ИНФОРМАТИКА, КИБЕРНЕТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

УДК 004.056.53, 004.422.832

Одна из возможных реализаций модели менеджера паролей для ОС Андроид

А. В. Черников
Пермский государственный национальный исследовательский университет; Пермь, Россия
e-mail: arsenyperm@mail.ru; **SPIN-код (РИНЦ): 6097-1133, ORCID: 0000-0001-9430-3587**

Рассматривается одна из возможных реализаций модели менеджера паролей для операционной системы (далее ОС) Андроид. Работа является продолжением статьи по разработке программного менеджера паролей для ОС Андроид [13], изданной ранее. В данной работе производится выбор конкретных методов, алгоритмов, на основе которых необходимо реализовывать программное обеспечение, предложена модель приложения с учетом результатов, полученных в предыдущей работе [13], а также приведены рекомендации по защите информации при работе приложения для ОС Андроид.

Ключевые слова: *информационная безопасность; менеджер паролей; операционная система Андроид*

Поступила в редакцию 27.01.2022, принята к опубликованию 14.02.2022

One of the possible implementations manager's password model for Android OS

A. V. Chernikov
Perm State University; Perm, Russia
e-mail: arsenyperm@mail.ru; **SPIN-код (РИНЦ): 6097-1133, ORCID: 0000-0001-9430-3587**

The paper considers one of the possible implementations of the password manager model for the Android operating system (hereinafter OS). The work is a continuation of the article on the development of a software password manager for the Android OS [13], published earlier. In this paper, the choice of specific methods, algorithms, on the basis of which it is necessary to implement the software, an application model is proposed, taking into account the results obtained in the previous work [13], and recommendations are given for protecting information when running an application for the Android OS.

Keywords: *information security; password manager; operating system Android*

Received 27.01.2022, accepted 14.02.2022

DOI: 10.17072/1993-0550-2022-1-38-47

Введение

В предыдущей работе [13], которая была посвящена анализу предметной области, а именно: анализу наиболее популярных в использовании программных/программно-аппаратных менеджеров паролей, которые существуют для ОС Андроид, были выявлены основные положительные и отрицательные стороны представленных средств.

В конце работы были сделаны выводы о том, какие проблемные места стоит исключить в дальнейших реализациях менеджеров паролей, какие положительные стороны существующих менеджеров паролей необходимо включать в дальнейшие реализации.

В результате проблематикой данной работы становится:

- выбор конкретных методов, алгоритмов, способов защиты информации, которые необходимо реализовать в разрабатываемом менеджере паролей;
- разработка модели работы защищенного менеджера паролей, которая учитывала бы необходимые рекомендации [13];
- предложения по применению в системе необходимых конкретных средств защиты информации менеджера паролей для ОС Андроид.

1. Модель программной реализации менеджера паролей

Для начала стоит сделать ряд уточнений, что менеджер паролей будет представлять собой только программный продукт и состоять из совокупности нескольких программно-аппаратных компонентов.

На рис. 1 изображена предполагаемая модель будущего программного комплекса для хранения паролей и управления им.

Комплекс представлен в идеальном варианте, с учетом всех возможных в работе приложения устройств, как под управлением ОС Андроид, так и, например, под управлением ОС Windows (ПК для управления).

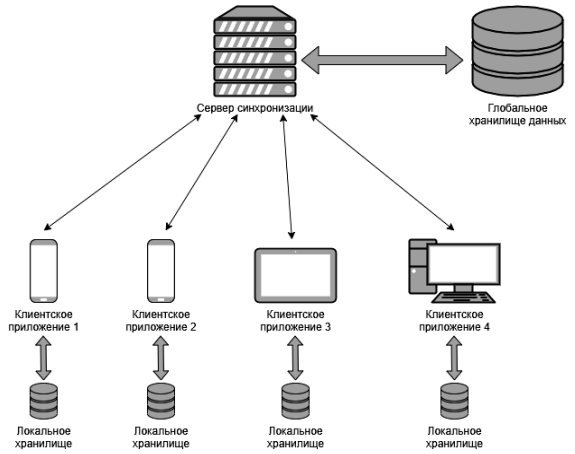

Рис. 1. *Общая модель системы*

Программный комплекс состоит из следующих компонентов:

- сервер синхронизации, который обеспечивает возможность работы пользователя с менеджером паролей на нескольких устройствах, предоставляя сервисы аутентификации пользователей в системе и синхронизации данных и является центральной частью системы.
- глобальное хранилище данных, которое предназначено для хранения данных, необходимых для работы сервера в зашифрованном виде.
- клиентское приложение, которое является основным компонентом системы, осуществляющим взаимодействие с пользователем. Пользователь сможет одновременно применять несколько клиентских приложений на различных устройствах.
- локальное хранилище данных, которое представляет собой структурированный набор записей об учетных данных пользователя на каком-либо электронном ресурсе.

Взаимодействие между клиентскими приложениями и сервером синхронизации осуществляется по средствам HTTP-запросов [11], что является классическим методом взаимодействия на сегодня в распределенных системах.

Представленная схема архитектуры системы подразумевает возможность масштабируемости, которая позволит в дальнейшем разработчикам создавать клиентские приложения для других операционных систем, а пользователям – не быть привязанными к ОС Андроид и работать с менеджером паролей со своих различных устройств.

В рамках данной работы будут рассмотрены только вопросы проектирования и программной реализации: сервера синхронизации, хранилищ данных и клиентского приложения для ОС Андроид.

2. Система хранения данных

Рассмотрение модели системы начнем с системы глобального и локального хранения данных.

2.1. Структура глобальной базы данных

Для реализации требований, описанных в статье [13] глобальная база данных должна состоять из пяти таблиц.

Схема глобальной базы данных представлена на рис. 2.

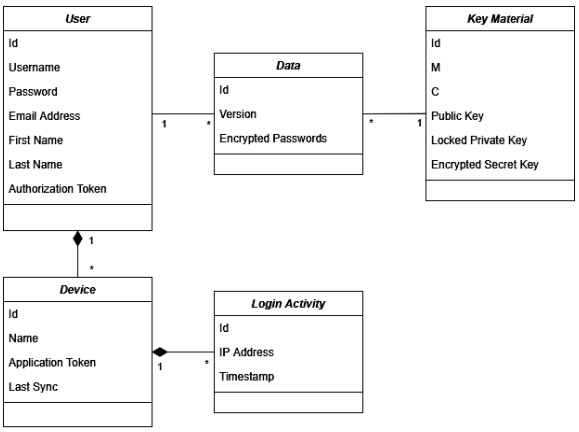

Рис. 2. *Схема глобальной базы данных*

В таблице **User** хранится информация о зарегистрированном в системе пользователе/пользователях: их логины, пароль в зашифрованном виде, адрес электронной почты, имя и фамилия. Также в таблице хранится билет (токен) авторизации пользователя, который имеет ограниченное время жизни. Время, в течение которого билет считается рабочим (активным), ограничивается периодом действия билета и отслеживается на уровне сервера синхронизации.

Так как пользователь может использовать приложения на разных устройствах, то серверу синхронизации необходимо знать, с какого устройства и когда был выполнен вход, а также фиксировать время последней синхронизации. Эта информация хранится в двух таблицах: **Device** и **LoginActivity**.

Таблицы **Data** и **KeyMaterial** предназначены для хранения данных пользователя.

В таблице **Data** хранится зашифрованный файл локальной базы данных (либо путь до файла, если он хранится в файловой системе сервера) и ее версия.

Таблица **KeyMaterial** предназначена для сбора ключевой информации по всей базе данных в одной таблице: хранит зашифрованные ключи (за исключением открытого ключа) и строки **M** и **C** для проверки аутентичности мастер-пароля.

Их применение более подробно будет рассмотрено далее.

2.2. Структура локальной базы данных

Локальная база данных содержит три основных таблицы.

Схема локальной базы данных представлена на рис. 3.

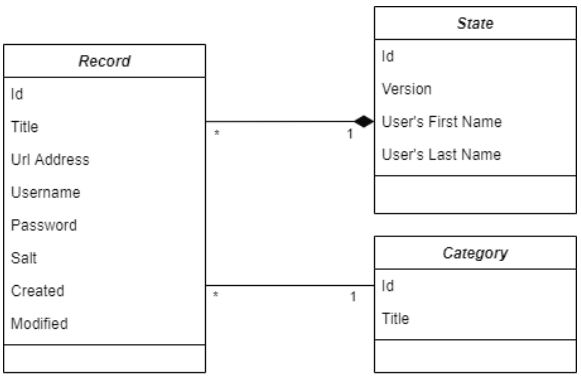

Рис. 3. *Схема локальной базы данных*

Структура локальной базы данных: все записи пользователя, с которыми он осуществляет работу, хранятся в таблице **Record**. Данная таблица состоит из таких полей, как название ресурса (**title**), URL-адрес ресурса (**urlAddress**), имя пользователя (**username**), зашифрованный пароль (**password**) и временные отметки о создании записи (**created**) и о последнем изменении (**modified**).

В отдельную таблицу выделена сущность **Category**. Эта таблица содержит список категорий записей, добавленных пользователем. Такой подход можно обосновать удобством группировки записей по категориям при отображении, а также невозможностью удаления категории, в которой еще существуют записи, на уровне СУБД.

Также в отдельную таблицу **State** была выделена служебная информация. На данном этапе разработки приложения в данную таблицу попадает такая информацию, как версия используемой базы данных, имя и фамилия пользователя. В дальнейшем данная таблица может быть использована для хранения любой информации, необходимой для расширения функционала приложения.

3. Сервер синхронизации и клиентское приложение для ОС Андроид

Далее будет рассмотрена модель системы на уровне сервера синхронизации и клиентского приложения как неотделимых друг от друга частей системы, так как требования, применяемые к одной части программного обеспечения, автоматически должны быть применены и к другой.

3.1. Защита данных

В приложении (клиентской части) обрабатываются и хранятся данные, представляющие ценность для пользователя. Получение доступа к ним третьим лицам может повлечь нежелательные последствия для пользователя. Поэтому при проектировании приложения проработаны вопросы, касающиеся обеспечения конфиденциальности и целостности обрабатываемых данных (согласно предъявленным критериям к приложению [13]).

Для этого должны быть реализованы следующие механизмы, описанные ниже.

3.2. Шифрование пользовательских данных

Данные пользователя в локальной базе данных должны храниться на сервере в зашифрованном виде, причем сервер никогда не должен получать эти данные в открытом виде. Поэтому при синхронизации, перед отправкой на сервер, файл базы данных шифруется в клиентском приложении. Для шифрования данных необходимо применять алгоритмы симметричного шифрования (например, AES[1] или ГОСТ 34.12[7]), поскольку по сравнению с ассиметричными криптографическими преобразованиями они отличаются скоростью и отлично подходят для шифрования файлов.

На устройстве пользователя база данных хранится в открытом виде. Если на сервере есть более свежая версия данных, то приложение получает новый файл с сервера, расшифровывает его и продолжается работа с новой базой данных. Несмотря на то, что файл базы данных хранится в папке приложения и доступен только ему, существует риск утечки паролей пользователя, поэтому поля паролей в базе данных дополнительно должны шифроваться. Они будут расшифровываться только по запросу пользователя при просмотре записи или копировании пароля в буфер обмена.

Может существовать вероятность того, что пользователь будет использовать одинаковые пароли на нескольких электронных ресурсах. Поэтому необходимо избежать подобной ситуации, когда по шифрограммам паролей можно понять, на каких электронных ресурсах пользователь использовал одни и те же авторизационные данные.

Эту проблему возможно решить добавлением к паролю случайных последовательностей – некоторой строки, генерируемой случайным образом, и хранить ее с паролем.

В рекомендациях по разработке менеджера паролей [13], было отмечено, что помимо паролей необходимо шифровать и все метаданные, чтобы избежать компрометации пользователя по интересам, убеждениям и т. п. Но в конечной реализации на уровне локальной базы данных это будет избыточно, так как вне устройства пользователя эти данные передаются и хранятся только в зашифрованном виде.

3.3. Защита данных при передаче между устройствами

В клиент-серверной среде для защиты информационного взаимодействия необходимо решить ряд задач:

- подключаясь к серверу синхронизации, пользователь должен удостовериться, что обмен данными происходит с сервером, с которым должен происходить;
- после установки соединения между сервером и клиентом данные должны быть защищены от получения третьими лицами;
- в процессе обмена информацией сервер и клиент должны быть уверены, что информация, приходящая к ним друг от друга, не искажена при передаче.

Для решения поставленных задач необходимо использовать существующие методы защиты информации: протоколы безопасности SSL [3, 4] или TLS [6], предназначенные для решения поставленных выше задач.

Поскольку современные веб-серверы имеют встроенную поддержку обоих протоколов, то будем использовать TLS как наиболее актуальный и безопасный [5]. Более того, веб-сервера различных разработчиков поддерживают расширение протокола HTTP – HTTPS, данные в котором передаются поверх протокола TLS.

Для обработки HTTPS-соединений между сервером и клиентом, необходимо установить в систему сертификат открытого и закрытого ключа для веб-сервера.

Данный сертификат необходимо получать в специализированном Центре сертификации. Поэтому этот момент крайне важно учесть при написании готовой реализации программного менеджера паролей.

3.4. Управление ключами

Без грамотного построения системы управления ключами ставятся под вопрос надежность приложения, так как администрирование ключей шифрования в системе является важнейшим вопросом с точки зрения шифрования со стороны обеспечения конфиденциальности обмена информацией, идентификации и целостности данных [14]. Основной целью администрирования ключей становится исключение следующих угроз:

- раскрытие закрытых и открытых ключей шифрования;
- использование ключей, срок действия которых истек.

Процесс администрирования ключей, в результате, состоит из задач, обеспечивающих:

- аутентификации пользователя в системе;
- создание и распределение закрытых и открытых ключей шифрования;
- контроль использования ключей в системе;
- замену устаревших ключей и их уничтожение внутри системы;
- сжатие, хранение и восстановление ключей шифрования [14].

В модели приложения указанные выше процедуры необходимы в реализации, однако данный вопрос нет необходимости выносить отдельно, так как для решения этих задач можно использовать готовые решения по реализации протокола безопасности, предоставляемые средствами разработки.

3.5. Генерация ключей

Согласно принципу Кирхгофа, стойкость криптографического алгоритма определяется секретностью ключа [12]. Это говорит о том, что если в качестве функции для генерации ключа используется алгоритм, неустойчивый к атакам, то, независимо от используемого метода шифрования, система будет нестойкой к атакам аналогично алгоритму. Криптостойкий ключ, предназначенный для работы в системе симметричного шифрования, представляет собой случайную бинарную последовательность.

Тогда для генерации криптостойких ключей может использоваться ряд методов (в порядке возрастания качества) [8]:

- программная реализация, представляющая вычисление последовательности псевдослучайных чисел как функцию от введенной информации пользователя, плюс текущего времени и возможной вспомогательной информации, такой как, например, серийных номеров аппаратной/программной составляющей системы, особенность клавиатурного почерка и т. п. информации;
- программная реализация, основанная на работе программного ГСПЧ (генератора псевдослучайных чисел), основанного на различных алгоритмах генерации;
- аппаратная реализация, основанная на аппаратных генераторах псевдослучайных чисел, построенных на алгоритмах с равномерным законом распределения;
- аппаратная реализация, основанная на аппаратных генераторах случайных чисел, основанных на физических явлениях, например уровень шума в атмосфере, скачки напряжения в сети питания и т.п. явлениях, имеющих случайных характер [9].

Так как в разрабатываемом приложении не предъявляется высоких требований к защищенности информации, то для генерации ключей шифрования можно использовать программные генераторы ключей, которые вычисляют случайные числа как сложную функцию от текущего времени или числа, введенного пользователем. Однако в идеальном случае (в коммерческой реализации, например) необходимо использовать встроенные генераторы случайных чисел ОС Андроид, у которых документально подтверждена криптографическая стойкость.

3.6. Хранение ключей

Секретные ключи не должны храниться в памяти устройств в открытом, не зашифрованном виде. Одним из возможных в использовании инструментов, который может обеспечить конфиденциальность ключей, является разделение ключей по уровням. Необходимо использовать двух- или трехуровневую иерархию. Так как в разрабатываемой системе в явном виде используется мастер-пароль, то выбрана трехуровневая модель разделения ключей, которую можно описать следующим образом:

- На верхнем уровне расположен так называемый мастер-ключ (главного), который необходимо защищать не криптографическими программными методами, а физическими.

В роли главного ключа в системе используется мастер-пароль (**Master Key**), который является известным только пользователю и ни в каком виде не должен храниться в системе. Физическая защита мастер-пароля остается за пользователем (например, не записывать его на бумажный носитель, не передавать его посторонним лицам и т. п.). Задача разрабатываемой системы – не допустить компрометацию пароля (хищение вредоносными программами) при вводе в клиентском приложении, что опять же частично решается средствами ОС Андроид (виртуализацией задач, отдельные адресные пространства для каждого запущенного ПО и т.д.) и пользователем (установка антивирусных программ, ответственный подход к работе в сети Интернет и т.д.).

Стоит отметить, что мастер-пароль не является криптографическим ключом в классическом понимании [10]. В схеме шифрования ключей используется его отображение (**Unlock Key**), полученное путем применения хеш-функции. Поскольку между **Master Key** и **Unlock Key** существует прямая зависимость, то необходимо защищать именно мастер-пароль, а не производный от него ключ.

• На втором уровне иерархии – ключи для шифрования – ключи, используемые для шифрования данных перед передачей или при хранении других ключей шифрования. В роли таких ключей в системе выступает пара из открытого и закрытого ключа (**Public Key** и **Private Key**). Они предназначены для шифрования и дешифрования ключа **Secret Key**. В свою очередь **Private Key** защищен с помощью **Unlock Key**, используемого в качестве парольной фразы при генерации пары ассиметричных ключей и получения доступа к закрытому ключу по стандарту PKCS#8 [2].

• На третьем уровне иерархии расположены ключи для шифрования данных, которые предназначены для защиты данных пользователей. В разрабатываемом приложении для шифрования данных пользователя (локальной БД) используется ключ симметричного шифрования **Secret Key**. **Secret Key** хранится в системе в зашифрованном виде.

Таким образом, ключи более высоких уровней используются для шифрования ключей и данных на следующем низком уровне. Этот подход снижает возможность вскрытия ключей шифрования и данных на различных уровнях, и при этом уменьшает размер информации, нуждающейся в физической защите.

Описанные выше методы можно представить в виде следующих схем шифрования и дешифрования ключей в системе (рис. 4 и 5).

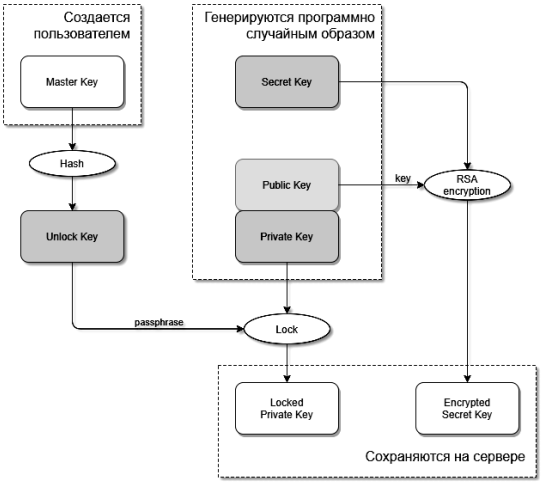

Рис. 4. *Схема шифрования ключей*

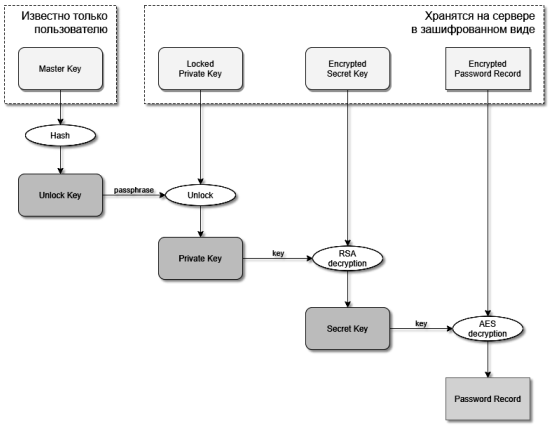

Рис. 5. *Схема дешифрования ключей*

3.7. Время жизни ключей шифрования

Один из важных моментов, который необходимо рассмотреть – это "время жизни" ключей шифрования. Ключи шифрования должны использоваться в течение ограниченного промежутка времени, продолжительность которого зависит от частоты использования ключа и величины нанесенного ущерба при компрометации ключа.

Для разрабатываемого приложения время жизни ключей можно интерпретировать как период времени **P**, на протяжении которого ключи доступны в расшифрованном виде.

Так, при вводе пользователем мастер-пароля и успешной его аутентификации приложение расшифровывает ключи и фиксирует метку времени. По истечении периода **P** доступ к ключам либо блокируется средствами операционной системы (если существуют такие возможности), либо их расшифрованные значения удаляются из памяти программно.

Стоит отметить, что в течение установленного промежутка времени **P** приложение может использовать альтернативные способы аутентификации (например, графический ключ или проверку биометрических данных), чтобы упростить вход пользователя в приложение при частом обращении к нему. Однако полностью исключить аутентификацию по мастер-паролю невозможно, так как, чем реже пользователь вводит мастер-пароль, тем быстрее он может его забыть. Именно поэтому "временем жизни" **P** мастер-пароля в приложении можно определить период в 24 часа.

Таким образом, ввод мастер-пароля один раз в сутки не даст пользователю забыть его, а альтернативные, более быстрые способы аутентификации, сделают пользование приложением более комфортным.

3.8. Замена ключей

Среди всех ключей, используемых в приложении, легче всего может быть скомпрометирован мастер-пароль. Несмотря на требования безопасности использования паролей, пользователь может сообщить его третьему лицу, записать его на бумажном носителе информации или же злоумышленник может подсмотреть мастер-пароль при вводе пользователем символов на клавиатуре устройства.

Учитывая эти факторы, в приложении предусмотрена возможность смены мастер-пароля. В соответствии со схемой шифрования ключей данная операция будет влиять на значение **Unlock Key**, являющегося парольной фразой для **Private Key**.

Поэтому при смене мастер-пароля будет установлена блокировка закрытого ключа с использованием новой парольной фразы. При этом не произойдет смены ключа шифрования **Secret Key** и не будет необходимости заново производить шифрование базы данных вместе с паролями. Остальные ключи надежно защищены, поэтому их периодическая замена не требуется.

3.9. Аутентификация мастер-пароля

Как было отмечено ранее, мастер-пароль и его хэш-функция не хранятся в системе. Поэтому, при очередном вводе пользователем мастер-пароля, у системы нет возможности проверить его корректность до тех пор, пока не будет расшифрована база данных с помощью ключа **Secret Key**, в результате чего может быть получен ″нечитаемый″ файл с данными с сервера. Подобная ситуация недопустима в системе.

Поэтому, для того чтобы исключить доступ к пользовательским данным при проведении процесса аутентификации мастер-пароля, необходимо применить другой способ, отличающийся от описанного выше.

Таким способом может стать метод, когда мастер-пароль будет использован в качестве функции для реализации процесса шифрования и последующего дешифрования данных. Для использования подойдет известный метод, описанный ниже.

Рассмотрим следующую схему аутентификации мастер-ключа [8].

Пусть **Km** – мастер-ключ. В памяти устройства хранится пара (**M**,**C**), где **M** – некоторый массив данных, **C=Km {M}** – результат его шифрования с помощью мастер-ключа **Km**. Каждый раз, когда требуется проверка аутентичности мастер-ключа, берется код **M** из памяти и подается на вход криптомодуля. Полученная на выходе шифрограмма **C^** сравнивается с шифрограммой, хранящейся в памяти.

При положительном результате сравнения аутентичность мастер-ключа считается установленной [8].

В разрабатываемой системе описанная схема проверки аутентичности мастер-пароля в виде, описанном в источнике [8, 12], не реализуема, поскольку в качестве мастер-пароля выступает пользовательская строка случайной длины, которая не пригодна для использования в качестве ключа шифрования.

Поэтому можно предложить следующую модификацию метода: в роли мастер-ключа **Km** в описанной схеме будет ключ **Unlock Key**, полученный хешированием мастер-пароля и представляющий строку фиксированной длины.

3.10. Восстановление данных при утере мастер-пароля

И последним моментом, но не менее важным, является вопрос по восстановлению локальной базы данных при утере мастер-пароля. В разрабатываемой системе решение было принято в пользу отказа от данной операции, так как мнения большинства разработчиков существующих менеджеров паролей ставят под сомнение безопасность операций. Однако периодические запросы на ввод мастер-пароля в разрабатываемом приложении в какой-то степени могут снизить риск утери мастер-пароля.

3.11. Синхронизация данных

Синхронизация данных актуальна в том случае, если пользователь работает с приложением на нескольких устройствах. Более того, наличие в системе сервера синхронизации позволяет пользователю безболезненно перенести данные с одного устройства на другое, например, при покупке нового смартфона.

Для осуществления заявленных функций сервер должен поддерживать следующие запросы:

- Регистрация пользователя. При регистрации сервер должен проверить, не существует ли пользователя с таким же логином и адресом электронной почты. После успешной регистрации сервер отправляет в ответ билет (токен) авторизации.
- Авторизовать пользователя. Сервер осуществляет проверку учетных данных пользователя и в случае успеха отправляет в ответ билет (токен) авторизации.
- Синхронизация: отправить данные. В рамках данного запроса сервер обновляет соответствующее пользователю содержимое таблицы **Data**, если версия полученных данных выше, чем у тех, которые уже хранятся в базе данных.
- Синхронизация: получить данные. Сервер отправляет клиентскому приложению данные из таблицы **Data** и, если необходимо, соответствующий ключевой материал из таблицы **KeyMaterial**. При этом, если сервер хранит "старые" копии, то пересылаться клиенту должна всегда быть последняя версия данных.
- Синхронизация: проверка актуальности данных. Клиенту требуется проверять, владеет ли он свежей версией данных. Чтобы избежать избыточной пересылки данных, сервер должен поддерживать запрос на проверку соответствия локальной версии данных клиента с версией хранящихся на сервере данных.

Таким образом, с учетом рассмотренных выше специфик управления ключами, процессов синхронизации разработана следующая модель работы приложения в виде диаграммы активностей (рис. 6).

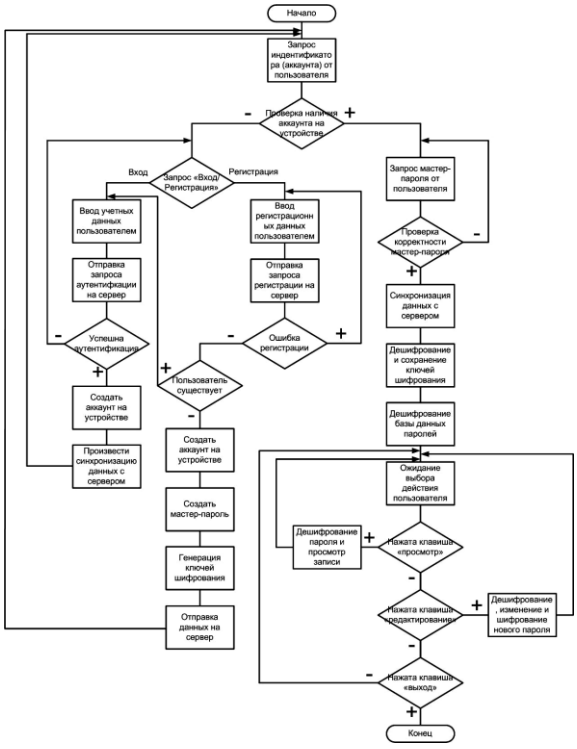

Рис. 6. *Диаграмма активностей, демонстрирующая логику работы приложения*

4. Результаты исследования

В результате проведенного анализа методов и алгоритмов защиты информации, взаимодействия сетевых сервисов получены следующие результаты:

- разобраны и обоснованно выбраны основные принципиальные решения по построению модели/архитектуры работы системы;
- разработаны и обоснованно выбраны форматы хранения данных;
- проработаны вопросы защиты данных в системе, в частности, по их шифрованию и управлению ключами;
- рассмотрены принципы взаимодействия клиентского приложения с сервером синхронизации.

Полученные результаты позволяют сформировать детальное представление о модели работы системы и построить схему работы системы как единого комплекса.

По разработанной модели в дальнейшем можно проводить разработку программного комплекса менеджера паролей для любой ОС. Отдельно стоит отметить, что разработанная модель не является идеальной, но позволяет разработать программный комплекс, исключающий проблемы существующих менеджеров паролей [13].

Список литературы

1. *Advanced* Encryption Standart, 2021. URL: https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm (дата обращение: 26.01.2022).
2. *RFC* 5208 – Public-Key Cryptography Standards (PKCS) #8: Private-Key Information Syntax Specification Version 1.2. 2008. URL: https://datatracker.ietf.org/doc/html/rfc5208 (дата обращения: 20.01.22).
3. *SSL* Certificate & Digital Certificate Authority, 2021. URL: https://www.ssl.com (дата обращения: 20.01.2022).
4. *SSL*-сертификаты бывают разные. 2021. URL: https://www.kaspersky.ru/blog/certificates-are-different/20227/ (дата обращения: 12.01.2022).
5. *US-CERT*. TA14-290A: SSL 3.0 Protocol Vulnerability and POODLE Attack. 2014. URL: https://www.cisa.gov/uscert/ncas/alerts/TA14-290A (дата обращения: 25.01.2022).
6. *What* is SSL, TLS? | How to check SSL certificate Validity. 2021. URL: https://www.encryptionconsulting.com/education-center/ssl-tls-certificates/ (дата обращения: 12.01.2022).
7. *ГОСТ* 34.12-2018. Информационная технология (ИТ). Криптографическая защита информации. Блочные шифры. 2018. URL: https://docs.cntd.ru/document/1200161708 (дата обращения: 20.01.2022).
8. *Иванов М.* Криптографические методы защиты информации в компьютерных системах и сетях. М.: КУДИЦ-ОБРАЗ, 2001.
9. *Краткие* сведения из теории построения генераторов, 2020. URL: https://cyberpedia.su/12x397.html (дата обращения: 12.01.2022).
10. *Криптографический* ключ: что это, для чего и где применяется, 2020. URL: https://itglobal.com/ru-ru/company/glossary/kriptograficheskij-klyuch/ (дата обращения: 12.01.2022).
11. *Методы* HTTP запроса. 2020. URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Methods (дата обращения: 25.01.2022).
12. *Методы* защиты информации в компьютерных системах и сетях, 2019. URL: https://intuit.ru/studies/courses/3580/822/lecture/ 30592?page=8 (дата обращения: 11.01.2022).
13. *Черников А.В.* Рекомендации по разработке менеджеров паролей для ОС Андроид // Вестник Пермского университета. Механика. Математика. Информатика. 2021. Вып. 4 (55). С. 49–57.
14. *Электронная* цифровая подпись на основе алгоритмов с открытым ключом. 2018. URL: https://cyberpedia.su/4x4bc9.html (дата обращения: 24.01.2022).

References

1. *Advanced* Encryption Standart, 2021. URL: https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm (дата обращение: 26.01.2022).
2. *RFC* 5208 – Public-Key Cryptography Standards (PKCS) #8: Private-Key Information Syntax Specification Version 1.2. 2008. URL: https://datatracker.ietf.org/doc/html/rfc5208 (дата обращения: 20.01.22).
3. *SSL* Certificate & Digital Certificate Authority, 2021. URL: https://www.ssl.com (дата обращения: 20.01.2022).
4. *SSL*-sertifikaty byvayut raznye. 2021. URL: https://www.kaspersky.ru/blog/certificates-are-different/20227/ (data obrashcheniya: 12.01.2022).
5. *US-CERT*. TA14-290A: SSL 3.0 Protocol Vulnerability and POODLE Attack. 2014. URL: https://www.cisa.gov/uscert/ncas/alerts/TA14-290A (дата обращения: 25.01.2022).
6. *What i*s SSL, TLS? | How to check SSL certificate Validity. 2021. URL: https://www.encryptionconsulting.com/education-center/ssl-tls-certificates/ (дата обращения: 12.01.2022).
7. *GOST* 34.12-2018. Informacionnaya tekhnologiya (IT). Kriptograficheskaya zashchita informacii. Blochnye shifry. 2018. URL:

https://docs.cntd.ru/document/1200161708 (data obrashcheniya: 20.01.2022).
8. *Ivanov M.* Kriptograficheskie metody zashchity informacii v komp'yuternyh sistemah i setyah // M.: KUDIC-OBRAZ, 2001.
9. *Kratkie* svedeniya iz teorii postroeniya generatorov, 2020. URL: https://cyberpedia.su/12x397.html (data obrashcheniya: 12.01.2022).
10. *Kriptograficheskij* klyuch: chto eto, dlya chego i gde primenyaetsya, 2020. URL: https://itglobal.com/ru-ru/company/glossary/kriptograficheskij-klyuch/ (data obrashcheniya: 12.01.2022).
11. *Metody* HTTP zaprosa. 2020. URL: https://developer.mozilla.org/ru/docs/Web/HTTP/Methods (data obrashcheniya: 25.01.2022).
12. *Metody* zashchity informacii v komp'yuternyh sistemah i setyah, 2019. URL: https://intuit.ru/studies/courses/3580/822/lecture/30592?page=8 (data obrashcheniya: 11.01.2022).
13. *CHernikov A.V.* Rekomendacii po razrabotke menedzherov parolej dlya OS Android // Vestnik Permskogo Universiteta. Mekhanika. Matematika. Informatika. 2021. Vyp. 4 (55). S. 49–57.
14. *Elektronnaya* cifrovaya podpis' na osnove algoritmov s otkrytym klyuchom. 2018. URL: https://cyberpedia.su/4x4bc9.html (data obrashcheniya: 24.01.2022).

Просьба ссылаться на эту статью:

Черников А.В. Одна из возможных реализаций модели менеджера паролей для ОС Андроид // Вестник ПГУ. Математика. Механика. Информатика. 2022. Вып. 1 (56). С. 38–47. DOI: 10.17072/1993-0550-2022-1-38-47.

Please cite this article as:

Chernikov A.V. One of the possible implementations manager’s password model for Android OS // Bulletin of Perm University. Mathematics. Mechanics. Computer Science. 2022. Vyp. 1 (56). P. 38–47. DOI: 10.17072/1993-0550-2022-1-38-47.