Утвержден
КБДЖ.01559 ПП

АВТОМАТИЗИРОВАННОЕ РАБОЧЕЕ МЕСТО ПОЛЬЗОВАТЕЛЯ СКЗИ «РУТОКЕН ЭЦП 3.0» (АРМ ПОЛЬЗОВАТЕЛЯ СКЗИ «РУТОКЕН ЭЦП 3.0»)

Руководство пользователя

КБДЖ.01559 ПП

Листов 9

2022

Содержание

Введение 3
1. Комплектность 4
2. Описание функционирования 4
3. Работа с программой 5
3.1. Запуск программы и аутентификация 5
3.1.1. Использование на ПЭВМ 5
3.1.2. Использование на мобильных устройствах 5
3.2. Информация об устройстве 5
3.3. Хэширование 6
4. Возвращаемые коды ошибок 6
5. Журнал событий 6
6. Контроль целостности 7
6.1. Контроль целостности ПЭВМ 7
6.2. Контроль целостности на мобильных устройствах Android 7
6.3. Контроль целостности на мобильных устройствах iOS 8

Введение

Настоящий документ является руководством пользователя и содержит общую информацию по использованию АРМ Пользователя СКЗИ «Рутокен ЭЦП 3.0», входящего в состав средства криптографической защиты информации «Рутокен ЭЦП 3.0», предназначенного для шифрования и расшифрования сообщений, создания и проверки электронных подписей, хэширования сообщений.

СКЗИ «Рутокен ЭЦП 3.0» выпускается в трех исполнениях:

- Исполнение 1
  - Смарт-карта «Рутокен ЭЦП 3.0 NFC 3100» (смарт-карта);
  - «Рутокен ЭЦП 3.0 NFC 3100» (полноразмерный токен / micro / Type-C);
  - «Рутокен ЭЦП 3.0 3100» (полноразмерный токен / micro / Type-C);
- Исполнение 2
  - Рутокен ЭЦП 3.0 3200 (полноразмерный токен / micro / полноразмерный токен с поддержкой microSD);
- Исполнение 3
  - Рутокен ЭЦП 3.0 3100 Исполнение А (полноразмерный токен);
  - Рутокен ЭЦП 3.0 3200 Исполнение А (полноразмерный токен).

1. Комплектность

В состав СКЗИ «Рутокен ЭЦП 3.0» входит:

- аппаратный модуль СКЗИ «Рутокен ЭЦП 3.0» (КБДЖ.01558);
- интерфейсная библиотека PKCS#11 (КБДЖ.01561);
- АРМ Пользователя СКЗИ «Рутокен ЭЦП 3.0» (КБДЖ.01559) - только для Windows, Linux, Mac, Android;
- модуль контроля целостности (КБДЖ.01455);
- аппаратно-программный модуль доверенной загрузки, имеющий действующий сертификат по требованиям ФСБ России (для уровня защиты КС2).

2. Описание функционирования

АРМ Пользователя позволяет выполнять криптографические операции библиотеки PKCS#11, используя ключевую информацию, хранящуюся на аппаратном модуле.

АРМ Пользователя поддерживает выполнение следующих российских криптографических алгоритмов: ГОСТ Р 34.11–2012.

АРМ Пользователя позволяет работать АМ только в режиме пользователя.

АРМ Пользователя предназначен для выполнения следующих функций:

- получение информации об устройстве (метке устройства, уникальном идентификаторе устройства, контрольной сумме микропрограммы);
- хэширование в областях памяти АМ.

АРМ Пользователя дает возможность использовать только библиотеку PKCS#11, входящую в состав сертифицированного продукта СКЗИ «Рутокен ЭЦП 3.0», и не позволяет изменять предустановленные режимы функционирования СКЗИ.

3. Работа с программой

3.1. Запуск программы и аутентификация

3.1.1. Использование на ПЭВМ

АРМ Пользователя реализован в виде исполняемого файла **userARM.exe** (для Windows) или **userARM** (для ОС Linux, FreeBSD, macOS) и библиотеки функций стандарта PKCS#11, входящей в состав СКЗИ. Данная библиотека должна располагаться в той же директории, что и исполняемый файл **userARM**.

Для начала работы вставьте АМ в считыватель смарт-карт или приложите к NFC считывателю смарт-карт или подключите к USB-порту и запустите утилиту.

Для запуска необходимо перейти в папку, содержащую исполняемый файл, и выполнить следующую команду[1]:

```
./userARM <команда> [<опции и файлы>]
```

userARM — имя исполняемого файла;

команда — одна из возможных команд, описанных ниже, начинающаяся с «--» (вывод информации об устройстве, хэширование);

опции — параметры команды, начинающиеся с «-»; порядок опций должен быть строго такой, как указано в описании команды;

файлы — файлы, зависящие от команды. Порядок файлов должен быть строго таким, как указано в описании команды.

Примечания:

Библиотека PKCS#11 должна находиться в той же директории, что и исполняемый файл АРМ Пользователя.

3.1.2. Использование на мобильных устройствах

Для работы в ОС Android используется исполняемый файл **userarm.apk**.

Для начала работы вставьте АМ в считыватель смарт-карт, приложите к NFC считывателю смарт-карт или подключите к USB-порту и запустите приложение.

После запуска приложения пользователю в интерактивном режиме доступна командная строка, поддерживающая команды **--info**, **--hash2012_256** и **--hash2012_512** с параметрами, описанными в п. 3.2, 3.3 данного документа.

3.2. Информация об устройстве

Для получения информации об устройстве необходимо выполнить команду:

```
./userARM --info
```

Результат работы данной команды будет выведен на экране:

---

[1] Для ОС Windows команда имеет вид
userARM.exe <команда> [<опции и файлы>]

ID — уникальный идентификатор устройства;

label — метка устройства;

Checksum — контрольная сумма микропрограммы.

3.3. Хэширование

Вычисление значения хэш-функции по алгоритму ГОСТ Р 34.11–2012 длины 256 бит выполняется при помощи команды:

```
./userARM --hash2012_256 -in <infile> -out <outfile>
```

Вычисление значения хэш-функции по алгоритму ГОСТ Р 34.11–2012 длины 512 бит выполняется при помощи команды:

```
./userARM --hash2012_512 -in <infile> -out <outfile>
```

<infile> — имя входного файла, для которого необходимо вычислить значение хэш-функции.

<outfile> — имя выходного файла с результатом работы функции с расширением *.dig.

4. Возвращаемые коды ошибок

Сообщения об ошибках при работе с программой выдаются пользователю в MessageBox. Соответствие между сообщениями и возвращаемыми значениями функций интерфейса PKCS#11 приведены в Приложении Б «СКЗИ «Рутокен ЭЦП 3.0». Правила пользования».

5. Журнал событий

В журнале сохраняется детальная информация о событиях, связанных с работой АМ, результаты данных событий: выполняемая функция интерфейса PKCS#11, возвращаемое значение функции. Запись в журнал ведется в фоновом режиме и выводится на панель АРМ пользователя.

6. Контроль целостности

Для обеспечения контроля целостности модулей используется утилита контроля целостности **checksum.exe** (для Windows), **checksum** (для ОС Linux, FreeBSD, macOS, Android) или **checksum.framework** (для iOS), входящая в состав дистрибутива СКЗИ «Рутокен ЭЦП 3.0». Проверка осуществляется в соответствии с алгоритмом HMAC_GOSTR3411_2012_256 на нулевом ключе.

6.1. Контроль целостности ПЭВМ

Утилита контроля целостности представляет собой приложение командной строки.

Перед запуском утилиты необходимо создать файл, содержащий хэш-значения контролируемых файлов (checksum, userARM, rtpkcs11ecp) и пути к ним: в начале строки указывается предварительно вычисленное значение хэш-функции, печатается два пробела, затем указывается путь к файлу, для которого вычислялся хэш.

Пример:

50c7ef376e75368fccafa00d197ef93bb9bcb5c0571d4ed31776f6d7401ace35 C:\\Windows\System32\rtpkcs11ecp.dll

Для выполнения проверки утилита **checksum.exe** (**checksum** в случае ОС Linux, FreeBSD, macOS) запускается в командной строке при помощи команды[**]

```
>checksum.exe --key 0 --file <path-to-file>
```

<path-to-file> — путь к файлу, содержащему хэш-значения контролируемых файлов и пути к ним.

Результатом работы данной команды будет заново вычисленное значение хэш-функции для указанных модулей и результат его сравнения со значением, прописанным в созданном файле.

В случае использования утилиты под ОС типа *nix **<file>** должен формироваться редактором, поддерживающим unix-переносы строк. На таких ОС утилита имеет имя **checksum**. Пути к контролируемым файлам должны прописываться в **<file>** с учетом программно-аппаратной платформы.

6.2. Контроль целостности на мобильных устройствах Android

Перед установкой ПО СКЗИ «Рутокен ЭЦП 3.0» на мобильное устройство пользователь должен загрузить его на ПЭВМ с предустановленной утилитой **checksum.exe** (или **checksum**) из состава сертифицированного СКЗИ «Рутокен ЭЦП 3.0». С помощью утилиты необходимо вычислить контрольную сумму и сравнить со значениями, представленными в формуляре для модулей:

- **checksum.apk** — утилита контроля целостности;
- **userarm.apk** — утилита АРМ Пользователя.

[**] Для ОС типа *nix команда имеет вид
./checksum --key 0 --file <path-to-file>

Порядок вычисления контрольных сумм на ПЭВМ с помощью утилиты контроля целостности описан в п. 6.1 данного документа.

Если значения контрольных сумм верны, файлы **checksum.apk**, **userarm.apk** переносятся и устанавливаются на мобильное устройство/смарт-терминал.

Перед использованием СКЗИ на мобильном устройстве необходимо убедиться, что на нем установлена утилита «Модуль контроля целостности». Контроль целостности утилиты контроля целостности **checksum**, АРМ Пользователя **userARM** и библиотеки **librtpkcs11ecp.so** выполняется автоматически с помощью утилиты «Модуль контроля целостности» при запуске приложений.

6.3. Контроль целостности на мобильных устройствах iOS

Исполняемые файлы в iOS имеют формат Mach-O. Контроль целостности осуществляется через специальный фреймворк **checksum.framework,** функционал которого позволяет подсчитывать контрольную сумму бинарных модулей, входящих в состав приложения (в том числе позволяет производить контроль целостности самого фреймворка **checksum.framework**).

Для встраивания утилиты **checksum.framework** в проект необходимо выполнить следующие действия:

1. Добавить фреймворк **checksum.framework** в проект.
2. Импортировать утилиту, добавив в модуль проекта строку import checksum.
3. Вызвать функцию **getBinaryDigest()** для каждой бинарной зависимости и исполняемого файла и отобразить результат пользователю.

Функция **getBinaryDigest()** используется для вычисления контрольной суммы с использованием алгоритма HMAC_GOSTR3411_2012_256.

```
func getBinaryDigest(for name: String, key: Data) throws -> Digest
```

name — название бинарной зависимости или исполняемого файла в Mach-O формате, для которой необходимо вычислить контрольную сумму;

key — ключ для вычисления контрольной суммы **(**по умолчанию используется нулевой ключ).

Результатом работы функции является структура **Digest**, содержащая рассчитанные значения контрольных сумм:

```
struct Digest {
    let arch: MachOArch
    let cs1: String
    let cs2: String?
}
```

arch — архитектура;

cs1 — контрольная сумма сегмента __TEXT модуля из оперативной памяти устройства;

cs2 — контрольная сумма сегмента __DATA файла модуля из flash-памяти устройства.

Пример:

```
...
private func getDefaultKey() -> Data {
    return Data(repeating: 0, count: 256/8)
}
...
let key = getDefaultKey()
do {
    let digest = try getBinaryDigest(for: "binary", key: key)
    print(digest.arch)
    print(digest.cs1)
    print(digest.cs2)
} catch let error {
    print(error)
}
...
```

Лист регистрации изменений

№ изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Входящий № сопрово-дительного докум. и дата	Подп.	Дата
	измененных	заменен-ных	новых	аннулиро-ванных					