Руководство оператора для ПО «Вершина»

Оглавление

1. Термины и определения

Аббревиатура	Описание
ПО	Программное обеспечение для карт
NVM	Non Volatile Memory – тип памяти, который может сохранить информацию даже после отключения питания
USB	Universal Serial Bus – универсальная последовательная шина
MDK	Microcontroller Development Kit – комплексный набор программных или аппаратных средств для создания и отладки программ для микроконтроллеров
IDE	Integrated Development Environment (интегрированная среда разработки) – программный комплекс, объединяющий базовые инструменты для написания, проверки и компиляции кода
API	Application Interface (программный интерфейс приложения) – набор правил и протоколов, позволяющий разным программам общаться друг с другом
JCRE	Java Card Runtime Environment – специальная среда для выполнения технологии Java Card, которая отвечает за запуск и управление приложениями (апплетами) на устройтсвах с крайне ограниченными ресурсами, такими как смарт-карты, SIM-карты, банковские чипы и токены
JCVM	Java Card Virtual Machine – специализированная облегченная версия стандартной виртуальной машины Java (JVM), созданная для выполнения программ (апплетов) на устройствах с крайне ограниченными ресурсами: от нескольких килобайт памяти до пары сотен килобайт постоянной памяти
CATRE	Card Application Toolkit Runtime Environment – это защищенная программная среда выполнения на SIM/UICC-картах, которая позволяет запускать мини-приложения (апплеты) и обеспечивает их взаимодействие с телефоном
APDU	Application Protocol Data Unit – тип управляющих команд, используемых для интегральных схем. ГОСТ Р ИСО 7816-4-2013 «Карты идентификационные. Карты на интегральных схемах. Часть 4. Организация, защита и команды для обмена»

2. Определение ПО

2.1 Краткое введение

Программное обеспечение «Вершина» — это стандартный телекоммуникационный продукт JAVA USIM-карта, разработанный компанией ООО «Ватчдата Технологии» на основе технологии смарт-карт и соответствующий таким спецификациям, как ISO7816, Global Platform и JavaCard. Оно может быть установлено в терминалах, поддерживающих сети GSM, WCDMA, TD-SCDMA, LTE и 5G, включая мобильные телефоны и другие терминальные устройства. Построенное на базе аппаратного обеспечения смарт-карт, ПО «Вершина» надежно хранит пользовательские ключи или цифровые сертификаты и использует встроенные криптографические алгоритмы для аутентификации. Программное обеспечение поддерживает контактные протоколы, соответствует стандартам 3GPP и ESTI и позволяет запускать разнообразные приложения на телекоммуникационных картах.

2.2 Ключевые функции

Кроссплатформенная совместимость: в ПО «Вершина» могут запускаться апплеты от разных разработчиков.

Совместное использование нескольких приложений: на одной карте можно одновременно установить несколько независимых апплетов, а изоляция данных между приложениями достигается с помощью механизма брандмауэра.

Изоляция безопасности: каждый апплет имеет свой собственный независимый контекст выполнения.

Установка после выпуска: приложение может быть удаленно загружено, установлено, обновлено или удалено по защищенному каналу OTA/SCP после выпуска карты.

Атомарность данных: даже если при записи данных во Flash происходит сбой питания, то механизм транзакций обеспечивает атомарность операций записи.

2.3 Архитектура ПО

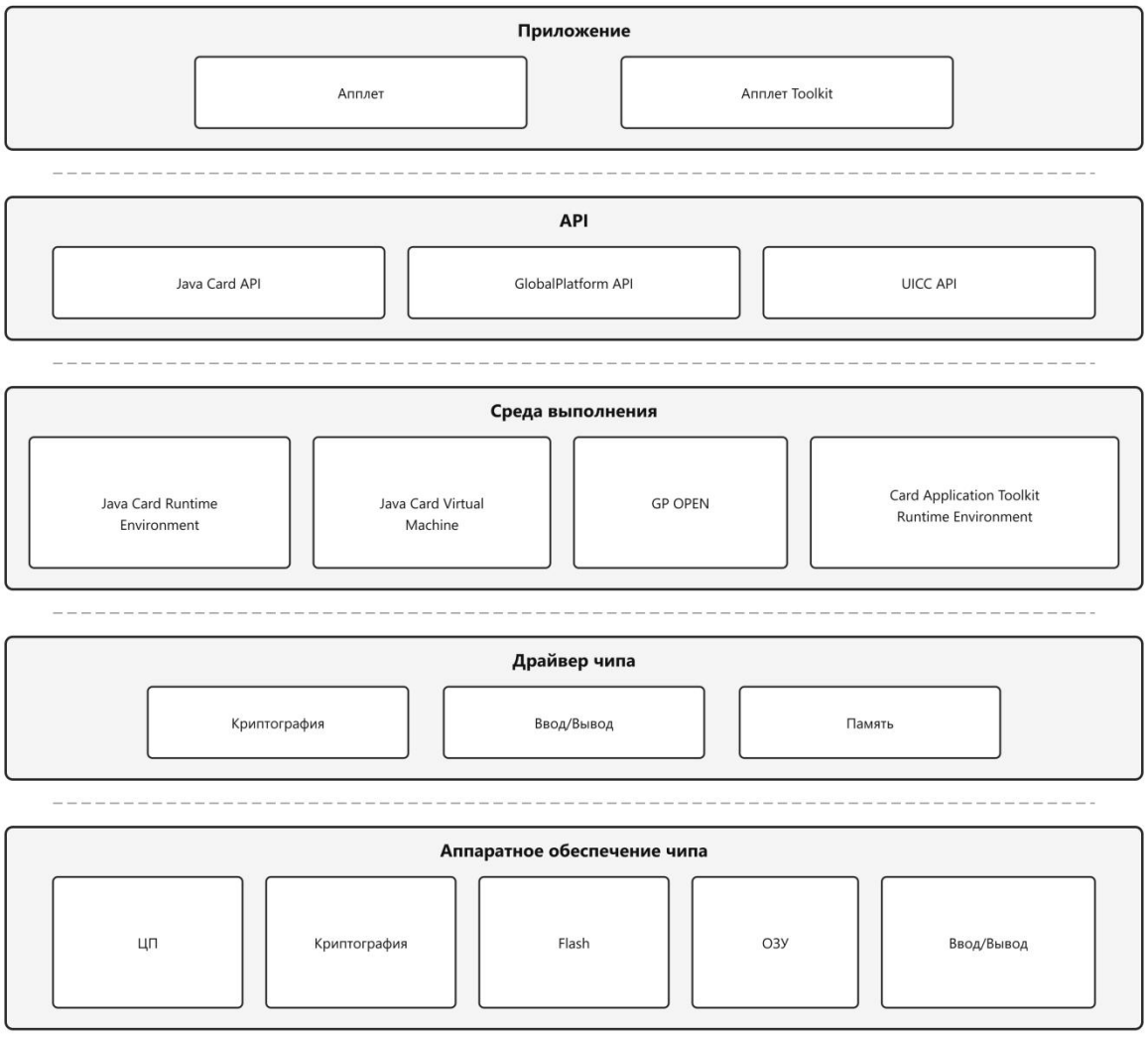

Рис.2.1 Структурная схема ПО

Ниже приведен обзор общей пятиуровневой архитектуры ПО «Вершина»:

2.3.1 Приложение

Уровень приложений – это самый верхний уровень программного обеспечения «Вершина» непосредственно взаимодействующий с бизнес-логикой и обеспечивающий размещение всех приложений, запущенных на карте.

Этот уровень содержит два основных модуля:

- Апплет:

Базовый модуль приложения Java Card; все приложения карт должны быть зарегистрированы и запускаться как апплеты. Каждый апплет создается с помощью метода install() и получает и обрабатывает команды APDU с помощью метода process().

- Апплет Toolkit:

Приложение STK (SIM Toolkit) обрабатывает упреждающие команды, поддерживая карты для инициирования взаимодействия с терминалами (например, отображение меню, отправка SMS-сообщений или совершения звонков).

2.3.2 Уровень интерфейса API

Уровень API предоставляет стандартизированные программные интерфейсы для приложений верхнего уровня.

- Java Card API：

  Стандартный набор API Java Card определяет таки пакеты, как javacard.framework, javacard.security, и javacardx.crypto. Он предоставляет основные функциональные возможности, включая управление жизненным циклом апплета, обработку APDU, основные типы данных и обработку исключений.

- GlobalPlatform API：

  The GlobalPlatform стандартизирует API, предоставляя интерфейсы для безопасного управления доменами, персонализацию установки/удаления приложений и работы с каналами безопасности SCP (Secure Channel Protocol).

- UICC API：

  Телекоммуникационный API UICC предоставляет доступ к файловой системе UICC, операциям EF/DF, телекоммуникационной аутентификации (например, алгоритм Milenage) и интерфейсам, связанным с CAT (Card Application Toolkit). Он соответствует спецификации ETSI TS 102-241.

2.3.3 Среда выполнения

Уровень среды выполнения – это ядро программного обеспечения «Вершина», отвечающее за выполнение приложений, управление ресурсами и планирование задач.

- JCRE:

  Среда выполнения Java Card управляет жизненным циклом апплета, распределением APDU, механизмами транзакций, сборкой мусора объектов, изоляцией брандмауэром и т.д..

- JCVM:

  Java Card Virtual Machine отвечает за интерпретацию и выполнение байт-кода.

- GP OPEN:

  GlobalPlatform's Open Platform Environment управляет архитектурой безопасности карты, жизненным циклом приложения (ЗАГРУЗКА → УСТАНОВКА → ВЫБОР → ПЕРСОНАЛИЗАЦИЯ → БЛОКИРОВКА), а также регистрацией и авторизацией внешних объектов.

- CATRE:

  Набор инструментов работает в среде выполнения, управляет планированием и выполнением упреждающих команд STK и поддерживает такие функции, как ОТОБРАЖЕНИЕ ТЕКСТА, НАСТРОЙКА МЕНЮ и ОТПРАВКА КОРОТКИХ СООБЩЕНИЙ.

2.3.4 Уровень драйвера чипа

Уровень драйверов маскирует аппаратные различия на базовом уровне, обеспечивая единый интерфейс доступа к оборудованию для среды выполнения верхнего уровня.

- Криптография:

  Уровень упаковки чипа обеспечивает аппаратные интерфейсы шифрования, поддерживающие симметричные алгоритмы, алгоритмы хэширования и генерацию случайных чисел (TRNG). Такие классы, как Cipher, Signature и KeyAgreement в JCVM, в конечном счете, вызывают этот уровень.

- Ввод/вывод:

  Управление передачей данных ввода/вывода с карты памяти, поддержка протокола T=0 (передача символов), обработка передачи/приема APDU и повторной передачи ошибок на уровне протокола.

- Память:

  Управление операциями чтения/записи/стирания с Flash-памяти, обеспечивая основные функции, такие как стирание страниц, ввод слов, управление хранилищем данных и защита от разрывов.

2.3.5 Аппаратное обеспечение чипа

На самом низком уровне находится физический чип, который обеспечивает реальные вычислительные возможности, хранение данных и коммуникацию.

- ЦП: 32-битный процессор.
- Криптография: аппаратный сопроцессор шифрования, поддерживающий аппаратное ускорение алгоритмов DES/AES.
- Flash: хранит программное обеспечение, код апплета, ключи и файловые данные.
- ОЗУ: используется для хранения стека среды выполнения и временных переменных.
- Интерфейс ввода/вывода: интерфейс ISO 7816 обеспечивает коммуникационные возможности физического уровня.

2.4 Основной механизм

2.4.1 Жизненный цикл Апплета

Состояние жизненного цикла Апплета:

ЗАГРУЗКА → УСТАНОВКА → ВЫБОР → ПЕРСОНАЛИЗАЦИЯ → БЛОКИРОВКА

- ЗАГРУЗКА: загрузите файл Cap на карту памяти через защищенный канал GP SCP
- УСТАНОВКА: JCRE вызывает статический метод install() апплета, чтобы созать его экземпляр
- ВЫБОР: терминал отправляет команду SELECT APDU для выбора апплета в качестве текущего местоположения.
- ПЕРСОНАЛИЗАЦИЯ: войдите в цикл process() для получения и обработки последующих инструкций APDU
- БЛОКИРОВКА: карта заблокирована

2.4.2 Управление APDU

Связь между терминалами и картой осуществляется через APDU в соответствии со стандартом ISO 7816.

- Команда APDU：CLA | INS | P1 | P2 | Lc | Data | Le
- Ответ APDU: Данные | SW1 SW2 (9000 указывает на успех выполнения)

JCRE отвечает за распространение APDU: получив инструкцию, он вызывает метод process(APDU apdu) на основе выбранного в данный момент апплета.

2.4.3 Механизм транзакции

Если во время процесса атомарной записи в несколько полей произойдет сбой питания, программное обеспечение автоматически откатит все несохраненные транзакции при следующем включении, чтобы обеспечить согласованность данных.

2.4.4. Брандмауэр

Механизм брандмауэра программного обеспечения гарантирует:

- Каждый апплет запускается в отдельном контексте.
- Объекты в одном контексте не могут напрямую обращаться к объектам в другом контексте.

3. Описание функций ПО

3.1 Формат команд ПО

3.1.1 Формат инструкции APDU

Команда APDU состоит из заголовка данных и тела данных (Таблица 1). Заголовок данных содержит поля CLA, INS, P1 и P2, которые являются обязательными компонентами команды. Тело данных является необязательным и содержит Lc, Data и Le.

Код	Длина	Описание	Тип
CLA	1	Байт класса команды	Заголовок данных
INS	1	Код инструкций	
P1	1	Параметр 1	
P2	1	Параметр 2	
Lc	0 или 1	Количество байт в теле данных команды	Тело данных
Data	Lc	Тело данных команды	
Le	0 или 1	Максимальный размер данных ответа в байтах	

Таб.1 Содержание команды APDU

Командная структура APDU может иметь следующие четыре возможные комбинации (Таблица 2):

Вариант	Структура
1	CLA INS P1 P2
2	CLA INS P1 P2 Le
3	CLA INS P1 P2 Lc Data
4	CLA INS P1 P2 Lc Data Le

Таб.2 Содержание команд APDU

- Кодировка поля CLA

Значение старших четырех битов (биты 8–5) поля Class приведены Таблице 3: биты 3 и 4 представляют идентификатор данных безопасности, в то время как биты 1 и 2 указывают используемый логический канал (в диапазоне от 0 до 3). Если карта UICC поддерживает механизм логических каналов, то максимальное доступное количество каналов указывается в объекте данных о совместимости карт в ATR; если этот объект данных отсутствует, поддерживается только канал 0.

Для приложений, работающих на UICC с поддержкой логических каналов, удаляются поля класса или устанавливаются для них значения по умолчанию при вычислении подписи для проверки

сообщения.

b8	b7	b6	b5	b4	b3	b2	b1	Значение	Описание
0	0	0	0	-	-	-	-	0X	См. 7816-4
1	0	1	0	-	-	-	-	AX	См. 7816-4, если не указано иное.
1	0	0	0	-	-	-	-	8X	См. 1876-4 и данный документ.
-	-	-	-	X	X	-	-	-	Идентификатор сообщения о безопасности – см. таблицу ниже
-	-	-	-	-	-	X	X	-	Номер логического канала

Таб.3 Кодирование байта CLA

Примечание: Номера логических каналов назначаются картой UICC, при этом логический канал с номером 0 остается постоянно доступным.

b4	b3	Значение
0	0	Между терминалом и картой не используется сообщения безопасности (SM)
0	1	Частный формат сообщения безопасности (SM)
1	0	Заголовок команды без аутентификации
1	1	Заголовок команды с аутентификацией

Таб.4 Идентификационный код сообщения безопасности (SM)

По умолчанию карта не использует защищенный обмен сообщениями, если это явно не указано приложением.

- Другие поля:

Поле команды: обратитесь к соответствующим главам ниже.

Поле параметров: использование байтов параметров P1 и P2 зависит от конкретной команды. Если параметры не используются, им присваивается значение «00».

Поле Lc: Это поле указывает длину данных (от 1 до 255) и является необязательным. Если оно присутствует, за ним следуют байты данных соответствующей длины.

Поле данных: коды связаны с определенными командами.

Le: указывает максимальную ожидаемую длину данных, возвращаемых после передачи команды (необязательно). Если это поле задано, ответ будет содержать информацию о его длине. Когда Le

настроен как «00», это означает, что терминал ожидает получить 256 байт данных, в то время как карта может возвращать данные в диапазоне от 1 до 256 байт.

3.1.2 Структура APDU ответа

APDU ответ содержит необязательный текст данных и обязательное слово состояния, состоящее из двух байт (SW1 и SW2), при этом длина данных обозначается символом Lr.

Кодирование	Длина	Описание
Data	xx	Байт данных ответа
SW1	1	SW1
SW2	1	SW2

Таб.5 Ответ на содержимое APDU

Определения общих кодов состояния:

SW1/SW2	Описание
Стандартный процесс	
90 00	Команда успешно выполнена
91 XX	Команда выполнена успешно и вернула длину данных, полученных от UICC (обозначенную как XX).
Процесс задержки	
93 00	STK занят; эта команда не может быть выполнена.
Предупреждение	
62 00	Справочная информация отсутствует; энергозависимая память остается неизменной.
62 81	Часть возвращаемых данных может быть повреждена
62 82	При чтении файла или записи длины Le достигается конец
62 83	Выбранный файл не действителен
63 CX	Команда была выполнена успешно после внутренних попыток повтора X. Проверка не пройдена. У пользователя осталось X попыток. Для команды проверки PIN-кода значения SW1 и SW2 указывают на то, что команда была выполнена успешно, но PIN-код был введен неверно; его можно повторить до X раз. Для любых других команд это означает успешное выполнение после X внутренних попыток.
Обнаружение ошибки	
67 00	Длина ошибки
67 XX	За исключением SW2 = 00, это слово состояния относится к команде.
6B 00	Неверные параметры P1-P2
6D 00	Это поле команды не поддерживается или содержит ошибку
6E 00	CLA-байт не поддерживается
6F 00	Техническая ошибка: точная причина отсутствует.
6F XX	За исключением SW2 = 00, это слово состояние относится к команде.
Функции в поле CLA не поддерживаются	
68 00	Справочная информация отсутствует
68 81	Логический канал не поддерживается
68 82	Сообщения безопасности не поддерживаются

Команда не может быть выполнена	
69 00	Справочные сообщения отсутствуют
69 81	Команда не совместима с данной файловой структурой
69 82	Не соответствует требованиям безопасности
69 83	Аутентификация/PIN-код запрещены
69 84	Указанные данные неверны
69 85	Условия использования не соблюдены
69 86	Команда не может быть выполнена (EF не выбран)
Неверный параметр	
6A 80	Данные параметра неверны
6A 81	Функция не поддерживается
6A 82	Файл не найден
6A 83	Запись не найдена
6A 86	Параметры P1-P2 неверны
6A 87	LC не соответствует требованиям P1-P2
6A 88	Указанные данные не найдены
Ошибка приложения	
98 50	Команда увеличения не может быть выполнена, так как она достигла максимального значения
98 62	Ошибка аутентификации: неверный MAC
98 64	Ошибка аутентификации: контекст безопасности GSM не поддерживается

Таб.6 Значения SW

3.2 Команда GP

Команды GP и соответствующие им определения CLS и INS приведены в таблице:

Команда	CLA	INS
DELETE («УДАЛИТЬ»)	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	E4
GET DATA («ПОЛУЧИТЬ ДАННЫЕ»)	'00' - '0F', '40' - '4F', '60' - '6F', '80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	CA
GET STATUS («ПОЛУЧИТЬ СТАТУС»)	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	F2
INSTALL («УСТАНОВИТЬ»)	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	E6
LOAD («ЗАГРУЗИТЬ»)	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	E8
MANAGE CHANNEL («УПРАВЛЯТЬ КАНАЛОМ»)	'00' - '03' or '40' - '4F'	70
PUT KEY («ВВЕСТИ КЛЮЧ»)	'80' - '8F', 'C0' - 'CF' or 'E0' - 'EF'	D8
SELECT («ВЫБРАТЬ»)	'00' - '03' or '40' - '4F'	A4
SET STATUS («УСТАНОВИТЬ СТАТУС»)	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	F0
STORE DATA («ХРАНЕНИЕ ДАННЫХ»)	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	E2

INITIALIZE UPDATE («ЗАПУСТИТЬ ПОДГОТОВКУ ОБНОВЛЕНИЯ»)	'80' - '83' or 'C0' - 'CF'	50
EXTERNAL AUTHENTICATION («ВНЕШНЯЯ АУТЕНТИФИКАЦИЯ»)	'84' - '87' or 'E0' - 'EF'	82

Таб.7 Набор команд GP

3.2.1 Команда INITIALIZE UPDATE

Функции команды:

Команда INITIALIZE UPDATE используется при явной инициализации защищенного канала для передачи данных карты и сеанса между картой и хостом. Данная команда инициирует установление сеанса защищенного канала.
В любой момент в рамках действующего защищенного канала на карту может быть отправлена команда INITIALIZE UPDATE для инициирования нового сеанса защищенного канала.
Команду INITIALIZE UPDATE не следует путать с одноименными командами в устаревших приложениях.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	'80' - '83' or 'C0' - 'CF'	Байт класса
INS	'50'	INITIALIZE UPDATE
P1	'xx'	Контрольный параметр для сравнения P1
P2	'00'	Контрольный параметр для сравнения P2
Lc	'08'	Длина поля данных
Data	'xxxxxx'	Запрос хоста
Le	'00'	

Таб.8 Сообщение команды INITIALIZE UPDATE

3.2.2 Команда EXTERNAL AUTHENTICATION

Функции команды:

Команда EXTERNAL AUTHENTICATION используется картой при явной инициализации защищенного канала для аутентификации хоста и определения уровня безопасности, требуемого для всех последующих команд.
Перед выполнением данной команды должно быть успешное выполнение команды INITIALIZE UPDATE.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	'84' - '87' or 'E0' - 'EF	Байт класса

INS	‘82’	EXTERNAL AUTHENTICATION
P1	'xx'	Уровень безопасности
P2	'00'	Контрольный параметр для сравнения P2
Lc	'10'	Длина поля данных
Data	'xxxxxx'	Криптограмма хоста и MAC
Le		Не представлено

Таб.9 Сообщение команды EXTERNAL AUTHENTICATION

3.2.3 Команда INSTALL

Функции команды:

Команда INSTALL подается домену безопасности для инициирования или выполнения различных этапов, необходимых для управления содержимым карты.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	Байт класса
INS	'E6'	INSTALL
P1	‘xx’	Контрольный параметр для сравнения P1
P2	'00', '01', or '03'	Контрольный параметр для сравнения P2
Lc	'xx'	Длина поля данных
Data	'xxxxxx'	Установить данные (и C-MAC при наличии)
Le	'00'	

Таб.10 Сообщение команды INSTALL

3.2.4 Команда DELETE

Функции команды:

Команда DELETE используется для удаления однозначно идентифицируемого объекта, такого как исполняемый загрузочный файл, приложение, исполняемый загрузочный файл и связанные с ним приложения или ключ. Для удаления объекта он должен однозначно идентифицироваться выбранным приложением.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	'80' - '8F', 'C0' - 'CF', or 'E0' - 'EF'	Байт класса
INS	'E4'	DELETE
P1	'xx'	Контрольный параметр для сравнения P1

P2	'xx'	Контрольный параметр для сравнения P2
Lc	'xx'	Длина поля данных
Data	'xxxxxx'	Объекты закодированные в формате TLV(и C-MAC при наличии)
Le	'00'	

Таб.11 Сообщение команды DELETE

3.2.5 Команда PUT KEY

Функции команды:

Команда PUT KEY используется для:

- Замены существующего ключа новым: новый ключ имеет тот же или иной номер версии, тот же идентификатор, что и заменяемый ключ;
- Замены нескольких существующих ключей новыми: новые ключи имеют тот же или иной номер версии ключа (одинаковый для всех новых ключей), но те же идентификаторы ключей, что и заменяемые ключи;
- Добавления одного нового ключа: новый ключ имеет иную комбинацию идентификатора ключа и номер версии ключа по сравнению с существующими ключами;
- Добавления нескольких новый ключей: новые ключи имеют иные комбинации идентификаторов ключа и номера версии ключа (последний одинаков для всех новых ключей) по сравнению с существующими ключами.

Если операция управления ключами требует выполнения нескольких команд PUT KEY, рекомендуется объединять эти команды в цепочку для обеспечения целостности операции.
В данной версии спецификации открытые значения асимметричных ключей представлены в открытом виде.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	'80' - '8F', 'C0' - 'CF' or 'E0' - 'EF'	Байт класса
INS	'D8'	PUT KEY
P1	'xx'	Контрольный параметр для сравнения P1
P2	'xx'	Контрольный параметр для сравнения P2
Lc	'xx'	Длина поля данных
Data	'xxxxxx'	Данные ключа (и C-MAC при наличи)
Le	'00'	

Таб.12 Сообщение команды PUT KEY

3.3 Команда UICC

Спецификации UICC вместе с соответствующими определениями CLA и INS представлены в таблице:

Команда	CLA	INS
SELECT FILE («ВЫБРАТЬ ФАЙЛ»)	0X	A4
STATUS («СТАТУС»)	8X	F2
READ BINARY («ЧТЕНИЕ БИНАРНЫХ ДАННЫХ»)	0X	B0
UPDATE BINARY («ОБНОВИТЬ БИНАРНЫЕ ДАННЫЕ»	0X	D6
READ RECORD («ЧТЕНИЕ ЗАПИСИ»)	0X	B2
UPDATE RECORD («ОБНОВЛЕНИЕ ЗАПИСИ»)	0X	DC
SEARCH RECORD («ПОИСК ЗАПИСИ»)	0X	A2
INCREASE («УВЕЛИЧЕНИЕ»)	8X	32
VERIFY PIN («ПРОВЕРИТЬ PIN-КОД»)	0X	20
CHANGE PIN («ПОМЕНЯТЬ PIN-КОД»)	0X	24
DISABLE PIN («ОТКЛЮЧИТЬ PIN-КОД»)	0X	26
ENABLE PIN («ВКЛЮЧИТЬ PIN-КОД»)	0X	28
UNBLOCK PIN («РАЗБЛОКИРОВАТЬ PIN-КОД»)	0X	2C
DEACTIVATE FILE («ДЕАКТИВИРОВАТЬ ФАЙЛ»)	0X	04
ACTIVATE FILE («АКТИВИРОВАТЬ ФАЙЛ»)	0X	44
AUTHENTICATE («АУТЕНТИФИКАЦИЯ»)	0X	88
GET CHALLENGE («ПОЛУЧИТЬ ЧЕЛЛЕНДЖ»)	0X	84
TREMINAL PROFILE	80	10

(«ПРОФИЛЬ ТЕРМИНАЛА»)		
ENVELOP («ОБОЛОЧКА»)	80	C2
FETCH («ВЫБОРКА»)	80	12
TERMINAL RESPONSE («ОТВЕТ ТЕРМИНАЛА»)	80	14
MANAGE CHANNEL («УПРАВЛЕНИЕ КАНАЛОМ»)	0X	70
GET IDENTITY («ПОЛУЧИТЬ ИДЕНТИФИКАЦИЮ»)	8X	78
GET RESPONSE («ПОЛУЧИТЬ ОТВЕТ»)	0X	C0

Таб.13 Команды UICC

3.3.1 Команда файлового оператора

3.3.1.1 Команда SELECT FILE

Функции команды:

Данная функция выбирает файл в соответствии с методами, описанными в п.8.4. После успешного выбора указатели записи и указатель текущего тега не определены.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	0X	Байт класса
INS	A4	SELECT FILE
P1	'xx'	Контрольный параметр для сравнения P1
P2	'xx'	Контрольный параметр для сравнения P2
Lc	'xx'	Длина последующего поля данных или пустое поле
Data	'xxxxxx'	ID файла, имя DF или путь к файлу в соответствии с P1
Le	'00'	

Таб.14 Сообщение команды SELECT FILE

3.3.1.2 Команда DELETE FILE

Функции команды:

Данная команда инициирует удаление EF, на который указывает ссылка, непосредственно под текущим DF или DF с полным поддеревом.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	‘0x’	Байт класса
INS	‘E4’	Команда DELETE FILE
P1	'00'	Контрольный параметр для сравнения P1
P2	'00'	Контрольный параметр для сравнения P2
Lc	'xx'	Длина поля данных
Data	'xxxxxx'	Данные, направленные UICC
Le		Не представлено

Таб.15 Сообщение команды DELETE FILE

3.3.1.3 Команда CREATE FILE

Функции команды:

Данная функция позволяет создать новый файл в текущем каталоге.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	‘0x’	Байт класса
INS	‘E0’	Команда CREATE FILE
P1	'00'	Контрольный параметр для сравнения P1
P2	'00'	Контрольный параметр для сравнения P2
Lc	'xx'	Длина поля данных
Data	'xxxxxx'	
Le	'00'	

Таб.16 Сообщение команды CREATE FILE

3.3.1.4 Команда UPDATE BINARY

Функции команды:

Данная функция считывает последовательность байтов из текущего прозрачного элемента EF. Выполнение этой функции допускается только при соблюдении условия доступа READ для данного элемента EF.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	0x	Байт класса
INS	D6	Команда UPDATE BINARY
P1	'xx'	Контрольный параметр для сравнения P1
P2	'xx'	Контрольный параметр для сравнения P2
Lc		Не представлено
Data		Не представлено
Le		Не представлено

Таб.17 Сообщение команды UPDATE BINARY

3.3.1.5 Команда READ BINARY

Функции команды:

Данная функция считывает строку байтов из текущего прозрачного элемента EF. Выполнение этой функции допускается только при соблюдении условия доступа READ для данного элемента EF.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	0x	Байт класса
INS	'B0'	Команда READ BINARY
P1	'xx'	Контрольный параметр для сравнения P1
P2	'xx'	Контрольный параметр для сравнения P2
Lc		Не представлено
Data		Не представлено
Le	'xx'	Количество байтов для чтения

Таб.18 Сообщение команды READ BINARY

3.3.1.6 Команда UPDATE RECORD

Функции команды:

Данная функция обновляет текущий прозрачный элемент EF последовательностью байтов. Выполнение этой функции допускается только при соблюдении условия доступа UPDATE для данного EF. Операцию обновления можно рассматривать как замену последовательности, уже содержащейся в EF, на последовательность, указанную в команде обновления.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	0x	Байт класса
INS	DC	Команда UPDATE RECORD
P1	'xx'	Контрольный параметр для сравнения P1
P2	'xx'	Контрольный параметр для сравнения P2
Lc	'xx'	Длина последующего поля данных
Data	'xx'	Строка данных для обновления
Le		Не представлено

Таб.19 Сообщение команды UPDATE RECORD

3.3.1.7 Команда READ RECORD

Функции команды:

Данная функция считывает одну полную запись из текущего элемента EF линейного типа с записями фиксированной длины или циклического типа. Считываемая запись определяется указанными ниже режимами. Выполнение этой функции допускается только при соблюдении условия доступа READ для данного EF. В случае неудачного выполнения функции READ RECORD указатель записи не изменяется.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	0x	Байт класса
INS	B2	Команда READ RECORD
P1	'xx'	Номер записи
P2	'xx'	Контрольный параметр для сравнения P2
Lc		Не представлено
Data		Не представлено
Le	'xx'	Количество байтов для чтения

Таб.20 Сообщения команды READ RECORD

3.3.2 Команда AUTHENTICATION

Функции команды:

Данная функция используется в контексте безопасности GSM или 3G во время процедуры аутентификации USIM на его НЕ и наоборот. Кроме того, вычисляются ключ шифрования и ключ целостности. Для выполнения команды USIM использует ключ аутентификации абонента K, который хранится USIM.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	'00'	Байт класса
INS	'88'	Команда AUTHENTICATION
P1	'xx'	Контрольный параметр для сравнения P1
P2	'xx'	Контрольный параметр для сравнения P2
Lc	'xx'	Длина поля данных
Data	'xxxxxx'	
Le	'00'	

Таб.21 Сообщения команды AUTHENTICATION

3.3.3 Команда CAT

3.3.3.1 Команда TERMINAL PROFILE

Функции команды:

Данная функция используется терминалом для передачи своих возможностей CAT приложениям, присутствующим UICC.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	80	Байт класса
INS	10	Команда Terminal Profile
P1	‘00’	Контрольный параметр для сравнения P1
P2	‘00’	Контрольный параметр для сравнения P2
Lc	'xx'	Длина поля данных
Data	'xxxxxx'	
Le		Не представлено

Таб.22 Сообщения команды TERMINAL PROFILE

3.3.3.2 Команда ENVELOPE

Функции команды:

Данная функция используется для передачи информации CAT из UE в UICC.

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	80	Байт класса
INS	C2	Команда ENVELOPE
P1	'00'	Контрольный параметр для сравнения P1
P2	'00'	Контрольный параметр для сравнения P2
Lc	'xx'	Длина поля данных
Data	'xxxxxx'	
Le	'00'	Пустая или максимальная длина данных ответа

Таб.23 Сообщения команды ENVELOPE

3.3.3.3 Команда FETCH

Функции команды:

Данная функция используется для передачи упреждающей команды из UICC в терминал (например, из приложения CAT).

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	80	Байт класса
INS	12	Команда FETCH
P1	'00'	Контрольный параметр для сравнения P1
P2	'00'	Контрольный параметр для сравнения P2
Lc		Не представлено
Data		Не представлено
Le	'xx'	Объем ожидаемых данных

Таб.24 Сообщения команды FETCH

3.3.3.4 Команда TERMINAL RESPONSE

Функции команды:

Данная функция используется для передачи с терминала в UICC ответа на ранее полученную упреждающую команду (например, команда CAT).

Сообщение команды кодируется в соответствии со следующей таблицей:

Код	Значение	Описание
CLA	80	Байт класса
INS	14	Команда TERMINAL RESPONSE
P1	'00'	Контрольный параметр для сравнения P1
P2	'00'	Контрольный параметр для сравнения P2
Lc	'xx'	Длина поля данных
Data	'xxxxxx'	
Le		Не представлено

Таб.25 Сообщение команды TERMINAL RESPONSE

4. Условия реализации программы

4.1 Состав инструментов разработки

Инструменты разработки в основном состоят из двух компонентов: аппартного и программного обеспечения. Аппаратное обеспечение включает в себя блоки питания, USB-кабели, платы расширения и комплекты разработки. Программное обеспечение включает в себя MDK, пакеты установки программного обеспечения, USB-драйверы, вспомогательные инструменты Eclipse IDE.

1. Аппаратные средства разработки

Основными аппаратными компонентами для инструментов разработки являются блок питания, USB-кабели, платы расширения и набор инструментов.

1) Источник питания: источник питания постоянного тока 5 В, 2 А;

2) USB-соединение: порты A и B, используемые для подключения ПК к интерфейсу JTAG разработчика;

3) Плата расширения 7816: обеспечивает аппаратный интерфейс для протокола ISO7816 на чипе, позволяя подключаться к устройству чтения карт памяти через интерфейс 7816 для целей отладки и разработки;

4) Плата разработки: предназначена для поддержки разработки встроенных программ для микросхем, имеет интерфейс 7816 и интерфейс JTAG для подключения к ПК.

2. Программное обеспечение для разработки инструментов

Основное программное обеспечение для разработки включает в себя Eclipse, MDK, установочные пакеты программного обеспечения и USB-драйверы.

1) MDK: используйте MDK версии 5.23 или более поздней (при использовании IDE Keil uVision 5)

2) Установочный пакет программного обеспечения: пакеты файлов, связанные с программированием и отладкой микросхем.

3) Файл на USB-накопителе.

4) Eclipse IDE

4.2 Аппаратные харектеристики чипа

Чип работает на процессере ARM.

RAM	10.5 КБ однопортовый RAM
Flash	340 КБ для хранения программ и данных, поддерживает загрузку. Способен выполнять более 100,000 циклов записи-считывания, поддерживает запись страниц, символов и запись половины страницы.
Интерфейс	Интерфейс 7816 совместим с ISO/IEC 7816 T=0/1 и поддерживает такие функции, как автоматическая передача 0x60.
Система синхронизации	Частота системы синхронизации может быть настроена как 30 МГц, 15 МГц или 7.5 МГц, при этом по умолчанию установлено значение 7.5 МГц.
Сброс системы	Полный сброс: сброс при включении питания, сброс при обнаружении неисправностей VDE, сброс при обнаружении неисправностей GDE, сброс по 7816 RST и т.д.
	Мягкое подавление: ручное уменьшение
Рабочий диапазон	Рабочее напряжение: от 1,62 В до 5,5 В
	Рабочий ток: ≤10 мА при частоте 30 МГц, при температуре 25°C
	Внешние тактовые частоты: от 1 МГц до 10 МГц
	Температура соединения: от -25°C до 85°C

Таб.26 Характеристики чипа

5. Примеры работы с приложением

5.1 Загрузка ПО

(1) Используйте Keil для компиляции и вывода Hex-файла;

(2) Преобразуйте Hex-файл в формат инструкций, совместимый с ChipLoader.

(3) Отправьте преобразованные инструкции для загрузки ПО в чип.

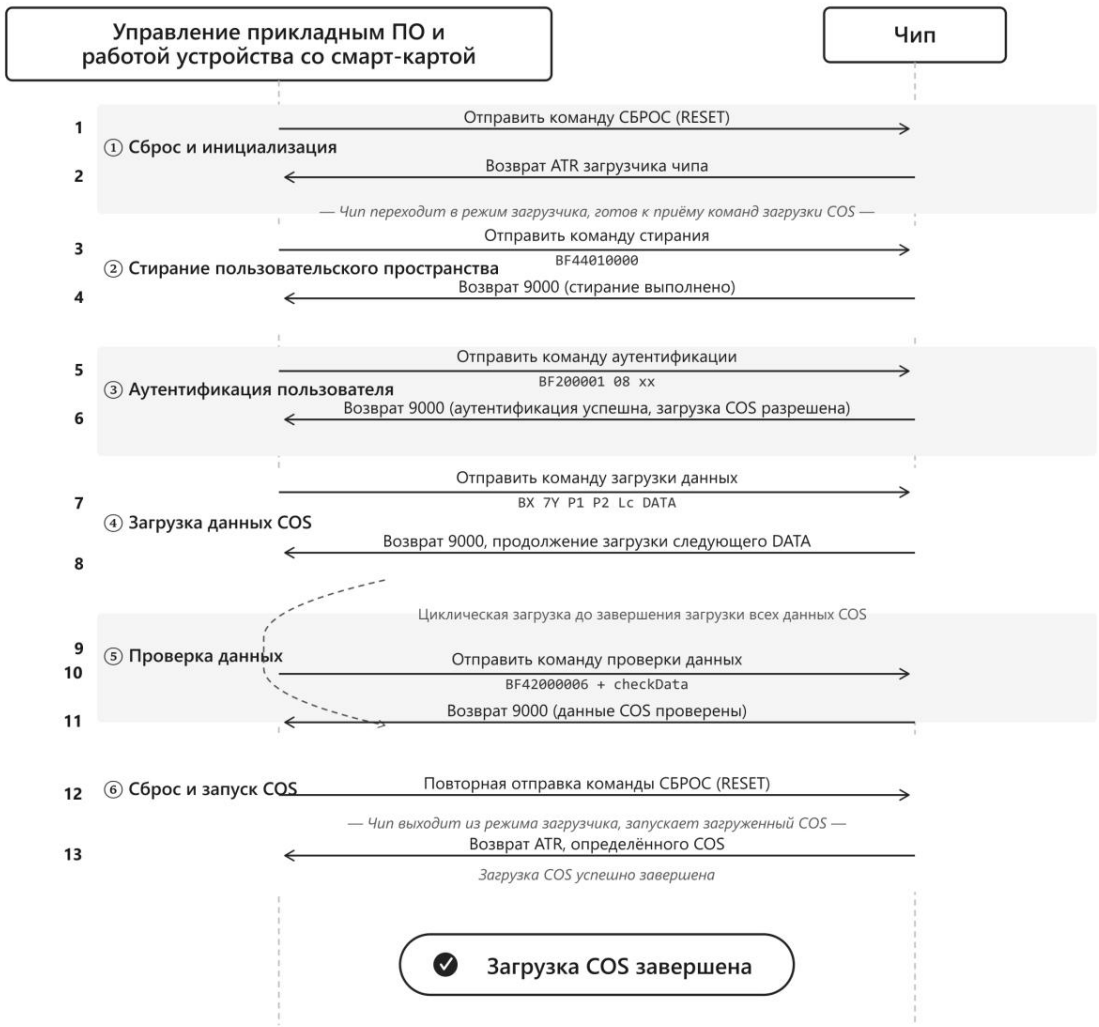

Рис.5.1 Загрузка ПО

Схема последовательности загрузки ПО состоит из шести фаз:

Фаза	Этап	Содержание
Фаза 1: Сброс и инициализация карты	1-2	Устройство отправляет команду сброса; карта возвращается в режим загрузки ATR и переходит в режим загрузки.
Фаза 2: Удаление пользовательского пространства	3-4	Отправьте BF44010000 для стирания пользовательского пространства и дождитесь ответа SW9000 для подтверждения успеха операции.
Фаза 3: Аутентификация личности пользователя	5-6	Отправьте BF200001 08 xx для аутентификации; ответ SW9000 означает подтверждение.
Фаза 4: загрузка ПО	7-9	Загрузите ПО через петлю записи данных BX 7Y P1 P2; продолжайте процесс после возврата каждого комплекта SW9000, пока не будут загружены все данные.
Фаза 5: Проверка данных	10-11	Отправьте BF42000006 + checkData для проверки целостности загруженных данныз; возврат SW9000 указывает на успешную загрузку.
Фаза 6: Сброс настроек карты	12-13	Повторный сброс: карта выходи из режима загрузки и возвращается к режиму ATR, определенному ПО; загрузка успешно завершена.

Таб.27 Загрузка ПО

5.2 Инициализация ПО

Преобразуйте файл профиля в исполняемый скрипт, а затем передайте данные профиля на карту в виде команд APDU для завершения инициализации ПО.

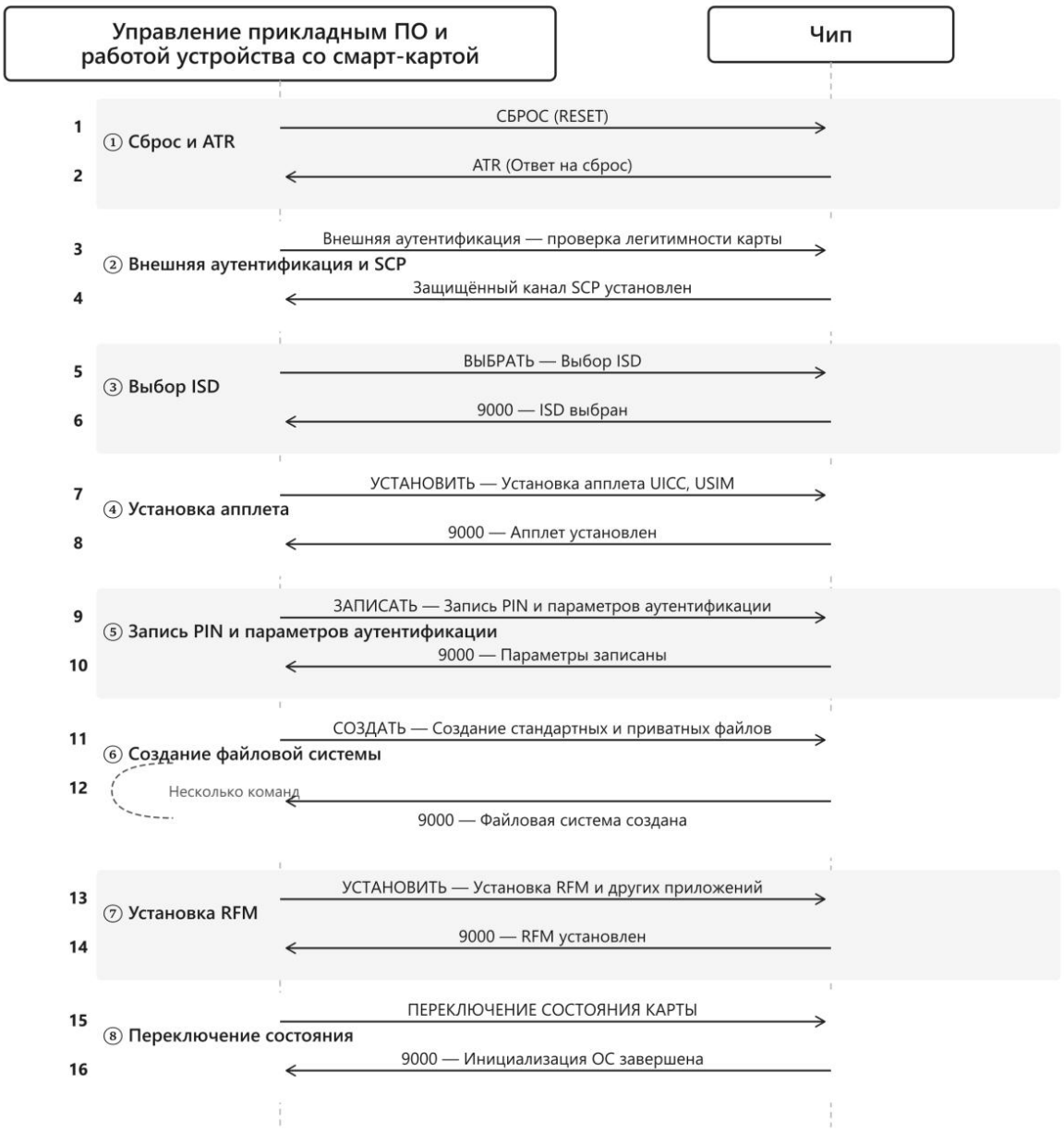

Рис. 5.2 Инициализация ПО

Схема последовательности инициализации ПО состоит из восьми фаз:

Фаза	**Этап**	**Содержание**
Фаза 1: Сброс карты и ATR	1-2	Сброс → Карта возвращает ATR
Фаза 2: Внешняя аутентификация и канал SCP	3-4	Устройство инициирует внешнюю аутентификацию → После аутентификации карты защищенный канал SCP установлен.
Фаза 3: Выбор ISD	5-6	Выбор ISD → Карта возвращает SW9000
Фаза 4: Установка апплета UICC/USIM	7-8	Установка апплета UICC/USIM → Карта возвращает SW9000
Фаза 5: Написание PIN-кода и аутентификация параметров	9-10	Укажите параметры инициализации, такие как PIN-код и аутентификация → Карта возвращает SW9000
Фаза 6: Создание файловой системы	11-12	Последовательно отправьте несколько команд для создания файловой системы → Карта возвращает SW9000
Фаза 7: Установка приложения RFM	13-14	Установите приложение RFM → Карта возвращает SW9000
Фаза 8: Переключение состояний и завершение	15	Значение статуса переключения карты после инициализации ПО

Таб.28 Последовательность загрузки ПО

5.3 Загрузка приложения

Преобразование приложения в формат команд APDU и его загрузка на чип:

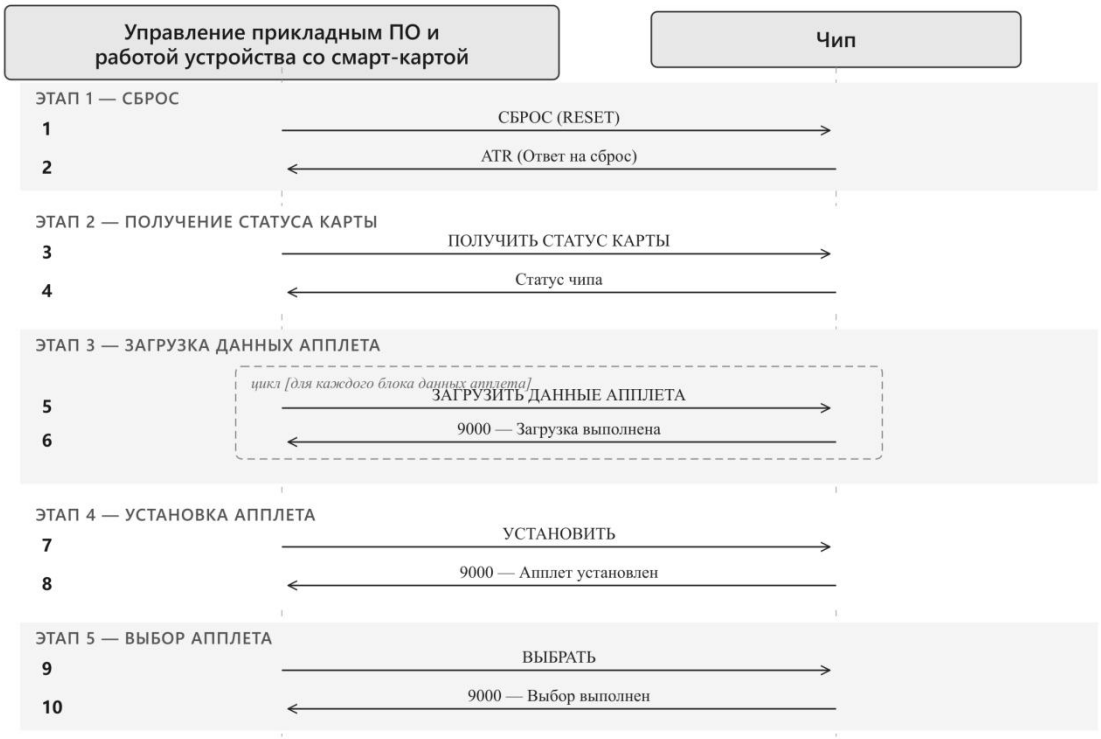

Рис. 5.3 Загрузка приложения

Фаза	**Этап**	**Содержание**
Фаза 1: Сброс карты и ATR	1-2	Устройство отправляет запрос на сброс → Возврат карты ATR
Фаза 2: Получить статус карты	3-4	Устройство получает статус карты → Карта возвращает статус ПО
Фаза 3: Загрузка данных апплета с устройства	5-6	Загрузка данных апплета → Карта возвращает SW9000
Фаза 4: Отправка инструкций по установке	7-8	Установка апплета → Карта возвращает SW9000
Фаза 5: Отправка инструкций по выбору	9-10	Выбор апплета → Карта возвращает SW9000

Таб.29 Загрузка приложения