Encrypted Messenger and Multiple Authentication
(InSilent)

Team Name: Encrypted Wecooking

Date: 11th of December, 2017

Course: Information Security

Professor: Lee Hee Jo

Table of Contents

Project Objective

Preface brief

In our current modern society, huge portion of the population is connected to the Internet. Now, information is being stored on digital devices, and is being transferred from one to another very commonly, especially when communicating. Thus, privacy of information, and its integrity has become an important issue. Therefore, to keep personal information private people started to use programs with security functions.

You can find lots of different messenger applications which provide secure communication with various encryption levels. However, it is very hard to know how secure one's application is. "How can you know which application is better than the other one?", and "How can you know if the application is secure enough?", these kinds of questions arise when trying to pick between numerous applications available in the current market. We were interested in encrypted messenger applications, "How do they work?", "What kind of encryption is being used?", and "How to achieve maximized security?". Thus, we chose to base our team project on encrypted messenger.

The aim of this project was to acquire more knowledge about encryption, and encrypted messengers, while creating a highly secure encrypted messenger which could potentially compete with the current available applications. Therefore, our end-product would be an open source encrypted messenger application that would be available to anyone.

Objective Specification

To fulfil our objective of creating a highly secure encrypted messenger, we have decided to consider the authentication methods. We have learnt in class that, both humans and machines are able to attack any system. Thus, our focus was to implement protocols, such that it can prevent both types of attacks. Through extensive research, we were finally able to select couple of methods that would be implemented in our program to prevent such attacks.

1. Password
2. CAPTCHA
3. Security Question

We believe that using three level authentication you can minimize the vulnerability of being hacked the most to the minimum without applying extraneous methods.

Research

During the project process several authentication processes were found and evaluated, but only couple were chosen to be implement, as others had more potential liabilities or weaknesses or limitations.

Graphical User Authentication

It was brought to our attention that image password can be used as one of the authentication techniques. Through voluminous research and investigation, we have found out that image passwords can be essentially broken down into two categories, recall based technique and recognition based technique. Recognition based graphical user authentication algorithms require the user to click certain image(s) in a particular order to authenticate oneself. On the other hand, recall based technique requires the user to replicate the password that was created before. Each technique has its own merits over the other one, and each technique has several multiple schemes. For example DAS scheme, signature scheme, and pass point scheme are part of recall based technique, while Dhamija and Perrig scheme, passface scheme, and Sobrado and Birget scheme goes under recognition based technique. We thought of implementing a graphical password that is similar to what Windows 10 has, because in our opinion it was relatively easy to implement within our time constraints and it could be used to increase factor of authentication. However, it was later regarded as not worth pursuing due to being prone to *over the shoulder* attacks.

CAPTCHA

Another method of authentication that was considered is CAPTCHA. During this time we were learning about bots, and how they can be used to attack the system. Therefore, we thought that we should also consider such a possibility, and find a way to prevent, or make it harder for machines to gain authorization. Thus, we thought of utilizing CAPTCHA as it is being used world-wide and well known by many users so they will not be flustered when asked to pass the test. Once again, there are currently many CAPTCHA algorithms, such as the ones that use numbers, and those which use images. We decided to implement a CAPTCHA question where several images along with the question in text would be given to a user, and he/she would have to select correct pictures. (**CHECK AND ADD IF WE USED IT OR CHANGED IT DUE TO… OR MENTION LATER IN THE DESIGN PART**)

Security Questions

Third authentication level technique that we thought of was to implement the use of security questions. One of the members suggested this idea, as this person had experience of being hacked, and thought that more personal questions would make it harder for an attacker to bypass. Of course as seen in movies and tv shows, hackers have complete information about their target by founding information online or by other sources. However, we believe if you are careful enough, configure all security measures, and keep a low profile online then it will be relatively hard to acquire the necessary information. Thus, we decided to use this method as our third level of authentication.

SMS Verification & Validation

Through a consultation with a professor, we have decided to look into SMS verification and validation techniques. By further looking into this kind of method of authentication, it was found out that you need to have a server to send sms verification codes to mobile phones from the machine. Since we did not have a server that we could use, we had to abandon the idea of using sms for verification or as a validation technique (SMS being sent to the user notifying about logging in). In addition, this limitation restricted us in the ability of achieving multi-factor authentication, as we would have covered what the user knows and what the user has.

Design & Implementation

Initially, we sought to write the code for the security questions and captcha system individually and then later implement it into the program, however as we became more familiarized with the program and its code, it came to realize that this strategy would be overly tedious and very perplexing. Therefore, we decided to begin within the program’s code itself. Hence, this allowed us to program and implement the code simultaneously.

Security Questions:

We decided to split the creation of the security questions into two parts. The first part covers the registration, encryption, and saving of the security questions and answers. And the second part covers loading the questions on startup and authenticating the user's input.

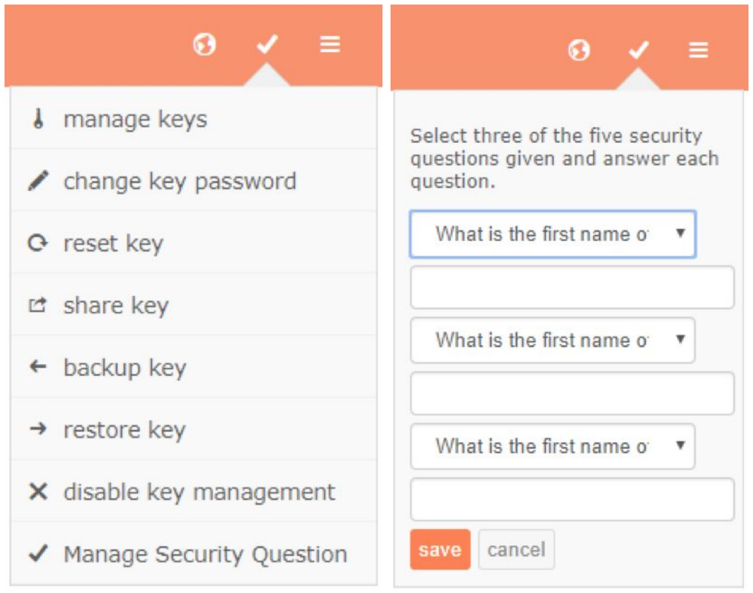
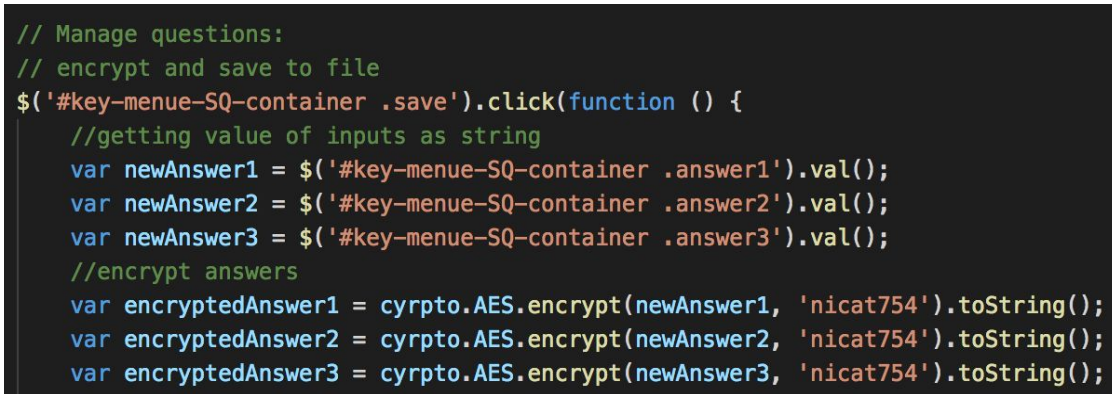


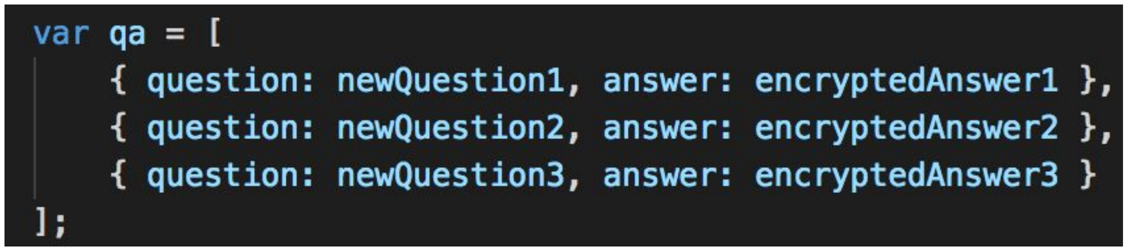
Figure: Manage Security Question tab and Question Select screen

As can be seen in the figure above, we coded a new tab on the menu for registering and saving (subsequently enabling) the security questions. When clicking the tab, the user can select three questions to answer from five pre-selected security questions. An additional feature we added was, we prevented the user from selecting two or more of the same question. This was done to somewhat combat Social Engineering where a user might not want to go through the hassle of selecting and answering three different questions.



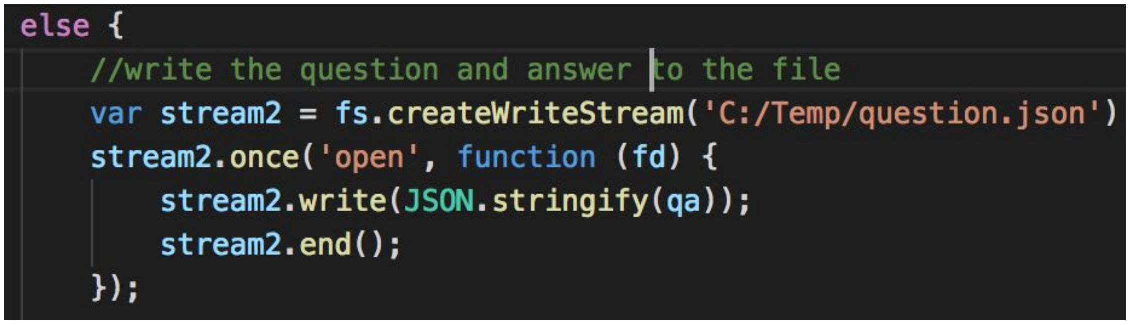
```
// Manage questions:
// encrypt and save to file
$('#key-menue-SQ-container .save').click(function () {
    //getting value of inputs as string
    var newAnswer1 = $('#key-menue-SQ-container .answer1').val();
    var newAnswer2 = $('#key-menue-SQ-container .answer2').val();
    var newAnswer3 = $('#key-menue-SQ-container .answer3').val();
    //encrypt answers
    var encryptedAnswer1 = cyrpto.AES.encrypt(newAnswer1, 'nicat754').toString();
    var encryptedAnswer2 = cyrpto.AES.encrypt(newAnswer2, 'nicat754').toString();
    var encryptedAnswer3 = cyrpto.AES.encrypt(newAnswer3, 'nicat754').toString();
```

Figure: Extracting Answers and Encrypting using AES



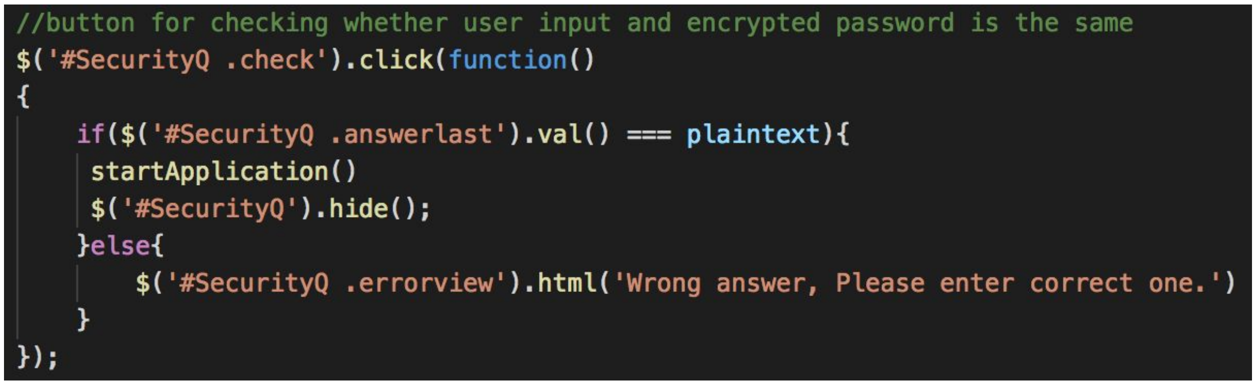
```
var qa = [
    { question: newQuestion1, answer: encryptedAnswer1 },
    { question: newQuestion2, answer: encryptedAnswer2 },
    { question: newQuestion3, answer: encryptedAnswer3 }
];
```

Figure: Creation of Question and Answer Dictionary



```
else {
    //write the question and answer to the file
    var stream2 = fs.createWriteStream('C:/Temp/question.json')
    stream2.once('open', function (fd) {
        stream2.write(JSON.stringify(qa));
        stream2.end();
    });
```

Once the user selects and answers each question, the questions are saved to a file unencrypted (due to the fact that the questions are preselected therefore any attacker can know the security question by simple installing the program themselves, therefore making question encryption futile), whereas the answers are first encrypted using an AES algorithm and then saved to a file. The question and answers are joined together into a dictionary and saved together.



```
//button for checking whether user input and encrypted password is the same
$('#SecurityQ .check').click(function()
{
    if($('#SecurityQ .answerlast').val() === plaintext){
     startApplication()
     $('#SecurityQ').hide();
    }else{
        $('#SecurityQ .errorview').html('Wrong answer, Please enter correct one.')
    }
});
```

Figure: Checking for if user input and encrypted password are equal

The plaintext variable is storing the encrypted answer, when the Question and the answer page shows on screen,the user inputting the the answer,so program grabbing the user input and compare it with the encrypted answer ,if the user input same with the encrypted answer the main interface come out otherwise program gives error that the user input is wrong.

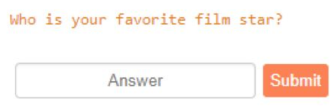
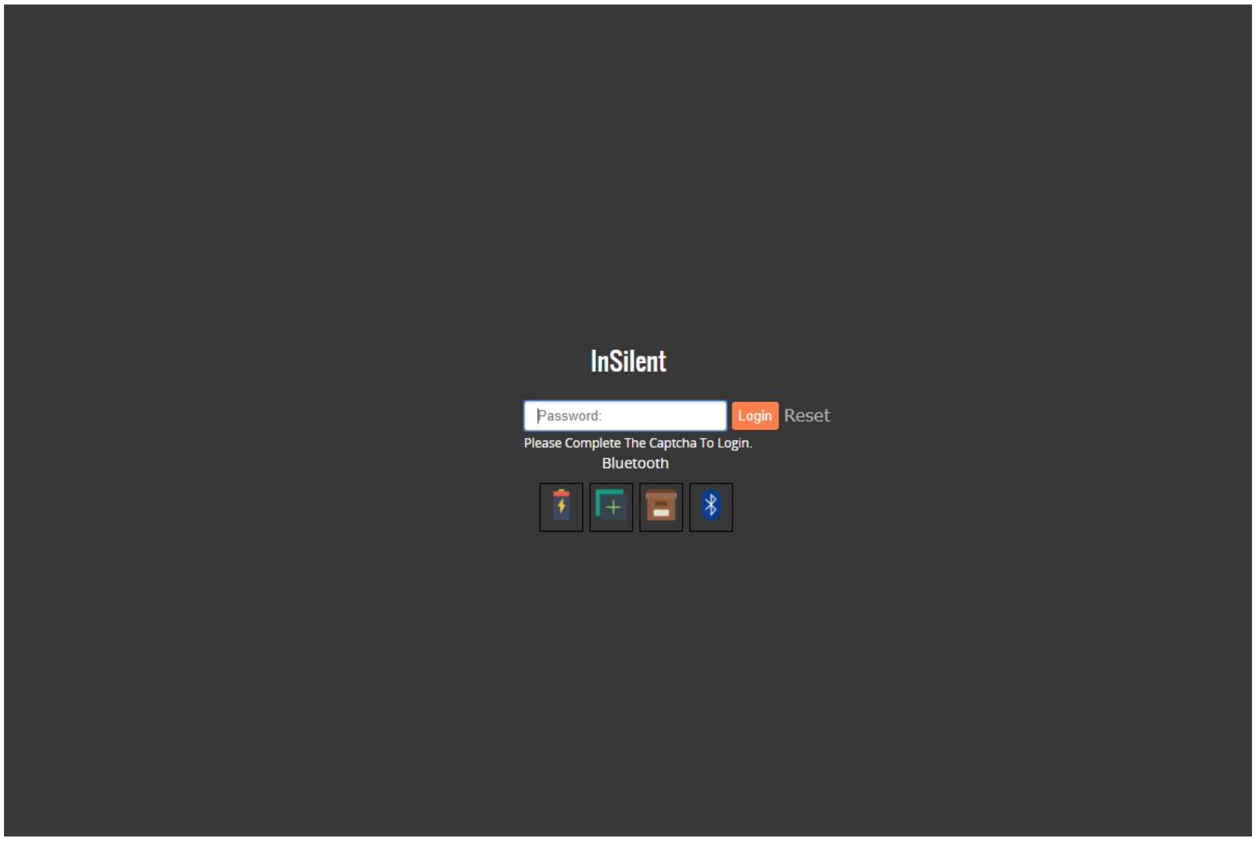


Figure: Interface of Security Question

Finally, after successfully inputting the password and correctly answering the CAPTCHA, the user reaches this page where they must answer the randomly selected security question. If the user does not input the correct answer, they will be prompted to input the correct answer.

Captcha System:

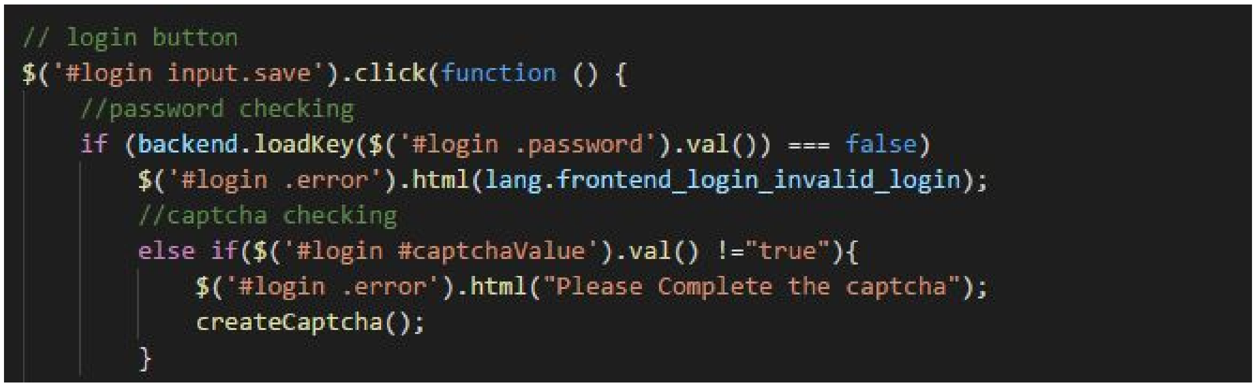
Captcha system is full written in the frontend of the program.It consist of 4 randomly chosen icons.For completing the captcha check,the user should select the correct icon which is the name of the icon is written on the program interface.When the user select the correct icon it has permission to pass next third authentication part.The main purpose of implementing the captcha system is preventing the bots and brute force attacks.



Please Complete The Captcha To Login.

Bluetooth

Figure: General view of the login interface with the image captcha system.



```
// login button
$('#login input.save').click(function () {
    //password checking
    if (backend.loadKey($('#login .password').val()) === false)
        $('#login .error').html(lang.frontend_login_invalid_login);
        //captcha checking
        else if($('#login #captchaValue').val() !="true"){
            $('#login .error').html("Please Complete the captcha");
            createCaptcha();
        }
```

Figure: Password and Captcha Authentication

This the frontend code of the captcha verification system.When the user press the login button , first the program is doing password validation then checking the selection of the user for the captcha image.If the correct image is chosen captchaValue variable changes to true and let the user pass the security check otherwise it shows error and re-new the captcha images with new ones.There are 16 icons for using captcha checking but this number is variable and can be changed by developer.The captcha system create itself when the program start.

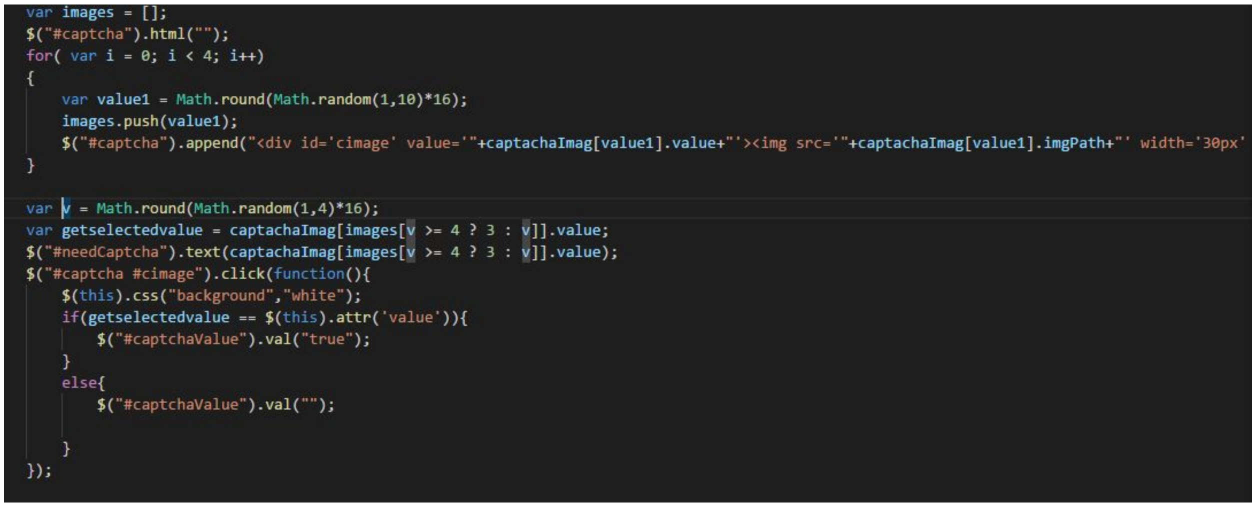
```
$(document).ready(function(){
            createCaptcha();

        })
```

There is a captcha image array which contains the image is names in the app folder.

```
function createCaptcha(){

            var captachaImag = [{

                imgPath:'png/add.png',

                value:'Add Sign'

            },.....
```

First program chose 4 images randomly from `captachaImag` array and pushing them to the new image array which called `var images = [];` and showing the images on the Interface of the programs.

```
for( var i = 0; i < 4; i++)

            {

        var value1 = Math.round(Math.random(1,10)*16);

        images.push(value1);

         $("#captcha").append("<div id='cimage'

value='"+captachaImag[value1].value+"'><img

src='"+captachaImag[value1].imgPath+"' width='30px'

value='"+captachaImag[value1].value+"'></div>");}
```

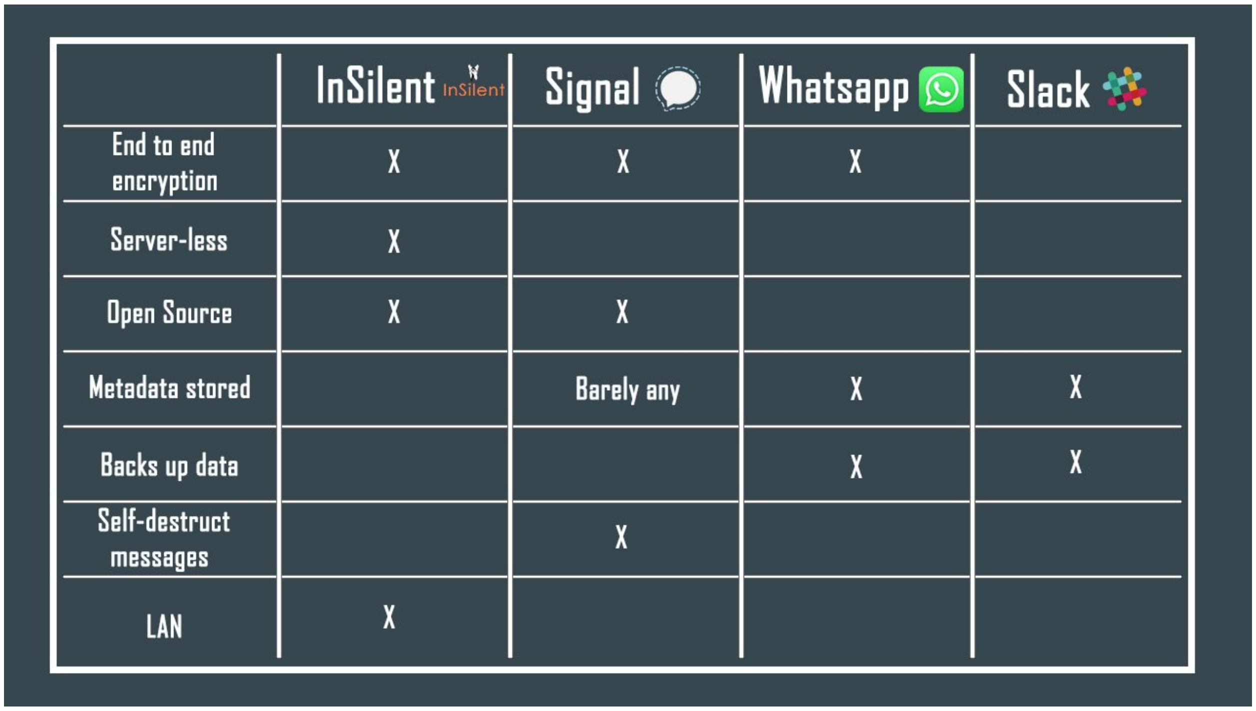


```
var images = [];
$("#captcha").html("");
for( var i = 0; i < 4; i++)
{
    var value1 = Math.round(Math.random(1,10)*16);
    images.push(value1);
    $("#captcha").append("<div id='cimage' value='"+captachaImag[value1].value+"'><img src='"+captachaImag[value1].imgPath+"' width='30px'
}

var v = Math.round(Math.random(1,4)*16);
var getselectedvalue = captachaImag[images[v >= 4 ? 3 : v]].value;
$("#needCaptcha").text(captachaImag[images[v >= 4 ? 3 : v]].value);
$("#captcha #cimage").click(function(){
    $(this).css("background","white");
    if(getselectedvalue == $(this).attr('value')){
        $("#captchaValue").val("true");
    }
    else{
        $("#captchaValue").val("");

    }
});
```

At the last steps function grabbing names of the icons which is given by developer and showing the as the indicator of icon.While the user select the icon, program makes it white for remembering the choice, and user cannot change the choice if selected a icon, login button should be pressed for making new choices.

Analysis & Discussion

Comparison Analysis



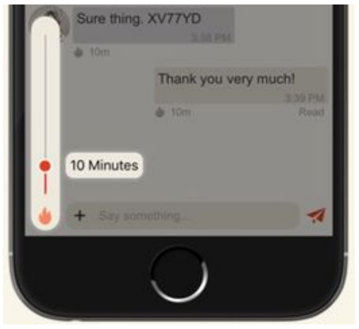
	InSilent	Signal	Whatsapp	Slack
End to end encryption	X	X	X	
Server-less	X			
Open Source	X	X		
Metadata stored		Barely any	X	X
Backs up data			X	X
Self-destruct messages		X		
LAN	X			

We have compared InSilent encrypted messenger with other encrypted messengers that are currently available online for use. While there are numerous programs online, we have selected the three best programs that are currently in the market in our opinion, and they are, Signal, Whatsapp, and Slack. All of these products, have desktop clients and this was one of the reasons we have chose them as well. Signal and WhatsApp are quite similar as the use the same Signal protocol. However, the difference is that WhatsApp is registered as a messaging app and therefore it is strongly concentrated on communication side, whereas Signal focuses more on the security. Slack is another program that is widely used by companies for communication, and thus was chosen to be included in the comparison analysis.

All four programs have some level of security built in, but as it can be seen from the table the level of security differs from one program to another. All three programs, InSilent, Signal, and WhatsApp, have end to end encryption, while Slack utilises encryption protocols as well it is not end to end encrypted. InSilent application is the only one that uses LAN connection to communicate between users, which can be a potential downfall of this program as you can only communicate with people who are on the same LAN network. However, it could also be seen as a potential advantage, because you no longer have to be connected to Internet and thus reduce the threat of being attacked from the outside. This also allows InSilent program to be "server-less", even though a file with user credentials is saved on a shared network which acts as a server. Both Signal and InSilent applications, are open source which means that they are available for use and scrutiny by other professionals. One of the unique features of InSilent is that it does not store any metadata, unlike most of the messaging applications available in the current market. WhatsApp and Slack have a capability of backing up data, which can be very useful for the users who use the program. However, we believe that it opens another port from where a potential intruder can intercept data if not backed up correctly and safely. One feature that Signal has over other applications used in the comparison is the ability of self-destructing messages. Overall, it can be seen that all four programs have their benefits and drawbacks, and that no program can be considered to be the best one.

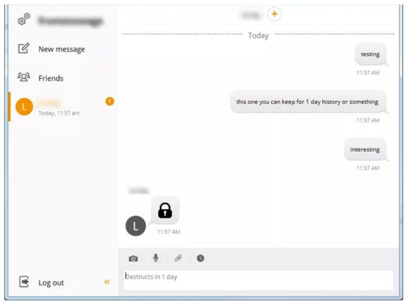
<u>Unique Features of other Applications</u>

Silent


Sure thing. XV77YD
3:38 PM
10m
Thank you very much!
3:39 PM
10m Read

Silent is a mobile application, that is used by users to send messages to each other. Just like any other messenger Silent has the capability of sending pdf files, photos, audio, and video. One unique feature of Silent is that it allows the user to put a timer on the message he/she wants to send before it gets deleted.

Wickr


Today
testing
11:57 AM
this one you can keep for 1 day history or something
11:57 AM
interesting
11:57 AM
11:57 AM
Destructs in 1 day

Wickr is another encrypted messenger that is used by teams and enterprises for secure communication. It also has the capability of not only sending text messages, but also files, such as video, audio, and photos. A unique feature of this application is the ability of making messages viewable for one day after which you no longer able to see the message, and see a lock instead.

Future Potential Improvements

Current InSilent application obviously has certain advantages over programs which were used in the comparison analysis, but it also lacks some certain functions as well. Thus, we believe that further improvement of the program is needed. One possible improvement could be the inclusion of self-destruct message capability, like Signal or Silent has, or the implementation of timer before you are no longer able to view the message, like Wickr. Another possible improvement of the software, that we thought of while creating multiple level authentication was SMS verification/validation technique. This could definitely increase the level of security of the application, especially if validation technique is being applied. This will create another factor of authentication, what the user has.

Concluding Remarks

(nadya, prince, sattam)

The purpose of our project was to create a secure encrypted messenger application, aiming to improve and increase the security of the application. During the process, we tried our best to consider the Eight Principles of Trusted System Design, and implemented some of the principles on the InSilent application we designed:

1. **Economy of mechanism: Two simple steps that covers both botnet and human vulnerabilities**
   - The first principle says to keep the mechanism as simple as possible. After considering for a way that would be simple, yet at the same time effectively deals with bot and human vulnerabilities, we decided to add two more authentication steps after normal password authentication. CAPTCHA, which is the second authentication .step after password, is implemented to deal with botnets, while the security question is implemented to prevent access from unauthorized users (human).
2. **Separation of privilege: Three conditions to grant privilege**
   - The three step authentication is an implementation of this principle, which suggest for multiple conditions to grant privilege. For InSilent, the user must go through all three (password, CAPTCHA, security question) authentication steps to grant privilege. One single authentication failure would immediately deny access.
3. **Least common mechanism: LAN based**
   - This principle suggests that mechanisms should not be shared. With our application being LAN based, there is no shared network for the application meaning there are no links

between chatrooms in different places. The LAN connection also prevents events such as security breaches to spread other than the location being affected.

After a 9 week progress, the end-product is the encrypted messenger InSilent, secured with a three step authentication (password, CAPTCHA, security question):

- https://github.com/Glockx/InSilent

Team Members :

Gidiglo Prince Delator 2014210115 (Group leader)

Aldhargham Abdulmajeed 2015410134

Alkhalaf Musab Saleh S 2016320262

Almuzaini Sattam Esam A 2016320245

Anas Alawa 2017952771

Andrei Son 2014210120

Nijat Muzaffarli 2016320242

Scholasticha Nadya Caesar 2014320054