Федеральное государственное бюджетное учреждение науки
Санкт-Петербургский институт информатики и автоматизации
Российской академии наук
(СПИИРАН)

На правах рукописи

Биричевский Алексей Романович

Методы защиты информации на основе псевдовероятностного преобразования для мобильных устройств телекоммуникационных систем

Специальность 05.13.19 – Методы и системы защиты информации, информационная безопасность

Диссертация на соискание ученой степени
Кандидата технических наук

Научный руководитель
д.т.н, профессор
Молдовян Н.А.

Санкт-Петербург – 2017

Оглавление

Введение

Актуальность темы исследования. Большинство практических задач обеспечения информационной безопасности информационно-телекоммуникационных систем решается на программно-аппаратном уровне с использованием разнотипных операционных систем (ОС), хотя встроенные механизмы их функционирования являются весьма схожими. Встраивание в ОС механизмов аутентификации и защиты информации сокращает сроки и затраты на разработку прикладного программного обеспечения для мобильных устройств различного типа и назначения.

Средства алгоритмической защиты информации широко применяются на практике, однако их уязвимой частью является согласование параметров защитного преобразования взаимодействующими в ходе информационного обмена пользователями. Для перехвата этих параметров злоумышленник может применять принуждающие атаки, т.е. средства подкупа и специальные средства воздействия на пользователя. Для защиты пользователей от принуждающих атак предложено применение псевдовероятностного защитного преобразования. Интеграция псевдовероятностного защитного преобразования в подсистему защиты информации универсальной ОС обеспечит пользователю системы более высокий уровень защиты от принуждающих атак.

Данная работа посвящена решению актуальных научных и практических проблем: расширения функциональности средств защиты информации, обеспечения переносимости программных средств защиты информации на различные типы мобильных устройств (на различные типы технических платформ) и встраивания механизмов защиты от атак с принуждением.

Степень разработанности темы. Исследования методов обеспечения информационной безопасности операционных систем освещены в работах Зыль С. и Махилёва В., Оладько А.Ю., Столлингс В. [1], Шаньгина В.Ф. [2][3], Безбогова А.А. [4], Котенко И.В, Молдовяна А.А. [5], Саенко И.Б., Лорина Г. [6], Дейтеля Х.М. и др.. Вопросы защиты информации от несанкционированного

доступа освещены в работах Девянина П.Н., Семкина С.Н. и др. [7] - [11]. Исследования в области разработки псевдовероятностных защитных преобразований приведены в работах Молдовяна Н.А.[12], Щербаковой В.А. [13], Березина А.Н. [14] и др.[15] - [17].

Цель и задачи исследования. Цель данной работы состоит в сокращении сроков и уменьшении затрат по разработке защищенных мобильных информационных технологий за счет расширения функциональности и обеспечения переносимости программных средств защиты информации на различные типы мобильных устройств (на различные типы технических платформ) и встраивание механизмов защиты от атак с принуждением.

Для решения поставленной цели были сформулированы и решены следующие исследовательские **задачи:**

– выполнение анализа функциональных возможностей и особенностей реализации существующих мобильных операционных систем и на его основе разработка модели угроз информационной безопасности объекта исследования, архитектуры и программного кода универсальной защищенной операционной системы для мобильных систем;

– разработка метода аутентификации пользователей, стойкого к принуждающим атакам;

– разработка метода защитного преобразования передаваемой по открытым каналам информации, стойкого к атакам с принуждением пользователя раскрыть ключ защитного преобразования;

– разработка метода защиты программного обеспечения от дизассемблирования;

– разработка метода защиты хранимой информации, стойкого к атакам с принуждением пользователя раскрыть ключ защитного преобразования.

Научная новизна диссертационного исследования заключается в следующем:

1. Разработан метод аутентификации пользователей, отличающийся использованием одноразовых паролей, генерируемых с помощью алгебраического алгоритма псевдовероятностного защитного преобразования.

2. Разработан метод защитного преобразования передаваемой по открытым каналам информации, отличающийся выполнением требования вычислительной неразличимости по шифртексту от вероятностного защитного преобразования.

3. Разработан метод защиты программного обеспечения от дизассемблирования, отличающийся введением ложных веток кода с помощью псевдовероятностного защитного преобразования машинного кода.

4. Разработан новый метод хранения ключей шифрования, отличающийся выполнением псевдовероятностного защитного преобразования ключей.

Теоретическая и практическая значимость работы. Теоретическая значимость работы состоит в разработке архитектуры универсальной защищенной мобильной ОС и новых алгоритмах защитных преобразований информации, обеспечивающих вычислительную неразличимость по шифртексту от вероятностного защитного преобразования. Практическая значимость состоит в том, что применение универсальной операционной системы в мобильных устройствах телекоммуникационных и информационных систем, в том числе в системах защиты информации, позволит унифицировать подходы к обеспечению безопасности при разработке таких систем. Данный подход упростит разработку и производство мобильных устройств. Область применения разработанной ОС включает разработку защищенных аутентифицирующих устройств (токенов, идентификаторов), систем охраны, устройств защиты программного обеспечения, персональных устройств хранения данных (защищенных файловых хранилищ), аппаратных средств для выполнения защитных преобразований данных.

Методология и методы исследования. В работе использован аппарат и методы математической статистики, теории вероятности, алгебры, теории чисел, криптографии и вычислительные эксперименты. *Объектом* исследования являются мобильные операционные системы; *предметом* – способы, алгоритмы и

протоколы обеспечения информационной безопасности в операционных системах.

Положения, выносимые на защиту:

1. Метод аутентификации пользователей по одноразовым паролям, обеспечивающий защиту от принуждающего несанкционированного доступа.

2. Метод защитного преобразования передаваемой по открытым каналам информации, обеспечивающий защиту от атак с принуждением к раскрытию ключа защитного преобразования.

3. Метод защиты программного обеспечения от активного и пассивного дизассемблирования, существенно повышающий вычислительную трудоемкость дизассемблирования машинного кода.

4. Метод хранения ключей шифрования, обеспечивающий возможность сокрытия наличия резервных серий ключей.

Степень достоверности и апробация результатов. Обоснованность научных положений, выводов и практических рекомендаций, полученных в диссертационной работе, обеспечивается анализом состояния исследований в данной области на сегодняшний день, формальными доказательствами, вычислительным экспериментом и апробацией результатов на всероссийской научно-практической конференции с международным участием «Комплексная защита объектов информатизации и измерительные технологии» (Санкт-Петербург, 16-18 июня 2014), юбилейной XIII Санкт-Петербургской международной конференции «Региональная информатика (РИ-2012)» (Санкт-Петербург, 24-26 октября 2012), VI межрегиональной научно-практической конференции «Информационная безопасность и защита персональных данных» (Брянск, 2014), VIII Санкт-Петербургской межрегиональной конференции «Информационная безопасность регионов России (ИБРР-2013)» (Санкт-Петербург, 23-25 октября 2013 г), IX Санкт-Петербургской межрегиональной конференции «Информационная безопасность регионов России (ИБРР-2015)» (Санкт-Петербург, 28-30 октября 2015 г).

Результаты диссертационной работы внедрены в учебный процесс кафедры информационной безопасности Института точных наук и информационных технологий Сыктывкарского государственного университета на старших курсах обучения студентов по специальности «090900 – Информационная безопасность».

Основные результаты диссертации изложены в 12 публикациях, в том числе, в 3 статьях, опубликованных в ведущих рецензируемых журналах, входящих в перечень ВАК, в 3 докладах на международной конференции и 6 докладах на российских конференциях.

Структура и объем работы. Диссертационная работа изложена на 154 страницах, включает 4 главы, 38 рисунков, 7 таблиц и список литературы из 112 наименований.

В первой главе была рассмотрена структура объекта исследования – мобильной операционной системы.

Далее были рассмотрены особенности реализации мобильных операционных систем на примере операционных систем для смарт-карт. В каждой из представленных операционных систем были выделены особенности реализации. На основе полученных данных об архитектуре существующих мобильных систем была построена модель угроз операционной системы для мобильных систем.

Вторая глава посвящена разработке методов аутентификации пользователей, контроля и управления доступом и методов активной защиты. Была рассмотрена подсистема разграничения доступа, которая является наиболее важной в составе операционной системы.

С целью повышения эффективности функционирования подсистемы аутентификации были разработаны: алгоритм аутентификации пользователей системы, протокол аутентификации на одноразовых паролях, которые обеспечивают защиту от атак с принуждением. Данные алгоритмы предназначены для защиты пользователей от принуждающей атаки.

Третья глава посвящена разработке методов защиты хранимой и передаваемой информации, стойких к атакам с принуждением пользователя раскрыть ключ защитного преобразования. В работе представлены разработанные алгоритмы преобразований данных, обеспечивающих защиту от атак с принуждением. Также описаны эффективные методы применения данных алгоритмов для защиты данных в операционной системе.

В четвертой главе описывается архитектура разработанной защищенной операционной системы для мобильных систем. Описываются подсистемы разработанной операционной системы, такие как методы защиты программного обеспечения от анализа, архитектуру виртуальной программной среды, резервирования, безопасного обновления и т.д.. Далее был произведен анализ эффективности применения выделенных средств защиты в мобильной ОС. При оценке принималась во внимание специфика применения мобильной ОС.

Основные обозначения и сокращения

JCOP	–	Java Card Open Platform
PAM	–	Pluggable Authentication Modules
НДВ	–	недекларированными возможностями
НСД	–	несанкционированный доступ
ОС	–	операционная система
ОШ	–	отрицаемое шифрование
СЗИ	–	средство защиты информации
СКЗИ	–	средство криптографической защиты информации
УЭК	–	универсальная электронная карта
ЧПУ	–	числовое программное управление

Глава 1. Обзор существующих мобильных операционных систем

1.1 Понятие операционной системы

Операционная система — комплекс программ, который управляет ресурсами компьютерной системы, осуществляет организацию вычислительных процессов в широком смысле и обеспечивает взаимодействие между пользователями, программистами, прикладными программами, системными приложениями и аппаратным обеспечением компьютера.[18]

Процессор имеет большое количество аппаратных ресурсов (регистры, оперативная память, порты ввода - вывода). Аппаратная организация ресурсов зависит от используемой архитектуры ядра процессора. Поэтому прикладному программному обеспечению придется адаптироваться под конкретную аппаратную архитектуру. В работах [19] - [21] описаны основные требования к функциональности операционных систем. Одна из основных задач операционной системы - это управление ресурсами. Операционная система предлагает прикладному программному обеспечению простой и унифицированный механизм доступа к ресурсам. В данном случае операционная система представляет собой прослойку между аппаратным обеспечением и прикладной программой (рисунок 1).

Рисунок 1. Взаимодействие приложений с аппаратными ресурсами

Так как операционная система расположена между аппаратными ресурсами и прикладной программой, система не только может организовать унифицированный доступ к ресурсам, но и проконтролировать правомочность такого доступа.

Операционная система представляет собой достаточно сложный продукт, состоящий из множества различных функциональных подсистем. «Наиболее общим подходом к структуризации операционной системы является разделение всех ее модулей на две группы ...» [22].

Первую группу модулей системы условно называют **ядром** системы. Ядро — центральная часть операционной системы, обеспечивающая приложениям координированный доступ к ресурсам компьютера, таким как процессорное время, оперативная память, внешнее оборудование. Ядро включает в себя наиболее критичные для работы операционной системы (низкоуровневые) подсистемы (подсистемы управления памятью, обработчики системных прерываний, стеки протоколов и другие).

Вторая группа модулей выполняет вспомогательные (дополнительные или высокоуровневые) функции операционной системы. Примером вспомогательного модуля может служить модуль архивирования данных. Прямого влияния на работу операционной системы модуль архивирования не оказывает.

Для надежного управления ходом выполнения приложений операционная система должна иметь по отношению к приложениям определенные привилегии. Иначе некорректно работающее приложение может вмешаться в работу ОС и, например, разрушить часть ее кодов. Все усилия разработчиков операционной системы окажутся напрасными, если их решения воплощены в незащищенные от приложений модули системы, какими бы элегантными и эффективными эти решения ни были. [18]

Ядро системы может выполняться в привилегированном (резидентном) режиме. Вспомогательные модули ОС, в зависимости от подразделения, могут иметь различные группы приоритетов. Например, можно разделить на функциональные группы:

утилиты – программы, необходимые для сопровождения операционной системы (файловый менеджер, архиватор);

системные программы – текстовый редактор, консольный интерпретатор, компилятор, отладчики;

прикладные приложения – пользовательский интерфейс, калькулятор, офисные приложения;

библиотеки процедур – набор, импортируемых функций операционной системы, который используется при написании прикладных приложений (например, библиотека процедур доступа к файловой системе).

Некоторые аппаратные платформы имеют возможность реализации приоритезации процессов на аппаратном уровне. К таким платформам относятся, например, микроконтроллеры на базе платформ ARM (ARM7TDMI, CORTEX-M3 и др.)**.** В ядре CORTEX-M3 поддерживает приоритеты процессов на аппаратном уровне:

– **privileged mode** – привилегированный режим (используется для выполнения функций ядра);

– **user mode –** пользовательский режим(используется в прикладных приложениях).

В зависимости от выбранного режима работы ядра CORTEX-M3 аппаратные (низкоуровневые) команды имеют различные права на доступ ресурсам контроллера. Например, в пользовательском режиме операция чтения из регистра состояния процессора приведет к возникновению исключения типа HardFault.

Дополнительно для выделения дополнительных групп приоритетов в ядре CORTEX-M3 возможно использовать 15 приоритетов прерываний (например, для реализации функций файловой системы). Кроме того, данное ядро имеет в своем составе специализированный модуль управления прерываниями (с учетом приоритета), который называется NVIC (Nested Vectored Interrupt Controller) – контроллер приоритетных векторных прерываний.

Структурное разделение модулей операционной системы повышает расширяемость операционной системы. В зависимости от особенностей

применения операционной системы в ядро включают различный набор подсистем. В работах [23] - [26] описаны примеры применения различных архитектур ядра операционной системы в зависимости от назначения операционной системы. Выделяют несколько основных архитектур ядра операционных систем:

- монолитное ядро;
- модульное ядро;
- микроядро:
- экзоядро;
- др.

Монолитное ядро – классическая архитектура ядер операционных систем. Все части монолитного ядра работают в одном адресном пространстве.

Модульное ядро – модификация архитектуры монолитных ядер.

В данной архитектуре некоторые функции выделяются в отдельные модули (например, драйвера). Изменение аппаратной архитектуры не влияет на работу всей ОС (необходимо лишь заменить необходимый модуль).

Микроядро включает минимальный набор функций для работы с оборудованием. Второстепенные функции ОС выполняются специализированными служебными приложениями (так называемыми сервисами).

Экзоядро – ядро операционной системы, предоставляющее лишь функции для взаимодействия между процессами и управления ресурсами. Данное ядро нередко применяется в гипервизорах виртуальных машин. В данном случае нет необходимости реализовывать большое количество функций (так как они также реализованы в дочерних операционных системах).

1.1.1 Механизм прерываний

Также крайне важным в рамках операционных систем является вопрос обработки системных прерываний. Согласно [18] «прерывания представляют собой механизм, позволяющий координировать параллельное функционирование

отдельных устройств вычислительной системы и реагировать на особые состояния, возникающие при работе процессора». В зависимости от аппаратной платформы реализация прерываний может быть различным. Общим остается только принцип работы. Прерывание - это событие, которое немедленно переключает процессор на выполнение специализированного участка кода - обработчика прерывания. Аппаратный модуль процессора, выполняющий операции по обработке прерываний, называют контроллером прерываний. Прерывания могут быть различного рода. Основные распространенные типы прерываний:

1. прерывания системного таймера (обычно применяется для выполнения периодичных операций);
2. прерывания интерфейсов ввода-вывода;
3. прерывания сброса (возвращает процессор в начальное состояние);
4. прерывание "критическая ошибка" (происходит при выполнении процессором "невыполнимой" команды – например, ссылка на несуществующее пространство в памяти).

В контроллерах также имеет смысл использовать принцип приоритетов. В работах [27], [28] рассмотрена проблема реализации приоритезации прерываний.

Процесс выполнения прерывания в наиболее простом контроллере прерываний имеет одинаковый приоритет всех прерываний и два режима работы ядра процессора (пользовательский и привилегированный режим). Процесс прерывания начинается с возникновения прерывания (рис. 2). Далее контроллер прерываний сохраняет контекст (окружение) пользовательского приложения, переключает процессор в привилегированный режим и передает управление соответственному обработчику прерывания. После выполнения обработчика прерывания происходит переход процессора обратно в пользовательский (не привилегированный) режим. Далее контроллер прерываний восстанавливает контекст (окружение) пользовательского приложения и приложение продолжает «прерванное» выполнение программы.

Рисунок 2. Процесс выполнения процесса прерывания**.**

В современных процессорах применяется дополнительная приоритезация прерываний. Приоритеты прерываний позволяют эффективно обрабатывать события параллельно возникающих прерываний. Прерывание может, как прерывать обработчики прерываний с более низким приоритетом, так и выполняться сразу после завершения прерывания с более высоким приоритетом.

1.1.2 Понятие процесса

Следующим ключевым понятием в операционных системах является процесс. Согласно [22] «процессом является выполняемая программа, вместе с текущими значениями счетчика команд, регистров и переменных». Третья задача операционной системы это распределение ресурсов процессора между процессами (реализация многопоточного режима выполнения прикладных программ). К таким ресурсам относятся оперативная память, регистры процессора и процессорное время.

На рис. 3 изображено распределение процессорного времени между несколькими прикладными процессами.

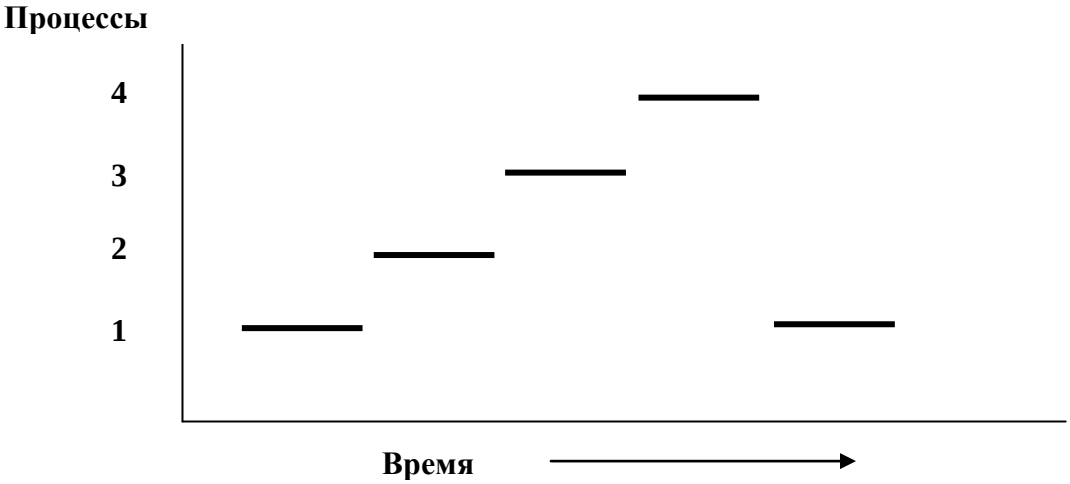


Рисунок 3. Распределение процессорного времени между процессами

Жизненный цикл процесса в системе представляет собой множество состояний, в которых процесс может находиться. «Самую простую модель можно построить, исходя из того, что в любой процесс времени процесс либо выполняется, либо не выполняется» [1]. На рисунке 4а изображена модель процесса с двумя состояниями. Однако данная модель является крайне ограниченной. Модель с двумя состояниями не описывает в частности такие состояния как:

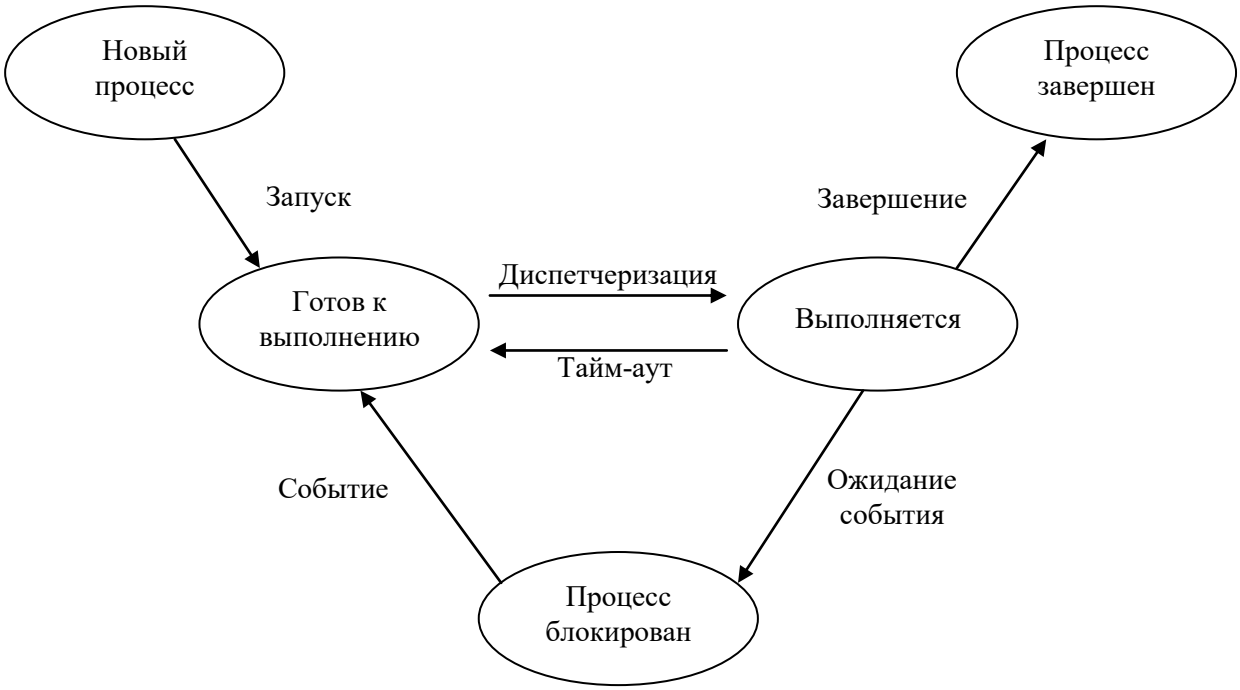
– запуск (создание) процесса;

– блокирование процесса (происходит, например, когда процесс ожидает окончания операций ввода-вывода);

– завершение процесса.

Большей информативностью обладает модель процесса с пятью состояниями (рис. 4б). Данная модель процесса применяется в большинстве современных операционных систем (например, в ядре операционной системы Linux[29])**.**

а) Модель процесса с двумя состояниями



б) Модель процесса с пятью состояниями

Рисунок 4. Модели процесса

Модель с пятью состояниями позволяет описать весь цикл жизни процесса в операционной системе:

– создание (запуск) процесса (переход из состояния "Новый процесс" в состояние "Готов к выполнению");

– выделение процессорного времени процессу при диспетчеризации (переход из состояния "Готов к выполнению" в состояние "Выполняется");

– переход в ждущий режим (переход из состояния "Выполняется" в состояние "Готов к выполнению");

– блокировка процесса (переход из состояния "Выполняется" в состояние "Процесс блокирован") - происходит вследствие "вынужденной" приостановки программы процесса для ожидания выполнения внешнего события;

– активация процесса (переход из состояния "Процесс блокирован" в состояние "Готов к выполнению") - происходит при наступлении ожидаемого события;

– завершение процесса (переход из состояния "Выполняется" в состояние "Процесс завершен").

Каждая прикладная программа представляет собой набор низкоуровневых команд, которые располагаются в основной памяти процессора.

Для старта программы в качестве процесса ядру операционной системы необходимо иметь следующие данные:

– указатель начала программы в основной памяти;

– указатель на стек процесса (область в оперативной памяти);

– размер стека процесса;

– дополнительные аргументы (приоритет процесса, родитель и т.д.).

Данный набор сведений является минимально необходимым для выполнения ядром операционной системы операции переключения контекста (окружения) процесса. Переключение "окружения" процесса необходимо для организации многопоточного режима работы операционной системы. Процесс переключения контекста процесса изображен на рисунке 5.

Переключение контекста начинается с перехода процессора в привилегированный режим (системное прерывание). Ядро системы производит перенос (копирование) активного контекста (счетчик команд, регистры процессора) в область оперативной памяти (стек) соответственного активного процесса (процесс №3). Далее выбирается процесс, который будет загружен (процесс №1). После этого ядро системы копирует контекст процесса №1 из области оперативной данных (стек) в регистры процессора. На заключительном этапе происходит возврат процессора из привилегированного режима в обычный. Процессор "продолжает" выполнение процесса №3.

Рисунок 5. Процесс переключения контекста

С точки зрения процесса выполнение его программы идет в последовательном (однопоточном) режиме. Каждый процесс имеет свое собственное окружение (состояние регистров, счетчик команд, стек для хранения переменных, и тд.). Данный факт является крайне полезным при обеспечении безопасности функционирования процессов. Так как регистры, счетчик команд и стек имеет фиксированное расположение в памяти (например, выход за границы стека в архитектуре процессоров ARM приведет к исключению "критическая ошибка"), остается только проследить за прямым доступом процесса к памяти. Данная проблема может быть исключена на аппаратном уровне, используя контроллер памяти (при его наличии) или запретом прямого доступа к памяти.

1.1.3 Управление памятью

Вторым по значимости ресурсом в операционной системе является быстродействующая оперативная память процессора. В мобильных устройствах количество оперативной памяти весьма ограничено. Так как абсолютно каждое прикладное приложение в системе в каком-либо виде использует некоторое количество оперативной памяти, то эффективность использования ограниченного количества оперативной памяти напрямую влияет на функциональность системы (ограничивается количество одновременно запущенных процессов).

Повышение эффективности использования оперативной памяти может осуществляться различными методами. Один из таких методов — это анализ и оптимизация использования переменных на этапе проектирования приложения. Методы контроля эффективности использования оперативной памяти:

1. контроль неиспользуемых переменных (переменные, которые были задекларированы, но не были использованы в теле программы);
2. перенос постоянных переменных из области оперативной памяти в постоянную память (применение директивы static);
3. контроль полного использования массива данных (в случае, когда программист декларирует массив данных с заведомо большим количеством элементов);
4. контроль возврата выделенной памяти (после использования памяти выделенной методом malloc() следует освободить память методом free()).

Контроллер памяти - программный или программно-аппаратный модуль, входящий в состав ядра операционной системы, который обеспечивает контроль использования оперативной памяти процессора. Контроллер памяти также может применяться для контроля эффективности использования оперативной памяти. Контроллер памяти применяется и для обеспечения безопасности данных процессов.

Кроме контроля эффективности использования оперативной памяти модуль контроля памяти должен обеспечивать следующие методы:

1. выделение области памяти;
2. освобождение ранее выделенной области памяти;
3. операции доступа к памяти (чтение, запись).

Использование контроллера памяти может помочь полностью исключить прямой доступ к оперативной памяти процессами.

Важным вопросом использования памяти является принцип организации адресации ячеек памяти. Выделяют следующие типы адресации.

Абсолютная адресация представляет собой наиболее простой способ адресации. Каждая ячейка в памяти имеет свой собственный уникальный адрес. Данный вид адресации может быть использован как низкоуровневыми приложениями (ядром, системными службами), так и пользовательскими приложениями. Минусом такого вида адресации является ограничение размера общей памяти от размера адреса (например, при размере адреса в 4 бита может быть всего 16 блоков памяти).

Сегментная адресация чаще всего используется для организации адресации к памяти внутри определенного приложения. Адрес высчитывается относительно начала сегмента (например, начала блока данных приложения).

Дополнительно выделяют следующие типы адресации:

– относительная – указывается смещение относительно определенного значения;

– косвенная– указывается адрес ячейки, в которой содержится адрес необходимой ячейки (ссылка на ячейку);

– индексная – порядковый номер элемента в массиве данных;

– регистровая – представляет собой уникальный идентификатор регистра процессора (регистра общего назначения, регистра аппаратного обеспечения).

1.1.4 Файловая система

При работе процесса иногда необходимо иметь возможность долговременного хранения данных. Для быстрого, структурированного доступа к памяти в операционной системе должна быть реализована файловая система. Согласно [30] «под файлом обычно понимают именованный набор данных, организованных в виде совокупности записей одинаковой структуры». Для структуризации хранения файлов и организации доступа к файлам и необходима система управления файлами (файловая система).

Файловая система представляет собой набор методов и объектов (таблиц размещения файлов, таблиц свойств файлов, и т.д.), предназначенных для

организации систематизированного доступа к постоянной памяти. Минимально файловая система операционной системы должна предоставлять ряд следующих операций:

1. стандартные операции с файлами (создание файла, удаление файла, переименование файла);
2. стандартные операции доступа к файлам (чтение из файла, запись в файл, чтение и запись атрибутов файла);
3. операции управления аппаратными устройствами хранения данных (FLASH-память, NAND-память, NOR-память и другие).

Классическая файловая система размещается в памяти в трех основных блоках (областях памяти):

1. загрузочный блок - область памяти, где содержатся служебные данные, которые необходимы для первичной загрузки системы;
2. блок хранения заголовков объектов файловой системы (иногда данную область дублируют для достижения большей отказоустойчивости);
3. блок хранения данных.

Для организации простого и логичного доступа к файлам в операционной системе применяют:

1. «простое» именование файлов - файлу присваивается дополнительное имя (обычно строковое значение) удобное для представления человеком;
2. применение специального типа файлов - «каталог».

«Каталог» представляет собой специальный тип файла, в котором хранятся указатели на другие файлы (аналогично «папке» в бумажном делопроизводстве). Понятие каталога позволяет применять в файловой системе полноценную иерархическую (древовидную) организацию файлов (рис. 6). Древовидная структура всегда содержит в себе хотя бы один каталог. Данный каталог называется корневым каталогом дерева файлов. Корневой каталог имеет фиксированное расположение в памяти. Каждый каталог может включать в себя как файлы, так и другие (дочерние) каталоги.

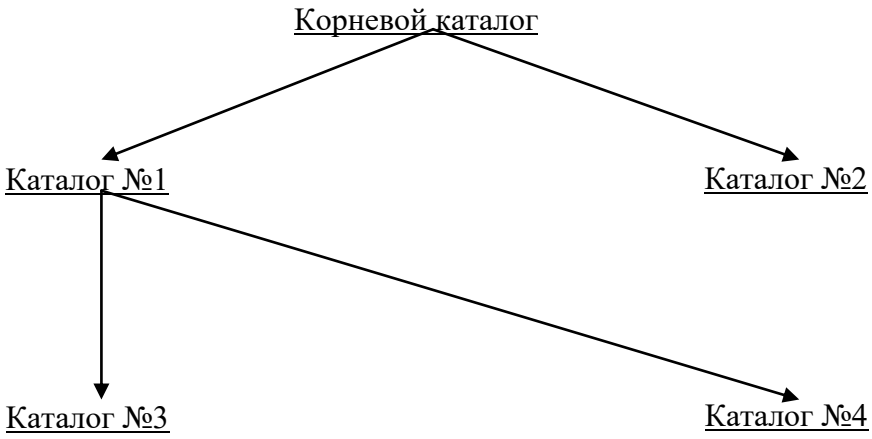


Рисунок 6. Древовидная структура каталогов

Так как каждый файл поименован (имеет уникальный идентификатор), то до каждого объекта в файловой системе может быть найден уникальный путь.

Простое именование файлов в виде текстовой строки (обычно ограниченной длины) позволяет еще больше упростить построение пути до объекта файловой системы. Например, путь до каталога №4 будет выглядеть следующим образом:

Корневой каталог / Каталог №1 / Каталог №4

Древовидная система каталогов совместно с простым именованием фалов представляет достаточно эффективную структуру. В дереве легко реализуются быстрые алгоритмы сортировки и поиска. Различные ветви дерева каталогов могут быть на различных физических носителях.

1.2 Существующие операционные системы

Наиболее подходящими и близкими к теме работы являются операционные системы, предназначенные для использования в смарт-картах. В данной главе будут рассмотрены наиболее популярные операционные системы для смат-карт:

– CardOS (C) Siemens AG [31];

– операционная система смарт-карты проекта УЭК (универсальной электронной карты) [32];

– Java Card OpenPlatform (JCOP) [33];

– MULTOS [34];

– ОС Магистра [35].

К сожалению, основная часть производителей вышеуказанных операционных систем крайне скудно описывают особенности архитектуры своих продуктов. Это связанно с высокой конкуренцией среди средств защиты. Все дальнейшие данные предоставляются на основании информации представленной в инструктивных материалах, которые доступны на официальных сайтах производителей.

Ввиду предназначения, большинство операционных систем для смарт-карт реализуют функции международного стандарта ISO/IEC 7816. Стандарт состоит из 14 основных частей:

1. описывает физические параметры карт (Physical characteristics);
2. описывает расположение и назначение контактов (Cards with contacts — Dimensions and location of the contacts);
3. описывает электрические параметры интерфейса и некоторые принципы установления связи для карт с асинхронным интерфейсом (Cards with contacts — Electrical interface and transmission protocols);
4. описывает протокол обмена и механизм действия команд (Organization, security and commands for interchange);
5. специфицирует процедуру регистрации в регулирующих органах идентификатора приложения (Registration of application providers).
6. специфицирует номера тэгов и формат записи для популярных типов данных: имен, дат, фотографий, биометрики и т. п. (Interindustry data elements for interchange);
7. специфицирует специализированный язык запросов;
8. специфицирует формат команд для доступа к криптографическим процедурам и менеджменту криптоключей. (Commands for security operations);
9. специфицирует формат команд для доступа к файловой системе карты. (Commands for card management);

10. специфицирует назначение контактов и принципы установления связи для карт с синхронным интерфейсом. (Electronic signals and answer to reset for synchronous cards);

11. специфицирует методы биометрической аутентификации;

12. специфицируетназначениеконтактовипринципыустановлениясвязидля картсинтерфейсом USB (Cards with contacts — USB electrical interface and operating procedures);

13. специфицирует команды управления приложениями;

14. специфицирует криптографические функции (Cryptographic information application).

Реализация стандарта ISO/IEC 7816 позволит использовать операционную систему в самых популярных платежных системах. Имеется и локализация данного стандарта - ГОСТ Р ИСО/МЭК 7816.[36] -[40]

1.2.1 CardOS SiemensAG

Операционная система CardOS SiemensAG [31] разработана одноименной компанией Siemens и в настоящее время широко применяется (к примеру, в средствах аутентификации компании Aladdin). Текущая версия операционной системы - 4.2. Данная операционная система поддерживает большое количество специализированных контроллеров.

Отличительной особенностью операционной системы является ее тесная взаимосвязь с аппаратной платформой (в части реализации механизмов обеспечения безопасности)**.** Командный интерфейс CardOS полностью поддерживает стандарт ISO 7816-4. Файловая система операционной системы защищается аппаратно-реализованными механизмами:

– аппаратно-защищенные области памяти;

– динамическое управление памятью (оптимизация EEPROM);

– защита памяти от дефектов и ошибок.

В операционной системе реализовано большое количество криптографических алгоритмов – RSA, SHA-1, Triple-DES (CBC), DES (ECB,

CBC), MAC, Retail-MAC (большинство реализовано в контроллере на аппаратном уровне). Аппаратная реализация механизмов обеспечения безопасности, которая так широко применяется в CardOS, имеет ряд положительных особенностей:

– относительно высокая скорость выполнения вычислительно сложных операций (например, криптографических операций);

– защита от отладки приложений.

1.2.2 Операционная система смарт-карты проекта УЭК

УЭК (универсальная электронная карта) представляет собой универсальный российский электронный носитель, который используется для идентификации пользователя. Согласно [41] универсальная электронная карта представляет собой материальный носитель, содержащий зафиксированную на нем в визуальной (графической) и электронной (машиносчитываемой) формах информацию о пользователе картой и обеспечивающий доступ к информации о пользователе картой, используемой для удостоверения прав пользователя картой на получение государственных и муниципальных услуг, а также иных услуг, в том числе для совершения в случаях, предусмотренных законодательством Российской Федерации, юридически значимых действий в электронной форме.

Специально для данного проекта была разработана аппаратная платформа - чип в формате смарт-карты с поддержкой российских криптографических алгоритмов.

Технические характеристики аппаратной платформы:

– защищенный 8-разрядный микроконтроллер MIK51 (частота до 30МГц);

– Объем EEPROM - 72K;

– Криптографические сопроцессоры блочных алгоритмов (ГОСТ 28147-89, DES, 3DES, AES);

– Криптографический сопроцессор модулярной арифметики для вычисления и проверки ЭЦП по алгоритмам ГОСТ Р34.10-2001, ECDSA, RSA;

– Поддержка протокола Mifare (протокол беспроводных меток).

Универсальная электронная карта также имеет и специально разработанную для данного проекта операционную систему для мобильных систем. Ядро операционной системы УЭК поддерживает различные аппаратные платформы, такие как MIK51 или 32-битные системы. Архитектура ОС изображена на рисунке 7.

В качестве среды выполнения прикладных программ используется известная платформа java (адаптированная для использования в системах с ограниченным количеством системных ресурсов).

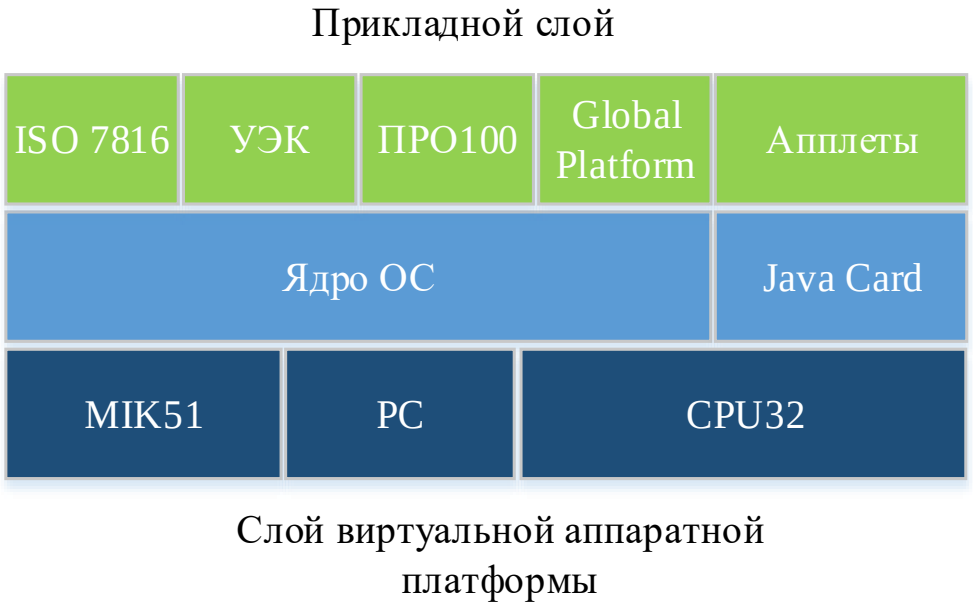


Рисунок 7. Архитектура операционной системы УЭК

Программы на данной платформе представляют собой специальные выполняемые сценарии (апплеты). Данное решение имеет как положительные, так и отрицательные стороны. К положительным можно отнести:

– популярный язык программирования,

– простой механизм апплетов,

– возможность портирования (адаптации) апплетов с других операционных систем.

Для обеспечения положительных особенностей платформы java придется пожертвовать производительностью. Механизм апплетов очень удобен для программиста, но для выполнения сценария необходимо выполнять преобразование в машинный код (компиляцию) либо каждый раз до выполнения приложения, либо в режиме реального времени. И первый**,** и второй вариант приведет к неизбежному увеличению времени выполнения приложений. Конечно, для не очень сложных приложений для смарт-карт, где многие функции реализованы аппаратно, скорость выполнения не является самым важным параметром операционной системы. Однако не самая эффективна в плане использования ресурсов платформа java отнимает вычислительные мощности, которые могут быть использованы для реализации большего количества функций конечного продукта при одинаковой аппаратной платформе.

2.2.3 Java Card Open Platform (JCOP)

Одна из наиболее популярных операционных систем которая применяется в смарт-картах. Данная операционная система для смарт-карт является очень популярной. В ее состав входят большое количество полезных подсистем. Очень большое внимание разработчики JCOP уделили вопросу отказоустойчивости. Так как смарт-карта существует в очень неблагоприятных условиях (неожиданные отключения питания, статическое электричество), необходимо**,** чтобы операционная система имела механизмы резервирования всех жизненно важных данных. JCOP представляет собой виртуальную среду выполнения (виртуальную машину) java. О положительных сторонах JAVA уже упоминалось в предыдущем разделе. Прикладные программы также представляют собой сценарии на языке java (апплеты). Применение виртуальной среды разработки позволяет производить резервирование данных. По информации из руководства администратора операционной системы [42]: «большинство информации**,** которая используется в операционной системе, зарезервирована в энергонезависимой памяти». Данный факт позволяет при неожиданном отключении (к примеру, в случае, когда пользователь намеренно отключил смарт-карту от платежного

терминала) избежать значительного повреждения операционной системы и потери пользовательских данных.

2.2.4 MULTOS AG

Особенностью файловой системы является занимательная реализация файловой системы. А точнее наличие в файловой системе файловых объектов различных типов, которые применяются в зависимости от поставленных целей. На рисунке 8 изображена структура различных типов файлов операционной системы MULTOS AG (оригинальное изображение[34]).

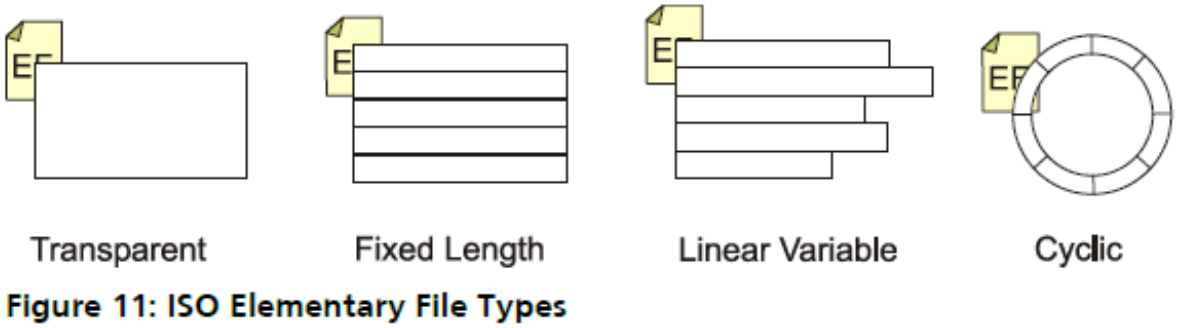


Figure 11: ISO Elementary File Types

Рисунок 8. Типы файловых объектов операционной системы MULTOS AG

В операционной системе представлены следующие типы файлов.

«Сырой файл» представляет собой файловый объект в виде непрерывного блока данных определенного размера. Размер файла может быть увеличен. Доступ к сохраненным данным осуществляется с применением функций бинарного (побайтного) доступа к файловой системе.

«Файл фиксированного размера» - это файл, поделенный на определенное количество блоков данных одинакового размера. Размер блока данных обычно обуславливается стандартным блоком записи на устройство хранения (например, карты памяти с файловой системой FAT32 поддерживают блоки размером 512 байт).

«Файл линейно переменного размера» состоит из блоков различного размера. Данный вид файлов может быть полезен при записи переменных различного размера (например, переменных различных типов - float, int, byte и

т.д.). Минусом данного типа файлов является нерациональность хранения данных (файл содержит достаточно большое количество служебной информации).

«Циклический файл» является наиболее интересным. Для организации журналов фиксированного размера целесообразно использовать файлы данного типа. Данный тип фалов представляет собой блок данных фиксированного размера. Запись в данный файл ведется с периодической перезаписью более старых блоков.

Применение каждого из вышеуказанных типов файловых объектов в конкретных случаях может быть достаточно эффективным. Крайне перспективным может считаться «Циклический файл». Так как без применения различного типа журналов нельзя представить ни одной операционной системы, возможность организовать эффективный журнал, просто создав файл на диске**,** является крайне полезной.

Для доступа к файлам смарт-карты (чтения, обновления) используется специализированная системная служба (Data Unit в терминологии операционной системы MULTOS AG). Данное решение является очень удачным. Организация доступа к файлам, расположенным на различных носителях, является ресурсоемким процессом. Более подробно об особенностях организации доступа к файловой системе в мобильных операционных системах будет рассказано далее.

Для защиты хранимых в смарт-карте данных (кода программ, флэш-памяти, энергонезависимой памяти) в MULTOS AG предусмотрены специализированные меры защиты данных. На рисунке 9 (оригинальное изображение [34]) изображена логическая схема прикладного программного продукта в контексте операционной системы MULTOS AG.

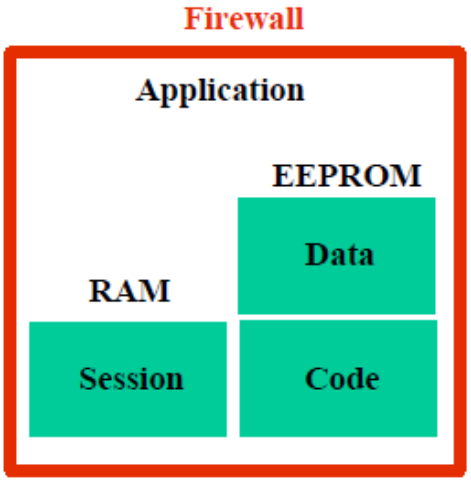


Рисунок 9. Логическая схема прикладного приложения в операционной системе MULTOS AG

По заверению производителя ОС каждое прикладное приложение имеет свое собственное исполняемое пространство. К пространству приложения относится:

– код программы расположен в статическом защищенном пространстве (пространство кода не может быть прочтено или записано);

– данные приложения располагаются в защищенном пространстве (данные могут быть прочитаны и записаны, но не выполнены);

– временные оперативные данные (данные сессии) располагаются в оперативной памяти.

Пространство выполнения каждого приложения защищено от других процессов защитным «экраном», который производит фильтрацию запросов на доступ к пространству процесса. Анализ поступающих запросов необходим в связи с тем, что у операционной системы нет возможности полной изоляции прикладного процесса от внешнего мира. Процессу необходим обмен с портами ввода-вывода и общими ресурсами. На рисунке 10 (оригинальное изображение [34]) изображена архитектура операционной системы MULTOS AG.

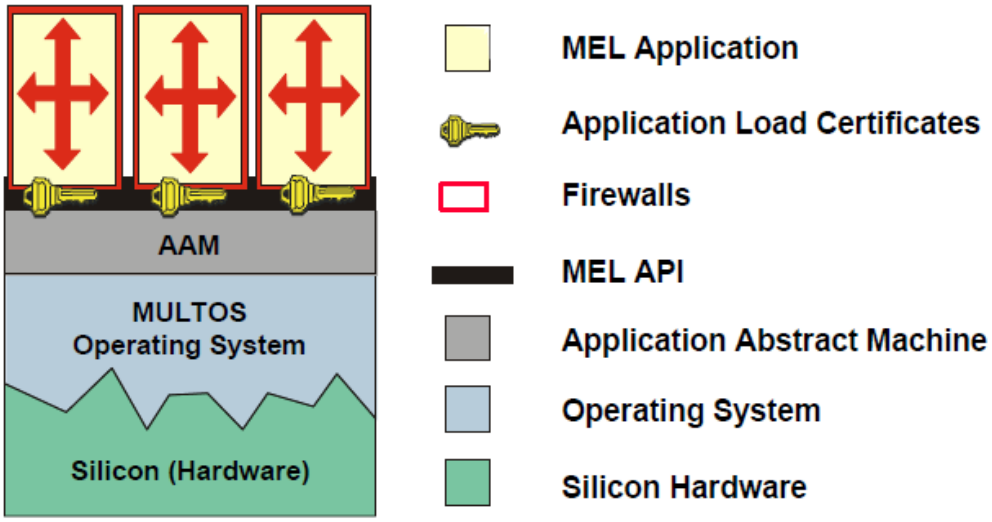


Рисунок 10. Архитектура операционной системе MULTOS AG

Операционная система имеет несколько функциональных уровней (в порядке возрастания уровня):

– физическая платформа (функции на этом уровне представляют собой низкоуровневые команды процессора);

– операционная система (ОС обеспечивает универсальные коммуникации, управление памятью, виртуальную машину);

– модуль абстракции приложения – модуль реализует интерфейс доступа к ресурсам операционной системы (API функции);

– модуль языка MULTOS AG(реализует специализированный язык для написания апплетов, поддерживает язык C и Java);

– Сертификат загрузки приложения (необходим для загрузки подписанного приложения);

– Специализированный защитный «экран» (в терминологии производителя – firewall);

– Прикладные приложения.

Для защиты операционной системы от выполнения потенциально опасного программного кода в MULTOS AG применяются специализированный защитный «экран», виртуальная среда выполнения прикладных программ и система проверки подписи производителя программного продукта до запуска приложения.

2.2.5 ОС Магистра AG

Операционная система Магистра AG разработана ООО "СмартПарк". Текущая версия операционной системы - 1.3. Согласно информации из руководства программиста [35]операционная система Магистра AG имеет следующие отличительные особенности:

- поддержка Российской криптографии;
- поддержка контактного интерфейса ISO 7816 (протокол T0);
- поддержка файловой системы на основе группы стандартов ISO 7816;
- поддержка механизма расширений.

ОС реализована на микроконтроллере ST23YL18 производства компании STMicroelectronics. Приложение смарт-карты разрабатывается как совокупность структурных элементов (файлов, директорий) для хранения разнообразных данных, доступ к которым (запись, чтение, использование) определяется самим разработчиком [35].

Как уже упоминалось ранее, операционные системы для мобильных систем функционируют в неблагоприятной среде, и существует большая вероятность возникновения нештатных ситуаций (например, вследствие неожиданного отключения электропитания). Резервирование должно являться неотъемлемой частью мобильной системы. Однако не существует ни одного стопроцентно гарантированного механизма резервирования. В случае возникновения ошибок в механизмах резервирования необходимо определить факт появления повреждений операционной системы. Для этих целей применяются механизмы самоконтроля и контроля целостности. Кроме того, контроль целостности критичных компонентов операционной системы позволяет обнаружить результаты злоумышленных действий нарушителя.

Отличительной особенностью данной операционной системы является наличие подсистемы самоконтроля. В ОС карты предусмотрены ряд автоматических (не отключаемых) и дополнительных (запускаемых вручную)

средств контроля исправности карты и целостности данных. К автоматическим средствам контроля относятся:

– самотестирование карты при старте;

– аппаратный механизм коррекции однократных ошибок в EEPROM и детектирования многократных ошибок;

– программно-вычисляемая контрольная сумма заголовков файлов, содержимого служебных файлов и пользовательских данных;

– механизм буферизации записи и поддержки транзакций. [35]

При невозможности восстановить целостность аппаратно-программной среды исполнения, карта операционная в зависимости от настроек может производить переход в специализированный режим ожидания или производить принудительный сброс.

1.3 Описание нарушителя

Под безопасностью информации (согласно [43]) понимают состояние защищенности информации [данных], при котором обеспечены ее [их] конфиденциальность, доступность и целостность.

Целью любого нарушителя безопасности является воздействие на состояние защищаемой информации (нарушение конфиденциальности, доступности и целостности информации). Перед выработкой необходимых мер защиты крайне необходимо иметь сведения о возможностях и умениях потенциального нарушителя.

Для построения как можно более стойкой системы защиты необходимо принять, что вероятный нарушитель информационной безопасности обладает следующими качествами:

– нарушитель является специалистом во всех требуемых областях знаний (информационных технологиях, методах защиты информации, и т.д.);

– нарушитель имеет знания о применяемой системе защиты информации;

– нарушитель имеет достаточно высокие материальные возможности;

– нарушитель имеет доступ к новейшим достижениям современной техники.

Наиболее простой классификацией нарушителей информационной безопасности является деление злоумышленников по положению относительно защищаемой информации:

- внутренний нарушитель,
- внешний нарушитель.

Не секрет, что порой действия внутреннего (легитимного или доверенного) пользователя могут нанести защищаемой информации больший урон, нежели внешний нарушитель. Ввиду данного факта следует понимать, что большинство подсистем системы защиты потенциально могут стать подвергнуты атаке со стороны нарушителя.

В качестве нарушителя может выступать не только отдельный человек, но организованная группа лиц. В данную группу могут входить внутренние нарушители (доверенные пользователи)

1.4 Угрозы безопасности операционной системы

Определение уязвимых мест операционной системы является важной задачей процесса оценки ее уровня защищенности. Возможны различные подходы к определению критериев защищенности операционной системы ([44]-[46]). В данной работе будут определены актуальные угрозы для мобильной операционной системы и обозначены методы защиты, нейтрализующие данные угрозы.

Угрозы безопасности в общем случае можно классифицировать по самым разным признакам. Согласно [47] выделяют следующие основные направления.

По природе возникновения выделяют искусственные и естественные угрозы.

По степени преднамеренности выделяют случайные и преднамеренные угрозы.

По степени воздействия выделяют пассивные и активные угрозы. К пассивным угрозам можно отнести анализ журналов операционной системы. К активным угрозам можно отнести реализацию вирусной атаки.

Можно привести и другие примеры классификации угроз [48], но наиболее популярной является классификация **по методу нарушения состояния безопасности информации** (см. определение в предыдущем разделе):

– угрозы нарушения конфиденциальности информации;

– угрозы нарушения целостности информации;

– угрозы нарушения доступности информации.

По принципу воздействия на операционную систему выделяют [49]:

– использование известных (легальных) каналов получения информации (например, несанкционированное чтение файла пользователем вследствие неверной настройки прав доступа);

– использование скрытых каналов получения информации (например, недокументированные возможности программного обеспечения);

– создание дополнительных собственных каналов получения информации (например, посредством внедрения программных закладок).

По типу используемой злоумышленником уязвимости защиты выделяют:

– неадекватная настройка политики безопасности ОС;

– ошибки и недокументированные возможности программного обеспечения;

– внедренная программная закладка.

Далее в работе будут рассмотрены угрозы безопасности, которые характерны для исследуемого объекта (операционной системы).

1.4.1. Неправомерный доступ к ресурсам

Операционная система может оперировать большим количеством различных ресурсов (процессорное время, оперативная память, объекты файловой системы). Любой ресурс имеющий ценность для нарушителя может стать его целью. Правилами разграничения доступа в операционной системе может быть

обеспечено разграничение доступа пользователей к ресурсам системы. В классической операционной системе предусмотрено ограничение доступа пользователей к системным файлам, которые необходимы для работы операционной системы. Для организации контроля доступа к защищаемым объектам прикладных программных продуктов в операционных системах реализуют различные механизмы разграничения доступа. Нарушение установленных ограничений с целью получения доступа к защищаемым ресурсам является одной из наиболее актуальных угрозой безопасности операционной системы.

К атакам, которые позволяют осуществить неправомерный доступ к ресурсам операционной системы, можно отнести:

– атаки на подсистему аутентификации (подбор пароля, подбор по словарям, перехват паролей специальными программными средствами – кейлогерами);

– атака превышения полномочий (путем использования ошибок программного обеспечения, применения вредоносных программ и др.).

1.4.2. Анализ (отладка) подсистем операционной системы

Анализ работы операционной системы может дать злоумышленнику данные для дальнейших атак на операционную систему. Операционная система имеет в своем составе большое количество потенциально подверженных анализу объектов. К таким объектам могут относиться:

– конфигурационные данные подсистем ОС;

– журналы работы ОС и прикладного программного обеспечения;

– программный код программ и функциональных частей ОС;

– оперативно и постоянно хранимые данные.

К атакам анализа ОС можно отнести атаки следующего типа:

– сканирование файловой системы (оперативной памяти);

– сборка мусора (анализ данных, подлежащих уничтожению);

– анализ программного обеспечения.

Отладка приложения представляет собой процесс анализа программного продукта специализированными средствами с целью устранения ошибок.[6]

Невозможно представить процесс разработки программного продукта (любой сложности), в котором отсутствует этап отладки приложения. Исправление ошибок и доработка программных продуктов является частью жизненного цикла любого приложения. Существует большое количество программных и программно-аппаратных средств, предназначенных для отладки приложений. Но и как многие другие полезные инструменты, средства отладки имеют двойное назначение. Злоумышленник может использовать средства отладки (анализа) приложений для различных целей:

– для получения эксклюзивных алгоритмов, реализованных в исследуемом приложении;

– для анализа системы защиты приложения с целью поиска уязвимых мест;

– для внедрения в алгоритм приложения вредоносного кода.

В соответствии с алгоритмом проведения анализа программного продукта выделяют следующие средства:

– дизассемблеры - реализуют статические методы отладки программных продуктов;

– отладчики - реализуют методы активного анализа приложений (отладка в процессе выполнения программы);

– эмуляторы - комплексные средства отладки (анализа), которые реализуют виртуальное окружение приложения.

Средства анализа приложений также могут строиться и по комбинированной схеме.

Статический метод анализа приложений основан на изучении низкоуровневых (машинных) команд приложения. Статический анализатор кода исследует собранное (скомпилированное) приложение, обнаруживает типичные блоки кода (циклы, условия, и т.д.), находит логические связи между блоками

кода. На выходе злоумышленник получает простой для понимания код программы или блок-схему ее работы.

Современный программист редко использует для написания сложных и функциональных приложений машинный язык программирования. Для таких задач используются специализированные автоматизированные среды разработки программных продуктов. Среды разработки предоставляют разработчикам все больше возможностей, таких как мастера построения типизированных проектов и многофункциональные программные библиотеки (например, библиотека Microsoft.Netframework). Программные библиотеки содержат наиболее типичные блоки кода. Современные программные библиотеки крайне обширны и охватывают различные области применения.

Широкое использование программных библиотек приносит пользу не только программистам, но и злоумышленникам. Современные языки программирования представляют собой "иерархические" структуры алгоритмов, такие как функции, макросы или библиотеки. По этой причине в программной библиотеке большое количество "постоянных" блоков кода (блоков с предопределенным кодом). Ввиду данного факта, зная тип библиотеки, на которой написан программный продукт, злоумышленник может с небольшими усилиями восстановить исходный код исследуемой программы.

Активные методы анализа приложений обычно используются для анализа особо сложных блоков кода. Активные методы более информативны, чем статические. Принцип действия отладчиков представляет собой пошаговое выполнение исследуемого приложения. Во время выполнения отладчик исследует и внешнее окружение программы (регистры процессора, обращения к файловой системе, значения переменных в оперативной памяти).

Еще одной крайне функциональной особенностью активного метода отладки является возможность изменения внешнего окружения исследуемого программного продукта. Например, отладчик может поменять значение некоторой переменной в оперативной памяти при входе в определенную функцию

и посмотреть на реакцию программы. Данный факт дает злоумышленнику очень большие возможности и может позволить обойти даже самые сложные системы защиты. По такому принципу взламываются многие коммерческие программы. По этой причине производители многое производители программных продуктов вносят в систему защиты своих продуктов элементы неопределенности (например, активацию программы через сеть интернет). Даже если злоумышленнику известен алгоритм аутентификации, он может не иметь сведений о преобразованиях на стороне сервера.

Эмуляторы позволяют злоумышленнику создать вокруг исследуемого программного продукта собственное "виртуальное" окружение. Тем самым нарушитель может "заставить" приложения выполнять некоторые действия. Стратегий применения эмуляторов огромное количество. Существуют как самые простые эмуляторы (одной определенной подсистемы), так и эмуляторы, полностью воспроизводящие среду функционирования программы.

Все вышерассмотренные методы анализа программ могут быть использованы в связке друг с другом. Угроза анализа подсистем операционной системы является одной из наиболее важных.

1.4.3.Методы атак на криптографическую подсистему

Анализ надежности уязвимостей криптографических алгоритмов является не тривиальной задачей. В [50] выделяют четыре основных формальных метода анализа криптографических алгоритмов:

– моделирование и проверка работы протокола;

– создание экспертных систем для отладки различных нестандартных режимов работы алгоритмов;

– моделирование требований к семейству криптопротоколов;

– разработка формальных моделей, основанных на алгебраических свойствах криптографических систем.

Вышеуказанные алгоритмы позволяют проанализировать работу криптографических алгоритмов.

Целью атаки на криптографическую подсистему операционной системы могут являться:

– секретные данные;

– нахождение секретного ключа.

В первом случае злоумышленник получит доступ к конкретному секретному сообщению. Получив ключевую информацию, злоумышленник получит возможность перехватывать все зашифрованные сообщения. Следующие подсистемы могут быть интересны для злоумышленника:

– защищенные объекты виртуальной файловой системы;

– криптографические контейнеры пользователей, которые могут содержать ключевую информацию и другие важные данные;

– данные передаваемые через подсистему передачи данных.

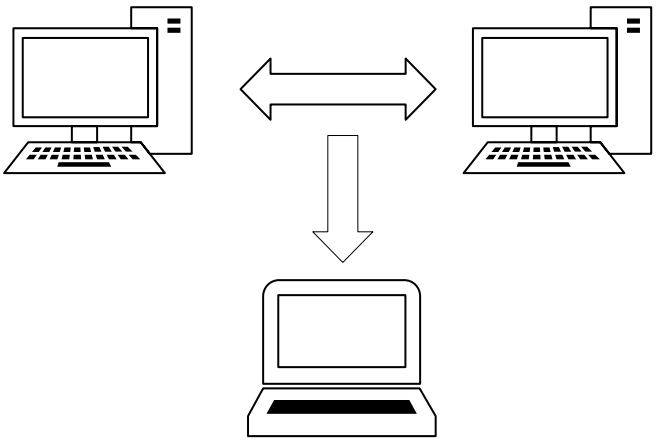


Рисунок 11. Пассивный перехват зашифрованных данных

В [51] предлагается классифицировать атаки на алгоритмы шифрования в зависимости от набора информации, которая имеется у злоумышленника. К первой категории относятся атаки, при которых криптоаналитик имеет возможность пассивного прослушивания канала связи (рисунок 11). В данном случае злоумышленник имеет доступ шифртексту. Данный вид атаки называется **атакой с известным шифртекстом**. Во второй категории атак целью

криптоаналитика является ключ шифрования. В данном случае криптоаналитик обладает шифратором с введенным ключом и может проводить криптографические преобразования с помощью шифратора (рисунок 12).

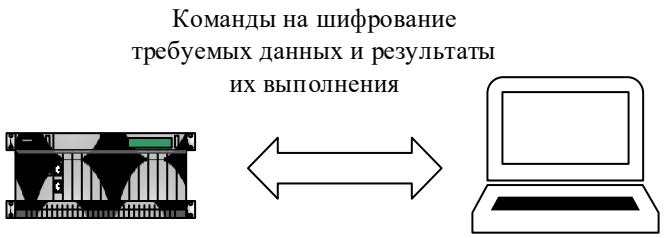


Рисунок 12. Активное воздействие на шифратор

В зависимости от данных, которые криптоаналитик использует для анализа, выделяют следующие виды атак:

- атака с известным открытым ключом;
- атака с выбранным открытым текстом;
- адаптивная атака с выбором открытого текста;
- атака с выбором шифртекста;
- адаптивная атака с выбором шифртекста.

Так как мобильная операционная система выполняется на завершенных аппаратных платформах, актуальность имеют атаки на криптосистему**,** основанные на ошибках вычислительной платформы. Так в [52] отмечается, что алгоритм симметричного шифрования ГОСТ 28147-89 является устаревшим алгоритмом шифрования из-за его подверженности атакам на основе аппаратных ошибок.

1.4.4. Нарушение работоспособности операционной системы

Операционная система представляет собой сложный программный продукт. Неработоспособность операционной системы может быть вызвана целым рядом различных причин. Начиная от программных ошибок, допущенных на этапе разработки операционной системы и заканчивая преднамеренными действиями

злоумышленника (например, удалением или модификацией системных файлов ОС).

К характерным угрозам, нацеленным на нарушение работоспособности ОС, относятся:

– программные закладки и вирусы;

– программные ошибки (например, ошибки чрезмерного использования ресурсов операционной системы);

– аппаратные ошибки (исключительные особенности работы конкретной аппаратной платформы).

1.4.5. Недекларированные возможности программного обеспечения

Согласно [53] под недекларированными возможностями понимается функциональные возможности ПО, не описанные или не соответствующие описанным в документации, при использовании которых возможно нарушение конфиденциальности, доступности или целостности обрабатываемой информации.

В операционной системе проблема НДВ в прикладном (клиентском) программном обеспечении имеет очень большое значение. Так как данное программное обеспечение разрабатывается сторонними производителями, имеется большая вероятность появления непредвиденных ошибок при выполнении. Также имеется возможность, что НДВ может быть организовано производителем преднамеренно с целью завладения пользовательскими данными.

1.5 Постановка задачи

В данной главе была рассмотрена структура объекта исследования. Операционная система представляет собой программный продукт, предназначенный для эффективного управления вычислительными ресурсами. К задачам операционной системы также относится организация псевдо параллельного выполнения нескольких прикладных программных продуктов (многозадачности). Ключевым понятием в понимании принципа организации многозадачности является прерывания. Прерывая работу прикладной задачи,

операционная система производит сохранение окружения (регистров и оперативной памяти) и дальнейшую передачу управления другой прикладной программе. Задачами управления переключениями процессов, контроля использования оперативной памяти и организации протоколов доступа к аппаратным ресурсам занимается специализированный модуль операционной системы – ядро операционной системы. От архитектуры ядра (центральной части операционной системы) зависит сфера применения ОС.

Большое количество элементов, которые взаимодействуют друг с другом, в процессе работы операционной системы (части ядра, системные службы, драйвера аппаратных устройств и прикладные процессы), делают операционную систему идеальным объектом для исследования методов защиты информации. Основная задача данной главы выделить наиболее актуальные для мобильной операционной системы угрозы информационной безопасности.

Были рассмотрены особенности реализации мобильных операционных систем на примере операционных систем для смарт-карт. Каждая из представленных операционных систем имеет в себе особенности реализации. Основной функцией операционной системы для смарт-карт является обеспечение функционирования протоколов работы с картой, которые описаны в стандарте ISO/IEC 7816. Данные протоколы взаимодействуют с криптографическими контейнерами, которые могут храниться как во внутренней (защищенной) памяти микроконтроллера, так и на внешнем хранилище данных. Для реализации защищенного хранилища криптографических контейнеров производители смарт-карт применяют два основных подхода:

– реализация смарт-карты на основе специализированного микроконтроллера,

– использование универсального микроконтроллера совместно с дополнительными мерами защиты.

Оба вышеуказанных подхода имеют свои положительные и отрицательные качества. В первом случае производитель применяет специализированные

микроконтроллеры, которые имеют в своей архитектуре защищенное хранилище данных (пример – смарт-карты проекта УЭК). Чаще всего данные хранилища представляет собой специализированный блок, который включает в себя собственно хранилище (энергонезависимая память или флэш-память) и контролер управления памятью. В задачи контроллера памяти входит управление хранением данных и контроль доступа к данным. Встроенный блок защищенного хранения данных имею сравнительно высокую производительность. Отрицательной стороной использования специализированных контроллеров является привязка операционной системы к одной конкретной аппаратной платформе, что делает функции устройства зависимыми от аппаратной платформы.

Второй подход использует, например, электронные идентификаторы компании «Актив» – Рутокены.[54] В аппаратной архитектуре данных устройств применяются универсальные микроконтроллеры архитектуры ARM. Для реализации защищенного хранилища на базе универсального контроллера необходимо применять дополнительные методы защиты данных. Это обусловлено тем, что универсальные контроллеры нацелены на большой круг устройств и имеют в своем составе средства диагностики и отладки. Нередко универсальные микроконтроллеры имею в своем составе и блоки защиты данных (например, MPU – блок защиты данных в контроллерах фирмы STMElectronics). Но, к сожалению, такие средства защиты обычно имеют ограниченный функционал. Обычно они имеют только механизмы дискреционного разграничения доступа в ограниченном исполнении. Для данных в универсальных контроллерах применяют защищенные виртуальные файловые системы. Данные в памяти хранятся в зашифрованном виде.

Механизмы доступа к памяти также имеют важное значение. Так, например, MULTOS AG. имеет в своем составе подсистему контроля использования ресурсов системы прикладными процессами. Многое производители концертирует ресурсы системы в файловой системе. Файловая система является наиболее удобной структурированной иерархической структурой, которая может

включать в себя объекты различного типа (файлы, каталоги, ссылки на функции и т.д.). Реализация файловой системы, представленной в операционной системе MULTOS AG, имеет особенный интерес. В структуре файловой системы включены специализированный файловые объекты, такие как «цикличный файл».

В ходе работы над моделью угроз мобильной операционной системы были выделены актуальные угрозы мобильной операционной системе. Угрозы сгруппированы в классы. В соответствии с данными классами угроз будет производиться дальнейшая разработка структуры системы защиты мобильной операционной системы.

Операционные системы успешно применяются в средствах защиты информации. Многие классы средств защиты на аппаратном уровне имеют весьма схожее строение. Персональные идентификаторы (USB-токены, например, Rutoken компании Актив [54]) и ключи защиты программных продуктов (например, HASP фирмы Aladdin[55]) имеют в своей аппаратной платформе универсальный микроконтроллер с интерфейсом USB. Чуть более сложную структуру имеют продукты ОКБ САПР. В основе СЗИ от НСД Аккорд-АМДЗ [56] котроллер общего назначения и матрица программируемой логики. Применение универсальной операционной системы в системах защиты информации позволит унифицировать подходы к обеспечению безопасности при разработке таких систем. Это значительно снизит затраты при разработке средств защиты информации и соответственно сделает такие продукты более конкурентно способным.

Цель данной работы состоит в сокращении сроков и уменьшении затрат по разработке мобильных информационных технологий. Для достижения этой цели решается общая научно-техническая задача расширения функциональности средств защиты информации, обеспечения переносимости программных средств защиты информации на различные типы мобильных устройств (на различные типы технических платформ) и встраивание механизмов защиты от атак с

принуждением. Для решения общей задачи необходимо выполнить следующие задачи:

1. выполнение анализа функциональных возможностей и особенностей реализации существующих мобильных операционных систем и на его основе разработать модель угроз информационной безопасности объекта исследования, архитектуру и программный код универсальной защищенной операционной системы для мобильных систем;

2. разработка метода аутентификации пользователей, стойкого к принуждающим атакам;

3. разработка метода защитного преобразования передаваемой по открытым каналам информации, стойкого к атакам с принуждением пользователя раскрыть ключ защитного преобразования;

4. разработка метода защиты программного обеспечения от дизассемблирования;

5. разработка метода защиты хранимой информации, стойкого к атакам с принуждением пользователя раскрыть ключ защитного преобразования.

Глава 2. Разработка методов аутентификации и разграничения доступа

Данная глава посвящена разработке методов аутентификации. В главе будут представлены разработанные алгоритмы усиленной аутентификации пользователей, описаны механизмы разграничения доступа в разработанной операционной системе. Также будут описаны принципы построения систем разграничения доступа.

2.1. Архитектура систем разграничения доступа

Подсистема разграничения доступа к ресурсам является одной из основных частей любой многопользовательской системы. Даже при самом пренебрежительном отношении к безопасности данных в системе, разработчик операционной системы ограничивает доступ пользователей к ресурсам ядра операционной системы. Неприкосновенность ресурсов ядра играет немалую роль в стабильности работы операционной системы. Из вышесказанного следует, что даже при отсутствии в системе каких-либо средств обеспечения безопасности данных, операционная система все равно имеет в своем составе хотя бы минимальную подсистему разграничения доступа пользователей к данным.

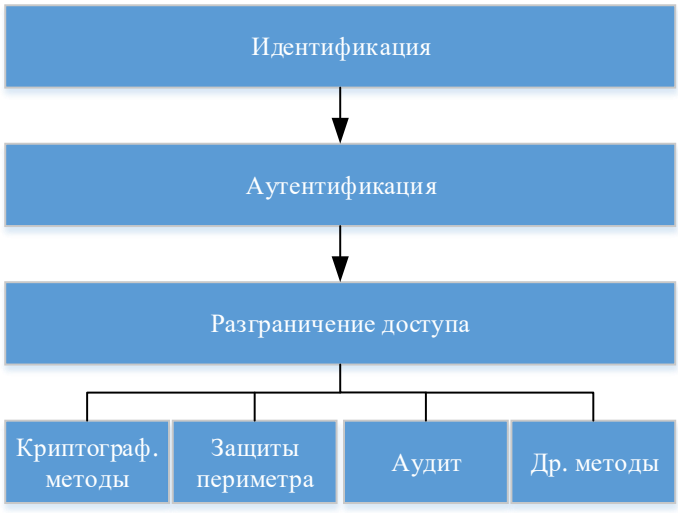


Рисунок 13. Структура системы защиты от угроз нарушения конфиденциальности

Для эффективной работы любой системы защиты информации необходимо четко определить субъекты и объекты доступа. **Объект доступа** — единица

информационного ресурса АС, доступ к которой регламентируется правилами разграничения доступа. [7]. **Субъект доступа** ─ лицо или процесс, действие которого регламентируется правилами разграничения доступа [7]. Объектами доступа операционной системы являются: файловые объекты, процессы, объекты памяти, периферия. Все объекты разграничения доступа в операционной системе имеют свой уникальный идентификатор (для объекта оперативной памяти - уникальный адрес, для периферии - уникальный дескриптор, и т.д.). Чтобы сформировать четкие правила разграничения доступа к объектам, необходимо однозначно определить субъекты доступа. На рисунке 13 изображена структура системы защиты от угроз нарушения конфиденциальности. Процесс определения субъекта доступа начинается с идентификации и проверки подлинности. В ходе процесса идентификации будет определен субъект доступа. В операционной системе будет сознан уникальный идентификатор пользователя.

2.1.1. Понятие пользователя системы

Известных моделей разграничения доступа к ресурсам существует несколько: мандатная, ролевая и т.д..

Пользователь системы - абстрактный, агрегированный субъект доступа в системе. По данной схеме субъектом доступа к системе является пользователь системы. Остальные субъекты (например, процессы) являются производными от родительского субъекта пользователь (процесс запускается от имени пользователя). Несомненно, в рамках определенной технологии могут быть использованы и другие субъекты доступа, но в рамках операционной системы все равно будет предусмотрен некоторый механизм наследования.

Абстрактный объект "пользователь" ассоциируется с "пользователем - человеком". По данной причине вышеуказанная модель наследования прав доступа легка для понимания человеком. В операционной системе могут быть предусмотрены различные группы пользователей (системные, удаленные пользователи и т.д.).

Неотъемлемой частью любой такой модели являются процессы идентификации и аутентификации пользователей в системе.

Идентификация - присвоение субъектам и объектам доступа идентификатора и (или) сравнение предъявляемого идентификатора с перечнем присвоенных идентификаторов. [57]

Аутентификация - проверка принадлежности субъекту доступа предъявленного им идентификатора; подтверждение подлинности. [57]

Так как под пользователем операционной системы в большинстве случаев понимают человека, принято выделять три основных метода аутентификации пользователя в операционной системе [58]:

1. по владению некой информации (знание секрета);
2. по владению уникальным предметом (смарт-карты, карты памяти и т.д.);
3. аутентификация по тому, что является неотъемлемой частью пользователя (биометрическая аутентификация).

Аутентификация по знанию секрета (пароля). Пароли применяется людьми уже очень много времени. По данной причине "парольный" метод аутентификации является наиболее "привычным", простым и понятным для человека. По виду применяемых паролей методы парольной аутентификации делятся на:

– методы с применением многоразовых паролей;
– методы с применением одноразовых паролей.

Аутентификацию по многоразовому паролю легко реализовать как в программных, так и в аппаратных устройствах. В операционной системе пароли пользователей могут храниться: в отрытом виде (в виде значения пароля), в виде значения односторонней функции (хеш-функции), в зашифрованном виде.

К сожалению, применение многоразовых паролей имеет и ряд очевидных недостатков:

1. зависимость безопасности системы от сложности выбранного пароля;

2. зависимость безопасности системы от сохранения пароля в секрете (пользователь не редко записывает пароли на бумажные носители);

3. возможность подбора пароля (в том числе с применением словарей паролей) злоумышленником;

4. возможность перехвата (выведывания) пароля злоумышленником.

Экзотические способы аутентификации (графический пароль и др.) описаны в работах [59],[60].

Одноразовые пароли. Аутентификация в удаленных системах является более сложной задачей [61]. У злоумышленника появляется возможность перехвата передаваемых аутентификационных данных. Применение одноразовых паролей решает ряд проблем парольных систем аутентификации. Суть одноразовых паролей заключается в том, что каждый пароль действителен только при выполнении одной отдельной операции доступа к ресурсу операционной системы. Перехват одноразового пароля не даст злоумышленнику особых возможностей (особенно при применении многофакторной аутентификации).

Для генерации одноразового пароля в протоколах аутентификации на одноразовых паролях используется обычно два параметра:

1. секретное значение (например, аппаратный идентификатор генератора);

2. случайное число (полученное от сервера) или значение текущего времени.

Широко известным примером протокола аутентификации на одноразовых паролях является протокол S/KEY (стандарт интернета RFC1760). Данный протокол основан на использовании алгоритмов хеш-функций MD4, MD5.

Одноразовые пароли решают множество проблем классических (многоразовых) парольных систем, но, к сожалению, не все. Протоколы аутентификации становятся еще более уязвимыми при значительном удалении пользователя от операционной системы. Для обеспечения безопасного удаленного доступа операционная система должна обеспечить конфиденциальность

передаваемого по сети пароля. Существует достаточное количество протоколов "удаленной" аутентификации. Данные протоколы аутентификации подразделяют на:

1. протоколы простой аутентификации, в которых пароль тем или иным образом передается по сети проверяющей стороне;
2. протоколы строгой аутентификации, в которых пользователь не передает пароль по сети, а тем или иным образом доказывает знание секретного пароля.

"Удаленная" аутентификация в операционной системе подразумевает некий "диалог" между клиентом и операционной системой. Классической является клиент-серверная организация протокола аутентификации. Чаще всего протокол "удаленной" аутентификации является частью протокола удаленного взаимодействия с операционной системы.

Простые методы "удаленной" аутентификации основываются на передаче каким-либо методом пароля (или хеш-значения функции от пароля) от клиента к проверяющей стороне (серверу). Для сокрытия секретного пароля может применяться шифрование (симметричное или ассиметричное) или хеш-функции.

Отличительной особенностью **протоколов строгой аутентификации является** то, что пользователь доказывает свою аутентичность перед сервером (выполняя ряд действий) без фактической передачи пароля (в любом виде) через канал связи.

Примеры протоколов строгой аутентификации описаны в работах [62],[63]. Популярным является протокол, описанный в стандарте ISO/IEC 9798-2. Данный стандарт предусматривает три варианта аутентификации:

1. односторонняя аутентификация с использованием меток времени;
2. односторонняя аутентификация с использованием случайных чисел;
3. взаимная (двусторонняя) аутентификация.

Рассмотрим простой пример односторонней аутентификации с использованием случайных чисел (сторона $\boldsymbol{A}$ аутентифицируется перед стороной $\boldsymbol{B}$):

$$\boldsymbol{A} \leftarrow \boldsymbol{B}: \boldsymbol{r_b};$$

$$\boldsymbol{A} \rightarrow \boldsymbol{B}: \boldsymbol{E_k}(\boldsymbol{r_b}, \boldsymbol{B}).$$

Обозначения:

$$\boldsymbol{A}, \boldsymbol{B} - \text{идентификаторы сторон;}$$

$$\boldsymbol{r_b} - \text{случайное число стороны } \boldsymbol{B};$$

$$\boldsymbol{E_k} - \text{функция шифрования на общем секретном ключе } \boldsymbol{k}.$$

Сторона $\boldsymbol{B}$ передает стороне $\boldsymbol{A}$ случайное значение (значение не является секретным). Сторона $\boldsymbol{A}$ шифрует данное значение совместно с идентификатором стороны $\boldsymbol{B}$ на общем симметричном ключе $\boldsymbol{k}$. Далее сторона $\boldsymbol{A}$ передает зашифрованное значение стороне $\boldsymbol{B}$. Сторона $\boldsymbol{B}$ расшифровывает полученное значение и проверяет свой идентификатор и случайное число. Если значения совпали, сторона $\boldsymbol{B}$ считает сторону $\boldsymbol{A}$ аутентифицированной. Заметим, что передачи секретного значения (пароля) в процессе аутентификации не было.

Аутентификация с использованием уникального предмета. В данном случае пользователь доказывает свою подлинность операционной системе, предъявляя уникальный предмет. Разработано большое количество предметов-аутентификаторов (RFID-метки, идентификаторы TouchMemory, USB токены). Безопасность метода аутентификации очень сильно зависит от конкретной реализации уникального предмета. Например, производитель персональных идентификаторов iButton (Touch Memory) компания Dallas Semiconductor гарантирует, что каждый произведенный ею идентификатор имеет уникальный идентификационный номер. Применение данных идентификаторов в схеме аутентификации по уникальному предмету могло бы быть достаточно безопасным. Однако так как iButton использует для связи известный протокол 1-Wire, китайские производители реализовали аналогичные по функционалу идентификаторы, но с перезаписываемым идентификатором. Аутентификация по

уникальному предмету чаще всего используется совместно с другими методами аутентификации (например, аутентификация уникальный предмет и ПИН-код).

Так как для организации **биометрической системы аутентификации** необходимы специализированные считыватели, данный вид аутентификации не будет рассматриваться в работе.

Идентификации и аутентификации объектов доступа в операционной системе применяется в различных сферах (аутентификация в системе, аутентификация в различных сетевых протоколах, таких как FTP). Для этого подсистема аутентификации в операционной системе должна иметь унифицированный (универсальный) характер. Пример унифицированной подсистемы аутентификации можно рассмотреть на основе операционной системы Linux. В данной ОС предусмотрена система подключаемых модулей аутентификации (с англ. PAM – pluggable authentication module).

PAM представляет собой универсальную библиотеку ─ PAM API, которая используется для подключения модулей аутентификации. Разработчики модуля аутентификации могут, используя библиотеку PAM API, производить разработку своих собственных алгоритмов аутентификации. Подключаемый модуль аутентификации представляет собой скомпилированную библиотеку, описывает функции аутентификации. В операционной системе обычно по умолчанию реализованы модули, в которых описывают протоколы TACACS, RADIUS и др. Решение об аутентификации принимается в подключаемом модуле. Модули подключаются и регистрируются в операционной системе путем записи пути к библиотеке в конфигурационном файле (файл /ets/pam.conf).

Клиентские приложения также взаимодействуют с библиотекой PAMAPI. При необходимости аутентификации прикладное приложение обращается к библиотеке PAMAPI. Библиотека читает настройки с конфигурационного файла. Далее производится процедура аутентификации, которая описана в подключаемом модуле.

В работе [64] проведен анализ системы аутентификации на подключаемых модулях. Система PAM имеет удобный, унифицированный и прозрачный механизм аутентификации, который позволяет легко интегрировать новые модули аутентификации.

Применение многофакторной унификации позволяет многократно повысить эффективность системы аутентификации ([65], [66], [67]). Многофакторная аутентификация в концепции PAM также реализуема.

Анализ эффективности применения в мобильной ОС.

Метод защиты: Аутентификация на многоразовых паролях.

Уровень эффективности: средний.

Достоинства: простота применения, понятность конечному пользователю.

Недостатки: защищенность системы аутентификации значительно зависит от сложности применяемого пароля, вероятна атака подбора паролей, многоразовые пароли мало применимы для удаленных систем (отсутствует защита при передаче пароля по сети).

Метод защиты: Усиленная аутентификация

Уровень эффективности: высокий.

Достоинства: высокая эффективность метода (в связи с применение криптографических алгоритмов), адаптирована для применения для удаленных пользователей, отсутствует фактическая передача секрета по сети.

Недостатки: сложность реализации (как клиентская, так и серверная сторона должна быть оснащена средствами вычисления).

Метод защиты: Протоколы одноразовых паролей

Уровень эффективности: средний.

Достоинства: простота реализации алгоритма.

Недостатки: отсутствует защита от атак типа «человек посередине».

2.1.2. Модели разграничения доступа

Если пользователями операционной системы становится более чем один субъект, возникает необходимость в прозрачном, универсальном механизме разграничения доступа к ресурсам системы.

Модель систем дискреционного разграничения доступа. Данная модель характеризуется разграничением доступа между поименованными субъектами и объектами. Каждый субъект, который имеет определенное право, может делегировать его другому субъекту. Доступ субъекта к объекту определяется конечным множеством возможных операций. Для каждой тройки объект-субъект-операция в системе строго определено правило. Данная система может быть реализована в виде матрицы доступа. Столбцы и строки данной матрицы описывают соответственно множество субъектов и объектов. В ячейках данной матрицы указывается тип разрешенной операции. По меньшей мере**,** имеется два подхода к построению дискреционного управления доступом:

– каждый объект системы имеет привязанного к нему субъекта, называемого владельцем. Именно владелец устанавливает права доступа к объекту;

– система имеет одного выделенного субъекта – администратора, который имеет право устанавливать права владения для всех остальных субъектов системы.

Мандатное управление доступом. Для реализации этого принципа каждому субъекту и объекту должны сопоставляться специальные метки (мандаты). Каждый объект имеет уровень конфиденциальности, а каждый субъект имеет соответственно уровень доступа. В мандатной системе разграничения доступа должно быть реализовано два основных правила:

– «нет записи вниз» – субъекту запрещено писать в объект, если уровень доступа субъекта выше уровня конфиденциальности объекта;

– «нет чтения вверх» – субъекту запрещено читать объект, если уровень доступа субъекта ниже уровня конфиденциальности объекта.

Ролевое разграничение. Основной идеей управления доступом на основе ролей является идея о связывании разрешений доступа с ролями, назначаемыми каждому пользователю. Данная модель разграничения доступа**,** очень понятна человеку. В других сферах деятельности человека часто встречаются роли (работник-начальник, студент-преподаватель и т.д.). Каждая роль имеет свой строго определенный набор прав доступа к объектам. Как правило, данный подход применяется в системах защиты СУБД, а отдельные элементы реализуются в сетевых операционных системах.

2.2. Контроль и управление доступом

В разработанной операционной системе контроль и управление доступом представлен двумя подсистемами: подсистемой аутентификации и системой контроля доступа к файловой системе. Применение концепции «все есть файл» позволило значительно унифицировать подходы разграничения доступа к файловым объектам, оперативной памяти и аппаратным устройствам.

2.2.1. Подсистема аутентификации

Подсистема аутентификации в разработанной операционной системе играет одну из важных ролей. В подсистеме аутентификации ОС реализован целый ряд функций:

1. аутентификация локальных пользователей;
2. аутентификация удаленных пользователей.

Подсистема аутентификации представляет собой набор библиотек, которые используются для проведения процедур аутентификации. Прототипом структуры подсистемы аутентификации послужила применяемая в операционных системах семейства Linux архитектура PAM (Pluggable Authentication Modules - подключаемые модули аутентификации).

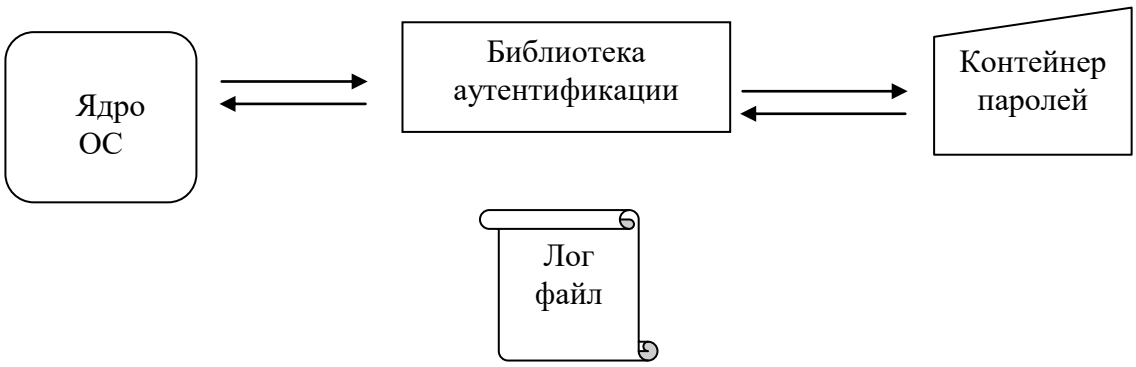


Рисунок 14. Подсистема аутентификации пользователей

Подсистема аутентификации локальных пользователей (рисунок 14) обеспечивает классическую аутентификацию пользователей с использование многоразовых паролей. Имеется возможность настройки параметров парольной политики:

- длины пароля;
- алфавита пароля;
- условий блокировки пользователей.

Любые операций аутентификации и изменения аутентификационных данных пользователей пишутся в системный журнал.

Для защиты пользователей от принуждающей атаки в операционной системе был разработан алгоритм аутентификации пользователей с защитой от принуждающей атаки.

2.2.2 Алгоритм аутентификации пользователей

Далее предложен алгоритм защиты пользователей от принуждающей атаки в процессе аутентификации в ОС. Для защиты пользователя от принуждающей атаки в ОС генерируется дополнительный набор аутентификационных данных. При этом для защиты пользователей было выбрано достаточно простое и эффективное решение.

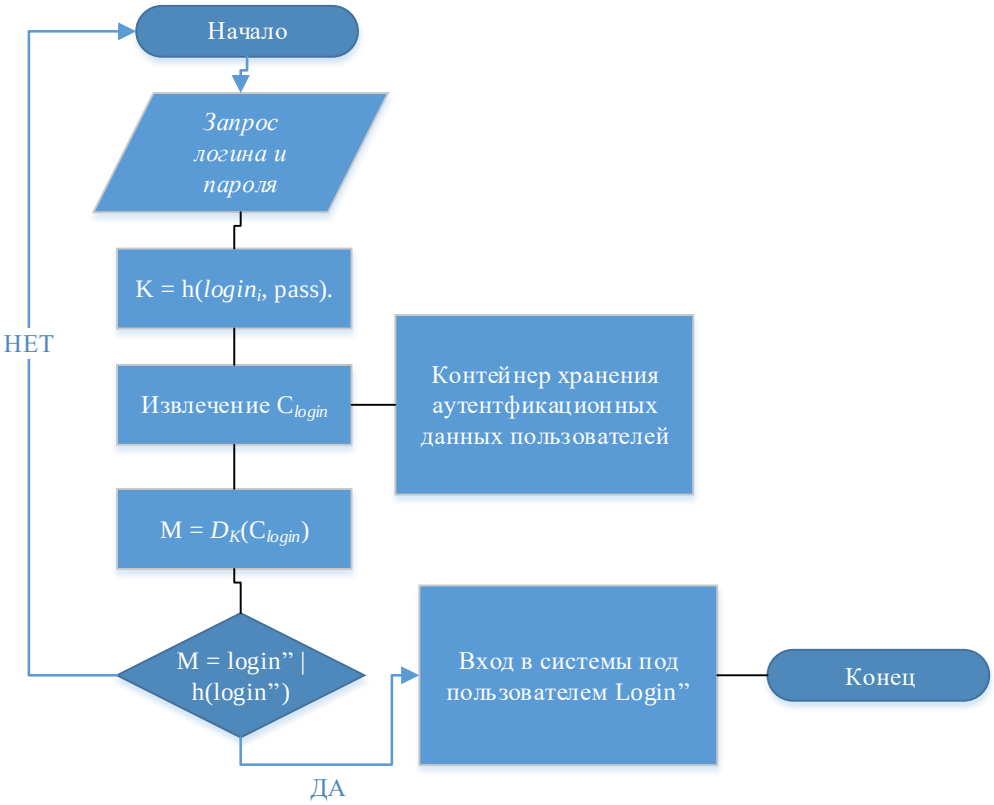


Рисунок 15. Блок-схема процесса аутентификации локальных пользователей

Для этого выбирается имя два имени пользователя с похожим названием (к примеру, user1 и userl). Один пользователь имеет необходимые права в системе. Второй пользователь имеет минимальные права. В случае ввода резервного пароля (при принуждающей атаке) будет выполнен вход ограниченным пользователем. Алгоритм аутентификации пользователей с защитой от принуждающей атаки изображен на блок-схеме (рисунок 15).

Процесс аутентификации заключается в следующем:

1. запросить у пользователя логин и пароль;
2. вычислить ключ шифрования как хеш-значение логина и пароля;
3. извлечь из хранилища шифртекст, в соответствии с логином пользователя;
4. расшифровать сообщение ключом, полученным на шаге 2;

5. если сообщение состоит из login" (возможно отличного от введенного) и хеш-значения полученного login" (для проверки результата), то вход в систему под пользователем login", иначе переход к шагу 1.

При вводе резервного пароля будет расшифровано значение резервного логина и пользователь зайдет под пользователем с усеченными правами.

Анализ эффективности применения в мобильной ОС

Метод защиты: метод парольной аутентификации с защитой от принуждающей атаки.

Отличительные особенности: защита от принуждающей атаки, возможность применения в любых пользовательских ОС (ориентированность на архитектуру PAM).

2.2.3. Способ аутентификации на одноразовых паролях

В разработанной операционной системе предусмотрена возможность удаленного доступа. В дополнение к стандартным многоразовым паролям в ОС предусмотрена аутентификация с использованием одноразовых паролей.

Наиболее популярный протокол аутентификации на одноразовых паролях S/KEY описан в стандарте интернета RFC1760. Данный протокол основан на хеш-функции MD4. В разработанной операционной системе применен протокол одноразовых паролей, дополнительно усиленный отрицаемым шифрованием, предложенным в [68], что позволяет противодействовать вынуждающей атаке. Алгоритм работает по клиент-серверной схеме. Рассмотрим данный алгоритм более подробно.

Условные обозначения:

N -количество генерируемых паролей;

файл криптографических значений - файл, который содержит значения результатов шифрования, используется при проверке одноразового пароля (хранится на сервере);

R -случайное значение;

C - шифртекст;

ID пользователя - уникальный идентификационный номер пользователя;

h( ) - хеш функция ГОСТ Р 34.11-94 [69];

P -парольная фраза;

S -случайное значение, вектор инициализации.

На первом шаге пользователь выбирает парольную фразу **P**. Вектор инициализации **C** - случайное число - позволяет использовать пользователем одну и ту же парольную фразу для нескольких серверов. Далее производится операции "исключающий ИЛИ" значений P и S. Вычисление временных паролей производятся по следующим формулам:

$$K_N = h(PxorS), \text{для } N = 1,$$

$$K_N = h(K_{N-1}), \text{ для } N > 1\,.$$

Так генерируются 2 списка из N паролей, при этом для каждого списка вектор инициализации также должен быть уникальный.

Для генерации файла криптографических значений для N одноразовых паролей выбираются 2N случайных значений. Дополнительно выбирается случайное значение K_0 первого криптографического значения C.

$$M_N = h(ID)|R_{1N}, \bar{M}_N = h(ID)|R_{2N};$$

$$K = R_{1(N-1)};$$

$$C_N = \left[{M_N}^{K_{11}} Kp_2(p_2^{-1} mod p_1) + {\bar{M}_N}^{K_{21}} Kp_1(p_1^{-1} mod p_2)\right] mod p_1 p_2.$$

Для получения последнего криптографического значения C используется следующие формулы:

$$M_1 = h(ID)|R_{12}, \bar{M}_1 = h(ID)|R_{22};$$

$$K = K_0;$$

$$C_1 = \left[{M_1}^{K_{1N}} Kp_2(p_2^{-1} mod p_1) + {\bar{M}_1}^{K_{2N}} Kp_1(p_1^{-1} mod p_2)\right] mod p_1 p_2.$$

На стороне клиента хранятся:

1. список из 2N паролей;
2. парольная фраза и 2 вектора инициализации (если необходима генерация паролей).

Использование одноразовых паролей начинается с последнего пароля (K_{1N}).

На стороне сервера хранятся:

1. файл криптографических значений (содержит N криптографических значений);
2. номер текущего пароля пользователя;
3. значения простых чисел p_1, p_2;
4. временный общий ключ K.

Для дешифрования проверочного сообщения необходимо вычислить следующие значения.

$$M = (CK^{-1})^{K_{1N}^{-1}} \, mod \, p_1$$

$$\bar{M} = (CK^{-1})^{K_{2N}^{-1}} \, mod \, p_2$$

В первоначальном состоянии временный ключ $K = K_0$, а номер текущего пароля - 1. Для проверки одноразового пароля K_{1N} сервер производит дешифрование криптографического значения C_1 по следующей формуле.

$$M_1 = (C_1K^{-1})^{K_{1N}^{-1}} \, mod \, p_1$$

Стоит заметить, так как сервер не имеет сведений, какой серии пользователь предоставил пароль (K_{1N}или K_{2N}), серверу необходимо попробовать дешифровать значение, используя и простое число p_2.

$$M_1 \neq (C_1K^{-1})^{K_{1N}^{-1}} \, mod \, p_2$$

В данном случае сервер не сможет дешифровать проверочное сообщение. Для проверки правильности проверочного сообщения сервер сравнивает первые 256 бит проверочного сообщения с вычисленным значением $h(ID)$. В случае удачной аутентификации в качестве временного общего ключа принимается расшифрованное значение R_{12} (таким образом, производится завязка последовательности паролей).

В случае если пользователь передаст серверу пароль K_{2N} (при осуществлении вынуждающей атаки), проверка подлинности пройдет успешно. Однако в качестве временного общего ключа будет принято неверное

значение (а точнее значение R_{22}) и следующий пароль ($K_{1(N-1)}$или $K_{2(N-1)}$) будет признан неправильным.

Анализ эффективности применения в мобильной ОС

Метод защиты: Способ применения отрицаемого шифрования в системах аутентификации на одноразовых паролях.

Отличительные особенности: защита пользователя от принуждающей атаки, возможность реализации в виде генераторов паролей и парольных карточек (заранее сгенерированных).

2.2.4 Сервис контроля доступа к файловой системе

В работах [70] - [73] представлены системы контролируемого разграничения доступа к файлам. Иерархическая структура файловой системы является идеальным объектом применения моделей разграничения доступа.

Для обеспечения контроля доступа к ресурсам в операционной системе предусмотрено несколько различных механизмов. Одним из таких механизмов является специализированный сервис доступа к файловой системе. Данная системная служба предоставляет пользователям возможности доступа к файловой системе. Конечно, используя функции динамической библиотеки, прикладное программное обеспечение также может получить высокоскоростной доступ к файлам. Однако для осуществления процесса записи в файл процессу необходимо зарезервировать достаточно большой объем оперативной памяти (около 2 килобайт). Для процессов, которые не могут позволить себе такую расточительность (в условиях мобильной системы это большинство процессов) и существует служба доступа к файловой системе. Данная служба имеет схожее строение с сервисом криптопровайдера, который был рассмотрен ранее. На рисунке 16 изображена структурная схема службы доступа к файловой системе.

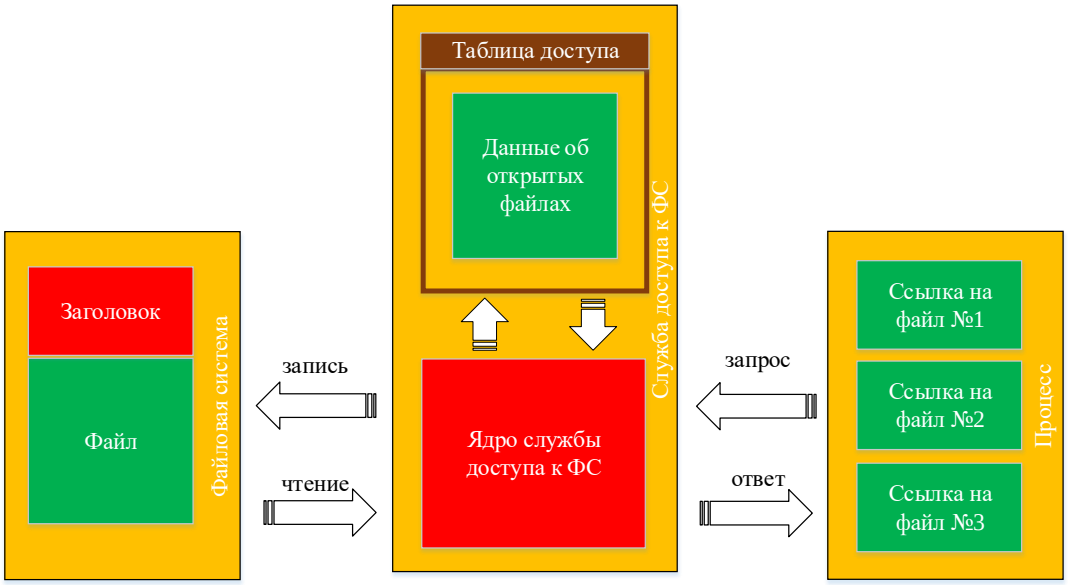


Рисунок 16. Структурная схема службы доступа к файловой системе

Служба доступа к файловой системе состоит из двух основных блоков:

– ядра службы, которое выполняет запросы прикладных процессов (в ядро входит очередь запросов на выполнение операции);

– таблицы доступа.

Ядро службы обеспечивает очередное выполнение заданий процессов на операции доступа к файловой системе. Очередность выполнения операций может быть основана на приоритетах процессов в операционной системе. В качестве критерия очередности, в зависимости от настроек операционной системы, могут быть использованы и другие регулярные переменные.

Таблица доступа представляет собой матрицу, в которой отражается взаимосвязь номера процесса операционной, объекта доступа (ссылка на файл), метода доступа (запись, чтение, и т.д.) и время последнего доступа. Данная таблица заполняется ядром сервиса по мере обращения прикладных процессов к новым файловым объектам (функция fopen).

Сервис контроля доступа к ФС обеспечивает:

– оптимизацию занятой оперативной памяти (в процессе анализа "брошенных файлов");

– анализ последствий аварийного завершения работы операционной системы (в зависимости от настройки ОС таблица доступа может писаться в энергонезависимую память и при аварии таблица останется невредимой);

– обеспечения безопасности (аудит процессов доступа к файлам, работа сенсоров системы обнаружения вторжения).

К дополнительным функциям сервиса доступа к файловой системе относится:

– обеспечение доступа к криптографическим функциям для выполнения операций над защищенными файлами;

– унификация доступа к файловой системе.

Анализ эффективности применения в мобильной ОС

Метод защиты: сервис контроля доступа к файловой системе.

Отличительные особенности: эффективное распределение ресурсов, прозрачные правила разграничения доступа (дискреционное разграничение), выполняет роль СЗИ от НСД мобильной ОС (так как в системе применяется принцип "все есть файл"), унификация методов доступа к ФС.

2.3 Выводы ко второй главе

В данной главе была рассмотрена подсистема разграничения доступа, которая является наиболее важной в составе операционной системы. Было рассмотрено такое понятие как пользователь системы (как особый вид субъекта доступа). Далее была описана универсальная модель разграничения доступа, которая включает в себя процессы идентификации, аутентификации и др.

Была проанализирована эффективность алгоритмов аутентификации, применяемых в операционных системах. С целью повышения эффективности функционирования подсистемы были разработаны: алгоритм аутентификации пользователей системы, способ аутентификации на одноразовых паролях, которые обеспечивают защиту от атак с принуждением.

В данной главе решены следующие задачи исследования:

– разработка метода аутентификации пользователей, стойкого к принуждающим атакам.

Глава 3. Разработка методов защиты хранимой и передаваемой информации

Данная глава посвящена разработке методов защиты хранимой и передаваемой информации, стойких к атакам с принуждением пользователя раскрыть ключ защитного преобразования. В задачи данной главы также входит выделение и анализ эффективности методов защитного преобразования информации, применяемых в операционных системах. Будут описаны разработанные методы защитного преобразования информации.

3.1. Методы защитного преобразования информации в ОС

В современных операционных система криптографические методы защиты информации используются повсеместно. Существует большое количество реализованных алгоритмов [74]. Для простого и унифицированного доступа к криптографическим функциям в операционной системе реализуется специализированный модуль. В данном модуле (модуль иногда называют криптопровайдером) реализуются различные криптографические функции. Остальные подсистемы операционной системы могут использовать криптопровайдер по своему усмотрению. В рамках криптопровайдера классически реализуются следующие криптографические алгоритмы.

Алгоритмы симметричного шифрования. Алгоритмы шифрования данной категории отличаются высокой производительностью. Это свойство делает симметричное шифрование крайне эффективным при обработке большого количества данных (например, при прозрачном шифровании на магнитном диске).

Симметричное шифрование имеет неоспоримый недостаток ─ один ключ для шифрования и расшифрования. На первый взгляд очевидным решением данной проблемы является применение двухключевых (ассиметричных) алгоритмов шифрования. Однако ассиметричное шифрование имеет сравнительно низкую производительность и при этом оно имеет достаточно высокую вычислительную сложность. Именно по этой причине ассиметричное шифрование крайне редко реализуется в критопровайдерах.

Для компенсации недостатков симметричного шифрования в криптопровайдерах чаще всего используют **алгоритмы обмена ключевой информацией**. Такие алгоритмы также иногда называют алгоритмами создания общего ключа обмена. К таким алгоритмам относится протокол Ди́ффи-Хе́ллмана (протокол описан в [75]). Данный протокол позволяет создать одинаковый ключ обмена у двух абонентов. Общий ключ обмена вычисляется из открытого ключа удаленного абонента и собственного закрытого ключа. После генерации общего ключа шифрования появляется возможность для обмена с использованием скоростных симметричных шифров. Вышеуказанная схема (применение протокола генерации общего ключа совместно с симметричным шифрованием) чаще всего применяется в современных средствах криптографической защиты информации. Примером удачного комбинирования ассиметричных и симметричных протоколов может служить протокол "безопасного переходника" SSL (secure socket slayer) [76]. Данный протокол применяется в web-браузерах, и комбинирует в себе комбинацию протоколов RSA и DES или RSA и тройной DES.

Хранение ключевой информации. Безопасное хранение ключей в операционной системе является важной задач подсистемы защиты ОС. В процессе эксплуатации криптографических протоколов иногда приходится производить процедуру смены (обновления ключевой информации). Необходимость смены ключей может быть вызвана различными причинами:

– истек срок действия ключевой информации;

– ключевая информация скомпрометирована.

Нередко для обеспечения непрерывности защищенного взаимодействия между участниками информационного обмена применяется резервирование ключевой информации. Для этого на этапе распределения ключей формируются несколько комплектов ключей. Стороны договариваются о номере (серии) применяемого ключа. В данной схеме важно защитить резервные ключ (например, зашифровав резервные ключи дополнительным ключом). Стоит

заметить, что получение резервной ключевой информации и серии активного ключа может быть целью злоумышленника.

Алгоритмы хешифрования. Алгоритмы хешифрования представляют собой алгоритм преобразования информации, которые позволяют обнаружить любые изменения в исходном файле. Требования к хеш-функции:

– невозможность фабрикации (высокую сложность подбора сообщения с заданным значением хеш);

– невозможность модификации (незначительные изменения в исходном файле должны приводить к значительному изменению значения хеш);

– однонаправленность (ложность вычисления исходного сообщения по заданному значению хеш);

– устойчивость к коллизиям (сложность нахождения пары сообщений с одним значением хеш);

– устойчивость к нахождению второго прообраза (сложность нахождения второго сообщения с тем же значением хеш по имеющейся паре сообщение - хеш).

Алгоритмы электронной подписи. Алгоритмы электронной подписи представляют собой криптографические алгоритмы, позволяющие обеспечить механизмы проверки подлинности и аутентичности сообщений. В основе алгоритмов электронной подписи лежит ассиметричное шифрование.

Алгоритмы генерации случайных чисел. Алгоритмы генерации случайных чисел с определенными параметрами является важной частью системы криптографической защиты информации. Малейшая предсказуемость в алгоритме генерации ключевой информации может стать предпосылкой к атаке на ключевую систему. В некоторых криптографических алгоритмах (например, в алгоритмах ассиметричного шифрования) необходимо сгенерировать случайное число большого размера, которое отвечает свойствам простоты. В [77] выделяются следующие основные подходы к генерации случайных чисел:

– программная генерация, предполагающая вычисление очередного псевдослучайного числа как функции текущего времени, последовательности символов, введенных пользователем, особенностей его клавиатурного почерка;

– программная генерация, основанная на моделировании качественного генератора псевдослучайных кодов с равномерным законом распределения;

– аппаратная генерация с использованием физических величин (шум радио эфира, физические характеристики полупроводников в нестандартных режимах).

Отрицаемое шифрование. Под понятием отрицаемого шифрования понимается способ криптографического преобразования, в котором зашифровываются совместно два или более различных сообщений на двух или более различных ключах, и обосновывается принципиальная реализуемость таких преобразований [68]. Для нахождения пар СООБЩЕНИЕ-КЛЮЧ для функции шифрования F, таких чтобы они имели идентичную криптограмму C, необходимо решить следующую систему уравнений.

$$\begin{cases} F(K_1, C) = M \\ F(K_2, C) = \bar{M} \end{cases},$$

где $K_1, K_2 -$ ключи шифрования; $M, \bar{M} -$ сообщения; $F -$функция шифрования; $C -$ криптограмма.

В концепции отрицаемого шифрования рассматривается цель обеспечения достаточно высокой стойкости к принуждающим атакам. В модели таких атак предполагается, что атакующий имеет некоторый ресурс воздействия на отправителя, получателя или хранителя криптограммы, принуждающий последнего представить ключ расшифрования криптограммы. Стойкость к атакам с принуждением обеспечивается тем, что, по крайней мере, одно из зашифрованных сообщений не является секретным и атакующему предоставляется ключ, по которому расшифрование криптограммы приводит к раскрытию этого сообщения. При этом процедура расшифрования выполняется таким образом, что у атакующего нет обоснованных доводов, которые он мог бы

привести в пользу утверждения, что с криптограммой связаны еще какие-то другие сообщения. [68]

Анализ эффективности применения в мобильной ОС.

Метод защиты: способ отрицаемого шифрования "подбор пары СООБЩЕНИЕ-КЛЮЧ".

Уровень эффективности: средний.

Достоинства: защита от принуждающей атаки, неотличимость от применяемого алгоритма шифрования.

Недостатки: крайне низкая производительность алгоритма.

Метод защиты: Способ хранения резервных серий ключей.

Уровень эффективности: средний.

Достоинства: отсутствует необходимость использования протоколов распределения ключей.

Недостатки: необходимость решать вопросы хранения ключевой информации.

3.2. Криптографическая подсистема

Для того чтобы в операционной системе имелась возможность повсеместно применять криптографические алгоритмы, в ОС было необходимо реализовать функциональную и гибкую криптографическую подсистему. Данная подсистема предоставляет пользователям (подпрограммам, службам и сервисам) целый комплекс криптографических алгоритмов.

Криптографическая подсистема ОС представляет комплекс из двух механизмов.

Подключаемая библиотека криптографических функций. Данный механизм применяется в низкоуровневых приложениях ОС (приложения уровня ядра). Выполнение криптографических функций производится из процесса, вызвавшего функцию, что определяет минимальные задержки получения ответа. Количество памяти, которое необходимо для выполнения функции, должно быть предварительно зарезервировано процессом "родителем". Так как мобильные

системы имеют ограниченное количество оперативной памяти, использовать библиотеку повсеместно не представляется возможным.

Для решения вышеуказанной проблемы в операционной системе предусмотрен специализированный сервис (криптопровайдер), который осуществляет криптографические преобразования по "заказу" других процессов. Такой механизм имеет ряд положительных особенностей:

– экономия ресурсов операционной системы (например, оперативной памяти);

– унификация алгоритма защиты ключевой информации;

– унификация алгоритма доступа к криптографическим операциям.

Так как операционной системе необходимо удовлетворить запросы на криптографические преобразования множества процессов, в криптопровайдере необходимо предусмотреть:

– механизм последовательного выполнения заданий (очередь заданий);

– механизм разграничения доступа (для защиты ключевой информации различных процессов);

– протокол взаимодействия криптопровайдера с процессом.

На рисунке 17 представлен процесс шифрования блока данных. Как уже упоминалось, криптопровайдер представляет собой самостоятельный процесс, который поочередно выполняет задания других процессов.

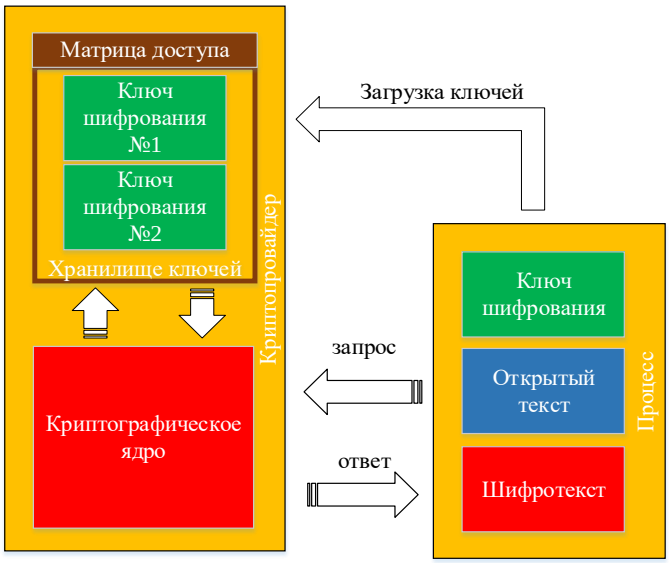


Рисунок 17. Процесс взаимодействия криптопровайдера с прикладными процессами

Для защиты ключевой информации пользователей криптопровайдер имеет в своем составе специализированное хранилище ключей. Хранилище ключей располагается в выделенной для криптопровайдера памяти (постоянной или оперативной). Если на аппаратной платформе предусмотрен механизм контроля памяти, то он обязательно должен быть использован для защиты вышеуказанных участков памяти. В микроконтроллерах архитектуры ARM фирмы STMictolectronics данная технология называется MPU (от англ. Memory Protect Unit). Также в хранилище ключей дополнительно реализован механизм дискреционного разграничения доступа (субъектом доступа является идентификатор процесса в системе).

Протокол взаимодействия процессов с криптопровайдером состоит нескольких групп операций:

– операции с ключевой информации (генерация ключа, запись ключа в файл, чтение ключа из файла);

– операции шифрования данных (шифрование блока данных, расшифрование блока данных, шифрование файла, расшифрование файла);

– операции хеширования (рассчитать хеш блока данных, рассчитать хеш файла);

– операции электронной подписи (выработать подпись файла, проверить подпись файла).

При необходимости зашифровать файл, процесс выполняет поочередно запросы на выполнение операции:

1. генерация ключа шифрования;
2. запись ключа шифрования в ключевой файл (для возможности дальнейшего расшифрования файла);
3. загрузка ключа шифрования из ключевого файла;
4. шифрование файла.

Также стоит заметить, что операции работы с ключами (запись ключа в файл, чтение ключа из файла) могут оперировать хранилищами различного типа. В зависимости от критичности хранимы на различных устройствах хранения ключей (смарт-карте, устройстве Touch memory).

В подсистеме криптографической защиты реализованы следующие криптографические алгоритмы.

Симметричное шифрование - ГОСТ 28147-89[78].

Алгоритм обмена ключами - протокол Диффи-Хеллмана (англ. Diffie-Hellman, DH).

Алгоритм электронной подписи - ГОСТ Р 31.10-2001, ГОСТ Р 31.10-94 [79][80].

Алгоритм хеширования ─ ГОСТ Р 34.11-2012.

Отрицаемое шифрование. В операционной системе реализован алгоритм отрицаемого шифрования, описанный в [81]. В данном случае в основе алгоритма ОШ лежит алгоритм симметричного шифрования ГОСТ 28147-89. Также применяется алгоритм отрицаемого шифрования, предложенный в работе [68]. Более подробно описание вышеуказанных алгоритмов будет приведено далее.

Анализ эффективности применения в мобильной ОС

Метод защиты: метод применения криптопровайдеров в низкопроизводительных системах.

Отличительные особенности: метод адаптирован для снижения потребления вычислительных ресурсов, наличие выполнения в режиме очереди, возможность многопользовательского применения.

3.2.1 Алгоритм алгебраического алгоритма псевдовероятностного защитного преобразования

В работах представлены практически применимые алгоритмы шифрования [68][81][12]. В операционной системе были применены несколько алгоритмов отрицаемого шифрования. Первый алгоритм ОШ описан в работе[68]. Рассмотрим данный алгоритм более подробно.

Условные обозначения:

***K*₁, *K*₂, *K*₃, *K*₄ –** подключи шифрования (K_1, K_2 для основного сообщения, K_3, K_4 для «фальшивого» сообщения);

p – простое число;

C - шифртекст;

M – сообщение.

Выполнение отрицаемого шифрования начинается с генерации секретного ключа в виде набора подключей K_1, K_2, K_3, K_4 и двух простых чисел p_1 и p_2. Шифрование блоков M ($p_1 > M$) и $\overline{M}$ ($p_2 > \overline{M}$) двух сообщений осуществляют путем вычисления значения C_1 по формуле $C_1 = M^{K_1} K_2 \bmod p_1$, вычисления значения C_2 по формуле $C_2 = \overline{M}^{K_3} K_4 \bmod p_2$ и формирования блока криптограммы C, которое является решением системы сравнений

$$\begin{cases} C \equiv C_1 \bmod p_1 \\ C \equiv C_2 \bmod p_2 \end{cases},$$

систему можно представить в виде

$$\begin{cases} C \equiv M^{K_1} K_2 \bmod p_1 \\ C \equiv \bar{M}^{K_3} K_4 \bmod p_2 \end{cases}.$$

В соответствии с китайской теоремой об остатках решение вычисляется по следующей формуле:

$$C = \left[M^{K_1} K_2 p_2 \left(p_2^{-1} \bmod p_1 \right) + \bar{M}^{K_3} K_4 p_1 \left(p_1^{-1} \bmod p_2 \right) \right] \bmod p_1 p_2 .$$

При вынуждающей атаке необходимо предоставить атакующему одно из сообщений. Пусть в качестве "ложного" сообщения выступает $\bar{M}$. Тогда атакующему предоставляется в качестве ключа шифрования тройка значений K_3, K_4, p_2. Расшифрование выполняется по формуле

$$\bar{M} = \left(C K_4^{-1} \right)^{K_3^{-1}} \bmod p_2 .$$

В последней формуле обратные значения для подключей K_3 и K_4 вычисляются по модулям $p_2 - 1$ и p_2, соответственно. При необходимости расшифрования "истинного" секретного сообщения M выполняется вычисления по той же формуле, но с использованием ключа, представляющего собой тройку значений (K_1, K_2, p_1):

$$M = \left(C K_2^{-1} \right)^{K_1^{-1}} \bmod p_1 .$$

Вышеуказанный протокол применяется в протоколе аутентификации по одноразовым паролям. Более подробно данный протокол будет описан в работе далее.

Анализ эффективности применения в мобильной ОС

Метод защиты: способ алгебраического алгоритма псевдовероятностного защитного преобразования.

Отличительные особенности: обладает свойствами неотличимости от вероятностного шифрования и единообразия процедуры расшифрования криптограммы по всему пространству ключей [68].

3.2.2 Алгоритм защитного преобразования с использованием труднообратимых операций

В работе [12] описан общий способ построения алгоритмов отрицаемого шифрования с использованием любых труднообратимых операций (в том числе блочных шифров). В самом простом случае алгоритм шифрования может быть реализован подбором таких двух пар СООБЩЕНИЕ-КЛЮЧ, чтобы при проведении одинаковых криптографических преобразований был получен одинаковый шифртекст. Данный алгоритм ОШ может быть реализован на базе блочного алгоритма шифрования с необходимой длиной блока (рисунок 18). Рассмотрим данный алгоритм более подробно.

Условные обозначения:

$\boldsymbol{K_1}, \boldsymbol{K_2}$ – ключи шифрования;

$\boldsymbol{p}$ – простое число;

$\boldsymbol{C}$ – шифртекст;

$\boldsymbol{E}$– алгоритм блочного шифрования (n – длина блока в битах);

$\boldsymbol{M}$ – сообщение.

Для нахождения пар СООБЩЕНИЕ-КЛЮЧ необходимо решить систему уравнений.

$$\begin{cases} E_{K1}(C)\ mod\ 2^n = M \\ E_{K2}(C)\ mod\ 2^n = \bar{M} \end{cases}$$

Предложенный алгоритм ОШ легко программно реализуем. Для шифрования двух сообщений М1 и М2 необходимо выполнить следующие действия:

1. сгенерировать ключ К1 и произвести шифрование сообщения М1;
2. сгенерировать ключ К2 и расшифровать шифртекст, который был получен на шаге 1;
3. сравнить полученное на шаге 2 с исходным сообщением М2;
4. если условие верно – закончить, иначе перейти к шагу 2.

Шифруемые сообщения должны иметь одинаковую длину (равную длине блока шифрования).

$$|M| = |\bar{M}| = n$$

Такой способ ОШ, конечно, не отличается особой эффективностью. Для шифрования необходимо выполнить 2^{2n} подборок.

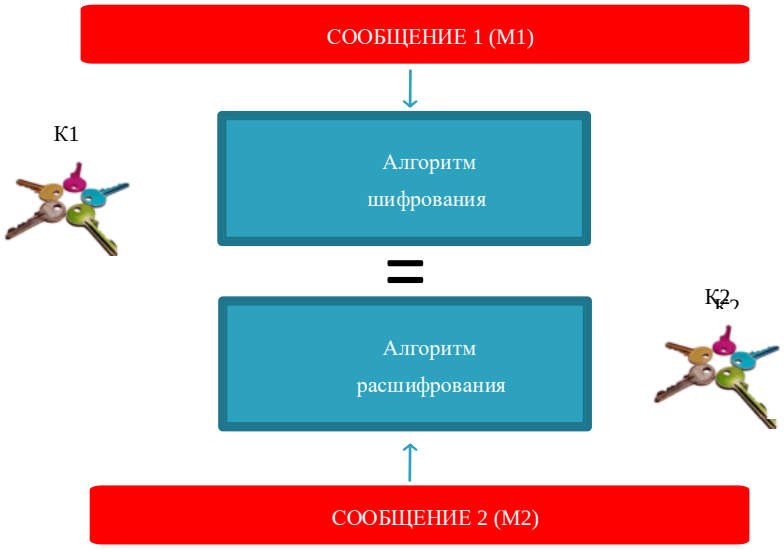


Рисунок 18. Алгоритм отрицаемого шифрования

Существуют и другие более эффективные алгоритмы отрицаемого шифрования.

3.2.3 Алгоритм защитного преобразования на базе блочного шифрования

В ходе исследования были разработаны эффективный алгоритм отрицаемого шифрования. В [81] был предложен алгоритм быстрого отрицаемого шифрования с использованием блочных алгоритмов шифрования. На рисунке 19 изображена блок-схема вышеизложенного алгоритма. Далее будет рассмотрен данный алгоритм.

Условные обозначения:

$\boldsymbol{K_1}$, $\boldsymbol{K_2}$ – ключи шифрования;

$\boldsymbol{C}$ – шифртекст;

$\boldsymbol{E}$– функция шифрования;

$\boldsymbol{D}$– функция расшифрования;

$\boldsymbol{\bar{M}}$– сообщение;

$\boldsymbol{M}$ – сообщение.

Сообщение, которое необходимо зашифровать, делится на части (например, посимвольно).

$$M = \{t_1, t_2, \dots, t_{1n}\}$$

$$\bar{M} = \{m_1, m_2, \dots, m_{1n}\}$$

К символу из СООБЩЕНИЯ 1 присоединяется СЛУЧАЙНОЕ ЗНАЧЕНИЕ 1, далее производится шифрование с использованием ключа К1 и расшифрование шифртекста с использованием ключа К2. Если получен нужный символ СООБЩЕНИЯ 2, то данная криптограмма сохраняется. Основная задача алгоритма подобрать такое СЛУЧАЙНОЕ ЗНАЧЕНИЕ 1, чтобы выполнилось это условие.

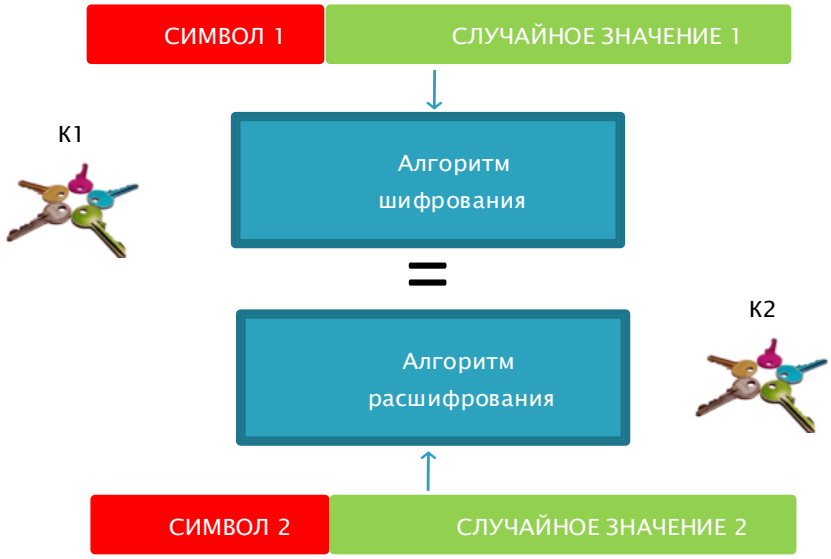


Рисунок 19. Алгоритм отрицаемого шифрования

Для вычисления искомого значения шифртекста для каждого символа сообщения необходимо решить систему уравнений.

$$\begin{cases} E_{K1}(t_i, r) = C_i \\ E_{K2}(m_i, \bar{r}) = C_i \end{cases}$$

Эффективность алгоритма зависит от значения выбранных параметров. Рассмотрим процедуру шифрования (рисунок 20) пары сообщений *T* и *M*:

1. установить значение счетчика $i = 1$;

2. установить значение счетчика $j = 1$;

3. сгенерировать случайное k-битовое число r_j;

4. вычислить значение $c_j = E_{K_M}(m_i, r_j)$;

5. вычислить значение $D_{K_T}(c_j) = (t_j, r'_j)$, где выходное n-битовое значение функции расшифрования интерпретируется как конкатенация u-битового значения t_j и k-битового значения r'_j;

6. сравнить значения t_j и t_i. Если $t_j = t_i$, то взять в качестве значения c_i значение c_j и перейти к шагу 7, в противном случае перейти к шагу 6;

7. если $j < N$, то прирастить значение счетчика $j \leftarrow j + 1$, вычислить $r_j \leftarrow r_{j-1} + 1$ и перейти к шагу 4, иначе вывести сообщение «Шифрование пары знаков t_i и m_i не выполнено»;

8. если $i < z$, то прирастить значение счетчика $i \leftarrow i + 1$ и перейти к шагу 2, иначе СТОП.

Ниже показана блок-схема вышеуказанной процедуры шифрования.

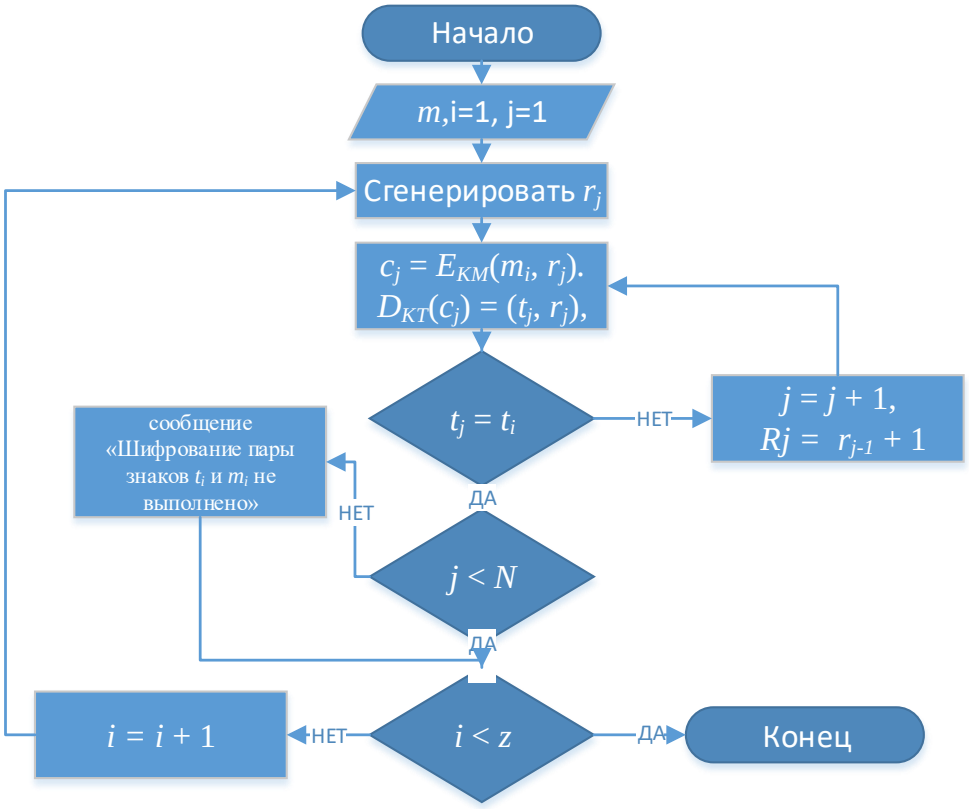


Рисунок 20. Блок-схема алгоритма разработанного способа ОШ

Так как предложенный алгоритм в своей основе имеет алгоритм блочного шифрования (например, алгоритм ГОСТ28147-89 в режиме простой замены), процедура расшифрования алгоритма очень проста. Для расшифрования блока шифртекста применяется процедура расшифрования согласно спецификации применяемого алгоритма блочного шифрования (в нашем примере, процедура расшифрования ГОСТ28147-89).

В работе [81] были описаны критерии эффективности функционирования предложенного алгоритма. Согласно работе, скорость алгоритма отрицаемого шифрования может быть оценена при использовании следующей формулы.

$$\lambda'_{\text{ОШ}} \approx (2^{-u-1}u/n)\lambda_{\text{БШ}}, \quad (1)$$

где

$\lambda'_{\text{ОШ}}$ – расчетная скорость алгоритма отрицаемого шифрования;

$\lambda_{\text{БШ}}$– скорость базового алгоритма шифрования;

u – количество битов данных;

n – количество битов случайной последовательности.

Скорость отрицаемого шифровая зависит от среднего числа η выбираемых значений r_j при шифровании одной пары знаков t_i и m_i(шаги 3-7 алгоритма ОШ). Значение η зависит от вероятности выполнения на шаге 6 условия $t_j = t_i$. Значение η можно вычислить по следующей формуле.

$$\eta \approx [\Pr(t_j = t_i)]^{-1} = 2^u, \quad (2)$$

Из формулы (1) следует, что эффективность представленного алгоритма ОШ зависит в основном от 2 параметров:

– скорости базового алгоритма шифрования;

– отношения размера блока данных к блоку случайных значений.

Также в ходе исследования были экспериментально проверены данные утверждения. Была подготовлена программа, реализующая предложенный алгоритм ОШ (с возможностью настройки параметров ОШ). В Таблице 2 показаны результаты измерений скоростных характеристик алгоритма ОШ при различных выбранных параметрах. В столбце 2 показано соотношение

скоростных характеристик алгоритма ОШ: эксперимента и формулы (1). Экспериментальное среднее числа η выбираемых значений r_jтакже соизмеримо с теоретическим значением.

Таблица 2. Результаты измерений скоростных характеристик алгоритма ОШ

Режим работы алгоритма ОШ	**Скорость шифрования секретного сообщения** эксперимент/формула (1),бит/с	**среднее число η выбираемых значений** эксперимент/формула (2),	**Скорость шифрования базового алгоритма,** бит/с
1	**2**	**3**	**4**
Размер блока – 32 бита (*u* = 8, *k* = 24*)*; алгоритм шифрования – RC5.	1150 / 1360	257 / 256	2728211
Размер блока – 32 бита (*u* = 4, *k* = 28*)*. алгоритм шифрования – RC5;	10260 / 10920	14 / 16	2728211
Размер блока – 64 бита (*u* = 8, *k* = 56*)*; алгоритм шифрования – ГОСТ28147.	1176 / 1000	240 / 256	4093953

Размер блока – 64 бита (u = 4, k = 60); алгоритм шифрования – ГОСТ28147.	9627 / 8000	15 / 16	4093953

Как уже упоминалось, скорость отрицаемого шифрования в основном зависит от скорости алгоритма шифрования блока. Также ощутимое влияние на скорость выполнения раунда отрицаемого шифрования оказывает размер блоков данных (значение u). Например, вероятность совпадения 4 бит (шаг 6 алгоритма) в 16 раз выше вероятности совпадения 8 бит. Но, к сожалению, уменьшение размера блоков данных неизбежно приводит к увеличению отношения размера криптограммы к размеру шифруемых сообщений. В данном случае зашифрованное сообщение будет больше исходных данных.

$$|t_i| + |m_i| < |C_i|$$

В случае принуждающей атаки предлагается утверждать, что в данном случае применялось вероятностное шифрование. По этой причине блок шифртекста больше шифруемых данных. Злоумышленник не сможет доказать обратного.

На рисунке 21 изображен график отношения количества выборок значений r_jк номеру шифруемого блока (ОШ ГОСТ28147, u = 4, k = 60).

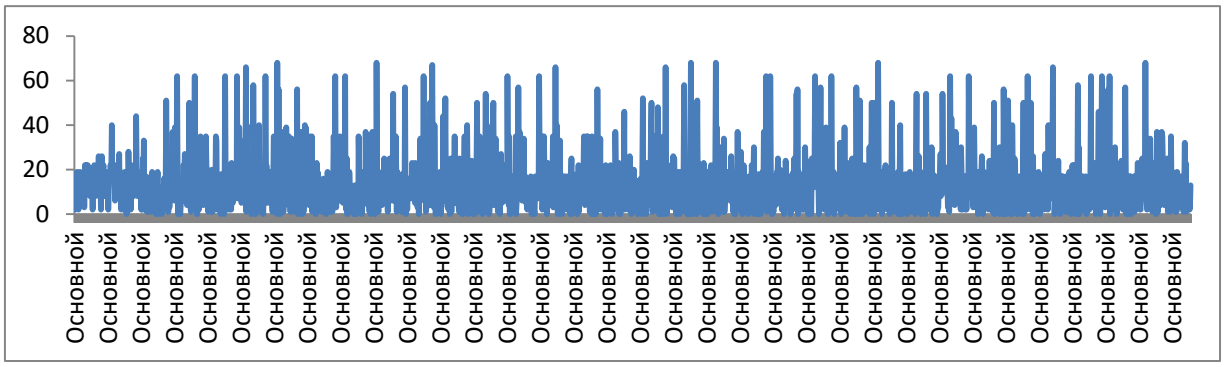


Рисунок 21. График зависимости числа пробных выборок значения rj от номера текущей пары шифруемых символов при использовании блочного шифра ГОСТ28147 и параметров u = 4, n = 64

Из графика видно, что большинство значений лежит в пределах 20 выборок на блок. Среднее число η выбираемых значений при подборе значений r_j близко к теоретическим значениям.

Представленные в данном разделе алгоритмы ОШ имеют практическую значимость и могут применяться в различных подсистемах защиты операционных систем. Используя современные вычислительные мощности и соблюдая вышеуказанные правила выбора параметров отрицаемого шифрования (с базовым алгоритмом шифрования ГОСТ 28147-89), можно достичь скоростей до 6Мбит/сек. Применение скоростных шифров [82] может значительно увеличить данное значение.

Анализ эффективности применения в мобильной ОС

Метод защиты: способ отрицаемого шифрования.

Отличительные особенности: обладает свойствами неотличимости от вероятностного шифрования, применяется блочное шифрование, имеет высокие скоростные характеристики.

3.2.4 Применение методов защитного преобразования, стойких к атакам с принуждением, в ОС

Высокая производительность предложенного в предыдущем разделе алгоритма ОШ позволяет применять алгоритмы отрицаемого шифрования на практике самыми различными способами. В ходе работы было предложено несколько практических применений алгоритмов отрицаемого шифрования[83]-[85], часть из которых реализована в разработанной операционной системе. Далее представлены направления практического применения алгоритмов отрицаемого шифрования.

При хранении важных данных в криптоконтейнерах. Хранение важной информации представляет собой наиболее очевидное применение отрицаемого шифрования. Так как скорость шифрования не позволяет шифровать значительные объемы данных, на первоначальном этапе предлагается использовать отрицаемое шифрование для сокрытия небольших сообщений

(например, криптографических ключей). В данной работе предложен способ применения ОШ для хранения ключевой информации (раздел 3.3)

В различных алгоритмах аутентификации. Алгоритмы отрицаемого шифрования также можно использовать при реализации подсистем аутентификации. В данном случае отрицаемое шифрование используется для маскировки рабочей области легитимного пользователя. Например, при загрузке системы в случае принуждающей атаки загрузится не профиль пользователя, а гостевой. Так как идея отрицаемого шифрования подразумевает невозможность идентифицировать в шифртексте наличие дополнительных сообщений, даже само наличие учетной записи пользователя в операционной системе может успешно скрываться от злоумышленника. В данной работе предложен алгоритм аутентификации, обладающий защитой от принуждающей атаки (разделы 2.2.2).

В протоколах одноразовых паролей. Разработаны алгоритмы одноразовых паролей, в которых в случае принуждающей атаки блокируются последующие действия пользователя (например, злоумышленник заставил пользователя войти в систему электронных платежей, но дальнейшие операции проводить уже невозможно). В данной работе предложен протокол одноразовых паролей, обладающий защитой от принуждающей атаки (разделы 2.2.3).

В алгоритмах защиты программного обеспечения от отладки (анализа). Поскольку при шифровании обоих сообщений в алгоритмах отрицательного шифрования проводятся одни и те же криптографические преобразования, злоумышленник не может предугадать дальнейшее поведение программы, просто анализируя код программы. К примеру, зашифрованными могут быть адреса следующей команды, а в качестве ключа может служить некоторый изменяемый параметр. Далее в работе предложен метод применения ОШ для защиты программного обеспечения от анализа (разделы 4.4).

Согласно [12] неоспоримой положительной особенностью алгоритмов отрицаемого шифрования является возможность применения в них блочных алгоритмов шифрования (в том числе отечественных алгоритмов). Данная

возможность позволяет применять отрицаемое шифрование на базе уже готовых программных продуктов, которые реализуют блочные алгоритмы. Это позволит значительно уменьшить время внедрения отрицаемого шифрования.

3.2.5 Ключевая инфраструктура

Криптографические алгоритмы широко применяются в разработанной операционной системе для защиты передаваемых данных, усиленной аутентификации, безопасного обновления и т.д. В процессе эксплуатации при взаимодействии с множеством защищенных объектов (серверы обновления, защищенные абоненты и т.д.) ставится вопрос доверия объектам взаимодействия. В каждом экземпляре операционной системы имеется достаточно большое количество идентификаторов и криптографических ключей.

Идентификаторы применяются в операционной системе, как по своему прямому назначению (идентификации), так и в протоколах формирования ключей. В таблице 3 показаны основные общесистемные виды идентификаторов.

Таблица 3. Общесистемные идентификаторы

№	Имя	размер	назначение
1	HW_ID	До 64бит	идентификатор аппаратной платформы
2	OS_ID	64 бит	идентификатор экземпляра операционной системы
3	GR_ID	64 бит	групповой идентификатор
4	AP_ID	64 бит	идентификатор приложения

Идентификатор аппаратной платформы применяется для защиты операционной системы от запуска на не доверенном оборудовании. В зависимости от возможностей аппаратной платформы размер идентификатора может быть различен. В контроллерах фирмы STMElectronics существует запрограммированный уникальный аппаратный идентификатор размером 48 бит. В случае если аппаратная платформа не имеет уникального идентификатора, в качестве идентификатора может быть использовано любое не перезаписываемое значение (например, серийный номер микросхемы).

Каждый экземпляр операционной системы должен иметь уникальный идентификационный номер. Данный идентификатор используется в подсистеме безопасного обновления операционной системы.

Групповой идентификатор также используется в процессах обновления операционной системы. Групповой идентификатор применяется в случае необходимости произвести обновление операционных систем определенного класса устройств. В 64 битах данного идентификатора предусмотрены разделы для описания типа аппаратной платформы, номера операционной системы и номера сборки. Также имеется несколько байт резерва.

Следующим важным вопросом является организация ключевой инфраструктуры. В операционной системе предусмотрено хранилище ключей. Это специально выделенное пространство в энергонезависимой памяти, в котором могут быть сохранены как пользовательские, так и встроенные (предустановленные) ключи операционные системы. Как уже упоминалось, в операционной системе активно используется как симметричное шифрование, так и ассиметричное шифрование. Симметричное шифрование используется тогда, когда необходимо обработать большое количество данных. Ключи для симметричного шифрования формируются на основе алгоритма открытого распределения ключей Диффи-Хеллмана [86]. По этой причине в операционной системе отсутствуют системные ключевые контейнеры симметричного шифрования. В таблице 4 отражены основные типы ключей (ассиметричные).

Таблица 4. Общесистемные типы ключей

№	Имя	Назначение
1	HW_KEY	главный ключ
2	OS_KEY	ключ операционной системы
3	GR_KEY	групповой ключ
4	AP_KEY	ключ приложения
5	UP_KEY	ключ обновления

Стоит отметить, что каждая категория ключей из таблицы содержит открытый и закрытый ключ. Особенностью использования ассиметричного шифрования является необходимость применять процессы контроля аутентичности открытой части ключа.

3.3.Способ применения методов защитного преобразования, стойких к атакам с принуждением, для хранения ключей

В разработанной операционной предлагается применять отрицаемое шифрование для сокрытия наличия резервных серий ключевой информации. На рисунке 22 представлен алгоритм хранения набора резервных серий ключевой информации. Контейнер ключей представляет собой файл, который содержит в себе значение шифртекста C. Для вычисления данного значения для n серий ключей необходимо найти решение системы уравнений

$$\begin{cases} E_{K1}(C)\ mod\ 2^r = M_1 \\ \dots \\ E_{Kn}(C)\ mod\ 2^r = M_n \end{cases}$$

где $K_1 \dots K_n -$ ключи шифрования ключей (пароль фиксированной длины); $M_1 \dots M_n -$ защищаемые серии ключей; $E -$ функция шифрования; $C -$ криптограмма; r-разрядность криптограммы. Информация о наличии резервных серий ключевой информации может иметь значительную ценность для злоумышленника. В качестве алгоритма шифрования применяется ГОСТ 28147-89. Так как длина файла шифртекста не зависит от количества зашифрованных серий ключевой информации, у злоумышленника отсутствует возможность доказать, что существуют несколько серий ключей.

Дополнительной положительной особенностью данного способа хранения ключевой информации является возможность совместить процесс генерации ключевой информации и выработки файла - контейнера ключей.

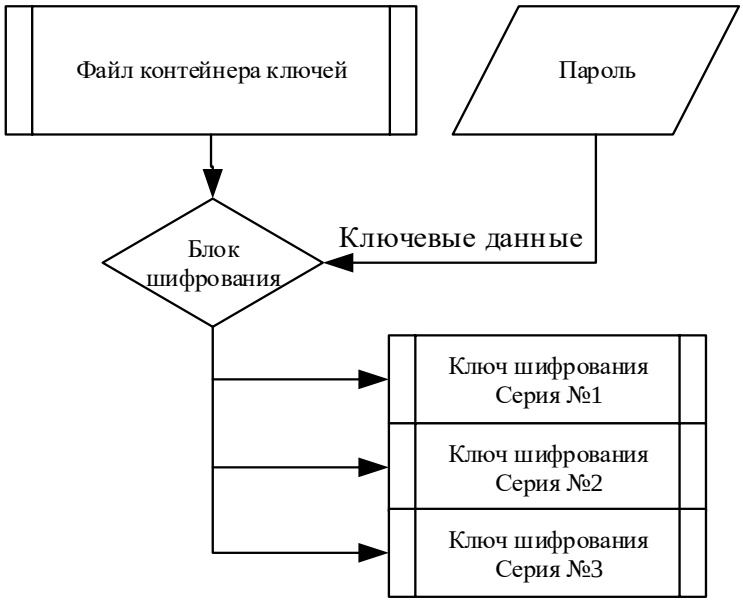


Рисунок 22. Использование отрицаемого шифрования для хранения резервных серий ключевой информации

В данном случае вычисление сводится к нахождению n значений ключей шифрования ключей $K_1 \dots K_n$ по формуле

$$E_{Kn}(C)\ mod\ 2^r = M_n.$$

Стоит отметить, что данный метод может применяться только с учетом требований целевых алгоритмов шифрования для ключевой информации.

Анализ эффективности применения в мобильной ОС

Метод защиты: Способ применения отрицаемого шифрования для хранения ключей.

Отличительные особенности: сокрытие факта наличия резервных серий ключевой информации, возможность интеграции с процессом генерации ключевой информации (при учете снижений требований к ключам).

3.4. Защищенная файловая система

Наличие в операционной системе подсистемы защиты хранимых файлов значительно повышает общий уровень защищенности операционной системы. Важность обеспечения безопасности хранимой информации рассмотрена в работах [87], [88]. Какие бы совершенные средства управления доступом ни

применялись в операционной системе, данные средства действуют только на субъекты, существующие в рамках модели ОС. Злоумышленник может просто извлечь жесткий диск из ПК и произвести анализ хранимых данных.Для защиты от данного вида угроз используют защищенные файловые системы.

Одной из наиболее популярных реализаций защищенной файловой системы является Encrypted File System (EFS) разработки компании Microsoft. Данная файловая система имеет достаточно простой и как следствие "прозрачный" для понимания алгоритм работы.

Высокая производительность EFS обусловлена применением различных типов алгоритмов шифрования. Как известно симметричное шифрование имеет высокие скоростные характеристики. Для шифрования основного объема файла (тела файла) в EFS применяется симметричные алгоритмы 3DES или AES. Применение симметричного шифрования неизбежно приведет к необходимости решать проблемы управления ключевой информации в операционной системе:

– хранение ключевой информации для каждого файла (применять один ключ для шифрования всех файлов нецелесообразно);

– невозможность совместного использования одного файла.

Для решения данных проблем в EFS для шифрования кличей шифрования файлов используется ассиметричное шифрование. Применение ассиметричного шифрования обеспечивает:

– безопасное хранение ключа шифрования файла;

– интеграцию с корпоративной инфраструктурой открытых ключей (PKI).

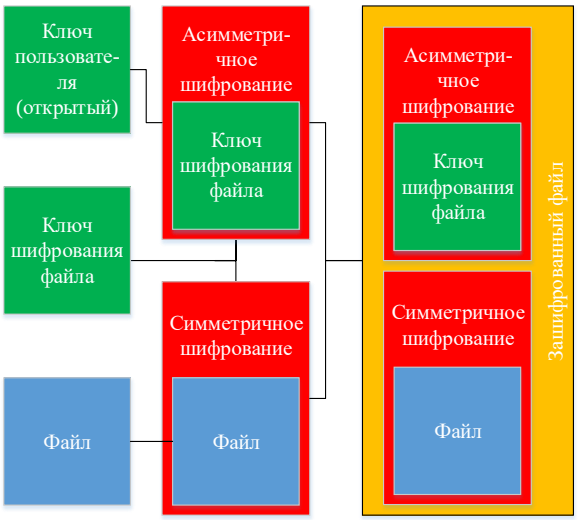


Рисунок 23. Алгоритм шифрования файла в файловой системе EFS

На рисунке 23 изображена блок схема алгоритма шифрования файла в файловой системе EFS. Процесс шифрования файла состоит из следующих шагов:

– генерация ключа шифрования файла (для каждого файла собственный);

– шифрование тела файла ключом шифрования файла (симметричный алгоритм шифрования);

– шифрования ключа шифрования файла персональным открытым ключом пользователя;

– скрепление зашифрованного ключа шифрования файла к зашифрованному телу файла.

Вышеуказанные операции выполняются в EFS, прозрачно для пользователя. Производительность современных процессоров позволяет сделать процесс шифрования файла незаметным для пользователя (процессоры фирмы Intel в данный момент имеют собственный криптографический сопроцессор).

Для расшифрования файла необходимо выполнить ряд следующих шагов:

– откреплення зашифрованного ключа шифрования файла от тела фала;

– расшифрования ключа шифрования файла с использованием персонального закрытого ключа пользователя;

– расшифровать тело файла с использованием полученного ключа шифрования файла.

EFS представляет собой пример простой и эффективной защищенной файловой системы.

Анализ эффективности применения в мобильной ОС.

Метод защиты: файловая система EFS.

Уровень эффективности: высокий.

Достоинства: высокая производительность файловой системы, высокая защищенность данных.

Недостатки: отсутствует возможность проверки аутентичности данных, уязвимость ключевой подсистемы (хранение ключей производится на том же жестком диске).

Стоит упомянуть, что на территории РФ предусмотрена сертификация средств криптографической защиты информации в соответствии с [89].

Потребность в безопасном и прозрачном хранении данных обуславливает несомненную необходимость реализации в операционной системе защищенной файловой системы. Термин виртуальная память обычно ассоциируется с возможностью адресовать пространство памяти, гораздо большее, чем емкость первичной (реальной, физической) памяти конкретной вычислительной машины. [90].

В структуре разработанной операционной системы представлена защищенная виртуальная файловая система. В операционной системе используется известная концепция "все есть файл". То есть все возможные объекты файловой системы, доступ к которым может быть необходим процессам, проецируются на файловую систему. К таким объектам могут относиться:

– запущенные процессы;

– хранилища данных различного типа (флеш-память, энергонезависимая память и др.);

– интерфейсы взаимодействия с внешней средой (интерфейсы RS-232, RS-485, TWI).

Структура виртуальной файловой системы разработана на основе анализа существующих файловых систем.

Ввиду того факта, что данная файловая является виртуальной, а значит «высокоуровневой», перед файловой системой не ставились такие специализированные задачи файловых систем как:

– журналирования (для организации возможности «отката» изменений),

– реализации сред загрузки (загрузочные сектора),

– реализация различного вида средства резервирования данных.

Все вышеуказанные функции реализуются на низком (по отношению к виртуальной файловой системе) уровне - в конкретной файловой системе низкого уровня (FAT32, NTFS и т.д.) К одной из основных задач, которые решает виртуальная защищенная файловая система, относится унификация доступа к файлам на различных носителях (флэш-память, энергонезависимая память и т.д.). Не секрет, что в различных устройствах хранения данных используются различные методы записи или чтения информации. В энергонезависимую память некоторых производителей возможна только последовательная запись (блок за блоком). Флэш-память (на примере файловой системы FAT32) позволяет производить операции записи-чтения минимальными блоками данных (минимум 512 байт). В концепции виртуальной файловой системы запись на вышеуказанные устройства хранения данных может различаться лишь по двум характеристикам:

– максимально-возможный размер файла (зависит от объема доступных блоков данных устройства хранения),

– скорости записи (скорость в энергонезависимую память много меньше записи данных на современные флэш-накопители).

Защита хранимых данных закладывалась как в равной степени важная для существования файловой системы, как и реализация объектов файловой системы

(файлов и каталогов). По этой причине на этапе проектирования к виртуальной файловой системе выдвигались следующие требования:

– обеспечение безопасности хранимой информации путем ее криптографического преобразования,

– возможность реализации различных методов разграничения доступа к ресурсам файловой системы (наличие у объектов файловой системы меток доступа, матриц доступа и т.д.)

Файловая система состоит из файловых двух типов файловых объектов: собственно, файлов и каталогов. В контексте виртуальной файловой системы файл представляет собой ссылку. Данная ссылка содержит заголовок файла, который включает информацию о файле (таблица 5)

Таблица 5. Заголовок файла

	Имя	**Назначение**
1	char name [16]	символическое имя фала
2	U16 flag	характеристика файла (тип фала, место хранения, и т.д.)
3	U8 metka	зарезервировано
4	U8 ovner	владелец файла
5	U8 group	группа владельца файла
6	U32 size	размер файла (количество стандартных блоков)
7	void * fo	признак «открытого» файла (ссылка на заголовок открытой операции)
8	void * p	ссылка на тело файла

Заголовок файла имеет размер 17 байт. В таблице 6 отражена структура каталогов разработанной операционной системы.

Таблица 6. Структура файловой системы

№	Имя	Назначение
1	/	корневой каталог
2	/etc	основной каталог настроек приложений
3	/mnt	каталог монтирования
4	/mnt/CSA	каталог защищенного пространства хранения криптографических контейнеров
5	/mnt/0	точка монтирования внешнего запоминающего устройства
6	/bin	каталог приложений
7	/proc	каталог оперативной памяти

Структура каталогов файловой системы представляет дерево каталогов, которое начинается с корневого каталога. Корневой каталог представляет собой специальный динамический каталог. В процессе запуска операционной системы формируется и структура корневого каталога. В зависимости от фактического расположения данных (внутренняя память, внешнее устройство хранения) в тело корневого каталога записываются значения адресов дочерних каталогов, размер стандартного кластера и количество доступных для записи кластеров.

Механизмы защиты, которые применяются для предотвращения несанкционированных действий с файлами, зависят от хранимой информации и**,** конечно**,** от места физического хранения (внутренняя память, внешнее устройство хранения). Для защиты файлов на внешних и отчуждаемых носителях применяются функции прозрачного шифрования, подписи данных. На рисунке 24 показана структура файла, хранимого на внешнем носителе.

```
1  ##################################################
2  [header]
3  name= 1.txt
4  flag= 29ff
5  metka= 0000
6  oyner= 0000
7  group= 0000
8  offset= 0116
9  sign_h= AAAAAAAAAAAAAAAAAAAAAA
10 sign_m= BBBBBBBBBBBBBBBBBBBBBB
11 key_m= CCCCCCCCCCCCCCCCCCCCCC
12 ##################################################
13 12345testtesttesttest
```

Рисунок 24. Структура защищенного файла

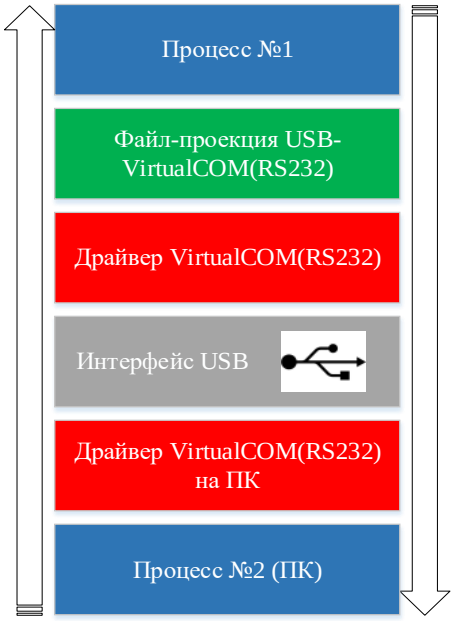


Рисунок 25. Взаимодействие процесса с интерфейсом USB

Структура защищенного файла состоит из дополнительного заголовка файла и тела файла**,** в котором хранится исходный файл целиком (в зашифрованном виде).

Концепция "все есть файл" позволяет унифицировать доступ процессов к абсолютно любым периферийным устройствам. На рисунке 25 представлен процесс передачи данных по интерфейсу USB.

Процесс передачи данных через интерфейс состоит из следующих этапов:

– процесс №1, используя стандартные функции доступа к файловой системе, производит запись в файл данных которые необходимо передать;

– ядро операционной системы использует драйвер последовательного интерфейса USB (в данном случае Virtual COM - виртуальный интерфейс RS232);

– аппаратный интерфейс USB передает данные на ПК;

– драйвер последовательного интерфейса USB на ПК (также Virtual COM);

– процесс №2 подключается к виртуальному COM-порту, и получает переданные данные.

С одной стороны, процесс передачи данных не выглядит простым. Однако для процесса №1 передача данных это обыкновенная операция записи в файл. Этот факт позволяет сделать операционную систему унифицированной и абстрагировать прикладное программное обеспечение от необходимости взаимодействовать с аппаратной платформой.

Анализ эффективности применения в мобильной ОС

Метод защиты: защищенная файловая система.

Отличительные особенности: интеграция в ФС электронной подписи, возможность применения различных моделей разграничения доступа (мандатной, дискреционной), наличие файловых объектов различных типов (метод, файл, каталог) – реализация метода "все есть файл".

3.5. Защита передаваемых данных

Защита передаваемых по сети данных осуществляется посредством системной службы - службы передачи данных. Данная служба представляет собой специализированный процесс уровня ядра. Служба передачи данных, как и уже рассмотренные службы криптопровайдера и доступа к файловой системе, обслуживает прикладные процессы в порядке очереди. Как и криптопровайдер**,** служба передачи данных имеет в своем составе хранилище ключей, в которое загружаются ключи шифрования данных различных процессов. При необходимости передачи данных в зашифрованном виде процесс предварительно

загружает ключи в хранилище. На рисунке 26 изображена структурная схема службы передачи данных.

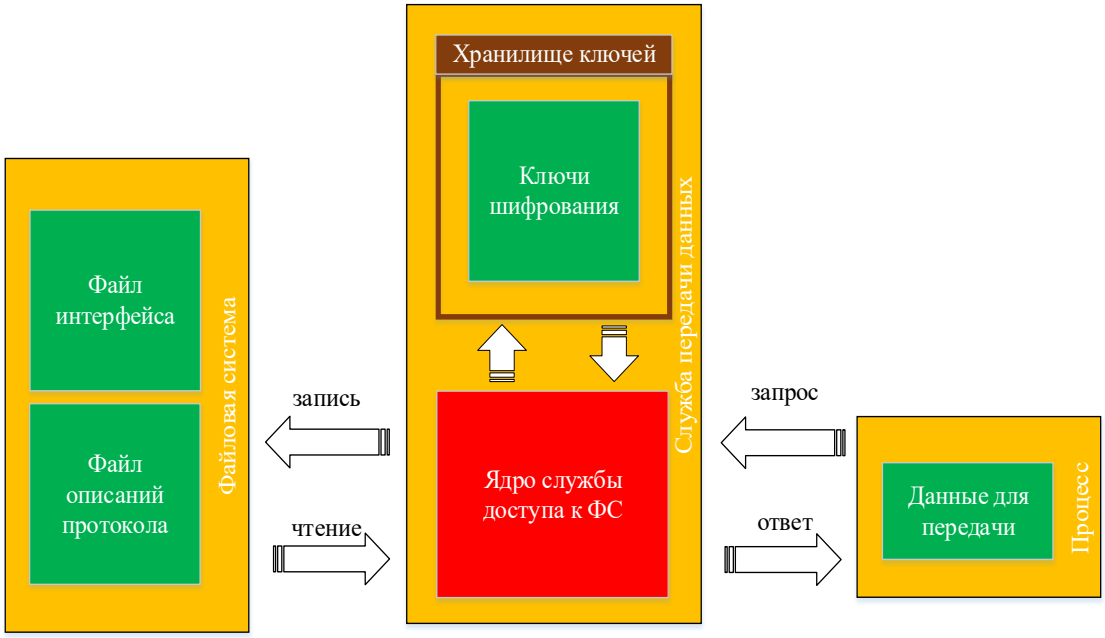


Рисунок 26. Структурная схема службы передачи данных

Особенностью службы передачи данных передачи является наличие возможности модификации прикладных протоколов взаимодействия при необходимости. Все поддерживаемые протоколы передачи данных представляют собой специализированные файлы описания, расположенные в предопределенных каталогах:

– для общесистемных протоколов (такой как удаленный интерпретатор командной строки): путь к файлам описания «/lib/net/proto/»;

– для прикладных процессов (путь объявляется предварительно, до использования вызова функций протокола): путь к файлам описания является относительным, например, «/mnt/0/proc1/lib/proto/».

Файл описания протокола передач данных представляет собой файл определенной структуры, который содержит все необходимые для обмена. К таким параметрам относятся:

– системное имя протокола,

– направление взаимодействия (симплекс, дуплекс, полудуплекс),

– формат передаваемого пакета,

– описание вызываемых процедур.

Создать свой собственный протокол производителю прикладного программного обеспечения не составит труда. Самой сложной частью файла описания является описание вызываемых процедур. В данном блоке файла описания можно использовать ссылки на исполняемые файлы с необходимыми параметрами. В основе службы передачи данных лежит механизм инкапсуляции (аналогично модели взаимодействия открытых систем[91]), что позволяет изолировать разработчика от необходимости взаимодействия с аппаратными интерфейсами.

Анализ эффективности применения в мобильной ОС

Метод защиты: подсистема передачи данных.

Отличительные особенности: поддержка настройки протоколов передачи, интеграция сервиса передачи данных с криптографической защитой данных.

3.6 Выводы к третьей главе

Данная глава была посвящена методам защиты передаваемой и хранимой информации. Криптографическая защита является одной из наиболее простых для реализации. Поэтому криптографические алгоритмы часто применяются в различных подсистемах операционных систем. В данной главе были описаны разработанные эффективные алгоритмы защитного преобразования данных. Описаны методы практического применения данных преобразований методов защитного преобразования, стойких к атакам с принуждением в ОС.

Далее была описана подсистема защиты передаваемых данных и защищенная файловая система.

В данной главе решены следующие задачи исследования:

– разработка метода защиты хранимой информации стойкий к атакам с принуждением пользователя раскрыть ключ защитного преобразования;

– разработка метода защитного преобразования передаваемой по открытым каналам информации, стойкий к атакам с принуждением пользователя раскрыть ключ защитного преобразования.

Глава 4. Разработка безопасной мобильной ОС и подсистемы защиты ПО

Данная глава посвящена описанию разработанной модели защищенной операционной системы для мобильных систем. Будут предложены усовершенствованные методы защиты информации. Для предложенных методов защиты будут выделены отличительные особенности, которые позволяют эффективно применять данные методы в мобильных операционных системах. Будет описана архитектура подсистемы защиты программного обеспечения разработанной мобильной операционной системы.

4.1. Аппаратная платформа

Для выполнения отладки и испытаний разработанной операционной системы были спроектированы отладочные аппаратные платформы (макетные стенды). Стенды (рисунок 27) имитируют некоторые распространенные классы средств защиты информации, в том числе:

– криптографическое хранилище информации,

– персональное идентифицирующее устройство для интерфейса USB (USB-токен).

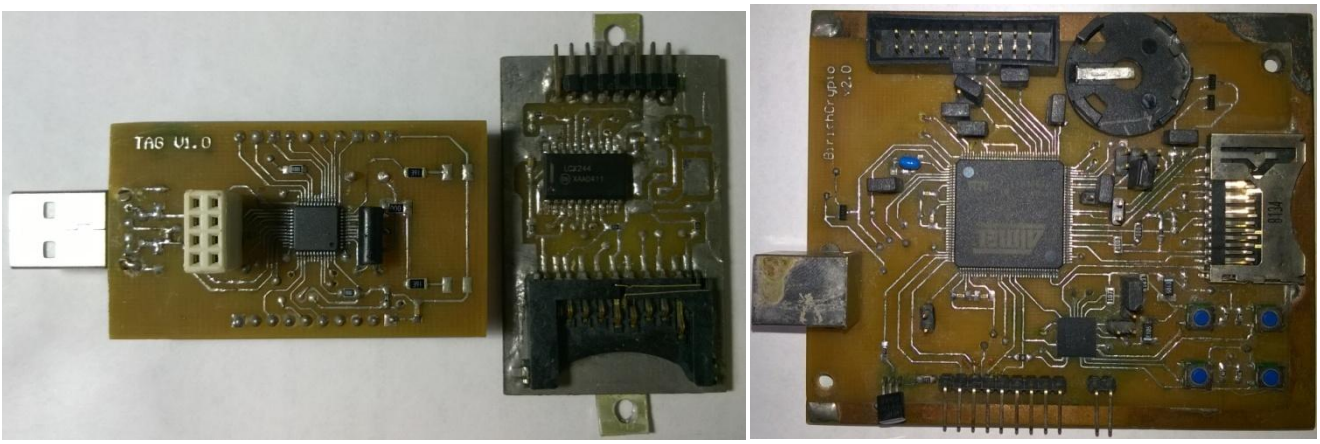


Рисунок 27. Стенды идентифицирующего устройства (слева) и криптографического хранилища информации (справа)

Макетные стенды разрабатывались с учетом специфики использования устройств, существующих классов аппаратных средств защиты информации [92] и нововведений в микроэлектронике. Технические характеристики устройств приведены в таблице 1.

Таблица 1.Технические характеристики

Назначение стенда	Криптографическое хранилище информации	Идентифицирующее устройство
Вычислительный блок	ATSAM3U4C x32, ARM, Cortex-M3, 96МГц,ПЗУ 256Кб, ОЗУ 52Кб, I/O 57, встроены часы реального времени LPC1111FHN33 x32, ARM, Cortex-M0, 50МГц,ПЗУ 8Кб, ОЗУ 2Кб, I/O 28	STM32F103RE x32, ARM, Cortex-M3, 72МГц, ПЗУ 256Кб, ОЗУ 64Кб, I/O 51, встроены часы реального времени
Интерфейс коммуникаций	SPI, MMC, USB 2.0, JTAG	SPI, USB 2.0, JTAG
Хранение данных	SDHC карта памяти, встроенная флеш память	Micro SDHC карта памяти, встроенная флеш память
Особенности	Скорость USB 400 Мбит/сек, быстрый интерфейс карты памяти, управление (4 клавиши), ЖКИ дисплей (24 символа 2 строки), поддержка SDHC карт до 32 Гб, производительный контроллер для обработки больших объемов данных, наличие второго контроллера для реализации механизма "черный ящик"	Компактный размер, форм-фактор USB-токен, поддержка micro SDHC карт до 16 Гб, низкое потребление электроэнергии, защита от статики

Для адаптации работы операционной системы на различных аппаратных платформах для исследования намеренно были выбраны контроллеры различных производителей, в том числе:

- высокопроизводительный ATSAM3U4C фирмы Atmel[93];
- энергоэффективный STM32F103RE фирмы STMicroelectronics [94];
- сверхкомпактный LPC1111FHN33 фирмы NXP[95].

Как можно видеть из таблицы каждая из макетных плат имеет свои положительные стороны. Криптографическое хранилище информации представляет собой устройство, к которому подключается хранилище

информации (SDHC карта памяти), которое подключается к персональному компьютеру посредством интерфейса USB. Структурная схема стенда изображена на рисунке 28. Стенд имеет достаточно большие вычислительные возможности. В состав схемы входят два высокопроизводительных 32-х разрядных микроконтроллера архитектуры ARM. Первый – центральный микроконтроллер ATSAM3U4C имеющий производительность практически 100 миллионов операций в секунду. Данный контроллер предназначен для обеспечения потокового шифрования данных на диске. Также особенность данного контроллера является встроенный контроллер высокоскоростной шины USB 2.0 (режим High Speed) 400 Мбит/сек., что обеспечит комфортную работу с устройством хранения. Второй контролер – LPC1111FHN33 имеет меньшую производительность (50 миллионов операций в секунду). Данный контроллер применяется для организации пользовательского интерфейса, хранения ключей пользователя. Оба контроллера соединены скоростным интерфейсом SPI (скорость 25 Мбит/сек.).

Стенд с двумя вычислительными платформами также был использован для испытания механизма "черный ящик", предложенного в [96]. Данный механизм позволяет использовать менее стойкие криптографические алгоритмы без ущерба общей защищенности криптографической системы.

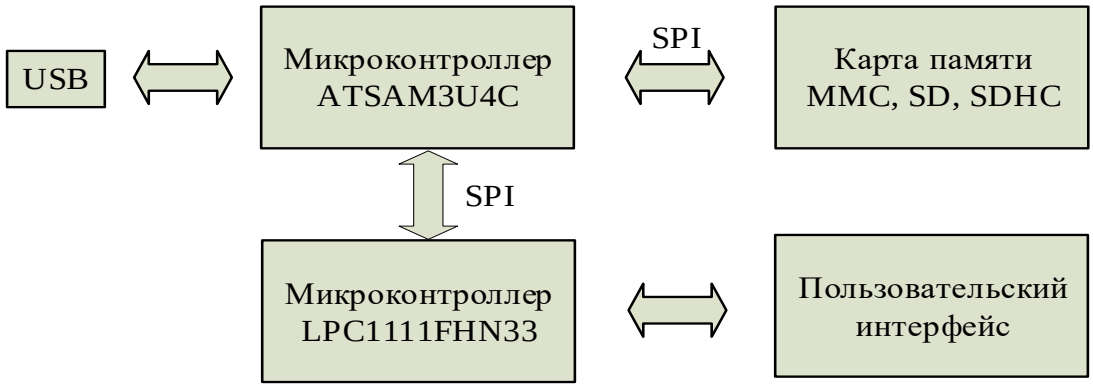


Рисунок 28.Структурная схема стенда криптографического хранилища информации

На рисунке 29 представлена структурная схема второго стенда, разработанного для откладки операционной системы на базе классического персонального идентифицирующего устройства. Устройства данного класса предназначены для организации систем усиленной (многофакторной) аутентификации, хранения ключевых контейнеров и др. Устройство должно отличатся компактностью, энергоэффективностью (а также энергонезависимостью) и наличием достаточного количество постоянной памяти.

Рисунок 29. Структурная схема стенда идентифицирующего устройства

Вычислительная платформа стенда представлена микроконтроллером STM32F103RE. Данный контроллер имеет высокую производительность (72 миллиона операций в секунду). Также в контроллер интегрированы часы реального времени. Энергоэффективность контроллера обусловлена наличием нескольких планов электроснабжения ядра (высокая производительность, работа от батарей). Различные режимы работы необходимы для выполнения операций резервирования важных операций при отлучении устройства от постоянного электропитания.

4.2. Ядро операционной системы

Наиболее распространенные архитектуры ядра операционной систем были рассмотрены в первой главе. Разработанная операционная мобильная операционная система организована по классической архитектуре типа «микроядро». Выбор архитектуры данного типа был сделан по следующим причинам:

– необходимостью реализации в системе режима «аварийного завершения» (так как система будет эксплуатироваться в неблагоприятных условиях мобильных устройств);

– теоретически достаточно низкая производительность аппаратной платформы (мобильные системы имеют сравнительно небольшую производительность);

– многоцелевой характер применения операционной системы (различным видам устройств необходим различный набор сервисных приложений).

В разработанной операционной системе реализовано микроядро, которое, в зависимости от условий эксплуатации, может находиться в одном из трех режимов работы.

Нормальный режим функционирования. Нормальный (штатный) режим работы ядра реализует все необходимые службы (набор служб различается в зависимости от применения операционной системы). Операционная система выделяет прикладным приложениям всю необходимую память. Процессорное время в равной мере делится между процессами. Аппаратная платформа находится в производительном режиме (в случае если аппаратная платформа поддерживает управление электропитанием).

Предаварийный режим функционирования. В данный режим переходит ядро при возникновении события опасного функционирования операционной системы (например, при отключении постоянного электропитания аппаратной платформы). Задачи данного состояния ядра:

– по возможности завершить (закрыть или сохранить состояние) прикладные процессы для предотвращения потери пользовательских данных;

– записать данные оперативной памяти (критические области - например, криптографические контейнеры) в энергонезависимую память;

– завершить все существующие операции записи на диск для предотвращения повреждения структуры хранилища данных.

В ходе обеспечения вышеуказанных задач ядро выделяет под свою работу большинство процессорного времени.

Энергосберегающий режим. В аппаратных платформах, в которых реализована возможность работы от автономного источника питания, может применяться энергосберегающий режим работы ядра.

Выбор конкретного режима работы производится операционной системой в автоматическом режиме. Выбор режима работы основывается на правилах, которые описаны в динамических настройках операционной системы. Файл конфигурации ядра операционной системы расположен по адресу «**/etc/kernel.conf**». Все необходимые параметры представлены в виде текстовых значений. В листинге 1 отражена часть конфигурационного файла операционной системы.

```
/*!<
System frequency (Hz).
*/
#define CFG_CPU_FREQ              (72000000)

/*!<
systick frequency (Hz).
default 100 Hz - 10 ms
*/
#define CFG_SYSTICK_FREQ          (100)
#define MAX_TASKS                 10

/* Process stack */
#define SP_PROCESS                0x02

/* Process stack size */
#define DEFAULT_STACK_SIZE        0x200
```

Листинг 1. Параметры ядра операционной системы

Для настройки того или иного параметра достаточно изменить значение в файле. В зависимости от типа параметра в качестве значения может быть число, логическое значение или текстовое поле. Применительно к представленному примеру, в файле конфигурации могут быть установлены следующие параметры:

– частота тактирования ядра (CFG_CPU_FREQ); данный параметр влияет на время работы системных обработчиков событий;

– частота срабатывания события переключения контекста потока (CFG_SYSTICK_FREQ); данный параметр имеет важное значение, так как он определяет минимальное время выполнения процесса (по истечению данного времени управление будет передано другому процессу).

– максимальное количество прикладных процессов (CFG_CPU_FREQ); количество запущенных процессов устанавливается в зависимости от количества оперативной памяти;

– адрес расположения и размер стека, который используется для хранения значений переменных выполняемых процессов.

Концепция динамической конфигурации с одной стороны позволит операционной системе быть достаточно гибкой для реализации задач самого разного характера. С другой стороны, динамические параметры обуславливает появление некоторого количества уязвимостей. Для защиты ядра операционной системы от атаки, основанной на вызове ошибок путем модификации конфигурационных файлов, в операционной системе используется механизм электронной подписи. Подробно данный механизм будет рассмотрен позднее.

Анализ эффективности применения в мобильной ОС

Метод защиты: адаптированное ядро.

Отличительные особенности: адаптированность к эксплуатации в мобильном режиме, защита критических данных от потери, возможность конфигурирования ядра для работы в различных условиях, поддержка различных архитектур.

4.3. Подсистема виртуальной программной среды

Виртуализация является перспективным направлением в разработке операционных систем. В работах [97], [98], [99] рассмотрены перспективные методы применения виртуальных сред в автоматизированных системах.

Подсистема виртуальной программной среды предназначена для защиты ресурсов операционной системы от алгоритмических атак через выполняемое программное обеспечение (рисунок 30).

В операционной системе предусмотрена система виртуальных команд. Для удобства разработчиков программного обеспечения виртуальные команды имеют схожий синтаксис с языком ASSEMBLER. Данное решение позволяет использовать при компиляции синтаксические анализаторы различных языков программирования высокого уровня (C, Basic, Pascal и т.д.).

Рисунок 30. Архитектура виртуальной программной среды

Во многих операционных системах применяются системы виртуального выполнения команд (например, виртуальная среда операционной системы Microsoft Windows [100], [101]). Основными задачами производителя операционных систем является:

– подготовить достаточно безопасную для операционной системы среду выполнения;

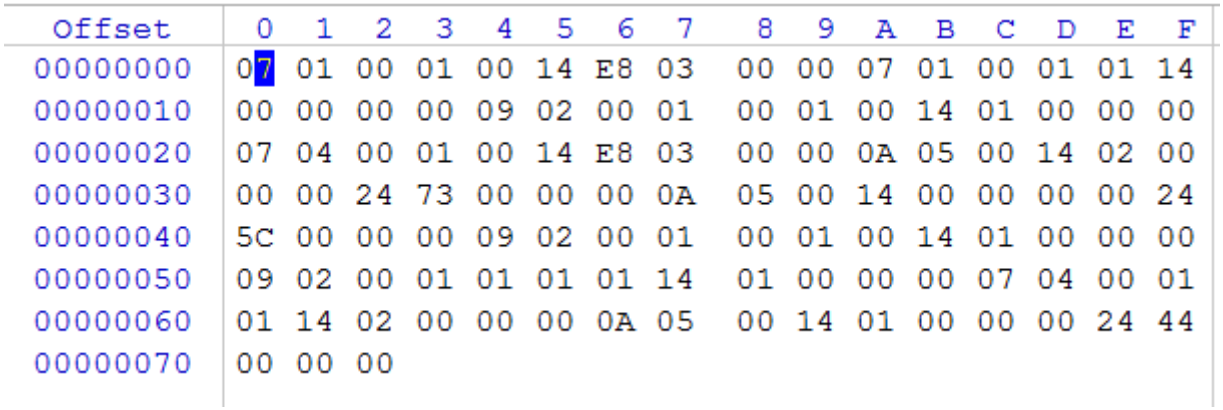
– обеспечить приемлемый интерфейс взаимодействия для разработчиков виртуальных программ.

Как уже упоминалось, в разработанной операционной системе в качестве базового языка выбран язык ASSEMBLER. А точнее используемый язык синтаксически схож с языком ASSEMBLER. Структура программы, типы данных и некоторые другие аспекты все же имеют собственную специфику. В листинге 2 показан пример текста файла программы. В данном примере можно видеть общеизвестные функции копирования памяти (оператор MOV), функции сравнения (CMP). Данные операторы применяются с небольшими особенностями (отличаются типы применяемых параметров, а точнее типы данных). Одновременное с этим в программе есть и специфические команды ядра ARM. К таким командам, например, относятся функции условного и безусловного перехода (оператор B).

```
[VAR]=0x0
[M]=0x3
[DATA]
[START]
MOV r0,#0x3e8
MOV r1,#0x0
ADD r0,r0,#0x1
CMP  r0,#0x3e8
B      #0x2,&0x2
NOP
B      #0x0,&0x1
|0x0|
ADD r0,r0,#0x1
ADD r1,r1,#0x1
|0x1|
CMP  r1,#2
B      #0x1,&0x0
|0x2|
[ENDP]
```

Листинг 2. Файл текста программы

После написания программы производится компиляция программы. Для этого применяется специальное приложение - компилятор. Компилятор выполняется на аппаратной платформе. Данное решение позволяет испытать разработанное приложение на целевой платформе и избежать ошибок, которые могут быть обусловлены ошибкой программиста и особенностями применяемой аппаратной платформы. На вход компилятора поступает файл текста программы. Который преобразуется в машинный код программы (рисунок 31).



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00000000	07	01	00	01	00	14	E8	03	00	00	07	01	00	01	01	14
00000010	00	00	00	00	09	02	00	01	00	01	00	14	01	00	00	00
00000020	07	04	00	01	00	14	E8	03	00	00	0A	05	00	14	02	00
00000030	00	00	24	73	00	00	00	0A	05	00	14	00	00	00	00	24
00000040	5C	00	00	00	09	02	00	01	00	01	00	14	01	00	00	00
00000050	09	02	00	01	01	01	01	14	01	00	00	00	07	04	00	01
00000060	01	14	02	00	00	00	0A	05	00	14	01	00	00	00	24	44
00000070	00	00	00													

Рисунок31. Файл скомпилированной программы

Бинарный код программы состоит из 16 байтовых команд с набором параметров. Например, на рисунке 31 программа начинается с команды **0x070100010014E8030000,** где **0x070100** – идентификатор. За идентификатором команды следует перечисление операндов. Каждый операнд имеет свой тип (регистр, переменная и т.д.) и значение. **0x0100** – первая операнда (тип операнды регистр, значение **0x00**). **0x14E8030000** – вторая операнда (тип операнды переменная 4 байта, значение **0xE803**). В итоге вышеуказанная машинная команда будет распознана виртуальной средой выполнения как команда: «**MOVR0, 1000».** В виртуальный регистр **R0** будет записано десятичное значение **1000.**

В операционной системе реализована подсистема управления памятью. Так как управление памятью является одной из самых важных задач операционной системы, подсистема управления памятью встроена в ядро операционной системы. Управление память построено по классической схеме. В подсистеме реализованы следующие схемы организации памяти:

– куча — это специализированная структура данных типа дерево;

– стек – динамическая структура данных (основное правило: последним пришёл – первым ушёл);

– очередь – это линейная динамическая структура данных (основное правило: добавление новых данных – в конец, удаление – с начала).

Все вышеуказанные структуры доступны для применения прикладными программными продуктами. Модули уровня ядра имеют возможность прямого использования основного стека (общего) операционной системы. К основным задачам подсистемы управления памятью относится:

– выделение областей памяти процессам;

– освобождение памяти (после завершения выполнения процесса);

– оптимизация памяти.

В мобильных системах доступная оперативная память операционной системы является весьма ограниченным ресурсом. Оптимизацию (дефрагментацию) памяти необходимо выполнять постоянно в процессе работы операционной системы.

Как уже упоминалось, в операционной системе применяется принцип «все есть файл». Каждое системное устройство, порт ввода-вывода имеет свое отображение в файловой системе. С точки зрения памяти каждое такое отображение представляет собой ссылку на функцию ядра, файл и т.д. Каждый объект, размещенный в памяти, имеет собственное именование (адрес). В операционной системе применяются классические типы адресации**,** описанные ранее.

Абсолютная адресация (прямое указание на адрес в памяти микроконтроллера) применяется только в ядре ОС и некоторых системных службах. Данное решение было принято в целях обеспечения безопасности. Абсолютная адресация может дать злоумышленнику информацию об аппаратной платформе, которая может быть использована, например, для атаки переполнения памяти. Возможны и другие атаки.

В операционной системе предусмотрены механизмы системы обнаружения вторжения. В частности, выполняется контроль использования оперативной памяти и контроля доступа к файловым объектам. Так как для запуска пользовательских приложений применяется виртуальная машина**,** основным видом адресации является относительная адресация. Пользовательское приложение нет необходимости получать данные извне своего пространства памяти. Все объявленные переменные размещаются в заранее определенном месте. Адресация памяти приложения является относительной по отношению к стартовому адресу начала стека, который выделен приложению перед его запуском. Виртуальная машина следит, чтобы приложение ни в коем случае не попало за пределы своего адресного пространстве. При попытке получить доступ вне выделенного адресного пространства приложение будет завершено с ошибкой. Кроме этого все попытки доступа могут производиться только с использованием функций доступа виртуальной машины, в которых применяются только относительные адреса. Регистровая адресация также применяется в операционной системе.

Анализ эффективности применения в мобильной ОС

Метод защиты: подсистема виртуального выполнения команд.

Отличительные особенности: применение языка ASSEMBLER упрощает "портирование" существующих приложений в мобильную ОС, интеграция виртуальной среды с подсистемой защиты памяти, интеграция виртуальной среды с системой обнаружения вторжений.

4.4.Защита от анализа приложений

Методы защиты от анализа (отладки) приложений целесообразно разделить по методу анализа (отладки) приложений:

– методы противодействия статическому анализу (дизассемблерирование),

– методы противодействия активному анализу (отладке),

– методы противодействия эмуляции.

Так как статический анализ основан на исследовании низкоуровнего (машинного) кода программы, все методы противодействия статическому анализу основаны на сокрытии основного кода программы. Среди методов противодействия статическому анализу можно выделить (согласно [6]):

— шифрование тела приложения (ключевых блоков) — при запуске приложение расшифровывается и производится запуск основной программы;

— запутывание кода (упаковывание) — перемешивание кода, создание большого количества дополнительных ссылок;

— алгоритмическая защита — внедрение в код программы (на этапе ее разработки) специализированной логики, нацеленной на защиту приложения от анализа; примером может быть маскировка передачи управления (использование косвенной передачи управления).

Анализ эффективности применения в мобильной ОС.

Метод защиты: шифрование тела приложения.

Уровень эффективности: высокий.

Положительные стороны: высокая эффективность метода (эффективность зависит от применяемого алгоритма шифрования).

Отрицательные стороны: зависимость области применения от производительности процессора, подверженность атакам на криптографический алгоритм (например, атака на ключевую систему).

Метод защиты: запутывание кода.

Уровень эффективности: средний.

Положительные стороны: простота реализации, наличие универсальных средств (упаковщики).

Отрицательные стороны: части запутанного кода все равно связны и могут быть проанализированы.

Не стоит недооценивать и другие методы защиты. Например, при грамотном применении алгоритмической защиты можно достичь весьма неплохих результатов.

Противодействовать методу активного анализа (отладке) приложения теоретически также не так сложно. Согласно [6] выделяют следующие методы.

Контроль времени. Суть метода заключается в измерении времени выполнения ключевых блоков кода. Так как при отладке (анализе) приложения выполнение команд происходит по команде человека ("пошаговый" режим отладки), время выполнения таких команд значительно больше. Время выполнения команд может быть измерено как запросом системного времени, так и по количеству тактов, выполненных процессором.

Контроль контрольных сумм. В данном случае программа подсчитывает контрольные суммы ключевых блоков (например, при старте приложения). Так как при анализе приложения отладчики нередко устанавливают так называемые контрольные точки (точки останова) в коде приложения, контрольные суммы ключевых блоков приложения будут изменены.

Алгоритмическая защита. В случае защиты от активного анализа приложения также имеется много различных специальных приемов программирования. Примером будет служить применение в приложении прерывания (int 01h и int 03h). Вышеуказанные прерывания используются программами-отладчиками для реализации "пошагового" режима выполнения программы. Если в защищаемом приложении намеренно использовать данные прерывания, то это создаст проблемы злоумышленнику.

Как уже упоминалось эмуляция (как метод анализа программ) обычно эффективнее применять совместно другими методами анализа. Как для защиты от анализа активного приложения, так и для обнаружения факта эмуляции (подделки окружения) применяют метод контроля внешней среды. Для этого защищаемое приложение может контролировать, например, следующие параметры: количество оперативной памяти, контрольные суммы системных библиотек и другие параметры системы.

Анализ эффективности применения в мобильной ОС.

Метод защиты: контроль времени.

Уровень эффективности: низкий.

Положительные стороны: –.

Отрицательные стороны: данные замера времени выполнения зависят от внешней среды и могут быть сфальсифицированы.

Метод защиты: контроль контрольных сумм.

Уровень эффективности: средний.

Положительные стороны: –.

Отрицательные стороны: для эффективного применения средства подсчета контрольных сумм должны быть защищены.

Метод защиты: алгоритмическая защита.

Уровень эффективности: низкий.

Положительные стороны: простота применения.

Отрицательные стороны: метод прямо не препятствует анализу, а лишь мешает работе программ отладчиков (замедляет работу отладчиков).

4.5. Способ защиты программного обеспечения от анализа

В данном разделе предложены эффективные методы защиты программного обеспечения от анализа. В разработанной операционной системе реализованы функциональные возможности защиты программного обеспечения.

В предыдущем разделе описаны классические алгоритмы защиты прикладного программного обеспечения от отладки (статического анализа, отладки, эмуляции).

Методы защиты приложения от статической отладки могут быть дополнительно усилены посредством применения отрицаемого шифрования.

Наиболее часто используемой в приложении структурой является «условие». "Условие" используется как самостоятельно, так и в более сложных структурах, таких как «цикл».

Для защиты от статического анализа (дизассемблирования) целесообразно в ключевых блоках программы вместо структуры «условие» использовать отрицаемое шифрование. Вместо ключа использовать входные данные «условия».

На выходе необходимо реализовать ложные ветви кода (например, в зашифрованном сообщении может содержаться адрес следующего блока программы). Такая структура (далее блок шифрования) будет применена и в других методах защиты от отладки, которые будут описаны далее. На вход блока шифрования будет подаваться параметр (ключ). На выходе блока будет получен адрес следующей команды (открытый текст). На рисунке 32 изображена блок-схема функции, которая в качестве конструкции типа «условие» использует блок шифрования. Как можно видеть результата выполнения «если» можно реализовать любое количество ветвей кода истинных или ложных. При этом для злоумышленника последующее выполнение каждой из ветвей кода будет равновероятно.

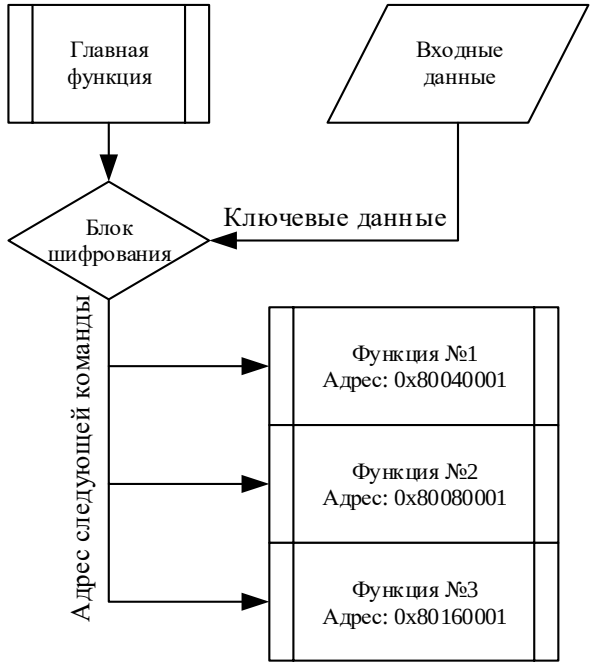


Рисунок 32. Использование отрицаемого шифрования в качестве конструкции типа «если»

Применение отрицаемого шифрования в качестве конструкции типа «условие» позволит значительно усложнить (особенно при многократном применении) статический анализ приложения (дизассемблирование).

При защите от статического анализа также применяется и шифрование тела или наиболее критичных блоков программы. Для шифрования тела программы или критических блоков приложения возможно применение отрицаемого шифрования. В данном случае в качестве ключа может быть использовать целый ряд внешних параметров (контроль внешней среды) таких как состояние операционной системы. На выходе так же, как и в предыдущем случае, необходимо реализовать ложные ветви кода.

Для защиты программы от активного анализа средствами отладки также может применяться отрицаемое шифрование.

Метод защиты от активной отладки "контроль времени" дополнительно усложняется введением в алгоритм метода отрицаемого шифрования.

Первая стратегия применения отрицаемого шифрования для контроля времени - контроль времени выполнения.

На входе блока шифрования: разница во времени (например, может быть передано количество тактов процессора, которое прошло за период времени).

На выходе блока шифрования: адрес следующего блока основной программы (либо ложной ветви алгоритма).

Вторая стратегия применения отрицаемого шифрования для контроля времени - контроль момента времени выполнения.

На входе блока шифрования: настоящее время.

На выходе блока шифрования: адрес следующего блока основной программы (либо ложной ветви алгоритма).

Такая стратегия может также быть применена также в пробных версиях программных продуктов (например, для работы программы в течении конкретного месяца).

Стратегий применения отрицаемого шифрования для контроля внешней среды можно разработать достаточно много (стратегия зависит от входных данных - количества оперативной памяти, наличие библиотек и т.д.). Перспективной выглядит следующая стратегия, которая предназначена для

контроля адреса flash-памяти микроконтроллера, откуда запущен экземпляр приложения.

На входе блока шифрования: адрес первой команды тестируемого блока (в микроконтроллерах архитектуры ARM - значение счетчика команд).

На выходе блока шифрования: адрес следующего блока основной программы (либо ложной ветви алгоритма).

Используя данный метод, экземпляр программы может определить, что он запущен вне защищенного блока памяти, и активизировать защитные действия.

Применение шифрования тела программы может быть использовано и для защиты от активной отладки приложения.

На входе блока шифрования: некоторое условие (в зависимости от используемой стратегии) в качестве ключа, блок зашифрованного кода.

На выходе блока шифрования: блок расшифрованного кода программы (либо ложный блок программы).

При внедрении системы защиты от отладки следует принимать что:

программный продукт в любой своей части может быть подвержен анализу;

внешняя среда (оперативная память, внешние библиотеки, средства операционной системы) постоянно находится под наблюдением;

злоумышленник может иметь большие возможности и обладать всеми необходимыми знаниями для анализа приложения.

В каждый момент времени на входе блока шифрования может быть намеренно направлено (злоумышленником в ходе анализа приложения) определенное значение. Использование для таких целей программ отладчиков может рассматриваться как "принуждение" программы к выполнению неких действий. В таком случае применение отрицаемого шифрования поможет более эффективно применять классические методы защиты приложения от отладки.

Анализ эффективности применения в мобильной ОС.

Метод защиты: Способы применения отрицаемого шифрования в методах защиты приложений от анализа.

Отличительные особенности: повышенная защищенность метода защиты от анализа за счет противодействия прямого анализа кода функции защиты (результат не предсказуем для злоумышленника).

4.6. Подсистема резервирования данных

Обеспечение доступности оперативных данных операционной системы является важной проблемой разработчиков операционной системы. Организация резервирования в различных системах рассмотрены в работах [102], [103]. В связи с особенными условиями эксплуатации мобильные операционные системы должны включать в себя различные дополнительные механизмы резервирования данных. В процессе работы операционной системы в любой момент может произойти «нештатная ситуация». В операционной системе предусмотрена защита от:

1. внезапное отключение питания аппаратной платформы выполнения (например, вследствии действий пользователя);
2. прерывание процесса передачи данных (потеря или искажение пакетов);
3. извлечение отчуждаемого носителя информации (например, карты памяти MMC);
4. искажение файловой системы вследствие воздействия статического электричества (флеш-память подвержена стиранию от воздействия статического электричества).

Стоит сказать, что поскольку резервирование представляет собой введение дополнительной избыточности, а мобильная операционная система имеет ограниченные ресурсы, целесообразно производить резервирование только тех данных, которые действительно имеют критически большое влияние на операционную систему.

Система резервирования данных постоянной памяти разработанной операционной системы реализована на базе виртуальной файловой системы. Для

хранения особо критичных файлов используется специальное файловое пространство, которое физически располагается в энергонезависимой памяти.

Защита данных оперативной памяти (особенно оперативных данных ядра ОС) осуществляется применением механизмов защиты аппаратной платформы (энергонезависимая память).

Анализ эффективности применения в мобильной ОС

Метод защиты: подсистема резервирования данных.

Отличительные особенности: интеграция с аппаратными механизмами защиты.

4.7. Доверенная загрузка операционной системы

В связи с возросшими возможностями современной электроники большое количество внедряемых систем, бытовой техники, сетевого оборудования, а также средств защиты информации и другой продукции, которая имеет в своей основе вычислительные операции, разработано на базе операционных систем. Большинство производителей применяют операционные системы в своих продуктах в связи с тем, что это позволяет не тратить время разработчиков на написание драйверов для унифицированной периферии. В операционных системах применяются различные методы защиты данных пользователей, а также авторских прав производителей на программные решения. Применение унифицированного аппаратного обеспечения совместно с операционными системами, которые используют средства защиты, порождают еще одну сложную задачу - доверенную загрузку операционной системы. Многие производители операционных систем уделяют огромное внимание защите операционной системы в процессе ее загрузки. Примером может служить разработка корпорации Microsoft**,** описанная в работе [104]. Загрузчик операционной системы Microsoft Windows обеспечивает проверку подлинности ядра операционной системы и корректность аппаратной платформы. В работах [105], [106] также глубоко рассмотрен процесс доверенной загрузки операционной системы.

Загрузка операционной системы представляет собой процесс выхода программных модулей операционной системы в штатный режим работы после включения. В процесс загрузки в зависимости от архитектуры ОС может включаться:

– передача управления ядру операционной системы;
– выполнение процедур инициализации ядра операционной системы;
– запуск сервисов операционной системы;
– запуск прикладного программного обеспечения.

Передача управления операционной системе является первым и неотъемлемым этапом загрузки любой ОС. Рассмотрим процесс загрузки на примере ядра CORTEX-M3. Согласно [94] память ядра CORTEX-M3 состоит из нескольких функциональных блоков памяти. К таким блокам относятся оперативная память, постоянное запоминающее устройство (флеш-память). Процедуры операционной системы расположены в флеш-памяти контроллера (пространство памяти 0x8000000-0x0801FFFF). Принцип работы ядра CORTEX-M3 состоит в том, что ядро контроллера выполняет команду, расположенную в памяти по адресу, который задан в специализированном регистре ядра контроллера (счетчике команд). Выполнение команды может происходить практически из любого разрешенного места памяти (например, из оперативной памяти).

Процесс передачи управления операционной системе заключается в записи адреса первой процедуры ядра ОС в регистр счетчика команд микроконтроллера. Защищенной операционной системе необходимо иметь информацию о месте ее запуска. Также важна информация о том, является ли операционная система единственным (резидентным) пользователем аппаратных ресурсов. Подсистемы защиты операционных систем сами представляют собой обычное приложение (процедуры). Так же, как и другие приложения, подсистемы защиты могут быть подвержены отладке. Для защиты оперативных данных процессов нередко используется аппаратное разграничение доступа к памяти. Например, в

контроллерах архитектуры ARM фирмы STMicroelectronics имеют специализированный модуль управления памятью (MPU – Memory Protect Unit, модуль защиты памяти). Но даже данный модуль не поможет, если контроллер находится в режиме отладки. Для минимизации вероятности вышеописанных угроз ядро операционной системы и должно иметь полное управление над аппаратной платформой. Некоторые аппаратные платформы (например, контроллеры ARM фирмы Atmel) предоставляют возможности контроля загрузки с использованием загрузчиков. Загрузчик представляет собой программу, предназначенную для безопасной передачи управления операционной системы (рисунок 33).

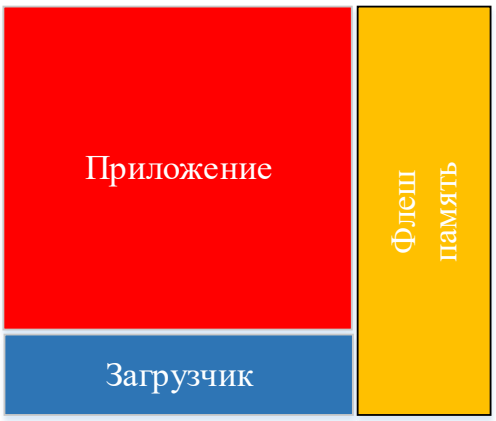


Рисунок 33. Расположение загрузчика в памяти.

Загрузчик расположен в пространстве стартовой загрузки процессора (обычно этот адрес равен 0). Дополнительно этот загрузчик защищен аппаратно. Перезаписать загрузчик можно только совместно со стиранием основной памяти.

Второй важной проблемой загрузки операционной системы является вопрос аутентичности модулей операционной системы. При использовании унифицированных контроллеров не всегда имеется возможность ограничить применение средств отладки в процессе выполнения операционной системы. Теоретически любой модуль операционной системы может быть изменен злоумышленником. Для защиты важных частей операционной системы

применяют средства контроля целостности и электронной подписи, которые также могут быть интегрированы в программу-загрузчик.

Анализ эффективности применения в мобильной ОС.

Метод защиты: контроль целостности и электронной подписи модулей в процессе загрузки ОС.

Уровень эффективности: высокий.

Положительные стороны: высокая эффективность методов электронной подписи.

Отрицательные стороны: проблема аутентичности ключа проверки подписи (ключ также хранится на устройстве и может быть подменен злоумышленником).

В разработанной операционной системе реализована подсистема доверенной загрузки. В операционной системе имеется специализированный модуль загрузки. Загрузчик решает целый ряд проблем, связанных возможностью загрузки неаутентичного программного продукта в системе:

– проверка аутентичности важнейших модулей операционной системы до ее загрузки;

– расшифрование ядра операционной системы;

– безопасное обновление ядра операционной системы;

– самоконтроль аппаратных средств.

На рисунке 34 изображен алгоритм работы загрузчика разработанной операционной системы.

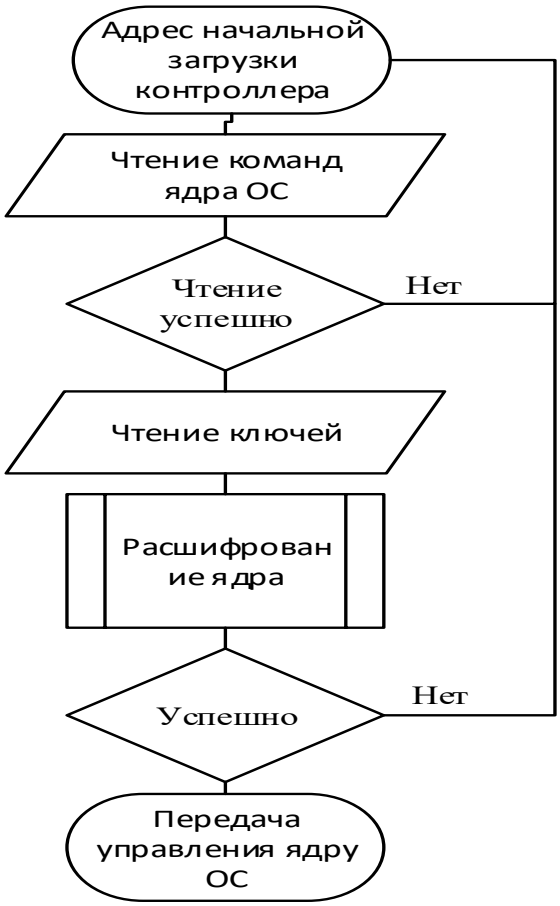


Рисунок 34.Алгоритм работы загрузчика

Для защиты самого важного модуля операционной системы (ядра) применяется шифрование тела программы. Перед началом загрузки ядра операционной системы загрузчик производит расшифрование тела ядра операционной системы. В качестве ключа используются следующие параметры:

– уникальный идентификатор контроллера (имеется у наиболее популярных производителей контроллеров архитектуры ARM, например, фирм STMElectronics, Atmel);

– хеш наиважнейших областей постоянной памяти;

– персональный ключ устройства.

Используя вышеуказанные данные, выполняется формирования ключа шифрования. Отсутствие явного хранения ключа в памяти контроллера позволяет исключить атаки, основанные на анализе постоянной памяти контроллера.

Анализ эффективности применения в мобильной ОС

Метод защиты: применение ресурсов аппаратной платформы при формировании ключа шифрования.

Отличительные особенности: отсутствует необходимость хранения результирующего ключа шифрования, привязанность ключа шифрования к конкретной аппаратной платформе.

4.8. Безопасное обновление операционной системы

Операционная система является крайне сложным программным продуктом и содержит в своем составе большое количество процедур. И, как все созданное человеком, может содержать ошибки. Поэтому производители операционных систем постоянно совершенствуют свои продукты. В работах [107], [108] рассматриваются основные принципы построения загрузчиков, обеспечивающих обновление модулей программного обеспечения. Помимо обеспечения замены программных модулей дополнительно необходимо обеспечить безопасность процесса обновления операционной системы.

На рисунке 35 (оригинальное изображение из документа [109]) изображен алгоритм работы безопасного загрузчика фирмы Atmel для контроллеров архитектуры ARM собственного производства.

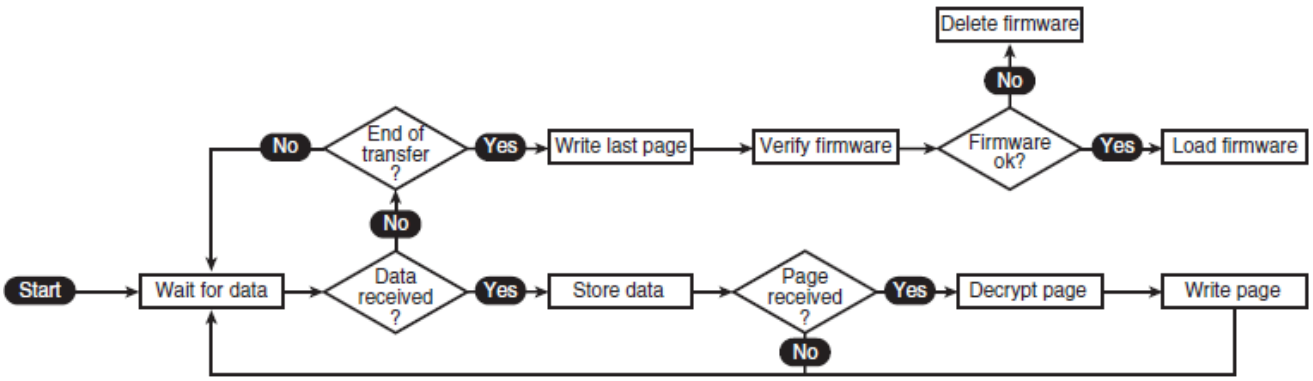


Рисунок 35. Алгоритм работы безопасного загрузчика фирмы Atmel

В данном загрузчике предусмотрены различные средства передачи данных (USB, USART). Процесс передачи образа операционной системы контролируется по блокам. При получении очередного блока данных производится проверка корректности передачи. После передачи последнего блока данных производится

проверка полученного образа. На последнем этапе производится обновление операционной системы из образа.

Анализ эффективности применения в мобильной ОС.

Метод защиты: Алгоритм безопасного обновления.

Уровень эффективности: средний.

Достоинства: защита обновления при передаче по сети.

Недостатки: необходимость перезагрузки для обновления программных модулей, отсутствует проверка подлинности обновлений.

Наличие подсистемы обновления является важной конкурентной характеристикой операционной системы. Данная подсистема не обязательно должна иметь возможность «обновления через интернет». Основной ее задачей является обеспечение возможности безопасного обновления программного обеспечения. Данная функция может быть реализована и простейшим путем «перепрошивки» микроконтроллера. Разработанная операционная система имеет в своем составе средства безопасного обеспечения программного обеспечения. Подсистема обновления включает в себя:

– безопасный загрузчик операционной системы с функцией обновления (для обновления ядра операционной системы);

– прикладной сервис обновления (для обновления модулей операционной системы и пользовательских приложений).

В загрузчик операционной системы, который был описан в предыдущем разделе, встроены функции безопасного обновления. Для осуществления обновления ядра операционной системы загрузчику необходимо перейти из стандартного режима «загрузки» в режим обновления. В данном режиме загрузчик не будет выполнять свою прямую задачу – загрузку операционной системы. В данном режиме загрузчик принимает на себя управление аппаратными средствами контроллера, а конкретно инициирует подключение к интерфейсу USB. С точки зрения компьютера подключение устройства никак не отличается от стандартного режима работы устройства. Используя интерфейс USB, загрузчику

передается пакет обновления. Далее производится проверка подлинности пакета обновления и обновление. Процесс проверки передачи пакета в контроллер построен по классической схеме, которая была описана в третьей главе. Более важным является процесс проверки подлинности пакета обновления.

Блок-схема алгоритма проверки подлинности пакета обновления изображена на рисунке 36.

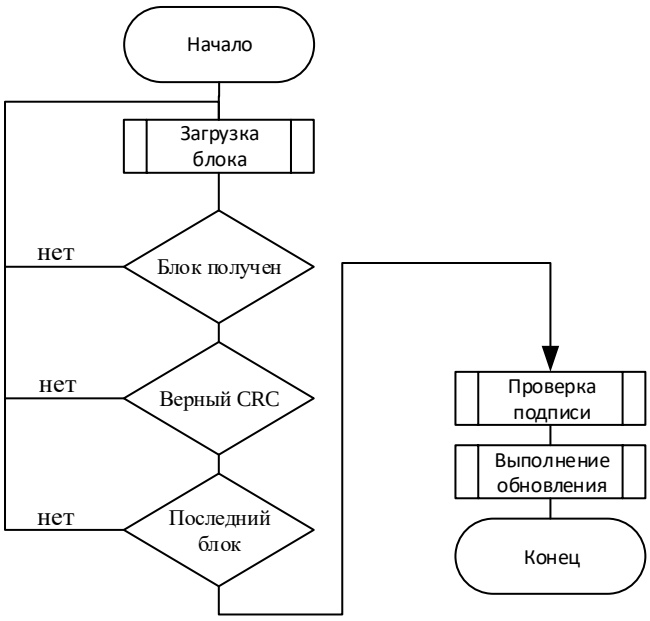


Рисунок 36. Алгоритма проверки подлинности пакета обновления

Первоначально стоит сказать, что в операционной системе предлагается применять три группы обновлений:

– персональные обновления (обновления, которые необходимо установить на определенное устройство);

– групповые обновления (обновления, которые предназначены для целого класса устройств);

– общие обновления (обновления, предназначенные для всех устройств, работающих на основе разработанной ОС).

Кроме вышеуказанного деления, обновления также имеют различный уровень критичности:

– критические обновления (обновления, которые исправляют фатальные ошибки в работе операционной системы);

– обновления безопасности (обновления, влияющие на процесс обеспечения безопасности операционной системы);

– обновления прикладного программного обеспечения.

Анализ эффективности применения в мобильной ОС

Метод защиты: безопасное обновление ПО.

Отличительные особенности: проверка подлинности обновлений, категоризация обновлений.

4.9. Выводы к четвертой главе

В данном разделе будет акцентировано внимание на результатах проделанной работы. Технические характеристики разработанной операционной системы во многом обусловлены двумя факторами:

– средой применения операционной системы - мобильные системы;

– требованиями информационной безопасности.

В разработанной операционной системе подсистема информационной безопасности глубоко интегрирована в системные службы. ОС обеспечивает эшелонированную систему защиты от наиболее актуальных угроз безопасности, которые были описаны во второй главе.

Вся система защиты операционной системы построена в ключе, предложенном в [110]. Два основных подхода**,** описывающих данную концепцию построения подсистем защиты это:

– обеспечение безопасности взаимодействия процессов является прямой задачей операционной системы (операционная система обязана обеспечить безопасность взаимодействия своих "клиентов");

– для обеспечения требований безопасности операционная система может требовать от процесса выполнения некоторых требований.

Далее будет обобщена общая структура разработанных подсистем защиты мобильной операционной системы.

Подсистема аутентификации. Количественная оценка стойкости стандартной парольной системы аутентификации пользователей выполняется согласно следующей формуле.

$$P = \frac{VT}{A^L}$$

где

A – мощность алфавита символов, из которых состоит пароль;

L – длина пароля;

V – скорость перебора паролей злоумышленником;

T – срок действия паролей;

P – вероятность подбора паролей злоумышленником за время t, меньшее срока действия паролей.

Подсистема аутентификации операционной системы поддерживает гибкую настройку парольной политики. Пароль пользователя может включать в себя символы латинского алфавита, цифры, специальные символы. Предусмотрена защита от перебора (стандартно: блокировка доступа на 5 минут после 3 неправильного ввода пароля). Для примера, пароль в 6 символов (символы латинского алфавита, цифры, специальные символы) и сроком действия в 1 год имеет вероятность подбора пароля в течение срока действия пароля в худшем случае составляет примерно $1{,}2 * 10^{-8}$.

Для безопасного удаленного управления операционной системой предусмотрена возможность использования аутентификации с использованием одноразовых паролей. Карточка паролей предусматривает использования цифровых паролей размером в 8 символов. В данном случае пространство паролей значительно меньше – порядка 10^8. Но так как сокращается время использования одного пароля, значение вероятности подбора пароля в течение срока действия пароля имеет соизмеримые значения. Что позволяет также эффективно применять данный метод аутентификации.

Дополнительно подсистема аутентификации ОС предоставляет возможность защиты пользователей от вынуждающих атак. При необходимости

может быть сгенерирован дополнительный набор альтернативной парольной информации. В работе была предложена схема организации подсистемы аутентификации локальных пользователей на основе отрицаемого шифрования.

Криптографическая подсистема операционной системы имеет развитую структуру ключевой инфраструктуры. Данный подход позволяет в некоторых случаях экономить ресурсы системы путем применения менее стойких алгоритмов шифрования без ущерба ее безопасности [111]. Подсистема реализует как стандартные криптографические алгоритмы, так и дополнительно разработанные.

В ходе исследования был предложен эффективный алгоритм отрицаемого шифрования на основе блочных шифров[81]. Скорость простого алгоритма отрицаемого шифрования с использованием блочного шифра (см. раздел Алгоритм отрицаемого шифрования).

$$\lambda_{\text{ОШ}} \approx (2^{-2u-1}u/n)\lambda_{\text{БШ}}$$

где

$\lambda'_{\text{ОШ}}$ – расчетная скорость алгоритма отрицаемого шифрования;

$\lambda_{\text{БШ}}$– скорость базового алгоритма шифрования;

u – количество битов данных;

n – количество битов случайной последовательности.

Предложенный алгоритм обеспечивает существенный прирост скорости шифрования

$$\lambda'_{\text{ОШ}} \approx (2^{-u-1}u/n)\lambda_{\text{БШ}}$$

где

$\lambda'_{\text{ОШ}}$ – расчетная скорость алгоритма отрицаемого шифрования;

$\lambda_{\text{БШ}}$– скорость базового алгоритма шифрования;

u – количество битов данных;

n – количество битов случайной последовательности.

Было достигнуто увеличение производительности ОШ в производительности в 2^u раз. Данный факт позволяет использовать отрицаемое

шифрование в разработанной мобильной операционной системе. Отрицаемое шифрование применяется в операционной системе:

– в подсистеме аутентификации;
– в подсистеме защиты от отладки;
– в криптографической подсистеме (как функция).

Экономия ресурсов системы. В связи с ограниченным количеством ресурсов было принято решения применять в операционной системе схемы поочередного выполнения заданий специализированным сервисом. Данная схема доступна для использования в прикладных программных продуктах с низкоприоритетными задачами. Схема применяется в следующих подсистемах:

– сервис контроля доступа к ФС;
– систему защита передаваемых данных;
– криптографическая подсистема.

Данное решение позволило сэкономить оперативную память мобильной операционной системы. Что позволит запускать большее количество приложений. К примеру, реализация функций доступа к файловой системе в режиме API функций для одного процесса требует выделения оперативной памяти в размере не менее 2 кбайт. Учитывая размер оперативной памяти целевых платформ примерно 20-40 кбайт, такое расточительство недопустимо в мобильных системах. Схема применения специализированных служб выполнения заданий позволяет не выделять память для реализации функций доступа к файловой системы каждому процессу.

Защита программного обеспечения от отладки. В ходе исследования были разработаны методы защиты прикладного программного обеспечения от отладки с использованием алгоритмов отрицаемого шифрования [6]. В данных методах ОШ применяется как дополнительная преграда для использования алгоритмов анализа программного обеспечения. В работе предложены методы защиты от:

– статического анализа машинного кода (дизассемблирования);

– активного анализа приложения с использованием отладчиков путем введения ложных ветвей кода;

– использования элементов анализа внешней среды.

Виртуальная среда выполнения. Очень большой угрозой операционной системе является прикладное программное обеспечение. Прикладные программы приносят в системы ошибки программистов, исключительные ситуации и даже специальные программные закладки. Для защиты от данного вида угроз была выбрана наиболее безопасная схема контроля программного обеспечения – виртуальная среда выполнения приложений. На базе операционной системы реализован компилятор программного обеспечения, который приводит программный код в удобный вид для виртуальной среды выполнения. Применение штатного компилятора позволит в дальнейшем внедрить в систему алгоритмы анализа программного кода на наличие недекларированных возможностей. Для написания программного обеспечения предлагается использовать язык**,** подобный языку ASSEMBLER. В виртуальной среде выполнения реализованы основные функции языка ASSEMBLER. Для дополнительной защиты в связке с виртуальной средой выполнения работает подсистема контроля памяти, которая позволяет надежно изолировать память процессов.

Резервирование ресурсов. Мобильная операционная система отличается наличием естественных угроз функционирования, которые связаны с особенностями эксплуатации операционной системы (например, возможностью внезапного отключения). Защита программного обеспечения от возможных последствий эксплуатации выполняется целым рядом решений:

– система контроля памяти обеспечивает резервирование оперативной памятью важных процессов в энергонезависимой памяти;

– наличие различных режимов работы ядра операционной системы;

– журналирования файловых операций для возможности дальнейшего восстановления.

Трехкратное резервирование данных прикладных приложений позволяет эксплуатирования разработанную ОС в мобильных системах.

Комплексное использование подсистем защиты разного типа позволяет значительно увеличить общую защищенность системы. Далее представлена таблица 7 соответствие класса угроз ИБ мобильной ОС применяемым средствам защиты.

В данной главе решены следующие задачи исследования:

– разработка методов защиты программного обеспечения от дизассемблирования.

Таблица 7. Таблица соответствие класса угроз ИБ мобильной ОС применяемым средствам защиты

№ п/п	Подсистема ОС	Применяемые средства защиты				
		Класс №1[1]	**Класс №2**[2]	**Класс №3**[3]	**Класс №4**[4]	**Класс №5**[5]
1	Подсистема аутентификации	+			+	
2	Сервис контроля доступа к ФС	+	+	+	+	
3	Криптографическая подсистема	+	+	+		
4	Подсистема виртуальной программной среды	+	+	+	+	+
5	Защищенная файловая система	+	+	+		
6	Подсистема защиты удаленного доступа к ОС(Аутентификация на одноразовых паролях)	+				
7	Защита передаваемых данных	+				
8	Подсистема управления памятью (изоляция данных прикладных процессов)	+			+	+
9	Подсистема доверенной загрузка операционной системы	+			+	
10	Подсистема защиты от отладки		+	+		
11	Подсистема резервирования данных				+	
12	Подсистема безопасного обновления				+	+

[1]Неправомерный доступ к ресурсам
[2] Анализ (отладка) подсистем операционной системы
[3] Атаки на криптографическую подсистему
[4] Нарушение работоспособности операционной системы
[5] Не декларированные возможности программного обеспечения

Заключение

В результате выполненного диссертационного исследования решены важные научно-технические задачи разработки производительных методов и алгоритмов псевдовероятностного защитного преобразования, удовлетворяющих критерию вычислительной неразличимости от вероятностного защитного преобразования, и универсальной защищенной мобильной операционной системы, которая обеспечивает переносимость программных средств защиты информации на различные типы мобильных устройств (технических платформ) и обеспечивает защиту пользователей системы и информации от атак с принуждением за счет применения псевдовероятностного защитного преобразования. Решение данных задач позволило достигнуть заявленного сокращения сроков и уменьшения затрат по разработке мобильных защищенных информационных технологий.

Основные результаты диссертационного исследования:

1. разработан метод аутентификации пользователей с использованием одноразовых паролей, генерируемых с помощью алгебраического алгоритма псевдовероятностного защитного преобразования, который обеспечивает защиту от принуждающих атак;

2. разработан метод псевдовероятностного защитного преобразования информации, обеспечивающий защиту информации от несанкционированного доступа в случае атак с принуждением;

3. разработан метод защиты программного обеспечения от дизассемблирования, основанный на введении ложных веток кода с помощью псевдовероятностного защитного преобразования кода;

4. разработан метод противодействия активной отладке программного обеспечения с использованием псевдовероятностного защитного преобразования для введения ложных веток кода;

5. разработан метод хранения ключей шифрования, основанный на применении псевдовероятностного защитного преобразования для обеспечения возможности сокрытия наличия резервных серий ключей;

6. разработан протокол аутентификации с использованием одноразовых паролей на основе алгебраического алгоритма псевдовероятностного защитного преобразования. Протокол обеспечивает дополнительную защиту удаленных пользователей от принуждающей атаки;

7. предложена схема подсистемы аутентификации локальных пользователей операционной системы, в которой предусмотрена защита от принуждающей атаки. В основе данного алгоритма лежит алгоритм псевдовероятностного защитного преобразования.

В процессе работы проводилась разработка механизмов применения отрицаемого шифрования в механизмах защиты информации операционных систем.

Новизна работы состоит в том, что применение универсальной операционной системы в мобильных устройствах телекоммуникационных и информационных систем, в том числе в системах защиты информации, позволит унифицировать подходы к обеспечению безопасности при разработке таких систем. Данный подход значительно сократит расходы на разработку и производство мобильных устройств на схожих аппаратных платформах. Область применения ОС: аутентифицирующие устройства (токены, идентификаторы), системы охраны, устройства защиты программного обеспечения, персональные устройства хранения данных (защищенные файловые хранилища), аппаратные средства шифрования (криптопровайдеры).

По теме диссертационной работы были выполнены 3 публикации в журналах, входящих в перечень ВАК:

1. **Молдовян Н. А., Биричевский А. Р., Мондикова Я.А**. Отрицаемое шифрование на основе блочных шифров // Информационно управляющие системы. № 5. 2014. С. 80-86.

2. **Биричевский А.Р., Молдовян Н.А., Березин А.Н., Рыжков А.В.,** Способ отрицаемого шифрования. // Вопросы защиты информации: Науч.-практ. журн. Москва.:ФГУП "ВИМИ", 2013. Вып. 2 (101). С. 18-21.

3. **Биричевский А.Р.** Универсальная мобильная операционная система с подсистемами аутентификации и защиты информации на основе псевдовероятностного преобразования // Труды СПИИРАН. СПб.: Наука, 2016. №3. С.128-138. [112].

Основные результаты диссертационной работы докладывались на научных конференциях:

1. **Биричевский А.Р.** Подход к обеспечению безопасности взаимодействия процессов при разработке операционных систем. Информационная безопасность регионов России (ИБРР-2013). И 74 VIII Санкт-Петербургская межрегиональная конференция. Санкт-Петербург, 23-25 октября 2013 г.: Материалы конференции / СПОИСУ. – СПб., 2013. С. 49-50

2. **Березин А.Н., Биричевский А.Р., Молдовян Н.А.** Особенности задачи дискретного логарифмирования по составному модулю как криптографического примитива // Труды VII Санкт-Петербургской межрегиональной конференции «Информационная безопасность регионов России (ИБРР-2011)». Санкт-Петербург, 26-28 октября. СПб.: СПОИСУ, 2012. С. 104-108.

3. **Биричевский А.Р., Мирин А.Ю., Молдовян Н.А.** Нетрадиционные приложения блочных шифров // Инновационная деятельность в Вооруженных силах Российской Федерации: Труды всеармейской научно-практической конференции. 29-30 ноября 2012, г. Санкт-Петербург / СПб.: ВАС, 2011. С. 72-76.

4. **Биричевский А.Р., Молдовян Н.А., Рыжков А.В.** Отрицаемое шифрование как механизм защиты информации, хранимой на удаленных носителях // Инновационная деятельность в Вооруженных силах Российской Федерации: Труды всеармейской научно-практической конференции. 29-30 ноября 2012, г. Санкт-Петербург / СПб.: ВАС, 2011. С. 77-81.

5. **Березин А.Н., Биричевский А.Р., Мондикова Я.А.** Отрицаемое шифрование для защиты информации от НСД // 5-я научно-практическая конференция "Информационная безопасность. Невский диалог". Санкт-Петербург, 12-13 ноября 2013 г. / СПб.: «Студия «НП-Принт». С. 21-22.

6. **Биричевский А.Р.** Отрицаемое шифрование как механизм защиты приложений от отладки // Инновационная деятельность в Вооруженных силах Российской Федерации: Труды всеармейской научно-практической конференции. 21-22 ноября 2013, г. Санкт-Петербург / СПб.: ВАС, 2013. С. 81-85.

7. **Биричевский А.Р.** Практическое применение алгоритмов отрицаемого шифрования // Информационная безопасность и защита персональных данных: Проблемы и пути их решения [Текст]+[Электронный ресурс]: материалы VI Межрегиональной научно-практической конференции / под ред. О.М. Голембиовской. – Брянск: БГТУ,2014. С. 17-18

8. **Биричевский А.Р.** Отрицаемое шифрование как механизм защиты приложений от отладки // Комплексная защита объектов информатизации и измерительные технологии: сб. науч. тр. Всероссийской науч.-практической конф. с международным участием. 16-18 июня 2014. - СПб.: Изд-воПолитех. ун-та, 2014 С. 8-12

9. **Биричевский А.Р.** Способ применения отрицаемого шифрования для хранения ключей // Информационная безопасность регионов России (ИБРР-2015). IX Санкт-Петербургская межрегиональная конференция. Санкт-Петербург, 28-30 октября 2015 г.: Материалы конференции / СПОИСУ. – СПб., 2015. С. 98-99

Список литературы

1. Столлингс, Вильямс. Операционные системы, 4-е издание. Москва: Издательский дом "Вильямс", 2014.
2. Шаньгин В.Ф. Защита в компьютерных системах и сетях. Москва: ДМК Пресс, 2012.
3. Щаньгин В.Ф. Информационная безопасность компьютерных систем и сетей: учебное пособие. М.: ИД "ФОРУМ"Ж ИНФРА-М, 2008.
4. Безбогов А.А., Яковлев А.В., Мартемьянов Ю.Ф. Безопасность операционных систем : учебное пособие. Москва: "Издательство Машиностроение-1", 2007.
5. Молдовян А.А., Молдовян Н.А., Советов Б.Я. Криптография. СПб.: Лань, 2000.
6. Лорин Г., Лейтел Х.М. Операционные системы. Москва: "Финансы и статистика", 1984.
7. Девянин П.Н. Модели безопасности компьютерных систем: Учеб. пособие для студ. высш. учеб. заведений. М.: Издательский центр "Академия", 2005.
8. Оладько А.Ю. Подсистема мониторинга и аудита информационной безопасности в операционной системе Linux // Известия Южного федерального университета. Технические науки, No. 12, 2012.
9. Качанов М.А. Анализ безопасности информационных потоков в операционных системах семейства GNU/Linux // Прикладная дискретная математика, No. 3, 2010.
10. Брыкова Е.И. Основные принципы построения безопасных операционных систем // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика , No. 2, 2013. С. 52-57.
11. Сумкин К.С., Байков И.В., Горелкин Д.О. Безопасность в операционных системах, модель безопасности операционных систем // Промышленные АСУ и контроллеры, No. 7, 2011. С. 59-60.

12. Морозова Е.В., Мондикова Я.А., Молдовян Н.А. Способы отрицаемого шифрования с разделяемым ключом // Информационно-управляющие системы, No. 6, 2013. С. 73-78.
13. Молдовян Н.А., Щербаков А.В. Протокол бесключевого отрицаемого шифрования // Вопросы защиты информации, No. 2, 2016. С. 9-14.
14. Березин А.Н., Рыжков А.В., Гурьянов Д.Ю. Понятие отрицаемого шифрования в дисциплине «криптографические методы защиты информации» // Современное образование: содержание, технологии, качество, 2012. С. 246-248.
15. Березин А.Н., Молдовян А.А., Молдовян Н.А. Потоковое отрицаемое шифрование, вычислительно неотличимое от вероятностного шифрования // Международная конференция по мягким вычислениям и измерениям. 2015. С. 95-97.
16. Молдовян Н.А., Баширов З.С., Солнышкин Ж.А. Протокол поточного отрицаемого шифрования с разделяемым ключом // Вопросы защиты информации, No. 3, 2005. С. 27-31.
17. Молдовян Н.А., Горячев А.А., Вайчикаускас М.А. Расширение криптосхемы рабина: алгоритм отрицаемого шифрования по открытому ключу // Вопросы защиты информации, No. 2, 2014.
18. Олифер В.Г., Олифер Н.А. Сетевые операционные системы. СПб.: Питер, 2002.
19. Семенцов С.Г., Аксенов А.В., Кобзарев А.Н. Общий подход к построению операционных систем реального времени в системах управления // Вопросы электромеханики. Труды ВНИИЭМ, 2004. С. 188-199.
20. Горбунов В.В. Требования к операционным системам реального времени // Журнал научных публикаций аспирантов и докторантов, No. 9, 2012. С. 78-79.
21. Ли И.В., Балса А.Р. Современные подходы к разработке операционных

систем для масштабируемых многоядерных систем // Информационные технологии и системы: управление, экономика, транспорт, право, No. 1, 2014. С. 6-14.

22. Таненбаум Э., Вудхалл А. Операционные системы. Разработка и реализация (+CD). Классика CS. 3-е изд. СПб.: Питер, 2007.
23. Хилл Б.М. 3 в 1 : операционная система Ubuntu Linux. Москва: Триумф, 2008.
24. Дубовцев А.В. Tango. Операционная система из будущего. Санкт-петербург: БХВ-Петербург.
25. Шахнов В.А., Мороз А.А., Михненко А.Е., Власов А.И. Операционная система реального времени - matrixrealtime как основа для построения экспериментальных систем обработки сигналов в реальном времени // Сборник научных докладов и тезисов 2-й международной конференции стран снг.:молодые ученые - науке, технологиям и профессиональному образованию для устойчивого развития. 2000. С. 100-103.
26. Титов В.С., Бобырь М.В., Милостная Н.А. Операционные системы реального времени для систем ЧПУ // Промышленные АСУ и контроллеры, No. 7, 2008. С. 31-33.
27. Скорубский В.И., Хмылко Ф.В. Распределение прерываний по уровням в микропроцессорных системах // Научно-технический вестник информационных технологий, механики и оптики, No. 19, 2005. С. 186-190.
28. Андреева О.Н. Создание процесса обработки в вычислительных системах реального времени // Вестник науки и образования северо-запада России, No. 1, 2015. С. 173-177.
29. Лав, Роберт. Разработка ядра Linux, 2-е издание. : Пер. с англ. М.: ООО "И.Д. Вильяме", 2006.
30. Гордеев А.В. Операционные системы, 2-е издание.: учебное пособие. СПб.: Питер, 2004.
31. CardOS V4.2B FIPS 2007. URL: http://www.t-systems-zert.de/pdf/

ein_01_zer_itsec_cc/st_04191_e.pdf (дата обращения: 01.11.2016).

32. Попов А.Ю. Статус реализации проекта «Универсальная электронная карта» Аутентификации и хранения ключевой информации 2014. URL: http:// www.uecard.ru/upload/iblock/e7e/e7ebcd07446fbcaa2f3e8c8a54cdae9a.pdf (дата обращения: 01.11.2016).
33. Java Card™ OS for NXP’s SmartMX™ family of secure microcontrollers URL: http://www.mifare.net/files/1013/3777/5220/ NXP_05_0048_JCOP_%20leaflet_9397750170260_v6.pdf (датаобращения: 01.11.2016).
34. MULTOS Developer's Guide 2016. URL: http://www.multos.com/uploads/ MDG.pdf (датаобращения: 01.11.2016).
35. Руководство программиста Операционная система "Магистра" 2012. URL: http://www.smart-park.ru/images/magistra_1_3.pdf (дата обращения: 01.11.2016).
36. ГОСТ Р ИСО/МЭК 7816-3-2013. Карты идентификационные. Карты на интегральных схемах. Часть 3. Карты с контактами. Электрический интерфейс и протоколы передачи. М.: ИПК Издательство стандартов., 2013.
37. ГОСТ Р ИСО/МЭК 7816-6-2013.Карты идентификационные. Карты на интегральных схемах. Часть 6. Межотраслевые элементы данных для обмена.. М.: ИПК Издательство стандартов., 2013.
38. ГОСТ Р ИСО/МЭК 7816-4-2013.Карты идентификационные. Карты на интегральных схемах. Часть 4. Организация, защита и команды для обмена. М.: ИПК Издательство стандартов., 2013.
39. ГОСТ Р ИСО/МЭК 7816-9-2011.Карты идентификационные. Карты на интегральных схемах. Часть 9. Команды для управления картами.. М.: ИПК Издательство стандартов., 2011.
40. ГОСТ Р ИСО/МЭК 7816-2-2010. Карты идентификационные. Карты на интегральных схемах. Часть 2. Карты с контактами. Размеры и расположение

контактов. М.: ИПК Издательство стандартов., 2010.

41. Федеральный закон от 27 июля 2010 г. N 210-ФЗ "Об организации предоставления государственных и муниципальных услуг".
42. Java Card 3 PlatformRuntime Environment Specification, Classic EditionVersion 3.0.4 [Электронныйресурс] // Официальныйсайт Oracle Corporation: [сайт]. URL: http://www.oracle.com/technetwork/java/javame/javacard/download/releasenotes-jsp-1440109.html (дата обращения: 01.11.2016).
43. ГОСТ Р 53114-2008. Защита информации. Обеспечение информационной безопасности в организации. Основные термины и определения. М.: ИПК Издательство стандартов, 2008.
44. Бокова О.И., Михайлов Д.М., Фроимсон М.И. Выработка и анализ требований к защищенной мобильной операционной системе // Вестник воронежского института МВД России, No. 4, 2013. С. 242-247.
45. Сычев В.М. Угрозы безопасности операционным системам // Вопросы защиты информации, No. 3, 2007. С. 13-15.
46. Черемушкин А.В. Криптографические протоколы: основные свойства и уязвимости // Прикладная дискретная математика., No. 2, 2009. С. 115-150.
47. Барабанова М.И., Кияев В.И. Информационные технологии: открытые системы, сети, безопасность в системах и сетях: Учебное пособие. СПб.: СПбГУЭФ, 2010.
48. ГОСТ Р 51275-99. Защита информации. Объект информатизации. Факторы, воздействующие на информацию. М.: ИПК Издательство стандартов, 1999.
49. Шаньгин В.Ф. Защита в компьютерных системах и сетях. Москва: ДМК Пресс, 2012.
50. Петров А.А. Компьютерная безопасность. Криптографические методы защиты. М.: ДМК, 2000.
51. Панасенко С.П. Алгоритмы шифрования. Специальный справочник. СПб.: БХВ-Петербург, 2009.

52. Молдовян Н.А. Каким быть новому стандарту шифрования? // "Компьютерра", No. 2, Jan 2000.
53. Руководящий документ. Защита от несанкционированного доступа к информации. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей // Официальный сайт федеральной службы по техническому и экспортному контролю. 1999. URL: http://fstec.ru/component/attachments/download/294. (дата обращения: 01.11.2016).
54. Архитектура РУТОКЕН [Электронный ресурс] // Официальный сайт компании "Актив": [сайт]. URL: http://dev.rutoken.ru/pages/viewpage.action?pageId=2228235 (дата обращения: 01.11.2016).
55. Аутентификации и хранения ключевой информации // Официальный сайт компании Aladdin. URL: http://www.aladdin-rd.ru/support/downloads/get?ID=27383 (дата обращения: 01.11.2016).
56. Программно-аппаратный комплекс защиты информации от НСД для ПЭВМ (PC) «Аккорд-АМДЗ». Описание применения // Официальный сайт ОКБ САПР. URL: http://www.accord.ru/docs/amdz/AMDZ_DOS_Application_guide.pdf (дата обращения: 01.11.2016).
57. Руководящий документ. Защита от несанкционированного доступа к информации. Термины и определения // Официальный сайт федеральной службы по техническому и экспортному контролю. 1992. URL: http://fstec.ru/component/attachments/download/298 (дата обращения: 01.11.2016).
58. Сидоркина И.Г., Канаев Р.В., Меркушев О.Ю. Классификация методов аутентификации человека // Вестник волжского университета им. В.Н. Татищева, No. 12, 2009.
59. Абденов А.Ж., Голяков С.А. Аутентификация методом динамического графического пароля // Сборник научных трудов Новосибирского государственного технического университета. Новосибирск. 2013. С. 78-83.

60. Капустин Ф.А., Долгова Т.Г. Двухэтапная аутентификация в интернет-сервисах // актуальные проблемы авиации и космонавтики, No. 9, 2013.
61. Бельфер Р.А., Богомолова Н.Е. Аутентификация в сетях передачи данных на базе виртуальных каналов // Фундаментальные проблемы радиоэлектронного приборостроения, No. 6, 2012. С. 34-37.
62. Злонов Т., Комарова Н. Строгая аутентификация при удаленной работе с корпоративными ресурсами // Защита информации. Инсайд, No. 5, 2009. С. 46-49.
63. Сабанов А.Г. Аутентификация в распределенных информационных системах // Защита информации. Инсайд, No. 4, 2008. С. 69-73.
64. Васильев А.С., Керш С.В. Анализ алгоритма аутентификации подключаемых модулей аутентификации // Россия молодая: передовые технологии – в промышленность!, No. 2, 2015.
65. Гроссе Э., Упадхаяй М. Многофакторная аутентификация // Открытые системы. СУБД, No. 2, 2013. С. 42-47.
66. Афанасьев А.А., Шелупанов А.А. Аутентификация. Теория и практика обеспечения безопасного доступа к информационным ресурсам : учебное пособие. М.: Издательство: горячая линия - телеком, 2009.
67. Царёв Е. Аутентификация сегодня и завтра // Защита информации. Инсайд , No. 4, 2014. С. 39-41.
68. Березин А.Н., Биричевский А.Р., Молдовян Н.А., Рыжков А.В. Способ отрицаемого шифрования // Вопросы защиты информации, No. 2, 2013. С. 18-21.
69. ГОСТ Р 34.11.-94. Информационная технология. Криптографическая защита информации. Функция хэширования.. М.: ИПК Издательство стандартов, 1994.
70. Козачок А.В., Голембиовская О.М., Туан Л.М. Прототип системы контролируемого разграничения доступа к файлам документальных

форматов // Вестник брянского государственного технического университета, No. 4, 2015.

71. Королев И.Д., Поддубный М.И., Носенко С.В. Применение сегмента матрицы доступов хру в анализе информационной безопасности систем, реализующих мандатное разграничение доступа // Политематический сетевой электронный научный журнал кубанского государственного аграрного университета, No. 101, 2014. С. 1811-1823.
72. Шапченко К.А. Способ проверки свойств безопасности в моделях логического разграничения доступа с древовидной иерархией объектов доступа // Информационные технологии, No. 10, 2009. С. 13-17.
73. Сизоненко А.Б. Арифметико-логическое представление матрицы доступа в дискреционной модели разграничения доступа // Вестник воронежского института МВД России, No. 3, 2012. С. 201-206.
74. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке СИ. М.: ТРИУМФ, 2002.
75. Молдовян Н.А., Молдовян А.А. Введение в криптосистемы с открытым ключом. СПб: «БХВ-Петербург», 2005.
76. Венбо М. Современная криптография. Теория и практика. М., СПб, Киев: Издательский дом «Вильямс».
77. Иванов М.А. Криптографические методы защиты информации в компьютерных системах и сетях. М.: Кудиц Образ, 2001.
78. ГОСТ Р 28.147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. М.: ИПК Издательство стандартов, 1989.
79. ГОСТ Р 34.10-94. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. М.: Издательство стандартов, 1994.
80. ГОСТ Р 34.10-2001. Информационная технология. Криптографическая

защита информации. Процессы формирования и проверки электронной цифровой подписи.. М.: ИПК Издательство стандартов, 2001.

81. Молдовян Н.А., Биричевский А.Р., Мондикова Я.А. Отрицаемое шифрование на основе блочных шифров // Информационно управляющие системы, No. 5, 2014. С. 80-86.
82. Молдовян А.А., Молдовян Н.А., Гуц Н.Д., Изотов Б.В. Криптография: скоростные шифры. СПб: БХВ-Петербург, 2002.
83. Биричевский А.Р. Практическое применение алгоритмов отрицаемого шифрования. Информационная безопасность и защита персональных данных: Проблемы и пути их решения // Материалы VI Межрегиональной научно-практической конференции. Брянск. 2014. С. 17-18.
84. Биричевский А.Р. Отрицаемое шифрование как механизм защиты приложений от отладки. // Комплексная защита объектов информатизации и измерительные технологии: сб. науч. тр. Всероссийской науч.-практической конф. с международным участием. 16-18 июня 2014. СПб. 2014. С. 8-12.
85. Биричевский А.Р. Способ применения отрицаемого шифрования для хранения ключей // Информационная безопасность регионов России (ИБРР-2015). IX Санкт-Петербургская межрегиональная конференция. Санкт-Петербург, 28-30 октября 2015 г. СПб. 2015. С. 98-99.
86. Молдовян Н.А. Практикум по криптосистемам с открытым ключом. СПб: «БХВ-Петербург», 2005.
87. Таныгин М.О. Методика обеспечения безопасного хранения данных на постоянных носителях // Известия юго-западного государственного университета., No. 2, 2013. С. 45-48.
88. Таныгин М.О., Глазков А.С. Принципы организации программно-аппаратной защиты файлов на жёстких дисках персональных компьютеров // Молодой ученый, No. 11, 2010. С. 104-106.
89. ГОСТ Р ИСО/МЭК 15408-1-2002. Информационная технология. Методы и

средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 1. Введение и общая модель.. М.: ИПК Издательство стандартов.

90. ГОСТ Р 50922-2006. Защита информации. Основные определения.. М.: ИПК Издательство стандартов, 2006.
91. ГОСТ 28906-91. Системы обработки информации. Взаимосвязь открытых систем. Базовая эталонная модель. М.: ИПК Издательство стандартов, 1991.
92. Зайцев А.П., Голубятников И.В., Мещеряков Р.В., Шелупанов А.А. рограммно-аппаратные средства обеспечения информационной безопасности: Учебное пособие. Издание 2-е испр. и доп. М.: Машиностроение-1, 2006.
93. AT91SAM ARM-based Flash MCU // Официальныйсайткомпании ATMEL. 2012. URL: http://www.atmel.com/ru/ru/Images/6430s.pdf (дата обращения: 01.11.2016).
94. Medium-density performance line ARM-based 32-bit MCU // Официальныйсайткомпании STMicroelectronics. URL: http://www.st.com/st-web-ui/static/active/en/resource /technical/document/datasheet/CD00161566.pdf (датаобращения: 01.11.2016).
95. 32-bit ARM Cortex-M0 microcontroller // Официальныйсайткомпании NXP Semiconductors. 2014. URL: http://www.ru.nxp.com/documents/data_sheet/ LPC111X.pdf (дата обращения: 01.11.2016).
96. Биричевский А.Р. Использование архитектуры «черного ящика» в аппаратных средствах криптографической защиты информации // Юбилейная XIII Санкт-Петербургская международная конференция региональная информатика «РИ-2012». Санкт-Петербург, 24-26 октября 2012. С. 19.
97. Максимов В.А. Архитектура системы управления виртуальными машинами // Электронное периодическое издание информационная среда образования и науки, No. 11, 2012. С. 30-34.

98. Хашковский В.В., Прокопенко А.В. О применении интерфейса прикладного программирования для управления виртуальными машинами // Материалы v международной научно-практической конференции. 21 век: фундаментальная наука и технологии. 2014. С. 128-129.
99. Петрищев Д.В. Проблемы управления виртуальной памятью в вычислительных машинах с архитектурой потока данных // Информационные технологии моделирования и управления, No. 9, 2006. С. 1179-1182.
100. Оуни Э.Д., Торнтон Э.Д., Ошинс Д. Управление состоянием распределенных аппаратных средств в виртуальных машинах, Патент на изобретение 2429530, 2011.
101. Траут Э. Системы и способы использования синтезированных команд в виртуальной машине, Патент на изобретение 2374675, 2009.
102. Иванов Д.Г. Организация резервирования в системах распределенного хранения данных // Весник национального техничного университету Украины "Киевский политехничний институт", No. 56, 2012. С. 160-164.
103. Черноусов Н.С., Зуев М.С. Системы резервирования данных // Психолого-педагогический журнал Гаудеамус, Vol. 2, No. 20, 2012. С. 161-162.
104. Уиллмэн Б.М., Инглэнд П., Д Р.К., Хантер Д., Макмайкл Л.Д., Ласалл Д.Н., Жакомет П., Пэли М.Э., Куриен Т.В., Кросс Д. Система и способ для защищенной начальной загрузки операционной системы с использованием проверки состояния, Патент на изобретение 2413295, 2011.
105. Гатчин Ю.А., Теплоухова О.А. Реализация контроля целостности образа операционной системы, загружаемого по сети на тонкий клиент // Научно-технический вестник информационных технологий, механики и оптики, No. 6, 2015. С. 1115-1121.
106. Авезова Я.Э., Фадин А.А. Вопросы обеспечения доверенной загрузки в физических и виртуальных средах // Вопросы кибербезопасности, No. 1, 2016. С. 24-30.

107. Петров Е.В. Обновление программного обеспечения в распределенных управляющих системах // Научно-технический вестник информационных технологий, механики и оптики, No. 45, 2007. С. 65-70.

108. Шадрин В.В. Реализация системы синхронизированного обновления компонентов операционной системы, прикладного программного обеспечения и средств защиты // Горный информационно-аналитический бюллетень (научно-технический журнал), No. 2, 2008. С. 303-309.

109. Atmel AT02333: Safe and Secure Bootloader Implementation for SAM3/4 // Официальныйсайткомпании ATMEL. 2013. URL: http://www.atmel.com/ Images/Atmel-42141-SAM-AT02333-Safe-and-Secure-Bootloader-Implementation-for-SAM3-4_Application-Note.pdf (дата обращения: 01.11.2016).

110. Биричевский А.Р. Подход к обеспечению безопасности взаимодействия процессов при разработке операционных систем. // **И**нформационная безопасность регионов России (ИБРР-2013). И 74 VIII Санкт-Петербургская межрегиональная конференция. Санкт-Петербург, 23-25 октября 2013 г. СПб. 2013. С. 49-50.

111. Биричевский А.Р. Использование распределенной ключевой системы в аппаратных средствах криптографической защиты информации // Юбилейная XIII Санкт-Петербургская международная конференция региональная информатика «РИ-2012». Санкт-Петербург, 24-26 октября 2012. С. 19.

112. Биричевский А.Р. Универсальная мобильная операционная система с подсистемами аутентификации и защиты информации на основе псевдовероятностного преобразования // Труды СПИИРАН, No. 3, 2016. С. 128-138.

Приложение А

Акты о внедрении результатов

Минобрнауки России

Федеральное государственное
бюджетное образовательное учреждение высшего образования
«Сыктывкарский государственный университет имени Питирима Сорокина»
(ФГБОУ ВО «СГУ им. Питирима Сорокина»)
Институт точных наук и информационных технологий

Октябрьский пр-т, д. 55, г. Сыктывкар, Республика Коми, 167001
телефон: (8212) 39-03-08, факс: (8212) 30-04-40, e-mail: ssu@svktsu.ru
ОКПО 02069547 ОГРН 1021100507230 ИНН/КПП 1101483236/110101001

03.05.2017 № б/н

А К Т
об использовании результатов диссертационной работы

Настоящий Акт составлен в том, что результаты диссертационной работы Биричевского Алексея Романовича на соискание ученой степени кандидата технических наук на тему «**Методы защиты информации на основе псевдовероятностного преобразования для мобильных устройств телекоммуникационных систем**», а именно:

– алгоритмы псевдовероятностного защитного преобразования, удовлетворяющие требованию вычислительной неотличимости от вероятностного шифрования;

– способы практического применения высокопроизводительных алгоритмов псевдовероятностного защитного преобразования в операционных системах;

– методы аутентификации пользователей с использованием алгоритмов псевдовероятностного защитного преобразования для защиты от принуждающей атаки;

– методы защиты программного обеспечения от дизассемблирования, основанные на введении ложных веток кода с помощью псевдовероятностного защитного преобразования кода;

– метод хранения ключей шифрования, основанный на применении псевдовероятностного защитного преобразования для обеспечения возможности сокрытия наличия резервных серий ключей;

используются кафедрой информационной безопасности Института точных наук и информационных технологий Сыктывкарского государственного университета имени Питирима Сорокина в учебном процессе на старших курсах обучения студентов по направлению «10.03.01 – Информационная безопасность» по дисциплинам «Информационная безопасность автоматизированных систем», «Программно-аппаратная защита информация», «Криптографические методы защиты информации», «Безопасность вычислительных сетей».

Заведующий кафедрой информационной безопасности,
к.ф.-м.н., доцент Л.С. Носов

Директор Института,
к.ф.-м.н, доцент С.В. Некипелов

Рисунок А.1. Акт внедрения в учебный процесс ФГБОУ ВО «СГУ им. Питирима Сорокина».

Crafe®

ИНН: 1101024296, КПП: 110101001
р/с: 40702810000030040717
в Северо-Западном филиале
ПАО «МТС-Банк» г. Санкт-Петербург
к/с: 30101810700000000893
БИК: 044030893

8(8212)24-37-41 e-mail: 11@8212.ru

А К Т

об использовании результатов диссертационной работы

г. Сыктывкар 3 мая 2017

Настоящий Акт составлен в том, что результаты диссертационной работы Биричевского Алексея Романовича на соискание ученой степени кандидата технических наук на тему «**Методы защиты информации на основе псевдовероятностного преобразования для мобильных устройств телекоммуникационных систем**», а именно:

- алгоритмы псевдовероятностного защитного преобразования, удовлетворяющие требованию вычислительной неотличимости от вероятностного шифрования;
- способы практического применения высокопроизводительных алгоритмов псевдовероятностного защитного преобразования в операционных системах;
- методы аутентификации пользователей с использованием алгоритмов псевдовероятностного защитного преобразования для защиты от принуждающей атаки;
- методы защиты программного обеспечения от дизассемблирования, основанные на введении ложных веток кода с помощью псевдовероятностного защитного преобразования кода;
- метод хранения ключей шифрования, основанный на применении псевдовероятностного защитного преобразования для обеспечения возможности сокрытия наличия резервных серий ключей,

используются в работе специалистами ООО «Крейф». Компания «Крейф» занимается разработкой перспективных средств криптографической защиты информации и имеет ряд патентов в данной области.

Генеральный директор Д.А. Гертнер

Рисунок А.2. Акт внедрения в производственный процесс ООО «Крейф».