Some useful code of the android application are as following-

```
@TargetApi(Build.VERSION_CODES.M)
public class FingerprintHandler extends
FingerprintManager.AuthenticationCallback {

    private Context context;

    public FingerprintHandler(Context context) {

        this.context = context;

    }

    public void startAuth(FingerprintManager fingerprintManager,
FingerprintManager.CryptoObject cryptoObject) {

        CancellationSignal cancellationSignal = new
CancellationSignal();
        fingerprintManager.authenticate(cryptoObject,
cancellationSignal, 0, this, null);

    }

    @Override
    public void onAuthenticationError(int errorCode, CharSequence
errString) {


this.update(context.getString(R.string.auth_error_try_again) +
errString, false);

    }

    @Override
    public void onAuthenticationFailed() {


this.update(context.getString(R.string.auth_failed_try_again),
false);

    }

    @Override
    public void onAuthenticationHelp(int helpCode, CharSequence
helpString) {
        this.update(context.getString(R.string.error_colon) +
helpString, false);
```

```
    }

    @Override
    public void
onAuthenticationSucceeded(FingerprintManager.AuthenticationResult
result) {
        this.update(context.getString(R.string.auth_successful),
true);
    }

    private void update(String s, boolean b) {

        TextView paraLabel = ((Activity)
context).findViewById(R.id.fingerprintLabel);
        ImageView imageView = ((Activity)
context).findViewById(R.id.fingerprintImage);

        paraLabel.setText(s);

        if (!b) {
            ((PasscodeActivity)context).goTOHome();
            paraLabel.setTextColor(ContextCompat.getColor(context,
R.color.colorAccent));
        } else {
            paraLabel.setTextColor(ContextCompat.getColor(context,
R.color.textColorPrimary));
            imageView.setImageResource(R.mipmap.action_done);
        }

    }
}
```

```
public class PasscodeActivityLM extends AppCompatActivity {

    String TAG = this.getClass().getSimpleName();

    private int[] mPattern = {11, // Backspace Button Position
            9,   // Other Button Position
            R.drawable.ic_exit // OtherButton Source
    };

    private PassGridViewAdapter gridViewAdapter;
    private String pin = "";
    private String pre_pin = "";
    private int cnt;
    private TextView mTitle, mMessage;
    private PrimePreference spc;
```

```
    private int stage = 0;
    private int entry_type;
    private boolean hasDestination = false;

    private LinearLayout mDotsLayout;
    private TextView mDots[];
    private int previous_pos = -1;

    private Animation upTODown;
    private TranslateAnimation translateAnimation;
    private int passcodeSize;
    String tempSize = passcodeSize + "";
    private View setting, skip;
    private ImageView fingerprintImage;
    private TextView fingerprintLabel;
    private boolean isFinishAffinity;
    private boolean isSizeChangeFromThere = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Intent intent = getIntent();
        spc = new PrimePreference(getApplicationContext());
        if (!intent.hasExtra(getString(R.string.from_there_only))
|| !intent.getBooleanExtra(getString(R.string.from_there_only),
false))
            spc.setPasscodeSize(spc.getOrgPasscodeSize());
        passcodeSize = spc.getPasscodeSize();

        setContentView(R.layout.activity_passcode);
//        ActionBar actionBar = getSupportActionBar();

        initializeAnimation();
        hasDestination =
intent.hasExtra(getString(R.string.destination));
        /*if (actionBar != null) {
            hideBackActionBar(actionBar);
        }*/
        mDotsLayout = findViewById(R.id.dotsLayout);
        GridView gridView = findViewById(R.id.passGridView);
        mTitle = findViewById(R.id.passcode_title);
        mMessage = findViewById(R.id.message);
        setting = findViewById(R.id.setting);
        skip = findViewById(R.id.skip);
        fingerprintImage = findViewById(R.id.fingerprintImage);
        fingerprintLabel = findViewById(R.id.fingerprintLabel);
//        mView = findViewById(R.id.margin_view);
        gridView.setAnimation(upTODown);
        gridViewAdapter = new PassGridViewAdapter(this, mPattern);
        gridView.setAdapter(gridViewAdapter);
        addDots();
        setFingerprint();
```

```
        if (intent.hasExtra(getString(R.string.entry_pass))) {
            entry_type =
intent.getIntExtra(getString(R.string.entry_pass), -1);
            switch (entry_type) {
                case R.string.setup_passcode:
                    mTitle.setText(R.string.create_passcode);

mMessage.setText(R.string.create_passcode_message);
                    setSetting(true);
                    isFinishAffinity =
intent.getBooleanExtra("isFinishAffinity", true);
                    break;
                case R.string.pass_check_point:
                    setSetting(false);
                    setFingerprint();
                    mTitle.setText(R.string.enter_passcode);

mMessage.setText(R.string.pass_check_point_message);
                    break;
                case R.string.reset_pass:

findViewById(R.id.skip).setVisibility(View.GONE);
                    if
(intent.getBooleanExtra(getString(R.string.from_there_only),
false))
                        changeActivityToCreatePasscode();
                    else {
                        setSetting(false);
                        mTitle.setText(R.string.confirm_passcode);

mMessage.setText(R.string.pass_check_point_message);
                    }
                    break;
            }
        }


        gridView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View
view, int position, long id) {
                if (position == mPattern[1]) {
                    goToBack();
                } else if (position == mPattern[0]) {
                    if (cnt > 0) {
                        cnt--;
                        pin = pin.substring(0, pin.length() - 1);

translateAnimation.setAnimationListener(new
Animation.AnimationListener() {
```

```
                            @Override
                            public void onAnimationStart(Animation
animation) {

                            }

                            @Override
                            public void onAnimationEnd(Animation
animation) {
                                addDotsIndicator(cnt);
                            }

                            @Override
                            public void
onAnimationRepeat(Animation animation) {

                            }
                        });

mDots[cnt].startAnimation(translateAnimation);
                    }
                } else if (cnt < passcodeSize) {
                    TextView t = view.findViewById(R.id.number);

                    try {
                        final int n =
Integer.parseInt(t.getText().toString());

translateAnimation.setAnimationListener(new
Animation.AnimationListener() {
                            @Override
                            public void onAnimationStart(Animation
animation) {

                            }

                            @Override
                            public void onAnimationEnd(Animation
animation) {
                                addDotsIndicator(cnt);
                                pin = pin + n;
                                cnt++;
                                if (cnt == passcodeSize)
changeLevel();
                            }

                            @Override
                            public void
onAnimationRepeat(Animation animation) {

                            }
                        });
```

```
mDots[cnt].startAnimation(translateAnimation);
                    } catch (NullPointerException e) {

messageErrorPIN(R.string.exception_passcode);
                    }
                }
            }
        });

        setting.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showEditAlertBox();
            }
        });

        skip.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                skipView();
            }
        });
    }

    private void setFingerprint() {
        fingerprintImage.setVisibility(View.GONE);
        fingerprintLabel.setVisibility(View.GONE);
    }

    private void changeLevel() {
        switch (stage) {
            case 0:
                if (entry_type == R.string.setup_passcode)
                    changeActivityToReenterPasscode();
                else {
                    if (spc.matchPasscode(pin)) {
                        if (entry_type == R.string.reset_pass)
                            changeActivityToCreatePasscode();
                        else
                            goTOHome();
                    } else {
                        int wal = spc.getWrongAttemptsLeft();
                        if (wal <= 0) {
                            Const.logOut(getApplicationContext());
                            finishAffinity();
                            startActivity(new
Intent(getApplicationContext(), LoginActivity.class));
                            finishAffinity();
                        } else {
                            regenerateKey();
```

```
gridViewAdapter.notifyDataSetChanged();
                            Snackbar snackbar =
Snackbar.make(getWindow().getDecorView(),
getString(R.string.wrong_passcode, wal), Snackbar.LENGTH_LONG);
                            View sbView = snackbar.getView();

sbView.setBackgroundColor(Color.TRANSPARENT);
                            TextView textView =
sbView.findViewById(android.support.design.R.id.snackbar_text);
                            textView.setTextColor(Color.RED);
                            snackbar.show();
                            showShakeEffect();
                            cnt = 0;
                            addDotsClearAll();
                            pin = "";
                        }
                    }
                }
                break;
            case 1:
                if (pre_pin.equals(pin)) {
                    spc.setPasscode(pin);
                    spc.setOrgPasscodeSize(pin.length());
                    showSuccessfulMessage();
                    goTOHome();
                } else {
                    messageErrorPIN(R.string.passcode_not_match);
                }
                break;
            default:
                messageErrorPIN(R.string.exception_passcode);
                finish();
                break;
        }
    }

    private void initializeAnimation() {
        int animationDuration = 1000;
        upTODown = AnimationUtils.loadAnimation(this,
R.anim.uptodown);
        upTODown.setDuration(animationDuration);
        translateAnimation = new TranslateAnimation(0.0f, 0.0f,
0.0f, -100.0f);
        translateAnimation.setDuration(100);
    }

    private void addDots() {
        LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);
        int textSizeUnit = TypedValue.COMPLEX_UNIT_SP;
```

```
        float textSize =
getResources().getInteger(R.integer.passcode_dots_size);
        int textColor =
getResources().getColor(R.color.colorLightGray);
        mDots = new TextView[passcodeSize];
        for (int i = 0; i < mDots.length; i++) {
            mDots[i] = new TextView(this);
            mDots[i].setText("\u25E6");
            mDots[i].setLayoutParams(params);
            mDots[i].setTextSize(textSizeUnit, textSize);
            mDots[i].setTextColor(textColor);
            mDotsLayout.addView(mDots[i]);
        }
    }

    private void addDotsIndicator(int pos) {
        upTODown.setDuration(100);
        if (mDots.length > 0) {
            if (pos > previous_pos) {
                mDots[pos].setText("\u2022");

mDots[pos].setTextColor(getResources().getColor(R.color.colorAccen
t));
                mDots[pos].startAnimation(upTODown);
                previous_pos = pos;
            } else {
                mDots[pos].setText("\u25E6");

mDots[pos].setTextColor(getResources().getColor(R.color.colorLight
Gray));
                previous_pos = pos - 1;
            }
        }
    }

    private void addDotsClearAll() {
        for (TextView mDot : mDots) {
            mDot.setText("\u25E6");

mDot.setTextColor(getResources().getColor(R.color.colorLightGray))
;
        }
        previous_pos = -1;
    }/*

    private void hideBackActionBar(ActionBar actionBar) {
        getSupportActionBar().hide();
        actionBar.setHomeAsUpIndicator(null);
        actionBar.setHomeButtonEnabled(false); // disable the
button
        actionBar.setDisplayHomeAsUpEnabled(false); // remove the
left caret
```

```
        actionBar.setDisplayShowHomeEnabled(false);
    }*/

    public void goTOHome() {
        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            public void run() {
                if (!hasDestination) {
                    startActivity(new
Intent(PasscodeActivityLM.this, HomeActivity.class));
                } else {
                    setResult(Const.ON_RESTART_RESULT_CODE, new
Intent().putExtra(getString(R.string.security_check), true));
                }
                overridePendingTransition(R.anim.gofrom_leftright,
R.anim.goto_leftright);
                try {
                    if (isFinishAffinity)
                        finishAffinity();
                    else finish();
                } catch (Exception e) {
                    finish();
                }
            }
        }, 500);
    }

    private void changeActivityToReenterPasscode() {
        setSetting(false);
        stage++;
        regenerateKey();
        gridViewAdapter.notifyDataSetChanged();
        mTitle.setText(R.string.reenter_passcode);
        mMessage.setText(R.string.reenter_passcode_message);
        mPattern[2] = R.drawable.ic_navigate_previous;
        gridViewAdapter.notifyDataSetChanged();
        cnt = 0;
        addDotsClearAll();
        pre_pin = pin;
        pin = "";
    }

    private void changeActivityToCreatePasscode() {
        setSetting(true);
        stage = 0;
        regenerateKey();
        mPattern[2] = R.drawable.ic_exit;
        gridViewAdapter.notifyDataSetChanged();
        entry_type = R.string.setup_passcode;
        mTitle.setText(R.string.create_passcode);
        mMessage.setText(R.string.create_passcode_message);
        cnt = 0;
```

```
            addDotsClearAll();
            pre_pin = "";
            pin = "";
        }

        private void goToBack() {
            if (stage == 0) {
                finish();
                spc.setPasscodeSize(spc.getOrgPasscodeSize());
            } else {
                setSetting(true);
                stage--;
                regenerateKey();
                if (stage == 0) mPattern[2] = R.drawable.ic_exit;
                gridViewAdapter.notifyDataSetChanged();
                mTitle.setText(R.string.create_passcode);
                mMessage.setText(R.string.create_passcode_message);
                cnt = 0;
                addDotsClearAll();
                pre_pin = "";
                pin = "";
            }
        }

        private void setSetting(boolean status) {
            if (status) {
                setting.setVisibility(View.VISIBLE);
                skip.setVisibility(View.VISIBLE);
//                mView.setVisibility(View.GONE);
            } else {
                setting.setVisibility(View.GONE);
                skip.setVisibility(View.GONE);
//                mView.setVisibility(View.VISIBLE);
            }
        }

        private void showSuccessfulMessage() {
            Snackbar snackbar =
Snackbar.make(getWindow().getDecorView(),
R.string.passcode_successfully_set, Snackbar.LENGTH_LONG);
            View sbView = snackbar.getView();
            sbView.setBackgroundColor(Color.TRANSPARENT);
            TextView textView =
sbView.findViewById(android.support.design.R.id.snackbar_text);
            textView.setTextColor(Color.GREEN);
            snackbar.show();
            snackbar.setDuration(1000);
        }

        private void messageErrorPIN(int message) {
            regenerateKey();
            gridViewAdapter.notifyDataSetChanged();
```

```
        Snackbar snackbar =
Snackbar.make(getWindow().getDecorView(), message,
Snackbar.LENGTH_LONG);
        View sbView = snackbar.getView();
        sbView.setBackgroundColor(Color.TRANSPARENT);
        TextView textView =
sbView.findViewById(android.support.design.R.id.snackbar_text);
        textView.setTextColor(Color.RED);
        snackbar.show();
        showShakeEffect();
        cnt = 0;
        addDotsClearAll();
        pin = "";
    }

    private void showShakeEffect() {
        Animation shake = AnimationUtils.loadAnimation(this,
R.anim.shake);
        findViewById(R.id.dotsLayout).startAnimation(shake);
        findViewById(R.id.passGridView).startAnimation(shake);

        Vibrator vibrator = (Vibrator)
getSystemService(VIBRATOR_SERVICE);
        long[] pattern = {200, 1500, 1500};
        Objects.requireNonNull(vibrator).vibrate(pattern, -1);
    }

    @Override
    public void onBackPressed() {
        if (!hasDestination) {
            super.onBackPressed();
            spc.setPasscodeSize(spc.getOrgPasscodeSize());
        }
    }


    private void showEditAlertBox() {
        Log.e(TAG, "spc.getPasscodeSize() " +
spc.getPasscodeSize());
        final AlertDialog.Builder alertDialog = new
AlertDialog.Builder(this);
        alertDialog.setTitle(R.string.change_passcode_size);
        alertDialog.setIcon(R.drawable.ic_settings);
        final String[] sizePass = new String[]{"3", "4", "5",
"6"};

        alertDialog.setSingleChoiceItems(sizePass,
spc.getPasscodeSize() - 3, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface,
int i) {
                tempSize = sizePass[i];
```

```
            }
        });


        alertDialog.setNegativeButton(R.string.no, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface,
int i) {
            }
        });

        alertDialog.setPositiveButton(R.string.yes, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface,
int i) {
                spc.setPasscodeSize(Integer.parseInt(tempSize));
                isSizeChangeFromThere = true;
                Log.e(TAG, "spc.getPasscodeSize() after " +
spc.getPasscodeSize());
                Intent intent = getIntent();

intent.putExtra(getString(R.string.from_there_only), true);
                startActivity(intent);
                finish();
            }
        });
        alertDialog.show();
    }

    public void skipView() {
        spc.setPasscodeStatus(false);
        spc.setPasscodeSize(spc.getOrgPasscodeSize());
        startActivity(new Intent(PasscodeActivityLM.this,
HomeActivity.class));
        overridePendingTransition(R.anim.gofrom_leftright,
R.anim.goto_leftright);
        finishAffinity();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (!isSizeChangeFromThere)
            spc.setPasscodeSize(spc.getOrgPasscodeSize());
    }
}
```

```
class InfoPagerAdapter extends PagerAdapter {

    private PrimePreference prime;
    private final Context context;
    private int childCount = 1;
    private int[] sliderTitleImage = {
            R.drawable.ic_language,
            R.drawable.ic_user_agreement,
            R.drawable.ic_permission,
            R.drawable.ic_create_acc
    };
    private int[] sliderTitle = {
            R.string.info_slider_one_title,
            R.string.info_slider_two_title,
            R.string.info_slider_three_title,
            R.string.info_slider_four_title
    };
    private int[] sliderDescription = {
            R.string.info_slider_one_desc,
            R.string.info_slider_two_desc,
            R.string.info_slider_three_desc,
            R.string.info_slider_four_desc
    };
    private int[] sliderButtonText = {
            R.string.info_slider_one_button_text,
            R.string.info_slider_two_button_text,
            R.string.info_slider_three_button_text,
            R.string.info_slider_four_button_text
    };
    private int[] sliderMessage = {
            R.string.info_slider_two_message,
            R.string.info_slider_three_message,
            R.string.info_slider_four_message
    };

    InfoPagerAdapter(Context context) {
        this.context = context;
    }

    public void setChildCount(int childCount) {
        this.childCount = childCount;
    }

    @Override
    public int getCount() {
        return childCount;
    }

    @Override
```

```
    public boolean isViewFromObject(@NonNull View view, @NonNull
Object o) {
        return view == o;
    }

    @NonNull
    @Override
    public Object instantiateItem(@NonNull ViewGroup container,
final int position) {
        LayoutInflater layoutInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View v =
Objects.requireNonNull(layoutInflater).inflate(R.layout.view_info_
slider, container, false);

        ImageView sliderTitleImage =
v.findViewById(R.id.sliderTitleImage);
        TextView sliderTitle = v.findViewById(R.id.sliderTitle);
        final TextView sliderDescription =
v.findViewById(R.id.sliderDescription);
        final Button sliderButton =
v.findViewById(R.id.sliderButton);
        final TextView sliderMessage =
v.findViewById(R.id.sliderMessage);


sliderTitleImage.setImageResource(this.sliderTitleImage[position])
;
        sliderTitle.setText(this.sliderTitle[position]);

sliderDescription.setText(this.sliderDescription[position]);
        if (position == 0) {
            prime = new
PrimePreference(context.getApplicationContext());
            sliderButton.setText(String.format(" %s",
prime.getLanguage()));
            sliderButton.setTextSize(TypedValue.COMPLEX_UNIT_SP,
15);

sliderButton.setCompoundDrawablesWithIntrinsicBounds(prime.getLang
uageImgId(), 0, 0, 0);
            sliderButton.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    showLangAlertSpin(sliderButton);
                }
            });
        }
        if (position < 1) {
            sliderMessage.setVisibility(View.GONE);
        } else {
```

```
sliderButton.setText(getString(this.sliderButtonText[position]));
            sliderButton.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    getActivity().sliderOnClickListener(position,
position == 2 ? sliderButton : null, position == 2 ? sliderMessage
: null);
                }
            });
            if (position == 1) {

sliderMessage.setText(getString(this.sliderMessage[position -
1]));

sliderMessage.setTextColor(getActivity().getResources().getColor(R
.color.info_slider_mess_permission_denied_color));
            } else if (position == 2 &&
getActivity().isAllGranted()) {
                sliderButton.setClickable(false);
                sliderButton.setAlpha(0.5f);

sliderButton.setTextColor(getActivity().getResources().getColor(R.
color.info_button_disable_txt_color));

sliderMessage.setTextColor(getActivity().getResources().getColor(R
.color.info_slider_mess_permission_granted_color));

sliderMessage.setText(R.string.all_permission_granted);
            } else {

sliderMessage.setText(getString(this.sliderMessage[position -
1]));
            }
        }
        container.addView(v);
        return v;
    }

    @Override
    public void destroyItem(@NonNull ViewGroup container, int
position, @NonNull Object object) {
        container.removeView((NestedScrollView) object);
    }

    private InformationActivity getActivity() {
        return ((InformationActivity) context);
    }

    private String getString(int id) {
        return context.getResources().getString(id);
```

```
    }


    private List<LangData> data;

    private void showLangAlertSpin(final Button sliderButton) {
        data = LangData.getDefaultList();
        AlertDialog.Builder builder = new
AlertDialog.Builder(getActivity());
        builder.setTitle(R.string.choose_language);
        final CustomLangAdapter adapter = new
CustomLangAdapter(getActivity(), R.layout.view_spinner_item,
data);
        builder.setSingleChoiceItems(adapter, -1, new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which)
{
                LangData selected = data.get(which);
                String name = selected.getName();
                String code = selected.getCode();
                int id = selected.getImageId();
                sliderButton.setText(String.format(" %s", name));

sliderButton.setTextSize(TypedValue.COMPLEX_UNIT_SP, 15);

sliderButton.setCompoundDrawablesWithIntrinsicBounds(id, 0, 0, 0);

                if (Const.updateResourcesLegacy(context, code))
                    prime.setLanguage(name, code, id);
                else Const.showCustomMess(context,
R.string.unable_to_change_language, false);

                ((InformationActivity) context).recreate();
                dialog.dismiss();
            }
        });
        builder.show();
    }
}
```