Implémenter le droit à l'oubli

Simon Bouget

Rapport de stage
M2 Recherche :
Implémenter le droit à l'oubli

Dégradation des données par publication éphémère

Simon Bouget
Encadrants : Sébastien Gambs & Guillaume Piolle
Équipe CIDRE, INRIA/Supelec

06/06/2013

Résumé — Le respect de la vie privée est un droit fondamental mais difficile à protéger dans un monde numérique : copier et transmettre de l'information est devenu extrêmement simple. Aussi, le droit à l'oubli est-il une composante de plus en plus importante de la protection de la vie privée. Toutefois, une solution technique et complète pour implémenter le droit à l'oubli parait impossible à l'heure actuelle. Nous présentons donc les systèmes existants et leurs domaines d'applications respectifs. Leur mise en œuvre se heurte cependant à des intérêts contradictoires entre utilisateurs et hébergeurs des données, qui engendrent un manque de confiance.

Puis nous introduisons le mécanisme de dégradation des données, qui recoupe les principes de rétention et de collecte minimales, et qui permet, combiné à une technique de publication éphémère, de réconcilier partiellement les intérêts des hébergeurs et des utilisateurs et de passer outre les problèmes. Enfin, nous présentons une architecture générale utilisant ces deux mécanismes ainsi qu'une illustration sur le cas d'un réseau géo-social.

Mots-clé : vie privée ; droit à l'oubli ; dégradation des données ; publication éphémère ;

Table des matières

1 Introduction

La *protection de la vie privée* est un droit fondamental, énoncé notamment dans l'article 12 de la Déclaration Universelle des Droits de l'Homme [1]. Pourtant, ce droit est de plus en plus menacé. Avec le développement d'Internet, chaque individu laisse constamment des traces numériques qui peuvent être reliées à son identité. De plus, les communications sont devenues très rapides, la copie de données numériques extrêmement facile. Aujourd'hui il est donc relativement aisé pour un acteur malveillant de s'immiscer dans la vie privée d'un internaute si ce dernier n'y prend pas garde. Or la plupart des gens sont imprudents et publient énormément d'informations sensibles, sur les réseaux sociaux par exemple.

Ainsi, le *droit à l'oubli* devient une composante essentielle de la protection de la vie privée. Mayer-Schönberger [18] explique que pendant longtemps, l'oubli a été la norme, mais que depuis l'avènement du numérique, la rétention des données est devenu le standard, avec les conséquences importantes évoquées ci-dessus. Il insiste également sur le risque à moyen terme d'une dérive vers une société « panoptique », c'est-à-dire une société disciplinaire où tout le monde est observé en permanence et qui utilise la pression sociale de l'observation pour imposer ses règles. Cette situation nuirait à la fois à la spontanéité de notre société et à la liberté d'expression, autre droit fondamental.

Cependant, la mise en pratique du droit à l'oubli se heurte à un problème de confiance entre les hébergeurs de données et leurs utilisateurs : l'intérêt des hébergeurs est d'accumuler un maximum de données pour pouvoir les monétiser. En conséquence, ils rechignent souvent à effacer des données. De l'autre côté, les utilisateurs en ont conscience et ne croient pas toujours les hébergeurs lorsque ces derniers affirment que l'effacement a bien été effectué.

Il est donc important que la recherche propose de nouvelles technologies qui offrent des garanties pour le droit à l'oubli sans reposer sur un hébergeur de confiance.

1. http ://www.un.org/en/documents/udhr/index.shtml#a12

1.1 Définitions

Avant de décrire plus précisément notre problématique, nous allons définir les concepts centraux à ce rapport.

1.1.1 Protection de la vie privée

En protection des données, on distingue différents types d'acteurs en fonction de leur relation aux données en question :

- le *sujet des données* (ou utilisateur, ou client) est la personne physique à laquelle les données se rapportent ;
- le *contrôleur des données* (ou hébergeur) est l'entité qui accueille les données sur ses infrastructures physiques et décide de leur utilisation. Selon les cas (utilisation de cryptographie, ...), il peut ou non lire ces données ;
- le *processeur des données* (ou responsable de traitement) est l'entité qui utilise les données, pour calculer des données dérivées, personnaliser le profil d'un client ou fournir tout autre service.

D'autre part, la littérature dans ce domaine a défini un ensemble de bonnes pratiques qui permettent de limiter les risques pour la vie privée des utilisateurs : le principe de *minimisation des données*. On relèvera en particuliers deux de ses composantes : la *collecte minimale* — il ne faut pas obliger un utilisateur à révéler plus de données personnelles que ce qui est strictement nécessaire au service dont il souhaite bénéficier — et la *rétention minimale* — un contrôleur ou un processeur ne doivent pas conserver une donnée plus longtemps que ce qui est strictement nécessaire pour effectuer le service demandé.

1.1.2 Droit à l'oubli

Le droit à l'oubli n'est à l'heure actuelle pas formellement inscrit dans les lois. Il est considéré comme un pendant du droit au respect de la vie privée, et non comme un droit autonome. Cependant, des réformes sont en cours, et l'Article 17 de la *Proposition de RÈGLEMENT DU PARLEMENT EUROPÉEN ET DU CONSEIL relatif à la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données (règlement général sur la protection des données)*[1] [2], émise par la Commission Européenne en janvier 2012, évoque un « Droit à l'oubli et à

2. Le texte complet et ses différentes traductions officielles sont disponibles ici : http ://eur-lex.europa.eu/LexUriServ/LexUriServ.do ?uri=CELEX :52012PC0011 :en :NOT

l'effacement ». Il peut se définir comme le droit dont disposerait tout individu de supprimer les traces numériques de ses actions.

Plus précisément, le sujet des données peut exiger d'un contrôleur qu'il supprime toutes les données personnelles le concernant. Il faut noter que toute personne physique, identifiée ou identifiable *même indirectement* à partir des données est considérée comme sujet d'après cette définition, et le terme « contrôleur » se réfère à l'organisme ou l'individu qui collecte ou utilise les données. Toute donnée « liée » à un sujet est considérée comme personnelle. Cependant, cette définition juridique ne permet pas de répondre de manière satisfaisante à toutes les questions relatives au droit à l'oubli, comme le montre un récent rapport de l'ENISA [3][9].

Portée : La définition même de ce qu'est une donnée personnelle est problématique. Aussi, la définition juridique est-elle volontairement générale, pour laisser la place à une interprétation contextuelle par un juge. Il s'agit d'une pratique standard en droit, mais pour une mise en application technologique du droit à l'oubli, il faudra trancher sur certaines questions. Une information permettant d'identifier un sujet avec une grande probabilité mais sans certitude est-elle personnelle ? Par exemple, une photo ou la liste des déplacements d'un individu permettent souvent (mais pas tout le temps) de déterminer cette identité même si elle n'est pas explicitement inscrite dans les données. De même, une information qui permet de déterminer l'appartenance d'une personne à un groupe, sa famille ou l'entreprise qui l'emploie par exemple, sans l'identifier de manière unique est-elle personnelle ?

Légitimité : Certaines données peuvent être liées à plusieurs sujets. Par exemple, Alice et Bob peuvent apparaitre ensemble sur une photo. Dans ce cas, comment doit s'appliquer le droit à l'oubli ? Comment trancher lorsque plusieurs sujets expriment des souhaits différents ? Plus généralement, il est difficile de trouver un équilibre satisfaisant entre le droit à l'oubli et d'autres droits fondamentaux tels que la liberté d'expression ou la liberté de la presse. Que doit faire un journaliste qui cite un *tweet* si l'auteur du *tweet* en question réclame son oubli ? Le journalisme, l'étude historique, l'enquête policière ou la recherche scientifique nécessitent tous une certaine conservation de données, bien que pour des raisons totalement différentes.

En conséquence, la nécessité d'un droit à l'oubli autonome — inscrit dans la loi pour lui-même et non plus le simple pendant d'autres droits, comme celui du respect de la vie privée — n'est pas une évidence pour les juristes.

3. *European Network and Information Security Agency*

Définition de l'oubli : Un autre problème fondamental est de définir exactement ce que signifie « oublier une information ». Dans son interprétation la plus stricte, la donnée originale, toutes les copies et toutes les informations dérivées devraient être supprimées. Ainsi, si Alice souhaite que ses réponses à un sondage soient effacées, toutes les statistiques tirées de ce sondage devraient être recalculées ou supprimées. Si ce retrait n'est pas effectué, il existe un risque théorique de pouvoir inférer la donnée brute en corrélant différentes formes dérivées, par exemples les résultats de plusieurs sondages auxquels Alice a répondu. Mais si le retrait est effectué, il faut aussi garantir l'oubli de l'ancien résultat ! Sinon, il est possible d'inférer les réponses d'Alice à partir de la différence entre les deux résultats.

En pratique, on considère généralement que le risque d'inférer une donnée brute à partir des formes dérivées est très faible alors que le retrait récursif de tous les dérivés d'une information est particulièrement complexe. Aussi, les interprétations suivantes ne tiennent-elles pas compte de ce problème.

Une interprétation moins stricte, mais plus pragmatique, autorise la survie de copies chiffrées de la donnée brute si elles ne peuvent plus être déchiffrées. Enfin, une interprétation encore plus libérale consiste à dire qu'une donnée est oubliée, même si des copies en clair existent encore, du moment qu'elles ne sont plus facilement accessibles de manière publique, notamment dans les résultats des moteurs de recherche ou lors de requêtes faites à une base de données.

1.2 Problématique

La définition du droit à l'oubli est donc une question complexe. En outre, une implémentation **complète** et **purement technique** du droit à l'oubli semble **impossible**, pour toute interprétation raisonnablement stricte. En particulier, à partir du moment où un utilisateur autre que le sujet accède (légitimement) à une donnée, il est ultimement impossible — de par la nature numérique de l'information — de l'empêcher d'en faire une copie, puis de diffuser la copie, même s'il efface d'abord l'exemplaire original de la donnée.

De plus, le grand public commence à prendre conscience des problèmes de vie privée. Les gens sont de plus en plus hostiles au fait de divulguer leurs informations personnelles, comme le montrent Herder et Kawase dans leur travaux [14]. Ainsi, la série de scandales sur les réseaux sociaux ces dernières années a érodé la confiance des utilisateurs envers les grandes entreprises qui collectent et manipulent habituellement leurs données personnelles.

Tout récemment, en avril 2013, un chercheur a réussi a récupérer des images

envoyées via Snapchat et soit-disant périmées [4]. Depuis, aucune annonce n'a été faite par le créateur de l'application. Plus ancien mais plus grave parce que volontaire, le cas de Hushmail est également parlant : Hushmail est un service en ligne d'e-mails chiffrés dont le slogan affirmait que même ses propres employés ne pouvaient pas lire les messages de ses utilisateurs. Pourtant en 2007, l'entreprise a transmis à une agence fédérale américaine les messages en clair de certains utilisateurs [5].

Ce manque de confiance du public est d'autant plus justifié que les intérêts des utilisateurs et des hébergeurs de données sont la plupart du temps divergents. Dans le modèle le plus courant à l'heure actuel, l'utilisateur souhaite bénéficier du service proposé tout en protégeant au maximum sa vie privée, tandis que l'hébergeur propose le service pour collecter des données personnelles qu'il pourra ensuite monétiser, via une régie publicitaire par exemple. Ainsi, le principe de minimisation des données est rarement appliqué et les délais de rétention sont quasi-systématiquement surévalués, voire non explicitement déclarés, sous prétexte de constituer un profil client, de fournir des services personnalisés plus performants, etc.

Pour minimiser ce conflit, un nouveau mécanisme a été proposé [2] : la *dégradation des données*, qui recoupe les principes de collecte et de rétention. Au lieu d'une situation binaire — soit conservée soit effacée —, une donnée peut exister dans différents états, différents niveaux de précision, qui protègent de manière plus ou moins forte la vie privée de l'utilisateur mais, inversement, permettent de pouvoir réaliser plus ou moins de services. Chaque niveau de précision peut ensuite être conservé plus ou moins longtemps.

Toutefois, à l'heure actuelle, les seuls systèmes de dégradation des données qui existent en pratique sont mis en œuvre par des SGBD [6] spécialement adaptés pour cette tâche. L'utilisateur doit donc faire confiance au contrôleur des données et le croire sur parole lorsque ce dernier déclare appliquer la dégradation.

Étant donné notre motivation de départ — réduire l'impact des intérêts divergents entre utilisateur et contrôleur —, cette hypothèse nous paraît difficile à admettre, et nous proposons ici une architecture qui offre des garanties techniques sur l'application de la dégradation des données, sans nécessiter l'existence d'un acteur de confiance dans le système, et fondée sur un système de publication éphémère décentralisé qui assure la dégradation des données.

Dans la suite de ce rapport, nous allons tout d'abord présenter les tech-

4. Voir R. Hickman, *Snapchat unveiled : An examination of Snapchat on Android devices*, http ://decipherforensics.com/publications.

5. Voir R. Singel. *Encrypted E-Mail Company Hushmail Spills to Feds.* http ://www.wired.com/threatlevel/2007/11/encrypted-e-mai/, 2007.

6. Système de Gestion de la Base de Données.

niques existantes qui permettent d'assurer un droit à l'oubli dans certains contextes (Chapitre 2), puis nous introduirons un mécanisme complémentaire : la dégradation des données (Chapitre 3). Ensuite, nous montrerons comment garantir la dégradation en utilisant des techniques de publication éphémère et nous proposerons une architecture générale combinant ces deux mécanismes (Cahpitre 4), que nous illustrerons par un exemple de réseau géo-social (Chapitre 5) avant de conclure (Chapitre 6).

2 État de l'art

Dans le chapitre précédent, nous avons montré qu'il y avait une forte demande pour le développement de solutions techniques garantissant le droit à l'oubli. Malgré le résultat d'impossibilité évoqué plus haut, diverses méthodes ont été développées. Si elles n'offrent pas une solution générale, elles permettent toutefois de résoudre partiellement le problème du droit à l'oubli, dans certains contextes bien définis.

Dans ce chapitre, nous allons passer en revue les méthodes existantes en fonction des application considérées, en progressant du contexte le plus étroit vers le plus général.

2.1 Oubli programmé

Toutes les techniques que nous étudions dans cette section ont en commun un point essentiel : un mécanisme d'auto-destruction est intégré au système. Ainsi, toute donnée est ultimement condamnée à disparaitre, et ce dès son introduction dans le système. Le droit à l'oubli est ici assuré par la nature même de la technique de publication ou de stockage des données. Certaines techniques permettent toutefois de prolonger activement la durée de vie d'une donnée, mais assurer un stockage à long terme n'est clairement pas l'objectif principal de ce type d'architecture.

Pour éviter de se reposer uniquement sur la bonne foi de l'hébergeur, ces techniques font intervenir des acteurs extérieurs. Une approche possible est par exemple de publier des données chiffrées et de garantir l'effacement des clés de chiffrement après un certain délai. Les deux systèmes de ce type les plus connus sont Ephemerizer [22] et Vanish [12].

2.1.1 Ephemerizer : une approche centralisée

Le concept d'Ephemerizer a été introduit par Perlman en 2005 [22]. Le système a été conçu au départ pour garantir la suppression d'e-mails ou de fichiers locaux après une date T. Dans ce qui suit, nous nous consacrons uniquement au cas des messages, par soucis de simplicité. En effet, on peut

simuler des fichiers locaux éphémères si on sait réaliser des messages éphémère en s'envoyant un message à soi-même.

Une solution naïve consisterait à envoyer des messages chiffrés dont la clé est stockée sur une carte à puce sécurisée (pour éviter que la clé soit compromise ou copiée par erreur) et à détruire la carte après la date T. Cependant, cette solution nécessite autant de cartes à puce sécurisées que de dates d'expiration différentes. Elle serait donc coûteuse (prix des cartes) et peu pratique (gestion d'un grand nombre de cartes), donc irréaliste.

L'idée centrale d'Ephemerizer est d'assigner la gestion des clés à un service tiers appelé *ephemerizer*, qui concentrera ainsi l'expertise cryptographique nécessaire tout en amortissant les coûts sur plusieurs utilisateurs et plusieurs messages. De plus le système garantit que même si l'*ephemerizer* ou le destinataire du message est *compromis*[1], aucun message dont la date d'expiration T est passée ne pourra être lu. Cette propriété est appelée confidentialité persistante, ou *Perfect Forward Secrecy* en anglais (PFS).

Considérons un scénario où un utilisateur, par exemple Alice, veut envoyer un message éphémère à un destinataire appelé Bob. Dans la suite, $\{M\}K$ désigne le message M chiffré à l'aide de la clé K. Le principe est le suivant :

- L'*ephemerizer* diffuse des triplets (clé éphémère publique, identifiant, date d'expiration) $= (K_{eph}, keyId, T)$. Lorsque T est passée, il oublie la clé privée associée à K_{eph} ;
- Alice choisit une date d'expiration T et une chaine aléatoire S. Elle utilise un algorithme de chiffrement symétrique pour obtenir $\{M\}S$, puis elle utilise à la fois K_{eph} et la clé publique de Bob K_{Bob} pour obtenir $\{\{S\}K_{Bob}\}K_{eph}$. Elle transmet ensuite à Bob $\Big(\{M\}S,\ \{\{S\}K_{Bob}\}K_{eph},\ keyId\Big)$.
- Bob transmet $\Big(\{\{S\}K_{Bob}\}K_{eph},\ keyId\Big)$ à l'*ephemerizer* et lui demande de déchiffrer $\{S\}K_{Bob}$;
- Si K_{eph} est encore valide, l'*ephemerizer* lui retourne effectivement $\{S\}K_{Bob}$, et Bob peut déchiffrer S avec sa clé privée, puis M à l'aide de S. Sinon, l'*ephemerizer* rejette la requête, le message est « oublié » puisqu'il ne peut pas être lu.

Pour respecter la propriété PFS annoncée, il ne faut pas stocker les données éphémères chiffrées uniquement avec des clés privées à long terme comme K_{Bob}. En effet, si on transmettait directement $\{\{M\}K_{Bob}\}K_{eph}$, un attaquant qui compromettrait Bob, même après la date T, pourrait déchiffrer M. Pour cette raison, on utilise un secret aléatoire S. Ainsi, si Bob est compromis, $\{M\}S$ ne peut quand même pas être déchiffré, et si K_{eph} a expiré,

1. On dit qu'un acteur du système est *compromis* si ses clés privées sont révélées, par un agent malveillant ou sur décision judiciaire, ou si ses communications sont écoutées.

l'*ephemerizer* ne peut plus déchiffrer $\{\{S\}K_{Bob}\}K_{eph}$. En conclusion, tout message dont la date d'expiration est passée au moment de la brèche de sécurité ne pourra pas être lu [2].

De nombreuses extensions d'Ephemerizer ont été développées, en particulier les deux suivantes :

IBE-based Ephemerizer : Il s'agit d'une amélioration proposée par Nair, Dashti, Crispo et Tanenbaum en 2007 [19], qui révèle une faille dans le protocole initial d'Ephemerizer et propose un schéma alternatif résistant à cette attaque. De plus, ce nouveau schéma utilise une technique cryptographique appelé chiffrement fondé sur l'identité (*Identity Based Encryption*, IBE [6]) où l'on peut choisir arbitrairement la clé publique utilisée pour chiffrer les données. Le destinataire des données doit ensuite solliciter la clé de déchiffrement auprès d'une autorité de confiance qui la calcule à la volée. Les auteurs utilisent cette propriété pour permettre à l'émetteur de choisir la date d'expiration qui sera encodée dans la clé IBE, au lieu de lui imposer une liste de dates pré-définies.

Timed-Ephemerizer : En 2009, Tang propose [25] une solution à la faille évoquée ci-dessus, ainsi qu'un modèle de sécurité rigoureux et prouvé mathématiquement, qui garantit que le système résiste aux attaques passives par écoute (même venant de l'*ephemerizer* lui-même) et qu'un destinataire compromis après l'expiration ne permet pas à l'attaquant de récupérer le message éphémère. Il introduit également dans le schéma de communication la possibilité de bloquer l'accès à un message **avant** une date pré-définie (*Timed-Release Encryption*), d'où le titre de l'article. Au lieu de donner uniquement la date d'expiration du message, on peut ainsi définir précisément une fenêtre pendant laquelle il sera disponible. Cette fenêtre ne débute pas forcément dès son émission.

2.1.2 Vanish : exploiter le P2P

Vanish, développé depuis 2009 par Geambasu, Kohno, Levy et Levy [12] vise à résoudre le même problème qu'Ephemerizer mais sans utiliser un tiers de confiance. En effet, l'approche de Perlman implique deux inconvénients : (a) il faut faire confiance à l'*ephemerizer*, qui peut potentiellement être malveillant et représente un point de vulnérabilité unique dans le système (ainsi, si l'*ephemerizer* est indisponible, un message ne peut pas être lu) ; (b) il faut

2. Nous supposons bien entendu que la technique cryptographique utilisée est suffisamment forte.

créer et maintenir un nouveau service. Ces deux inconvénients posent une barrière importante à l'adoption d'un tel système.

Les concepteurs de Vanish ont donc décidé d'utiliser des Tables de Hachage Distribuées (DHT, d'après le nom anglais) pour remplacer l'*ephemerizer*. Une DHT implémente de manière robuste une base de données stockant des paires (clé, valeur) sur un ensemble de nœuds d'un réseau pair-à-pair. Les DHT possèdent quatre propriétés particulièrement appropriées au contexte de Vanish :

1. Leur passage à l'échelle (plus d'1 million de nœuds pour la DHT de Vuze, un des clients BitTorrent les plus utilisés [10]) et la répartition géographique des nœuds à travers le monde entier les protègent efficacement d'adversaires légalement puissants et influents. En effet, un tribunal n'a pas forcément autorité pour exiger la divulgation de données stockées dans un autre pays et un gouvernement pourrait difficilement négocier avec l'ensemble de tous les autres pays impliqués ;
2. Une DHT fournit un stockage redondant et décentralisé, donc plus fiable qu'un serveur unique comme dans Ephemerizer. En effet, même si certains nœuds sont indisponibles, il y a une grande probabilité pour que la donnée requise soit disponible sur un autre nœud. Le système ne possède plus de point de défaillance unique. Un message a donc plus de chances d'être disponible jusqu'à sa date d'expiration ;
3. En même temps, le phénomène d'attrition (*churn* en anglais) garantit de manière probabiliste que le message sera bien supprimé et qu'aucune copie de sauvegarde ne survivra par erreur. Une DHT évolue en permanence, au fur et à mesure que des nœuds rejoignent ou quittent le réseau, changent d'identifiant ou nettoient leur mémoire interne, et les données qui ne sont pas activement répliquées disparaissent ainsi de manière irrévocable ;
4. Enfin, des infrastructures implémentant des DHT existent et sont déjà disponibles. Il n'y a donc pas besoin de développer et maintenir un nouveau service uniquement pour Vanish (contrairement à Ephemerizer).

Vanish repose également sur la technique du partage de secret de Shamir [24] qui permet de découper un message en n morceaux et de garantir que, pour un entier $k < n$ prédéfini : (a) tout sous-ensemble de k morceaux permet de reconstituer le texte initial ; (b) aucun sous-ensemble de $k - 1$ morceaux ne donne d'information sur le contenu du texte initial.

Pour envoyer un message éphémère avec Vanish, Alice chiffre le message en utilisant une clé K pour obtenir $\{M\}K$, elle coupe K en n parts et les répartit dans la DHT en utilisant une *clé d'accès* L choisie aléatoirement, et

elle envoie à Bob $(\{M\}K, L)$. Bob utilise alors L pour récupérer au moins k parts sur la DHT (k est prédéfini par Alice au départ) et reconstitue la clé K. Il peut ainsi déchiffrer $\{M\}K$.

Ici, le canal de communication entre Alice et Bob est supposé sécurisé : en effet, quiconque possède L peut reconstituer le message. Vanish, comme Ephemerizer, ne protège que les messages dont la date est expirée au moment de la brèche de sécurité. En pratique, le message $(\{M\}K, L)$ peut être simplement chiffré avec un logiciel comme PGP ou GPG [3] : c'est la suppression de K dans la DHT qui garantit que le message ne sera plus disponible au-delà d'une certaine date, et cela même si la clé de chiffrement GPG est compromise a posteriori.

Initialement, Vanish était proposé comme un plug-in Firefox et son implémentation reposait au choix sur Vuze [10], réseau ouvert avec plus d'un million de nœuds, ou sur un réseau à accès restreint, construit à partir d'OpenDHT [23], ce qui permet d'éviter certains types d'attaques. Par exemple, un attaquant peut entrer sur la DHT de Vuze et copier toutes les clés disponibles pour empêcher leur effacement. Avec OpenDHT, un attaquant externe à la DHT (donc utilisant simplement l'interface standard de la DHT) n'aura pas connaissance de la localisation des clés, qui est pseudo-aléatoire, et ce type d'attaque sera beaucoup plus difficile. Depuis, de nombreuses extensions ont été proposées :

Cascade [11] est un *framework* modulaire qui permet de répartir des parts de secret dans différents systèmes hétérogènes. De cette manière, chaque système apporte des garanties de sécurité complémentaires et un attaquant doit être capable de corrompre tous les systèmes pour reconstituer le message initial ;

Tide [11] est un système de stockage des parts s'appuyant sur Apache. Il permet notamment d'éviter les attaques par *crawling*, où un attaquant récupère activement un maximum de clés pendant leur période de validité, et les stocke pour un usage ultérieur (sans savoir lesquelles seront effectivement utiles) ;

Comet [13] est une DHT personnalisable en fonction de l'application visée. L'idée est qu'un objet stocké dans la DHT n'est pas qu'une valeur passive mais peut transporter avec lui un fragment de code qui va contrôler son

3. http ://www.gnupg.org/

comportement. Appliqué à Vanish, cela permet par exemple de contrôler précisément la réplication des clés, qui était trop agressive dans Vuze [4] ;

EphPub [8] est un système de publication éphémère qui stocke les clés dans des serveurs DNS au lieu d'utiliser une DHT. Les serveurs DNS sont naturellement résistants aux attaques Sybil, qui consistent à générer un grand nombre d'identités virtuelles pour contrôler une large proportion des nœuds d'un réseau distribué. Il s'agit d'une amélioration par rapport à Vanish car les DHT ouvertes comme celle de Vuze sont particulièrement sensibles à ce type d'attaques, même si les inventeurs de Vanish proposent déjà [11] des solutions pour limiter ce problème.

À noter que des plug-in Firefox et Thunderbird ont été publiés pour Vanish et EphPub. Ils ne sont toutefois pas maintenus et ne sont plus compatibles avec les versions actuelles.

2.1.3 Solutions grand public

Les systèmes précédents n'ont jamais dépassé le stade du prototype de recherche. Cependant, des systèmes de publication éphémère disponibles pour le grand public existent, et plusieurs ont récemment suscité l'intérêt.

Le premier, **X-pire !** [5], a été proposé dès 2011 pour publier des photos sur Facebook ou Flickr de manière éphémère. En effet, Facebook et Flickr (et la plupart des services actuels) ré-encodent les photos uploadées. Or Ephemerizer ou Vanish ne permettent pas la modification du chiffré et ne pouvaient donc pas être utilisé dans ce contexte. X-pire ! est une solution payante [5] fondée sur le principe d'Ephemerizer.

Peu de détails techniques sont disponibles sur **Poke** [6], développé de manière fermé par Facebook, mais le système semble également fondé sur Ephemerizer, avec Facebook dans le rôle du tiers de confiance central. Facebook précise cependant que la suppression des clés de chiffrement effectuée n'est pas parfaite et que des traces peuvent en subsister dans les logs ou les copies de sauvegarde pendant 90 jours. [7]

Parmi les solutions évoquées ici, **Snapchat** [8] est malheureusement à la

4. Vuze étant au départ destiné au partage de fichiers, les réglages utilisés favorisaient la disponibilité au détriment du contrôle précis de la durée de vie.

5. disponible ici : http ://www.backes-srt.de/produkte/x-pire/. Attention, site en Allemand.

6. https ://itunes.apple.com/fr/app/facebook-poke/id588594730 ?mt=8

7. http ://techcrunch.com/2012/12/22/your-facebook-pokes-are-stored-for-two-days-then-their-encryption-keys-are-deleted/

8. http ://www.snapchat.com/

fois la plus populaire [9] et celle qui offre le moins de garanties [10]. En réalité, l'application consiste simplement en une interface avec le système de fichier, et n'utilise ni cryptographie ni suppression sécurisée. Il est donc possible de récupérer les images même après leur prétendue expiration.

2.2 Oubli envisagé

Les techniques étudiées dans la section précédente garantissent que les données seront supprimées après une certaine date. Cependant, beaucoup de données ne sont pas a priori destinées à être supprimées. Par exemple, les données stockées dans un nuage sont souvent là pour être conservées sur le long terme. Pourtant, un utilisateur qui décide de supprimer un fichier veut être sûr que le fichier est réellement supprimé, et que le service d'hébergement n'en conserve pas une copie en dépit de sa demande de suppression, se contentant de retirer les liens publics vers le fichier.

Pour répondre à ces situations, un sujet a besoin d'un système qui lui permette de garder le contrôle sur ses données et de tracer les copies réalisées, pour être en mesure de supprimer une donnée et ses différentes copies s'il le souhaite, quel que soit l'hébergeur actuel. Les politiques adhésives (*sticky policies* en anglais) permettent de réaliser toutes ces fonctionnalités.

2.2.1 Concept général de politique adhésive

Le concept de politique adhésive a été introduit pour la première fois par Karjoth et Schunter [15], puis formalisé par Casassa Mont, Pearson et Bramhall [7] de la manière suivante :

- Les données personnelles du sujet sont chiffrées avant qu'il ne les diffuse ;
- Une politique d'utilisation est définie par l'utilisateur [11]. Elle est ensuite liée aux données de manière persistante et résistante à la falsification, d'où le terme de politique « adhésive » ;
- Les données ne peuvent être lues par un processeur potentiel que s'il satisfait aux exigences de la politique associée ;
- Pour pouvoir déterminer les responsabilités des différents processeurs et hébergeurs, toute divulgation consécutive des données est enregistrée et vérifiée.

9. http ://venturebeat.com/2013/04/16/snapchat-150m-images-a-day/
10. http ://rezonances.blog.lemonde.fr/2013/05/10/snapchat-ces-photos-ephemeres-qui-ne-seffacent-pas/
11. Ou un tiers de confiance agissant en son nom, comme un agent le représentant ou un moteur de politique automatisé. C'est un domaine de recherche à part entière, qui est connexe à notre sujet mais ne sera pas évoqué ici par manque de place.

La description ci-dessus est une spécification de haut niveau, purement fonctionnelle. Elle ne présage rien des moyens techniques à mettre en œuvre, mais décrit simplement les propriétés que doit satisfaire un système utilisant des politiques adhésives. Dans le même article, les auteurs suggèrent cependant quelques techniques qui seront détaillées plus loin.

Toutefois, il faut signaler que le passage du concept à l'implémentation est loin d'être évident. En effet, il est difficile de donner des garanties sur le respect des propriétés décrites plus haut, en particulier de s'assurer que les politiques resteront bien « adhésives » tout au long de l'utilisation des données et qu'un contrôleur qui divulgue les données à une tierce partie le reporte effectivement à l'autorité de vérification compétente. De plus, la spécification même des politiques n'est pas une tâche évidente.

Enfin, il faut noter que si les politiques adhésives permettent effectivement d'implémenter le droit à l'oubli, leur champ d'application est beaucoup plus vaste. Définir quand une donnée doit être supprimée n'est que l'un des nombreux aspects qui peuvent être spécifiés dans une politique d'utilisation des données et un utilisateur peut également définir les traitements autorisés sur ses données, s'il doit être notifié lorsque ces traitements sont effectués, quels sont les contrôleurs autorisés, etc.

2.2.2 Spécification des politiques

À l'heure actuelle, les politiques de confidentialité et les conditions d'utilisation des sites web ou des services en ligne sont toujours exprimées en langage naturel, la plupart du temps sous forme de termes juridiques [12]. Ainsi, elles sont à la fois inaccessibles à la compréhension de l'utilisateur moyen et non interprétables par une machine. Or pour toute mise en pratique réaliste des politiques adhésives, elles **doivent** être interprétables par une machine, afin d'automatiser le traitement décrit par Casassa Mont, Pearson et Bramhall [7]. Dans cette optique, différents langages et normes ont été développés :

- Le projet **P3P** (*Platform for Privacy Preference*) [20] est un standard développé par le consortium W3C, débuté en 2000 ;
- **XACML** (*eXtensible Access Control Markup Language*, actuellement en v3.0) [17] définit un schéma XML pour représenter des politiques de contrôle d'accès. Il est développé par OASIS, organisation encourageant l'adoption de standards pour le e-business et les services webs ;
- **PPL** (*PrimeLife Policy Language*) [26], une extension de XACML 2.0 qui permet d'exprimer de manière symétrique les préférences de l'utili-

12. Une exception notable : tumblr. Voir http ://gizmodo.com/5896017/tumblr-has-the-only-likable-terms-of-service-weve-ever-seen

sateur et les politiques appliquées par le contrôleur des données.

2.2.3 Garantie sur l'application des politiques

Garantir l'application des politiques adhésives pose deux problèmes distincts :

1. Comment vérifier que les politiques appliquées par un contrôleur ou un processeur sont compatibles avec les préférences d'un utilisateur avant de l'autoriser à lire ses données personnelles ?
2. Comment être sûr qu'un processeur, après avoir accédé aux données en clair, les chiffrera de nouveau avant de les transmettre à un tiers et rapportera bien la divulgation à l'autorité de vérification appropriée ?

Pour le premier point, il est possible d'utiliser des techniques de chiffrement IBE. Comme une clé publique IBE peut être une chaine de caractères arbitraire, on peut en particulier utiliser la politique d'utilisation des données comme clé publique. Ceci garantit que la politique est liée aux données chiffrées de manière résistante à la falsification, puisque si la politique est modifiée, la clé publique n'est plus la bonne et les données ne peuvent plus être déchiffrées. Le destinataire des données (un processeur) doit ensuite solliciter la clé de déchiffrement privée auprès d'une autorité de confiance, éventuellement distribuée, qui la calcule à la volée à partir de la clé publique non altérée. Si une technique de certification forte est mise en place (voir paragraphe suivant), cette autorité peut vérifier que le processeur satisfait les conditions exigées par la politique avant de retourner la clé. Toutefois, même en l'absence de certification, la clé publique utilisée est une trace fiable que le processeur s'est engagé à respecter la politique d'utilisation, et il pourra être tenu responsable des infractions constatées par la suite. On parle alors de *privacy through accountability* par opposition à *privacy by design*, que l'on pourrait traduire par « protéger la vie privée par la responsabilisation plutôt que par conception ». Casassa Mont, Pearson et Bramhall suggèrent d'utiliser cette technique [7].

Pour le second point, la technologie du *Trusted Computing* – littéralement « informatique de confiance » – semble être la voie privilégiée dans la littérature à l'heure actuelle [16]. Une des implémentations du *Trusted Computing* repose sur un dispositif résistant à la falsification qui dispose d'un processeur, d'une mémoire sécurisée, de primitives cryptographiques, d'une horloge interne et d'une source d'énergie indépendantes et appelé TPM (pour *Trusted Platform Module*). À l'aide de son TPM, une plateforme de confiance est capable de garantir qu'elle n'a pas été compromise, de chiffrer des données en les « scellant » au matériel et de témoigner de certaines propriétés tout

en conservant son anonymat [21]. En particulier dans le contexte qui nous intéresse, une plateforme de confiance peut attester qu'elle ne transmet les données personnelles que sous forme chiffrée et qu'elle ne diffusera pas la copie en clair après l'avoir déchiffrée.

2.3 Oubli sans mesure préventive

Il faut garder à l'esprit qu'à l'heure actuelle, les données sont persistantes par défaut, principalement de part leur nature numérique. Les solutions présentées dans les sections précédentes permettent, pour certains cas d'utilisation ou certains contextes, de changer ce comportement par défaut, mais elles ne sont pas toujours applicables. En particulier, elles exigent de prendre des précautions au moment de la publication, et ne peuvent être appliquées rétroactivement pour toutes les données déjà existantes. Il est donc aussi fondamental que les utilisateurs prennent conscience des problèmes de vie privée et assimilent ce que représente la publication de données en ligne. Le rapport de l'ENISA [9] souligne bien qu'un changement de comportement est nécessaire et doit être encadré par une évolution politique et juridique.

Toutefois, si les données sont aujourd'hui conservées indéfiniment, elles ne sont pas toujours exploitables. L'avènement du web collaboratif et de l'« Internet des objets » [4] a provoqué une explosion de la quantité d'informations disponibles, et certains spécialistes parlent aujourd'hui d'un « déluge de données ». Pour accéder aux données, les utilisateurs se reposent de plus en plus sur des intermédiaires, par exemple les moteurs de recherche comme Google ou Bing, ou encore des systèmes de recommandations du type de Reddit. En un certain sens, une donnée qui est disponible en ligne en accès direct, mais qui n'est pas trouvable (ou difficilement) par les intermédiaires sus-mentionnés peut donc être considérée comme « oubliée » car non-visible.

2.3.1 Intervention judiciaire

Plusieurs décisions judiciaires ont demandé à des moteurs de recherche de censurer leurs résultats en se fondant sur ce raisonnement [13]. Il faut cependant noter que cette approche n'est qu'une réparation a posteriori et ne peut être mise en œuvre que si l'atteinte à la vie privée a déjà eu lieu.

De plus, elle ne passe pas à l'échelle en raison des délais et des lourdeurs du système judiciaire, ainsi que des différences de législation entre pays. Enfin, elle ne s'applique qu'aux cas graves. D'un point de vue scientifique, ce n'est

13. Voir par exemple http ://www.legavox.fr/blog/maitre-anthony-bem/google-condamne-supprimer-referencement-contenus-4764.htm

donc pas une direction de recherche très satisfaisante. Malgré tout, c'est aujourd'hui ce qui se rapproche le plus d'une mise en pratique effective d'un « droit à l'effacement ».

2.3.2 Enfouissement

Plusieurs sociétés de services proposent également aux internautes de gérer leur réputation en ligne et de camoufler des données personnelles compromettantes sous des données anodines [14]. Mais parfois, ces sociétés proposent également de retrouver toutes les informations en ligne concernant une personne donnée. Si les mêmes individus s'occupent à la fois de cacher des données puis de les retrouver, on peut légitimement s'interroger sur l'efficacité des services proposés. Dans tous les cas, il s'agit de manipulation des mécanismes d'accès, via intervention judiciaire ou optimisation des paramètres de recherche sur les données anodines.

Preuve que la question suscite l'intérêt du grand public, la compagnie d'assurance AXA a récemment lancé une offre protégeant la « e-réputation » [15]. Mais là encore, il ne s'agit que d'accompagnement pour les démarches administratives et juridiques, pas de solutions techniques.

14. https ://www.123people.com/
15. http ://www.assurances-ereputation.fr/2012/01/assurance-e-reputation-axa-protection-familiale-integrale/

3 Application de la dégradation des données au droit à l'oubli

Dans le chapitre précédent, nous avons détaillé les différentes techniques applicables au droit à l'oubli et leurs contextes respectifs. Toutefois, il faut bien avoir conscience que l'application de ces techniques nécessite la coopération des hébergeurs alors que le droit à l'oubli est contraire aux intérêts de ces derniers. De plus, même si un hébergeur accepte de faire des concessions, il n'est pas garanti que les utilisateurs lui fassent suffisamment confiance pour le croire sur parole lorsqu'il affirme effacer des données.

Il est donc nécessaire d'introduire de nouveaux mécanismes permettant :

- d'une part, de réconcilier, au moins partiellement, les intérêts contradictoires des hébergeurs et des utilisateurs ;
- d'autre part, de donner des garanties techniques à l'utilisateur sans exiger qu'il fasse confiance à l'hébergeur.

Après avoir présenté le mécanisme de dégradation des données tel qu'il a été initialement proposé, nous montrons dans ce chapitre qu'il permet de réconcilier les intérêts des deux partis, avant de développer une extension qui facilitera son utilisation dans notre contexte. Le problème de la confiance sera traité au chapitre suivant.

3.1 Durée de rétention optimale

En 2009, van Heerde, Fokkinga et Anciaux ont développé un modèle économique de la valeur d'une donnée [28]. Dans ce modèle, pour le sujet, la valeur d'une donnée est d'autant plus haute que sa vie privée est respectée. Du point de vue du contrôleur, plus la précision et le délai de rétention sont grands, plus la donnée est utile. La *valeur totale* d'une donnée est alors définie comme le produit des valeurs pour le sujet et pour le contrôleur.

En l'absence de dégradation, la valeur totale dépend uniquement de la durée de rétention t et les auteurs démontrent que sous des hypothèses raisonnables (en particulier, monotonie des fonctions de valeur du sujet et du contrôleur), une durée de rétention optimale existe.

3.2 Dégradation progressive

De plus, ils affirment qu'un mécanisme de *dégradation progressive* permet d'augmenter encore cette valeur. L'idée centrale derrière la dégradation est qu'une même donnée peut avoir plusieurs niveaux de précision, pas seulement un état binaire (présente ou effacée). Les niveaux les plus précis ont plus de valeur pour le responsable de traitement mais sont plus sensibles en terme de vie privée pour l'utilisateur. À l'inverse, les niveaux fortement dégradés protègent efficacement la vie privée mais n'ont pas une utilité importante. Par exemple, la géolocalisation d'un utilisateur peut être indiquée comme une coordonnée GPS précise au mètre, comme un quartier, une ville ou même un pays. Bien entendu, il est plus sensible pour l'utilisateur de révéler sa coordonnée GPS exacte que son pays, mais une localisation précise permet en contrepartie de lui envoyer des offres promotionnelles à proximité.

Au lieu de supprimer simplement une donnée après un temps t, on peut alors définir un *scénario de dégradation progressive* : à n durées $t_1, ..., t_n$ (avec $t_1 \leq t \leq t_n$) on associe n opérations de dégradation progressives (par ex. ajout de bruit ou généralisation), la n^e opération étant généralement la suppression complète de la donnée. Le cas $n = 1$ correspond à la suppression simple. Un scénario judicieusement choisi permet d'atteindre une valeur totale supérieure au scénario de suppression simple. En effet, la valeur pour le sujet commence à croître plus tôt, tandis que la valeur pour le contrôleur décroît plus lentement. Le gain en valeur totale dépend des vitesses de variation relative entre les fonctions de valeur du sujet et du contrôleur. Pour un scénario de dégradation optimal, qui existe forcément, il est parfois négligeable mais jamais négatif. Le mécanisme de dégradation des données permet donc effectivement de réconcilier les intérêts des hébergeurs et des utilisateurs.

Toutefois, cette étude purement théorique laisse en suspens le problème de la quantification précise de la valeur d'une donnée (*i.e.* comment déterminer les fonctions de valeur), qui est nécessaire pour calculer le scénario optimal. De plus, van Heerde propose une implémentation du système de dégradation qui repose entièrement sur le contrôleur, avec une sémantique SQL légèrement modifiée et un SGBD spécialement adapté. Cette approche ne résout donc pas le problème de confiance évoqué plus haut.

3.3 Extension de la dégradation

Les travaux de Van Heerde *et al.* [2, 3, 28, 27] considèrent que la dégradation des données est, sur le plan théorique, un processus temporel. Certaines des implémentations proposées pré-dégradaient les données au moment de

leur insertion dans la base de données, mais ils définissent le cycle de vie d'une donnée comme une suite de dates et d'opérations de dégradation associées.

De notre côté, ce point de vue nous paraît limitatif et nous proposons de considérer la dégradation comme un processus plus riche, qui peut par exemple être lié à une utilisation. Ainsi, chaque utilisation d'une donnée ne doit accéder qu'au niveau de précision minimal nécessaire pour accomplir le service : on retrouve le principe de minimisation. De plus, associer de manière explicite un niveau de précision à une utilisation, et non plus à une durée de conservation, permet de mieux comprendre les enjeux de ce niveau, donc de choisir l'opération de dégradation associée avec plus de pertinence.

3.4 Les pratiques de la dégradation à l'heure actuelle

Contrairement au principe de minimisation des données, la dégradation ne fait pas partie des grands principes admis dans la littérature sur la vie privée. Toutefois, la législation autorise dans certains cas l'anonymisation des données en lieu et place de leur suppression une fois la durée de rétention maximale atteinte. En particulier, la plupart des moteurs de recherches [1], ont fait ce choix concernant leurs archives. Cette anonymisation peut être vue comme une dégradation, mais elle n'est pas de même nature que celle évoquée précédemment : ici, il s'agit de retirer les liens entre différentes données alors qu'avant il s'agissait de dégrader une information indépendamment des autres données l'accompagnant.

1. Pour une rapide comparaison, voir http ://blogs.technet.com/b/microsoft_on_the_issues/archive/2009/02/search-data-retention-policies-of-major-search-engines-before-the-eu.aspx. Attention, l'« anonymisation complète » revendiquée par Microsoft n'est pas détaillée donc sujette à caution.

4 Garantir la dégradation des données

Dans le chapitre précédent, nous avons montré que la dégradation des données permet de réconcilier les intérêts des hébergeurs et des utilisateurs. Cependant, si la dégradation est mise en œuvre par les hébergeurs, les utilisateurs risquent de ne pas faire confiance au système. Dans ce chapitre, nous proposons d'utiliser les techniques de publication éphémère présentées en section 2.1 pour garantir techniquement l'application de la dégradation et nous présentons une architecture détaillée qui combine ces deux mécanismes.

4.1 Acteurs en présence

Nous considérons des systèmes qui mettent en relation des utilisateurs (ou *sujets*) et des hébergeurs (ou *contrôleurs*). Ces systèmes peuvent être de purs systèmes d'hébergement (par ex. Dropbox [1]) ou des réseaux sociaux (par ex. Facebook [2], Foursquare [3]) qui hébergent les données relatives à l'utilisation du système (liste des pages « aimées » par l'utilisateur, des lieux visités, etc.).

En réalité, le rôle de l'hébergeur est souvent double : en plus d'être contrôleur des données, il en est aussi le *processeur*, c'est-à-dire celui qui effectue des traitements sur les données pour les valoriser. En particulier dans les réseaux sociaux, les données que l'utilisateur publie permettent de lui construire un profil que le réseau social pourra ensuite monétiser (auprès d'une régie publicitaire par exemple).

Par la suite, nous distinguerons systématiquement les rôles de contrôleur et de processeur, même s'il s'agit de la même entité dans les deux cas. De plus, pour des services multi-utilisateurs comme les réseaux sociaux, on modélisera les interactions entre un utilisateur et ses voisins dans le réseau par des interactions entre un utilisateur et des responsables de traitement autorisés à consulter ses données, même si le traitement se limite à la seule consultation.

1. https ://www.dropbox.com/
2. https ://www.facebook.com/
3. https ://foursquare.com/

4.2 Modèle d'attaque

L'adversaire considéré ici est un hébergeur qui ne respecte pas la vie privée de ses utilisateurs et n'applique pas le principe de minimisation des données. Au contraire, il tentera d'accumuler un maximum de connaissance sur ses utilisateurs. Toutefois, nous supposons qu'il restera dans le cadre d'un fonctionnement normal du service fourni, et qu'il ne cherchera pas pro-activement à accéder à des données confidentielles. Plus formellement, on se place donc dans le modèle d'attaque d'un hébergeur *honnête-mais-curieux* (parfois aussi appelé *semi-trusted*, mais ce terme est ambigu selon les auteurs).

Les responsables de traitement n'ont pas tous accès à toutes les données, mais ils sont supposés être de confiance *relativement* aux données auxquelles ils ont accès. Cette hypothèse est due à l'utilisation de système de publication éphémère : les entités qui accèdent légitimement à une copie en clair des données ne doivent pas en effectuer de copies. Dans le cas où l'hébergeur est aussi un responsable de traitement, on considère que la durée de vie du niveau de précision auquel il accède est infinie, et qu'il peut donc en faire une copie sans que cela porte à conséquence.

4.3 Cas pratique

À des fins d'illustrations, nous considérons dans la suite un exemple d'application spécifique. Nous avons choisi de nous intéresser à un réseau géo-social fictif, inspiré de Foursquare. En effet, les données géo-localisées se prêtent relativement bien à la dégradation des données [4], comme l'illustrent les quelques exemples suivants :

- un point GPS peut facilement se transformer en un intervalle rectangulaire, ou être brouillé par un ajout de bruit aléatoire ;
- une date peut se généraliser de la minute à l'heure ou à la journée ;
- un lieu précis peut être remplacé par sa catégorie, par exemple « La Pagode Dorée » deviendrait « restaurant asiatique » ou même simplement « restaurant ».

Nous montrerons que la majorité des services disponibles sur une plateforme du type Foursquare restent réalisables dans notre proposition, tout en protégeant bien mieux la vie privée des utilisateurs :

- géo-localisation des amis ;
- partage de conseils pour un lieu donné ;

4. Au moins dans un premier temps. Les dégradations triviales décrites ici peuvent être sensibles à des attaques par inférence utilisant des connaissances auxiliaires, dont l'étude dépasse toutefois le cadre de ce travail.

– offres promotionnelles ciblées par localisation ;
– etc.

4.4 Description de notre architecture

L'idée générale de notre système est de découpler strictement les rôles de contrôleur et de processeur. Ainsi, en utilisant un système de publication éphémère, un utilisateur envoie au contrôleur uniquement la version chiffrée des données et transmet directement à chaque processeur la clé nécessaire au déchiffrement des données auxquelles il a droit. La Figure 4.1 présente l'architecture générale qui sera détaillée ci-dessous.

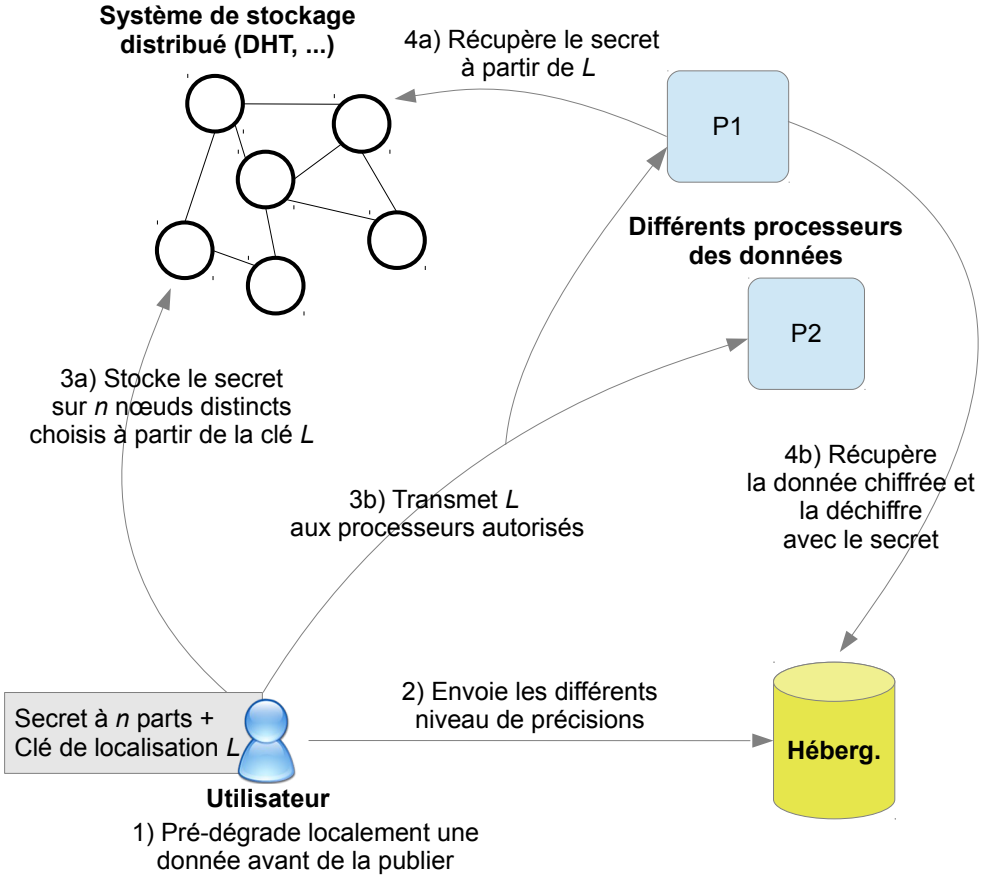

FIGURE 4.1: Architecture générale de notre solution.

4.4.1 Dégradation a priori

Pour respecter le principe de minimisation des données, nous avons montré précédemment que chaque utilisation d'une donnée doit être lié à un niveau de précision et une date d'expiration spécifiquement choisis par l'utilisateur.

En conséquence, nous exigeons que les données soient pré-dégradées côté utilisateur et que chaque niveau de précision soit chiffré séparément (étape 1 de la Figure 4.1) avant d'être envoyé au système d'hébergement (étape 2).

4.4.2 Hébergement

L'hébergement dans les systèmes centralisés actuels est la plupart du temps assuré par une base de données, car l'hébergeur étant aussi un processeur, il peut profiter des requêtes complexes autorisées par un tel système. Dans notre proposition, l'hébergement se fait sous forme chiffrée et la recherche et l'agrégation de différentes entrées n'est donc plus possible.

Par conséquent, il est possible de remplacer la base de donnée par un dictionnaire clé-valeur. Chaque publication de l'utilisateur est repérée par une clé unique, qui peut être transmise en même temps que la clé de localisation.

4.4.3 Gestion des clés

Chaque clé de chiffrement est tout d'abord coupée en n parts en utilisant un partage de secret de Shamir. Puis les parts sont stockées, chacune sur un nœud distinct d'une structure distribuée, par exemple une DHT dans un système de type Vanish, les nœuds étant choisis à partir d'une clé de localisation aléatoire L (étape 3a).

L'utilisateur doit ensuite transmettre les clés de localisations (et les identifiants des données associées) aux processeurs autorisés (étape 3b). Cette transmission doit être :

- sûre : si la clé de localisation est interceptée par un adversaire, il peut interroger la structure distribuée et reconstruire la clé de déchiffrement, puis accéder aux données. Autrement dit, il peut effectuer les étapes 4a et 4b comme s'il était un processeur légitime pour les données associées à la clé interceptée ;
- asynchrone : les responsables de traitement, et plus encore les utilisateurs, ne sont pas forcément toujours disponible en ligne.

On peut imaginer diverses solutions pour remplir ces contraintes. La plus simple est d'envoyer un courrier électronique, chiffré avec GPG ou un autre programme équivalent, contenant clé et identifiant. Même si l'intégration au système global n'est pas parfaite et poserait problème pour une adoption à

large échelle, cette solution est suffisante pour un prototype de démonstration. Dans l'idéal, cependant, l'envoi et la réception des clés seraient intégrés à l'interface du service et exécutés de manière transparente pour l'utilisateur.

Il est également possible d'utiliser des solutions plus avancées, en fonction de la confiance accordée à l'hébergeur. Par exemple, si seule la clé de localisation est chiffrée mais que le destinataire et l'identifiant de la donnée sont en clair, alors l'hébergeur peut potentiellement optimiser la localité de ses structures de stockage et son utilisation de bande passante. L'étude détaillée des différentes variantes dépasse cependant le cadre de ce travail.

4.4.4 Traitement des données

Une fois qu'il a reçu clé de localisation et identifiant, le responsable de traitement peut reconstruire la clé de chiffrement à partir de la structure distribuée (étape 4a) et interroger l'hébergeur avec l'identifiant associé pour récupérer la donnée correspondante (étape 4b).

4.5 Avantages

Notre solution met en œuvre la dégradation des données et garantit son application à l'utilisateur, même sans la coopération pleine et entière de l'hébergeur, grâce à un système de publication éphémère de type Vanish. Il s'agit d'une architecture partiellement distribuée, qui ne peut pas être ajoutée directement comme une sur-couche dans les grands services actuels (Facebook, Google, etc.). Cependant, elle présente des avantages pour les deux parties :

- l'utilisateur bénéficie d'une bien meilleure protection de sa vie privée et d'un contrôle plus direct sur qui a accès à quelles données ;
- l'hébergeur peut gagner de nouveaux utilisateurs qui n'utilisaient pas du tout le service par manque de confiance. Et, sous réserve que le schéma de dégradation soit bien choisi, il peut conserver des données avec toute leur valeur pratique pendant plus longtemps.

Nous pensons donc que notre solution est une alternative viable aux systèmes complètement centralisés actuels et qu'il est envisageable de la voir déployée à grande échelle.

4.6 Limitations et extensions possibles

L'architecture générale présentée ci-dessus présente certaines limitations, malgré les avantages que nous avons mis en avant jusqu'ici. Notamment, elle

ne rentre pas dans la spécification de certains détails, trop dépendants de l'application visée. Nous tenons cependant à attirer l'attention du lecteur sur ces questions qui ont leur importance.

La décision de lier chaque niveau de précision à l'utilisation qui va en être faite permet de faciliter le choix de la dégradation à appliquer, mais cette approche présente un inconvénient. Il devient très difficile de proposer un schéma de dégradation a priori, par exemple à partir des préférences de l'utilisateur, et qui conviendrait pour toutes les utilisations potentielle de la donnée.

Chaque fois qu'une nouvelle utilisation est envisagée, le processeur doit donc contacter l'utilisateur pour qu'il donne son accord et choisisse l'opération de dégradation appropriée. Une piste possible pour résoudre ce problème serait de déterminer automatiquement si l'un des niveaux de précision existant convient à la nouvelle utilisation et si l'utilisateur accepte de transmettre la clé associée au processeur. Les travaux effectués dans le domaine de l'automatisation des politiques d'utilisation [ref] laissent supposer qu'une telle automatisation sera possible à moyen terme.

5 Vers un réseau géo-social avec dégradation des données

Dans le chapitre précédent, nous avons proposé une architecture théorique, présenté son fonctionnement et ses limitations. Dans ce chapitre, nous allons utiliser cette architecture pour concevoir un simulateur de réseau géo-social simplifié. Dans la première section, cette application concrète va nous permettre de détailler les points délicats seulement mentionnés dans le modèle général. L'objectif à terme est d'implémenter un prototype de démonstration, mais ce travail est encore en cours. Nous présenterons quelques éléments de cette implémentation dans la seconde section.

5.1 Scénario de la simulation

Nous nous plaçons du point de vue d'un utilisateur que nous appellerons l'*utilisateur principal*. Cet utilisateur souhaite participer à un réseau géo-social. Il va donc publier son activité en ligne après avoir effectué localement les dégradations appropriées. L'activité élémentaire que nous considérons par la suite est un *check-in*, constitué des informations suivantes avec entre parenthèses les différents niveau de précision possibles :

- coordonnée GPS (généralisation par troncature des décimales) ;
- lieu (nom, catégories suivant un arbre de généralisation pré-défini) ;
- adresse (pays, ville, quartier, rue, numéro).

Une fois dégradées et publiées, ces données doivent permettre de réaliser les fonctionnalités suivantes :

- localiser l'utilisateur ;
- lui envoyer des offres promotionnelles à proximité ;
- établir un profil publicitaire personnalisé.

Bien entendu, ces différentes fonctionnalités ne seront pas utilisées par les mêmes acteurs, et en fonction de la confiance que l'utilisateur principal accorde à chacun d'entre eux, il choisira des opérations de dégradation plus ou moins fortes et des délais d'expiration plus ou moins long. Le Tableau 5.1

récapitule les contraintes pour chaque fonction et le schéma de dégradation proposé.

Fonction	localisation	promotion	profilage
Utilisée par	amis	commerçants	hébergeur
Niveau de confiance	élevé	faible-moyen	faible
Dégradation appliquée	faible	moyenne	importante
Durée de conservation	courte	courte	infinie
Valeurs possibles	coordonnées GPS (exacte) 1 heure	adresse (quartier) 1 heure	lieu (catégorie) pas d'expiration

TABLE 5.1: Contraintes et schéma de dégradation proposé

Ces fonctionnalités, bien que largement simplifiées par rapport à un système réel, permettent déjà de mettre en évidence certains mécanismes intéressants. Tout d'abord, même si l'on a confiance en certains processeurs, le principe de rétention minimale exige d'utiliser la durée de conservation la plus courte possible. Ainsi, pour la fonctionnalité de localisation, même si l'utilisateur fait confiance à ses amis, il ne sert à rien de garder l'historique de déplacement pour retrouver l'utilisateur à un moment donné : la durée de conservation a été fixée à 1 heure.

D'autre part, l'utilisateur dispose de deux leviers pour se protéger des processeurs auxquels il ne fait pas confiance. Par exemple, les commerçants ont besoin d'un niveau de précision élevé, c'est donc la durée de conservation qui est choisie très courte. À l'inverse, l'hébergeur honnête-mais-curieux conserve potentiellement les données sans limite de durée. On lui donne donc accès qu'à des données très dégradées.

Enfin, il faut noter que toutes les informations d'un *check-in* ne sont pas pertinentes pour toutes les fonctionnalités. D'après le principe de collecte minimale, on ne transmettra que les informations nécessaires. Dans notre exemple, chaque processeur n'accède en fait qu'à un seul attribut.

En conclusion, ce scénario illustre bien que l'utilisation combinée de dégradation et de publication éphémère redonne à l'utilisateur le contrôle de ses données et lui permet de mettre en application le principe de minimisation des données. Toutefois, le grand nombre de degrés de libertés dans la sélection du schéma de dégradation approprié

5.2 Remarques sur l'implémentation

L'architecture proposée fait intervenir de nombreux sous-composants. Nous présentons ici les options à notre disposition et les décisions que nous avons prise.

5.2.1 Publication éphémère

Les sources du projet Vanish ayant été publiées sous une licence académique, nous avons décidé de le ré-utiliser. Il a fallu ensuite choisir quelle plateforme utiliser pour stocker les parts de secret. Les sources disponibles contenaient un back-end pour VuzeDHT ou OpenDHT, mais OpenDHT a été mis hors-ligne en 2009. Nous nous sommes donc tourné vers Vuze, qui est un logiciel grand public (un client BitTorrent) encore actif aujourd'hui. Cependant, Vanish n'a pas été maintenu depuis 2010 tandis que Vuze continuait d'évoluer. Un travail non négligeable a donc été nécessaire pour ré-adapter Vanish à la nouvelle version de Vuze.

5.2.2 Machines distantes

Pour simuler des machines distantes en réseau, nous proposons d'utiliser des processus Java communiquant par RPC [1]. Cette solution permet de simplifier la gestion du réseau et de bénéficier des facilités proposées par le langage. Les processus qui représentent des utilisateurs fournissent une interface graphique basique mais adaptée à leur rôle : l'utilisateur principal a accès aux différents paramètres du schéma de dégradation et ses amis dans le réseau peuvent consulter sa position. Le processus représentant l'hébergeur peut utiliser les collections disponibles dans la librairie standard pour représenter le dictionnaire contenant les données de l'utilisateur.

1. *Remote Procedure Call.*

6 Conclusion

La protection de la vie privée est un droit fondamental, mentionné notamment dans la Déclaration Universelle des Droits de l'Homme. Pourtant, ce droit est de plus en plus menacé, car les technologies numériques modernes permettent une dissémination de l'information très rapide. Il est donc nécessaire de développer des solutions pour proposer un droit à l'oubli aux utilisateurs imprudents. Si une solution générale semble inaccessible à l'heure actuelle, nous avons montré dans ce rapport que des solutions techniques existent dans certains contextes précis, en particulier lorsque des précautions sont prises au moment de la publication de la donnée.

Ainsi, les techniques de publication éphémère permettent de fixer dès le départ une date au-delà de laquelle les données seront oubliées, tandis que les politiques adhésives permettent à un utilisateur de garder le contrôle sur la dissémination et l'utilisation de ses données. Elles sont également complétées par des approches juridiques et des techniques empiriques de manipulation des mécanismes d'accès aux données, qui, sans donner de garanties rigoureuses, permettent parfois d'oublier une donnée, même si toutes ses copies ne sont pas effacées.

L'application des ces techniques repose toutefois sur des hypothèses fortes (tiers de confiance, matériel sécurisé, ...) et se heurtent à deux obstacles majeurs : d'une part, il n'est pas dans l'intérêt des hébergeurs de mettre en place ces techniques, d'autre part, les utilisateurs font de moins en moins confiance aux hébergeurs.

Pour résoudre ce problème, nous proposons de combiner deux mécanismes : un système de dégradation des données a priori par l'utilisateur, qui permet de garantir l'application du principe de collecte minimale et un système de publication éphémère, qui assure le respect de la rétention minimale. Nous présentons également une architecture générale qui permet la mise en place de ces mécanismes, et une application concrète à un réseau géo-social qui permet d'illustrer les décisions que doit prendre un utilisateur.

L'implémentation du droit à l'oubli reste un problème ouvert et des efforts importants sont nécessaires en terme d'intégration et d'adoption des technologies. Une solution purement technique est improbable, et la recherche en informatique doit être accompagnée par des évolutions juridiques et sociales..

Bibliographie

[1] Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation), 2012.

[2] Nicolas Anciaux, Luc Bouganim, Harold Van Heerde, Philippe Pucheral, and Peter M.G. Apers. Data Degradation : Making Private Data Less Sensitive Over Time. In *Proceedings of the 17th ACM Int. Conf. on Information and Knowledge Management (CIKM)*, pages 1401–1402, 2008.

[3] Nicolas Anciaux, Luc Bouganim, Harold van Heerde, Philippe Pucheral, and Peter M. G. Apers. InstantDB : enforcing timely degradation of sensitive data. *Proceedings of the 24th IEEE Int. Conf. on Data Engineering (ICDE)*, pages 1373–1375, 2008.

[4] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things : A survey. *Computer Networks*, 54(15) :2787–2805, October 2010.

[5] Julian Backes, Michael Backes, Markus Dürmuth, Sebastian Gerling, and Stefan Lorenz. X-pire ! : A digital expiration date for images in social networks. Technical report, 2011.

[6] Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3) :586–615, January 2001.

[7] Marco Casassa Mont, Siani Pearson, and Pete Bramhall. Towards accountable management of identity and privacy : Sticky policies and enforceable tracing services. *DEXA*, pages 377–382, 2003.

[8] Claude Castelluccia, Emiliano De Cristofaro, Aurelien Francillon, and Mohamed-Ali Kaafar. EphPub : Toward robust Ephemeral Publishing. In *Procs 19th ICNP*, October 2011.

[9] Peter Druschel, Michael Backes, and Rodica Tirtea. The right to be forgotten – Between expectations and practice. *ENISA*, 2011.

[10] Jarret Falkner, Michael Piatek, John P. John, Arvind Krishnamurthy, and Thomas Anderson. Profiling a million user DHT. In *Procs 7th SIGCOMM*, 2007.

[11] Roxana Geambasu, Tadayoshi Kohno, Arvind Krishnamurthy, Amit Levy, Henry Levy, Paul Gardner, and Vinnie Moscaritolo. New directions for self-destructing data systems. Technical report, 2011.

[12] Roxana Geambasu, Tadayoshi Kohno, Amit A. Levy, and Henry M. Levy. Vanish : Increasing data privacy with self-destructing data. *Procs 18th USENIX Security*, 2009.

[13] Roxana Geambasu, Amit Levy, Yoshi Kohno, Arvind Krishnamurthy, and Hank Levy. Comet : an active distributed key-value store. *Procs 9th OSDI*, 2010.

[14] Eelco Herder and Ricardo Kawase. Considerations for recruiting contributions to anonymised data sets. *IJTEL*, X, 2012.

[15] Günter Karjoth, Matthias Schunter, and Michael Waidner. Platform for enterprise privacy practices. In *Procs 2nd PET Workshop*, 2002.

[16] Gina Kounga and Liqun Chen. Enforcing sticky policies with TPM and virtualization. In *Procs 3rd InTrust*, 2011.

[17] Andreas Matheus. How to declare access control policies for XML structured information objects using OASIS' extensible access control markup language (XACML). *Procs 38th HICSS*, 2005.

[18] Viktor Mayer-Schönberger. Useful void : The art of forgetting in the age of ubiquitous computing. *Working Paper, John F. Kennedy School of Government, Harvard University*, (April), 2007.

[19] Srijith K. Nair, Mohammad T. Dashti, Bruno Crispo, and Andrew S. Tanenbaum. A hybrid PKI-IBC based ephemerizer system. *Procs 22nd IFIP SEC*, 2007.

[20] Muyiwa Olurin, Carlisle Adams, and Luigi Logrippo. Platform for privacy preferences (P3P) : Current status and future directions. In *Procs 10th PST*, July 2012.

[21] Siani Pearson. Trusted computing : Strengths , weaknesses and further. In *Procs 3rd iTrust*. 2005.

[22] Radia Perlman. The Ephemerizer : Making data disappear. *Journal of Information System Security (JISSec)*, 2005.

[23] Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu. OpenDHT : a public DHT service and its uses. *Procs 5th SIGCOMM*, 2005.

[24] Adi Shamir. How to share a secret. *Communications of the ACM*, November 1979.

[25] Qiang Tang. From Ephemerizer to Timed-Ephemerizer : Achieve assured lifecycle enforcement for sensitive data (extended version). *initially EUROPKI*, 2009.

[26] Slim Trabelsi, Akram Njeh, Laurent Bussard, and Gregory Neven. PPL engine : A symmetric architecture for privacy policy handling. *W3C Workshop on Privacy and data usage control*, October :5, 2010.

[27] Harold van Heerde. *Privacy-aware data management by means of data degradation : making private data less sensitive over time*. PhD thesis, University of Twente, Enschede, The Netherlands, June 2010.

[28] Harold van Heerde, Maarten Fokkinga, and Nicolas Anciaux. A framework to balance privacy and data usability using data degradation. In *Procs 12th CSE*. Ieee, 2009.