# Background

The Checkmate application deals with sensitive law enforcement information. It is a primary requirement for this application to secure the information contained inside the app. This will require encryption to ensure data is not available to anyone that may obtain either the device or intercept data transmitted from the app (via email or other electronic transfer).

This project will be comprised of a few different components:

1. Apply encryption to secure data stored within the application.
2. Define the requirements for transferring the data from the app to a law enforcement secured server
3. Client application to access the information.

## Repository

https://cisbjmaclean@bitbucket.org/cisbjmaclean/yesis\_nserc\_checkmate.git

## 

## 

## Architecture

The suggested approach for Checkmate is to allow two options for the user.

#### Encrypt Phone

The easiest manner to allow encryption on the device is to be encrypted. Any data on the device would be encrypted until the device is unlocked by the user with the passcode. The files could be uploaded when the phone is unlocked and files would be available. Research done on Android. Google introduced security in 2011 to allow phones to be encrypted. This will have an impact on speed but provides security of data on phones. This would be valuable for other data than checkmate for phones being used for workplace sensitive data (emails, docs etc).

#### Encrypt/Decrypt Files

As a second option, the files that are used by the application can use the passcode of the user to encrypt the data. All files that are stored in the app would be encrypted as they are created. These would then have to be decrypted when accessed in the app. This would require some overhead but during testing files were encrypted/decrypted with minimal delay.

When the user chooses to upload an checklist, the encrypted files are all transferred to the server. The server can implement a similar process. A service can be provided using USB tethering to connect the device to the computer. This service can be made available as a wireless network service in the future. The files are secured during transfer and decrypted using the same passcode they were used to encrypt.

The server application can save the files and make them available for viewing.

Notes:

* Using the default mode for the Crypto JS
* Testing was done using Android

## 

## 

## 

## Encrypting data

|  |
| --- |
| The information can be secured on the device using device encryption but the application should also have the ability to encrypt the data stored on the device. The data will be stored using an open standard format (json) for the data and binary for the audio and image files. The mechanism to provide this encryption is part of the project. |

* A symmetrical encryption approach was followed using a Javascript implmentation of AES encryption. Testing was completed to read a file, encrypt, transfer, and write files. The speed was good.
* Application is a javascript based application so used an AES js library. This should be transferrable between platforms.
* There will be an issue to be resolved for all of the images. If they are stored encrypted then when you access the images in the app you will have to decrypt them to make them viewable in the app.
* The file reading/writing is where there was a lot of challenges. The Cordova FileReader/Writer works with different encoding. The encoding can be base64 / utf8 so have to test to ensure this encoding works correctly. Have tested Android and am able to transfer successfully.

### Testing

|  |
| --- |
| 1.5MB mp4 - 4 seconds to encrypt  4MB mp4 - 9 seconds to encrypt  8.5MB mp4 - 22 seconds to encrypt |

10MB+ caused crash initially and further analysis required.

### Outstanding Issues

* Testing for the encryption of large files
* test Checkmate with an encrypted phone
* Add a progress

## 

## 

## 

## Data transfer

|  |
| --- |
| There are several methods available to get the data from the device to the law agency secured servers. This could be USB connection, email, secure ftp, or over a wireless network. Recommendation and demonstration of transfer is part of the project. |

The process followed was to access a network service provided by a Node.js web service. This was done using usb Tethering. Once the device is connected via usb the tethering would have to be turned on. From there the user could access their checklist and choose upload. The files could be triggered to be sent to the service running behind the organization firewall available on the computer.

### Outstanding Issues:

* Check for connectivity <http://stackoverflow.com/questions/9887621/accessing-localhost-of-pc-from-usb-connected-android-mobile-device>
  + Implement an error if not able to connect or other error sent from client desktop service.
* Can we access an ip address accessible only on the device or hit an ip address that is accessible on the pc from the device. Ie. Can we have a server running on the network that the computer is attached to and hit that service from the device.
* <https://brendonparker.wordpress.com/2015/06/22/reverse-tethering-solution/> It may be possible to open a socket over usb to connect the device to the pc.
* Pass some metadata to the service as well which could be used to save the file.

## 

## 

## 

## Client Application

|  |
| --- |
| When the data is transferred from the devices the records must be organized for storage and retrieval. A client application will be designed and developed to decrypt, read, and save. This application will provide a basic mechanism to view the incident information transferred from the Checkmate mobile application. |

A Node.js web application was created which will provide the app with a service to receive the checklist files. As the checklist is received it can be decrypted and saved to the server.

### Outstanding Issues

* Pass metadata (json) to the service to be used when saving the file to the computer. Currently am passing the name of the checklist which is used to create filenames and folder names.

Demonstration

Tethering device to pc

* This has been tested with Android
* Testing with ios required.
* This differs by device
* The transfer is setup to go to a networked serice. This could be a service which is available on the wifi network or internet. The app wouldn’t care as long as it can see the service.
* Would have to have a setting to send the data to the correct ip address.

Creating an image in the app with tethering turned on.

* Image is stored in the app data. This is stored as an image. It could be stored encrypted. When loading the checklist, all of the images could be decrypted as a specific checklist is accessed (ie. Initial Complaint). The images would have to be decrypted to be viewed in the app. Issue would be that the decrypted version would have to be deleted before leaving the app.
* Note how the image is encrypted and sent to the service for decryption/storage on pc.

Upload checklist

* Have added an upload button. This is currently uploading the pdf of the checklist only. The images are encrypted and sent individually for testing when the images are taken.
* Checklist name is used to create the folder.
* In the future suggest that all components of the checklist would be sent over one by one. This would replace the zip file creation.

## Notes

* In the future the research activities could be a spring/summer activity.
* Was a great learning experience (Node.js, Angular.js, Cordova, Encryption technology, File storage)
* Efficiency of working with an experienced resource (related to scheduling during low class time).

## 

## Resources:

<https://www.cs.utexas.edu/users/byoung/cs361/lecture44.pdf>

<https://www.ciphercloud.com/blog/cloud-information-protection-symmetric-vs-asymmetric-encryption/>

<http://stackoverflow.com/questions/16600509/aes-encrypt-in-cryptojs-and-decrypt-in-coldfusion>

<https://dzone.com/articles/interoperable-aes256-encryption-between-cryptojs-p>

<http://cryptojs.altervista.org/secretkey/aes_cryptojs-v3.html>

Cordova website in detail