Научная статья
УДК 004.413
DOI: 10.26583/bit.2025.2.12

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ ePKI ДЛЯ БЕЗОПАСНОГО ОБНОВЛЕНИЯ ВСТРОЕННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДОВЕРЕННЫХ ПРОГРАММНО-АППАРАТНЫХ КОМПЛЕКСОВ

Олег Н. Дьяков[1], Данила С. Беляков[2], Евгений О. Калинин[3]
[1]АО «Аладдин Р.Д.», ул. Докукина, 16, стр. 1, Москва, 129226, Россия,
[2,3]Томский государственный университет систем управления и радиоэлектроники, пр–кт Ленина, 40, Томск, 634050, Россия
[1]O.Dyakov@aladdin.ru, https://orcid.org/0009-0004-5696-8595
[2]bds2@csp.tusur.ru, https://orcid.org/0000-0002-6111-455X
[3]keo@csp.tusur.ru, https://orcid.org/0000-0003-4952-4043

Аннотация. Создание доверенных информационных систем требует построения доверенных цепочек поставок аппаратных и программных компонентов. С точки зрения информационных технологий доверие – это уверенность в том, что все компоненты системы, процессы и данные являются подлинными, целостными и безопасными, а сервисы являются доступными. Одним из общепринятых приёмов обеспечения доверия в информационных системах является построение доверенных цепочек на основе использования технологии инфраструктуры открытых ключей (public key infrastructure, PKI). В статье рассматриваются аспекты создания доверенной цепочки поставки на основе PKI при выполнении операции обновления встраиваемого программного обеспечения доверенных программно-аппаратных комплексов. Дано общее описание стадий процесса обновления встроенного программного обеспечения, приведены форматы и основные операции по подготовке и загрузке образа встроенного программного обеспечения. Приведены оценки затрат вычислительных ресурсов, необходимые для реализации описанных операций. Следует отметить, что рассматриваемая технология может также использоваться для обновления системного и прикладного программного обеспечения. Данная статья будет интересна специалистам по разработке доверенной электронно-компонентной базы, доверенных программно-аппаратных комплексов (ПАК), системным интеграторам КИИ, разработчикам сквозных технологических процессов по разработке и производству доверенных устройств и информационных систем.

Ключевые слова: *PKI, доверенный ПАК, информационная безопасность, встроенное программное обеспечение, доверенная цепочка поставки, корень доверия, обновление встроенного ПО.*

Для цитирования: *Дьяков, Олег Н.; Беляков, Данила С.; Калинин, Евгений О. Использование технологии ePKI для безопасного обновления встроенного программного обеспечения доверенных программно-аппаратных комплексов. Безопасность информационных технологий, [S.l.], т. 32, № 2, с. 152–177. 2025. ISSN 2074-7136. URL: https://bit.spels.ru/index.php/bit/article/view/1787. DOI: 10.26583/bit.2025.2.12.*

Scientific article

USING ePKI TECHNOLOGY TO SECURELY UPDATE OF EMBEDDED SOFTWARE OF TRUSTED HARDWARE AND SOFTWARE SYSTEM

Oleg N. Dyakov[1], Danila S. Belyakov[2], Evgeny O. Kalinin[3]
[1]JSC "Aladdin R.D.", Dokukina str., 16, building 1, Moscow, 129226, Russia,
[2,3]Tomsk State University of Control Systems and Radioelectronics, Lenin Ave., 40, Tomsk, 634050, Russia

[1]*O.Dyakov@aladdin.ru, https://orcid.org/0009-0004-5696-8595*
[2]*bds2@csp.tusur.ru, https://orcid.org/0000-0002-6111-455X*
[3]*keo@csp.tusur.ru, https://orcid.org/0000-0003-4952-4043*

Abstract. The creation of trusted information systems requires the establishment of trusted supply chains for both hardware and software components. From the perspective of information technology, trust means confidence that all system components, processes, and data are authentic, integral, secure. One of the widely accepted approaches to ensuring trust in information systems is the construction of trusted chains based on Public Key Infrastructure (PKI) technology. This paper examines the aspects of building a trusted supply chain based on PKI in the context of updating embedded software in trusted hardware-software systems. It provides a general description of the steps in the firmware upgrade process, describes the formats and basic operations for preparing and loading the firmware image, and estimates the computational resources required to perform these operations. It should be noted that the described technology can also be applied to update system and application software. This publication will be of interest to specialists in the development of trusted electronic components, trusted devices, system integrators of critical information infrastructure, developers of end-to-end technological processes for the development and production of trusted devices and information systems.

Keywords: *PKI, trusted hardware-software system, information security, embedded software, trusted supply chain, root of trust, embedded software update.*

For citation: *Dyakov, Oleg N.; Belyakov, Danila S.; Kalinin, Evgeny O. Using ePKI technology to securely update of embedded software of trusted hardware and software system. IT Security (Russia), [S.l.], v. 32, no. 2, p. 152–177, 2025. ISSN 2074-7136. URL: https://bit.spels.ru/index.php/bit/article/view/1787. DOI: 10.26583/bit.2025.2.12.*

Введение

Для обеспечения функциональной и информационной безопасности доверенных программно-аппаратных комплексов (ПАК) критически важно корректно реализовать процессы доверенной поставки, установки и обновления встроенного, системного и прикладного программного обеспечения.

Цепочка разработки, дистрибуции, установки и обновления ПО представляет собой распределённый процесс с множеством участников. Ключевым аспектом для обеспечения безопасности работы доверенной системы является обеспечение целостности этой цепочки.[1]

Основные задачи обновления программного обеспечения включают:

- аутентификацию устройства системой управления;
- аутентификацию программного обеспечения устройством перед установкой или обновлением;
- проверку полномочий устройства на использование ПО.

Для решения этих задач хорошо подходят технологии, основанные на инфраструктуре открытых ключей ((public key infrastructure, PKI).[2,3,4]

В контексте межмашинного взаимодействия используются технологические или машинные сертификаты. Механизмы поддержки PKI должны быть встроены в доверенные устройства и цепочку поставки ПО на самых ранних стадиях. В дальнейшем будем использовать термин ePKI, чтобы отличать эту технологию от традиционного применения PKI в поддержке инфраструктуры электронной подписи.

[1]TCG Guidance for Secure Update of Software and Firmware on Embedded Systems. Version 1.0, Revision 72, February 10, 2020.
[2]RFC 9019. A Firmware Update Architecture for Internet of Things. April 2021.
[3]GPC_SPE_134. GlobalPlatform Open Firmware Loader for Tamper Resistant Element Version 1.3, April 2021.
[4]TCG, DICE Attestation Architecture. Version 1.1, Revision 0.17, April 2023.

В данной работе рассматриваются:
- базовый сценарий обновления программного обеспечения,
- основные угрозы цепочке поставки программного обеспечения,
- вопросы обеспечения корня доверия,
- основные участники цепочки поставки,
- схема системы сертификатов,
- принцип работы безопасного загрузчика,
- формат образа программного обеспечения,
- основные операции при подготовке и установке программного обеспечения,
- результаты экспериментальной реализации системы доверенной установки и обновления программного обеспечения.

1. Базовый сценарий безопасного обновления программного обеспечения

Для предотвращения кибератак все этапы процесса разработки и развёртывания программного обеспечения должны быть должным образом защищены. На рис. 1 показаны основные стадии процесса безопасного обновления программного обеспечения[5]. Первичная установка (инсталляции) программного обеспечения является частным случаем обновления программного обеспечения. Далее будем использовать термин обновление в том числе и для описания процесса первичной установки.

Необходимость постоянных обновлений и повторных оценок безопасности обозначена элементом «Запрос Обновления» в левой части рис. 1.

Рис. 1. Базовый сценарий обновления встроенного программного обеспечения

Программное обеспечение, подготовленное для загрузки в память вычислительного устройства, называется образом (image).

1.1. Безопасная разработка программного обеспечения

Обеспечение безопасности процесса разработки ПО, начинается с внедрения и согласования проверенных методов разработки программного обеспечения. Рекомендуемые практики включают в себя [1, 2]:

– обеспечение безопасности на всех этапах процесса разработки;

– тщательный анализ угроз и выбор мер противодействия при проектировании и обслуживании устройств;

– применение лучших практик обеспечения безопасности и их постепенное совершенствование для устранения новых и возникающих угроз;

– согласование измеримых требований безопасности, которые должны быть выполнены до выпуска устройства;

– обеспечение безопасности среды разработки;

– использование надёжных инструментов, языков и библиотек;

– тщательная проверка ввода и обработка ошибок;

[5]TCG Guidance for Secure Update of Software and Firmware on Embedded Systems. Version 1.0, Revision 72, February 10, 2020.

– обучение всех участников разработки вопросам безопасности;
– создание надёжного процесса реагирования на инциденты.

Дополнительные методы для более высоких уровней безопасности могут включать:

– установление строгой физической безопасности (например, шлюзы) и сетевой безопасности (например, изоляцию сетей) для компьютеров разработчиков;
– независимые проверки безопасности исходного кода;
– независимые аудиты безопасности и сертификация продукта и процесса разработки;
– настройка системы управления информационной безопасностью;
– независимый анализ безопасности внешних зависимостей (инструментов, библиотек и т.д.);
– тестирование на проникновение;
– нагрузочное тестирование;
– автоматизированное тестирование, включая фаззинг, для поиска ошибок и уязвимостей безопасности;
– статический анализ двоичного или исходного кода для поиска ошибок и небезопасных методов кодирования;
– строгий контроль персонала.

1.2. Подпись и зашифрование обновлений

Создание безопасного обновления программного обеспечения сложнее, чем создание программного обеспечения с нуля. Помимо сложности, связанной с обновлением уже установленного программного обеспечения (например, обновление файлов, созданных в более ранней версии), обновление программного обеспечения должно быть подписано (известное как «подпись кода»), чтобы получатель мог проверить в его происхождение и целостность перед инсталляцией.

Подписание кода – один из наиболее важных шагов в процессе безопасного обновления программного обеспечения. Успешная атака на этом этапе позволяет злоумышленнику распространять вредоносный код.

В случае неудачной попытки обновления встроенного программного обеспечения необходимо выполнить откат к предыдущей версии прошивки. Управление откатом должно выполняться под строгим контролем системы безопасности: злоумышленник может повторно использовать легитимную предыдущую версию с известной уязвимостью. Если нет механизма предотвращения несанкционированного отката, уязвимость может быть использована, и злоумышленник потенциально может получить контроль над системой. Обновления должны включать последовательный номер версии для поддержки защиты от отката.

При создании безопасной криптографической системы защиты обновлений рекомендуемые методы включают в себя [3]:

– тщательный анализ угроз и выбор мер противодействия во время проектирования и обслуживания;
– применение лучших практик обеспечения безопасности;
– использование отдельных ключей и сертификатов для подписи производственного кода и кода разработки;
– использование надёжных, проверенных криптографических алгоритмов и инструментов;
– выбор ключа подписи с достаточной надёжностью;

– обеспечение того, чтобы продукты клиентов принимали только код, подписанный производственным ключом;

– тщательная проверка всех центров сертификации и других сторон, доверенных в процессе подписания;

– использование информации о версии для предотвращения отката;

– разработка процесса отзыва исправлений (в случае, если они окажутся плохими) и ключей (в случае, если они скомпрометированы) и проверка этой информации об отзыве перед обновлением;

– планирование истечения срока действия ключа и его смены;

– разработка гибких криптографических схем с возможностью использования в будущем более стойких алгоритмов (поскольку алгоритмы со временем становятся слабее, например, с переходом на квантово-устойчивые криптографические алгоритмы).

Дополнительные методы для более высоких уровней безопасности могут включать:

– размещение рабочего ключа подписи в аппаратном модуле безопасности (HSM, SE)[6] для предотвращения его извлечения;

– использование выделенного изолированного компьютера для подписи производственного кода;

– строгий контроль физического и логического доступа к системе подписи производственного кода;

– использование нескольких участников для авторизации подписи производственного кода или получения доступа к системе подписи производственного кода;

– тщательная проверка кода на наличие признаков компрометации перед его подписанием.

Хотя эти меры могут показаться избыточными, любые недостатки в безопасности подписи кода могут дать злоумышленникам полную свободу действий в обновляемых системах, поэтому данные меры оправданы.

1.3. Надёжная доставка

После подписания обновления программного обеспечения его необходимо доставить до устройства, которое будет обновляться. Если распределительная сеть работает с перебоями, устройствам может потребоваться собирать обновления из данных, полученных через нерегулярные промежутки времени, когда канал связи становится доступным.

Если сети имеют ограниченную пропускную способность и/или высокую задержку, обновления могут содержать только различия между текущим состоянием устройства и обновлённым состоянием.

Помимо фундаментальной проблемы отправки больших файлов на множество конечных точек, необходимо решать такие задачи, как обеспечение аутентичности и надёжности службы распространения.

Если установка исправлений задерживается слишком долго, конечные точки могут подвергнуться большему риску компрометации.

При построении надёжной системы распространения обновлений рекомендуемые методы включают в себя [4, 5]:

– тщательный анализ угроз и выбор мер противодействия во время проектирования и обслуживания;

[6]NIST IR 8320 Hardware-Enabled Security. May 2022. DOI: https://doi.org/10.6028/NIST.IR.8320.

– применение лучших практик обеспечения безопасности;

– защита связи с помощью тщательно проверенных протоколов безопасности;

– установление аутентичности и надёжности службы распространения, как правило, через доверенную третью сторону;

– автоматизация распространения и установки обновлений программного обеспечения для минимизации риска того, что обновления не будут должным образом распространяться и устанавливаться. Необходимо обеспечить административный контроль процесса со стороны организаций, которым необходимо контролировать и управлять обновлением программного обеспечения;

– разработка механизмов распространения обновлений программного обеспечения таким образом, чтобы избежать перегрузки сетей или серверов;

– обеспечение административного доступа к серверам обновлений для обеспечения распространения только авторизованных обновлений;

– использование проверенных механизмов распространения обновлений программного обеспечения для снижения административной нагрузки;

– предоставление возможности администраторам планировать обновления систем своей организации;

– обнаружение и противодействие атакам типа «отказ в обслуживании» на серверах обновлений или на механизме распространения.

Дополнительные методы для более высоких уровней безопасности могут включать:

– размещение ключей безопасности связи в аппаратном модуле безопасности (HSM, SE) для предотвращения их извлечения;

– аутентификация конечных точек для определения того, какие обновления им разрешено получать;

– отслеживание установки обновлений, чтобы гарантировать обновление конечных точек;

– оповещение администраторов, если обновления не могут быть установлены на некоторых конечных точках;

– обеспечение администраторам системы возможности противодействия попыткам пользователей остановить обновления или установить обновления без одобрения администратора;

– предполагая, что серверы обновлений могут быть скомпрометированы, необходимо обеспечить конечным точкам возможность проверки всех обновлений, загруженных с серверов, а также должны быть включены другие механизмы для проверки правильности загрузки и установки обновлений.

1.4. Безопасная установка обновлений

Существует множество сложных аспектов эксплуатации и безопасности при выполнении обновления программного обеспечения.

Время простоя является основной проблемой для некоторых систем, особенно тех, которые контролируют критически важные процессы. По этой причине (а также для управления рисками) владельцы устройств должны иметь возможность планировать обновления, по крайней мере, для критически важных устройств.

Как альтернативный вариант, устройствам может потребоваться принимать обновления, не прекращая своей работы. Обычно для этого требуется обеспечение резервирования ресурсов устройства, чтобы часть устройства можно было обновить, в то время как остальная часть устройства продолжает работать, а также способ определения,

когда устройство может безопасно переключиться с существующей функциональности на обновлённую функциональность.

Выполнение обновлений в фоновом режиме может позволить перезаписать зашифрованные данные обновлёнными зашифрованными данными, избегая необходимости расшифровки во время обновления. Ещё одной альтернативой является использование одного или нескольких резервных устройств во время обновления основного устройства.

По разным причинам обновление программного обеспечения может завершиться ошибкой (сбой питания во время обновления, невозможность преобразования данных или доступа к внешним ресурсам и т.д.). Устройства должны иметь возможность восстанавливаться после таких сбоев, например, путём возврата в режим восстановления.

Обновления часто требуют внесения изменений в данные, хранящиеся во встроенной системе. Например, может потребоваться изменить при установке новой версии программного обеспечения формат базы данных или формат файла конфигурации. При ограниченных ресурсах на устройстве или желании обеспечить плавное восстановление в случае сбоя обновления, любые сложные преобразования данных лучше всего выполнять путём отправки данных в облако или на внешний сервер для резервного копирования и преобразования.

Рекомендуемые методы безопасной установки обновлений включают в себя [6]:

– тщательный анализ угроз и выбор мер противодействия во время проектирования и обслуживания;

– применение лучших практик обеспечения безопасности;

– ограничение привилегий и действий по установке обновлений. Использование тщательно спроектированного и строго контролируемого механизма обновления, включая защиту ключей и кода, которые имеют решающее значение для процесса обновления;

– проверку обновлений перед установкой на предмет: аутентификации и авторизации источника, целостности, соответствия программного обеспечения устройству и любых других необходимых полномочий (например, владельца устройства);

– избежание атак Time Of Check To Time Of Use [7] и подобных состояний гонки, например за счёт гарантии того, что никакие другие операции не могут выполняться во время обновления;

– использование процесса восстановления после неудачных или вредоносных обновлений программного обеспечения;

– обеспечение возможности администраторам планировать обновления систем своей организации.

Дополнительные методы для более высоких уровней безопасности могут включать:

– использование зашифрованных обновлений для повышения сложности обратного проектирования для обнаружения и использования уязвимостей в коде;

– использование физически защищённых от несанкционированного доступа устройств (HSM, SE) для хранения ключей и выполнения кода, которые имеют решающее значение для процесса обновления;

– тщательное проектирование, тестирование и защита коня доверия (Root of Trust, RoT)[7] , чтобы гарантировать, что он может обнаруживать вредоносные входные данные, предотвращая тем самым компрометацию программного обеспечения;

– обновления с подписью администратора, при этом даже подлинные обновления не могут применяться без разрешения лиц (таких как администратор платформы), имеющих законный интерес в определении того, следует ли и когда применять обновление;

– обеспечение невозможности использования процесса восстановления, как способа отката к уязвимой прошивке.

1.5. Проверка и аттестация устройства после обновления

Системы распространения обновлений должны определить, было ли устройство правильно обновлено, предоставляет ли оно все свои API и имеет ли доступ к сетям. Этот процесс проверки может включать ручное или автоматическое тестирование функциональности и целостности.

В рамках процесса обновления или в процессе эксплуатации владельцам и производителям устройств (и другим сторонам) может потребоваться проконтролировать версию и целостность программного обеспечения, установленного на устройстве. Данный процесс называется аттестацией [8].

Для обеспечения аттестации, необходимо выполнить процедуру измеренной загрузки (запуска). При такой загрузке криптографические хэши образа программного обеспечения, рассчитываются (измеряются) корнем доверия. Данный вариант загрузки увеличивает сложность устройства, но упрощает инфраструктуру аттестации.

После измеренной загрузки можно выполнить удалённую аттестацию. Измерения, выполненные во время загрузки, можно безопасно отправить на сервер аттестации. Сервер аттестации может сравнить измерения с набором «золотых измерений», чтобы определить, какая прошивка или программное обеспечение работает на устройстве. Устройства со старым программным обеспечением можно изолировать, чтобы защитить их от заражения, и обновить до новейших версий. Устройства с вредоносной или неизвестным программным обеспечением можно изолировать для экспертизы.

При использовании удалённой аттестации рекомендуемые методы включают в себя:

– тщательный анализ угроз и выбор мер противодействия во время проектирования и обслуживания;

– следование практикам безопасного жизненного цикла разработки программного обеспечения;

– использование измеренной загрузки (запуска) и удалённой аттестации, чтобы позволить серверам удалённого управления обнаруживать версии программного обеспечения устройств и управлять ими.

Дополнительные методы для более высоких уровней безопасности могут включать:

– внедрение контроля доступа к сети для изоляции устройств со старыми, вредоносными или неизвестными версиями программного обеспечения;

– использование локальных политик, для обнаружения неправильных версий встроенного ПО и программного обеспечения и запуска восстановления устройства или, по крайней мере, предотвращения вредоносного поведения.

[7]GlobalPlatform Technology. Root of Trust Definitions and Requirements. Version 1.1.1, Public Release, June 2022.

1.6. Угрозы и меры противодействия

В табл. 1 приведены угрозы жизненного цикла обновления программного обеспечения и меры противодействия[8], [9, 10, 3].

Таблица 1. Угрозы и меры противодействия

	Контрмеры				
Угрозы	Безопасная разработка	Безопасная подпись обновлений	Надёжная доставка	Безопасная инсталляция обновлений	Удалённая аттестация
Компрометация цепочки поставок (например, недоверенный компилятор)	X				
Компрометация ключей подписи		X			
Устаревание криптоалгоритма		X			
Отказ в обслуживании при распространении обновлений			X		
Компрометация распространения обновлений		X		X	X
Обновление без местного одобрения				X	X
Неудачное обновление, устройство в небезопасном состоянии				X	X
Эксплуатация уязвимости в процессе обновления	X			X	

2. Корень доверия

Для эффективной реализации функций поддержки обновления встроенного программного обеспечения рекомендуется использовать технологию инфраструктуры открытых ключей (Public Key Infrastructure, PKI) [11].

Основная цель любой PKI – создание защищённой среды, используемой участниками, приложениями и оборудованием, через управление ключами и сертификатами. Сертификаты и ассиметричные ключи составляют основу PKI, с помощью которой может быть поддержано доверие к конечной точке[9].

Для поддержки PKI процессор (микроконтроллер) и программное обеспечение должны реализовать: алгоритмы ассиметричной криптографии, вычисление хэш функций, процедуры безопасного хранения ключей [12, 13].

В доверенных ПАК поддержка PKI должна быть встроена на уровне аппаратной платформы (Embedded PKI или ePKI).

[8]RFC 9124. A Manifest Information Model for Firmware Updates in Internet of Things (IoT) Devices.
[9]RFC 5280. Internet X.509 PKI Certificate and CRL Profile.

Наличие цифрового сертификата у компонента системы означает, что данный компонент имеет пару ассиметричных ключей, которые могут использоваться для строгой аутентификации, проверки целостности, защиты от клонирования, шифрования информации.

Технология ePKI позволяет создавать цепочки доверия (рис. 2), которые предполагают проверку целостности и аутентичности последовательно запускаемых компонент программного обеспечения системы [14, 15]. Каждый предыдущий элемент цепочки доверия проверяет (измеряет) следующий элемент, при успешном выполнении процедуры проверки выполняется передача управления следующему элементу цепочки. В начале цепочки доверия находится корень доверия (RoT) – код и данные, использующие некоторый вычислительный механизм, который стартует первым и гарантирует, что устройство запускается правильно, а его компоненты и операционная система защищены от фальсификации.

Основные свойства корня доверия (RoT):

– RoT должен включать код, который выполняется первым при инициализации вычислительного ядра во время холодного старта платформы;

– код RoT должен быть неизменяемым или изменения должны контролироваться только уникальным идентифицируемым владельцем;

– RoT должен быть проверен независимой стороной (лабораторией тестирования), способной оценить вычислительный механизм, код и данные;

– RoT должен обеспечить безопасное хранение криптографических ключей и иных конфиденциальных данных;

– производитель аппаратной платформы должен создать и инициализировать RoT в процессе производства.

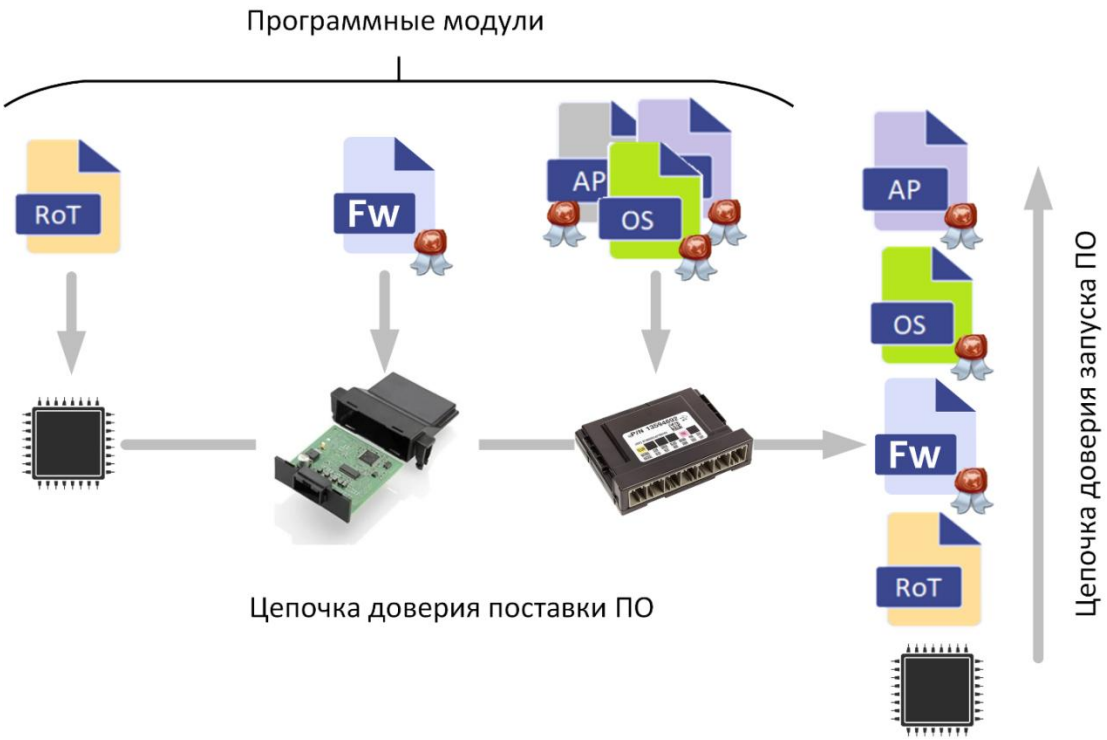

Рис. 2. Цепочки доверия, используемые в ПАК

Основное назначение корня доверия – это обеспечение процедуры безопасного старта системы и поддержка операций безопасной инсталляции/обновления программного обеспечения системы. Дополнительно корень доверия может предоставлять для программного обеспечения вышележащих уровней следующие сервисы безопасности:

- безопасное хранение криптографических ключей;
- аутентификация программных и аппаратных компонент и данных;
- обеспечение конфиденциальности программных компонент и данных;
- контроль целостности программных компонент и данных;
- измерение (поддержка цепочки доверия);
- отчётность (поддержка удалённого аудита);
- инсталляция и обновление программного обеспечения;
- защита канала связи.

3. Компоненты системы обновления программного обеспечения

Обобщённая схема процесса обновления программного обеспечения на основе технологии ePKI представлена на рис. 3. Схема построена на основе спецификаций GlobalPlatform[10], Trusted Connectivity Alliance[11], IETF[12].

Процесс безопасного обновления опирается на следующие компоненты:

1. Владелец Образа (Поставщик Услуг, Оператор Информационной Сети, Image Owner) - определят основные требования и разрешения для использования программного обеспечения на конкретном устройстве/типе устройств.

2. Разработчик программного обеспечения (Software Maker) создает и тестирует программное обеспечение в соответствии с требованиями Владельца Образа.

3. Издатель Образа (Image Maker) получает исполняемый код от Разработчика программного обеспечения, соответствующий требованиям и входным данным от Владельца Образа. Издатель Образа помещает программное обеспечение в безопасный контейнер, называемый Образом, который передается на Сервер Доставки Образов (IDS) под контролем и по согласованию с Владельцем Образа.

4. Сервер Доставки Образов (Image Delivery Server, IDS) – отвечает за доставку Образов Безопасному Загрузчику (SL). Между Сервером IDS и устройством выполняется взаимная аутентификация, на основе обмена сертификатами. В процессе проведения аутентификации выполняется выработка индивидуальных ключей шифрования образа. Образ, зашифрованный на индивидуальных для конкретного Устройства ключах шифрования, называется Связанным Образом (Bound Image) и может использоваться только в данном Устройстве.

5. Центр Сертификации (CA) управляет инфраструктурой открытых ключей (PKI), выдает сертификаты участникам.

6. Модуль Безопасности (SE) – это безопасная среда Устройства, в которой хранятся криптографические ключи и выполняются экземпляры корня доверия.

7. Изготовитель Модуля Безопасности (Chip Maker, CM) производит Модули Безопасности в соответствии со спецификациями отраслевых организаций и/или стандартами.

8. Безопасный Загрузчик (Secure Loader) – программный код, предназначенный для: выполнения взаимной аутентификации Сервера доставки Образов и Устройства,

[10]GPC_FST_134. GlobalPlatform Open Firmware Loader for Tamper Resistant Element Version 1.3.
[11]SIMA. TCA – Image Delivery Server (IDS) to Open Firmware Loader (OFL) Agent Interfaces.
[12]RFC 9124. A Manifest Information Model for Firmware Updates in Internet of Things (IoT) Devices.

загрузки Связанного Образа в память устройства, извлечение программного кода из безопасного контейнера Образа. Безопасный Загрузчик для выполнения своей работы использует Модуль безопасности.

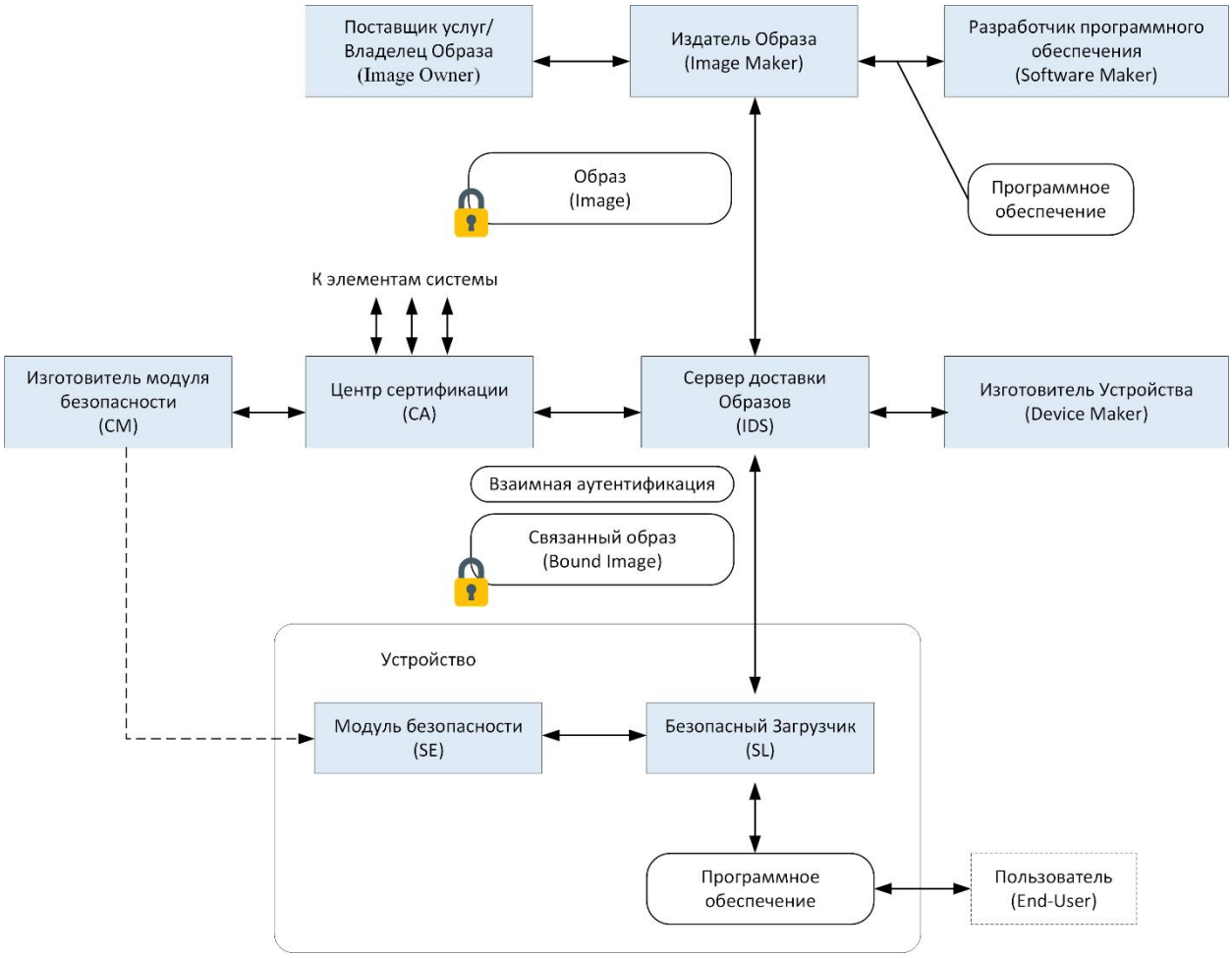

Рис. 3. Основные компоненты процесса безопасного обновления программного обеспечения

9. Изготовитель Устройств (Device Maker) производит Устройство, в которое встроен Безопасный Загрузчик и Модуль Безопасности. Изготовитель Устройств устанавливает Безопасный Загрузчик.

10. Конечный Пользователь (End-User/Subscribe) взаимодействует с Устройством. Конечному Пользователю может быть предложено дать своё явное согласие на проверку некоторых операций Безопасного Загрузчика или получение конкретных учётных данных от Поставщика Услуг.

Базовым компонентом ePKI инфраструктуры является Центр Сертификации (CA), который формирует сертификаты для участников и модулей системы (рис. 4). В ePKI системе циркулируют технологические сертификаты:

- цифровой сертификат участника цепочки поставки,
- цифровой сертификат электронных компонентов,
- цифровой сертификат программного обеспечения,
- цифровой сертификат устройства,
- цифровой сертификат управляющих систем,
- и т.д.

Для обозначения ключей и сертификатов используются следующие сокращения: SK (Secret Key) – закрытый ключ, PK (Public Key) – открытый ключ, CERT – сертификат открытого ключа.

Например: SK.CM – закрытый ключ Изготовителя модуля Безопасности; PK.SE – открытый ключ Модуля Безопасности; CERT.IDS.CA – сертификат открытого ключа Сервера доставки Образов, подписанный Центром Сертификации.

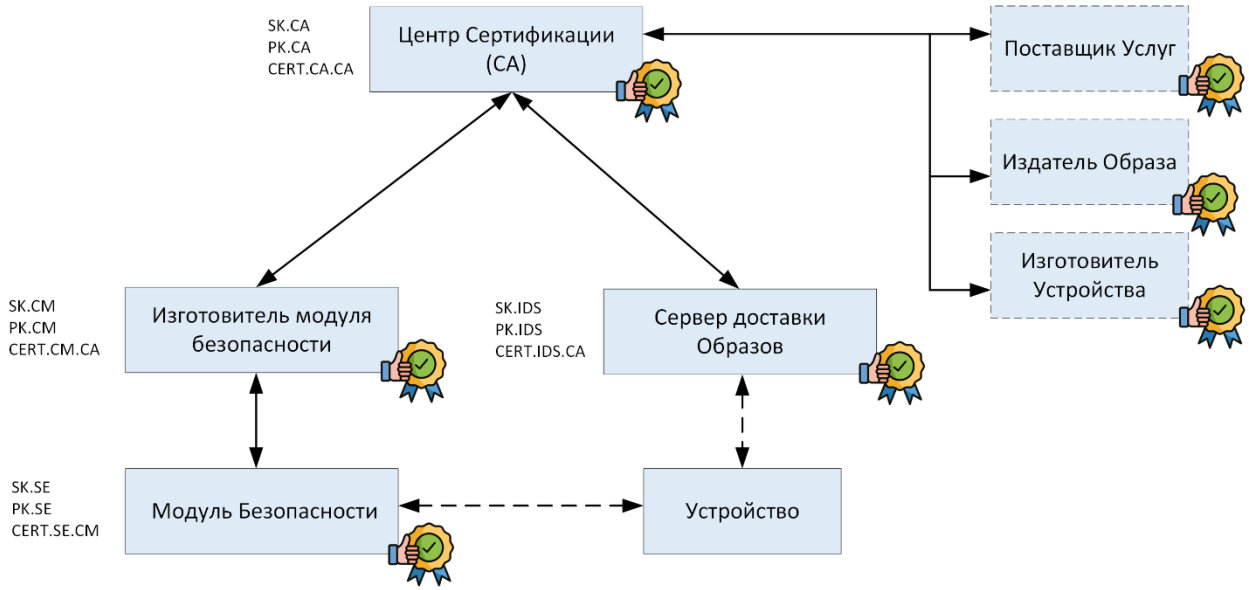

Рис. 4. Цепочка сертификатов процесса обновления программного обеспечения

Сертификаты Поставщика Услуг, Издателя Образа, Изготовителя Устройства не принимают непосредственного участия в формировании Образа программного обеспечения и используются для обеспечения безопасного обмена данными между участниками системы.

4. Криптографические преобразования для подготовки и загрузки образа

Для процесса безопасного обновление программного обеспечения на основе технологии ePKI необходимо реализовать следующие криптографические механизмы:

– генерация асимметричной ключевой пары;

– подпись данных и проверка подписи с использованием криптографии эллиптических кривых;

– вычисление общего секрета с помощью протокола Диффи-Хеллмана на эллиптических кривых;

– формирование сессионных ключей на основании общего секрета с помощью функции формирования ключа (Key Derivation Function, KDF);

– зашифрование и расшифрование данных с использованием симметричных ключей;

– вычисление и проверка имитовставки;

– преобразование симметричных ключей с помощью функции “поворота”;

– формирование сертификата X.509.

Перечисленные криптографические преобразования имеют общий характер и допускают использование как криптографических алгоритмов NIST, так и ГОСТ.

В табл. 2–7 представлено описание ключевых объектов, необходимых для и загрузки Образа программного обеспечения.

Таблица 2. Описание ключевых объектов Центра сертификации

п/п	Наименование	Описание
1	SK.CA	Закрытый ключ Центра сертификации. Применяется для подписи сертификатов Изготовителя модуля безопасности (CERT.CM.CA) и Сервера доставки образов (CERT.IDS.CA).
2	PK.CA	Открытый ключ Центра сертификации. Применяется для проверки подписи сертификатов Изготовителя модуля безопасности (CERT.CM.CA) и Сервера доставки образов (CERT.IDS.CA).
3	CERT.CA.CA	Самоподписанный сертификат открытого ключа Центра сертификации.

Таблица 3. Описание ключевых объектов Изготовителя модуля безопасности

п/п	Наименование	Описание
1	SK.CM	Закрытый ключ Изготовителя модуля Безопасности. Применяется для подписи сертификатов Модулей безопасности – CERT.SE.CM.
2	PK.CM	Открытый ключ Изготовителя модуля Безопасности. Применяется для проверки подписи сертификатов Модулей безопасности - CERT.SE.CM.
3	CERT.CM.CA	Сертификат открытого ключа Изготовителя модуля Безопасности, подписанный Центром Сертификации.

Таблица 4. Описание ключевых объектов Модуля безопасности

п/п	Наименование	Описание
1	SK.SE	Закрытый ключ Модуля безопасности. Применяется для подписи сертификатов ATK.SE.
2	PK.SE	Открытый ключ Модуля безопасности. Применяется для проверки подписи сертификатов ATK.SE.
3	CERT.SE.CM	Сертификат открытого ключа Модуля безопасности, подписанный закрытым ключом Изготовителя модуля безопасности.

Таблица 5. Описание ключевых объектов Сервера доставки образов

п/п	Наименование	Описание
1	SK.IDS	Закрытый ключ Сервера доставки образов. Применяется для подписи сертификатов ATK.IDS.
2	PK.IDS	Открытый ключ Сервера доставки образов. Применяется для проверки подписи сертификатов ATK.IDS.
3	CERT.IDS.CA	Сертификат открытого ключа Сервера доставки образов, подписанный закрытым ключом Центра сертификации.

Таблица 6. Описание ключевых объектов Издателя образов

п/п	Наименование	Описание
1	KT_{SSD}	Транспортный ключ для шифрования дескриптора образа IMD и дескрипторов сегментов SDS.
2	K_{SEG}	Набор ключей для шифрования сегментов. Отдельные ключи нумеруются от 1 до N: $K_{SEG1}…K_{SEGn}$ и хранятся в соответствующих дескрипторах сегментов. Где N – количество сегментов в образе.

Таблица 7. Описание сессионных ключевых объектов

п/п	Наименование	Описание
1	eSK.SE	Закрытый эфемерный ключ Модуля безопасности.
2	ePK.SE	Открытый эфемерный ключ Модуля безопасности.
3	ATK.SE	Токен аутентификации Модуля безопасности, который является укороченным сертификатом открытого эфемерного ключа Модуля безопасности, подписанный закрытым ключом Модуля безопасности. Позволяет выработать ключи для перешифрования дескрипторов сегментов.
4	eSK.IDS	Закрытый эфемерный ключ Сервера доставки образов.
5	ePK.IDS	Открытый эфемерный ключ Сервера доставки образов.
6	ATK.IDS	Токен аутентификации IDS, который является укороченным сертификатом открытого эфемерного ключа Сервера доставки образов, подписанный закрытым ключом Сервера доставки образов. Позволяет выработать ключи для перешифрования дескрипторов сегментов.
7	K_{IMD}	Ключ для шифрования IMD.
8	K_{SDS}	Набор ключей для шифрования дескрипторов сегментов. Отдельные ключи нумеруются от 1 до N: $K_{SDS1}…K_{SDSn}$,

5. Формат образа программного обеспечения

Образ программного обеспечения представляет собой упорядоченную структуру, включающую: дескриптор образа, набор дескрипторов сегментов и соответствующих сегментов исполняемого кода. На рис. 5 приведена схема структуры Связанного Образа после его формирования на Сервере Доставки Образов (IDS).

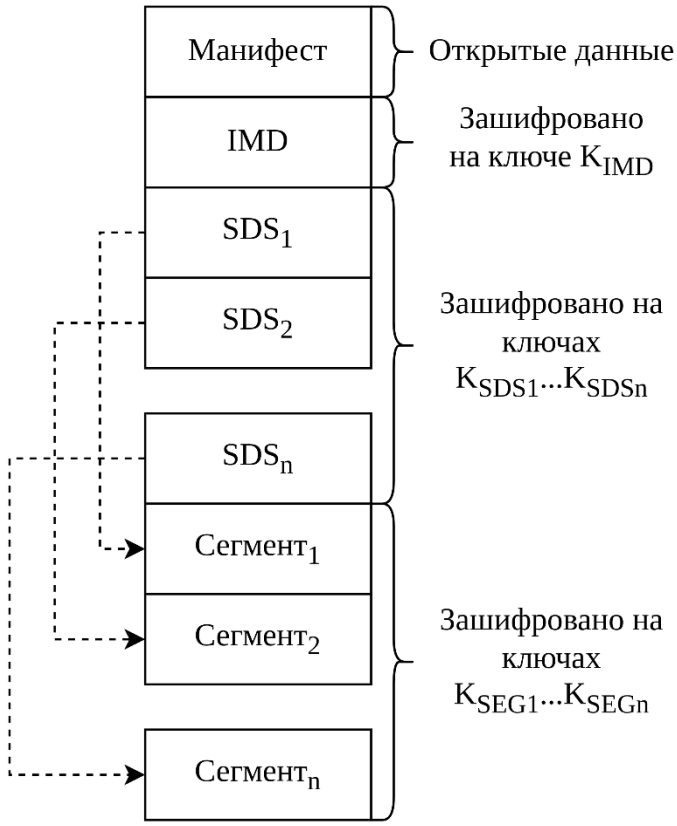

Рис. 5. Структура Связанного Образа

Манифест размещается в начале образа. Он содержит информацию о количестве и расположении структурных компонентов образа, включая дескрипторы образа, дескрипторы сегментов и сведения о самих сегментах исполняемого кода. Кроме структурных данных, манифест включает значение имитовставки, используемое для проверки целостности образа при его передаче от Издателя Образа (Image Maker) к Серверу Доставки Образов (IDS). Манифест передаётся в открытом виде и размещается в начале Образа.

Дескриптор образа (Image Descriptor, IMD) представляет собой структуру, содержащую описание Образа. Он включает служебную информацию, необходимую для обработки Образа, а также данные, обеспечивающие его однозначную идентификацию. В частности, дескриптор включает следующие поля:

– $UUID_P$ – частный (уникальный) идентификатор прошивки, используемый для внутреннего контроля;
– $UUID_I$ – общедоступный идентификатор прошивки;
– $UUID_G$ – идентификатор группы, к которой принадлежит прошивка;
– $UUID_F$ - идентификатор семейства прошивок;
– Текущая версия Образа;

– Параметры загрузки, определяющие допустимые операции при взаимодействии с Безопасным загрузчиком, включая разрешение на обновление, локальное удаление, резервное копирование и восстановление прошивки.

Для обеспечения конфиденциальности дескриптор зашифровывается с использованием отдельного ключа K_{IMD}.

Дескрипторы сегментов (Segment Descriptors, $SDS_1 \ldots SDS_n$) описывают соответствующие им сегменты исполняемого кода. Дескриптор сегмента содержит сведения о размере сегмента, адресе его размещения в памяти целевого устройства, а также сведения об криптографических параметрах. Дескрипторы сегментов зашифровываются с использованием индивидуальных ключей $K_{SDS1} \ldots K_{SDSn}$, что обеспечивает защиту их содержимого при передаче и хранении.

Сегменты исполняемого кода ($Сегмент_1 \ldots Сегмент_n$) содержат зашифрованные фрагменты программы, предназначенные для выполнения на устройстве. Шифрование каждого сегмента осуществляется с использованием отдельного ключа из набора $K_{SEG1} \ldots K_{SEGn}$, хранящиеся в соответствующем дескрипторе сегмента $SDS_1 \ldots SDS_n$.

На рис. 6 представлена последовательность формирования, шифрования и передачи Образа в рамках подготовки к безопасной загрузке на устройство:

1. Разработчик программного обеспечения (SM) формирует исполняемый программный код, разделённый на сегменты (например, в формате elf).
2. Издатель образа (IM) преобразует исполняемый код в Образ программного обеспечения. Этот процесс включает следующие этапы:

– зашифрование каждого сегмента с использованием индивидуальных ключей $K_{SEG1} \ldots K_{SEGn}$;

– генерация дескрипторов сегментов ($SDS_1 \ldots SDS_n$);

– формирование дескриптора Образа IMD;

– создание манифеста;

– зашифрование заголовочной части образа (IMD и SDS дескрипторов) с использованием ключа KT_{SSD}.

3. На следующем этапе Сервер Доставки Образов (IDS) выполняет преобразование Образа в Связанный Образ, путём выполнения:

– перешифрования каждого дескриптора сегмента $SDS_1 \ldots SDS_n$ с использованием ключей $K_{SDS1} \ldots K_{SDSn}$;

– перешифрования дескриптора IMD с применением ключа K_{IMD}.

4. Безопасный загрузчик (SL) на стороне устройства:

– принимает связанный Образ, содержащий зашифрованные сегменты и дескрипторы;

– выполняет расшифровку дескрипторов и на их основе, расшифровку соответствующих сегментов;

– загружает расшифрованные сегменты в область памяти среды исполнения.

5. Устройство (Среда исполнения) получает исполняемый код и инициирует выполнение исполняемого кода.

6. Безопасный загрузчик, основные команды, последовательность загрузки

Безопасный Загрузчик (SL) предназначен для выполнения процедуры аутентификации Сервера Доставки Образов (IDS), для обеспечения безопасной загрузки прошивки, проверки её целостности и подлинности, а также для осуществления последующей записи прошивки в память устройства. В своей работе Загрузчик

взаимодействует с Модулем Безопасности, обеспечивающим выполнение криптографических операций.

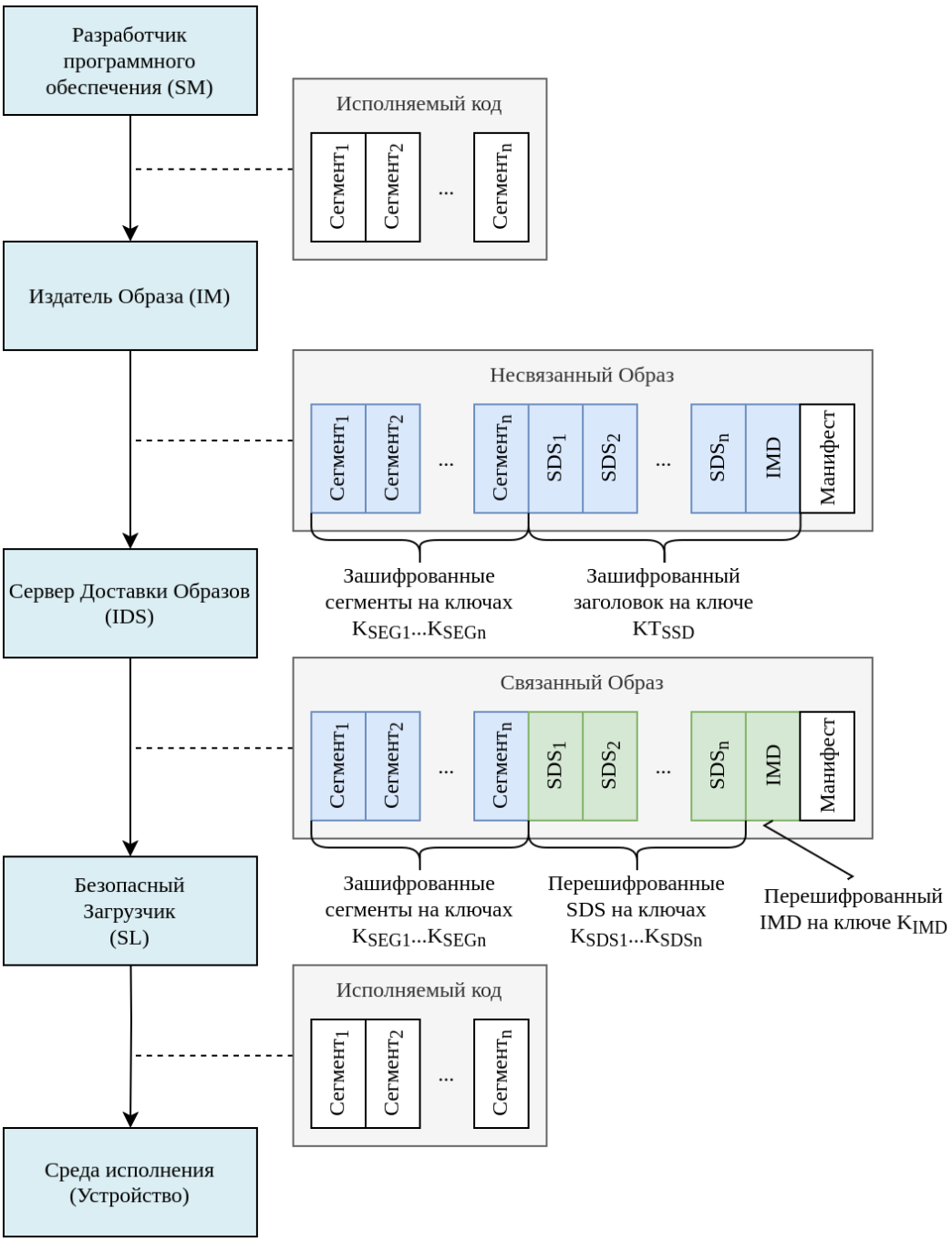

Рис. 6. Последовательность преобразования Образа

Одной из ключевых внутренних структур данных Безопасного загрузчика является реестр Безопасного Загрузчика. Реестр используется для управления процессом загрузки и хранения параметров, связанных с конкретным Образом (сессия). Безопасный загрузчик может поддерживать несколько сессий для разных загружаемых прошивок. На основе идентифицирующих данных, содержащихся в Образе (например, $UUID_P$ прошивки), Сервер активизирует необходимую сессию прошивки.

Для организации процесса загрузки Образа используется набор специализированных команд, каждая из которых соответствует определённому этапу обработки. В табл. 8 приведено краткое описание этих команд.

Таблица 8. Команды безопасной загрузки образа

Наименование команды	Описание команды
УСТАНОВИТЬ ПАРАМЕТР	Команда для записи параметра в реестр Безопасного Загрузчика.
ПОЛУЧИТЬ ПАРАМЕТР	Команда для чтения параметра из реестра Безопасного Загрузчика.
ВЫПОЛНИТЬ ОПЕРАЦИЮ	Команда, инициирующая процесс загрузки Связанного Образа в память Устройства. Используется для идентификации Связного Образа на Устройстве. Выполняется однократно в начале процесса загрузки.
СМЕНИТЬ СЕГМЕНТ	Выполняется перед загрузкой каждого сегмента прошивки. Команда передаёт на Устройство информацию о сегменте исполняемого кода из дескриптора SDS, включая ключ шифрования и значение имитовставки сегмента.
ЗАГРУЗИТЬ СЕГМЕНТ	Команда, непосредственно передающая зашифрованный сегмент образа прошивки. Используется для записи содержимого сегмента в память Устройства для последующего выполнения.

После каждого выполнения команды, Устройство должно возвращать результат операции: OK – для подтверждения успешного выполнения, и NOK – в случае неудачного выполнения операции.

На начальном этапе загрузки Образа осуществляется процедура взаимной аутентификации между Сервером Доставки Образов (IDS) и Безопасным Загрузчиком (SL) с Модулем Безопасности (SE). По окончанию данной процедуры в IDS выполняется генерация Связанного Образа.

На рис. 7 представлена последовательность выполнения взаимной аутентификации и формирования Связанного Образа, включающая следующие шаги:

1. IDS передаёт в SL сертификат CERT.IDS.CA;
2. SL проверяет полученный сертификат и извлекает из него открытый ключ сервера PK.IDS;
3. SL генерирует эфемерную ключевую пару eSK.SE и ePK.SE;
4. SL формирует токен аутентификации ATK.SE, который содержит ePK.SE, и подписывает его с помощью собственного закрытого ключа SK.SE;
5. SL передаёт серверу IDS весь сформированный криптографический материал: сертификаты CERT.SE.CM, CERT.CM.CA, токен ATK.SE.
6. IDS выполняет проверку полученной цепочки сертификатов CERT.SE.CM, CERT.CM.CA и извлекает PK.SE из CERT.SE.CM;
7. IDS проверяет подпись токена ATK.SE, используя PK.SE, и извлекает из него эфемерный ключ ePK.SE;
8. IDS генерирует эфемерную ключевую пару eSK.IDS и ePK.IDS;

9. Используя эфемерный открытый ключ Безопасного загрузчика ePK.SE и собственный eSK.IDS, ранее полученный из токена ATK.SE, IDS формирует набор сессионных ключей K_{IMD} и K_{SDS1}…K_{SDSn};

10. IDS формирует токен ATK.IDS, включающий открытый ключ ePK.IDS и цифровую подпись, созданную с использованием ключа SK.IDS;

11. IDS передаёт токен ATK.IDS на устройство SL;

12. SL проверяет подлинность токена ATK.IDS с использованием открытого ключа PK.IDS;

13. SL используя эфемерные ключи, полученные ранее и включённые в токен ATK.IDS, формирует идентичный набор ключей K_{IMD} и K_{SDS1}…K_{SDSn};

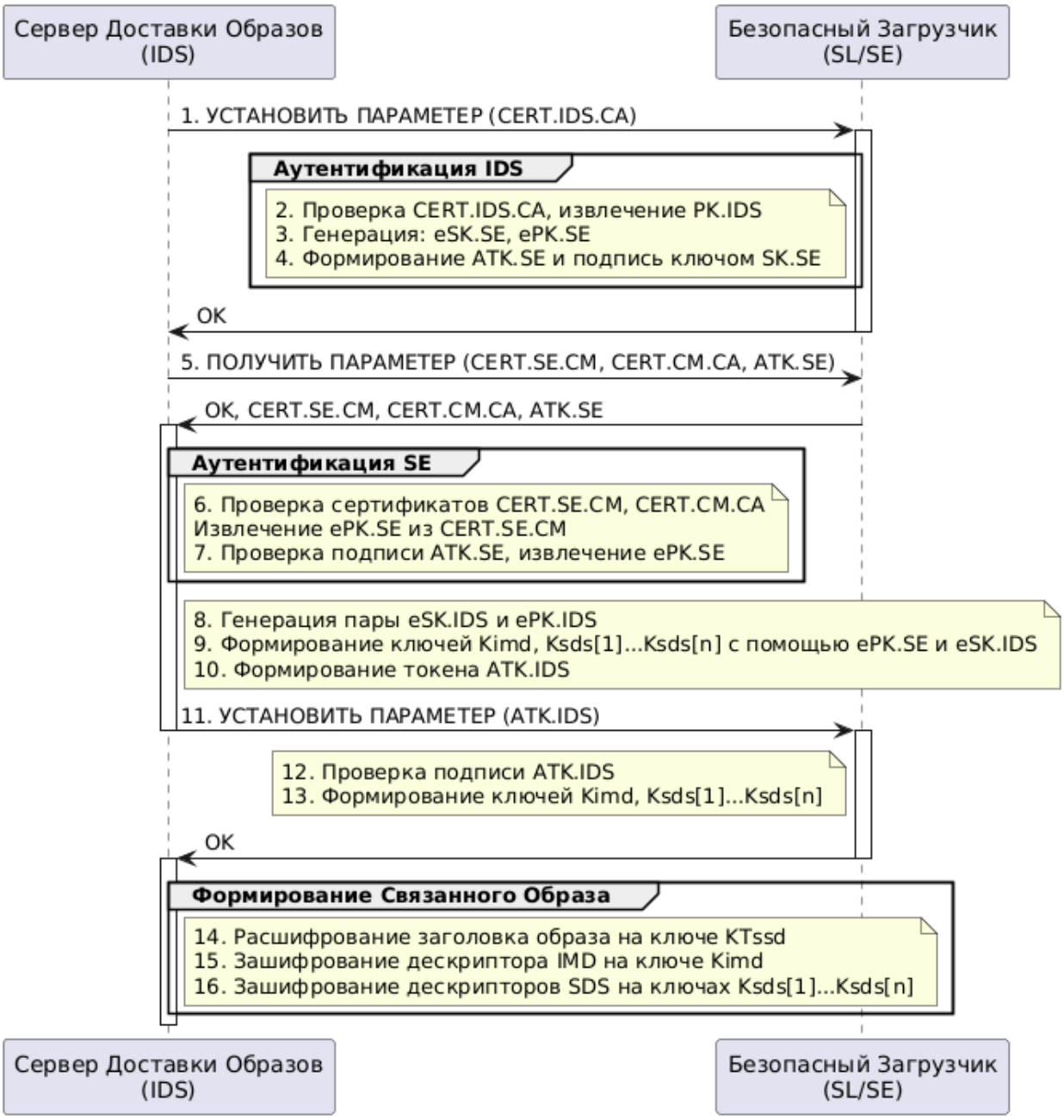

Рис. 7. Аутентификация и формирование Связанного Образа

14. IDS выполняет расшифрование заголовка Образа с помощью транспортного ключа KT_{SSD};

15. IDS выполняет зашифрование дескриптора IMD загружаемого Образа с помощью ключа K_{IMD};

16. IDS выполняет зашифрование дескрипторов сегментов загружаемого Образа с помощью ключей $K_{SDS1}…K_{SDSn}$.

Следующий этап загрузки образа прошивки заключается в загрузки Связанного Образа на устройство через Безопасный Загрузчик. Схема выполняемых действий приведена на рис. 8.

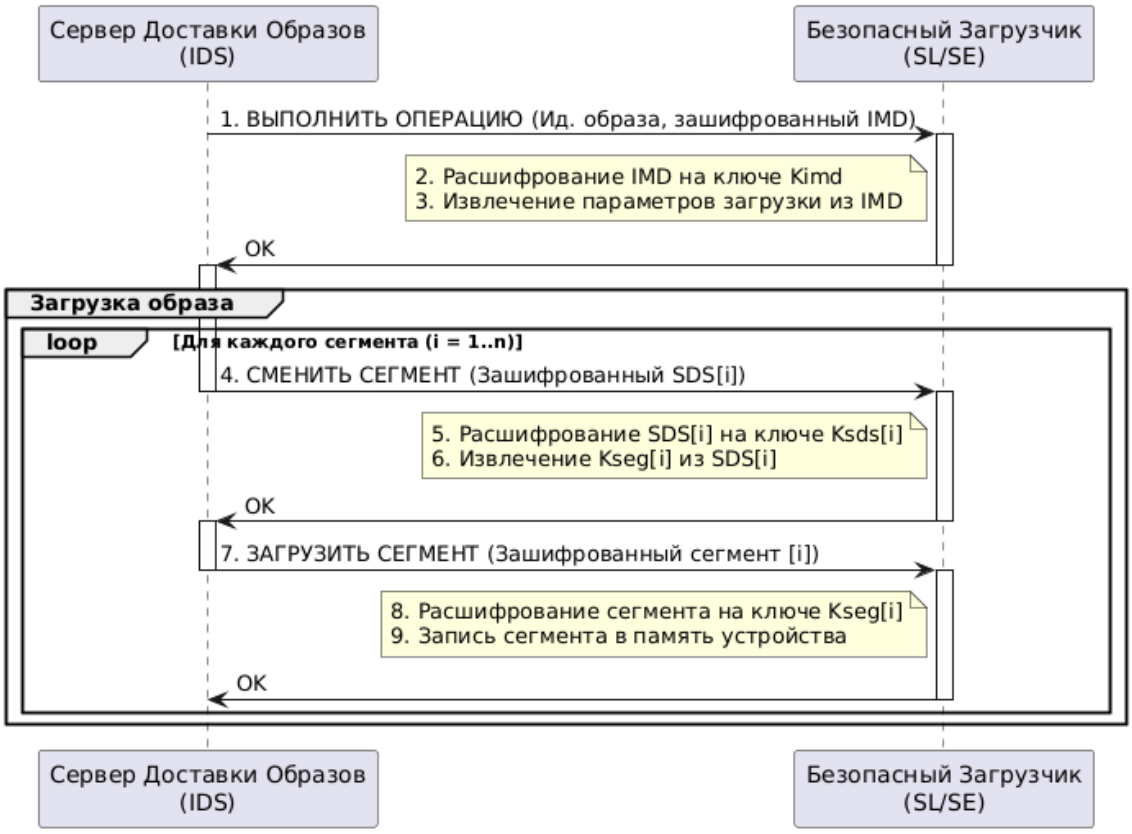

Рис. 8. Загрузка Связанного Образа

При загрузке Связанного Образа на устройство, выполняются следующие действия:

1. IDS инициирует операцию загрузки Связанного Образа на устройство, передавая в качестве аргументов команды идентификатор образа и данные Связанного Образа;

2. SL расшифровывает дескриптор IMD загружаемого образа с использованием ключа K_{IMD};

3. SL осуществляет извлечение параметров загрузки и перечень допустимых операций из дескриптора IMD;

4. IDS передаёт дескриптор сегмента SDS_i на устройство SL;

5. SL расшифровывает дескриптор сегмента SDS_i Связанного Образа с использованием ключа K_{SDSi};

6. SL извлекает данные о ключе шифрования K_{SEGi} и адрес расположения сегмента в оперативной памяти из дескриптора SDS_i;

7. IDS передаёт соответствующий зашифрованный сегмент на устройство;

8. SL расшифровывает сегмент прошивки с использованием ключа K_{SEGi}, полученного из дескриптора сегмента SDS_i;

9. SL осуществляет запись исполняемого сегмента в энергонезависимую память устройства, используя адрес расположения сегмента из SDS_i.

7. Описание макета системы безопасного обновления ПО

Для определения минимальных требований к ресурсам контроллера, необходимых для работы Безопасного Загрузчика, был разработан макет системы, состоящий из следующих программных компонентов (рис. 9):

- Центр сертификации (CA);
- Разработчик программного обеспечения (SM);
- Издатель образа (IM);
- Сервер доставки Образов (IDS);
- Изготовитель модуля безопасности (CM);
- Безопасный Загрузчик (SL);
- Модуль безопасности (SE).

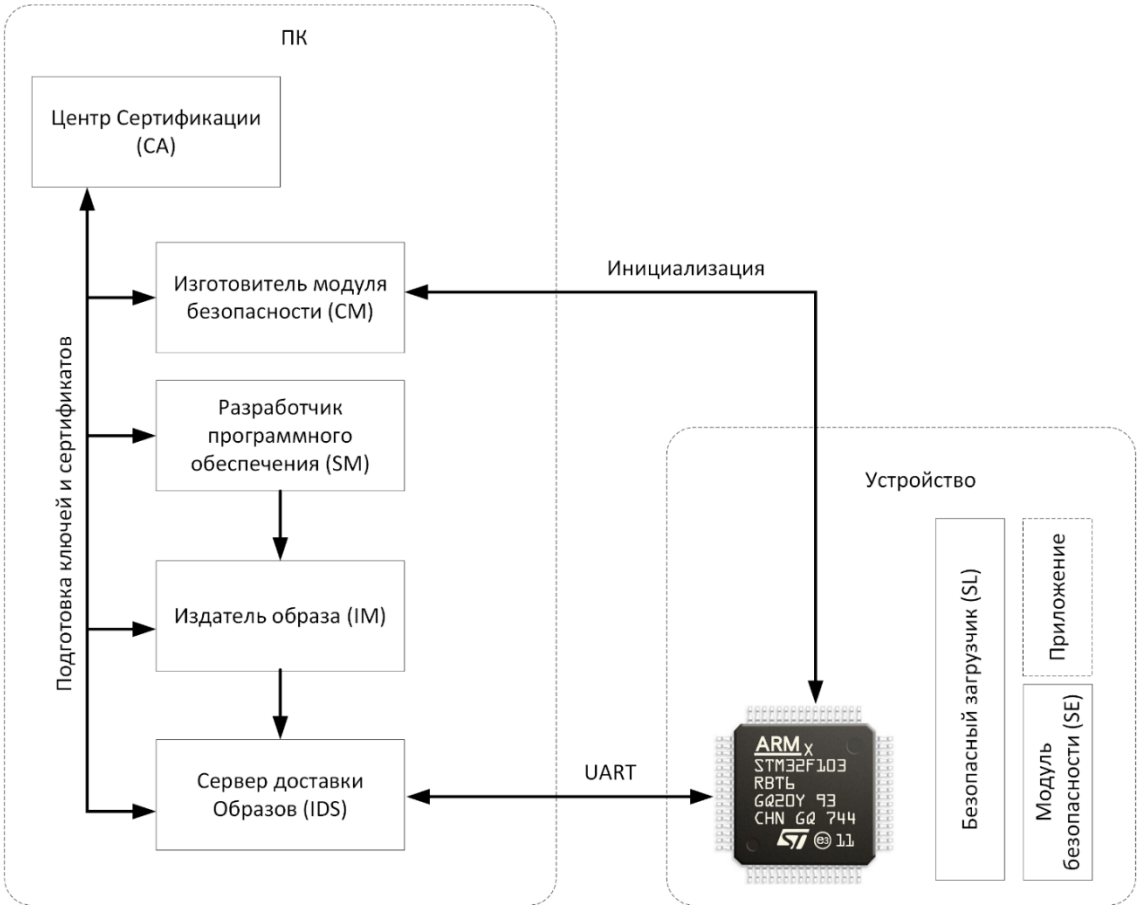

Рис. 9. Структура макета системы безопасного обновления встроенного программного обеспечения

Реализация программных компонентов Центра сертификации (CA), Издателя образов (IM), Сервера доставки образов (IDS) и Изготовителя модуля безопасности (CM) выполнена в виде программных модулей на языке Python. Для реализации криптографических операций использована библиотека pyca/cryptography,

предоставляющая интерфейсы к широко используемым криптографическим алгоритмам и примитивам.

В качестве аппаратной платформы для реализации Безопасного Загрузчика был выбран микроконтроллер STM32F103C8T6, который представляет собой систему на кристалле начального уровня:

- Ядро: ARM Cortex-M3, 32-бит;
- Максимальная тактовая частота: 72 МГц;
- ПЗУ (Flash Memory): 64 КБ;
- ОЗУ (SRAM): 20 КБ;
- Поддержка интерфейсов: 3xUART, 1xCAN, 1xUSB.

Модуль Безопасности был реализован исключительно на программном уровне, без привлечения аппаратных ускорителей.

Разработанное на языке С программное обеспечение Безопасного Загрузчика вместе с Модулем Безопасности занимает 20352 байт ПЗУ и требует для своей работы 7864 байт ОЗУ. При этом, алгоритмы, реализующие криптографические преобразования, занимают 10309 байт ПЗУ (табл. 9). Оставшиеся 10043 байт приходится на реализацию протокола обмена и организацию среды исполнения.

Таблица 9. Размер алгоритмов криптографических преобразований

п/п	Наименование	Описание	Размер, байт
1	AES-128	Алгоритм симметричного шифрования, размер ключа 128 бит	3918
2	AEAD-GCM	Режим шифрования AES	1562
3	SHA-256	Алгоритм хеширования	1091
4	KDF	Функция получения ключа	150
5	RNG	Алгоритм генерации случайных чисел	690
6	ECDSA NIST P-256	Алгоритмы работы с эллиптическими кривыми P-256 (генерация, подпись, проверка подписи, вычисление общего секрета)	2898

Образ загружаемого/обновляемого программного обеспечения (приложение) формировался на персональном компьютере и передавался на устройство по интерфейсу UART. Размер незашифрованного образа составляет 11136 байт. Связанный Образ (зашифрованные сегменты плюс дескрипторы IMD и SDS) занимает 12468 байт. Размеры используемых криптографических объектов представлены в табл. 10.

В ходе эксперимента была проведена оценка времени загрузки Связанного Образа размером 11 КБ (табл. 11) в устройство. Затраченное время составило приблизительно 11 с.

На устройстве при выполнении этапа «Взаимная аутентификация и выработка наборов ключей» (в соответствии с рис. 7) происходит четыре проверки подписи сертификатов формата X.509, одна генерация эфемерных ключей, два вычисления общего секрета и одна генерация сертификата формата X.509.

При выполнении этапа «Загрузка и расшифрование Связанного Образа» (в соответствии с рис. 8) на устройстве происходит одно расшифрование IMD, одиннадцать расшифрований SDS и одиннадцать расшифрований сегментов.

Таблица 10. Размер криптографических объектов

п/п	Наименование	Размер, байт
1	CERT.CA.CA	504
2	CERT.CM.CA	501
3	CERT.SE.CM	496
4	CERT.IDS.CA	504
5	ATK.SE	297
6	ATK.IDS	299
7	ePK.SE	64
8	eSK.SE	32

Таблица 11. Оценка времени загрузки Связанного Образа размером 11 КБ

п/п	Этап	Длительность, мс.
1	Взаимная аутентификация и выработка наборов ключей	10277
2	Загрузка и расшифрование Связанного Образа	943

Время выполнения первого этапа процесса загрузки Связанного Образа практически не зависит от размера прошивки.

Время второго этапа линейно зависит от размера прошивки и в данной реализации составляет около 8 мс на килобайт передаваемых данных. Необходимо отметить, что в указанное время входит передача данных по интерфейсу UART, расшифрование сегментов прошивки, проверка имитовставок, запись во Flash память контроллера.

Заключение

Использование технологии ePKI, базирующейся на криптографии эллиптических кривых, позволяет разрабатывать весьма гибкие и защищённые системы управления обновлениями встроенного программного обеспечения, которые применяют цепочки доверия на основе цифровых технологических сертификатов.

Основой для построения цепочек доверия в системах информационной безопасности является Корень доверия (RoT), который должен быть неизменяемым, хорошо верифицированным, обеспечивать безопасное хранение криптографических ключей и иных конфиденциальных данных.

Наибольшую защиту обеспечивают корни доверия, выполненные на базе аппаратных модулей безопасности.

Пилотная реализация системы безопасного обновления встроенного программного обеспечения продемонстрировала, что:

– увеличение размера образа программного обеспечения за счёт добавления дескрипторов безопасности не превышает двух килобайт вне зависимости от размера прошивки;

– криптографические объекты, используемые в протоколах безопасности имеют достаточно малые размеры (сотни байт);

– тестовая реализация Безопасного загрузчика вместе с программным Модулем безопасности, написанная на языке C, занимает около 20KB энергонезависимой памяти;

– время обновления приложения размером 11KB для микроконтроллера начального уровня составляет около 11 с (без использования криптографических ускорителей).

Полученные результаты позволяют сделать обоснованный вывод, что описанная технология может успешно применяться даже для маломощных микроконтроллеров с ограниченным количеством памяти и низкой производительностью.

Для доверенных ПАК, созданных на основе производительных процессоров приложений, описанная технология может использоваться не только для обновления программного обеспечения, но и для проверки аутентичности и целостности загружаемого в RAM системного и прикладного программного обеспечения при старте системы и запуске приложений (доверенная загрузка). В данном случае на локальном или сетевом устройстве хранения данных Связанные Образы программного обеспечения будут записаны в зашифрованном виде, что увеличивает безопасность системы. Скорость загрузки при этом будет существенно выше, чем для описанного в данной статье макета.

СПИСОК ЛИТЕРАТУРЫ:

1. Fauzi, Muhammad & Mohan, Vinod & Qi, Yang & Chandrasegar, Christal & Muzafar, Saira. Secure Software Development Best Practices. International Journal of Emerging Multidisciplinaries: Computer Science & Artificial Intelligence. 2023, v. 2, p. 1–18. DOI: http://dx.doi.org/10.54938/ijemdcsai.2023.02.1.256.
2. Alauthman, M., Al-Qerem, A., Aldweesh, A., Almomani, A. Secure SDLC Frameworks: Leveraging DevSecOps to Enhance Software Security. Modern Insights on Smart and Secure Software Development. IGI Global Scientific Publishing. 2025, p. 77–118. DOI: https://doi.org/10.4018/979-8-3693-9851-7.ch003.
3. Bakhshi, T., Ghita, B., Kuzminykh, I. A Review of IoT Firmware Vulnerabilities and Auditing Techniques. Sensors. 2024, v. 24, no. 2. DOI: https://doi.org/10.3390/s24020708.
4. El Jaouhari S., Bouvet E. Secure firmware Over-The-Air updates for IoT: Survey, challenges, and discussions. Internet of Things. 2022, v. 18, p. 100508. DOI: http://dx.doi.org/10.1016/j.iot.2022.100508.
5. Bast C., Yeh K.-H. Emerging Authentication Technologies for Zero Trust on the Internet of Things. Symmetry. 2024, v. 16, no. 8, p. 993. DOI: http://dx.doi.org/10.3390/sym16080993.
6. Hoang, Trong-Thuc & Duran, Ckristian & Serrano, Ronaldo & Sarmiento, Marco & Nguyen, Khai-Duy & Tsukamoto, Akira & Suzaki, Kuniyasu & Pham, Cong-Kha. Trusted Execution Environment Hardware by Isolated Heterogeneous Architecture for Key Scheduling. IEEE Access. 2022, v. 10, p. 46014–46027. DOI: http://dx.doi.org/10.1109/ACCESS.2022.3169767.
7. Wei, J., & Pu, C. (2005). TOCTTOU vulnerabilities in UNIX-style file systems: an anatomical study. USENIX Conference on File and Storage Technologies. URL: https://www.usenix.org/legacy/event/fast05/tech/full_papers/wei/wei.pdf (дата обращения: 18.05.2025).
8. S. El Jaouhari. Toward a Secure Firmware OTA Updates for constrained IoT devices. IEEE International Smart Cities Conference (ISC2), Pafos, Cyprus, 2022, p. 1–6. DOI: https://doi.org/10.1109/ISC255366.2022.9922087.
9. Конев А.А. Модель угроз безопасности защищенного микроконтроллера и обрабатываемой им информации. Доклады ТУСУР. 2022, т. 25, № 4, с. 80–87. DOI: http://dx.doi.org/10.21293/1818-0442-2022-25-4-80-87.
10. Беляков Д. С., Калинин Е.О., Конев А.А., Шелупанов А.А., Мицель А.А. Модели жизненного цикла и угрозы безопасности микросхемы во время ее разработки и эксплуатации. Доклады ТУСУР. 2023, т. 26, № 1, с. 76–81. DOI: http://dx.doi.org/10.21293/1818-0442-2023-26-1-76-81.
11. Горбатов В.С., Полянская О.Ю. Основы технологии PKI. Монография. ISBN: 978-5-9912-0213-8. М.: Горячая линия – Телеком. 2011. – 248 с. – EDN: KWAAHG.
12. Молдовян А.А., Молдовян Д.Н., Молдовян Н.А. Новый подход к разработке алгоритмов многомерной криптографии. Вопросы кибербезопасности. 2023, № 2(54), с. 52–64. DOI: 10.21681/2311-3456-2023-2-52-64. – EDN: JXHQMI.

13. Беляев С.С., Будько М.Б., Будько М.Ю., Гирик А.В., Грозов В.А. Построение функции генерации криптографически стойких псевдослучайных последовательностей на базе алгоритма шифрования «Кузнечик». Вопросы кибербезопасности. 2021, № 4(44), с. 25–34. DOI: 10.21681/2311-3456-2021-4-25-34. – EDN: GBNJBU.
14. Энциклопедия Касперского, Глоссарий, Цепочка доверия (chain of trust). https://encyclopedia.kaspersky.ru/glossary/chain-of-trust/ (дата обращения 18.05.2025).
15. Дьяков, Олег Н. Специальное встроенное программное обеспечение доверенной электронной компонентной базы. Безопасность информационных технологий, [S.l.], т. 31, № 4, с. 67–86, 2024. ISSN 2074-7136. DOI: http://dx.doi.org/10.26583/bit.2024.4.04.

REFERENCES:

[1] Fauzi, Muhammad & Mohan, Vinod & Qi, Yang & Chandrasegar, Christal & Muzafar, Saira. Secure Software Development Best Practices. International Journal of Emerging Multidisciplinaries: Computer Science & Artificial Intelligence. 2023, v. 2, p. 1–18. DOI: http://dx.doi.org/10.54938/ijemdcsai.2023.02.1.256.
[2] Alauthman, M., Al-Qerem, A., Aldweesh, A., Almomani, A. Secure SDLC Frameworks: Leveraging DevSecOps to Enhance Software Security. Modern Insights on Smart and Secure Software Development. IGI Global Scientific Publishing. 2025, p. 77–118. DOI: https://doi.org/10.4018/979-8-3693-9851-7.ch003.
[3] Bakhshi, T., Ghita, B., Kuzminykh, I. A Review of IoT Firmware Vulnerabilities and Auditing Techniques. Sensors. 2024, v. 24, no. 2. DOI: https://doi.org/10.3390/s24020708.
[4] El Jaouhari S., Bouvet E. Secure Firmware Over-The-Air Updates for IoT: Survey, Challenges, and Discussions. Internet of Things. 2022, v. 18, p. 100508. DOI: http://dx.doi.org/10.1016/j.iot.2022.100508.
[5] Bast C., Yeh K.-H. Emerging Authentication Technologies for Zero Trust on the Internet of Things. Symmetry. 2024, v. 16, no. 8, p. 993. DOI: http://dx.doi.org/10.3390/sym16080993.
[6] Hoang, Trong-Thuc & Duran, Ckristian & Serrano, Ronaldo & Sarmiento, Marco & Nguyen, Khai-Duy & Tsukamoto, Akira & Suzaki, Kuniyasu & Pham, Cong-Kha. Trusted Execution Environment Hardware by Isolated Heterogeneous Architecture for Key Scheduling. IEEE Access. 2022, v. 10, p. 46014–46027. DOI: http://dx.doi.org/10.1109/ACCESS.2022.3169767.
[7] Wei, J., & Pu, C. (2005). TOCTTOU vulnerabilities in UNIX-style file systems: an anatomical study. USENIX Conference on File and Storage Technologies. URL: https://www.usenix.org/legacy/event/fast05/tech/full_papers/wei/wei.pdf (accessed: 18.05.2025).
[8] S. El Jaouhari. Toward a Secure Firmware OTA Updates for constrained IoT devices. IEEE International Smart Cities Conference (ISC2), Pafos, Cyprus, 2022, p. 1–6. DOI: https://doi.org/10.1109/ISC255366.2022.9922087.
[9] Konev A.A. Security threat model for protected microcontroller and the information it processes. TUSUR Reports. 2022, vol. 25, no. 4, p. 80–87. DOI: http://dx.doi.org/10.21293/1818-0442-2022-25-4-80-87 (in Russian).
[10] Belyakov D.S., Kalinin E.O., Konev A.A., Shelupanov A.A., Mitsel A.A. Life-cycle models and security threats to the microchip during its development and exploitation. TUSUR Reports. 2023, vol. 26, no. 1, p. 76–81. DOI: http://dx.doi.org/10.21293/1818-0442-2023-26-1-76-81 (in Russian).
[11] Gorbatov V.S., Polyanskaya O.Yu. Fundamentals of PKI Technology. Monograph. M.: Goryachaya liniya – Telekom, 2011. – 248 p. – ISBN: 978-5-9912-0213-8. – EDN: KWAAHG (in Russian).
[12] Moldovyan A.A., Moldovyan D.N., Moldovyan N.A. A New Approach to the Development of Multidimensional Cryptography Algorithms. Cybersecurity Issues. 2023, no. 2(54), p. 52–64. DOI: http://dx.doi.org/10.21681/2311-3456-2023-2-52-64. – EDN: JXHQMI (in Russian).
[13] Belyaev S.S., Budko M.B., Budko M.Yu., Girik A.V., Grozov V.A. Construction of a Cryptographically Secure Pseudorandom Sequence Generator Based on the “Kuznyechik” Encryption Algorithm. Cybersecurity Issues. 2021, no. 4(44), p. 25–34. DOI: 10.21681/2311-3456-2021-4-25-34. – EDN: GBNJBU (in Russian).
[14] Kaspersky Encyclopedia. Glossary. Chain of Trust. URL: https://encyclopedia.kaspersky.ru/glossary/chain-of-trust/ (accessed: 18.05.2025) (in Russian).
[15] Dyakov, Oleg N. Special embedded software of trusted electronic component. IT Security (Russia), [S.l.], v. 31, no. 4, p. 67–86, 2024. ISSN 2074-7136. DOI: http://dx.doi.org/10.26583/bit.2024.4.04 (in Russian).

Статья поступила в редакцию 20.03.2025; одобрена после рецензирования 20.05.2025;
принята к публикации 20.05.2025
The article was submitted 20.03.2025; approved after reviewing 20.05.2025;
accepted for publication 20.05.2025