment en le récupérant du réseau P2P et puis diminuer le niveau de fiabilité du nœud qui a modifié sa longueur réelle (un nœud à toujours intérêt de diminuer la longueur de ces fragments et d'augmenter la longueur des fragments qu'il stocke).
- **Cas 5 :** si une même clé DHT d'un fragment FBL appartient à la fois à la liste L1 et L2 mais avec des condensés de fragments différents (le condensé envoyé par le propriétaire du fragment est différent de celui calculé par le nœud de sauvegarde) ➔ le fragment FBL a été modifié : le SP demande au nœud de sauvegarde de le supprimer et à son propriétaire de le retransmettre.

Dans le cas où un pair sauvegarde un fichier sans informer le SP de la liste des fragments qu'il a créée et sauvegardée dans le réseau P2P. La liste L1 sera vide et c'est la liste L2 qui contient la liste des clés DHT des différents fragments créés : il s'agit du troisième cas et donc le SP va demander la suppression de tous ces fragments créés.

7.3 Mise à jour d'un fichier

Pour la mise à jour d'un fichier, l'utilisateur suit le même scénario décrit à la figure 8. Pour seules différences, on notera que la nouvelle description du fichier contient uniquement les nouveaux fragments à créer et que l'utilisateur ne sauvegardera (étape 1)

que les fragments qui n'existent pas dans les anciennes versions (cette liste peut être déterminée par l'algorithme rsync [11]). Le seul fragment qui sera mis à jour est le fragment FBL. Le nœud qui le sauvegardera doit vérifier son intégrité avant de supprimer son ancienne version.

7.4 Suppression d'un fichier

Le système DisPairSe doit donner à l'utilisateur la possibilité de supprimer les versions anciennes d'un fichier ou un fichier tout entier afin de libérer de l'espace disque pour sauvegarder de nouvelles versions ou des fichiers plus importants. Cette tâche est très difficile à réaliser dans un réseau P2P totalement anonyme. En effet, un nœud de sauvegarde doit s'assurer de l'identité d'un pair et vérifier ses droits d'accès avant d'accepter sa demande de suppression d'un fragment, ce qui est contradictoire avec le souhait de respect de l'anonymat des utilisateurs. Du fait de cette complexité, le système PeerStore [3] n'autorise pas la suppression des fichiers. Les autres systèmes de sauvegarde distribués tels que : Farsite [12], pStore [2],… associent à chaque fragment la clé publique de son propriétaire pour que la demande de suppression soit signée avec la clé privée correspondante, mais ces solutions ne sont pas satisfaisantes car elle ne respecte pas l'anonymat des utilisateurs. En effet, par connaissance de la clé publique du propriétaire du fragment, il est facile de déterminer l'identité associée.

Pour permettre la suppression des fragments tout en assurant l'anonymat des utilisateurs, nous avons proposé la solution suivante : chaque utilisateur inclut dans ses messages périodiques envoyés au SP les clés DHT de ses fragments à supprimer. Avec l'identifiant du propriétaire et du nœud de sauvegarde de chaque fragment, le SP peut vérifier, tout d'abord, les droits de suppression. Puis il inclut dans ses messages périodiques envoyés aux nœuds de sauvegarde les clés DHT des fragments à supprimer. De cette façon, l'opération de suppression est assurée sans aucun transfert supplémentaire, ni entre les nœuds, ni entre nœud et SP. De plus, les droits de suppression sont gérés sans coût additionnel de stockage dans le système P2P.

7.5 Restauration d'

En cas de perte d'un fichier, préciser le nom et le che restauration. En effet, à p système détermine tout d chiffrement du fragment Formules 1 et 2), puis il extr chiffrement associées aux système P2P récupère et nécessaires et reconstitue le f

8. AUTRES CARA

Les éléments techniques de ayant été décrits dans la sec décrire plus précisément ce retenue dans DisPairSe.

8.1 Incitation des pairs à contribuer au service

D'après le processus de sauvegarde d'un fichier dans le réseau P2P (voir section 7.1), le SP est informé de la liste des fragments sauvegardés dans le réseau P2P ainsi que de la longueur, du propriétaire et du nœud de sauvegarde de chaque fragment. Pour détecter et ensuite remédier au comportement égoïste des nœuds, le SP peut choisir de comptabiliser pour chaque nœud du réseau P2P l'espace disque consommé et l'espace disque fourni au système, ceci afin de lui attribuer une facture débitrice lorsqu'il consomme plus de ressources qu'il n'en fournit ou bien une rémunération dans le cas contraire. De cette façon, chaque pair se trouve obligé de fournir une partie de son espace disque au système soit pour bénéficier d'un service de sauvegarde gratuit soit pour avoir une rémunération.

8.2 Contrôle d'accès au service de sauvegarde

D'après le processus de sauvegarde d'un nouveau fichier, l'autorisation d'accès au service DisPairSe est équivalente à l'autorisation d'accès au fournisseur de service (SP), pour les raisons suivantes :

- Tout fragment créé sans informer le SP de sa clé DHT est supprimé. Ainsi, chaque pair doit avoir préalablement obtenu l'accès au SP afin de pouvoir lui déclarer la liste des fragments qu'il crée.
- Chaque nœud recevant un nouveau fragment doit informer le SP de sa clé DHT et de sa longueur pour que son effort de contribution au service soit comptabilisé dans le SP (en vue d'une rémunération éventuelle). Par conséquent, un pair n'ayant pas accès au SP n'a aucun intérêt à sauvegarder les fragments d'autres nœuds.

Ainsi, chaque utilisateur doit s'authentifier auprès du serveur d'authentification pour avoir accès au SP et donc bénéficier du service de sauvegarde.

Le fait de rendre l'accès au SP indispensable pour qu'un nœud du

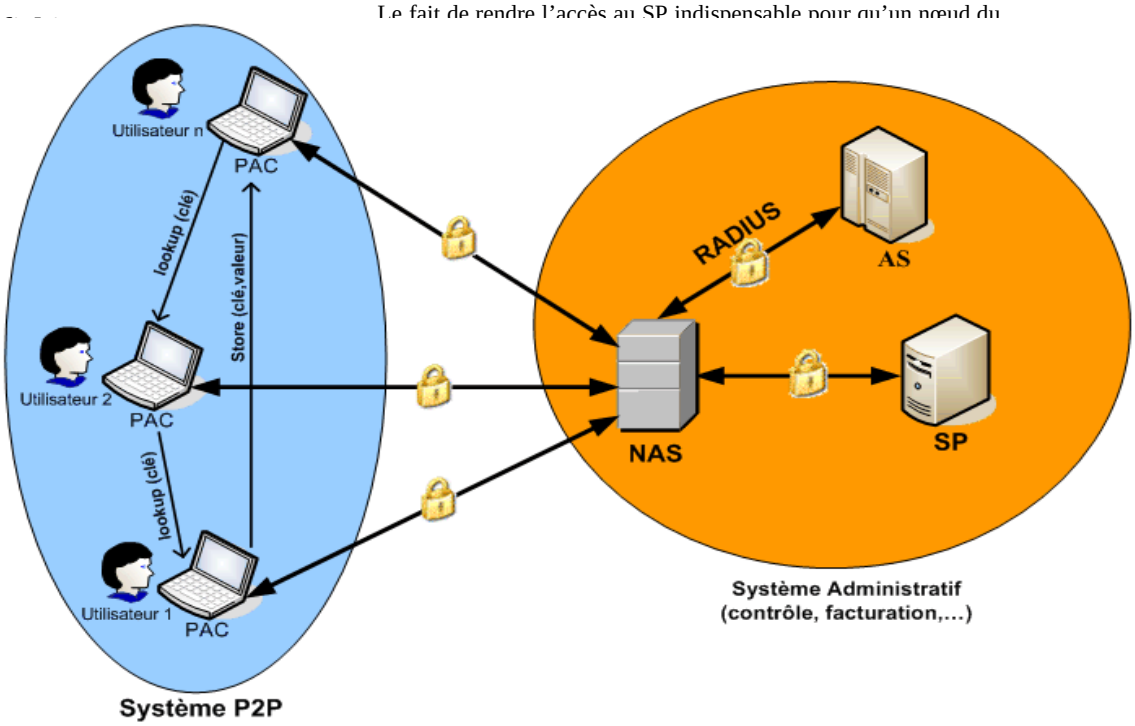


Figure 9. Fonctionnement autonome du système de sauvegarde

- L'échange entre les nœuds : cet échange concerne les données et les métadonnées des fichiers (les fragments FB et FBL). Ce type d'échange n'exige aucune authentification.

Le fait de sauvegarder les données et les métadonnées des fichiers dans le réseau P2P et d'autoriser une communication entre pairs sans aucune obligation d'authentification permet d'augmenter la disponibilité du service de sauvegarde et d'assurer son autonomie (voir Figure 9). En effet, un pair peut toujours récupérer ses données et sauvegarder de nouveaux fichiers dans le réseau P2P, même en cas d'indisponibilité du système administratif (indisponibilité du SP ou de l'AS). Notons que le contrôle effectué par le SP sera en fait simplement retardé jusqu'à que le système administratif soit de nouveau accessible.

9. CONCLUSIONS ET PERSPECTIVES

Dans le projet DisPairSe, nous avons défini un nouveau système de sauvegarde distribuée et sécurisée, respectant entièrement l'anonymat des utilisateurs et présentant de nombreux avantages comparativement aux systèmes de sauvegarde distribuée existants (voir section 2). Le système introduit dans le fonctionnement du service P2P de sauvegarde, un tiers de confiance agissant comme un fournisseur de service et permettant de contrôler le fonctionnement du système et d'assurer une sauvegarde de qualité et sécurisée des données d'un utilisateur.

Il nous semble important de poursuivre ces travaux et de veiller à mieux contrôler le comportement des différents nœuds du réseau P2P. En effet, par analyse des déclarations reçues des pairs consommateurs et des pairs fournisseurs d'espace disque (liste des fragments créés, liste des fragments acquis, liste des fragments à supprimer, …), le fournisseur de service peut découvrir les comportements malveillants des différents pairs. Une piste de recherche possible pourrait donc consister à attribuer à chaque nœud un niveau de fiabilité et à faire dépendre le prix d'un octet offert au système en fonction de la fiabilité de son propriétaire. Ainsi, chaque pair sera obligé de bien respecter les règles de fonctionnements du système, s'il veut garder un niveau de fiabilité élevé et augmenter le prix de son espace disque.

10. REMERCIEMENTS

Nous tenons à remercier l'Institut TELECOM, et la Fondation TELECOM (ex Fondation Louis Leprince-Ringuet) pour leur soutien financier ainsi que Laurent Toutain de TELECOM Bretagne, initiateur du projet DisPairSe.

11. REFERENCES

[1] https://dispairse.point6.net

[2] C. Batten, K. Barr, A. Saraf and S. Trepetin, "pStore: A secure peer-to-peer backup system", Technical Memo MIT-LCS-TM-632, Massachusetts Institute of Technology Laboratory for Computer Science, October 2002.

[3] M. Landers, H. Zhang and K-L. Tan, "PeerStore : Better Performance by Relaxing in Peer-to-Peer Backup", Proceedings of the Fourth International Conference on PeertoPeer Computing, 2004.

[4] L.P. Cox, C.D. Murray and B.D. Noble, "Pastiche: Making backup cheap and easy", In Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 2002.

[5] A. Rowstron and P. Druschel, "Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems", Microsoft Research, 2001.

[6] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon and M. Theimer, "Reclaiming Space from Duplicate Files in a Serverless Distributed File System", Technical Report MSR-TR-2002-30, July 2002.

[7] J. Cooley, C. Taylor and A. Peacock, "ABS: the apportioned backup system. Technical report", IT Laboratory for Computer Science, 2004.

[8] A. Tridgell, "Efficient Algorithms for Sorting and Synchronization", PhD thesis, The Australian National University, 1999.

[9] A. Muthitacharoen, B. Chen and D. Maziéres. "A low-bandwidth network file system", In Proceedings of the 18th ACM Symposium on Operating Systems Principles, 2001.

[10] V. Henson, "An analysis of compare by hash", in 9th Workshop on Hot Topics in Operating Systems (HotOS IX), (Lihue Hawaii), Sun Microsystems, May 2003.

[11] http://rsync.samba.org/

[12] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J. R. Lorch, M. Theimer, R. P. Wattenhofer, "FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment",Microsoft Research, December 2002.

[13] H. Jarraya, M. Laurent-Maknavicius, “ Un système de sauvegarde P2P sécurisé s'appuyant sur une architecture AAA ”, Rapport de recherche TELECOM & Management SudParis, 08-002 LOR, 2008.

[14] A. Yegin, Y. Ohba, R. Penno, G. Tsirtsis, C. Wang, "Protocol for Carrying Authentication for Network Access (PANA) Requirements", RFC4058, May 2005.