



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К НАЧУНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ
НА ТЕМУ:

Анализ существующих реализаций доверенных сред
исполнения (ТЭЕ)

Студент группы ИУ7-32М

(Подпись, дата)

А. В. Романов

(И.О. Фамилия)

Руководитель НИР

(Подпись, дата)

Д. Е. Бекасов

(И.О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 Анализ предметной области	5
1.1 Кольца привилегий	5
1.2 Доверенная среда исполнения	6
2 Существующие реализации ДСИ	7
2.1 ARM TrustZone	7
2.1.1 Обычный и безопасный мир	8
2.1.2 Режимы работы процессора	10
2.1.3 Разделение памяти	12
2.1.4 Проверка целостности	13
2.2 Intel SGX	13
2.2.1 Processor Reserved Memory	14
2.2.2 Жизненный цикл анклава	16
2.2.3 Проверка целостности	18
2.3 Keystone	19
2.3.1 Компоненты системы	19
2.3.2 Жизненный цикл анклава	20
2.3.3 Проверка целостности	22
3 Сравнение реализаций ДСИ	23
3.1 Критерии сравнения	23
ЗАКЛЮЧЕНИЕ	24
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	25

ВВЕДЕНИЕ

Необходимость повышения безопасности исполнения приложений, работающих в системах безопасности и обрабатывающих защищаемую информацию, привела к разработке программно-аппаратных решений, создающих доверенные среды исполнения (англ. TEE – Trusted Execution Environment [1]) на базе аппаратных средств, доверенных загрузок или аппаратно-программных модулей доверенной загрузки. Intel [2] и ARM [3] являются лидерами в этой области. Целью данной работы является анализ и сравнение существующих реализаций доверенных сред исполнения (ДСИ).

Для достижения поставленной цели необходимо решить следующие задачи:

- провести обзор существующих реализаций ДСИ;
- описать плюсы и недостатки каждой из реализаций;
- сформулировать критерии сравнения;
- сравнить существующие реализации.

1 Анализ предметной области

В этом разделе будут проведен анализ предметной области. Описаны методы обеспечения защиты информации на современных процессорах. Дано понятие и характеристика доверенной среды исполнения.

1.1 Кольца привилегий

В целях безопасности, компоненты любой системы разделены на уровни привилегий – кольца защиты, за реализацию которых отвечает разработчик процессора. Во всех современных системах, реализована кольцевая система уровней привилегий. От внешнего кольца к внутреннему идёт увеличение полномочий для инструкций кода, выполняемых на процессоре в данный момент (рис. 1).

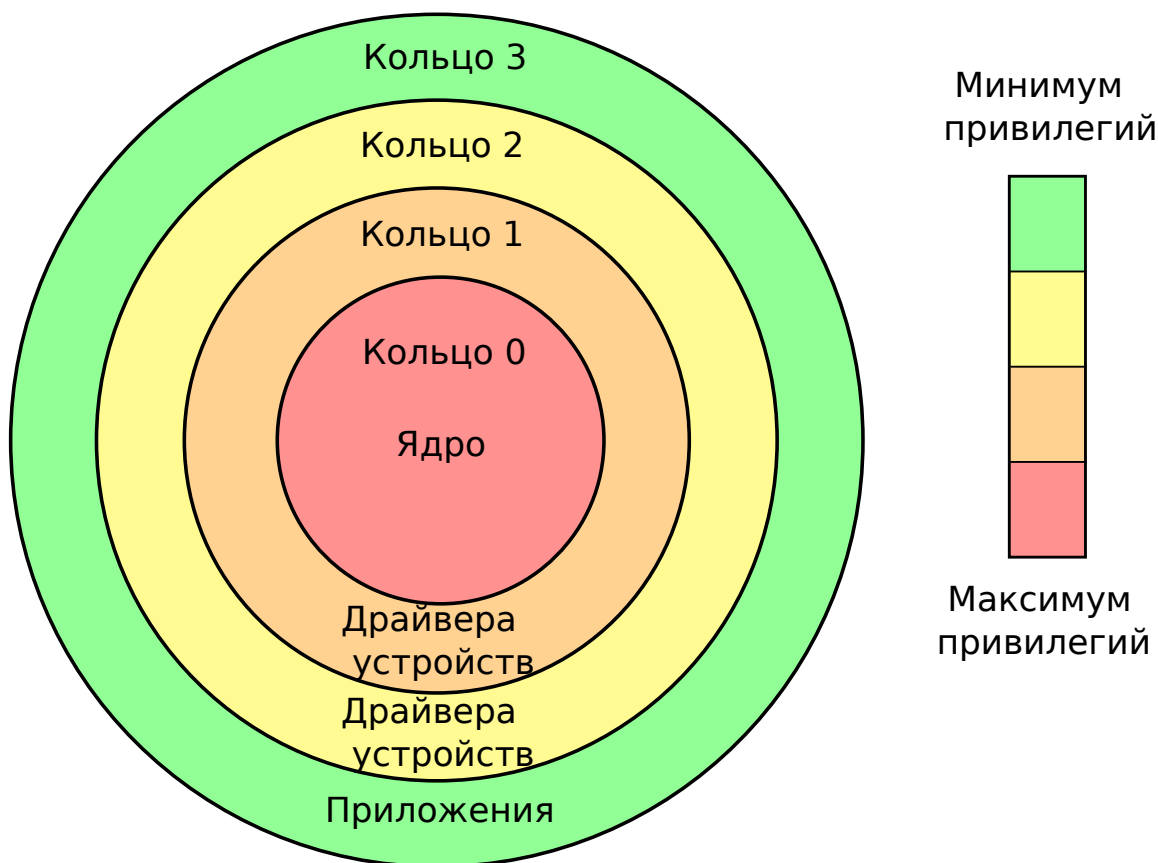


Рисунок 1 – Концептуальная схема представления колец защиты в современных системах

Можно создавать ещё более сложную систему – формировать ещё больше колец защиты, для ограничения каждого из компонентов системы. Однако, чем сложнее архитектура системы и чем больше количества кода в ней, тем проще злоумышленнику найти уязвимость и эксплуатировать её [4].

Главной задачей злоумышленника является получение доступ к привилегиям, которые бы позволили получить доступ к необходимым ресурсам системы. Может показаться, что архитектурно верным является решение размещать код, который отвечает за управление пользовательским данным, и конфиденциальные данные нужно исключительно на последнем кольце защиты – ведь получить доступ туда сложнее всего. Но у такого подхода есть свои недостатки [4]. Данный подход был переосмыслен – в настоящее используется схема, когда и код, и конфиденциальные данные хранятся на одном и том же уровне, что и пользовательские предложения, однако, доступ к ним имеет только лишь процессор. Такой метод защиты информации называется анклавом или доверенной средой исполнения.

1.2 Доверенная среда исполнения

Доверенная среда исполнения (ДСИ) – специальная изолированная область, предоставляемая процессором, которая позволяет вынести из системы часть функциональности приложений и ОС в отдельное окружение, содержащее память и выполняемый код в которой будут недоступны из основной системы, независимо от уровня текущих привилегий. Так, например, в ДСИ выполняется код отвечающих за реализацию различных алгоритмов шифрования, обработки закрытых ключей, паролей, процедур аутентификации и работы с конфиденциальными данными.

В случае, если система была скомпрометирована, информация хранящаяся в ДСИ не может быть определена, и доступ к ней будет ограничен лишь внешним программным интерфейсом. В отличии от других методов защиты информации, таких как например гомоморфное шифрование, аппаратная реализация ДСИ практически не влияет на производительность системы и

уменьшает время разработки программного обеспечения [1].

С другой стороны, аппаратная реализация ДСИ имеет свои недостатки:

- некоторые современные процессоры имеют лишь частичную поддержку ДСИ либо не имеют её вовсе;
- нет возможности программно исправить уязвимость. Найденные уязвимости в реализации ДСИ могут быть исправлены лишь в новых ревизиях процессора, т.е. без его физической замены, злоумышленник сможет эксплуатировать уязвимость.
- увеличивается издержки производства на разработку таких процессоров – их конечная стоимость возрастает.

Таким образом, в некоторых случаях, появляется необходимость в программной реализации ДСИ. Стоит отметить, что в конечном счёте, программная реализация всё равно использует другие аппаратные механизмы предоставляемые процессором [?].

2 Существующие реализации ДСИ

2.1 ARM TrustZone

ARM TrustZone – технология аппаратного обеспечения ДСИ, разрабатываемая компанией ARM. Большинство процессоров разработанных ARM имеют поддержку TrustZone [5]. Данная технология основана на разделении режимов работы процессора на два ”мира”: обычный мир (Normal World) и безопасный мир (Secure World). Процессор переключается в безопасный мир по запросу (с помощью специальной инструкции), при работе с конфиденциальными данными. Всё остальное время, процессор работает в режиме обычного мира. Процессоры с поддержкой данной технологии имеют способность разделять память, независимо от её типа, на ту, которая доступна только в безопасном мире, и ту, которую можно использовать в обычном мире. ARM предоставляют открытый исход программного обеспечения для поддержки данной аппаратной

технологии – ARM Trusted Firmware [6].

2.1.1 Обычный и безопасный мир

Ключевой особенной ARM TrustZone является способность процессора переключаться между обычным и безопасным миром. Каждый из этих миров управляется собственной операционной системой, которые обеспечивают необходимую функциональность. Основное различие между этими ОС заключается в предоставляемых гарантия безопасности. В один момент времени, процессор может находиться только в одном из двух миров, что определяется значением специального бита NS (Non-Secure), бит является частью регистра Secure Configuration Register (SCR). Этот регистр доступен для периферии только для чтения, изменять его значение может лишь сам процессор. Когда процессор находится в обычном режиме исполнения кода, значение бита NS равно 1, и наоборот, когда процессор находится в безопасном мире, значение бита NS равно 0.

За связь между обычным и безопасным миром отвечает специальный механизм – Secure Monitor. Он соединяет оба мира и является единственной точкой входа в безопасный мир. Для того, чтобы из обычного мира перейти в безопасный, существует специальная инструкция процессоров ARM – Secure Monitor Call (SMC). При вызове данной инструкции процессор передает управление Secure Monitor. Тот в свою очередь готовит систему к переходу из одного мира в другой и передает управление соответствующей ОС. Инструкция SMC используется как для перехода из нормального мира в безопасный, так и для перехода из безопасного в нормальный. Некоторые прерывания или исключения могут быть настроены так, чтобы они так же проходили через Secure Monitor и были обработаны в безопасном мире. ARM предоставляет спецификацию Secure Monitor Call Calling Convention (SMCCC) [7], которая является стандартом при реализации вызовов SMC.

На рисунке 2 представлена цонепутальная схема взаимодействия двух миров.

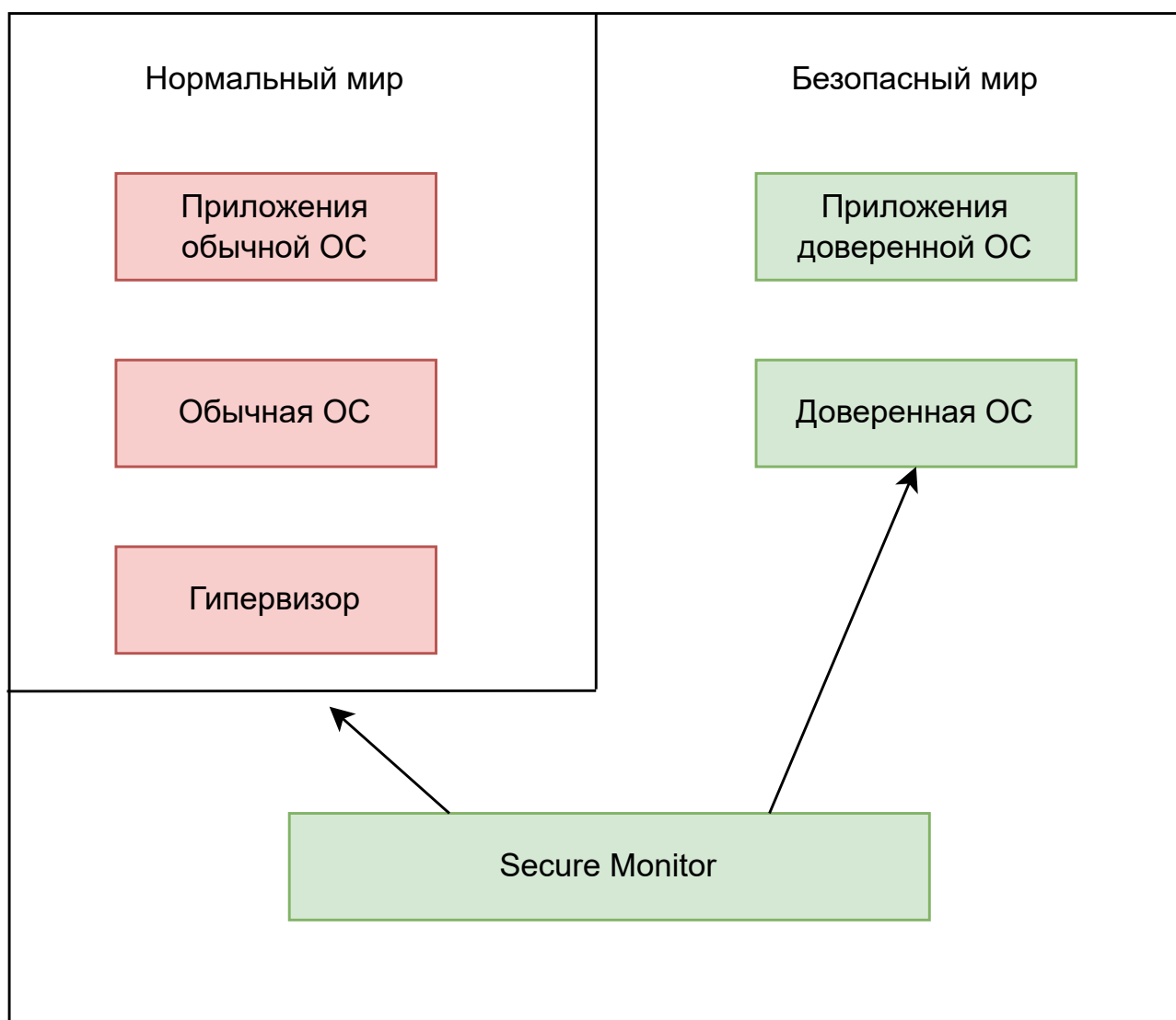


Рисунок 2 – Концептуальная схема взаимодействия двух миров для процессоров ARM Cotrex-A

Доверенная операционная система, так же как и обычная, может запускать приложения. Они, как и в обычном мире, обращаются к доверенной ОС при необходимости получения каких-либо ресурсов или обработки прерываний и исключений. Таким образом, Secure Monitor передаёт управление одному из доверенных приложений, и уже те, в свою очередь, обращаются к доверенной ОС.

ARM предоставляют открытый исходной код эталонной доверенной операционной системы, которая называется OP-TEE [8]. Global Platform предоставляет спецификацию для реализации API взаимодействия доверенных приложе-

ний [9] с доверенной ОС [10].

Физически миры разделены таким образом, что часть регистров доступны только в безопасном мире. Периферия, например память, может быть настроена так, что она может быть доступна лишь в определенном мире. Технология TrustZone в нормальном режиме работы процессора не позволяет программному обеспечению получить доступ к аппаратным средствам, которые могут быть доступны лишь только в безопасном мире.

При сборке компонентов системы, производитель устройства должен позаботиться о конфигурации периферии для работы с TrustZone:

- если предполагается, что периферия может получать доступ к безопасному режиму исполнения, процессор и внешнее устройство должны быть соединены (помимо различных шин) линией NS. Получение сигнал $NS=0$ от процессора к периферии означает, что команда является доверенной (например операция чтения или записи).
- в обратном случае, линия NS может быть опущена. Предполагается, что такая периферия не имеет никаких привилегий, т.е. $NS=1$ всегда.

На рисунке 3 представлена схема взаимодействия процессора и периферии для поддержки TrustZone. Периферийные устройства №1 и №3 соединены линией NS, а устройство №2 нет.

Стоит отметить, что чаще всего не вся периферия соединена сигналом NS с процессором. Например, тот факт, что производитель устройства не соединил сигналом NS камеру и ядра процессора, полностью исключает возможность предоставления пользователю доступа к устройству с помощью технологии распознавания лица.

2.1.2 Режимы работы процессора

Современная архитектура ARM поддерживается три режима работы процессора:

- EL0 – непривилегированный режим работы, предназначенный для исполнения обычных программ;

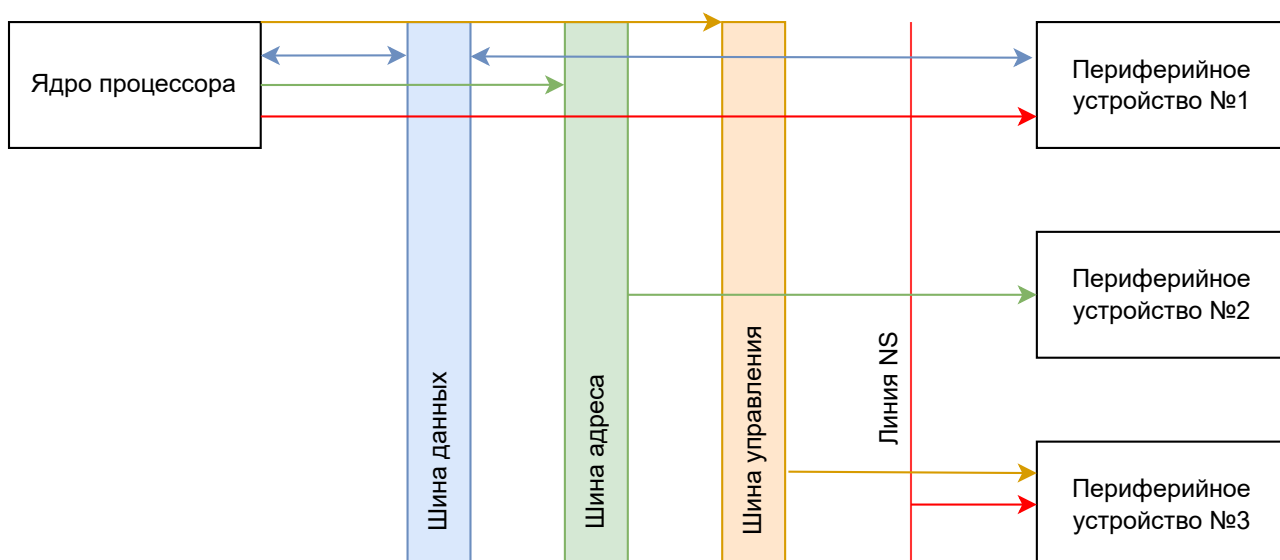


Рисунок 3 – Пример взаимодействия процессора и периферии для поддержки TrustZone

- EL1 – привилегированный режим работы – исполняется кода ОС, обработчиков прерываний и исключений;
- EL2 – режим работы гипервизора.

Для того, чтобы программа исполняемая на уровне EL0 могла перейти в EL1 (например, обратиться к ресурсам доступным только ОС), в архитектуре ARM существует команда Supervisor Call (SVC). Аналогично, команда Hypervisor Call (HVC) предназначена для перехода из режима EL1 в EL2.

Каждый из этих уровней могут исполняться в нормальных (Non-Secure) так и безопасных (Secure) режимах (рис. 4)

- обычные приложения исполняются на уровне Non-Secure EL0, а приложения доверенной ОС на Secure EL0;
- обычная ОС выполняется на уровне Non-Secure EL1; все прерывания и исключения произошедшие в нормальном мире так же обрабатываются на этом уровне; доверенная ОС и прерывания произошедшие в безопасном мире исполняются на уровне Secure EL1;
- Secure Monitor всегда исполняется на уровне Secure EL1;
- гипервизор исполняется в режиме Non-Secure EL2.

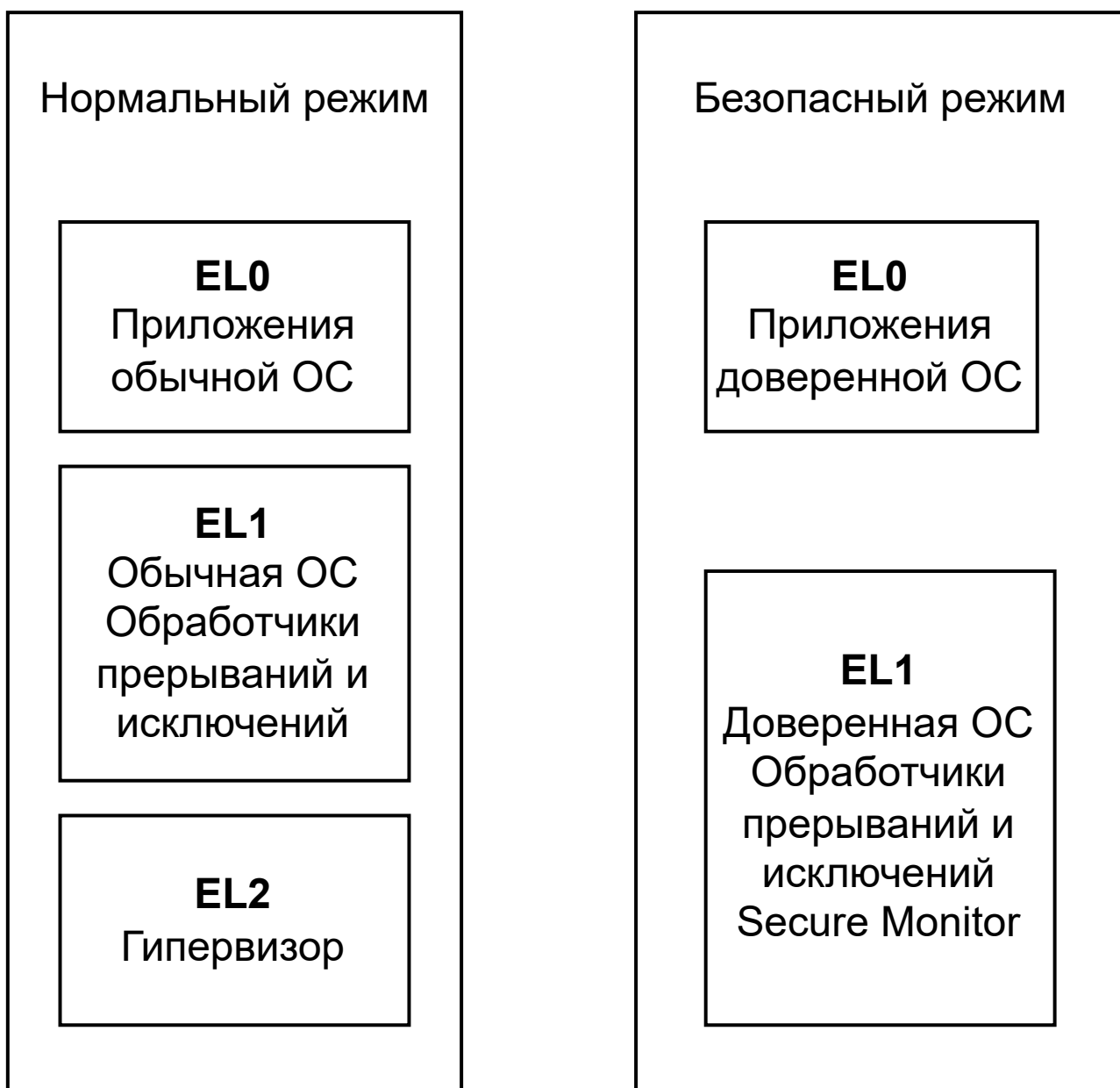


Рисунок 4 – Режимы работы процессоров архитектуры ARM

Бит NS, который был описан в предыдущей главе, определяет то, в каком режиме сейчас выполняется код: обычном (NS=1) или безопасном (NS=0).

Благодаря дублированию нормального и безопасного режима архитектура ARM позволяет запустить сразу две ОС: обычную и доверенную.

2.1.3 Разделение памяти

В целях безопасности в ARM TrustZone память разделяется на защищенную и незащищенную область. Благодаря контроллеру адресного пространства (TZASC – TrustZone Address Space Controller) незащищенная область памяти

может использоваться только из обычного мира, а защищенная из безопасного. Кэш память так же разделяется на защищенную и незащищенную. Необходимо отметить, что данный контролер не является обязательным в архитектуры ARM TrustZone, поэтому разработчики могут принять решение отказаться от них в пользу более компактных и менее энергоемких устройств.

2.1.4 Проверка целостности

Проверка целостности ДСИ является важным механизмом, который не позволит злоумышленнику изменить исходный код доверенной ОС или доверенных приложений. В ARM TrustZone данный механизм реализован на аппаратном уровне как для ОС, так и для приложений: каждый раз, при загрузке доверенной ОС в память, с помощью цифровой подписи проверяются её целостность. Аналогичная схема используется и при загрузке доверенных приложений. Запустить ОС может только подписанные приложения, подпись формируется разработчиками, на стадии компиляции в исполняемый файл.

На данный момент, доверенная среда исполнения от компании ARM не поддерживает процедуру удалённой проверенной проверки (например, с помощью сервера). Существуют лишь программные решения этой проблемы от сторонних разработчиков [5].

2.2 Intel SGX

Intel Software Guard Extension (SGX) – реализация ДСИ от компании Intel, включена в большинство современных процессоров Intel Core. Конфиденциальность и целостность в этой технологии достигается с помощью использования анклава – специальной, зашифрованной области кода и данных. Это достигается с помощью различных компонентов и протоколов, одним из которых является специальная область памяти, называемая Processor Reserved Memory (PRM), обеспечивающая безопасное хранилище, к которому не может обращаться никто, кроме самого процессора. Для выполнения кода, после череды его проверок, он загружается извне в PRM. Процессор с помощью специальных инструкций переходит в режим анклава (enclave mode) и выполняет загружен-

ный код. В технологии Intel SGX именно анклав является доверенной средой исполнения.

2.2.1 Processor Reserved Memory

Processor Reserved Memory (PRM) – защищенная часть оперативной памяти, к которой не имеет доступ код, который исполняет в режиме non-enclave. Это реализуется аппаратно, с помощью специальных контроллеров доступа к памяти.

Данный участок памяти подразделяется на дополнительные разделы:

- Encalve Page Cache (EPC) – разделенная страницы размером 4Кб область памяти, которые хранят код конкретного анклава, к которому относятся; это позволяет использовать в системе несколько анклавов одновременно. EPC управляется программным путём с помощью ОС. Доступ к нему может быть получен только программным обеспечением входящим в данный анклав;
- Enclave Page Cache Map (EPCM) – массив с одной записью для каждой страницы, хранимой в EPC; запись содержит в себе метаданные этой страницы: владелец, виртуальный адрес и т. д.;
- SGX Enclave Control Structure (SECS) – содержит в себе метаданные соответствующего анклава; хранится в специальной страничной части EPC. Страница, ассоциированная с SECS не отображается в память напрямую и доступна только для реализации SGX – это сделано в целях дополнительной безопасности.

Схема представления памяти с использованием Intel SGX представлена на рисунке 5.

Анклав получает доступ к EPC, выделяя часть своей виртуальной памяти под линейный диапазон адресов анклава (Enclave Linear Address Range, ELRANGE), который содержит адреса сопоставленные с EPC. Другие адреса виртуальной памяти отображаются на память расположенную за пределами EPC. Процессор проверяет что в результате трансляции физического адреса

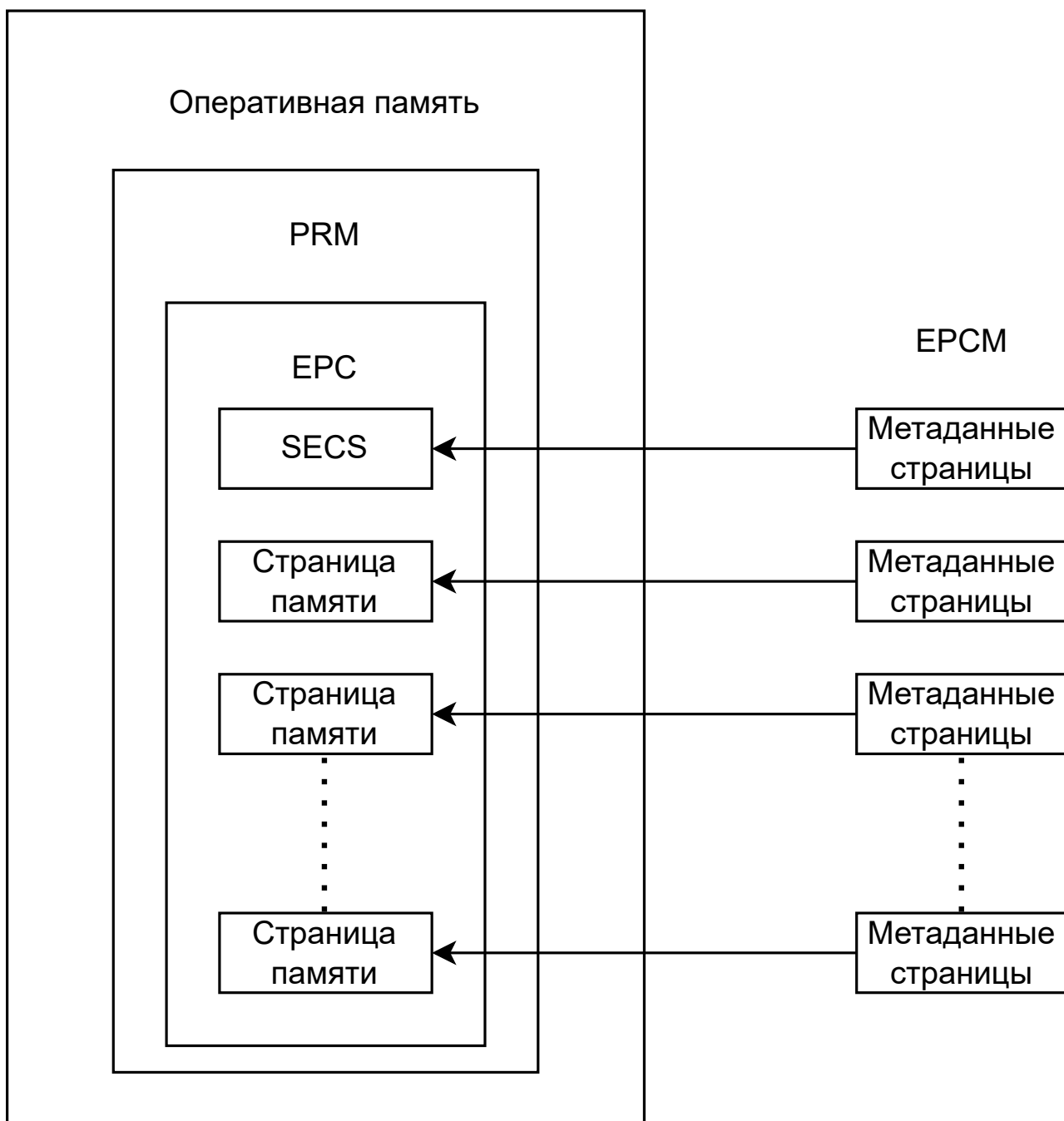


Рисунок 5 – Схема представления памяти Intel SGX

страницы, находящейся в EPC, виртуальный адрес совпадает с адресом хранящимся в EPCM – это позволяет предотвратить атаки на трансляцию адресов.

Каждая страница обладают индивидуальными правами доступа, которые устанавливаются при выделении соответствующей страницы и определяются автором анклава, что так же является дополнительной мерой безопасности. Страницы разделены на те, которым разрешено чтение, запись и выполнение кода

анклава. Эта информация так же находится в метаданных страницы, хранящихся EPCM.

State Save Area (SSA) – ещё один компонент Intel SGX, отвечающий за сохранение текущего состава анклава. Это специальная область памяти, которая используется для хранения контекста выполнения кода анклава. Эта область памяти необходима, например, когда в системе произошло прерывание – процессору необходимо перейти в нормальный режим исполнения для его обработки. После этого, процессор может загрузить состояние анклава из SSA и возобновить выполнение кода.

Ещё одной особенностью с точки зрения безопасности и производительности, является возможность вытеснения страниц из PRM в обычную (не-PRM) память. Большинство современных компьютеров поддерживают избыточное использование оперативной памяти, выгружая некоторые страницы во вторичные устройства. Технология Intel SGX поддерживает это, добавляя меры, которые гарантируют целостность и конфиденциальность вытесняемых страниц: вытесняемая страницы шифруется с помощью симметричного шифрования, а ключ хранится в специально отведенных для этого страницах EPC.

2.2.2 Жизненный цикл анклава

Для управления анклавами Intel предоставляет набор специальных инструкций:

- ECREATE – создаёт новый анклав и сохраняет его метаданные в SECS;
- EADD – используется для добавления новых страниц в анклав; ОС загружает новые страницы в EPC, заполняя необходимые метаданные. Вызывать эту инструкцию можно только после создания анклава, попытка добавить страницы после этого этапа приведёт к ошибке;
- с помощью инструкции EINIT и специального токена от Launch Enclave, процессор начинает выполнять код анклава. Launch Enclave – специальный анклав, проходящий все те же стадии инициализации, что и другие. Его основная цель является выдача токена другим анклавам на основе

списка одобренных анклавов;

- приложения могут выполнить команду `EENTER` для входа в режиме анклава и выполнения там свой кода, и по окончании выйти оттуда используя команду `EEXIT`. В виртуальном адресном пространстве этих приложений должны иметься соответствующие страницы EPC;
- команда `AEX` используется при возникновении прерывания или исключения, после её выполнения процессор переходит в обычный режим исполнения, сохраняя контекст в `SSA`;
- `ERESUME` – возобновить выполнение анклава с контекстом, сохраненным в `SSA`. Стоит отметить, что анклав может иметь более одного `SSA` в случаях, если при выполнении одного и того же блока происходит несколько прерываний.

После выполнения кода, в метаданных страниц, ассоциированных с этим анклавом, выставляется пометка, что они невалидны и очищается TLB кэш. Это позволяет защитить Intel SGX от атак на память. На рисунке 6 представлена схема жизненного цикла анклава с использованием команд, описанных выше.

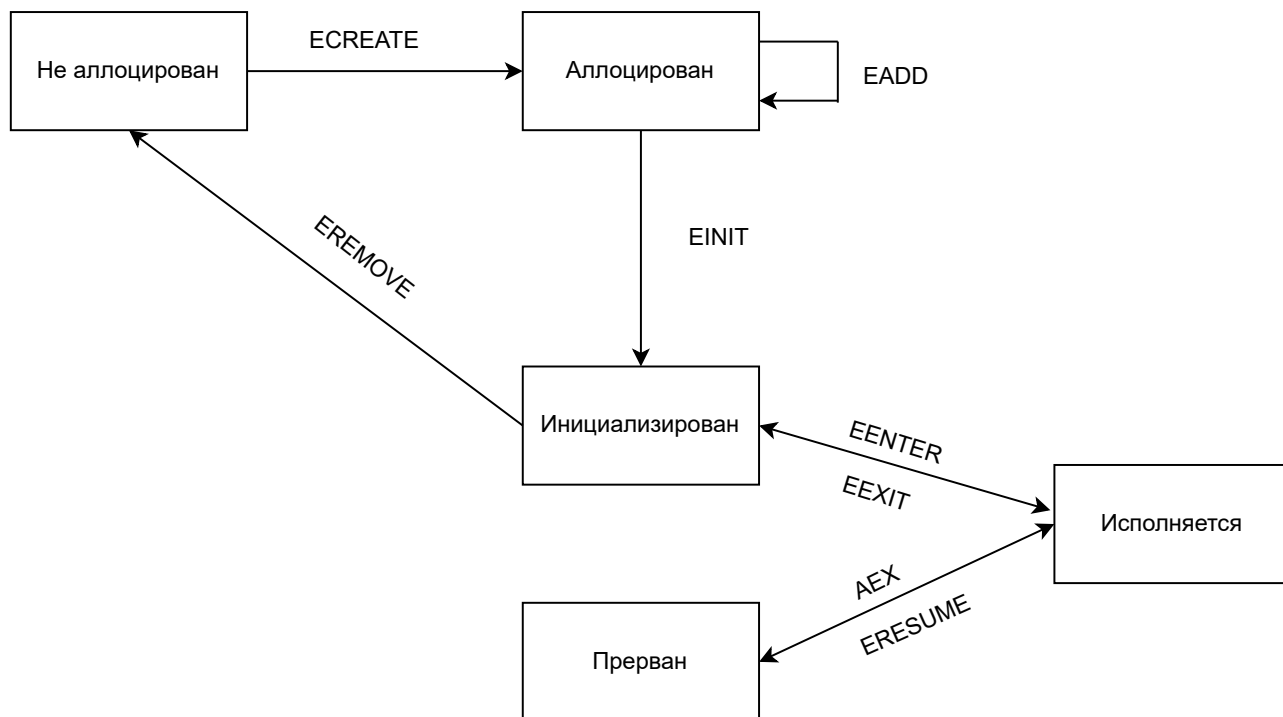


Рисунок 6 – Жизненный цикл анклава

2.2.3 Проверка целостности

В Intel SGX реализована поддержка механизма локальной и удаленной проверки целостности.

Локальная проверка используется для установления канала связи, который гарантирует конфиденциальность, двумя анклавами на одном устройстве. Для обмена симметричным ключом используется протокол Диффи-Хеллмана [11]. Локальная проверка начинается с того, что один из анклавов отправляет значение MRENCLAVE (индивидуальный идентификатор анклава) другому анклаву, который находится на том же устройстве. Отправить называется верификатором, а получающий анклав утверждающим. Отправитель использует полученное от верификатора значение MRENCLAVE для создания отчёта (claimer), который он отправляет обратно верификатору. Отчёт может быть проверен с помощью специального ключа REPORT KEY, который хранится на устройстве и доступен всем анклавам. Он так же содержит данные обмена ключами Диффи-Хеллмана, который в дальнейшем будут использованы для создания защищенного канала связи. После проверки отчёта, верификатор создает и отправляет отчёт для утверждающего анклава. Затем обе стороны могут создать защищенный канал используя данные Диффи-Хеллмана, содержащиеся в обоих отчетах. На рисунке 7 представлена схема локальной проверки целостности в Intel SGX.

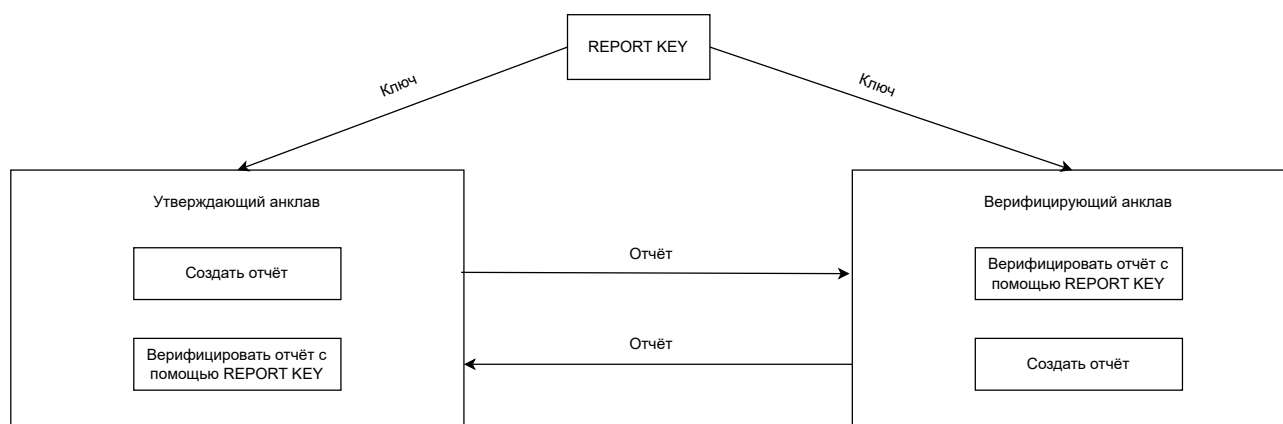


Рисунок 7 – Схема локальной проверки целостности в Intel SGX

Удаленная проверка целостности реализуется с помощью удаленной

службы Intel Attestation Service [13]. В процессе проверки, используется подсчёт хэш-суммы анклава с помощью хэш-функции SHA-2 [12], специального анклава находящегося в однократно записываемой памяти и ключей, которые были расположены в устройстве на стадии его производства. С помощью ключей и хэш-суммы, анклав расположенный в однократно записываемой памяти формирует специальный отчёт, отправляемый в службу Intel Attestation Service, и та, в свою очередь, на основе этого отчёта проверяет целостность анклава.

2.3 Keystone

Keystone – реализация доверенной среды с открытым исходным кодом для процессоров на базе архитектуры RISC-V. В отличие от Intel SGX и ARM TrustZone, эта технология является полностью программным решением с открытым исходным кодом, построенным на использовании аппаратных особенностей архитектуры RISC-V. Keystone предоставляет спецификацию для разработчиков устройств, выполнение которой гарантирует поддержку этого механизма [14].

2.3.1 Компоненты системы

Keystone состоит из нескольких компонентов, каждый из которых выполняется на разном уровне привилегий [15].

Всего есть три уровня привилегий

- 1) U-mode (user) – режим исполнения пользовательских процессов.
- 2) S-mode (supervisor) – режим выполнения кода ядра.
- 3) M-mode (machine) – режим, в котором осуществляется доступ к периферии устройства.

Ниже будут описаны компоненты, с помощью которых строится доверенная среда исполнения Keystone.

Trusted Hardware – совместимые со спецификацией Keystone ядра архитектуры RISC-V и ключи (открытый и закрытый), используемые для подписи анклава. Аппаратное обеспечение также может содержать дополнительные функции, например разделение кэша, шифрование памяти, криптографический

защищенный источник случайных чисел.

Security Monitor (SM) – выполняется в режиме M. Предоставляет интерфейс для управления жизненным циклом анклава, а так же для использования специфических возможностей платформы. SM обеспечивает выполнение гарантий безопасности Keystone, поскольку он отвечает за изоляцию анклавов и обычной (недоверенной) ОС.

Анклавы представляет собой среду, изолированную от операционной системы и других анклавов. Каждому анклаву выделяется отдельная область физической памяти, получить доступ к которой может только он сам и Security Monitor. Каждый анклав состоит из анклавного приложения, выполняемого на уровне пользователя (режим U) и Runtime (режим S).

Приложения анклава (EAPP) – приложение пользовательского уровня, которого выполняется в анклаве. Можно создавать собственное приложение с нуля или просто запустить существующий исполняемый файл.

Runtime – программное обеспечение выполняющиеся в режиме S, реализующие такие следующие возможности: системные вызовы, управление виртуальной памятью, обработка прерываний и так далее.

На рисунке 8 представлена концептуальная схема компонентов системы ДСИ Keystone.

2.3.2 Жизненный цикл анклава

Анклав Keystone может находится в трех состояниях [15].

- 1) Создание. Анклав загружается на непрерывный диапазон физической памяти, которая называется приватной памятью анклава. Недоверенный код (например, операционная система) выделяет эту память и инициализирует таблицу страниц анклава, загружает код компонента Runtime и приложение анклава. Для создания анклава вызывается Secure Monitor, которые изолирует и защищает приватную память с помощью механизма защиты физической памяти, что позволяет защитить память анклава от изменений и чтений для любого ядра процессора. После выполнения этих действий,

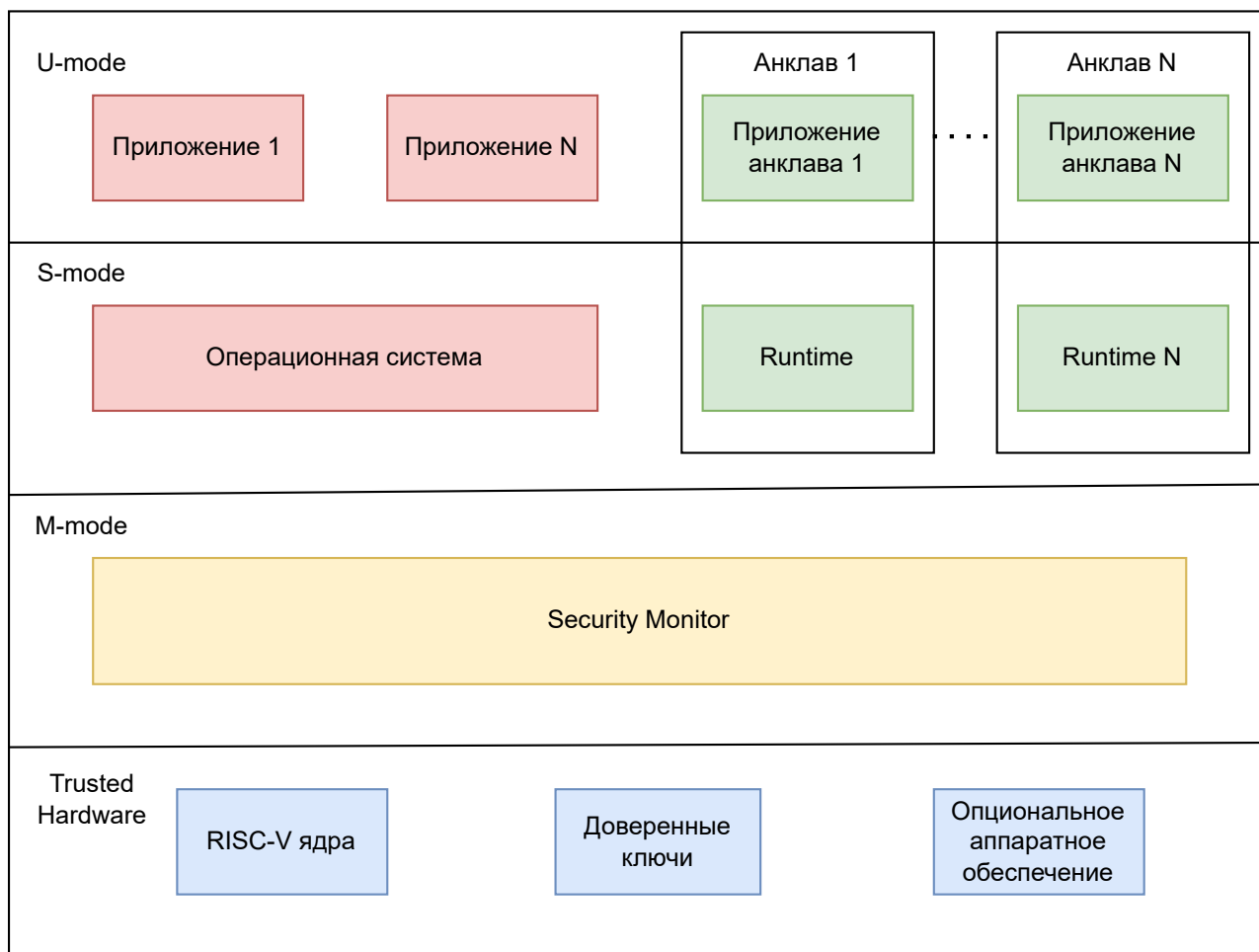


Рисунок 8 – Компоненты системы доверенной среды исполнения Keystone

SM помечает анклав готовым для выполнения;

- 2) **Выполнение.** Недоверенный код запрашивает у SM разрешение на исполнение кода анклава на одном из ядер процессора; SM выдает разрешение на выполнение и ядро начинает выполнять его код. Процесс выполнения может быть прерван (например для обработки прерывания), и, в таком случае, ядро перестанет выполнять код анклава и уберёт разрешение на выполнение его кода.
- 3) **Разрушение.** Недоверенный код может уничтожить анклав в любое время. При его уничтожении, SM освобождает выделенную память для анклава, снимает с неё защиту и передаёт её в распоряжение операционной системы, предварительно заполнив нулями её содержимое.

На рисунке 9 представлена схема жизненного цикла анклава Keystone.

Состояние анклава	Карта памяти	Команды
Создание	<div>Свободная память</div> <div>Таблица страниц RT EAPP Free</div>	<ul style="list-style-type: none"> • Выделить память • Загрузить исполняемый файл • Создать анклав • Пометить готовым к исполнению
Выполнение	<div>Свободная память</div> <div>Память анклава (защищена)</div>	<ul style="list-style-type: none"> • Исполнять код анклава • Прервать/возобновить исполнение • Остановить исполнение
Разрушение	<div>Свободная память</div> <div>0000 0000</div>	<ul style="list-style-type: none"> • Уничтожить анклав • Освободить память

Рисунок 9 – Жизненный цикл анклава

При создании анклава, участок его памяти состоит из таблицы страницы, компонента Runtime, приложения и свободного участка памяти. При его уничтожении, она предварительно заполняется нулями.

2.3.3 Проверка целостности

Архитектура ДСИ Keystone предполагает использование как локальной, так и удаленной проверки целостности анклава.

Для того чтобы локально проверить целостность анклава, при его создании вычисляется хэш-сумма (на основе кода и данных). Далее, анклав генерирует ключ шифрования, который будет использоваться для аутентификации анклава и защиты его кода и данных. Ключ подписывается с помощью доверенного ключа (см. рис. 8), для удостоверения его подлинность. Ключ анклава и информация о нем, включая его хэш-сумму и сертификат, передаются клиенту (например, хост-системе), который будет проверять анклав. Клиент проверяет аутентичность хэш-суммы анклава и его ключа, используя доверенный откры-

тый ключ. Если анклав и его ключ прошли проверку, клиент может быть уверен в его подлинности и целостности.

3 Сравнение реализаций ДСИ

3.1 Критерии сравнения

Для сравнения ранее описанных реализаций ДСИ были выделены следующие критерии:

- К1 - безопасность;
- К2 - производительность;
- К3 - полнота и надежность механизмов проверки целостности ДСИ;
- К4 - наличие открытого исходного кода.

ЗАКЛЮЧЕНИЕ

В ходе выполнения научно исследовательский работы была достигнута ее цель – проведен анализ и сравнение существующих реализаций ДСИ.

Для достижения данной цели были решены следующие задачи:

- проведён обзор существующих реализаций ДСИ;
- описаны плюсы и недостатки каждой из реализаций;
- сформулированы критерии сравнения;
- проведено сравнение существующих реализации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Introduction to Trusted Execution Environments – Global Platform [Электронный ресурс]. – Режим доступа: <https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf>, свободный – (09.10.2023)
2. Intel | Data Center Solutions, IoT, and PC Innovation [Электронный ресурс]. – Режим доступа: <https://www.intel.com/>, свободный – (10.11.2023)
3. Building the Future of Computing – Arm® [Электронный ресурс]. – Режим доступа: <https://www.arm.com>, свободный – (09.10.2023)
4. The Security Paradox of Complex Systems, 2002. David Woods, Nancy Leveson, Brian Rebentisch. с. 1 - 15.
5. Comparison of Prominent Trusted Execution Environments, 2022. Xiaoyu Zhang [Электронный ресурс]. – Режим доступа: https://elib.uni-stuttgart.de/bitstream/11682/12171/1/Zhang_Xiaoyu_Prominent_Trusted_Execution_E – (22.10.2023)
6. TrustedFirmware-A (TF-A) | ARM [Электронный ресурс]. – Режим доступа: <https://www.trustedfirmware.org/projects/tf-a/> – (22.10.2023)
7. Secure Monitor Calling Convention (TF-A) | ARM [Электронный ресурс]. – Режим доступа: <https://developer.arm.com/Architectures/SMCCC> – (22.10.2023)
8. OP-TEE | ARM [Электронный ресурс]. – Режим доступа: <https://www.trustedfirmware.org/projects/op-tee/> – (22.10.2023)
9. Global Platform – TEE Client API Specification [Электронный ресурс]. – Режим доступа: https://globalplatform.org/wp-content/uploads/2010/07/TEE_Client_API_Specification-V1.0.pdf – (22.10.2023)

10. Global Platform – TEE Internal Core API Specification [Электронный ресурс]. – Режим доступа: <https://globalplatform.org/specs-library/tee-internal-core-api-specification/> – (22.10.2023)
11. Diffie-Hellman key agreement | IBM [Электронный ресурс]. – Режим доступа: <https://www.ibm.com/docs/en/zos/2.1.0?topic=ssl-diffie-hellman-key-agreement> – (27.10.2023)
12. FIPS 180-2, Secure Hash Standard [Электронный ресурс]. – Режим доступа: <https://csrc.nist.gov/files/pubs/fips/180-2/final/docs/fips180-2.pdf> – (27.10.2023)
13. Attestation Services for Intel® Software Guard Extensions [Электронный ресурс]. – Режим доступа: <https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/attestation-services.html> – (27.10.2023)
14. Keystone Enclave Documentation [Электронный ресурс]. – Режим доступа: <https://buildmedia.readthedocs.org/media/pdf/keystone-enclave/docswork/keystone-enclave.pdf> – (01.11.2023)
15. Keystone Basics | Keystone Enclave [Электронный ресурс]. – Режим доступа: <http://docs.keystone-enclave.org/en/latest/Getting-Started/How-Keystone-Works/Keystone-Basics.html#overview> – (01.11.2023)