**Université du Québec en Outaouais INF4533**

Rapport : projet 2

Mini-serveur HTTP, couriel, pour le partage de messages

**Aline Landry LANA25609207**

**Ariella Sota SOTA18599506**

**Arnaud Niyonkuru NIYA26109309**

**Reagan Shukuru SHUR23019005  
Vestine Mukeshimana MUKV01568701**

**18 avril 2016**

**Professeur Fraczak Wojciech**

# Table des matières

[Rapport : projet 2 1](#_Toc448305280)

[Table des matières 2](#_Toc448305281)

[Description du système 3](#_Toc448305282)

[Spécification fonctionnelle 3](#_Toc448305283)

# Tables des figures

[Figure 1 Enregistrer un nouvel utilisateur 3](#_Toc448305897)

[Figure 2 Se connecter 3](#_Toc448305898)

[Figure 3 Fenêtre d'authentification 4](#_Toc448305899)

[Figure 4 Afficher la liste sans détails des messages 4](#_Toc448305900)

[Figure 5 Afficher le message reçu en détail 5](#_Toc448305901)

[Figure 6 Fenêtre d'alerte 5](#_Toc448305902)

[Figure 7 Fenêtre d'alerte 7](#_Toc448305903)

[Figure 8 Fenêtre d'alerte 9](#_Toc448305904)

## Description du système



Figure 1 Architecture du système

## Spécifications fonctionnelles

1. Enregistrer et connecter l’utilisateur : Au premier accès du serveur dans le navigateur, une fenêtre permet d’enregistrer les informations suivantes :
   1. le nom
   2. le mot de passe

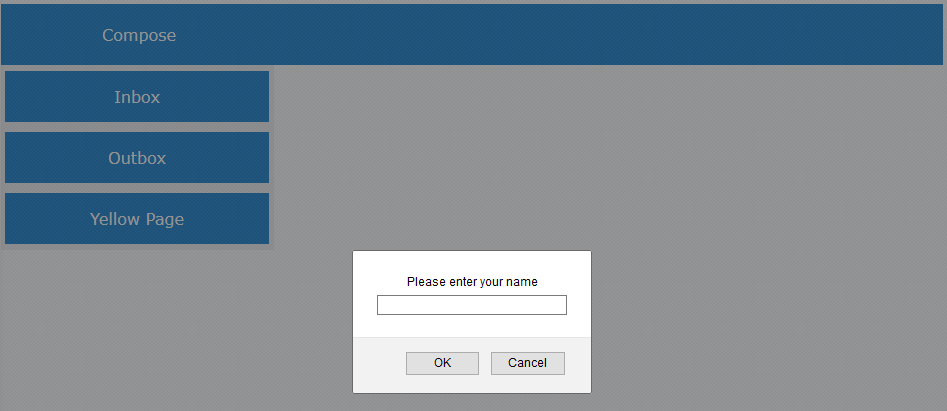


Figure 2 Entrée du nom d’utilisateur

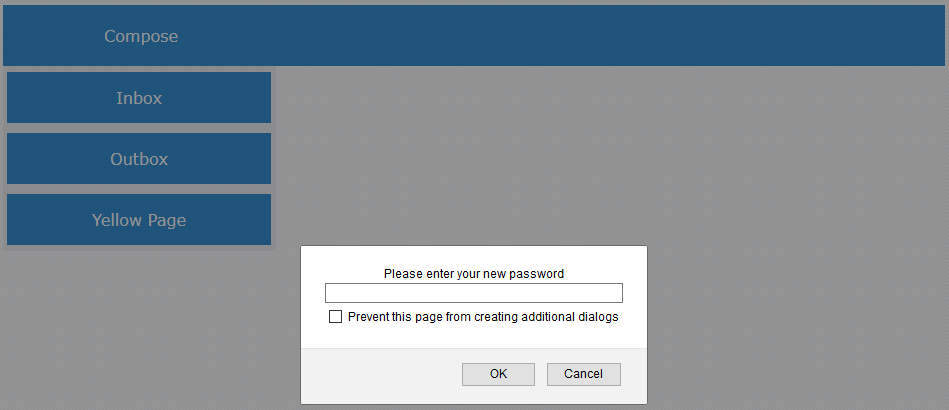


Figure 3 Entrée du mot de passe

1. Afficher la liste sans détails des messages reçus : le bouton de navigation ***Inbox*** permet l’affichage d’une liste incluant le nom d’utilisateur de l’envoyeur et la date de réception. Sur chaque ligne on peut soit lire le message en question ou l’effacer.

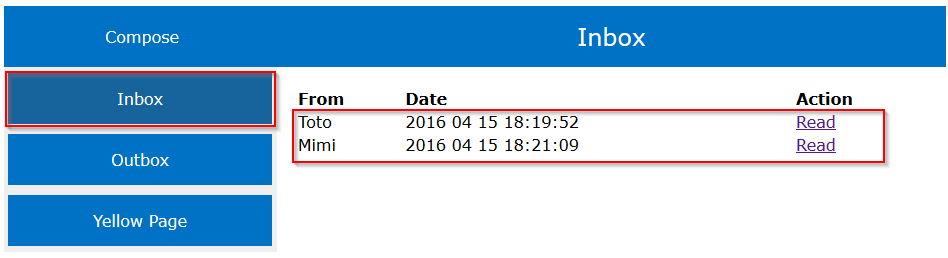


Figure 4 Afficher la liste sans détails des messages reçus

1. Afficher le message reçu en détail : le lien ***Read*** dans la page ***Inbox*** permet l’affichage en détail du nom d’utilisateur de l’envoyeur, la date de réception et le message en question.

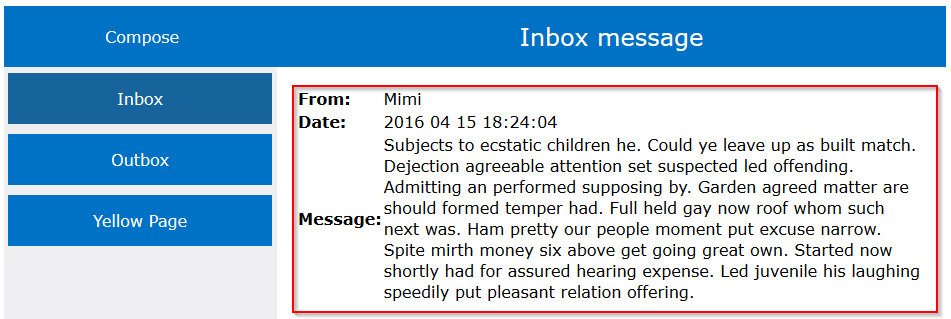


Figure 5 Afficher le message reçu en détail

1. Afficher une liste sans détails des messages envoyés : le bouton de navigation ***Outbox*** permet l’affichage d’une liste incluant le nom d’utilisateur du destinataire et la date d’envoi. Sur chaque ligne on peut soit lire le message en question ou l’effacer.

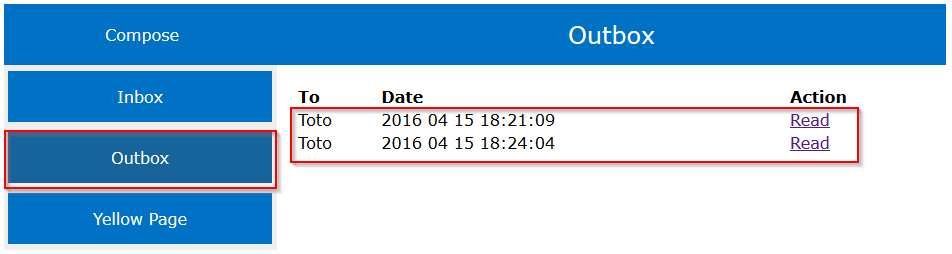


Figure 6 Afficher la liste sans détails des messages envoyées

1. Afficher le message envoyé en détail : le lien ***Read*** dans la page ***Outbox*** permet l’affichage en détail du nom d’utilisateur du destinataire, la date d’envoi et le message en question.

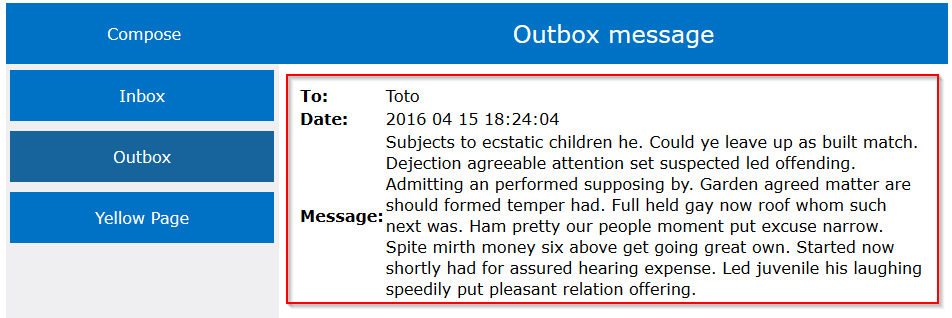


Figure 7 Afficher le message envoyé en détail

1. Créer un nouveau message : le bouton de navigation ***Compose*** permet d’afficher le formulaire avec le nom d’utilisateur du destinataire, le message et le bouton d’envoi.

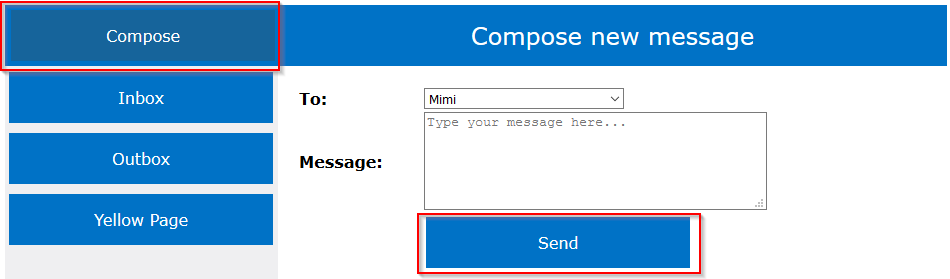


Figure 8 Écriture du message

1. Afficher une liste des contacts : le bouton de navigation ***Yellow Page*** permet l’affichage d’une liste des contacts.

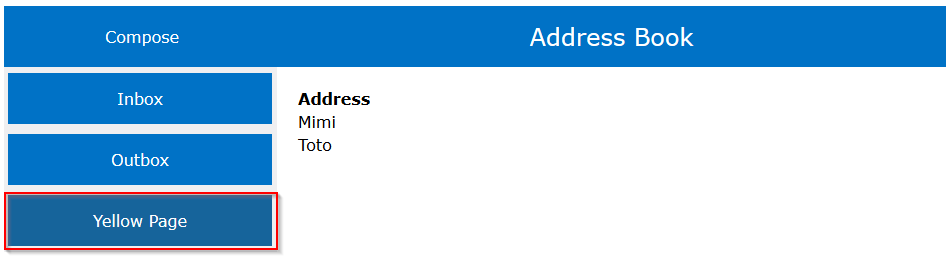


Figure 9 Liste des contacts

## Modèle pour le stockage des messages

Le système de fichier a été utilisé (Annexe A) comme modèle pour le stockage de l’état des messages en simulant la base de données et l’état est périodiquement stocké. Pour ce modèle, le module **json\_file\_object** de **node.js** a été utilisé.

Voici ses avantages :

1. Simplicité : vu la petite envergure de notre projet ce modèle est adéquat.
2. Éviter la complexité des bases de données : on n’a pas besoin d’installer de base de données ce qui va faciliter l’évaluateur du projet.
3. Visualisation simple des données : en ouvrant le fichier on peut visualiser les données sous le format JSON.

# Réalisation du serveur HTTP coté client web

## Implantation du serveur HTTP

Voici les modules utilisés pour réaliser le serveur

1. Le module **express** de **node.js** a été utilisé comme serveur HTTP
2. Le module **body-parser** de **node.js** a été utilisé pour lire les données pour les messages envoyés par les formulaires avec la méthode **POST**.
3. Le module **superagent** de **node.js** a été utilisé pour faire des requêtes synchrones ou asynchrones qu’on appelle communément **ajax**.
4. Le module **lodash** de **node.js** a été utilisé pour la synchronisation des tableaux.

## Redesign de la page web du projet 1

Pour le redesign de la page web du projet 1, le module **jade** de **node.js** a été utilisé pour remplacer les fichiers HTML est ce module permet de passer des paramètres dynamiques aux pages et génère le code HTML en dernier lieu.

## Définition d'un protocole de communication entre les pairs (peers)

Le protocole pour synchroniser l’échange de messages est basé sur l’échange des états par les serveurs.

* L’état est sauvegardé périodiquement entre le serveur et le système de fichier grâce au module **json\_file\_object**.
* L’état est synchronisé entre les 2 serveurs toutes les 5 secondes grâce aux modules **superagent** et **lodash.**
* L’état est sychronisé entre le client est son serveur L’état est sychronisé entre le client est son serveur lorsque le bouton ou lien sont appuyé, ou toutes les 5 secondes.



Figure 10 Protocole de communication

## Réalisation du serveur HTTP coté "pair"

## Considérations sur la sécurité

Pour des raisons de confidentialité, l’outil OPENSSL a été utilisé pour générer la clé publique et privée qui sont la base du chiffrement asymetrique RSA. En utilisant le module **crypto** de **node.js**, le fichier utilisé qui contient l’état (base de données) est codé à chaque fois qu’il y a un changement dans le système, et ce même fichier et décoder pour lire les données et les mettre en mémoire.

Voir Annexe B et C.

# Annexe A : Modèle système de fichier non codé

{

"encryptedKey": "",

"yp": {

"-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAI09z1lqLumDiN+gveSTkIejXioebu4M\nQTqR1mcWpdF1sUEnk1aEq/kSDm8lgS7KgSmg7NTRvpfuc47OIPVhhgECAwEAAQ==\n-----END PUBLIC KEY-----": {

"pem": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAI09z1lqLumDiN+gveSTkIejXioebu4M\nQTqR1mcWpdF1sUEnk1aEq/kSDm8lgS7KgSmg7NTRvpfuc47OIPVhhgECAwEAAQ==\n-----END PUBLIC KEY-----",

"name": "Mimi"

},

"-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAOManGnctGo+NxN41XIY9GxzhJdbCxoz\nnKCgOc4zvtgiWAxWR0C+3Ogq5XVqWvseaao3wlP/pSaB2kcli/Y40oMCAwEAAQ==\n-----END PUBLIC KEY-----": {

"pem": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAOManGnctGo+NxN41XIY9GxzhJdbCxoz\nnKCgOc4zvtgiWAxWR0C+3Ogq5XVqWvseaao3wlP/pSaB2kcli/Y40oMCAwEAAQ==\n-----END PUBLIC KEY-----",

"name": "Toto"

}

},

"letters": [

{

"date": "2016 04 15 18:19:52",

"from": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAOManGnctGo+NxN41XIY9GxzhJdbCxoz\nnKCgOc4zvtgiWAxWR0C+3Ogq5XVqWvseaao3wlP/pSaB2kcli/Y40oMCAwEAAQ==\n-----END PUBLIC KEY-----",

"to": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAOManGnctGo+NxN41XIY9GxzhJdbCxoz\nnKCgOc4zvtgiWAxWR0C+3Ogq5XVqWvseaao3wlP/pSaB2kcli/Y40oMCAwEAAQ==\n-----END PUBLIC KEY-----",

"msgTo": "mBJfnf9+zD9/0z9Q//CR/hvfkNkzM03FZh9p/dPLLhoJC6xq8MNDM/AeBu5GHmO86blXIfCkqA3AknpoaG2lig==",

"msgFrom": "4eak+TyA5IjeXFqV2nGaysFwHNFvZ+S/XEPxcFNBgahfULSN3oq+GIiSHYm3RXkX9mjrr7hMIZAVOgGfQKVgnA=="

},

{

"date": "2016 04 15 18:20:11",

"from": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAI09z1lqLumDiN+gveSTkIejXioebu4M\nQTqR1mcWpdF1sUEnk1aEq/kSDm8lgS7KgSmg7NTRvpfuc47OIPVhhgECAwEAAQ==\n-----END PUBLIC KEY-----",

"to": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAI09z1lqLumDiN+gveSTkIejXioebu4M\nQTqR1mcWpdF1sUEnk1aEq/kSDm8lgS7KgSmg7NTRvpfuc47OIPVhhgECAwEAAQ==\n-----END PUBLIC KEY-----",

"msgTo": "Zyq+nvjsj1IstXt1OgsuV38tjJnw20UVacm+pLWr9sL2tyuLUaNXA4IQdLPFi4hAP+3y7AVJH2N0UFHWcYLLtw==",

"msgFrom": "KyopUvjLog5EcxKiX/wCtzAMYpP9wIYnSMwkh5N4jQq9r9B9j5oUAwIEH20HhKtE95NsHKBABe0npmRczVIGcw=="

},

{

"date": "2016 04 15 18:20:35",

"from": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAI09z1lqLumDiN+gveSTkIejXioebu4M\nQTqR1mcWpdF1sUEnk1aEq/kSDm8lgS7KgSmg7NTRvpfuc47OIPVhhgECAwEAAQ==\n-----END PUBLIC KEY-----",

"to": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAI09z1lqLumDiN+gveSTkIejXioebu4M\nQTqR1mcWpdF1sUEnk1aEq/kSDm8lgS7KgSmg7NTRvpfuc47OIPVhhgECAwEAAQ==\n-----END PUBLIC KEY-----",

"msgTo": "Ko/ZmRQBpbCOneE4oPS2Cl/XOm1jpAivSsbXyLOuQD7IWwTUCB/wIUoSVE/V6fLSj7lbtt21iTSiR5LeW6fElA==",

"msgFrom": "Ss+KoQQ9Z6p80G96B4xHPKYNyKZYx/+1XXx+/z73scrVQy1aPo8FCleXzUcmwJuKQRnb79wWgJzgzuY2ymvhbQ=="

},

{

"date": "2016 04 15 18:21:09",

"from": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAI09z1lqLumDiN+gveSTkIejXioebu4M\nQTqR1mcWpdF1sUEnk1aEq/kSDm8lgS7KgSmg7NTRvpfuc47OIPVhhgECAwEAAQ==\n-----END PUBLIC KEY-----",

"to": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAOManGnctGo+NxN41XIY9GxzhJdbCxoz\nnKCgOc4zvtgiWAxWR0C+3Ogq5XVqWvseaao3wlP/pSaB2kcli/Y40oMCAwEAAQ==\n-----END PUBLIC KEY-----",

"msgTo": "iXJiCU/iFjjJDdGEor8UNSJQ5BtK1avy5/FU+w4BJrnmv24XqyGdEGbqBTu32EEZfScuknBA9xbjT3OZG88qrQ==",

"msgFrom": "IhyJqkbklNMsBp/jwK7izswox6hz28TpNAelzsaA06TTFH981LrS2lj48SUiFPeCRUgYxpK6EhGZ0K2QRxKsmQ=="

},

{

"date": "2016 04 15 18:24:04",

"from": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAI09z1lqLumDiN+gveSTkIejXioebu4M\nQTqR1mcWpdF1sUEnk1aEq/kSDm8lgS7KgSmg7NTRvpfuc47OIPVhhgECAwEAAQ==\n-----END PUBLIC KEY-----",

"to": "-----BEGIN PUBLIC KEY-----\nMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAOManGnctGo+NxN41XIY9GxzhJdbCxoz\nnKCgOc4zvtgiWAxWR0C+3Ogq5XVqWvseaao3wlP/pSaB2kcli/Y40oMCAwEAAQ==\n-----END PUBLIC KEY-----",

"msgTo": "",

"msgFrom": ""

}

]

}

# Annexe B: Modèle système de fichier codé

VU5GWnRMzfsE+GX2RunfETGNwHoahvsaTMkDCyFApt4803Hpbi+iVG002UpL+pI1yFy5ZvqJFXtEDFVPhIKrwIEVjmNLAqvH0AW1dNo6CAxm/iwY7GGeq04eDFYQyCP8mndRMA4KTmiSIHjOShQZ6XPSTq08t2ARhsfTtA4ABFk=lysdqQuJ2M4Hwmcb4253gLaS0jXPIia7ImvCUK+V90M2tcPK8rjuYapgZeE07y58gf3vGP6R8qtNP9aM4cgXOUG/kkibGMLkBB7TRkfAmSlWkPFLfKQqWzwsdbukr+H2loyCXnjQdpvxMwK5KE8HDlM060vylL1NMYxeATsHsvQ=i4DgtdNcRXQYoVja/1oBxEFheJXwBMp+EEh0NICr32kpLxcEZqPtktzD+YP6V+Ug9lX0pyFLqQB+MEmJo76anGZA15KHRQCjj24hmpQCO65Ur86EAoNFZfGFMJy9tiQCqUMlwiWX9JGMg0ZTPwemvi6whBzIzGR+xlIYgOamUa0=oa2olwEcA6yFeqbRJEmIapX2PXuU/7S9PLeg8h5UYbhqJZvuZ3vt7R4WtZk0u0P4xoZ3T4eriROJmHwNseEA2YuqM8Jw5m2MZB2d271iZcdDX2V4p1N2Q5uXmuxwp4y+NNfcO5z6YR/tyPCl3feUojXkM/k5/yl0NGAratp4/J0=OyXp5PGFgTSJeIc1RBUml8UOAPiJU5VooJ+DccIJU3blFdyibqO3ZQ1SZ2l6EfS4yXCI/Cv+Ssja0Ug47PzpUglE6kgGLivjVhwNxtjPwbgR8AEqcQcDv9JpRKSGNwcW0XhmscQ6oT6CByMIBz9cSIxjZozh64p2XzC8qMBdu8c=awyJrHCcjFCf1/981fkth0Q7jSAunYulWYKMYpy8iNJs71vAbS2hGgUsI60A+lPTg7EDtRhlqKTXvUZmjUWjxMtZ1cijTezmWcedkYldDhVmyXA3sUUUSTR9QkjIoSd3mSB3gk5AqQ203skcdZYXTvRS1ocDllDUMBeV6zPYdpw=Nt/jpvrRudQBnn7HWJ2lwcR7YR/k/E2knVD8nMMqphkTE0anTjnn91xTfubGYu9L9bSH1rp/5emScXnmbhNnJIhqPqYn5PQ7tZCpTYOHSY8A2V908Zs7Gwb6ah21eGTnuBYLRGmGbc0wa2JNpOR2L9AgejYKHFNUPp9tGO2SoJs=AFjjzp10VlNDphkGpKghU0iy1NHqk76rbzKUYA6qAzD+VVoFpEN7X3LIpdSNTfMFOqYctc4o+OK+Ja/bf8rCl7ZAZ5iKp9EBrbGjxQy/tuvBdg/wshsJBAWnAEzOfoeucLW6dMzogSKopHlE/hiZ0VJSMcBFz7Syfyg/w9XGLDI=QVm/Eu4ECn1lV5VMjO1Oje2zp89b8NKlosUCyic5vWN6j/2HecXSeL/OqLhG+T16Q5ctNh9EZu9bfDExb76xNKhSVImapGaMKf5c/7dH+JQkmyDnnFJMePuJd5knArKeNQaNJqdN9RotffahfXXSSFJNJQ0Ejjag2h0SCBmZ2r8=Q1NI6q9AvkWHgTKzInx51Wx/+VeVuGPNik4swlRMKn2uFMrcdGjAM/NBBiCUTbu182Xyxr2gWCGyCLLs80T/+wKvtTP6CLDy3+MiOf7hO3NnfEHYcjTmNKYRNMenqIhOTQQKH/CL+dnVFcX0E34STJgLc9ONgq/dp+dIEEK7+RI=i3xRudjDkvHeEVJkNZMJOk0K8xhlnXGT8aqJp1ndYuz2fcFz7WUa+m7XVANvzD0TXvnSx1HVujgsjSs4oKlyv4DPdt86aisY4raXA+PxoZF92z53NmsUT7HFL/WzH9vbWPBvYVNCd8ar3ZhlEMR/IqynQLuDRwVZ3paFL03YIc4=

# Annexe C: Chiffrement asymétrique RSA avec le module crypto

/\*\*

\* Fonction pour coder le text avec une clé publique

\* @reference http://stackoverflow.com/questions/21951304/encrypting-and-decrypting-string-using-ursa-with-nodes-throws-decoding-error

\* @param {String} plainText text non-codé

\* @param {String} relativePathtoPrivateKey chemin relatif de la clé publique

\* @return {String} le text codé

\*/

var encryptStringWithRsaPublicKey = function(plainText, relativePathToPublicKey) {

var key = fs.readFileSync(relativePathToPublicKey, "utf8");

var publicKey = {'key':key,'padding':1};

var buffer = new Buffer(plainText);

var maxBufferSize = 117;

var encryptedBuffersList = [];

var bytesDecrypted = 0;

//boucle à travers toutes les données dans buffer morceau par morceau(chunk by chunk)

while(bytesDecrypted < buffer.length){

//calculer la taille du buffer temporaire et créer ce tampon(buffer)

var amountToCopy = Math.min(maxBufferSize, buffer.length - bytesDecrypted);

var tempBuffer = new Buffer(amountToCopy);

//copies next chunk of data to the temporary buffer

buffer.copy(tempBuffer, 0, bytesDecrypted, bytesDecrypted + amountToCopy);

//encrypter et stocker le morceau courrant(chunk)

var encrypted = crypto.publicEncrypt(publicKey, tempBuffer);

encryptedBuffersList.push(encrypted.toString('base64'));

bytesDecrypted += amountToCopy;

}

//coller tous les tampons encryptés et retourner une string

return encryptedBuffersList.join("");

};

function chunk(str, size) {

return str.match(new RegExp('.{1,' + size + '}', 'g'));

}

/\*\*

\* Fonction pour décoder le text codé avec une clé privée

\* @param {String} cipherText text non-codé

\* @param {String} relativePathtoPrivateKey chemin relatif de la clé privée

\* @return {String} le text décodé

\*/

var decryptStringWithRsaPrivateKey = function(cipherText, relativePathtoPrivateKey) {

var key = fs.readFileSync(relativePathtoPrivateKey, "utf8");

var privateKey = {'key':key,passphrase:KEY,'padding':1};

var maxBufferSize = 172;

var stringChunkList = [];

var decryptedBuffersList = [];

stringChunkList = chunk(cipherText,maxBufferSize);

var totalBuffers = stringChunkList.length;

for(var i = 0 ; i < totalBuffers; i++){

//copies next buffer chunk to be decrypted in a temp buffer

var data = stringChunkList[i].toString('utf8');

var tempBuffer = new Buffer(stringChunkList[i],'base64');

//décrypter et stocker le morceau courrant(chunk)

var decrypted = crypto.privateDecrypt(privateKey, tempBuffer);

decryptedBuffersList.push(decrypted);

}

//coller tous les tampons décryptés et retourner une string

return Buffer.concat(decryptedBuffersList).toString("utf8");

};