Mostafa Elkhouly

34-0871
T14

Secure WhatsApp

Summary

A similar whatsApp application supposed to be known as SayHi Chat and this application serves like a whatsApp authenticates the user and protect him/her by encrypting messages,attachments,shared locations.This chat application is different than any other application as if any attacker accessed the servers the backend where the messages are stored cannot retrieve or understand anything from it.As this message is encrypted in a special fashion and decrypted only to the authorised recipient.

Motivation

1. Exploring the technology ,techniques and features behind using using **Back4App Server for creating backend server for our chatting application**
2. Exploring the benefit of using the Virgil Security for encrypting the whole **application linking it with the Back4App through connecting it through Cloud**.

Implementation of Features

1. Authentication
2. Encryption
3. Access Control

4. Tokens

All these features have been demonstrated through BackApp Server and Virgil Security Api.Basically authentication is 90% build upon the implementation of the Back4App libraries,frameworks and dependencies used in the project.The remaining 10% was updated through virgil security to enable the secure authentication and so no one else see the authentication user session chat messages.Along with the Encryption basically, encryption techniques built almost completely through virgil security giving tokens,establishing the keys and encryption hashing algorithms being used here.Access control implemented in a fashion such that read,write and controlled by the server only.Tokens are created by the application in virgil security and this token is passed through the cloud server linking the virgil security to the Back4App .

Cryptographic Algorithms and Protocols:

1. During sign-up: we'll generate the individual private & public keys for new users (remember: the recipient's public key encrypts messages and the matching recipient's private key decrypts them).
2. Before sending messages, you'll encrypt chat messages with the recipient's public key.
3. After receiving messages, you'll decrypt chat messages with the recipient's private key.

We'll publish the users' public keys to Virgil's Cards Service so that chat users are able to look up each other and able to encrypt messages for each other. The private keys will stay on the user devices.

After setting up the App with the Credentials from your new Back4App App's Dashboard, enable Live Query to get live updates for messages and chat threads.Then encryption process will begin.

COLLECT ACCOUNT INFORMATION:

The first thing you need to do is grab all the necessary information from your Virgil account. To set up your Client and Server Sides, you need the following values from your account:

Account values	Description
ACCESS_TOKEN	Used to authenticate your users on Virgil Services.
APP_KEY	Private Key that is generated during an Application registration on your dashboard.

APP_KEY_PASSWORD	A password that established for a Private Key of your Application.
APP_ID	Your application identifier.

SET UP YOUR SERVER SIDE:

Your server should be able to authorize your users, store your Application's Virgil Key and use the Virgil SDK for cryptographic operations or for requests to Virgil Services. You can configure your server using the next steps:

Your server should be able to authorize your users, store your Application's Virgil Key and use the Virgil SDK for cryptographic operations or for requests to Virgil Services. You can configure your server using the next steps:

Install SDK & Setup Virgil Crypto

Set up authentication on a server side:

You need to set up server-side SDK to sign and approve user's Card.

SET UP YOUR CLIENT SIDE:

Set up the client side to provide your users with an access token after their registration at your Application Server to authenticate them for further operations and transmit their Virgil Cards to the server. Configure the client side using the next steps:

Install SDK & Setup Virgil Crypto

Set up authentication on a client side

In order to make call to Virgil Services, for example, to publish user's Card on Virgil Cards Service you need to have a Access Token.

With the Access Token we can initialize the Virgil SDK on the client-side to start doing fun stuff like sending and receiving messages.

To initialize the Virgil SDK on a client-side you need to use the following code:

```
VirgilApi virgil = new VirgilApiImpl("[YOUR_ACCESS_TOKEN_HERE]");
```

REGISTER USERS:

Now you need to register users who will participate in encrypted communications.

In order to sign and encrypt a message, each user must have his or her own tools, which allow him or her to perform cryptographic operations, and these tools must contain the necessary information to identify users. In Virgil Security, these tools are the Virgil Key and the Virgil Card.

When you have set up the Virgil SDK on the server & client sides, you can finally create Virgil Cards for the users and transmit the Cards to your Server for further publication on Virgil Services.

Generate Keys and Create Virgil Card

Use the Virgil SDK on the client side to generate a new Key Pair, and then create a user's Virgil Card using the recently generated Virgil Key. All keys are generated and stored on the client side.

In order for the Sender to be able to send a message, we also need a Virgil Card associated with the Recipient. Thus, create one more user Virgil Card to be used..

Transmit Cards to Your Server:

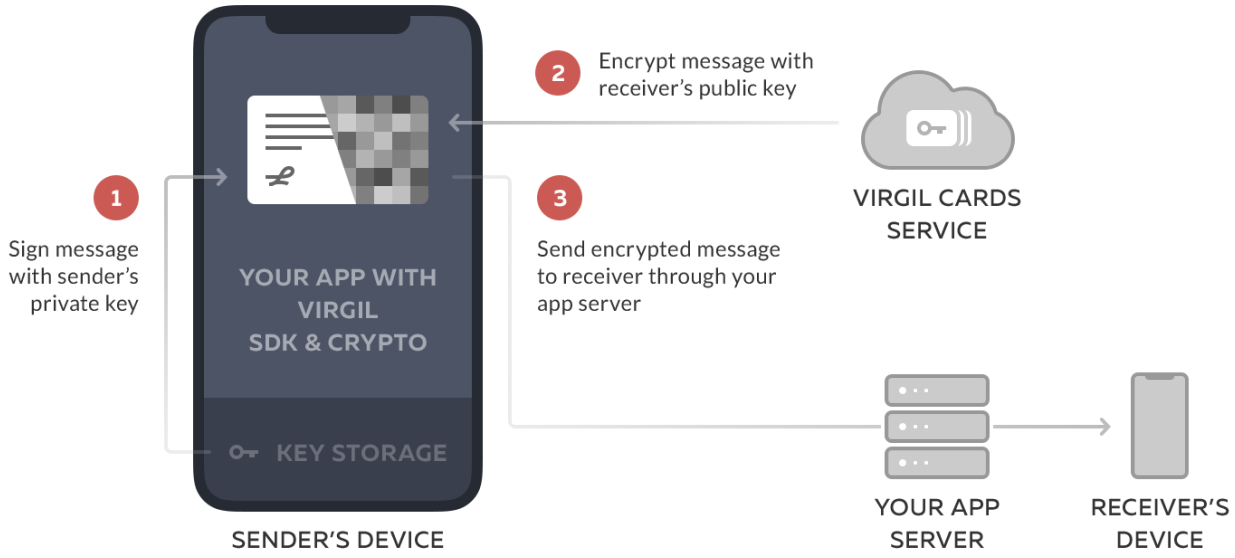
In order to add the signature of your app server to a user's Card you need to transmit an existing user's Card to your server. You can use any suitable way to transmit the Card.

Sign a transmitted user's Card with App Key and publish the Card on Virgil Cards Service:

With the user's Private Key and Cards in place, you will be ready to sign and encrypt a message for encrypted communication. Also, once the Recipient receives the signed and encrypted message, he or she can decrypt the message and verify the signature.

SIGN & ENCRYPT A MESSAGE

As previously stated, we encrypt data for secure communication, but a recipient also must be sure that no third party modified any of the message's content and that they can trust a sender, which is why we provide Data Integrity by adding a Digital Signature. Therefore we must sign the data first and then encrypt it.



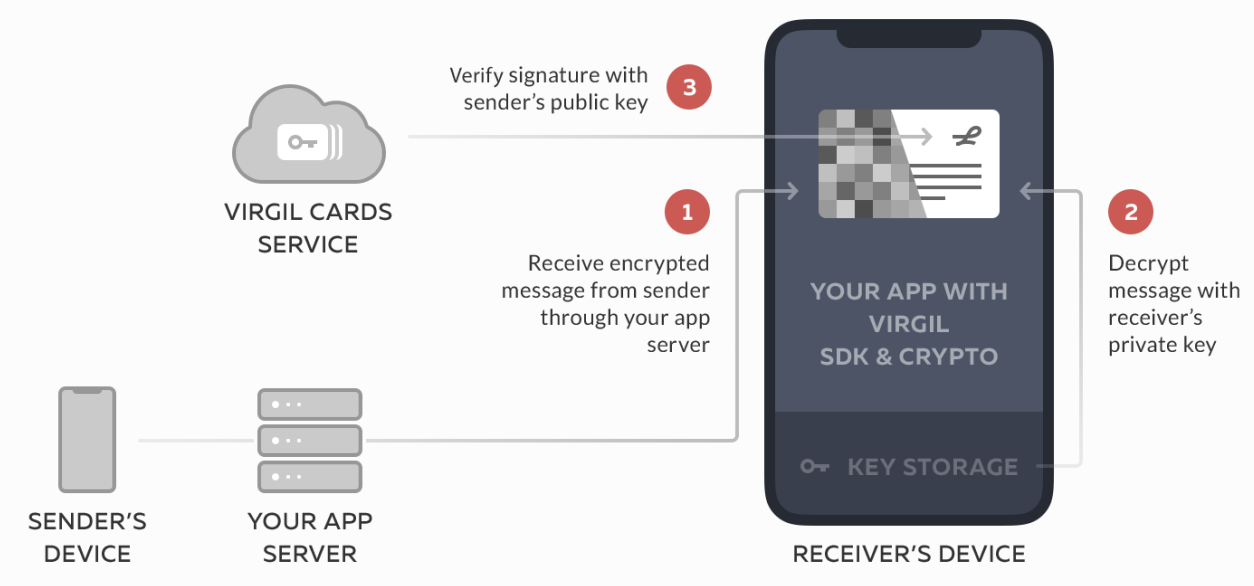
In order to first sign and then encrypt messages, the Sender must load his or her own recently generated Private Key and search for the receiver's Virgil Cards on Virgil Services, where all Virgil Cards are saved.

With the signature in place, the Sender is now ready to transmit the signed and encrypted message to the Receiver.

Use your application to transmit a message.

DECRYPT A MESSAGE & VERIFY ITS SIGNATURE

Once the Recipient receives the signed and encrypted message, he or she can decrypt and validate the message. Verifying the signature against the Sender's Virgil Card proves that the message has not been tampered with.



In order to decrypt the encrypted message and then verify the signature, we need to load a private receiver's Key and search for the sender's Virgil Card at Virgil Services.

My approach for encrypting messages using E2Ev4 have protect the user data or documents with End-to-End Encryption and cryptographic verification.

Security methods, Protocols and Algorithms:

- ➔ IPsec - Internet Protocol Security:
- ➔ IKE - Internet Key Exchange:
- ➔ SSH - Secure Shell
- ➔ SSL - Secure Socket Layer
- ➔ HTTPS - Secure Embedded Web Server:

Integrating an Embedded Web Server in a dedicated device presents special requirements on the server in terms of memory consumption, performance, security and functional requirements. The Secure Embedded Web Server is a versatile, configurable, high performance HTTP server that has low ROM and RAM footprint. It is specifically designed for operating in an embedded environment.

Since the HTTP protocol does not contain any security features, the HTTPS protocol was invented. It introduces Secure Socket Layer (SSL) functionality in the communication between the Web Server and the browser. This eliminates the risk of most security breaches, and has now become the de facto standard for secure web communication.

The Secure Embedded Web Server has built-in support for SSL which is configurable, and can be removed to get minimum footprint. Secure Embedded Web Server features

Secure Web Server features:

- ➔ Supports HTTP/1.0 (RFC1945) and HTTP/1.1 (RFC2616)
- ➔ Supports HTTP methods GET, HEAD and POST
- ➔ Supports incoming entities
- ➔ Implements persistent connections (HTTP/1.1)
- ➔ Supports pipelined requests (HTTP/1.1)
- ➔ Supports chunked mode transfer encoding
- ➔ Supports SSL v2, SSL v3 and TLS 1.0 (RFC2246)*
- ➔ Sends target system files upon client requests
- ➔ Supports precompiled HTML files (HTML compiler included)
- ➔ Supports custom function hooks in a CGI-like fashion
- ➔ Implements public API for sending HTML responses

My approach using HTTPS - Secure Embedded Web Server .

End-to-end Encryption for Twilio IP Messaging with Virgil Security

Twilio is partnering with Virgil Security to make building end-to-end encrypted applications a breeze with Twilio IP Messaging and Virgil Crypto SDK and Key Management. Now every developer can build security into their Twilio apps without having to become a crypto expert themselves.

The Virgil Crypto SDK is used to encrypt the content of messages so that only the message originators and recipients have the ability to decrypt and read it. Virgil Key Management provides the message recipient’s public key. The SDK uses that public key

to encrypt the message on the sender's device before the message is sent. The result is that the content transmitted or stored in the cloud is undecipherable gibberish. The recipient uses their private key to decrypt the message on their device. This ensures that no one else, including Twilio, Virgil Security, telecom providers, Internet providers, or eavesdroppers can decrypt the message.

By default, Twilio IP Messaging is already encrypted in transit with HTTPS. End-to-end encryption is an additional layer of security designed to address the needs of companies that operate under strict regulatory environments. At the same time, encrypting end-to-end may remove the ability to access advanced functionality such as searching chat history.

Attack Scenarios:

Attack scenarios that you thought about (and researched) and how your system is secure against them. Examples and screenshots would be much appreciated.

Scenario 1:

If an attacker knew details about any authenticated username on the server this attacker will not be able to log in successfully.

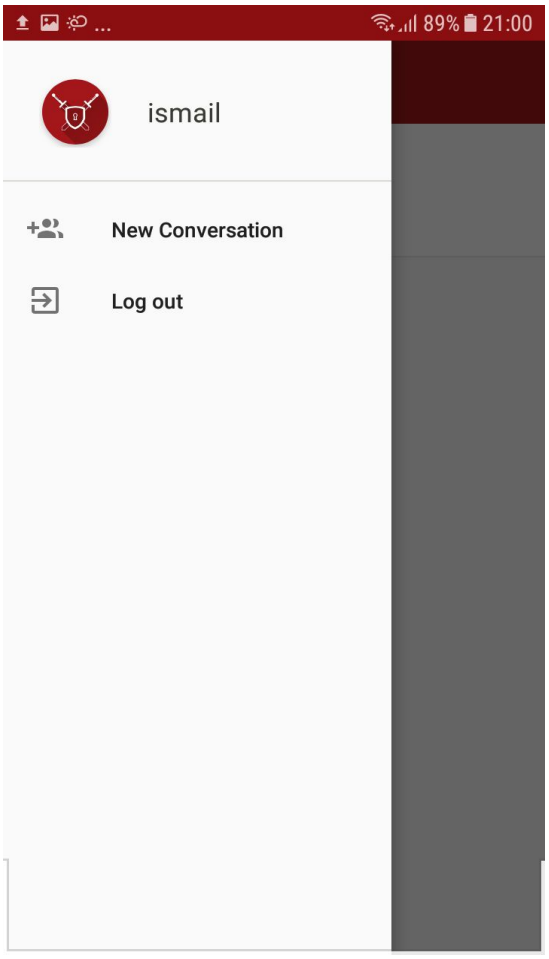


Figure (1)

In figure 1 this is the authenticated real user on his/her device logged in.

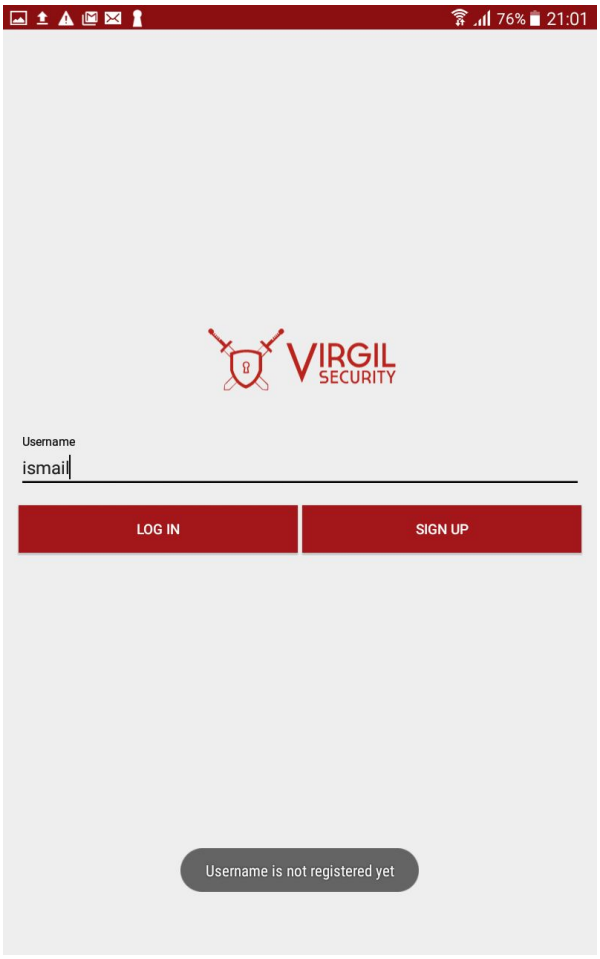
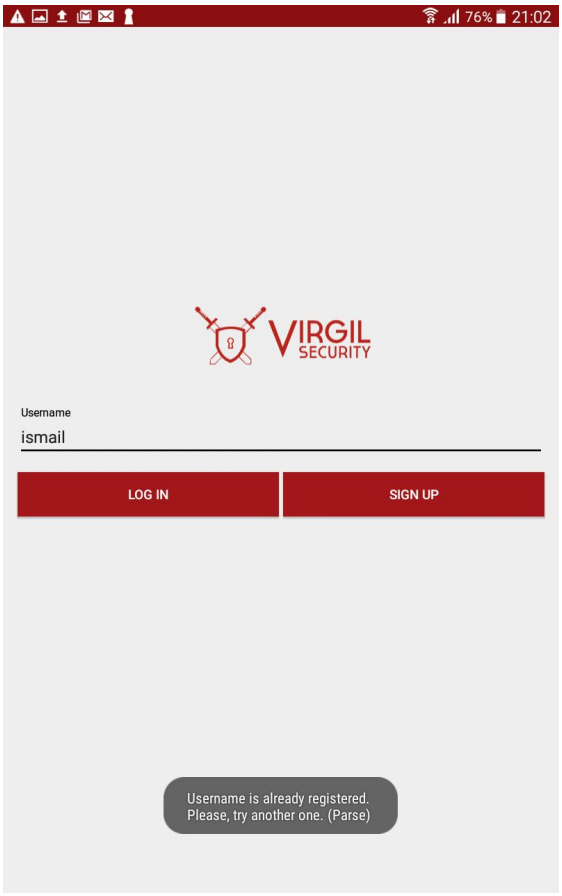


Figure (2)

In figure 2 here is the attacker from another mobile device , the attacker tried to hack to know username , this step of hacking succeed for the attacker but when the attacker tries to log in with the correct stolen username he/she won't be able to log in successfully as this username is unique for only one device so if the device is configured

successfully with that unique username and stored on the server no one else will have the right to log in with this username except the first device logged in successfully.Here there will be a message displayed as a toast which indicated that no username is registered yet this means that this username which is said to be stolen by the attacker will not get verified on the attacker mobile device.

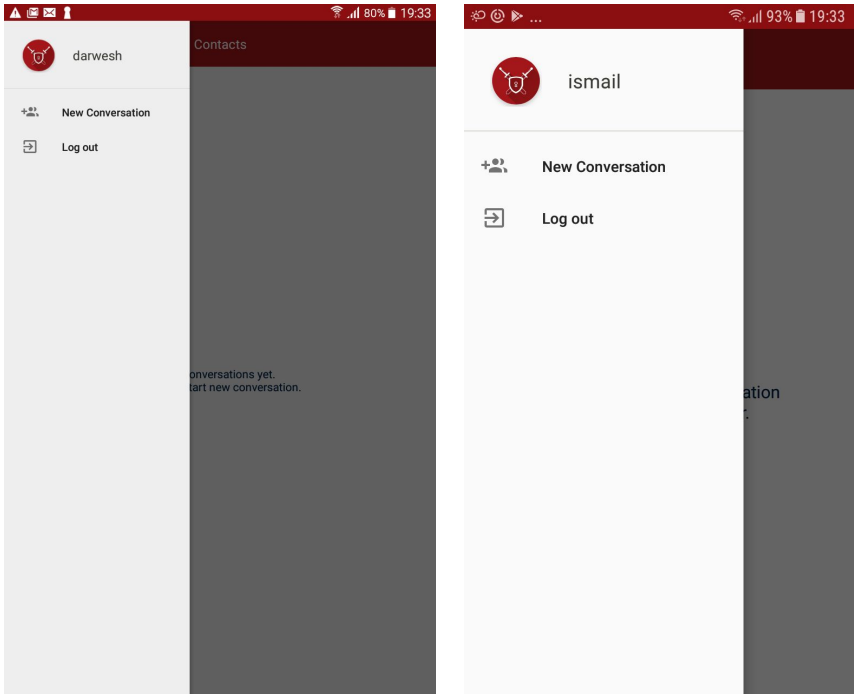


Figure(3)

When the attacker tries to sign up with that username , he or she will not be allowed to and there will be a toast message displayed indicating that the username is already registered.

So in scenario 1 the system is secure enough to prevent the attacker from logging or signing up with the stolen username.

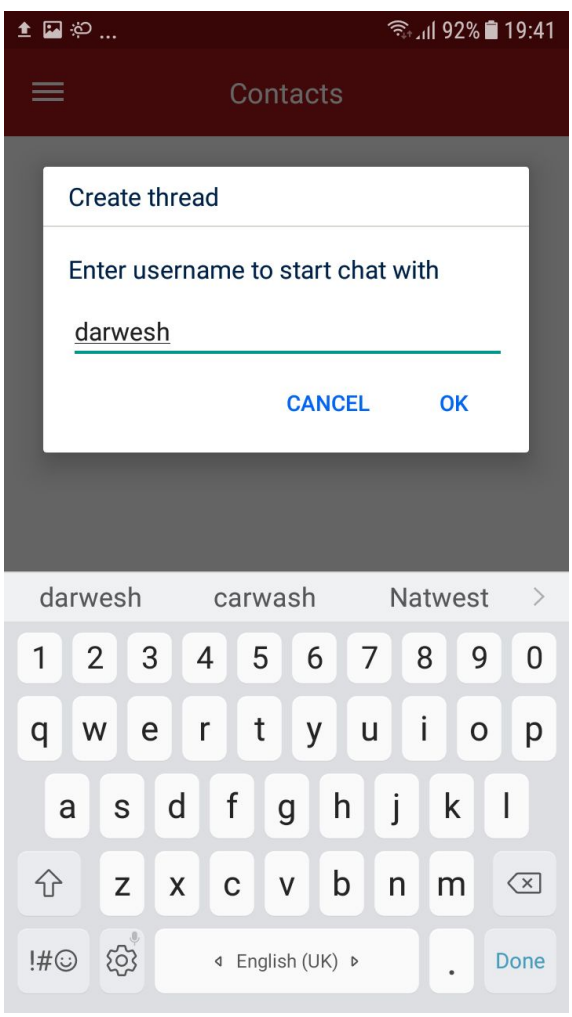
Scenario 2:

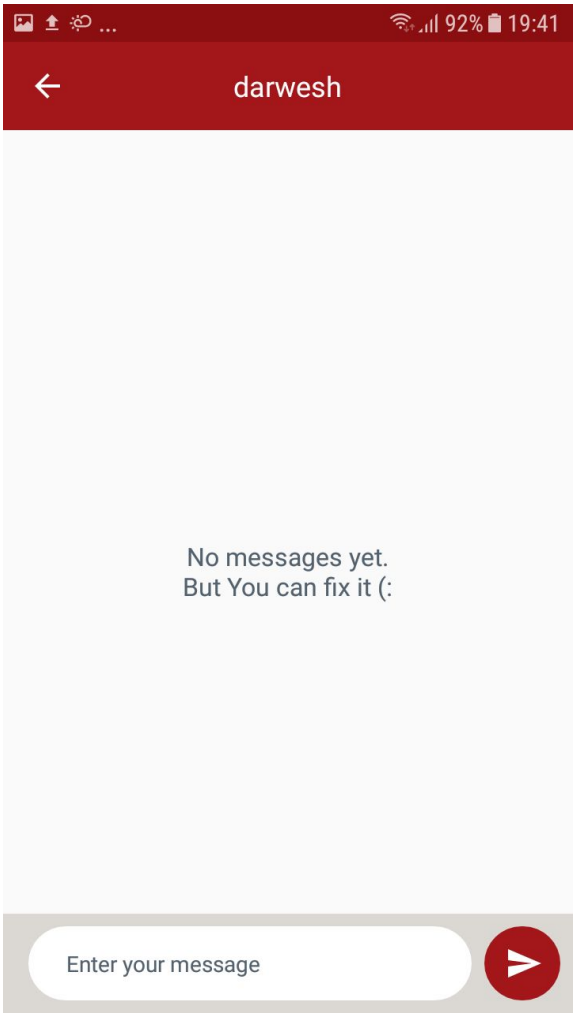


Figure(4)

Here in Figure 4 there are two users trying to communicate,chat with each other.

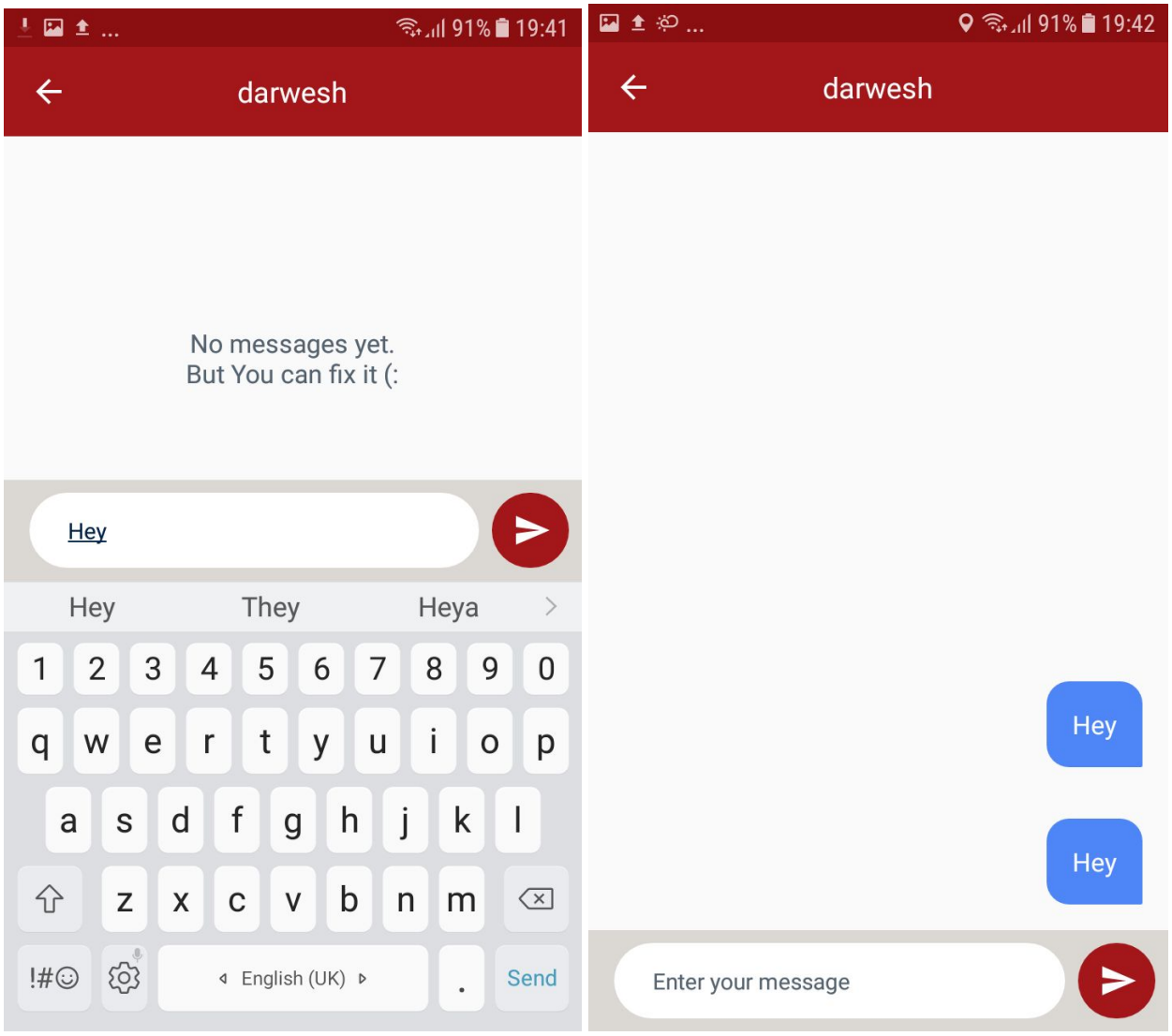
These are the authenticated users.





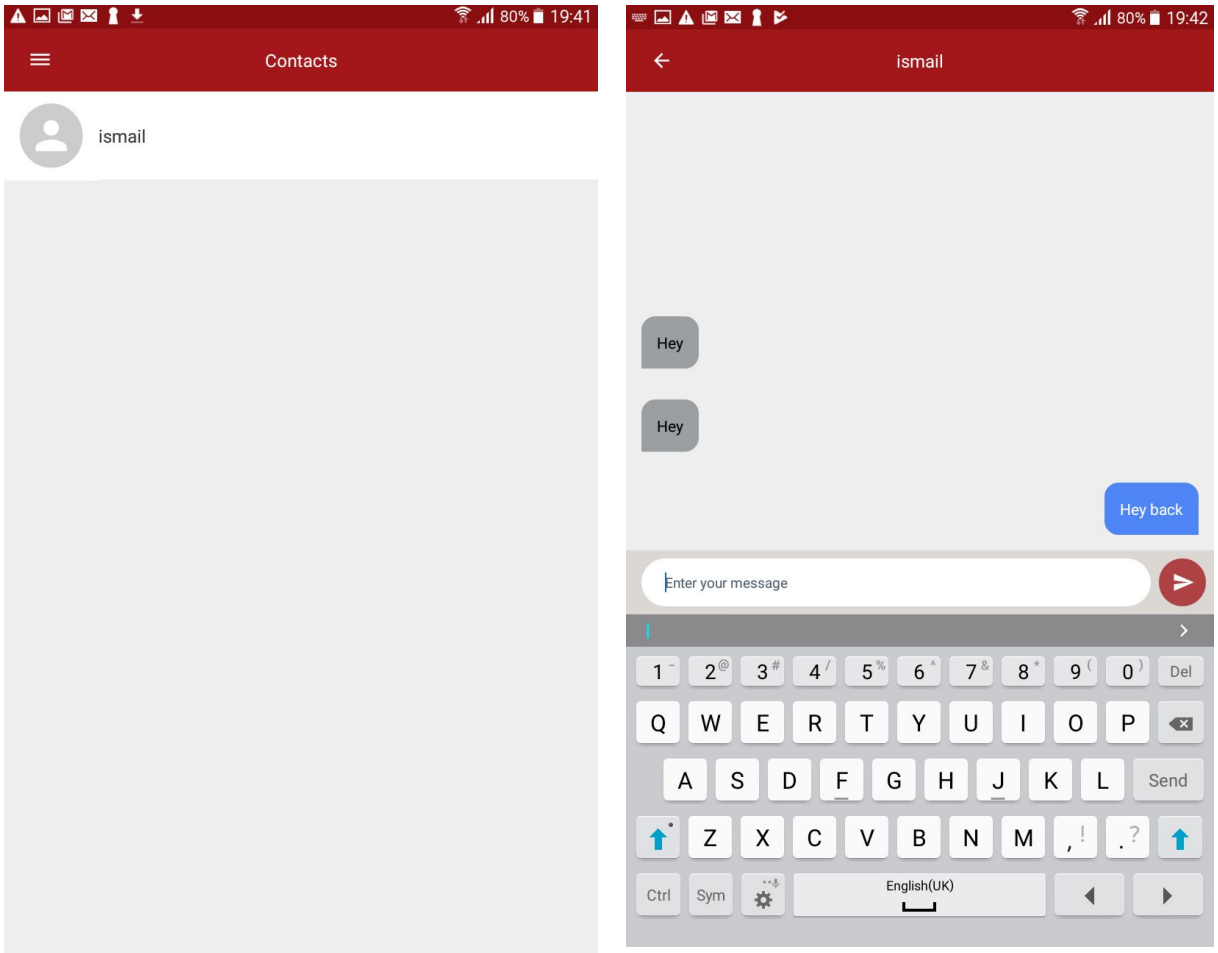
Figure(5)

Here is the process of starting a chat between two users.
Ismail wants to start a chat with darwish.



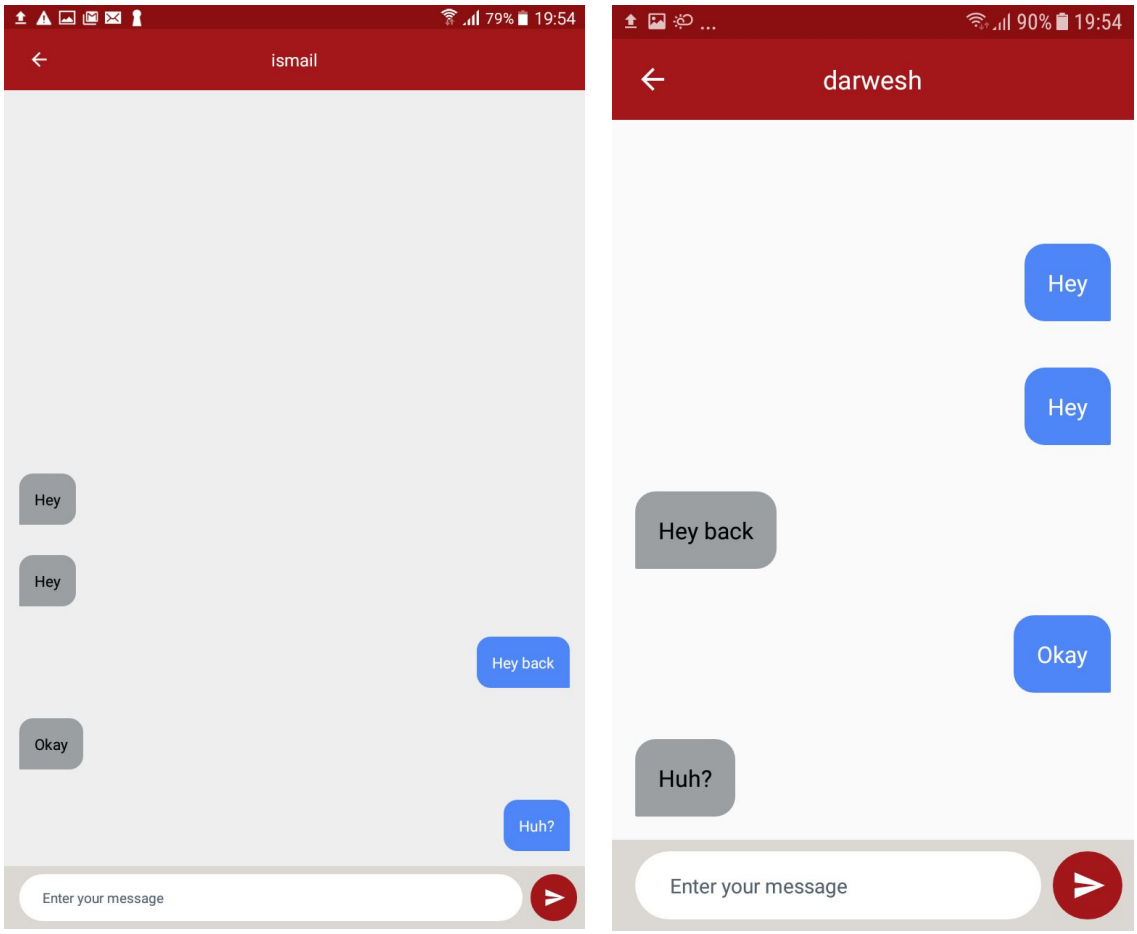
Figure(6)

Here Ismail sends a message of “Hey” to Darwesh.



Figure(7)

Here when darwesh open his account ismail will be added automatically to his contact list.Here darwish will retrive the message of ismail and he even replied by a message of hey back



Figure(8)

A Total of 5 message between those 2 users.

objectId String	emailVerified Boole…	ACL ACL	updatedAt Date	authData Object	username String	createdAt Date	virgilCardId String
BRdZ9u9VlQ	(undefined)	Public Read, BRdZ9…	3 May 2019 at 17:33:02 UTC	(undefined)	darwesh	3 May 2019 at 17:33:02 UTC	e584065befc57d49a1606456f3a
UxLn8bOHcj	(undefined)	Public Read, UxLn8…	3 May 2019 at 17:32:57 UTC	(undefined)	ismail	3 May 2019 at 17:32:57 UTC	16148a5e074b1f5b7a373f0d317

User Class

objectId String	updatedAt Date	createdAt Date	ACL ACL	createdAtDate String	senderUsername Str…	recipientUsername …	senderId String	recipientId String
APjiBh3aJQ	3 May 2019 at 17:4…	3 May 2019 at 17:4…	UxLn8bOHcj, BRdZ9u…	Fri May 03 19:41:0…	ismail	darwesh	UxLn8bOHcj	BRdZ9u9VlQ
EEBBxyYP9j	2 May 2019 at 20:3…	2 May 2019 at 20:3…	9Z0pEuyUxP, BgMnuZ…	Thu May 02 22:38:4…	ramil122	mostafa.mohd96@hot…	9Z0pEuyUxP	BgMnuZwUWj

ChatThread Class

objectId String	updatedAt Date	createdAt Date	ACL ACL	times…	body String
PYD6cOBbKJ	3 May 2019 at 17:51:50 UTC	3 May 2019 at 17:51:50 UTC	UxLn8bOHcj, BRdZ9u9VlQ	Fri Ma…	MIICewIBADCCAnQGCSqGSIb3DQEHA6CCAmUwggJhAgECMYICMjCCA
gjFy7Igr0M	3 May 2019 at 17:42:40 UTC	3 May 2019 at 17:42:40 UTC	UxLn8bOHcj, BRdZ9u9VlQ	Fri Ma…	MIICewIBADCCAnQGCSqGSIb3DQEHA6CCAmUwggJhAgECMYICMjCCA

Message Class

objectId String	updatedAt Date	createdAt Date	ACL ACL	times…	body String
PYD6cOBbKJ	3 May 2019 at 17:51:50 UTC	3 May 2019 at 17:51:50 UTC	UxLn8bOHcj, BRdZ9u9VlQ	Fri Ma…	MIICewIBADCCAnQGCSqGSIb3DQEHA6CCAmUwggJhAg
gjFy7Igr0M	3 May 2019 at 17:42:40 UTC	3 May 2019 at 17:42:40 UTC	UxLn8bOHcj, BRdZ9u9VlQ	Fri Ma…	MIICewIBADCCAnQGCSqGSIb3DQEHA6CCAmUwggJhAg
HDzbKVg5IH	3 May 2019 at 17:41:57 UTC	3 May 2019 at 17:41:57 UTC	UxLn8bOHcj, BRdZ9u9VlQ	Fri Ma…	MIICewIBADCCAnQGCSqGSIb3DQEHA6CCAmUwggJhAg
BPwrOKAKHl	3 May 2019 at 17:52:05 UTC	3 May 2019 at 17:52:05 UTC	UxLn8bOHcj, BRdZ9u9VlQ	Fri Ma…	MIICewIBADCCAnQGCSqGSIb3DQEHA6CCAmUwggJhAg
ZLz6xfwUvg	3 May 2019 at 17:41:48 UTC	3 May 2019 at 17:41:48 UTC	UxLn8bOHcj, BRdZ9u9VlQ	Fri Ma…	MIICewIBADCCAnQGCSqGSIb3DQEHA6CCAmUwggJhAg

If the attacker tried to log in the database server which is back4app by hacking the username and the password the attacker will not be able to retrieve or understand anything about the message because the body in the Message Class is encrypted and only the sender and the receiver have the right to decrypt and see the messages so the system is able to protect the messages

Libraries/Frameworks/Dependencies:

1. SupportLibrary
2. ConstraintLayout
3. RxJava
4. RxAndroid
5. Retrofit
6. Gson:For enabling Json Data like package.json and these data are useful as some of these data like token keys,app id ,private and public keys
7. ButterKnife
8. NetworkTracker:For Tracking Network and enable the request to be networked by the server
9. Firebase
10. Nucleus
11. Back4appCore:To access the core of Back4App Server so that to create class and link these classes the android classes
12. Back4appLiveQuery:To acess the Web Hosting and Live Query here you enter the subdomain that you register and for seeing the session details and host quering details.
13. VirgilSecurity: For accessing the Virgil sdk and the api on which they have predefined functions that is accessed and modified in the project

References

1. https://www.back4app.com
2. https://virgilsecurity.com
3. https://github.com
4. https://stackoverflow.com

5. https://maven.apache.org/plugins/maven-dependency-plugin/list-mojo.html

6. https://developer.android.com/docs