Санкт-Петербургский государственный университет

Программная инженерия
Кафедра системного программирования

Дина Дмитриевна Фунт

Анализ шифрования резервной копии Huawei

Курсовая работа

Научный руководитель:
доц., к.т.н. Ю.В. Литвинов

Консультант:
разработчик ПО, ООО “Белкасофт” М.В. Виноградов

Санкт-Петербург
2020

Оглавление

Введение

В современном мире данные, хранящиеся на мобильных устройствах, могут служить одними из главных доказательств виновности или невиновности подозреваемого в преступлении. Исследованием этих данных занимаются эксперты в области цифровой криминалистики. Однако извлекать информацию из мобильных устройств становится все труднее из-за непрерывных обновлений и использования функций ее шифрования, или же вовсе невозможно из-за утраты или поломки смартфона. Резервные данные обычно хранятся в зашифрованном виде с целью защиты конфиденциальности пользователей. Данные в зашифрованном виде не могут быть прочитаны экспертами из области цифровой криминалистики, поэтому важно уметь преобразовывать зашифрованные резервные данные в форму, которая может быть проинтерпретирована человеком или программно. Для создания инструментария разбора резервных копий необходимо ориентироваться в многообразии методов шифрования этих резервных копий.

Согласно исследованию мирового рынка смартфонов [5], во втором квартале 2018 года компания Huawei заняла второе место в мире, превзойдя долю смартфонов Apple, с долей рынка 20,9% для устройств Samsung, 15,8% для Huawei и 12,1% для Apple. Как видно, распространенность смартфонов Huawei возросла, поэтому возможность получать данные из их устройств становится все более существенной для цифровых криминалистов. Однако при анализе резервных копий возникает проблема, заключающаяся в том, что все резервные копии Huawei зашифрованы, у компании реализован свой проприетарный протокол для создания резервных копий. При этом данные резервных копий шифруются либо на основе введенного пользователем пароля, либо без него. Для разбора резервных копий необходимо знать, какой алгоритм шифрования и с какими параметрами используется в каждом из случаев.

1. Цели и задачи

Целью данного исследования является изучение механизма создания резервных копий смартфонов Huawei в случае, когда пароль не задан пользователем, и создание прототипа программы, извлекающей данные из резервных копий.

Для достижения цели были поставлены следующие задачи:

1. Изучить существующие решения.
2. Проанализировать приложение на смартфоне с целью выявления применяемых алгоритмов шифрования.
3. Проанализировать приложение для компьютера.
4. Проанализировать USB трафик между ними.
5. Разработать инструмент, дешифрующий данные резервной копии.

2. Исследование шифрования резервных копий Huawei на основе пользовательского пароля

В статье [4] авторы проводят исследование шифрования резервных копий Huawei, основанного на пароле, введенном пользователем. С помощью обратного инжиниринга программ HiSuite и KoBackup они выявили алгоритмы генерации ключей и шифрования, которые будут более подробно рассмотрены в данной работе. Также изучили алгоритмы верификации пароля и предложили способы атаки на пароль и расшифровки резервных данных.

В исследовании [4] и подобных работах подробно разобрано шифрование резервной копии с паролем пользователя, но отсутствует информация про алгоритм генерации пароля по умолчанию. Данная курсовая работа ставит целью устранить этот пробел.

2.1. Методы создания резервной копии Huawei

Создание резервных копий смартфонов Huawei может проводиться как на телефоне (локальное копирование), так и на компьютере. Резервная копия может включать в себя сторонние приложения, файлы баз данных различных приложений и медиа файлы, среди которых изображения, аудиозаписи, документы и видео.

2.1.1. Локальное резервное копирование

Копирование выполняется на смартфоне, и данные резервного копирования хранятся либо в памяти устройства, либо на SD карте, либо на съемном USB накопителе. При введенном пользовательском пароле данные шифруются, иначе — нет (рис. 1). Приложение, управляющее процессом создания резервной копии на смартфоне — KoBackup.apk.

Как видно из рис. 1, шифрование данных в случае локального копирования без заданного пароля не производится, поэтому этот случай

выходит за рамки данной работы; в ней анализируется вариант резервного копирования при помощи ПК.

Backup Item	Local backup		PC backup	
	with password	without password	with password	without password
DB file	C	P	C	C
Application(apk)	P	P	P	P
Media file	P	P	C	P

Рис. 1: Список зашифрованных файлов при локальном и ПК копировании, C (ciphertext) — зашифрован, P (plaintext) — нет, [4]

2.1.2. Резервное копирование на ПК

В случае создания резервной копии на компьютере используется программа HiSuite. В процессе копирования HiSuite обменивается данными с KoBackup.apk, установленным на устройстве, при помощи USB (рис. 2). Шифрование файлов резервной копии, сохраняемых на компьютере, в этом случае также зависит от того, задан пароль пользователем или нет (рис. 1).

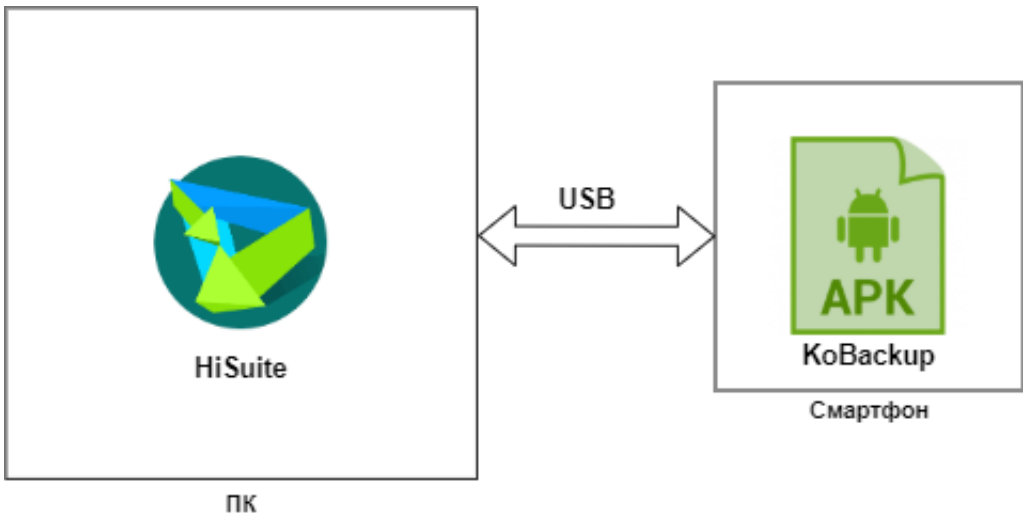

Рис. 2: Модули комплекса, используемые для резервного копирования.

2.2. Методы шифрования

В ходе исследования [4] авторами было выяснено, что алгоритм шифрования данных и генерации ключа зависит от значения целочисленного параметра ”type_attch”. Он может принимать значения 0, 2, 3, но файлы баз данных шифруются только при значениях 2, 3. Этот параметр хранится в файле info.xml, создающемся после выполнения резервного копирования. На рис. 3 представлены функции генерации ключей и алгоритмы шифрования, соответствующие значениям параметра ”type_attch”.

type_attch	KDF	Encryption Algorithm
2	MD5(*password*)	AES128-CTR(*P*, *Key*, *Counter*)
3	PBKDF2-HMAC-SHA256 (*password*, *salt*, *iteration*))	AES256-CTR(*P*, *Key*, *Counter*)
–	SHA256(*password*)	AES128-CTR(*P*, *Key*, *Counter*)

Рис. 3: Алгоритмы шифрования, генерации ключей, [4]

3. Исследование приложения HiSute и его компонентов

Для решения поставленных задач был взят HiSuite версии 9.0.3.300_OVE.

Существуют сборки HiSuite под Windows и под Mac OS, они обе были изучены в данной работе.

3.1. HiSuite для Windows

3.1.1. Анализ трафика USB

Как видно из рис. 2, общение между приложением KoBackup для смартфона и приложением HiSuite для персонального компьютера происходит посредством USB. Для анализа данных, передаваемых по USB, существуют средства перехвата пакетов трафика. Захват пакетов был проведен инструментом USB Packet Capture (USBPcap) [8], просмотр их содержимого проводился в Wireshark [9]. Трафик был проанализирован в различных ситуациях работы приложения: при создании копии и ее восстановлении. В каждом из этих случаев были просмотрены данные и установлено, что существуют команды (рис. 4), исходящие от компьютера, для инициирования различных действий на телефоне, однако все они зашифрованы, сериализованы или сжаты, отчего нечитаемы.

```
0000  1b 00 a0 89 58 47 0f d7  ff ff 00 00 00 00 09 00   ····XG·· ········
0010  00 01 00 05 00 03 03 4d  00 00 00 00 00 00 ca 00   ·······M ········
0020  00 00 39 02 00 00 00 00  44 39 37 32 46 37 39 41   ··9····· D972F79A
0030  30 30 31 46 43 43 36 41  31 34 36 38 45 30 32 32   001FCC6A 1468E022
0040  39 41 41 31 33 39 37 46  46 46 30 34 42 34 43 33   9AA1397F FF04B4C3
0050  32 36 45 31 36 32 42 32  37 37 32 33 38 43 34 37   26E162B2 77238C47
0060  33 34 37 42 39 30 43 35                           347B90C5
```

Рис. 4: Пример шифрованного запроса, посылаемого на смартфон.

На начальных этапах создания резервной копии в трафике можно увидеть шифрованные запросы с компьютера и ответы с телефона (рис. 5), представляющие собой короткие команды WRTE, OKAY,

CLSE, OPEN и т.д. А также, в самом конце процесса, компьютер получает ответ, в виде данных результирующих файлов (зашифрованные базы данных (рис. 6), медиафайлы, info.xml, содержащий метаинформацию о резервной копии).

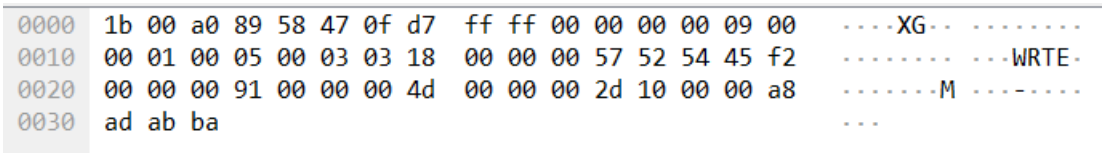

Рис. 5: Пример ответа с телефона.

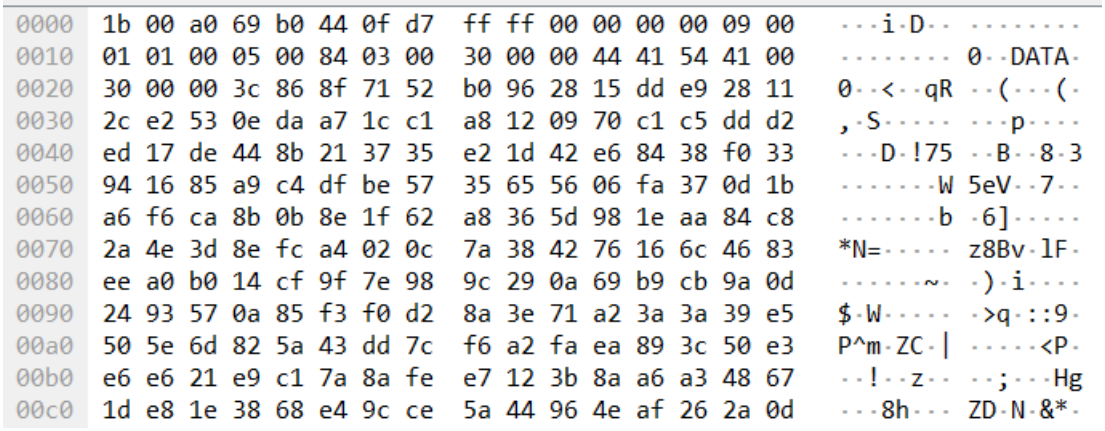

Рис. 6: Данные файла резервной копии в конечном виде, приходящего на компьютер.

Так как данные передаваемых файлов, полученные из трафика, полностью совпадали с данными из созданной резервной копии, хранящейся на компьютере, можно было сделать вывод, что процесс шифрования файлов осуществляется на смартфоне. Пользовательский пароль вводится на компьютере, а значит, он передается в какой-то из команд, предшествующих передаче файлов. Каким образом передается пароль, используемый в случае, если пользователь не задал собственный, на данном этапе установить не удалось. Поэтому для решения этого вопроса было принято решение исследовать алгоритмы, использующиеся в HiSuite.

3.1.2. Анализ HiSuite

Вначале были изучены ресурсы HiSuite, среди которых было найдено два приложения, устанавливаемых программой на телефон и участвующих в процессе резервного копирования, — HiSuite.apk и KoBackup.apk. Также обнаружены XML-документы с переводами различных строк из графического интерфейса HiSuite на множество языков, что впоследствии оказалось полезно в изучении программы.

Изучение HiSuite проводилось в интерактивном дизассемблере IDA Pro [7], предоставляющем возможности статического и динамического анализа программ.

Код, получившийся после дизассемблирования, был сложен для статического анализа, так как был обфусцирован. Причиной для такого вывода послужило наличие в нем множества конструкций, исполнение которых в конечном итоге не влияет на результат выполнения функции, и команд, до которых исполнение программы никогда не дойдет.

Однако, используя информацию из XML-файлов для перевода языка, удалось отследить некоторые функции, вызывающиеся в процессе создания резервной копии, например, проверка введенного пользователем пароля на корректность или формирование имени директории для сохранения резервной копии.

Также IDA имеет плагин, позволяющий в некоторых случаях преобразовать ассемблерный код функции в псевдокод. Для HiSuite эта возможность была недоступна, что сильно затрудняло анализ кода, поэтому было решено изучить приложение под Mac OS.

3.2. HiSuite для Mac OS

3.2.1. Анализ трафика USB

Перед анализом приложения был проанализирован USB трафик. Отличий от трафика приложения на Windows не было найдено. Так же можно было видеть запросы с компьютера, ответы смартфонного приложения, названия баз данных и данные результирующих прило-

жений.

3.2.2. Анализ HiSuite

Программа была проанализирована с помощью дизассемблера для Mac OS и Linux Hopper [1]. В отличие от версии HiSuite для Windows, получившийся код не был обфусцирован, тем самым он был более удобен для анализа. Также его изучение было облегчено возможностью Hopper преобразовывать ассемблерный код в псевдокод на Objective-C.

В коде удалось найти несколько функций, используемых для создания запроса на проведение резервного копирования. Особый интерес из них представляет функция "backupRequestWithRequestModel", т.к. в ней определяются начальные данные, необходимые для проведения резервного копирования, в том числе и пароль, пользовательский или нет.

С помощью метода "useWord" проверяется, введен пароль пользователем или нет, "var_40" содержит в себе строку пользовательского пароля и используется, когда пароль задан, а в случае с резервным копированием без пароля, в коде программы можно увидеть указатель на фиксированную строку, используемую вместо пароля (рис. 7). Hopper имеет функциональность, позволяющую смотреть непосредственно на данные, лежащие в исполняемом файле по определенному адресу, что оказалось полезно в данном случае для извлечения нужной нам строки.

```
rax = [var_8 backupPath];
rax = [rax retain];
[var_28 setPath:rax];
[rax release];
if ([var_28 useWord] != 0x0) {
        rax = [var_40 dataUsingEncoding:0x4];
        rax = [rax retain];
        [var_28 setWord:rax];
        [rax release];
}
if ([var_28 useWord] == 0x0) {
        [var_28 setUseWord:0x1];
        rax = [*qword_100cbae28 dataUsingEncoding:0x4];
        rax = [rax retain];
        [var_28 setWord:rax];
        [rax release];
}
rdx = var_28;
[var_8 backupRequestWithRequestModel:rdx andBlock:0x0];
rax = objc_storeStrong(&var_40, 0x0);
var_34 = 0x0;
```

Рис. 7: Фрагмент кода метода, участвующего в создании резервной копии.

Для подтверждения предположения о том, что полученный пароль такой же и в сборке под Windows, было решено изучить клиентскую часть приложения ”KoBackup”, располагаемую на телефоне.

3.3. Анализ KoBackup

3.3.1. Обратный инжиниринг приложения KoBackup

Существует 2 варианта установки KoBackup.apk на смартфон: вручную из Play Market или при первом запуске HiSuite, приложение устанавливается автоматически, из ресурсов HiSuite.

Был взят KoBackup.apk из ресурсов HiSuite и декомпилирован с помощью JADX [11], декомпилятора из dex[1] в java. Полученный код на java изучен при помощи статического анализа.

[1]dex(dalvik executable) — исполняемый файл в Dalvik, регистровой виртуальной машине для выполнения программ на Android

В процессе изучения кода было установлено, что в случае создания копии данных с компьютера, приложение на смартфоне работает как привязанная служба[2]. Также обнаружен метод, разбирающий сообщение с компьютера и вызывающий один универсальный способ шифрования, с параметрами, полученными из сообщения. Таким образом, для расшифровки резервной копии без пользовательского пароля можно использовать метод, предложенный в исследовании [4].

Также был найден метод, инициализирующий некоторые параметры, полученные в сообщении от HiSuite, которые необходимы для дальнейшей работы.

Было принято решение изменить исполняемый код KoBackup и добавить вывод пароля в лог.

3.3.2. Вывод пароля шифрования

Приложение было дизассемблировано с использованием инструмента apktool [3], преобразующего исполняемые файлы для Android в файлы smali[3], и обратно. Далее был изменен метод, устанавливающий пароль, и добавлен вызов стандартной функции вывода логов (android.util.Log.e()), с сообщением, содержащим значение переменной, отведенной под пароль.

После изменения была испробована обычная установка модифицированного приложения на смартфон, с помощью ADB[4], однако возникала ошибка "**install_failed_shared_user_incompatible**". Причиной этой ошибки являлось то, что приложение имеет значение "android.uid.phone" у параметра sharedUserId, описываемого в манифесте. Данное значение параметра означает, что приложение делит ресурсы с некоторыми другими системными программами, установленными на телефоне. Система безопасности Android сверяет подписи приложе-

[2]Привязанная служба — сервис Android, работающий в фоновом режиме, пока другая компонента приложения(в данном случае HiSuite) привязана к нему.

[3]это язык ассемблера для виртуальной машины Android Dalvik, основанный на языке ассемблера Jasmin Java.

[4]ADB (Android Debug Bridge) – инструмент, входящий в Android-SDK и позволяющий управлять устройством на базе Android.

ний с таким же идентификатором с подписью устанавливаемого приложения, в случае если они не совпадают, установка прерывается с данной ошибкой.

Приложение Android подписывается закрытым ключом. С каждым таким ключом связан открытый сертификат, содержащий хеши файлов приложения, с помощью которого устройства и сервисы могут убедиться в безопасности приложений и их обновлений. Если внести вручную изменения в любой из файлов, их хеши не будут совпадать с исходными, и приложение не пройдет проверку, поэтому требуется подписывать приложения заново после изменения. Однако из-за отсутствия закрытого ключа невозможно получить сертификат, проходящий проверку совместимости с другими приложениями, подписанными тем же ключом.

Для решения возникшей проблемы понадобилось отключить на смартфоне проверки системы безопасности, препятствующие установке измененного KoBackup.apk.

Был взят смартфон модели Honor 6X, на нем проведено разблокирование загрузчика и получение root прав через twrp [6].

На следующем этапе проведена установка программы Lucky Patcher[5] и отключена верификация подписей приложения и проверка целостности apk.

После отключения проверок, приложение установилось, однако при запуске приложения происходило немедленное его закрытие. Это было связано с тем, что при простой сборке apktool отсутствовали нужные для прохождения проверки совместимости сертификаты, имеющиеся у исходного приложения.

Решение этой проблемы заключалось в использовании ключа сборки –copy-original. При его использовании в результирующий архив копируются сертификаты исходного приложения, необходимые для прохождения проверки безопасности. После того, как все проблемы, связанные с установкой и запуском приложения были решены, были сняты логи

[5]Lucky Patcher - патчер приложений, позволяющий получить дополнительные возможности в Android играх и приложениях, например можно отключить проверку лицензии, бесплатно покупать во внутриигровых магазинах и блокировать рекламу.

системных сообщений командой ADB – logcat, в период создания бэкапа с пользовательским паролем. Действительно, в снятых сообщениях, можно было видеть заданный нами пароль (рис. 8). Операция была повторена еще раз, для резервного копирования без пароля, и строка пароля, полученная в этом случае, совпадала с той, что была извлечена из HiSuite для операционной системы Mac OS.

```
03-26 11:48:33.786  2079  7175 W BackupModule: delete file fail : /storage/emulated/0/Android/data/com.huawei.hisuite/cache/MicroMsg
03-26 11:48:33.789  7392  7406 E HwBackup8.0.1.306_OVE: [BackupPasswordHere]: 123456
03-26 11:48:33.797  7392  7406 I HwBackup8.0.1.306_OVE: [BackupLogicService]: handleMessage callback,msg.what=28, msg.arg1=0, msg.arg2=0
03-26 11:48:33.797  7392  7406 I HwBackup8.0.1.306_OVE: [PMSUtil]: finishBackupSessionMethod :true
03-26 11:48:33.797  7392  7406 I HwBackup8.0.1.306_OVE: [BackupObject]:  moduleName = sms
```

Рис. 8: Пользовательский пароль ”123456”, выведенный в лог.

4. Проверка результатов

Для подтверждения верности полученных результатов, пароля и методов расшифровки данных, на языке C# было реализовано тестовое приложение. В его основе лежит написанный на Python скрипт [10] для дешифрации резервных копий, созданных с использованием пользовательского пароля.

На вход программе подается зашифрованная резервная копия (рис. 9), созданная без пароля, и вводится пароль (рис. 10). На выходе получаются расшифрованные данные, в частности, файлы баз данных (рис. 11), содержимое которых теперь можно прочитать в тестовом приложении (рис. 12).

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	54	8B	B7	2B	34	78	FF	FE	02	2B	95	4E	F3	23	43	8E	T‹ ·+4xяю.+•Ny#CЋ
00000010	C8	7A	38	6B	B2	D8	03	99	C8	FD	9A	B6	A0	22	3A	BA	Иz8kІШ.™ИэљЉ ":є
00000020	98	9B	7D	3D	5C	20	DA	BF	3A	C6	D3	86	94	78	09	0A	.›}=\ Ъї:ЖУ†”x..
00000030	BC	A2	ED	87	68	37	B3	0B	61	DB	72	92	80	D2	8B	E7	jўн‡h7i.aЫr’ЂТ‹з
00000040	E7	53	84	E8	22	B2	F1	39	2D	05	BE	52	D7	01	13	4E	зS„и"Ic9-.sRЧ..N
00000050	02	BF	46	2E	6B	4F	AD	34	B1	F4	A2	B4	F7	DC	8F	2D	.їF.kO.4±фўґчЬЏ-
00000060	9D	D8	0E	8E	47	C2	6E	67	09	DC	46	43	3A	7B	EB	BC	ќШ.ЋGBng.ЬFC:{лј
00000070	C7	CB	79	77	55	4D	84	F3	66	BA	14	FE	AD	D6	02	A0	ЗЛywUM„yfє.ю.Ц.
00000080	A4	E2	07	CD	B5	F4	82	3B	8E	19	47	82	24	FC	32	47	¤в.Нµф‚;Ћ.G‚$ь2G

Рис. 9: Данные зашифрованной базы данных в hex-редакторе.

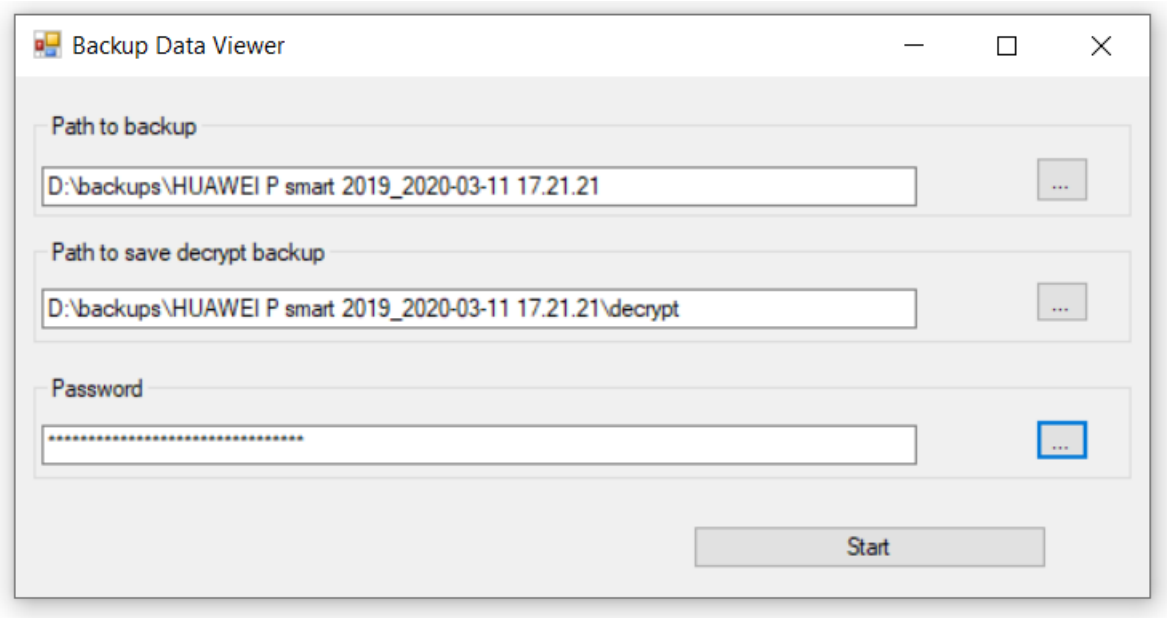

Рис. 10: Ввод начальной информации, директории с резервной копией, пароля.

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	53	51	4C	69	74	65	20	66	6F	72	6D	61	74	20	33	00	SQLite format 3.
00000010	10	00	01	01	00	40	20	20	00	00	00	03	00	00	00	08	.....@ .....
00000020	00	00	00	00	00	00	00	00	00	00	00	03	00	00	00	04	.....
00000030	00	00	00	00	00	00	00	05	00	00	00	01	00	00	00	00	.....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	03	.....
00000060	00	2E	1C	B0	0D	00	00	00	03	0D	AE	00	0F	8F	0F	33	...°.....®..Џ.3
00000070	0D	AE	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.®.....
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

Рис. 11: Данные дешифрованной базы данных в hex-редакторе.

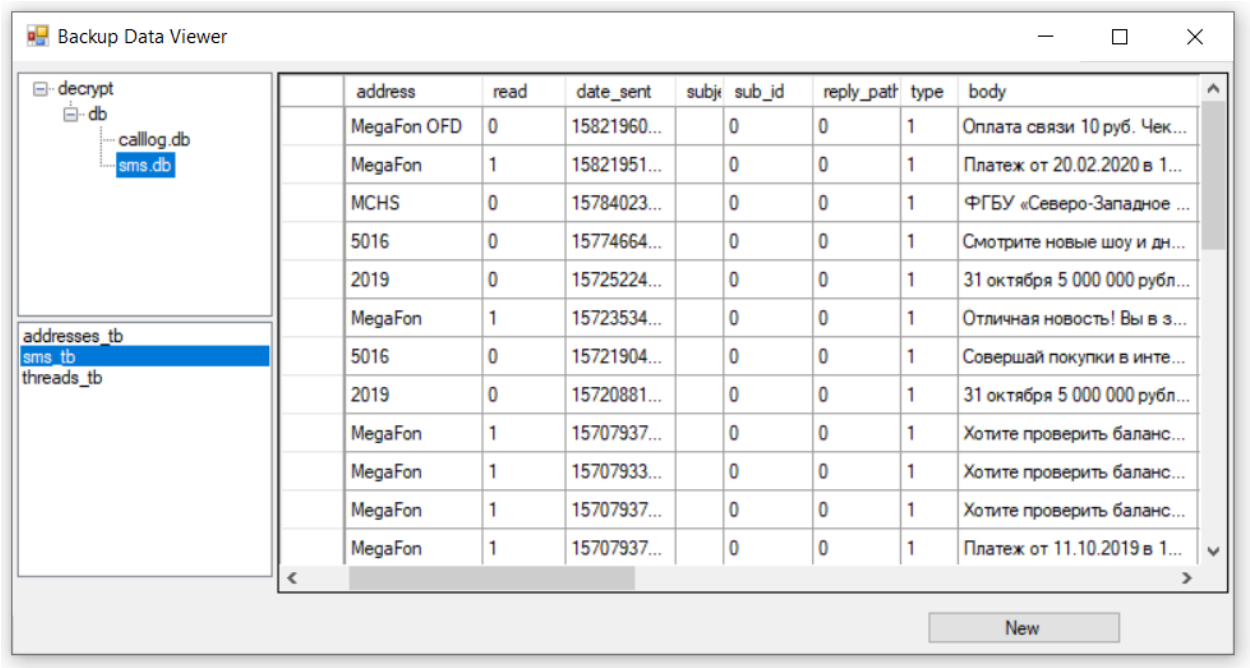

Рис. 12: Расшифрованные базы данных, содержимое таблицы sms_tb базы данных sms.db

Заключение

В рамках данной курсовой были выполнены следующие задачи:

1. Изучены алгоритмы шифрования, используемые в создании резервных копий без пользовательского пароля.
2. Получена строка, используемая в качестве пароля.
3. Предложены алгоритмы для расшифровки резервных копий.
4. Реализовано тестовое приложение для дешифрации данных, с использованием предложенных алгоритмов.

Результаты данной работы в ближайшее время будут интегрированы в Belkasoft Evidence Center [2].

Автор выражает признательность компании Belkasoft за предоставление темы данной курсовой работы. А также отдельную благодарность ее сотрудникам, Никите Тимофееву и Михаилу Виноградову, за ценные советы при планировании исследования, рекомендации по оформлению курсовой и предоставленное оборудование.

Список литературы

[1] Apps Cryptic. Домашняя страница продукта Hopper. — URL: https://www.hopperapp.com/index.html (online; accessed: 15-04-2020).

[2] Belkasoft. Домашняя страница продукта Belkasoft Evidence Center 2020. — 2019. — URL: https://belkasoft.com/ec (online; accessed: 2019-12-18).

[3] Connor Tumbleson Ryszard Wiśniewski. A tool for reverse engineering Android apk files. — URL: https://ibotpeaches.github.io/Apktool/ (online; accessed: 15-04-2020).

[4] Decrypting password-based encrypted backup data for Huawei smartphones / Myungseo Park, Giyoon Kim, Younjai Park et al. // Digital Investigation. — 2019. — 01. — Vol. 28.

[5] Holst Arne. Global market share held by leading smartphone vendors from 4th quarter 2009 to 3rd quarter 2019. — URL: https://www.statista.com/statistics/271496/global-market-share-held-by-smartphone-vendors-since-4th-quarte

[6] LLC Team Win. TeamWin - TWRP. — URL: https://twrp.me/ (online; accessed: 15-04-2020).

[7] SA Hex-Rays. About IDA. — URL: https://www.hex-rays.com/products/ida/ (online; accessed: 2019-12-18).

[8] USBPcap. USBPcap - USB Packet capture for Windows. — URL: https://desowin.org/usbpcap/ (online; accessed: 2019-12-18).

[9] Wireshark. Wireshark User's Guide. — URL: https://www.wireshark.org/docs/wsug_html_chunked/ (online; accessed: 2019-12-18).

[10] dfirfpi. Huawei backup decryptor. — 2019. — URL: https://blog.digital-forensics.it/2019/07/huawei-backup-decryptor.html (online; accessed: 2019-12-18).

[11] skylot. Инструмент декомпиляции JADX. — URL: https://github.com/skylot/jadx (online; accessed: 15-04-2020).