Master 1 Sécurité des systèmes informatiques

Rapport de projet : Chat sécurisé

Rédigé par	Charles Ango, Ismael Kabore, Julien Legras, Yves Nouafo, Jean-Baptiste Souchal
Relu par	Julien Legras
À l'attention de	Magali Bardet
Date de rendu	24 mai 2013

Introduction

Les outils de messagerie instantanée sont de nos jours largement utilisés par les particuliers et entreprises. Ces communications donnent parfois lieu à des échanges de données confidentielles. Or, la sécurité des données et l'authentification des utilisateurs n'y sont généralement pas assurées. Il est donc important d'avoir à disposition des outils permettant d'assurer la confidentialité d'une communication. C'est donc dans le cadre de notre projet annuel de première année de master en sécurité des systèmes informatiques que nous avons étudié les protocoles cryptographiques permettant à plusieurs utilisateurs de s'authentifier et de communiquer de manière sécurisée à travers un outil de messagerie instantanée assurant la confidentialité et l'intégrité des données échangées.

La couche de sécurité mise en place sur un outil de messagerie instantanée va permettre de dialoguer avec des utilisateurs authentifiés, d'assurer l'intégrité des messages reçus, c'est-à-dire que le message n'a pas été modifié entre deux interlocuteurs puis une confidentialité assurée des contenus échangés grâce à des outils cryptographiques forts mis en place empêchant toute intelligibilité des données transitant entre utilisateurs pour quelqu'un de non autorisé.

Dans un premier temps nous allons faire une présentation plus détaillée du sujet en analysant les besoins du client puis nous présenterons les solutions proposées. Nous détaillerons ensuite les résultats obtenus après l'implémentation des solutions proposées en amont et leur pertinence par rapport aux besoins du client. Finalement un manuel d'utilisation viendra compléter ce rapport afin d'assurer une bonne prise en main de l'outil de messagerie instantanée par le client.

Table des matières

Chapitre 1

Présentation du projet

Dans le cadre de notre projet, il nous à été demandé de réaliser un système de bavardage sécurisé. Pour le mener à bien, le travail à été découpé en deux grandes phases qui sont l'analyse des besoins du client et les solutions proposées.

1.1 Besoins du client

Les fonctionnalités du logiciel sont les points indispensables, nécessaires et imposés par le client. Dans notre sujet, on le résume de la manière suivante :

Étudier les protocoles cryptographiques permettant à plusieurs utilisateurs de s'authentifier et de communiquer de manière sécurisée à travers un outil de messagerie instantanée répondant au standards IRC.

À la fin du développement, l'outil de messagerie instantanée devra répondre aux besoins du client, garantir que les messages soient échangés entre utilisateurs, prouver la mise en place du système de sécurité ainsi que les protocoles déployés pour parvenir à un système de bavardage sûr.

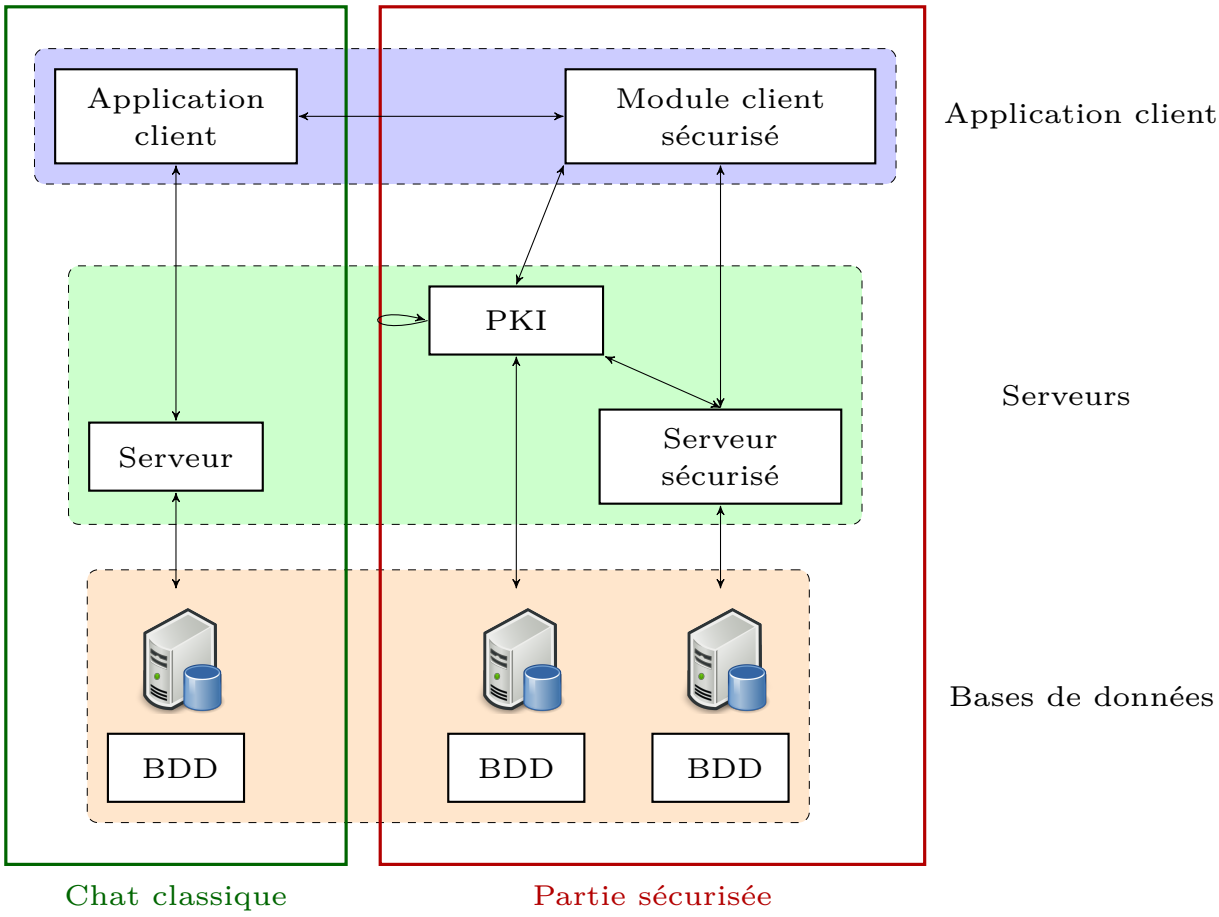
1.2 Solutions proposées

À l'issue de l'analyse des besoins du client, il a été convenu de traiter sa demande en décomposant la réalisation du logiciel en trois phases de développement :

- **Phase 1 :** Mise en place du système de bavardage modèle client-serveur ;
- **Phase 2 :** Installation d'une PKI (infrastructure à clés publiques) ;
- **Phase 3 :** Mise en place d'une couche sécurisée pour permettre une communication chiffrée, authentifiée et d'en garantir l'intégrité.

1.2.1 Structure du logiciel

Le logiciel est conçu et pensé de manière modulable de tel sorte que chaque partie puisse être utilisée de manière indépendante. Son architecture se présente ainsi :



Sur le schéma précédent on voit bien la séparation de la partie couche classique de la partie couche sécurisée, mais aussi les différentes couches qui interviennent dans le développement du logiciel. Ces couches sont la partie client, la partie serveurs et la partie base de données.

La partie Application client comporte deux modules qui sont l'application client et le module client sécurisé. L'application client permet à l'utilisateur de pouvoir communiquer avec d'autres utilisateurs de manière classique. Autrement dit, chaque message pourra être lu par tout utilisateur présent dans un salon.
Quant au module client sécurisé, il permet d'établir une communication sécurisée entre plusieurs utilisateurs. Une communication sécurisée masque le message à tout utilisateur n'étant pas destinataire du message en question.
Cette partie implémente les principales fonctionnalités suivantes :

- création d'un compte sécurisé ;
- envoi de message sur salon privé ;
- envoi de message sur salon général ;
- rejoindre un serveur de chat sans authentification ;
- quitter un serveur de chat ;

- envoi d'un message privé non sécurisé ;
- créer un salon privé ;
- joindre un salon privé ;
- fermeture d'un salon privé.

La partie serveurs est décomposée en trois éléments qui sont le serveur, le serveur sécurisé et la PKI. Chacun de ces éléments ont un rôle bien défini.
Le rôle du serveur est de distribuer les messages aux destinataires.
Le rôle du serveur sécurisé est d'authentifier les utilisateurs sécurisés et de distribuer les clés symétriques permettant le chiffrement, le déchiffrement et l'intégrité des messages entre utilisateurs sécurisés.
La PKI est une autorité de confiance qui permet d'établir l'authenticité d'un utilisateur sécurisé.
Nous avons opté pour une PKI clé en main EJBCA qui remplira les fonctionnalités suivantes :

- ajout d'un certificat à un utilisateur ;
- révocation d'un certificat.

La partie base de données décrit les données manipulées et utiles à l'application. L'application globale en possède trois.
La première permet la gestion des utilisateurs non sécurisés. Cette base de données est rattachée au serveur non sécurisé. Elle est non persistante, ce qui veut dire que lors de la fin de l'utilisation du logiciel par un utilisateur, celui-ci ne sera pas conservé en base. Cette base à pour rôle de lister tout utilisateur déjà présent sur le service de chat.
La seconde permet la gestion des utilisateurs sécurisés. Contrairement à la première, cette base est persistante, ce qui permet à tout utilisateur sécurisé de réserver son nom et de pouvoir l'utiliser pour les prochaines connections à venir sur le service.
La dernière permet le stockage des certificats générés par la PKI. Ces certifcats définissent l'identité d'un utilisateur et sont stockés dans une base de données interne à l'application EJBCA.

1.2.2 Détail de la couche sécurisée

La couche sécurisée contient le module sécurisé, la PKI, le serveur sécurisé et la base de données qui lui est jointe. Cette couche est nécessaire pour échanger et garantir l'échange des messages sécurisés sans que celui-ci ne puisse être lu par un autre utilisateur. Pour mettre en place cette partie, on utilise la bibliothèque OpenSSL pour assurer la création d'un canal de communication sécurisé entre deux utilisateurs sécurisés.
Comme dit ci-dessus, la PKI, autorité de confiance, permettra de garantir et d'identifier les utilisateurs présents sur le système de chat. On a aussi vu précédemment quel rôle joue le serveur sécurisé. Toute la mise en œuvre ainsi décrite permet de garantir un système de chat respectant les normes et les exigences du client.

Pour s'assurer que l'application fournie respecte bien les exigences du client, elle implémente des exigences fonctionnelles que l'on peut retrouver dans le cahier de recette, où on peut voir les différents tests garantissant la mise en place de la couche sécurisée.

1.2.3 Choix des langages

Le choix des langages pour réaliser le projet sont le C pour la réalisation du serveur non sécurisé et du serveur sécurisé ainsi que des clients, sécurisé et non sécurisés.
La réalisation de la base de données embarquée s'est faite en sqlite.
La partie graphique quant à elle a été faite en Vala, car ce langage s'interface avec le C très facilement.
Pour la partie sécurisée on a choisi la bibliothèque OpensSSL car elle fourni une bibliothèque riche en fonctions cryptographiques.

1.3 Partie technique

Le résultat a été réparti en trois livraisons dont voici les contenus.

1.3.1 Partie non-sécurisée

La première partie consistait en le développement d'un client et d'un serveur de chat. Pour la communication entre les clients et le serveur, une structure C a été mise en place :

```
typedef struct {
        int code;
        char sender[MAX_NAME_SIZE];          // MAX_NAME_SIZE = 64
        char content[MAX_MESS_SIZE];         // MAX_MESS_SIZE = 512
        char receiver[MAX_ROOM_NAME_SIZE];   // MAX_ROOM_NAME_SIZE = 64
} message;
```

Chaque action est représentée par un code (`CONNECT, CREATE_ROOM, MESSAGE...`). Le champ `sender` représente l'émetteur du message, `receiver` le destinataire. Enfin le champ `content` contient le(s) argument(s) associé(s) à l'action souhaitée. Côté client, pour envoyer une telle structure, il suffit d'appeler la fonction `int send_message(const char *mess, char **error_mess)` avec `mess` de la forme : `"/<ACTION> <ARGS>"` et `err_mess` le message d'erreur, si une erreur survient. L'affichage de message reçu se fait à la couche supérieure, dans l'interface graphique ou en lignes de commandes en appelant la fonction `int receive_message( message *m)`. Ces fonctions se trouvent dans une bibliothèque écrite pour le client (`libclient.a`) afin de les utiliser dans l'interface graphique et en préparation à la surcouche sécurisée de la troisième livraison.

L'interface du client de chat a été développée en Vala à l'aide de la bibliothèque graphique GTK. La fenêtre a été dessinée à l'aide de Glade qui est un User Interface Designer produisant un fichier `.xml`, ce dernier étant utilisé dans le code Vala.

Côté serveur, les clients sont gérés dans des fils d'exécution séparés. Chaque message reçu est filtré selon son code d'action et le traitement correspondant est effectué. Si la connexion avec un client est perdue, le fil d'exécution s'arrête en déconnectant l'utilisateur.

1.3.2 Certification

La seconde livraison consistait en la mise en place d'une infrastructure à clefs publiques (PKI). Cette infrastructure permet de garantir l'identité d'un utilisateur en lui délivrant un certicat numérique qui permet d'effectuer des opérations cryptographiques. Pour cela nous avons utilisé EJBCA (Enterprise Java Bean Certificate Authority) qui est une application utilisant un serveur JBoss. L'installation s'est faite en quelques étapes :

Récupération de JBoss et d'EJBCA

```
$ wget http://sourceforge.net/projects/jboss/files/JBoss/JBoss-5.1.0.GA/jboss-5.1.0.GA-jdk6.zip
$ wget http://sourceforge.net/projects/ejbca/files/ejbca4/ejbca_4_0_10/ejbca_4_0_10.zip
$ unzip jboss-5.1.0.GA-jdk6.zip
$ unzip ejbca_4_0_10.zip
```

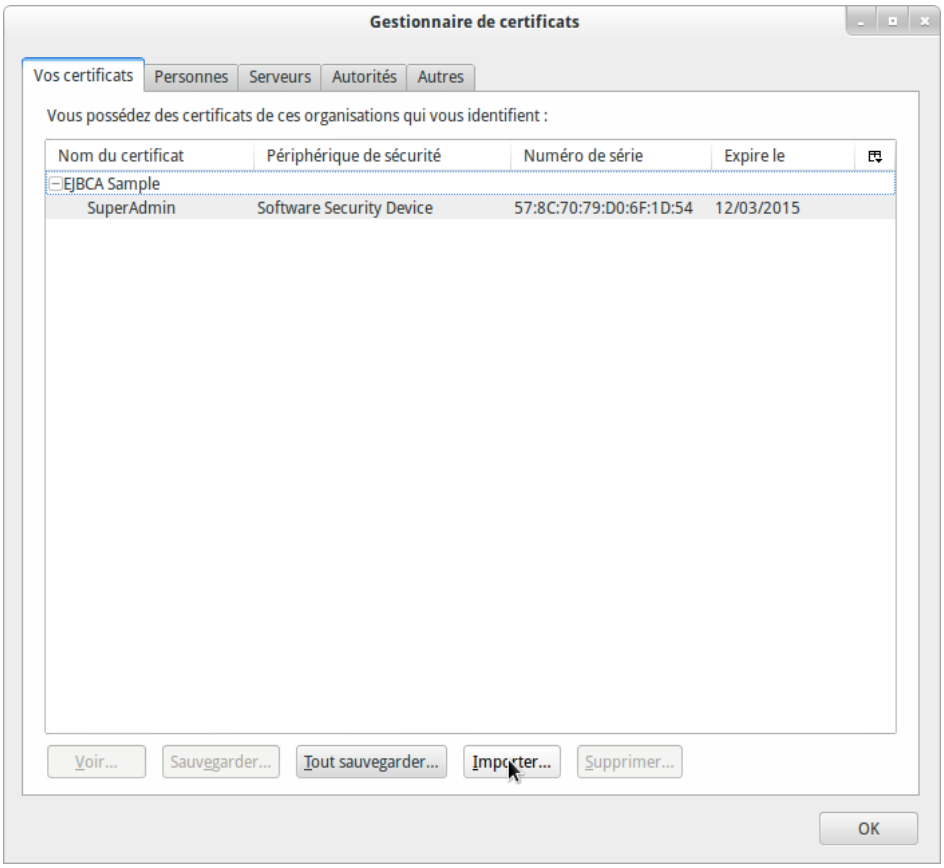
Configuration et construction d'EJBCA

```
$ echo "appserver.home=/home/user/jboss-5.1.0.GA" >> ejbca_4_0_10/conf/ejbca.properties
$ cd ejbca_4_0_10; ant bootstrap
```

Installation et déploiement d'EJBCA

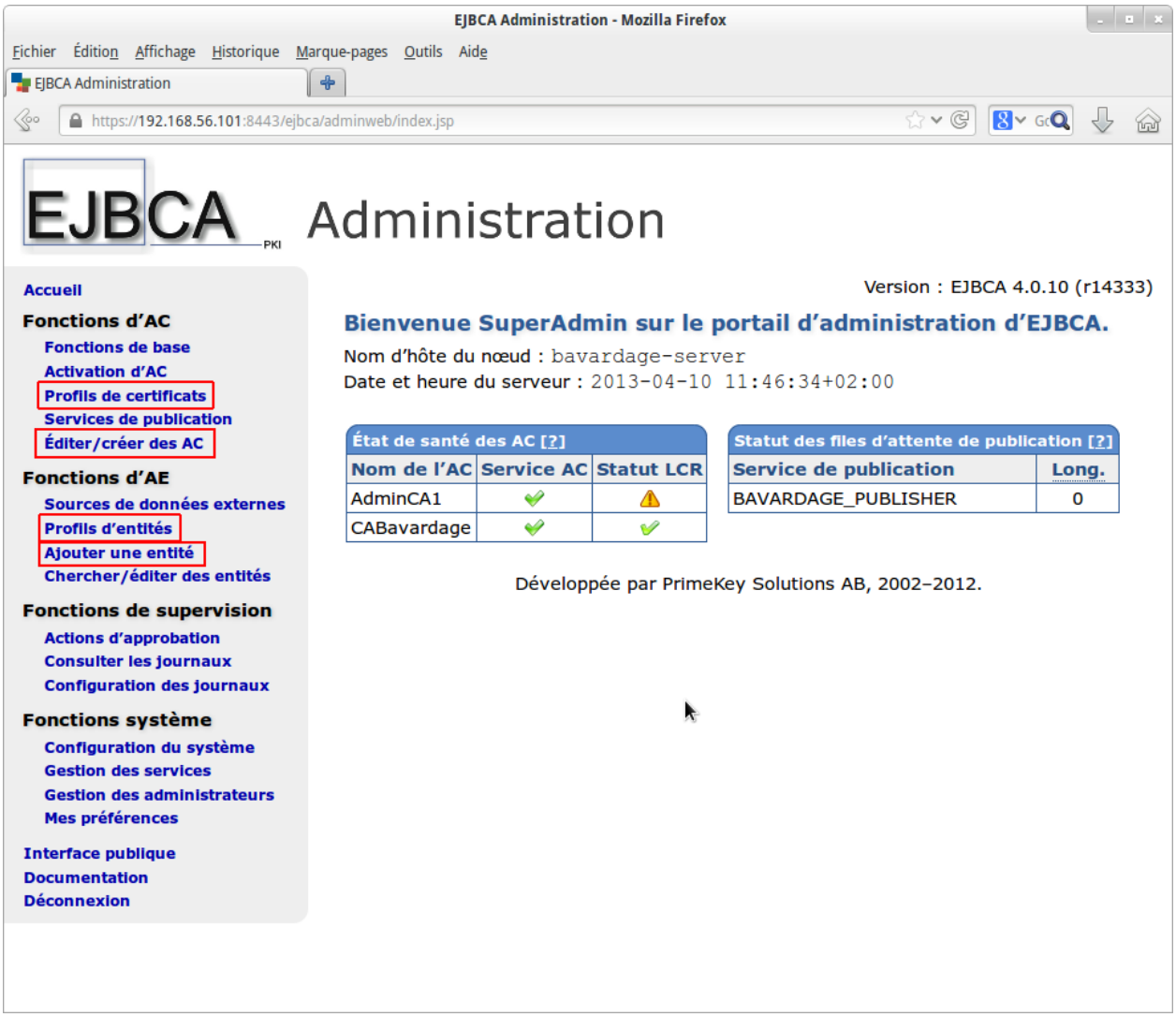
```
$ /home/user/jboss-5.1.0.GA/bin/run.sh &
$ ant install
$ ant deploy
```

Pour la configuration des autorités de certification et des différents profils, EJBCA met à disposition une interface web. Il faut tout d'abord récupérer le certificat administrateur qui a été généré lors de l'installation se trouvant dans `/home/user/ejbca_4_0_10/p12/superadmin.p12` puis de l'ajouter dans les certificats du navigateur comme suit (Firefox) :



On peut alors accéder à la partie administration à partir de l'adresse : `https://inf-srv-securechat:8443/ejbca/adminweb/index.jsp`

La page obtenue est la suivante :



Les parties importantes ont été entourées en rouge et voici leurs descriptions.

Les profils contiennent les informations nécessaires et utiles qui permettent de générer les certificats. Comme on peut voir sur le schéma précédent, il en existe deux sortes.

Profils de certificats C'est dans cette section que l'on crée les différents profils de certificats :
- `BAVARDAGESERVER` : profil pour le serveur sécurisé ;
- `BAVARDAGEUSER` : profil pour les utilisateurs sécurisés.

Profils d'entités Chaque type d'entité a un profil associé :
- `BAVARDAGE_SERVER` : profil pour le serveur sécurisé ;
- `BAVARDAGE_USER` : profil pour les utilisateurs sécurisés.

On utilise deux profils d'entités différents car les rôles de chacun sont différents. Le premier pour déterminer l'entité du server et le second l'entité de l'utilisateur.
Ces profils sont décrits en détail dans le document annexe : Politique de certification

Éditer/créer des AC Une AC (autorité de certification) est une entité dont la mission est de vérifier les données du demandeur de certificat, de signer et de maintenir une liste des certificats et une liste des révocations. Sur la capture d'écran ci-dessus, on voit la liste des AC contenant celle relative à notre projet : `CABavardage`

Ajouter une entité Lorsqu'un utilisateur désire obtenir un certificat, un administrateur doit au préalable lui créer une entité associée. Il pourra ensuite récupérer son certificat pour sa clef RSA.

1.3.3 Partie sécurisée

La troisième livraison consistait en la surcouche sécurisée du client de chat et un serveur « sécurisé » dont le rôle est de gérer les utilisateurs sécurisés et la sécurité des messages.

Nous avons choisi tout d'abord d'ouvrir une connexion SSL entre le client sécurisé et le serveur sécurisé grâce aux fonctions d'OpenSSL. Ensuite, nous avons crée des codes d'actions spécifiques à l'utilisation sécurisée du chat comme `CONNECT_SEC, CREATE_ROOM_SEC`... La gestion des clients dans le serveur sécurisé est similaire au serveur classique avec un fil d'exécution par client et une socket SSL.

Côté client, la surcouche sécurisée se base sur la bibliothèque du client classique et utilise la même logique. Ainsi nous avons deux fonctions principales :

- `int send_message_sec (const char *mess, char **err_mess)`
- `int receive_message_sec (message *m)`

On rajoute également des fonctions de chiffrement/déchiffrement de messages :

- `char *aes_encrypt (unsigned char *key, unsigned char *iv, char *plaintext, int *len)`
- `char *aes_decrypt (unsigned char *key, unsigned char *iv, char *ciphertext, int *len)`

Pour le chiffrement et le déchiffrement nous utilisons une fonction `char *gen_keyiv(key_iv keyiv, unsigned char *key_data, int key_data_len)` qui génère la clé et l'IV (valeur d'initialisation). C'est cette paire (clé, iv) qui est utilisée par les fonctions `aes_encrypt` et `aes_decrypt` pour chiffrer et déchiffrer les messages.

Comme les noms des fonctions l'indiquent, nous utilisons AES-256-CBC pour chiffrer les messages envoyés (champ `content`). AES est un chiffrement par bloc qui supporte des clés et des blocs de 128, 192 et 256 bits. Le mode de chiffrement CBC consiste à diviser les données en plusieurs blocs de même taille et à les chiffrer de manière chaînée (le résultat précédent est utilisé lors du chiffrement suivant). L'IV est utilisée pour le chiffrement du premier bloc. Avec le mode CBC l'IV est unique et aléatoire, ce qui rend la clé réutilisable sans risque, et il n'y a pas de risque de répétition de bloc. Nous avons choisi AES-256-CBC pour avoir une plus grande taille des clés et la sécurité qu'offre le mode de chiffrement CBC.

1.4 Problèmes rencontrés

Nous avons rencontré un problème pour la livraison du sprint 2. En effet, nous devions configurer un serveur, ce que nous avions fait dans une machine virtuelle. Cependant, le format de livraison utilisé jusqu'alors n'était plus utilisable car la machine faisait plusieurs giga octets de données. Nous avons alors convenu avec Mme Bardet de livrer sur une clef USB dans un

premier temps et de configurer une machine mise à disposition par Mr Macadré en début du sprint 3.

Pour le sprint 3, nous avons également eu des problèmes au niveau de l'apprentissage d'OpenSSL. En effet, nous n'avions aucune expérience avec cette bibliothèque et les exemples sont nombreux mais très différents. Finalement, Mme Bardet nous a conseillé un très bon ouvrage : Network Security with OpenSSL.

Chapitre 2

Manuel d'utilisation

2.1 Récupération du projet

Les sources du projet sont disponibles sur un dépôt git. Elles peuvent être récupérées en ligne de commande ou en ligne.

2.1.1 En ligne de commande

Placez vous dans un terminal et exécutez la commande suivante :

```
$ git clone git://github.com/legrajul/bavardage.git
```

2.1.2 En ligne

Une archive contenant le projet peut être téléchargée à l'adresse :
`https://github.com/legrajul/bavardage/archive/master.zip`

2.2 Compilation

2.2.1 Dépendances Ubuntu

Pour la compilation du projet il faut d'abord vérifier si toutes les dépendances sont satisfaites :

- gcc : permet de compiler le code C.
- cmake : permet de précompiler un projet pour différentes plateformes et de générer des makefile.
- valac-0.18 : compilateur Vala qui précompile (dans notre cas) le code source vala en code source C.
- libgtk-3-dev : bibliothèque multi plateformes pour créer des interfaces graphiques.
- libgee-dev : bibliothèque de collections fournissant des classes basées sur GObject.
- libglib2.0-dev : fichiers de développement pour la bibliothèque GLib.
- libssl-dev : bibliothèque de développement OpenSSL.
- libsqlite3-dev : bibliothèque de développement SQLite.

2.2.2 Compilation des sources

Les commandes suivantes doivent être exécutées dans le dossier racine du projet git bavardage :

```
$ mkdir src/build
$ cd src/build
$ cmake ..
$ make
```

2.3 Exécution

2.3.1 Demande de certificat

Pour faire une demande de création de compte sécurisé, l'utilisateur doit envoyer un e-mail à julien.legras@etu.univ-rouen.fr . Il faut renseigner les informations suivantes :

- les noms et prénoms
- le pays
- la ville
- l'organisation
- l'adresse mail
- le département

L'utilisateur peut entrer ce qu'il veut dans les différents champs. Seul les éléments entrés par l'administrateur sur EJBCA seront pris en compte. Après avoir reçu le message de confirmation de compte créé par l'administrateur, il faut générer une clé RSA 2048 avec la commande :

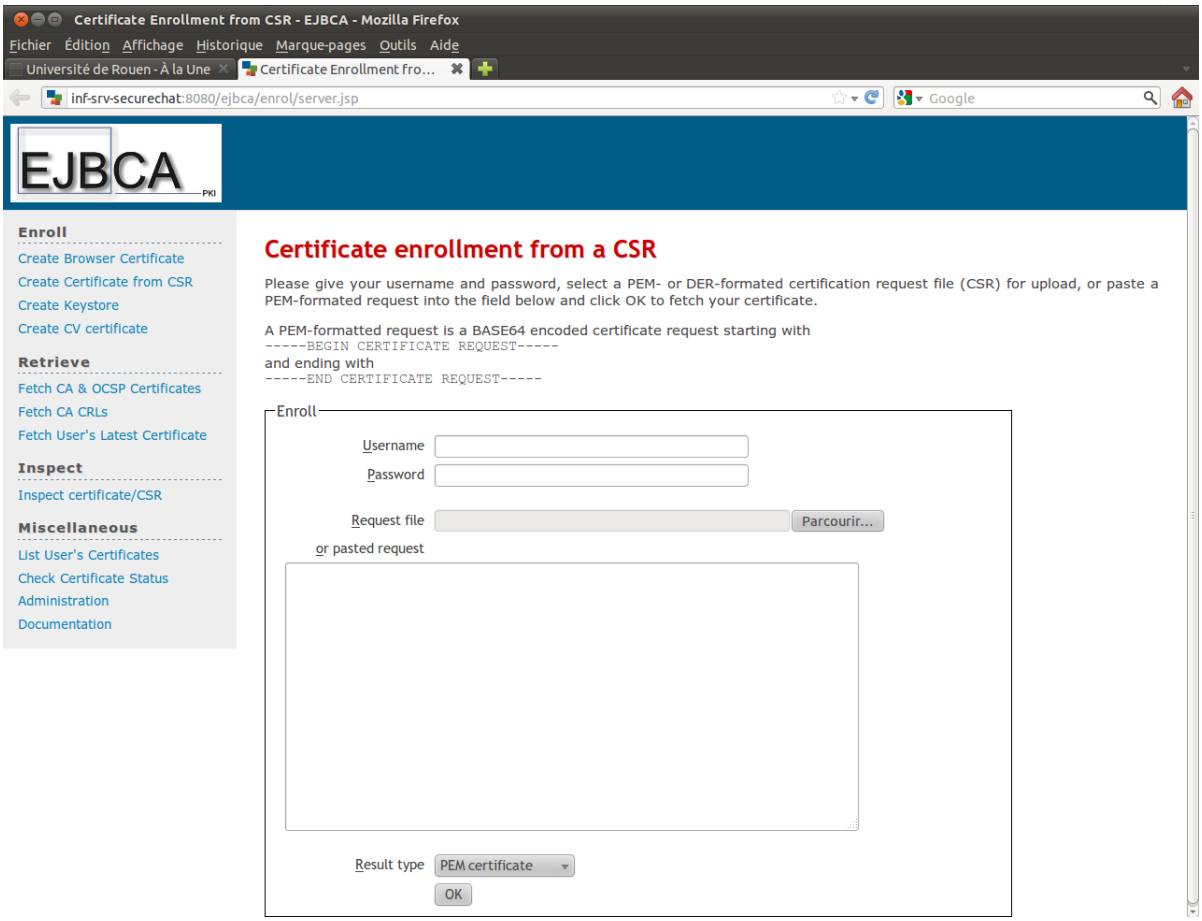
```
$ openssl genrsa -out maclef.pem 2048 -aes256
```

Puis on crée la requête :

```
$ openssl req -new -key maclef.pem -out marequete.req
```

Pour récupérer son certificat, il faut aller dans la partie publique du site web à l'adresse `http://inf-srv-securechat:8080/ejbca/enrol/server.jsp` puis :

- Entrer les identifiants (login/mot de passe) fournis par l'administrateur ;
- Selectionner le fichier "marequete.req" généré sur votre ordinateur ;
- Valider avec le bouton "OK".



2.3.2 Serveur non sécurisé

Pour lancer le serveur non sécurisé, il faut d'abord se rendre dans le répertoire "server" du répertoire build et taper les instructions suivantes :

```
$ ./server ADRESSE_SERVEUR PORT_SERVEUR
```

`ADRESSE_SERVEUR` : l'adresse IP ou le nom de domaine du serveur non sécurisé
`PORT_SERVEUR`: le numéro du port d'écoute du serveur

2.3.3 Serveur sécurisé

Pour lancer le serveur sécurisé, il faut d'abord se rendre dans le répertoire "serversec" du répertoire "build" et taper les instructions suivantes :

```
$ ./serversec ADRESSE_SERVEUR_SEC PORT_SERVEUR_SEC
```

`ADRESSE_SERVEUR_SEC` : l'adresse IP ou le nom de domaine du serveur sécurisé
`PORT_SERVEUR_SEC` : le numéro du port d'écoute du serveur sécurisé

Contraintes :
Le certificat client obtenu lors de la demande devra être placé dans le dossier client dont le chemin est : src/build/client.
Le certificat server devra être placé dans le dossier server_sec dont le chemin est : src/build/server_sec.
Le certificat racine root.pem devra être placé à la fois dans le dossier client et dans le dossier server_sec.

2.3.4 Client

Pour lancer le client, il faut se rendre dans le répertoire "client" du répertoire "build" et lancer les instrutions suivantes :

– Client console :

```
$ ./clientsec-cli ADRESSE_SERVEUR PORT_SERVEUR ADRESSE_SERVEUR_SEC
    PORT_SERVEUR_SEC CERTIFICAT CLE_PRIVEE [MOT_DE_PASSE_CLE]
```

– Client Graphique :

```
$ ./Client
```

`ADRESSE_SERVEUR` : l'adresse IP ou le nom de domaine du serveur non sécurisé.
`PORT_SERVEUR` : le numéro du port d'écoute du serveur.
`ADRESSE_SERVEUR_SEC` : l'adresse IP ou le nom de domaine du serveur sécurisé.
`PORT_SERVEUR_SEC` : le numéro du port d'écoute du serveur sécurisé.
`CERTIFICAT` : le chemin d'accès au certificat délivré par l'autorité de certification.
`CLE_PRIVE` : la clé privée générée par l'utilisateur
`MOT_DE_PASSE_CLE` : le mot de passe de la clé privée si elle a été généré avec un mot de passe.

2.3.5 Utilisation du client console

La forme générale des commandes est :

```
/NOM_COMMANDE [paramètre_1 [Paramètre_2 [Paramètre_3 Paramètre_4]]]
```

La commande de base de l'application est /HELP. Elle permet de lister toutes les commandes de l'application.

Commandes du client classique et les actions

`/CREATE_ROOM room_name` : Permet de créer un salon ayant le nom "room_name"
`/DELETE_ROOM room_name` : Permet de supprimer le salon du nom de "room_name"
`/QUIT_ROOM room_name` : Permet de quitter le salon du nom de "room_name"
`/JOIN_ROOM room_name` : Rejoindre un salon du nom de "room_name"
`/DISCONNECT` : Se déconnecter du serveur non sécurisé
`/CONNECT user` : Se connecter avec l'identifiant "user"
`/MP user message` : Envoyer un message privé à l'utilisateur "user"

Commandes du client sécurisé et les actions

`/CONNECT_SEC user user_certif user_key` : Connexion au serveur sécurisé avec le nom d'utilisateur "user" avec le certificat "user_certif" et la clé privée "user_key" (Si la clé privée a été créée avec un mot de passe, il faut ajouter le mot de passe après "user_key")

`/CREATE_ROOM_SEC room_name` : Permet de créer un salon sécurisé du nom de "room_name"

`/DELETE_ROOM_SEC room_name` : Permet de supprimer le salon sécurisé du nom "room_name"

`/DISCONNECT_SEC` : Se déconnecter du serveur sécurisé et du serveur non sécurisé

`/QUIT_ROOM_SEC room_name` : Quitter le salon sécurisé du nom de "room_name"

`/JOIN_ROOM_SEC room_name` : Rejoindre le salon sécurisé du nom de "room_name"

`/DEL_ACCOUNT_SEC` : Supprimer le compte utilisateur sécurisé

`/MP_SEC user message` : Envoyer un message privé sécurisé

`/MESSAGE room_name message` : Envoyer le message "message" sur le salon "room_name"

`/ACCEPT_JOIN_ROOM_SEC room_name user` : Accepter l'ajout d'un utilisateur qui fait la demande pour rejoindre le salon "room_name". Seul l'administrateur du salon a les droits pour effectuer cette opération.

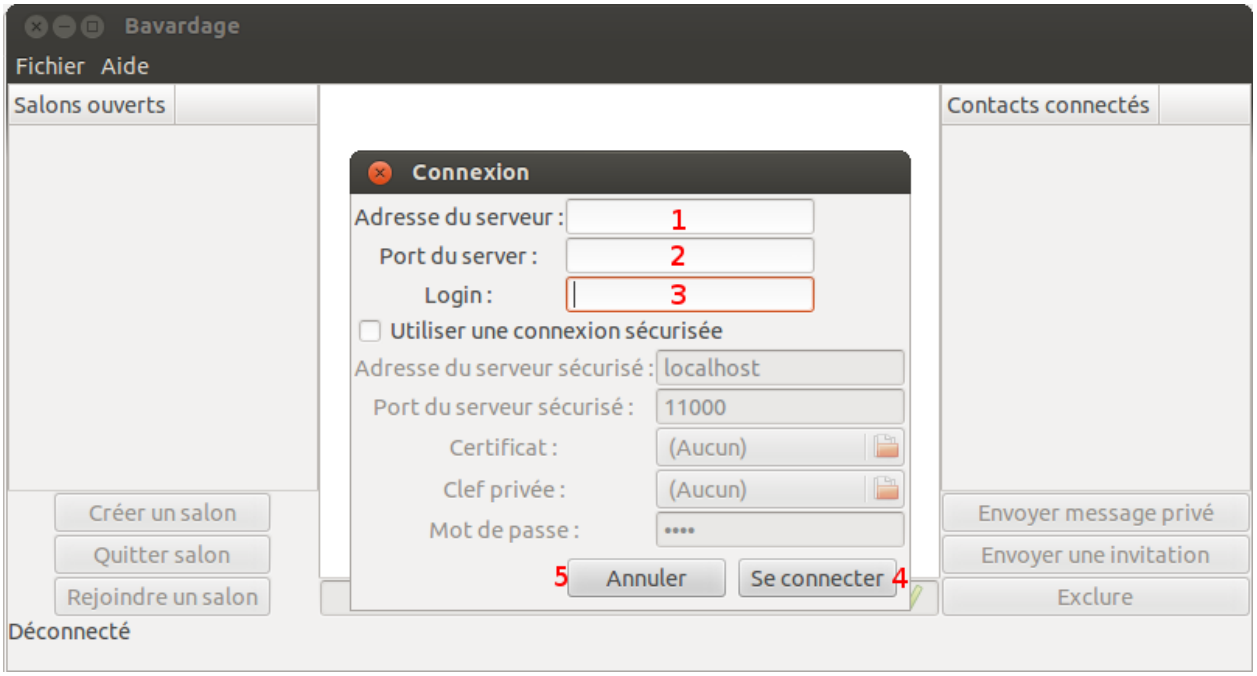
`/REFUSE_JOIN_ROOM_SEC room_name user` : Refuser la requête d'ajout qu'un utilisateur a envoyé pour rejoindre un salon. Seul l'administrateur du salon a les droits pour effectuer cette opération.

`/EJECT_FROM_ROOM_SEC room_name user` : Supprimer un utilisateur "user" du salon "room_name". Seul l'administrateur du salon a les droits pour effectuer cette opération.

2.3.6 Utilisation du client graphique

Le client graphique permet d'effectuer les mêmes opérations que le client console mais avec l'intuitivité que nous offre l'interface graphique.

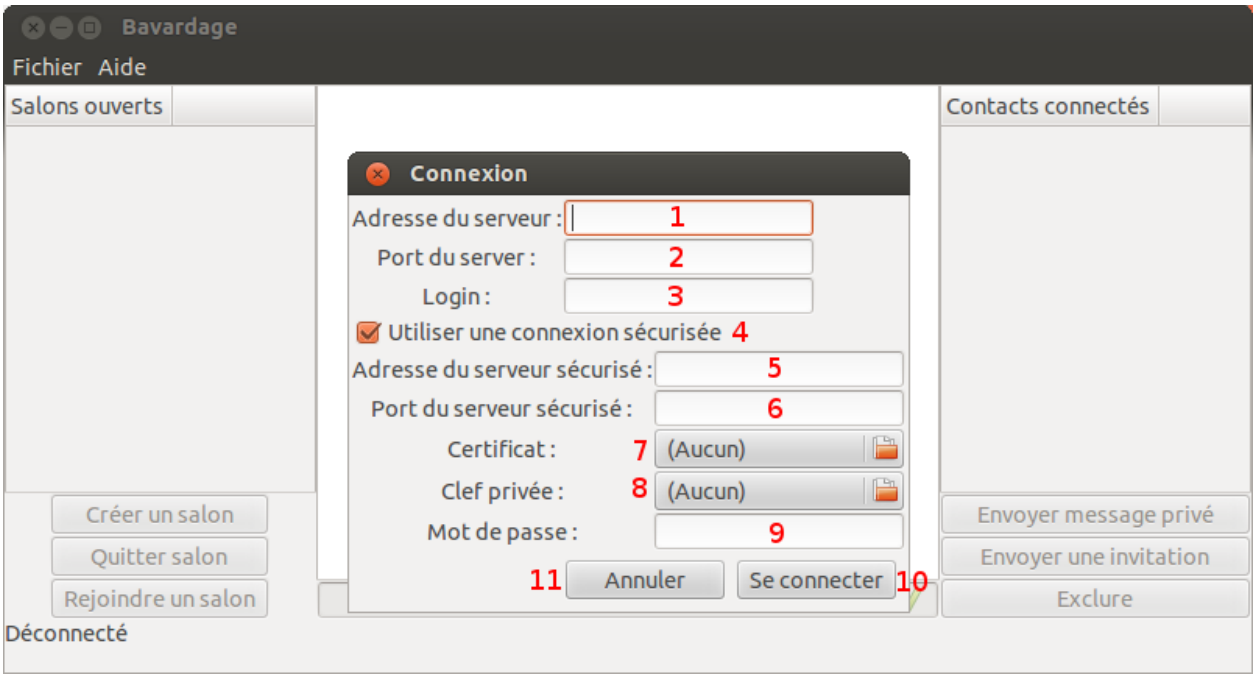
Connexion non sécurisée



Pour effectuer une connexion non sécurisée, il faut :

1. Entrer l'adresse du serveur non sécurisé ;
2. Entrer le port d'écoute du serveur non sécurisé ;
3. Entrer le login voulu pour se connecter ;
4. Cliquer sur le bouton "Se connecter" pour lancer la connexion ;
5. Cliquer sur le bouton "Annuler" pour annuler la connexion ;

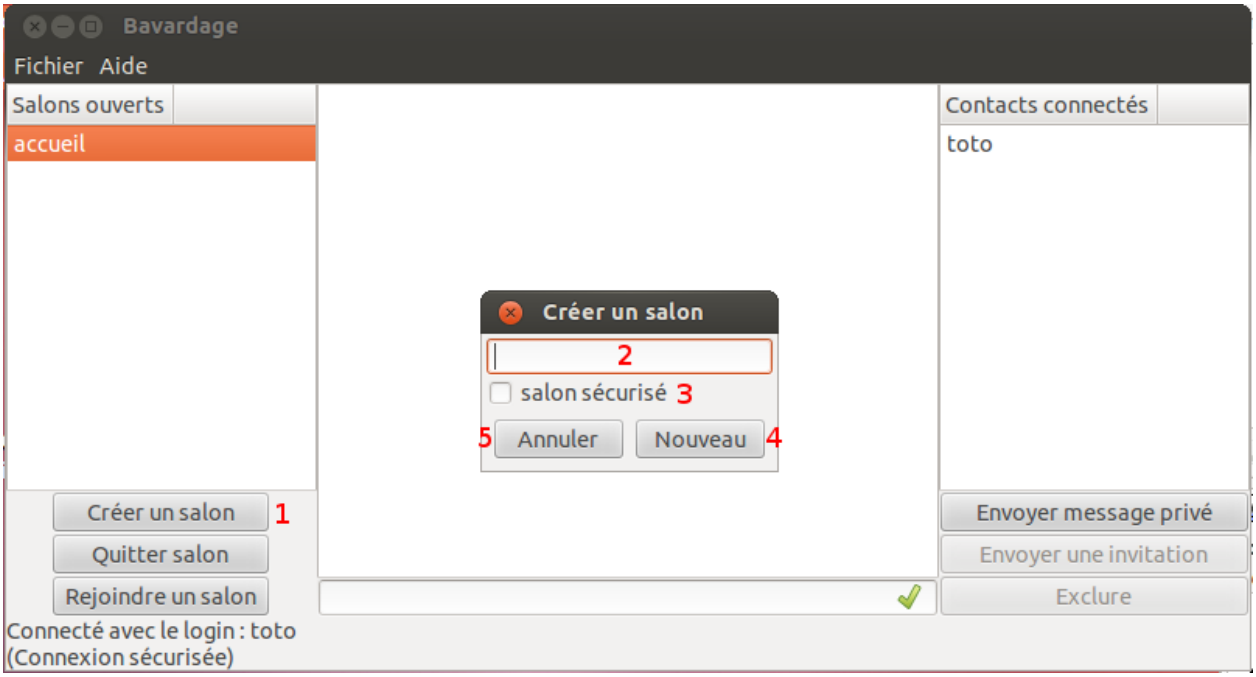
Connexion sécurisée



Pour effectuer une connexion sécurisée, il faut :

1. Entrer l'adresse du serveur non sécurisé ;
2. Entrer le port d'écoute du serveur non sécurisé ;
3. Entrer le login voulu pour se connecter ;
4. Cocher l'option de connexion sécurisée ;
5. Entrer l'adresse du serveur sécurisé ;
6. Entrer le port d'écoute du serveur sécurisé ;
7. Sélectionner le fichier du certificat délivré par l'autorité de certification ;
8. Sélectionner le fichier de la clé privée ;
9. Entrer le mot de passe correspondant à la clé privée (Si aucun mot de passe n'a été utilisé pour générer la clé privée, ne rien entrer dans ce champ) ;
10. Cliquer sur le bouton "Se connecter" pour lancer la connexion ;
11. Cliquer sur le bouton "Annuler" pour annuler la connexion ;

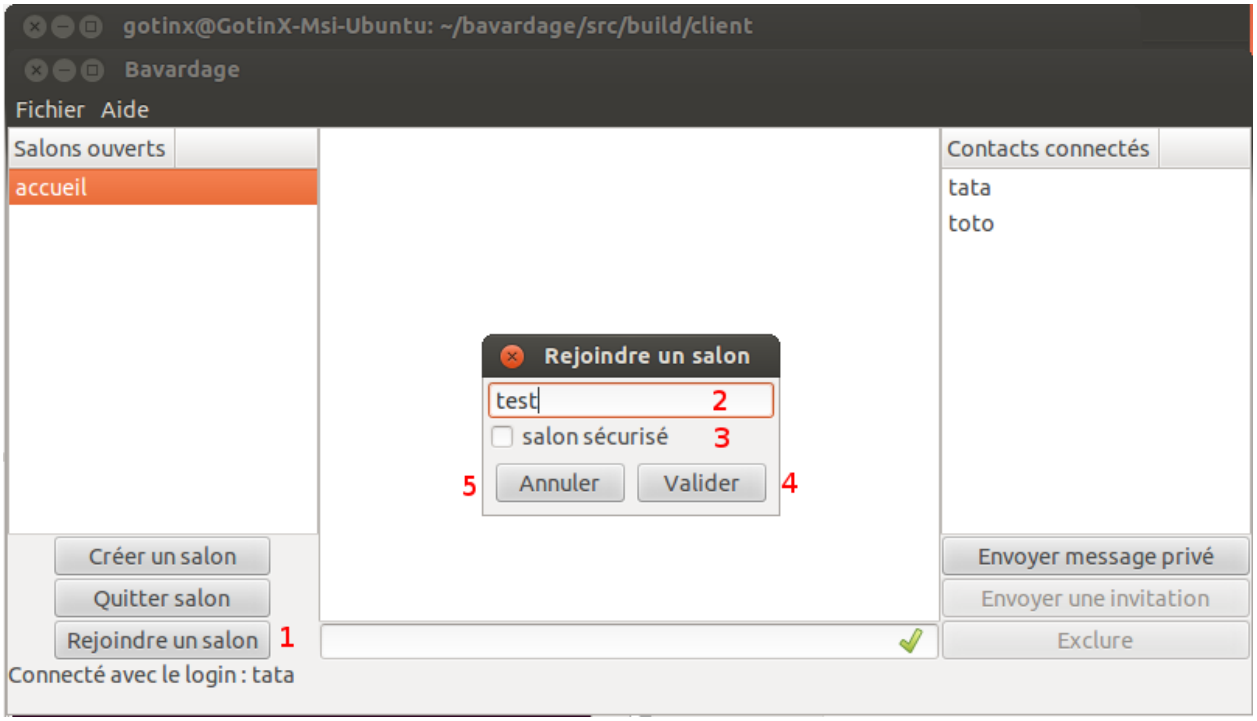
Création d'un salon (sécurisé ou non sécurisé)



Pour effectuer la création d'un salon (sécurisé ou non sécurisé), il faut :

1. Cliquer sur le bouton "Créer un salon" ;
2. Entrer le nom du salon à créer dans le champ ;
3. Cocher le champ "salon sécurisé" s'il s'agit d'un salon sécurisé ;
4. Cliquer sur le bouton "Nouveau" pour créer le salon ;
5. Cliquer sur le bouton "Annuler" pour annuler la création du salon.

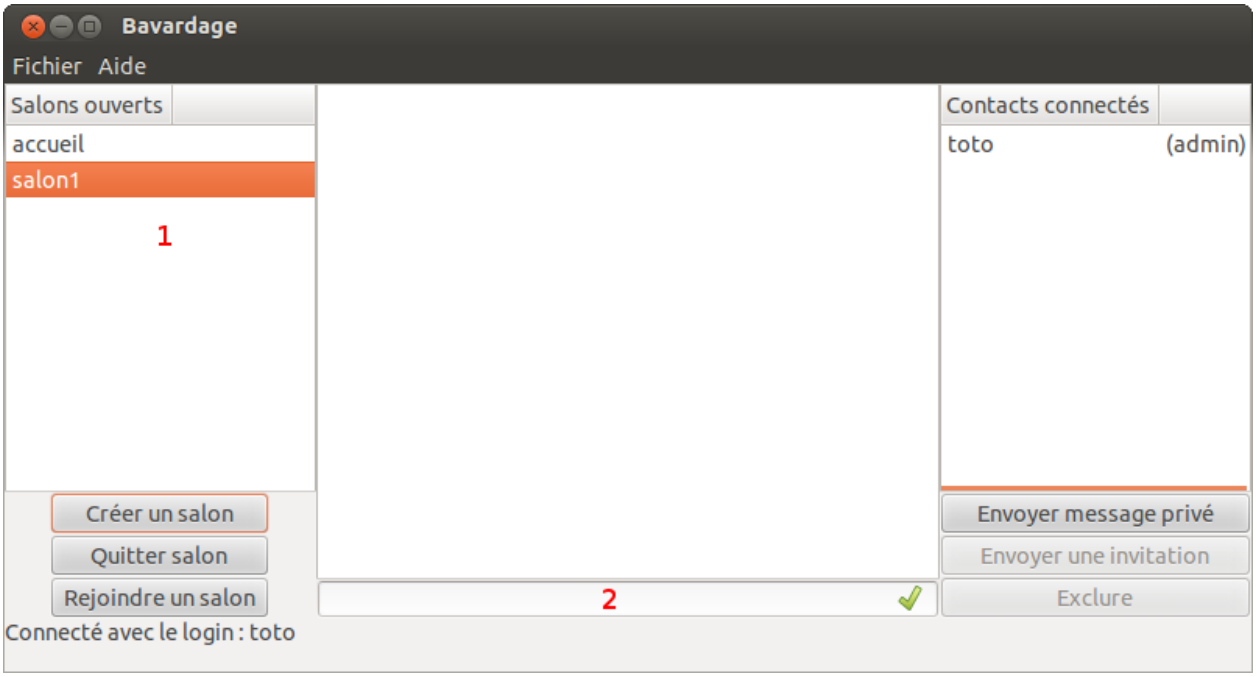
Rejoindre un salon (sécurisé ou non sécurisé)



Pour rejoindre un salon (sécurisé ou non sécurisé), il faut :

1. Cliquer sur le bouton rejoindre un salon ;
2. Entrer le nom du salon à rejoindre ;
3. Cocher la case "salon sécurisé" ;
4. Cliquer sur le bouton "Se connecter" pour rejoindre le salon s'il n'est pas sécurisé ou envoyer une demande à l'administrateur du salon si le salon est sécurisé ;
5. Cliquer sur le bouton "Annuler" pour annuler la connexion.

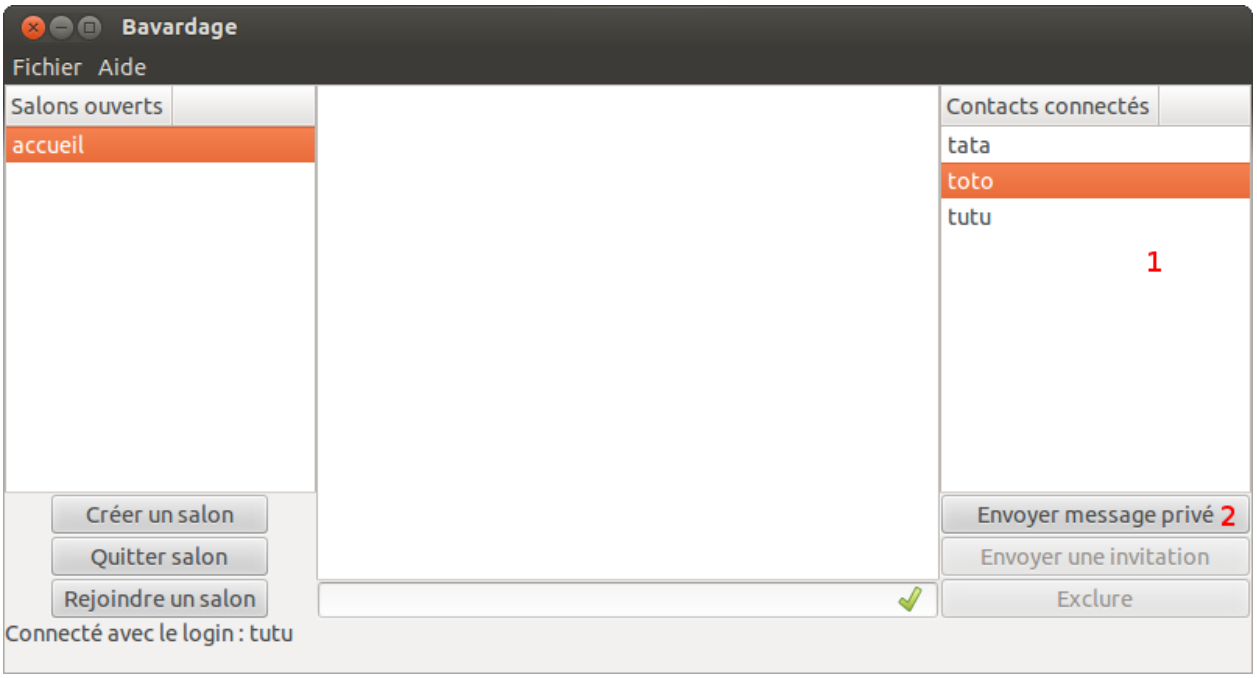
Envoyer un message dans un salon



Pour envoyer un message dans un salon, il faut :

1. Sélectionner le salon dans lequel on veut envoyer le message dans la liste des salons ouverts ;
2. Saisir le message dans la zone puis appuyer sur la touche "Entrée" pour envoyer le message.

Envoyer un message privé à un utilisateur

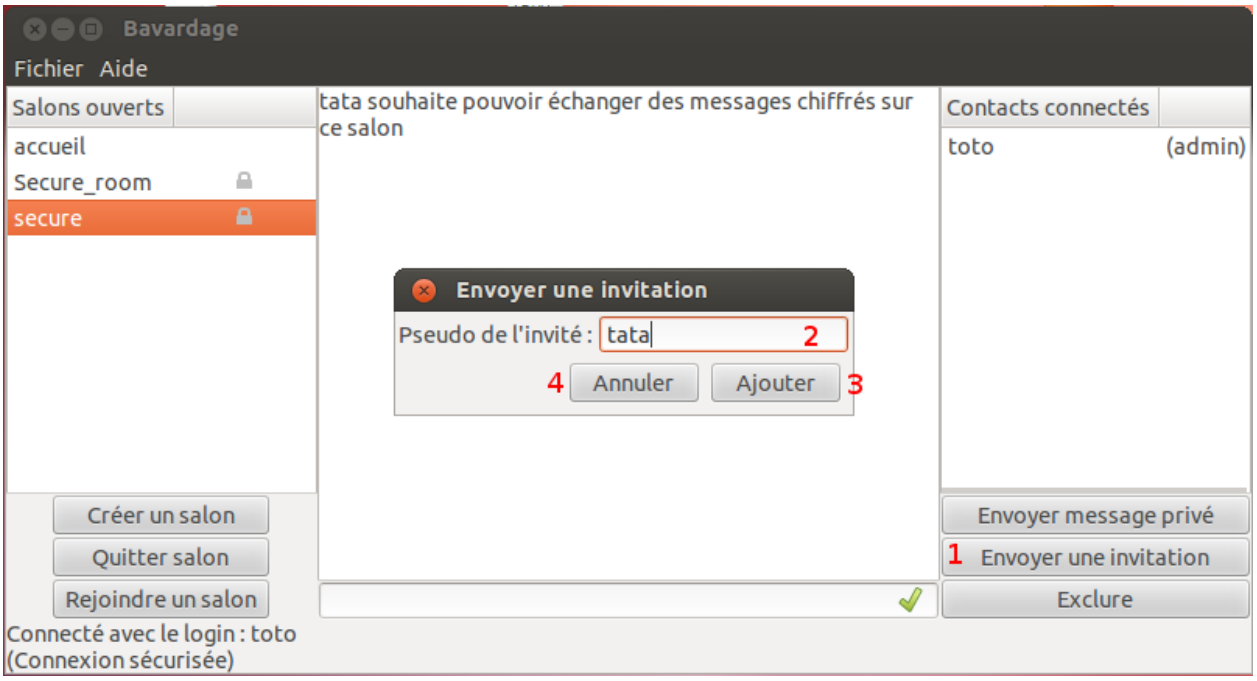


Pour envoyer des messages en mode privé à un utilisateur, il faut :

1. Sélectionner l'utilisateur dans la liste des contacts connectés ;

2. Cliquer sur le bouton "Envoyer message privé".
3. Une boite de dialogue s'afficher pour demander si l'on veut une discussion sécurisée avec l'utilisateur

Accepter la requête d'un utilisateur qui souhaite rejoindre un salon



Pour accepter une demande d'invitation reçu,

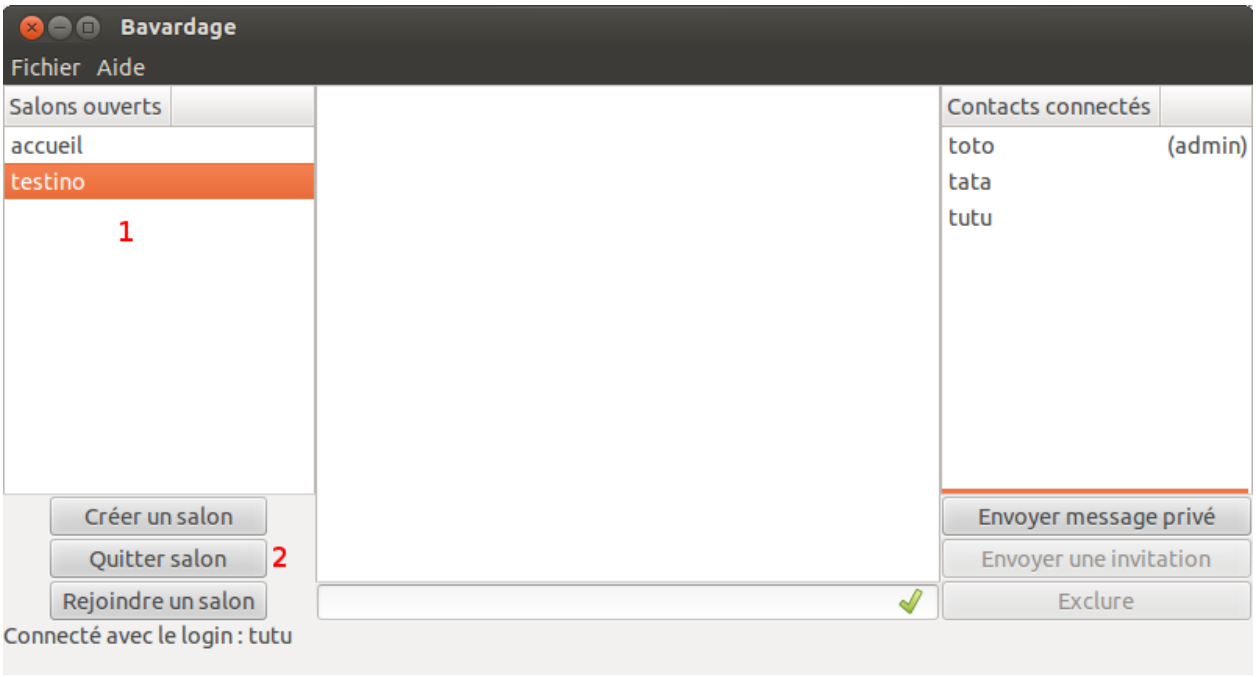
1. Cliquer le bouton "Envoyer une invitation" ;
2. Entrer le pseudo de l'utilisateur ayant envoyé la demande ;
3. Cliquer le bouton "Ajouter" pour rajouter l'utilisateur au salon sécurisé
4. Cliquer sur le bouton "Annuler" pour annuler l'ajout de l'utilisateur

Exclure un utilisateur

Pour exclure un utilisateur,

1. Sélectionner le pseudo de l'utilisateur dans la liste des contacts connectés ;
2. Cliquer sur "Exclure".

Quitter un salon



Pour quitter un salon, il faut :

1. Sélectionner le salon que l'on veut quitter dans la liste des salons ouverts ;
2. Cliquer sur le bouton "Quitter salon".

Se déconnecter

Pour se déconnecter,

1. Aller dans le menu "Fichier" ;
2. Cliquer sur "Déconnecter".

Chapitre 3

Conclusion

Ce projet a été découpé en deux parties distinctes, l'analyse et l'étude d'un sujet associées à une gestion de projet afin de proposer une solution au client puis une deuxième phase qui a été l'implémentation de la solution proposée également accompagnée d'une gestion de projet.

L'étude et l'analyse théorique du sujet lors de la première phase de projet nous a permis d'apprendre à mettre en place des documents spécifiques et détaillés sur chaque fonctionnalités en utilisant des méthodes de conception professionnelle. La méthode agile scrum que nous avons choisie a permis d'établir un planning de gestion et un découpage astucieux du projet en plusieurs étapes et ainsi proposer une solution structurée respectant au mieux les besoins du client.

La deuxième phase d'implémentation a été un moyen de mettre en pratique nos techniques de programmation acquises au cours de nos formations antérieures, nos connaissances théoriques en cryptographie mais également d'apprendre de nouvelles technologies, en particulier pour la mise en place des techniques de cryptographie et la gestion d'une autorité de certification avec certificats dans une application concrète et fonctionnelle.

La méthode agile scrum utilisée pour ce projet a simplifié les phases afin d'en raccourcir la durée. Les méthodes agiles, se basent sur la notion de communauté de projet dans laquelle les développeurs et les utilisateurs sont présents en permanence pour exprimer ou répondre à une question liée au projet. C'est une communication permanente entre les membres du groupe de projet, et le client qui a fait que nous avons su rapidement prendre en compte les opinions et idées de chacun et de réagir aux différents problèmes que ce soit lors de l'analyse ou de l'implémentation du projet. Nous en tirons donc une expérience très enrichissante sur la gestion et la communication au sein d'un groupe de projet dans un cadre orienté professionnel.

Annexe A

Documents de gestion de projet

- Spécification technique des besoins : `https://github.com/legrajul/bavardage-doc/raw/master/STB/stb.pdf` ;
- Document d'architecture logicielle : `https://github.com/legrajul/bavardage-doc/raw/master/DAL/dal.pdf` ;
- Analyse des risques : `https://github.com/legrajul/bavardage-doc/raw/master/ADR/adr.pdf` ;
- Cahier de recette : `https://github.com/legrajul/bavardage-doc/raw/master/CdR/cdr.pdf`
  `https://github.com/legrajul/bavardage-doc/raw/master/CdR/CdR.xlsx` ;
- Plan de développement : `https://github.com/legrajul/bavardage-doc/raw/master/PDD/pdd.pdf` ;

Annexe B

Déclaration des pratiques de certification

https://github.com/legrajul/bavardage-doc/raw/master/DPC/dpc.pdf

Annexe C

Politique de certification

https://github.com/legrajul/bavardage-doc/raw/master/PC/pc.pdf