

## Telecoms Assignment 2

14315135

### Introduction:

Approaching this assignment I found the idea of it quite difficult, however after a bit of research on the topic I found that there were multiple ways to accomplish this, some being more simple than others. I decided to make a chrome extension without using local storage memory in case for if it actually were to be deployed to the chrome web store, it would have all the same functionality by just changing the localhost links in my code to the host server.

After looking into different ways of encryption, I decided against openpgp, to use an S/MIME encryption library called cryptico instead. I also realised there were many ways in which the data could be read from the server, I decided to use json files as they are native to javascript which was the main language that I was using.

On the adding and deleting front, I decided to use php as it was a server side language that I was familiar with, and it had some handy methods for writing to and parsing json files.

### Encryption:

So in order to encrypt, decrypt and sign my emails I used the encryption library Cryptico. It is a very well documented encryption library built for javascript. How I encrypted the files was to decide on the number of bits I wanted the RSA Key to be, I used 1024, and have each user create a passphrase for their private key. Once this was set I could create all of the private keys by using

```
var RSAKey = cryptico.generateRSAKey(passPhrase,bits);
```

And from here I could create a public key from each user by writing:

```
var PublicKeyString = cryptico.publicKeyString(RSAkey);
```

Once this was done, it was very simple to encrypt and sign any email by encrypting with the recipients public keys and signing the cipher with the senders RSA Keys.

To decrypt, I then run the decryption using the recipients RSAKey.

In order to find the data I need for these, I have the json file being parsed with jquery. This loads the elements of the json file into an array containing the email, public key and passphrase. For encryption, I pull the recipients email from the compose view on gmail, I then check this against my session json file, if it finds the email then it check for their public key to encrypt the message with. If its not found it throws a simple error and will not encrypt the message.

\*\*\*\*\* Recipient has no public key\*\*\*\*\*


ksjrgb


Callum Duffy


ksjrgb


TEST EMAIL|


Send





















ksjrgb


Callum Duffy


ksjrgb


mLkKh0NrOcVYhMwStKIQmuhkdqaOHyb5+zGreJeGFIMiM53dpn4/zvLa+XBCsWQM9qmlTwc5AKDIPQPIb+IZ1gr/Sg2a+w/qzFi2dFGEipDQYtKQEAxwjPk8BOgF2NSogNx3CcgeJlvrVr8cexRXJyGHtslxELBgSnYqNebuBQg=?  
sawrm/6NSG2SGJuaoJtlzPoTVn7rLl6bz2VuKdAkI29BWGSQX3C51644ntSeBZK/FXXZNOEBICKDAMEFcGkzbxiv/4jQQG3A17j+6DxDCpw6eFpTKQwm/jFeBQFSAVjCsMqOq6narL5JzPtloobUQ7ur19y03r8kSXubp9dxs/OX+B7Hvi3Po18PgPNeX7xmjiIa8AD84SOFhqQcORs7555YncpLasQyyZF8Zqi9RDo8zFEY90Xodip8ncVmvmtcbfojklIMER5hRnW/yulieVhj4dOKYeanXsx7OtQWEMv7b91aBURM7jMNRsW0m9nDavW0fkHUdp0TnzZnhhLjscQwIkOe9HOOfhujaaIw4+62/vvSgenxIhHWyp6+Lmv98YrjNZf90I/pODBA/wKLWmUOdTGI9aUVcVWgYd0HPvz6tAfWkzCAH5J05/k7eLwnrcrVIW0g0UmlW0Nml44VCa2sV6w1BIFZLlogHZwZq6R5U6wreuqwxY5QNfBOLpGr68aobCuJ334IS8OEnhjCqOW4uT9XKxYnVQh95xdR0OZWWZHhvNKjCSp4OpckOv9GxmcQXoFFC8u7khDROhjmH3XA==


Send






















(In the above message, the highlighted lock button in the compose view encrypts the message).

For decryption I do a very similar thing, checking for the email in the json file, if present it will pull their passphrase and generate their rsa key to decrypt the message. This was only done this way for testing on one machine without networking, if I could I would have had each user have their own private key store locally on their machine and then only their email and public key stored on the server, however for testing purposes this wasn't practical.




For decryption I have a handler that runs whenever the thread view of gmail is opened, when this happens it loads the messages into an array and then uses this to obtain information. I have a sidebar, which displays the decrypted message if the email of the sender is in the json file and if the recipients public key is correct. The decrypt method also checks whether this message has been signed, and if so prints that it is validated or unsigned. It does so by checking the certificate of the sender in the message via verifying that they have signed with their private rsa key.

laefhjkj

Inbox x



Decrypted Message

**Callum Duffy** <duffyc8@tcd.ie> 09:35 (30 minutes ago) ☆  

to me ▾

gQ3OTHzTyOmHG4G0HOsxUtC/CnxyMst1Fp4YY+VDGHWov/  
6mukMGFxV1qi5+g8tGuR6FIW37eWZyxc75TjyiUnPEg  
o/TOY5PDjtwrzloi3+9Ky6JxL7KLwfNRIs+  
6JLgyCsR7gX9zDJFI7ThfvW1uQfltU+gDXDIZl7iL2OkqA=?  
qGBfkuRq07v0S4Ymmn2Xc3P5ukgmNEmQWqSoJiglggNw5KDd19+  
Wcdeax4v7Z3uwvDC1PL6MyS4JGpWuv0s9CPM2+  
2jirT5dWvhZpbnd4TOBEQvWiFoulsGKimEcvqBSSflN5KglCYVza9U08x5n3  
JAX1b3jtA37fpOVHjVjIdPa16Pevf6wa0lloFrMPKyRGwg+  
RdEJiv90D1jMJYOZXf8GfW7as4sSbr+Db8T9tZ8nNRPgCi2D3c27Mc76Z/  
hS8d+EtPCgHLRyLtDOSJiWmYgmuA4z2a+tMX4QqfDq/veJhReY55l7J+  
Vy6VIHZOrQaZTSxjDFSLNi9s5A0XMr  
5KRQvjmeRLmkcKY07L5ZMilXfubeLmW823F4GY/  
0SuEcPd9NFPFLMeSk7klIGU0GiKi+  
pSGIM8L4J2R6R8HsQJgBSy9qZNJzug  
mYWSdnJotafWNlrAk6Jdp7ACivVUEzTKYXjaah/  
4HhvCagAy1rUvLEJmTV4TyOafR/Olev4vFsnuTXcRq1NMbnMsAiaGrt3G  
au84c90wKOvrDPKCPxQTK5Xy2jErjv4JhH9dNBb+  
jpqUJpryQzbHYNsHl6cSpyg==

TEST EMAIL And the signature was:  
verified

### Users:

In order to add the users to the session, I am running a php script which is currently located on my local host, this has a form containing email, public key and private key fields which adds them to the json file on the localhost using php's simple json functions.

# Mail Encryption

Email:

Public key:

Private key:

Submit

This is what allows my javascript credential checks to run, in order to allow it to check whether the encryption can be done by pulling their public keys and checking for the private key in the decryption of the cipher text. To check whether the user is in session or has the correct keys in order to encrypt or decrypt the message. Once new credentials are added, emails can be securely sent to this user as they are added to the group.

## Code files:

Main.js

```
InboxSDK.load('1.0', 'sdk_14315135_2d9300ddff').then(function(sdk){

    var myArr;

    //handler for the compose section of mail

    //used for encryption

    sdk.Compose.registerComposeViewHandler(function(composeView){

        var bits = 1024;

        var PassPhrase = "Hello";

        var RSAKey = cryptico.generateRSAKey(PassPhrase, bits);

        var PublicKeyString = cryptico.publicKeyString(RSAKey);

        console.log(PublicKeyString);

        //xmlhttp request to create array from reading json
```

```

//json read from local host
var xmlhttp = new XMLHttpRequest();
var url = "http://localhost/keys.json";

//check request is in correct state
xmlhttp.onreadystatechange = function() {
if (this.readyState == 4 && this.status == 200) {
    //parse json file
    myArr = JSON.parse(this.responseText);
}
};

//send request
xmlhttp.open("GET", url, true);
xmlhttp.send();

//Encrypt Button
composeView.addButton({
    title: "Encrypt Message",
    imageUrl : 'https://image.freepik.com/free-icon/lock-button_318-36478.jpg',
    onClick: function(event) {
        //get email of recipient of email
        var recips = composeView.getToRecipients();
        var recip = recips[0].emailAddress;
        //query json for this recipient
        //console.log(myArr);

        if(recip === myArr.keys.email){
            //result from query for public key
            var content = composeView.getTextContent();
            var pass = myArr.keys.priv;
            var PublicKeyString = myArr.keys.pub;

```

```

        var RSAKey = cryptico.generateRSAKey(pass,bits);
        var encrypted = cryptico.encrypt(content,PublicKeyString,RSAKey);
        event.composeView.setBodyText(encrypted.cipher);
    }
    else{
        event.composeView.insertTextIntoBodyAtCursor("\n ***** Recipient has no public key*****
\n");
    }
}
});
});

```

//handler to manage the threads, to view messages

```

sdk.Conversations.registerThreadViewHandler(function(threadView){
    var bits = 1024;

```

//getting email info for queries

```

var mviews = threadView.getMessageViews();
var contact = mviews[0].getSender();
var email = contact.emailAddress;

```

//get message text to decrypt

//horrible scraping but it does the job

```

var bodyEl = mviews[0].getBodyElement();
var inner = bodyEl.innerHTML;
var string = "";
string = inner.replace(/\<wbr>/g,"");
var index = string.indexOf("ltr")+5;
string = string.slice(index);
var end = string.indexOf("</div>");
string = string.substring(0,end);

```

```

//query priv key from db
var pass;
var message;
var rkey;
//query for the email in the json
if(myArr.keys.email === email){
    pass = myArr.keys.priv;
    var RSAKey = cryptico.generateRSAKey(pass,bits);
    message = cryptico.decrypt(string,RSAKey).plaintext;
    message += "\nAnd the signature was: " + (cryptico.decrypt(string,RSAKey)).signature;
}
else{
    message = "Private key incorrect, cannot decrypt message";
}

//add text to the html element for decrypted message
var el = document.createElement("div");
el.innerHTML = message;

//sidebar for the decrypted message to be displayed
threadView.addSidebarContentPanel({
    title: 'Decrypted Message',
    el: el
});

});
});

```

Test.php

```
<!DOCTYPE html>
```

```
<?php
```

```
    $serverName = "localhost";
```

```
    $username = "root";
```

```
    $password = "";
```

```
    $dbname = "keys";
```

```
    $email = htmlspecialchars($_POST['email']);
```

```
    $pubkey = htmlspecialchars($_POST['public']);
```

```
    $privkey = htmlspecialchars($_POST['private']);
```

```
?>
```

```
<?php
```

```
    $file = "keys.json";
```

```
    $json = json_decode(file_get_contents($file),TRUE);
```

```
    echo "$email";
```

```
    $json["keys"] = array("email"=> $email, "pub" => $pubkey, "priv" => $privkey);
```

```
    file_put_contents($file, json_encode($json));
```

```
?>
```

```
<html>
```

```
    <body>
```

```
        <h1>Mail Encryption</h1>
```

```
        <form method = "post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

```
            Email:<br><input type="text" name="email" value=""><br>
```

```
            Public key:<br><input type="password" name="public" value=""><br><br>
```

```
            Private key:<br><input type="password" name="private" value=""><br><br>
```

```
                <input type="submit" value="Submit">
```

```
        </form>
```

```
    </body>
```

```
</html>
```



Manifest.json

```
{
  "manifest_version": 2,

  "name": "Mail Encryption",
  "description": "Encrypting and decrypting emails",
  "version": "1.0",
  "permissions": [
    "activeTab",
    "https://ajax.googleapis.com/",
    "https://mail.google.com/",
    "https://inbox.google.com/"
  ],
  "content_scripts" : [
    {
      "matches": ["https://mail.google.com/*", "https://inbox.google.com/*"],
      "js": ["inboxsdk.js", "main.js", "cryptico-master/cryptico.js", "jquery-3.2.0.min.js"]
    }
  ],

  "browser_action": {
    "default_icon": "icon.png",
    "default_popup": "http://localhost:80/test.php",
    "default_title": "Encrypt that shit!"
  }
}
```

Keys.json

```
{"keys":{"email":"duffyc8@tcd.ie","pub":"tCmfUlJw6p9zxm85mFKL1vyxsm+CH7VhpHofeN4Ca33wBdtVLHN1pRY4tkU4LZGUj5ohVRepl26BtJp2nl6abf8XmD7qOCAPaM3wTsxW9HYqa\Xbr7h7nTROWJwuS7sAcN7WZnm28LeWNNdxK0gVNutAk0Phao1cea4LaMze0Ys=","priv":"Hello"}}
```