

## Использование биометрической аутентификации в Android

*Андриенко Иван Сергеевич*

*Приамурский государственный университет имени Шолом-Алейхема*

*Студент*

### Аннотация

В данной статье описан процесс интеграции биометрической аутентификации в Android-приложении с использованием библиотеки BiometricPrompt. Рассмотрены ключевые аспекты настройки биометрической аутентификации, включая проверку доступности биометрических данных, создание диалогового окна для аутентификации и обработку результатов. В качестве примера приведена реализация простого приложения, где пользователь может подтвердить свою личность с помощью биометрии. Показано, как с помощью BiometricPrompt можно обеспечить безопасность и удобство использования приложения.

**Ключевые слова:** Android, биометрическая аутентификация, BiometricPrompt, безопасность данных, проверка личности, идентификация пользователя.

## Using Biometric Authentication in Android

*Andrienko Ivan Sergeevich*

*Sholom-Aleichem Priamursky State University*

*Student*

### Abstract

This article describes the process of integrating biometric authentication into an Android application using the BiometricPrompt library. The key aspects of setting up biometric authentication are considered, including checking the availability of biometric data, creating an authentication dialog, and handling results. As an example, a simple application is implemented where the user can verify their identity using biometrics. It shows how BiometricPrompt can be used to ensure the security and convenience of the application.

**Keywords:** Android, biometric authentication, BiometricPrompt, data security, identity verification, user identification.

## 1 Введение

### 1.1 Актуальность

С ростом использования мобильных приложений повышается важность защиты пользовательских данных. Биометрическая аутентификация является одним из наиболее безопасных и удобных методов подтверждения личности пользователя. Интеграция биометрической аутентификации позволяет

обеспечить высокий уровень безопасности, снижая риск несанкционированного доступа к данным. В Android платформе BiometricPrompt предоставляет стандартизированный способ внедрения биометрии в приложения, что делает процесс интеграции быстрым и простым для разработчиков. В условиях увеличивающихся требований к безопасности мобильных приложений, биометрическая аутентификация становится важным инструментом для защиты пользовательских данных.

## **1.2 Обзор исследований**

В работе А.В. Якимовича и А.С. Прохоренко рассматривается использование библиотеки Jetpack Security для шифрования данных в операционной системе Android. Авторы анализируют возможности библиотеки и ее роль в обеспечении безопасности данных, а также приводят примеры использования шифрования в мобильных приложениях [1]. П.А. Ребещенко в своей статье исследует биометрические системы аутентификации в мобильных устройствах, делая акцент на безопасности и надежности использования биометрических данных в различных сценариях [2]. В работе Д.И. Сычева рассматриваются проблемы безопасности в Android-приложениях, написанных на языках программирования Java и Kotlin. Автор анализирует основные уязвимости и предлагает методы их устранения [3]. Р.Э. Замалетдинова в своем исследовании обсуждает встроенные средства защиты данных на Android и их эффективность. Рассматриваются различные методы защиты, предоставляемые операционной системой [4]. И.А. Федонина проводит анализ уязвимостей системы хранения биометрических данных в операционной системе Android, предлагая рекомендации по их устранению [5].

## **1.3 Цель исследования**

Цель исследования – разработать подход к интеграции биометрической аутентификации в Android-приложение с использованием библиотеки BiometricPrompt. Это включает проверку доступности биометрических данных на устройстве, создание диалогового окна для аутентификации и обработку результатов.

## **2 Материалы и методы**

Для исследования использовались Android Studio, язык программирования Java и библиотека BiometricPrompt. В ходе разработки был создан интерфейс для проверки биометрической аутентификации, реализованы методы обработки данных и проверки успешности авторизации.

## **3 Результаты и обсуждения**

Биометрическая аутентификация в Android реализуется через библиотеку BiometricPrompt, которая предоставляет удобный интерфейс для работы с различными биометрическими данными, такими как отпечатки

пальцев и распознавание лица. Она включает в себя несколько ключевых компонентов:

1. **BiometricManager** – класс, позволяющий проверить, поддерживает ли устройство биометрию и настроены ли биометрические данные. Он используется для проверки условий перед запуском процесса аутентификации.

2. **BiometricPrompt** – класс, который отображает диалоговое окно для биометрической аутентификации. Он позволяет настроить параметры окна, такие как заголовок, подзаголовок и текст для кнопки отмены.

3. **PromptInfo** – объект, который определяет параметры для **BiometricPrompt**, включая заголовок, описание и текст кнопок (рис. 1).

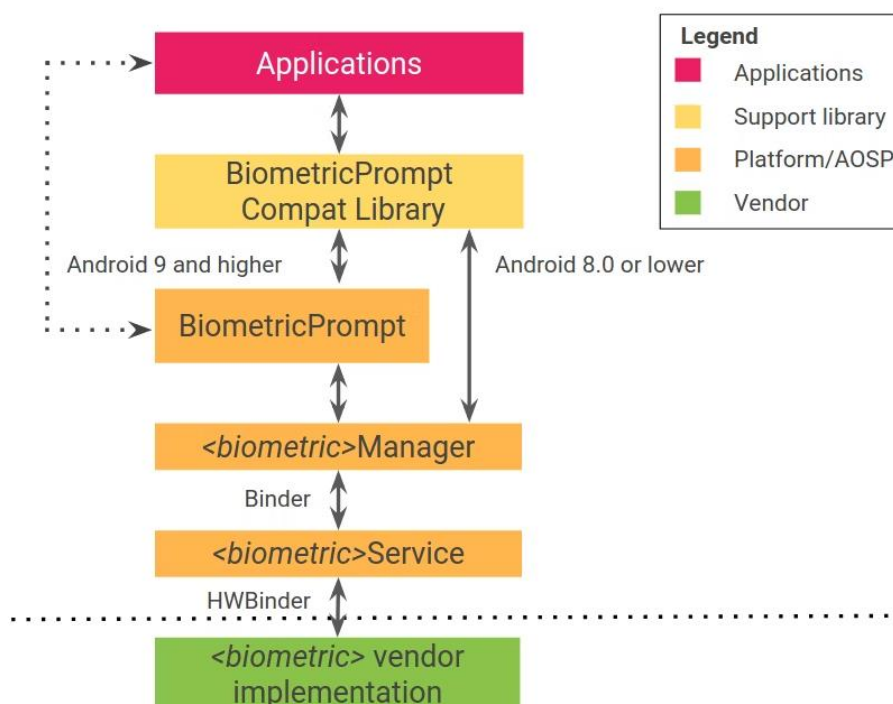


Рис. 1. Схема работы с BiometricPrompt

Для проверки биометрической аутентификации в Android-приложении был реализован следующий процесс. Сначала с помощью **BiometricManager** проверяется поддержка биометрии на устройстве. Затем создается **BiometricPrompt** для отображения диалогового окна и определения поведения при успешной или неуспешной аутентификации. В случае успешной аутентификации пользователь получает доступ к защищенным функциям приложения, например, к просмотру конфиденциальных данных.

В начале происходит проверка, поддерживает ли устройство биометрическую аутентификацию. Вызывается метод `canAuthenticate()` для проверки доступности биометрической аутентификации. Метод `canAuthenticate()` возвращает одно из значений, которое анализируется в конструкции `switch`. Если устройство поддерживает биометрию и она настроена, возвращается `BiometricManager.BIOMETRIC_SUCCESS`, и кнопка

для аутентификации (authButton) становится доступной для нажатия, так как аутентификация возможна. В случае, если на устройстве отсутствует биометрический сенсор, выводится сообщение с помощью Toast, информирующее пользователя об отсутствии датчика, и кнопка аутентификации отключается. Если биометрический сенсор временно недоступен, и пользователю также показывается соответствующее сообщение, после чего кнопка аутентификации отключается. Если на устройстве не зарегистрированы биометрические данные, пользователю выводится сообщение о необходимости регистрации биометрических данных, и кнопка аутентификации также отключается (рис. 2).

```
20 @Override
21 protected void onCreate(Bundle savedInstanceState) {
22     super.onCreate(savedInstanceState);
23     setContentView(R.layout.activity_main);
24
25     Button authButton = findViewById(R.id.authButton);
26
27     BiometricManager biometricManager = BiometricManager.from(context);
28     switch (biometricManager.canAuthenticate(authenticators: BiometricManager.Authenticators.BIOMETRIC_STRONG | BiometricManager.Authenticators.DEVICE_CREDENTIAL)) {
29         case BiometricManager.BIOMETRIC_SUCCESS:
30             authButton.setEnabled(true);
31             break;
32         case BiometricManager.BIOMETRIC_ERROR_NO_HARDWARE:
33             Toast.makeText(context, "На устройстве нет биометрических датчиков", Toast.LENGTH_SHORT).show();
34             authButton.setEnabled(false);
35             break;
36         case BiometricManager.BIOMETRIC_ERROR_HW_UNAVAILABLE:
37             Toast.makeText(context, "Биометрические датчики временно недоступны", Toast.LENGTH_SHORT).show();
38             authButton.setEnabled(false);
39             break;
40         case BiometricManager.BIOMETRIC_ERROR_NONE_ENROLLED:
41             Toast.makeText(context, "Не зарегистрированы биометрические данные", Toast.LENGTH_SHORT).show();
42             authButton.setEnabled(false);
43             break;
44     }
```

Рис. 2. Создание и настройка BiometricManager

Далее реализуем функциональность биометрической аутентификации в приложении. Сначала создается объект Executor с использованием метода `getMainExecutor()` из класса `ContextCompat`. Этот объект управляет выполнением задач в основном потоке пользовательского интерфейса, что необходимо для взаимодействия с элементами интерфейса, такими как кнопки и уведомления.

Далее создается объект `BiometricPrompt`, который является основным компонентом для управления биометрической аутентификацией. Метод `onAuthenticationSucceeded()` вызывается, когда аутентификация прошла успешно. Внутри этого метода отображается уведомление с сообщением "Аутентификация прошла успешно" с помощью `Toast`. Метод `onAuthenticationFailed()` вызывается, если аутентификация не удалась, и в этом случае пользователю отображается сообщение "Аутентификация не удалась".

Затем создается объект `BiometricPrompt.PromptInfo` с помощью `Builder`, который настраивает параметры интерфейса биометрической аутентификации. В настройках задаются заголовок окна ("Подтверждение личности"), подзаголовок ("Используйте биометрию для авторизации") и текст кнопки отмены ("Отмена"). Эти параметры определяют внешний вид диалога, который будет отображаться пользователю при запуске аутентификации.

В конце, устанавливается обработчик событий для кнопки аутентификации (authButton). При нажатии на кнопку вызывается метод authenticate(), который иницирует процесс аутентификации с использованием ранее созданного объекта promptInfo (рис. 3).

```

45
46     Executor executor = ContextCompat.getMainExecutor( context: this);
47     BiometricPrompt biometricPrompt = new BiometricPrompt( activity: MainActivity.this, executor, new BiometricPrompt.AuthenticationCallback() {
48         @Override
49         public void onAuthenticationSucceeded(BiometricPrompt.AuthenticationResult result) {
50             super.onAuthenticationSucceeded(result);
51             Toast.makeText(getApplicationContext(), text: "Аутентификация прошла успешно", Toast.LENGTH_SHORT).show();
52         }
53
54         @Override
55         public void onAuthenticationFailed() {
56             super.onAuthenticationFailed();
57             Toast.makeText(getApplicationContext(), text: "Аутентификация не удалась", Toast.LENGTH_SHORT).show();
58         }
59     });
60
61     final BiometricPrompt.PromptInfo promptInfo = new BiometricPrompt.PromptInfo.Builder()
62         .setTitle("Подтверждение личности")
63         .setSubtitle("Используйте биометрию для авторизации")
64         .setNegativeButtonText("Отмена")
65         .build();
66
67     authButton.setOnClickListener(view -> biometricPrompt.authenticate(promptInfo));
68 }
69

```

Рис. 3. Создание BiometricPrompt

После настройки компонентов следующий шаг включает в себя интеграцию BiometricPrompt в пользовательский интерфейс приложения. В рассматриваемом примере была создана кнопка, по нажатию на которую отображается диалоговое окно аутентификации. (рис. 4).

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/a
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:id="@+id/main"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".MainActivity">
9
10     <Button
11         android:id="@+id/authButton"
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:text="Авторизоваться"
15         app:layout_constraintBottom_toBottomOf="parent"
16         app:layout_constraintEnd_toEndOf="parent"
17         app:layout_constraintHorizontal_bias="0.498"
18         app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintTop_toTopOf="parent"
20         app:layout_constraintVertical_bias="0.207" />
21
22 </androidx.constraintlayout.widget.ConstraintLayout>

```

Рис. 4. Создание макета

Далее тестируем приложение (рис. 5, 6).

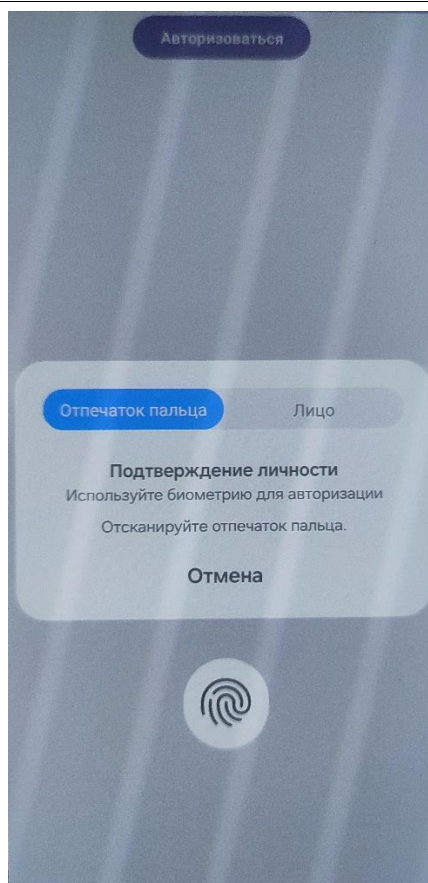


Рис. 5. Тестирование приложения



Рис. 6. Тестирование приложения

### **Выводы**

В данной статье рассмотрен процесс интеграции биометрической аутентификации в Android-приложении с использованием библиотеки BiometricPrompt. Рассмотрены основные этапы настройки, включая проверку доступности биометрии на устройстве, создание диалогового окна для аутентификации и обработку результатов. Приведенные примеры показывают, как можно легко и эффективно внедрить биометрию в мобильные приложения, повышая их безопасность и удобство использования.

### **Библиографический список**

1. Якимович А.В., Прохоренко А.С. Шифрование данных в операционной системе Android с Jetpack Security // Студенческий. 2020. № 15-1 (101). С. 45-46.
2. Ребещенко П.А. Биометрические системы аутентификации в мобильных устройствах // В сборнике: Студент: наука, профессия, жизнь. Материалы X всероссийской студенческой научной конференции с международным участием. В 5-ти частях. Омск, 2023. С. 57-61.
3. Сычев Д.И. Проблемы безопасности в Android-приложениях на Java и Kotlin // Обществознание и социальная психология. 2022. № 12 (42). С. 510-519.
4. Замалетдинова Р.Э. Встроенные средства защиты данных на Android // В сборнике: В мире научных открытий. Материалы V Всероссийской студенческой научной конференции (с международным участием). 2016. С. 232-234.
5. Федонина И.А. Исследование и анализ уязвимостей системы хранения биометрических данных в ОС Android // В сборнике: Современные проблемы телекоммуникаций. Материалы Международной научно-технической конференции. Новосибирск, 2020. С. 127-130.
6. URL: <https://developer.android.com/identity/sign-in/biometric-auth>(дата обращения 12.08.2024)