Securing a social media application

Dylan Lewis

Introduction

For this assignment we tasked with creating something allows users to encrypt and decrypt messages put on a social media website be it twitter, facebook, etc. But it must also allow for the user to create and delete groups and to add and remove users from these groups. For this I chose to build a chrome extension for reddit.

Key Management Server

The key management server was implemented in node js. It uses the http, cryptico and crypto-js library. For this system the user must keep track of existing groups and users in groups. To do this groups are stored as a javascript object which is a hashmap, where the keys are the group names and they map to another javascript object. This javascript object maps a username to that user's role within the group, admin or regular user and the user's public key. The server also has a public private key pair to allow for encrypted message to be sent to server. The server receives requests and each request must have the action field which specifies what operation the server must perform. The actions the server can perform are get server public key, add user to group, remove user from group, encrypt, decrypt, make group and delete group. When the make group command is made, the user must supply their public key and username, then it creates a javascript object which maps the username to it's public key and it's role, which in this case is set to admin. Then it generates a random aes key using crypto-js library which it will use to encrypt all of the groups' messages with.

For deleting a group it checks that the user issuing the command is an admin in the group then it will just remove the group from the javascript object. It will also delete the aes key for that group meaning that old messages cannot be recovered. Next when adding a user to the group first it must be an admin adding the user, the admin must also supply the username and the public key of the user being added to the group. This will allow them to see messages now. For removing the user from a group it checks if the user sending the request is

an admin and if the user to be deleted is in the group. If they're in the group then they are deleted from it and unable to see group's messages anymore.

For encryption the user sends the plaintext encrypted with the server's public key to the server along with the group they want to be able to see it. The server then checks if the group is valid and if the user is in the group. Then it decrypts the plaintext with it's private key and encrypts it again with the aes key of the group and sends this ciphertext back to the user. We use the aes 256 bits with cipher block chain mode as is it secure and prevents unwanted users from accessing it. This aes key is never seen by the user and is only ever used by the server and never sent to anyone so it cannot be gotten by anyone.

For decryption the user sends the ciphertext to the server along with the group's name and the username. It checks that the group is valid and that the user is in the group. The ciphertext is then decrypted with the aes key and the resulting plaintext is encrypted with the user's public key and sent back to the user. This ensures that only that user can get the message as even if non-user gets the message they need the user's private in order to decrypt it.

Chrome Extension

The chrome extension makes the login with a username and password. This will then store the username and rsa key in the chrome storage system. When the user is logged in it allows them to make a group, delete a group, add user to a group, remove user, get their public key and logout. For making a group the chrome extension takes a group name and since all the other necessary parameters are in the chrome storage then it sends the correct parameters to the server and prints out the servers response. The extension does this for all options. Upon logging out the details are removed from the chrome storage system.

The extension also adds an option on reddit post pages to encrypt the reddit post. It does this by having an input box to specify the group this is message if for and an encrypt button which takes the text the username and the server's public key and encrypts the plaintext with the public key and sends it to the server for encryption with the username and the group's name. Then the server's response is automatically entered into the post's text box. The user is then able to submit the encrypted post to the subreddit.

For decrypting a reddit post a button is added to the options list of any reddit post along with an input box which allows the user to specify which group this encrypted message belongs to. When the group is specified and the decrypt button is pressed the ciphertext is sent to the server along with the username group gotten from the input for decryption. The server's response is decrypted using the user's private key and replaces the ciphertext in the post. If the

user cannot decrypt the message then the message is replaced with the word "undefined".

Installation

- Install node js
- Install npm Then do following commands

```
npm i cryptico

npm i crypto-js

npm install -g browserify

browserify popup.js > popup_bundle.js

browserify content.js > bundle.js

browserify decrypt.js > bundle2.js
```

Code Snippets

server.js

```
const http = require('http');
const fs = require('fs');
const url = require('url');
const cryptico = require('cryptico');
const CryptoJS = require("crypto-js");

const Bits = 2048;
const PassPhrase = "Veteran of a thousand psychic wars";
const RSAKey = cryptico.generateRSAKey(PassPhrase, Bits);

const SendPubKey = 100;

const CreateGroup = 202;

const DeleteGroup = 205;

const AddUserToGroup = 206;
const RemoveUserFromGroup = 207;

const Encrypt = 300;
const Decrypt = 301;
```

```
var Groups = {};

var keys = [];
function getUrlVars(url) {
    var hash;
    var myJson = {};
    var hashes = url.slice(url.indexOf('?') + 1).split('&');
    for (var i = 0; i < hashes.length; i++) {
        hash = hashes[i].split('=');
        myJson[hash[0]] = hash[1];
    }
    return myJson;
}
server = http.createServer( function(req, res) {

    //console.dir(req.param);

    if (req.method == 'POST') {
        console.log("POST");
    }
    else
    {
        console.log("GET");
        var responseText = "Invalid Action";

        var newDict = url.parse(req.url, true).query;
    console.log(JSON.stringify(newDict));
        if ("action" in newDict){
            console.log(newDict['action']);
            if (newDict['action'] == SendPubKey){
                responseText =  cryptico.publicKeyString(RSAKey);
            }
            else if(newDict['action'] == CreateGroup && "user" in newDict && "pubKey" in new
        if (newDict['group'] in Groups) {
          responseText = newDict + " already exists";
        }
        else {
          users = {};
                users[newDict['user']] = {'role':'admin'};

                admin = users[newDict['user']];

                admin['key'] = newDict['pubKey'];
                Groups[newDict['group']] = {};
                Groups[newDict['group']]['users'] = users;
```

```
        var key = new Buffer(cryptico.generateAESKey());
        key = key.toString();
        Groups[newDict['group']]['key'] = key;
  console.log("Groups: " + JSON.stringify(Groups));
        responseText = "Created group " + newDict['group'];
}

    }
    else if(newDict['action'] == AddUserToGroup){
            if('admin' in newDict && 'group' in newDict  && 'user' in newDict && 'ke
                var user = newDict['user'];
                var admin = newDict['admin'];
                var group = newDict['group'];
                var key = newDict['key'];
                if(group in Groups && admin in Groups[group]['users'] && Groups[grou
                    var users = Groups[group]['users'];
                    users[user] = {'role':'user','key':key};
      console.log("users: " + JSON.stringify(users));
                    responseText= user + " added to " + group;
                }
                else{
                    responseText = "Not admin";
                }
            }
            else{
                responseText = "Invalid parameters";
            }
    }
    else if(newDict['action'] == RemoveUserFromGroup){
            if('admin' in newDict && 'group' in newDict  && 'user' in newDict){
                var user = newDict['user'];
                var admin = newDict['admin'];
                var group = newDict['group'];

                if(group in Groups && admin in Groups[group]['users'] && Groups[grou
                    var users = Groups[group]['users'];
                    delete users[user];
      console.log("users: " + JSON.stringify(users));
                    responseText= user + " removed from " + group;
                }
                else{
                    responseText = "Not admin";
                }
            }
            else{
                responseText = "Invalid parameters";
```

```
        }
}
else if(newDict['action'] == DeleteGroup){
    if('group' in newDict && 'admin' in newDict){
        var group = newDict['group'];
        var admin = newDict['admin'];
        if(group in Groups){
            if( admin in Groups[group]['users'] && Groups[group]['users'][admin]

                delete Groups[group];
  console.log("Groups: " + JSON.stringify(Groups));

                responseText = group + " has been deleted";
            }
            else{
                responseText = "Not a valid admin";
            }
        }
        else{
            responseText = "Not valid group";
        }
    }
    else {
        responseText = "Not enough parameters for request";
    }
}
else if (newDict['action'] == Encrypt && 'user' in newDict && 'group' in newDict
    var group = newDict['group'];
    if(group in Groups && newDict['user'] in Groups[group]['users']){
        var str = cryptico.decrypt(newDict['text'], RSAKey).plaintext;
        var key = Groups[group]['key'];
        var result = CryptoJS.AES.encrypt(str, key).toString();
        console.log("Encrypted text: " + result);
        responseText = result;
    }
    else {
        console.log(" group details");
        responseText = "invalid details";
    }
}
else if (newDict['action'] == Decrypt && 'user' in newDict && 'group' in newDict
    var group = newDict['group'];
    var user = newDict['user'];
    if(group in Groups && newDict['user'] in Groups[group]['users']){
        console.log("Decrypt");
        var str = newDict['text'];
```

```
                    console.log(str);
                    var AESKey = Groups[group]['key'];

                    var pubKey = Groups[group]['users'][user]['key'];
                    var result = CryptoJS.AES.decrypt(str, AESKey).toString(CryptoJS.enc.Utf
                    result = cryptico.encrypt(result, pubKey).cipher;
                    responseText = result;
                    console.log("--------------");
                }
                else {
                    responseText = "invalid details";
                }
            }

        }
        res.writeHead(200, {'Content-Type': 'text/html'});
        res.end(responseText + " ");
    }

});

port = 3000;
host = '127.0.0.1';
server.listen(port, host);
console.log('Listening at http://' + host + ':' + port);
```

content.js

```
ar cryptico = require("cryptico");


var getKey = new  XMLHttpRequest();
getKey.open("GET", "http://localhost:3000?action=100", false);
getKey.send();
cryptico;
var key = getKey.responseText;
console.log("Server key is " + key);
var btn = document.createElement("BUTTON");// Create a <button> element
btn.setAttribute("type", "button");
btn.setAttribute("id", "myBtn");
var t = document.createTextNode("Encrypt");       // Create a text node
btn.appendChild(t);
var userInp  = document.createElement("INPUT");// Create a <button> element
userInp.setAttribute("type", "text");
userInp.setAttribute("value", "user");
```

```
userInp.setAttribute("id", "username-field");

var groupInp = document.createElement("INPUT");// Create a <button> element
groupInp.setAttribute("type", "text");
groupInp.setAttribute("value", "group");
groupInp.setAttribute("id", "groupname-field");


document.getElementById("text-field").appendChild(btn);
document.getElementById("text-field").appendChild(userInp);
document.getElementById("text-field").appendChild(groupInp);

document.getElementById("myBtn").addEventListener("click", function(){

    var user = document.getElementById("username-field").value;
    var group = document.getElementById("groupname-field").value;

    var text = document.getElementsByName("text")[0].value;
    var cipher = cryptico.encrypt(text, key).cipher;
    var xhr = new XMLHttpRequest()
    xhr.open("GET", "http://localhost:3000?action=300&user=" + user + "&group=" + group  +"&
    xhr.send();
    var result = xhr.responseText;
    console.log(result);
    if(result == "invalid details "){
        alert("Invalid details for encryption");
    }
    else{
        document.getElementsByName("text")[0].value = result;
    }
});
```

decrypt.js

```
var cryptico = require('cryptico');
function RSAParse(rsaString) {
    var json = JSON.parse(rsaString);
    var rsa = new cryptico.RSAKey();

    rsa.setPrivateEx(json.n, json.e, json.d, json.p, json.q, json.dmp1, json.dmq1, json.coef

    return rsa;
}
function decrypt(group){
```

```
  chrome.storage.local.get(['user'],  function(result){
    if (result['user'] != undefined && result['user'] != null) {
      var user = result['user']['user'];
      var rsaKey = RSAParse(result['user']['key']);

      var getPost = document.getElementsByClassName("md")[1];
      var post = getPost.getElementsByTagName('p')[0].innerHTML;
      var xhr = new XMLHttpRequest();
      xhr.open("GET", "http://localhost:3000?action=301&user="+ user +"&group=" + group + "&
      xhr.send();
      var resp = xhr.responseText;
      var result = cryptico.decrypt(resp, rsaKey).plaintext;
      document.getElementsByClassName("md")[1].getElementsByTagName('p')[0].innerHTML = resu
      document.getElementById("groupname-field").value = "";
  }
  else {
    alert("Please sign in");
  }
});
}
var groupInp = document.createElement("INPUT");// Create a <button> element
groupInp.setAttribute("type", "text");
groupInp.setAttribute("value", "group");
groupInp.setAttribute("id", "groupname-field");
document.getElementsByClassName("top-matter")[0].appendChild(groupInp);


var decryptBtn = document.createElement("LI");
decryptBtn.setAttribute("id", "decryptBtn");
decryptBtn.setAttribute("class", "crosspost-button");
var a = document.createElement("A");
a.setAttribute("id", "decryption");


a.innerHTML= "decrypt";
decryptBtn.appendChild(a);


document.getElementsByClassName("flat-list buttons")[0].appendChild(decryptBtn);
document.getElementById("decryption").addEventListener("click", function(){
  var group = document.getElementById("groupname-field").value;
  decrypt(group);

});
// var rsaKey = cryptico.generateRSAKey(PassPhrase, 2048);
```

```
// var getPost = document.getElementsByClassName("md")[1];
// var post = getPost.getElementsByTagName('p')[0].innerHTML;
// var xhr = new XMLHttpRequest();
// xhr.open("GET", "http://localhost:3000?action=301&user=test&group=test&text=" + encodeURI
// xhr.send();
// var resp = xhr.responseText;
// console.log(resp)
// var result = cryptico.decrypt(resp, rsaKey).plaintext;
// console.log(result);
// document.getElementsByClassName("md")[1].getElementsByTagName('p')[0].innerHTML = result;
```

popup.js

```
// Copyright 2018 The Chromium Authors. All rights reserved.  Use of
// this source code is governed by a BSD-style license that can be
// found in the LICENSE file.
var cryptico = require('cryptico');
'use strict';
function loadPage(href) {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("GET", href, false);
    xmlhttp.send();
    return xmlhttp.responseText;
}

function RSAParse(rsaString) {
    var json = JSON.parse(rsaString);
    var rsa = new cryptico.RSAKey();

    rsa.setPrivateEx(json.n, json.e, json.d, json.p, json.q, json.dmp1, json.dmq1, json.coef

    return rsa;
}
function serverReq(data){
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("GET", "http://localhost:3000/?" + data, false);
    xmlhttp.send();
    return xmlhttp.responseText;

}

function back(){

    document.getElementById("action").innerHTML=loadPage("action.html");
    document.getElementById("login").onclick = login_page;
```

```
    document.getElementById("mkgrp").onclick = mkgrp;
    document.getElementById("rmgrp").onclick = rmgrp;
    document.getElementById("delgrp").onclick = delgrp;
    document.getElementById("adduser").onclick = addgrp;
    document.getElementById("logout").onclick = logout;
    document.getElementById('pubKey').onclick = getPublicKey;
}

function login_page(){
    document.getElementById("action").innerHTML=loadPage("login.html");
    document.getElementById('loginForm').onclick = login;
    document.getElementById("back").onclick = back;

}

function mkgrp(){
    document.getElementById("action").innerHTML=loadPage("mkgrp.html");
    document.getElementById('groupForm').onclick = createGroup;
    document.getElementById("back").onclick = back;

}

function createGroup(){
    var group = document.getElementById("groupname").value;
    if(group != ""){
        chrome.storage.local.get(['user'], function(result){
            if (result['user'] != undefined && result['user'] != null) {
                var user = result['user']['user'];
                var key = cryptico.publicKeyString(RSAParse(result['user']['key']));

                var data = "action=202&user="+ user + "&pubKey=" + encodeURIComponent(key) +
                alert(serverReq(data));
                back();
            }
            else {
                alert("Please login");
                login_page();
            }
        });
    }
    else {
        alert("Please enter group name");
        mkgrp();
    }

}
```

```
function rmgrp(){
    document.getElementById("action").innerHTML=loadPage("rmgrp.html");
    document.getElementById("removeForm").onclick = removeFromGroup;
    document.getElementById("back").onclick = back;

}
function removeFromGroup(){
    var user = document.getElementById("username").value;
    var group = document.getElementById("groupname").value;
    if(user != "" && group != ""){
        chrome.storage.local.get(['user'], function(result){
            if (result['user'] != undefined && result['user'] != null) {
                var admin = result['user']['user'];
                var data = "action=207&admin="+ admin + "&group=" + group +
                "&user=" +user;
                alert(serverReq(data));
            }
            else {
                alert("Please login");
                login_page();
            }
        });

    }
    else {
        alert("Please fill out all fields");

    }
}
function addgrp(){
    document.getElementById("action").innerHTML=loadPage("adduser.html");
    document.getElementById("removeForm").onclick = addToGroup;
    document.getElementById("back").onclick = back;

}
function addToGroup(){
    var user = document.getElementById("username").value;
    var group = document.getElementById("groupname").value;
    var key = document.getElementById("pubKey").value;
    if(user != "" && group != ""){
        chrome.storage.local.get(['user'], function(result){
            if (result['user'] != undefined && result['user'] != null) {
                var admin = result['user']['user'];
                var data = "action=206&admin="+ admin + "&group=" + group +
                "&user=" +user + "&key=" + encodeURIComponent(key);
```

```
                alert(serverReq(data));
            }
            else {
                alert("Please login");
                login_page();
            }
        });

    }
    else {
        alert("Please fill out all fields");

    }
}

function delgrp(){
    document.getElementById("action").innerHTML=loadPage("delgrp.html");
    document.getElementById("delForm").onclick = deleteGroup;
    document.getElementById("back").onclick = back;
}

function deleteGroup(){
    var group = document.getElementById("groupname").value;
    if(group != ""){
        chrome.storage.local.get(['user'], function(result){
            if (result['user'] != undefined && result['user'] != null) {
                var user = result['user']['user'];
                var data = "action=205&admin="+ user + "&group=" + group;
                alert(serverReq(data));
                back();
            }
            else {
                alert("Please login");
                login_page();
            }
        });
    }
    else {
        alert("Please enter group name");
        mkgrp();
    }

}
```

```
function logout(){
    chrome.storage.local.set({'user':null}, function(){
        alert("You have been logged out");

    });
    chrome.storage.sync.set({'user':null}, function(){

    });
    back();
}

function login(){
    console.log("got here");
    var user = document.getElementById("username").value;
    var pass = document.getElementById("passphrase").value;
    if (user == null || user=="" || pass==null || pass==""){
        login_page();
        return null;
    }
    var username ="";
    var passphrase = "";
    chrome.storage.local.get(['user'], function(result) {

        username = user;
        passphrase = pass;
        var key = JSON.stringify(cryptico.generateRSAKey(passphrase, 2048).toJSON());
        var details = {'key': key, 'pass':passphrase, 'user':username};
        chrome.storage.local.set({'user':{'key':key,'pass':pass,'user':user}},function() {
            console.log("Logged in");
            back();
        });
  });
}

function getPublicKey(){
    chrome.storage.local.get(['user'], function(result){
        if (result['user'] != undefined && result['user'] != null) {
            alert(cryptico.publicKeyString(RSAParse(result['user']['key'])));
            back();
        }
        else {
            alert("Please login");
            login_page();
        }
    });
```

```
}

back();
```

manifest.json

```
{
  "name": "Encryption extension",
  "version": "1.0",
  "description": "Allows users to decrypt and encrypt their social media messages.",
    "permissions": ["tabs","activeTab", "declarativeContent", "storage",
                    "http://localhost:3000/*",
                   "https://github.com/wwwtyro/cryptico/blob/master/cryptico.js"],
    "content_scripts":[
        {
            "matches":["https://www.reddit.com/r/*/submit*"],
            "js": ["bundle.js"]
        },
        {
            "matches":["https://www.reddit.com/r/*/comments/*"],
            "js": ["bundle2.js"]
        }
    ],
  "background": {
    "scripts": ["background.js"],
    "persistent": false
  },
  "browser_action": {
    "default_popup": "popup.html",
    "default_icon": {
      "16": "images/get_started16.png",
      "32": "images/get_started32.png",
      "48": "images/get_started48.png",
      "128": "images/get_started128.png"
    }
  },
  "icons": {
    "16": "images/get_started16.png",
    "32": "images/get_started32.png",
    "48": "images/get_started48.png",
    "128": "images/get_started128.png"
  },
  "manifest_version": 2
}
```

popup.html

```
<!DOCTYPE html>
<html >
  <head>
    <style>
      body{
      width:200px;
      text-align:center;
      }
      button{
      width:100px;
      }
    </style>
  </head>
  <body >
    <div id="action">
    <button type="button" id="login">Login</button>
    <button type="button" id="adduser">Add user to group</button>
    <button type="button" id="rmgrp">Remove from group</button>
    <button type="button" id="mkgrp">Make Group</button>
    <button type="button" id="delgrp">Delete Group</button>
  <button type="button" id="pubKey">Get Public key</button>
    <button type="button" id="logout">Logout</button>
    <script src="popup_bundle.js"></script>
    </div>
  </body>
</html>
```

action.html

```
<button type="button" id="login">Login</button>
<button type="button" id="adduser">Add user to group</button>
<button type="button" id="rmgrp">Remove from group</button>
<button type="button" id="mkgrp">Make Group</button>
<button type="button" id="delgrp">Delete Group</button>
<button type="button" id="pubKey">Get Public key</button>
<button type="button" id="logout">Logout</button>
```

adduser.html

```
User:
<input type="text" id="username" value="">
<br>
```

```
User's key:
<input type="text" id="pubKey" value="">
<br>
Group:
<input type="text" id="groupname" value="">
<br>
<button type="button" id="removeForm">Submit</button>
</div>
<button type="button" id="back">back</button>
```

delgrp.html

```
Group:
<input type="text" id="groupname" value="">
<br>
<button type="button" id="delForm">Submit</button>
</div>
<button type="button" id="back">back</button>
```

login.html

```
User:
<input type="text" id="username" value="">
<br>
Pass:
<input type="password" id="passphrase" value="">
<br>
<button type="button" id="loginForm">Submit</button>
</div>
<button type="button" id="back">back</button>
```

mkgrp.html

```
Group:
<input type="text" id="groupname" value="">
<br>
<button type="button" id="groupForm">Submit</button>
<button type="button" id="back">back</button>
```

rmgrp.html

```
User:
<input type="text" id="username" value="">
<br>
Group:
<input type="text" id="groupname" value="">
<br>
<button type="button" id="removeForm">Submit</button>
</div>
<button type="button" id="back">back</button>
```

images/get_started16.png

Figure 1: icon

images/get_started32.png

Figure 2: icon

images/get_started48.png

Figure 3: icon

images/get_started128.png

Figure 4: icon