

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет приборостроения,
информационных технологий и
электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность 100503 –

Специализация -

Информационная безопасность
автоматизированных систем
Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ПРОГРАММНОЕ СРЕДСТВО ЗАЩИТЫ ЦЕЛОСТНОСТИ И
ДОСТУПНОСТИ ВИЗУАЛЬНОЙ И ТЕКСТОВОЙ ИНФОРМАЦИИ

Студент

Старостин Евгений Владимирович

Руководитель

Кашаев Е.Д.

Консультант

Присталов П.М.

Нормоконтролер

Иванов А.П.

Рецензент начальник отдела АО НПП «Рубин»

Байкин А.А.

Работа допущена к защите (протокол заседания кафедры от 20.06.16 № 14)

Заведующий кафедрой

С.Л. Зефирев

Работа защищена с отметкой Одн.м.ч.д (протокол заседания ГЭК
от 24.06.16 № 18)

Секретарь ГЭК

А.П. Иванов

Продана ВКР
— 1084116
 С.Л. Зефирев

Пенза 2016

РЕФЕРАТ

Отчет содержит 111 с., 20 рис., 9 табл., 8 источников, 9 прил.

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ, ЦЕЛОСТНОСТЬ
ИНФОРМАЦИИ, ДОСТУПНОСТЬ ИНФОРМАЦИИ, ВИЗУАЛЬНАЯ
ИНФОРМАЦИЯ, ТЕКСТОВАЯ ИНФОРМАЦИЯ

Объектом исследований в данном дипломном проекте является программное средство защиты целостности и доступности визуальной и текстовой информации.

Цель дипломного проекта — разработка и реализация программного средства защиты целостности и доступности визуальной и текстовой информации.

В процессе дипломного проектирования был проведен анализ предметной области: местоположение приложения в системе, анализ потоков с которыми работает приложение; была разработана модель угроз для разрабатываемого приложения; был произведен анализ и выбор защитных мер для защиты от актуальных угроз, которые могут быть использованы для реализации программного средства защиты целостности и доступности визуальной и текстовой информации, было произведено тестирование разработанного программного средства.

В результате дипломного проектирования было разработано программное средство защиты целостности и доступности визуальной и текстовой информации.

СОДЕРЖАНИЕ

Введение	7
1 Анализ предметной области	8
1.1 Местоположение приложения в системе	8
1.2 Анализ потоков информации, с которыми работает приложение	9
1.3 Выбор инструментальных средств	10
2 Модель угроз	12
3 Выбор защитных мер для актуальных угроз	22
3.1 Защитные меры, применяемые для защиты от угрозы несанкционированной модификации защищаемой информации	24
3.2 Защитные меры, применяемые для защиты от угрозы доступности защищаемой информации	25
3.3 Защитные меры, применяемые для защиты от угрозы навязывания ложной информации	26
3.4 Защитные меры, применяемые для защиты от угрозы сбора информации о системе защиты	26
4 Программная реализация средств защиты	28
4.1 Класс MainWindow	31
4.2 Класс ScreenViewerWidget	35
4.3 Класс LogViewerWidget	36
4.4 Класс Hash3411	37
4.5 Класс ECPoint	42
4.6 Класс Sign3410	44
4.7 Класс Cipher3412	50
4.8 Класс StegoThread	54
4.9 Класс VerificationThread	56
5 Тестирование разработанной программы	58
Заключение	61

Список использованных источников	62
Приложение А. Листинг класса MainWindow	63
Приложение Б. Листинг класса ScreenViewerWidget	72
Приложение В. Листинг класса LogViewerWidget	76
Приложение Г. Листинг класса Hash3411	79
Приложение Д. Листинг класса ECPoint	89
Приложение Е. Листинг класса Sign3410	92
Приложение Ж. Листинг класса Cipher3412	97
Приложение З. Листинг класса StegoThread	105
Приложение И. Листинг класса VerificationThread	108

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящем отчете используются следующие термины с соответствующими определениями.

Информационно-управляющая система – цифровая система контроля или управления некоторым реальным объектом.

Авиационный комплекс радиообнаружения и наведения – электронная система разведки и управления, устанавливаемая на летательном средстве и предназначенная для дальнего обнаружения объектов противника (в воздухе, на земле, на воде), наведения на них средств поражения или перехвата, а также координации действий сил союзников и смежных задач.

Целостность информации – обеспечение достоверности и полноты информации и методов её обработки.

Доступность информации – состояние информации, при котором субъекты, имеющие права доступа, могут реализовать их беспрепятственно.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БИУС – бортовая информационно-управляющая система;

ДРЛОиУ – самолет дальнего радиолокационного обнаружения и управления;

ИС – информационная система;

АРМ – автоматизированное рабочее место;

ОС – операционная система.

ВВЕДЕНИЕ

Одной из основных задач информационных систем является обеспечение доступности и целостности информации, которая циркулирует в этих системах. Считается, что целостность и доступность информации защищены, если информация хранится или передается своевременно (с минимальным временем передачи сообщений), достоверной и полной, т.е. защищенной от случайных и злонамеренных искажений.

Цель дипломного проекта — разработка и реализация программного средства защиты целостности и доступности визуальной и текстовой информации.

В результате дипломного проектирования были проведены:

- анализ предметной области: местоположение приложения в системе, анализ потоков, с которыми работает приложение;
- разработка модели угроз для разрабатываемого приложения;
- анализ и выбор защитных мер для защиты от актуальных угроз, которые могут быть использованы для реализации программного средства защиты целостности и доступности визуальной и текстовой информации,
- программная реализация выбранных защитных мер;
- тестирование разработанного программного средства.

В результате дипломного проектирования было разработано программное средство защиты целостности и доступности визуальной и текстовой информации.

1 Анализ предметной области

1.1 Местоположение приложения в системе

Информационная система (ИС), для которой разрабатывается данное программное средство, представляет собой бортовую информационно-управляющую систему (БИУС) самолета дальнего радиолокационного обнаружения и управления (ДЛРОиУ).

Информационно-управляющая система - цифровая система контроля или управления некоторым реальным объектом. БИУС предназначена для сбора, обработки и отображения навигационной информации, обработки информационных данных от датчиков, установленных на системах и агрегатах самолета, а также для осуществления навигационных расчётов.

Авиационный комплекс радиообнаружения и наведения — электронная система разведки и управления, устанавливаемая на летательном средстве и предназначенная для дальнего обнаружения объектов противника (в воздухе, на земле, на воде), наведения на них средств поражения или перехвата, а также координации действий сил союзников и смежных задач. Подобные системы рассчитаны на использование в боевых действиях, однако могут быть использованы и для управления гражданским воздушным движением.

Структурно все авиационные комплексы радиообнаружения и наведения состоят из летательного аппарата (чаще всего самолёт), мощного радиолокатора, вспомогательной аппаратуры и средств связи.

Летательный аппарат выбирается с тем расчётом, чтобы в его фюзеляже хватило места для аппаратуры и операторов, а мощность электросистемы позволяла запитывать радиолокационную станцию. Это обуславливает выбор в качестве носителей серийных транспортных самолётов или бомбардировщиков.

Наиболее используемой сейчас является компоновка, при которой антенна радиолокатора размещена над фюзеляжем, а комплект аппаратуры и персонал — в грузовом отсеке. Однако существуют и другие варианты размещения оборудования — использование продольных или вращающихся

плоскостей фазированной антенной решетки, разнесённых по корпусу летательного аппарата.

Большая мощность радиолокатора даёт возможность проводить дальнейшее обнаружение и сопровождение целей, находясь вне зоны действия большинства комплексов стационарной ПВО и противовоздушных ракет истребительной авиации противника. Однако сам комплекс радиообнаружения хорошо заметен для противника (так как является источником мощного радиоволнового излучения, а также радио- и ИК-контрастен), небольшие скорости и неповоротливость самолётов-носителей делают их удобной мишенью, а высокая значимость и стоимость таких систем означает, что авиационные комплексы радиообнаружения являются высокоприоритетными целями. Всё это приводит к тому, что в реальных условиях комплексы радиообнаружения без прикрытия системами ПВО или истребительной авиацией не применяются[1].

Разрабатываемая программа предназначена для автоматизированного рабочего места (АРМ) инженера-оператора и представляет собой отдельный модуль, реализующий обеспечение целостности и доступности сохраняемой информации на жестком магнитном диске.

1.2 Анализ потоков информации, с которыми работает приложение

Во время работы оператора на автоматизированном рабочем месте ведется журнал передаваемых команд, а также оператор может произвести снимок экрана.

Снимки экрана сохраняются в формате JPEG, а журнал передаваемых команд сохраняется как текстовый файл в формате TXT.

Созданные файлы хранятся на жестком магнитном диске компьютера и используются при оценке правильности работы оператора, поэтому необходимо обеспечить целостность и доступность этих данных.

Целостность информации – обеспечение достоверности и полноты информации и методов её обработки.

Доступность информации – состояние информации, при котором субъекты, имеющие права доступа, могут реализовать их беспрепятственно.

Считается, что целостность и доступность информации защищены, если информация хранится или передается своевременно (с минимальным временем передачи сообщений), достоверной и полной, т.е. защищенной от случайных и злонамеренных искажений.

1.3 Выбор инструментальных средств

В соответствии с техническим заданием программное обеспечение написано с использованием библиотеки Qt версии 4.8.

Qt – кроссплатформенный инструментальный набор для разработки программного обеспечения на языке программирования C++. Qt позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем путём простой компиляции программы для каждой операционной системы (ОС) без изменения исходного кода.

Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования. Существуют версии библиотеки для Microsoft Windows, систем класса UNIX.

Отличительная особенность Qt от других библиотек — использование Meta Object Compiler (MOC) — предварительной системы обработки исходного кода. MOC позволяет во много раз увеличить мощь библиотек, вводя такие понятия, как слоты и сигналы. Кроме того, это позволяет сделать код более лаконичным.

Программное обеспечение написано для следующих ОС:

— ОС Astra Linux – операционная система специального назначения на базе ядра Linux, созданная для нужд российских силовых ведомств и спецслужб. Сертифицирована в системах сертификации средств

защиты информации Минобороны, ФСТЭК и ФСБ России. Включена в Единый реестр российских программ Минкомсвязи России.

— ОС Windows – операционная система корпорации Microsoft, ориентированная на применение графического интерфейса при управлении. При разработке и тестировании использовалась ОС Windows 7.

2 Модель угроз

Основной целью защиты является предотвращение угроз нарушения конфиденциальности, целостности и доступности информации, содержащейся в информационной системе. Данные угрозы могут привести к наступлению негативных последствий (ущерба) для обладателя информации.

Определение угроз безопасности информации должно носить систематический характер и осуществляться как на этапе создания информационной системы и формирования требований по ее защите, так и в ходе эксплуатации информационной системы.

В качестве источников угроз безопасности информации могут выступать субъекты (физические лица, организации, государства) или явления (техногенные аварии, стихийные бедствия, иные природные явления).

Источники угроз безопасности информации могут быть следующих типов:

- антропогенные источники (антропогенные угрозы);
- техногенные источники (техногенные угрозы);
- стихийные источники (угрозы стихийных бедствий, иных природных явлений).

В качестве источников антропогенных угроз безопасности информации могут выступать:

- внутренние нарушители - физические лица, имеющие право пребывания на территории контролируемой зоны, в пределах которой размещается оборудование ИС.
- внешние нарушители - физические лица, не имеющие права пребывания на территории контролируемой зоны, в пределах которой размещается оборудование ИС.

В рамках данного дипломного проекта рассматривается ИС, представляющая собой БИУС самолета ДРЛОиУ. Данная ИС является

закрытой, т.к. не имеет подключения к сетям общего пользования и международного обмена.

Предполагается, что внешний нарушитель не может воздействовать на защищаемую информацию по техническим каналам утечки. Организационными мерами полностью исключена возможность доступа в контролируемую зону, поэтому данный тип нарушителя не рассматривается.

К внутренним нарушителям могут относиться:

- сотрудники, имеющие санкционированный доступ в служебных целях в помещения, в которых размещаются ресурсы ИС, но не имеющие право доступа к ним;

- обслуживающий персонал (охрана, работники инженерно-технических служб и т.д.)

К операторам ввиду его исключительной роли в ИС должен применяться комплекс особых организационных мер по их подбору, принятию на работу и контролю выполнения функциональных обязанностей.

Операторы являются доверенными, однако необходимо учитывать человеческий фактор, и поэтому в качестве источника угрозы выступают непреднамеренные действия оператора (например, случайное удаление информации).

Внутренний нарушитель может использовать только штатные средства.

Источники техногенных угроз напрямую зависят от свойств техники. В качестве техногенных источников могут выступать:

- некачественные технические средства обработки информации;
- некачественные программные средства обработки информации;

Стихийные источники в данном проекте не рассматриваются.

Источники угроз ИБ, рассматриваемой ИС представлены в таблице 1.

Таблица 1 – Источники угроз ИБ

№	Источник угрозы	Возможные цели (мотивация) реализации угроз безопасности информации	Возможности по реализации угроз безопасности информации
1 Антропогенные источники			
1.1	оператор	<ul style="list-style-type: none"> - любопытство; - непреднамеренные или неквалифицированные действия; 	<p>имеют возможность получить информацию об уязвимостях отдельных компонентов информационной системы, опубликованных в общедоступных источниках</p>
1.2	сотрудники, имеющие санкционированный доступ в служебных целях в помещения, в которых размещаются ресурсы ИС, но не имеющие право доступа к ним	<ul style="list-style-type: none"> - причинение имущественного ущерба; - выявление уязвимостей с целью их дальнейшей продажи и получения финансовой выгоды; - внедрение ложной информации; - любопытство; - непреднамеренные или неквалифицированные действия 	<p>имеют возможность получить информацию об уязвимостях отдельных компонентов информационной системы, опубликованных в общедоступных источниках</p>

Продолжение таблицы 1

№	Источник угрозы	Возможные цели (мотивация) реализации угроз безопасности информации	Возможности по реализации угроз безопасности информации
1.3	обслуживающий персонал (охрана, работники инженерно-технических служб и т.д.)	<ul style="list-style-type: none"> - причинение имущественного ущерба; - выявление уязвимостей с целью их дальнейшей продажи и получения финансовой выгоды; - внедрение ложной информации; - любопытство; - непреднамеренные или неквалифицированные действия 	имеют возможность получить информацию об уязвимостях отдельных компонентов информационной системы, опубликованных в общедоступных источниках
2 Техногенные источники			
2.1	некачественные технические средства обработки информации	-	-
2.2	некачественные программные средства обработки информации	-	-

Уязвимости, которые могут быть использованы при реализации угроз ИБ, рассматриваемой ИС представлены в таблице 2.

Таблица 2 – Уязвимости

№	Уязвимости
1	ошибка персонала ИС
2	ошибка при проектировании ИС
3	недостаточность организационных мер на территории контролируемой зоны ИС

Методы нападений, которые могут быть использованы при реализации угроз ИБ, рассматриваемой ИС представлены в таблице 3.

Таблица 3 - Методы нападений

№	Методы нападений
1	несанкционированный доступ к защищаемой информации с использованием штатных средств ИС и недостатков механизмов разграничения доступа
2	использование оставленных без присмотра незаблокированных АРМ
3	физическое воздействие на элементы информационной системы
4	сбои и отказы программно-технических компонентов ИС
5	получения сведений об ИС, способствующих созданию благоприятных условий для применения других методов

Объекты защиты ИС представлены в таблице 4.

Таблица 4 – Объекты защиты

№	Объекты защиты
1	графическая информация (снимок экрана в формате JPEG)
2	текстовая информация (журнал переданных команд в формате TXT).
3	информация о системе защиты

Угрозы ИБ, рассматриваемой ИС представлены в таблице 5. В качестве косвенной угрозы безопасности информации может рассматриваться: угроза сбора информации о системе защиты. Знание алгоритмов работы системы защиты не дает возможности ее преодоления, однако эта информация не должна быть общедоступна.

Таблица 5 – Угрозы ИБ

№	Угрозы ИБ
1	угроза несанкционированной модификации защищаемой информации;
2	угроза несанкционированного удаления защищаемой информации
3	угроза навязывания ложной информации
4	угроза сбора информации о системе защиты

Далее были описаны все возможные сценарии инцидентов. Описание сценария «Модификация защищаемой информации, сохраненной на жестком магнитном диске» представлено в таблице 6.

Таблица 6 – Описание сценария №1 «Модификация защищаемой информации, сохраненной на жестком магнитном диске»

Характеристика	Возможные варианты
Угроза	угроза несанкционированной модификации защищаемой информации
Источник угрозы	<ul style="list-style-type: none"> - сотрудники, имеющие санкционированный доступ в служебных целях в помещения, в которых размещаются ресурсы ИС, но не имеющие право доступа к ним; - обслуживающий персонал (охрана, работники инженерно-технических служб и т.д.);
Объект нападения:	<ul style="list-style-type: none"> - графическая информация (снимок экрана в формате JPEG); - текстовая информация (журнал переданных команд в формате TXT);

Продолжение таблицы 6

Характеристика	Возможные варианты
Метод нападения	<ul style="list-style-type: none"> - несанкционированный доступ к защищаемой информации с использованием штатных средств ИС и недостатков механизмов разграничения доступа; - использование оставленных без присмотра незаблокированных АРМ;
Уязвимость	<ul style="list-style-type: none"> - ошибка при проектировании ИС; - недостаточность организационных мер на территории контролируемой зоны ИС;
Последствия инцидента:	<ul style="list-style-type: none"> - нарушение целостности защищаемой информации; - принятие неправильных решений; - необходимость дополнительных (незапланированных) затрат на восстановление информации;

Описание сценария «Удаление защищаемой информации, сохраненной на жестком магнитном диске» представлено в таблице 7.

Таблица 7 – Описание сценария №2 «Удаление защищаемой информации, сохраненной на жестком магнитном диске»

Характеристика	Возможные варианты
Угроза	угроза несанкционированного удаления защищаемой информации
Источник угрозы	<ul style="list-style-type: none"> - оператор; - сотрудники, имеющие санкционированный доступ в служебных целях в помещения, в которых размещаются ресурсы ИС, но не имеющие право доступа к ним; - обслуживающий персонал (охрана, работники инженерно-технических служб и т.д.); - некачественные технические средства обработки информации; - некачественные программные средства обработки информации;

Продолжение таблицы 7

Характеристика	Возможные варианты
Объект нападения:	<ul style="list-style-type: none"> - графическая информация (снимок экрана в формате JPEG); - текстовая информация (журнал переданных команд в формате TXT);
Метод нападения	<ul style="list-style-type: none"> - несанкционированный доступ к защищаемой информации с использованием штатных средств ИС и недостатков механизмов разграничения доступа; - использование оставленных без присмотра незаблокированных АРМ; - физическое воздействие на элементы информационной системы; - сбои и отказы программно-технических компонентов ИС;
Уязвимость	<ul style="list-style-type: none"> - ошибка персонала ИС; - ошибка при проектировании ИС, - недостаточность организационных мер на территории контролируемой зоны ИС;
Последствия инцидента:	<ul style="list-style-type: none"> - нарушение доступности защищаемой информации; - необходимость дополнительных (незапланированных) затрат на восстановление информации;

Описание сценария «Сохранение на жестком магнитном диске ложного снимка или журнала» представлено в таблице 8.

Таблица 8 – Описание сценария №3 «Сохранение на жестком магнитном диске ложного снимка или журнала»

Характеристика	Возможные варианты
Угроза	угроза навязывания ложной информации

Продолжение таблицы 8

Характеристика	Возможные варианты
Источник угрозы	<ul style="list-style-type: none"> - сотрудники, имеющие санкционированный доступ в служебных целях в помещения, в которых размещаются ресурсы ИС, но не имеющие право доступа к ним; - обслуживающий персонал (охрана, работники инженерно-технических служб и т.д.);
Объект нападения:	<ul style="list-style-type: none"> - ложная графическая информация (снимок экрана в формате JPEG); - ложная текстовая информация (журнал переданных команд в формате TXT);
Метод нападения	<ul style="list-style-type: none"> - несанкционированный доступ к защищаемой информации с использованием штатных средств ИС и недостатков механизмов разграничения доступа; - использование оставленных без присмотра незаблокированных АРМ;
Уязвимость	недостаточность организационных мер на территории контролируемой зоны ИС
Последствия инцидента:	- принятие неправильных решений;

Описание сценария «Раскрытие параметров системы защиты» представлено в таблице 9.

Таблица 9 – Описание сценария №4 «Раскрытие параметров системы защиты»

Характеристика	Возможные варианты
Угроза	угроза сбора информации о системе защиты
Источник угрозы	обслуживающий персонал (охрана, работники инженерно-технических служб и т.д.)

Продолжение таблицы 9

Характеристика	Возможные варианты
Объект нападения:	информация о системе защиты
Метод нападения	получения сведений об ИС, способствующих созданию благоприятных условий для применения других методов
Уязвимость	недостаточность организационных мер на территории контролируемой зоны ИС
Последствия инцидента:	последствия реализации не причиняют какой-либо ущерб обрабатываемой информации, но дают возможность реализовать непосредственные угрозы, перечисленные выше.

3 Выбор защитных мер для актуальных угроз

Был произведен анализ существующих защитных мер на операционных системах, для которых разрабатывается данная программа.

В ОС Astra Linux представлены следующие особенности по реализации требований безопасности информации:

- мандатное разграничение доступа. Принятие решения о запрете или разрешении доступа субъекта к объекту принимается на основе типа операции (чтение/запись/исполнение), мандатного контекста безопасности, связанного с каждым субъектом, и мандатной метки, связанной с объектом;

- изоляция модулей. Ядро операционной системы обеспечивает для каждого процесса в системе собственное изолированное адресное пространство. Данный механизм изоляции основан на страничном механизме защиты памяти, а также механизме трансляции виртуального адреса в физический. Любой доступ нескольких процессов к одному и тому же участку памяти обрабатывается диспетчером доступа в соответствии с дискреционными и мандатными правилами разграничения доступа;

- очистка оперативной и внешней памяти и гарантированное удаление файлов. Операционная система выполняет очистку неиспользуемых блоков файловой системы непосредственно при их освобождении;

- маркировка документов. Разработанный механизм маркировки позволяет серверу печати (CUPS) проставлять необходимые учетные данные в выводимых на печать документах. Вывод на печать документов без маркировки субъектами доступа, работающими в мандатном контексте с грифом выше "несекретно", невозможен;

- регистрация событий. Реализована подсистема протоколирования, осуществляющая надёжную регистрацию событий с использованием специального сервиса;

- механизм контроля замкнутости программной среды. Реализован механизм, обеспечивающий проверку неизменности и подлинности

загружаемых исполняемых файлов в формате ELF. Проверка производится на основе проверки векторов аутентичности, рассчитанных в соответствии с ГОСТ Р 34.10-2001 и внедряемых в исполняемые файлы в процессе сборки;

- контроль целостности. Для решения задач контроля целостности применяется функция хэширования в соответствии с ГОСТ Р 34.11-94;

- защищенная реляционная СУБД. В состав операционной системы входит объектно-реляционная СУБД PostgreSQL, в которой реализованы дискреционный и мандатный механизмы контроля доступа к защищаемым ресурсам БД;

- защищенный комплекс программ гипертекстовой обработки данных. В состав защищенного комплекса программ гипертекстовой обработки данных входят браузер Mozilla Firefox и web-сервер Apache, интегрированный со встроенными средствами защиты информации для обеспечения мандатного разграничения доступа при организации удаленного доступа к информационным ресурсам[2].

В ОС Windows 7 представлены следующие особенности по реализации требований безопасности информации:

- контроль учётных записей пользователей. Этот компонент запрашивает подтверждение действий, требующих прав администратора, в целях защиты от несанкционированного использования компьютера;

- ограничение выполнения приложений AppLocker. Обеспечивает не только защиту, но и соответствие правилам и политикам использования программ в организации за счет поддержки запрета запуска на компьютере нелегитимного ПО, вредоносного ПО или приложений;

- шифрование дисков BitLocker. Благодаря BitLocker похитители, использующие другую операционную систему или запускающие вредоносные программные средства, не смогут взломать защиту файлов и системы Windows или просмотреть в автономном режиме файлы, хранящиеся на защищенном диске[3].

3.1 Защитные меры, применяемые для защиты от угрозы несанкционированной модификации защищаемой информации

Для защиты целостности в ОС Astra Linux используется ГОСТ 34.11-94 "Информационная технология. Криптографическая защита информации. Функция хэширования", который на данный момент является устаревшим российским стандартом вычисления хэш-функции. В ОС Windows встроенных механизмов защиты целостности не предусмотрено.

Для защиты от угрозы несанкционированной модификации защищаемой информации в программе применяется электронно-цифровая подпись (ЭЦП).

Использование ЭЦП позволит также привязать сохраняемую информацию к определенному сеансу работы. При запуске программы генерируется сессионная пара открытого и закрытого ключа электронно-цифровой подписи, закрытый ключ по завершению сеанса уничтожается, открытый ключ хранится в журнале. Открытый ключ, используется для проверки информации при следующем запуске программы. Чтобы найти открытый ключ в журнале необходимо осуществить поиск по метке. В качестве метки применяется дата и время запуска программы. После закрытия программы необходимо производить проверку удаления информации о секретном ключе из оперативной памяти АРМ.

Для ЭЦП используется ГОСТ Р 34.10-2012. "Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи".

ГОСТ Р 34.10-2012 основан на эллиптических кривых. Стойкость этого алгоритма основывается на сложности вычисления дискретного логарифма в группе точек эллиптической кривой, а также на стойкости хэш-функции. В программе используется размер цифровой подписи равный 1024 бит, размер ключа – 512 бит.

Для ГОСТ Р 34.10-2012 используется хэш-функция по ГОСТ Р 34.11-2012 "Информационная технология. Криптографическая защита информации. Функция хэширования".

В программе используется размер хеш-функции равный 512 бит, размер блока входных данных — 512 бит.

В ГОСТ Р 34.11-2012 используется функции сжатия, в основе которой лежат три преобразования: нелинейное биективное преобразование, перестановка байт, линейное преобразование.

3.2 Защитные меры, применяемые для защиты от угрозы доступности защищаемой информации

В операционных системах Astra Linux и Windows не предусмотрены механизмы обеспечения доступности информации.

Основным методом реализации доступности является резервное копирование информации. Для обеспечения защиты от данной угрозы программа осуществляет соединением с сервером резервного копирования по протоколу FTP. Вся информация, сохраняемая программой на жесткий магнитный диск, копируется на сервер. При запуске программы сверяется наличие файлов на компьютере и на сервере.

Для обеспечения санкционированного удаления информации оператору необходимо доложить администратору об удалении файлов. Администратор выполняет удаление этих же файлов с сервера резервного копирования. В случае получения несанкционированного доступа нарушителя к программе, возможно удаление защищаемой информации, только с жесткого магнитного диска. При следующем запуске произведется проверка файлов, и выведется сообщение о удаленных файлах. Оператор сможет восстановить эти файлы с сервера резервного копирования. Для дополнительной защиты возможно применение программно-аппаратных средств защиты, которые должны предусматривать идентификацию и аутентификацию пользователя, а также ограничения по запуску программы после окончания сеанса оператора.

3.3 Защитные меры, применяемые для защиты от угрозы навязывания ложной информации

Во время работы программы на жесткий магнитный диск сохраняется следующая информация:

- снимки экрана;
- журнал передаваемых команд;
- журнал открытых ключей, используемых при проверке информации.

Для защиты от угрозы навязывания ложной информации необходимо шифровать журнал открытых ключей, чтобы нарушитель не смог вставить информацию в журнал о ложном открытом ключе. В случае реализации данной угрозы, нарушитель с легкостью сможет сохранить ложную информацию (снимок или журнал). Также при проверке доступности необходимо сверять количество файлов на сервере и на жестком магнитном диске АРМ, что уже было отмечено выше.

В качестве шифра используется блочный шифр "Кузнечик" описанный в ГОСТ Р 34.12-2015 "Информационная технология. Криптографическая защита информации. Блочные шифры". Используется размер блока шифрования 128 бит с длиной ключа равного 256 бит.

3.4 Защитные меры, применяемые для защиты от угрозы сбора информации о системе защиты

Данная угроза является косвенной. Знание алгоритмов работы системы защиты не дает возможности ее преодоления, однако эта информация не должна быть общедоступна.

Для обеспечения защиты от угрозы сбора информации о системе защиты необходимо скрыть от возможных нарушителей применяемые средства защиты.

Для того чтобы не хранить отдельный журнал для снимков экрана, в котором будет содержать ЭЦП и отметка, применяется стеганография.

Программа работает с графической информацией в формате JPEG. Для проверки ЭЦП, необходимо после изъятия данных из изображения, восстанавливать его к первоначальному виду (до встраивания информации). Это возможно только с добавлением избыточности в изображение. В программе используется метод ложных таблиц квантования. Он состоит в создании дополнительных ложных таблиц квантования. Обычно файлы JPEG содержат одну или две таблицы квантования (размер одной таблицы квантования равен 64 байтам). В стандарте JPEG учтена возможность использования нескольких таблиц квантования, т.е. это не нарушает внутреннюю организацию формата. В программе используются три ложные таблицы квантования: одна для метки, и две другие для ЭЦП в 1024 бит (в каждую таблицу помещается 512 бит)[4].

После извлечения скрытой информации, ее можно удалить из изображения и восстановить исходное изображение, и провести проверку целостности. Визуально обнаружить скрытую информацию не возможно, изображение не изменяется. Информацию можно обнаружить только hex-редактором, также нарушитель должен обладать необходимыми знаниями о формате JPEG.

Для журналов передаваемых команд в конец журнала записывается ЭЦП и метка. Затем журнал шифруется с помощью ГОСТ Р 34.12-2015.

4 Программная реализация средств защиты

Для более наглядного использования диаграмма последовательности была разбита на три части:

- инициализация программы;
- работа программы;
- закрытие программы.

Для того чтобы убрать возможные зависания программы, все вычисления требующие большого времени, были выделены в отдельный поток. В программы используются следующие потоки:

- главный поток, в данном потоке отображается интерфейс программы, создаются снимки экрана, рассчитывается ЭЦП журнала передаваемых команд, а также происходит взаимодействие программы с сервером резервного копирования

- поток расчета ЭЦП, в данный поток выделены расчет хэш-функции, ЭЦП снимков и вставка информации в снимки с помощью стеганографии;

- поток проверки ЭЦП, в данном потоке происходит проверка целостности и доступности снимков экрана и журналов передаваемых команд.

Т.к. снимки могут создавать каждую секунду, то в один момент времени может быть запущено несколько потоков расчета ЭЦП.

Диаграмма последовательности инициализации программы представлена на рисунке 1. При открытии программы проверяется ключ шифрования, в виде файла, который может быть сохранен на USB – флеш – накопителе или на любом портативном носителе информации. Если ключ правильный то главное окно программы запускается, если нет, программа завершает свою работу. При запуске программы происходит расшифрование журналов переданных команд и журнала открытых ключей, сохраненных на жестком магнитном диске АРМ. Далее генерируются сессионная пара открытого и закрытого ключа ЭЦП. Открытый ключ записывается в журнал

открытых ключей, закрытый ключ уничтожается при завершении сеанса. Инициализация программы считается завершенной.

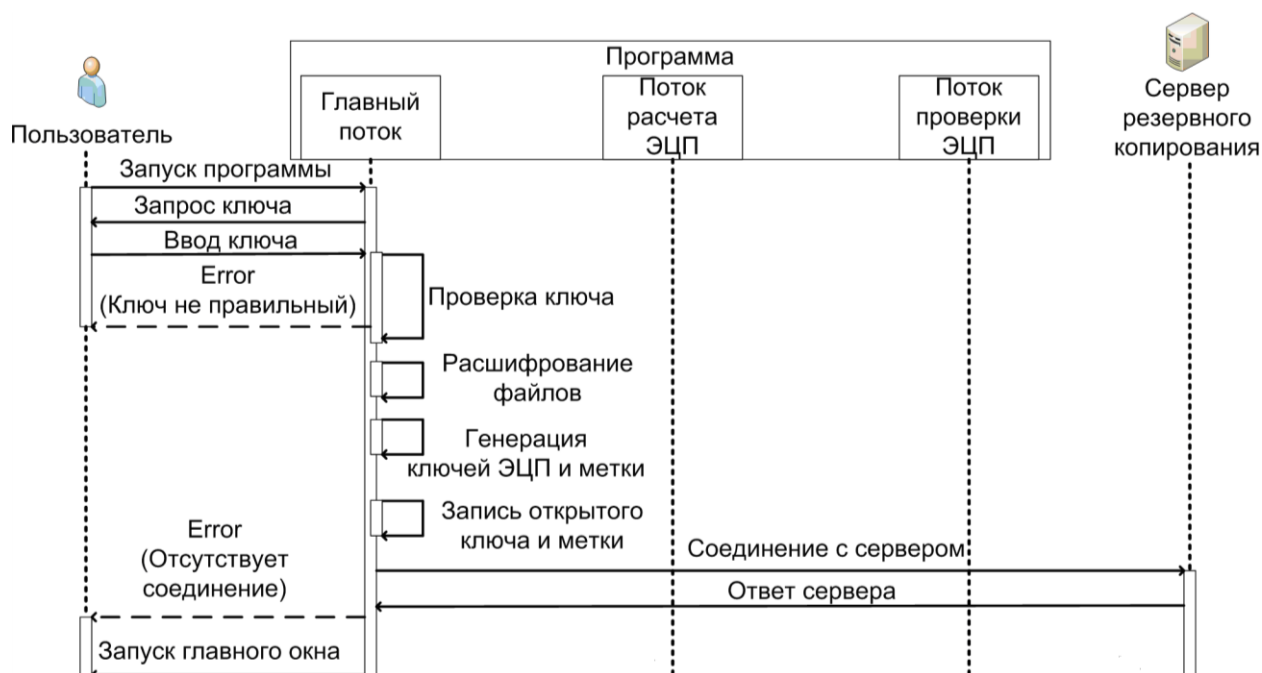


Рисунок 1 – Диаграмма последовательности инициализации программы

Диаграмма последовательности работы программы представлен на рисунке 2. Во время работы программы оператором могут быть созданы снимки экрана. Снимок экрана создается в главном потоке, расчеты ЭЦП и внедрение информации в изображение вынесены в отдельный поток. Все переданные оператором команды сохраняются в журнал передаваемых команд. Также после инициализации в отдельном потоке проверяется целостность и доступность информации на жестком магнитном диске. В этом потоке проверяется совпадение количества файлов на жестком магнитном диске АРМ и на сервере, далее по метке происходит поиск открытого ключа в журнале открытых ключей, и проверка ЭЦП.

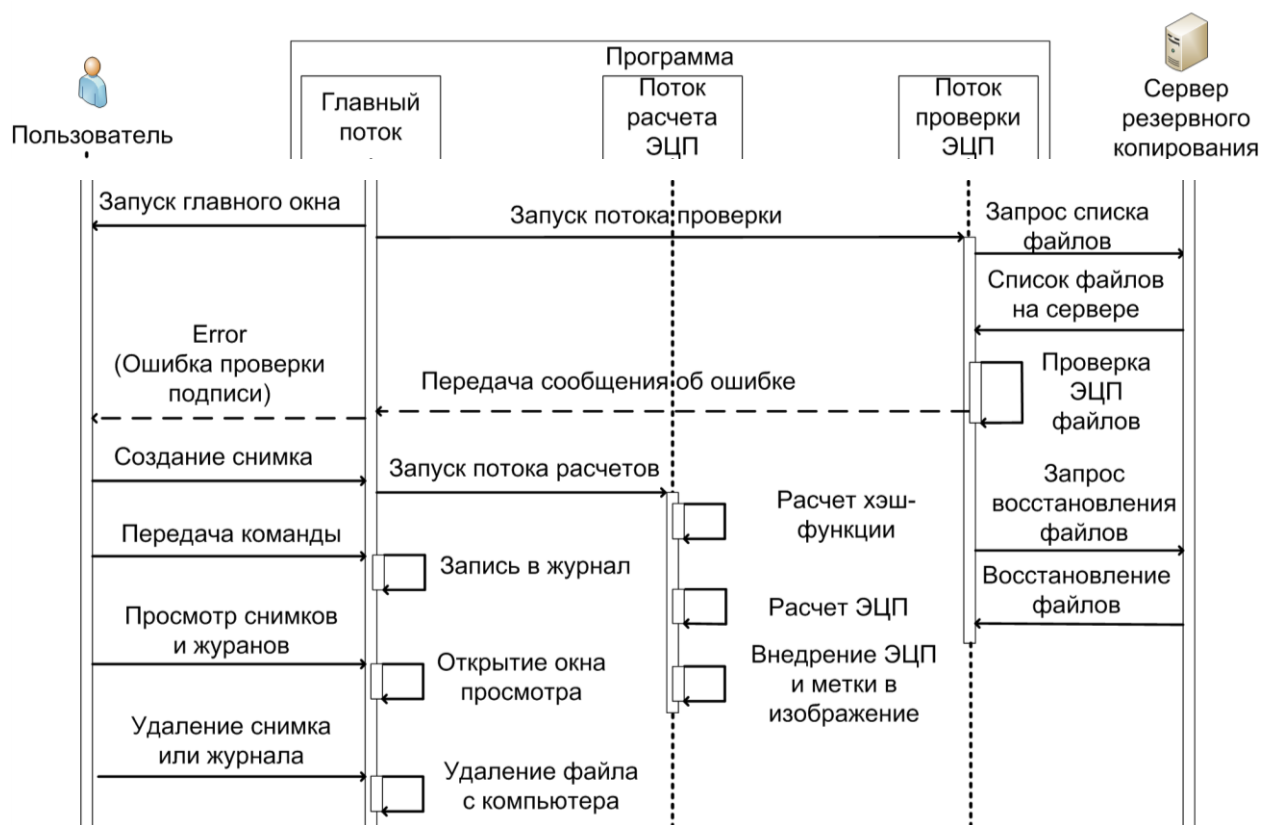


Рисунок 2 – Диаграмма последовательности работы программы

Диаграмма последовательности завершения работы программы представлена на рисунке 3.

При нажатии оператором кнопки выход, происходит ожидание завершения всех потоков расчета ЭЦП, чтобы сохранились все созданные снимки. Далее осуществляется резервное копирование созданных снимков и журнала переданных команд за время сеанса программы, а также на сервер отправляется журнала открытых ключей. После завершения операции резервного копирования программа завершается.

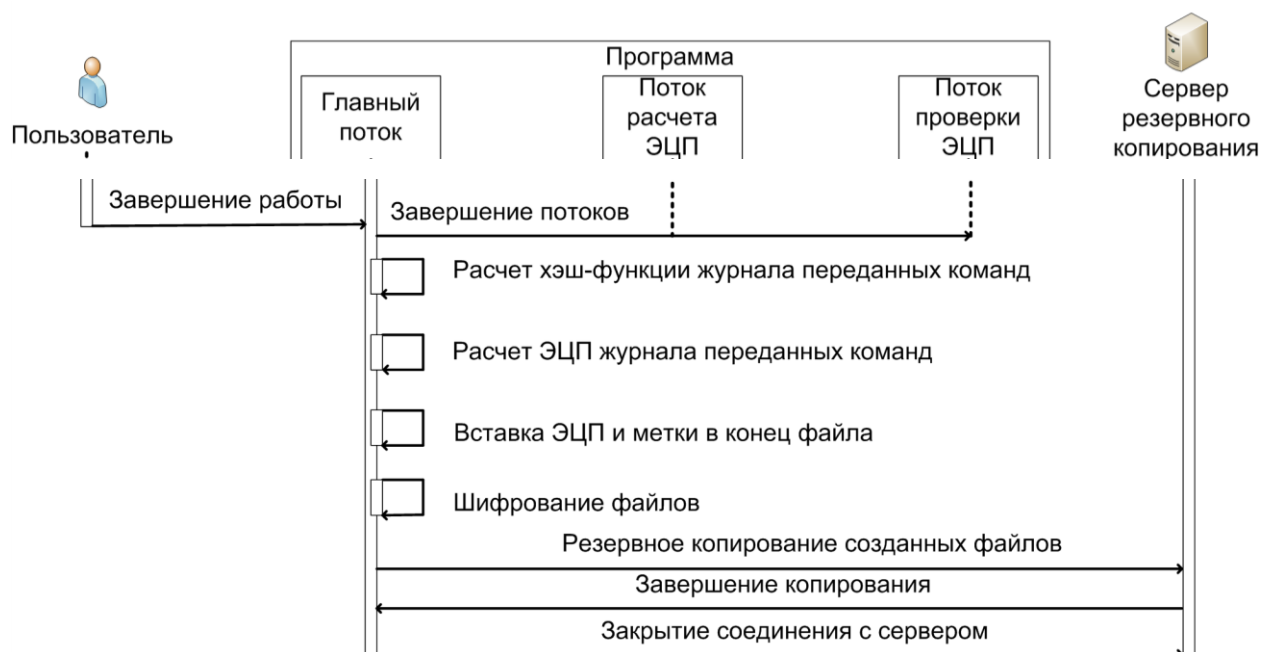


Рисунок 3 – Диаграмма последовательности завершения работы программы

4.1 Класс MainWindow

Данный класс реализует главное окно программы. Реализация класса представлена в приложение А. Перед началом работы происходит проверка ключа шифрования, представленная на рисунке 4.

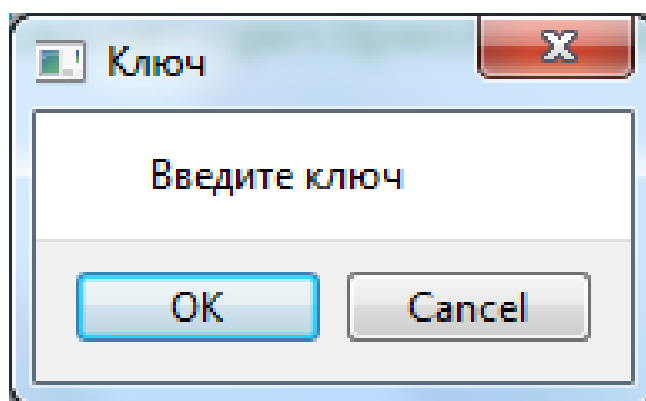


Рисунок 4 – Окно ввода ключа

При нажатии «Ок» выводит окно выбора файла-ключа, представленное на рисунке 5. При нажатии «Cancel» программа завершается.

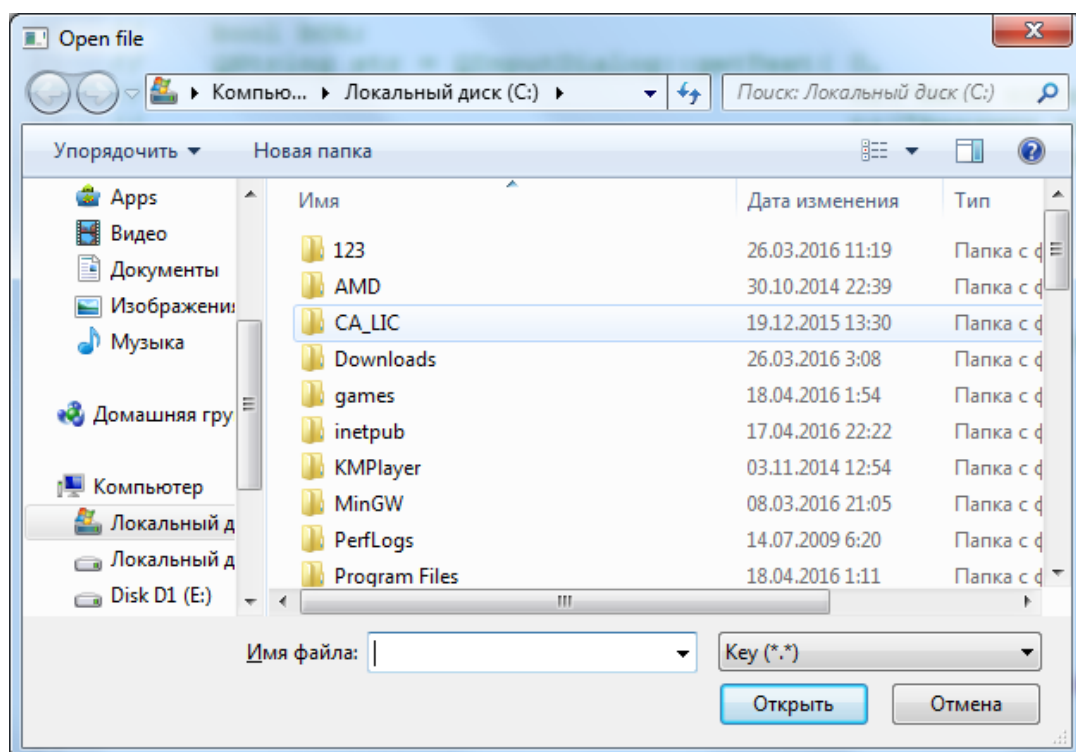


Рисунок 5 – Окно открытия ключа

Если ключ правильный программа запускается. Главное окно программы представлено на рисунке 6.

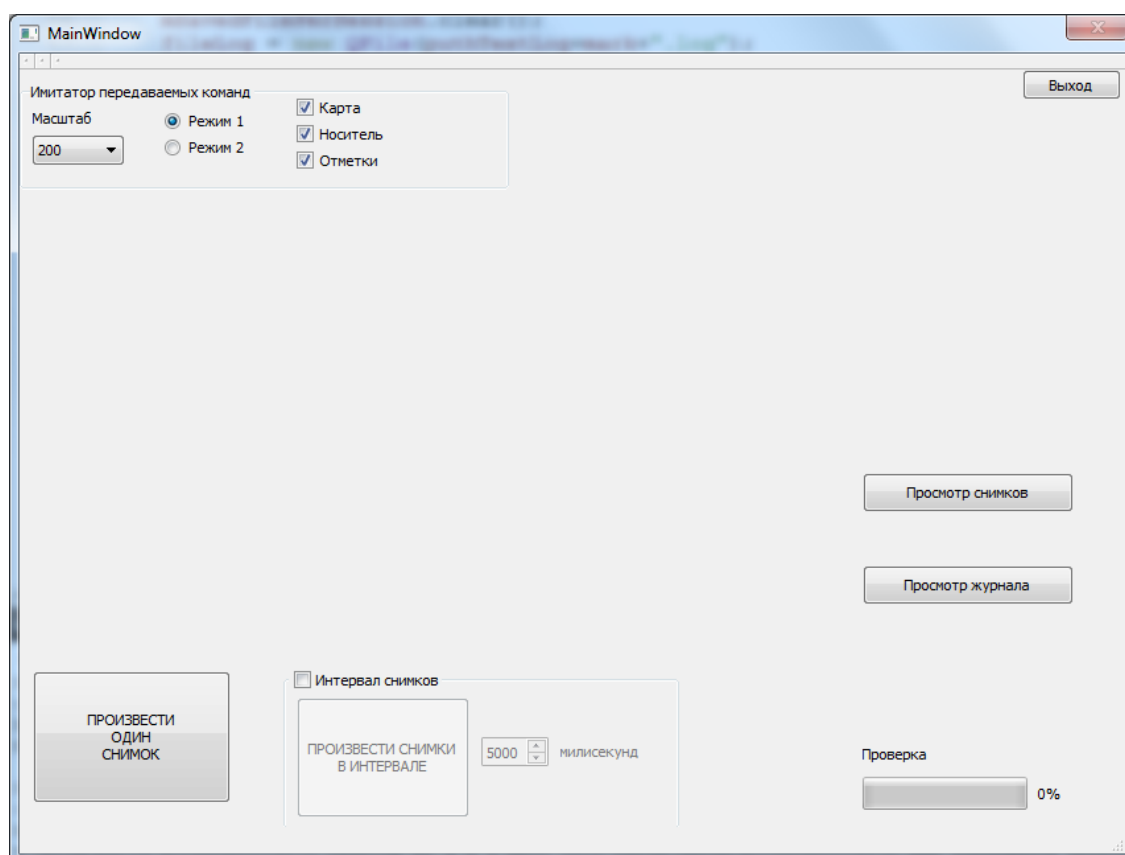


Рисунок 6 – главное окно программы

Далее расшифровываются все журналы, генерируется сессионная пара открытого и закрытого ключа, открытый ключ сохраняется в журнале.

Затем запускается проверка файлов на жестком магнитном диске на целостность и доступность.

Оператору доступно создание одного снимка или создание снимков в интервале. Т.к. данная программа является только модулем программы, то для передачи команд оператором используется «Имитатор передаваемых команд». Все команды записываются журнал передаваемых команд.

В нижнем правом углу отображается текущее состояние проверки. При нажатии кнопки «Просмотр снимков» отображается окно представленное в классе `ScreenViewerWidget`, описанным в подразделе 4.2. Оператору доступен просмотр и удаление снимков с жесткого магнитного диска. Для удаления с сервера резервного копирования, оператор должен обратиться к администратору.

При нажатии кнопки «Просмотр журнала» в главном окне программы отображается окно представленное в классе `LogViewerWidget`, описанным в подразделе 4.3. Оператору доступен просмотр и удаление журналов с жесткого магнитного диска. Для удаления с сервера резервного копирования, оператор должен обратиться к администратору.

В случае если сервер резервного копирования недоступен оператору выдается сообщение, представленное на рисунке 7.

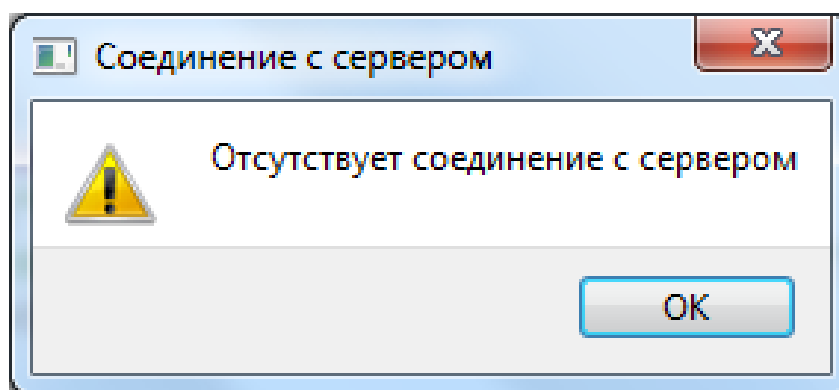


Рисунок 7 – Сообщение об отсутствии соединения с сервером

При возникновении ошибки при открытии файла оператору выдается сообщение, представленное на рисунке 8.

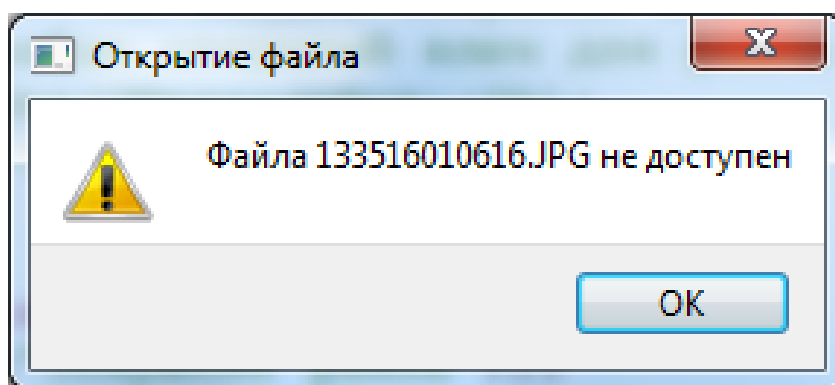


Рисунок 8 – Сообщение об ошибке открытия файла

В классе реализованы следующие функции:

- функция `slotGoStopInterval` – запускает/останавливает создание снимков экрана в интервале флажком интервал снимков;
- функция `slotInterval` – при выборе флажка «Интервал снимков» блокирует кнопку «Произвести один снимок», а активной становится кнопка «Произвести снимки в интервале» и наоборот;
- функция `screen` – создает снимок и новый поток, для расчета ЭЦП;
- функция `showScreenViewer` – отображение окна просмотра снимков;
- функция `showLogViewer` – отображение окна просмотра журналов;
- функция `ftpCommandFinished` - обработка команд, выполненных на сервере резервного копирования; параметрами является индекс операции и флаг выполнения команды или ошибки;
- функция `addToList` записывает список файлов, находящихся на сервер;
- функция `slotExit` выполняется при нажатии кнопки «Выход», осуществляется резервное копирование созданных файлов за сеанс работы и выход из программы.

4.2 Класс ScreenViewerWidget

Данный класс реализует окно просмотра и удаления снимков экрана, представленное на рисунке 9. Реализация класса представлена в приложении Б.

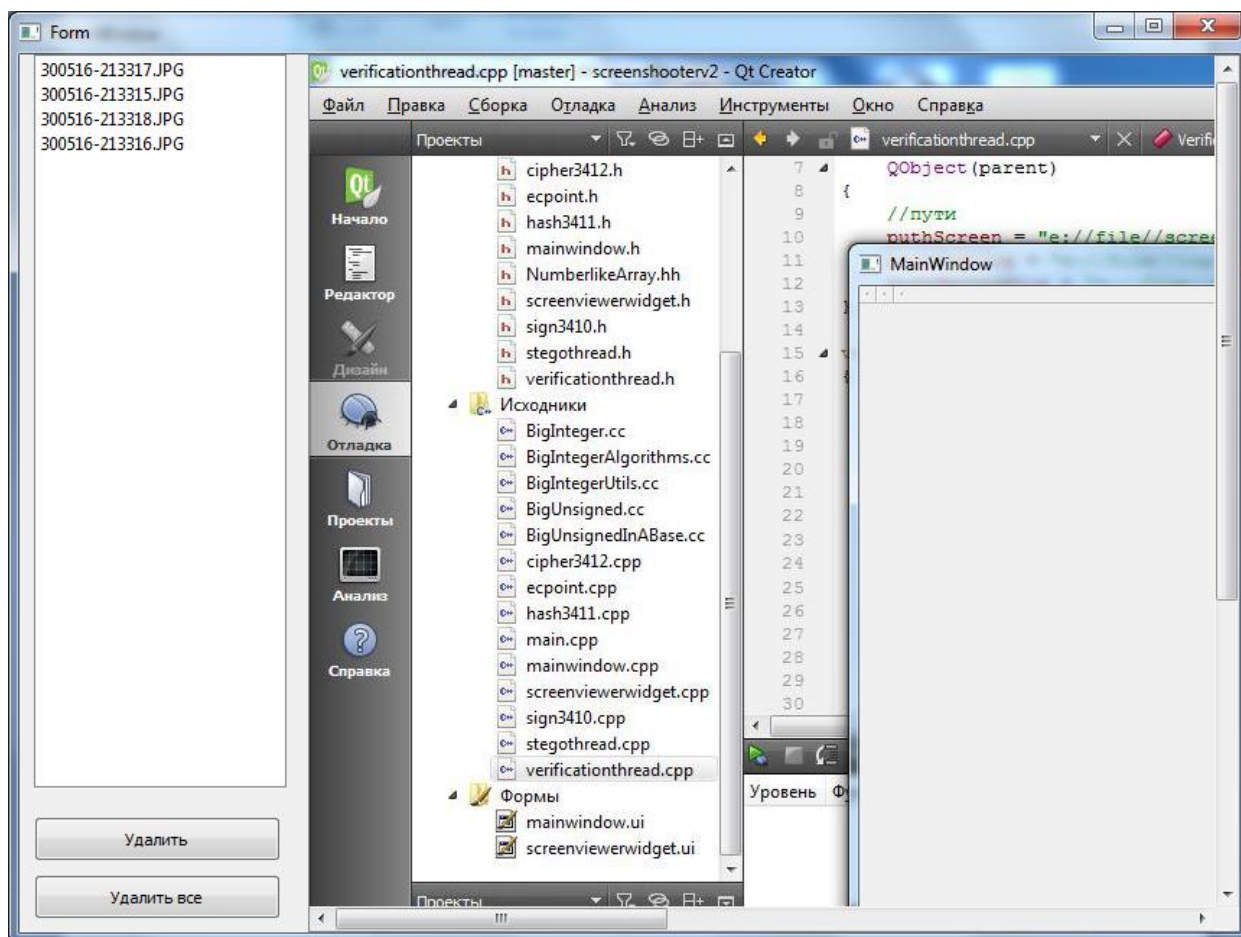


Рисунок 9 – Окно просмотра и удаления снимков экрана

При невозможности открытия файла для отображения оператору выводится на экран следующее сообщение, представленное на рисунке 10.

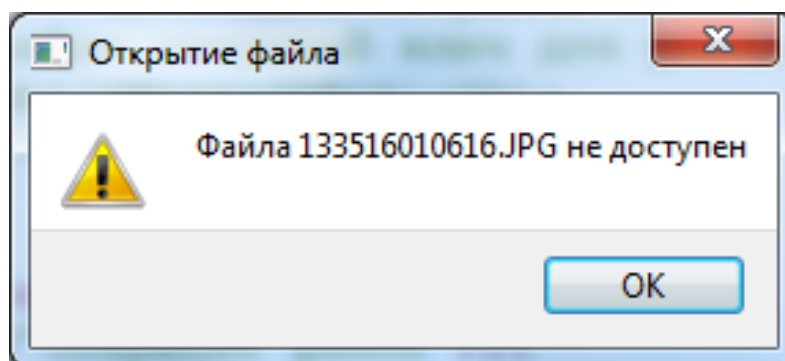


Рисунок 10 – Ошибка открытия файла

Данный класс, предоставляет оператору возможность просмотра созданных снимков экрана и их удаление с жесткого магнитного диска и с сервера резервного копирования.

В классе реализованы следующие функции:

- функция `refresh` – выполняется перед запуском окна, и загружает список файлов в класс;

- функция `slotCurrentScreenChanged` – выполняется при создании нового снимка. Входным параметром функции является имя файла, которое добавляется в список файлов указанный слева;

- функция `slotOnItemChanged` – отвечает за отображение выбранного снимка. В параметрах функции передается предыдущий элемент списка файлов, и тот который выбрали;

- функция `slotDeleteAll` – выполняется при нажатии кнопки «Удалить» и удаляет выбранный файл с НЖМД. Для полного удаления оператору необходимо обратиться к администратору;

- функция `slotDeleteCurrent` – выполняется при нажатии кнопки «Удалить все» и удаляет все файлы с НЖМД. Для полного удаления оператору необходимо обратиться к администратору.

4.3 Класс `LogViewerWidget`

Данный класс реализует окно просмотра и удаления журналов переданных команд, представленное на рисунке 11. Реализация класса представлена в приложении В.

Функции данного класса такие же, как и у класса описанного выше.

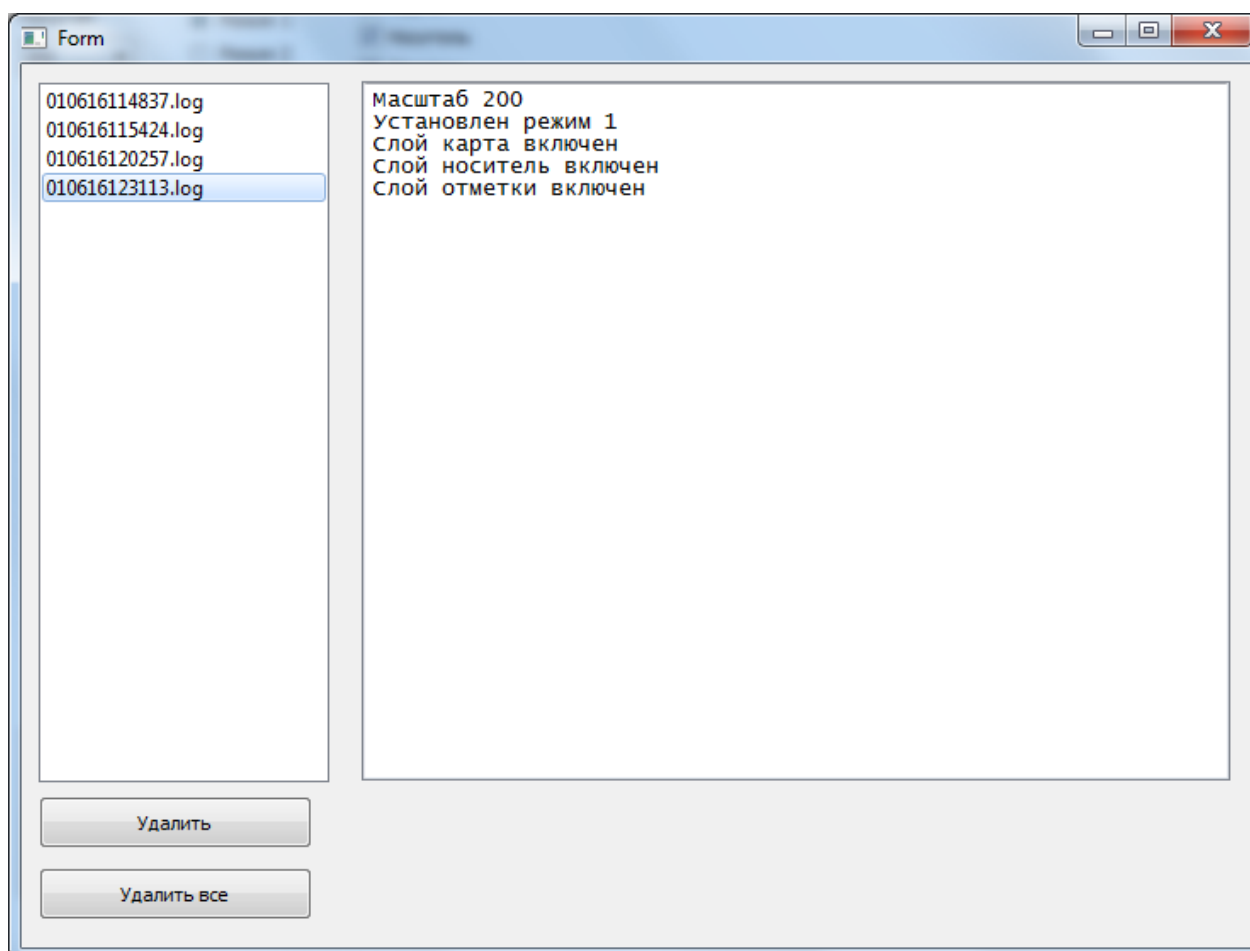


Рисунок 11 – Окно просмотра и удаления журналов переданных команд

4.4 Класс Hash3411

Данный класс реализует хэш-функцию описанную в ГОСТ Р 34.11-12 .
Класс представлен в приложении Г.

Для хеш-функции в ГОСТ Р 34.11-12 определены следующие значения параметров:

— инициализационные векторы: значение вектора IV для функции с длиной хэш-кода 512 бит равно 0^{512} - заполняется в функции hash_512, для функций с длиной хэш-кода 256 бит - $(00000001)^{64}$ - заполняется в функции hash_256;

— нелинейное биективное преобразование для множества двоичных векторов V_8 задается подстановкой:

$$\pi = \text{Vec}_8 \pi' \text{Int}_8 : V_8 \longrightarrow V_8,$$

$$\pi' : Z_{2^8} \longrightarrow Z_{2^8}$$

Значения подстановки π' заданы в массиве subs, представленным в приложении Г, листинг Г.2;

— значения перестановки τ заданы в массиве tau, представленным в приложении Г, листинг Г.2;

— линейное преобразование l множества двоичных векторов V_{64} задается умножением справа на матрицу A над полем $GF(2)$, строки которой записаны в шестнадцатеричном виде в матрице A в приложении Г, листинг Г.2;

— итерационные константы записаны в шестнадцатеричном виде в матрице C в приложении Г, листинг Г.2.

Для хеш-функции в ГОСТ Р 34.11-12 определены следующие преобразования:

— X преобразование. Представлено в классе функцией addXor512. На вход функции подаются две последовательности длиной 512 бит каждая, выходом функции является XOR этих последовательностей.

$$X[k] : V_{512} \longrightarrow V_{512}, X[k](a) = k \oplus a, k, a \in V_{512}$$

— S преобразование. Представлено в классе функцией S . Является обычной функцией подстановки. Каждый байт из 512-битной входной последовательности заменяется соответствующим байтом из таблицы подстановок subs.

$$S:V_{512} \longrightarrow V_{512}, S(a) = S(a_{63} \parallel \dots \parallel a_0) = \pi(a_{63}) \parallel \dots \parallel \pi(a_0),$$

$$a = a_{63} \parallel \dots \parallel a_0 \in V_{512}, a_i \in V_8, i = 0, \dots, 63$$

— Р преобразование. Представлено в классе функцией Р. Функция перестановки. Для каждой пары байт из входной последовательности происходит замена одного байта другим.

$$P:V_{512} \longrightarrow V_{512}, P(a) = P(a_{63} \parallel \dots \parallel a_0) = a_{\tau(63)} \parallel \dots \parallel a_{\tau(0)},$$

$$a = a_{63} \parallel \dots \parallel a_0 \in V_{512}, a_i \in V_8, i = 0, \dots, 63$$

— L преобразование. Представлено в классе функцией L. Представляет собой умножение 64-битного входного вектора на бинарную матрицу а[64].

$$L:V_{512} \longrightarrow V_{512}, L(a) = L(a_7 \parallel \dots \parallel a_0) = l(a_7) \parallel \dots \parallel l(a_0),$$

$$a = a_7 \parallel \dots \parallel a_0 \in V_{512}, a_i \in V_{64}, i = 0, \dots, 7$$

— функция сжатия. Значение хэш-кода сообщения $M \in V^*$ вычисляется с использованием итерационной процедуры. На каждой итерации вычисления хэш-кода используется функция сжатия:

$$g_N:V_{512} \times V_{512} \longrightarrow V_{512}, N \in V_{512}$$

Значение которой вычисляется по формуле:

$$g_N(h, m) = E(LSP(h \oplus N), m) \oplus h \oplus m,$$

$$E(K, m) = X[K_{13}]LSPX[K_{12}] \dots LSPX[K_2]LSPX[K_1](m)$$

Значения $K_i \in V_{512}, i = 1, \dots, 13$ вычисляются следующим образом:

$$K_1 = K;$$

$$K_i = LSPX(K_{i-1} \oplus C_{i-1}), i = 2, \dots, 13$$

Функция сжатия представлена в классе функцией g_N , в параметры которой передаются векторы N, h и m . В данной функции выполняется addXor512 с входными векторами N и h . Далее с полученным результатом выполняются последовательно S , P и L преобразования. Далее выполняется функция E , в каждом раунде которой вызывается функция $\text{KeySchedule}(K, i)$ отвечающая за формирование временного ключа K на каждом раунде функции $E(k, m)$. Результат функции $E \oplus h \oplus m$.

Исходными данными для процесса вычисления хэш-кода $H(M)$ является подлежащее хэшированию сообщение $M \in V^*$ и $IV \in V_{512}$ - инициализационный вектор хэширования[5].

Процесс вычисления хэш-функции представлен на рисунке 12.

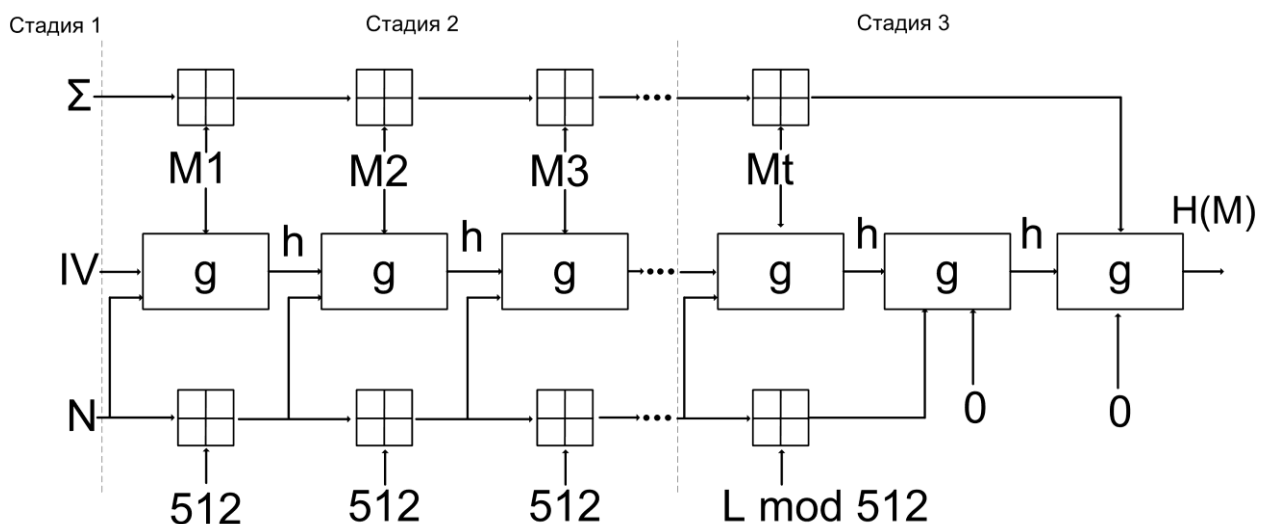


Рисунок 12 – Процесс вычисления хэш-функции

Процесс вычисления функции H состоит из следующих стадий:

— стадия 1: присвоить начальные значения текущих величин:

$$h := IV,$$

$$N := 0^{512} \in V_{512},$$

$$\Sigma := 0^{512} \in V_{512}$$

— стадия 2: проверить условие $|M| < 512$. При положительном исходе перейти к этапу 3. В противоположном случае l -битное сообщение сначала дополняется до размера, кратного 512 битам. Единичный бит присоединяется к концу сообщения; после него следуют $512 - 1 - (l \bmod 512)$ нулевых бит. Пусть $M = M_t \parallel M_{t-1} \parallel \dots \parallel M_1$ – это сообщение, состоящее из t блоков после дополнения, представленное форме от старших бит к младшим. Далее выполнить последовательность вычислений:

$$\begin{aligned} h &:= g_N(h, m), \\ N &:= \text{Vec}_{512}(\text{Int}_{512}(N) \oplus 512), \\ \Sigma &:= \text{Vec}_{512}(\text{Int}_{512}(\Sigma) \oplus \text{Int}_{512}(m)) \end{aligned}$$

— стадия 3 выполняем вычисления:

$$\begin{aligned} h &:= g_N(h, m), \\ N &:= \text{Vec}_{512}(\text{Int}_{512}(N) \oplus |M|), \\ \Sigma &:= \text{Vec}_{512}(\text{Int}_{512}(\Sigma) \oplus \text{Int}_{512}(m)), \\ h &:= g_0(h, m), \\ h &:= \begin{cases} g_0(h, \Sigma) \\ \text{MSB}_{256}(g_0(h, \Sigma)) \end{cases} \end{aligned}$$

Данный процесс представлен в классе функциями hash_512, hash_256 и hash_X.

В функциях hash_512, hash_256 задается инициализационный вектор хэширования и вызывается функция hash_X с параметрами:

- инициализационный вектор хэширования;
- файл для которого рассчитывается хеш-функция в виде байтового массива;

- длина файла;
- переменная для сохранения значения хеш-функции.

Для операции вычисления сложения в кольце Z_2^n вызывается функция addModulo512 - сложение по модулю 2^{512} .

4.5 Класс ECPoint

Данный класс реализует операции с точками на эллиптической кривой.

Реализация класса представлена в приложении Д.

Эллиптической кривой E , определенной над конечным полем F_p (где $p > 3$ - простое число), называется множество пар чисел $(x, y), x, y \in F_p$, удовлетворяющих тождеству:

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

где $a, b \in F_p$ и $4a^3 + 27b^2$ не сравнимо с нулем по модулю p .

Пары (x, y) удовлетворяющие тождеству, называются "точками эллиптической кривой E ", x и y - соответственно "координатами точки"[6].

Класс ECPoint выполняет действия над точками на эллиптической кривой. Каждая точка обладает следующими параметрами:

- a - коэффициент эллиптической кривой;
- b - коэффициент эллиптической кривой;
- x - координата точки на эллиптической кривой;
- y - координата точки на эллиптической кривой;
- fieldChar - модуль эллиптической кривой.

В классе реализованы следующие функции:

- operator + - сложение двух точек $p1$ и $p2$. Сложение двух точек представлено графически на рисунке 13.

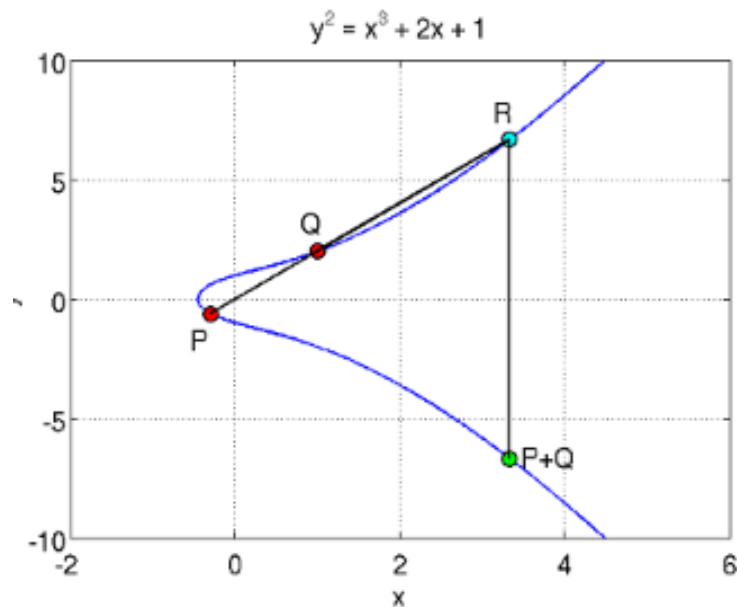


Рисунок 13 – Сложение двух точек на эллиптической кривой

На множестве всех точек эллиптической кривой E операцию сложения обозначают знаком "+".

Для точек P_1 и P_2 , координаты которых удовлетворяют условию $x_1 \neq x_2$, их суммой называется точка $P_3(x_3, y_3)$, координаты которой определяются сравнениями

$$\begin{cases} x_3 \equiv \lambda^2 - x_1 - x_2 \pmod{p}, \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 \pmod{p} \end{cases}$$

$$\text{где } \lambda \equiv \frac{y_2 - y_1}{x_2 - x_1} \pmod{p}$$

— функция `doublePoint` - сложение точки p с собой же. Если выполнены неравенства $x_1 = x_2$ и $y_1 = y_2 \neq 0$, то координаты P_3 , определяются следующим образом:

$$\begin{cases} x_3 \equiv \lambda^2 - 2x_1 \pmod{p}, \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 \pmod{p} \end{cases}$$

$$\text{где } \lambda \equiv \frac{3x_1^2 + a}{2y_1} \pmod{p}$$

— функция multiply - умножение точки на число x , по сути представляет сложение точки самой с собой X раз.

Алгоритм умножения точки эллиптической кривой на число состоит в следующем. Сначала полагаем результат равным несобственной точке. Затем просматриваем биты числа n от старших к младшему. Если встретили 0, умножаем текущее значение результата на 2. Если встретили 1 — умножаем результат на 2, после чего прибавляем P . Таким образом, для нахождения результата потребуется порядка $\log_2(k)$ шагов вместо k .

4.6 Класс Sign3410

Данный класс реализует ЭЦП описанную в ГОСТ Р 34.10-12 . Класс представлен в приложении Е.

Для ЭЦП в ГОСТ 34.10-12 установлены следующие параметры:

- простое число p - модуль эллиптической кривой;
- эллиптическая кривая E , задаваемая коэффициентами $a, b \in F_p$;
- целое число m - порядок группы точек эллиптической кривой E ;
- простое число q - порядок циклической подгруппы группы точек эллиптической кривой E , для которого выполнены следующие условия:

$$\begin{cases} m = nq, n \in \mathbb{Z}, n \geq 1 \\ 2^{254} < q < 2^{256} \text{ и } 2^{508} < m < 2^{512} \end{cases}$$

— точка $P \neq Q$ эллиптической кривой E , с координатами (x_p, y_p) , удовлетворяющая равенству $qP=0$

— хэш-функция $h(*): V^* \rightarrow V_l$, отображающая сообщения, представленные в виде двоичных векторов произвольной конечной длины, в двоичные вектора длины l бит. Если $2^{254} < q < 2^{256}$, то $l=256$. Если $2^{508} < q < 2^{512}$, то $l=512$.

Каждый оператор схемы цифровой подписи должен обладать личными ключами:

— ключ подписи - целым числом d , удовлетворяющим неравенству $0 < d < q$;

— ключ проверки подписи - точкой эллиптической кривой Q с координатами (x_q, y_q) , удовлетворяющей равенству $dP = Q$.

Для определения процессов формирования и проверки цифровой подписи необходимо установить соответствие между целыми числами и двоичными векторами длины l бит.

Вектор длиной l бит, в котором младшие биты расположены справа, а старшие - слева:

$$\bar{h} = (\alpha_{l-1}, \dots, \alpha_0), \bar{h} \in V_l,$$

где $a_i, i = 0, \dots, l-1$ равно либо 1, либо 0.

Число $a \in Z$ соответствует двоичному вектору $\bar{h}$, если выполнено равенство[7]:

$$a = \sum_{i=0}^{l-1} a_i 2^i$$

В классе Sign3410 применяются следующие параметры:

— простое число p модуль эллиптической кривой = 3623986102229
003635907788753683874306021320925534678605086546150450856166624002
482588482022271496854025090823603058735163734263822371964987228582
907372403;

— коэффициенты эллиптической кривой E $a = 7$; $b =$
151865506921082854508950034714043154928747527740206436194018823352
809982443793732829756914785974674866041605397883677596626326413990
136959047435811826396;

— порядок m группы точек эллиптической кривой =
362398610222900363590778875368387430602132092553467860508654615045
085616662396916489830503286306849996140407943793658545586519221297
0734808812618120619743;

— порядок циклической подгруппы q точек эллиптической кривой =
362398610222900363590778875368387430602132092553467860508654615045
085616662396916489830503286306849996140407943793658545586519221297
0734808812618120619743;

— координаты точки P эллиптической кривой E :
 $P_x=192835694406702284939930940124313759899778663545950797435707549$
 $130776659268583544106555768100318487481965800490321233288425233583$
 $0250729527632383493573274$;
 $P_y=228872869337197285997001215552947841635356232732950618031449742$
 $593110286030157281414199707227170880706659385065033415238185734779$
 $8885864807605098724013854$;

В классе реализованы следующие функции:

— функция перевода вектора в число `vectorToBigInteger`. Входными параметрами функция являются вектор, представленный символьным массивом и его длина. Результатом является число. Для определения процессов формирования и проверки цифровой подписи необходимо установить соответствие между целыми числами и двоичными векторами длины L бит. Двоичный вектор длиной L бит, в котором младшие биты расположены справа, а старшие – слева представлен ниже:

$$\bar{h} = (\alpha_{l-1}, \dots, \alpha_0), \bar{h} \in V_l$$

где $\alpha_i, i = 0, \dots, l-1$ равно либо 1, либо 0.

Число $a \in Z$ соответствует двоичному вектору $\bar{h}$, если выполнено равенство:

$$\alpha = \sum_{i=0}^{L-1} a_i 2^i$$

— функция `toHexString`. Входные параметры: число. Возвращаемый результат: строка представляющая двоичный вектор. Обратная функция `vectorToBigInteger`.

— функция `randomLess`. Входные параметры: максимальное число, результатом является случайно сгенерированное число от 0 до числа указанного в параметрах.

— функция генерации секретного ключа и расчета открытого ключа `genKeys()`. Секретный ключ генерируется с помощью функции указанной выше. Открытый ключ – точка эллиптической кривой Q с координатами (x_q, y_q) , удовлетворяющей равенству $dP=Q$.

— функция `setPublicKey`. Устанавливает открытый ключ для объекта класса числами указанными в параметрах функции.

Формирование подписи осуществляется функция `signGen` в соответствии с алгоритмом представленным на рисунке 14.

Для получения цифровой подписи под сообщением $M \in V^*$ необходимо выполнить следующие действия (шаги) по алгоритму:

1) вычислить хэш-код сообщения $M : \bar{h} = h(M)$ хэш-код вычисляется в классе `Nash3411`, представленным в приложении Б;

2) вычислить целое число a , двоичным представлением которого является вектор $\bar{h}$ и определить $e \equiv a \pmod{q}$. Если $e=0$, то определить $e=1$. Для вычисления двоичного представления используется функция `vectorToBigInteger(unsigned char vector[], unsigned int length);`

3) сгенерировать случайное (псевдослучайное) целое число, удовлетворяющее равенству: $0 < k < q$ Для генерации используется функция `randomLess (BigInteger Max)`, где в качестве параметра `Max` указывается число q ;

4) вычислить точку эллиптической кривой $C=kP$ и определить $r \equiv x_c \pmod{q}$. Если $r=0$, то вернуться к шагу 3. Точка рассчитывается в классе ECPPoint представленным в приложении В;

5) вычислить значение $s \equiv (rd + ke) \pmod{q}$. Если $s=0$, то вернуться к шагу 3;

6) вычислить двоичные векторы $\bar{r}$ и $\bar{s}$, соответствующие r и s , и определить цифровую подпись $\zeta = (\bar{r} \parallel \bar{s})$ как конкатенацию двух двоичных векторов.

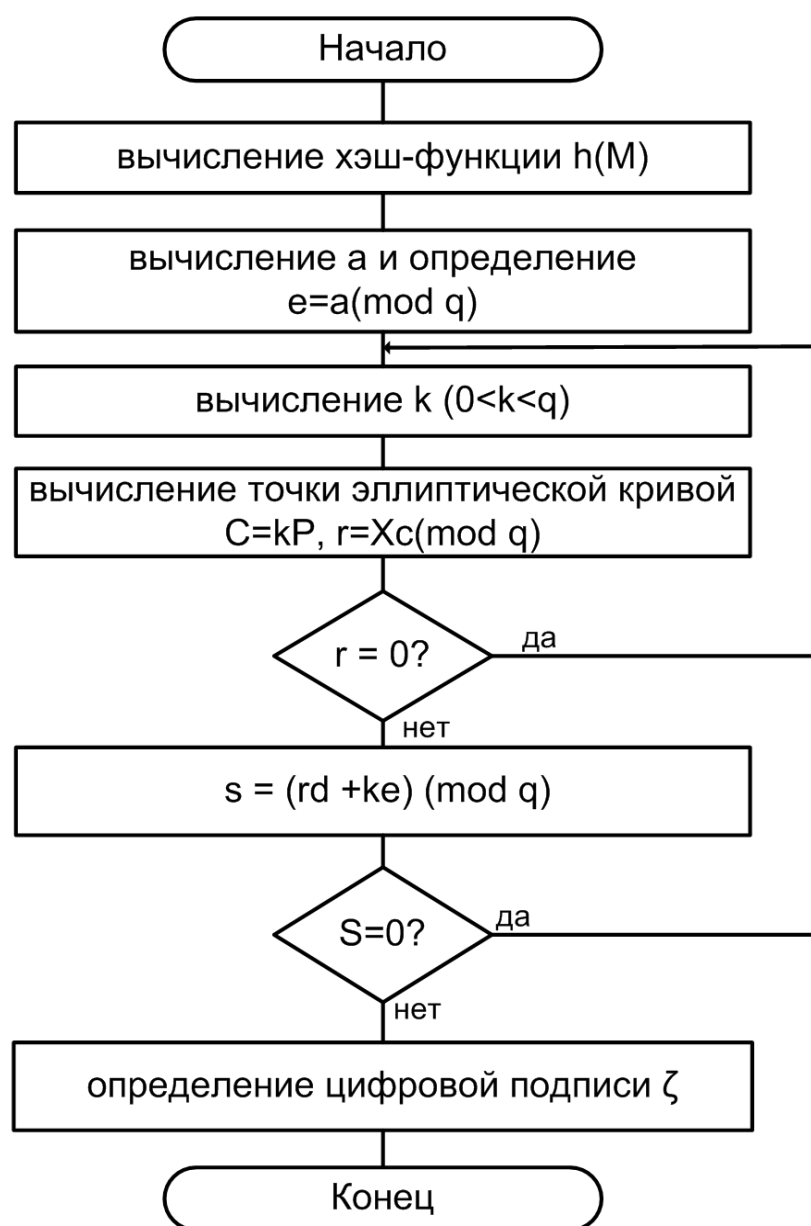


Рисунок 14– Алгоритм формирования электронно-цифровой подписи

Проверка цифровой подписи реализована функцией signVer. Проверка подписи осуществляется в соответствии с алгоритмом, представленным на рисунке 15.

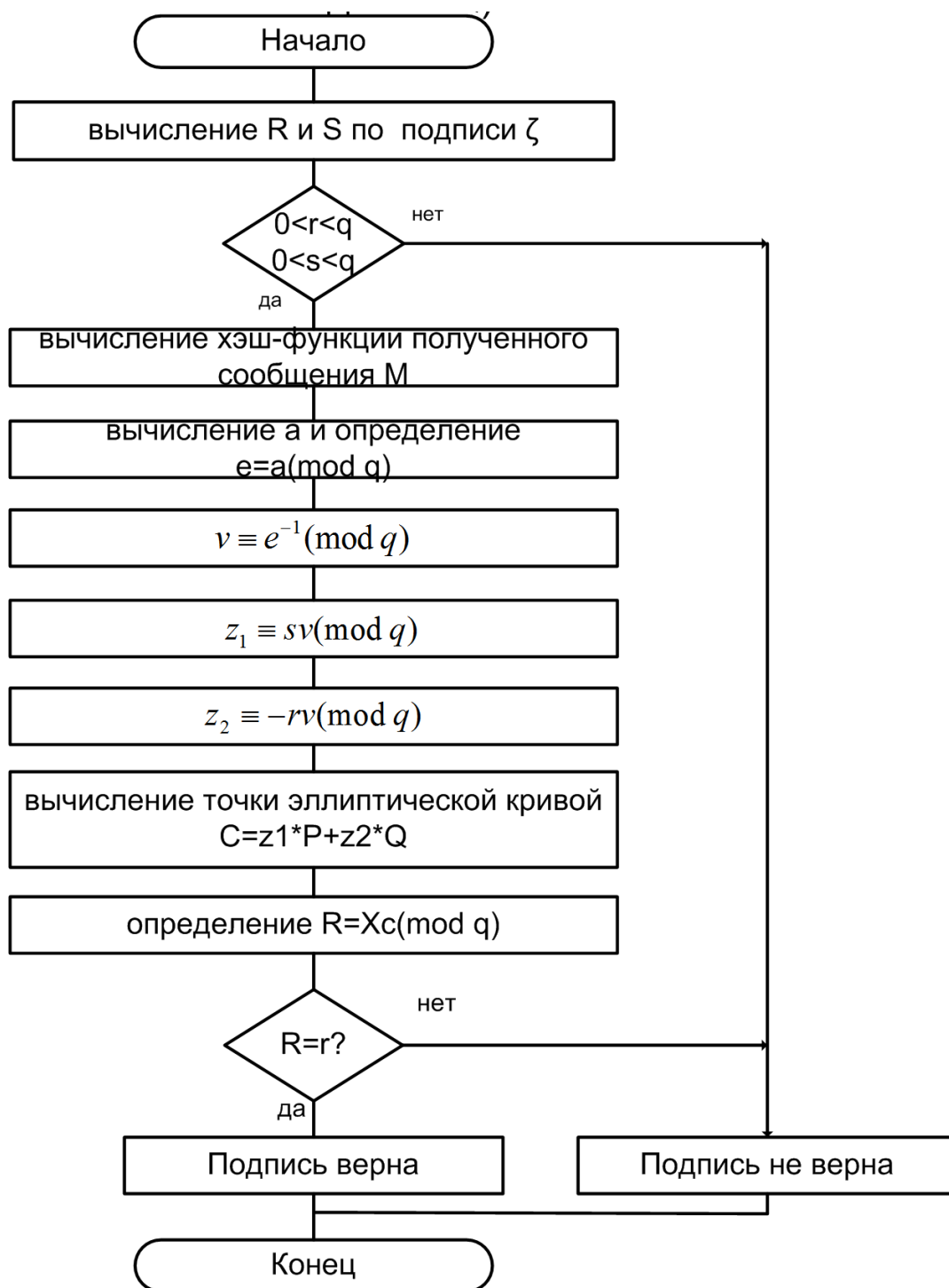


Рисунок 15– Алгоритм проверки электронно-цифровой подписи

Для проверки подписи необходимо выполнить следующие действия:

1) по полученной подписи ζ вычислить целые числа r и s . Если выполнены неравенства $0 < r < q$, $0 < s < q$, то перейти к следующему шагу. В противном случае подпись не верна;

2) вычислить хэш-код сообщения $M : \bar{h} = h(M)$. Хэш-код вычисляется в классе Hash3411 представленным в приложении Б;

3) вычислить целое число a , двоичным представлением которого является вектор $\bar{h}$ и определить $e \equiv a(\bmod q)$. Если $e=0$, то определить $e=1$. Для вычисления двоичного представления используется функция `vectorToBigInteger(unsigned char vector[], unsigned int length);`

4) вычислить значение $v \equiv e^{-1}(\bmod q)$;

5) вычислить значения $z_1 \equiv sv(\bmod q)$ и $z_2 \equiv -rv(\bmod q)$;

6) вычислить точку эллиптической кривой $C = z_1P + z_2Q$ и определить $R \equiv x_c(\bmod q)$ точка рассчитывается в классе ECPoint;

7) если выполнено равенство $R=r$, то подпись принимается, в противном случае - подпись неверна

4.7 Класс Cipher3412

Данный класс представлен в приложении Ж. В классе используются шифр с длиной блоков $n=128$ бит и длинами ключей $k = 256$ бит.

Для алгоритма блочного шифрования в ГОСТ Р 34.12-15 определены следующие значения параметров:

— нелинейное биективное преобразование для множества двоичных векторов V_8 задается подстановкой:

$$\pi = Vec_8 \pi' Int_8 : V_8 \longrightarrow V_8,$$
$$\pi' : Z_{2^8} \longrightarrow Z_{2^8}$$

значения подстановки π' заданы в массиве subs[256], представленным в приложении Г, листинг Г.2;

— линейное преобразование задается отображением $l: V_8^{16} \longrightarrow V_8$.

Для алгоритма блочного шифрования в ГОСТ Р 34.11-12 определены следующие преобразования:

— X преобразование. Реализация представлена в функции encrypt и decrypt. Представляет собой XOR двух значений.

$$X[k]: V_{128} \longrightarrow V_{128}, X[k](a) = k \oplus a, k, a \in V_{128}$$

— S преобразование. Реализация представлена в функции S – является обычной функцией подстановки из массива pi.

$$S: V_{128} \longrightarrow V_{128}, S(a) = S(a_{15} \parallel \dots \parallel a_0) = \pi(a_{15}) \parallel \dots \parallel \pi(a_0), \\ a = a_{15} \parallel \dots \parallel a_0 \in V_{128}, a_i \in V_8, i = 0, \dots, 15$$

— S^{-1} преобразование. Обратная функция S

$$S^{-1}: V_{128} \longrightarrow V_{128}, S^{-1}(a) = S^{-1}(a_{15} \parallel \dots \parallel a_0) = \pi^{-1}(a_{15}) \parallel \dots \parallel \pi^{-1}(a_0), \\ a = a_{15} \parallel \dots \parallel a_0 \in V_{128}, a_i \in V_8, i = 0, \dots, 15$$

— R преобразование Реализация представлена в функции R.

$$R(a) = R(a_{15} \parallel \dots \parallel a_0) = l(a_{15}, \dots, a_0) \parallel a_{15} \parallel \dots \parallel a_1 \\ a = a_{15} \parallel \dots \parallel a_0 \in V_{128}, a_i \in V_8, i = 0, 1, \dots, 15$$

— L преобразование. Реализация представлена в функции L

$$L(a) = R^{16}(a)$$

— R^{-1} преобразование Обратная функция R

$$R(a) = R(a_{15} \parallel \dots \parallel a_0) = l(a_{15}, \dots, a_0) \parallel a_{15} \parallel \dots \parallel a_1$$

$$a = a_{15} \parallel \dots \parallel a_0 \in V_{128}, a_i \in V_8, i = 0, 1, \dots, 15$$

— L^{-1} преобразование Обратная функция L

$$L^{-1}(a) = (R^{-1})^{16}(a)$$

Алгоритм развертывания ключа использует итерационные константы $C_i \in V_{128}, i = 1, 2, \dots, 32$ которые определены следующим образом:

$$C_i = L(\text{Vec}_{128}(i)), i = 1, 2, \dots, 32$$

Итерационные ключи $K_i \in V_{128}, i = 1, 2, \dots, 10$ вырабатываются на основе ключа $K = k_{255} \parallel \dots \parallel k_0 \in V_{256}, k_i \in V_1, i = 0, 1, \dots, 255$ и определяются равенствами[8]:

$$K1 = k_{255} \parallel \dots \parallel k_{128}$$

$$K2 = k_{127} \parallel \dots \parallel k_0$$

$$(K_{2i+1}, K_{2i+2}) = F[C_{8(i-1)+8}] \dots F[C_{8(i-1)+1}](K_{2i-1}, K_{2i}), i = 1, 2, 3, 4.$$

Данный алгоритм представлен функцией `key_setup(unsigned long long int * master_key, unsigned long long int ** subkeys)`. Параметрами этой функции передаются ключ и массив для сохранения итерационных ключей.

Алгоритм зашифрования представлен на рисунке 16.

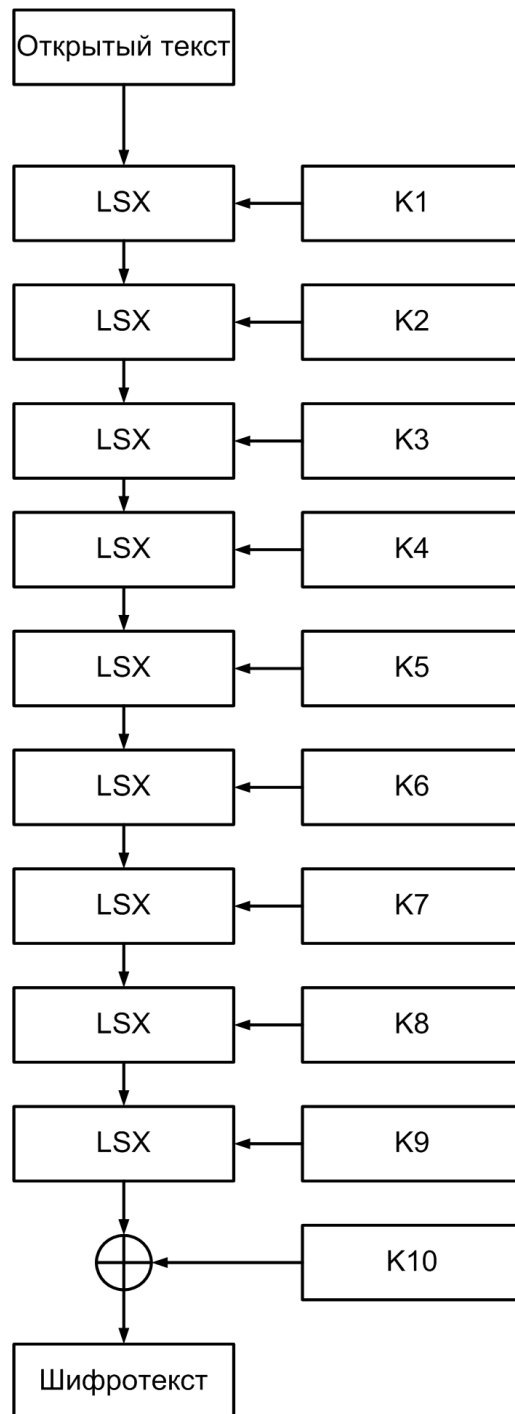


Рисунок 16 – Алгоритм зашифрования

Алгоритм зашифрования в зависимости от значений итерационных ключей $K_i \in V_{32}, i = 1, 2, \dots, 32$ реализует подстановку:

$$E_{K_1, \dots, K_{32}}(a) = G * [K_{32}] G[K_{31}] \dots G[K_2] G[K_1](a1, a0)$$

Данный алгоритм представлен в функции encrypt(unsigned long long int * * subkeys, unsigned long long int * plaintext, unsigned long long int * ciphertext). Параметрами этой функции являются массив итерационных ключей, а также открытый текст в виде побайтового массива и переменная для записи зашифрованного текста.

Алгоритм расшифрования представлен на рисунке 17. Алгоритм расшифрования в зависимости от значений итерационных ключей $K_i \in V_{32}, i = 1, 2, \dots, 32$ реализует подстановку:

$$D_{K_1, \dots, K_{32}}(a) = G * [K_1] G[K_2] \dots G[K_{31}] G[K_{32}](a_1, a_0)$$

Данный алгоритм представлен в функции decrypt(unsigned long long int * * subkeys, unsigned long long int * ciphertext, unsigned long long int * plaintext). Параметрами этой функции являются массив итерационных ключей, а также закрытый текст в виде побайтового массива и переменная для записи зашифрованного текста.

4.8 Класс StegoThread

Данный класс представляет собой отдельный поток расчетов ЭЦП для снимков экрана. Реализация класса представлена в приложении 3.

В данном классе реализована функция stego. Входными параметрами являются:

- имя файла для которого рассчитывается подпись;
- закрытый ключ;
- отметка – дата и время начала сеанса работы программы.

Функция устанавливает переданный секретный ключ для класса Sign3411 с помощью функции setPrivateKey.

Далее открывается файл указанный во входных параметрах. Для данного файла рассчитывается хэш-функция, и ЭЦП. Далее подпись и отметка вставляются в изображение при помощи метода стеганографии

указанного в подразделе 3.4. Если в качестве файла используется текстовый журнал, то подпись и метка записывается в конец файла и файл при завершении сеанса программы будет зашифрован.

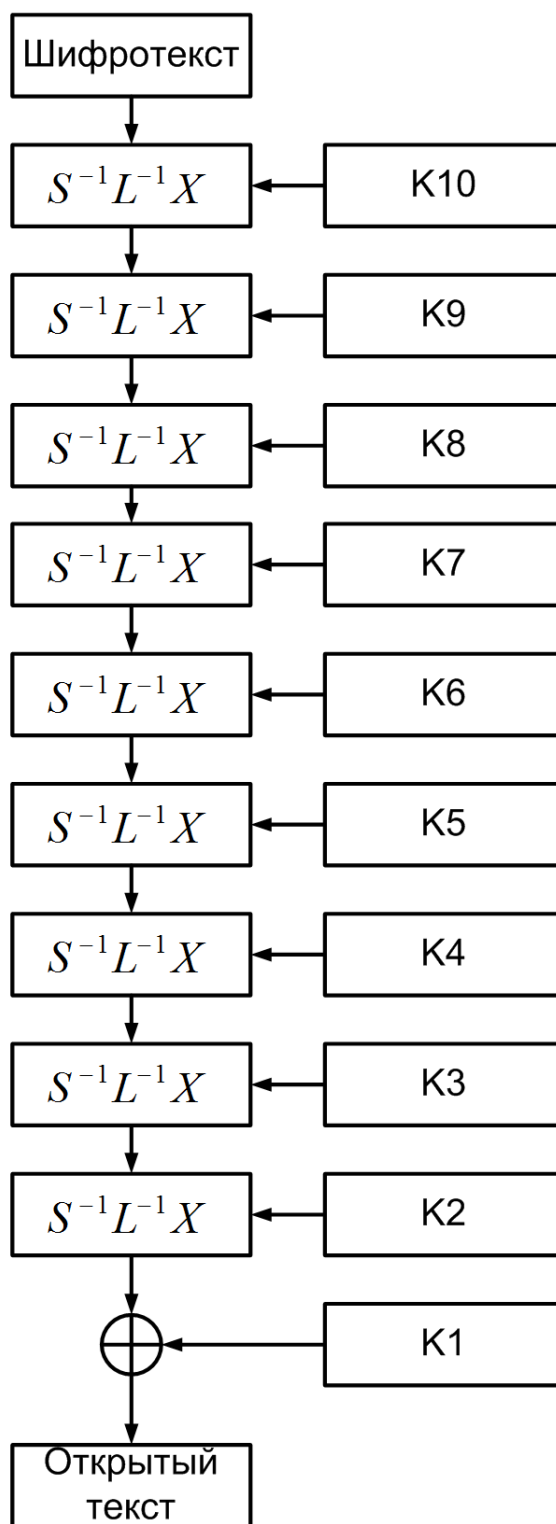


Рисунок 17 – Алгоритм расшифрования

Если открыть файл не получилось, программа выводит оператору ошибку, представленную на рисунке 18.

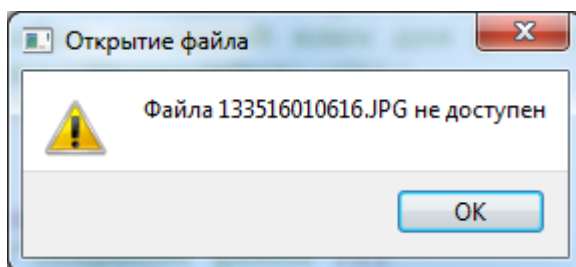


Рисунок 18 – Ошибка открытия файла

4.9 Класс VerificationThread

Данная функция запускается с открытием главного окна и осуществляет проверку целостности и доступности файлов на НЖМД. Реализация класса представлена в приложении И.

Функция verification проверяет соответствие количества файлов на сервера и на НЖМД. Входными параметрами являются:

- список снимков, хранимых на сервере резервного копирования;
- список журналов, хранимых на сервер резервного копирования.

Для каждого снимка извлекается информация из ложных таблиц квантования. Ложные таблицы удаляются, в результате восстанавливается исходный файл. Далее выполняется проверка подписи по открытому ключу. Ключа считывается из журнала открытых ключей по отметке.

Для каждого журнала подпись и отметка извлекаются из конца файла. Далее выполняется проверка подписи по открытому ключу. Ключа считывается из журнала открытых ключей по отметке.

При несоответствии подписи или при отсутствии файла на жестком магнитном диске выводится сообщение, представленное на рисунке 18. Файл не прошедший проверку восстанавливается с сервера резервного копирования.

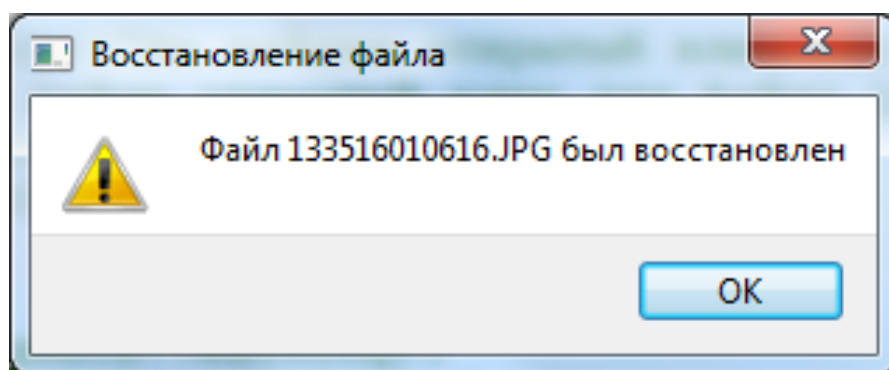


Рисунок 18 – Сообщение о восстановлении файла

В случае если открытый ключ не найден в журнале. Оператору выводится сообщение, представленное на рисунке 19. Возможно, был изменен журнал, или проверяемый файл. Оператору необходимо восстановить журнал открытых ключей с сервера резервного копирования.

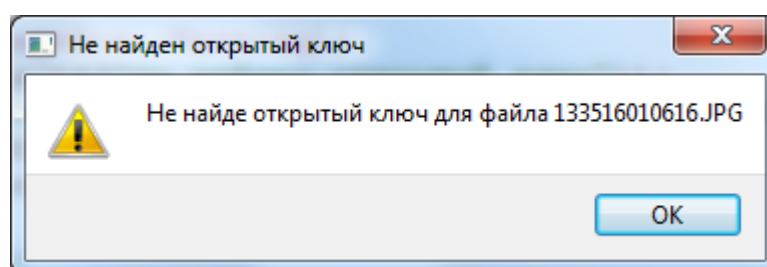


Рисунок 19 – Сообщение об отсутствии открытого ключа для файла

В случае если проверяемый файл не открывается выводится сообщение об ошибке представленное на рисунке 20.

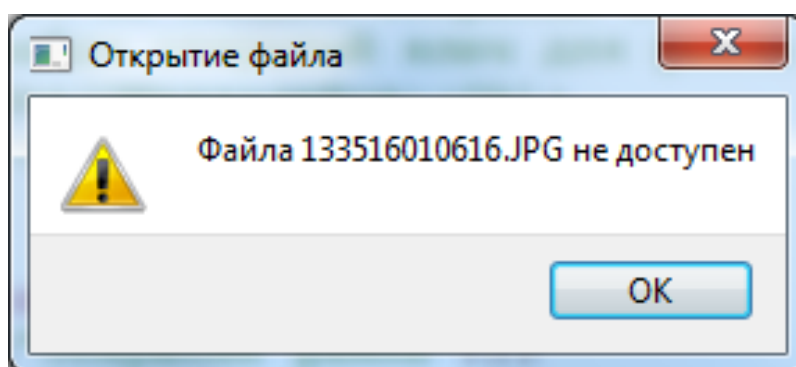


Рисунок 20 – Сообщение об ошибке открытия файла

5 Тестирование разработанной программы

На этапе тестирования класс `Nash3411`, был запущен с тестовыми данными указанными в ГОСТ 34.10-12.

Необходимо было вычислить хэш-код сообщения M1

M1=323130393837363534333231303938373635343332313039383736353
433323130393837363534333231303938373635343332313039383736353433323
130 .

Хэш-кодом сообщения М1 является значение

H(M1)=486f64c1917879417fef082b3381a4e211c324f074654c38823a7b76f830ad00fa1fbae42b1285c0352f227524bc9ab16254288dd6863dccd5b9f54a1ad0541b.

Результат выполнения тестовых данных сошелся с указанным в ГОСТ 34.10-12.

На этапе тестирования класс Sign3410, был запущен с тестовыми данными указанными в ГОСТ 34.11-12.

Параметры схемы цифровой подписи:

— модуль эллиптической кривой

p=57896044618658097711785492504343953926634992332820282019
728792003956 5648 21041

— коэффициенты эллиптической кривой

$$a = 7$$

b = 433088765467672769057659045956509319959421117944510395832
5296884203 3849580414

— порядок группы точек эллиптической кривой

m=57896044618658097711785492504343953927082934583725450622380
97359213763106 9619

— порядок циклической подгруппы группы точек эллиптической кривой

q=57896044618658097711785492504343953927082934583725450622380
97359213 7631069 619

— коэффициенты точки эллиптической кривой

В данном примере координаты точки Р принимают следующие значения:

$$x = 2$$

y=40189740565390375033354494229370597756357393899055450806909
793652 1343156 6280

В данном примере считается, что пользователь обладает следующим ключом подписи d :

d=55441196065363246126355624130324183196576709222340016572108
097750006 097525544

В данном примере считается, что пользователь обладает ключом проверки подписи Q, координаты которого имеют следующие значения:

xq=5752021612617680844363140502333807117663010490631363218289
674134220 6604859403

yq=1761494441921378154380939194965408003194266204536363926070
98478594 382867 63994

При проведении теста условие $R = r$ было выполнено.

На этапе тестирования класс Cipher3412, был запущен с тестовыми данными указанными в ГОСТ 34.12-15

Было выполнено преобразование S

S(ffeeddccbbaa99881122334455667700)=b66cd8887d38e8d77765aeea0c9a7efc,S(b66cd8887d38e8d77765aeea0c9a7efc)=559d8dd7bd06cbfe7e7b262523280d39,S(559d8dd7bd06cbfe7e7b262523280d39)=0c3322fed531e4630d80ef5c5a81c50b,S(0c3322fed531e4630d80ef5c5a81c50b)=23ae65633f842d29c5df529c13f5acda.

Было выполнено преобразование R

[illegible][illegible]

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта были решены следующие задачи:

- произведен анализ предметной области: местоположение приложения в системе, анализ потоков с которыми работает приложение;
- произведена разработка модели угроз для разрабатываемого приложения;
- произведен анализ и выбор защитных мер для защиты от актуальных угроз, которые могут быть использованы для реализации программного средства защиты целостности и доступности визуальной и текстовой информации,
- произведена программная реализация выбранных защитных мер;
- произведено тестирование разработанного программного средства.

В результате дипломного проектирования все пункты технического задания были выполнены, было разработано программное средство защиты целостности и доступности визуальной и текстовой информации.

Результат выполнения работы используется в рабочем процессе АО НПП «Рубин», что подтверждено актом о внедрении.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Авиационный комплекс радиобнаружения и наведения [Электронный ресурс]. – Режим доступа свободный: https://ru.wikipedia.org/wiki/Авиационный_комплекс_радиобнаружения_и_наведения;
2. Ключевые особенности Astra Linux Special Edition по реализации требований безопасности информации [Электронный ресурс]. – Режим доступа свободный: <http://www.astralinux.com/klyuchevye-osobennosti.html>;
3. Windows 7: новые возможности и перспективы применения [Электронный ресурс]. – Режим доступа свободный: <http://compress.ru/article.aspx?id=20888>;
4. Методы сокрытия информации в графических изображениях [Электронный ресурс]. – Режим доступа свободный: http://ru.bmstu.wiki/Методы_сокрытия_информации_в_графических_изображениях;
5. ГОСТ Р 34.11-2012 "Информационная технология. Криптографическая защита информации. Функция хэширования";
6. Криптография с использованием эллиптических кривых [Электронный ресурс] – Режим доступа свободный: <http://www.intuit.ru/studies/courses/28/28/lecture/20430>;
7. ГОСТ Р 34.10-2012 "Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи";
8. ГОСТ Р 34.12-2015 "Информационная технология. Криптографическая защита информации. Блочные шифры".

ПРИЛОЖЕНИЕ А

Листинг класса MainWindow

Листинг А.1 Заголовочный файл класса MainWindow:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTimer>
#include "hash3411.h"
#include "ecpoint.h"
#include "BigIntegerLibrary.hh"

#include <iostream>
#include "string.h"

#include "sign3410.h"
#include "cipher3412.h"
#include "verificationthread.h"
#include "stegothread.h"
#include <QThread>
#include <QFtp>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private:
    Ui::MainWindow *ui;
    QTimer timer;
    Hash3411 hash;
    Sign3410 sign;
    QString mark; //<Метка сессии
    QString puthScreen;
    QString puthTextLog;
    QString puthServiceLog;
    VerificationThread *objectThread;
    QThread *threadVerification;
    QFtp* ftp;
    QList <QString> mSavedFilePerSession;
    QList <QString> mListServerScreen;
    QList <QString> mListServerLog;
    bool isListScreen;
    StegoThread* objStegoThread1;
    StegoThread* objStegoThread2;
    StegoThread* objStegoThread3;

    QThread *threadStego1;
    QThread *threadStego2;
    QThread *threadStego3;

    QFile *fileCopyServer;
    int nNumberScreenSavedPerSession;
```

```

    QFile *fileLog;

    int isCopyFile;

    enum numberCopyFile
    {
        nFileScreen,
        nFileLog,
        nFileMark
    };
    void resizeEvent(QResizeEvent* event);

public slots:
    void slotGoStopInterval(bool);
    void slotInterval(bool);
    void screen();
    void showScreenViewer();
    void showLogViewer();
    void ftpCommandFinished(int,bool);
    void addToList(const QUrlInfo &urlInfo);
    void slotExit();

signals:
    void screenshotSaved(QString);
    void sigStego1 (QString,QString,QString);
    void sigStego2 (QString,QString,QString);
    void sigStego3 (QString,QString,QString);

    void sigVerifivation (QList<QString>,QList<QString>);
    void closeW();
};

#endif // MAINWINDOW_H

```

Листинг А.2. Исходный файл класса MainWindow:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDesktopWidget>
#include <QDateTime>
#include <QFile>
#include <QDebug>
#include <QMessageBox>
#include <ctime>

#include "screenviewerwidget.h"
#include "ui_screenviewerwidget.h"

#include "logviewerwidget.h"
#include "ui_logviewerwidget.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setUpUi(this);
    connect(ui->Button_Go_One, SIGNAL(clicked()), this, SLOT(screen()));
    connect(&timer, SIGNAL(timeout()), this, SLOT(screen()));
    connect(ui->groupBoxInterval, SIGNAL(toggled(bool)), this, SLOT(slotInterval(bool)));
    connect(ui->Button_Go_Interval, SIGNAL(toggled(bool)), this, SLOT(slotGoStopInterval(bool)));
    connect(ui->buttonScreenViewer, SIGNAL(clicked()), this, SLOT(showScreenViewer()));
    connect(ui->buttonLogViewer, SIGNAL(clicked()), this, SLOT(showLogViewer()));
    connect(ui->buttonExit, SIGNAL(clicked()), this, SLOT(slotExit()));
    connect(this, SIGNAL(closeW()), QApplication, SLOT(closeAllWindows()));

    connect(this, SIGNAL(screenshotSaved(QString)), ScreenViewerWidget::instance(),
        SLOT(slotCurrentScreenChanged(QString)));
    this->setWindowFlags(Qt::Dialog | Qt::CustomizeWindowHint |
        Qt::WindowTitleHint | Qt::WindowSystemMenuHint);

    //пути
    puthScreen = "e://file//screen//";
    puthTextLog = "e://file//log//";
    puthServiceLog = "e://file//service//serviceLog.txt";

    //генерация секретного и открытого ключа
    ui->Button_Go_One->setEnabled(false);
    sign.genKeys();
    ui->Button_Go_One->setEnabled(true);
    //-----
    //генерация отметки сессии
    QDateTime dateTime = QDateTime::currentDateTime();
    mark = dateTime.toString("ddMMyyhhmmss");
    QFile fileServiceLog(puthServiceLog);
    if (!fileServiceLog.open(QIODevice::Append)) {

    }
    QTextStream strText(&fileServiceLog);
    strText << "Mark=" << mark
        << "Qx=" <<
    QString::fromStdString(BigIntegerToString(sign.getQx()))
}
```

```

        << "Qy="<<
QString::fromStdString(bigIntegerToString(sign.getQy()));
fileServiceLog.close();
//обнуляем список файлов сделанных скриншотов за сессию
mSavedFilePerSession.clear();
fileLog = new QFile(puthTextLog+mark+".log");
if (!fileLog->open(QIODevice::ReadWrite)) /*ошибка*/return;
QByteArray writeLog;
writeLog.append("Масштаб 200");
writeLog.append('\r');
writeLog.append('\n');
writeLog.append("Установлен режим 1");
writeLog.append('\r');
writeLog.append('\n');
writeLog.append("Слой карта включен");
writeLog.append('\r');
writeLog.append('\n');
writeLog.append("Слой носитель включен");
writeLog.append('\r');
writeLog.append('\n');
writeLog.append("Слой отметки включен");
fileLog->write(writeLog);
fileLog->close();////////////////////////

//5 портоков stego
objStegoThread1 = new StegoThread;
objStegoThread2 = new StegoThread;
objStegoThread3 = new StegoThread;

threadStego1 = new QThread(this);
threadStego2 = new QThread(this);
threadStego3 = new QThread(this);

objStegoThread1->setNumThread(1);
objStegoThread2->setNumThread(2);
objStegoThread3->setNumThread(3);

connect(this,
SIGNAL(sigStego1(QString,QString,QString)),objStegoThread1,SLOT(stego(QString
,QString,QString)));
connect(this,
SIGNAL(sigStego2(QString,QString,QString)),objStegoThread2,SLOT(stego(QString
,QString,QString)));
connect(this,
SIGNAL(sigStego3(QString,QString,QString)),objStegoThread3,SLOT(stego(QString
,QString,QString)));

objStegoThread1->moveToThread(threadStego1);
threadStego1->start();

objStegoThread2->moveToThread(threadStego2);
threadStego2->start();

objStegoThread3->moveToThread(threadStego3);
threadStego3->start();

//проверка скриншотов в новый поток
objectThread = new VerificationThread;
threadVerification = new QThread(this);
qRegisterMetaType< QList<QString> >("QList<QString>");
connect(this, SIGNAL(sigVerifivation(QList<QString>,QList<QString>)),
        objectThread,SLOT(verification(QList<QString>,QList<QString>)));
connect(objectThread,SIGNAL(setProgress(int)),

```

```

        ui->progressBar,SLOT(setValue(int)));
        objectThread->moveToThread(threadVerification);
        threadVerification->start();
//-----
        //FTP
        ftp = new QFtp(this);
        connect(ftp, SIGNAL(commandFinished(int,bool)),
                this, SLOT(ftpCommandFinished(int,bool)));
        connect(ftp, SIGNAL(listInfo(QUrlInfo)), this, SLOT(addToList(QUrlInfo)));
        ftp->connectToHost("192.168.0.4", 21); //id=1
        ftp->login("user", "1"); //id=2
        mListServerScreen.clear();
        isListScreen = true;
        ftp->list("screen"); //id=3
        mListServerLog.clear();
        ftp->list("log"); //id=4
//-----

}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::resizeEvent(QResizeEvent *event)
{
    QSize sizeWidget = size();
    int h = sizeWidget.height();
    int w = sizeWidget.width();

    if ((h>300) && (w>700))
    {
        //кнопки просмотра
        ui->buttonScreenViewer->move(w-200, h-300);
        ui->buttonLogViewer->move(w-200, h-230);
        //объекты проверки
        ui->label_Text->move(w-200, h-100);
        ui->progressBar->move(w-200, h-70);
        //кнопки скриншотов
        ui->Button_Go_One->move(10, h-150);
        ui->groupBoxInterval->move(200, h-150);
    }
}

void MainWindow::slotGoStopInterval(bool isEnabled)
{
    if (isEnabled) timer.start(ui->spinBox->value());
    else timer.stop();
}

void MainWindow::slotInterval(bool isEnabled)
{
    if (isEnabled)
    {
        ui->Button_Go_One->setEnabled(false);
    }
    else
    {
        ui->Button_Go_One->setEnabled(true);
        timer.stop();
    }
}

```

```

    }
}

void MainWindow::screen()
{
    //-----screen-----
    unsigned int start_time = clock(); // начальное время

    //делаем скриншот
    QImage image;
    image = QPixmap::grabWindow(QApplication::desktop()->winId()).toImage();

    //сохраняем с названием даты и времени
    QDateTime dateTime = QDateTime::currentDateTime();
    QString str = dateTime.toString("ddMMyy-hhmmss");
    str+=".JPG";
    image.save(puthScreen+str,"JPG");

    unsigned int end_time = clock(); // конечное время
    unsigned int search_time = end_time - start_time; // искомое время
    qDebug() << "time do screen=" << search_time;

    emit screenshotSaved(str);
    mSavedFilePerSession.push_back(str);

    if (objStegoThread1->isRunning())
    {
        if (objStegoThread2->isRunning())
        {
            if (objStegoThread3->isRunning())
            {
            }
            else
            {
                emit
sigStego3(str,QString::fromStdString(bigIntegerToString(sign.getD())),mark);
                return;
            }
        }
        else
        {
            emit
sigStego2(str,QString::fromStdString(bigIntegerToString(sign.getD())),mark);
            return;
        }
    }
    else
    {
        emit
sigStego1(str,QString::fromStdString(bigIntegerToString(sign.getD())),mark);
        return;
    }
}

void MainWindow::showScreenViewer()
{
    ScreenViewerWidget::instance()->setVisible(true);
}

void MainWindow::showLogViewer()
{

```

```

        LogViewerWidget::instance()->setVisible(true);
    }

void MainWindow::ftpCommandFinished(int id, bool error)
{
    if (ftp->currentCommand() == QFtp::ConnectToHost)
    {
        if (error)
        {
            QMessageBox::information(this, "FTP","Unable to connect to the
FTP server ");
            return;
        }
    }
    if (ftp->currentCommand() == QFtp::Login)
    {
        if (error)
        {
            QMessageBox::information(this, tr("FTP"), tr("Unable to login to
the FTP server"));
            return;
        }
    }

    if (ftp->currentCommand() == QFtp::List)
    {
        isListScreen = false;
        if (id == 4)
        {
            emit sigVerifivation(mListServerScreen,
                                mListServerLog);
        }
    }
    if (ftp->currentCommand() == QFtp::Get)
    {
        if (error) {
            QMessageBox::information(this, tr("FTP"), tr("Canceled download
file of %1.").arg(fileCopyServer->fileName()));
            fileCopyServer->close();
            fileCopyServer->remove();
        } else {
            fileCopyServer->close();
        }
        delete fileCopyServer;
    }
    if (ftp->currentCommand() == QFtp::Put)
    {
        if (error) {
            QMessageBox::information(this, tr("FTP"), tr("Canceled download
filt to server of %1.").arg(fileCopyServer->fileName()));
            fileCopyServer->close();
        } else
        {
            nNumberScreenSavedPerSession--;
            if (nNumberScreenSavedPerSession > 0)
            {
                mSavedFilePerSession.removeFirst();
                fileCopyServer->close();
                fileCopyServer = new
QFile(puthScreen+mSavedFilePerSession.first());
                ftp->put(fileCopyServer,mSavedFilePerSession.first());
            }
            if (nNumberScreenSavedPerSession == 0)

```

```

        {
            isCopyFile = nFileLog;
            ftp->cd("/log");
            //посылаем лог
            fileCopyServer->close();
        }
        if (nNumberScreenSavedPerSession == -1)
        {
            isCopyFile = nFileMark;
            ftp->cd("/service");
            //посылаем лог
            fileCopyServer->close();
        }
        if (nNumberScreenSavedPerSession == -2)
        {
            //fileCopyServer->close();
            fileLog->close();
            ftp->close();

            threadStego1->quit();
            threadStego1->wait();
            delete threadStego1;

            threadStego2->quit();
            threadStego2->wait();
            delete threadStego2;

            threadStego3->quit();
            threadStego3->wait();
            delete threadStego3;

            emit closeW();
        }
    }
}
if (ftp->currentCommand() == QFtp::Cd)
{
    if (isCopyFile = nFileScreen)
    {
        fileCopyServer = new
QFile(puthScreen+mSavedFilePerSession.first());
        ftp->put(fileCopyServer,mSavedFilePerSession.first());
    }
    if (isCopyFile = nFileLog)
    {
        ftp->put(fileLog,mark+".log");
    }
    if (isCopyFile = nFileMark)
    {
        QFile fileMark (puthServiceLog);
        //ftp->put(fileMark,"service"+mark+".log");
    }
}
}

void MainWindow::addToList(const QUrlInfo &urlInfo)
{
    if (isListScreen)
    {
        mListServerScreen.append(urlInfo.name());
    }
    else
    {

```

```

        mListServerLog.append(urlInfo.name());
    }
}

void MainWindow::slotExit()
{
    nNumberScreenSavedPerSession = mSavedFilePerSession.length();
    if (mSavedFilePerSession.isEmpty())
    {
        isCopyFile = nFileLog;
        ftp->cd("/log");
    }
    else
    {
        isCopyFile = nFileScreen;
        ftp->cd("/screen");
    }
}

```

ПРИЛОЖЕНИЕ Б

Листинг класса ScreenViewerWidget

Листинг Б.1 Заголовочный файл класса ScreenViewerWidget:

```
#ifndef SCREENVIEWERWIDGET_H
#define SCREENVIEWERWIDGET_H

#include <QWidget>
#include <QListWidgetItem>
#include <QLabel>

namespace Ui {
class ScreenViewerWidget;
}

class ScreenViewerWidget : public QWidget
{
    Q_OBJECT

private:
    ScreenViewerWidget(QWidget *parent = 0, Qt::WindowFlags f=0);
    ~ScreenViewerWidget();

    Ui::ScreenViewerWidget *ui;
    QLabel* imageLabel;
    QString screensDirectory;

    static ScreenViewerWidget* mInstance;

    QString puthScreen;
    QString puthTextLog;
    QString puthServiceLog;

    void resizeEvent(QResizeEvent* event);

public:
    static ScreenViewerWidget* instance();
    void refresh();

public slots:
    void slotCurrentScreenChanged(QString);
private slots:
    void slotOnItemChanged(QListWidgetItem*,QListWidgetItem*);
    void slotDeleteAll();
    void slotDeleteCurrent();
};

#endif // SCREENVIEWERWIDGET_H
```

Листинг Б.2 Исходный файл класса ScreenViewerWidget:

```
#include "screenviewerwidget.h"
#include "ui_screenviewerwidget.h"

#include <QFile>
#include <QDir>
#include <QDateTime>
#include <QMessageBox>

class QScreenListWidgetItem : public QListWidgetItem
{
public:
    QScreenListWidgetItem(const QString &text, const QString &logFileName)
        :QListWidgetItem(text),screenFileName(logFileName) {}
    QString screenFileName;
};

ScreenViewerWidget* ScreenViewerWidget::mInstance = NULL;

ScreenViewerWidget::ScreenViewerWidget(QWidget *parent, Qt::WindowFlags f) :
    QWidget(parent, f | Qt::WindowStaysOnTopHint),
    ui(new Ui::ScreenViewerWidget)
{
    ui->setupUi(this);
    imageLabel = new QLabel(ui->scrollAreaScreenshot);
    ui->scrollAreaScreenshot->setWidget(imageLabel);

    //setScreenDir();

    connect(ui->buttonDeleteScreenshot,SIGNAL(clicked()),this,SLOT(slotDeleteCurrent()));
    connect(ui->buttonDeleteAll,SIGNAL(clicked()),this,SLOT(slotDeleteAll()));
    connect(ui->listWidget,SIGNAL(currentItemChanged(QListWidgetItem*,QListWidgetItem*)),
            this,SLOT(slotOnItemChanged(QListWidgetItem*,QListWidgetItem*)));

    //пути
    puthScreen = "e://file//screen//";
    puthTextLog = "e://file//log//";
    puthServiceLog = "e://file//service//serviceLog.txt";

    refresh();
}

ScreenViewerWidget::~ScreenViewerWidget()
{
    delete ui;
    delete imageLabel;
}

void ScreenViewerWidget::resizeEvent(QResizeEvent *event)
{
    QSize sizeWidget = size();
    int h = sizeWidget.height();
    int w = sizeWidget.width();

    ui->scrollAreaScreenshot->setGeometry(200,0,w-200,h);

    if (h>100)
    {
```

```

        ui->listWidget->setGeometry(10,0,175,h-100);
        ui->buttonDeleteScreenshot->move(10,h-80);
        ui->buttonDeleteAll->move(10,h-40);
    }
}

ScreenViewerWidget *ScreenViewerWidget::instance()
{
    if (mInstance == NULL)
    {
        mInstance = new ScreenViewerWidget();
    }
    return mInstance;
}

void ScreenViewerWidget::refresh()
{
    QDir dir(puthScreen);
    dir.setFilter(QDir::Files);
    dir.setSorting(QDir::Time | QDir::Reversed);
    dir.setNameFilters(QStringList(QString("*.JPG")));
    ui->listWidget->clear();
    QFileInfoList list = dir.entryInfoList();
    QList<QFileInfo>::Iterator it;
    for (it=list.begin(); it!=list.end(); ++it)
    {
        slotCurrentScreenChanged((*it).fileName());
    }
    ui->listWidget->scrollToBottom();
}

void ScreenViewerWidget::slotCurrentScreenChanged(QString msg)
{
    QScreenListWidgetItem* shooterItem = new
    QScreenListWidgetItem(msg,puthScreen+msg);
    ui->listWidget->addItem(shooterItem);
}

void ScreenViewerWidget::slotOnItemChanged(QListWidgetItem *curr,
QListWidgetItem */*prev*/)
{
    QScreenListWidgetItem* currItem = (QScreenListWidgetItem*)curr;
    if (!currItem) return;

    QDir dir("");
    if(!dir.exists(currItem->screenFileName))
    {
        qWarning("ScreenViewerWidget::slotOnItemChanged: cannot open
screenfile=%s", currItem->screenFileName.toUtf8().data());
    }
    QPixmap pixmap(currItem->screenFileName);
    imageLabel->setPixmap(pixmap);
}

void ScreenViewerWidget::slotDeleteAll()
{
    QDir dir(puthScreen);
    dir.setFilter(QDir::Files);
    dir.setSorting(QDir::Time | QDir::Reversed);
    dir.setNameFilters(QStringList(QString("*.JPG")));
    QFileInfoList list = dir.entryInfoList();

```

```

        QList<QFileInfo>::Iterator it;
        for (it=list.begin(); it!=list.end(); ++it)
        {
            QFile((*it).fileName()).remove();
        }
    }

void ScreenViewerWidget::slotDeleteCurrent()
{
    QScreenListWidgetItem* currItem = (QScreenListWidgetItem*)curr;
    if (!currItem) return;

    QFile(currItem->screenFileName).remove();
}

```

ПРИЛОЖЕНИЕ В

Листинг класса LogViewerWidget

Листинг В.1 Заголовочный файл класса LogViewerWidget:

```
#ifndef LOGVIEWERWIDGET_H
#define LOGVIEWERWIDGET_H

#include <QWidget>
#include <QListWidgetItem>
#include <QTextBrowser>

namespace Ui {
class LogViewerWidget;
}

class LogViewerWidget : public QWidget
{
    Q_OBJECT

private:
    explicit LogViewerWidget(QWidget *parent = 0, Qt::WindowFlags f=0);
    ~LogViewerWidget();

    Ui::LogViewerWidget *ui;
    QTextBrowser* logText;
    QString logDirectory;

    static LogViewerWidget* mInstance;
    QString puthScreen;
    QString puthTextLog;
    QString puthServiceLog;
public:
    static LogViewerWidget* instance();
    void refresh();

public slots:
    void slotCurrentScreenChanged(QString);
private slots:
    void slotOnItemChanged(QListWidgetItem*,QListWidgetItem*);
    void slotDeleteAll();
    void slotDeleteCurrent();
};

#endif // LOGVIEWERWIDGET_H
```

Листинг В.1 Исходный файл класса LogViewerWidget:

```
#include "logviewerwidget.h"
#include "ui_logviewerwidget.h"

#include <QFile>
#include <QDir>
#include <QDateTime>
#include <QMessageBox>

class QLogListWidgetItem : public QListWidgetItem
{
public:
    QLogListWidgetItem(const QString &text, const QString &fileName)
        :QListWidgetItem(text),logFileName(fileName) {}
    QString logFileName;
};

LogViewerWidget* LogViewerWidget::mInstance = NULL;

LogViewerWidget::LogViewerWidget(QWidget *parent, Qt::WindowFlags f) :
    QWidget(parent, f | Qt::WindowStaysOnTopHint),
    ui(new Ui::LogViewerWidget)
{
    ui->setupUi(this);
    logText = new QTextBrowser(ui->scrollArea);
    ui->scrollArea->setWidget(logText);

    connect(ui->buttonDel,SIGNAL(clicked()),this,SLOT(slotDeleteCurrent()));
    connect(ui->buttonDelAll,SIGNAL(clicked()),this,SLOT(slotDeleteAll()));
    connect(ui->listWidget,SIGNAL(currentItemChanged(QListWidgetItem*,QListWidgetItem*)),
        this,SLOT(slotOnItemChanged(QListWidgetItem*,QListWidgetItem*)));
    //пути
    puthScreen = "e://file//screen//";
    puthTextLog = "e://file//log//";
    puthServiceLog = "e://file//service//serviceLog.txt";

    refresh();
}

LogViewerWidget::~LogViewerWidget()
{
    delete ui;
    delete logText;
}

LogViewerWidget *LogViewerWidget::instance()
{
    {
        if (mInstance == NULL)
        {
            mInstance = new LogViewerWidget();
        }
        return mInstance;
    }
}

void LogViewerWidget::refresh()
{
    QDir dir(puthTextLog);
    dir.setFilter(QDir::Files);
    dir.setSorting(QDir::Time | QDir::Reversed);
    dir.setNameFilters(QStringList(QString("*.log")));
    ui->listWidget->clear();
```

```

        QFileInfoList list = dir.entryInfoList();
        QList<QFileInfo>::Iterator it;
        for (it=list.begin(); it!=list.end(); ++it)
        {
            slotCurrentScreenChanged((*it).fileName());
        }
        ui->listWidget->scrollToBottom();
    }

void LogViewerWidget::slotCurrentScreenChanged(QString msg)
{
    QLogListWidgetItem* shooterItem = new
    QLogListWidgetItem(msg, puthTextLog+msg);
    ui->listWidget->addItem(shooterItem);
}

void LogViewerWidget::slotOnItemChanged(QListWidgetItem *curr,
QListWidgetItem */*prev*/)
{
    QLogListWidgetItem* currItem = (QLogListWidgetItem*)curr;
    if (!currItem) return;

    QDir dir("");
    if(!dir.exists(currItem->logFileName))
    {
        qWarning("LogViewerWidget::slotOnItemChanged: cannot open
logfile=%s",
                currItem->logFileName.toUtf8().data());
    }
    QFile file(currItem->logFileName);

    if (!file.open(QIODevice::ReadWrite)) /*ошибка*/return;
    QByteArray data = file.readAll();
    file.close();
    logText->setText((QString(data)));
}

void LogViewerWidget::slotDeleteAll()
{
    QDir dir(puthTextLog);
    dir.setFilter(QDir::Files);
    dir.setSorting(QDir::Time | QDir::Reversed);
    dir.setNameFilters(QStringList(QString("*.log")));
    QFileInfoList list = dir.entryInfoList();
    QList<QFileInfo>::Iterator it;
    for (it=list.begin(); it!=list.end(); ++it)
    {
        QFile((*it).fileName()).remove();
    }
}

void LogViewerWidget::slotDeleteCurrent()
{
    QLogListWidgetItem* currItem = (QLogListWidgetItem*)curr;
    if (!currItem) return;

    QFile(currItem->screenFileName).remove();
}

```

ПРИЛОЖЕНИЕ Г

Листинг класса Hash3411

Листинг Г.1 Заголовочный файл класса Hash3411:

```
#ifndef HASH3411_H
#define HASH3411_H
//#include <QObject>

class Hash3411
{
public:
    Hash3411();
    void addModulo512(const unsigned char *a, const unsigned char *b,
unsigned char *c);
    void addXor512(const void *a, const void *b, void *c);
    void S(unsigned char *state);
    void L(unsigned char *state);
    void P(unsigned char *state);
    void keySchedule(unsigned char *K, int i);
    void E(unsigned char *K, const unsigned char *m, unsigned char *state);
    void g_N(const unsigned char *N, unsigned char *h, const unsigned char
*m);
    void hash_X(unsigned char *IV, const unsigned char *message, unsigned
long long length, unsigned char *out);
    void hash_512(const unsigned char *message, unsigned long long length,
unsigned char *out);
    void hash_256(const unsigned char *message, unsigned long long length,
unsigned char *out);
    void print_hex(unsigned char a[], int s);

private:
    static const unsigned long long a[64]; //бинарная матрица a
    static const unsigned char subs[256]; // Таблица подстановок subs
    static const unsigned char tau[64]; // Таблица перестановок tau
    static const unsigned char c[12][64]; // Итерационные константы C

//public slots:
    // void hash_512(const unsigned char *message, unsigned long long length,
unsigned char *out);
};

#endif // HASH3411_H
```

Листинг Г.2 Исходный файл класса Hash3411:

```
#include "hash3411.h"
#include <string.h>
#include <QFile>
const unsigned long long Hash3411::a[64]= {
0x8e20faa72ba0b470, 0x47107ddd9b505a38, 0xad08b0e0c3282d1c,
0xd8045870ef14980e,
0x6c022c38f90a4c07, 0x3601161cf205268d, 0x1b8e0b0e798c13c8,
0x83478b07b2468764,
0xa011d380818e8f40, 0x5086e740ce47c920, 0x2843fd2067adea10,
0x14aff010bdd87508,
0x0ad97808d06cb404, 0x05e23c0468365a02, 0x8c711e02341b2d01,
0x46b60f011a83988e,
0x90dab52a387ae76f, 0x486dd4151c3dfdb9, 0x24b86a840e90f0d2,
0x125c354207487869,
0x092e94218d243cba, 0x8a174a9ec8121e5d, 0x4585254f64090fa0,
0xacc9ca9328a8950,
0x9d4df05d5f661451, 0xc0a878a0a1330aa6, 0x60543c50de970553,
0x302a1e286fc58ca7,
0x18150f14b9ec46dd, 0x0c84890ad27623e0, 0x0642ca05693b9f70,
0x0321658cba93c138,
0x86275df09ce8aaa8, 0x439da0784e745554, 0xafc0503c273aa42a,
0xd960281e9d1d5215,
0xe230140fc0802984, 0x71180a8960409a42, 0xb60c05ca30204d21,
0x5b068c651810a89e,
0x456c34887a3805b9, 0xac361a443d1c8cd2, 0x561b0d22900e4669,
0x2b838811480723ba,
0x9bcf4486248d9f5d, 0xc3e9224312c8c1a0, 0xeffa11af0964ee50,
0xf97d86d98a327728,
0xe4fa2054a80b329c, 0x727d102a548b194e, 0x39b008152acb8227,
0x9258048415eb419d,
0x492c024284fbaec0, 0xaa16012142f35760, 0x550b8e9e21f7a530,
0xa48b474f9ef5dc18,
0x70a6a56e2440598e, 0x3853dc371220a247, 0x1ca76e95091051ad,
0x0edd37c48a08a6d8,
0x07e095624504536c, 0x8d70c431ac02a736, 0xc83862965601dd1b,
0x641c314b2b8ee083};
const unsigned char Hash3411::subs[256]= {
0xFC, 0xEE, 0xDD, 0x11, 0xCF, 0x6E, 0x31, 0x16, 0xFB, 0xC4, 0xFA, 0xDA,
0x23, 0xC5, 0x04, 0x4D,
0xE9, 0x77, 0xF0, 0xDB, 0x93, 0x2E, 0x99, 0xBA, 0x17, 0x36, 0xF1, 0xBB,
0x14, 0xCD, 0x5F, 0xC1,
0xF9, 0x18, 0x65, 0x5A, 0xE2, 0x5C, 0xEF, 0x21, 0x81, 0x1C, 0x3C, 0x42,
0x8B, 0x01, 0x8E, 0x4F,
0x05, 0x84, 0x02, 0xAE, 0xE3, 0x6A, 0x8F, 0xA0, 0x06, 0x0B, 0xED, 0x98,
0x7F, 0xD4, 0xD3, 0x1F,
0xEB, 0x34, 0x2C, 0x51, 0xEA, 0xC8, 0x48, 0xAB, 0xF2, 0x2A, 0x68, 0xA2,
0xFD, 0x3A, 0xCE, 0xCC,
0xB5, 0x70, 0x0E, 0x56, 0x08, 0x0C, 0x76, 0x12, 0xBF, 0x72, 0x13, 0x47,
0x9C, 0xB7, 0x5D, 0x87,
0x15, 0xA1, 0x96, 0x29, 0x10, 0x7B, 0x9A, 0xC7, 0xF3, 0x91, 0x78, 0x6F,
0x9D, 0x9E, 0xB2, 0xB1,
0x32, 0x75, 0x19, 0x3D, 0xFF, 0x35, 0x8A, 0x7E, 0x6D, 0x54, 0xC6, 0x80,
0xC3, 0xBD, 0x0D, 0x57,
0xDF, 0xF5, 0x24, 0xA9, 0x3E, 0xA8, 0x43, 0xC9, 0xD7, 0x79, 0xD6, 0xF6,
0x7C, 0x22, 0xB9, 0x03,
0xE0, 0x0F, 0xEC, 0xDE, 0x7A, 0x94, 0xB0, 0xBC, 0xDC, 0xE8, 0x28, 0x50,
0x4E, 0x33, 0x0A, 0x4A,
0xA7, 0x97, 0x60, 0x73, 0x1E, 0x00, 0x62, 0x44, 0x1A, 0xB8, 0x38, 0x82,
0x64, 0x9F, 0x26, 0x41,
0xAD, 0x45, 0x46, 0x92, 0x27, 0x5E, 0x55, 0x2F, 0x8C, 0xA3, 0xA5, 0x7D,
0x69, 0xD5, 0x95, 0x3B,
```

```

    0x07, 0x58, 0xB3, 0x40, 0x86, 0xAC, 0x1D, 0xF7, 0x30, 0x37, 0x6B, 0xE4,
    0x88, 0xD9, 0xE7, 0x89,
    0xE1, 0x1B, 0x83, 0x49, 0x4C, 0x3F, 0xF8, 0xFE, 0x8D, 0x53, 0xAA, 0x90,
    0xCA, 0xD8, 0x85, 0x61,
    0x20, 0x71, 0x67, 0xA4, 0x2D, 0x2B, 0x09, 0x5B, 0xCB, 0x9B, 0x25, 0xD0,
    0xBE, 0xE5, 0x6C, 0x52,
    0x59, 0xA6, 0x74, 0xD2, 0xE6, 0xF4, 0xB4, 0xC0, 0xD1, 0x66, 0xAF, 0xC2,
    0x39, 0x4B, 0x63, 0xB6};
const unsigned char Hash3411::tau[64]={
    0,  8, 16, 24, 32, 40, 48, 56,
    1,  9, 17, 25, 33, 41, 49, 57,
    2, 10, 18, 26, 34, 42, 50, 58,
    3, 11, 19, 27, 35, 43, 51, 59,
    4, 12, 20, 28, 36, 44, 52, 60,
    5, 13, 21, 29, 37, 45, 53, 61,
    6, 14, 22, 30, 38, 46, 54, 62,
    7, 15, 23, 31, 39, 47, 55, 63};
const unsigned char Hash3411::c[12][64] = {
    {

0xb1,0x08,0x5b,0xda,0x1e,0xca,0xda,0xe9,0xeb,0xcb,0x2f,0x81,0xc0,0x65,0x7c,0x
1f,

0x2f,0x6a,0x76,0x43,0x2e,0x45,0xd0,0x16,0x71,0x4e,0xb8,0x8d,0x75,0x85,0xc4,0x
fc,

0x4b,0x7c,0xe0,0x91,0x92,0x67,0x69,0x01,0xa2,0x42,0x2a,0x08,0xa4,0x60,0xd3,0x
15,

0x05,0x76,0x74,0x36,0xcc,0x74,0x4d,0x23,0xdd,0x80,0x65,0x59,0xf2,0xa6,0x45,0x
07
    },
    {

0x6f,0xa3,0xb5,0x8a,0xa9,0x9d,0x2f,0x1a,0x4f,0xe3,0x9d,0x46,0x0f,0x70,0xb5,0x
d7,

0xf3,0xfe,0xea,0x72,0x0a,0x23,0x2b,0x98,0x61,0xd5,0x5e,0x0f,0x16,0xb5,0x01,0x
31,

0x9a,0xb5,0x17,0x6b,0x12,0xd6,0x99,0x58,0x5c,0xb5,0x61,0xc2,0xdb,0x0a,0xa7,0x
ca,

0x55,0xdd,0xa2,0x1b,0xd7,0xcb,0xcd,0x56,0xe6,0x79,0x04,0x70,0x21,0xb1,0x9b,0x
b7
    },
    {

0xf5,0x74,0xdc,0xac,0x2b,0xce,0x2f,0xc7,0x0a,0x39,0xfc,0x28,0x6a,0x3d,0x84,0x
35,

0x06,0xf1,0x5e,0x5f,0x52,0x9c,0x1f,0x8b,0xf2,0xea,0x75,0x14,0xb1,0x29,0x7b,0x
7b,

0xd3,0xe2,0x0f,0xe4,0x90,0x35,0x9e,0xb1,0xc1,0xc9,0x3a,0x37,0x60,0x62,0xdb,0x
09,

0xc2,0xb6,0xf4,0x43,0x86,0x7a,0xdb,0x31,0x99,0x1e,0x96,0xf5,0x0a,0xba,0x0a,0x
b2
    },
    {

```

```

0xef, 0x1f, 0xdf, 0xb3, 0xe8, 0x15, 0x66, 0xd2, 0xf9, 0x48, 0xe1, 0xa0, 0x5d, 0x71, 0xe4, 0x
dd,

0x48, 0x8e, 0x85, 0x7e, 0x33, 0x5c, 0x3c, 0x7d, 0x9d, 0x72, 0x1c, 0xad, 0x68, 0x5e, 0x35, 0x
3f,

0xa9, 0xd7, 0x2c, 0x82, 0xed, 0x03, 0xd6, 0x75, 0xd8, 0xb7, 0x13, 0x33, 0x93, 0x52, 0x03, 0x
be,

0x34, 0x53, 0xea, 0xa1, 0x93, 0xe8, 0x37, 0xf1, 0x22, 0x0c, 0xbe, 0xbc, 0x84, 0xe3, 0xd1, 0x
2e
    },
    {

0x4b, 0xea, 0x6b, 0xac, 0xad, 0x47, 0x47, 0x99, 0x9a, 0x3f, 0x41, 0x0c, 0x6c, 0xa9, 0x23, 0x
63,

0x7f, 0x15, 0x1c, 0x1f, 0x16, 0x86, 0x10, 0x4a, 0x35, 0x9e, 0x35, 0xd7, 0x80, 0x0f, 0xff, 0x
bd,

0xbf, 0xcd, 0x17, 0x47, 0x25, 0x3a, 0xf5, 0xa3, 0xdf, 0xff, 0x00, 0xb7, 0x23, 0x27, 0x1a, 0x
16,

0x7a, 0x56, 0xa2, 0x7e, 0xa9, 0xea, 0x63, 0xf5, 0x60, 0x17, 0x58, 0xfd, 0x7c, 0x6c, 0xfe, 0x
57
    },
    {

0xae, 0x4f, 0xae, 0xae, 0x1d, 0x3a, 0xd3, 0xd9, 0x6f, 0xa4, 0xc3, 0x3b, 0x7a, 0x30, 0x39, 0x
c0,

0x2d, 0x66, 0xc4, 0xf9, 0x51, 0x42, 0xa4, 0x6c, 0x18, 0x7f, 0x9a, 0xb4, 0x9a, 0xf0, 0x8e, 0x
c6,

0xcf, 0xfa, 0xa6, 0xb7, 0x1c, 0x9a, 0xb7, 0xb4, 0x0a, 0xf2, 0x1f, 0x66, 0xc2, 0xbe, 0xc6, 0x
b6,

0xbf, 0x71, 0xc5, 0x72, 0x36, 0x90, 0x4f, 0x35, 0xfa, 0x68, 0x40, 0x7a, 0x46, 0x64, 0x7d, 0x
6e
    },
    {

0xf4, 0xc7, 0x0e, 0x16, 0xee, 0xaa, 0xc5, 0xec, 0x51, 0xac, 0x86, 0xfe, 0xbf, 0x24, 0x09, 0x
54,

0x39, 0x9e, 0xc6, 0xc7, 0xe6, 0xbf, 0x87, 0xc9, 0xd3, 0x47, 0x3e, 0x33, 0x19, 0x7a, 0x93, 0x
c9,

0x09, 0x92, 0xab, 0xc5, 0x2d, 0x82, 0x2c, 0x37, 0x06, 0x47, 0x69, 0x83, 0x28, 0x4a, 0x05, 0x
04,

0x35, 0x17, 0x45, 0x4c, 0xa2, 0x3c, 0x4a, 0xf3, 0x88, 0x86, 0x56, 0x4d, 0x3a, 0x14, 0xd4, 0x
93
    },
    {

0x9b, 0x1f, 0x5b, 0x42, 0x4d, 0x93, 0xc9, 0xa7, 0x03, 0xe7, 0xaa, 0x02, 0x0c, 0x6e, 0x41, 0x
41,

0x4e, 0xb7, 0xf8, 0x71, 0x9c, 0x36, 0xde, 0x1e, 0x89, 0xb4, 0x44, 0x3b, 0x4d, 0xdb, 0xc4, 0x
9a,

```

```

0xf4,0x89,0x2b,0xcb,0x92,0x9b,0x06,0x90,0x69,0xd1,0x8d,0x2b,0xd1,0xa5,0xc4,0x
2f,

0x36,0xac,0xc2,0x35,0x59,0x51,0xa8,0xd9,0xa4,0x7f,0x0d,0xd4,0xbf,0x02,0xe7,0x
1e
    },
    {

0x37,0x8f,0x5a,0x54,0x16,0x31,0x22,0x9b,0x94,0x4c,0x9a,0xd8,0xec,0x16,0x5f,0x
de,

0x3a,0x7d,0x3a,0x1b,0x25,0x89,0x42,0x24,0x3c,0xd9,0x55,0xb7,0xe0,0x0d,0x09,0x
84,

0x80,0x0a,0x44,0x0b,0xdb,0xb2,0xce,0xb1,0x7b,0x2b,0x8a,0x9a,0xa6,0x07,0x9c,0x
54,

0x0e,0x38,0xdc,0x92,0xcb,0x1f,0x2a,0x60,0x72,0x61,0x44,0x51,0x83,0x23,0x5a,0x
db
    },
    {

0xab,0xbe,0xde,0xa6,0x80,0x05,0x6f,0x52,0x38,0x2a,0xe5,0x48,0xb2,0xe4,0xf3,0x
f3,

0x89,0x41,0xe7,0x1c,0xff,0x8a,0x78,0xdb,0x1f,0xff,0xe1,0x8a,0x1b,0x33,0x61,0x
03,

0x9f,0xe7,0x67,0x02,0xaf,0x69,0x33,0x4b,0x7a,0x1e,0x6c,0x30,0x3b,0x76,0x52,0x
f4,

0x36,0x98,0xfa,0xd1,0x15,0x3b,0xb6,0xc3,0x74,0xb4,0xc7,0xfb,0x98,0x45,0x9c,0x
ed
    },
    {

0x7b,0xcd,0x9e,0xd0,0xef,0xc8,0x89,0xfb,0x30,0x02,0xc6,0xcd,0x63,0x5a,0xfe,0x
94,

0xd8,0xfa,0x6b,0xbb,0xeb,0xab,0x07,0x61,0x20,0x01,0x80,0x21,0x14,0x84,0x66,0x
79,

0x8a,0x1d,0x71,0xef,0xea,0x48,0xb9,0xca,0xef,0xba,0xcd,0x1d,0x7d,0x47,0x6e,0x
98,

0xde,0xa2,0x59,0x4a,0xc0,0x6f,0xd8,0x5d,0x6b,0xca,0xa4,0xcd,0x81,0xf3,0x2d,0x
1b
    },
    {

0x37,0x8e,0xe7,0x67,0xf1,0x16,0x31,0xba,0xd2,0x13,0x80,0xb0,0x04,0x49,0xb1,0x
7a,

0xcd,0xa4,0x3c,0x32,0xbc,0xdf,0x1d,0x77,0xf8,0x20,0x12,0xd4,0x30,0x21,0x9f,0x
9b,

0x5d,0x80,0xef,0x9d,0x18,0x91,0xcc,0x86,0xe7,0x1d,0xa4,0xaa,0x88,0xe1,0x28,0x
52,

0xfa,0xf4,0x17,0xd5,0xd9,0xb2,0x1b,0x99,0x48,0xbc,0x92,0x4a,0xf1,0x1b,0xd7,0x
20
    }

```

```

    };
Hash3411::Hash3411()
{
}
/*-----
    Сложение по модулю 2^512.
-----*/
void Hash3411::addModulo512(const unsigned char *a, const unsigned char
*b, unsigned char *c)
{
    int i = 0, t = 0;

    for(i = 63; i >= 0; i--)
    {
        t = a[i] + b[i] + (t >> 8);
        c[i] = t & 0xFF;
    }
}
/*-----
    X-преобразование. На вход функции X подаются две
    последовательности длиной 512 бит каждая, выходом функции
    является XOR этих последовательностей.
-----*/
void Hash3411::addXor512(const void *a, const void *b, void *c)
{
    int i = 0;
    const long long *A = (const long long*)a, *B = (const long long*)b;
    unsigned long long *C = (unsigned long long*)c;

    for(i = 0; i < 8; i++)
    {
        C[i] = A[i] ^ B[i];
    }
}
/*-----
    S-преобразование. Функция S является обычной функцией подстановки.
    Каждый байт из 512-битной входной последовательности заменяется
    соответствующим байтом из таблицы подстановок Subs.
-----*/
void Hash3411::S(unsigned char *state)
{
    int i = 0;

    for(i = 0; i < 64; i++)
    {
        state[i] = subs[state[i]];
    }
}
/*-----
    L-преобразование. Представляет собой умножение 64-битного входного
    вектора на бинарную матрицу A размерами 64x64.
-----*/
void Hash3411::L(unsigned char *state)
{
    unsigned long long v = 0;
    int i = 0, j = 0, k = 0;

    for(i = 0; i < 8; i++)
    {
        v = 0;
        for(k = 0; k < 8; k++)
        {
            for(j = 0; j < 8; j++)

```

```

        {
            if ((state[i*8+k] & (1<<(7-j))) != 0)
                v ^= a[k*8+j];
        }
    }
    for(k = 0; k < 8; k++)
    {
        state[i * 8 + k] = (v & ((unsigned long long)0xFF << (7 - k) *
8)) >> (7 - k) * 8;
    }
}
}
/*-----
P-преобразование. Функция перестановки. Для каждой пары байт из входной
последовательности происходит замена одного байта другим.
-----*/
void Hash3411::P(unsigned char *state)
{
    int i = 0;
    unsigned char t[64];

    for(i = 0; i < 64; i++)
    {
        t[i] = state[tau[i]];
    }

    memcpy(state,t,64);
}
/*-----
Функция KeySchedule(K, i), отвечающая за формирование временного ключа K
на каждом раунде функции E(K, m).
-----*/
void Hash3411::keySchedule(unsigned char *K, int i)
{
    addXor512(K, c[i], K);

    S(K);
    P(K);
    L(K);
}

void Hash3411::E(unsigned char *K, const unsigned char *m, unsigned char
*state)
{
    int i = 0;

    memcpy(K, K, 64);

    addXor512(m, K, state);

    for(i = 0; i < 12; i++)
    {
        S(state);
        P(state);
        L(state);
        keySchedule(K, i);
        addXor512(state, K, state);
    }
}
/*-----
Функция сжатия.
-----*/

```

```

void Hash3411::g_N(const unsigned char *N, unsigned char *h, const unsigned
char *m)
{
    unsigned char t[64], K[64];

    addXor512(N,h,K);

    S(K);
    P(K);
    L(K);

    E(K,m,t);

    addXor512(t,h,t);
    addXor512(t,m,h);
}
/*-----
    Вычисление хеш-функции
-----*/
void Hash3411::hash_X(unsigned char *IV, const unsigned char *message,
unsigned long long length, unsigned char *out)
{
    unsigned char v512[64] = {

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x
00

    };
    unsigned char v0[64] = {

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00

    };
    unsigned char Sigma[64] = {

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00
    }
}

```

```

};
unsigned char N[64] = {
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00
};
unsigned char m[64], *hash = IV;
unsigned long long len = length;

// Стадия 2
while (len >= 512)
{
    memcpy(m, message + len/8 - 63 - ( (len & 0x7) == 0 ), 64);

    g_N(N,hash,m);
    addModulo512(N,v512,N);
    addModulo512(Sigma,m,Sigma);
    len -= 512;
}

memset(m,0,64);
memcpy(m + 63 - len/8 + ( (len & 0x7) == 0 ), message, len/8 + 1 - ( (len
& 0x7) == 0 ));

// Стадия 3
m[ 63 - len/8 ] |= (1 << (len & 0x7));

g_N(N, hash, m);
v512[63] = len & 0xFF;
v512[62] = (unsigned char)(len >> 8);
addModulo512(N, v512, N);

addModulo512(Sigma, m, Sigma);

g_N(v0, hash, N);
g_N(v0, hash, Sigma);

memcpy(out, hash, 64);
}
/*-----
Вычисление 512-битной хеш-функции
-----*/
void Hash3411::hash_512(const unsigned char *message, unsigned long long
length, unsigned char *out)
{
    unsigned char IV[64] = {

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,

```

```

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x
00
    };

    hash_X(IV, message, length, out);
}
/*-----
    Вычисление 256-битной хеш-функции
-----*/
void Hash3411::hash_256(const unsigned char *message, unsigned long long
length, unsigned char *out)
{
    unsigned char IV[64] = {

0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x
01,

0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x
01,

0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x
01,

0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x
01
    };
    unsigned char hash[64];

    hash_X(IV, message, length, hash);

    memcpy(out, hash, 32);
}

```

ПРИЛОЖЕНИЕ Д

Листинг класса ECPPoint

Листинг Д.1 Заголовочный файл класса ECPPoint:

```
#ifndef ECPPOINT_H
#define ECPPOINT_H
#include "BigIntegerLibrary.hh"

class ECPPoint
{
public:
    ECPPoint();
    ECPPoint(ECPPoint &p);
    //сложение двух точек P1 и P2
    friend const ECPPoint operator +( ECPPoint p1,  ECPPoint p2);
    //сложение точки P с собой же
    static ECPPoint doublePoint (ECPPoint p);
    //умножение точки на число x, по сути своей представляет x сложений точки
    //самой с собой
    static ECPPoint multiply(BigInteger x, ECPPoint p);

    //параметры
    BigInteger x;
    BigInteger y;
    BigInteger a;
    BigInteger b;
    BigInteger fieldChar;

};

#endif // ECPPOINT_H
```

Листинг В.2 Исходный файл ECPPoint:

```
#include "ecpoint.h"

ECPPoint::ECPPoint()
{

}

ECPPoint::ECPPoint(ECPPoint &p)
{

    x=p.x;
    y=p.y;
    a=p.a;
    b=p.b;
    fieldChar=p.fieldChar;

}

const ECPPoint operator+( ECPPoint p1,  ECPPoint p2)
{

    ECPPoint p3;

    p3.a = p1.a;
    p3.b = p1.b;
    p3.fieldChar = p1.fieldChar;

    BigInteger dy,dx;
    dy = p2.y - p1.y;
    dx = p2.x - p1.x;

    if (dx < 0 /*BigInteger(0)*/) dx += p1.fieldChar;
    if (dy < 0 /*BigInteger(0)*/) dy += p1.fieldChar;

    BigInteger m;

    std::string str =  bigIntegerToString(p1.fieldChar);
    BigUnsigned n =  stringToBigUnsigned(str);

    m = (dy * modinv(dx,n))% p1.fieldChar;

    if (m < BigInteger(0))
        m += p1.fieldChar;
    p3.x = (m * m - p1.x - p2.x) % p1.fieldChar;
    p3.y = (m * (p1.x - p3.x) - p1.y) % p1.fieldChar;
    if (p3.x < 0/*BigInteger(0)*/)
        p3.x += p1.fieldChar;
    if (p3.y < 0/*BigInteger(0)*/)
        p3.y += p1.fieldChar;
    return p3;

}

ECPPoint ECPPoint::doublePoint(ECPPoint p)
{
    ECPPoint p2;

    p2.a=p.a;
    p2.b = p.b;
    p2.fieldChar = p.fieldChar;
```

```

    BigInteger dy,dx;

    dy = p.x*3 /*BigInteger(3)*/ * p.x + p.a;
    dx = p.y*2 /*BigInteger(2)*/;

    if (dx < 0/*BigInteger(0)*/)
        dx += p.fieldChar;
    if (dy < 0/*BigInteger(0)*/)
        dy += p.fieldChar;

    BigInteger m;
    std::string str = BigIntegerToString(p.fieldChar);
    BigInteger n = stringToBigInteger(str);

    m = (dy * modinv(dx,n))% p.fieldChar;
    p2.x = (m*m - p.x - p.x) % p.fieldChar;
    p2.y = (m*(p.x - p2.x) - p.y) % p.fieldChar;
    if (p2.x < 0/*BigInteger(0)*/)
        p2.x += p.fieldChar;
    if (p2.y < 0/*BigInteger(0)*/)
        p2.y += p.fieldChar;

    return p2;
}

ECPoint ECPoint::multiply(BigInteger x, ECPoint p)
{
    ECPoint temp = p;

    x = x - 1/*BigInteger(1)*/;

    while (x != 0/*BigInteger(0)*/)
    {
        if ((x % 2/*BigInteger(2)*/) != 0/*BigInteger(0)*/)
        {
            if ((temp.x == p.x) || (temp.y == p.y))
                temp = doublePoint(temp);
            else
                temp = temp + p;
            x = x - 1/*BigInteger(1)*/;
        }
        x = x / 2 /*BigInteger(2)*/;
        p = doublePoint(p);
    }
    return temp;
}

```

ПРИЛОЖЕНИЕ Е

Листинг класса Sign3410

Листинг Е.1 Заголовочный файл класса Sign3410:

```
#ifndef SIGN3410_H
#define SIGN3410_H
#include "BigIntegerLibrary.hh"
#include "ecpoint.h"

#include <string.h>

class Sign3410
{
public:
    Sign3410();
    BigInteger vectorToBigInteger(unsigned char vector[], unsigned int len);

    std::string signGen(unsigned char hash[]);

    BigInteger randomLess(BigInteger Max);

    std::string toHexString(BigInteger x);

    bool signVer(unsigned char hash[], std::string strSing);

    void genKeys();

    void setPublicKey(BigInteger Qx, BigInteger Qy);
    void setPrivateKey(BigInteger privateKey) {d = privateKey;}
    //для проверки
    BigInteger getD() {return d;}
    BigInteger getQx() {return pointQ.x;}
    BigInteger getQy() {return pointQ.y;}
    //-----

    //Параметры
private:
    //простое число p модуль эллиптической кривой
    BigInteger p;
    //коэффициенты эллиптической кривой E
    BigInteger a;
    BigInteger b;
    //порядок группы точек эллиптической кривой
    BigInteger m;
    //порядок циклической подгруппы точек эллиптической кривой
    BigInteger q;
    //точка P эллиптической кривой E
    BigInteger Px;
    BigInteger Py;
    ECPoint pointP;
    //ключ подписи d
    BigInteger d;
    //ключ проверки подписи
    BigInteger Qx;
    BigInteger Qy;
    ECPoint pointQ;
};

#endif // SIGN3410_H
```

Листинг Г.2 Исходный файл класса Sign3410:

```
#include "sign3410.h"
#include <QDebug>
#include <ctime>
#include <sstream>

Sign3410::Sign3410()
{
    //простое число p модуль эллиптической кривой
    p =
    stringToBigInteger("362398610222900363590778875368387430602132092553467860508
65461504508561666240024825884820222714968540250908236030587351637342638223719
64987228582907372403");
    //коэффициенты эллиптической кривой E
    a = 7;
    b =
    stringToBigInteger("151865506921082853450895003471404315492874752774020643619
40188233528099824437937328297569147859746748660416053978836775966263264139901
36959047435811826396");
    //порядок группы точек эллиптической кривой
    m =
    stringToBigInteger("362398610222900363590778875368387430602132092553467860508
65461504508561666239691648983050328630684999614040794379365854558651922129707
34808812618120619743");
    //порядок циклической подгруппы точек эллиптической кривой
    q =
    stringToBigInteger("362398610222900363590778875368387430602132092553467860508
65461504508561666239691648983050328630684999614040794379365854558651922129707
34808812618120619743");
    //координаты точки P эллиптической кривой E
    Px =
    stringToBigInteger("192835694406702284939930940124313759899778663545950797435
70754913077665926858354410655576810031848748196580049032123328842523358302507
29527632383493573274");
    Py =
    stringToBigInteger("228872869337197285997001215552947841635356232732950618031
44974259311028603015728141419970722717088070665938506503341523818573477988858
64807605098724013854");
    //параметры точки P
    pointP.a = a;
    pointP.b = b;
    pointP.x = Px;
    pointP.y = Py;
    pointP.fieldChar = p;
}

//длина в байтах
//только вектор с длиной делящейся на 4
BigInteger Sign3410::vectorToBigInteger(unsigned char vector[], unsigned int
len)
{
    int dataLength = len/4;

    BigInteger::Blk *blocks = new BigInteger::Blk[dataLength];

    for (int i = len-1,j=0; i>=0 ; i-=4, j++)
    {
        //blocks[j] = 1;
        blocks[j] = ( vector[i-3] << 24 )+( vector[i-2] << 16 )+
            ( vector[i-1] << 8 )+( vector[i] );
    }
}
```

```

        BigInteger result (blocks,dataLength, BigInteger::Sign::positive);
        return result;
    }
    //для хеша 512
    std::string Sign3410::signGen(unsigned char hash[])
    {
        BigInteger alpha = vectorToBigInteger(hash,64);
        BigInteger e;
        e = alpha % q;
        if (e == BigInteger(0))
            e = BigInteger(1);

        BigInteger k;
        ECPoint C;
        BigInteger r;
        BigInteger s;
        do {
            k = randomLess(q);

            C = C.multiply(k,pointP);

            r = C.x % q;
            //if (r == BigInteger(0)) signGen(hash,len); //по ГОСТу нужно
            //вернуться к пункту расчета k
            s = ((r * d) + (k * e)) % q;
            } while ((r == 0) || (s == 0));
            //if (s == BigInteger(0)) signGen(hash,len); //по ГОСТу нужно вернуться к
            //пункту расчета k

            std::string rVector = toHexString(r);

            std::string sVector = toHexString(s);

            return rVector + sVector;
        }

    BigInteger Sign3410::randomLess(BigInteger Max)
    {
        //получить число блоков
        unsigned int lengthMax = Max.getLength();
        srand (time(0));
        //рандомное число блоков
        int lengthBlocks = 1+rand() % lengthMax;
        //максимальное число в блоке unsigned int
        unsigned int uiMax = 0xffffffff;
        BigInteger::Blk *blocks = new BigInteger::Blk[lengthBlocks];
        for (int i=0 ; i<lengthBlocks; i++)
        {
            srand (time(0));
            //рандомное число в блоке
            blocks[i]=rand() % uiMax;
        }
        BigInteger result (blocks,lengthBlocks, BigInteger::Sign::positive);
        //Проверка
        while (result > Max) {
            blocks[lengthBlocks-1] = blocks[lengthBlocks-1] >> 1;
            BigInteger temp (blocks,lengthBlocks, BigInteger::Sign::positive);
            result = temp;
        }

        return result;
    }
}

```

```

std::string Sign3410::toHexString(BigInteger x)
{
    unsigned int length = x.getLength();
    BigUnsigned blk;
    std::string result;
    for (int i=length-1 ; i>=0; i--)
    {
        blk=x.getBlock(i);
        unsigned long num = blk.toUnsignedLong();
        std::stringstream stream;
        stream << std::hex << num;
        std::string res(stream.str());

        if (res.length() < 8) {
            for (int j= res.length(); j<8; j++) {
                std::string null="0";
                res = null+res;
            }
        }
        result += res;
    }
    return result;
}

bool Sign3410::signVer(unsigned char hash[], std::string strSing)
{
    //BigInteger r
    std::string rString = strSing.substr(0,128);
    QString qstr = QString::fromStdString(rString);
    unsigned char rVector[64];
    for (int i=0;i<128;i=i+2)
    {
        QString byte = qstr.mid(i,2);
        bool bStatus = false;
        int num = byte.toUInt(&bStatus,16);
        if(bStatus) {
            rVector[i/2] = num;
        }
        else qDebug() << "Errorr string to char";
    }
    BigInteger r = vectorToBigInteger(rVector,64);
    //BigInteger s
    std::string sString = strSing.substr(128);
    qstr = QString::fromStdString(sString);
    unsigned char sVector[64];
    for (int i=0;i<128;i=i+2)
    {
        QString byte = qstr.mid(i,2);
        bool bStatus = false;
        int num = byte.toUInt(&bStatus,16);
        if(bStatus) {
            sVector[i/2] = num;
        }
        else qDebug() << "Errorr string to char";
    }
    BigInteger s = vectorToBigInteger(sVector,64);

    if ((r < 1) || (r > (q - 1)) || (s < 1) || (s > (q - 1))) return
false;

    BigInteger alpha = vectorToBigInteger(hash,64);
    BigInteger e;

```

```

    e = alpha % q;
    if (e == BigInteger(0))
        e = BigInteger(1);
    BigInteger v;
    std::string str = BigIntegerToString(q);
    BigUnsigned n = stringToBigUnsigned(str);
    v=modinv(e,n);

    BigInteger z1;
    z1 = (s * v) % q;

    BigInteger z2;
    z2 = -(r * v) % q;

    ECPoint pointC;
    ECPoint pointA;
    ECPoint pointB;

    pointA = pointA.multiply(z1,pointP);
    pointB = pointB.multiply(z2,pointQ);
    pointC = pointA + pointB;

    BigInteger R;
    R = pointC.x % q;
    if ( R - r == 0) return true;
    else return false;
}

void Sign3410::genKeys()
{
    //генерируем секретный ключ
    BigInteger privateKey;
    privateKey = randomLess(q);
    d = privateKey;
    //генерируем открытый ключ
    ECPoint pointPublicKey;
    pointPublicKey = pointPublicKey.multiply(privateKey,pointP);
    pointQ.a = a;
    pointQ.b = b;
    pointQ.x = pointPublicKey.x;
    pointQ.y = pointPublicKey.y;
    pointQ.fieldChar = p;
}

void Sign3410::setPublicKey(BigInteger Qx, BigInteger Qy)
{
    pointQ.a = a;
    pointQ.b = b;
    pointQ.x = Qx;
    pointQ.y = Qy;
    pointQ.fieldChar = p;
}

```

ПРИЛОЖЕНИЕ Ж

Листинг класса Cipher3412

Листинг Ж.1 Заголовочный файл класса Cipher3412:

```
#ifndef CIPHER3412_H
#define CIPHER3412_H

class Cipher3412
{
public:
    Cipher3412();
    void S(unsigned long long int * block);
    void S_inv(unsigned long long int * block);
    unsigned char ll(unsigned long long int block[2]);
    void R(unsigned long long int * block);
    void R_inv(unsigned long long int * block);
    void L(unsigned long long int * block);
    void L_inv(unsigned long long int * block);
    void F(unsigned long long int * k, unsigned long long int * block1,
unsigned long long int * block0);
    void key_setup(unsigned long long int * master_key, unsigned long long
int * * subkeys);
    void encrypt(unsigned long long int * * subkeys, unsigned long long int *
plaintext, unsigned long long int * ciphertext);
    void decrypt(unsigned long long int * * subkeys, unsigned long long int *
ciphertext, unsigned long long int * plaintext);

private:
    static const unsigned char pi[256]; // Таблица подстановок
    static const unsigned char pi_inv[256]; // Таблица подстановок^-1
    /* Precomputation of  $X \cdot Y \bmod p(x) = x^8 + x^7 + x^6 + x + 1$  where X and Y
are elements in  $GF(2)[X]/p(x)$  */
    static const unsigned char mult_mod_poly[256][256];
};

#endif // CIPHER3412_H
```

Листинг Ж.2 Исходный файл класса Cipher3412:

```
#include "cipher3412.h"
const unsigned char Cipher3412::pi[256] = {252, 238, 221, 17, 207, 110, 49,
22, 251, 196, 250, 218, 35, 197, 4, 77,
233, 119, 240, 219, 147, 46, 153,
186, 23, 54, 241, 187, 20, 205, 95, 193,
249, 24, 101, 90, 226, 92, 239,
33, 129, 28, 60, 66, 139, 1, 142, 79,
5, 132, 2, 174, 227, 106, 143,
160, 6, 11, 237, 152, 127, 212, 211, 31,
235, 52, 44, 81, 234, 200, 72,
171, 242, 42, 104, 162, 253, 58, 206, 204,
181, 112, 14, 86, 8, 12, 118,
18, 191, 114, 19, 71, 156, 183, 93, 135,
21, 161, 150, 41, 16, 123, 154,
199, 243, 145, 120, 111, 157, 158, 178, 177,
50, 117, 25, 61, 255, 53, 138,
126, 109, 84, 198, 128, 195, 189, 13, 87,
223, 245, 36, 169, 62, 168, 67,
201, 215, 121, 214, 246, 124, 34, 185, 3,
224, 15, 236, 222, 122, 148, 176,
188, 220, 232, 40, 80, 78, 51, 10, 74,
167, 151, 96, 115, 30, 0, 98,
68, 26, 184, 56, 130, 100, 159, 38, 65,
173, 69, 70, 146, 39, 94, 85,
47, 140, 163, 165, 125, 105, 213, 149, 59,
7, 88, 179, 64, 134, 172, 29,
247, 48, 55, 107, 228, 136, 217, 231, 137,
225, 27, 131, 73, 76, 63, 248,
254, 141, 83, 170, 144, 202, 216, 133, 97,
32, 113, 103, 164, 45, 43, 9,
91, 203, 155, 37, 208, 190, 229, 108, 82,
89, 166, 116, 210, 230, 244, 180,
192, 209, 102, 175, 194, 57, 75, 99, 182};

const unsigned char Cipher3412::pi_inv[256] = {165, 45, 50, 143, 14, 48,
56, 192, 84, 230, 158, 57, 85, 126, 82, 145,
100, 3, 87, 90, 28, 96,
7, 24, 33, 114, 168, 209, 41, 198, 164, 63,
224, 39, 141, 12, 130, 234,
174, 180, 154, 99, 73, 229, 66, 228, 21, 183,
200, 6, 112, 157, 65, 117,
25, 201, 170, 252, 77, 191, 42, 115, 132, 213,
195, 175, 43, 134, 167, 177,
178, 91, 70, 211, 159, 253, 212, 15, 156, 47,
155, 67, 239, 217, 121, 182,
83, 127, 193, 240, 35, 231, 37, 94, 181, 30,
162, 223, 166, 254, 172, 34,
249, 226, 74, 188, 53, 202, 238, 120, 5, 107,
81, 225, 89, 163, 242, 113,
86, 17, 106, 137, 148, 101, 140, 187, 119, 60,
123, 40, 171, 210, 49, 222,
196, 95, 204, 207, 118, 44, 184, 216, 46, 54,
219, 105, 179, 20, 149, 190,
98, 161, 59, 22, 102, 233, 92, 108, 109, 173,
55, 97, 75, 185, 227, 186,
241, 160, 133, 131, 218, 71, 197, 176, 51, 250,
150, 111, 110, 194, 246, 80,
255, 93, 169, 142, 23, 27, 151, 125, 236, 88,
247, 31, 251, 124, 9, 13,
122, 103, 69, 135, 220, 232, 79, 29, 78, 4,
```

```

        235, 248, 243, 62, 61, 189,
138, 136, 221, 205, 11, 19, 152, 2, 147, 128,
        144, 208, 36, 52, 203, 237,
244, 206, 153, 16, 68, 64, 146, 58, 1, 38,
        18, 26, 72, 104, 245, 129,
139, 199, 214, 32, 10, 8, 0, 76, 215, 116};
Cipher3412::Cipher3412()
{
}
/*****
/* Base and conversion functions */
*****/

/* S conversion function */
void Cipher3412::S(unsigned long long *block)
{
    unsigned long long int valtmp;

    valtmp = ((unsigned long long int)(pi[(block[0] >> 56) & 0xFF]) << 56);
    valtmp |= ((unsigned long long int)(pi[(block[0] >> 48) & 0xFF]) << 48);
    valtmp |= ((unsigned long long int)(pi[(block[0] >> 40) & 0xFF]) << 40);
    valtmp |= ((unsigned long long int)(pi[(block[0] >> 32) & 0xFF]) << 32);
    valtmp |= ((unsigned long long int)(pi[(block[0] >> 24) & 0xFF]) << 24);
    valtmp |= ((unsigned long long int)(pi[(block[0] >> 16) & 0xFF]) << 16);
    valtmp |= ((unsigned long long int)(pi[(block[0] >> 8) & 0xFF]) << 8);
    valtmp |= (unsigned long long int)(pi[block[0] & 0xFF]);

    block[0] = valtmp;

    valtmp = ((unsigned long long int)(pi[(block[1] >> 56) & 0xFF]) << 56);
    valtmp |= ((unsigned long long int)(pi[(block[1] >> 48) & 0xFF]) << 48);
    valtmp |= ((unsigned long long int)(pi[(block[1] >> 40) & 0xFF]) << 40);
    valtmp |= ((unsigned long long int)(pi[(block[1] >> 32) & 0xFF]) << 32);
    valtmp |= ((unsigned long long int)(pi[(block[1] >> 24) & 0xFF]) << 24);
    valtmp |= ((unsigned long long int)(pi[(block[1] >> 16) & 0xFF]) << 16);
    valtmp |= ((unsigned long long int)(pi[(block[1] >> 8) & 0xFF]) << 8);
    valtmp |= (unsigned long long int)(pi[block[1] & 0xFF]);

    block[1] = valtmp;
}

/* Inverse of S conversion function */
void Cipher3412::S_inv(unsigned long long int * block)
{
    unsigned long long int valtmp;

    valtmp = ((unsigned long long int)(pi_inv[(block[0] >> 56) & 0xFF]) <<
56);
    valtmp |= ((unsigned long long int)(pi_inv[(block[0] >> 48) & 0xFF]) <<
48);
    valtmp |= ((unsigned long long int)(pi_inv[(block[0] >> 40) & 0xFF]) <<
40);
    valtmp |= ((unsigned long long int)(pi_inv[(block[0] >> 32) & 0xFF]) <<
32);
    valtmp |= ((unsigned long long int)(pi_inv[(block[0] >> 24) & 0xFF]) <<
24);
    valtmp |= ((unsigned long long int)(pi_inv[(block[0] >> 16) & 0xFF]) <<
16);
    valtmp |= ((unsigned long long int)(pi_inv[(block[0] >> 8) & 0xFF]) <<
8);
    valtmp |= (unsigned long long int)(pi_inv[block[0] & 0xFF]);

```

```

    block[0] = valtmp;

    valtmp = ((unsigned long long int) (pi_inv[(block[1] >> 56) & 0xFF]) <<
56);
    valtmp |= ((unsigned long long int) (pi_inv[(block[1] >> 48) & 0xFF]) <<
48);
    valtmp |= ((unsigned long long int) (pi_inv[(block[1] >> 40) & 0xFF]) <<
40);
    valtmp |= ((unsigned long long int) (pi_inv[(block[1] >> 32) & 0xFF]) <<
32);
    valtmp |= ((unsigned long long int) (pi_inv[(block[1] >> 24) & 0xFF]) <<
24);
    valtmp |= ((unsigned long long int) (pi_inv[(block[1] >> 16) & 0xFF]) <<
16);
    valtmp |= ((unsigned long long int) (pi_inv[(block[1] >> 8) & 0xFF]) <<
8);
    valtmp |= (unsigned long long int) (pi_inv[block[1] & 0xFF]);

    block[1] = valtmp;
}

/* Linear transformation */

unsigned char Cipher3412::ll(unsigned long long int block[2])
{
    unsigned char val;

    val = mult_mod_poly[148][(block[0] >> 56) & 0xFF];
    val ^= mult_mod_poly[ 32][(block[0] >> 48) & 0xFF];
    val ^= mult_mod_poly[133][(block[0] >> 40) & 0xFF];
    val ^= mult_mod_poly[ 16][(block[0] >> 32) & 0xFF];
    val ^= mult_mod_poly[194][(block[0] >> 24) & 0xFF];
    val ^= mult_mod_poly[192][(block[0] >> 16) & 0xFF];
    val ^= mult_mod_poly[  1][(block[0] >> 8) & 0xFF];
    val ^= mult_mod_poly[251][(block[0] >> 0) & 0xFF];

    val ^= mult_mod_poly[  1][(block[1] >> 56) & 0xFF];
    val ^= mult_mod_poly[192][(block[1] >> 48) & 0xFF];
    val ^= mult_mod_poly[194][(block[1] >> 40) & 0xFF];
    val ^= mult_mod_poly[ 16][(block[1] >> 32) & 0xFF];
    val ^= mult_mod_poly[133][(block[1] >> 24) & 0xFF];
    val ^= mult_mod_poly[ 32][(block[1] >> 16) & 0xFF];
    val ^= mult_mod_poly[148][(block[1] >> 8) & 0xFF];
    val ^= mult_mod_poly[  1][(block[1] >> 0) & 0xFF];

    return(val);
}

/* R conversion function */
void Cipher3412::R(unsigned long long int * block)
{
    unsigned long long int valtmp[2];
    unsigned char val;

    valtmp[0] = block[0];
    valtmp[1] = block[1];
    val = ll(valtmp);

    block[1] >>= 8;
    block[1] |= ((block[0]) & 0xFF) << 56);

    block[0] >>= 8;

```

```

    block[0] |= ((unsigned long long int)(val) << 56);
}

/* Inverse of R conversion function */
void Cipher3412::R_inv(unsigned long long int * block)
{
    unsigned long long int valtmp[2];
    unsigned char val;

    valtmp[0] = (block[0] << 8);
    valtmp[0] |= ((block[1] >> 56) & 0xFF);
    valtmp[1] = (block[1] << 8);
    valtmp[1] |= ((block[0] >> 56) & 0xFF);

    val = ll(valtmp);

    block[0] = valtmp[0];
    block[1] = (valtmp[1] & 0xFFFFFFFFFFFFFFF00L);
    block[1] |= val;
}

void Cipher3412::L(unsigned long long int * block)
{
    R(block); R(block); R(block); R(block);
    R(block); R(block); R(block); R(block);
    R(block); R(block); R(block); R(block);
    R(block); R(block); R(block); R(block);
}

/* Inverse of L conversion function (16 iteration of R_inv) */
void Cipher3412::L_inv(unsigned long long int * block)
{
    R_inv(block); R_inv(block); R_inv(block); R_inv(block);
    R_inv(block); R_inv(block); R_inv(block); R_inv(block);
    R_inv(block); R_inv(block); R_inv(block); R_inv(block);
    R_inv(block); R_inv(block); R_inv(block); R_inv(block);
}

/* F conversion function (used in key setup) */
void Cipher3412::F(unsigned long long int * k, unsigned long long int *
block1, unsigned long long int * block0)
{
    unsigned long long int tmpblock[2];

    /* Bitwise xor of k with block2 */
    tmpblock[0] = (block1[0] ^ k[0]);
    tmpblock[1] = (block1[1] ^ k[1]);

    /* Apply S conversion function */
    S(tmpblock);

    /* Apply L conversion function */
    L(tmpblock);

    /* Add a0 to the the result */
    tmpblock[0] ^= block0[0];
    tmpblock[1] ^= block0[1];

    /* Update the results */
    block0[0] = block1[0];
    block0[1] = block1[1];

    block1[0] = tmpblock[0];

```

```

    block1[1] = tmpblock[1];
}

/*****
/*****
/* The three core functions */
/*****
/*****

/*****
/* Key setup procedure */
/*****
void Cipher3412::key_setup(unsigned long long int * master_key, unsigned long
long int * * subkeys)
{
    unsigned long long int c_values[32][2], blocktmp[2], blocktmp2[2];
    unsigned char i;

    /* Compute the C_i values */
    for(i = 0; i < 32; i++)
    {
        blocktmp[0] = 0L;
        blocktmp[1] = (unsigned long long int)(i + 1);
        L(blocktmp);
        c_values[i][0] = blocktmp[0]; c_values[i][1] = blocktmp[1];
    }

    /* Compute subkeys K_1 and K_2 */
    subkeys[0][0] = master_key[0]; subkeys[0][1] = master_key[1];
    subkeys[1][0] = master_key[2]; subkeys[1][1] = master_key[3];

    /* Compute subkeys K_3 and K_4 */
    blocktmp[0] = subkeys[0][0]; blocktmp[1] = subkeys[0][1];
    blocktmp2[0] = subkeys[1][0]; blocktmp2[1] = subkeys[1][1];
    F(c_values[0], blocktmp, blocktmp2); F(c_values[1], blocktmp, blocktmp2);
    F(c_values[2], blocktmp, blocktmp2); F(c_values[3], blocktmp, blocktmp2);
    F(c_values[4], blocktmp, blocktmp2); F(c_values[5], blocktmp, blocktmp2);
    F(c_values[6], blocktmp, blocktmp2); F(c_values[7], blocktmp, blocktmp2);

    subkeys[2][0] = blocktmp[0]; subkeys[2][1] = blocktmp[1];
    subkeys[3][0] = blocktmp2[0]; subkeys[3][1] = blocktmp2[1];

    /* Compute subkeys K_5 and K_6 */
    F(c_values[ 8], blocktmp, blocktmp2); F(c_values[ 9], blocktmp, blocktmp2);
    F(c_values[10], blocktmp, blocktmp2); F(c_values[11], blocktmp, blocktmp2);
    F(c_values[12], blocktmp, blocktmp2); F(c_values[13], blocktmp, blocktmp2);
    F(c_values[14], blocktmp, blocktmp2); F(c_values[15], blocktmp, blocktmp2);

    subkeys[4][0] = blocktmp[0]; subkeys[4][1] = blocktmp[1];
    subkeys[5][0] = blocktmp2[0]; subkeys[5][1] = blocktmp2[1];

    /* Compute subkeys K_7 and K_8 */
    F(c_values[16], blocktmp, blocktmp2); F(c_values[17], blocktmp, blocktmp2);
    F(c_values[18], blocktmp, blocktmp2); F(c_values[19], blocktmp, blocktmp2);
    F(c_values[20], blocktmp, blocktmp2); F(c_values[21], blocktmp, blocktmp2);
    F(c_values[22], blocktmp, blocktmp2); F(c_values[23], blocktmp, blocktmp2);

    subkeys[6][0] = blocktmp[0]; subkeys[6][1] = blocktmp[1];
    subkeys[7][0] = blocktmp2[0]; subkeys[7][1] = blocktmp2[1];

    /* Compute subkeys K_9 and K_{10} */
    F(c_values[24], blocktmp, blocktmp2); F(c_values[25], blocktmp, blocktmp2);
    F(c_values[26], blocktmp, blocktmp2); F(c_values[27], blocktmp, blocktmp2);

```

```

F(c_values[28], blocktmp, blocktmp2); F(c_values[29], blocktmp, blocktmp2);
F(c_values[30], blocktmp, blocktmp2); F(c_values[31], blocktmp, blocktmp2);

subkeys[8][0] = blocktmp[0]; subkeys[8][1] = blocktmp[1];
subkeys[9][0] = blocktmp2[0]; subkeys[9][1] = blocktmp2[1];
}

/*****
/* Encryption procedure */
*****/
void Cipher3412::encrypt(unsigned long long int * * subkeys, unsigned long
long int * plaintext, unsigned long long int * ciphertext)
{
    /* Round 1 with subkey 1 */
    ciphertext[0] = plaintext[0] ^ subkeys[0][0]; ciphertext[1] = plaintext[1]
^ subkeys[0][1];
    S(ciphertext); L(ciphertext);

    /* Round 2 with subkey 2 */
    ciphertext[0] ^= subkeys[1][0]; ciphertext[1] ^= subkeys[1][1];
    S(ciphertext); L(ciphertext);

    /* Round 3 with subkey 3 */
    ciphertext[0] ^= subkeys[2][0]; ciphertext[1] ^= subkeys[2][1];
    S(ciphertext); L(ciphertext);

    /* Round 4 with subkey 4 */
    ciphertext[0] ^= subkeys[3][0]; ciphertext[1] ^= subkeys[3][1];
    S(ciphertext); L(ciphertext);

    /* Round 5 with subkey 5 */
    ciphertext[0] ^= subkeys[4][0]; ciphertext[1] ^= subkeys[4][1];
    S(ciphertext); L(ciphertext);

    /* Round 6 with subkey 6 */
    ciphertext[0] ^= subkeys[5][0]; ciphertext[1] ^= subkeys[5][1];
    S(ciphertext); L(ciphertext);

    /* Round 7 with subkey 7 */
    ciphertext[0] ^= subkeys[6][0]; ciphertext[1] ^= subkeys[6][1];
    S(ciphertext); L(ciphertext);

    /* Round 8 with subkey 8 */
    ciphertext[0] ^= subkeys[7][0]; ciphertext[1] ^= subkeys[7][1];
    S(ciphertext); L(ciphertext);

    /* Round 9 with subkey 9 */
    ciphertext[0] ^= subkeys[8][0]; ciphertext[1] ^= subkeys[8][1];
    S(ciphertext); L(ciphertext);

    /* Round 10 with subkey 10 */
    ciphertext[0] ^= subkeys[9][0]; ciphertext[1] ^= subkeys[9][1];
}

/*****
/* Decryption procedure */
*****/
void Cipher3412::decrypt(unsigned long long int * * subkeys, unsigned long
long int * ciphertext, unsigned long long int * plaintext)
{
    /* Round 1 with subkey 10 */
    plaintext[0] = ciphertext[0] ^ subkeys[9][0]; plaintext[1] = ciphertext[1]
^ subkeys[9][1];

```

```

L_inv(plaintext); S_inv(plaintext);

/* Round 2 with subkey 9 */
plaintext[0] ^= subkeys[8][0]; plaintext[1] ^= subkeys[8][1];
L_inv(plaintext); S_inv(plaintext);

/* Round 3 with subkey 8 */
plaintext[0] ^= subkeys[7][0]; plaintext[1] ^= subkeys[7][1];
L_inv(plaintext); S_inv(plaintext);

/* Round 4 with subkey 7 */
plaintext[0] ^= subkeys[6][0]; plaintext[1] ^= subkeys[6][1];
L_inv(plaintext); S_inv(plaintext);

/* Round 5 with subkey 6 */
plaintext[0] ^= subkeys[5][0]; plaintext[1] ^= subkeys[5][1];
L_inv(plaintext); S_inv(plaintext);

/* Round 6 with subkey 5 */
plaintext[0] ^= subkeys[4][0]; plaintext[1] ^= subkeys[4][1];
L_inv(plaintext); S_inv(plaintext);

/* Round 7 with subkey 4 */
plaintext[0] ^= subkeys[3][0]; plaintext[1] ^= subkeys[3][1];
L_inv(plaintext); S_inv(plaintext);

/* Round 8 with subkey 3 */
plaintext[0] ^= subkeys[2][0]; plaintext[1] ^= subkeys[2][1];
L_inv(plaintext); S_inv(plaintext);

/* Round 9 with subkey 2 */
plaintext[0] ^= subkeys[1][0]; plaintext[1] ^= subkeys[1][1];
L_inv(plaintext); S_inv(plaintext);

/* Round 10 with subkey 1 */
plaintext[0] ^= subkeys[0][0]; plaintext[1] ^= subkeys[0][1];
/*L_inv(plaintext); S_inv(plaintext);*/
}

```

ПРИЛОЖЕНИЕ 3

Листинг класса StegoThread

Листинг 3.1 Заголовочный файл класса StegoThread:

```
#ifndef STEGOTHREAD_H
#define STEGOTHREAD_H

#include <QObject>
#include <QFile>
#include "hash3411.h"
#include "sign3410.h"
#include <QDebug>

class StegoThread : public QObject
{
    Q_OBJECT
public:
    explicit StegoThread(QObject *parent = 0);
    bool isRunning() {return mIsRunning;};
    void setNumThread(int n) {nThread = n;};

signals:
    void threadFinished();

public slots:
    void stego(QString file, QString privateKey, QString mark);
private:
    QString puthScreen;
    QString puthTextLog;
    QString puthServiceLog;
    Hash3411 hash;
    Sign3410 sign;
    bool mIsRunning;
    int nThread;
};

#endif // STEGOTHREAD_H
```

Листинг 3.2. Исходный файл класса StegoThread:

```
#include "stegothread.h"
#include <ctime>

StegoThread::StegoThread(QObject *parent) : QObject(parent)
{
    //пути
    puthScreen = "e://file//screen//";
    puthTextLog = "e://file//log//";
    puthServiceLog = "e://file//service//serviceLog.txt";
    mIsRunning = false;
}

void StegoThread::stego(QString file, QString privateKey, QString mark)
{
    qDebug() << nThread << " thread started";
    unsigned int start_time = clock(); // начальное время
    mIsRunning = true;
    sign.setPrivateKey(stringToBigInteger(privateKey.toStdString()));
    //-----hash-----
    //считываем файл
    QFile fileImage(puthScreen+file);
    QByteArray data;
    if(!fileImage.open(QIODevice::ReadWrite))
    {
        qDebug() << "file not open";
        return;
    }
    data = fileImage.readAll();
    fileImage.close();

    //конвертируем QByteArray to unsigned char*
    char *dataMessage = data.data();
    unsigned char *message = new unsigned char(data.length());
    message = reinterpret_cast<unsigned char*>(dataMessage);

    unsigned char out512[64];
    //вычисляем хэш
    hash.hash_512(message, data.length() * 8, out512);
    //-----sign-----
    std::string strSign = sign.signGen(out512);
    QString qStrSign = QString::fromStdString(strSign);
    int n = qStrSign.length();
    //-----stego-----
    QByteArray tableMark, table1, table2;
    tableMark.clear();
    table1.clear();
    table2.clear();
    char headerMark[5] = {0xFF, 0xDB, 0x00, 0x43, 0x02};
    tableMark.append(headerMark, 5);
    char header1[5] = {0xFF, 0xDB, 0x00, 0x43, 0x03};
    table1.append(header1, 5);
    char header2[5] = {0xFF, 0xDB, 0x00, 0x43, 0x04};
    table2.append(header1, 5);

    char dataTable1[64] = "";
    char dataTable2[64] = "";
    char dataTableMark[64] = "";

    for (int i=0; i<64; i++)
```

```

{
    if (i<6)
    {
        QString qNum = mark.mid(i*2,2);
        bool ok;
        int number = qNum.toInt(&ok,10);
        dataTableMark[i] = char(number);
    }
    else
    {
        srand (time(0));
        //рандомное число
        dataTableMark[i] = char(rand() % 255);
    }
}

for (int i=0; i<128; i=i+2)
{
    QString qNum = qStrSign.mid(i,2);
    bool ok;
    int number = qNum.toInt(&ok,16);
    dataTable1[i/2] = char(number);
}

for (int i=128; i<256; i=i+2)
{
    QString qNum = qStrSign.mid(i,2);
    bool ok;
    int number = qNum.toInt(&ok,16);
    dataTable2[i/2-64] = char(number);
}
tableMark.append(dataTableMark,64);
table1.append(dataTable1,64);
table2.append(dataTable2,64);

char cFind[2]={0xFF,0xDB};
QByteArray baFind(cFind,2);
int numberLastTable = data.lastIndexOf(baFind);

numberLastTable = numberLastTable+69;
data.insert(numberLastTable,tableMark);
numberLastTable = numberLastTable+69;
data.insert(numberLastTable,table1);
numberLastTable = numberLastTable+69;
data.insert(numberLastTable,table2);

QFile(puthScreen+file).remove();
QFile stego(puthScreen+file);
if(!stego.open(QIODevice::WriteOnly))
{
    qDebug() << "file not open";
    return;
}
stego.write(data);
stego.close();
//-----
unsigned int end_time = clock(); // конечное время
unsigned int search_time = end_time - start_time; // искомое время
qDebug() << nThread << " thread finished at =" << search_time;
mIsRunning = false;
}

```

ПРИЛОЖЕНИЕ И

Листинг класса VerificationThread

Листинг И.1 Заголовочный файл класса VerificationThread:

```
#ifndef VERIFICATIONTHREAD_H
#define VERIFICATIONTHREAD_H
#include <QObject>
#include "hash3411.h"
#include "sign3410.h"

class VerificationThread : public QObject
{
    Q_OBJECT
public:
    explicit VerificationThread(QObject *parent = 0);

signals:
    void setProgress(int);

public slots:
    void verification(QList<QString>,QList<QString>);

private:
    QString puthScreen;
    QString puthTextLog;
    QString puthServiceLog;
    Hash3411 hash;
    Sign3410 signVer;
};

#endif // VERIFICATIONTHREAD_H
```

Листинг И.2. Исходный файл класса VerificationThread:

```
#include "verificationthread.h"
#include <QDebug>
#include <QFile>
#include <QDir>

VerificationThread::VerificationThread(QObject *parent) :
    QObject(parent)
{
    //пути
    puthScreen = "e://file//screen//";
    puthTextLog = "e://file//log//";
    puthServiceLog = "e://file//service//serviceLog.txt";
}

void VerificationThread::verification(QList<QString>
listScreenServer, QList<QString> listLogServer)
{
    qDebug() << "thread start";

    //служебный журнал с открытыми ключами и метками
    QFile serviceFile(puthServiceLog);
    QByteArray dataServiceFile;
    if (!serviceFile.open(QIODevice::ReadOnly)) return;
    dataServiceFile = serviceFile.readAll();
    serviceFile.close();

    QDir dir(puthScreen);
    dir.setFilter(QDir::Files);
    dir.setSorting(QDir::Time | QDir::Reversed);
    dir.setNameFilters(QStringList(QString("*.JPG")));
    QFileInfoList list = dir.entryInfoList();
    int numberScreenServer=listScreenServer.length();
    for (int i=0; i<numberScreenServer; i++)
    {
        QString fileScreen = listScreenServer.first();
        int position = list.indexOf(puthScreen+fileScreen);
        if (position == -1)
        {
            qDebug() << "file " << fileScreen << "not found";
            continue;
        }
        else
        {
            //скриншот
            QFile file(puthScreen+list.at(position).fileName());
            QByteArray dataFile;
            if (!file.open(QIODevice::ReadOnly))
            {
                qDebug() << "file not open";
                return;
            }
            dataFile = file.readAll();
            file.close();

            QString verificationMark = "";

            int lengthFileService = dataServiceFile.length();

            //progress bar
            double progress = (double(i)/double(numberScreenServer))*100;
            emit setProgress(int(progress));
        }
    }
}
```

```

char cFind[2]={0xFF,0xDB};
QByteArray baFind(cFind,2);
int numberLastTable = dataFile.lastIndexOf(baFind);

QByteArray mark = dataFile.mid(numberLastTable-69-64,6).toHex();
QString sMark = "";
for (int n=0; n<mark.length(); n+=2)
{
    bool ok;
    QString bNum = mark.mid(n,2);
    if (bNum.toInt(&ok,16) < 10) sMark+="0";
    sMark += QString::number(bNum.toInt(&ok,16),10);
}

QByteArray sign = dataFile.mid(numberLastTable-64,64).toHex();
sign += dataFile.mid(numberLastTable+5,64).toHex();
dataFile.remove(numberLastTable-69-69,207);

QByteArray QxServiceLog, QyServiceLog;

if (sMark != verificationMark)
{
    int k=0;
    do
    {
        int m = dataServiceFile.indexOf("Mark=",k);
        k=m+5;
        QByteArray markServiceLog = dataServiceFile.mid(k,12);
        k=k+12;
        k=k+3;// "Qx="
        if (sMark == QString(markServiceLog))
        {
            verificationMark = QString(markServiceLog);
            int y = dataServiceFile.indexOf("Qy=",k);
            QxServiceLog = dataServiceFile.mid(k,y-k);

            m = dataServiceFile.indexOf("Mark=",y);
            if (m == -1)
            {
                QyServiceLog =
dataServiceFile.mid(y+3,lengthFileService-(y+3));
                break;
            }
            QyServiceLog = dataServiceFile.mid(y+3,m-(y+3));
            break;
        }
        m = dataServiceFile.indexOf("Mark=",k);
        if (m==-1) k=lengthFileService;
        k=m;
    } while (k < lengthFileService);

    signVer.setPublicKey(stringToBigInteger(QString(QxServiceLog).toStdString()),
stringToBigInteger(QString(QyServiceLog).toStdString()));
}

//-----hash-----
//конвертируем QByteArray to unsigned char*
char *dataMessage = dataFile.data();
unsigned char *message = new unsigned char(dataFile.length());
message = reinterpret_cast<unsigned char*>(dataMessage);
unsigned char out512[64];

```

```

        //вычисляем хэш
        hash.hash_512(message,dataFile.length() * 8, out512);
        //-----
        //-----sign-----

        bool ok = false;
        ok = signVer.signVer(out512,QString(sign).toStdString());
        if (!ok) qDebug() << "file " << list.at(position).fileName() <<
"change";
        //-----
    }
    listScreenServer.removeFirst();
}
emit setProgress(100);
qDebug() << "thread finished";
}

```

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

УДК 004.056.55

Политехнический институт
Факультет приборостроения,
информационных технологий и
электроники

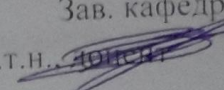
Выпускающая кафедра:

Информационная безопасность систем
и технологий

Учебная группа 11ПИ1

УТВЕРЖДАЮ

Зав. кафедрой ИБСТ

к.т.н., доцент  С.Л. Зефилов

«27» 04 2016 г.

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ
РАБОТУ**

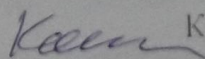
студента группы 11ПИ1 Старостина Евгения Владимировича

Специальность 100503 – Информационная безопасность
автоматизированных систем

Тема дипломного проекта: Программное средство защиты целостности и
доступности визуальной и текстовой информации.

Руководитель ВКР

д.т.н., профессор кафедры ИБСТ



Кашаев Евгений Дмитриевич

Консультант

начальник сектора №09323

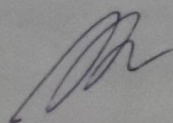
АО НПП «Рубин»



Присталов Павел Михайлович

Нормоконтролер

к.т.н., доцент



Иванов Алексей Петрович

1 Цели и задачи ВКР

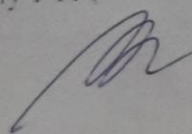
Цели работы:

- разработка модели угроз для целевой информационной системы;
- разработка программного средства защиты графической информации и текстовых журналов передаваемых команд от несанкционированной модификации и удаления;
- оценка эффективности реализованной защиты.

2 Тактико-технические требования к выполнению ВКР

Технические требования:

- программное обеспечение должно быть написано с использованием библиотеки Qt версии 4.8 для ОС Astra Linux и ОС Windows;
- программа должна работать с графической информацией в формате JPEG и текстовыми данными;
- для криптографической защиты информации должны использоваться алгоритмы, описанные в:
 - ГОСТ Р 34.10-2012. «Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи»;
 - ГОСТ Р 34.11-2012 «Информационная технология. Криптографическая защита информации. Функция хэширования»;
 - ГОСТ Р 34.12-2015 «Информационная технология. Криптографическая защита информации. Блочные шифры».
- приложение должно обеспечивать возможность легитимного просмотра и удаления защищаемых графических и текстовых данных;
- приложение не должно хранить какую-либо служебную информацию (ключи, хэши и т.д.) в файлах на диске в незашифрованном виде;
- приложение должно осуществлять соединение с сервером резервного копирования по протоколу FTP;



- приложение должно генерировать сессионную пару открытого и закрытого ключа электронно-цифровой подписи, закрытый ключ по завершению сеанса уничтожается, открытый ключ хранится в журнале.

В ВКР должны быть рассмотрены следующие вопросы:

- анализ предметной области: местоположение приложения в системе, анализ потоков информации с которыми работает приложение;

- модель угроз для разрабатываемого приложения;
- выбор защитных мер для защиты от актуальных угроз;
- программная реализация выбранных защитных мер;
- тестирование разработанной программы.

3 Требования к разрабатываемой документации

Отчет по ВКР выполняется в соответствии с требованиями ГОСТ 7.32.

Перечень и содержание графической части ВКР:

- модель угроз - 1 л., ф. А1 (плакат);
- диаграмма последовательности - 1 л., ф. А1 (плакат);
- интерфейс ПО - 1 л., ф. А1 (плакат);
- алгоритм процесса формирования и проверки цифровой подписи - 1 л., ф. А1 (плакат);
- процесс вычисления хэш-функции - 1 л., ф. А1 (плакат);
- алгоритм блочного шифрования - 1 л., ф. А1 (плакат);
- демонстрация работы (презентация).

Электронная копия отчета и графической части ВКР должна быть представлена на магнитном или оптическом носителе.

4 Сроки выполнения ВКР

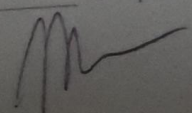
Тема утверждена приказом ректора ПГУ № 491/с от «25» 04 2016 г.

Дата выдачи задания «09» февраля 2016 г.

Время дипломного проектирования с 18.04.2016 г. по 10.07.2016 г.

Задание к исполнению принял «09» февраля 2016 г.

Исполнитель ВКР  Старостин Е.В.



Ректору Пензенского
государственного университета
А. Д. Гулякову
студента 5 курса ФПИТО
очной формы обучения
Старостина Евгения Владимировича

Заявление

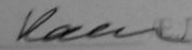
Я, Старостин Евгений Владимирович, студент 5 курса очной формы обучения специальности 10.05.03 – «Информационная безопасность автоматизированных систем», прошу провести проверку с использованием системы «Антиплагиат» дипломного проекта на тему «Программное средство защиты целостности и доступности визуальной и текстовой информации», выполненной мной самостоятельно, на содержание элементов плагиата.

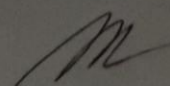
Все прямые заимствования из печатных и электронных источников, а также из интернетных ранее выпускных квалификационных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Я ознакомлен с действующим Стандартом университета «Выпускная квалификационная работа обучающихся по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры», согласно которому обнаружение плагиата является основанием для недопуска ВКР к защите и отчисления из университета.

 /Е.В. Старостин/
02.06.16

ВКР представлена на проверку

Руководитель ВКР  Е.Д. Кашаев/

 02.06/16.

Протокол
проверки на оригинальность в системе «Антиплагиат.ВУЗ» выпускной
квалификационной работы на тему «Программное средство защиты
целостности и доступности визуальной и текстовой информации» студента 5
курса Е.В. Старостина специальности 10.05.03 – «Информационная
безопасность автоматизированных систем».

Руководитель ВКР д.т.н., профессор кафедры ИБСТ Кашаев Евгений
Дмитриевич

Отчет о проверке № 1

ФИО: Евгений Евгеньевич Кашаев
дата выпуска: 04.06.2016 16:42:31
пользователь: 80.100.000.000 / 100.126.1.846
адрес: 100.126.1.846 / 100.126.1.846
на сайте: <http://www.antiplagiat.ru>

Информация о документе

№ документа: 1
Имя исходного файла: ВКР_Старостина_ЕВ_11791.docx
Размер текста: 1740 кБ
Тип документа: 100.126.1.846
Символы в тексте: 151757
Слов в тексте: 11791
Число предложений: 1280

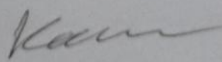
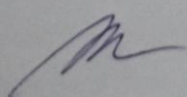
Информация об отчете

Дата: 04.06.2016 16:42:31 – Последний полный отчет
Комментарий: 100.126.1.846
Оценка оригинальности: 83.15%
Заимствования: 16.85%
Цитирование: 0%

Оригинальность: 83.15%
Заимствования: 16.85%
Цитирование: 0%

08.06.2016 г.

Секретарь ГЭК

Е.Д.Кашаев

А.П.Иванов

Отзыв
на выпускную квалификационную работу студента
Пензенского государственного университета
Старостина Е.В. по теме:
«Программное средство защиты целостности и доступности визуальной и текстовой информации»

Целью квалификационной работы Старостина Е.В. является разработка программного средства защиты целостности и доступности визуальной и текстовой информации.

Во время выполнения задания автором был изучен большой объем теоретического и практического материала по различным аспектам защиты целостности и доступности информации. Был произведен анализ предметной области, анализ потоков информации с которыми работает программное средство, также была разработана модель угроз. Для выделенных угроз были выбраны защитные меры. В соответствии с техническим заданием и проведенным анализом научно-технической литературы, автором разработано программное средство защиты целостности и доступности визуальной и текстовой информации.

Следует признать, что выполненная работа потребовала большого терпения и усердия, которые Старостин Е.В. проявил в полной мере. Техническое задание по все пунктам выполнено полностью. Пояснительная записка содержит все необходимые таблицы и рисунки, повышающие наглядность текстового материала.

Считаю, что дипломный проект выполнен в полном соответствии с заданием, заслуживает высокой оценки и может быть допущен к защите. Старостин Е.В. достоин присвоения квалификации специалиста по информационной безопасности по специальности 10.05.03 – «Информационная безопасность автоматизированных систем».

Количество баллов: 15

Руководитель ВКР
д.т.н., профессор кафедры ИБСТ

 Кашаев Е.Д.

РЕЦЕНЗИЯ

на выпускную квалификационную работу
студента

Пензенского государственного университета

Старостина Евгения Владимировича

Тема проекта: "Программное средство защиты целостности и доступности
визуальной и текстовой информации"

Проблемы защиты целостности и доступности информации являются актуальными проблемами информационной безопасности для любой информационной системы.

В дипломном проекте проведен анализ предметной области, разработана модель угроз и выбраны защитные меры, а также представлена реализация выбранных мер в программном средстве.

Студентом обработано большое количество теоретического материала. Содержание дипломного проекта полностью соответствует выбранной теме. Все пункты технического задания выполнены в полном объеме. Дипломный проект может быть допущен к защите.

Выбранные защитные меры хорошо обоснованы, произведен анализ существующих защитных мер на операционных системах, для которых разрабатывается программное средство.

Проект оформлен в соответствии с ГОСТ 7.32-2001, содержит все необходимые таблицы и рисунки, повышающие наглядность текстового материала.

Необходимо отметить, что проведенная работа требует больших трудозатрат и хороших тематических знаний.

В качестве недочета стоит отметить, что в работе не представлено обоснование выбора программного обеспечения (Qt, ОС AstraLinux, ОС Windows).

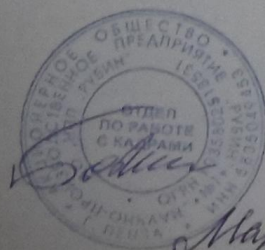
Дипломный проект соответствует предъявляемым требованиям и заслуживает отметки "отлично", а дипломник - присвоения квалификации специалиста по информационной безопасности по специальности 10.05.03 - "Информационная безопасность автоматизированных систем"

Количество баллов: 9

Рецензент:

начальник отдела 5800

АО "НПП "Рубин"



Байкин Александр Анатольевич

УТВЕРЖДАЮ

Зам. генерального директора

АО «НПП «Рубин»

Цофин А.С.

2016 г.



АКТ О ВНЕДРЕНИИ

результатов выпускной квалификационной работы

Старостина Евгения Владимировича

на тему «Программное средство защиты целостности и доступности визуальной и
текстовой информации»

на предприятии АО «НПП «Рубин»

Комиссия в составе: Боровков В.Г. (председатель), Бубнов М.В., Пристатов П.М. установила, что результаты выпускной квалификационной работы «Программное средство защиты целостности и доступности визуальной и текстовой информации» использованы в программном средстве «Комплекс функциональных программ пользовательского интерфейса инженера-оператора изделия Б1».

Использование «Программного средства защиты целостности и доступности визуальной и текстовой информации», предложенного в выпускной квалификационной работе, позволяет обеспечить целостность и доступность информации, сохраняемой комплексом функциональных программ пользовательского интерфейса инженера-оператора изделия Б1 на жёстком магнитном диске автоматизированного рабочего места.

Председатель:

начальник отделения №09300

A handwritten signature in blue ink, likely belonging to V.G. Borovkov.

Боровков В.Г.

Члены комиссии:

начальник отдела №09320

начальник сектора №09323

Two handwritten signatures in blue ink, likely belonging to M.V. Bubnov and P.M. Pristatov.

Бубнов М.В.

Пристатов П.М.