2010-2011

Olivier MARIDAT
Elève ingénieur en bi-cursus à l'**ENSIIE** et au master **SEM** du **CNAM**

TRIALOG

ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

le cnam

Gestion de données personnelles dans un réseau social

[MEMOIRE]

[STAGE DE FIN D'ETUDE]

Maître de stage : M. Bruno Jean-Bart
Tuteur ENSIIE : Mme Catherine Dubois
Tuteur CNAM : Mme Selma Boumerdassi

Remerciements

Avant même de commencer ce mémoire, je tiens à remercier mon maître de stage Bruno Jean-Bart pour la confiance qu'il m'a accordé, les opportunités de projets qu'il m'a permis d'aborder, et bien sûr pour m'avoir encadré et conseillé durant mon stage. J'aimerai aussi remercier l'équipe de direction de Trialog, notamment Madeleine Francillard, pour m'avoir fait confiance en me permettant même de participer à des meetings du projet européen SOCIETIES.

Je voudrai aussi remercier tout spécialement Rafik Said-Mansour, Franck Gauthier et Mourad Tiguercha, pour leur disponibilité, leur écoute, et l'aide précieuse qu'ils m'ont apportés dans mes recherches.

Et je ne pourrai conclure ces remerciements sans exprimer ma gratitude à tous les membres de la société Trialog pour leur chaleureux accueil.

ÉCOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE POUR L'INDUSTRIE ET L'ENTREPRISE

FICHE SIGNALÉTIQUE

Mémoire d'ingénieur ENSIIE

Gestion de données personnelles dans un réseau social

Auteur : Olivier Maridat

Directeur du stage : M. Bruno Jean-Bart, directeur, société Trialog

Descriptif : La protection de la vie privée est un défi grandissant face à l'émergence des réseaux sociaux et la croissance des communications. Minimiser au maximum les données personnelles à partager dans un environnement social est la clé pour protéger la vie privée des utilisateurs. J'ai étudié en détails ces aspects de la protection de la vie privée, et spécifié, conçu et implémenté un prototype d'une authentification anonyme sans perte de données personnelles, ainsi qu'un moyen d'obscurcir des données de géolocalisation afin de diminuer leur teneur en informations personnelles.

Résumé

La protection de la vie privée est un défi grandissant face à l'émergence des réseaux sociaux et la croissance des communications. SOCIETIES, un projet européen, se propose de relever de nombreux défis à ce sujet, dans l'environnement social et ubiquitaire qu'il a pour but de créer. Minimiser au maximum les données personnelles à partager est la clé pour protéger la vie privée des utilisateurs dans ce cadre, et pour cela j'ai spécifié, conçu et implémenté un prototype d'une authentification anonyme sans perte de données personnelles, ainsi qu'un moyen d'obscurcir des données de géolocalisation afin de diminuer leur teneur en informations personnelles.

Mots clefs : vie privée, Android, authentification, minimisation de données, obscurcissement, réseaux sociaux, informatique ubiquitaire

Summary

Privacy is a real challenge of today with rising of social networks and over-communications. SOCIETIES is an European project which aims to product pervasive social networks with a good focus on privacy. Personal data minimization in communities is a good way to protect users' privacy. That is why, I have specified, designed and implemented a prototype of anonymous authentication in order to authenticate without sharing personal information, and a mean to obfuscate geolocation data to reduce their degree of personal degree information.

Keywords: privacy, android, authentication, data minimization, data obfuscation, social networks, pervasive computing

Table des matières

1 Introduction

Dans le cadre de mes études à l'ENSIIE et de mon année de master Systèmes Embarqués et Mobiles en bi-cursus au CNAM, j'ai eu le privilège de réaliser mon stage de fin d'étude dans la société Trialog. Mon stage a débuté le 4 avril 2011, s'est terminé le 30 septembre 2011, et ce mémoire rend compte de mes six mois d'expérience au sein de cette entreprise.

Mon sujet de stage s'inscrit dans le cadre du projet européen SOCIETIES, en se focalisant plus particulièrement sur la protection de la vie privée dans les réseaux sociaux. Devant l'émergence des architectures sociales et la montée en puissance toujours croissante des flux de communications, ce domaine de recherche gagne en intérêt, et SOCIETIES est un projet plein de défis à ce sujet.

Je vais tout d'abord présenter le contexte de mon stage : l'entreprise Trialog, le projet européen SOCIETIES, et la méthodologie que j'ai suivies. Je formaliserai ensuite la mission qui m'a été confiée, puis j'expliciterai les deux études que j'ai réalisées sur ce sujet. Pour finir, je conclurai par un bilan d'expérience sur ce que m'a apporté ce stage de fin d'étude.

Des annexes et un glossaire à la fin de ce rapport fournissent des informations complémentaires ne pouvant trouver leur place dans les parties présentées ci-dessus sans en embrouiller le sens.

2 Contexte du stage

2.1 Présentation de Trialog

2.1.1 La société

Trialog est une société de conseil, d'étude et d'ingénierie dans les domaines des réseaux de télécommunication et des systèmes temps réel. Plus particulièrement, ses activités concernent l'étude et le développement de systèmes et de produits innovants pour l'électronique automobile et grand public, les télécommunications (dont la téléphonie mobile), la domotique et la distribution d'énergie.

La signification du mot « Trialog » résume d'ailleurs bien ces activités :

- « TR » pour Temps Réel (notamment pour l'informatique dans les voitures)
- « IA » pour Intelligence Artificielle (notamment avec la domotique)
- « log » pour Logiciel

Fondée en 1987, cette société est aujourd'hui basée à Paris 8ème et possède un capital réparti entre ses fondateurs, ses employés et des investisseurs privés. En moyenne, l'effectif de Trialog oscille aux alentours d'une vingtaine d'employés qui sont des ingénieurs venant de divers horizons dans le domaine de l'informatique. Trialog est donc une société à « taille humaine ». Cela permet notamment une répartition des projets selon les compétences et les affinités de chacun, et un investissement personnel des employés pour la bonne marche et la progression de l'entreprise.

2.1.2 Ses clients, ses projets

Projets industriels

Cette société compte parmi ses clients des grands groupes comme Renault, ou encore ERDF

notamment sur un projet de « compteurs communicants » (1), (2), (3).

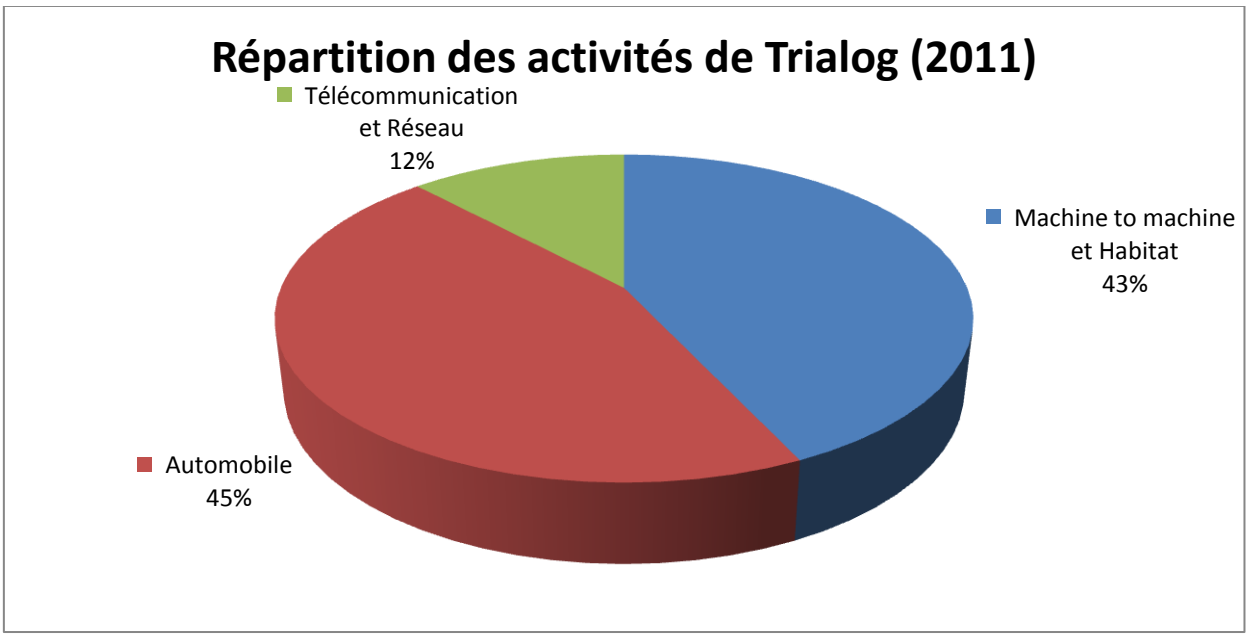
Elle effectue également, à la demande de ses clients, des audits de projets et produits qui conduisent à des actions de conseil en management stratégique.

Projet européens

De plus, Trialog participe de manière active à des programmes européens de recherche, ce qui enrichit constamment son expérience et diversifie son expertise afin de mieux servir ses clients. Le projet « MonAMI » (4) est un de ces programmes européens auquel a participé Trialog. Il est terminé depuis peu, et son but était de faciliter et de sécuriser la vie à domicile pour des personnes âgées ou handicapées.

Dans de tels projets, européens ou industriels, Trialog contribue à l'architecture, à la conception et au développement des systèmes et des produits. Le métier de Trialog consiste donc à conjuguer les techniques de pointe avec les contraintes industrielles, économiques, humaines, et celles du marché afin d'améliorer la compétitivité des systèmes et des produits, ainsi que leur cycle de création.

2.1.3 Répartition de ses activités



2.2 Présentation du projet européen SOCIETIES

2.2.1 Le cadre

SOCIETIES est un projet européen sur quatre ans qui a débuté en octobre 2010. C'est dans le cadre de ce programme que s'est déroulée la mission qui m'a été confiée durant mon stage. De nombreuses sociétés basées en Europe y participent, comme Telecom Italia, Intel, IBM, NEC et bien sûr Trialog ; ainsi que des établissements académiques comme Telecom Sud Paris ou Herriot Watt University (Edimburg). La liste complète des partenaires du projet est consultable sur le site Internet de SOCIETIES (5).

Cela fait donc moins d'un an que ce projet a été lancé. La première phase du projet a consisté à définir les concepts, les termes et les éléments du projet, les scénarios d'usage ainsi que l'étude des besoins. Ensuite l'architecture globale a été définie, ce qui va permettre de spécifier en détails chaque sous-partie. Pour autant, c'est une vaste entreprise que je ne prétends pas appréhender dans son ensemble, et dont certains aspects sont encore en mutation. Un travail est d'ailleurs actuellement effectué par des partenaires du projet de manière à pouvoir expliquer en deux ou trois diapositives les tenants et les aboutissants de SOCIETIES. Mais tâchons ici de le décrire succinctement du mieux possible.

2.2.2 Le but

SOCIETIES signifie « *Self Orchestrating Community ambIEnT intelligEnce Spaces* » ce que l'on pourrait traduire par « Espaces communautaires pervasifs, intelligents, et auto-orchestrés ». Il a pour but de créer des espaces intelligents de coopération (appelés *Cooperating Smart Space* ou CSS) permettant à un utilisateur d'interagir avec des communautés généralement sous forme de réseaux sociaux, en les intégrant à des espaces pervasifs (utilisant des réseaux de capteurs, la géolocalisation et les données de l'environnement en général). De manière plus pratique, l'idée est que les utilisateurs vont utiliser leur CSS, par exemple au travers de leur téléphone, pour participer de manière durable ou temporaire à une ou plusieurs communautés (appelées *Community Interaction Space* ou CIS) qui leur fourniront des services. Le tout de manière pervasive, c'est-à-dire en s'intégrant et en réagissant à l'environnement.

Le cahier des charges du projet décrit quelques exemples d'utilisation facilitant la compréhension :

- **Dans le cadre d'une entreprise :** il est possible de créer une communauté, c'est-à-dire un CIS, pour une entreprise de manière à fournir aux employés des services de messagerie, de réservation de salles de réunions, d'agenda partagé, d'accès à l'imprimante et au vidéoprojecteur, d'utilisation des outils domotiques existant dans les locaux de l'entreprise, etc.
- **Dans le cadre d'un match de foot :** les supporters présents dans le stade peuvent rejoindre la communauté créée pour l'occasion et commenter en direct en donnant leurs impressions sur le match, ou bien revoir la dernière action au ralenti.
- **Dans le cadre d'un cataclysme humanitaire ou écologique :** la mise en place rapide et efficace d'un réseau de communication et d'échange tirant parti de l'environnement et non forcément lié au réseau Internet peut permettre de sauver des vies ou de protéger notre planète.

On voit dans ces exemples très simples que l'on a la notion de communauté

- durable (scénario entreprise) ou temporaire (scénario match de foot),
- liée à un lieu et donc à la notion de géolocalisation (scénarii entreprise, match de foot)
- tirant parti de l'environnement pour offrir des services ludiques (scénario match de foot) ou utilitaires (scénario cataclysme)

2.2.3 Fonctionnement global de SOCIETIES

Expliquons maintenant le fonctionnement global de SOCIETIES. Le système SOCIETIES permet de mettre en place des communautés d'utilisateurs possédant un *smartphone* (comme le « User 1 » et le « User 2 » dans le schéma ci-dessous) leur permettant d'interagir entre eux en utilisant des services partagés par la communauté. Le schéma ci-dessous représente de manière simplifiée une communauté SOCIETIES.

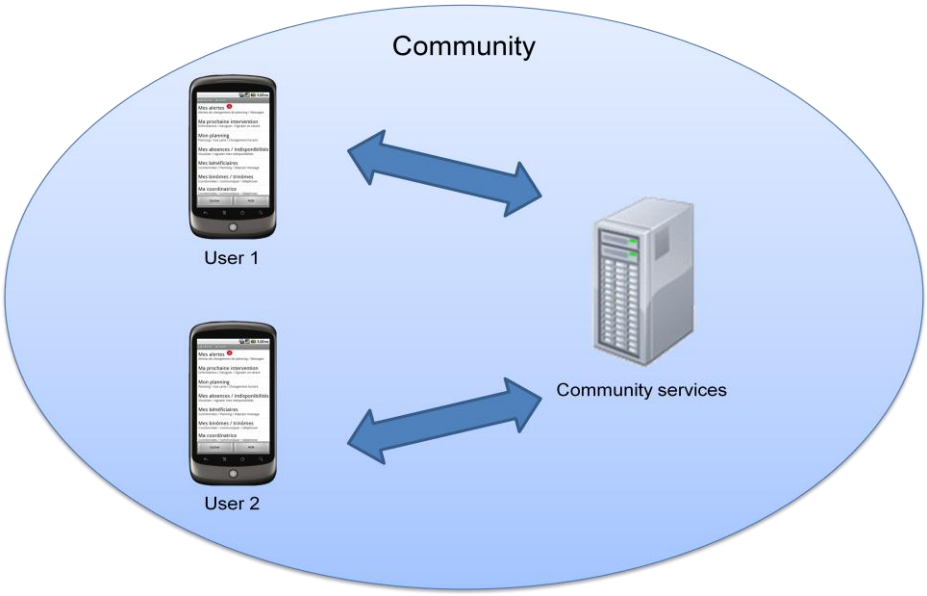


Figure 1 : Représentation simplifiée d'une communauté SOCIETIES

En réalité, l'architecture est légèrement plus complexe et nous allons la détailler.

Un *Cooperating Smart Space* (CSS) représente un utilisateur, par exemple une personne ou une organisation, et lui permet de créer ou de faire partie d'une communauté, appelée *Community Interaction Space* (CIS), regroupant plusieurs CSS. Un CSS peut faire partie de plusieurs communautés.

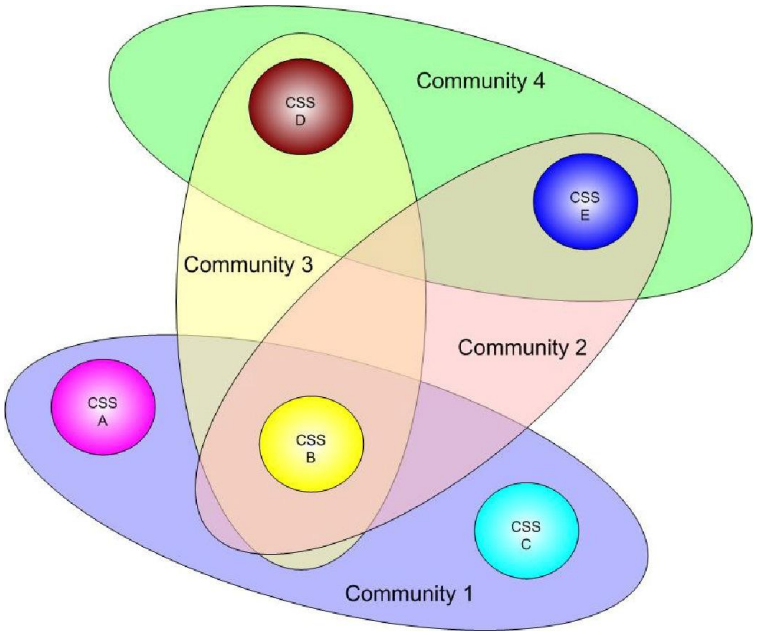


Figure 2 : Des CSS faisant partis de plusieurs communautés

Le créateur (ou un administrateur) de la communauté y associe des données et des ressources (capteurs, …) au moyen de services qui sont accessibles à tous les membres ou une partie des

membres de la communauté.

Un utilisateur utilise donc son CSS comme moyen de création d'une communauté, ou d'accès à une communauté. Un CSS est utilisable à l'aide d'un ou plusieurs périphériques appelés CSS *Node* : un *smartphone*, une tablette, un ordinateur ou un serveur... Mais bien plus, un CSS est un système distribué où les CSS Nodes mobiles utilisent des CSS Nodes plus robustes (comme un serveur) pour réaliser pleinement leurs fonctionnalités, et où les différentes Nodes n'ont pas forcément les mêmes capacités. Un CSS possède des données, des capteurs et des services propres. Ce sont ces informations qu'il peut mettre à disposition comme il le souhaite dans une communauté qu'il crée.

N'importe quel CSS peut créer une communauté que peuvent rejoindre d'autres utilisateurs s'ils respectent certains critères : intérêts communs, possesseurs d'un code particulier, à proximité d'un lieu particulier, …

2.2.4 Différences entre SOCIETIES et les réseaux sociaux actuels

A l'heure actuelle il existe de nombreux réseaux sociaux, les plus connus étant sûrement Facebook et Twitter, et le tout récent Google+ dont il est difficile de prédire l'impact qu'il aura sur ce marché. En réalité, on pourrait parler plutôt de bases de données sociales pour nommer ces réseaux. En effet, ce sont en fait de gigantesques bases de données d'utilisateurs créant des liens de différentes natures entres eux (ami, suivant/suivi, cercle, etc), et leur permettant d'interagir les uns avec les autres. A la différence de la vision de SOCIETIES, où les utilisateurs ont une identité stockée chez eux ou sur une base de données sociale externe, et grâce à laquelle ils accèdent à une multitude de réseaux sociaux. Ces réseaux sociaux peuvent alors être de tailles différentes et offrir des fonctionnalités variées tout en étant fédérés entre eux pour accroître leur intérêt. D'autre part, SOCIETIES souhaite aussi adapter ces réseaux à l'environnement et leur fournir ainsi de nouvelles fonctionnalités pervasives.

SOCIETIES n'a donc pas pour but de remplacer Facebook, ou Google+, mais de s'y adosser en y offrant de nouvelles fonctionnalités. Pour autant, il est peu probable que Facebook soit disposé à s'ouvrir suffisamment pour devenir fédéré avec d'autres réseaux concurrents, puisqu'il est aujourd'hui le réseau social majoritaire en situation de quasi monopole. Cela pourrait changer si un business model émergeait dans cet environnement fédéré. Par contre, il est fort probable que les autres réseaux sociaux, émergents, soient favorables à la vision de SOCIETIES, soit parce qu'ils souhaitent rendre la main à l'utilisateur, soit parce qu'ils ont conscience que c'est leur seule solution de survie.

2.3 Organisation et méthodologie

Le cadre de mon stage a été guidé par celui du projet SOCIETIES et la vision de Trialog. Une fois mes objectifs fixés j'ai travaillé en autonomie mais toujours en contact avec mon maître de stage et mes collègues assignés au même projet. Des réunions fréquentes ont permis de prendre des décisions sur les orientations à prendre pour mes recherches, et les cinq meetings avec des partenaires européens de SOCIETIES auxquels j'ai participé m'ont permis de préciser ma vision d'ensemble du projet.

Au niveau de la méthodologie, durant mon stage nous avons adopté un cycle de développement en V avec quelques touches d'agilité pour ma deuxième étude. Voici comment j'ai réparti mon travail :

- **Etat de l'art** : recherche de ce qui a déjà été fait dans le domaine au travers d'articles scientifiques ou d'articles sur Internet et écriture de synthèses résumant ou comparant les différentes possibilités.
    - ex : vie privée, algorithmes d'authentification, méthodes de développement pour une application Android, …

- **Création de scénarii et de cas d'utilisation :** de manière à définir ce que fera le système au final et la manière dont il sera utilisé, il est utile d'écrire des scénarii de manière plus ou moins formels (grâce à des cas d'utilisation UML par exemple).
- **Exigences** : les objectifs du projet étant fixés et grâce à la vision globale acquise lors de la période de recherche : écriture d'un document détaillant les exigences requises pour le système à créer. Les étapes d'état de l'art de d'écriture des exigences m'ont permis de déterminer ce qu'il était important d'étudier, et ce qui pouvait être effectué dans un deuxième temps. J'ai ainsi pu affiner ma problématique de manière à réduire mon sujet de stage à sa substantifique moelle.
- **Spécifications**
    - **fonctionnelles** : écriture d'un document décrivant le système à créer, les acteurs, et les fonctionnalités de ce système. A partir de cette étape, l'utilisation d'UML entre en jeu pour faciliter la formalisation, l'écriture, la compréhension et le partage avec des partenaires.
    - **logicielles/techniques** : écriture d'un document spécifiant le système d'un point de vue technique : données, communications, fonctions et algorithmes.
- **Conception** : écriture d'un document précisant l'architecture choisie pour l'implémentation du système spécifié, donnant le détail quant à l'implémentation des algorithmes si besoin.
    - ex : l'algorithme « A » nécessitant la résolution d'un système d'équation sera implémenté grâce à la méthode de Newton.
- **Implémentation** : développement du système. C'est dans cette étape que j'ai réutilisé quelques principes de la méthode agile, en choisissant de développer un algorithme par itération successive : d'abord un algorithme simple, puis à nouveau une phase de spécification/conception d'un algorithme plus complexe, et développement de cette solution, et ainsi de suite.
- **Tests et validation** : lorsque cela a été possible et utile, mise en place d'une étape de tests du système, et dans tous les cas pour finir, validation du respect des exigences et des spécifications.

Au début de mon stage, j'ai commencé par écrire ces documents en français, mais le projet étant européen, tous les documents de SOCIETIES sont en anglais. Il s'est donc avéré rapidement plus efficace d'écrire directement les spécifications (et autres) en anglais, de manière à pouvoir les partager rapidement avec d'autres partenaires si besoin.

Dans ce mémoire, de nombreux termes du domaine de la vie privée sont en général plus connus sous leur forme anglaise que française. J'utiliserai cependant une traduction française en précisant la forme anglaise entre parenthèse lors de la première utilisation. Exemple : vie privée (*privacy*).

3 Objectif de ma recherche : protéger des données personnelles dans un réseau social

3.1 Quelques notions de vie privée

3.1.1 L'Union Européenne

L'Union Européenne a pris plusieurs initiatives pour promouvoir et légiférer l'utilisation des données personnelles parmi ses pays membres. En plus de leur importance humaine, ces initiatives sont primordiales pour harmoniser les lois des différents états membres et faciliter ainsi le libre-échange d'informations, notamment à des fins commerciales. En 1995 l'Union Européenne a ainsi adopté la « Data Protection Directive » aussi nommé « Directive 95/45/EC » (6) qui fournit entre autres un cadre au traitement automatisé de données personnelles, et encourage l'institution d'autorités de protection des données personnelles comme la CNIL en France.

Lors d'un discours à Bruxelles le 16 mars 2011, Viviane Reding, vice-présidente de la commission européenne à la Justice, aux Droits fondamentaux et à la Citoyenneté, résume la vision présente et future de l'Union Européenne sur la vie privée en quatre points (7). Son point de vue est d'autant plus intéressant qu'il prend en compte l'aspect juridique du sujet.

- **La transparence**

  L'utilisateur doit avoir accès facilement à des explications claires et simples sur les données collectées à son sujet, les traitements qui sont ou seront effectués dessus, et la finalité de cette utilisation. Ces données étant, par définition, personnelles, elles appartiennent à l'utilisateur, en tout cas dans leur état d'origine. Il est donc logique que celui-ci puisse s'informer de leur utilisation.

  D'autre part, cette notion introduit la mise en place de vérifications et d'audits du respect des engagements pris par les organismes (sociétés, gouvernements, …) traitant des données personnelles.

- **La protection de la vie privée par défaut**

  Par défaut, la vie privée des utilisateurs doit être protégée au maximum, c'est-à-dire : aucune collecte, aucune utilisation et aucun partage de données personnelles. En effet, l'utilisateur doit consentir de manière plus ou moins explicite à l'utilisation de ses données. Ce qui implique qu'il doit savoir pour quoi elles seront utiles, ce qui rejoint la notion de transparence. Pour des données peu sensibles et utilisées uniquement dans un but facilement identifiable, il est acceptable d'avoir un accord global de l'utilisateur (case à cocher, ou simple phrase « conformément à la loi informatique et liberté… » en France), généralement lié à une étape d'inscription ou de soumission de formulaire. On parle dans ce cas d'*opt-out*, accord tacite. Mais pour d'autres données comme la localisation d'un individu, surtout si elle est liée à son identité (un prénom par exemple), il est nécessaire d'avoir l'accord express de l'utilisateur. Dans ce cas, on parle d'*opt-in*.

- **Le droit à l'oubli**

  Viviane Reding introduit la notion de droit à l'oubli, c'est-à-dire le droit et la possibilité (ce qui est une nuance de taille !) de retirer un consentement à un traitement des données. Cet aspect du respect de la vie privée apparaît déjà dans la loi informatique et liberté française (8) avec le droit d'opposition et le droit de rectification pour des raisons justifiables. En mettant en avant le droit à l'oubli, on assure à l'utilisateur qu'il pourra revenir en arrière en cas de changement de situation, ou si sa confiance dans le service utilisant ses données a baissé. Cependant, d'un point de vue technique et organisationnel, cela peut poser des difficultés puisque cela implique d'avoir la possibilité de modifier ou supprimer aussi les données chez les tierces-

parties qui ont pu y avoir accès. Par exemple : une agence immobilière devrait pouvoir modifier / supprimer les données personnelles de ses clients, y compris celles transmises aux assurances, …

- **La protection indépendamment de l'emplacement de stockage des données**

  La Directive 95/45/EC fixe des règles, non seulement pour les états membres de l'union européenne, mais aussi pour les états extérieurs lorsque ceux-ci traitent des données européennes : article 4 « *Controllers from outside the EU, processing data in the EU, will have to follow data protection regulation* ». Ce qui implique que des sociétés comme Google, Facebook, … doivent (ou devraient) se plier aux règles de protection des données personnelles européennes.

3.1.2 Architecture dirigée par la protection de la vie privée

D'un point de vue « architecture des systèmes », il est nécessaire de mettre en place des processus permettant l'application des lois et des bonnes pratiques concernant la vie privée. C'est d'autant plus important que la gestion de la vie privée a un coût. Par exemple, l'étude « The Cost of Reading Privacy Policies » d'Aleecia M. McDonald et Lorrie Faith Cranor (9), montre que la perte de temps passé à lire des politiques de gestion des données personnelles coûte environ 3 534$ par an et par internaute américain : « We estimate that reading privacy policies carries costs in time of approximately 201 hours a year, worth about $3,534 annually per American Internet user » (9).

C'est dans ce cadre qu'un consultant de Trialog Antonio Kung, ainsi que Johann-Christoph Freytag et Frank Kargl, ont proposé le principe « Privacy-By-Design » (10). L'idée générale est que la gestion de la vie privée guide toutes les étapes de développement. Ce principe s'appuie sur trois points clefs :

- **Minimisation de l'utilisation des données personnelles** (*data minimisation*)

  La notion de minimisation des données est capitale et doit diriger les réflexions en amont du projet, lors de l'établissement des exigences et des grandes lignes de l'architecture. Partant du principe qu'une donnée personnelle non partagée présente un risque zéro en termes de protection de la vie privée, l'idée ici est de réduire au maximum le nombre d'informations personnelles échangées. D'où l'importance d'avoir cette réflexion très en amont du projet de manière à choisir l'architecture utilisant le moins de données privées, et à prévoir l'utilisation de certificats de sécurité et d'algorithmes d'anonymisation si besoin.

  On rejoint ici les points clefs de Viviane Reding (7) en ce qui concerne la protection de la vie privée par défaut.

  Exemple : soit un système de paiement automatisé aux péages autoroutiers. Ce système enregistre régulièrement la géolocalisation du véhicule de manière à pouvoir calculer ensuite le tarif dû à la société de péage. Au lieu d'envoyer toutes les données de localisation à la société de péage pour qu'elle effectue le calcul, il est préférable de faire faire ce calcul par le véhicule et de l'envoyer à la société de péage en y joignant un certificat de sécurité prouvant la véracité du calcul. Ceci implique de fortes exigences en termes de sécurité.

- **Mise en application** (*enforcement*)

  Durant les étapes de conception et d'implémentation, il faut mettre en pratique et vérifier la mise en pratique de ce qui a été prévu en termes de protection des données personnelles, notamment en ce qui concerne le stockage et la manipulation de ces informations.

- **Transparence** (*transparency*)

  La vie privée étant très liée à la confiance des utilisateurs, cette notion de transparence est indispensable pour rendre utile (ou en tout cas « utilisé ») les efforts fournis pour la protection des données personnelles. Cette notion se partage en deux volets :

- Transparence au sujet des données personnelles collectées et de leurs utilisations. Dans la même idée que la transparence évoquée par Viviane Reding (7).
- Transparence durant les étapes de test et de validation pour s'assurer de la mise en application des principes de protection de la vie privée. Ces étapes doivent être ouvertes et les résultats doivent être accessibles.

Le schéma suivant récapitule ces trois points clefs :

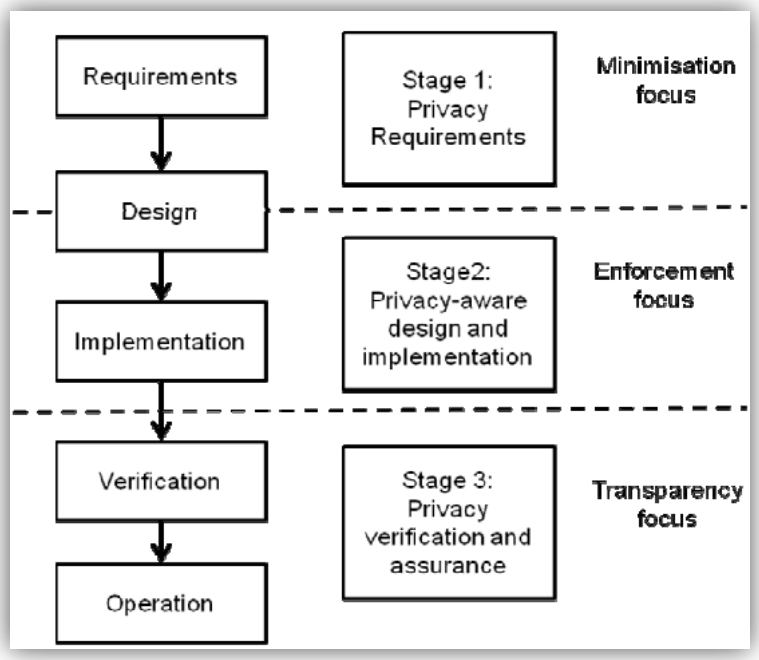


Figure 3 : Impact de Privacy-By-Design sur les processus

3.2 Problématique : comment minimiser l'utilisation de données personnelles dans des réseaux sociaux ?

Les quelques notions de vie privée ci-dessus montrent bien que la gestion des données personnelles est un vaste sujet, à la fois juridique, architecturale et technique. L'objet de ma recherche se focalise sur la manière de minimiser l'utilisation de données personnelles (*data minimisation*). Pour ce faire, j'ai étudié deux manières de réduire le nombre de données privées utilisées au travers de deux exemples concrets :

- En modifiant l'architecture du système pour éviter de partager des données personnelles
  - En mettant en place la notion d'authentification anonyme : un mode d'authentification où l'identité de l'utilisateur n'est pas transmise.
- En obscurcissant les données utilisées de manière à réduire leur teneur en informations personnelles
  - En travaillant sur l'obscurcissement de données de géolocalisation afin de fournir des informations sur la géolocalisation d'un individu mais en réduisant la précision de cette localisation.

Dans SOCIETIES, un module « Privacy Manager » se chargera de gérer la protection des données personnelles à l'aide de différents sous-modules. Mes recherches doivent donc s'inscrire dans ce cadre dans le but de valider l'architecture que l'on a choisie pour l'aspect vie privée de SOCIETIES, et d'explorer les possibilités qu'elle propose ou celles que l'on n'a pas encore envisagées. De plus, mon travail sera intégré plus ou moins directement à ce module.

Une autre contrainte est la gestion de l'embarqué. SOCIETIES est un projet où l'utilisateur accédera à des communautés à l'aide de ses CSS[1] qui peuvent être un smartphone, un ordinateur de bureau, un ordinateur portable, une tablette, un serveur, … Il faut donc prendre en compte plusieurs aspects :

- La consommation en CPU et la complexité des algorithmes de gestion de la vie privée afin de déterminer s'il est possible de les faire fonctionner sur un mobile (par exemple un smartphone de type Android),
- Dans cette architecture distribuée, il faut déterminer sur quel support allouer les différents modules de protection des données personnelles (ex : module A sur le CSS Node smartphone, module B sur le CSS Node serveur)

Mon étude prend donc en compte ces contraintes pour spécifier des moyens de minimiser les données personnelles utilisées, et implémenter des prototypes.

[1] CSS : un terme du projet Societies signifiant *Cooperating Smart Space*. C'est à la fois le représentant de l'utilisateur, ainsi que le moyen possédé par celui-ci pour interagir avec des communautés.

4 Authentification anonyme

En étudiant un scénario SOCIETIES, par exemple celui d'une communauté d'entreprise, et en mettant en valeur tout ce qui a trait à la vie privée, on s'aperçoit que des données personnelles sont échangées lors de l'étape d'authentification. En effet, lorsqu'un utilisateur souhaite s'authentifier auprès d'une communauté, c'est-à-dire qu'il cherche à prouver son identité afin de prouver son appartenance à la communauté, il fournit généralement un login et un mot de passe. Or ces informations sont personnelles. Avec la connaissance de ces informations, un CIS[2] est capable d'associer des actions à une identité (ici un login), or une communauté n'a pas forcément à savoir les faits et gestes de ses membres. C'est parfois nécessaire, mais pas toujours. Par exemple pour utiliser un service fournissant le planning d'une salle de réunion, l'identité de l'utilisateur n'est pas requise, pas plus que pour recevoir sur son téléphone le score d'un match de foot. Cela est d'autant plus vrai qu'un organisme peut gérer différentes communautés, et si un utilisateur participe à plusieurs d'entres elles, il ne se doute pas forcément qu'un organisme est capable de faire des liens entre ses données dans le CIS 1 et d'autres données dans le CIS 2. D'où la nécessité d'un nouveau mode d'authentification nommé « **authentification anonyme** » (*anonymous authentication*) par opposition à un mode plus classique que l'on peut appeler : « **authentification identifiée** » (*identified authentification*).

Posons le contexte :

- Un utilisateur est membre d'une communauté
- Il souhaite s'authentifier
    - Prouver qu'il est le membre qu'il prétend être
    - Pour utiliser des services de la communauté

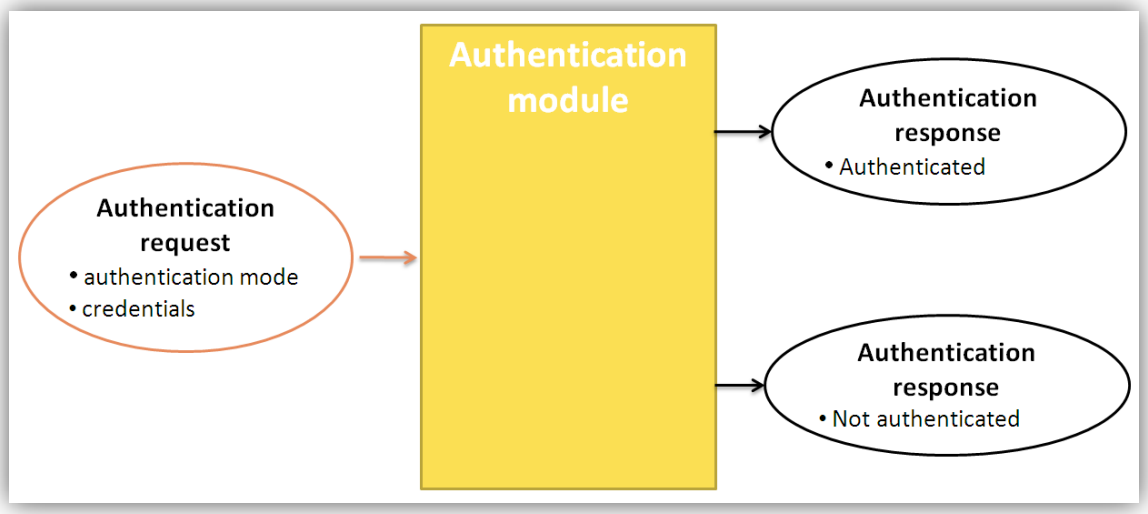


Figure 4 : Schéma bloc d'un module d'authentification (en anglais)

Toute la première moitié de mon stage a porté sur la recherche d'une solution pour

[2] CIS : *Community Interactive Space*, aussi appelé « Communauté ». Ce terme du projet Societies désigne un élément créé par un CSS qui lui a associé des données et des services. Un CIS est associé à plusieurs CSS qui sont ses membres.

l'authentification anonyme. L'objectif de cette section est donc de définir davantage ce concept, de présenter des solutions envisageables et de voir plus en détails celle que choisie et la manière dont je l'ai implémentée dans un prototype.

A noter que pour ce prototype, c'est un développement sous Android qui a été choisi pour des questions de simplicité (le développement pour iPhone nécessitant un ordinateur Apple et une licence payante). De plus, c'est la solution choisie pour le développement des premières versions de SOCIETIES. Nous verrons plus loin ce qu'implique un développement sur cette plateforme.

4.1 Objectifs et exigences

Traduisons l'authentification anonyme en termes d'exigences et d'objectifs.

4.1.1 Authentification anonyme

On veut :

- S'assurer que l'utilisateur est membre de la communauté visée
- Ne rien savoir d'autre au sujet de cet utilisateur (ni son identité, ni son et ses codes d'accès, …)
- Rendre impossible le traçage des authentifications successives d'un membre
    - Sinon, on pourrait récolter des informations sur un individu anonyme et être en mesure de déduire des informations utiles sur son identité.

Dans le but :

- D'utiliser des services de la communauté sans révéler l'identité de l'utilisateur
    - En effet, le CIS peut fournir des services où l'identité n'est pas requise
    - Et les utilisateurs peuvent ne pas vouloir révéler leur identité aux autres membres, ou la communauté elle-même
- De protéger la sécurité et la vie privée de l'utilisateur durant l'étape d'authentification
    - Empêcher quiconque d'usurper une identité
    - Empêcher quiconque de tracer les authentifications successives d'un membre
    - Assurer l'utilisateur qu'il est anonyme pour les autres membres et la communauté

C'est donc un mode d'authentification innovant où la communauté ne sait rien sur le membre, mais peut tout de même lui donner accès à tous les services ne nécessitant pas une identification. Cette fonctionnalité est utile puisqu'elle permet à un utilisateur de protéger sa vie privée en minimisant le nombre de données connues par des tiers en s'adaptant à la situation souhaitée.

4.1.2 Authentification identifiée

On veut simplement :

1. S'assurer que l'utilisateur est le membre qu'il prétend être dans la communauté

Dans le but :

- D'utiliser des services de la communauté avec sa propre identité
    - Ex : être « Albert » ou « Alb' » durant une discussion dans un service de messagerie
    - Ex : consulter l'historique de ses conversations sur le service de messagerie

- De protéger la sécurité de l'utilisateur durant l'étape d'authentification
    - Empêcher quiconque d'usurper une identité

4.2 Etat de l'art

4.2.1 Etudes des architectures possibles pour l'authentification

Il existe de nombreuses méthodes pour s'authentifier. Sur Internet par exemple, l'utilisation d'un couple login+mot de passe est très courante, mais on voit apparaître de plus en plus l'authentification par OpenID à l'aide d'un URL. En étudiant ces différentes méthodes, il apparaît que l'authentification anonyme dépend certes du protocole, mais surtout de l'architecture utilisée.

Quatre architectures sont envisageables :

- Code d'accès partagé par tous les membres de la communauté
- Stockage des codes d'accès séparément des identités
- Passeport certifiant l'appartenance à une communauté
- Preuve d'appartenance à une communauté à divulgation nulle d'identité

Nous allons maintenant les détailler et les comparer.

4.2.1.1 Code d'accès partagé par tous les membres de la communauté

Dans cette architecture, une clé est connue par tous les membres de la communauté et ils l'utilisent chaque fois qu'ils veulent s'authentifier anonymement.

Cette architecture possède des failles puisque si le secret de la clé est révélé, c'est toute la sécurité du système qui s'écroule. Or tout le monde possède la même clé, donc la sécurité correspond à celle de l'utilisateur le moins prévoyant, ce qui avoisine parfois le zéro. Qui plus est, il faut fournir à nouveau la clé à tous les utilisateurs pour la changer.

Une telle architecture n'est pas envisageable dans SOCIETIES, puisque l'utilisateur final installera sur son support un « logiciel SOCIETIES » et la clé ne pourrait être suffisamment protégée. Cependant, cette architecture est utilisée avec succès dans les voitures communicantes (11) par exemple. En effet, dans ces dernières, les constructeurs peuvent protéger physiquement la clé et/ou utiliser des moyens cryptographiques pour éviter sa perte ou son vol.

4.2.1.2 Stockage des codes d'accès séparément des identités

En stockant à deux endroits différents les identités et les codes d'accès (par exemple dans deux bases de données), on met en place une architecture permettant l'anonymat. En couplant cette technique à certains protocoles d'authentification qui permettent d'empêcher qu'un attaquant (voir le système lui-même) traque un utilisateur, il est possible de réaliser une authentification anonyme. Il existe notamment trois protocoles qui permettent cela :

- **Secure Remote Protocol** (SRP) : un protocole développé par Thomas Wu de Stanford (12) (13) présenté comme à divulgation nul de connaissance (« Zero-Knowledge Proof » (14)) même s'il n'y ressemble pas dans la forme. Il est notamment performant pour une application sur le Web et a l'avantage de fournir une clé de session permettant de chiffrer ensuite les messages échangés.
- **Feige-Fiat-Shamir** (FFS) : un protocole proposé par Feige, Fiat et Shamir (15) (16) aussi nommé « Zero-Knowledge Proof of Identity ». Il permet notamment de prouver son identité sans fournir d'informations sur ses codes d'accès.
- Sur le Web, on « hashe » souvent les mots de passe stockés pour augmenter la sécurité. Un

protocole basé sur cette technique peut permettre l'authentification anonyme. Il a l'avantage d'être simple d'implémentation mais est moins performant que les algorithmes précédents.

Néanmoins, cette architecture possède de sérieux inconvénients. Tout d'abord, elle nécessite d'avoir confiance dans l'entité auprès de laquelle on s'authentifie : confiance qu'elle séparera bien le stockage identité / codes d'accès, et qu'elle ne créera pas de liens entre les deux. Or d'après le principe de Kerckhoffs (17), la sécurité ne doit dépendre que de la clé, ce qui n'est pas le cas ici.

Plus problématique, cette architecture ne passe pas à l'échelle. Pour un trop grand nombre d'utilisateurs, il serait difficile, voir impossible de la mettre en œuvre. Etudions plus en détails le protocole SRP pour mieux comprendre cet inconvénient.

Le schéma ci-dessous représente le fonctionnement du protocole SRP adapté pour correspondre à l'architecture proposée et permettre l'authentification anonyme. Un serveur et un client s'envoient des messages et effectuent des actions (schématisées dans des rectangles noirs) à chaque réception. Il n'est pas important de comprendre les messages qui sont envoyés, mais par contre, on peut remarquer qu'il y a deux échanges question-réponse successifs et que deux d'entre eux envoient autant de fois les données requises qu'il y a d'utilisateurs dans la communauté.

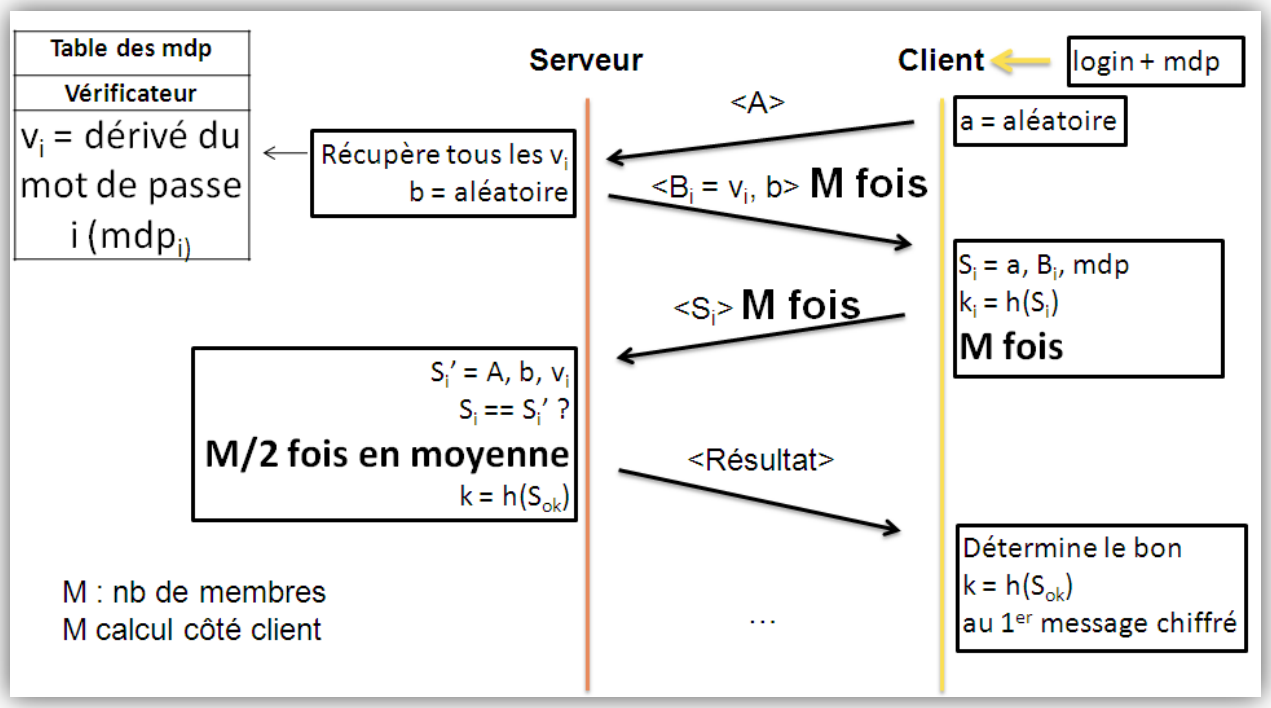


Figure 5 : SRP adapté pour l'authentification anonyme

Maintenant, reprenons le même schéma mais intéressons-nous uniquement à la durée nécessaire pour exécuter les calculs et les communications réseaux. Pour cela, prenons les hypothèses suivantes :

- 10 000 utilisateurs dans cette communauté, soit M = 10^4
- Notre connexion transfère en moyenne 200ko/s
- Le serveur est un ordinateur de bureau récent, mais le client est un smartphone sous Android.
- Les calculs de S et de k nécessitent une ou plusieurs opérations de hashage. On utilise l'un des meilleurs algorithmes de hashage du moment mais le plus long à calculer : Sha-512 qui

produit 512 bits.

- Sur un Android moyen, cette opération dure en moyenne 10^{-2}s, mais possède en plus un coût en consommation CPU (et donc en batterie) et en mémoire qui peuvent poser d'autres problèmes.
- Sur une machine récente, cette opération est effectuée très rapidement et on va voir que sa durée d'exécution est négligeable.

- Les dérivés de mot de passe v_i correspondent à un hashage, donc 512 bits soit 64 octets, auquel on ajoute 8 octets aléatoires pour ralentir drastiquement les attaques par dictionnaire (18). Un dérivé de mot de passe fait donc 72 octets.

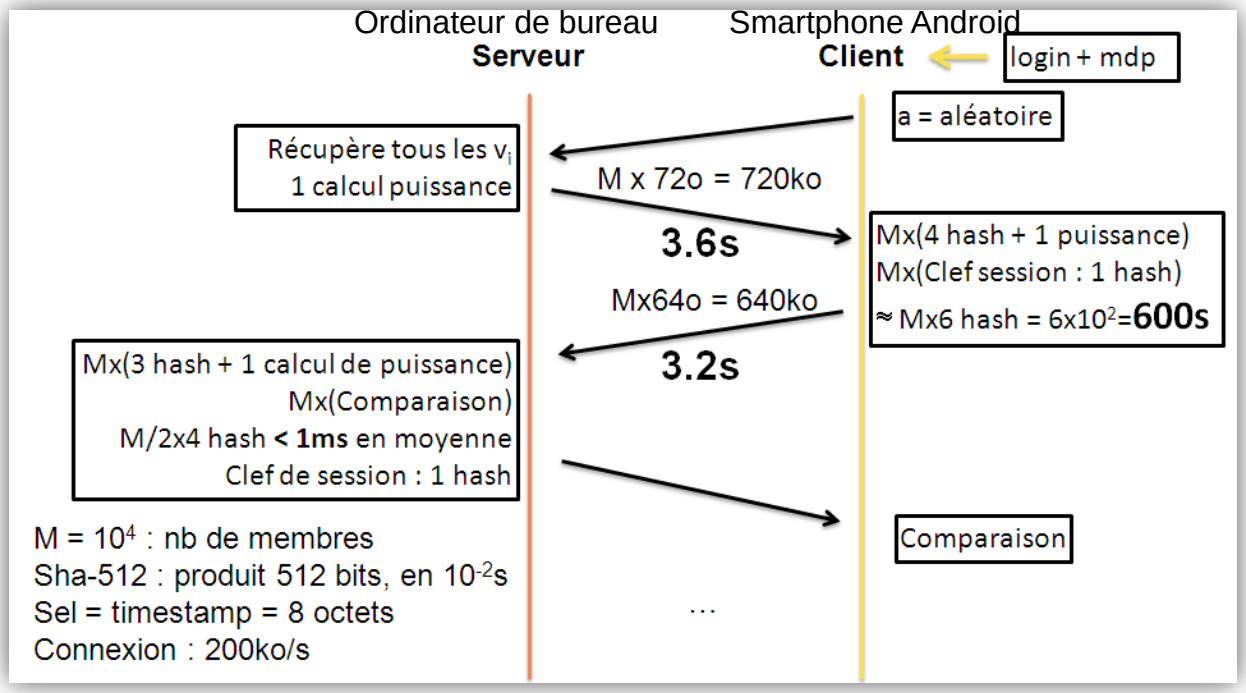


Figure 6 : simulation du temps d'exécution du protocole SRP adapté à l'authentification anonyme

Avec ces quelques hypothèses proches de la réalité, on s'aperçoit rapidement que le facteur temps est problématique. Les temps d'échanges sont longs, environ 8 secondes, mais encore envisageables. Côté serveur, les calculs sont réalisables. Mais côté client, les temps de calculs, environ 600 secondes, soit 10 minutes, sont rédhibitoires. D'autant plus qu'effectuer des calculs pendant 10 minutes sans discontinuer sur un Android n'est pas conseillé du tout ! (consommation excessive de la batterie, risque d'être stoppé par l'OS, …)

Les autres protocoles font face à des problèmes similaires. Cette architecture ne peut donc être utilisée en l'état dans le cadre de SOCIETIES pour l'authentification anonyme.

4.2.1.3 Passeport certifiant l'appartenance à une communauté

Dans cette architecture, l'utilisateur s'identifie à une autorité de confiance qui lui fournit alors un passeport lui permettant de prouver son appartenance à une communauté et/ou son identité. Je me suis inspiré du système Kerberos (14) pour concevoir cette architecture. Un fonctionnement similaire est utilisée par le protocole OpenID de plus en plus utilisé sur le Web, ou encore WebID (19) sur lequel travaille actuellement un Working Group du W3C (l'organisme de standardisation du Web).

Cette architecture a l'inconvénient de nécessiter une autorité de confiance, mais en contrepartie, on

obtient réellement une authentification anonyme. La communauté, ou un attaquant écoutant le réseau, n'a aucun moyen de savoir l'identité de l'utilisateur (puisqu'elle n'est pas transmise), ni de pister ses authentifications (puisque le passeport change à chaque fois). Dans SOCIETIES, on aura de toute façon besoin d'une ou plusieurs autorités de confiance pour héberger les comptes des utilisateurs, ou effectuer des audits sur le respect de la vie privée par exemple. L'inconvénient de notre architecture n'en est donc pas un dans SOCIETIES.

4.2.1.4 *Preuve d'appartenance à une communauté à divulgation nulle d'identité*

L'idée est ici d'appliquer le principe de *Zero-Knowledge Proof* ou ZKP (14) à l'identité. La connaissance à prouver est l'appartenance à une communauté, et la preuve de cette connaissance ne doit fournir aucune autre information sur le membre. Les protocoles connus s'intéressent seulement à l'identification. D'après mes recherches, un protocole permettant cette architecture n'existe donc pas (encore) aujourd'hui. Et il n'existe pas de solution triviale. Cela pourrait être un axe de recherche futur intéressant, d'autant plus que les algorithmes ZKP commencent à être utilisés aussi sur le Web. On pourrait appeler un tel protocole : "*Zero-Knowledge about identity to prove membership*".

4.2.1.5 *Comparaison des différentes architectures*

Cet aperçu de différentes architectures d'authentification nous permet de mieux comprendre les différents modes possibles, et ce qui est réalisable aujourd'hui. Le tableau suivant met en comparaison différents protocoles des architectures ci-dessus, à partir d'une liste de critères pondérés de manière arbitraire en fonction des besoins dans SOCIETIES.

Libellé	Poids	Clef partagée	SRP	FFS	Hashage	WebID	OpenID
Mode anonyme disponible	5	1	0	1	1	1	0,5
Mode identifié disponible	3	0	1	1	1	1	1
Protéger contre le pistage externe	3	1	1	1	0,5	1	0,5
Protéger contre le pistage interne	2	1	0	0	0	1	0
Permet la double authentification	0,5	1	1	0	0	0	0
Algorithme sécurisé	2	0	1	1	1	1	1
Code d'accès sécurisé	1	0	1	1	1	1	1
Partage clef de session	0,5	0	1	0	0	1	0
Autorité de confiance non nécessaire	1	1	1	1	1	0	0
Calcul faisable par un ordinateur	3	1	0,8	0,5	0,3	1	1
Calcul faisable par un *smartphone*	3	1	0,1	1	1	1	1
Login + code d'accès	1	0	1	0	1	0	1
Aucun stockage chez le client	1	0	1	0	1	0	1
Implémentation existante	2	0	1	1	1	0,5	1
Facilité d'implémentation (de 1 à 3)	0,5	3	2	2	3	1	1
Total		**10,0**	**12,9**	**10,5**	**13,0**	**10,5**	**10,0**
Total pondéré		**19,0**	**18,7**	**22,5**	**23,5**	**24,0**	**20,5**

Sans qu'il soit nécessaire de le décrypter à la case près, ce tableau met particulièrement en avant deux protocoles :

- Architecture : stockage des codes d'accès séparément des identités, avec un protocole

adapté de techniques couramment utilisées sur le Web de hashage de mot de passe. On a certes le même problème que pour SRP mais dans une bien moindre mesure.

- Architecture : passeport certifiant l'appartenance à une communauté, avec le protocole WebID.

On a vu que l'architecture utilisant des passeports pour certifier l'identité et l'appartenance à une communauté offre davantage d'intérêts que l'architecture séparant les identités des codes d'accès. De plus, le protocole WebID (qui s'intègre dans cette première architecture) est plus intéressant que celui utilisant des techniques de hashage. En effet, d'autres partenaires européens souhaitent l'utiliser pour l'authentification classique dans SOCIETIES notamment car il permet d'utiliser la même identité pour s'authentifier sur plusieurs communautés (*Single-Sign-On*). De plus, ce protocole est en lien avec les notions de « Web of trust » et de « Web Sémantique » ce qui est un avantage dans le domaine des réseaux sociaux. J'ai donc spécifié l'authentification anonyme en me basant sur une architecture nécessitant des passeports certifiant l'identité et l'appartenance à une communauté, et en utilisant le protocole WebID particulièrement adapté à cette architecture et aux besoins de l'authentification anonyme. Voilà pourquoi la section suivante de cet état de l'art s'intéresse davantage à WebID.

4.2.2 Protocole d'authentification retenu : WebID

WebID est une technologie actuellement en cours de standardisation par un Working Group du W3C (19) (20). L'idée globale est qu'un utilisateur possède un certificat de sécurité prouvant son identité. En effet, ce certificat est associé à un fichier d'identité WebID stocké quelque part sur Internet, modifiable uniquement par cet utilisateur, et possédant dans un champ particulier une clé de sécurité du certificat. Ainsi, en envoyant son certificat à un système sachant lire les fichiers WebID, un utilisateur peut prouver son identité. Ce certificat de sécurité est en fait un certificat X.509 permettant le transport et la vérification d'une clé publique (qui peut être connue de tous) et d'une clé privée (qui est secrète et connue seulement du possesseur du certificat) afin de réaliser des opérations de chiffrement de données. Un exemple de certificat X.509 est présent en annexe 9.2.2.1.

Au final :

- Chaque utilisateur possède une ou plusieurs identités (une qui le représente vraiment et une qui représente l'un de ses pseudonymes par exemple)
- Chaque identité possède
    - Un certificat X.509
        - Contenant l'URL du fichier WebID
    - Un fichier WebID
        - Contenant la clé publique du certificat X.509
        - Modifiable seulement par l'utilisateur
- Pour vérifier l'identité d'un utilisateur
    - Les protocoles de sécurité SSL ou TLS utilisés par exemple avec HTTP permettent de vérifier le certificat X.509 de l'utilisateur (et de chiffrer les échanges)
    - Il faut vérifier la correspondance entre la clé publique du certificat et celle contenu dans le fichier WebID
- Un fichier WebID contient des informations sur l'identité qu'il représente, il est donc possible d'étendre son mode de fonctionnement initial en vérifiant d'autres données. Par exemple pour vérifier l'appartenance d'un utilisateur à une communauté :

- Il suffit de vérifier la présence de l'URL du WebID de l'utilisateur dans le WebID de la communauté
- Cependant : un utilisateur peut écrire n'importe quoi dans son fichier WebID ! En vérifiant son identité, on ne fait que s'assurer qu'il en est le possesseur et non la véracité des informations que son fichier contient. Voilà pourquoi on utilise les informations contenues dans un fichier WebID d'un tiers (par exemple une communauté) pour vérifier les informations contenues dans le WebID de l'utilisateur (par exemple son appartenance à cette communauté).

Un exemple de fichier WebID est présenté en annexe 9.2.2.2.

4.2.3 Développement sous Android

Android est un système d'exploitation open source pour terminaux mobiles conçu par Google. Il est possible de développer des applications pour Android, qui pourront ensuite être téléchargées et installées par les détenteurs de smartphones, tablettes ou téléviseurs possédant cet OS. Aujourd'hui, il existe différentes méthodes pour développer de telles applications, principalement :

- **Le SDK natif d'Android fourni par Google**. C'est la méthode prônée par Google et la plus répandue. Android étant basé sur Java, le développement est rapide à prendre en main et de nombreux outils sont disponibles.
- **HTML5** : le nouveau standard pour le Web en cours de finition par le W3C. Une application mobile conçue en HTML5 s'appelle « webapp ». Ce langage a été conçu pour s'adapter aux nouveaux périphériques mobiles et aux nouveaux usages des Internautes. D'ailleurs de plus en plus d'IHM sont produites en HTML.
- **Titanium** conçu par Appcelerator. Basé sur Javascript, ce framework permet de développer facilement des applications pour Android, iPhone, …

Face à la montée en puissance des smartphones, de la multiplication des terminaux, il convient de se poser la question du développement sur mobile de manière sérieuse. SOCIETIES et son architecture répartie n'y coupent pas. Une étude comparative de ces différentes méthodes de développement, que l'on peut retrouver en annexe 9.1 de ce mémoire, m'a permis de déterminer laquelle choisir pour mettre en œuvre un prototype de l'authentification anonyme. Cette recherche contribue de plus aux réflexions de SOCIETIES à ce sujet.

Au final, il s'avère que le SDK Android est la solution la plus adaptée aujourd'hui pour développer efficacement des applications sous Android. Voilà pourquoi c'est celle que j'ai choisie pour mon prototype. Cependant, HTML5 tend à devenir un concurrent sérieux d'autant plus qu'il est multiplateforme. Il est peu implémenté aujourd'hui et ne fonctionne pas de la même manière sur tous les supports, mais des outils comme le framework PhoneGap permettent de pallier au moins en partie cette faiblesse en attendant les évolutions futures. C'est donc une solution envisageable pour un projet sur quatre ans comme SOCIETIES.

4.3 Acteurs de l'authentification anonyme

Dans l'authentification anonyme, plusieurs acteurs entrent en jeu :

- **L'utilisateur** : grâce à son ou ses CSS, c'est la personne qui utilise les services des communautés dont il est membre.
- **L'autorité de confiance ou *trusted authority*** : c'est l'entité utilisée pendant la phase d'authentification pour prouver au système que l'utilisateur est bien celui qu'il prétend être et qu'il est bien membre de la communauté visée.

- **Le gestionnaire d'identités ou *ID Manager*** : c'est l'entité chargée de fournir des identités aux utilisateurs en les créant, les stockant et les rendant accessibles en ligne, et leur permettant de les supprimer ou modifier. Dans notre cas, il s'agira de créer et de gérer un WebID pour chaque utilisateur.
- **Le CSS** : c'est le client utilisé par l'utilisateur. Ce CSS s'authentifie auprès d'une communauté en utilisant la fonctionnalité d'authentification anonyme ou identifiée.
- **Le CIS ou communauté** : c'est l'élément qui propose des services à ses membres qui sont authentifiés.

4.4 Fonctionnalités du CSS et du CIS permettant l'authentification anonyme

Plaçons-nous d'abord du point de vue CSS et CIS. Le système à spécifier est donc l'ensemble CSS + CIS, et dans ce cas les acteurs sont l'utilisateur et l'autorité de confiance.

4.4.1 Diagrammes de cas d'utilisation

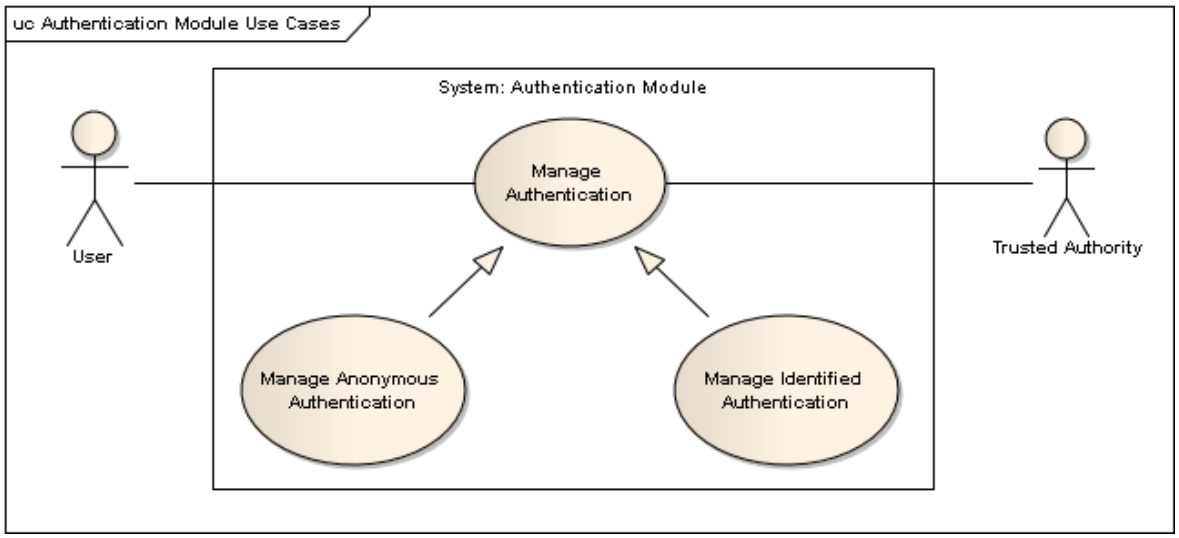


Figure 7 : Cas d'utilisation de l'authentification anonyme

Dans une seconde étape, on peut raffiner ce diagramme de cas d'utilisation en séparant CSS et CIS.

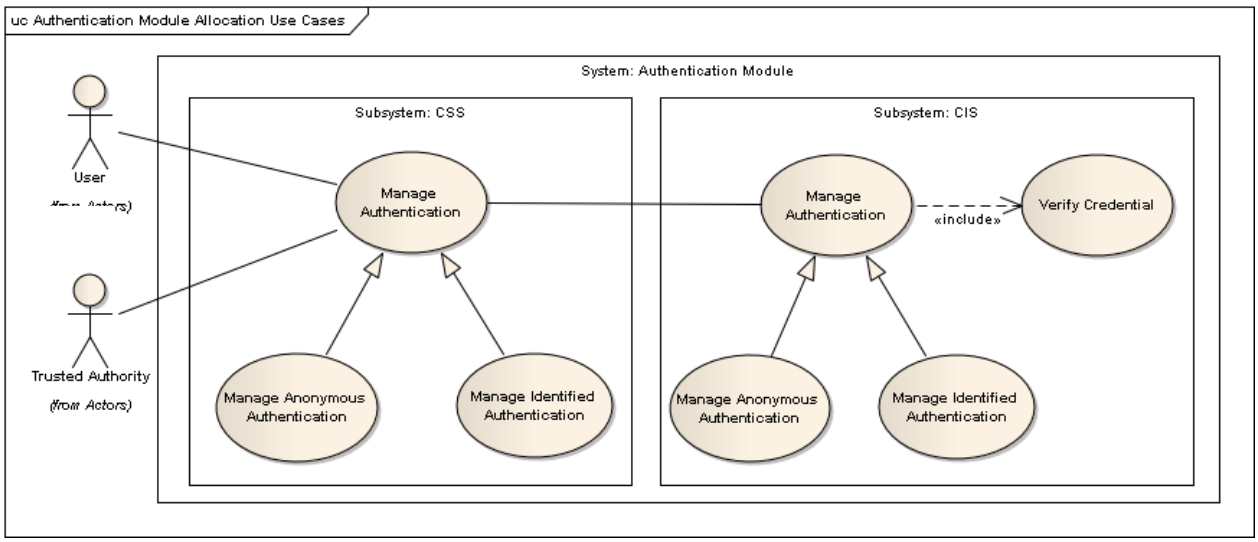


Figure 8 : Cas d'utilisation raffiné de l'authentification anonyme

Mon prototype, dans sa version actuelle, s'intéresse exclusivement à la partie authentification et laisse pour l'instant de côté les fonctionnalités d'inscription et de gestion de l'identité WebID des utilisateurs.

Le CSS et le CIS interagissent donc pour exécuter la fonctionnalité « Manage Authentication ». Cette fonctionnalité est abstraite et est spécialisée par deux sous-fonctionnalités : « Manage Anonymous Authentication » et « Manage Identified Authentication » (ce qui correspond à l'authentification classique). Etudions-les plus en détails.

4.4.2 Manage Authentification (anonyme ou identifié)

4.4.2.1 Description

L'authentification, qu'elle soit anonyme ou identifiée, s'effectue en trois étapes :

1. D'abord le CSS prépare l'authentification avec la communauté visée en échangeant une variable aléatoire (que l'on appelle souvent « nonce » en cryptographie) qui servira plus tard à vérifier les preuves d'identité et d'appartenance qu'enverra le CSS. C'est une étape de sécurité. La communauté profite de cet échange pour indiquer au CSS une ou plusieurs autorités de confiance auprès desquelles il peut s'authentifier.
2. Le CSS s'authentifie ensuite à l'autorité de confiance choisie en utilisant le protocole WebID. Si le CSS est bien membre du CIS indiqué, alors l'autorité de confiance lui remet deux certificats :
    - Le premier prouve son identité
    - Le second prouve son appartenance à la communauté indiquée
3. Le CSS va alors enfin s'authentifier auprès de la communauté en lui fournissant un ou plusieurs certificats :
    - Les deux certificats pour une identification.
    - Seulement le certificat d'appartenance à la communauté pour une authentification anonyme. Ce certificat ne fournit aucune autre information sur le membre.

    Ces certificats étant signés par l'autorité de confiance, la communauté est assurée que celle-ci a bien authentifiée l'utilisateur, et que c'est bien l'utilisateur qui s'est authentifié auprès de l'autorité de confiance grâce à la variable aléatoire échangée à l'étape 1. C'est le cas d'utilisation « Verify Credential ».

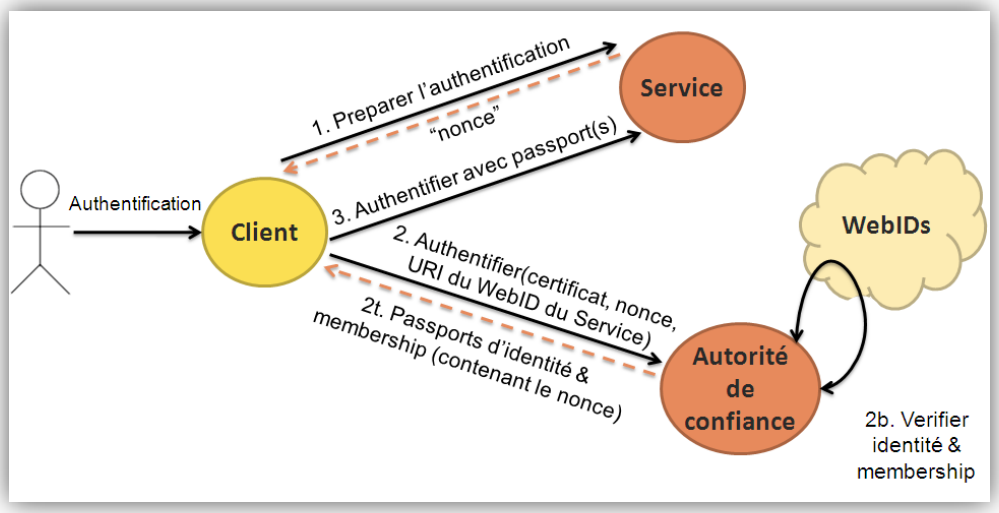


Figure 9 : Schéma explicatif de l'authentification

Ainsi, une communauté peut authentifier un membre et lui proposer tous ses services (mode identifié) ou seulement ceux ne nécessitant pas d'identification (mode anonyme).

Ce choix d'architecture offre de nombreux avantages. Tout d'abord, il exporte toutes les opérations de calculs un peu complexes hors du smartphone vers des supports à moindres contraintes. Et cette exportation n'ajoute qu'un seul échange réseau supplémentaire par rapport à un algorithme classique. Ainsi, le smartphone n'a pas à gérer tout ce qui traite du chiffrement ou de la certification, qui sont consommateurs de temps (et d'énergie) même sur un ordinateur de bureau.

De plus, c'est le client (le CSS) qui est à l'origine des transactions et qui veille donc à ne pas envoyer d'informations personnelles à la communauté ou au service auquel il souhaite s'authentifier. L'utilisateur n'a donc même plus besoin d'avoir confiance dans la communauté puisque de toute façon elle ne saura rien de lui s'il s'authentifie en mode anonyme.

Néanmoins, l'autorité de confiance doit obligatoirement bien porter son nom en possédant en effet la confiance de ses utilisateurs. Cet état de fait n'est pas un problème majeur, puisqu'en utilisant la technologie WebID on a séparé l'identité de l'organisme qui authentifie (ici l'autorité de confiance) et on permet ainsi à une communauté d'accepter plusieurs autorités de confiance. Ainsi, un utilisateur pourra choisir l'autorité à laquelle il souhaite s'authentifier et sélectionnera donc celle qui a le plus de crédit à ses yeux.

4.4.2.2 Diagramme de séquence

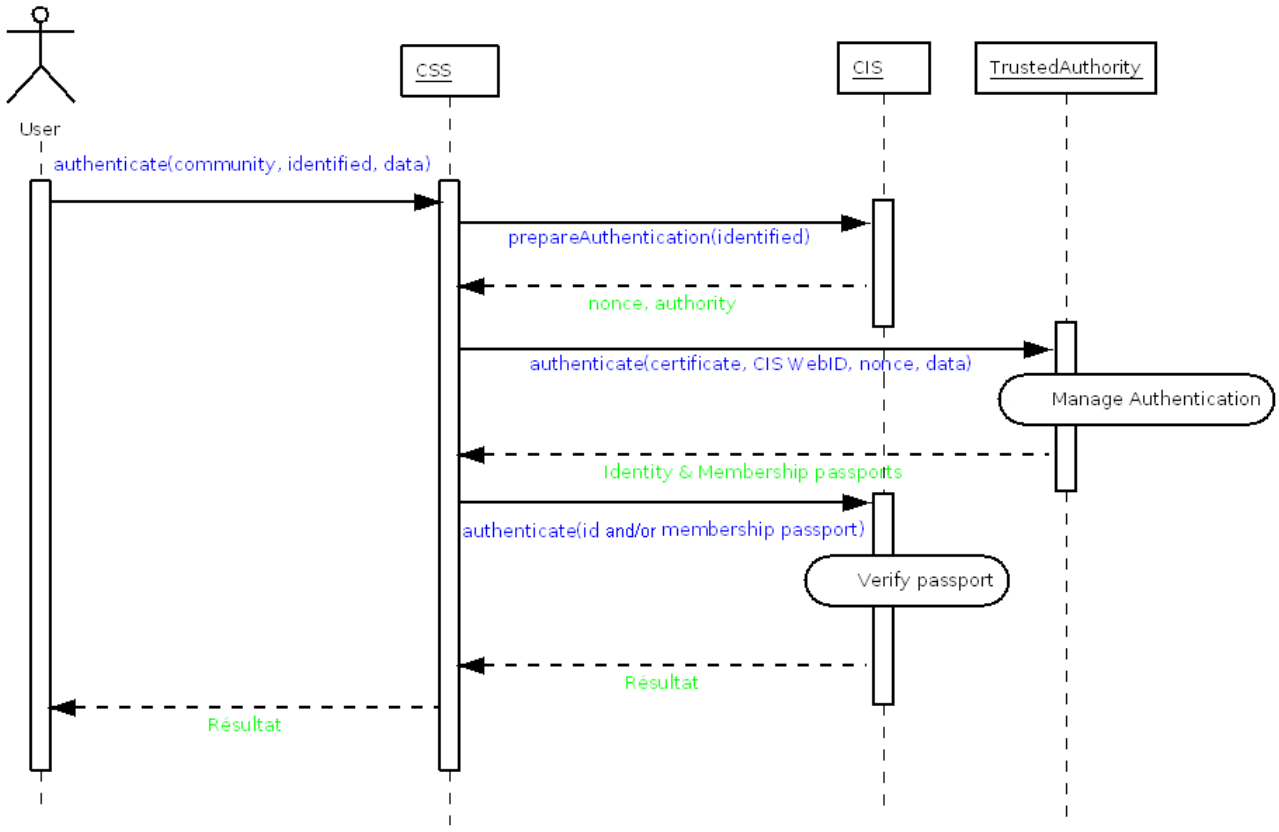
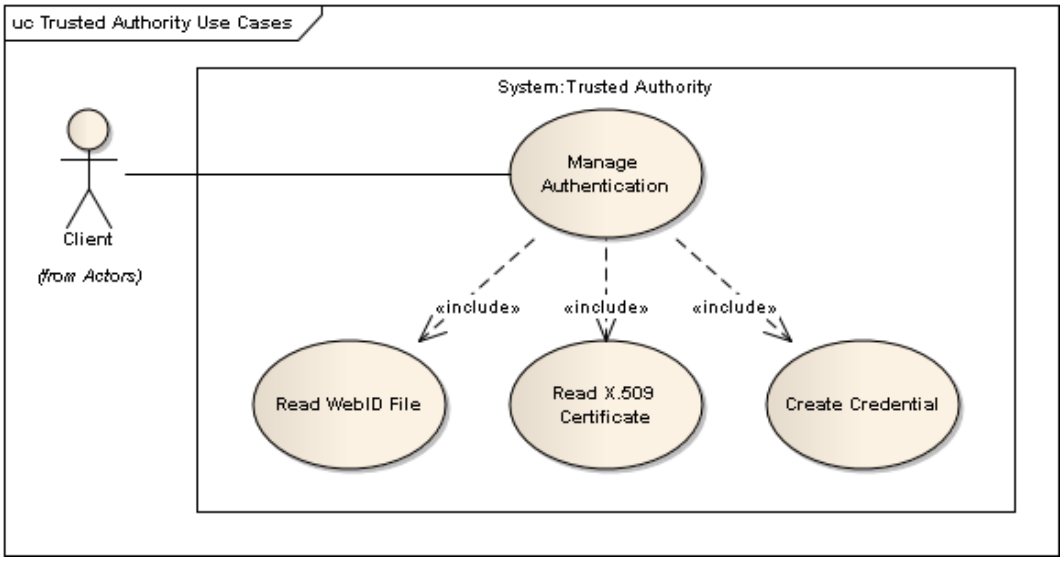


Figure 10: Diagramme de séquence authentification anonyme (CSS+CIS)

4.5 Fonctionnalités de l'autorité de confiance permettant l'authentification anonyme

Plaçons-nous maintenant du point de vue de l'autorité de confiance. Dans ce cas, le système à spécifier est constitué de l'autorité de confiance, et l'acteur principal est un client qui, dans SOCIETIES, est en fait un CSS.

4.5.1 Diagramme de cas d'utilisation



4.5.2 Manage Authentication

4.5.2.1 Description

Lorsque l'autorité de confiance reçoit une requête d'authentification, elle effectue les vérifications requises puis renvoie le résultat au client.

- Tout d'abord, le client envoie à l'autorité une requête d'authentification en précisant :
  - Le WebID de la communauté visée
  - Le certificat X.509 associé au WebID de l'utilisateur, preuve de son identité
  - Une variable aléatoire fournie par la communauté permettant la vérification future des passeports
- L'autorité vérifie alors l'identité de l'utilisateur en s'assurant :
  - De la validité du certificat X.509, ce qui est en général réalisé par la transaction TLS ou SSL,
  - Et de la correspondance entre la clé publique du certificat et celle du WebID associé
    - Pour se faire, l'autorité utilise les fonctions « Read WebID file » et « Read X.509 Certificate ».
- L'autorité vérifie ensuite l'appartenance de l'utilisateur à la communauté visée en s'assurant qu'il apparait dans la liste des membres du WebID de la communauté. Seule la communauté possède la liste de ses membres. Elle la rend accessible via son WebID à certaines autorités de confiance choisie.
  - Pour cette étape, l'autorité utilise les fonctions « Read WebID file ».
- Si tout se passe bien, l'autorité crée alors un passeport d'identité et un passeport d'appartenance à la communauté, qu'elle envoie au client afin de lui certifier ses

vérifications.

- Pour ce faire, l'autorité utilise la fonction « Create Credential ».

4.5.2.2 *Diagramme de séquence*

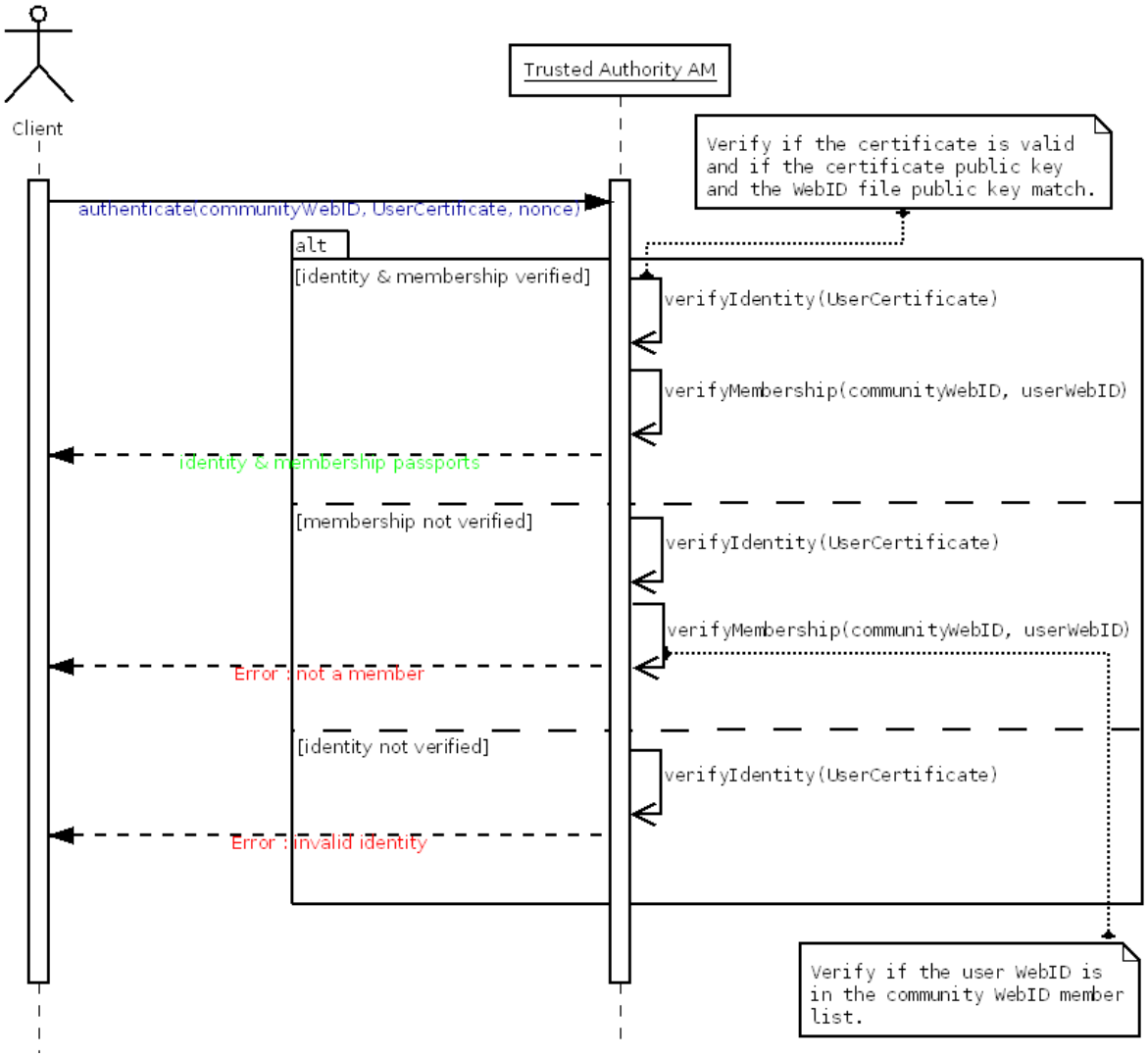


Figure 11 : Diagramme de séquence de l'authentification à une autorité de confiance

4.6 Spécifications techniques

Les spécifications fonctionnelles décrites de manière non formelles ci-dessus, permettent de comprendre le fonctionnement et les besoins de l'authentification anonyme. Mais il est nécessaire d'entrer davantage dans les détails d'un point de vue technique, en précisant chaque fonctionnalité et l'algorithme associé, de manière à ce que deux implémentations différentes de l'authentification anonyme (qui suivent les spécifications !) possèdent les mêmes fonctionnalités et puissent interagir entres elles.

4.6.1 Protocole de communication

J'ai donc précisé le protocole de communication entre les différents éléments. L'utilisation du protocole HTTP est recommandé, mais d'autres protocoles peuvent convenir (comme XMPP par exemple, un

protocole d'échange de messages XML via TCP), c'est un choix d'implémentation. Par contre, j'ai fixé tout ce qui concerne les requêtes et les réponses en choisissant le format JSON, simple d'utilisation et peu consommateur en taille, il est d'ailleurs recommandé par Google pour les communications sur Android. J'ai ensuite fixé la liste des requêtes possibles entre les trois protagonistes (CSS, CIS et autorité de confiance), en précisant :

- les champs nécessaires de chaque requête,
- les champs nécessaires pour chaque réponse associée, ainsi que les différents codes d'erreur possibles.

L'annexe 9.2.1 fournit les détails du format choisi et donne un exemple de requête et de réponse.

4.6.2 Gestion des données

Il faut de plus spécifier les données de chaque élément et leur format. Par exemple, le CSS doit posséder :

- une base de données SQL où il liste les CIS dont il est membre avec un format de table SQL précis
- un certificat X.509 associé à son WebID dont le format un petit peu particulier est détaillé dans l'annexe 9.2.2.1
- un fichier au format JSON définissant son identité (quel est l'URL de son WebID, quel est sa clé publique, …)

Et il faut faire de même pour le CIS et l'autorité de confiance, qui possèdent sensiblement les mêmes informations mais dont il faut bien préciser la teneur.

4.7 Conception

Pour implémenter un prototype à partir de spécifications, il faut faire des choix d'implémentations. D'abord des choix sur les technologies à utiliser, puis d'autres sur la manière d'implémenter les différents algorithmes.

4.7.1 Choix technologiques

Pour mon prototype, j'ai donc fait des choix technologiques suivantes :

- Le CSS est donc une application Android servant d'interface pour s'authentifier à une communauté
- La communauté est un serveur qui répond à des requêtes. J'ai donc créé un serveur Apache avec PHP et MySQL. On parle parfois de serveur AMP.
- De même pour l'autorité de confiance, c'est un serveur qui répond à des requêtes particulières, donc un serveur AMP.
- Le gestionnaire d'identité permet de simuler l'élément qui stocke en ligne les identités, ici des fichiers WebID. Dans mon prototype, c'est donc aussi un serveur AMP, mais qui ne fait que stocker des fichiers.

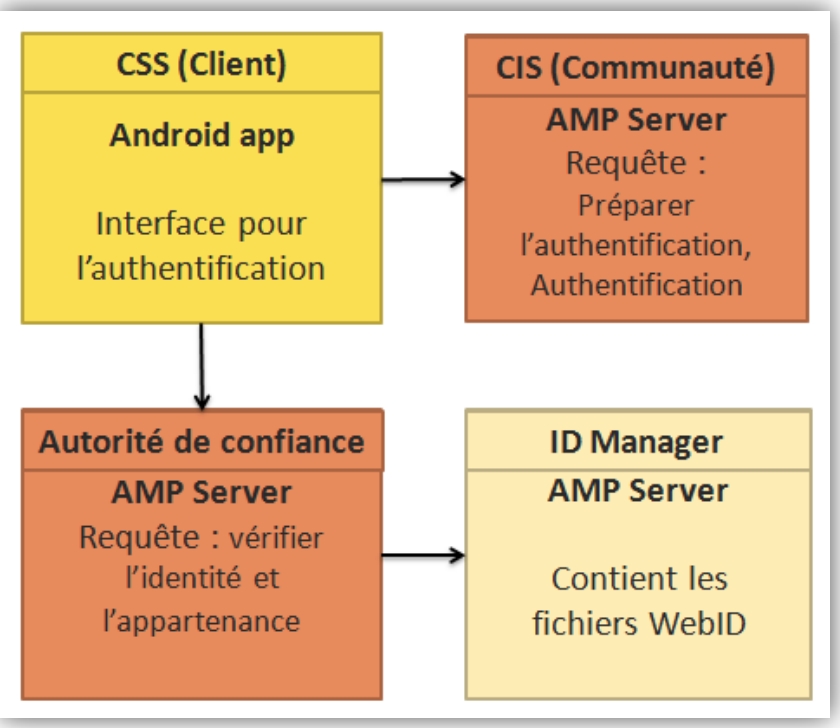


Figure 12 : Architecture du prototype d'authentification anonyme

D'autre part, on a vu que les différents éléments avaient besoin de tables SQL pour stocker leurs données. Pour cela, les serveurs AMP ont accès à une base de données MySQL. Quant à l'application Android, elle tirera partie de la base de données SQLite nativement présente sur cette plateforme. Conçue spécialement pour les systèmes à faibles ressources, cette base de données possède moins de fonctionnalités qu'une base MySQL, mais cela suffit largement pour un stockage simple comme c'est le cas ici.

L'utilisation du format JSON se fera nativement en PHP sur les serveurs AMP, mais nécessitera l'installation de la bibliothèque GSON pour Android, spécialement adaptée aux OS mobiles et offrant des fonctionnalités avancées comme la transformation JSON → objet Java et objet Java → JSON.

Pour tout ce qui traite de la cryptographie (vérification de certificat X.509, vérification d'une signature à partir d'une clé publique, etc.), il existe une implémentation native OpenSSL en PHP. Malheureusement, cette bibliothèque n'est pas tout à fait au point, et même sa documentation précise que tout n'est pas complet. De plus, il existe plusieurs formats possible pour afficher une clé publique. Le format PEM convient bien au stockage des clés de chiffrement, mais certaines fonctionnalités d'OpenSSL requièrent un autre format et bien sûr la transformation d'un format à un autre n'est pas prévue par cette bibliothèque ! Tout cela entraîne quelques difficultés… Au final, il convient de se munir : de la bibliothèque OpenSSL de PHP, d'une bibliothèque non native appelée PHPSecLib (21) (moins performante mais fonctionnelle) pour remplacer les fonctionnalités boguées d'OpenSSL, et des documentations OpenSSL de PHP (22) ou autre pour créer les fonctionnalités manquantes, notamment celle de transformation inter-format.

Quant à la manipulation des fichiers WebID, cela peut se faire avec le langage de requêtes SPARQL spécialisé dans la lecture de fichiers au format XML/RDF qui est le format des fichiers WebID. Pour cela, il existe une librairie PHP nommée RAP, qui est difficile à prendre en main, mais permet finalement, en utilisant SPARQL, de chercher une chaine de caractères (par exemple une clé publique) dans un fichier WebID.

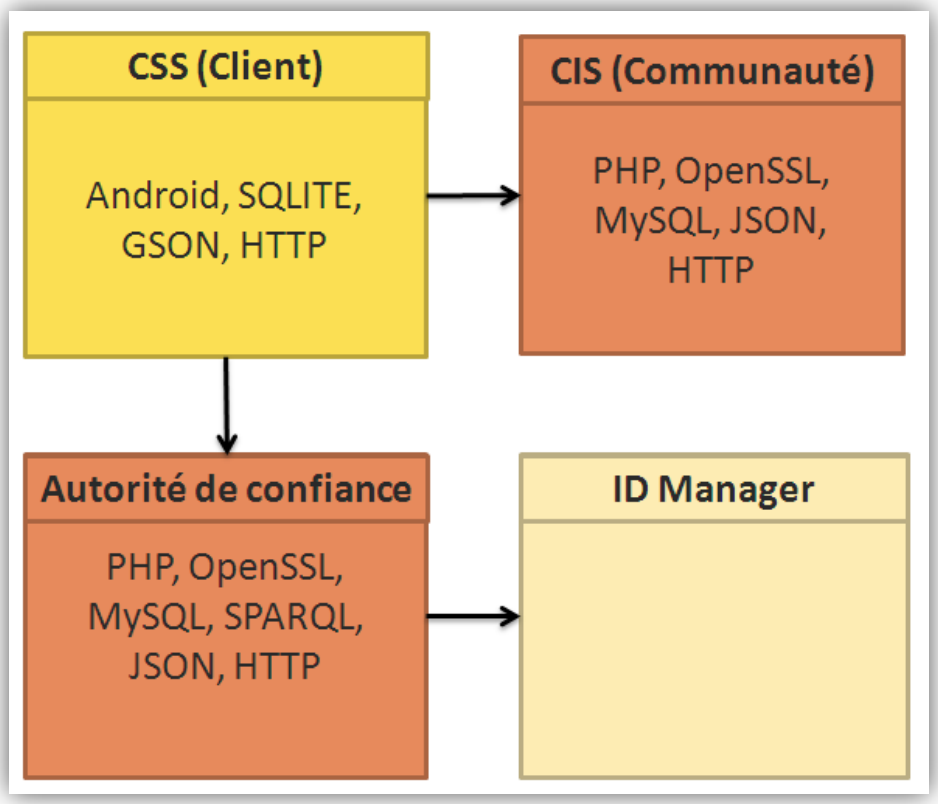


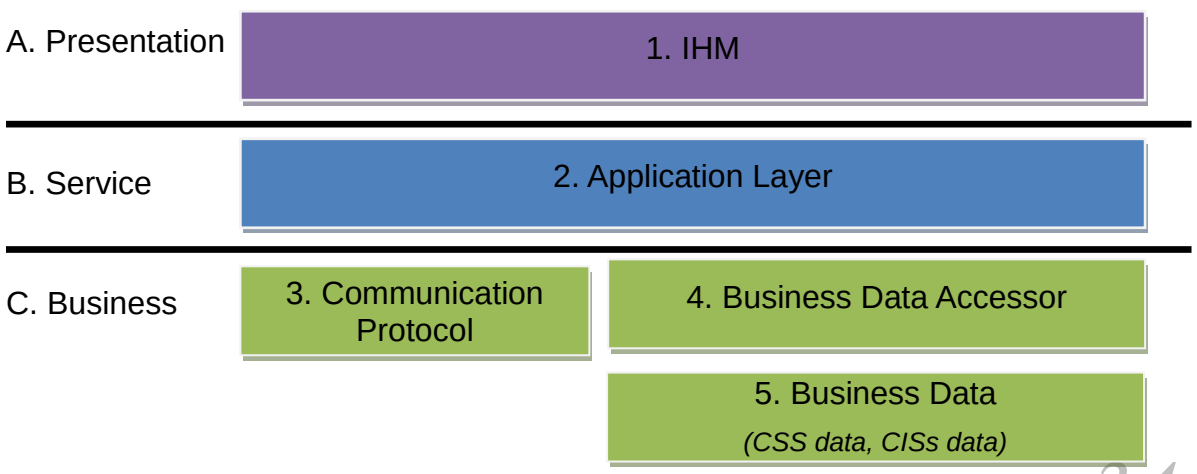
Figure 13 : Technologies utilisées par les différents éléments pour l'authentification anonyme

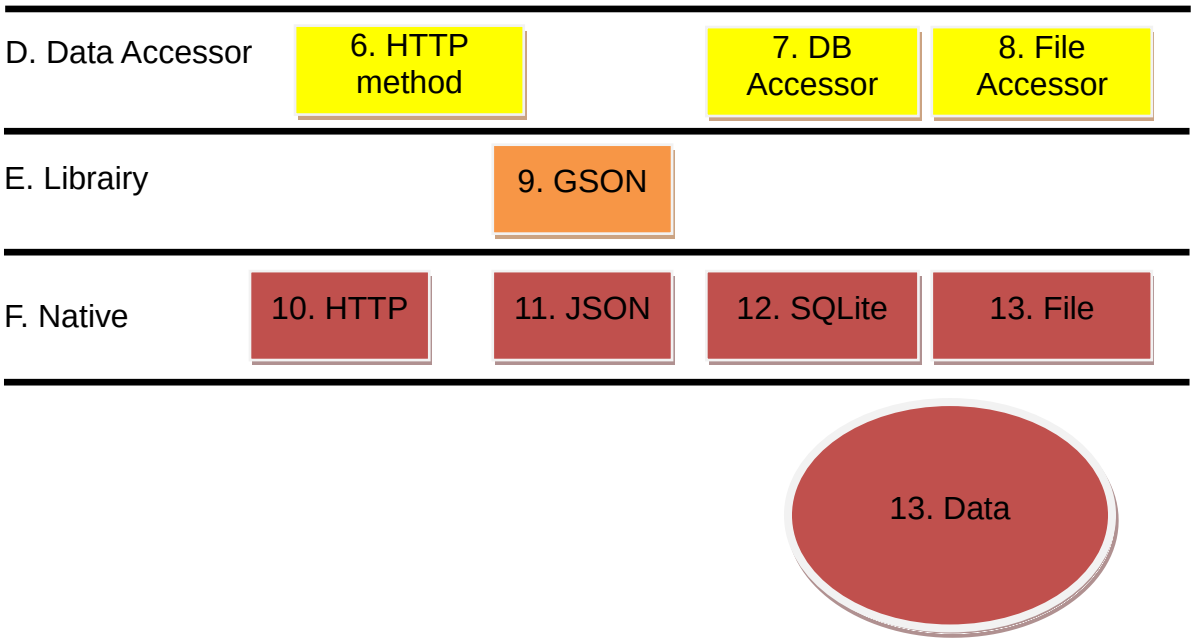
4.7.2 Séparation en couches

De manière à obtenir un code propre et rigoureux, j'ai séparé mon prototype en plusieurs couches, les plus indépendantes possibles les unes des autres. Cette architecture modulaire permet de séparer les différentes parties du code, et rend possible le remplacement d'une couche (pour l'améliorer ou changer de technologie) en impactant au minimum le reste du code.

Cette séparation en couche m'a permis de créer des diagrammes de classes pour chaque éléments, puis de générer automatiquement un squelette de code Android (avec le plugin Eclipse eUML2) et PHP (avec le logiciel Dia) qui m'a servi de base de travail pour mon prototype. Le diagramme UML de l'application Android est disponible en annexe 9.2.3.

Etudions plus en détails la séparation en couche pour le CSS qui est une application Android. On retrouve des couches similaires dans le CIS et l'autorité de confiance, mais en y ajoutant une couche pour gérer la cryptographie, et une autre pour manipuler des fichiers WebID.





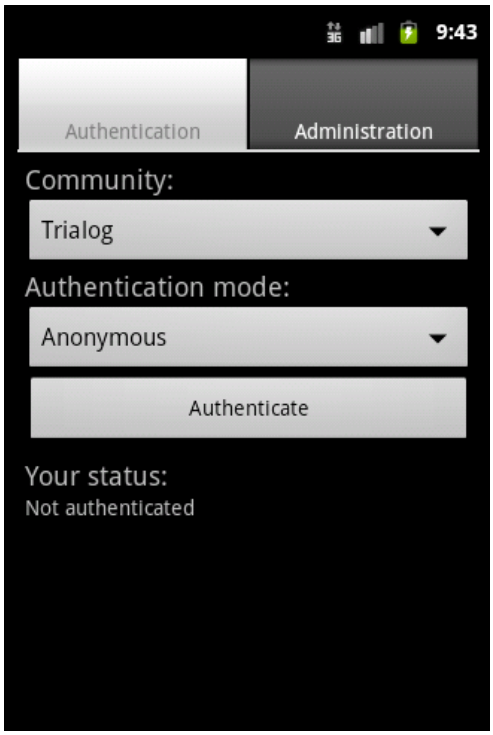
1. **IHM** : Cette couche se charge de l'affichage des différents écrans et des messages envoyés à l'utilisateur. Elle effectue des traitements locaux (contrôles effectués au niveau du dialogue avec l'IHM : action sur un clic, etc.) et effectue des appels à la couche applicative. Les messages reçus de la couche applicative sont traités pour être affichés à l'écran.
2. **Couche applicative** : Cette couche se charge des traitements globaux, sous l'action de l'IHM elle exécute les algorithmes correspondants en utilisant les couches plus basses, elle regroupe les messages résultats et les retransmet à l'IHM.
3. **Protocole de communication** : Cette couche est une couche d'abstraction utilisant le protocole d'échange spécifié. Elle contient des méthodes pour chaque requête à effectuer, auxquels on fournit en paramètre des objets classiques ou des objets métiers
4. **Business Data Accessor** : Cette couche permet d'accéder facilement aux objets métiers, c'est-à-dire les données stockées dans le CSS
5. **Business Data** : données stockées dans le CSS sous une forme transparente pour le système, indépendamment du stockage utilisé
6. **HTTP method** : cette couche implémente le protocole d'échange spécifié en utilisant REST et les couches plus basses. C'est elle qui se charge de créer la requête HTTP, de l'exécuter et d'en récupérer la réponse
7. **DB Accessor** : Couche d'accès aux données de la BDD SQLite
8. **File Accessor** : Couche d'accès aux données stockés dans des fichiers (format JSON)
9. **GSON** : GSON est une bibliothèque permettant de manipuler facilement le format JSON sur Android

4.8 Résultats de l'implémentation et validation

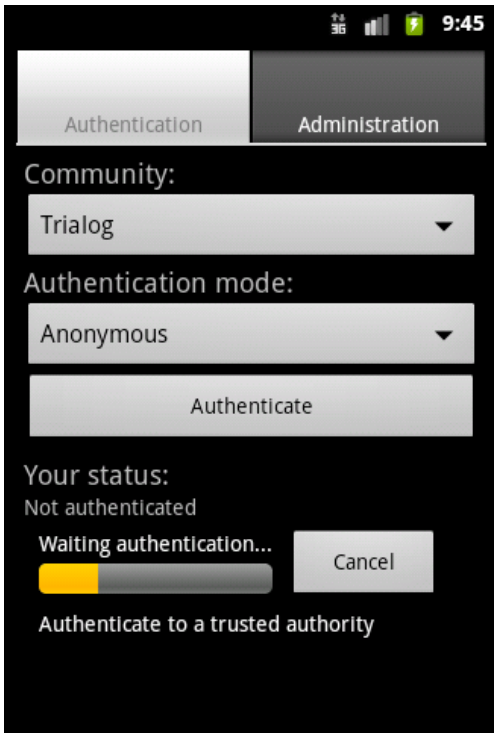
Au final, j'ai pu trouver une solution pour minimiser l'échange de données personnelles durant l'authentification en spécifiant le concept d'authentification anonyme. Le prototype que j'ai réalisé permet de s'authentifier en mode anonyme ou identifié à une communauté. L'authentification anonyme via WebID s'effectue entre 1 à 3 secondes selon la connexion ce qui

est donc tout à fait viable pour une utilisation réelle. C'est une preuve de concept que l'on peut s'authentifier à une véritable communauté qui fournira des services en conséquences du mode d'authentification choisi.

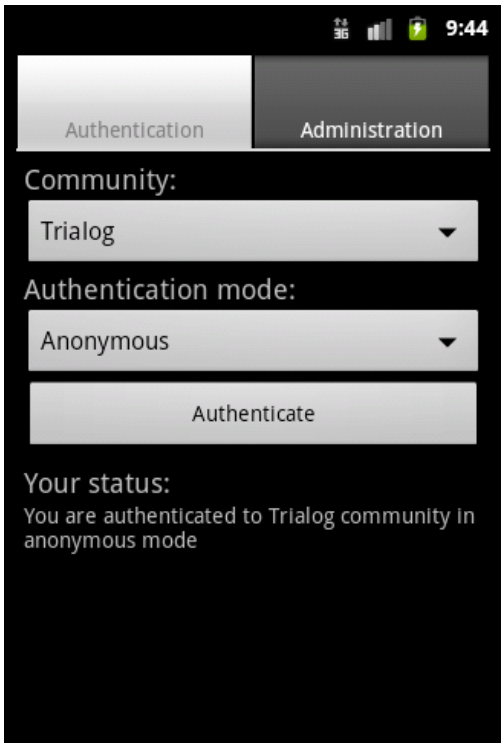
Voici quelques impressions d'écran de mon prototype.



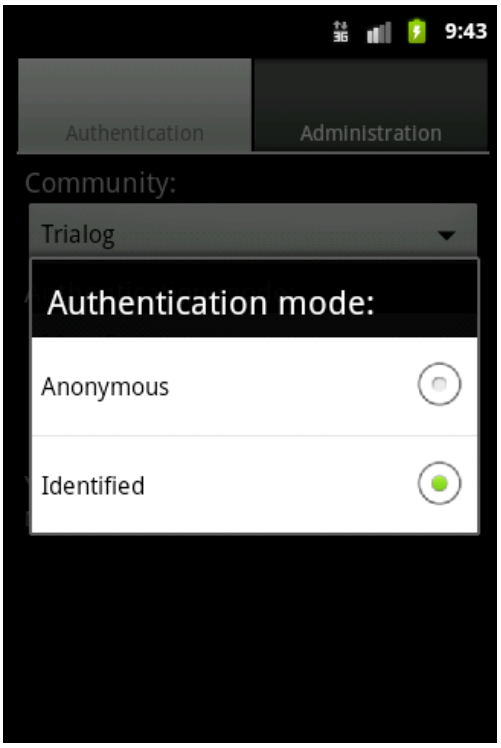
L'interface d'authentification permet de lancer l'authentification en cliquant sur « Authenticate »



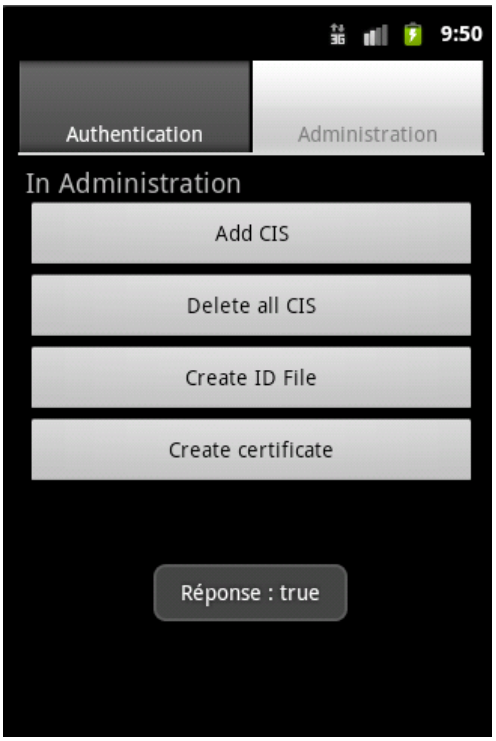
Lorsque l'on a cliqué il faut attendre entre 1 à 3 secondes durant lesquelles les actions effectuées s'affichent accompagnée d'une barre de progression. L'authentification est lancé dans un thread différent du thread de l'IHM ce qui évite de bloquer l'application.



Si l'authentification a réussi, un message s'affiche pour confirmer l'authentification.



Il est possible de sélectionner le mode d'authentification et le CIS visé.



L'onglet d'administration permet d'initialiser facilement les données d'un CSS de manière à pouvoir utiliser la fonctionnalité. En réalité, ces actions seraient effectuées lors de l'installation de l'application et lors de l'inscription à un CIS.

L'application Android est téléchargeable à l'adresse suivante : http://css.societies.maridat.com.

Et les serveurs du CIS et de l'autorité de confiance sont accessibles via les adresses suivantes : http://cis.societies.maridat.com, http://ta.societies.maridat.com. Ils ne font que répondre aux requêtes qu'on leur envoie, il n'y a donc pas d'interface pour les tester indépendamment du CSS.

Dans le cadre d'exposés d'explorations des technologies et concepts potentiellement utiles pour SOCIETIES, j'ai pu présenter l'authentification anonyme et le prototype que j'ai réalisé lors du meeting SOCIETIES à Paris du 21, 22 et 23 juin aux autres partenaires du projet. Mon exposé a reçu un accueil favorable, d'autant plus que le protocole WebID se profile de plus en plus comme protocole d'authentification sur SOCIETIES. La question cruciale est finalement de savoir dans quelle mesure implémenter ce mode d'authentification dans SOCIETIES. J'ai donc écrit un document récapitulatif de mon travail pour qu'il soit ajouté à l'un des délivrables intermédiaires de SOCIETIES. La décision finale de l'ajout ou non de cette fonctionnalité à SOCIETIES sera donc prise lorsque les partenaires de SOCIETIES auront fini de définir les bornes de ce projet.

5 Obscurcissement de données

Un autre moyen de minimiser l'utilisation et le partage de données personnelles est de les obscurcir, c'est-à-dire de trouver un moyen de réduire leur teneur en informations personnelles. L'exemple le plus parlant est celui d'une localisation : il est possible de connaître la localisation précise d'un utilisateur possédant un système GPS (ex : « 137, rue des Lilas, 75000 Paris »), et pour éviter d'enfreindre sa vie privée, il est possible de proposer à cet utilisateur d'obscurcir sa localisation, en y ôtant quelques précisions, lorsqu'il souhaite la partager avec d'autres personnes (ex : l'adresse précédente devient simplement « Paris »). A partir de cet exemple simple, on peut saisir le principe de l'obscurcissement des données, mais on peut aussi remarquer que la méthode d'obscurcissement décrite ne fonctionnerait pas pour protéger le genre d'un utilisateur (féminin ou masculin), ou ses photos (pour lesquels on pourrait rendre les visages flous par exemple). Il apparait donc qu'il n'y a pas de méthode générale pour obscurcir tout type d'information : à chaque type de données son algorithme d'obscurcissement.

Dans SOCIETIES, un module « Data Obfuscation Manager » se chargera de réaliser cet obscurcissement des données lorsque cela est utile. L'objet de mon étude se concentre sur la manière d'effectuer cet obscurcissement sur une donnée de type géolocalisation (généralement composée de : latitude, longitude, précision de la mesure), tout en construisant une architecture pouvant supporter une méthode d'obscurcissement pour chaque type de données.

5.1 Objectifs et exigences

L'objectif de mon étude au sujet de l'obscurcissement est de partir d'une géolocalisation et d'un niveau d'obscurcissement et d'obscurcir cette donnée en conséquence. Cependant, ce système doit se situer dans le contexte de SOCIETIES et respecter ses règles de conception puisqu'il sera intégré plus ou moins directement au module « Data Obfuscation Manager ». Par conséquent, l'architecture de ce système doit aussi fournir le niveau d'abstraction nécessaire pour permettre la mise en place d'une API utilisable pour plusieurs algorithmes d'obscurcissement de données, sans se limiter à la géolocalisation.

D'autre part, dans SOCIETIES, on ne sait pas encore si l'obscurcissement sera effectué sur un smartphone ou sur un serveur. Une exigence supplémentaire est donc de rendre faisable cette opération sur un support mobile avec des ressources limitées (notamment en terme de batterie) ou sur un support fixe comme un ordinateur de bureau ou un serveur.

5.2 Etat de l'art

5.2.1 Définition de la géolocalisation

La géolocalisation est le terme employé pour parler de localisation d'une position à la surface d'une planète, généralement la planète Terre. La manière la plus courante de définir une position est d'utiliser des coordonnées géographiques constituées d'une latitude et d'une longitude et de l'altitude par rapport au niveau de la mer. Plusieurs techniques permettent de calculer une position, notamment le GPS, ou des techniques basées sur la connaissance des émetteurs Wi-Fi ou des Cell-ID (les numéros d'identification des stations BTS du réseau GSM). Cela a été particulièrement démocratisé par l'émergence des smartphones proposant des solutions de géolocalisation, souvent même très précises grâce à l'utilisation d'une puce GPS.

Le standard HTML5 en cours de définition par le W3C, fournit une API pour utiliser la géolocalisation (23) via un navigateur Web, et définit celle-ci comme étant composée des éléments suivants :

Libellé	Type de donnée	Unité	Intervalle	Description
latitude	Double	degré	[-90, 90]	Angle formé par l'équateur et la position à localiser par rapport au centre de la Terre
longitude	Double	degré	[-180, 180]	Angle formé par l'équateur et la position à localiser par rapport au centre de la Terre
horizontal accuracy (souvent appelée accuracy)	Float	mètre	≥ 0	Puisque l'on effectue une mesure, celle-ci est plus ou moins précise. La véritable position de l'utilisateur est donc quelque part dans le cercle de centre <latitude, longitude> et de rayon horizontal accuracy
altitude	Double	mètre	> -422	Altitude par rapport au niveau de la mer
vertical accuracy	Float	mètre	≥ 0	
heading	Double	degré	≥ 0	Décalage en degré par rapport au Nord magnétique
Speed	Double	mètre / seconde	≥ 0	S'il y a un déplacement
timestamp	Date			Date de la mesure

Pour l'obscurcissement d'une géolocalisation, toutes ces données ne sont pas utiles. Ainsi, Ardagna et al. (24) se contentent des trois premières : latitude, longitude, précision (i.e. *horizontal accuracy*), et se servent énormément de la notion de cercle dans lequel la véritable position de l'utilisateur est uniformément répartie.

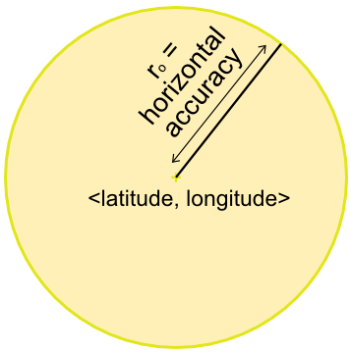


Figure 14 : Schématisation de la géolocalisation
L'utilisateur se trouve quelque part n'importe où dans ce cercle

5.2.2 Définition approfondie de l'obscurcissement

Jusqu'ici, on a défini l'obscurcissement comme un moyen de réduire la teneur en informations personnelles d'une donnée. Mais Duckham & Kulik vont plus loin en différenciant trois catégories d'obscurcissement : réduction de la précision, atténuation avec du flou, ajout d'inexactitude (25).

Prenons un exemple pour préciser leur pensée : « Alice est située (exactement et précisément) au croisement de la rue Vaugirard et de l'avenue Grande Aire à Créteil. ».

- **Réduire de la précision (*imprecision*)** : en faisant en sorte que plusieurs lieux puissent correspondre à la localisation obscurcie.
    - Ex : « Alice est située dans la rue Vaugirard à Créteil. »
    - C'est la manière intuitive de concevoir les choses, facilement transcriptible en termes d'algorithme.
- **Atténuer avec du flou (*vagueness*)** : en rendant floue la frontière permettant d'apprécier si l'on se situe au bon endroit ou non.
    - Ex : « Alice est située près de la rue Vaugirard à Créteil. »
    - Ce type d'obscurcissement permet bien de réduire la teneur personnelle de l'information « géolocalisation », mais sa transcription en algorithme n'est pas aisée. En effet, comment traduire cette notion de « proche de », « aux alentours de » ?
- **Ajouter de l'inexactitude (*inaccuracy*)** : en introduisant de fausses informations dans une donnée.
    - Ex : « Alice est située au croisement du boulevard des Vaches et de l'avenue Grande Aire à Créteil. »
    - Cette solution est efficace pour un obscurcissement drastique, mais n'est pas applicable dans tous les cas et peut parfois poser des questions d'éthique.

Ces précisions permettent de classer les différents algorithmes d'obscurcissement, les plus efficaces étant ceux qui combinent ces trois catégories.

5.2.3 Niveau d'obscurcissement

La notion d'obscurcissement étant mieux définie, il convient de préciser comment l'utilisateur final va percevoir et choisir le taux d'informations personnelles qu'il souhaite partager. On utilise pour cela la notion intuitive de « niveau d'obscurcissement » (*obfuscation level*) (24). L'utilisateur choisit le niveau qu'il souhaite, ce qui revient finalement à choisir le degré de vie privée qu'il est prêt à exposer, et l'algorithme se charge d'obscurcir la donnée voulue en conséquence. Mais ceci entraîne quelques questions.

5.2.3.1 Comment formaliser ce niveau d'obscurcissement, à quoi correspond-t-il ?

Ardagna and al. (24) ont effectué des travaux pour formaliser cette notion dans le domaine de la géolocalisation et utilise le terme de « *relevance* » c'est-à-dire « pertinence ». Ils définissent la pertinence d'un obscurcissement comme le quotient au carré de la précision (horizontal accuracy) théorique optimale mesurable r_o, et la précision que l'on obtient finalement r_i. Donc la « relevance » $R_i = \frac{{r_o}^2}{{r_i}^2}$. Ainsi, si on a les coordonnées mesurées d'une position, on a alors la précision r_m de la mesure et on peut calculer la « relevance » $R_m = \frac{{r_o}^2}{{r_m}^2}$ associée. Si l'utilisateur choisit une « relevance » $R_f = \frac{{r_o}^2}{{r_f}^2}$, il possible de retrouver alors la précision r_f de la géolocalisation à partager.

Les travaux d'Ardagna and al. (24) formalisent bien la notion de niveau d'obscurcissement d'une manière tout à fait adaptée pour la géolocalisation. Le fait de faire référence à une valeur théorique optimale r_o permet d'avoir un obscurcissement suffisamment efficace pour des données précises (pour lesquelles le cercle est petit), et qui réduit l'ajout de flous pour des données déjà peu précises (où le cercle est déjà grand). Malheureusement, il est extrêmement difficile de connaître ce r_o

puisqu'il varie en fonction de l'emplacement où on se trouve, de la position des satellites dans le cas du GPS, et qu'il n'est pas calculable. Connaître une constante qui varie en fonction de la situation n'est définitivement pas trivial. Les travaux d'Ardagna et al. sont peu précis à ce sujet et un approfondissement de cette piste de recherche serait intéressant.

5.2.3.2 Comment faire la correspondance niveau d'obscurcissement – obscurcissement ?

Il existe plusieurs algorithmes d'obscurcissement, et généralement chaque type de données doit être obscurci de manière particulière. De même, faire la correspondance entre le niveau d'obscurcissement et l'obscurcissement à opérer dépend du type de données. Et cela est plus ou moins difficile à réaliser. Pour répondre à cette question, il faut séparer le problème en deux cas :

- Niveau d'obscurcissement continu : comme pour la géolocalisation, l'*obfuscation level* est alors par exemple une valeur entre 0 et 1 ou un pourcentage.
- Niveau d'obscurcissement discret : comme pour le nom d'un individu, il faut séparer l'obscurcissement en classes (0 : pas d'obscurcissement Nom+Prénom+Pseudonyme, 1 : Nom+Prénom, 2 : Pseudonyme, 3 : rien du tout, 4 : un faux nom) et proposer un choix de niveaux en conséquence.

A partir de là, nous avons de bonnes bases pour établir une correspondance adaptée pour chaque type de données.

5.2.3.3 Comment traduire cela en termes d'expérience utilisateur ?

Ou comment permettre à l'utilisateur de choisir intuitivement le bon niveau d'obscurcissement ? La question reste ouverte et a de l'avenir comme tout ce qui concerne l'ergonomie de la protection des données personnelles.

De manière générale, il convient d'utiliser un sélecteur graphique. Un curseur à déplacer pour un niveau continu et un choix de vignettes pour un niveau discret par exemple. La figure ci-dessous est le choix que j'ai fait dans mon prototype d'obscurcissement d'une géolocalisation sous Android.

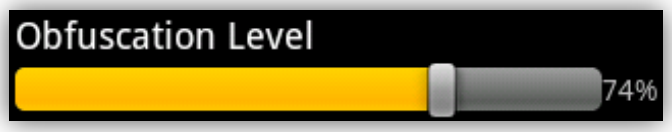


Figure 15 : un curseur à déplacer sous Android

Mais cette réflexion va plus loin, puisque le terme d'obscurcissement (*obfuscation*) est certes adapté et suffisamment précis pour le monde scientifique, mais il ne convient pas du tout pour une bonne expérience utilisateur. A Trialog nous avons commencé à réfléchir pour trouver un terme plus adapté, plus proche des mots comme « précision » ou « flou », mais nous n'avons encore arrêté aucune décision.

5.2.4 Etudes des différents moyens permettant d'obscurcir une géolocalisation

Plusieurs recherches ont été effectuées sur l'obscurcissement d'une géolocalisation. On peut classer ces recherches en trois types d'algorithmes :

- **Par reverse-geocoding puis geocoding**, c'est-à-dire la transformation de coordonnées géographiques en une adresse postale, puis l'inverse après avoir enlever des précisions à l'adresse.
- **Par camouflage en déplaçant la localisation dans une zone peuplée** à proximité comme un centre-ville une autoroute, etc. (26) (25) (27)

- **Par augmentation ou réduction de la précision de la mesure**, ou même déplacement du centre du cercle de géolocalisation (24).

Ce dernier type d'algorithme est aujourd'hui le plus simple à formaliser et à relier à la notion de niveau d'obscurcissement. C'est aussi le seul qui ne nécessite pas de communication avec des services externes, ou l'accès à une base de données de connaissances. Ces différentes raisons m'ont poussé à le choisir comme algorithme d'obscurcissement. Pour ce faire, je me base sur l'article d'Ardagna et al. (24) sans prendre en compte leur notion de précision optimale théorique r_o.

5.3 Description du système : module d'obscurcissement

Le système que j'ai étudié est composé du module d'obscurcissement (nommé « Data Obfuscation Manager »). Les acteurs avec lesquels ce module interagit se limitent au gestionnaire de vie privée (nommé « Privacy Manager »), puisque dans l'architecture de SOCIETIES il sera le seul à gérer les différentes actions de protection des données personnelles. Mais au final, c'est l'utilisateur qui est l'actionneur initial d'un obscurcissement lorsqu'il partage ses données ou utilise les services d'une communauté.

5.4 Cas d'utilisation

5.4.1 Scénario 1 : récupérer une donnée obscurcie

Alice est une photographe amateur de talent et sa passion est de publier ses photos sur le Web pour pouvoir les partager avec ses amis ou d'autres passionnés. Elle publie donc souvent des photos à l'aide de SOCIETIES en les associant à la géolocalisation de l'endroit où elles ont été prises. Cependant, elle a conscience qu'il est important de protéger sa vie privée, par conséquent, elle a configuré son profil pour que seuls sa famille et ses amis proches puissent avoir accès à la géolocalisation exacte de ses photos. Son cercle de connaissances passionnées de photographies a seulement le droit d'accéder à la région où ont été prises ses photos. Quant aux inconnus, seul le pays de la prise de vue leur est indiqué.

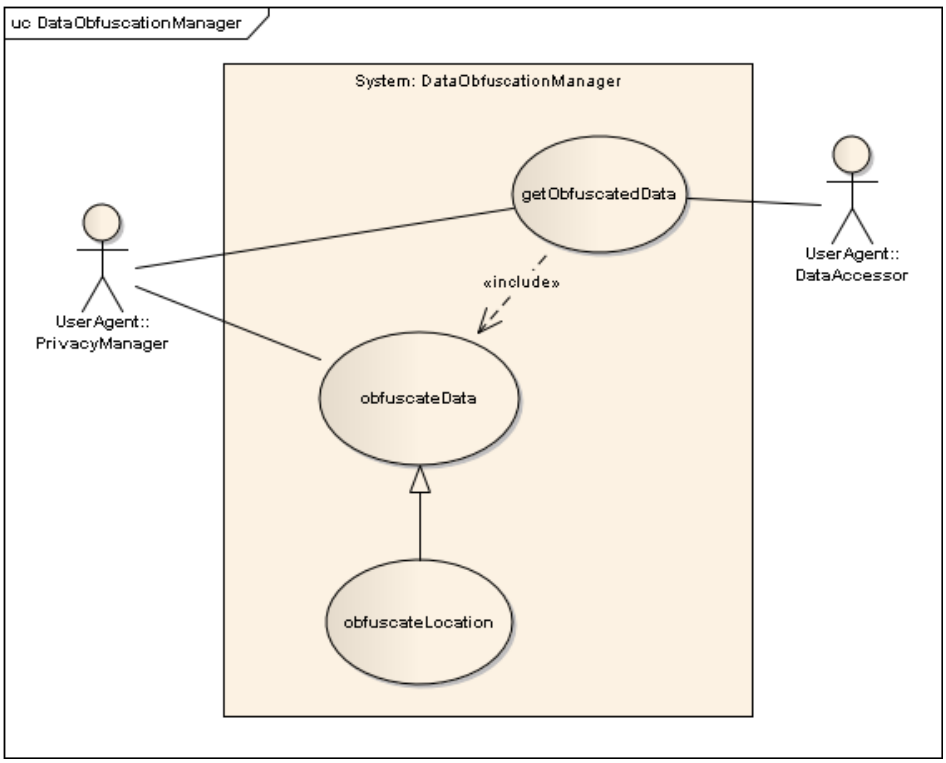
Donc, lorsque Bob (une connaissance de Londres) souhaite regarder les dernières photos d'Alice, le gestionnaire de vie privée de cette dernière effectue quelques vérifications. Définitivement, Bob ne fait pas parti de la famille d'Alice, ni de ses amis proches, et il n'est pas passionné de photos, le gestionnaire de vie privée précise donc que la position de la prise de vue devra être obscurcie pour atteindre une précision à l'échelle du pays. Il lance alors la récupération des photos et de leur géolocalisation, et obscurcit les géolocalisations à l'aide de l'obscurcisseur de données. Bob ne pourra ainsi jamais savoir où Alice a passé ses vacances.

5.4.2 Scénario 2: obscurcir une donnée

Alice a récemment découvert un service Web lui permettant, à partir de la géolocalisation d'une de ses photos, de découvrir les photos prises au même endroit par d'autres utilisateurs. Elle a plus ou moins confiance en ce service, mais on ne sait jamais. Elle décide donc de l'utiliser, mais en camouflant un peu la géolocalisation de ses photos, pour éviter que son itinéraire de vacances ne soit visible par tout le monde si le service s'avérait malveillant.

Donc, lorsqu'elle utilise ce service, le gestionnaire de vie privée va vérifier si Alice a autorisé le partage de sa géolocalisation, et obscurcir cette dernière à l'aide de l'obscurcisseur de données. Alice découvrira alors les photos prises par d'autres passionnés aux alentours de là où elle a pris les siennes.

5.4.3 Diagramme de cas d'utilisation



5.4.3.1 Get an obfuscated data

Lorsque le gestionnaire de vie privée souhaite avoir la main sur le gestionnaire d'accès aux données, il utilise ce cas d'utilisation pour récupérer la donnée auprès du gestionnaire d'accès aux données et lancer l'obscurcissement grâce au cas d'utilisation : obfuscate a data.

5.4.3.2 Obfuscate a data

Ce cas d'utilisation est générique. Son rôle est de sélectionner l'algorithme d'obscurcissement qui va réellement être utilisé, puis de l'exécuter.

Pour ce faire, il est nécessaire d'avoir :

- la donnée à obscurcir,
- le niveau d'obscurcissement que l'on souhaite obtenir,
- la sémantique de cette donnée, c'est-à-dire ce qu'elle signifie (ex : géolocalisation, nom, photo). Cette information est fournie en amont, elle est parfois directement liée à la donnée comme c'est le cas pour les données d'une identité (nom, prénom, géolocalisation, etc.), ou alors le lien est fait par le gestionnaire de vie privée en analysant son contexte.

A partir de la sémantique de la donnée fournie, cette fonctionnalité va choisir l'algorithme le plus adapté. Par exemple pour une géolocalisation : obfuscate a location.

5.4.3.3 Obfuscate a location

Ce cas d'utilisation est spécifique à un type d'obscurcissement, celui qui traite la géolocalisation. C'est le cœur de ce module, puisque c'est lui qui effectue réellement l'obscurcissement, on verra comment en abordant l'aspect technique.

5.5 Spécifications techniques

5.5.1 Gestion des données

Le module d'obscurcissement manipule trois types de données différentes :

- Une géolocalisation
- Un niveau d'obscurcissement qui est un entier naturel entre 0 exclus et 100 inclus. La valeur 100 correspond à la géolocalisation non-obscurcie. En effet, pour l'obscurcissement d'une géolocalisation, on définit ainsi **le niveau d'obscurcissement L : le cercle final formera L% du cercle initial, ou contiendra L% du cercle initial** (nous reviendrons sur cette définition en utilisant des schémas explicatifs).
- Un type d'obscurcissement, ce qui correspond à la sémantique de la donnée à obscurcir. En attendant des avancées de SOCIETIES dans ce domaine, j'ai choisi d'identifier ces types d'obscurcissement à l'aide d'URI (*Uniform Resource Identifier*), par exemple http://ict-SOCIETIES.eu/data/context/geolocation pour la géolocalisation. C'est la méthode employée dans le domaine du Web Sémantique (28).

5.5.2 Architecture du module d'obscurcissement des données

SOCIETIES a choisi le patron de conception « Inversion de contrôle » pour mettre en œuvre les fonctionnalités des différents modules. Dans ce patron, un module implémente des interfaces décrivant ses fonctionnalités, et les résultats de ses actions sont accessibles via un ou plusieurs éléments implémentant une interface Listener. Ainsi, il est possible d'avertir l'entité utilisant le Listener au moment même où le résultat est disponible, et de personnaliser l'action à effectuer à ce moment là. Mon architecture n'utilise qu'un seul Listener, mais il est possible d'en ajouter plusieurs et donc de rendre facilement accessible à plusieurs entités (implémentant ou utilisant chacune un Listener) le résultat d'une opération.

En respectant cette architecture, j'ai défini trois interfaces permettant d'obscurcir des données :

- **Obfuscator** : qui représente un algorithme d'obscurcissement et qui nécessite d'implémenter la méthode `obfuscateData`.
- **DataObfuscatedAccessor** : qui doit permettre l'accès à une donnée obscurcie et qui nécessite d'implémenter la méthode `getObfuscatedData`.
- **DataObfuscationManagerCallback** : qui est l'interface Listener et qui nécessite d'implémenter la méthode `obfuscationResult` (à appeler en cas de succès) et `cancel` (à appeler en cas d'échec ou d'arrêt).

Le diagramme de composants UML ci-dessous permet de visualiser l'architecture globale que j'ai choisie pour l'obscurcissement de données.

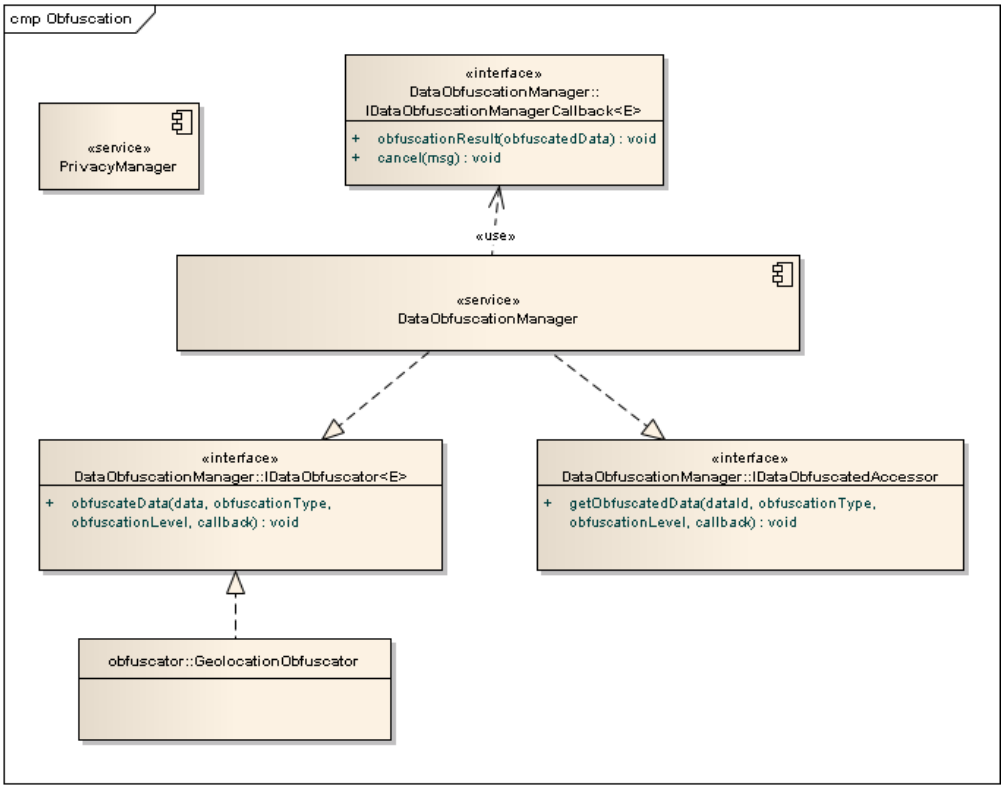


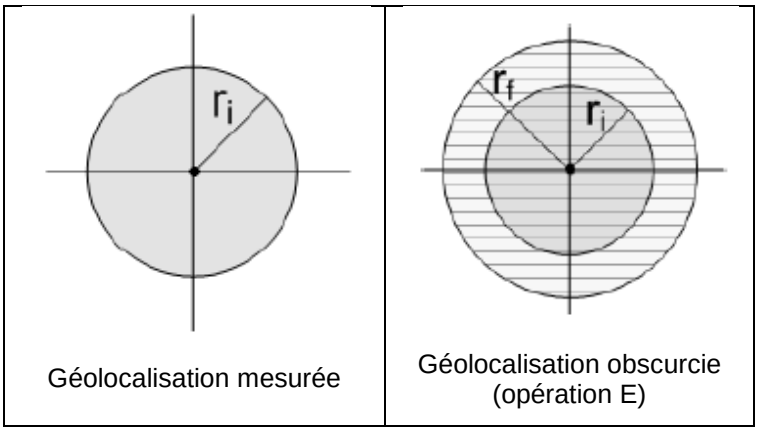
Figure 16 : Architecture globale de l'obscurcissement de données

L'idée générale de cette architecture est que le module DataObfuscationManager est un Obfuscator générique et que sa méthode obfuscateData va déterminer l'Obfuscator spécifique adapté à la situation et exécuter sa méthode obfuscateData.

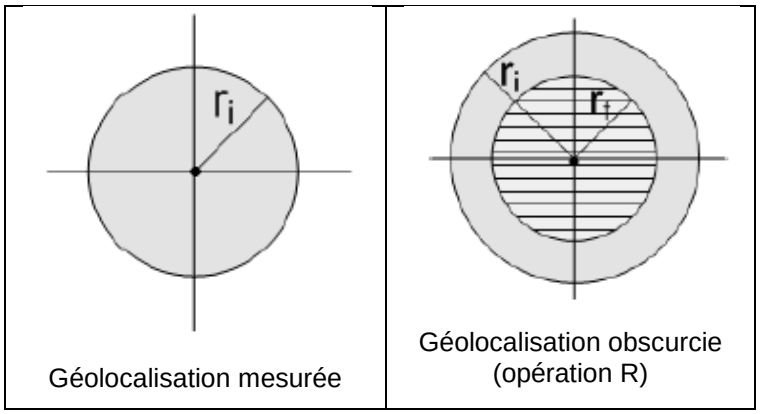
5.5.3 Algorithme d'obscurcissement d'une géolocalisation

L'algorithme d'obscurcissement que j'ai choisi utilise trois opérations d'obscurcissement :

- **Opération E**, pour « Enlargement » : augmentation du rayon du cercle de géolocalisation

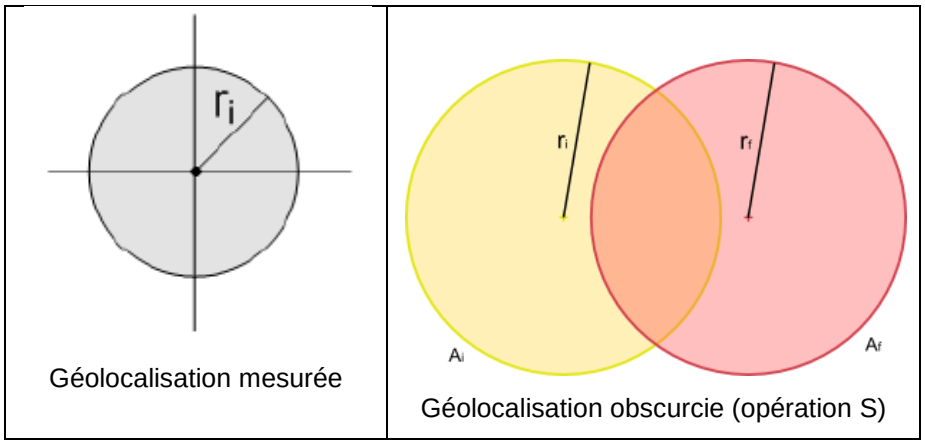


- **Opération R**, pour « Reduce » : diminution du rayon du cercle de géolocalisation



Géolocalisation mesurée | Géolocalisation obscurcie (opération R)

- **Opération S**, pour « Shift » : déplacement du centre du cercle de géolocalisation



Géolocalisation mesurée | Géolocalisation obscurcie (opération S)

Auxquelles s'ajoutent trois combinaisons des opérations précédentes :

- **Opération ES** : une première étape d'augmentation du rayon du cercle de géolocalisation, suivie du déplacement du centre de ce cercle
- **Opération SE** : une première étape de déplacement du centre du cercle de géolocalisation, suivie d'une augmentation du rayon de ce cercle
- **Opération SR** : une première étape de déplacement du centre du cercle de géolocalisation, suivie d'une diminution du rayon de ce cercle
- Il n'y a pas d'opération RS car cela revient au même que l'opération SR (24)

5.5.3.1 Algorithme général

L'algorithme d'obscurcissement sélectionne aléatoirement une opération à effectuer et il l'exécute. On augmente ainsi la résistance de notre algorithme, puisqu'en utilisant toujours la même opération, (par exemple l'augmentation du rayon), on pourrait permettre à un attaquant de se rapprocher de la géolocalisation initiale (en diminuant le rayon). Puisque plusieurs opérations sont possibles, l'attaquant n'a aucun moyen de deviner le sens de l'opération inverse à effectuer.

Cependant, si un attaquant arrive à obtenir, pour une même géolocalisation, le résultat de plusieurs opérations, il pourrait deviner les opérations effectuées et déduire une valeur proche de la géolocalisation mesurée. Par exemple, si un attaquant parvient à obtenir l'obscurcissement de sa géolocalisation par deux opérations, s'il obtient deux cercles l'un dans l'autre, il peut donc deviner qu'une opération E et une opération R ont été utilisées, et donc déduire que le rayon du cercle de géolocalisation non obscurci se situe entre les deux cercles obscurcis. Par conséquent, il faut un moyen de pouvoir réutiliser la même opération pour deux géolocalisations identiques. L'algorithme

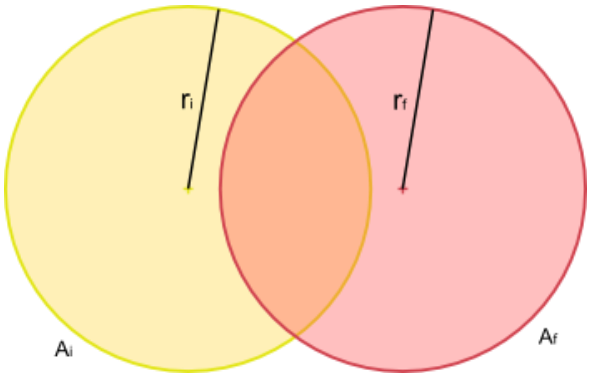
est donc raffiné en conséquence :

- Si une opération est définie en paramètre : l'exécuter. De manière à supprimer toute notion potentielle d'aléatoire dans les algorithmes des opérations, on peut transmettre des paramètres spécifiques à l'opération à exécuter.
- Sinon : choisir une opération de manière aléatoire et l'exécuter.

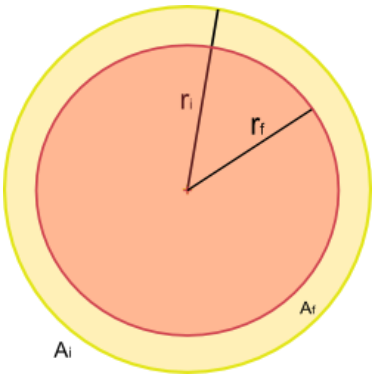
5.5.3.2 Algorithmes de chaque opération

Les algorithmes de ces différentes opérations sont relativement complexes et une étude probabiliste a été nécessaire pour les trouver. Ardagna et al. la présentent sans entrer dans les détails, et je l'ai donc repris depuis le début, d'une part pour bien la comprendre, la vérifier, et d'autre part pour l'adapter à ma formalisation de la notion de niveau d'obscurcissement.

De manière intuitive, si on limite le niveau d'obscurcissement L en le définissant comme le pourcentage du cercle mesuré se trouvant dans le cercle obscurci, on en déduit que L est égale au quotient de l'aire de l'intersection des deux cercles (géolocalisation mesurée et géolocalisation obscurcie), sur l'aire de l'intersection du cercle obscurci.



Si on définit L comme le pourcentage du cercle obscurci se trouvant dans le cercle mesuré (ce qui est le cas lorsque le cercle obscurci est à l'intérieur du cercle mesuré), alors le dénominateur de l'opération précédente est l'aire du cercle mesuré.



Opération E et R

L'algorithme de ces opérations est très simple.

- Opération E : le rayon r_f de la géolocalisation obscurcie est égale à $r_f = \frac{r_i}{\sqrt{L}}$
- Opération R : le rayon r_f de la géolocalisation obscurcie est égale à $r_f = r_i \times \sqrt{L}$

Opération S

L'algorithme de cette opération nécessite la résolution d'un système d'équations à trois inconnus. On peut simplifier ce système car la géolocalisation mesurée et la géolocalisation obscurcie ont la même

précision, donc leur cercle ont le même rayon. Il ne reste plus qu'à résoudre un système d'équations à deux inconnus.

$$\begin{cases} \alpha - \sin(\alpha) = \pi \times obfuscationLevel \\ d = 2 \times accuracy \times \cos\left(\frac{\alpha}{2}\right) \end{cases}$$

Le schéma suivant permet de comprendre le problème et la signification des variables.

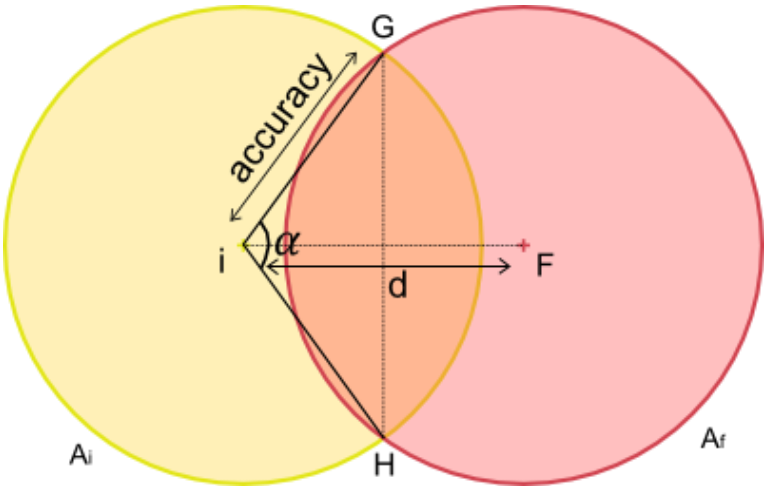


Figure 17 : Opération S, déplacement du centre

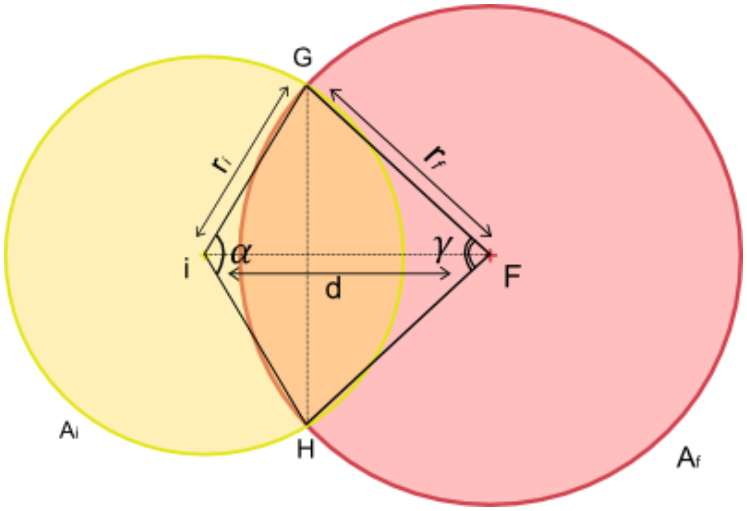
Une fois la distance de translation d déterminée, il faut générer un angle aléatoire permettant de choisir la direction du déplacement.

Opération ES, SE et SR

Aucune simplification n'est possible pour cet algorithme, on doit donc résoudre un système d'équations à trois inconnus.

$$\begin{cases} r_i sin\left(\frac{\alpha}{2}\right) = r_f sin\left(\frac{\gamma}{2}\right) \\ r_i^2(\alpha - \sin(\alpha)) + {r_f}^2(\gamma - \sin(\gamma)) = \left(\pi \times obfuscation\ level \times {r_f}^2\right) \\ d = r_i \cos\left(\frac{\alpha}{2}\right) + r_f \cos\left(\frac{\gamma}{2}\right) \end{cases}$$

Le schéma suivant permet de comprendre le problème et la signification des variables.



Lorsque la distance de translation d, les angles α et γ sont déterminés, il faut générer un angle aléatoire permettant de choisir la direction du déplacement.

Pour l'opération ES, les inconnues sont d, α et γ.

Pour les opérations SE et SR, les inconnues sont r_f, α et γ.

5.6 Conception

5.6.1 Raffinement de l'architecture pour gérer plusieurs algorithmes

Notre architecture doit supporter un algorithme d'obscurcissement pour chaque type de données. Chacun de ces algorithmes devra implémenter l'interface Obfuscator, comme la classe GeolocationObfuscator de l'architecture globale, qui est l'Obfuscator pour les données de type géolocalisation.

De manière à clarifier ce fonctionnement et à réutiliser de bonnes pratique de conception, j'ai mis en place le design-pattern Factory. Le module DataObfuscationManager utilise donc le constructeur DataObfuscatorFactory pour sélectionner l'Obfuscator adapté et l'instancier. Ainsi, il est facile de créer autant d'Obfuscator que nécessaire et tout ajout ou modification nécessite seulement une légère modification du DataObfuscatorFactory, sans toucher au reste du code. Ce raffinement de l'architecture rend donc le code plus facile à comprendre et plus résistant aux changements.

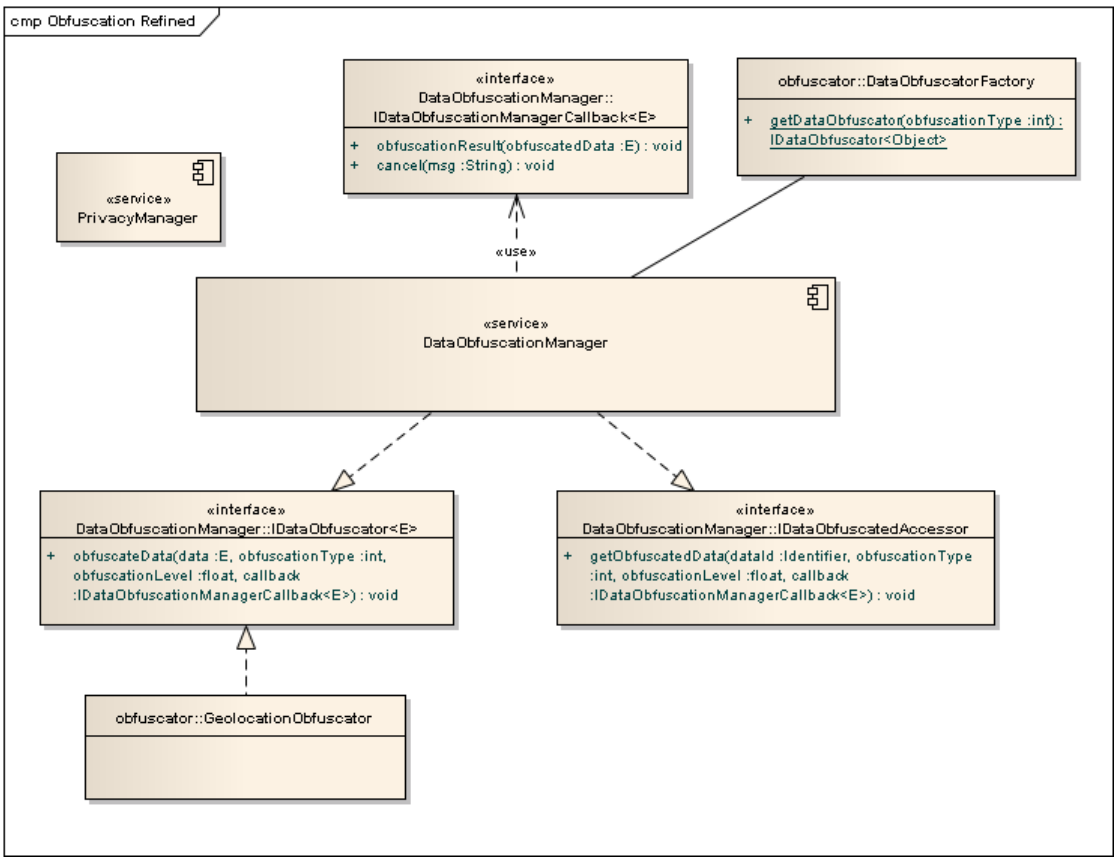


Figure 18 : architecture raffinée de l'obscurcissement des données

5.6.2 Implémentation de l'obscurcissement d'une géolocalisation

La principale difficulté de l'algorithme d'obscurcissement que j'ai décrit, et qu'il nécessite de résoudre des systèmes d'équations en Java pour les opérations S, ES, SE et SR. J'ai donc dû trouver des solutions d'implémentations pour surmonter ces difficultés.

5.6.2.1 *Bibliothèque Java pour résoudre des équations*

Il existe une bibliothèque de Math avancés en Java, créée par Apache (29), que j'ai utilisé quand cela était nécessaire. Elle a l'avantage d'être utilisable aussi sur Android. Cependant, cette bibliothèque est particulièrement difficile à prendre en main, d'autant plus qu'elle nécessite une bonne connaissance d'algorithmes d'approximation mathématiques. Une compréhension plus approfondie m'aurait donc certainement permis de l'utiliser davantage, et de rendre mon prototype plus générique et résistant à de futures améliorations (modification des équations, ajout de cas particuliers, …), et cela aurait certainement pu améliorer ses performances. J'ai donc résolu ces équations par des techniques d'approximations, en utilisant lorsque cela était possible la bibliothèque Commons Math d'Apache.

5.6.2.2 *Approximations successives : méthode de Newton*

La technique d'approximation que j'ai utilisée est la méthode de Newton. Dans cette méthode, on approxime la solution par itérations successives. Par exemple, si l'on souhaite trouver la solution de l'équation f(x)=0, il faut effectuer le calcul suivant un certain nombre de fois :

$$\begin{cases} x_0 \; donné \; pour \; l'intialissation \\ x_n = x_{n-1} - \frac{f(x)}{f'(x)} \end{cases}$$

En choisissant un x_0 relativement proche de la solution finale (ce qui peut être très délicat en pratique comme on le verra), et en arrêtant l'itération lorsque cela n'améliore plus suffisamment la précision de la solution, par exemple lorsque $x_n - x_{n-1} < 10^{-4}$.

Il est possible d'utiliser aussi cette méthode pour des systèmes d'équations à plusieurs variables. Le plus simple est alors d'utiliser une notation matricielle car cela permet de se servir de la bibliothèque Commons Math pour effectuer de manière performante des calculs sur les matrices (inversion, multiplication, etc.).

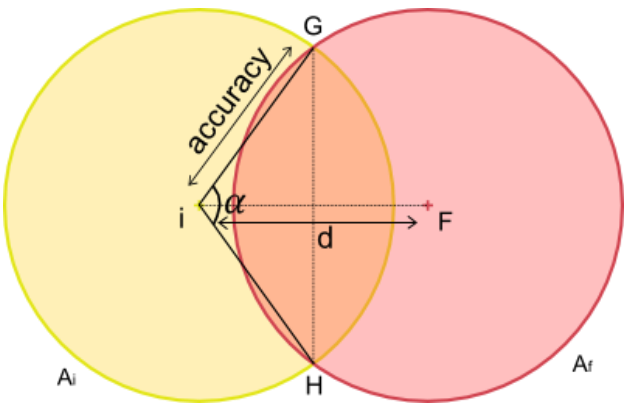
Voyons concrètement les difficultés qui apparaissent sur quelques opérations de l'algorithme d'obscurcissement d'une géolocalisation.

5.6.2.3 *Implémentation de l'opération S*

La difficulté de cette opération S consiste à résoudre l'équation suivante :

$$\alpha - \sin(\alpha) = Constante$$

Il est possible de résoudre cette équation par approximation successive en utilisant la méthode de Newton.



On a donc :

$$f(\alpha) = \alpha - \sin(\alpha) - Constante$$

$$f'(\alpha) = 1 - \cos(\alpha)$$

Et il faut que $\forall \alpha \in]0, 2\pi[, \; f'(\alpha) > 0$, ce qui est le cas car l'intersection des deux cercles n'est pas vide, donc $\alpha \in]0, 2\pi[$.

Comment initialiser l'algorithme

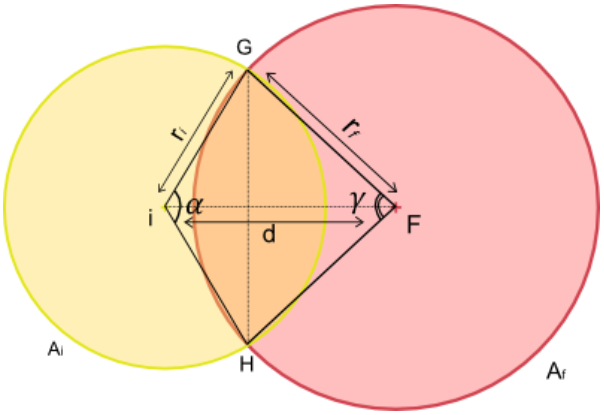
Pour obtenir un résultat rapide et correct (c'est-à-dire pour obtenir la bonne solution si jamais il y en existait plusieurs), il est nécessaire de fournir une initialisation proche de la solution finale. Dans le cas présent, une étude de signe de la fonction f permet de déterminer un intervalle dans laquelle se

trouve la solution, et il apparait que $x_0 = 2$ est une bonne initialisation.

5.6.2.4 *Implémentations des opérations ES, SE et SR*

L'implémentation de ces opérations est plus difficile que la précédente puisqu'il est nécessaire de résoudre un système de trois équations à trois variables, et les inconnues ne sont pas les mêmes selon l'opération.

$$\begin{cases} r_i sin\left(\frac{\alpha}{2}\right) = r_f sin\left(\frac{\gamma}{2}\right) \\ r_i^2(\alpha - \sin(\alpha)) + r_f{}^2(\gamma - \sin(\gamma)) = C \\ d = r_i \cos\left(\frac{\alpha}{2}\right) + r_f \cos\left(\frac{\gamma}{2}\right) \end{cases}$$



Là encore, il est possible d'utiliser la méthode de Newton pour approximer une solution.

Comment initialiser l'algorithme

Initialiser l'algorithme est particulièrement difficile dans ce cas. En effet, il existe plusieurs solutions et trouver un intervalle par une étude de signe n'est pas suffisant pour être sûr de tomber sur la bonne solution.

Par conséquent, l'idée est d'itérer cet algorithme jusqu'à trouver la bonne solution. En effet, des données physiques permettent de déterminer la solution valide (par exemple $\alpha \in]0, 360[$, et $\gamma \in]0, 90°[$ et varie de manière identifiable en fonction de d et α). Il faut donc de commencer par des valeurs d'initialisation données, lancer l'algorithme, vérifier le résultat, et si celui-ci n'est pas correct, re-lancer l'algorithme en augmentant les valeurs d'initialisation.

Pseudo code

```
α0, γ0, iterationMax
Itération tant que (la solution est invalide) {
     Itération tant que (precision>10^-10 ET currentIteration<iterationMax) {
          αn-1,γn-1 prennent les valeurs de αn, γn de la boucle précédente.
          Calcul de αn, γn.
     }
     Incrémentation de α0, γ0
}
```

Ces calculs répétés ont évidement un coût non négligeable, mais c'est actuellement le prix à payer pour que la résolution fonctionne dans tous les cas.

5.7 Résultats de l'implémentation

Au final j'ai pu formaliser la notion d'obscurcissement d'une donnée en me focalisant sur la géolocalisation, spécifier un système d'obscurcissement intégrable à l'architecture de SOCIETIES, et implémenter un prototype de ce système.

Mon prototype se présente sous la forme d'un fichier jar (Java ARchive) implémentant l'architecture précédemment spécifiée. De manière à tester mes résultats, j'ai donc créé une classe Java permettant d'exécuter l'obscurcissement et d'afficher le résultat au format JSON. J'ai pu ainsi tester mon prototype sur un ordinateur de bureau, et j'ai créé une interface Web qui, à partir du résultat, affiche les cercles de géolocalisation sur une carte Google Maps en utilisant l' API Google Maps en Javascript.

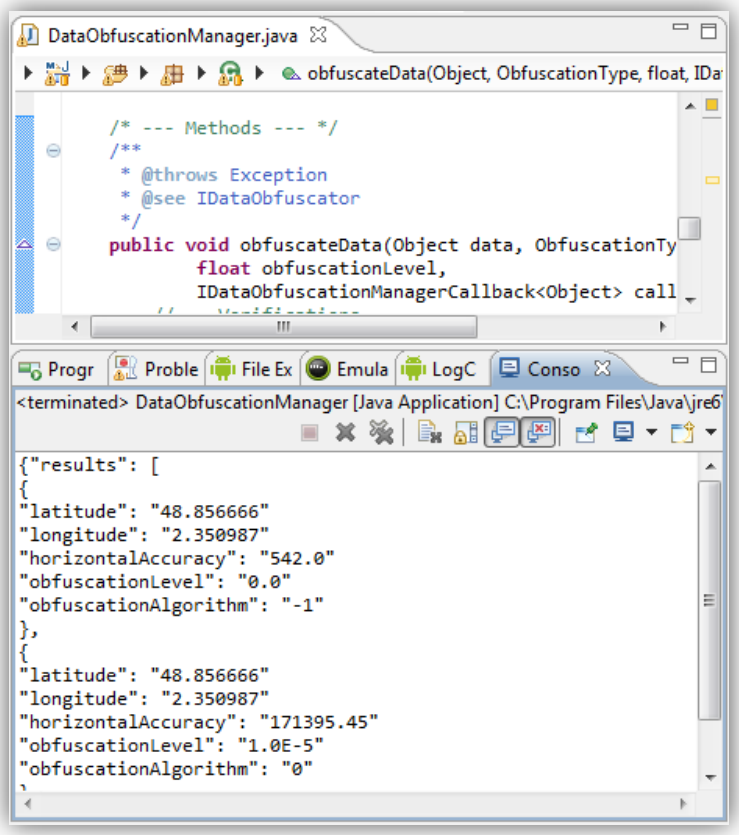


Figure 19 : Exécution de l'obscurcissement

La carte présentée Figure 20 : Interface Web pour visualiser l'obscurcissementFigure 20 permet de visualiser facilement l'obscurcissement d'une géolocalisation. La géolocalisation mesurée apparaît comme un cercle jaune, tandis que la géolocalisation obscurcie apparaît en rouge (et l'intersection des deux en orange).

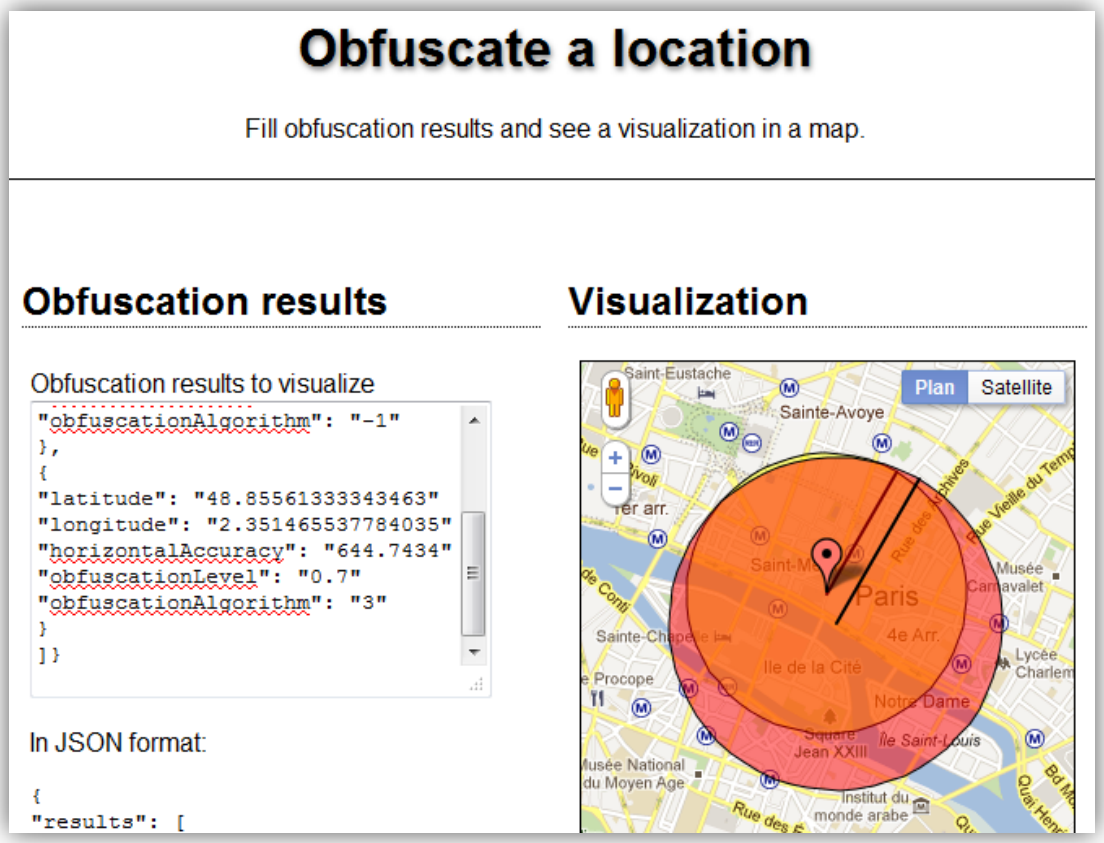


Figure 20 : Interface Web pour visualiser l'obscurcissement, ici 30% d'obscurcissement

Ces différents éléments permettent de tester mon prototype sur un serveur ou un ordinateur de bureau. J'ai donc aussi créé une application Android qui importe le fichier jar de mon prototype, de manière à vérifier si les calculs sont réalisables sur ce support. Un smartphone a l'avantage d'être généralement muni d'une puce GPS. Un bouton « Locate » dans l'application Android permet donc de connaître la géolocalisation actuelle du smartphone, puis de l'obscurcir ce qui permet de faire de tests ancrés dans la réalité.

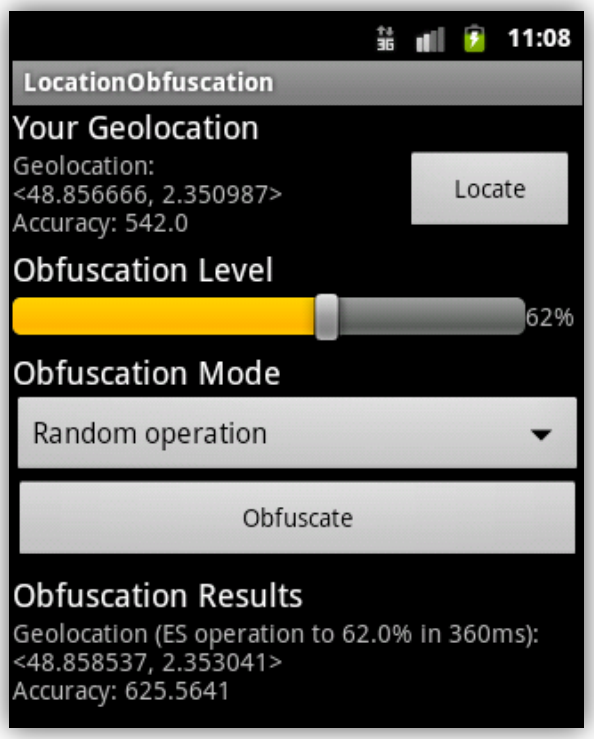


Figure 21 : application Android mettant en œuvre l'obscurcissement

5.8 Validation

Le système que j'ai spécifié correspond aux exigences fixées, et plus particulièrement le prototype implémenté fonctionne sur un ordinateur fixe et sur un smartphone. Certes, certaines opérations sont bien plus consommatrices en temps que d'autres (une opération S durera environ 6ms sur un smartphone, alors qu'une opération SE avec un niveau d'obscurcissement à 0.0001% peut durer 5-10 secondes ou plus, et même jusqu'à 2-3 secondes sur un ordinateur de bureau), mais pour une utilisation classique avec un niveau d'obscurcissement entre 10 et 100, le système a un temps de réponse tout à fait acceptable. Il y a donc des perspectives dans ce domaine, afin d'optimiser l'algorithme d'obscurcissement, notamment la résolution d'équations des opérations ES, SE et SR, mais cette implémentation est utilisable dès aujourd'hui.

D'autre part, ayant adopté une méthode de développement pseudo-agile par itérations successives pour ce module, il serait possible d'effectuer d'autres itérations pour améliorer encore l'algorithme et y ajouter des fonctionnalités des autres méthodes d'obscurcissement existantes pour faciliter son utilisation et son efficacité. Par exemple, au lieu de déplacer le cercle de géolocalisation dans une direction aléatoire, il pourrait être avantageux de la déplacer dans la direction la plus peuplée à proximité en utilisant une bonne de connaissances.

Et bien entendu, l'objet de mon étude s'est limité à la géolocalisation, mais dans SOCIETIES, ce sont toutes sortes de données qu'il sera nécessaire d'obscurcir. L'architecture de l'obscurcissement de données étant maintenant définie, de futures recherches porteront sur l'obscurcissement d'autres types de données.

6 Conclusion

Pour conclure, ces six mois de stage à Trialog auront été riches en expérience et en recherche.

Dans le cadre de la contribution de Trialog au projet SOCIETIES, j'ai eu l'occasion de travailler sur la notion de minimisation des données dans le domaine de la vie privée. Minimiser l'utilisation des données personnelles dans un système s'effectue :

- En aval en insérant la protection de la vie privée dans les exigences de conception de l'architecture. J'en ai étudié un exemple au travers de la notion d'authentification anonyme que j'ai conçue et dont j'ai implémenté un prototype.
- En amont en mettant en œuvre des actions d'obscurcissements des données avant leur partage, comme le permet le module d'obscurcissement de données que j'ai spécifié et implémenté.

J'ai adapté mes recherches à l'architecture du projet SOCIETIES, et j'ai pris en compte les contraintes techniques comme l'adaptation à un système distribué, et les spécificités du développement sur les systèmes mobiles.

Cela m'aura permis de rentrer pleinement dans le domaine de la protection de la vie privée, et d'avoir une vision globale des difficultés posées, des solutions existantes aujourd'hui, et des pistes de recherche en cours. J'ai pu ainsi approfondir et découvrir tout un ensemble de technologies, que ce soit des outils comme Maven ou Git, ou des systèmes comme Android, WebID, SPARQL ou OpenSSL. J'ai aussi apprécié de me heurter à des difficultés lors du passage des algorithmes théoriques à des implémentations, ce qui m'a permis de revoir ou voir des patrons de conception, ainsi que des méthodes d'algorithmique et de calcul particulièrement intéressantes et importantes pour un ingénieur. Et au-delà de cela, d'avoir pu mettre en pratique à plusieurs reprises l'écriture de documents d'état de l'art, de spécifications ou de conceptions, avec l'aide avisée de mon maître de stage, a été un véritable apprentissage concret de ce que je connaissais surtout de manière théorique. Sans compter la collaboration toujours riche d'enseignements avec mes collègues de Trialog, ou avec des partenaires européens durant les réunions plénières et les conférences téléphoniques. Bien souvent, discuter ou exprimer à haute voix un problème que l'on rencontre, permet de trouver plus rapidement une solution qui n'aurait pu être résolue si vite sans cet échange.

Beaucoup de recherches sont encore à effectuer pour améliorer la protection des données personnelles. Notre vie privée étant de plus en plus exposée, et pas seulement dans les réseaux sociaux, les besoins en termes de protection sont grandissants. Petit à petit, des solutions apparaissent pour permettre aux utilisateurs d'accéder à des services même lorsqu'ils ne souhaitent pas partager une partie de leurs données personnelles. Mais au-delà de ces recherches, un des grands défis reste encore de trouver la manière adéquate et ergonomique de proposer ces fonctionnalités à l'utilisateur. Et non seulement pour l'utilisateur final, mais aussi pour le développeur d'applications ! On le voit bien dans SOCIETIES, tous les partenaires sont convaincus de la nécessité de protéger la vie privée, mais quand il s'agit d'ajouter les fonctionnalités de protection à leurs systèmes, ils sont beaucoup moins enthousiastes. Le développeur de système doit donc lui aussi pouvoir comprendre facilement les principes de protection des données personnelles, et posséder des outils faciles à utiliser et efficaces. L'amélioration des techniques de protection de la vie privée, l'ergonomie de l'interface utilisateur et des outils des développeurs, voilà quelques perspectives intéressantes pour la suite de SOCIETIES, et des recherches dans ce domaine.

7 Bibliographie

1. **Trialog.** News Trialog. *Trialog.* [En ligne] [Citation : 2 septembre 2011.] http://www.trialog.com/French/NewsFR.html#erdf.

2. **Electroniques.** Compteurs ERDF communicants : le protocole CPL du projet pilote Linky est signé Trialog et Atos Origin . *Electroniques.* [En ligne] 15 octobre 2009. [Citation : 2 septembre 2011.] http://www.electroniques.biz/editorial/409108/compteurs-erdf-communicantsle-protocole-cpl-du-projet-pilote-linky-est-signe-trialog-et-atos-origin/.

3. —. Compteurs intelligents: première expérimentation enmars 2010. *Electroniques.* [En ligne] 23 avril 2009. [Citation : 2 septembre 2011.] http://www.electroniques.biz/pdf/EIH200904230688012.pdf.

4. **Union Européenne.** *MonAMI.* [En ligne] [Citation : 2 septembre 2011.] http://www.monami.info/.

5. —. Partenaires du projet Societies. *Societies.* [Online] [Cited: septembre 10, 2011.] http://www.ict-societies.eu/partners/.

6. **Wikipédia.** Data Protection Directive. *Wikipédia.* [Online] [Cited: 10 septembre 2011.] http://en.wikipedia.org/wiki/Data_Protection_Directive.

7. **Grand-Duché de Luxembourg.** Viviane Reding présente ses objectifs de révision de la directive sur la protection des données . *Commission Nationale pour la Protection des Données.* [En ligne] 18 mars 2011. [Citation : 10 septembre 2011.] http://www.cnpd.public.lu/fr/actualites/international/2011/03/reding_speech/.

8. **Etat français.** Loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés . *Légifrance.* [En ligne] http://www.legifrance.gouv.fr/affichTexte.do?cidTexte=LEGITEXT000006068624&dateTexte=vig.

9. *The Cost of Reading Privacy Policies.* **M. McDonald, Aleecia and Faith Cranor, Lorrie.** s.l. : A Journal of Law and Policy for the Information Society, 2008.

10. *Privacy-by-Design in ITS Applications.* **Kung, Antonio, Freytag, Johann-Christoph and Kargl, Frank.** s.l. : Second International Workshop on Data Security and PrivAcy in wireless Networks, 2011, p. 6.

11. Sevecom. [Online] http://www.sevecom.org/.

12. **Wu, Thomas.** *SRP-6: Improvements and Refinements of the SRP Protocol.* 2002.

13. —. *The Secure Remote Password Protocol.* 1998. ftp://srp.stanford.edu/pub/srp/srp.ps.

14. **Pioch, Nicolas.** *Protocoles de sécurité.* 2009. SEM - Cours de sécurité.

15. **Feige, Uriel, Fiat, Amos et Shamir, Adi.** *Zero-Knowledge Proofs of Identity.* s.l. : Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot 76100, Israel, 1988.

16. **Raffo, Daniele.** *Digital Certificates and the Feige-Fiat-Shamir zero-knowledge protocol.* 2002. Thèse.

17. **Wikipédia.** Kerckhoffs' principle. *Wikipédia.* [Online] [Cited: 13 septembre 2011.] http://en.wikipedia.org/wiki/Kerckhoffs%27_principle.

18. Attaque par dictionnaire. *Wikipédia.* [En ligne]

http://fr.wikipedia.org/wiki/Attaque_par_dictionnaire.

19. **W3C.** Spécifications WebID. *W3C.* [Online] 2005. http://www.w3.org/2005/Incubator/webid/spec/.

20. —. Wiki WebID. *W3C Wiki.* [Online] http://www.w3.org/wiki/WebID.

21. PHPSecLib. *Sourceforge.* [Online] http://phpseclib.sourceforge.net/.

22. Document OpenSSL PHP. *Manuel PHP.* [En ligne] http://www.php.net/manual/fr/book.openssl.php.

23. Spécification Géolocalisation HTML5. *W3C.* [En ligne] 2011. http://dev.w3.org/geo/api/spec-source.html.

24. *An Obfuscation-based Approach for Protecting Location Privacy.* **A. Ardagna, Claudio, et al.** 2011, Dependable and Secure Computing, IEEE Transactions on, p. 16.

25. *A Formal Model of Obfuscation and Negotiation for Location Privacy.* **Duckham, Matt and Kulik, Lars.** 2005.

26. *Hiding on the Road - Anonymous Usage of Location Based Services.* **Walsh, Alan and Pfeifer, Tom.** 2011.

27. *A Robust Data-obfuscation Approach for Privacy Preservation of Clustered Data.* **Parameswaran, Rupa et M. Blough, Douglas.** 2006.

28. Schema. *Google, Yahoo, Bing.* [Online] http://schema.org/.

29. Apache Commons Math. *Apache.* [Online] http://commons.apache.org/math/.

30. **W3C.** Spécifications SPARQL. *W3C.* [Online] http://www.w3.org/TR/rdf-sparql-query/.

8 Table des illustrations

9 Annexes

9.1 Méthodes de développement sous Android

9.1.1 Définition des critères de comparaison

Commençons par définir des critères de comparaison en fonction des besoins nécessaires dans SOCIETIES.

Critères	Définition
En ligne / Hors ligne	Une application peut être hors ligne (contenue dans le mobile) ou en ligne (non contenue dans le mobile). Dans ce critère, il est donc important de vérifier si une application peut accéder à un service en ligne de manière asynchrone. D'autre part, il faut voir si une application en ligne pourrait être stockée, au moins partiellement, hors ligne.
Exécution périodique	L'exécution périodique est l'un des trois modes d'utilisation d'une application. Ce mode permet, alors que l'application n'est pas lancée, d'exécuter un processus périodiquement.
Géolocalisation	La géolocalisation permet de localiser le mobile à l'aide de coordonnées latitude / longitude. Via GPS, ou en prenant les coordonnées GPS de la balise GSM la plus proche ou autre. La précision varie selon la technique utilisée et la qualité du matériel.
Graphisme avancé	Pouvoir dessiner des formes, un calendrier par semaine ou par mois, … nécessite des capacités graphiques avancés plus ou moins difficile à mettre en œuvre.
Lecture / écriture de tags RFID	La technologie RFID est une méthode efficace et peu coûteuse pour interagir avec l'environnement. Avoir la capacité de lire et écrire dans un tag RFID peut donc s'avérer utile dans un espace ambiant ou pour notifier rapidement un serveur de sa présence.
Multiplateforme	Une méthode multiplateforme permet de développer une seule fois l'applicatio, et de pouvoir la déployer ensuite sur Android, iPhone, Blackberry, … mais aussi smartphone, tablette, PC, TV, … avec un minimum d'adaptation.
Notification	La notification est l'un des trois modes d'utilisations d'une application. Elle permet de fournir des alertes à l'utilisateur sous la forme d'icônes apparaissant dans la barre de statut (avec signal sonore et vibration si on le souhaite), et ceci quelle que soit l'application en cours.

	 *Notification de l'arrivée d'un email dans le barre de status par une icône « @ ».* *Ouverture de la notification pour consulter les détails.* La mode Notification peut être utilisé pour signaler tout type d'événement à l'utilisateur et pourra donc être mis en œuvre dans les applications mobiles visées afin d'avertir l'utilisateur de l'occurrence d'événements auxquels il doit réagir.
Push	Une méthode de Push permet de demander au serveur de nous prévenir lorsqu'une information demandée est disponible ou qu'un événement s'est produit. Cela évite d'exécuter régulièrement et intensivement des requêtes http vers le serveur. Il serait utile qu'un Push puisse réveiller une Activity d'Android qui elle-même s'affichera (à éviter) ou affichera une notification si besoin.
Requête HTTP	La possibilité d'utiliser des requêtes HTTP permet d'appeler des services en ligne hébergés sur un serveur. C'est le moyen le plus simple pour utiliser des services. Néanmoins, il est préférable : • de lancer ces appels dans un Thread de manière à ne pas ralentir l'application, • de limiter le nombre de requêtes au maximum pour éviter une consommation excessive de la batterie • d'éviter le *pooling* intensif, c'est-à-dire de lancer régulièrement une requête HTTP pour attendre un changement d'état du serveur. Il est préférable d'utiliser une méthode de Push dans ce cas.

9.1.2 Comparaison des méthodes de développement

Continuons en comparant ensuite les trois principales méthodes de développement sous Android.

Caractéristiques	SDK Android	HTML5	Titanium
Autre		Technologie amenée à durer mais non totalement spécifiée et diversement implémentée.	Rend dépendant de Titanium et de ses évolutions. La rétrocompatibilité ne semble pas être au point.
En ligne / Hors	L'application est **hors ligne et**	L'application peut être **hors**	L'application est **hors ligne et**

ligne	**peut accéder au réseau**. Des données peuvent être stockées sur la machine.	**ligne** (grâce à HTML5 ou à PhoneGap par exemple. Le fonctionnement est alors similaire à Titanium) **ou en ligne** (on consulte un site Web). Stockage des données : • Local ou session storage : fonctionne parfaitement. Limité à 5mo. • Web SQL Database : disponible • IndexedDB : non disponible mais futur standard. • Cache : disponible (surtout utile pour une application hors ligne) Question en suspend : comment faire d'une webapp « en ligne », une application avec une icône ouvrable à partir du panneau des applications ?	**peut accéder au réseau**. Des données peuvent être stockées sur la machine.
Exécution périodique	**Disponible** Avec AlarmManager il est possible de prévoir, périodiquement ou non, de lancer un service ou une activité. L'élément lancé peut donc aussi démarrer une notification.	**Non disponible** Une webapp ne peut paramétrer une exécution future et il y a peu de chance que cela soit possible un jour. Une webapp peut cependant être lancée par un AlarmManager d'Android.	**Non disponible** Idem qu'HTML5.
Géolocalisation	**Disponible** Peut être lancé comme un service. Utilise une puce GPS ou à défaut une localisation IP, borne Wifi et balise GSM.	**Disponible** Mais la précision varie selon l'endroit. Utilise un service de localisation basé sur l'adresse IP (on a associé les adresses IP à une position) et les points Wifi environnants. Firefox utilise par exemple Google Location. Il est probable que dans un futur proche, les navigateurs puissent utiliser les puces GPS des téléphones.	**Disponible** Non testé. Une API de localisation et de cartographie est disponible.
Graphisme avancé	Une interface graphique permet d'utiliser des formes	Grandes possibilités avec CSS et notamment CSS3, de mieux	Plusieurs formes sont prédéfinies.

	prédéfinies. Des formes plus spécifiques, comme un calendrier, peuvent être dessiné en codant directement. Si besoin, il est possible d'utiliser Canvas.	en mieux supportés. Il faut redéfinir les formes soit même, mais de nombreuses ressources existent sur le Web. De plus, jQuery Mobile permet de dessiner des boutons, des sliders, … à la mode Android, iPhone, … Canvas fonctionne et des outils avancés si besoin.	Peut être lent.
Lecture / écriture de tags RFID	**Possible** Dépend du matériel du téléphone.	? Il est probable qu'une API soit un jour disponible.	?
Multiplateforme	**Non, ou presque** Mais il existe, ou existera, des systèmes permettant de porter une application Android sur iPhone, …	**Oui, à quelques adaptations près.** Cependant, tous les appareils n'implémentent pas tout, et pas forcément de la même manière. On retrouve les contraintes du Web classique.	**Oui à quelques adaptations près**. Particulièrement Android et iPhone.
Notification	**Disponible** Il est possible d'afficher une icône suivie d'un texte défilant. Un court message pour en savoir plus sur la notification, ainsi qu'une Activity de destination.	**Non disponible** Il est possible en HTML5 de créer une notification lorsque l'application est lancée. Mais cela ne fonctionne pas encore sous Android.	**Disponible**
Push	**Disponible a**vec C2DM de Google. Le push de C2DM permet, quand le serveur a une information à fournir, de démarrer une Activity ciblée (même si l'application qui s'est enregistré à C2DM est fermée) cette Activity pouvant afficher une notification. Cela a l'inconvénient de nécessiter un compte Gmail (qui peut cependant être le même pour tous les utilisateurs de l'application) D'autres technologies existent, notamment XMPP.	**Disponible avec quelques efforts** En utilisant les WebSocket. Ils ne sont pas encore supporté par le navigateur d'Android, ni PhoneGap, mais des librairies permettent de les utiliser. Les tests que j'ai réalisés sont prometteurs et quasi fonctionnels. L'avantage est que l'on contrôle parfaitement la plateforme de Push. D'autres technologies existent, notamment XMPP.	*?*
Requête HTTP	**Disponible** REST est préféré à SOAP. Le parsing du résultat est facile en XML ou en JSON. Une librairie GSon facilite notamment l'interaction avec JSON.	**Disponible** Plusieurs méthodes sont possibles. • Accès immédiat pour passer à la page suivante (fonctionnement classique	**Disponible**

		d'HTML) • Ajax XMLHttprequest : disponible • Websocket : disponible Le parsing du résultat est très facile en XML ou en JSON, surtout avec jQuery.	

9.2 Authentification anonyme

9.2.1 Protocole de communication

Dans le cas d'une utilisation avec le protocole HTTP, l'exécution d'une requête applicative se décompose en deux étapes : l'envoi de la requête POST par le client, puis la réception de la réponse émise par le serveur. Le contenu de ces deux étapes est décrit ci-dessous :

Requête POST envoyé par le client	

En-tête de la requête	`POST <URL relatif à appeler> HTTP/1.1`
	`Host: <URL de l'hôte sur lequel se basera l'URL relatif>`
	`Content-Length: <taille du corps de la requête en octet>`
	`Content-Type: application/json; charset=UTF-8`
	`Content-Encoding: UTF-8`
	`Connection: Keep-Alive`
	`Accept: text/html`
	`User-Agent: <Ex : Apache-HttpClient/4.1 (java 1.5)>`
	`*** Ligne vide ***`

Corps de la requête	`{`
	`commande:'<numéro de la commande>',`
	`data:'<données structurées au format text/plain ou plus généralement au format application/json>'`
	`}`

Réponse retournée par le serveur	

En-tête de la réponse	`HTTP/1.1 200 OK`
	`Date: <Format RFC 1123. Ex : Mon, 24 Jan 2011 10:01:43 GMT>`
	`Server: <Ex : Apache/2.2.16 (Unix) mod_ssl/2.2.16 OpenSSL/0.9.8>`
	`Content-Type: text/html`
	`Content-Length: <`*`taille du corps de la réponse en octet`*`>`
	`*** Ligne vide ***`

Corps de la réponse	Données structurées au format text/plain ou plus généralement au format application/json.

Pour mieux comprendre, voici un exemple de requête / réponse avec la requête « authenticate » exécutée par un CSS et réceptionnée par une autorité de confiance. Elle permet à l'utilisateur de s'authentifier auprès de l'autorité de confiance, en lui demandant de vérifier et de se porter garant de son appartenance à une communauté donnée. Ainsi, l'utilisateur se voit délivrer deux certificats d'authentification qui lui permettront de prouver son identité et/ou son appartenance à la

communauté donnée.

Paramètres de la requête

Libellé	Type	Description
typeVerification	Integer	Vérifications à effectuer 1 = ID by WebID (default) 2 = ID & membership by WebID 3 = ID by login/ password 4 = ID & membership by login/password
certificate	String	Certificat X.509 au format PEM associé au WebID de l'utilisateur. *Si typeVerification=1 ou 2*
WebIDCIS	String	URI du WebID du CIS auquel l'utilisateur souhaite s'authentifier *Si typeVerification= 2*
nonce	BigInteger	Variable aléatoire à ajouter au checksum *Si typeVerification= 2*
Login	String	Login de l'utilisateur *Si typeVerification= 3 ou 4*
Pwd	String	Mot de passe de l'utilisateur *Si typeVerification= 3 ou 4*

Exemple de requête : vérification de l'identité et de l'appartenance à la communauté avec WebID

```
{
commande:'2',
data:{
     typeVerification:'2',
     certificate:'-----BEGIN CERTIFICATE-----
     MIIDODCCAvagAwIBAgIERqplETALBgcqhkjOOAQDBQAwfzELMAkGA1UE...
     [...]
     Cgfs2kXj/IQCFDC5GT5IrLTIFxAyPUo1tJo2DPkK
     -----END CERTIFICATE-----',
     webIDCIS:'https://trialog.com/SOCIETIES/#community',
     nonce:'46687874987946848[...]4987486489746546'
}
}
```

Réponse du serveur

Libellé	Type	Description
Response	Integer	1 en cas de succès, code d'erreur sinon
identityPassport	IdentityPassport	Passeport prouvant que l'identité de l'utilisateur a été vérifiée

membershipPassport	MembershipPassport	Passeport prouvant que l'appartenance du membre à la communauté à été vérifiée

Exemple de réponse :

```
{
response:'1',
identityPassport:{
     type:1,
     webid:'https://name.me/#me',
     date:'Mon, 24 Jan 2011 10:01:43 GMT',
     hmac:'AFBA34A2A11AB13EEBA5D0A7AA22BBB6120E177B'
},
membershipPassport:{
     type:2,
     webid:'https://trialog.com/SOCIETIES/#community',
     date:'Mon, 24 Jan 2011 10:01:43 GMT',
     hmac:'9BEF2485410E9378BC9ADFB3E32236AF4F683FA2'
}
}
```

9.2.2 Gestion des données

9.2.2.1 Certificat X.509

Un certificat X.509 peut être stocké sous plusieurs formats. Ci-dessous au format PEM, un format courant pour le stockage car peu coûteux en place.

```
-----BEGIN CERTIFICATE-----
MIIDODCCAvagAwIBAgIERqplETALBgcqhkjOOAQDBQAwfzELMAkGA1UE...
...
Cgfs2kXj/IQCFDC5GT5IrLTIFxAyPUo1tJo2DPkK
-----END CERTIFICATE-----
```

Le format PEM est généré à partir d'un certificat X.509 au format CERT. Pour un certificat associé à un fichier WebID, le champ « X509v3 Subject Alternative Name » contient l'URI du WebID. Ci-dessous un exemple de certificat X.509 au format CERT avec en rouge les informations spécifiques au lien avec un fichier WebID.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5f:df:d6:be:2c:73:c1:fb:aa:2a:2d:23:a6:91:3b:5c
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: O=FOAF+SSL, OU=The Community of Self Signers, CN=Not a
Certification Authority
        Validity
            Not Before: Jun  8 14:16:14 2010 GMT
            Not After : Jun  8 16:16:14 2010 GMT
        Subject: O=FOAF+SSL, OU=The Community Of Self
Signers/UID=https://example.org/profile#me, CN=Robert (Personal)
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:cb:24:ed:85:d6:4d:79:4b:69:c7:01:c1:86:ac:
                    [...]
                    91:a1
                Exponent: 65537 (0x10001)
        X509v3 extensions:
```

```
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Key Usage: critical
                Digital Signature, Non Repudiation, Key Encipherment, Key
Agreement, Certificate Sign
            Netscape Cert Type:
                SSL Client, S/MIME
            X509v3 Subject Key Identifier:
                08:8E:A5:5B:AE:5D:C3:8B:00:B7:30:62:65:2A:5A:F5:D2:E9:00:FA
            X509v3 Subject Alternative Name: critical
                URI:https://robert.me/#me
    Signature Algorithm: sha1WithRSAEncryption
        cf:8c:f8:7b:b2:af:63:f0:0e:dc:64:22:e5:8a:ba:03:1e:f1:
        [...]
        d8:b8
```

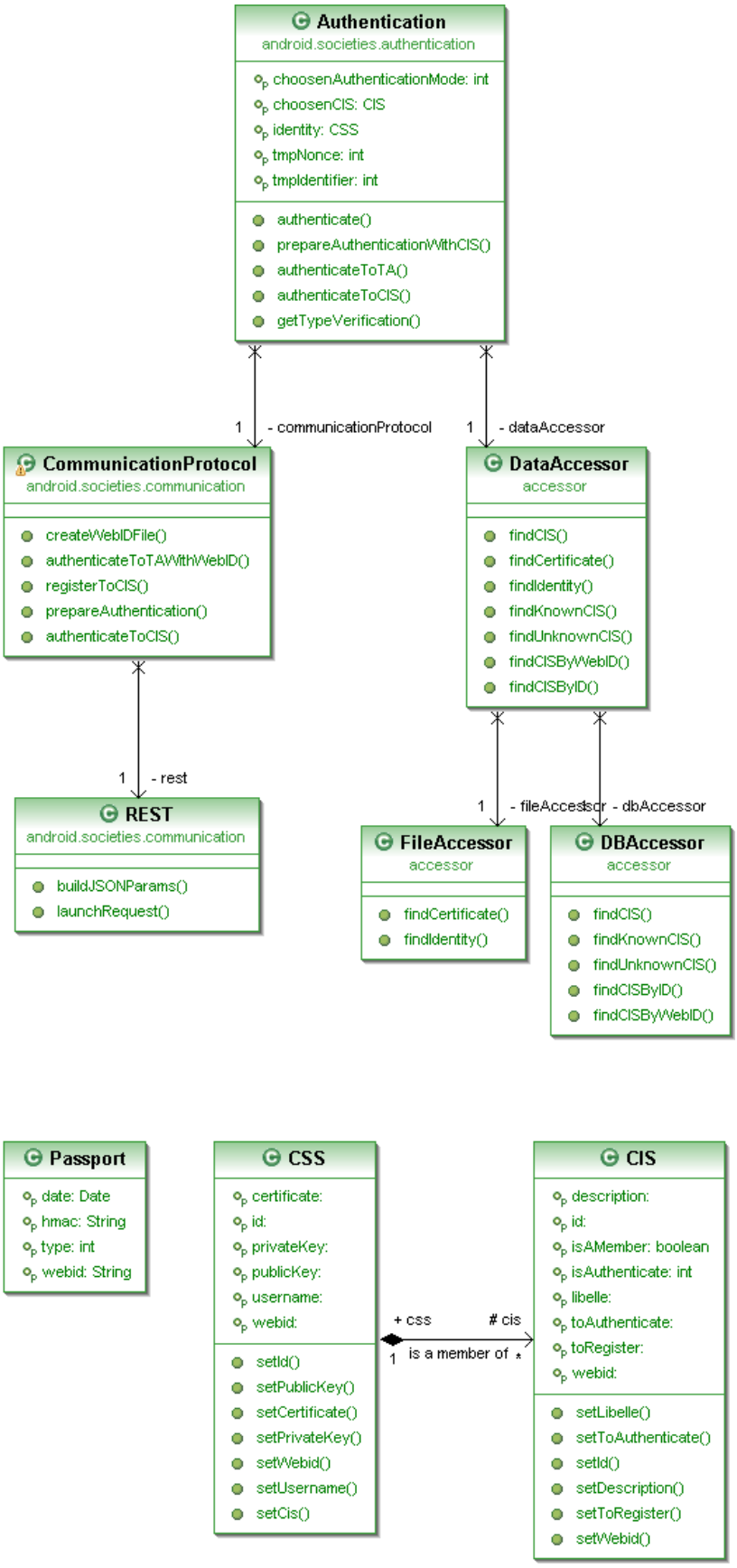
9.2.2.2 WebID

Exemple de fichier WebID pour l'utilisateur « Username ».

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:cert="http://www.w3.org/ns/auth/cert#"
 xmlns:rsa="http://www.w3.org/ns/auth/rsa#"
 xmlns:foaf="http://xmlns.com/foaf/0.1/">
    <foaf:PersonalProfileDocument rdf:about="">
          <foaf:maker rdf:resource="#me"/>
          <foaf:primaryTopic rdf:resource="#me"/>
    </foaf:PersonalProfileDocument>
    <foaf:Person rdf:ID="me">
          <foaf:nick>Username</foaf:nick>
    </foaf:Person>
    <rsa:RSAPublicKey>
          <cert:identity rdf:resource="#me" />
          <rsa:public_exponent cert:decimal="65537" />
          <rsa:modulus
cert:hex="9c0132cdd8614ca5c82b9650f61b24000fd69afd5b1b4f1daaeb5a4bff14c09a7
9c4c5a798fa779c1a17152b29e82c6ef4dcaecc3d73804d014b8bc0f7c3e17ed0457335a03d
f50a0b33ef1b0d0bbcfddc19c1d66972816d1ecdef887fbcb40c0ad63349d358ca1955ddc73
69913f81c74baefafa7bd5bdafbbc510676502da6ec8401b156a6c5902b74de04d62f5799fd
2bc20612dd204546b0d0c83f7bf828ad6e53c9a21c3f2438d20a7562f95341dee3adb5ea610
2a0fdb1953343791222f38f29f9c689cd1fa3f7d16eb215f447688e0d77bc7ecb3c27c5ca53
b2fe3fcf775a1e43b6199c7cf7607930d7be05568bef8d246c3ff86b70b7157c45de30b5"
/>
    </rsa:RSAPublicKey>
</rdf:RDF>
```

9.2.3 Diagramme UML de classes du CSS Android

Avec le plugin Eclipse eUML2 j'ai créé un diagramme de classes UML du CSS Android, ce qui m'a permis ensuite de générer le squelette Java de mon code. Voici ce diagramme, légèrement allégé de classes purement Android.


Authentication
android.societies.authentication
choosenAuthenticationMode: int
choosenCIS: CIS
identity: CSS
tmpNonce: int
tmpIdentifier: int
authenticate()
prepareAuthenticationWithCIS()
authenticateToTA()
authenticateToCIS()
getTypeVerification()
1
- communicationProtocol
1
- dataAccessor
CommunicationProtocol
android.societies.communication
createWebIDFile()
authenticateToTAWithWebID()
registerToCIS()
prepareAuthentication()
authenticateToCIS()
DataAccessor
accessor
findCIS()
findCertificate()
findIdentity()
findKnownCIS()
findUnknownCIS()
findCISByWebID()
findCISByID()
1
- rest
REST
android.societies.communication
buildJSONParams()
launchRequest()
1
- fileAccessor
- dbAccessor
FileAccessor
accessor
findCertificate()
findIdentity()
DBAccessor
accessor
findCIS()
findKnownCIS()
findUnknownCIS()
findCISByID()
findCISByWebID()
Passport
date: Date
hmac: String
type: int
webid: String
CSS
certificate:
id:
privateKey:
publicKey:
username:
webid:
setId()
setPublicKey()
setCertificate()
setPrivateKey()
setWebid()
setUsername()
setCis()
+ css
# cis
1
is a member of
*
CIS
description:
id:
isAMember: boolean
isAuthenticate: int
libelle:
toAuthenticate:
toRegister:
webid:
setLibelle()
setToAuthenticate()
setId()
setDescription()
setToRegister()
setWebid()

10 Glossaire

Mot	Explication
API	Signifie *Application Programming Interface* et désigne une interface facilitant l'utilisation d'un système pour un développeur. Très souvent, une API n'offre qu'une interface d'utilisation mais pas l'implémentation qui y est associée, elle est alors abstraite. Lorsque des implémentations d'une API existent pour plusieurs systèmes, il devient possible d'utiliser de la même manière tous ces systèmes pourtant bien différents.
Certificat X.509	Certificat de sécurité à clé asymétrique permettant de transporter la clé publique en prouvant que l'on détient la clé privée associée.
CIS	*Community Interaction Space*, aussi appelé « Communauté ». Ce terme du projet SOCIETIES désigne un élément créé par un CSS qui lui a associé des données et des services. Un CIS est associé à plusieurs CSS qui sont ses membres.
CSS	Un terme du projet SOCIETIES signifiant *Cooperating Smart Space*, ce qui est à la fois le représentant de l'utilisateur, ainsi que le moyen possédé par celui-ci pour interagir avec des communautés. Un CSS est constitué d'une ou plusieurs CSS Node c'est-à-dire des interfaces différentes (*smartphone*, ordinateur, …). Un CSS peut faire partie d'une ou plusieurs communautés dont il est alors membre.
Opt-in	Etendu du vocabulaire marketing, le terme d'opt-in s'emploie lorsque l'utilisateur doit explicitement donner son accord à l'utilisation de ses données personnelles.
Opt-out	Etendu du vocabulaire marketing, le terme d'opt-out s'emploie lorsque l'utilisateur donne son accord de manière tacite à l'utilisation de ses données personnelles.
SOCIETIES	Projet européen signifiant : *"Self Orchestrating Community ambIEnT intelligEnce Space".*
TA	*Trusted Authority* ou autorité de confiance pour désigner l'élément capable d'authentifier un utilisateur et de lui fournir des certificats lui permettant de prouver son identité à d'autres éléments.