Design and Analysis of Algorithms(DAA) Project Report

Customized Chat Application using RSA Algorithm

Our team :-

Arpan Sadhu, Pratyugna Manna, Suvam Mukhopadhyay, Shubham Kumar Singh and **Ipsita Sarkar**

Department of Information Technology(IT)

Institute of Engineering and Management, Kolkata, West Bengal-700091

Under the guidance and mentorship of

Prof. Susovan Jana, Prof. Avipsita Chatterjee and **Prof. Pulak Baral**

February 2022- May 2022

Submitted in partial fulfillment of the requirements of obtaining the degree of

Bachelor of Technology in Information Technology(IT)

Acknowledgements

First and foremost, we would like to thank our esteemed guides and mentors, Prof. Susovan Jana, Prof. Avipsita Chatterjee and Prof. Pulak Baral, for having suggested the topic of our project and for their constant valuable support and guidance, without which we would not have been able to attempt this project.

Next we would like to thank the faculty members of IT department for providing us the much required support and resources which assisted us in completing this project on time.

Last but not the least, we would also thank our friends and classmates who always supported and motivated us to complete this project.

Arpan Sadhu, ***Pratyugna Manna,*** ***Suvam Mukhopadhyay,***

Shubham Kumar Singh *and* ***Ipsita Sarkar***

Introduction

This project Customized Chat Application using RSA algorithm has been developed on **NodeJs** and **Javascript**. The android framework has been created by using **Kotlin**. The main aim of the simple project is to develop web application as per the user requirements. The main objective for developing this Chat Application project is to create a social media application for users of similar interest the application on which we have chat between groups or individuals calls or messages. It can help for managing communication between friends, employees, family, customers, etc. This project provides a lot of features to manage in very well manner.

The main objective for developing this project module is to provide all the functionality related to chat user. It tracks all the information and details of the chat user. We have developed all type of CRUD(Create, Read, Update and Delete) operations of the chat user. This is a role based project where admin can perform each and every operations on data but the chat user will be able to view only his/her data, so access level restrictions has also been implemented on the project.

Description

Chat Application

Communication is a mean for peole to exchange messages. It has started since the beginning of human creation. Distant communication began as early as 19th century with the introduction of telephone and telegram. The emergence of computer network and telecommunication technologies bears the same objective that is to allow people to communicate. All this while, much efforts has been drawn towards consolidating the device into one and therefore indiscriminate the sevices. Chatting is a method of using technology to bring people and ideas together despite of the geographical barriers. The technology has been available for years but the acceptance of it was quit recent. Our project is also an example of a chat application.

RSA Algorithm

The RSA algorithm (Rivest-Shamir-Adleman) is the basis of a cryptosystem -- a suite of cryptographic algorithms that are used for specific security services or purposes -- which enables public key encryption and is widely used to secure sensitive data, particularly when it is being sent over an insecure network such as the internet.

Public key cryptography, also known as asymmetric cryptography, uses two different but mathematically linked keys -- one public and one private. The public key can be shared with everyone, whereas the private key must be kept secret.

In RSA cryptography, both the public and the private keys can encrypt a message. The opposite key from the one used to encrypt a message is used to decrypt it. This attribute is one reason why RSA has become the most widely used asymmetric algorithm: It provides a method to assure the confidentiality, integrity, authenticity, and non-repudiation of electronic communications and data storage.

Kotlin

Kotlin is a programming language introduced by JetBrains in 2011, the official designer of the most intelligent Java IDE, named Intellij IDEA. Kotlin is free, has been free and will remain free. It is developed under the Apache 2.0 license and the source code is available on GitHub.

This is a strongly statically typed general-purpose programming language that runs on JVM. In 2017, Google announced Kotlin is an official language for android development. Kotlin is an open source programming language that combines object-oriented programming and functional features into a unique platform.

Currently, Kotlin mainly targets the Java Virtual Machine (JVM), but also compiles to JavaScript. Kotlin is influenced by other popular programming languages such as Java, C#, JavaScript, Scala and Groovy.

Android Studio

Android Studio is the official Integrated Development Environment (IDE) for android application development. Android Studio provides more features that enhance our productivity while building Android apps.

Android Studio was announced on 16th May 2013 at the Google I/O conference as an official IDE for Android app development. It started its early access preview from version 0.1 in May 2013. The first stable built version was released in December 2014, starts from version 1.0.

Since 7th May 2019, Kotlin is Google's preferred language for Android application development. Besides this, other programming languages are supported by Android Studio.

Algorithm

RSA Algorithm:-

RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption. It was invented by Rivest, Shamir and Adleman in year 1978 and hence name RSA algorithm.

Step 1: Generate the RSA modulus

The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown − N=p*q

Step 2: Derived Number (e)

Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1.

Step 3: Public key

The specified pair of numbers N and e forms the RSA public key and it is made public.

Step 4: Private Key

Private Key d is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows − ed = 1 mod (p-1) (q-1)

Encryption Formula

Consider a sender who sends the plain text message to someone whose public key is (N,e). To encrypt the plain text message in the given scenario, use the following syntax − C = Pe mod N

Decryption Formula

The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver C has the private key d, the result modulus will be calculated as − Plaintext = Cd mod N

Source code

Creation of the model

```
const User = require('../models/user');
const asynchandler = require('express-async-handler');
const jwt = require('jsonwebtoken');
const expressAsyncHandler = require('express-async-handler');
exports.register = asynchandler(async (req, res, next) => {
const { username, email, password, isOnline } = req.body;
try {
    const userExists = await User.findOne({ email })
    if (userExists) {
        return res.json({
            success: false,
            message: "user already exists"
        });
    }
    const user = await User.create({
        username: username,
        email: email,
        password: password,
        isOnline: isOnline
    });
    user.save();
   const token = user.getJwtToken()
   return res.status(200).json({
```

```
            success: true,
            msg: "user created successfully",
            user: user,
            token: token
        })
    }
    catch (error) {
        console.log(error);
        return res.status(400).json({
            success: false,
            msg: `user creation unsuccessfull , ${error.message}`,
            token: null
        })
    }
});

exports.login = asynchandler(async (req, res, next) => {
    const { email, password } = req.body;
    console.log(email, password);
    if (!email || !password) {
        console.log("enter password and email...");
        return res.status(400).json({
            success: false,
            msg: 'enter email and password',
            token: undefined,
            user: undefined
```

```
    });
}
const user = await User.findOne({ email }).select("+password");
if (!user) {
    return res.status(400).json({
        success: false,
        msg: 'User not found in db',
        token: undefined,
        user: undefined
    });
}

const isPasswordValid = await user.isValidatedPassword(password);
if (!isPasswordValid) {
    // const variable =
    // {
    //     success: false,
    //     msg: 'password wrong',
    //     token: undefined,
    //     user: undefined
    // }


    // return res.status(400).json(variable);
    return res.status(200).json({
        success: false,
```

```
        msg: 'password Wrong',
        token: undefined,
        user: user
    });
}

// const payload = {
//     user: {
//         id: user.id
//     }
// }

// jwt.sign(
//     payload, process.env.JWT_SECRET,
//     {
//         expiresIn: 360000
//     }, (err, token) => {
//         if (err) throw err;
const token = await user.getJwtToken()
console.log(token);
res.status(200).json({
    success: true,
    msg: 'User logged in',
    token: token,
    user: user
});
```

```
    // }
 // )
})
exports.isOnlineToggler = expressAsyncHandler(async (req, res, next) => {
    const user = await User.findById(req.user._id);
    const isOnline = user.isOnline;
    var loginstatus = false;
     if (isOnline == 0) {
        user.isOnline = 1;
        loginstatus = true;
    } else {
        user.isOnline = 0;
        loginstatus = false;
    }
    user.save({ validateBeforeSaving: false })
    return res.json({
        success: true,
        msg: `user is ${loginstatus ? "Online" : "Offline"} now`
    })
    // next();
})
```

Implementation of RSA Algorithm

```
const User = require('../models/user');
const asynchandler = require('express-async-handler');
const jwt = require('jsonwebtoken');
```

```
const expressAsyncHandler  = require('express-async-handler');
exports.register = asynchandler(async (req, res, next) => {
const { username, email, password, isOnline } = req.body;
try {
       const userExists = await User.findOne({ email })
       if (userExists) {
          return res.json({
             success: false,
             message: "user already exists"
          });
       }
       const user = await User.create({
          username: username,
          email: email,
          password: password,
          isOnline: isOnline
       });
       user.save();
      const token = user.getJwtToken()
      return res.status(200).json({
       success: true,
        msg: "user created successfully",
          user: user,
          token: token
       })
   }
```

```
    catch (error) {
        console.log(error);
        return res.status(400).json({
            success: false,
            msg: `user creation unsuccessfull , ${error.message}`,
            token: null
        })
    }
});

exports.login = asynchandler(async (req, res, next) => {
    const { email, password } = req.body;
    console.log(email, password);
    if (!email || !password) {
        console.log("enter password and email...");
        return res.status(400).json({
            success: false,
            msg: 'enter email and password',
            token: undefined,
            user: undefined
        });
    }
    const user = await User.findOne({ email }).select("+password");
    if (!user) {
        return res.status(400).json({
            success: false,
```

```
        msg: 'User not found in db',
        token: undefined,
        user: undefined
    });
}
const isPasswordValid = await user.isValidatedPassword(password);
if (!isPasswordValid) {
    // const variable =
    // {
    //     success: false,
    //     msg: 'password wrong',
    //     token: undefined,
    //     user: undefined
    // }
    // return res.status(400).json(variable);
    return res.status(200).json({
        success: false,
        msg: 'password Wrong',
        token: undefined,
        user: user
    });
}

// const payload = {
//     user: {
//         id: user.id
```

```
    //     }
    // }

    // jwt.sign(
    //     payload, process.env.JWT_SECRET,
    //     {
    //         expiresIn: 360000
    //     }, (err, token) => {
    //         if (err) throw err;
    const token = await user.getJwtToken()
    console.log(token);
    res.status(200).json({
        success: true,
        msg: 'User logged in',
        token: token,
        user: user
    });
    // }
    // )})

exports.isOnlineToggler = expressAsyncHandler(async (req, res, next) => {
    const user = await User.findById(req.user._id);
    const isOnline = user.isOnline;
    var loginstatus = false;

    if (isOnline == 0) {
```

```
    user.isOnline = 1;
    loginstatus = true;
  } else {
    user.isOnline = 0;
    loginstatus = false;
  }
  user.save({ validateBeforeSaving: false })
  return res.json({
    success: true,
    msg: `user is ${loginstatus ? "Online" : "Offline"} now`
  })
  // next();})
```

Encryption and Decryption of the messages

```
const User = require('../models/user');
const asynchandler = require('express-async-handler');
const jwt = require('jsonwebtoken');
const expressAsyncHandler = require('express-async-handler');
exports.register = asynchandler(async (req, res, next) => {
const { username, email, password, isOnline } = req.body;
try {
    const userExists = await User.findOne({ email })
    if (userExists) {
      return res.json({
        success: false,
```

```
            message: "user already exists"
        });
      }
      const user = await User.create({
        username: username,
        email: email,
        password: password,
        isOnline: isOnline
      });
      user.save();
     const token = user.getJwtToken()
     return res.status(200).json({
        success: true,
        msg: "user created successfully",
user: user,
 token: token
      })
  }
  catch (error) {
      console.log(error);
      return res.status(400).json({
        success: false,
        msg: `user creation unsuccessfull , ${error.message}`,
        token: null
      })
  }
```

```
});
exports.login = asynchandler(async (req, res, next) => {
   const { email, password } = req.body;
   console.log(email, password);
   if (!email || !password) {
      console.log("enter password and email...");
      return res.status(400).json({
         success: false,
         msg: 'enter email and password',
         token: undefined,
         user: undefined
      });
   }
   const user = await User.findOne({ email }).select("+password");
   if (!user) {
      return res.status(400).json({
         success: false,
         msg: 'User not found in db',
         token: undefined,
         user: undefined
      });
   }

   const isPasswordValid = await user.isValidatedPassword(password);
   if (!isPasswordValid) {
      // const variable =
```

```
    // {
    //     success: false,
    //     msg: 'password wrong',
    //     token: undefined,
    //     user: undefined
    // }


    // return res.status(400).json(variable);
    return res.status(200).json({
        success: false,
        msg: 'password Wrong',
        token: undefined,
        user: user
    });
}

// const payload = {
//     user: {
//         id: user.id
//     }
// }

// jwt.sign(
//     payload, process.env.JWT_SECRET,
//     {
```

```
    //        expiresIn: 360000
    //    }, (err, token) => {
    //        if (err) throw err;
    const token = await user.getJwtToken()
    console.log(token);
    res.status(200).json({
        success: true,
        msg: 'User logged in',
        token: token,
        user: user
    });
    // }
    // )
})
exports.isOnlineToggler = expressAsyncHandler(async (req, res, next) => {
    const user = await User.findById(req.user._id);
    const isOnline = user.isOnline;
    var loginstatus = false;
    if (isOnline == 0) {
        user.isOnline = 1;
        loginstatus = true;
    } else {
        user.isOnline = 0;
        loginstatus = false;
    }
    user.save({ validateBeforeSaving: false })
```

```
    return res.json({
        success: true,
        msg: `user is ${loginstatus ? "Online" : "Offline"} now`
    })
    // next();
})
```

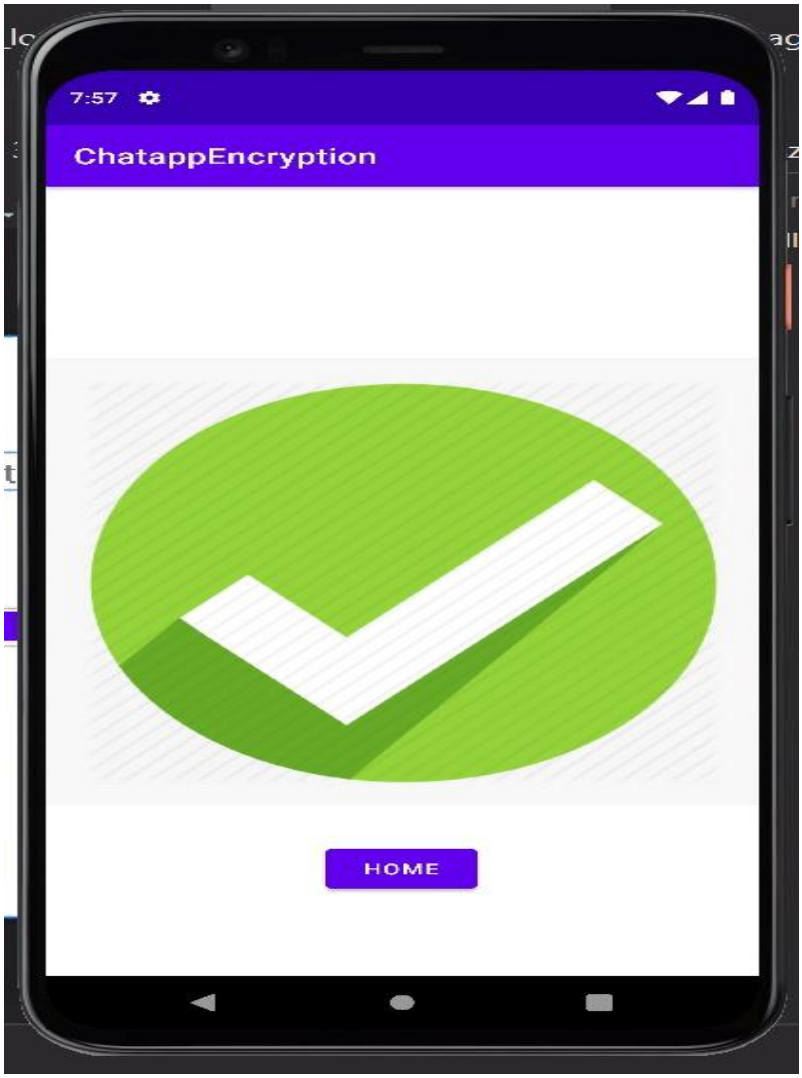
Idea Implementation and Results

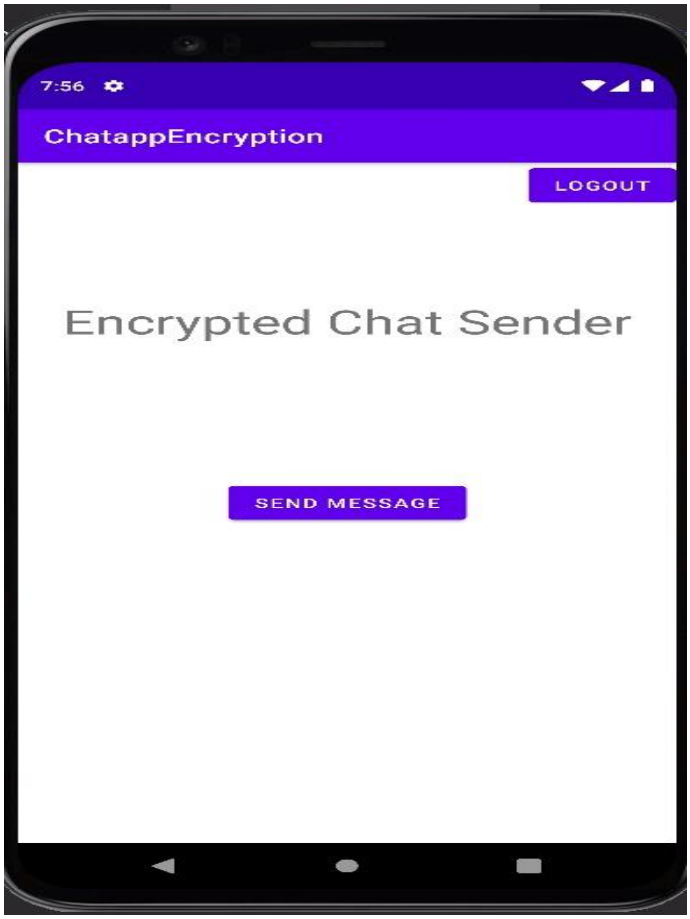
```
const mongoose = require('mongoose');  1M (gzipped: 280.7k)
const bcrypt = require('bcryptjs')  21.6k (gzipped: 9.8k)

const jwt = require('jsonwebtoken')  40.7k (gzipped: 12k)

const userSchema = new mongoose.Schema({
    username: {
        type: String,
        required: true,
        unique: true
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true

    },
    isOnline: {
        type: Number,
        default: 0
    }
```

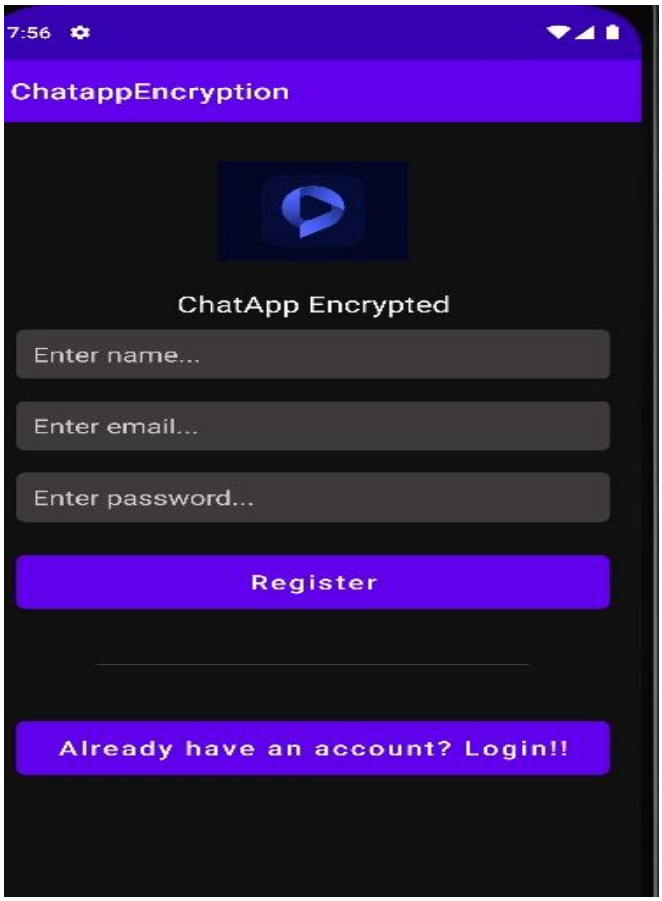
```
        })
        chatMessage_Encry.save()

        const chatMessage_Decry = await chat_Decry.create({
            sender: req.user._id,
            content: decryptData,
            receiver: sendMessageTo
        })
        chatMessage_Decry.save()
        return res.status(200).json({
            success: true,
            msg: "chat sent successfully",
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE node

```
POST /api/v1/sendchats 200 260 - 112.814 ms
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYyNjc3NGE4M2Q0YWFiZDFhMjY5N2Q4MSIsImlhdCI6MTY1MTY3NTc2NiwiZXhwIjoxNjUxOTM0OTY2fQ
.Yw0-9b13mL8TvYokqI4A9GkcBt8mKxEkKSkuxbQKOdU THIS IS TOKEN
{
  _id: new ObjectId("626774a83d4aabd1a2697d81"),
  username: 'a',
  email: 'a@a.com',
  password: '$2a$10$qpJ879PRnQWtsyuUAowuweWCu377e4bxA2b2Ag2xfP5QQL2Za6pKu',
  isOnline: 1,
  __v: 0
}
{
  message: 'hellooooooooooooo',
  sendMessageTo: '6267f6fc613ceb06e885ccd7'
}
hellooooooooooooo
encypted data:  id/X+XJg+6Otx+68i83ws6tMJzvSjZZkskjNwj8Zj3KrfHHyeZbFhlW6TOCN3c4Us7PK57AktygwzwMpocrI7Rz41Dl0tbSqUWz9atNKy+bmBfv
gGgyM2GfpqCVS4Dy59isvNuYHukHn0tNKVcE5Nw7UqlaijS0fJehoRJUooI7GxIYecDcTHQ5ujRgCMofvMm9otbnqtfuLAgsvViFnyK1JfWKfUFSOstA8IyNhDtVDL1
ABI2kjzt8EMJo2efK6sgrZRcNy5wqr51J2qOzdEYDvR8qgysPx7ZsoFMx6MqzYX+U2oll59ziu5vwXwUGO2GThOgN2kdWRJlbtlF51Qg==
decrypted data:  hellooooooooooooo
```




7:56
ChatappEncryption
LOGOUT
Encrypted Chat Sender
SEND MESSAGE


7:56
ChatappEncryption
ChatApp Encrypted
Enter name...
Enter email...
Enter password...
Register
Already have an account? Login!!

Discussion

An Android Chat Application using RSA algorithm can thus be implemented by using Node Js and Javascript. The programming language used for writing the code is kotlin. Android Studio is used for developing the frontend UI/UX design using XML.

The main objective for developing this Chat Application project is to create a social media application for users of similar interest the application on which we have chat between groups or individuals calls or messages. It can help for managing communication between friends, employees, family, customers, etc.

The functionalities and responsiveness of this application can further be improved. We can use more complex algorithms like Advanced encryption algorithm(AES) and Data encryption algorithm(DES). But writing these algorithms and implementing is quite complicated.

References

1) Geeksforgeeks.org
2) Android development youtube playlist by Code With Harry
3) Stackoverflow.com
4) Android development course by Edureka
5) Official Android Developer Website
6) My Codeschool youtube channel
7) Android Tutorials by Java Code Geeks