**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

|  |  |
| --- | --- |
| КР. 09.02.03.18.16-2.19.ПЗ |  |

Приложение «Шифровка данных»

|  |  |  |
| --- | --- | --- |
| Председатель ВЦК: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (М.А. Кудрявцева) |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.С. Некипелова) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Д.М.Овинов) |

Иркутск 2018

**Содержание**

[**Введение** 3](#_Toc531663876)

[**1** **Общая часть** 4](#_Toc531663877)

[**1.1** **Обзор программного обеспечения для разработки** 4](#_Toc531663878)

[**1.2** **Обоснование необходимости использования выбранных методов** 5](#_Toc531663879)

[**2** **Специальная часть** 6](#_Toc531663880)

[**2.1** **Постановка задач** 6](#_Toc531663881)

[**2.2** **Описание теоретической части** 6](#_Toc531663882)

[**2.3** **Алгоритм работы программы** 10](#_Toc531663883)

[**3** **Технологическая часть** 11](#_Toc531663884)

[**3.1** **Руководство пользователя** 11](#_Toc531663885)

[**3.2** **Системно – технические требования** 12](#_Toc531663886)

[**Заключение** 13](#_Toc531663887)

[**Список используемых источников** 14](#_Toc531663888)

[**Приложение** А 15](#_Toc531663889)

# **Введение**

В истории было много криптографических приложений, начиная от простых шифров, используемых Юлием Цезарем для отправления военных приказов своим генералам, до более сложных средневековых шифров, и знаменитых кодов Энигма во Второй Мировой войне.

Развитие компьютеров в 20-м веке позволило создать более сложные средства шифрования.

Цель курсовой работы заключается в разработке мобильного приложения, для шифрования данных. Задачи, направленные на достижение цели:

1. Изучение теории криптографии;
2. Выбор используемых шифров;
3. Разработка алгоритмов выбранных шифров;
4. Разработка дизайна (интерфейса) приложения;
5. Разработка основной логики приложения;
6. Отладка приложения;

# **Общая часть**

# **Обзор программного обеспечения для разработки**

Android Studio — интегрированная среда разработки производства Google, с помощью которой разработчикам становятся доступны инструменты для создания приложений на платформе Android OS. Android Studio можно установить на Windows, Mac и Linux. Создана на базе IntelliJ IDEA.

Среда Android Studio предназначена как для небольших команд [разработчиков мобильных приложений](http://wnfx.ru/) (даже в количестве одного человека), или же крупных международных организаций с GIT или другими подобными системами управления версиями. Опытные разработчики смогут выбрать инструменты, которые больше подходят для масштабных проектов. [Решения для Android](http://wnfx.ru/android-razrabotka-mobilnyih-prilozheniy/) разрабатываются в Android Studio с использованием Java или C++. В основе рабочего процесса Android Studio заложен концепт непрерывной интеграции, позволяющий сразу же обнаруживать имеющиеся проблемы. Продолжительная проверка кода обеспечивает возможность эффективной обратной связи с разработчиками. Такая опция позволяет быстрее опубликовать версию мобильного приложения в Google Play App Store. Для этого присутствует также поддержка инструментов LINT, Pro-Guard и App Signing.

С помощью средств оценки производительности определяется состояние файла с пакетом прикладных программ. Визуализация графики дает возможность узнать, соответствует ли приложение ориентиру Google в 16 миллисекунд. С помощью инструмента для визуализации памяти разработчик узнает, когда его приложение будет использовать слишком много оперативной памяти и когда произойдет «сборка мусора». Инструменты для анализа батареи показывают, какая нагрузка приходится на устройство.

Разработчики приложений для Android предпочитают именно эту программу. Согласно отзывам, которые оставляют пользователи IDE, софт дает больше возможностей, предоставляет более гибкую среду разработки, нежели конкуренты. О чем только говорит наличие такого многофункционального эмулятора и еще нескольких инструментов, которые включены в пакет Android Studio! Видно, что с каждым обновлением IDE становится только лучше, в него интегрируют новый функционал, не забывая про старый.

# **Обоснование необходимости использования выбранных методов**

Разработка приложения с помощью среды разработки Android studio и языка программирования Java обоснуется следующими причинами:

* Н­­­­е малые теоретические и практические знания языка Java;
* Имеющийся опыт работы в Android studio;
* Большое количество функциональных интегрированных в среду инструментов;

# **Специальная часть**

# **Постановка задач**

Приложение должно шифровать введенные данные по одному из выбранных шифров. После ввода информации, указания пароля и по желанию выбора дополнительных параметров, приложение должно будет зашифровать её.

# **Описание теоретической части**

Блочный шифр (block cipher) — это функция шифрования, которая применяется к блокам текста фиксированной длины. Текущее поколение блочных шифров работает с блоками текста длиной 128 бит (16 байт). Такой шифр принимает на вход 128-битовый открытый текст и выдает 128-битовый шифрованный текст. Блочный шифр является обратимым: существует функция дешифрования, которая принимает на вход 128-битовый шифрованный текст и выдает исходный 128-битовый открытый текст. Открытый и шифрованный текст всегда имеет один и тот же размер, который называется размером блока (block size).

Чтобы зашифровать сообщение, нужен секретный ключ. Невозможно скрыть сообщение, не сохраняя что-нибудь в секрете. Подобно открытому и шифрованному тексту, ключ шифрования также представляет собой строку битов. Наиболее распространены ключи размером 128 и 256 бит. Шифрование открытого текста p при помощи ключа K принято обозначать как E(K, p) или EK(p), а расшифровку шифрованного текста c при помощи ключа K — D(K, c) или DK(c).

Иногда блочный шифр было бы удобно представить в виде большой таблицы, построенной на основе конкретного ключа. Для каждого фиксированного ключа можно построить таблицу соответствий, которая бы отображала все варианты открытого текста в соответствующий шифрованный текст. Разумеется, это была бы очень большая таблица. Для блочного шифра с размером блока 32 бит понадобилась бы таблица размером 16 Гбайт, для шифра с размером блока 64 бит — 150 млн. Тбайт, а для шифра с размером блока 128 бит — 5·1039 байт. В таблице содержится ровно по одному экземпляру всех возможных вариантов шифрованного текста. Набор элементов шифрованного текста совпадает с набором элементов открытого текста, порядок расположения которых изменен. В математике это называется перестановкой (permutation). Блочный шифр с размером блока k бит задает перестановку k-битовых элементов для каждого из заданных значений ключа.

Блочные шифры применяются только к блокам текста фиксированного размера. Чтобы зашифровать какой-нибудь текст, размер которого не совпадает с размером блока, необходимо воспользоваться одним из режимов работы блочных шифров (block cipher modes).

Наиболее популярным режимом работы блочных шифров является сцепление шифрованных блоков (cipher block chaining — CBC). Каждый блок открытого текста складывается с помощью операции XOR с предыдущим блоком шифрованного текста. Стандартная формулировка режима CBC выглядит следующим образом:

Ci = E(K, Pi ⊕ Ci−1) для i = 1, . . . , k.

(1)

В открытый текст вносится “элемент случайности” путем его прибавления к предыдущему блоку шифрованного текста. В режиме CBC одинаковые блоки открытого текста преобразуются в разные блоки шифрованного текста, что значительно сокращает объем информации об открытом тексте, доступной злоумышленнику.

Следует заметить, что в качестве C0 используется значение, называемое вектором инициализации (initialization vector — IV). Использовать фиксированный вектор инициализации не следует. В противном случае в первом блоке каждого сообщения могут возникнуть проблемы. Если два разных сообщения начинаются с одного и того же блока открытого текста, соответствующие шифрованные сообщения будут начинаться с одинаковых блоков шифрованного текста. На практике сообщения часто начинаются с одинаковых или похожих блоков шифрованного текста, поэтому не стоит помогать злоумышленнику.

AES (Advanced Encryption Standard — улучшенный стандарт шифрования)

Алгоритм AES не относится к шифрам Файстеля. Ниже показан один раунд алгоритма AES (Рисунок 1).

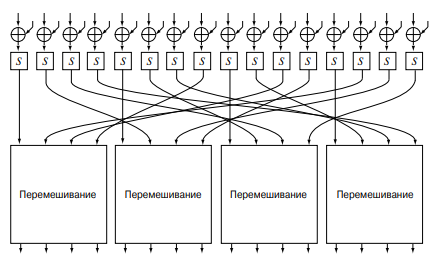
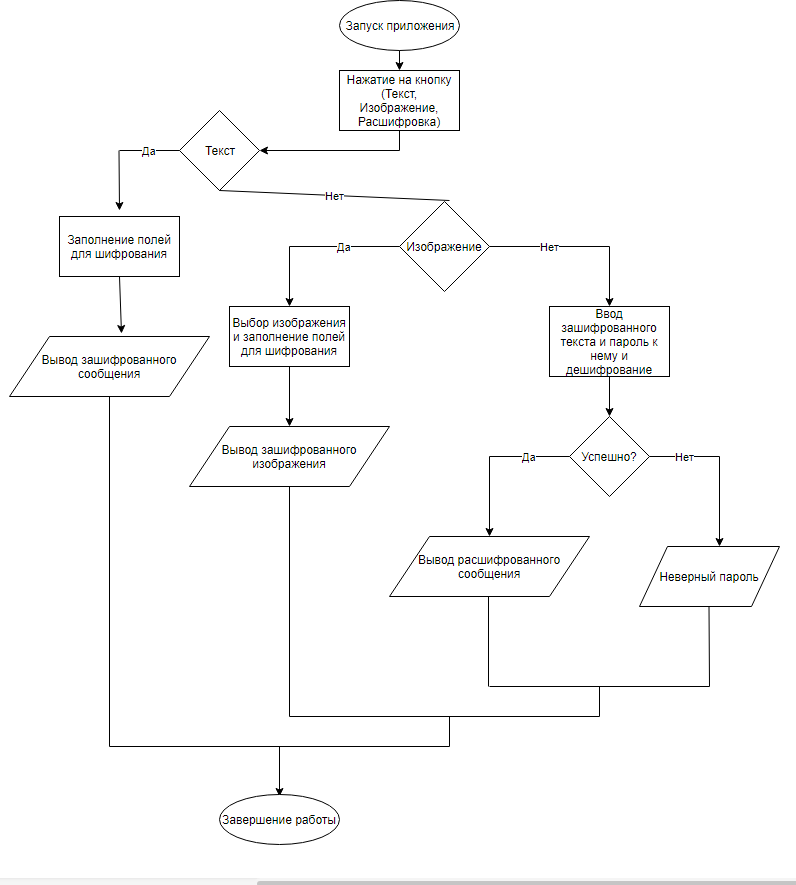


Рисунок 1 – Структура одного раунда AES

Последующие раунды имеют аналогичную структуру. На вход алгоритма подается блок открытого текста длиной 16 байт. Вначале открытый текст складывается с помощью операции XOR с 16-байтовым (128-битовым) подключом. На рисунке этот процесс обозначен операторами ⊕ (каждый байт подключа складывается с соответствующим байтом открытого текста). Затем каждый из 16 байт полученного результата подается на вход таблицы S матриц, которая отображает 8-битовые входные значения в 8-битовые выходные значения. Все S-матрицы одинаковы. Полученные байты переставляются в некотором заданном порядке. И наконец, каждая группа из 4 байт подвергается перемешиванию, которое осуществляется с помощью линейной функции перемешивания. Термин “линейная” означает лишь то, что каждый бит выходных данных функции перемешивания получен в результате применения операции XOR к нескольким входным битам. Применение функции перемешивания завершает раунд. Полный процесс шифрования состоит из 10-14 раундов, в зависимости от размера ключа. Подключи AES, генерируются на основе некоторого ключа шифрования. Для расшифровки текста необходимо использовать обратные S матрицы, да и функция, обратная перемешиванию, существенно отличается от самой функции перемешивания.

В алгоритме AES можно выделить несколько функциональных блоков, каждый из которых имеет свое назначение. Операции XOR складывают значение ключа с данными, S-матрицы обеспечивают нелинейность, а функции перестановки и перемешивания гарантируют наличие диффузии. Шифр AES имеет очень четкую структуру, каждая часть которой выполняет строго определенную задачу.

# **Алгоритм работы программы**



# **Технологическая часть**

# **Руководство пользователя**

# **Системно – технические требования**

Минимальные системные требования устройства:

1. Архитектура на базе операционной системы Android 4.4 “KitKat” и выше;
2. Оперативная память от 512 Мб;
3. 12 Мб свободного пространства;
4. Наличие камеры (Не обязательно);
5. Наличие GPS (Не обязательно);

# 

# **Заключение**

Цель курсовой работы, разработка мобильного приложения для шифрования данных, была достигнута.

В ходе работы были встречены трудности по следующим пунктам:

# **Список используемых источников**

Эккель.Б. Философия Java. 4-е полное изд. – СПБ.: Питер, 2017. –

1168 с. : ил. – (Серия «Классика computer scince»).

Фергюсон, Н. Практическая криптография. / Н. Фергюсон, Б. Шнайер. –

# **Приложение** А

**Листинг приложения**

public class MainActivity extends AppCompatActivity implements LinearViewHolder.ActionListener {  
  
 private RecyclerView itemresviews;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 Toolbar toolbar = findViewById(R.id.toolbar);  
 setSupportActionBar(toolbar);  
 SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);

String lang = preferences.getString("lang", "default");  
 if (lang.equals("default")) {  
 lang =getResources().getConfiguration().locale.getCountry();}  
 Locale locale = new Locale(lang);  
 Resources res = getResources();  
 DisplayMetrics dm = res.getDisplayMetrics();  
 Configuration conf = res.getConfiguration();  
 conf.locale = locale;  
 res.updateConfiguration(conf, dm);  
 List<EncryptedClass> cryptoList;  
 DataBase database = App.getInstance().getDatabase();  
 EncryptedDAO encryptedDAO = database.encryptedDAO();  
 cryptoList = encryptedDAO.getAll();  
 itemresviews = findViewById(R.id.itemresview);  
 LinearLayoutManager layoutManager = new LinearLayoutManager(this);  
 itemresviews.setLayoutManager(layoutManager);  
 itemresviews.setItemAnimator(new DefaultItemAnimator());  
 ResAdapter linearAdapter = new ResAdapter();  
 if (cryptoList.size() != 0)  
 linearAdapter.addAll(cryptoList);  
 linearAdapter.addHeader(inflateHeader());  
 linearAdapter.setActionListener(this);  
 itemresviews.setAdapter(linearAdapter);  
 }  
  
 private View inflateHeader() {  
 CardView Header = (CardView) getLayoutInflater().inflate(  
 R.layout.header, itemresviews, false);  
 Header.setUseCompatPadding(true);  
 return Header;  
 }  
  
 public void onCipherClick(View view) {  
 Intent intent = null;  
 switch (view.getId()) {  
 case (R.id.btn\_text):  
 intent = new Intent(MainActivity.this, MessageActivity.class);  
 break;  
 case (R.id.btn\_image):  
 intent = new Intent(MainActivity.this, ImageActivity.class);  
 break;  
 case (R.id.btn\_decrypt):  
 intent = new Intent(MainActivity.this, TestedActivity.class);  
 break;  
 }  
 startActivityForResult(intent, 1);  
 overridePendingTransition(R.anim.slide\_right, R.anim.alpha);  
 }  
  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.menu\_main, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 int id = item.getItemId();  
 if (id == R.id.action\_settings) {  
 Intent intent = new Intent(this, SettingsActivity.class);  
 startActivityForResult(intent, 1);  
 overridePendingTransition(R.anim.slide\_up, R.anim.alpha);  
 return true;  
 }  
 return super.onOptionsItemSelected(item);  
 }  
  
 @Override  
 public void OnItemClick(long id) {  
 Intent intent = new Intent(MainActivity.this, ViewItemActivity.class);  
 intent.putExtra("Position",id);  
 startActivityForResult(intent, 1);  
 overridePendingTransition(R.anim.slide\_right, R.anim.alpha);  
 }  
}

public class MessageActivity extends AppCompatActivity implements FragmentManager.OnBackStackChangedListener {  
  
 private EditText etMessage;  
 private EditText etPassword;  
 private TextView tvEMessage;  
 Button btnEncrypt;  
 Button cib;  
 SwipeItemsAdapter methodadapter;  
 List<PageItem> MethodEncryptedItems;  
 SecretKey secretKey;  
 ViewPager ChipherPager;  
 DialogPages dialogPages;  
 Encryptor encryptor;  
 private boolean mShowingBack = false;  
 private String Ct;  
 private String password;  
   
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_message);  
 Toolbar toolbar = findViewById(R.id.toolbar);  
 setSupportActionBar(toolbar);  
 Objects.requireNonNull(getSupportActionBar()).setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setDisplayShowHomeEnabled(true);  
 etMessage = findViewById(R.id.Message);  
 etPassword = findViewById(R.id.etPassword);  
 tvEMessage = findViewById(R.id.tv\_message);  
 btnEncrypt = findViewById(R.id.btn\_encrypted);  
 cib = findViewById(R.id.btn\_copy);  
 encryptor = new Encryptor();  
 MethodEncryptedItems = new ArrayList<>();  
 fillList();  
 ChipherPager = findViewById(R.id.pv);  
 methodadapter = new SwipeItemsAdapter(this);  
 methodadapter.setItems(MethodEncryptedItems);  
 ChipherPager.setAdapter(methodadapter);  
 btnEncrypt.setOnClickListener(onEncrypt);  
 if (savedInstanceState == null) {  
 getFragmentManager()  
 .beginTransaction()  
 .add(R.id.frame\_addParam, new CardFrontFragment())  
 .commit();  
 } else {  
 mShowingBack = (getFragmentManager().getBackStackEntryCount() > 0);  
 }  
 getFragmentManager().addOnBackStackChangedListener(this);  
 }  
   
 private void fillList() {  
 for (int i = 0; i < getResources().getStringArray(R.array.method\_encrypted).length; i++)  
 MethodEncryptedItems.add(new PageItem("Method", getResources().getStringArray(R.array.method\_encrypted)[i]));  
 }  
  
 private View.OnClickListener onEncrypt = new View.OnClickListener() {  
 public void onClick(View v) {  
 password = etPassword.getText().toString();  
 Log.i("TAG", "onDecryptClick: "+ password + getExtraOptionPassword());  
  
 if (!password.isEmpty()) {  
 secretKey = encryptor.deriveKey(password+getExtraOptionPassword());  
 byte[] iv = encryptor.generateIV(16);  
 String rowMessage = etMessage.getText().toString();  
 Ct = encryptor.encrypt(rowMessage, secretKey, iv);  
 tvEMessage.setText(Ct);  
 }else Snackbar.make(v,getResources().getString(R.string.errPass),Snackbar.LENGTH\_SHORT).show();  
 }  
 };  
  
 @SuppressLint({"MissingPermission", "HardwareIds"})  
 private String getExtraOptionPassword(){  
 String result = "";  
 @SuppressLint("SimpleDateFormat") SimpleDateFormat dateFormat = new SimpleDateFormat("H:mm");  
 TelephonyManager telephonyManager = (TelephonyManager)getSystemService(Context.TELEPHONY\_SERVICE);  
 Map<String,Boolean> map = isExOption();  
 result += map.get(getResources().getString(R.string.imei)) ? Objects.requireNonNull(telephonyManager).getDeviceId() : "";  
// result += map.get(getResources().getString(R.string.gps)) ? MyLocation.imHere : "";  
 result += map.get(getResources().getString(R.string.time)) ? dateFormat.format(new Date()) : "";  
  
 return result;  
 }  
  
 public void onCopyInBuffer(View v) {  
 ClipboardManager clipboard = (ClipboardManager) getSystemService(Context.CLIPBOARD\_SERVICE);  
 ClipData clip = ClipData.newPlainText("", dialogPages.getMessage());  
 Objects.requireNonNull(clipboard).setPrimaryClip(clip);  
 Toast.makeText(getApplicationContext(), "Текст скопирован в буфер обмена", Toast.LENGTH\_SHORT).show();  
 }  
  
 public void onSave(View v) {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 LayoutInflater inflater = this.getLayoutInflater();  
 final View dialogView = inflater.inflate(R.layout.dialog, (ViewGroup) findViewById(R.id.dialog\_rootView), false);  
 final EditText etTitle = dialogView.findViewById(R.id.etSetTitle);  
 final CheckBox checkSPass = dialogView.findViewById(R.id.check\_save\_pass);  
 builder.setView(dialogView);  
 builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 DateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy", Locale.getDefault());  
 EncryptedClass encrypted = new EncryptedClass(App.getInstance().getDatabase().encryptedDAO().getAll().size() + 1,etTitle.getText().toString(),  
 MethodEncryptedItems.get(ChipherPager.getCurrentItem()).getValue(), 'T', Ct, dateFormat.format(new Date().getTime()), checkSPass.isChecked() ? password : "",isExOption());  
 DataBase database = App.getInstance().getDatabase();  
 EncryptedDAO encryptedDAO = database.encryptedDAO();  
 encryptedDAO.insert(encrypted);  
 Intent intent = new Intent(MessageActivity.this, MainActivity.class);  
 startActivityForResult(intent, 1);  
 overridePendingTransition(R.anim.slide\_up, R.anim.alpha);  
 }  
 });  
 builder.setNegativeButton("Cancel", null);  
 AlertDialog dialog = builder.create();  
 Objects.requireNonNull(dialog.getWindow()).getAttributes().windowAnimations = R.style.DialogTheme;  
 dialog.show();  
 }  
  
 private Map<String,Boolean> isExOption(){  
 CheckBox cb1 = findViewById(R.id.checkBox);  
 CheckBox cb2 = findViewById(R.id.checkBox2);  
 CheckBox cb3 = findViewById(R.id.checkBox3);  
 Map<String,Boolean> map = new ArrayMap<>();  
 map.put(getResources().getString(R.string.imei), cb1 != null && cb1.isChecked());  
 map.put(getResources().getString(R.string.gps),cb2 != null && cb2.isChecked());  
 map.put(getResources().getString(R.string.time),cb3 != null && cb3.isChecked());  
 return map;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 switch (item.getItemId()) {  
 case android.R.id.home:  
 this.finish();  
 overridePendingTransition(R.anim.zero, R.anim.slide\_left);  
 return true;  
 default:  
 return super.onOptionsItemSelected(item);  
 }  
 }  
  
 @Override  
 public void onBackStackChanged() {}  
  
 public void onAddParam(View view) {  
 flipCard();  
 }  
  
 public void onPageArrowChange(View view) {  
 switch (view.getId()) {  
 case R.id.page\_left:  
 if (ChipherPager.getCurrentItem() != 0)  
 ChipherPager.setCurrentItem(ChipherPager.getCurrentItem() - 1);  
 break;  
 case R.id.page\_right:  
 if (ChipherPager.getCurrentItem() != Objects.requireNonNull(ChipherPager.getAdapter()).getCount() - 1)  
 ChipherPager.setCurrentItem(ChipherPager.getCurrentItem() + 1);  
 break;  
 }  
 }  
  
 public void onShowPassword(View view) {  
 etPassword.setInputType(etPassword.getInputType() == 129 ? 1 : 129);  
 }  
  
 public static class CardFrontFragment extends Fragment {  
 public CardFrontFragment() {}  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,Bundle savedInstanceState) {  
 return inflater.inflate(R.layout.fragment\_card\_front, container, false);  
 }  
 }  
  
 public static class CardBackFragment extends Fragment {  
 public CardBackFragment() {}  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,Bundle savedInstanceState) {  
 return inflater.inflate(R.layout.fragment\_card\_back, container, false);  
 }  
 }  
  
 private void flipCard() {  
 try {  
 if (mShowingBack) {  
 getFragmentManager().popBackStack();  
 return;  
 }  
 mShowingBack = true;  
 getFragmentManager()  
 .beginTransaction()  
 .setCustomAnimations(  
 R.animator.card\_flip\_right\_in, R.animator.card\_flip\_right\_out,  
 R.animator.card\_flip\_left\_in, R.animator.card\_flip\_left\_out)  
 .replace(R.id.frame\_addParam, new CardBackFragment())  
 .addToBackStack(null)  
 .commit();  
 }catch (Exception e) {  
 Log.e("TAG", "flipCard: ", e);  
 }  
 }  
}

public class ImageActivity extends AppCompatActivity implements FragmentManager.OnBackStackChangedListener {  
  
 static final int GALLERY\_REQUEST = 1;  
 static final int REQUEST\_IMAGE\_CAPTURE = 2;  
  
 private boolean mShowingBack = false;  
  
 private String password;  
  
 private Encryptor encryptor;  
  
  
 private ImageView image;  
 private TextView tv;  
 private EditText etPassword;  
 private Bitmap picture;  
  
 private SwipeItemsAdapter methodadapter;  
 private ViewPager ChipherPager;  
  
 List<PageItem> MethodEncryptedItems;  
 private String Ct;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_image);  
 Toolbar toolbar = findViewById(R.id.toolbar);  
 setSupportActionBar(toolbar);  
 Objects.requireNonNull(getSupportActionBar()).setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setDisplayShowHomeEnabled(true);  
  
 picture = null;  
  
 encryptor = new Encryptor();  
  
 Button button = findViewById(R.id.button2);  
 Button button1 = findViewById(R.id.button3);  
 Button btnEncrypted = findViewById(R.id.btn\_encrypted\_on\_image);  
 tv = findViewById(R.id.tv\_message\_on\_image);  
 etPassword = findViewById(R.id.etPassword\_on\_image);  
 image = findViewById(R.id.imageView);  
  
 MethodEncryptedItems = new ArrayList<>();  
 fillList();  
  
 ChipherPager = findViewById(R.id.pv\_on\_image);  
 methodadapter = new SwipeItemsAdapter(this);  
 methodadapter.setItems(MethodEncryptedItems);  
 ChipherPager.setAdapter(methodadapter);  
  
 if (savedInstanceState == null) {  
 getFragmentManager()  
 .beginTransaction()  
 .add(R.id.frame\_addParam\_on\_image, new ImageActivity.CardFrontFragment())  
 .commit();  
 } else {  
 mShowingBack = (getFragmentManager().getBackStackEntryCount() > 0);  
 }  
 getFragmentManager().addOnBackStackChangedListener(this);  
  
  
  
 button.setOnClickListener(new View.OnClickListener() {  
  
 @Override  
 public void onClick(View v) {  
 Intent photoPickerIntent = new Intent(Intent.ACTION\_PICK);  
  
 photoPickerIntent.setType("image/\*");  
 startActivityForResult(photoPickerIntent, GALLERY\_REQUEST);  
 }  
 });  
  
 button1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent takePictureIntent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);  
 if (takePictureIntent.resolveActivity(getPackageManager()) != null) {  
 startActivityForResult(takePictureIntent, REQUEST\_IMAGE\_CAPTURE);  
 }  
 }  
 });  
  
 btnEncrypted.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 byte[] b = getByteArrayfromBitmap(picture);  
 tv.setText(Arrays.toString(b));  
 password = etPassword.getText().toString();  
 SecretKey secretKey;  
 if (!password.isEmpty()) {  
 secretKey = encryptor.deriveKey(password);  
 byte[] iv = encryptor.generateIV(16);  
 Ct = encryptor.encrypt(Arrays.toString(b), secretKey, iv);  
 tv.setText(Ct.length() > 256 ? Ct.substring(0,256)+ "..." : Ct);  
 }else Snackbar.make(v,getResources().getString(R.string.errPass),Snackbar.LENGTH\_SHORT).show();  
 }  
 });  
 }  
  
 private void fillList() {  
 for (int i = 0; i < getResources().getStringArray(R.array.method\_encrypted).length; i++)  
 MethodEncryptedItems.add(new PageItem("Method", getResources().getStringArray(R.array.method\_encrypted)[i]));  
 }  
  
 @Override  
 public void onBackStackChanged() {}  
  
 public byte[] getByteArrayfromBitmap(Bitmap bitmap) {  
 ByteArrayOutputStream bos = new ByteArrayOutputStream();  
 bitmap.compress(Bitmap.CompressFormat.PNG, 0, bos);  
 return bos.toByteArray();  
 }  
  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, Intent imageReturnedIntent) {  
 super.onActivityResult(requestCode, resultCode, imageReturnedIntent);  
  
 switch(requestCode) {  
 case GALLERY\_REQUEST:  
 if(resultCode == RESULT\_OK){  
 Uri selectedImage = imageReturnedIntent.getData();  
 try {  
 picture = MediaStore.Images.Media.getBitmap(getContentResolver(), selectedImage);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 image.setImageBitmap(picture);  
 }break;  
 case REQUEST\_IMAGE\_CAPTURE:  
 if (resultCode == RESULT\_OK) {  
 Bundle extras = imageReturnedIntent.getExtras();  
 picture = (Bitmap) Objects.requireNonNull(extras).get("data");  
 image.setImageBitmap(picture);  
 }  
 }  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 switch (item.getItemId()) {  
 case android.R.id.home:  
 this.finish();  
 overridePendingTransition(R.anim.zero, R.anim.slide\_left);  
 return true;  
 default:  
 return super.onOptionsItemSelected(item);  
 }  
 }  
  
 public void onAddParam(View view) {  
 flipCard();  
 }  
  
 public void onCopyInBuffer(View v) {  
 ClipboardManager clipboard = (ClipboardManager) getSystemService(Context.CLIPBOARD\_SERVICE);  
 ClipData clip = ClipData.newPlainText("", Ct);  
 Objects.requireNonNull(clipboard).setPrimaryClip(clip);  
 Toast.makeText(getApplicationContext(), "Текст скопирован в буфер обмена", Toast.LENGTH\_SHORT).show();  
 }  
  
 public void onSave(View view) {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 LayoutInflater inflater = this.getLayoutInflater();  
 final View dialogView = inflater.inflate(R.layout.dialog, (ViewGroup) findViewById(R.id.dialog\_rootView), false);  
 final EditText etTitle = dialogView.findViewById(R.id.etSetTitle);  
 builder.setView(dialogView);  
 builder.setTitle("EncryptedClass Message");  
 builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 DateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy", Locale.getDefault());  
 EncryptedClass encrypted = new EncryptedClass(App.getInstance().getDatabase().encryptedDAO().getAll().size() + 1,etTitle.getText().toString(),  
 MethodEncryptedItems.get(ChipherPager.getCurrentItem()).getValue(), 'I', Ct, dateFormat.format(new Date().getTime()), password, isExOption());  
 DataBase database = App.getInstance().getDatabase();  
 EncryptedDAO encryptedDAO = database.encryptedDAO();  
 encryptedDAO.insert(encrypted);  
 Intent intent = new Intent(ImageActivity.this, MainActivity.class);  
 startActivityForResult(intent, 1);  
 overridePendingTransition(R.anim.slide\_up, R.anim.alpha);  
 }  
 });  
 builder.setNegativeButton("Cancel", null);  
 AlertDialog dialog = builder.create();  
 Objects.requireNonNull(dialog.getWindow()).getAttributes().windowAnimations = R.style.DialogTheme;  
 dialog.show();  
 }  
  
 public void onShowPassword(View view) {  
 etPassword.setInputType(etPassword.getInputType() == 129 ? 1 : 129);  
 }  
  
 private Map<String,Boolean> isExOption(){  
 Map<String,Boolean> map = new ArrayMap<>();  
 CheckBox cb1 = findViewById(R.id.checkBox);  
 CheckBox cb2 = findViewById(R.id.checkBox2);  
 CheckBox cb3 = findViewById(R.id.checkBox3);  
 map.put(cb1.getText().toString(),cb1.isChecked());  
 map.put(cb2.getText().toString(),cb2.isChecked());  
 map.put(cb3.getText().toString(),cb3.isChecked());  
 return map;  
 }  
  
  
 public static class CardFrontFragment extends Fragment {  
 public CardFrontFragment() {}  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 return inflater.inflate(R.layout.fragment\_card\_front, container, false);  
 }  
 }  
  
  
 public static class CardBackFragment extends Fragment {  
 public CardBackFragment() {}  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 return inflater.inflate(R.layout.fragment\_card\_back, container, false);  
 }  
 }  
  
 private void flipCard() {  
 try {  
 if (mShowingBack) {  
 getFragmentManager().popBackStack();  
 return;  
 }  
 mShowingBack = true;  
 getFragmentManager()  
 .beginTransaction()  
 .setCustomAnimations(  
 R.animator.card\_flip\_right\_in, R.animator.card\_flip\_right\_out,  
 R.animator.card\_flip\_left\_in, R.animator.card\_flip\_left\_out)  
 .replace(R.id.frame\_addParam\_on\_image, new ImageActivity.CardBackFragment())  
 .addToBackStack(null)  
 .commit();  
 }catch (Exception e) {  
 Log.e("TAG", "flipCard: ", e);  
 }  
 }  
}

public class ViewItemActivity extends AppCompatActivity{  
  
 Toolbar toolbar;  
 TextView tvRMes, tvEncMes,tvDate;  
 EditText etPassword;  
 EncryptedClass CrItem;  
 Encryptor encryptor;  
 long id;  
 char type;  
  
 private BottomNavigationView.OnNavigationItemSelectedListener mOnNavigationItemSelectedListener  
 = new BottomNavigationView.OnNavigationItemSelectedListener() {  
  
 @Override  
 public boolean onNavigationItemSelected(@NonNull MenuItem item) {  
 switch (item.getItemId()) {  
 case R.id.navigation\_share:  
 Intent intent=new Intent(Intent.ACTION\_SEND);  
 intent.putExtra(Intent.EXTRA\_SUBJECT, "subject");  
 intent.putExtra(Intent.EXTRA\_TEXT, tvEncMes.getText());  
 intent.setType("message/rfc822");  
 startActivity(Intent.createChooser(intent, "Choose an Email client :"));  
 return true;  
 case R.id.navigation\_delete:  
 onRemove();  
 return true;  
 }  
 return false;  
 }  
 };  
  
 private void onRemove() {  
 DataBase database = App.getInstance().getDatabase();  
 EncryptedDAO encryptedDAO = database.encryptedDAO();  
 encryptedDAO.delete(CrItem);  
 Intent intent = new Intent(this, MainActivity.class);  
 startActivityForResult(intent, 1);  
 overridePendingTransition(R.anim.slide\_up, R.anim.alpha);  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 switch (item.getItemId()) {  
 case android.R.id.home:  
 this.finish();  
 overridePendingTransition(R.anim.zero, R.anim.slide\_left);  
 return true;  
 default:  
 return super.onOptionsItemSelected(item);  
 }  
 }  
  
 @SuppressLint("InflateParams")  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_view\_item);  
 Bundle arguments = getIntent().getExtras();  
 if (arguments != null) {  
 id = arguments.getLong("Position");  
 }  
 DataBase database = App.getInstance().getDatabase();  
 EncryptedDAO encryptedDAO = database.encryptedDAO();  
 CrItem = encryptedDAO.getById(id);  
 type = CrItem.getType();  
 ViewGroup inclusionViewGroup = findViewById(R.id.include);  
 View child = LayoutInflater.from(this).inflate(R.layout.content\_view\_item, null);  
 inclusionViewGroup.removeAllViews();  
 inclusionViewGroup.addView(child);  
 toolbar = findViewById(R.id.item\_activity\_toolbar);  
 setSupportActionBar(toolbar);  
 Objects.requireNonNull(getSupportActionBar()).setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setDisplayShowHomeEnabled(true);  
 tvRMes = findViewById(R.id.Raw\_Message);  
 tvDate = findViewById(R.id.tv\_IV\_date);  
 tvEncMes = findViewById(R.id.Encrypted\_Message);  
 TextView tvIsExParam = findViewById(R.id.tvIsExParam);  
 etPassword = findViewById(R.id.etPassword\_on\_itemView);  
 encryptor = new Encryptor();  
 tvRMes.setText(CrItem.getMessage());  
 tvDate.setText(CrItem.getDate());  
 tvIsExParam.setText(getResources().getString(R.string.is\_ex\_param,  
 CrItem.getExOption().get(getResources().getString(R.string.imei)),  
 CrItem.getExOption().get(getResources().getString(R.string.gps)),  
 CrItem.getExOption().get(getResources().getString(R.string.time))));  
 etPassword.setText(CrItem.getPassword());  
 BottomNavigationView navigation = findViewById(R.id.navigation);  
 navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener);  
 }  
  
 @Override  
 protected void onResume() {  
 if(toolbar.getTitle() != CrItem.getTitle()) toolbar.setTitle(CrItem.getTitle());  
 super.onResume();  
 }  
  
 public void onShowPassword2(View view) {  
 etPassword.setInputType(etPassword.getInputType() == 129 ? 1 : 129);  
 }  
  
 public Bitmap getBitmapfromByteArray(byte[] bitmap) {  
 return BitmapFactory.decodeByteArray(bitmap , 0, bitmap.length);  
 }  
  
 @SuppressLint({"MissingPermission", "HardwareIds"})  
 private String getExtraOptionPassword(){  
 String result = "";  
 @SuppressLint("SimpleDateFormat") SimpleDateFormat dateFormat = new SimpleDateFormat("H:mm");  
 TelephonyManager telephonyManager = (TelephonyManager)getSystemService(Context.TELEPHONY\_SERVICE);  
 Map<String,Boolean> map = isExOption();  
 result += map.get(getResources().getString(R.string.imei)) ? Objects.requireNonNull(telephonyManager).getDeviceId() : "";  
// result += map.get(getResources().getString(R.string.gps)) ? : "";  
 result += map.get(getResources().getString(R.string.time)) ? dateFormat.format(new Date()) : "";  
  
 return result;  
 }  
  
 private Map<String,Boolean> isExOption() {  
 return CrItem.getExOption();  
 }  
  
 public void onDecryptClick(View view) {  
 Log.i("TAG", "onDecryptClick: "+ CrItem.getPassword() + getExtraOptionPassword());  
 String decrypted = encryptor.decrypt(CrItem.getMessage(), etPassword.getText() + getExtraOptionPassword());  
 if (decrypted != null)  
 switch (type) {  
 case 'T' : tvEncMes.setText(decrypted); break;  
 case 'I' :  
 ImageView imageView = findViewById(R.id.imageView2);  
 byte[] bs;  
 String[] dec = decrypted.substring(1, decrypted.length() - 1).split(",");  
 bs = new byte[dec.length];  
 int i = 0;  
 for (String s: dec) {  
 bs[i] = Byte.parseByte(s.trim());  
 i++;  
 }  
 imageView.setImageBitmap(getBitmapfromByteArray(bs));  
 }  
 else Toast.makeText(this, "Неверный пароль", Toast.LENGTH\_SHORT).show();  
 }  
}

@Entity  
public class EncryptedClass implements Serializable {  
  
 @PrimaryKey  
 private long id;  
  
 private String title;  
 private String cryptoMethod;  
 private String message;  
 private String date;  
 private char type;  
 @TypeConverters({MapConverter.class})  
 private Map<String,Boolean> exOption;  
 private String password;  
  
 public EncryptedClass(long id, String title, String cryptoMethod,

char type, String message, String date,

String password,Map<String,Boolean> exOption){  
 this.id = id;  
 this.title = title;  
 this.cryptoMethod = cryptoMethod;  
 this.type = type;  
 this.message = message;  
 this.date = date;  
 this.password = password;  
 this.exOption = exOption;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public String getCryptoMethod() {  
 return cryptoMethod;  
 }  
  
 public String getMessage() {  
 return message;  
 }  
  
 public void setMessage(String message) {  
 this.message = message;  
 }  
  
 public String getDate() {  
 return date;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public char getType() {  
 return type;  
 }  
  
 public void setType(char type) {  
 this.type = type;  
 }  
  
 public long getId() {  
 return id;  
 }  
  
 public Map<String,Boolean> getExOption() {  
 return exOption;  
 }  
}

public class Crypto {  
  
 private static final String TAG = Crypto.class.getSimpleName();  
 public static final String PBKDF2\_DERIVATION\_ALGORITHM = "PBKDF2WithHmacSHA1";  
 private static final String CIPHER\_ALGORITHM = "AES/CBC/PKCS5Padding";  
 private static String DELIMITER = "]";  
 private static final int PKCS5\_SALT\_LENGTH = 8;  
 private static SecureRandom random = new SecureRandom();  
 private Crypto() {  
 }  
  
 public static SecretKey generateKey(byte[] salt, String password) {  
 try {  
 int ITERATION\_COUNT = 1000;  
 int KEY\_LENGTH = 128;  
 KeySpec keySpec = new PBEKeySpec(password.toCharArray(), salt,  
 ITERATION\_COUNT, KEY\_LENGTH);  
 SecretKeyFactory keyFactory = SecretKeyFactory  
 .getInstance(PBKDF2\_DERIVATION\_ALGORITHM);  
 byte[] keyBytes = keyFactory.generateSecret(keySpec).getEncoded();  
 return new SecretKeySpec(keyBytes, "AES");  
 } catch (GeneralSecurityException e) {  
 throw new RuntimeException(e);  
 }  
 }  
  
 public static byte[] generateIv(int length) {  
 byte[] b = new byte[length];  
 random.nextBytes(b);  
 return b;  
 }  
  
 public static byte[] generateSalt() {  
 byte[] b = new byte[PKCS5\_SALT\_LENGTH];  
 random.nextBytes(b);  
 return b;  
 }  
  
 public static String encrypt(String plaintext, SecretKey key,byte[] iv, byte[] salt) {  
 try {  
 Cipher cipher = Cipher.getInstance(CIPHER\_ALGORITHM);  
// byte[] iv = generateIv(cipher.getBlockSize());  
 Log.d(TAG, "IV: " + toHex(iv));  
 IvParameterSpec ivParams = new IvParameterSpec(iv);  
 cipher.init(Cipher.ENCRYPT\_MODE, key, ivParams);  
 Log.d(TAG, "Cipher IV: "  
 + (cipher.getIV() == null ? null : toHex(cipher.getIV())));  
 byte[] cipherText = cipher.doFinal(plaintext.getBytes("UTF-8"));  
 if (salt != null) {  
 return String.format("%s%s%s%s%s", toBase64(salt), DELIMITER,  
 toBase64(iv), DELIMITER, toBase64(cipherText));  
 }  
 return String.format("%s%s%s", toBase64(iv), DELIMITER,  
 toBase64(cipherText));  
 } catch (GeneralSecurityException | UnsupportedEncodingException e) {  
 throw new RuntimeException(e);  
 }  
 }  
  
 public static String encrypt(byte[] plaintext, SecretKey key,byte[] iv, byte[] salt) {  
 try {  
 Cipher cipher = Cipher.getInstance(CIPHER\_ALGORITHM);  
  
// byte[] iv = generateIv(cipher.getBlockSize());  
 Log.d(TAG, "IV: " + toHex(iv));  
 IvParameterSpec ivParams = new IvParameterSpec(iv);  
 cipher.init(Cipher.ENCRYPT\_MODE, key, ivParams);  
 Log.d(TAG, "Cipher IV: "  
 + (cipher.getIV() == null ? null : toHex(cipher.getIV())));  
 if (salt != null) {  
 return String.format("%s%s%s%s%s", toBase64(salt), DELIMITER,  
 toBase64(iv), DELIMITER, toBase64(plaintext));  
 }  
 return String.format("%s%s%s", toBase64(iv), DELIMITER,  
 toBase64(plaintext));  
 } catch (GeneralSecurityException e) {  
 throw new RuntimeException(e);  
 }  
 }  
  
 public static String toHex(byte[] bytes) {  
 StringBuffer buff = new StringBuffer();  
 for (byte b : bytes) {  
 buff.append(String.format("%02X", b));  
 }  
 return buff.toString();  
 }  
  
 public static String toBase64(byte[] bytes) {  
 return Base64.encodeToString(bytes, Base64.NO\_WRAP);  
 }  
  
 public static byte[] fromBase64(String base64) {  
 return Base64.decode(base64, Base64.NO\_WRAP);  
 }  
  
 public static String decrypt(byte[] cipherBytes, SecretKey key, byte[] iv) {  
 try {  
 Cipher cipher = Cipher.getInstance(CIPHER\_ALGORITHM);  
 IvParameterSpec ivParams = new IvParameterSpec(iv);  
 cipher.init(Cipher.DECRYPT\_MODE, key, ivParams);  
 Log.d(TAG, "Cipher IV: " + toHex(cipher.getIV()));  
 byte[] plaintext = cipher.doFinal(cipherBytes);  
  
 return new String(plaintext, "UTF-8");  
 } catch (Exception e) {  
 Log.e(TAG, "decrypt: ", e);  
 }  
 return null;  
 }  
  
 public static String decryptPbkdf2(String ciphertext, String password) {  
 String[] fields = ciphertext.split(DELIMITER);  
 if (fields.length != 3) {  
 throw new IllegalArgumentException("Invalid encypted text format");  
 }  
 byte[] salt = fromBase64(fields[0]);  
 byte[] iv = fromBase64(fields[1]);  
 byte[] cipherBytes = fromBase64(fields[2]);  
 SecretKey key = generateKey(salt, password);  
  
 return decrypt(cipherBytes, key, iv);  
 }  
  
}

public class Encryptor {  
 SecretKey key;  
 byte[] salt;  
  
 public SecretKey deriveKey(String password) {  
 salt = Crypto.generateSalt();  
 return Crypto.generateKey(salt, password);  
 }  
 public byte[] generateIV(int length){  
 return Crypto.generateIv(length);  
 }  
  
 public String encrypt(String plaintext,SecretKey secretKey,byte[] iv) {  
  
// key = deriveKey(password, salt);  
// Log.d("TAG", "Generated key: " + getRawKey());  
  
 return Crypto.encrypt(plaintext, secretKey,iv, salt);  
 }  
 public String encrypt(byte[] plaintext,SecretKey secretKey,byte[] iv) {  
  
// key = deriveKey(password, salt);  
// Log.d("TAG", "Generated key: " + getRawKey());  
  
 return Crypto.encrypt(plaintext, secretKey,iv, salt);  
 }  
  
 public String decrypt(String ciphertext, String password) {  
 return Crypto.decryptPbkdf2(ciphertext, password);  
 }  
}