**ACKNOWLEDGEMENT**

“It is not possible to prepare this Project without the assistance and encouragement of other people. This one is certainly no exception.” We would like to thank my teacher Dr. Partha Sarathi Goswami for her encouragement, guidance and support which helped me to complete our project due time. A Special acknowledgement goes to our principal Narendra Nath Singha who gave us this golden opportunity of this wonderful project, which also helped me in doing a lot of projects and I came to know about so many new things. In the end, I want to thank my friends who displayed appreciation for my work and motivated me to continue my work.

Acknowledge by

Koushik Karan

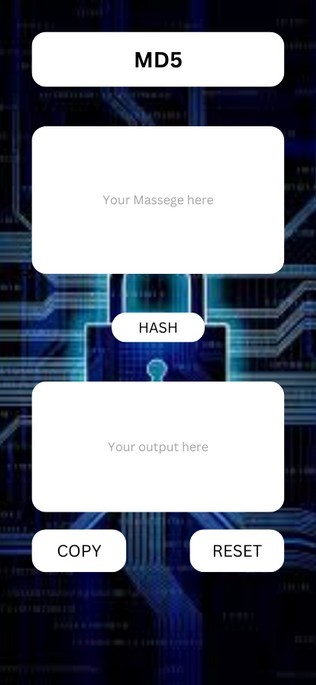
REG.No.-D20212263

1.**Objective of the work**: To secure the data communication many text encryption algorithms are previously made but no common people use the algorithm in their text and secure it. In today’s word 90% people have smartphone as because we create an application for smartphone with which a common people can encrypt and decrypt their text and make secure it with their own hand.

2.**Plan of Work**: We want to create an application for mobile device which is mainly encrypt data with various algorithm and hashing technique and decrypt the encrypted data. Main plan of the work is text message encryption. This application work as an encryption medium which is mainly used for the text or private message encryption and securely send to the other party. This app has 2 main Encryption Algorithm AES (Advance Encryption Standard) and DES (Data Encryption Standard) also it has 2 main Hashing Algorithm MD5 and SHA-256(Secure Hashing Algorithm.

3. **Motivation of work:** This system can be used by everyone who wants to send some confidential text via social media. And main motivation is that anybody is done this text encryption with their hand.

# Brief Description of the work



**Homepage Encryption Page** **Hash page**

In this App, the text entered in the edit text is encrypted using the app instance and the encrypted text is shown. On clicking decrypt button, the encrypted text is decrypted and real text is shown. Users communicate over all social media, but messages aren't secured when it passes through network. Intruder can access user’s message easily. We want to secure users’ communication over all social media. So here we proposed a system where user will enter the plain text and choose the algorithm type from **AES, DES, MD5**, and supply the key, a chipper text is going to be formed which will be sent via any communication application and user can decrypt the text by selecting an equivalent algorithm type and must enter an equivalent sender secret key. User can use our application and may enter the plain text and must select the algorithm type and must enter the key to encrypt the message and receiver can decrypt the message by specifying an equivalent algorithm used for encryption and must use an equivalent secret key employed by sender. Intruder will find difficult to decrypt the message. By using this method you'll double make sure that your secret message is shipped securely without outside interference of hackers or crackers. If sender sends this image publicly others won't know what's it, and it'll be received by receiver. You will need the key and algorithm type to decrypt the hidden text.

# S/W and H/W Requirements

1. Software: Windows 10, Linux, Mac, Windows XP, Windows 7(ultimate, enterprise) Android Studio
2. Hardware: Processor – i3

Hard Disk – 5 GB

Memory – 1GB RAM

Android Phone with 10 and higher.

# Algorithm of the Application

Main Application

Step 1: Select Encryption or Hash Step 2: If Encryption:

Step 2.1: EN= input (“Encryption Stranded (AES/TDES)”)

Step 2.2: T = input (“Your Message here”)

Step 2.3: K = input(“Key”)

Step 2.4: MT = input (“Select (Encrypt/Decrypt)”) Step 2.5: if EN == AES && MT == Encrypt:

Step 2.5.1: Derive the set of round keys from the cipher key Step 2.5.2: Initialize the state array with the block data (plaintext).

Step 2.5.3: Add the initial round key to the starting state array.

Step 2.5.4: Perform nine rounds of state manipulation.

Step 2.5.5: Perform the final round of state manipulation.

Step 2.5.6: Print the final state array as a cipher text or encrypted text Step 2.6: else if EN == TDES && MT == Encrypt:

Step 2.6.1: Encrypt the plaintext blocks using single DES with key K Step 2.6.2: Now decrypt the output of the step 1 using DES with k2.

Step 2.6.3: Finally, encrypt the output of the step 2 using single DES with k3.

Step 2.6.4: The output of step 3 is the ciphertext.

Step 2.6.5: Print the ciphertext as output.

Step 2.7: else if EN == AES && MT == Decrypt:

Step 2.7.1: Derive the set of round keys from the plain key.

Step 2.7.2: Initialize the state array with the block data (ciphertext).

Step 2.7.3: Perform nine step reverse state manipulation.

Step 2.7.4: Perform the final reverse state manipulation Step 2.7.5: Print the final manipulation array as plaintext.

Step 2.8: else if EN == TDES && MT == Decrypt:

Step 2.8.1: Decrypt the cipher text with k1.

Step 2.8.2: Decrypt the text with k2 which is the result of step 1

Step 2.8.3: Decrypt the text with k3 which is the result of step 2

Step 2.8.4: Print the 3rd step as the plain text Step 3: Else if select Hash:

Step 3.1: HS = input (“Select hash algo(md5/SHA-256)”)

Step 3.2: PT = input (“Enter your message here”)

Step 3.3: SU = input (“Sault”)

Step 3.4: HASH= input (“Enter hashing”)

Step 3.5: if HS == MD5 && HASH== Hashing:

Step 3.5.1: Adding the Sault with the original message

Step 3.5.2: Add the length bit with the output of the first step

Step 3.5.3: Start using 32bit buffer

Step 3.5.4: Perform 64 operations in 4 round (AND, OR, XOR Operations)

Step 3.5.5: Print the final text as an output

Step 3.6: if HS == SHA-256 && HASH==Hashing:

Step 3.6.1: Arbitrary of the message length ‘M’

Step 3.6.2: Convert the message into binary

Step 3.6.3: Padding a message M

Step 3.6.4: Parsing the Message M [(l+1+k) =448 mod 512 {64}]

Step 3.6.5: Setting initial hash values H0-H7

Step 3.6.6: Prepare the message schedule

Step 3.6.7: Initialize a,b,c,d,e,f,g and h

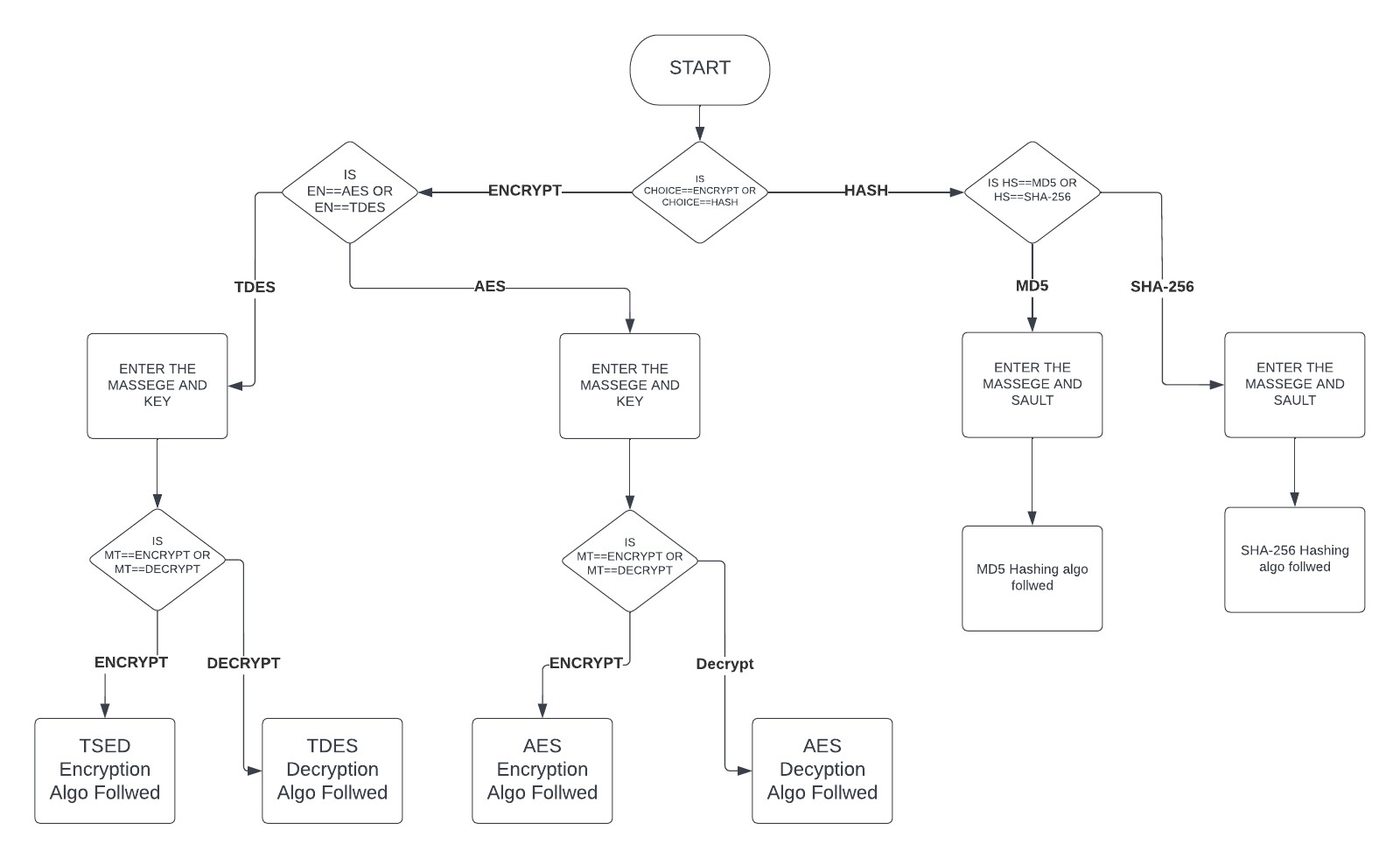
Step 3.6.8: Compute for the intermediate hash values

Step 3.6.9: Append hash values H0, H1,H2,H3,H4,H5,H6,H7

Step 3.6.10: 256-bit message output come

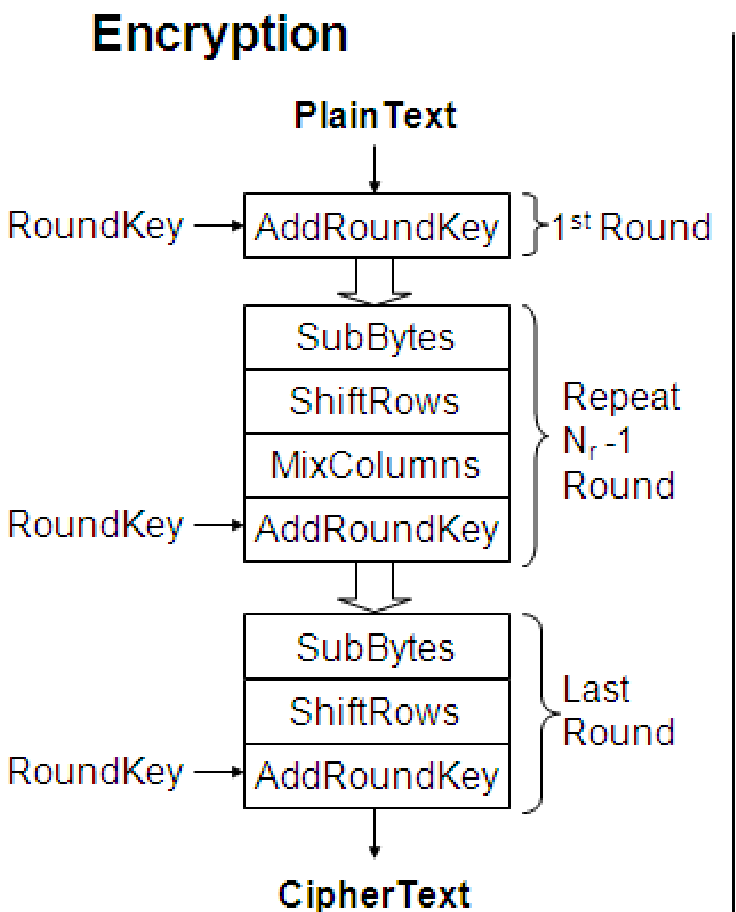
**Flow Diagram**

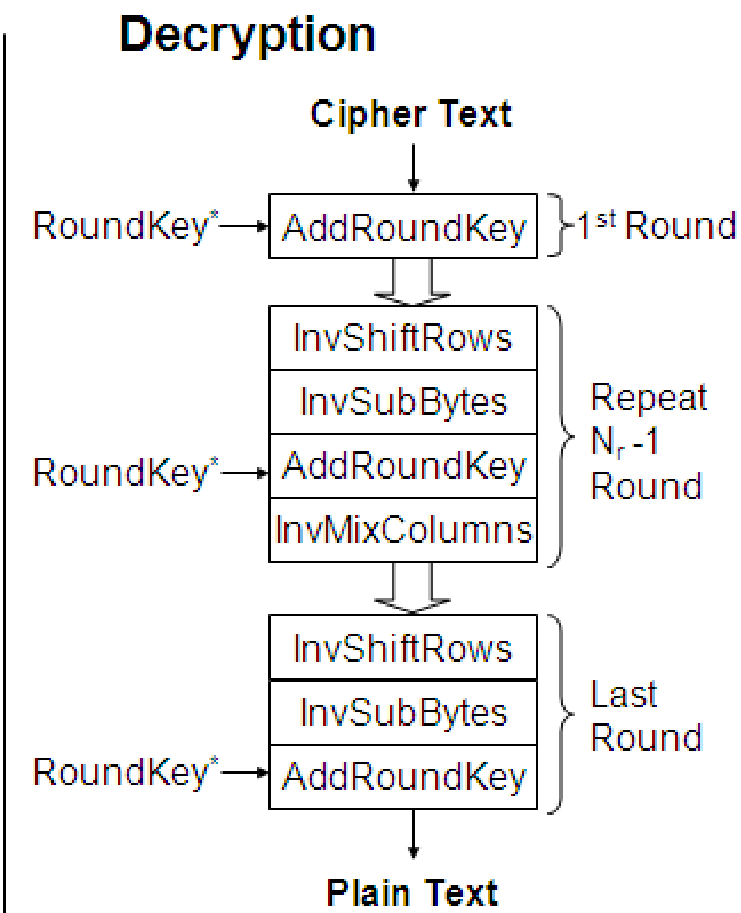
**Main Application Flow Diagram:**



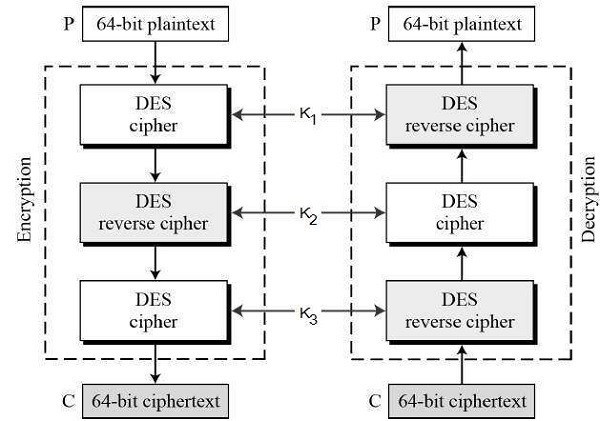
**Encryption Algorithm Flow Diagram:**

**AES Algorithm Flow Diagram:**



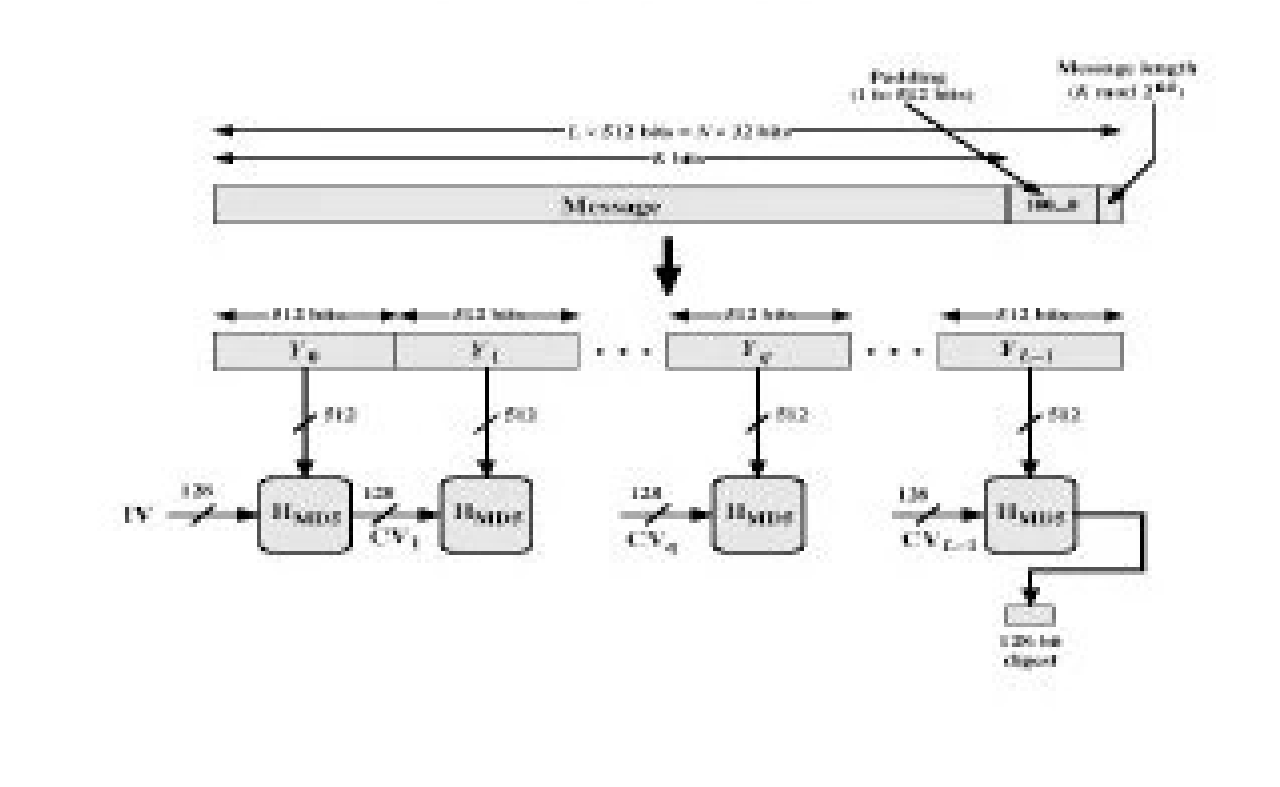


**TDES Algorithm Flow Diagram:**

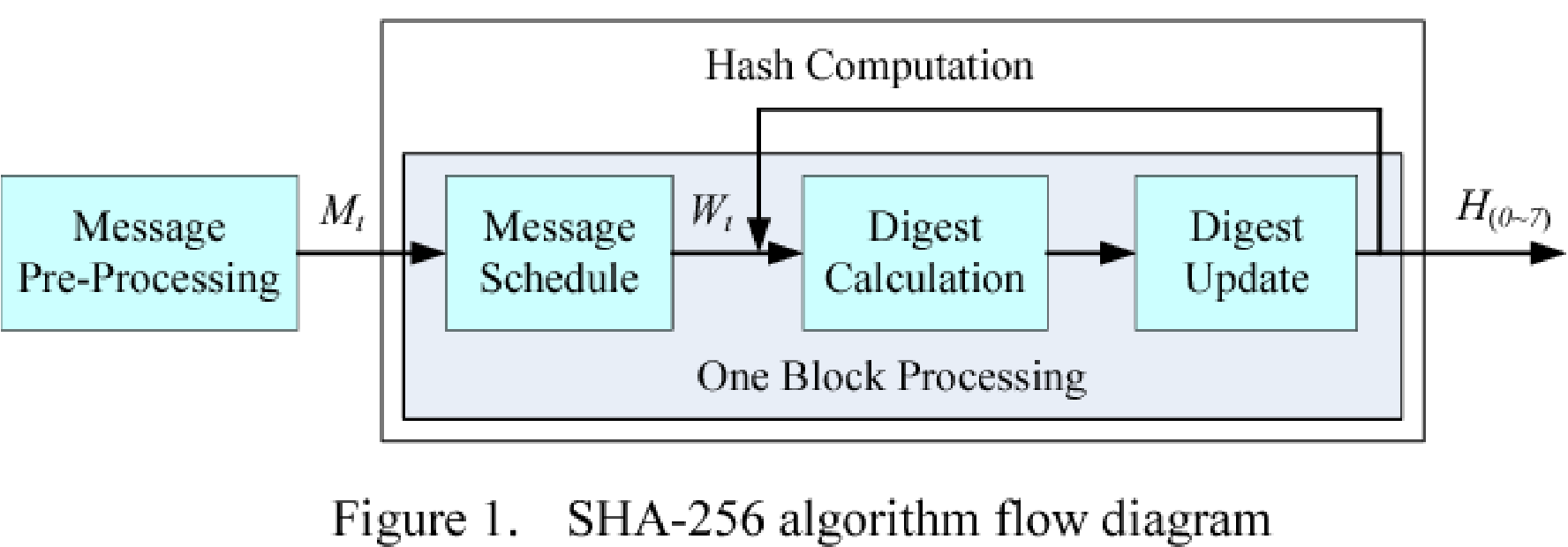


**Hash Algorithm Flow Diagram:**

**MD5 Encryption Flowdiagram:**



**SHA-256 Encryption Flowdiagram:**



# CODING

**Code for Encryption**: AES Algorithm Code:

package Encryption.Algorithms; import android.util.Base64; import android.util.Log; import java.security.MessageDigest; import javax.crypto.Cipher; import javax.crypto.spec.SecretKeySpec; public class AES {

public String AESencrypt ( byte[] key, byte[] clear) throws Exception {

MessageDigest md = MessageDigest.*getInstance*("md5"); byte[] digestOfPassword = md.digest(key);

SecretKeySpec skeySpec = new SecretKeySpec(digestOfPassword, "AES"); Cipher cipher = Cipher.*getInstance*("AES/ECB/PKCS7Padding"); cipher.init(Cipher.*ENCRYPT\_MODE*, skeySpec); byte[] encrypted = cipher.doFinal(clear);

return Base64.*encodeToString*(encrypted, Base64.*DEFAULT*);

} public String AESdecrypt (String key,byte[] encrypted) throws Exception { MessageDigest md = MessageDigest.*getInstance*("md5");

byte[] digestOfPassword = md.digest(key.getBytes("UTF-16LE")); SecretKeySpec skeySpec = new SecretKeySpec(digestOfPassword, "AES"); Cipher cipher = Cipher.*getInstance*("AES/ECB/PKCS7Padding"); cipher.init(Cipher.*DECRYPT\_MODE*, skeySpec); byte[] decrypted = cipher.doFinal(encrypted); return new String(decrypted, "UTF-16LE");

} }

**TDES Encryption Code**:

package Encryption.Algorithms; import android.util.Base64; import android.util.Log; import java.security.MessageDigest; import javax.crypto.Cipher; import javax.crypto.spec.SecretKeySpec; public class DES {

public String encrypt ( byte[] key, byte[] clear) throws Exception {

MessageDigest md = MessageDigest.*getInstance*("md5"); byte[] digestOfPassword = md.digest(key);

SecretKeySpec skeySpec = new SecretKeySpec(digestOfPassword, "DESede"); Cipher cipher = Cipher.*getInstance*("DESede/CBC/PKCS5Padding"); cipher.init(Cipher.*ENCRYPT\_MODE*, skeySpec); byte[] encrypted = cipher.doFinal(clear);

return Base64.*encodeToString*(encrypted, Base64.*DEFAULT*);

} public String decrypt (String key,byte[] encrypted) throws Exception {

MessageDigest md = MessageDigest.*getInstance*("md5");

byte[] digestOfPassword = md.digest(key.getBytes("UTF-16LE")); SecretKeySpec skeySpec = new SecretKeySpec(digestOfPassword, "DESede"); Cipher cipher = Cipher.*getInstance*("DESede/CBC/PKCS5Padding"); cipher.init(Cipher.*DECRYPT\_MODE*, skeySpec); byte[] decrypted = cipher.doFinal(encrypted); return new String(decrypted, "UTF-16LE");

} }

**Encryption Code**:

package Encryption; import android.content.Context; import android.os.Bundle; import android.text.InputType; import android.util.Base64; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.view.WindowManager; import android.widget.Button; import android.widget.EditText; import android.widget.TextView; import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity; import androidx.constraintlayout.widget.ConstraintLayout; import androidx.constraintlayout.widget.ConstraintSet; import androidx.fragment.app.Fragment; import com.example.Algorithms.R; import java.nio.charset.StandardCharsets; import java.security.KeyPair; import java.security.KeyPairGenerator; import java.security.MessageDigest; import java.security.NoSuchAlgorithmException; import java.security.PublicKey; import Encryption.Algorithms.AES; import Encryption.Algorithms.DES; import Encryption.Algorithms.PlayFair; import Encryption.Algorithms.Vigenere; import Encryption.Algorithms.Caesarcipher; public class EncryptionMain extends Fragment { private String message; private String key; private Button Switch; private Button Encrypt\_Buuton; private Button Decrypt\_Buuton; private PlayFair p; private TextView Answer; private EditText Textfield\_Text; private EditText Textfield\_Key; private TextView Matrix\_value; private TextView Play\_Fair\_VALUE; private View view; @Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

view = inflater.inflate(R.layout.*encryption\_main*, container, false); ((AppCompatActivity) getActivity()).getSupportActionBar().hide();

getActivity().getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*,

WindowManager.LayoutParams.*FLAG\_FULLSCREEN*);

Switch = view.findViewById(R.id.*Swtich*);

Encrypt\_Buuton = view.findViewById(R.id.*Encrypt\_Buuton*);

Decrypt\_Buuton = view.findViewById(R.id.*Decrypt\_Buuton*);

Answer = view.findViewById(R.id.*Answer*);

Textfield\_Text = view.findViewById(R.id.*TextArea*);

Textfield\_Key = view.findViewById(R.id.*Key*);

Matrix\_value = view.findViewById(R.id.*Matrix*);

Play\_Fair\_VALUE = view.findViewById(R.id.*Play\_Fair\_VALUE*); return view;

}

public void encrypt(View view) throws Exception {

if (Textfield\_Text.length() == 0) {

Toast.*makeText*(view.getContext(), "Enter a message to Encrypt",

Toast.*LENGTH\_SHORT*).show(); return;

}

message = String.*valueOf*(Textfield\_Text.getText()); key = String.*valueOf*(Textfield\_Key.getText()); String Algorithm = String.*valueOf*(Switch.getText()); switch (Algorithm) { case "Advanced Encryption Standard":

AES aes = new AES();

String enc = aes.AESencrypt(key.getBytes("UTF-16LE"), message.getBytes("UTF-16LE")); Answer.setText(enc); break; case "Triple Data Encryption Standard":

DES des = new DES();

String encData = des.encrypt(key.getBytes("UTF-16LE"), message.getBytes("UTF-16LE"));

Answer.setText(encData); break; case "Caesar Cipher": if (key.isEmpty()) {

Toast.*makeText*(view.getContext(), "Enter a key to Encrypt", Toast.*LENGTH\_SHORT*).show(); return;

} if (key.length() > 26) {

Toast.*makeText*(view.getContext(), "The Key must be less than

26 characters", Toast.*LENGTH\_SHORT*).show(); return;

}

Caesarcipher c = new Caesarcipher();

Answer.setText(c.caesarcipherEnc(message,

Integer.*parseInt*(key))); break; case "Vigenere Cipher": if (Textfield\_Key.length() == 0) {

Toast.*makeText*(view.getContext(), "Enter a key to Encrypt", Toast.*LENGTH\_SHORT*).show(); return;

}

for (char i : message.toUpperCase().toCharArray()) { if (i < 'A' || i > 'Z') {

Toast.*makeText*(view.getContext(), "Only Letters are allowed here", Toast.*LENGTH\_SHORT*).show(); return;

} }

for (char i : key.toUpperCase().toCharArray()) {

if (i < 'A' || i > 'Z') {

Toast.*makeText*(view.getContext(), "Only Letters are allowed here", Toast.*LENGTH\_SHORT*).show(); return;

}

}

Vigenere v = new Vigenere();

Answer.setText(v.Vigenereencrypt(message, key)); break; case "Play Fair": try {

p = new PlayFair("");

Play\_Fair\_VALUE.setText(p.Encrypt(message, key));

Matrix\_value.setText(p.getT1());

} catch (Exception e) {

Toast.*makeText*(view.getContext(), "Only Letters are allowed here", Toast.*LENGTH\_SHORT*).show();

} break;

} }

public void decrypt(View view) throws Exception {

if (Textfield\_Text.length() == 0) {

Toast.*makeText*(view.getContext(), "Enter a message to Decrypt",

Toast.*LENGTH\_SHORT*).show(); return;

}

message = String.*valueOf*(Textfield\_Text.getText()); key = String.*valueOf*(Textfield\_Key.getText()); String SwitchValue = Switch.getText().toString(); switch (SwitchValue) { case "Advanced Encryption Standard": AES aes = new AES(); try {

String decData = aes.AESdecrypt(key,

Base64.*decode*(message.getBytes("UTF-16LE"), Base64.*DEFAULT*));

Answer.setText(decData);

} catch (Exception e) {

Toast.*makeText*(view.getContext(), "Your key is wrong",

Toast.*LENGTH\_SHORT*).show(); } break; case "Triple Data Encryption Standard": DES des = new DES(); try {

String decData = des.decrypt(key,

Base64.*decode*(message.getBytes("UTF-16LE"), Base64.*DEFAULT*));

Answer.setText(decData);

} catch (Exception e) {

Toast.*makeText*(view.getContext(), "Your key is wrong",

Toast.*LENGTH\_SHORT*).show(); } break; case "Caesar Cipher": if (Textfield\_Key.length() == 0) {

Toast.*makeText*(view.getContext(), "Enter a key", Toast.*LENGTH\_SHORT*).show(); return;

} if (Integer.*parseInt*(key) >= 26) {

Toast.*makeText*(view.getContext(), "The Key must be less than

26 characters", Toast.*LENGTH\_SHORT*).show(); return;

}

Caesarcipher c = new Caesarcipher();

Answer.setText(c.caesarcipherDec(message,

Integer.*parseInt*(key))); break; case "Vigenere Cipher":

if (Textfield\_Key.length() == 0) {

Toast.*makeText*(view.getContext(), "Enter a key to Decrypt", Toast.*LENGTH\_SHORT*).show(); return;

}

for (char i : message.toUpperCase().toCharArray()) { if (i < 'A' || i > 'Z') {

Toast.*makeText*(view.getContext(), "Only Letters are allowed here", Toast.*LENGTH\_SHORT*).show(); return;

} }

for (char i : key.toUpperCase().toCharArray()) {

if (i < 'A' || i > 'Z') {

Toast.*makeText*(view.getContext(), "Only Letters are allowed here", Toast.*LENGTH\_SHORT*).show(); return;

}

}

Vigenere v = new Vigenere();

Answer.setText(v.Vigeneredecrypt(message, key)); break; case "Play Fair": try {

Play\_Fair\_VALUE.setText(p.Decrypt(message, key));

Matrix\_value.setText(p.getT1());

} catch (Exception e) {

Toast.*makeText*(view.getContext(), "Only Letters are allowed here", Toast.*LENGTH\_SHORT*).show();

} break;

} }

public void switchAlgho(View view) {

reset(null);

String SwitchValue = Switch.getText().toString(); switch (SwitchValue) {

case "Advanced Encryption Standard":

Switch.setText("Triple Data Encryption Standard"); break; case "Triple Data Encryption Standard":

Textfield\_Key.setInputType(InputType.*TYPE\_CLASS\_NUMBER*);

Switch.setText("Caesar Cipher"); break; case "Caesar Cipher":

Textfield\_Key.setInputType(InputType.*TYPE\_CLASS\_TEXT*);

Switch.setText("Vigenere Cipher"); break; case "Vigenere Cipher":

Textfield\_Key.setVisibility(View.*VISIBLE*);

Answer.setVisibility(View.*GONE*);

Matrix\_value.setVisibility(View.*VISIBLE*);

Play\_Fair\_VALUE.setVisibility(View.*VISIBLE*);

Switch.setText("Play Fair"); break; case "Play Fair":

Answer.setVisibility(View.*VISIBLE*);

Matrix\_value.setVisibility(View.*GONE*);

Play\_Fair\_VALUE.setVisibility(View.*GONE*); Switch.setText("Advanced Encryption Standard"); break;

} } public void reset(View view) { Textfield\_Text.setText("");

Textfield\_Key.setText("");

Answer.setText("");

Play\_Fair\_VALUE.setText(""); Matrix\_value.setText(""); if(view!=null)

Toast.*makeText*(view.getContext(), "All data has been deleted",

Toast.*LENGTH\_SHORT*).show();

}

public void copyToClipboard(View view) { if (Play\_Fair\_VALUE.length() == 0) {

String copyText = String.*valueOf*(Answer.getText()); if (Answer.length() == 0) {

Toast.*makeText*(view.getContext(), "There is no message to copy",

Toast.*LENGTH\_SHORT*).show(); return;

}

int sdk = android.os.Build.VERSION.*SDK\_INT*; if (sdk < android.os.Build.VERSION\_CODES.*HONEYCOMB*) { android.text.ClipboardManager clipboard =

(android.text.ClipboardManager)

view.getContext().getSystemService(Context.*CLIPBOARD\_SERVICE*); clipboard.setText(copyText);

} else {

android.content.ClipboardManager clipboard =

(android.content.ClipboardManager)

view.getContext().getSystemService(Context.*CLIPBOARD\_SERVICE*); android.content.ClipData clip = android.content.ClipData .*newPlainText*("Your message :", copyText); clipboard.setPrimaryClip(clip);

}

Toast.*makeText*(view.getContext(),

"Your message has be copied", Toast.*LENGTH\_SHORT*).show();

} else {

int sdk = android.os.Build.VERSION.*SDK\_INT*; if (sdk < android.os.Build.VERSION\_CODES.*HONEYCOMB*) { android.text.ClipboardManager clipboard =

(android.text.ClipboardManager)

view.getContext().getSystemService(Context.*CLIPBOARD\_SERVICE*);

clipboard.setText(Play\_Fair\_VALUE.getText().toString());

} else {

android.content.ClipboardManager clipboard =

(android.content.ClipboardManager)

view.getContext().getSystemService(Context.*CLIPBOARD\_SERVICE*);

android.content.ClipData clip = android.content.ClipData

.*newPlainText*("Your message :", Play\_Fair\_VALUE.getText().toString()); clipboard.setPrimaryClip(clip);

}

Toast.*makeText*(view.getContext(), "Your message has be copied",

Toast.*LENGTH\_SHORT*).show();

}

} } **Code for Hash**:

package Hash;

import android.content.Context; import android.os.Bundle; import android.text.InputType; import android.util.Base64; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.view.WindowManager; import android.widget.Button; import android.widget.EditText; import android.widget.TextView; import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity; import androidx.fragment.app.Fragment; import com.example.Algorithms.R; import java.math.BigInteger; import java.security.MessageDigest; import java.security.NoSuchAlgorithmException; import java.security.NoSuchProviderException; import java.security.SecureRandom; public class HashMain extends Fragment { private Button Switch; private Button Hash\_Buuton; private TextView Answer; private EditText Textfield\_Text; private EditText Textfield\_salt;

private String message; private String salt; private View view; @Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

view = inflater.inflate(R.layout.*hash\_main*, container, false); ((AppCompatActivity) getActivity()).getSupportActionBar().hide();

getActivity().getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*,

WindowManager.LayoutParams.*FLAG\_FULLSCREEN*);

Switch = view.findViewById(R.id.*Swtich*);

Hash\_Buuton = view.findViewById(R.id.*hash\_Buuton*);

Answer = view.findViewById(R.id.*Answer*);

Textfield\_Text = view.findViewById(R.id.*TextArea*); Textfield\_salt = view.findViewById(R.id.*salt*); return view;

}

public void hash(View view) throws Exception { if (Textfield\_Text.length() == 0) {

Toast.*makeText*(view.getContext(), "Enter a message to Hash",

Toast.*LENGTH\_SHORT*).show(); return;

}

message = String.*valueOf*(Textfield\_Text.getText()); salt = String.*valueOf*(Textfield\_salt.getText()); String Algorithm = String.*valueOf*(Switch.getText()); String answer=""; switch (Algorithm) { case "MD5": answer=hashText("MD5",salt,message);

Answer.setText(answer); break; case "SHA-256": answer=hashText("SHA-256",salt,message);

Answer.setText(answer); break; case "SHA-512": answer=hashText("SHA-512",salt,message);

Answer.setText(answer); break;

} }

public void switchAlgho(View view) {

reset(null);

String SwitchValue = Switch.getText().toString(); switch (SwitchValue) { case "MD5":

Switch.setText("SHA-256"); break; case "SHA-256":

Switch.setText("SHA-512"); break; case "SHA-512":

Switch.setText("MD5"); break;

}

}

public String hashText(String algo,String salt, String plainText)

throws NoSuchAlgorithmException {

MessageDigest m = MessageDigest.*getInstance*(algo); m.reset();

if (salt.length() != 0) {

m.update(salt.getBytes());

}

m.update(plainText.getBytes()); byte[] digest = m.digest();

BigInteger bigInt = new BigInteger(1,digest);

String hashtext = bigInt.toString(16);

*// Now we need to zero pad it if you actually want the full 32 chars.*

while(hashtext.length() < 32 ){ hashtext = "0"+hashtext;

} return hashtext;

} private byte[] getRandomSalt() throws NoSuchAlgorithmException,

NoSuchProviderException

{

*//Always use a SecureRandom generator*

SecureRandom sr = SecureRandom.*getInstance*("SHA1PRNG", "SUN");

*//Create array for salt*  byte[] salt = new byte[16]; *//Get a random salt*  sr.nextBytes(salt); *//return salt*  return salt;

} public void reset(View view) { Textfield\_Text.setText("");

Textfield\_salt.setText(""); Answer.setText(""); if(view!=null)

Toast.*makeText*(view.getContext(), "All data has been deleted",

Toast.*LENGTH\_SHORT*).show();

} public void copyToClipboard(View view) {

String copyText = String.*valueOf*(Answer.getText()); if (Answer.length() == 0) {

Toast.*makeText*(view.getContext(), "There is no message to copy",

Toast.*LENGTH\_SHORT*).show(); return;

}

int sdk = android.os.Build.VERSION.*SDK\_INT*; if (sdk < android.os.Build.VERSION\_CODES.*HONEYCOMB*) { android.text.ClipboardManager clipboard =

(android.text.ClipboardManager)

view.getContext().getSystemService(Context.*CLIPBOARD\_SERVICE*); clipboard.setText(copyText);

} else {

android.content.ClipboardManager clipboard =

(android.content.ClipboardManager)

view.getContext().getSystemService(Context.*CLIPBOARD\_SERVICE*); android.content.ClipData clip = android.content.ClipData .*newPlainText*("Your message :", copyText); clipboard.setPrimaryClip(clip);

}

Toast.*makeText*(view.getContext(),

"Your message has be copied", Toast.*LENGTH\_SHORT*).show();

} }

**Main Application Code**:

package Main; import android.os.Bundle; import android.view.View;

import androidx.appcompat.app.AppCompatActivity; import androidx.fragment.app.Fragment; import androidx.fragment.app.FragmentManager; import androidx.fragment.app.FragmentTransaction; import com.example.Algorithms.R; import Encryption.EncryptionMain; import Hash.HashMain;

public class MainActivity extends AppCompatActivity {

EncryptionMain encryptionMain;

HashMain hashMain; @Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.*activity\_main*); Fragment fragment = new MainFragment();

FragmentManager fragmentManager = getSupportFragmentManager(); fragmentManager.beginTransaction().replace(R.id.*container*, fragment).commit();

}

public void goToEncryption(View view) { encryptionMain = new EncryptionMain();

FragmentManager manager = getSupportFragmentManager();

FragmentTransaction transaction = manager.beginTransaction();

transaction.setCustomAnimations(android.R.anim.*fade\_in*,android.R.anim.*fade\_out*, android.R.anim.*fade\_in*, android.R.anim.*fade\_out*); transaction.replace(R.id.*container*, encryptionMain); transaction.addToBackStack(null); transaction.commit();

}

public void goToHash(View view) { hashMain = new HashMain();

FragmentManager manager = getSupportFragmentManager();

FragmentTransaction transaction = manager.beginTransaction();

transaction.setCustomAnimations(android.R.anim.*fade\_in*,android.R.anim.*fade\_out*, android.R.anim.*fade\_in*, android.R.anim.*fade\_out*); transaction.replace(R.id.*container*, hashMain); transaction.addToBackStack(null); transaction.commit();

}

public void encryptionButtonClick(View view) { try {

switch (view.getId()) { case R.id.*Swtich*: encryptionMain.switchAlgho(view); break; case R.id.*Encrypt\_Buuton*:

encryptionMain.encrypt(view); break; case R.id.*Decrypt\_Buuton*:

encryptionMain.decrypt(view); break; case R.id.*copy\_button*: encryptionMain.copyToClipboard(view); break; case R.id.*reset\_button*:

encryptionMain.reset(view); break;

} }

catch (Exception e){

e.printStackTrace();

} }

public void HashButtonClick(View view) { try {

switch (view.getId()) { case R.id.*Swtich*: hashMain.switchAlgho(view); break; case R.id.*hash\_Buuton*: hashMain.hash(view); break; case R.id.*copy\_button*: hashMain.copyToClipboard(view); break; case R.id.*reset\_button*: hashMain.reset(view); break;

} }

catch (Exception e){

e.printStackTrace();

}

} }

# Conclusion

The use of Android APP For Text Encryption using Various Algorithms on android application has been structured and executed. The application is running in the cell phone and does not require any extra encryption gadgets. The outcome demonstrated that a common people could encrypt and decrypt their text and make secure it with their own hand by using this application. An architecture was proposed with details about implementation of such a design and a demo was also created. This application is helpful for those users who want to send text messages to each other without the need for extra hardware or physical tokens. Users can be confident that nobody, even not the provider of the service, can read their messages. However due to time limitations, this research can be extended to provide more features to the chat application and better implementation of the infrastructure. In this chapter the conclusion and some of the writer’s suggestions are mentioned in order to open opportunities for future researchers and students to continue this project.