МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»**

**Московский техникум космического приборостроения**

### СПЕЦИАЛЬНОСТЬ: 09.02.07 Информационные системы и программирование (квалификация «Администратор баз данных»)

**П О Я С Н И Т Е Л Ь Н А Я З А П И С К А**

**к курсовому проекту по теме:**

## Использование фреймворков для безопасного программирования биометрической системы идентификации личности

|  |  |  |
| --- | --- | --- |
| Руководитель разработки от техникума | (подпись, дата) | Гончаренко С.В. |
| Разработчик | (подпись, дата) | Симонян П. Р. |

Москва 2021

**«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»**

**Московский техникум космического приборостроения**

УТВЕРЖДАЮ

Председатель ПЦК спец. 09.02.07

Н.А. Жилкина

« » 2021г.

**З А Д А Н И Е**

### на выполнение курсового проекта

по профессиональному модулю «ПМ.04 Сопровождение и обслуживание программного обеспечения компьютерных систем»

Студент Симонян П.Р. ТИП-51

(фамилия, инициалы, индекс группы)

Руководитель Гончаренко С.В.

(фамилия, инициалы)

График выполнения работы: 25% к 4 нед., 50% к 8 нед., 75% к 12 нед., 100% к 15 нед.

1. ***Тема курсового проекта***

Использование фреймворков для безопасного программирования биометрической идентификации личности

1. ***Техническое задание***

Сравнить возможности фреймворков для безопасного программирования биометрической идентификации личности

1. ***Оформление курсового проекта***
   1. Расчетно-пояснительная записка на 41 листе формата А4.
   2. Перечень графического материала КП (плакаты, схемы, чертежи и т.п.) – схемы алгоритма

Дата выдачи задания «\_ » 2021г.

Руководитель курсового проекта Гончаренко С.В.

# СОДЕРЖАНИЕ

[Введение 4](#_TOC_250005)

1. [Постановка задачи 5](#_TOC_250004)
2. [Теоретические основы 8](#_TOC_250003)
3. [Проведение сравнения 1](#_TOC_250003)3
4. Подведение итогов 22

[Заключение 2](#_TOC_250000)5

[Список использованных источников 2](#_TOC_250000)6

Приложение А 27

Приложение Б 28

Приложение В. 29

Приложение Г 31

Приложение Д 34

Приложение Е 39

# ВВЕДЕНИЕ

Благодаря развитию аппаратного обеспечения, в XXI веке большую популярность обрели биометрические датчики, которые позволяют сделать взаимодействие с различными устройствами проще и безопаснее. В отличие от классических данных, используемых при аутентификации, таких как пароль или электронные ключи (токенов), биометрические данные не могут быть переданы, так как индивидуальны для каждого человека. Биометрия не всегда может быть такой же безопасной, как сложные пароли, однако она значительно упрощает идентификацию личности.

Существует множество фреймворков для реализации функций биометрических сканеров и анализа собранной информации. В данной работе будут рассмотрены самые популярные из них, а также будет проведено их сравнение.

# ПОСТАНОВКА ЗАДАЧИ

Целью курсового проекта является сравнение возможностей фреймворков для безопасного программирования биометрической идентификации личности.

Основания для проведения работы: учебный план специальности, рабочая программа дисциплины, распоряжение по учебному заведению.

Требования к курсовому проекту: требования к структуре документов определены соответствующими стандартами ЕСКД и КСПД; требования к оформлению определены соответствующими методическими указаниями.

Порядок контроля и приёмки курсового проекта: контроль выполнения курсового проекта проводится руководителем поэтапно в соответствии с утверждённым графиком выполнения курсового проекта; на завершающем этапе руководитель осуществляет нормоконтроль представленной исполнителем документации и принимает решение о допуске (недопуске) курсового проекта к защите.

При работе над курсовым проектом учитывать: полноту раскрытия рассматриваемой темы, актуальность выбираемых фреймворков, корректность их сравнения.

Плановые сроки выполнения: 1 семестр 2021 учебного года.

Исходные данные: в курсовом проекте необходимо провести сравнение на основе официальной документации фреймворков и подтверждённой информации, находящейся в открытом доступе.

Состав курсового проекта:

* Текстовые и графические документы;
* Программная и технологическая документация (при необходимости);
* Пояснительная (расчётно-пояснительная) записка;
* Обязательные структурные элементы: титульный лист (одна страница), содержание (от одной страницы), ведение (от одной страницы), список литературы (от одной страницы), заключение (от одной страницы),
* Объём основной части работы: минимум два раздела с минимум двумя параграфами (от пяти страниц) в каждом разделе.

Содержание пояснительной записки:

* Титульный лист;
* Лист задания;
* Содержание;
* Введение;
* Исходные данные;
* Основной раздел (теоретическая, аналитическая и проектно-аналитическая часть);
* Заключение;
* Список использованной литературы;
* Приложения.

Детализированное техническое задание на курсовой проект:

* во введении обосновать актуальность исследуемой темы, степень её изученности, дать критический обзор работ, характеризующих сущность рассматриваемого вопроса, определить цель и задачи, описать применяемые методы исследования;
* в теоретической части изложить теоретические концептуальные основы исследуемой проблемы и собственное понимание существа вопросов исследуемой темы, сформировать своё аргументированное мнение по неоднозначным вопросам;
* в аналитической части изложить особенности функционирования объекта исследования, проанализировать и раскрыть особенности сбора, обработки и подготовки информации о процессах, целесообразно рассмотреть порядок формирования и представления информации об отдельных объектах исследования;
* в проектно-аналитической части разработать и внести предложения по рациональной методологии исследования объекта с использованием современных информационных технологий.
* в заключении изложить основные результаты исследования, сформулировать выводы, предложения и перспективы дальнейшей разработки темы.

# ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

**Биометрия** - это идентификация человека по уникальным биологическим признакам.

В современные устройства и специализированные средства защиты биометрия встраивается все чаще по следующим причинам:

1. Оказывается минимальная когнитивная нагрузка. Для проведения идентификации требуется совершить интуитивные, не требующие умственных усилий или запоминания, действия: коснуться сканера, посмотреть в камеру, произнести фразу.
2. С приходом в массово - потребительские устройства биометрия стала гораздо доступнее и востребованнее.
3. Биометрические сканеры не всегда работают безотказно, но все же остаются достаточно безопасным методом идентификации личности.
4. Тенденция усиления защиты приватности удачно сочетается с тем фактом, что биометрические данные невозможно передать и крайне тяжело подделать.

Остановимся подробнее на последнем пункте.

**Биометрические данные** - уникальные биологические признаки людей. Самыми востребованными являются:

1. Отпечатки пальцев
2. Сетчатка глаза
3. Голос

В отличие от паролей, которые сейчас используются повсеместно, биометрические данные обладают одинаковой сложностью. Это вносит дополнительную защиту, так как пользователь не может сделать этот способ идентификации личности менее надёжным, как если бы он придумал недостаточно сложный и легко угадываемый пароль.

Помимо этого, биометрические данные гораздо тяжелее украсть (или, в случае биометрии, подделать). Скопировать их представляется возможным только для специально обученных людей.

Однако возникают сложности, которые все же не дают гарантию безопасности данного метода идентификации:

1. Все же существуют люди с очень похожими биометрическими признаками. Например, близнецы или люди с похожим тембром голоса.
2. Ключевую роль в обеспечении безопасности и точности биометрии берет на себя разработчик биометрических датчиков и программного обеспечения для них.

Следует учитывать, что надёжность биометрии, в отличие от обычных паролей, сильнее зависит от качества аппаратного и программного продукта, которое обеспечивает её. Потому решения сильно отличаются по качеству, удобству и цене. Продукты для государственных структур, требующие исключительную надёжность и подразумевающие усиленную нагрузку, будут сильно отличаться от решений, используемых в современных смартфонах. Конкретные различия данных продуктов будут рассмотрены в контексте сравнения.

**Фреймворк** - это программный продукт, упрощающий создание и поддержку технически сложных и нагруженных проектов. Обычно задачу программирования биометрических датчиков, методов сбора и обработки информации, полученных с их помощью, берут на себя крупные и проверенные компании или разработчики данных сканеров. Примером вторых могут послужить решения компании BioLink, которая сама выпускает биометрические датчики и комплекты для разработки программного обеспечения (SDK) под них. Но чаще всего встречаются SDK компаний, агрегирующих аппаратное обеспечение разных производителей, чтобы предоставить разработчикам общий интерфейс для их использования. Такими являются Google и Microsoft.

Таким образом, фреймворки способствуют:

А) Усиленной безопасности. Перед тем, как перейти в использование конечным пользователям (в данном случае, это пользователи фреймворков - разработчики), решения для программирования биометрии тщательно тестируются организациями, которые специализируются на этом. В других случаях, такие компании, как, например, Microsoft, имеют все средства для того, чтобы провести качественное тестирование собственных программных решений и предоставить разработчикам безопасный программный продукт.

Б) Облегчённой интеграции. Решения для программирования биометрии, как правило, хорошо задокументированы и не требуют углублённых знаний программиста о работе биометрических сканеров. Ещё проще дела обстоят у разработчиков для массовых платформ: Apple, Google, Microsoft и некоторые другие компании предлагают единые решения для разработчиков под их экосистемы. Пользователю их фреймворков гораздо меньше приходится принимать во внимание разнообразие биометрических датчиков, используемых в конкретных устройствах разных производителей или моделей.

В) Популяризации усиленной безопасности с помощью биометрии. Использование особенностей современных биометрических датчиков может позволить себе любой разработчик, который должен лишь использовать готовое решение в виде фреймворка и изучить документацию к нему. Благодаря этому биометрия пришла в банковскую сферу, сферу онлайн - платежей, а также в другие массовые продукты, предъявляющие специальные требования к безопасности.

Теперь, разобравшись в преимуществах и важности биометрии, стоит рассмотреть некоторые самые популярные решения, которые на данный момент доступны разработчикам.

OpenBR - фреймворк с открытым исходным кодом, позволяющий распознавать лица, возраст и пол по фотографии.

NEUROtechnology Megamatcher SDK - Дорогой профессиональный продукт, ориентированный на крупные компании и государственные структуры, которые предъявляют к безопасности и отказоустойчивости особые требования. Введён в список сравнения для того, чтобы дать представление о том, чем профессиональные решения отличаются от массово - потребительских.

Huawei FIDO (Fast Identity Online) BioAuthn - Фреймворк для использования биометрии на смартфонах и планшетах компании Huawei.

Samsung Knox SDK - Масштабный фреймворк компании Samsung, включающий в себя возможность идентификации личности по отпечатку пальца или радужной оболочки глаза.

Apple Local Authentication Framework - Фреймворк, позволяющий использовать биометрию на всех устройствах компании Apple, аппаратно поддерживающих её: от iPhone до iMac с использованием соответствующей клавиатуры.

Google’s Android Biometric Library - Фреймворк для использования биометрии на устройствах под операционной системой Android.

Microsoft Windows Biometric Framework - Фреймворк для использования биометрии в операционной системе Windows с поддержкой самых разных производителей: от Lenovo до Honor.

Такое разнообразие программных решений определяется не только различием платформ для разработки, но и требованиями конечного пользователя. Например, пользователь смартфона предъявляет требования как к безопасности, так и к удобству и скорости считывания биометрических данных, не говоря уже о цене. Поэтому производители аппаратных и программных решений стараются балансировать между этими характеристиками, пока более профессиональные решения не представляются доступными для использования в массовых продуктах. Развитие в этом направлении происходит постоянно. Некоторое время назад использование достаточно надёжных датчиков отпечатка пальца в бюджетных устройствах было невозможно, но с развитием технологии данная технология стала обыденностью. Как и считывание топологии лица с помощью лазерного сканирования раньше было фантастикой, а сейчас доступно в мобильных устройствах высокого класса, а позже, вероятно, станет такой же обычной практикой, как и датчик отпечатка пальца сейчас.

С другой стороны, биометрия уже давно используется в специализированных решениях обеспечения безопасности. Так, систему Megamatcher используют правительства и силовые структуры некоторых стран, силовые и правоохранительные органы, предприятия, имеющие государственную важность, в том числе аэропорты и банки. Компромиссные и доступные решения не удовлетворяют высоким требованиям данных организаций. Поэтому именно в таких сферах встречаются инновационные и передовые решения, вроде массового распознавания лиц по видео, распознавание особых биометрических признаков вроде вен, отпечатков ладоней, технологии классификации распознанных людей по различным признакам, моментальная идентификация людей для поиска разыскиваемых преступников, безотказные решения, которые могут использоваться для идентификации личности при совершении важных финансовых операций. Ошибки, возможные при идентификации обычного пользователя смартфона, на промышленном и государственном уровне по очевидным причинам просто недопустимы.

Выделим следующие характеристики, которые позволят сравнить данные фреймворки:

1. **Доступные платформы.**

Какие платформы и операционные системы способны взаимодействовать с биометрическими данными через данный фреймворк?

1. **Доступные для сканирования биометрические данные.**

С какими данными может работать фреймворк? Среди них: данные отпечатков пальцев, ладоней, голоса, сетчатки глаза, рисунка вен.

1. **Доступность фреймворка.**

На каких языках программирования осуществляется использование фреймворка? Какие инструменты необходимо приобрести для разработки на нём? Какие представляются возможности отладки фреймворка?

1. **Поддерживаемые модели сканеров.**

Для какого аппаратного обеспечения предназначен фреймворк?

1. **Качество документации.**

Оценивается полнота и актуальность документации, наличие примеров и готовых шаблонов проектов, учебников.

1. **Доступность документации.**

Локализована ли документация? Имеются ли онлайн и оффлайн версии документации?

1. **Дороговизна разработки на данном фреймворке.**

Каких вложений требует минимальная разработка и отладка программ, использующих данный фреймворк?

1. **Уровень поддержки разработчиком фреймворка.**

Чьими силами и как обеспечивается поддержка и совместимость фреймворка в соответствии с развитием биометрии?

1. **Лицензирование.**

По какой лицензии распространяется фреймворк?

Для структурирования информации о фреймворках, было решено поделить их характеристики на три группы:

1. **Функциональные.** Это те характеристики, которые определяют реализацию функций фреймворка в конечных продуктах. Сюда относятся доступные платформы (пункт 1 предыдущего списка), доступные биометрические данные (пункт 2), доступность фреймворка (пункт 3), доступные модели сканеров (пункт 4).
2. **Уровень поддержки документации.** Данные характеристики определяют качество документации и удобство поддержки конечных продуктов, использующих конкретный фреймворк. Сюда относятся: качество документации (пункт 5) и доступность документации (пункт 6).
3. **Финансовая обоснованность использования.** Здесь подразумевается перспективность использования фреймворка. Например, хорошим показателем будет активная поддержка фреймворка или открытость, позволяющая модифицировать его в соответствии с нуждами разработчика. Сюда относится дороговизна разработки на данном фреймворке (пункт 7), уровень поддержки разработчиком (пункт 8) и лицензирование (пункт 9).

# ПРОВЕДЕНИЕ СРАВНЕНИЯ

Изучив все теоретические сведения и определив критерии оценивания фреймворков, можно представить их сравнительную характеристику в виде трёх таблиц, описывающих разные группы характеристик.

Таблица 3.1 - сравнение функциональных характеристик фреймворков.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Фреймворк** | **Доступные платформы** | **Доступные биометрические данные** | **Доступность фреймворка** | **Доступные модели сканеров** |
| OpenBR | Windows, Mac OS X, Debian Linux. | Фотография лица | Для использования OpenBR, его необходимо собрать с помощью CMake. Также необходимо наличие Qt и OpenCV при сборке.  API фреймворка доступен на языке C++, а также «обёртка», как её называют разработчики в официальной документации, в виде Python API.  Отладка фреймворка может производиться на устройстве разработчика, биометрические датчики не требуются. | Фреймворк поддерживает обработку лиц на любых фотографиях. Следовательно, он требует лишь фотографию, содержащую различимое лицо, а не специализированное оборудование. |

Продолжение таблицы 3.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Megamatcher SDK | Windows, Mac OS, Linux, Android, IOS, серверные конфигурации Microsoft Linux | Отпечаток пальца, радужная оболочка, лицо (фотография и 3D модель), голос, отпечаток ладони | Фреймворк необходимо приобрести. Доступен пробный период в 30 дней.  Поддерживаются следующие языки программирования: C#, C++, Java.  Отладка приложения производится на целевой платформе с использованием требующихся биометрических датчиков. | Фреймворк поддерживает десятки камер, датчиков отпечатков пальцев и ладоней. Поддержка также включает в себя биометрические датчики в устройствах на Android и IOS без покупки дополнительного профессионального оборудования. Поддерживается любой микрофон для идентификации по голосу или идентификация по звуковому файлу, содержащему голос. |

Продолжение таблицы 3.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Huawei FIDO BioAuthn | EMUI (смартфоны и планшеты Huawei) | Отпечаток пальца, лицо (фотография или 3D модель) | Для использования фреймворка необходимо завести аккаунт разработчика и приложения Huawei App Gallery, а затем скачать SDK. FIDO использует онлайн сервис для идентификации и дополнительных кроссплатформенных возможностей, тогда как BioAuthn позволяет использовать биометрические возможности устройства локально. Разработка доступна на самых популярных платформах, таких как: Java, Xamarin, Flutter и т.д. Отладка производится на мобильных устройствах Huawei. Сложность конфигурации фреймворка и требование к настройке онлайн сервиса может оказаться довольно трудновыполнимым. | Фреймворк поддерживает только датчики, встроенные в устройства Huawei. Совместимость с 3D распознаванием лица зависит от модели устройства. |
| Samsung Knox SDK Biometric Authentication. | One UI (смартфоны и планшеты Samsung) | Отпечаток пальца, лицо, радужная оболочка глаза | Для использования фреймворка необходимо зарегистрировать аккаунт Samsung и скачать Knox SDK. Для программирования используется Java и Android Studio. Отладка возможна только на устройствах Samsung. | Фреймворк поддерживает только датчики, встроенные в устройства Samsung. |

Продолжение таблицы 3.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Apple Local Authentication Framework | iOS, iPadOS, macOS, Mac | Отпечаток пальца, 3D модель лица | Фреймворк встроен в SDK Apple, разработанный Apple и использующий язык Swift и Objective-C. Официально фреймворк доступен только для разработки на компьютерах Apple, но существуют возможности использования на кроссплатформенных фреймворках (Flutter, React Native, Hybrid и т.д.). Для скачивания фреймворка необходимо зарегистрировать аккаунт разработчика Apple. Который, впрочем, все равно потребуется для публикации приложения. Существуют неофициальные порты на кроссплатформенные фреймворки вроде Flutter. | Фреймворк поддерживает только датчики, встроенные в устройства Apple. |
| Android Biometric Library | Android 9+ (смартфоны и планшеты) | Отпечаток пальца, лицо (фотография или 3D модель), радужная оболочка глаза | Для использования фреймворка нужно установить его в Android Studio или использовать другие кроссплатформенные фреймворки. Разработка доступна на множестве языков, соответствующих фреймворкам для разработки приложений: Java, Kotlin, Dart, Flutter, JavaScript и другие. | Фреймворк поддерживает камеры устройств и самые распространённые биометрические датчики в них: сканер отпечатка, сканер радужной оболочки глаза, 3D сканеры. Совместимость фреймворка с данными технологиями сильно зависит от производителя устройства. |

Продолжение таблицы 3.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Windows Biometric Framework | Начиная с Windows Server 2008 R2 и Windows 7, а также устройства, поддерживающие Windows Hello. | Отпечаток пальца, лицо модель или фотография), радужная оболочка глаза. | Официальная документация описывает использование биометрических функций только в UWP. Доступны языки C, C++ и C#. | Фреймворк поддерживает биометрические устройства, встроенные в компьютер под управлением Windows. |

Таблица 3.2 - сравнение уровня поддержки документации фреймворков.

|  |  |  |
| --- | --- | --- |
| **Фреймворк** | **Качество документации** | **Доступность документации** |
| OpenBR | Документация проекта покрывает все возможности фреймворка, включает в себя небольшое пошаговых учебников, подробное описание функций API, готовые примеры программ, руководство по установке фреймворка и всех его зависимостей с нуля. Документация содержит пояснительные изображения. | Документация доступна только на английском языке. Имеется онлайн версия, а оффлайн документация включена в открытый репозиторий GitHub, так что при установке фреймворка она в любом случае будет доступна в виде .md файлов. |
| Megamatcher SDK | Документация фреймворка насчитывает 2859 страниц с подробным описанием всех аспектов работы фреймворка, изображениями, примерами на разных языках программирования, описанием конфигурации баз данных MS SQL Server, MySQL, Oracle, PostgreSQL, SQLite. Нужно отметить полноту документации и удобство ориентирования по ней, а также наличие множества обучающих разделов. | Документация доступна только на английском языке. Файл документации хранится в PDF формате и доступен для скачивания. |
| Huawei FIDO BioAuthn | Документация содержит руководство по всем этапам разработки приложения с биометрией и видеоролики, обозревающие возможности фреймворка. Доступен форум разработчиков. Стоит отметить некоторые ошибки в переводе обучающих статей на английский с китайского. | Документация доступна онлайн на 8 языках. Русский недоступен. |

Продолжение таблицы 3.2.

|  |  |  |
| --- | --- | --- |
| Samsung Knox SDK | Доступен форум разработчиков. Ориентирование в документации удобно за счёт многостраничного интерфейса, списка классов. Содержит примеры использования классов. Не содержит полного руководства по внедрению. Встречаются неработающие ссылки. | Документация доступна онлайн на 7 языках. Русский недоступен. |
| Apple Local Authentication Framework | Фреймворк относительно простой за счёт монолитности дизайнерских решений Apple. Примеры приведены на языках Swift и Objective-C. Ориентирование по API пошаговое и больше опирается на готовые программные решения, нежели на описание инструментов для реализации, что должно позволить ускорить разработку и исключить ошибки в разработке. Компоненты классов отсортированы по назначению. Отдельное внимание уделяется оповещению разработчиков об изменении документации. | Документация доступна только на английском языке. |
| Android Biometric Library | Документация размещена на глобальной платформе, которая также имеет форум, множество примеров проектов и обучающие видеоролики. Примеры кода размещены на языках Java и Kotlin. Ориентирование по документации соблюдает логику наследования, удобное и интуитивное. Имеются автоматические рекомендации для перехода к соответствующим функциям и обучающим статьям по фреймворку. | Документация доступна онлайн на 7 языках. Русский язык доступен с помощью встроенного переводчика. |

Продолжение таблицы 3.2.

|  |  |  |
| --- | --- | --- |
| Windows Biometric Framework | Документация содержит несколько статей об использовании фреймворка, но эта информация находится довольно сложно, содержит продвинутый английский язык и будет трудна для изучения. Обучающие статьи поддерживаются разными участниками сообщества, заметен недостаток материала, его некоторая неактуальность и сложность для понимания. Автоматический перевод документации содержит ошибки, которые не позволяют его использовать. | Онлайн документация к фреймворку доступна только на английском. Возможно скачивание документации, но за ненадобностью эта функция может пропасть с сайта 3 Января 2022 года. |

Таблица 3.3 - сравнение финансовой обоснованности использования фреймворков.

|  |  |  |  |
| --- | --- | --- | --- |
| **Фреймворк** | **Дороговизна разработки на фреймворке** | **Уровень поддержки разработчиком** | **Лицензирование** |
| OpenBR | Фреймворк распространяется бесплатно. Это значит, что стоимость разработки на нём определяется только сложностью проекта, а тестирование может проходить на машине разработчика. | Активная стадия разработки фреймворка завершилась в 2015 году, но до сих пор разработчик выполняет оптимизацию и улучшает разработку, публикуя изменения на GitHub. | Фреймворк распространяется под лицензией Apache 2.0. Это значит, что исходный код доступен для изучения и модификации. Разработчик не даёт никаких гарантий безотказности и надёжности продукта. |

Продолжение таблицы 3.3.

|  |  |  |  |
| --- | --- | --- | --- |
| Megamatcher SDK | Цены на пожизненную лицензию начинаются от 216 тысяч рублей. Ценовая политика актуальна для крупных компаний и государственных организаций. | Фреймворк обновляется по сей день. Разработчик оказывает индивидуальную поддержку заказчикам и потенциальным покупателям. | Лицензия покупается один раз и на всегда для всех типов используемых компонентов и платформ, на которых планируется использование биометрии. Возможен перенос или обновление лицензии в случае обновления аппаратного обеспечения. SDK покупается один раз на всю компанию, которая собирается интегрировать продукт.  Лицензия содержит положение о том, что все официально заявленные функции фреймворка работают безотказно, но ПО распространяется «как есть» и может содержать какие-то ошибки. Разработчик обязуется заменить ПО с ошибкой в течение тридцати дней. |
| Huawei FIDO и BioAuthn | Использование фреймворка бесплатно. Для отладки приложения требуется устройство Huawei, поддерживающее биометрические функции. | Фреймворк не совместим с новой операционной системой компании Harmony OS, что помешает разработчику портировать свое приложение под новые версии ПО устройств Huawei.  Не смотря на это, ПО остается актуальным и активно поддерживается разработчиком. | Модификация продукта не допускается. Huawei не предоставляет никаких гарантий работы фреймворка и сохранения данных и возлагает всю ответственность за использование фреймворка на его пользователя согласно официальному документу «Agreement on Use of Huawei APIs». |
| Samsung Knox SDK | Использование фреймворка бесплатно, но доступны некоторые платные функции. Для отладки приложения требуется устройство Samsung, поддерживающее биометрические функции. | Фреймворк активно поддерживается и, помимо биометрии, развивается в других направлениях удобства и безопасности. | Модификация продукта не допускается. Для использования необходимо активировать бесплатную лицензию. Samsung не исключает свою ответственность в случае нарушения безопасности устройств по вине разработчиков фреймворка. |

Продолжение таблицы 3.3.

|  |  |  |  |
| --- | --- | --- | --- |
| Apple Local Authentication Framework | Использование фреймворка бесплатно. Для отладки требуется устройство Apple с возможностью биометрической идентификации. Для использования Xcode - официального инструмента разработчика под устройства Apple - требуется компьютер этой компании, что является большим вложением и доступно далеко не всем мелким разработчикам. Альтернативным решением будет использование кроссплатформенных фреймворков для разработки приложений. | Фреймворк активно поддерживается разработчиком. | Модификация продукта не допускается. В соглашении компания - разработчик отказывается от любых гарантий работы фреймворка и обязательного исправления его проблем и дополнительно подчёркивает, что данное ПО не предназначено для обеспечения безотказной работы и не соблюдает требования повышенной безопасности. |
| Android Biometric Library | Фреймворк бесплатный. Для отладки необходимо устройство на базе Android 9+ или его эмулятор. | Фреймворк активно поддерживается разработчиком. | Исходный код доступен для изучения и модификации. Фреймворк распространяется под лицензией GPLv2, предусматривающей отказ от ответственности за ошибочную работу ПО. |
| Windows Biometric Framework | Использование фреймворка бесплатно, для отладки необходимо совместимое устройство. | Из-за непопулярности использования фреймворка в приложениях, поддержка документации и обучающих ресурсов остаётся под вопросом. В основном возможности Windows Hello использует сама Microsoft в некоторых своих продуктах. | Модификация не представляется возможной. Разработчик не гарантирует безотказной работы фреймворка, если при заключении другого соглашения с ним не указано иное. |

В ознакомительных целях в приложении к курсовому проекту были предоставлены примеры использования биометрической аутентификации (отпечаток пальца или лицо) на предложенных фреймворках.

# ПОДВЕДЕНИЕ ИТОГОВ

Изучив подробную сравнительную характеристику, уже можно сделать определённые выводы касательно целей использования того или иного программного решения.

**OpenBR**

Данный фреймворк предназначен в основном для анализа лиц по фотографиям, а не для использования в целях безопасности. Об этом, помимо функциональных возможностей фреймворка, говорит его открытая лицензия, не подразумевающая требование к надёжности и безотказности. Эта библиотека достаточно небольшая, что может быть полезно для целей изучения кода и принципов распознавания лиц. Она может использоваться в небольших некоммерческих приложениях в целях развлечения или анализа.

**Megamatcher SDK**

Данное программное обеспечение на контрасте с предыдущим объектом исследования представляет из себя передовое решение для обеспечения безопасности на крупных предприятиях. В основном оно ориентировано на масштабируемость. Дополнительно стоит отметить поддержку и лицензионное соглашение, защищающее пользователя от неисправного ПО (если быть точным, дающее ему гарантию на то, что ПО будет исправлено в кратчайшие сроки). Дороговизна использования отражается в возможности использования непопулярных биометрических датчиков и их связок, качестве и полноте документации и обучающих материалов, возможностях оказания технической поддержки предприятиям. Главным недостатком является проблема доступа к полному программному и аппаратному обеспечению организации в России и многих других странах.

**Huawei FIDO и BioAuthn**

Сервисы HMS Core компании Huawei ориентированы на интеграцию с облачными технологиями для обеспечения тесной интеграции между различными платформами. Но даже если считать данный метод достаточно безопасным, сложность интеграции в собственные приложения остаётся серьёзным барьером для разработчиков небольших и даже средних организаций, которые не пожелают лишний раз регистрироваться в сервисе для разработчиков ради сомнительной экономической выгоды.. Дополнительно усугубляет положение переход Huawei на операционную систему Harmony OS, которая, по непонятным обстоятельствам, не поддерживает данный фреймворк. Я могу посоветовать его только тем, кто посчитает возможности сервиса FIDO достаточно важными для интеграции в свой продукт, но условия лицензионного соглашения и отказ от ответственности разработчика всё же не дадут использовать предложенное решение в серьёзных коммерческих и государственных предприятиях.

**Samsung Knox Biometric Authentication**

В последнее время Samsung предпринимает большие усилия для того, чтобы стать востребованным бизнес решением. Вместе с программным пакетом Knox, Samsung предоставил дополнительные возможности обеспечения безопасности в своих устройствах. Большинство компонентов Knox остаются бесплатными, но, не смотря на это, Samsung, хоть и в довольно расплывчатой формулировке, заявляет ответственность за свой продукт, что позволяет использовать его на уровне предприятия. Как итог, я бы посоветовал это решение тем, кому необходимо выстроить на предприятии тесную информационную структуру, так как в Knox Samsung предлагает множество дополнительных возможностей, повышающих безопасность устройств. Если в этих возможностях нет нужды, стоит воспользоваться универсальным решением, предлагаемым Google в Android SDK.

**Apple Local Authentication**

Для всех своих устройств - от компьютеров до мобильных устройств, Apple предлагает единое решение. За счёт его атомарности стала возможна простота внедрения для любого требуемого сценария. Главной проблемой для небольших команд может стать трудность разработки ПО на устройства Apple без Mac и Macbook, цена которых одна из самых высоких на рынке компьютеров. Лицензия ПО и ориентирование компании на простого пользователя «умными» устройствами не даёт возможность использовать это решение в целях обеспечения безопасности крупного предприятия. Я могу посоветовать данный фреймворк тем, у кого есть возможность нативной разработки по устройства Apple, так как это гарантирует лучший пользовательский опыт и качество массово - потребительского уровня.

**Android Biometric Library**

Фреймворк программирования биометрии Android гарантирует простоту портируемости приложения на самые разные устройства, поддерживающие данную операционную систему. Устройства под Android значительно дешевле, что позволит легко отладить создаваемую программу даже при малом бюджете. Открытый исходный код и лицензия позволяет взять код программного решения за основу для модификации. Однако на личном примере были замечены некоторые проблемы и неточности работы данного фреймворка, которые не позволяют назвать его достаточно безопасным. Я могу посоветовать фреймворк всем, для кого приватность имеет некритичную важность.

**Windows Biometric Framework**

Биометрия в компьютерах используется крайне редко. Я могу отметить два сценария: разблокирование устройства и скачивание ПО из встроенного магазина. По понятным причинам, обучающие материалы и документация по фреймворку для биометрической аутентификации в Windows находится в плохом состоянии. Данное решение сложно применить без кропотливого изучения документации, а для большинства приложений оно будет просто бесполезно. Потому я могу посоветовать этот фреймворк только для организаций, способных потратить ресурсы на его изучение и внедрение и имеющих такую необходимость. Разработчикам поменьше стоит посмотреть на другие решения или вовсе пересмотреть потребность в использовании биометрии в своей программе.

# ЗАКЛЮЧЕНИЕ

Как видно из данного сравнения, каждый фреймворк для программирования биометрии отвечает своим требованиям и стандартам безопасности, что позволяет сделать эту передовую технологию актуальной как для обычного пользователя, так и для крупнейших корпораций. Я могу утверждать: почти каждый из этих фреймворков является достаточно безопасным. Главным критерием в разработке простых приложений станет доступность той или иной технологии на разных платформах. Более крупным компаниям потребуется сильная безопасность. Благодаря развитию рынка биометрических сенсоров, способов обработки биометрических данных и поддержке крупных корпораций, могут быть обеспечены потребности обеих групп разработчиков.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кухарев Г. А.  Биометрические системы: Методы и средства идентификации личности человека / Политехника, 2001. — 240 с.
2. ГОСТ 2.120 – 73, ЕСКД. Отчёт о научно-исследовательской работе. – Введ. 01.01.1974, срок действия до 01.01.1982. – М.: Изд-во стандартов, 1974. – 17с.
3. <http://openbiometrics.org/>
4. <https://docs.microsoft.com/en-us/windows/win32/secbiomet/biometric-service-api-reference>
5. <https://developer.apple.com/documentation/localauthentication/>
6. <https://developer.huawei.com/consumer/en/codelab/HMSCoreFIDOBioAuthn/index.html#0>
7. <https://docs.samsungknox.com/admin/whitepaper/kpe/biometric-authentication.htm>
8. <https://docs.samsungknox.com/dev/knox-sdk/biometrics.htm>
9. <https://docs.microsoft.com/en-us/windows/security/identity-protection/hello-for-business/hello-biometrics-in-enterprise>
10. <https://www.computerworld.com/article/3244347/what-is-windows-hello-microsofts-biometrics-security-system-explained.html>

# ПРИЛОЖЕНИЕ А

## (обязательное)

Пример кода для сравнения фотографий лиц в фреймворке OpenBR на языке C++

Листинг файла face\_recognition.cpp:

#include <openbr/openbr\_plugin.h>

static void printTemplate(const br::Template &t)

{

const QPoint firstEye = t.file.get<QPoint>("Affine\_0");

const QPoint secondEye = t.file.get<QPoint>("Affine\_1");

printf("%s eyes: (%d, %d) (%d, %d)\n", qPrintable(t.file.fileName()), firstEye.x(), firstEye.y(), secondEye.x(), secondEye.y());

}

int main(int argc, char \*argv[]) {

br::Context::initialize(argc, argv);

// Retrieve classes for enrolling and comparing templates using the FaceRecognition algorithm

QSharedPointer<br::Transform> transform = br::Transform::fromAlgorithm("FaceRecognition");

QSharedPointer<br::Distance> distance = br::Distance::fromAlgorithm("FaceRecognition");

// Initialize templates

br::Template queryA("../data/MEDS/img/S354-01-t10\_01.jpg");

br::Template queryB("../data/MEDS/img/S382-08-t10\_01.jpg");

br::Template target("../data/MEDS/img/S354-02-t10\_01.jpg");

// Enroll templates

queryA >> \*transform;

queryB >> \*transform;

target >> \*transform;

printTemplate(queryA);

printTemplate(queryB);

printTemplate(target);

// Compare templates

float comparisonA = distance->compare(target, queryA);

float comparisonB = distance->compare(target, queryB);

// Scores range from 0 to 1 and represent match probability

printf("Genuine match score: %.3f\n", comparisonA);

printf("Impostor match score: %.3f\n", comparisonB);

br::Context::finalize();

return 0;

}

# ПРИЛОЖЕНИЕ Б

## (обязательное)

Пример программы, созданной с помощью Megamatcher SDK на C#

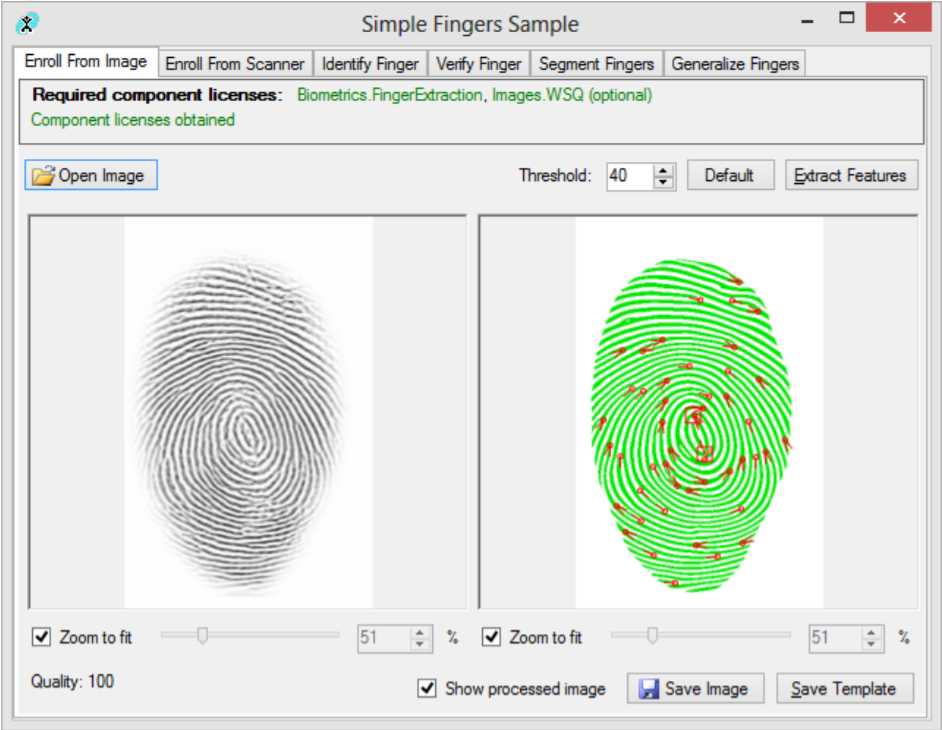


Рисунок - Программа для анализа отпечатка пальца

# ПРИЛОЖЕНИЕ В

## (обязательное)

Пример кода внедрения биометрии лица Huawei FIDO BioAuthn на языке Java

Листинг примера:

private BioAuthnPrompt createBioAuthnPrompt() {

// Callback.

BioAuthnCallback callback = new BioAuthnCallback() {

@Override

public void onAuthError(int errMsgId, CharSequence errString) {

showResult("Authentication error. errorCode=" + errMsgId + ",errorMessage=" + errString);

}

@Override

public void onAuthSucceeded(BioAuthnResult result) {

showResult("Authentication succeeded. CryptoObject=" + result.getCryptoObject());

}

@Override

public void onAuthFailed() {

showResult("Authentication failed.");

}

};

return new BioAuthnPrompt(this, ContextCompat.getMainExecutor(this), callback);

}

// Build prompt information.

BioAuthnPrompt.PromptInfo.Builder builder =

new BioAuthnPrompt.PromptInfo.Builder().setTitle("This is the title.")

.setSubtitle("This is the subtitle")

.setDescription("This is the description");

// The user will first be prompted to authenticate with biometrics, but also given the option to

// authenticate with their device PIN, pattern, or password. setNegativeButtonText(CharSequence) should

// not be set if this is set to true.

builder.setDeviceCredentialAllowed(true);

// Set the text for the negative button. setDeviceCredentialAllowed(true) should not be set if this button text

// is set.

// builder.setNegativeButtonText ("This is the Cancel button.");

BioAuthnPrompt.PromptInfo info = builder.build();

resultTextView.setText("Start fingerprint authentication without CryptoObject.\nAuthenticating......\n");

bioAuthnPrompt.auth(info);

// Callback.

BioAuthnCallback callback = new BioAuthnCallback() {

@Override

public void onAuthError(int errMsgId, CharSequence errString) {

showResult("Authentication error. errorCode=" + errMsgId + ",errorMessage=" + errString

+ (errMsgId == 1012 ? " The camera permission may not be enabled." : ""));

}

@Override

public void onAuthHelp(int helpMsgId, CharSequence helpString) {

resultTextView

.append("Authentication help. helpMsgId=" + helpMsgId + ",helpString=" + helpString + "\n");

}

@Override

public void onAuthSucceeded(BioAuthnResult result) {

showResult("Authentication succeeded. CryptoObject=" + result.getCryptoObject());

}

@Override

public void onAuthFailed() {

showResult("Authentication failed.");

}

};

// Cancellation signal.

CancellationSignal cancellationSignal = new CancellationSignal();

FaceManager faceManager = new FaceManager(this);

// Flags.

int flags = 0;

// Authentication message handler.

Handler handler = null;

// You are advised to set CryptoObject to null. The KeyStore is not associated with face authentication in the current

// version. KeyGenParameterSpec.Builder.setUserAuthenticationRequired() must be set to false in this scenario.

CryptoObject crypto = null;

faceManager.auth(crypto, cancellationSignal, flags, callback, handler);

# ПРИЛОЖЕНИЕ Г

## (обязательное)

Пример кода внедрения биометрии Apple Local Authentication на языке Swift

Листинг примера:

/\*

See LICENSE folder for this sample’s licensing information.

Abstract:

Login view controller.

\*/

import UIKit

import LocalAuthentication

class ViewController: UIViewController {

@IBOutlet weak var loginButton: UIButton!

@IBOutlet weak var stateView: UIView!

@IBOutlet weak var faceIDLabel: UILabel!

/// An authentication context stored at class scope so it's available for use during UI updates.

var context = LAContext()

/// The available states of being logged in or not.

enum AuthenticationState {

case loggedin, loggedout

}

/// The current authentication state.

var state = AuthenticationState.loggedout {

// Update the UI on a change.

didSet {

loginButton.isHighlighted = state == .loggedin // The button text changes on highlight.

stateView.backgroundColor = state == .loggedin ? .green : .red

// FaceID runs right away on evaluation, so you might want to warn the user.

// In this app, show a special Face ID prompt if the user is logged out, but

// only if the device supports that kind of authentication.

faceIDLabel.isHidden = (state == .loggedin) || (context.biometryType != .faceID)

}

}

override func viewDidLoad() {

super.viewDidLoad()

// The biometryType, which affects this app's UI when state changes, is only meaningful

// after running canEvaluatePolicy. But make sure not to run this test from inside a

// policy evaluation callback (for example, don't put next line in the state's didSet

// method, which is triggered as a result of the state change made in the callback),

// because that might result in deadlock.

context.canEvaluatePolicy(.deviceOwnerAuthentication, error: nil)

// Set the initial app state. This impacts the initial state of the UI as well.

state = .loggedout

}

/// Logs out or attempts to log in when the user taps the button.

@IBAction func tapButton(\_ sender: UIButton) {

if state == .loggedin {

// Log out immediately.

state = .loggedout

} else {

// Get a fresh context for each login. If you use the same context on multiple attempts

// (by commenting out the next line), then a previously successful authentication

// causes the next policy evaluation to succeed without testing biometry again.

// That's usually not what you want.

context = LAContext()

context.localizedCancelTitle = "Enter Username/Password"

// First check if we have the needed hardware support.

var error: NSError?

if context.canEvaluatePolicy(.deviceOwnerAuthentication, error: &error) {

let reason = "Log in to your account"

context.evaluatePolicy(.deviceOwnerAuthentication, localizedReason: reason ) { success, error in

if success {

// Move to the main thread because a state update triggers UI changes.

DispatchQueue.main.async { [unowned self] in

self.state = .loggedin

}

} else {

print(error?.localizedDescription ?? "Failed to authenticate")

// Fall back to a asking for username and password.

// ...

}

}

} else {

print(error?.localizedDescription ?? "Can't evaluate policy")

// Fall back to a asking for username and password.

// ...

}

}

}

}

# ПРИЛОЖЕНИЕ Д

## (обязательное)

Пример кода внедрения биометрии отпечатка пальца Android Biometric authentication на языке Java

Листинг примера:

/\*

\* Copyright (C) 2019 The Android Open Source Project

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License

\*/

package com.example.android.biometricauth

import android.content.Intent

import android.content.SharedPreferences

import android.os.Bundle

import android.preference.PreferenceManager

import android.security.keystore.KeyGenParameterSpec

import android.security.keystore.KeyPermanentlyInvalidatedException

import android.security.keystore.KeyProperties

import android.security.keystore.KeyProperties.BLOCK\_MODE\_CBC

import android.security.keystore.KeyProperties.ENCRYPTION\_PADDING\_PKCS7

import android.security.keystore.KeyProperties.KEY\_ALGORITHM\_AES

import android.util.Base64

import android.util.Log

import android.view.Menu

import android.view.MenuItem

import android.view.View

import android.widget.Button

import android.widget.TextView

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

import androidx.biometric.BiometricManager

import androidx.biometric.BiometricPrompt

import androidx.core.content.ContextCompat

import java.io.IOException

import java.security.InvalidAlgorithmParameterException

import java.security.InvalidKeyException

import java.security.KeyStore

import java.security.KeyStoreException

import java.security.NoSuchAlgorithmException

import java.security.NoSuchProviderException

import java.security.UnrecoverableKeyException

import java.security.cert.CertificateException

import javax.crypto.BadPaddingException

import javax.crypto.Cipher

import javax.crypto.IllegalBlockSizeException

import javax.crypto.KeyGenerator

import javax.crypto.NoSuchPaddingException

import javax.crypto.SecretKey

class MainActivity : AppCompatActivity(),

FingerprintAuthenticationDialogFragment.Callback {

private lateinit var keyStore: KeyStore

private lateinit var keyGenerator: KeyGenerator

private lateinit var sharedPreferences: SharedPreferences

private lateinit var biometricPrompt: BiometricPrompt

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

setSupportActionBar(findViewById(R.id.toolbar))

setupKeyStoreAndKeyGenerator()

val (defaultCipher: Cipher, cipherNotInvalidated: Cipher) = setupCiphers()

sharedPreferences = PreferenceManager.getDefaultSharedPreferences(this)

biometricPrompt = createBiometricPrompt()

setUpPurchaseButtons(cipherNotInvalidated, defaultCipher)

}

private fun setUpPurchaseButtons(cipherNotInvalidated: Cipher, defaultCipher: Cipher) {

val purchaseButton = findViewById<Button>(R.id.purchase\_button)

val purchaseButtonNotInvalidated =

findViewById<Button>(R.id.purchase\_button\_not\_invalidated)

if (BiometricManager.from(

application).canAuthenticate() == BiometricManager.BIOMETRIC\_SUCCESS) {

createKey(DEFAULT\_KEY\_NAME)

createKey(KEY\_NAME\_NOT\_INVALIDATED, false)

purchaseButtonNotInvalidated.run {

isEnabled = true

setOnClickListener(PurchaseButtonClickListener(

cipherNotInvalidated, KEY\_NAME\_NOT\_INVALIDATED))

}

purchaseButton.run {

isEnabled = true

setOnClickListener(PurchaseButtonClickListener(defaultCipher, DEFAULT\_KEY\_NAME))

}

} else {

showToast(getString(R.string.setup\_lock\_screen))

purchaseButton.isEnabled = false

purchaseButtonNotInvalidated.isEnabled = false

}

}

private fun setupKeyStoreAndKeyGenerator() {

try {

keyStore = KeyStore.getInstance(ANDROID\_KEY\_STORE)

} catch (e: KeyStoreException) {

throw RuntimeException("Failed to get an instance of KeyStore", e)

}

try {

keyGenerator = KeyGenerator.getInstance(KEY\_ALGORITHM\_AES, ANDROID\_KEY\_STORE)

} catch (e: Exception) {

when (e) {

is NoSuchAlgorithmException,

is NoSuchProviderException ->

throw RuntimeException("Failed to get an instance of KeyGenerator", e)

else -> throw e

}

}

}

private fun setupCiphers(): Pair<Cipher, Cipher> {

val defaultCipher: Cipher

val cipherNotInvalidated: Cipher

try {

val cipherString = "$KEY\_ALGORITHM\_AES/$BLOCK\_MODE\_CBC/$ENCRYPTION\_PADDING\_PKCS7"

defaultCipher = Cipher.getInstance(cipherString)

cipherNotInvalidated = Cipher.getInstance(cipherString)

} catch (e: Exception) {

when (e) {

is NoSuchAlgorithmException,

is NoSuchPaddingException ->

throw RuntimeException("Failed to get an instance of Cipher", e)

else -> throw e

}

}

return Pair(defaultCipher, cipherNotInvalidated)

}

private fun initCipher(cipher: Cipher, keyName: String): Boolean {

try {

keyStore.load(null)

cipher.init(Cipher.ENCRYPT\_MODE, keyStore.getKey(keyName, null) as SecretKey)

return true

} catch (e: Exception) {

when (e) {

is KeyPermanentlyInvalidatedException -> return false

is KeyStoreException,

is CertificateException,

is UnrecoverableKeyException,

is IOException,

is NoSuchAlgorithmException,

is InvalidKeyException -> throw RuntimeException("Failed to init Cipher", e)

else -> throw e

}

}

}

override fun onPurchased(withBiometrics: Boolean, crypto: BiometricPrompt.CryptoObject?) {

if (withBiometrics) {

// If the user authenticated with fingerprint, verify using cryptography and then show

// the confirmation message.

crypto?.cipher?.let { tryEncrypt(it) }

} else {

// Authentication happened with backup password. Just show the confirmation message.

showConfirmation()

}

}

// Show confirmation message. Also show crypto information if fingerprint was used.

private fun showConfirmation(encrypted: ByteArray? = null) {

findViewById<View>(R.id.confirmation\_message).visibility = View.VISIBLE

if (encrypted != null) {

findViewById<TextView>(R.id.encrypted\_message).run {

visibility = View.VISIBLE

text = Base64.encodeToString(encrypted, 0 /\* flags \*/)

}

}

}

private fun tryEncrypt(cipher: Cipher) {

try {

showConfirmation(cipher.doFinal(SECRET\_MESSAGE.toByteArray()))

} catch (e: Exception) {

when (e) {

is BadPaddingException,

is IllegalBlockSizeException -> {

Toast.makeText(this, "Failed to encrypt the data with the generated key. "

+ "Retry the purchase", Toast.LENGTH\_LONG).show()

Log.e(TAG, "Failed to encrypt the data with the generated key. ${e.message}")

}

else -> throw e

}

}

}

override fun createKey(keyName: String, invalidatedByBiometricEnrollment: Boolean) {

try {

keyStore.load(null)

val keyProperties = KeyProperties.PURPOSE\_ENCRYPT or KeyProperties.PURPOSE\_DECRYPT

val builder = KeyGenParameterSpec.Builder(keyName, keyProperties)

.setBlockModes(BLOCK\_MODE\_CBC)

.setUserAuthenticationRequired(true)

.setEncryptionPaddings(ENCRYPTION\_PADDING\_PKCS7)

.setInvalidatedByBiometricEnrollment(invalidatedByBiometricEnrollment)

keyGenerator.run {

init(builder.build())

generateKey()

}

} catch (e: Exception) {

when (e) {

is NoSuchAlgorithmException,

is InvalidAlgorithmParameterException,

is CertificateException,

is IOException -> throw RuntimeException(e)

else -> throw e

}

}

}

override fun onCreateOptionsMenu(menu: Menu): Boolean {

menuInflater.inflate(R.menu.menu\_main, menu)

return true

}

override fun onOptionsItemSelected(item: MenuItem): Boolean {

if (item.itemId == R.id.action\_settings) {

val intent = Intent(this, SettingsActivity::class.java)

startActivity(intent)

return true

}

return super.onOptionsItemSelected(item)

}

private fun createBiometricPrompt(): BiometricPrompt {

val executor = ContextCompat.getMainExecutor(this)

val callback = object : BiometricPrompt.AuthenticationCallback() {

override fun onAuthenticationError(errorCode: Int, errString: CharSequence) {

super.onAuthenticationError(errorCode, errString)

Log.d(TAG, "$errorCode :: $errString")

if (errorCode == BiometricPrompt.ERROR\_NEGATIVE\_BUTTON) {

loginWithPassword() // Because negative button says use application password

}

}

override fun onAuthenticationFailed() {

super.onAuthenticationFailed()

Log.d(TAG, "Authentication failed for an unknown reason")

}

override fun onAuthenticationSucceeded(result: BiometricPrompt.AuthenticationResult) {

super.onAuthenticationSucceeded(result)

Log.d(TAG, "Authentication was successful")

onPurchased(true, result.cryptoObject)

}

}

val biometricPrompt = BiometricPrompt(this, executor, callback)

return biometricPrompt

}

private fun createPromptInfo(): BiometricPrompt.PromptInfo {

val promptInfo = BiometricPrompt.PromptInfo.Builder()

.setTitle(getString(R.string.prompt\_info\_title))

.setSubtitle(getString(R.string.prompt\_info\_subtitle))

.setDescription(getString(R.string.prompt\_info\_description))

.setConfirmationRequired(false)

.setNegativeButtonText(getString(R.string.prompt\_info\_use\_app\_password))

// .setDeviceCredentialAllowed(true) // Allow PIN/pattern/password authentication.

// Also note that setDeviceCredentialAllowed and setNegativeButtonText are

// incompatible so that if you uncomment one you must comment out the other

.build()

return promptInfo

}

private fun loginWithPassword() {

Log.d(TAG, "Use app password")

val fragment = FingerprintAuthenticationDialogFragment()

fragment.setCallback(this@MainActivity)

fragment.show(fragmentManager, DIALOG\_FRAGMENT\_TAG)

}

private inner class PurchaseButtonClickListener internal constructor(

internal var cipher: Cipher,

internal var keyName: String

) : View.OnClickListener {

override fun onClick(view: View) {

findViewById<View>(R.id.confirmation\_message).visibility = View.GONE

findViewById<View>(R.id.encrypted\_message).visibility = View.GONE

val promptInfo = createPromptInfo()

if (initCipher(cipher, keyName)) {

biometricPrompt.authenticate(promptInfo, BiometricPrompt.CryptoObject(cipher))

} else {

loginWithPassword()

}

}

}

companion object {

private const val ANDROID\_KEY\_STORE = "AndroidKeyStore"

private const val DIALOG\_FRAGMENT\_TAG = "myFragment"

private const val KEY\_NAME\_NOT\_INVALIDATED = "key\_not\_invalidated"

private const val SECRET\_MESSAGE = "Very secret message"

private const val TAG = "MainActivity"

}

}

# ПРИЛОЖЕНИЕ Е

## (обязательное)

Пример кода внедрения биометрии Windows Biometric Framework на языке C#

Листинг примера:

HRESULT CaptureSample()

{

HRESULT hr = S\_OK;

WINBIO\_SESSION\_HANDLE sessionHandle = NULL;

WINBIO\_UNIT\_ID unitId = 0;

WINBIO\_REJECT\_DETAIL rejectDetail = 0;

PWINBIO\_BIR sample = NULL;

SIZE\_T sampleSize = 0;

// Connect to the system pool.

hr = WinBioOpenSession(

WINBIO\_TYPE\_FINGERPRINT, // Service provider

WINBIO\_POOL\_SYSTEM, // Pool type

WINBIO\_FLAG\_RAW, // Access: Capture raw data

NULL, // Array of biometric unit IDs

0, // Count of biometric unit IDs

WINBIO\_DB\_DEFAULT, // Default database

&sessionHandle // [out] Session handle

);

if (FAILED(hr))

{

wprintf\_s(L"\n WinBioOpenSession failed. hr = 0x%x\n", hr);

goto e\_Exit;

}

// Capture a biometric sample.

wprintf\_s(L"\n Calling WinBioCaptureSample - Swipe sensor...\n");

hr = WinBioCaptureSample(

sessionHandle,

WINBIO\_NO\_PURPOSE\_AVAILABLE,

WINBIO\_DATA\_FLAG\_RAW,

&unitId,

&sample,

&sampleSize,

&rejectDetail

);

if (FAILED(hr))

{

if (hr == WINBIO\_E\_BAD\_CAPTURE)

{

wprintf\_s(L"\n Bad capture; reason: %d\n", rejectDetail);

}

else

{

wprintf\_s(L"\n WinBioCaptureSample failed. hr = 0x%x\n", hr);

}

goto e\_Exit;

}

wprintf\_s(L"\n Swipe processed - Unit ID: %d\n", unitId);

wprintf\_s(L"\n Captured %d bytes.\n", sampleSize);

e\_Exit:

if (sample != NULL)

{

WinBioFree(sample);

sample = NULL;

}

if (sessionHandle != NULL)

{

WinBioCloseSession(sessionHandle);

sessionHandle = NULL;

}

wprintf\_s(L"\n Press any key to exit...");

\_getch();

return hr;

}