

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

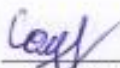
Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

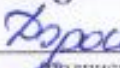
Система хранения облачных данных

Студент


(подпись, дата)

Соустин Константин Романович
(ФИО полностью)

Руководитель


(подпись, дата)

Дорошенко И. Н.
(фамилия, инициалы)

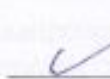
Нормоконтролёр


(подпись, дата)

29.08.19
А.В. Кучин
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 31.05.2019 г. № 38-09-19)

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 01.07.2019 № 29)

Секретарь ГЭК


(подпись)

Е.И. Гурин
(фамилия, инициалы)

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин М.А. Митрохин

«23» апреля 2019 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Соустин Константин Романович гр.15BB1 факультета вычислительной техники

2. Тема работы: Система хранения облачных данных

Тема утверждена приказом ПГУ № 554/о от "29" апреля 2019 г.

3. Руководитель бакалаврской работы Дорошенко Иван Николаевич

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать систему хранения зашифрованных данных в облаке. Разработать и отладить взаимодействие модулей приложения. Система должна быть представлена в виде установочного файла программы.

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной областей: изучить способы обеспечения безопасности данных в облаке, рассмотреть существующие средства защиты данных в облаке. Необходимо изучить методы шифрования данных.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	7	29.04.2019-06.05.2019	Хорошилкин
2.	Анализ требований	3	07.05.2019-10.05.2019	Хорошилкин
3.	Изучение методов шифрования	3	11.05.2019-14.05.2019	Хорошилкин
4.	Разработка системы хранения	10	15.05.2019-25.05.2019	Хорошилкин
5.	Оформление пояснительной записки	5	26.05.2019-31.05.2019	Хорошилкин

Дата выдачи задания "29" Апреля 2019 г.

Руководитель бакалаврской работы Хорошилкин Дорошенко И.Н.
(подпись, дата)

Задание к исполнению принял студент Соустин Соустин К.Р.
(подпись, дата)

Работу к защите допустить
Декан факультета Фионова Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Бакалаврская работа содержит 128 страниц, 10 рисунков, 8 источников, 2 приложения. Структура работы включает в себя введение, 5 глав, заключение, список использованной литературы и приложение.

Цель работы: Разработать систему хранения зашифрованных данных в облаке. Разработать и отладить взаимодействие модулей приложения. Система должна быть представлена в виде установочного файла программы.

Достижение поставленной цели требует решения следующих задач:

- 1) формирование требований к проектируемой системе и определение её функциональных возможностей;
- 2) изучить способы обеспечения безопасности данных в облаке;
- 3) изучить методы шифрования данных;
- 4) разработка необходимых программных модулей, для работы с облачными данными.

В ходе проделанной работы были изучены основные методы работы с программной библиотекой DOKAN. В процессе работы проводился анализ, сравнение существующих программ шифрования.

В результате выполнения работы была создана система хранения облачных данных.

В дальнейшем планируется добавлять алгоритмы шифрования данных пользователя, а также улучшать пользовательский интерфейс, для повышения безопасности и оптимизации скорости работы программы.

					ПГУ 2.090301.116.001 ПЗ			
Изм	Лист	№ докум.	Подпись	Дата				
Разраб.		Соустин К.Р.			Система хранения облачных данных Пояснительная записка	Лит.	Лист	Листов
Провер.		Дорошенко И.Н.					4	
Н. Контр.		Кучин А.В.				ФВТ, гр 15ВВ1		
Утверд.		Митрохин М.А.						

Содержание

РЕФЕРАТ.....	4
Содержание	8
Введение	9
1 Обзор предметной области	12
1.1 Облачные вычисления.....	12
1.2 Способы обеспечения безопасности "Облачных технологий"	14
1.2.1 Сохранность данных. Шифрование.....	14
1.2.2 Защита данных при передаче	15
1.2.3 Аутентификация	16
1.2.4 Изоляция пользователей.....	17
2.1 Vboxcryptor.....	18
2.2 Cryptomator	19
2.3 Tresorit	20
3 Анализ API для разработки хранилищ облачных систем	22
3.1 Точка монтирования.....	22
3.1.1 Применение точек монтирования в Windows	22
3.2. DokanAPI.....	25
3.2.1 Методы для реализации	26
4 Методы шифрования.....	29
4.1AES	29
4.1.1Особенности AES	29
4.1.2 Как работает шифрование AES.....	30
5 Разработка системы.....	32
5.1 Проектирование системы	32
5.2 Программная реализация системы	32
Заключение	34
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	35
ПРИЛОЖЕНИЕ А. Листинги программы	36

Введение

Человеческое общество по мере своего развития прошло этапы овладения веществом, затем энергией и, наконец, информацией. В рабовладельческом и феодальном обществах деятельность общества в целом и каждого человека в отдельности была направлена в первую очередь на овладение веществом.

С начала люди научились изготавливать простые орудия труда и охоты, в античности появились первые механизмы и первые средства передвижения. В средние века люди изобрели первые сложные орудия труда и механизмы.

Овладение энергией находилось в этот период на начальной ступени развития, в качестве источников энергии использовались солнце, вода, огонь, ветер и сила человека.

С начала человеческой истории возникла потребность в передаче, а также в хранении информации.

Начиная примерно с XVII века, в процессе развития машинного производства на первый план выходит проблема овладения энергией. Сначала совершенствовались способы овладения энергией ветра и воды, а затем человечество овладело солнечной энергией.

В конце XIX века началось овладение электрической энергией, были изобретены электрогенератор и электродвигатель. И наконец, в середине XX века человечество овладело атомной энергией, в 1954 году в СССР была запущена в эксплуатацию первая атомная электростанция.

Освоение энергией позволило перейти к массовому производству потребительских товаров. Было создано индустриальное общество. В этот период начали происходить существенные изменения в способах хранения и передачи информации.

В информационном обществе главным ресурсом является информация. Владея информацией о самых различных процессах и явлениях можно эффективно и оптимально строить любую деятельность.

Важно не только произвести большое количество продукции, но и нужную

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

продукцию в нужное время. С определёнными затратами и так далее. Поэтому в информационном обществе повышается не только качество потребления, но и качество производства. Человечество использует информационные технологии, имеет лучшие условия труда, труд становится творческим, интеллектуальным и так далее.

В настоящее время самые развитые страны мира (Япония, США, страны Западной Европы) фактически уже вступили в информационное общество. Другие же, в том числе и Россия, находятся на подступах к нему. В качестве критериев развитости информационного общества можно выбрать три: наличие компьютеров, уровень развития компьютерных сетей и количество населения, занятого в информационной сфере, а также использование информационных и коммуникационных технологий в своей повседневной деятельности.

Информация на сегодняшний день очень дорого стоит и её необходимо охранять. Массовое применение персональных компьютеров, к сожалению, оказалось связанным с появлением программ-вирусов, препятствующих нормальной работе компьютера, разрушающих файловую структуру дисков и наносящих ущерб хранимой в компьютере информации.

Информацией владеют все люди без исключения. Любой человек решает для себя, какую информацию ему необходимо получить, а какая информация не должна быть доступна другим и т.д. Человеку легко, хранить информацию, которая у него в голове, но как быть, если информация занесена в «мозг машины», к которой имеют доступ многие люди.

Чтобы не потерять информацию разрабатываются различные механизмы её защиты, которые используются на всех этапах работы с ней. Защищать от повреждений и внешних воздействий надо не только устройства, на которых хранится секретная и важная информация, но и каналы связи.

Повреждения могут быть вызваны не только поломкой оборудования или канала связи, но и подделкой или разглашением секретной информации.

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		10

Сбой системы оборудования может произойти в результате стихийного бедствия, либо его кражи.

Для сохранения информации используют различные способы защиты:

- безопасность зданий, где хранится секретная информация
- контролирование доступа к секретной информации
- разграничение доступа
- дублирование каналов связи и подключение резервных устройств
- криптографические преобразования информации

		.			ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

1 Обзор предметной области

1.1 Облачные вычисления

Самое современное направление повышения работоспособности использования информационных систем является переход к Cloud computing. В русском языке термин "Cloud computing" в результате перевода получил трактовку – "Облачные вычисления". Следует заметить, что дословный перевод выражения Cloud computing как "Облачные вычисления" не полностью отражает сущность современных процессов удаленного обслуживания пользователей информационных систем. Cloud computing, кроме удаленного выполнения приложений, предполагает также весь спектр информационных услуг, т.е. хранение, поиск и передача информации, обеспечение ее безопасности и многое другое. Поэтому логичнее употреблять термин "Облачные технологии".

Наиболее актуальны сейчас "Облачные технологии" (Cloud computing). Экономичность, легкость развертывания, многопользовательская архитектура, простота в использовании – все это способствует их быстрому распространению. "Облачные технологии" (Cloud computing) предполагают весь комплекс информационных услуг, включая хранение, поиск и передачу информации, обеспечение ее безопасности и многое другое.

"Облачные технологии" – это современный подход к размещению, предоставлению и потреблению приложений и компьютерных ресурсов, при котором приложения и ресурсы становятся доступны через Интернет в виде сервисов, потребляемых на платформах и устройствах.

Основными характеристиками облачных вычислений являются:

–**масштабируемость** (масштабируемое приложение обеспечивает большую нагрузку за счет увеличения количества запущенных экземпляров);

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

–**эластичность** (позволяет быстро нарастить мощность инфраструктуры без внедрения инвестиций в оборудование и программное обеспечение);

–**мультиотенантность** (позволяет снизить расходы на облачную платформу и использовать доступные вычислительные ресурсы);

–**оплата за использование** (позволяет перевести часть капитальных издержек в операционные);

–**самообслуживание** (позволяет потребителям запросить и получить требуемые ресурсы в короткое время).

"Облачные технологии" и предоставляемые ими сервисы можно сравнить с потреблением воды или электроэнергии. Когда в жару или холод меняется потребление воды и электричества, так и потребление сервисов, предоставляемых "облачными" платформами, может возрасти или уменьшиться в зависимости от повышения или понижения нагрузок.

Схожесть этих сервисов заключается в том, что:

во-первых, потребители услуг платят только за реальную утилизацию;

во-вторых, ресурсы берутся в аренду, т.е. поставщики таких сервисов обеспечивают их доступность в виде арендуемых "ресурсов", оставляя за собой вопросы для создания и поддержания инфраструктуры;

в-третьих, заключая договор с соответствующей организацией, подразумевается доступность тех или иных ресурсов, а организация обеспечивает своевременную оплату их аренды.

Самым значимым из достоинств "Облачных технологий" является безопасность ("облачные" сервисы имеют высокую безопасность при эффективном ее обеспечении, однако при халатном отношении эффект может быть полностью противоположным).

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

Как же убедить потребителя, что его данные будут в безопасности? Решением является соответствие облака к требованиям нормативных документов и стандартов в области обеспечения информационной безопасности. Но в российском законодательстве пока нет стандартов, описывающих принцип построения защиты информации в "Облачных технологиях". Поэтому многие поставщики "Облачных технологий" вынуждены сами выбирать способы защиты информации из огромного количества готовых решений. Но все средства защиты должны учитывать особенности "Облачной технологии".

1.2 Способы обеспечения безопасности "Облачных технологий"

1.2.1 Сохранность данных. Шифрование

Шифрование – один из самых эффективных способов защиты данных. Провайдер, предоставляющий доступ к данным, должен шифровать информацию клиента, хранящуюся в ЦОД (Центр обработки данных – совокупность серверов, размещенных на одной площадке с целью повышения эффективности и защищенности), а также в случае отсутствия необходимости, безвозвратно удалять.

При шифровании данных клиента всегда возникает вопрос о ключах. Хранение ключей на облачном сервере не разумно, потому что каждый, кто имеет доступ к облачным серверам или шаблонам, может получить доступ к ключу, а значит, и к зашифрованным данным. Набор пароля при запуске системы, как это принято в локальных решениях для шифрования данных, затруднен в связи с отсутствием настоящей консоли, однако идея неплоха. Физический ввод ключа заменяется запросом, который облачный сервер отправляет внешнему источнику — серверу управления ключами (Key Management Server, KMS).

Решающим фактором для обеспечения безопасности такого решения является раздельная эксплуатация облачного сервера и сервера управления ключами (см. Рисунок 1): если оба размещены у (одного и того же)

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		14

провайдера облачных сервисов, то вся информация снова оказывается собранной в одном месте. Решением этой проблемы является установка сервера KMS в локальном ЦОД или в качестве внешней услуги у другого сервис-провайдера.

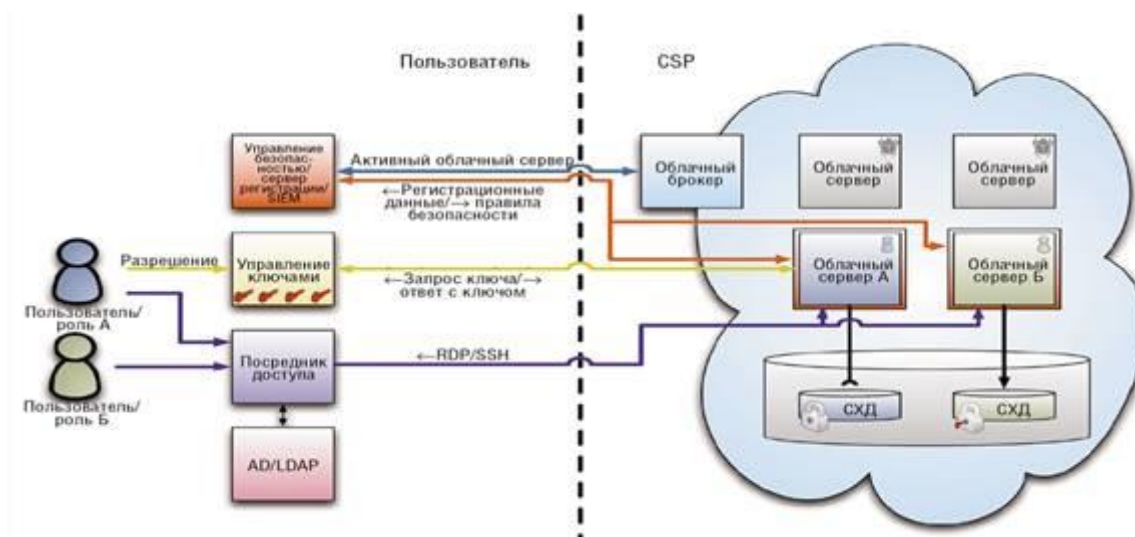


Рисунок. 1. Схема взаимодействия пользователя, сервера управления ключами и облачного сервера

1.2.2 Защита данных при передаче

В целях безопасной обработки данных необходимым условием является их шифруемая передача. Для того, чтобы защитить данные в публичном облаке используется туннель виртуальной частной сети (VPN), связывающей клиента и сервер для получения публичных облачных услуг. VPN-туннель способствует безопасным соединениям и позволяет использовать единое имя и пароль для доступа к различным облачным ресурсам. В качестве средства передачи данных в публичных облаках VPN - соединение использует общедоступные ресурсы, такие как Интернет. Процесс основан на режимах доступа с шифрованием при помощи двух ключей на базе протокола Secure Sockets Layer (SSL).

Большинство протоколов SSL и VPN в качестве опции поддерживают использование цифровых сертификатов для аутентификации, посредством которых проверяется идентификационная информация другой стороны,

причем еще до начала передачи данных. Эти цифровые сертификаты могут храниться на виртуальных жестких дисках в зашифрованном виде, и используются они только после того, как сервер управления ключами проверит идентификационную информацию и безопасность системы. Следовательно, такая цепочка взаимозависимостей позволит передавать данные только тем облачным серверам, которые прошли предварительную проверку. Такие зашифрованные данные при передаче должны быть доступны только после аутентификации. Данные не получится прочесть или сделать изменения в них, даже в случае доступа через ненадежные узлы. Такие технологии достаточно известны, алгоритмы и надежные протоколы AES, TLS, IPsec давно используются провайдерами.

1.2.3 Аутентификация

Аутентификация — это защита паролем. Для обеспечения более высокой надежности и работоспособности, часто прибегают к таким средствам, как токены (электронный ключ для доступа к чему-либо) и сертификаты. Самый простой и достаточно надежный метод аутентификации — это технология использования одноразовых паролей (One Time password, OTP). Эти пароли могут генерироваться либо специальными программами, либо дополнительными устройствами, либо сервисами, с пересылкой пользователю по SMS. Основное отличие облачной инфраструктуры заключается в большой масштабируемости и более широкой географической распределенности. В самом простом случае одноразовый пароль будет сгенерирован специальным сервером аутентификации и выслан в SMS на мобильный телефон пользователя после ввода правильного статического пароля на страницу доступа к облачному сервису. Для прозрачного взаимодействия провайдера с системой идентификации при авторизации, также рекомендуется использовать протокол LDAP (Lightweight Directory Access Protocol) и язык программирования SAML (Security Assertion Markup Language).

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		16

1.2.4 Изоляция пользователей

Использование индивидуальной виртуальной машины и виртуальной сети. Виртуальные сети должны быть развернуты с применением таких технологий, как VPN (Virtual Private Network), VLAN (Virtual Local Area Network) и VPLS (Virtual Private LAN Service). Часто провайдеры изолируют данные пользователей друг от друга за счет изменения кода в единой программной среде. Такой метод имеет опасность риска и позволяет получить доступ к данным. В случае возможной ошибки в коде пользователь может получить доступ к информации другого пользователя.

"Облачные технологии" представляют собой солидный прогресс в сфере развития информационных технологий и сервисов. Обеспечивая по требованию пользователя доступ к общим источникам вычислительных ресурсов в автономном, динамично масштабируемом и выверенном режиме, облачные вычисления предлагают несомненные преимущества работоспособности в скорости, оперативности и эффективности работы с информацией. В данной технологии безопасность играет значительную роль, этой проблеме специалисты уделяют огромное внимание. Но, несмотря на все сложности в области безопасности, преимущества предоставляемых через Интернет сервисов перевешивают возможные риски и "Облачные технологии" будут широко востребованы на рынке информационных технологий. Поэтому информационная безопасность пользователя - очень важная проблема всего мирового человечества.. Если эти данные попадают под действие нормативных или регулирующих ограничений, то выбор развертывания в облаке (частном, гибридном или общедоступном) зависит от понимания того, полностью ли поставщик соответствует этим требованиям. В другом случае возникает риск нарушения конфиденциальных и правовых требований. Обеспечения безопасности информации очень важны, когда речь идет о конфиденциальности пользователя.

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		17

2 Обзор существующих средств подключения к облаку

2.1 Boxcryptor

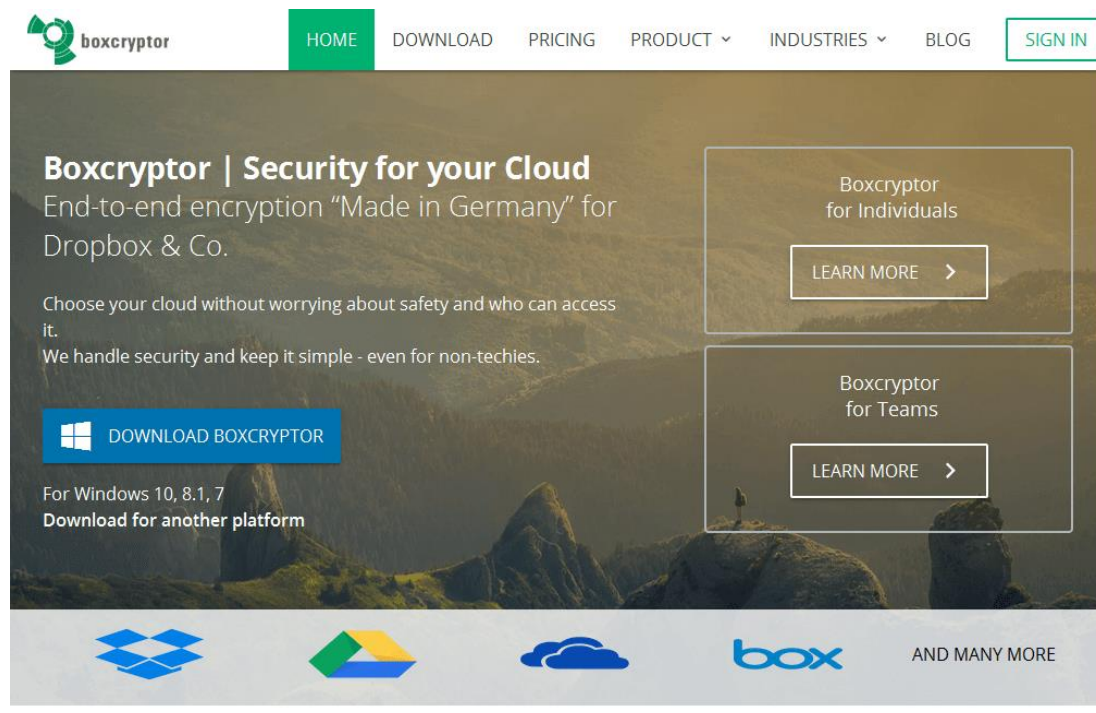


Рисунок. 2. Сайт программы ВохсCRYPTOR

Boxcryptor - немецкое программное обеспечение для облачного шифрования, известное своим удобным интерфейсом. Этот инструмент облачного шифрования берет на себя управление вашей учетной записью облачного хранилища, тем самым уменьшая нарушения безопасности.

Облачный инструмент шифрования Boxcryptor позволяет пользователям использовать его сквозной протокол шифрования, который шифрует их онлайн-активность от настольного компьютера до облака и наоборот.

Boxcryptor использует комбинированный алгоритм шифрования, включающий асимметричное RSA и симметричное шифрование AES; это гарантирует, что каждый файл сохраняет свой уникальный случайный ключ шифрования, который генерируется в процессе формирования файла.

Затем ключ файла используется для шифрования и дешифрования содержимого файла, что приводит к безопасному хранению.

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		18

2.2 Cryptomator



Рисунок. 3. Сайт программы Cryptomator

Этот инструмент облачного шифрования представляет собой технологию с открытым исходным кодом, которая может быть настроена по вашему желанию.

Одним из основных преимуществ Cryptomator является использование прозрачного шифрования, что означает, что пользователи не заметят никакой разницы при работе с файлами.

Кроме того, Cryptomator использует протокол шифрования военного уровня, то есть шифрование AES, для шифрования вашей облачной информации.

Ваша облачная учетная запись защищена от взлома и других попыток взлома благодаря расширенной защите и регулярным обновлениям безопасности от Cryptomator.

Вы также получаете виртуальный жесткий диск для доступа к вашим файлам в облачном хранилище, а также хранилище, в котором хранятся ваши зашифрованные файлы.

Кроме того, этот инструмент облачного шифрования имеет небольшой вес и поставляется с адаптивным пользовательским интерфейсом в отличие от других инструментов облачного шифрования.

Настройка проста, так как вам нужно только ввести имя пользователя и пароль, что устраняет необходимость создания учетных записей, управления ключами или конфигурации шифров.

С другой стороны, файлы шифруются индивидуально, даже если они размещены в группах или пакетах.

Это особенно полезно в связи с тем, что при внесении изменений в конкретный файл изменяется только конкретный файл, сохраняя при этом свое состояние шифрования.

2.3 Tresorit

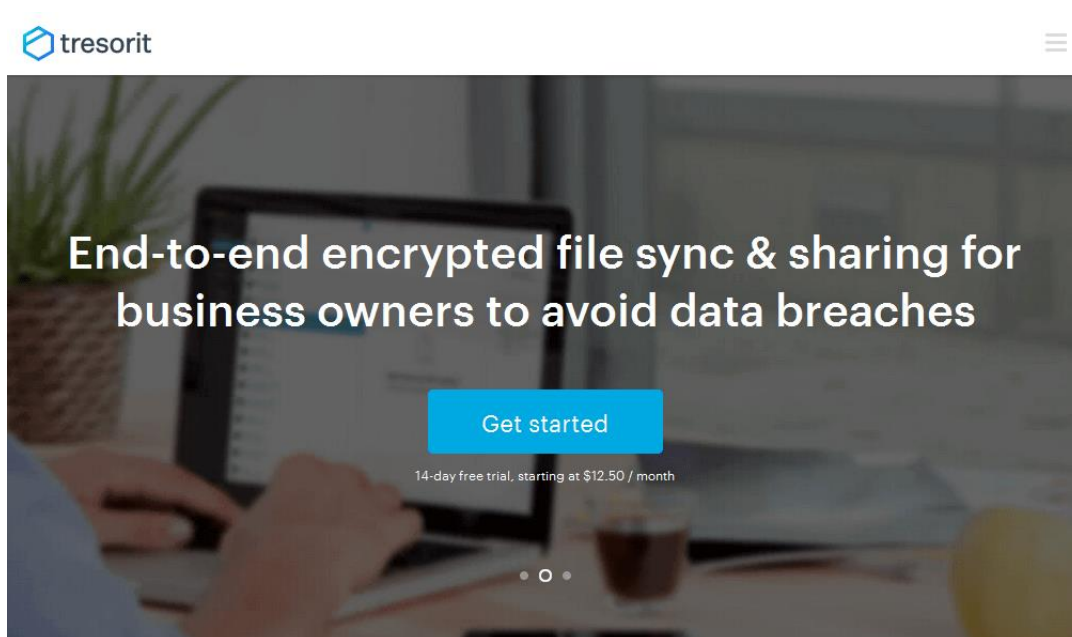


Рисунок. 4. Сайт программы Tresorit

Tresorit помогает защитить ваши файлы с помощью облачного шифрования, предотвращая тем самым доступ хакеров, правительства и отслеживающих устройств к вашему облачному хранилищу.

С помощью этого инструмента облачного шифрования вы можете легко защитить свои файлы, используя протокол шифрования AES 256 end-

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

to-end, который использует случайно сгенерированные ключи шифрования для обеспечения безопасности файлов.

Ключи, сгенерированные программным обеспечением, доступны только отдельным пользователям, так как их копии не отправляются на серверы Tresorit.

Следовательно, ваши файлы, хранящиеся в облаке, могут быть доступны только для вас с вашим уникальным закрытым ключом расшифровки.

		.			ПГУ 2.090301.116.001 ПЗ	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

3 Анализ API для разработки хранилищ облачных систем

3.1 Точка монтирования

Точка монтирования — это каталог или файл, с помощью которого обеспечивается доступ к новой файловой системе, каталогу или файлу.

Точка монтирования используется для реализации возможности динамически присоединять/отсоединять разделы диска к файловой системе во время работы операционной системы. Поддержка этой возможности в основанных на монолитных ядрах UNIX-подобных ОС обеспечивается самим ядром системы и системной программой, которая вызывается при помощи командной строки утилитой mount.

В операционной системе Windows точки монтирования доступны при использовании файловой системы NTFS. Точки монтирования реализованы через более общую технологию точек повторной обработки. Всего существует два вида точек монтирования: точка монтирования каталога и точка монтирования тома. Создание точек монтирования первого типа осуществляется через консольную команду mklink /J, создание точек монтирования второго типа — через команду mountvol.

3.1.1 Применение точек монтирования в Windows

Администраторы систем, отличных от Windows, знают и широко используют точки монтирования, но администраторы Windows только начинают осваивать их возможности. Парадигма управления хранением Windows всегда основывалась на буквенных символьных обозначениях, поэтому точки монтирования, независимые от связи с символьными обозначениями, представляют особую ценность. В прошлом, когда серверы Windows были простыми и число символов для обозначения дисков редко превышало 5 или 10, необходимости в точках монтирования почти не ощущалось. Однако в настоящее время точки монтирования просто необходимы, так как администраторы Windows строят все более сложные

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		22

серверы с подключением многочисленных систем хранения, таких как NAS (Network Attached Storage — устройство хранения данных, подключаемое к сети) и SAN (Storage Area Network — сеть устройств памяти). А если учесть потребности таких приложений, как Microsoft Exchange Server и Microsoft SQL Server, то нехватка символов для обозначения дисков станет еще более очевидной. Положение осложняет применение кластеров, так как на весь кластер отводится только 26 символов, а совместно используемые диски в кластере должны иметь единое символьное обозначение, независимо от того, какому кластеру они принадлежат. Microsoft дополнила Windows 2003 и Windows 2000 Server точками монтирования, чтобы устранить проблему нехватки символов и упростить управление памятью.

В Windows существует три способа настройки точек монтирования. Первый метод, вероятно, лучше остальных знаком администраторам Windows. Точка монтируется на ранее настроенные физические диски с помощью функции Add Mountpoint из интерфейса оснастки Disk Management (diskmgmt.msc) консоли управления Microsoft Management Console (MMC). Второй метод — запуск команды mountvol.exe — подходит тем администраторам, которые предпочитают интерфейс командной строки. Третий метод заключается в использовании вызовов Win32 API в подготовленном администратором .exe-файле. Функции SetVolumeMountPoint и DeleteVolumeMountPoint интерфейса Win32 API обеспечивают соответственно добавление и удаление точек монтирования.

Чтобы показать преимущества точек монтирования, приведу несколько практических примеров их использования в средах Windows Server и Exchange Server.

Сначала — простой пример. Предположим, что на системе Windows требуется установить второй жесткий диск. На ней уже имеется один жесткий диск (Drive 1), обозначенный символом C, и пользователь не хочет обозначать второй диск (Drive 2) символом D. Эту задачу можно решить, дополнив структуру каталогов Drive 1 точкой монтирования, указывающей

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

на Drive 2 (экран 1). В этом примере Drive 2 организован как точка монтирования C:\My Documents\Data. При перемещении по дереву каталогов к C:\My DocumentData пользователь или программа перенаправляется к корневому каталогу Drive 2. Благодаря этому простому приему можно не назначать второму диску нового символа и значительно упростить перемещение по каталогам.

Теперь — более сложный, практический пример. Группа OTG (Operations and Technology Group — производственно-технологическая группа) компании Microsoft располагает одной из крупнейших в мире систем Exchange с более чем 88 000 почтовых ящиков и емкостью памяти более 50 Тбайт, обрабатывающей более 6 млн. сообщений в день. В рамках смелого проекта консолидации, проведенного в 2003 г., специалисты OTG решили использовать для крупных серверов Exchange кластеры Microsoft Cluster Server (MSCS). Самый большой из них — пятиузловой кластер (четыре активных узла и один резервный), поддерживающий 16 000 почтовых ящиков (4000 почтовых ящиков на один активный узел).

Очевидно, что эта крупномасштабная система требует огромной памяти, состоящей из множества устройств с логическими номерами (LUN), используемыми в качестве томов базы данных, томов журналов и резервных томов. Первоначально специалисты OTG спроектировали эту конфигурацию без точек монтирования, так как до появления Windows 2003 точки монтирования нельзя было применять на общих дисках, объединенных в кластеры. Для обозначения дисков первоначальной конфигурации требовалось 36 символов; в результате построить и разместить кластер на платформе Windows 2000 Server оказалось невозможно.

Однако Windows 2003 поддерживает использование точек монтирования на объединенных в кластеры общих дисках, и специалисты OTG смогли сократить число символов для обозначения дисков с 36 до 16. Благодаря этому удалось построить и масштабировать кластеры Exchange, отвечающие

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		24

требованиям проекта консолидации. Для Microsoft точки монтирования томов стали ключом к более развитой архитектуре кластеров Exchange.

Точки монтирования томов — мощный инструмент для решения как простых, так и сложных задач. С их помощью можно преодолеть ограничение числа символов дисков и упростить перемещение по каталогам. Со временем администраторы Windows научатся использовать точки монтирования для построения крупномасштабных кластеров и управления томами файл-сервера Windows так же эффективно, как их коллеги по UNIX.

3.2. DokanAPI

Dokan — это драйвер, работающий на уровне ядра Windows, с которым могут взаимодействовать любые приложения из userspace-а. Драйвер существует как для 32-ух разрядных систем, так и для 64-ех разрядных. Драйвер не несет в себе реализацию какой-либо ФС, а лишь выступает в роли прокси, пропуская через себя все операции ввода-вывода (IO) из ядра ОС в userspace. Реализация функционала ФС лежит на наших плечах, мы создаем базовые функции ФС, такие как открытие/закрытие файла, чтение из файла, запись в файл и т.д., которые будут вызываться драйвером Dokan при возникновении в системе соответствующих событий. В момент регистрации нашей ФС в драйвере Dokan, мы указываем некоторые параметры работы нашей ФС (кол-во потоков-обработчиков операций ввода/вывода, точка монтирования нашей ФС (Dokan поддерживает монтирование в качестве съемного или сетевого диска и только в корень ФС), автоматическое размонтирование нашей ФС, в случае ошибок в работе, и некоторые другие параметры). После этого в ОС появляется новый диск в корне ФС, с которым любые приложения и сама ОС взаимодействует, как и с обычным диском. Какие операции разрешено производить в данной ФС, зависит только от разработчика этой ФС, то есть от нас.

3.2.1 Методы для реализации

Для создания файловой системы приложению необходимо реализовать функции в структуре DOKAN_OPERATIONS (объявленной в dokan.ч.) После реализации функция DokanMain может быть вызвана с помощью dokan_operations в качестве параметра для монтирования файловой системы. Семантика функции в DOKAN_OPERATIONS похожи на API-интерфейсы Windows с тем же именем. Поэтому параметры для этих функций такие же, как и для соответствующих API Windows. Эти функции вызываются из потоков, поэтому они должны быть потоко безопасными, иначе может возникнуть много проблем.

Эти функции обычно вызываются в этой последовательности:

DOKAN_OPERATIONS::ZwCreateFile

Другие функции (например, DOKAN_OPERATIONS.ReadFile или DOKAN_OPERATIONS.Функция WriteFile)

DOKAN_OPERATIONS:: Cleanup

DOKAN_OPERATIONS:: CloseFile

Перед операциями доступа к файлу (перечисление записей каталога, чтение атрибутов файла и т. д.), DOKAN_OPERATIONS.ZwCreateFile всегда вызывается. С другой стороны, функция DOKAN_OPERATIONS. Cleanup всегда вызывается драйвером файловой системы Dokan, когда файл закрыт CloseHandle Windows API.

Каждая функция должна возвращать STATUS_SUCCESS при успешном выполнении операции, в противном случае при возникновении ошибки верните правильный NTSTATUS (<https://support.microsoft.com/en-us/kb/113996>). Например, если ZwCreateFile не может открыть файл (Win32 error_file_not_found), он должен вернуть STATUS_OBJECT_NAME_NOT_FOUND. Dokan имеет экспортированный помощник, который может быть использован для преобразования ошибки Win32 в NTSTATUS с DokanNtStatusFromWin32.

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		26

Есть случай с DOKAN_OPERATIONS.ZwCreateFile, который должен возвращать STATUS_OBJECT_NAME_COLLISION вместо STATUS_SUCCESS, когда параметр CreationDisposition является CREATE_ALWAYS или OPEN_ALWAYS и файл, запрошенный успешно открыт, уже существовал.

DOKAN_OPERATIONS.FindFiles или DOKAN_OPERATIONS.FindFilesWithPattern призваны для того, чтобы отвечать на запросы каталога. Одна из этих двух функций должна быть реализована.

Для каждой записи каталога приложения файловой системы должны вызывать функцию FillFindData (передается как указатель функции на FindFiles, FindFilesWithPattern) со структурой WIN32_FIND_DATAW, заполненной информацией каталога: FillFindData (&win32FindDataW, DokanFileInfo). Это ответственность файловых систем для обработки шаблонов,. Когда приложения файловой системы реализуют только FindFiles.

Как указано выше, файловая система может быть смонтирована путем вызова функции DokanMain. Функция блокируется до тех пор, пока файловая система не будет размонтирована. Приложения файловой системы должны заполнить структуру DOKAN_OPTIONS для описания будущего устройства и DOKAN_OPERATIONS указателями функций для операций файловой системы (например, ZwCreateFile, ReadFile, CloseFile,...) перед передачей этих параметров функции DokanMain. Функции в структуре DOKAN_OPERATIONS должны быть потокобезопасными, поскольку они вызываются в нескольких потоках (а не в потоке, вызывающем DokanMain) с разными контекстами выполнения. DokanMain может мгновенно вернуть статус DokanMainResult в случае сбоя монтирования.

DOKAN_OPTIONS. Параметры с флагами DOKAN_OPTION описывают поведение устройства.

Файловую систему можно размонтировать, вызвав функцию DokanUnmount или DokanRemoveMountPoint. В большинстве случаев, когда

					ПГУ 2.090301.116.001 ПЗ	Лист
						27
Изм.	Лист	№ докум.	Подпись	Дата		

программы или оболочки, использующие файловую систему, зависят, операция размонтирования решит проблему, приведя систему в предыдущее состояние, когда файловая система не была смонтирована.

4 Методы шифрования

4.1 AES

Advanced Encryption Standard, или AES, представляет собой симметричный блочный шифр и внедряется в программное и аппаратное обеспечение по всему миру для шифрования конфиденциальных данных.

Национальный институт стандартов и технологий (NIST) начал разработку AES в 1997 году, когда объявил о необходимости использования алгоритма-преемника для стандарта шифрования данных (DES), который начинал становиться уязвимым для атак методом перебора.

Этот новый, усовершенствованный алгоритм шифрования был бы не классифицирован и должен был быть «способен защищать конфиденциальную правительственную информацию в следующем столетии», согласно объявлению NIST о процессе разработки усовершенствованного стандартного алгоритма шифрования. Он должен был быть прост в реализации в аппаратном и программном обеспечении, а также в ограниченных средах (например, в смарт-картах) и предлагать хорошую защиту от различных методов атаки.

4.1.1 Особенности AES

Процесс выбора для этого нового алгоритма симметричного ключа был полностью открыт для публичного изучения и комментирования; это обеспечило тщательный и прозрачный анализ представленных проектов.

NIST указал, что новый алгоритм усовершенствованного стандарта шифрования должен представлять собой блочный шифр, способный обрабатывать 128-битные блоки с использованием ключей размером 128, 192 и 256 бит; другие критерии выбора в качестве следующего усовершенствованного стандартного алгоритма шифрования:

Безопасность: конкурирующие алгоритмы должны были оцениваться по их способности противостоять атакам по сравнению с другими представленными шифрами, хотя уровень безопасности должен был считаться наиболее важным фактором в конкурентной борьбе.

Стоимость: Предназначенные для выпуска на глобальной, неисключительной и бесплатной основе, алгоритмы-кандидаты должны были быть оценены с точки зрения эффективности вычислений и памяти.

Реализация: алгоритм и характеристики реализации, подлежащие оценке, включают гибкость алгоритма; пригодность алгоритма для реализации в аппаратном или программном обеспечении; и в целом относительная простота реализации.

4.1.2 Как работает шифрование AES

AES состоит из трех блочных шифров: AES-128, AES-192 и AES-256. Каждый шифр шифрует и дешифрует данные блоками по 128 бит, используя криптографические ключи длиной 128, 192 и 256 бит соответственно.

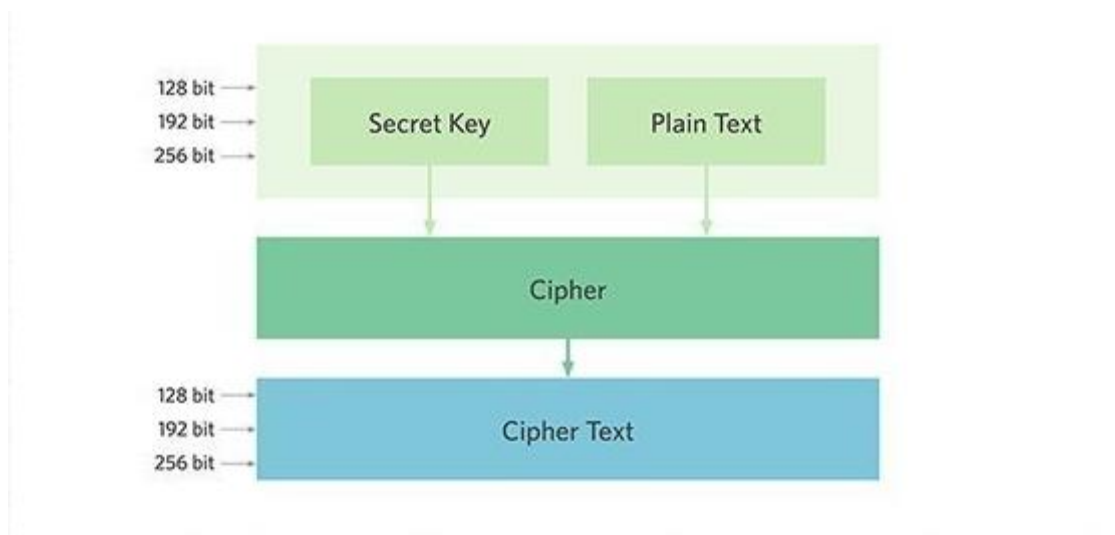


Рисунок. 5. Работа AES шифрования

Симметричные (также называемые секретным ключом) шифры используют один и тот же ключ для шифрования и дешифрования, поэтому отправитель и получатель должны как знать, так и использовать один и тот же секретный ключ. Все длины ключей считаются достаточными для защиты

секретной информации до уровня «Секрет», а для информации «Совершенно секретно» требуется длина ключа 192 или 256 бит. Существует 10 раундов для 128-битных ключей, 12 раундов для 192-битных ключей и 14 раундов для 256-битных ключей - раунд состоит из нескольких этапов обработки, которые включают замену, транспонирование и смешивание исходного текста и преобразование его в окончательный вывод зашифрованного текста.

Алгоритм шифрования AES определяет ряд преобразований, которые должны выполняться над данными, хранящимися в массиве. Первым шагом шифра является помещение данных в массив; после чего преобразования шифров повторяются в течение нескольких циклов шифрования. Количество раундов определяется длиной ключа: 10 раундов для 128-битных ключей, 12 раундов для 192-битных ключей и 14 раундов для 256-битных ключей.

Первым преобразованием в шифре шифрования AES является подстановка данных с использованием таблицы подстановок; второе преобразование сдвигает строки данных, третье смешивает столбцы. Последнее преобразование представляет собой простую эксклюзивную операцию (XOR), выполняемую для каждого столбца с использованием другой части ключа шифрования - более длинные ключи требуют больше раундов для завершения.

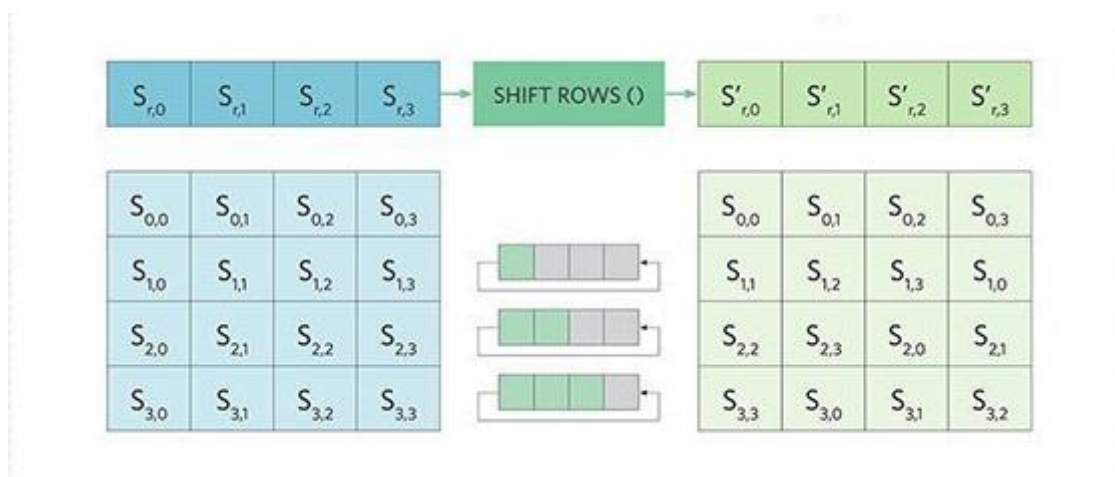


Рисунок. 6. AES подстановка данных

5 Разработка системы

5.1 Проектирование системы

Выделим следующие компоненты, которые необходимо реализовать в рамках данной работы:

1. Компонент создания точек монтирования, позволяющий монтировать облачные хранилища.
2. Компонент шифрования, позволяющий быстро зашифровать и расшифровать объект.
3. Компонент графического приложения необходим для визуализации результатов функционала.

5.2 Программная реализация системы

Разработанная программа состоит из следующих нескольких модулей, которые можно разделить на 2 группы:

1. Компоненты точки монтирования и шифрования
2. Компоненты графического приложения

Компоненты точки монтирования и шифрования, были реализованы в виде классов подключаемой библиотеки классов.

Компонент графического приложения реализован в виде формы, которая содержит компоненты управления для точек монтирования и шифрования.

Графическое приложение позволяет создавать точки монтирования для сетевых хранилищ. Результаты работы приложения представлены на рисунках 7 и 8;

Имя	Дата изменения	Тип	Размер
 A1RJQOcWGK6ycdiMHOy-yRX	28.06.2019 8:28	Файл	58 КБ
 aRalgntf6fQGmWQ-7mYlkkwXw5fzKt1zd...	28.06.2019 8:28	Файл	179 КБ
 RK24HtlRqzWlrxRhldokmXZ	29.06.2019 6:26	Файл	56 КБ
 rKTtpwFi4INF1ZxCYtcuAcjd	28.06.2019 8:28	Файл	58 КБ

Рисунок. 7. Зашифрованные данные на сетевом хранилище

 AVR_.docx	28.06.2019 8:28	Документ Micros...	56 КБ
 otchet_Coys_2.docx	28.06.2019 8:28	Документ Micros...	178 КБ
 PlanV_2.doc	28.06.2019 8:28	Документ Micros...	57 КБ
 PlanVCoys.doc	28.06.2019 8:28	Документ Micros...	58 КБ

Рисунок . 8. Расшифрованные данные на нашей точке монтирования

Заключение

В рамках ВКР разработана система хранения облачных данных.

Предложенная программа реализует алгоритм шифрования в облачном хранилище. Для защиты данных пользователя был реализован алгоритм AES. К облачным хранилищам позволяет подключаться программная библиотека DOKAN. Созданную программу можно использовать в качестве решения конкретных практических задач.

В дальнейшем планируется добавлять алгоритмы шифрования данных пользователя, а также улучшать пользовательский интерфейс, для повышения безопасности и оптимизации скорости работы программы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Котяшичев И. А. К вопросу о безопасности облачных технологий в информационной среде [Текст] / И. А. Котяшичев, С. В. Смоленцев // Молодой ученый. — 2014. — №5.1. — С. 25-28.
2. Социальные опасности и защита от них: учебник для студ. учреждений высшего проф. образования / [В. М. Губанов, Л. А. Михайлов, В. П. Соломин и др.]; под ред. Л.А. Михайлова. – М., Издательский центр «Академия», 2012.
3. Удо Шнайдер Использование облачных сервисов [Текст] / У. Шнайдер //
4. Журнал сетевых решений/LAN. — 2013. — № 04.
5. Peter Mell, Timothy Grance. «The NIST Definition of Cloud Computing (Draft)» // Recommendations of the National Institute of Standards and Technology, Special Publication 800 – 145 (Draft), сентябрь 2011 год.
6. Котяшичев И. А., Бырылова Е. А. Защита информации в «Облачных технологиях» как предмет национальной безопасности // Молодой ученый. — 2015. — №6.4. — С. 30-34.
7. К.С. Пан, М.Л. Цымблер Алгоритм блочного симметричного шифрования Advanced Encryption Standard (AES)// Технический отчет CELLAES-01
8. Документация API DOKAN//<https://dokan-dev.github.io/dokany-doc/html/>

ПРИЛОЖЕНИЕ А. Листинги программы

Файл «base64.h»

```
#ifndef _base64_incl_
#define _base64_incl_

inline int B64ToB256Bytes( int numB64Bytes )
{
    return (numB64Bytes * 6) / 8; // round down
}

inline int B32ToB256Bytes( int numB32Bytes )
{
    return (numB32Bytes * 5) / 8; // round down
}

inline int B256ToB64Bytes( int numB256Bytes )
{
    return (numB256Bytes * 8 + 5) / 6; // round up
}

inline int B256ToB32Bytes( int numB256Bytes )
{
    return (numB256Bytes * 8 + 4) / 5; // round up
}

/*
    convert data between different bases - each being a power of 2.
*/
void changeBase2(unsigned char *src, int srcLength, int srcPow2,
                unsigned char *dst, int dstLength, int dstPow2);

/*
    same as changeBase2, but writes output over the top of input data.
*/
void changeBase2Inline(unsigned char *buf, int srcLength,
                      int srcPow2, int dst2Pow,
                      bool outputPartialLastByte);

// inplace translation from values [0,2^6] => base64 ASCII
void B64ToAscii(unsigned char *buf, int length);
// inplace translation from values [0,2^5] => base32 ASCII
void B32ToAscii(unsigned char *buf, int length);

// inplace translation from values base64 ASCII => [0,2^6]
void AsciiToB64(unsigned char *buf, int length);
void AsciiToB64(unsigned char *out, const unsigned char *in, int length);

// inplace translation from values base32 ASCII => [0,2^5]
void AsciiToB32(unsigned char *buf, int length);
void AsciiToB32(unsigned char *out, const unsigned char *in, int length);

#endif
```

Файл «base64.cpp»

```
#include "base64.h"

#include <ctype.h>

// change between two powers of two, stored as the low bits of the bytes in the
```

					ПГУ 2.090301.116.001 ПЗ	Лист
						36
Изм.	Лист	№ докум.	Подпись	Дата		

```

// arrays.
// It is the caller's responsibility to make sure the output array is large
// enough.
void changeBase2(unsigned char *src, int srcLen, int src2Pow,
                unsigned char *dst, int dstLen, int dst2Pow)
{
    unsigned long work = 0;
    int workBits = 0; // number of bits left in the work buffer
    unsigned char *end = src + srcLen;
    unsigned char *origDst = dst;
    const int mask = (1 << dst2Pow) -1;

    // copy the new bits onto the high bits of the stream.
    // The bits that fall off the low end are the output bits.
    while(src != end)
    {
        work |= ((unsigned long) (*src++)) << workBits;
        workBits += src2Pow;

        while(workBits >= dst2Pow)
        {
            *dst++ = work & mask;
            work >>= dst2Pow;
            workBits -= dst2Pow;
        }

        // now, we could have a partial value left in the work buffer..
        if(workBits && ((dst - origDst) < dstLen))
            *dst++ = work & mask;
    }

    /*
    Same as changeBase2, except the output is written over the input data. The
    output is assumed to be large enough to accept the data.

    Uses the stack to store output values. Recurse every time a new value is
    to be written, then write the value at the tail end of the recursion.
    */
    static
    void changeBase2Inline(unsigned char *src, int srcLen,
                        int src2Pow, int dst2Pow,
                        bool outputPartialLastByte,
                        unsigned long work,
                        int workBits,
                        unsigned char *outLoc)
    {
        const int mask = (1 << dst2Pow) -1;
        if(!outLoc)
            outLoc = src;

        // copy the new bits onto the high bits of the stream.
        // The bits that fall off the low end are the output bits.
        while(srcLen && workBits < dst2Pow)
        {
            work |= ((unsigned long) (*src++)) << workBits;
            workBits += src2Pow;
            --srcLen;
        }

        // we have at least one value that can be output
        unsigned char outVal = work & mask;
        work >>= dst2Pow;
        workBits -= dst2Pow;

        if(srcLen)
        {
            // more input left, so recurse
            changeBase2Inline( src, srcLen, src2Pow, dst2Pow,
                            outputPartialLastByte, work, workBits, outLoc+1);
        }
    }
}

```

					ПГУ 2.090301.116.001 ПЗ	Лист
						37
Изм.	Лист	№ докум.	Подпись	Дата		

```

        *outLoc = outVal;
    } else
    {
        // no input left, we can write remaining values directly
        *outLoc++ = outVal;

        // we could have a partial value left in the work buffer..
        if(outputPartialLastByte)
        {
            while(workBits > 0)
            {
                *outLoc++ = work & mask;
                work >>= dst2Pow;
                workBits -= dst2Pow;
            }
        }
    }
}

void changeBase2Inline(unsigned char *src, int srcLen,
                      int src2Pow, int dst2Pow,
                      bool outputPartialLastByte)
{
    changeBase2Inline(src, srcLen, src2Pow, dst2Pow,
                      outputPartialLastByte, 0, 0, 0);
}

// character set for ascii b64:
// ",-0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
// a standard base64 (eg a64l doesn't use ',-' but uses './'. We don't
// do that because '/' is a reserved character, and it is useful not to have
// '.' included in the encrypted names, so that it can be reserved for files
// with special meaning.
static const char B642AsciiTable[] = ",-0123456789";
void B64ToAscii(unsigned char *in, int length)
{
    for(int offset=0; offset<length; ++offset)
    {
        int ch = in[offset];
        if(ch > 11)
        {
            if(ch > 37)
                ch += 'a' - 38;
            else
                ch += 'A' - 12;
        } else
            ch = B642AsciiTable[ ch ];

        in[offset] = ch;
    }
}

static const unsigned char Ascii2B64Table[] =
    "                                01  23456789:;          ";
    // 0123456789 123456789 123456789 123456789 123456789 123456789 1234
    // 0          1          2          3          4          5          6
void AsciiToB64(unsigned char *in, int length)
{
    return AsciiToB64(in, in, length);
}

void AsciiToB64(unsigned char *out, const unsigned char *in, int length)
{
    while(length--)
    {
        unsigned char ch = *in++;
        if(ch >= 'A')
        {
            if(ch >= 'a')

```

					ПГУ 2.090301.116.001 ПЗ	Лист
						38
Изм.	Лист	№ докум.	Подпись	Дата		

```

        ch += 38 - 'a';
    else
        ch += 12 - 'A';
    } else
        ch = Ascii2B64Table[ ch ] - '0';

    *out++ = ch;
}

}

void B32ToAscii(unsigned char *buf, int len)
{
    for(int offset=0; offset<len; ++offset)
    {
        int ch = buf[offset];
        if (ch >= 0 && ch < 26)
            ch += 'A';
        else
            ch += '2' - 26;

        buf[offset] = ch;
    }
}

void AsciiToB32(unsigned char *in, int length)
{
    return AsciiToB32(in, in, length);
}

void AsciiToB32(unsigned char *out, const unsigned char *in, int length)
{
    while(length--)
    {
        unsigned char ch = *in++;
        int lch = toupper(ch);
        if (lch >= 'A')
            lch -= 'A';
        else
            lch += 26 - '2';

        *out++ = (unsigned char)lch;
    }
}

```

Файл « BlockFileIO.h»

```

#ifndef _BlockFileIO_incl_
#define _BlockFileIO_incl_

#include "FileIO.h"
#include "FSConfig.h"

/*
    Implements block scatter / gather interface. Requires derived classes to
    implement readOneBlock() / writeOneBlock() at a minimum.

    When a partial block write is requested it will be turned into a read of
    the existing block, merge with the write request, and a write of the full
    block.
*/
class BlockFileIO : public FileIO
{
public:
    BlockFileIO( int blockSize, const FSConfigPtr &cfg );
    virtual ~BlockFileIO();

    // implemented in terms of blocks.

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		39

```

virtual ssize_t read( const IORequest &req ) const;
virtual bool write( const IORequest &req );

virtual int blockSize() const;

protected:

    int truncate( efs_off_t size, FileIO *base );
    void padFile( efs_off_t oldSize, efs_off_t newSize, bool forceWrite );

    // same as read(), except that the request.offset field is guarenteed to be
    // block aligned, and the request size will not be larger then 1 block.
    virtual ssize_t readOneBlock( const IORequest &req ) const =0;
    virtual bool writeOneBlock( const IORequest &req ) =0;

    ssize_t cacheReadOneBlock( const IORequest &req ) const;
    bool cacheWriteOneBlock( const IORequest &req );

    int _blockSize;
    bool _allowHoles;

    // cache last block for speed...
    mutable IORequest _cache;
};

#endif

```

Файл « BlockFileIO.cpp»

```

#include "BlockNameIO.h"

#include "Cipher.h"
#include "base64.h"

#include <cstring>
#include <rlog/rlog.h>
#include <rlog/Error.h>
#include <rlog/RLogChannel.h>

#include "i18n.h"

using namespace rlog;
using namespace rel;
using namespace boost;

static RLogChannel * Info = DEF_CHANNEL( "info/nameio", Log_Info );

static shared_ptr<NameIO> NewBlockNameIO( const Interface &iface,
    const shared_ptr<Cipher> &cipher, const CipherKey &key )
{
    int blockSize = 8;
    if(cipher)
        blockSize = cipher->cipherBlockSize();

    return shared_ptr<NameIO>(
        new BlockNameIO( iface, cipher, key, blockSize, false));
}

static shared_ptr<NameIO> NewBlockNameIO32( const Interface &iface,
    const shared_ptr<Cipher> &cipher, const CipherKey &key )
{
    int blockSize = 8;
    if(cipher)
        blockSize = cipher->cipherBlockSize();
}

```

					ПГУ 2.090301.116.001 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		

```

        return shared_ptr<NameIO>(
            new BlockNameIO( iface, cipher, key, blockSize, true));
    }

    static bool BlockIO_registered = NameIO::Register("Block",
        // description of block name encoding algorithm..
        // xgroup(setup)
        gettext_noop("Block encoding, hides file name size somewhat"),
        BlockNameIO::CurrentInterface(false),
        NewBlockNameIO);

    static bool BlockIO32_registered = NameIO::Register("Block32",
        // description of block name encoding algorithm..
        // xgroup(setup)
        gettext_noop("Block encoding with base32 output for case-sensitive systems"),
        BlockNameIO::CurrentInterface(true),
        NewBlockNameIO32);

    /*
    - Version 1.0 computed MAC over the filename, but not the padding bytes.
    This version was from pre-release 1.1, never publically released, so no
    backward compatibility necessary.

    - Version 2.0 includes padding bytes in MAC computation. This way the MAC
    computation uses the same number of bytes regardless of the number of
    padding bytes.

    - Version 3.0 uses full 64 bit initialization vector during IV chaining.
    Prior versions used only the output from the MAC_16 call, giving a 1 in
    2^16 chance of the same name being produced. Using the full 64 bit IV
    changes that to a 1 in 2^64 chance..

    - Version 4.0 adds support for base32, creating names more suitable for
    case-insensitive filesystems (eg Mac).
    */
    Interface BlockNameIO::CurrentInterface(bool caseSensitive)
    {
        // implement major version 4 plus support for two prior versions
        if (caseSensitive)
            return Interface("nameio/block32", 4, 0, 2);
        else
            return Interface("nameio/block", 4, 0, 2);
    }

    BlockNameIO::BlockNameIO( const rel::Interface &iface,
        const shared_ptr<Cipher> &cipher,
        const CipherKey &key, int blockSize,
        bool caseSensitiveEncoding )
        : _interface( iface.current() )
        , _bs( blockSize )
        , _cipher( cipher )
        , _key( key )
        , _caseSensitive( caseSensitiveEncoding )
    {
        // just to be safe..
        rAssert( blockSize < 128 );
    }

    BlockNameIO::~BlockNameIO()
    {
    }

    Interface BlockNameIO::get_interface() const

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		41

```

{
    return CurrentInterface(_caseSensitive);
}

int BlockNameIO::maxEncodedNameLen( int plaintextNameLen ) const
{
    // number of blocks, rounded up.. Only an estimate at this point, err on
    // the size of too much space rather than too little.
    int numBlocks = ( plaintextNameLen + _bs ) / _bs;
    int encodedNameLen = numBlocks * _bs + 2; // 2 checksum bytes
    if ( _caseSensitive )
        return B256ToB32Bytes( encodedNameLen );
    else
        return B256ToB64Bytes( encodedNameLen );
}

int BlockNameIO::maxDecodedNameLen( int encodedNameLen ) const
{
    int decLen256 = _caseSensitive ?
        B32ToB256Bytes( encodedNameLen ) :
        B64ToB256Bytes( encodedNameLen );
    return decLen256 - 2; // 2 checksum bytes removed..
}

int BlockNameIO::encodeName( const char *plaintextName, int length,
    uint64_t *iv, char *encodedName ) const
{
    // copy the data into the encoding buffer..
    memcpy( encodedName+2, plaintextName, length );

    // Pad encryption buffer to block boundary..
    int padding = _bs - length % _bs;
    if( padding == 0 )
        padding = _bs; // padding a full extra block!

    memset( encodedName+length+2, (unsigned char)padding, padding );

    // store the IV before it is modified by the MAC call.
    uint64_t tmpIV = 0;
    if( iv && _interface >= 3 )
        tmpIV = *iv;

    // include padding in MAC computation
    unsigned int mac = _cipher->MAC_16( (unsigned char *)encodedName+2,
        length+padding, _key, iv );

    // add checksum bytes
    encodedName[0] = (mac >> 8) & 0xff;
    encodedName[1] = (mac << 8) & 0xff;

    _cipher->blockEncode( (unsigned char *)encodedName+2, length+padding,
        (uint64_t)mac ^ tmpIV, _key);

    // convert to base 64 ascii
    int encodedStreamLen = length + 2 + padding;
    int encLen;

    if ( _caseSensitive )
    {
        encLen = B256ToB32Bytes( encodedStreamLen );

        changeBase2Inline( (unsigned char *)encodedName, encodedStreamLen,
            8, 5, true );
        B32ToAscii( (unsigned char *)encodedName, encLen );
    }
}

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		42

```

    } else
    {
        encLen = B256ToB64Bytes( encodedStreamLen );

        changeBase2Inline( (unsigned char *)encodedName, encodedStreamLen,
            8, 6, true );
        B64ToAscii( (unsigned char *)encodedName, encLen );
    }

    return encLen;
}

int BlockNameIO::decodeName( const char *encodedName, int length,
    uint64_t *iv, char *plaintextName ) const
{
    int declen256 = _caseSensitive ?
        B32ToB256Bytes( length ) :
        B64ToB256Bytes( length );
    int decodedStreamLen = declen256 - 2;

    // don't bother trying to decode files which are too small
    if(decodedStreamLen < _bs)
        throw RL_ERROR("Filename too small to decode");

    BUFFER_INIT( tmpBuf, 32, (unsigned int)length );

    // decode into tmpBuf,
    if (_caseSensitive)
    {
        AsciiToB32((unsigned char *)tmpBuf, (unsigned char *)encodedName, length);
        changeBase2Inline((unsigned char *)tmpBuf, length, 5, 8, false);
    } else
    {
        AsciiToB64((unsigned char *)tmpBuf, (unsigned char *)encodedName, length);
        changeBase2Inline((unsigned char *)tmpBuf, length, 6, 8, false);
    }

    // pull out the header information
    unsigned int mac = (((unsigned int)((unsigned char)tmpBuf[0])) << 8
        | ((unsigned int)((unsigned char)tmpBuf[1])));

    uint64_t tmpIV = 0;
    if( iv && _interface >= 3 )
        tmpIV = *iv;

    _cipher->blockDecode( (unsigned char *)tmpBuf+2, decodedStreamLen,
        (uint64_t)mac ^ tmpIV, _key);

    // find out true string length
    int padding = (unsigned char)tmpBuf[2+decodedStreamLen-1];
    int finalSize = decodedStreamLen - padding;

    // might happen if there is an error decoding..
    if(padding > _bs || finalSize < 0)
    {
        rDebug("padding, _bx, finalSize = %i, %i, %i", padding,
            _bs, finalSize);
        throw RL_ERROR( "invalid padding size" );
    }

    // copy out the result..
    memcpy(plaintextName, tmpBuf+2, finalSize);
    plaintextName[finalSize] = '\0';
}

```

```

// check the mac
unsigned int mac2 = _cipher->MAC_16((const unsigned char *)tmpBuf+2,
    decodedStreamLen, _key, iv);

BUFFER_RESET( tmpBuf );

if(mac2 != mac)
{
    rDebug("checksum mismatch: expected %u, got %u", mac, mac2);
    rDebug("on decode of %i bytes", finalSize);
    throw RL_ERROR( "checksum mismatch in filename decode" );
}

return finalSize;
}

bool BlockNameIO::Enabled()
{
    return true;
}

```

Файл « BlockNameIO.h»

```

#ifndef _BlockNameIO_incl_
#define _BlockNameIO_incl_

#include "NameIO.h"
#include "CipherKey.h"

#include <boost/shared_ptr.hpp>

class Cipher;

/*
    Implement NameIO interface for filename encoding. Uses cipher in block
    mode to encode filenames. The filenames are padded to be a multiple of the
    cipher block size.
*/
class BlockNameIO : public NameIO
{
public:
    static rel::Interface CurrentInterface(bool caseSensitive = false);

    BlockNameIO( const rel::Interface &iface,
        const boost::shared_ptr<Cipher> &cipher,
        const CipherKey &key, int blockSize,
        bool caseSensitiveEncoding = false );
    virtual ~BlockNameIO();

    virtual rel::Interface get_interface() const;

    virtual int maxEncodedNameLen( int plaintextNameLen ) const;
    virtual int maxDecodedNameLen( int encodedNameLen ) const;

    // hack to help with static builds
    static bool Enabled();
protected:
    virtual int encodeName( const char *plaintextName, int length,
        uint64_t *iv, char *encodedName ) const;
    virtual int decodeName( const char *encodedName, int length,
        uint64_t *iv, char *plaintextName ) const;

private:

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		44

```

    int _interface;
    int _bs;
    boost::shared_ptr<Cipher> _cipher;
    CipherKey _key;
    bool _caseSensitive;
};

```

```

#endif

```

Файл « Cipher.h »

```

#ifndef _Cipher_incl_
#define _Cipher_incl_

#include "encfs.h"

#include "Range.h"
#include "Interface.h"
#include "CipherKey.h"

#include <string>
#include <list>

#if defined(HAVE_INTTYPES_H)
#include <inttypes.h>
#endif

#if defined(HAVE_STDINT_H)
#include <stdint.h>
#endif

using boost::shared_ptr;

/*
    Mostly pure virtual interface defining operations on a cipher.

    Cipher's should register themselves so they can be instanciated via
    Cipher::New().
*/
class Cipher
{
public:
    // if no key length was indicated when cipher was registered, then keyLen
    // <= 0 will be used.
    typedef boost::shared_ptr<Cipher> (*CipherConstructor)( const rel::Interface &iface,
                                                             int keyLenBits );

    struct CipherAlgorithm
    {
        std::string name;
        std::string description;
        rel::Interface iface;
        Range keyLength;
        Range blockSize;
    };

    typedef std::list<CipherAlgorithm> AlgorithmList;
    static AlgorithmList GetAlgorithmList( bool includeHidden = false );

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		45

```

static boost::shared_ptr<Cipher> New( const rel::Interface &iface,
                                     int keyLen = -1);
static boost::shared_ptr<Cipher> New( const std::string &cipherName,
                                     int keyLen = -1 );

static bool Register(const char *cipherName,
                    const char *description,
                    const rel::Interface &iface,
                    CipherConstructor constructor,
                    bool hidden = false);
static bool Register(const char *cipherName,
                    const char *description,
                    const rel::Interface &iface,
                    const Range &keyLength, const Range &blockSize,
                    CipherConstructor constructor,
                    bool hidden = false);

Cipher();
virtual ~Cipher();

virtual rel::Interface get_interface() const =0;

// create a new key based on a password
// if iterationCount == 0, then iteration count will be determined
// by newKey function and filled in.
// If iterationCount == 0, then desiredFunctionDuration is how many
// milliseconds the password derivation function should take to run.
virtual CipherKey newKey(const char *password, int passwdLength,
                        int &iterationCount, long desiredFunctionDuration,
                        const unsigned char *salt, int saltlen) =0;
// deprecated - for backward compatibility
virtual CipherKey newKey(const char *password, int passwdLength ) =0;
// create a new random key
virtual CipherKey newRandomKey() =0;

// data must be len encodedKeySize()
virtual CipherKey readKey(const unsigned char *data,
                        const CipherKey &encodingKey,
                        bool checkKey = true) =0;
virtual void writeKey(const CipherKey &key, unsigned char *data,
                    const CipherKey &encodingKey) =0;

virtual std::string encodeAsString(const CipherKey &key,
                                const CipherKey &encodingKey );

// for testing purposes
virtual bool compareKey( const CipherKey &A, const CipherKey &B ) const =0;

// meta-data about the cypher
virtual int keySize() const=0;
virtual int encodedKeySize() const=0; // size
virtual int cipherBlockSize() const=0; // size of a cipher block

// fill the supplied buffer with random data
// The data may be pseudo random and might not be suitable for key
// generation. For generating keys, uses newRandomKey() instead.
// Returns true on success, false on failure.
virtual bool randomize( unsigned char *buf, int len,
                      bool strongRandom ) const =0;

// 64 bit MAC of the data with the given key
virtual uint64_t MAC_64( const unsigned char *src, int len,

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		46

```

        const CipherKey &key, uint64_t *chainedIV = 0 ) const =0;
// based on reductions of MAC_64
unsigned int MAC_32( const unsigned char *src, int len,
        const CipherKey &key, uint64_t *chainedIV = 0 ) const;
unsigned int MAC_16( const unsigned char *src, int len,
        const CipherKey &key, uint64_t *chainedIV = 0 ) const;

// functional interfaces
/*
    Stream encoding of data in-place. The stream data can be any length.
*/
virtual bool streamEncode( unsigned char *data, int len,
        uint64_t iv64, const CipherKey &key) const=0;
virtual bool streamDecode( unsigned char *data, int len,
        uint64_t iv64, const CipherKey &key) const=0;

/*
    These are just aliases of streamEncode / streamDecode, but there are
    provided here for backward compatibility for earlier ciphers that has
    effectively two stream modes - one for encoding partial blocks and
    another for encoding filenames.
*/
virtual bool nameEncode( unsigned char *data, int len,
        uint64_t iv64, const CipherKey &key) const;
virtual bool nameDecode( unsigned char *data, int len,
        uint64_t iv64, const CipherKey &key) const;

/*
    Block encoding of data in-place. The data size should be a multiple of
    the cipher block size.
*/
virtual bool blockEncode(unsigned char *buf, int size,
        uint64_t iv64, const CipherKey &key) const=0;
virtual bool blockDecode(unsigned char *buf, int size,
        uint64_t iv64, const CipherKey &key) const=0;
};

#endif

```

Файл « Cipher.cpp »

```

#include "config.h"

#include "Cipher.h"
#include "Interface.h"
#include "Range.h"
#include "base64.h"

#include <map>
#include <list>
#include <string>
#include <iostream>

// for static build. Need to reference the modules which are registered at
// run-time, to ensure that the linker doesn't optimize them away.
#include "NullCipher.h"
#include "SSL_Cipher.h"

using namespace std;
using namespace rel;
using boost::shared_ptr;

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		47

```

#define REF_MODULE(TYPE) \
    if( !TYPE::Enabled() ) \
        cerr << "referenceModule: should never happen\n";

static
void AddSymbolReferences()
{
    REF_MODULE(SSL_Cipher)
    REF_MODULE(NullCipher)
}

struct CipherAlg
{
    bool hidden;
    Cipher::CipherConstructor constructor;
    string description;
    Interface iface;
    Range keyLength;
    Range blockSize;
};

typedef multimap< string, CipherAlg> CipherMap_t;
static CipherMap_t *gCipherMap = NULL;

std::list<Cipher::CipherAlgorithm>
Cipher::GetAlgorithmList( bool includeHidden )
{
    AddSymbolReferences();

    list<CipherAlgorithm> result;

    if(!gCipherMap)
        return result;

    CipherMap_t::const_iterator it;
    CipherMap_t::const_iterator mapEnd = gCipherMap->end();
    for(it = gCipherMap->begin(); it != mapEnd; ++it)
    {
        if(includeHidden || !it->second.hidden)
        {
            CipherAlgorithm tmp;
            tmp.name = it->first;
            tmp.description = it->second.description;
            tmp.iface = it->second.iface;
            tmp.keyLength = it->second.keyLength;
            tmp.blockSize = it->second.blockSize;

            result.push_back( tmp );
        }
    }

    return result;
}

bool Cipher::Register(const char *name, const char *description,
    const Interface &iface, CipherConstructor fn, bool hidden)
{
    Range keyLength(-1,-1,1);
    Range blockSize(-1,-1,1);
    return Cipher::Register( name, description, iface,
        keyLength, blockSize, fn, hidden );
}

```

```

bool Cipher::Register(const char *name, const char *description,
    const Interface &iface, const Range &keyLength,
    const Range &blockSize,
    CipherConstructor fn, bool hidden)
{
    if(!gCipherMap)
        gCipherMap = new CipherMap_t;

    CipherAlg ca;
    ca.hidden = hidden;
    ca.constructor = fn;
    ca.description = description;
    ca.iface = iface;
    ca.keyLength = keyLength;
    ca.blockSize = blockSize;

    gCipherMap->insert( make_pair(string(name), ca) );
    return true;
}

boost::shared_ptr<Cipher> Cipher::New(const string &name, int keyLen)
{
    boost::shared_ptr<Cipher> result;

    if(gCipherMap)
    {
        CipherMap_t::const_iterator it = gCipherMap->find( name );
        if(it != gCipherMap->end())
        {
            CipherConstructor fn = it->second.constructor;
            // use current interface..
            result = (*fn)( it->second.iface, keyLen );
        }
    }

    return result;
}

boost::shared_ptr<Cipher> Cipher::New( const Interface &iface, int keyLen )
{
    boost::shared_ptr<Cipher> result;
    if(gCipherMap)
    {
        CipherMap_t::const_iterator it;
        CipherMap_t::const_iterator mapEnd = gCipherMap->end();

        for(it = gCipherMap->begin(); it != mapEnd; ++it)
        {
            // TODO: we should look for the newest implementation..
            if( it->second.iface.implements( iface ) )
            {
                CipherConstructor fn = it->second.constructor;
                // pass in requested interface..
                result = (*fn)( iface, keyLen );

                // if we're not going to compare the options, then just stop
                // now..
                break;
            }
        }
    }

    return result;
}

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		49

```

Cipher::Cipher()
{
}

Cipher::~~Cipher()
{
}

unsigned int Cipher::MAC_32( const unsigned char *src, int len,
    const CipherKey &key, uint64_t *chainedIV ) const
{
    uint64_t mac64 = MAC_64( src, len, key, chainedIV );

    unsigned int mac32 = ((mac64 >> 32) & 0xffffffff) ^ (mac64 & 0xffffffff);

    return mac32;
}

unsigned int Cipher::MAC_16( const unsigned char *src, int len,
    const CipherKey &key, uint64_t *chainedIV ) const
{
    uint64_t mac64 = MAC_64( src, len, key, chainedIV );

    unsigned int mac32 = ((mac64 >> 32) & 0xffffffff) ^ (mac64 & 0xffffffff);
    unsigned int mac16 = ((mac32 >> 16) & 0xffff) ^ (mac32 & 0xffff);

    return mac16;
}

bool Cipher::nameEncode( unsigned char *data, int len,
    uint64_t iv64, const CipherKey &key ) const
{
    return streamEncode( data, len, iv64, key );
}

bool Cipher::nameDecode( unsigned char *data, int len,
    uint64_t iv64, const CipherKey &key ) const
{
    return streamDecode( data, len, iv64, key );
}

string Cipher::encodeAsString(const CipherKey &key,
    const CipherKey &encodingKey )
{
    int encodedKeySize = this->encodedKeySize();
    unsigned char *keyBuf = new unsigned char[ encodedKeySize ];

    // write the key, encoding it with itself.
    this->writeKey( key, keyBuf, key );

    int b64Len = B256ToB64Bytes( encodedKeySize );
    unsigned char *b64Key = new unsigned char[ b64Len + 1 ];

    changeBase2( keyBuf, encodedKeySize, 8, b64Key,
        b64Len, 6 );
    B64ToAscii( b64Key, b64Len );
    b64Key[ b64Len - 1 ] = '\\0';

    return string( (const char *)b64Key );
}

```

Файл « CipherFileIO.cpp»

```
#include "CipherFileIO.h"

#include "Cipher.h"
#include "MemoryPool.h"

#include <rlog/rlog.h>
#include <rlog/Error.h>

#include <fcntl.h>
#include <cerrno>

using boost::shared_ptr;

/*
- Version 2:0 adds support for a per-file initialization vector with a
fixed 8 byte header. The headers are enabled globally within a
filesystem at the filesystem configuration level.
When headers are disabled, 2:0 is compatible with version 1:0.
*/
static rel::Interface CipherFileIO_iface("FileIO/Cipher", 2, 0, 1);

const int HEADER_SIZE = 8; // 64 bit initialization vector..

static bool checkSize( int fsBlockSize, int cipherBlockSize )
{
    int blockBoundary = fsBlockSize % cipherBlockSize ;
    if(blockBoundary != 0)
    {
        rError("CipherFileIO: blocks should be multiple of cipher block size");
        return true;
    } else
        return false;
}

CipherFileIO::CipherFileIO( const shared_ptr<FileIO> &_base,
                           const FSConfigPtr &cfg )
    : BlockFileIO( cfg->config->blockSize, cfg )
    , base( _base )
    , haveHeader( cfg->config->uniqueIV )
    , externalIV( 0 )
    , fileIV( 0 )
    , lastFlags( 0 )
{
    fsConfig = cfg;
    cipher = cfg->cipher;
    key = cfg->key;

    static bool warnOnce = false;

    if(!warnOnce)
        warnOnce = checkSize( fsConfig->config->blockSize,
                              fsConfig->cipher->cipherBlockSize() );
}

CipherFileIO::~CipherFileIO()
{
}

rel::Interface CipherFileIO::get_interface() const
{
    return CipherFileIO_iface;
}
```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		51

```

}

int CipherFileIO::open( int flags )
{
    int res = base->open( flags );

    if( res >= 0 )
        lastFlags = flags;

    return res;
}

void CipherFileIO::setFileName( const char *fileName )
{
    base->setFileName( fileName );
}

const char *CipherFileIO::getFileName() const
{
    return base->getFileName();
}

bool CipherFileIO::setIV( uint64_t iv )
{
    rDebug("in setIV, current IV = %" PRIu64 " ", new IV = %" PRIu64
        ", fileIV = %" PRIu64,
        externalIV, iv, fileIV);
    if(externalIV == 0)
    {
        // we're just being told about which IV to use. since we haven't
        // initialized the fileIV, there is no need to just yet..
        externalIV = iv;
        if(fileIV != 0)
        {
            rWarning("fileIV initialized before externalIV! (%" PRIu64
                ", %" PRIu64 ")", fileIV, externalIV);
        }
    } else
    if(haveHeader)
    {
        // we have an old IV, and now a new IV, so we need to update the fileIV
        // on disk.
        if(fileIV == 0)
        {
            // ensure the file is open for read/write..
            int newFlags = lastFlags | O_RDWR;
            int res = base->open( newFlags );
            if(res < 0)
            {
                if(res == -EISDIR)
                {
                    // duh -- there are no file headers for directories!
                    externalIV = iv;
                    return base->setIV( iv );
                } else
                {
                    rDebug("writeHeader failed to re-open for write");
                    return false;
                }
            }
        }
        initHeader();
    }

    uint64_t oldIV = externalIV;

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		52

```

        externalIV = iv;
        if(!writeHeader())
        {
            externalIV = oldIV;
            return false;
        }
    }

    return base->setIV( iv );
}

int CipherFileIO::getAttr( efs_stat *stbuf, void *statCache ) const
{
    int res = base->getAttr( stbuf, statCache );
    // adjust size if we have a file header
    if((res == 0) && haveHeader &&
        S_ISREG(stbuf->st_mode) && (stbuf->st_size > 0))
    {
        rAssert(stbuf->st_size >= HEADER_SIZE);
        stbuf->st_size -= HEADER_SIZE;
    }

    return res;
}

efs_off_t CipherFileIO::getSize() const
{
    efs_off_t size = base->getSize();
    // No check on S_ISREG here -- don't call getSize over getAttr unless this
    // is a normal file!
    if(haveHeader && size > 0)
    {
        rAssert(size >= HEADER_SIZE);
        size -= HEADER_SIZE;
    }
    return size;
}

void CipherFileIO::initHeader( )
{
    // check if the file has a header, and read it if it does.. Otherwise,
    // create one.
    efs_off_t rawSize = base->getSize();
    if(rawSize >= HEADER_SIZE)
    {
        rDebug("reading existing header, rawSize = %" PRIi64, rawSize);
        // has a header.. read it
        unsigned char buf[8] = {0};

        IORequest req;
        req.offset = 0;
        req.data = buf;
        req.dataLen = 8;
        base->read( req );

        cipher->streamDecode( buf, sizeof(buf),
                               externalIV, key );

        fileIV = 0;
        for(int i=0; i<8; ++i)
            fileIV = (fileIV << 8) | (uint64_t)buf[i];

        rAssert(fileIV != 0); // 0 is never used..
    } else

```

```

{
    rDebug("creating new file IV header");

    unsigned char buf[8] = {0};
    do
    {
        if(!cipher->randomize( buf, 8, false ))
            throw RL_ERROR("Unable to generate a random file IV");

        fileIV = 0;
        for(int i=0; i<8; ++i)
            fileIV = (fileIV << 8) | (uint64_t)buf[i];

        if(fileIV == 0)
        {
            rWarning("Unexpected result: randomize returned 8 null bytes!");
        }
    } while(fileIV == 0); // don't accept 0 as an option..

    if( base->isWritable() )
    {
        cipher->streamEncode( buf, sizeof(buf), externalIV, key );

        IORequest req;
        req.offset = 0;
        req.data = buf;
        req.dataLen = 8;

        base->write( req );
    } else
    {
        rDebug("base not writable, IV not written..");
    }
}
rDebug("initHeader finished, fileIV = %" PRIu64 , fileIV);
}

bool CipherFileIO::writeHeader( )
{
    if( !base->isWritable() )
    {
        // open for write..
        int newFlags = lastFlags | O_RDWR;
        if( base->open( newFlags ) < 0 )
        {
            rDebug("writeHeader failed to re-open for write");
            return false;
        }
    }

    if(fileIV == 0)
    {
        rError("Internal error: fileIV == 0 in writeHeader!!!");
    }
    rDebug("writing fileIV %" PRIu64 , fileIV);

    unsigned char buf[8] = {0};
    for(int i=0; i<8; ++i)
    {
        buf[sizeof(buf)-1-i] = (unsigned char)(fileIV & 0xff);
        fileIV >>= 8;
    }

    cipher->streamEncode( buf, sizeof(buf), externalIV, key );

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		54

```

IORequest req;
req.offset = 0;
req.data = buf;
req.dataLen = 8;

base->write( req );

return true;
}

ssize_t CipherFileIO::readOneBlock( const IORequest &req ) const
{
    // read raw data, then decipher it..
    int bs = blockSize();
    efs_off_t blockNum = req.offset / bs;

    ssize_t readSize = 0;
    IORequest tmpReq = req;

    if(haveHeader)
        tmpReq.offset += HEADER_SIZE;
    readSize = base->read( tmpReq );

    bool ok;
    if(readSize > 0)
    {
        if(haveHeader && fileIV == 0)
            const_cast<CipherFileIO*>(this)->initHeader();

        if(readSize != bs)
        {
            ok = streamRead( tmpReq.data, (int)readSize, blockNum ^ fileIV);
        } else
        {
            ok = blockRead( tmpReq.data, (int)readSize, blockNum ^ fileIV);
        }

        if(!ok)
        {
            rDebug("decodeBlock failed for block %" PRIi64 " , size %i",
                blockNum, (int)readSize );
            readSize = -1;
        }
    } else
    {
        rDebug("readSize zero for offset %" PRIi64, req.offset);
    }
    return readSize;
}

bool CipherFileIO::writeOneBlock( const IORequest &req )
{
    int bs = blockSize();
    efs_off_t blockNum = req.offset / bs;

    if(haveHeader && fileIV == 0)
        initHeader();

    bool ok;
    if( req.dataLen != bs )
    {
        ok = streamWrite( req.data, (int)req.dataLen,

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		55

```

        blockNum ^ fileIV );
    } else
    {
        ok = blockWrite( req.data, (int)req.dataLen,
            blockNum ^ fileIV );
    }

    if( ok )
    {
        if(haveHeader)
        {
            IORequest tmpReq = req;
            tmpReq.offset += HEADER_SIZE;
            ok = base->write( tmpReq );
        } else
            ok = base->write( req );
    } else
    {
        rDebug("encodeBlock failed for block %" PRIi64 " , size %i",
            blockNum, req.dataLen);
        ok = false;
    }
    return ok;
}

bool CipherFileIO::blockWrite( unsigned char *buf, int size,
    uint64_t _iv64 ) const
{
    if (!fsConfig->reverseEncryption)
        return cipher->blockEncode( buf, size, _iv64, key );
    else
        return cipher->blockDecode( buf, size, _iv64, key );
}

bool CipherFileIO::streamWrite( unsigned char *buf, int size,
    uint64_t _iv64 ) const
{
    if (!fsConfig->reverseEncryption)
        return cipher->streamEncode( buf, size, _iv64, key );
    else
        return cipher->streamDecode( buf, size, _iv64, key );
}

bool CipherFileIO::blockRead( unsigned char *buf, int size,
    uint64_t _iv64 ) const
{
    if (fsConfig->reverseEncryption)
        return cipher->blockEncode( buf, size, _iv64, key );
    else
    {
        if(_allowHoles)
        {
            // special case - leave all 0's alone
            for(int i=0; i<size; ++i)
                if(buf[i] != 0)
                    return cipher->blockDecode( buf, size, _iv64, key );

            return true;
        } else
            return cipher->blockDecode( buf, size, _iv64, key );
    }
}

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		56

```

bool CipherFileIO::streamRead( unsigned char *buf, int size,
                               uint64_t _iv64 ) const
{
    if (fsConfig->reverseEncryption)
        return cipher->streamEncode( buf, size, _iv64, key );
    else
        return cipher->streamDecode( buf, size, _iv64, key );
}

int CipherFileIO::truncate( efs_off_t size )
{
    int res = 0;
    if(!haveHeader)
    {
        res = BlockFileIO::truncate( size, base.get() );
    } else
    {
        if(0 == fileIV)
        {
            // empty file.. create the header..
            if( !base->isWritable() )
            {
                // open for write..
                int newFlags = lastFlags | O_RDWR;
                if( base->open( newFlags ) < 0 )
                {
                    rDebug("writeHeader failed to re-open for write");
                }
            }
            initHeader();
        }

        // can't let BlockFileIO call base->truncate(), since it would be using
        // the wrong size..
        res = BlockFileIO::truncate( size, 0 );

        if(res == 0)
            base->truncate( size + HEADER_SIZE );
    }
    return res;
}

bool CipherFileIO::isWritable() const
{
    return base->isWritable();
}

```

Файл « CipherFileIO.h»

```

#ifndef _CipherFileIO_incl_
#define _CipherFileIO_incl_

#include "BlockFileIO.h"
#include "CipherKey.h"
#include "FileUtils.h"

#if defined(HAVE_INTTYPES_H)
#include <inttypes.h>
#endif

#if defined(HAVE_STDINT_H)

```

									Лист
									57
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 2.090301.116.001 ПЗ				

```

#include <stdint.h>
#endif

class Cipher;

/*
    Implement the FileIO interface encrypting data in blocks.

    Uses BlockFileIO to handle the block scatter / gather issues.
*/
class CipherFileIO : public BlockFileIO
{
public:
    CipherFileIO( const boost::shared_ptr<FileIO> &base,
                  const FSConfigPtr &cfg);
    virtual ~CipherFileIO();

    virtual rel::Interface get_interface() const;

    virtual void setFileName( const char *fileName );
    virtual const char *getFileName() const;
    virtual bool setIV( uint64_t iv );

    virtual int open( int flags );

    virtual int getAttr( efs_stat *stbuf, void *statCache ) const;
    virtual efs_off_t getSize() const;

    virtual int truncate( efs_off_t size );

    virtual bool isWritable() const;

private:
    virtual ssize_t readOneBlock( const IORequest &req ) const;
    virtual bool writeOneBlock( const IORequest &req );

    void initHeader();
    bool writeHeader();
    bool blockRead( unsigned char *buf, int size,
                    uint64_t iv64 ) const;
    bool streamRead( unsigned char *buf, int size,
                     uint64_t iv64 ) const;
    bool blockWrite( unsigned char *buf, int size,
                     uint64_t iv64 ) const;
    bool streamWrite( unsigned char *buf, int size,
                      uint64_t iv64 ) const;

    boost::shared_ptr<FileIO> base;

    FSConfigPtr fsConfig;

    // if haveHeader is true, then we have a transparent file header which
    // contains a 64 bit initialization vector.
    bool haveHeader;
    bool externalIVChaining;
    uint64_t externalIV;
    uint64_t fileIV;
    int lastFlags;

    boost::shared_ptr<Cipher> cipher;
    CipherKey key;
};

#endif

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		58

Файл « ConfigReader.cpp»

```
#include "ConfigReader.h"
#include "fs_layer.h"

#include <rlog/rlog.h>

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#ifdef _WIN32
#include <unistd.h>
#endif
#include <cstring>

using namespace std;
using namespace rlog;

ConfigReader::ConfigReader()
{
}

ConfigReader::~ConfigReader()
{
}

// read the entire file into a ConfigVar instance and then use that to decode
// into mapped variables.
bool
ConfigReader::load(const char *fileName)
{
    efs_stat stbuf;
    memset( &stbuf, 0, sizeof(efs_stat));
    if( fs_layer::lstat( fileName, &stbuf ) != 0)
        return false;

    int size = stbuf.st_size;

    int fd = fs_layer::open( fileName, O_RDONLY );
    if(fd < 0)
        return false;

    char *buf = new char[size];

    int res = fs_layer::read( fd, buf, size );
    fs_layer::close( fd );

    if( res != size )
    {
        rWarning("Partial read of config file, expecting %i bytes, got %i",
            size, res);
        delete[] buf;
        return false;
    }

    ConfigVar in;
    in.write( (unsigned char *)buf, size );
    delete[] buf;

    return loadFromVar( in );
}
```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		59

```

bool
ConfigReader::loadFromVar(ConfigVar &in)
{
    in.resetOffset();

    // parse.
    int numEntries = in.readInt();

    for(int i=0; i<numEntries; ++i)
    {
        string key, value;
        in >> key >> value;

        if(key.length() == 0)
        {
            rError("Invalid key encoding in buffer");
            return false;
        }
        ConfigVar newVar( value );
        vars.insert( make_pair( key, newVar ) );
    }

    return true;
}

bool
ConfigReader::save(const char *fileName) const
{
    // write everything to a ConfigVar, then output to disk
    ConfigVar out = toVar();

    int fd = fs_layer::open( fileName, O_RDWR | O_CREAT, 0640 );
    if(fd >= 0)
    {
        int retVal = fs_layer::write( fd, out.buffer(), out.size() );
        fs_layer::close( fd );
        if(retVal != out.size())
        {
            rError("Error writing to config file %s", fileName);
            return false;
        }
    } else
    {
        rError("Unable to open or create file %s", fileName);
        return false;
    }

    return true;
}

ConfigVar
ConfigReader::toVar() const
{
    // write everything to a ConfigVar, then output to disk
    ConfigVar out;
    out.writeInt( vars.size() );
    map<string, ConfigVar>::const_iterator it;
    for(it = vars.begin(); it != vars.end(); ++it)
    {
        out.writeInt( it->first.size() );
        out.write( (unsigned char*)it->first.data(), it->first.size() );
        out.writeInt( it->second.size() );
        out.write( (unsigned char*)it->second.buffer(), it->second.size() );
    }
}

```

```

        return out;
    }

ConfigVar ConfigReader::operator[] ( const std::string &varName ) const
{
    // read only
    map<string, ConfigVar>::const_iterator it = vars.find( varName );
    if( it == vars.end() )
        return ConfigVar();
    else
        return it->second;
}

ConfigVar &ConfigReader::operator[] ( const std::string &varName )
{
    return vars[ varName ];
}

```

Файл « ConfigReader.h»

```

#ifndef _ConfigReader_incl_
#define _ConfigReader_incl_

#include <string>
#include <map>

#include "ConfigVar.h"

/*
    handles Configuration load / store for Encfs filesystems.

    loading existing config file example:

    ConfigReader cfg;
    cfg.load( filesystemConfigFile );

    Interface iface;
    cfg["cipher"] >> iface;

    creating new config example:

    ConfigReader cfg;
    cfg["cipher"] << cipher->get_interface();
*/
class ConfigReader
{
public:
    ConfigReader();
    ~ConfigReader();

    bool load(const char *fileName);
    bool save(const char *fileName) const;

    ConfigVar toVar() const;
    bool loadFromVar( ConfigVar &var );

    ConfigVar operator[](const std::string &varName) const;
    ConfigVar &operator[](const std::string &varName);

private:

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		61

```

        std::map<std::string, ConfigVar> vars;
    };

#endif

```

Файл « ConfigVar.cpp»

```

#include "ConfigVar.h"
#include <rlog/rlog.h>

#include <cstring>

using namespace rlog;

#ifdef MIN
inline int MIN(int a, int b)
{
    return (a < b) ? a : b;
}
#endif

ConfigVar::ConfigVar()
    : pd( new ConfigVarData )
{
    pd->offset = 0;
}

ConfigVar::ConfigVar(const std::string &buf)
    : pd( new ConfigVarData )
{
    pd->buffer = buf;
    pd->offset = 0;
}

ConfigVar::ConfigVar(const ConfigVar &src)
{
    pd = src.pd;
}

ConfigVar::~~ConfigVar()
{
    pd.reset();
}

ConfigVar & ConfigVar::operator = (const ConfigVar &src)
{
    if(src.pd == pd)
        return *this;
    else
        pd = src.pd;

    return *this;
}

void ConfigVar::resetOffset()
{
    pd->offset = 0;
}

int ConfigVar::read(unsigned char *buffer_, int bytes) const

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		62

```

{
    int toCopy = MIN( bytes, pd->buffer.size() - pd->offset );

    if(toCopy > 0)
        memcpy( buffer_, pd->buffer.data() + pd->offset, toCopy );

    pd->offset += toCopy;

    return toCopy;
}

int ConfigVar::write(const unsigned char *data, int bytes)
{
    if(pd->buffer.size() == (unsigned int)pd->offset)
    {
        pd->buffer.append( (const char *)data, bytes );
    } else
    {
        pd->buffer.insert( pd->offset, (const char *)data, bytes );
    }

    pd->offset += bytes;

    return bytes;
}

int ConfigVar::size() const
{
    return pd->buffer.size();
}

const char *ConfigVar::buffer() const
{
    return pd->buffer.data();
}

int ConfigVar::at() const
{
    return pd->offset;
}

void ConfigVar::writeString(const char *data, int bytes)
{
    writeInt( bytes );
    write( (const unsigned char *)data, bytes );
}

// convert integer to BER encoded integer
void ConfigVar::writeInt(int val)
{
    // we can represent 7 bits per char output, so a 32bit number may take up
    // to 5 bytes.
    // first byte:      0x0000007f                                0111,1111
    // second byte:     0x00003f80                                0011,1111 1000,0000
    // third byte:      0x001fb000                                0000,0000 0001,1111 1100,0000 0000,0000
    // fourth byte:     0x0fe00000                                0000,1111 1110,0000
    // fifth byte:      0xf0000000                                1111,0000
    unsigned char digit[5];

    digit[4] = (unsigned char)((val & 0x0000007f));
    digit[3] = 0x80 | (unsigned char)((val & 0x00003f80) >> 7);
    digit[2] = 0x80 | (unsigned char)((val & 0x001fc000) >> 14);
    digit[1] = 0x80 | (unsigned char)((val & 0x0fe00000) >> 21);
}

```

```

digit[0] = 0x80 | (unsigned char)((val & 0xf0000000) >> 28);

// find the starting point - we only need to output starting at the most
// significant non-zero digit..
int start = 0;
while(digit[start] == 0x80)
    ++start;

write( digit + start, 5-start );
}

int ConfigVar::readInt() const
{
    const unsigned char * buf = (const unsigned char *)buffer();
    int bytes = this->size();
    int offset = at();
    int value = 0;
    bool highBitSet;

    rAssert( offset < bytes );

    do
    {
        unsigned char tmp = buf[offset++];
        highBitSet = tmp & 0x80;

        value = (value << 7) | (int)(tmp & 0x7f);
    } while(highBitSet && offset < bytes);

    pd->offset = offset;

    // should never end up with a negative number..
    rAssert( value >= 0 );

    return value;
}

int ConfigVar::readInt( int defaultValue ) const
{
    int bytes = this->size();
    int offset = at();

    if(offset >= bytes)
        return defaultValue;
    else
        return readInt();
}

bool ConfigVar::readBool( bool defaultValue ) const
{
    int tmp = readInt( defaultValue ? 1 : 0 );
    return (tmp != 0);
}

ConfigVar & operator << (ConfigVar &src, bool value)
{
    src.writeInt( value ? 1 : 0 );
    return src;
}

ConfigVar & operator << (ConfigVar &src, int var)
{
    src.writeInt( var );
    return src;
}

```

```

}

ConfigVar & operator << (ConfigVar &src, const std::string &str)
{
    src.writeString( str.data(), str.length() );
    return src;
}

const ConfigVar & operator >> (const ConfigVar &src, bool &result)
{
    int tmp = src.readInt();
    result = (tmp != 0);
    return src;
}

const ConfigVar & operator >> (const ConfigVar &src, int &result)
{
    result = src.readInt();
    return src;
}

const ConfigVar & operator >> (const ConfigVar &src, std::string &result)
{
    int length = src.readInt();
    //rAssert(length > 0);

    int readLen;

    unsigned char tmpBuf[32];
    if(length > (int)sizeof(tmpBuf))
    {
        unsigned char *ptr = new unsigned char[length];
        readLen = src.read( ptr, length );
        result.assign( (char*)ptr, length );
        delete[] ptr;
    } else
    {
        readLen = src.read( tmpBuf, length );
        result.assign( (char*)tmpBuf, length );
    }

    if(readLen != length)
    {
        rDebug("string encoded as size %i bytes, read %i", length, readLen );
    }
    rAssert(readLen == length);

    return src;
}

```

Файл « ConfigVar.h»

```

#ifndef _ConfigVar_incl_
#define _ConfigVar_incl_

#include <string>
#include <boost/shared_ptr.hpp>

using boost::shared_ptr;

class ConfigVar
{
    struct ConfigVarData

```

									Лист
									65
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 2.090301.116.001 ПЗ				

```

{
    std::string buffer;
    int offset;
};

shared_ptr<ConfigVarData> pd;

public:
    ConfigVar();
    ConfigVar(const std::string &buffer);
    ConfigVar(const ConfigVar &src);
    ~ConfigVar();

    ConfigVar & operator = (const ConfigVar &src);

    // reset read/write offset..
    void resetOffset();

    // read bytes
    int read(unsigned char *buffer, int size) const;

    // write bytes..
    int write(const unsigned char *data, int size);

    int readInt() const;
    int readInt( int defaultValue ) const;
    void writeInt(int value);

    bool readBool( bool defaultValue ) const;

    void writeString(const char *data, int size);

    // return amount of data in var
    int size() const;
    // return data pointer - returns front of data pointer, not the current
    // position.
    const char *buffer() const;

    // return current position in data() buffer.
    int at() const;
};

ConfigVar & operator << (ConfigVar &, bool);
ConfigVar & operator << (ConfigVar &, int);
ConfigVar & operator << (ConfigVar &, const std::string &str);

const ConfigVar & operator >> (const ConfigVar &, bool &);
const ConfigVar & operator >> (const ConfigVar &, int &);
const ConfigVar & operator >> (const ConfigVar &, std::string &str);

#endif

```

Файл « Context.cpp»

```

#include "FileNode.h"
#include "Context.h"
#include "Mutex.h"
#include "FileUtils.h"
#include "DirNode.h"

#include <rlog/rlog.h>

using namespace rel;

```

									Лист
									66
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 2.090301.116.001 ПЗ				

```

using namespace rlog;

EncFS_Context::EncFS_Context() :
    wakeupCond(), wakeupMutex(), contextMutex()
{
    usageCount = 0;
}

EncFS_Context::~EncFS_Context()
{
    // release all entries from map
    openFiles.clear();
}

shared_ptr<DirNode> EncFS_Context::getRoot(int *errCode)
{
    shared_ptr<DirNode> ret;
    do
    {
        {
            Lock lock( contextMutex );
            ret = root;
            ++usageCount;
        }

        if(!ret)
        {
            std::ostringstream ostr;
            int res = remountFS( this, ostr );
            if(res != 0)
            {
                *errCode = res;
                break;
            }
        }
    } while(!ret);

    return ret;
}

void EncFS_Context::setRoot(const shared_ptr<DirNode> &r)
{
    Lock lock( contextMutex );

    root = r;
    if(r)
        rootCipherDir = r->rootDirectory();
}

bool EncFS_Context::isMounted()
{
    return root;
}

int EncFS_Context::getAndResetUsageCounter()
{
    Lock lock( contextMutex );

    int count = usageCount;
    usageCount = 0;

    return count;
}

```

```

int EncFS_Context::openFileCount() const
{
    Lock lock( contextMutex );

    return openFiles.size();
}

shared_ptr<FileNode> EncFS_Context::lookupNode(const char *path)
{
    Lock lock( contextMutex );

    FileMap::iterator it = openFiles.find( std::string(path) );
    if(it != openFiles.end())
    {
        // all the items in the set point to the same node.. so just use the
        // first
        return (*it->second.begin())->node;
    } else
    {
        return shared_ptr<FileNode>();
    }
}

void EncFS_Context::renameNode(const char *from, const char *to)
{
    Lock lock( contextMutex );

    FileMap::iterator it = openFiles.find( std::string(from) );
    if(it != openFiles.end())
    {
        std::set<Placeholder *> val = it->second;
        openFiles.erase(it);
        openFiles[ std::string(to) ] = val;
    }
}

shared_ptr<FileNode> EncFS_Context::getNode(void *p1)
{
    Placeholder *ph = (Placeholder*)p1;
    return ph->node;
}

void *EncFS_Context::putNode(const char *path,
    const boost::shared_ptr<FileNode> &node)
{
    Lock lock( contextMutex );
    Placeholder *p1 = new Placeholder( node );
    openFiles[ std::string(path) ].insert(p1);

    return (void *)p1;
}

void EncFS_Context::eraseNode(const char *path, void *p1)
{
    Lock lock( contextMutex );

    Placeholder *ph = (Placeholder *)p1;

    FileMap::iterator it = openFiles.find( std::string(path) );
    rAssert(it != openFiles.end());

    int rmCount = it->second.erase( ph );

    rAssert(rmCount == 1);
}

```

```

// if no more references to this file, remove the record all together
if(it->second.empty())
{
    // attempts to make use of shallow copy to clear memory used to hold
    // unencrypted filenames.. not sure this does any good..
    std::string storedName = it->first;
    openFiles.erase( it );
    storedName.assign( storedName.length(), '\\0' );
}

delete ph;
}

```

Файл « Context.h»

```

#ifndef _Context_incl_
#define _Context_incl_

#include "encfs.h"
#include <boost/shared_ptr.hpp>
#include <boost/thread.hpp>
#include <set>

#ifdef USE_HASHMAP
#include <ext/hash_map>
#else
#include <map>
#endif

using boost::shared_ptr;
struct EncFS_Args;
struct EncFS_Opts;
class FileNode;
class DirNode;

class EncFS_Context
{
public:
    EncFS_Context();
    ~EncFS_Context();

    shared_ptr<FileNode> getNode(void *ptr);
    shared_ptr<FileNode> lookupNode(const char *path);

    int getAndResetUsageCounter();
    int openFileCount() const;

    void *putNode(const char *path, const shared_ptr<FileNode> &node);

    void eraseNode(const char *path, void *placeholder);

    void renameNode(const char *oldName, const char *newName);

    void setRoot(const shared_ptr<DirNode> &root);
    shared_ptr<DirNode> getRoot(int *err);
    bool isMounted();

    shared_ptr<EncFS_Args> args;
    shared_ptr<EncFS_Opts> opts;
    bool publicFilesystem;

    // root path to cipher dir
    std::string rootCipherDir;
}

```

									Лист
									69
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 2.090301.116.001 ПЗ				

```

// for idle monitor
bool running;
boost::condition_variable wakeupCond;
boost::mutex wakeupMutex;

private:
/* This placeholder is what is referenced in FUSE context (passed to
 * callbacks).
 *
 * A FileNode may be opened many times, but only one FileNode instance per
 * file is kept. Rather than doing reference counting in FileNode, we
 * store a unique Placeholder for each open() until the corresponding
 * release() is called. shared_ptr then does our reference counting for
 * us.
 */
struct Placeholder
{
    shared_ptr<FileNode> node;

    Placeholder( const shared_ptr<FileNode> &ptr ) : node(ptr) {}
};

// set of open files, indexed by path
#ifdef USE_HASHMAP
typedef __gnu_cxx::hash_map<std::string,
    std::set<Placeholder*> > FileMap;
#else
typedef std::map< std::string,
    std::set<Placeholder*> > FileMap;
#endif

mutable boost::mutex contextMutex;
FileMap openFiles;

int usageCount;
shared_ptr<DirNode> root;
};

int remountFS( EncFS_Context *ctx, std::ostream &ostr );

#endif

```

Файл « DirNode.cpp»

```

#include "encfs.h"

#include "DirNode.h"
#include "FileUtils.h"

#include <sys/stat.h>
#include <sys/types.h>
#include <cerrno>
#include <cstdio>
#include <cstdlib>

#ifdef _MSC_VER
#include <direct.h>
#else
#include <pthread.h>
#include <unistd.h>
#endif
#ifdef HAVE_SYS_FSUID_H

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		70

```

#include <sys/fsuid.h>
#endif
#endif

#include <cstring>

#include "Context.h"
#include "Cipher.h"
#include "Mutex.h"
#include <rlog/rlog.h>
#include <rlog/Error.h>

#include <iostream>

using namespace std;
using namespace rel;
using namespace rlog;

static RLogChannel *Info = DEF_CHANNEL( "info/DirNode", Log_Info );

class DirDeleter
{
public:
    void operator () ( fs_layer::DIR *d )
    {
        fs_layer::closedir( d );
    }
};

DirTraverse::DirTraverse(const boost::shared_ptr<fs_layer::DIR> &_dirPtr,
    uint64_t _iv, const boost::shared_ptr<NameIO> &_naming)
: dir( _dirPtr )
, iv( _iv )
, naming( _naming )
{
}

DirTraverse::DirTraverse(const DirTraverse &src)
: dir( src.dir )
, iv( src.iv )
, naming( src.naming )
{
}

DirTraverse &DirTraverse::operator = (const DirTraverse &src)
{
    dir = src.dir;
    iv = src.iv;
    naming = src.naming;

    return *this;
}

DirTraverse::~DirTraverse()
{
    dir.reset();
    iv = 0;
    naming.reset();
}

static
bool _nextName(fs_layer::fs_dirent *&de, const boost::shared_ptr<fs_layer::DIR> &dir,
    int *fileType, ino_t *inode)

```

```

{
    de = fs_layer::readdir( dir.get() );

    if(de)
    {
        if(fileType)
        {
            #if defined(_DIRENT_HAVE_D_TYPE) || defined(__FreeBSD__)
                *fileType = de->d_type;
            #else
            #ifndef _WIN32
            #warning "struct dirent.d_type not supported"
            #endif
                *fileType = 0;
            #endif
        }
        if(inode)
            *inode = de->d_ino;
        return true;
    } else
    {
        if(fileType)
            *fileType = 0;
        return false;
    }
}

std::string DirTraverse::nextPlaintextName(int *fileType, ino_t *inode)
{
    fs_layer::fs_dirent *de=0;
    while(_nextName(de, dir, fileType, inode))
    {
        try
        {
            {
                uint64_t localIv = iv;
                return naming->decodePath( de->d_name, &localIv );
            } catch ( rlog::Error &ex )
            {
                // .. .problem decoding, ignore it and continue on to next name..
                rDebug("error decoding filename: %s", de->d_name);
            }
        }
    }

    return string();
}

std::string DirTraverse::nextInvalid()
{
    fs_layer::fs_dirent *de=0;
    // find the first name which produces a decoding error...
    while(_nextName(de, dir, (int*)0, (ino_t*)0))
    {
        try
        {
            {
                uint64_t localIv = iv;
                naming->decodePath( de->d_name, &localIv );
                continue;
            } catch( rlog::Error &ex )
            {
                return string( de->d_name );
            }
        }
    }
}

```

```

        return string();
    }

    struct RenameEl
    {
        // ciphertext names
        string oldCName;
        string newCName; // intermediate name (not final cname)

        // plaintext names
        string oldPName;
        string newPName;

        bool isDirectory;
    };

    class RenameOp
    {
    private:
        DirNode *dn;
        boost::shared_ptr< list<RenameEl> > renameList;
        list<RenameEl>::const_iterator last;

    public:
        RenameOp( DirNode *_dn, const boost::shared_ptr< list<RenameEl> > &_renameList )
            : dn(_dn), renameList(_renameList)
        {
            last = renameList->begin();
        }

        RenameOp(const RenameOp &src)
            : dn(src.dn)
            , renameList(src.renameList)
            , last(src.last)
        {
        }

        ~RenameOp();

        operator bool () const
        {
            return renameList;
        }

        bool apply();
        void undo();
    };

    RenameOp::~RenameOp()
    {
        if(renameList)
        {
            // got a bunch of decoded filenames sitting in memory.. do a little
            // cleanup before leaving..
            list<RenameEl>::iterator it;
            for(it = renameList->begin(); it != renameList->end(); ++it)
            {
                it->oldPName.assign( it->oldPName.size(), ' ' );
                it->newPName.assign( it->newPName.size(), ' ' );
            }
        }
    }

    bool RenameOp::apply()

```

```

{
    try
    {
        while(last != renameList->end())
        {
            // backing store rename.
            rDebug("renaming %s -> %s",
                last->oldCName.c_str(), last->newCName.c_str());

            efs_stat st;
            bool preserve_mtime = fs_layer::stat(last->oldCName.c_str(), &st) == 0;

            // internal node rename..
            dn->renameNode( last->oldPName.c_str(),
                last->newPName.c_str() );

            // rename on disk..
            if(fs_layer::rename( last->oldCName.c_str(),
                last->newCName.c_str() ) == -1)
            {
                rWarning("Error renaming %s: %s",
                    last->oldCName.c_str(), strerror( errno ));
                dn->renameNode( last->newPName.c_str(),
                    last->oldPName.c_str(), false );
                return false;
            }

            if(preserve_mtime)
            {
                struct utimbuf ut;
                ut.actime = st.st_atime;
                ut.modtime = st.st_mtime;
                fs_layer::utime(last->newCName.c_str(), &ut);
            }

            ++last;
        }

        return true;
    } catch( rlog::Error &err )
    {
        err.log( _RLWarningChannel );
        return false;
    }
}

void RenameOp::undo()
{
    rDebug("in undoRename");

    if(last == renameList->begin())
    {
        rDebug("nothing to undo");
        return; // nothing to undo
    }

    // list has to be processed backwards, otherwise we may rename
    // directories and directory contents in the wrong order!
    int undoCount = 0;
    list<RenameEl>::const_iterator it = last;

    while( it != renameList->begin() )
    {
        --it;
    }
}

```

```

        rDebug("undo: renaming %s -> %s",
                it->newCName.c_str(), it->oldCName.c_str());

        fs_layer::rename( it->newCName.c_str(), it->oldCName.c_str() );
        try
        {
            dn->renameNode( it->newPName.c_str(),
                            it->oldPName.c_str(), false );
        } catch( rlog::Error &err )
        {
            err.log( _RLWarningChannel );
            // continue on anyway...
        }
        ++undoCount;
    };

    rWarning("Undo rename count: %i", undoCount);
}

DirNode::DirNode(EncFS_Context *_ctx,
                  const string &sourceDir,
                  const FSConfigPtr &_config)
{
    Lock _lock( mutex );

    ctx = _ctx;
    rootDir = sourceDir;
    fsConfig = _config;

    // make sure rootDir ends in '/', so that we can form a path by appending
    // the rest..
    if( rootDir[ rootDir.length()-1 ] != '/' )
        rootDir.append( 1, '/' );

    naming = fsConfig->nameCoding;
}

DirNode::~DirNode()
{
}

bool
DirNode::hasDirectoryNameDependency() const
{
    return naming ? naming->getChainedNameIV() : false;
}

string
DirNode::rootDirectory()
{
    // don't update last access here, otherwise 'du' would cause lastAccess to
    // be reset.
    // chop off '/' terminator from root dir.
    return string( rootDir, 0, rootDir.length()-1 );
}

string
DirNode::cipherPath( const char *plaintextPath )
{
    return rootDir + naming->encodePath( plaintextPath );
}

string

```

```

DirNode::cipherPathWithoutRoot( const char *plaintextPath )
{
    return naming->encodePath( plaintextPath );
}

string
DirNode::plainPath( const char *cipherPath_ )
{
    try
    {
        if( !strncmp( cipherPath_, rootDir.c_str(),
                      rootDir.length() ) )
        {
            return naming->decodePath( cipherPath_ + rootDir.length() );
        } else
        {
            if ( cipherPath_[0] == '+' )
            {
                // decode as fully qualified path
                return string("/") + naming->decodeName( cipherPath_+1,
                                                            strlen(cipherPath_+1) );
            } else
            {
                return naming->decodePath( cipherPath_ );
            }
        }
    } catch( rlog::Error &err )
    {
        rError("decode err: %s", err.message());
        err.log( _RLWarningChannel );

        return string();
    }
}

string
DirNode::relativeCipherPath( const char *plaintextPath )
{
    try
    {
        if(plaintextPath[0] == '/')
        {
            // mark with '+' to indicate special decoding..
            return string("+") + naming->encodeName(plaintextPath+1,
                                                        strlen(plaintextPath+1));
        } else
        {
            return naming->encodePath( plaintextPath );
        }
    } catch( rlog::Error &err )
    {
        rError("encode err: %s", err.message());
        err.log( _RLWarningChannel );

        return string();
    }
}

DirTraverse DirNode::openDir(const char *plaintextPath)
{
    string cyName = rootDir + naming->encodePath( plaintextPath );
    //rDebug("openDir on %s", cyName.c_str() );
}

```

```

fs_layer::DIR *dir = fs_layer::opendir( cyName.c_str() );
if(dir == NULL)
{
    rDebug("opendir error %s", strerror(errno));
    return DirTraverse( boost::shared_ptr<fs_layer::DIR>(), 0,
boost::shared_ptr<NameIO>() );
} else
{
    boost::shared_ptr<fs_layer::DIR> dp( dir, DirDeleter() );

    uint64_t iv = 0;
    // if we're using chained IV mode, then compute the IV at this
    // directory level..
    try
    {
        if( naming->getChainedNameIV() )
            naming->encodePath( plaintextPath, &iv );
    } catch( rlog::Error &err )
    {
        rError("encode err: %s", err.message());
        err.log( _RLWarningChannel );
    }
    return DirTraverse( dp, iv, naming );
}
}

bool DirNode::genRenameList( list<RenameEl> &renameList,
    const char *fromP, const char *toP )
{
    uint64_t fromIV = 0, toIV = 0;

    // compute the IV for both paths
    string fromCPart = naming->encodePath( fromP, &fromIV );
    string toCPart = naming->encodePath( toP, &toIV );

    // where the files live before the rename..
    string sourcePath = rootDir + fromCPart;

    // ok..... we wish it was so simple.. should almost never happen
    if(fromIV == toIV)
        return true;

    // generate the real destination path, where we expect to find the files..
    rDebug("opendir %s", sourcePath.c_str() );
    boost::shared_ptr<fs_layer::DIR> dir = boost::shared_ptr<fs_layer::DIR>(
        fs_layer::opendir( sourcePath.c_str() ), DirDeleter() );
    if(!dir)
        return false;

    fs_layer::fs_dirent *de = NULL;
    while((de = fs_layer::readdir( dir.get() )) != NULL)
    {
        // decode the name using the oldIV
        uint64_t localIV = fromIV;
        string plainName;

        if((de->d_name[0] == '.') &&
            ((de->d_name[1] == '\0')
            || ((de->d_name[1] == '.') && (de->d_name[2] == '\0'))))
        {
            // skip "." and ".."
            continue;
        }
    }
}

```

```

try
{
    plainName = naming->decodePath( de->d_name, &localIV );
} catch( rlog::Error &ex )
{
    // if filename can't be decoded, then ignore it..
    continue;
}

// any error in the following will trigger a rename failure.
try
{
    // re-encode using the new IV..
    localIV = toIV;
    string newName = naming->encodePath( plainName.c_str(), &localIV );

    // store rename information..
    string oldFull = sourcePath + '/' + de->d_name;
    string newFull = sourcePath + '/' + newName;

    RenameEl ren;
    ren.oldCName = oldFull;
    ren.newCName = newFull;
    ren.oldPName = string(fromP) + '/' + plainName;
    ren.newPName = string(toP) + '/' + plainName;

    bool isDir;
#if defined(_DIRENT_HAVE_D_TYPE)
    if(de->d_type != DT_UNKNOWN)
    {
        isDir = (de->d_type == DT_DIR);
    } else
#endif
    {
        isDir = isDirectory( oldFull.c_str() );
    }

    ren.isDirectory = isDir;

    if(isDir)
    {
        // recurse.. We want to add subdirectory elements before the
        // parent, as that is the logical rename order..
        if(!genRenameList( renameList,
                           ren.oldPName.c_str(),
                           ren.newPName.c_str()))
        {
            return false;
        }
    }

    rDebug("adding file %s to rename list",
           oldFull.c_str());

    renameList.push_back( ren );
} catch( rlog::Error &err )
{
    // We can't convert this name, because we don't have a valid IV for
    // it (or perhaps a valid key).. It will be inaccessible..
    rWarning("Aborting rename: error on file: %s",
             fromCPart.append(1, '/').append(de->d_name).c_str());
    err.log( _RLDebugChannel );
}

```



```

        mode, strerror(eno));
    res = -eno;
} else
    res = 0;

return res;
}

int
DirNode::rename( const char *fromPlaintext, const char *toPlaintext )
{
    Lock _lock( mutex );

    string fromCName = rootDir + naming->encodePath( fromPlaintext );
    string toCName = rootDir + naming->encodePath( toPlaintext );
    rAssert( !fromCName.empty() );
    rAssert( !toCName.empty() );

    rLog( Info, "rename %s -> %s", fromCName.c_str(), toCName.c_str() );

    boost::shared_ptr<FileNode> toNode = findOrCreate( toPlaintext );

    boost::shared_ptr<RenameOp> renameOp;
    if( hasDirectoryNameDependency() && isDirectory( fromCName.c_str() ) )
    {
        rLog( Info, "recursive rename begin" );
        renameOp = newRenameOp( fromPlaintext, toPlaintext );

        if(!renameOp || !renameOp->apply())
        {
            if(renameOp)
                renameOp->undo();

            rWarning("rename aborted");
            return -EACCES;
        }
        rLog( Info, "recursive rename end" );
    }

    int res = 0;
    try
    {
        efs_stat st;
        bool preserve_mtime = fs_layer::stat(fromCName.c_str(), &st) == 0;

        renameNode( fromPlaintext, toPlaintext );
        toNode.reset();
        res = fs_layer::rename( fromCName.c_str(), toCName.c_str() );

        if(res == -1)
        {
            // undo
            res = -errno;
            renameNode( toPlaintext, fromPlaintext, false );

            if(renameOp)
                renameOp->undo();
        } else if(preserve_mtime)
        {
            struct utimbuf ut;
            ut.actime = st.st_atime;
            ut.modtime = st.st_mtime;
            fs_layer::utime(toCName.c_str(), &ut);
        }
    }
}

```

```

    } catch( rlog::Error &err )
    {
        // exception from renameNode, just show the error and continue..
        err.log( _RLWarningChannel );
        res = -EIO;
    }

    if(res != 0)
    {
        rLog( Info, "rename failed: %s", strerror( errno ));
        res = -errno;
    }

    return res;
}

int DirNode::link( const char *from, const char *to )
{
    Lock _lock( mutex );

    string fromCName = rootDir + naming->encodePath( from );
    string toCName = rootDir + naming->encodePath( to );

    rAssert( !fromCName.empty() );
    rAssert( !toCName.empty() );

    rLog(Info, "link %s -> %s", fromCName.c_str(), toCName.c_str());

    int res = -EPERM;
    if( fsConfig->config->externalIVChaining )
    {
        rLog(Info, "hard links not supported with external IV chaining!");
    } else
    {
        res = -ENOSYS;
    }
#if 0
    res = ::link( fromCName.c_str(), toCName.c_str() );
    if(res == -1)
        res = -errno;
    else
        res = 0;
#endif
    return res;
}

/*
The node is keyed by filename, so a rename means the internal node names
must be changed.
*/
boost::shared_ptr<FileNode> DirNode::renameNode( const char *from, const char *to )
{
    return renameNode( from, to, true );
}

boost::shared_ptr<FileNode> DirNode::renameNode( const char *from, const char *to,
    bool forwardMode )
{
    boost::shared_ptr<FileNode> node = findOrCreate( from );

    if(node)
    {
        uint64_t newIV = 0;

```

```

        string cname = rootDir + naming->encodePath( to, &newIV );

        rLog(Info, "renaming internal node %s -> %s",
            node->cipherName(), cname.c_str());

        if(node->setName( to, cname.c_str(), newIV, forwardMode ))
        {
            if(ctx)
                ctx->renameNode( from, to );
        } else
        {
            // rename error! - put it back
            rError("renameNode failed");
            throw RL_ERROR("Internal node name change failed!");
        }
    }

    return node;
}

boost::shared_ptr<FileNode> DirNode::findOrCreate( const char *plainName)
{
    boost::shared_ptr<FileNode> node;
    if(ctx)
        node = ctx->lookupNode( plainName );

    if(!node)
    {
        uint64_t iv = 0;
        string cipherName = naming->encodePath( plainName, &iv );
        node.reset( new FileNode( this, fsConfig,
            plainName,
            (rootDir + cipherName).c_str()));

        if(fsConfig->config->externalIVChaining)
            node->setName(0, 0, iv);

        rLog(Info, "created FileNode for %s", node->cipherName());
    }

    return node;
}

boost::shared_ptr<FileNode>
DirNode::lookupNode( const char *plainName, const char * requestor )
{
    (void)requestor;
    Lock _lock( mutex );

    boost::shared_ptr<FileNode> node = findOrCreate( plainName );

    return node;
}

/*
    Similar to lookupNode, except that we also call open() and only return a
    node on sucess.. This is done in one step to avoid any race conditions
    with the stored state of the file.
*/
boost::shared_ptr<FileNode>
DirNode::openNode( const char *plainName, const char * requestor, int flags,
    int *result )
{
    (void)requestor;

```

```

    rAssert( result != NULL );
    Lock _lock( mutex );

    boost::shared_ptr<FileNode> node = findOrCreate( plainName );

    if( node && (*result = node->open( flags )) >= 0 )
        return node;
    else
        return boost::shared_ptr<FileNode>();
}

int DirNode::unlink( const char *plaintextName )
{
    string cyName = naming->encodePath( plaintextName );
    rLog( Info, "unlink %s", cyName.c_str() );

    Lock _lock( mutex );

    int res = 0;
#ifdef 0 // TODO
    if( ctx && ctx->lookupNode( plaintextName ) )
    {
        // If FUSE is running with "hard_remove" option where it doesn't
        // hide open files for us, then we can't allow an unlink of an open
        // file..
        rWarning( "Refusing to unlink open file: %s, hard_remove option "
            "is probably in effect", cyName.c_str() );
        res = -EBUSY;
    } else
#endif
    {
        string fullName = rootDir + cyName;
        res = fs_layer::unlink( fullName.c_str() );
        if( res == -1 )
        {
            res = -errno;
            rDebug( "unlink error: %s", strerror(errno) );
        }
    }

    return res;
}

```

Файл « DirNode.h »

```

#ifndef _DirNode_incl_
#define _DirNode_incl_

#ifdef HAVE_INTTYPES_H
#include <inttypes.h>
#endif

#ifdef HAVE_STDINT_H
#include <stdint.h>
#endif

#ifdef !defined(_MSC_VER)
#    include <dirent.h>
#endif
#include <sys/types.h>

#include <map>

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		83

```

#include <list>
#include <vector>
#include <string>
#include <boost/thread.hpp>

#include "FileNode.h"
#include "NameIO.h"
#include "CipherKey.h"
#include "FSConfig.h"

using boost::shared_ptr;

class Cipher;
class RenameOp;
struct RenameEl;
class EncFS_Context;

class DirTraverse
{
public:
    DirTraverse(const boost::shared_ptr<fs_layer::DIR> &dirPtr, uint64_t iv,
                const boost::shared_ptr<NameIO> &naming);
    DirTraverse(const DirTraverse &src);
    ~DirTraverse();

    DirTraverse &operator = (const DirTraverse &src);

    // returns FALSE to indicate an invalid DirTraverse (such as when
    // an invalid directory is requested for traversal)
    bool valid() const;

    // return next plaintext filename
    // If fileType is not 0, then it is used to return the filetype (or 0 if
    // unknown)
    std::string nextPlaintextName(int *fileType=0, ino_t *inode=0);

    /* Return cipher name of next undecodable filename..
       The opposite of nextPlaintextName(), as that skips undecodable names..
    */
    std::string nextInvalid();
private:

    boost::shared_ptr<fs_layer::DIR> dir; // struct DIR
    // initialization vector to use. Not very general purpose, but makes it
    // more efficient to support filename IV chaining..
    uint64_t iv;
    boost::shared_ptr<NameIO> naming;
};

inline bool DirTraverse::valid() const { return dir != 0; }

#ifdef USE_HASHMAP
namespace __gnu_cxx
{
    template<> struct hash<std::string>
    {
        size_t operator() (const std::string &__s) const
        {
            return __stl_hash_string( __s.c_str() );
        }
    };
}
#endif

class DirNode

```

```

{
public:
    // sourceDir points to where raw files are stored
    DirNode(EncFS_Context *ctx,
            const std::string &sourceDir,
            const FSConfigPtr &config );
    ~DirNode();

    // return the path to the root directory
    std::string rootDirectory();

    // find files
    boost::shared_ptr<FileNode> lookupNode( const char *plaintextName,
            const char *requestor );

    /*
        Combined lookupNode + node->open() call. If the open fails, then the
        node is not retained. If the open succeeds, then the node is returned.
    */
    boost::shared_ptr<FileNode> openNode( const char *plaintextName,
            const char *requestor, int flags, int *openResult );

    std::string cipherPath( const char *plaintextPath );
    std::string cipherPathWithoutRoot( const char *plaintextPath );
    std::string plainPath( const char *cipherPath );

    // relative cipherPath is the same as cipherPath except that it doesn't
    // prepent the mount point. That it, it doesn't return a fully qualified
    // name, just a relative path within the encrypted filesystem.
    std::string relativeCipherPath( const char *plaintextPath );

    /*
        Returns true if file names are dependent on the parent directory name.
        If a directory name is changed, then all the filenames must also be
        changed.
    */
    bool hasDirectoryNameDependency() const;

    // unlink the specified file
    int unlink( const char *plaintextName );

    // traverse directory
    DirTraverse openDir( const char *plainDirName );

    // uid and gid are used as the directory owner, only if not zero
    int mkdir( const char *plaintextPath, mode_t mode,
            uid_t uid = 0, gid_t gid = 0);

    int rename( const char *fromPlaintext, const char *toPlaintext );

    int link( const char *from, const char *to );

    // returns idle time of filesystem in seconds
    int idleSeconds();

protected:

    /*
        notify that a file is being renamed.
        This renames the internal node, if any. If the file is not open, then
        this call has no effect.
        Returns the FileNode if it was found.
    */
    boost::shared_ptr<FileNode> renameNode( const char *from, const char *to );

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		85

```

boost::shared_ptr<FileNode> renameNode( const char *from, const char *to,
                                         bool forwardMode );

/*
    when directory IV chaining is enabled, a directory can't be renamed
    without renaming all its contents as well. recursiveRename should be
    called after renaming the directory, passing in the plaintext from and
    to paths.
*/
boost::shared_ptr<RenameOp> newRenameOp( const char *from, const char *to );

private:

    friend class RenameOp;

    bool genRenameList( std::list<RenameEl> &list, const char *fromP,
                       const char *toP );

    boost::shared_ptr<FileNode> findOrCreate( const char *plainName);

    boost::mutex mutex;

    EncFS_Context *ctx;

    // passed in as configuration
    std::string rootDir;
    FSConfigPtr fsConfig;

    boost::shared_ptr<NameIO> naming;
};

#endif

```

Файл « FileNode.cpp»

```

#include "encfs.h"

#include <errno.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
// #include <unistd.h>
#ifdef linux
#include <sys/fsuid.h>
#endif

#include <cstring>

#include "config.h"

#include "FileNode.h"
#include "FileUtils.h"
#include "Cipher.h"
#include "CipherFileIO.h"
#include "RawFileIO.h"
#include "MACFileIO.h"
#include "DirNode.h"
#include "fs_layer.h"

#include "FileIO.h"
#include "MemoryPool.h"
#include "Mutex.h"

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		86

```

#include <rlog/rlog.h>
#include <rlog/Error.h>

using namespace std;
using namespace rel;
using namespace rlog;
using boost::dynamic_pointer_cast;

/*
  TODO: locking at the FileNode level is inefficient, since this precludes
  multiple IO operations from going on concurrently within the same file.

  There is no reason why simultaneous reads cannot be satisfied, or why one
  read has to wait for the decoding of the previous read before it can be
  sent to the IO subsystem!
*/

static RLogChannel *Info = DEF_CHANNEL("info/FileNode", Log_Info);

FileNode::FileNode(DirNode *parent_, const FSConfigPtr &cfg,
  const char *plaintextName_, const char *cipherName_)
{
  Lock _lock( mutex );

  this->_pname = plaintextName_;
  this->_cname = cipherName_;
  this->parent = parent_;

  this->fsConfig = cfg;

  // chain RawFileIO & CipherFileIO
  boost::shared_ptr<FileIO> rawIO( new RawFileIO( _cname ) );
  io = boost::shared_ptr<FileIO>( new CipherFileIO( rawIO, fsConfig ) );

  if(cfg->config->blockMACBytes || cfg->config->blockMACRandBytes)
    io = boost::shared_ptr<FileIO>(new MACFileIO(io, fsConfig));
}

FileNode::~FileNode()
{
  // FileNode mutex should be locked before the destructor is called
  //pthread_mutex_lock( &mutex );

  _pname.assign( _pname.length(), '\0' );
  _cname.assign( _cname.length(), '\0' );
  io.reset();
}

const char *FileNode::cipherName() const
{
  return _cname.c_str();
}

const char *FileNode::plaintextName() const
{
  return _pname.c_str();
}

string FileNode::plaintextParent() const
{
  return parentDirectory( _pname );
}

static bool setIV(const boost::shared_ptr<FileIO> &io, uint64_t iv)

```

										Лист
										87
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 2.090301.116.001 ПЗ					

```

{
    efs_stat stbuf;
    if((io->getattr(&stbuf, 0) < 0) || S_ISREG(stbuf.st_mode))
        return io->setIV( iv );
    else
        return true;
}

bool FileNode::setName( const char *plaintextName_, const char *cipherName_,
                        uint64_t iv, bool setIVFirst )
{
    //Lock _lock( mutex );
    rDebug("calling setIV on %s", cipherName_);
    if(setIVFirst)
    {
        if(fsConfig->config->externalIVChaining && !setIV(io, iv))
            return false;

        // now change the name..
        if(plaintextName_)
            this->_pname = plaintextName_;
        if(cipherName_)
        {
            this->_cname = cipherName_;
            io->setFileName( cipherName_ );
        }
    } else
    {
        std::string oldPName = _pname;
        std::string oldCName = _cname;

        if(plaintextName_)
            this->_pname = plaintextName_;
        if(cipherName_)
        {
            this->_cname = cipherName_;
            io->setFileName( cipherName_ );
        }

        if(fsConfig->config->externalIVChaining && !setIV(io, iv))
        {
            _pname = oldPName;
            _cname = oldCName;
            return false;
        }
    }

    return true;
}

int FileNode::mknod(mode_t mode, dev_t rdev, uid_t uid, gid_t gid)
{
    Lock _lock( mutex );

    int res = 0;
    int olduid = -1;
    int oldgid = -1; /* TODO
    if(uid != 0)
    {
        olduid = setfsuid( uid );
        if(olduid == -1)
        {
            rInfo("setfsuid error: %s", strerror(errno));
            return -EPERM;
        }
    }

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		88

```

    }
}
if(gid != 0)
{
    oldgid = setfsgid( gid );
    if(oldgid == -1)
    {
        rInfo("setfsgid error: %s", strerror(errno));
        return -EPERM;
    }
}
}*/

/*
 * cf. xmp_mknod() in fusexmp.c
 * The regular file stuff could be stripped off if there
 * were a create method (advised to have)
 */
if (S_ISREG( mode )) {
    res = fs_layer::open( _cname.c_str(), O_CREAT | O_EXCL | O_WRONLY, mode );
    if (res >= 0)
        res = fs_layer::close( res );
}/* else if (S_ISFIFO( mode ))
    res = fs_layer::mkfifo( _cname.c_str(), mode );
else
    res = fs_layer::mknod( _cname.c_str(), mode, rdev );*/           // TODO
/* TODO
    if(oldduid >= 0)
        setfsuid( oldduid );
    if(oldgid >= 0)
        setfsgid( oldgid );
*/
if(res == -1)
{
    int eno = errno;
    rDebug("mknod error: %s", strerror(eno));
    res = -eno;
}

return res;
}

int FileNode::open(int flags) const
{
    Lock _lock( mutex );

    int res = io->open( flags );
    return res;
}

int FileNode::getAttr(efs_stat *stbuf, void *statCache) const
{
    Lock _lock( mutex );

    int res = io->getAttr( stbuf, statCache );
    return res;
}

efs_off_t FileNode::getSize() const
{
    Lock _lock( mutex );

    int res = io->getSize();
    return res;
}

```

```

ssize_t FileNode::read( efs_off_t offset, unsigned char *data, ssize_t size ) const
{
    IORequest req;
    req.offset = offset;
    req.dataLen = size;
    req.data = data;

    Lock _lock( mutex );

    return io->read( req );
}

bool FileNode::write(efs_off_t offset, unsigned char *data, ssize_t size)
{
    rLog(Info, "FileNode::write offset %" PRIi64 ", data size %i",
        offset, (int)size);

    IORequest req;
    req.offset = offset;
    req.dataLen = size;
    req.data = data;

    Lock _lock( mutex );

    return io->write( req );
}

int FileNode::truncate( efs_off_t size )
{
    Lock _lock( mutex );

    return io->truncate( size );
}

int FileNode::sync(bool datasync)
{
    Lock _lock( mutex );

    int fh = io->open( O_RDONLY );
    if(fh >= 0)
    {
        int res = -EIO;
#ifdef linux
        if(datasync)
            res = fs_layer::fdatasync( fh );
        else
            res = fs_layer::fsync( fh );
#else
        // no fdatasync support
        // TODO: use autoconfig to check for it..
        res = fs_layer::fsync(fh);
#endif

        if(res == -1)
            res = -errno;

        return res;
    } else
        return fh;
}

```

Файл « FileNode.h»

```

#ifndef _FileNode_incl_
#define _FileNode_incl_

#include "config.h"

#ifdef HAVE_STDINT_H
#include <stdint.h>
#endif

#include <boost/thread.hpp>

#include "encfs.h"
#include "CipherKey.h"
#include "FileUtils.h"
#include "fs_layer.h"

#ifdef HAVE_INTTYPES_H
#include <inttypes.h>
#endif

#ifdef HAVE_STDINT_H
#include <stdint.h>
#endif

#ifdef HAVE_SYS_TYPES_H
#include <sys/types.h>
#endif

#include <string>

class Cipher;
class FileIO;
class DirNode;
using boost::shared_ptr;

class FileNode
{
public:
    FileNode(DirNode *parent,
             const FSConfigPtr &cfg,
             const char *plaintextName,
             const char *cipherName);
    ~FileNode();

    const char *plaintextName() const;
    const char *cipherName() const;

    // directory portion of plaintextName
    std::string plaintextParent() const;

    // if setIVFirst is true, then the IV is changed before the name is changed
    // (default). The reverse is also supported for special cases..
    bool setName( const char *plaintextName, const char *cipherName,
                  uint64_t iv, bool setIVFirst = true);

    // create node
    // If uid/gid are not 0, then chown is used change ownership as specified
    int mknod(mode_t mode, dev_t rdev, uid_t uid = 0, gid_t gid = 0);

    // Returns < 0 on error (-errno), file descriptor on success.
    int open(int flags) const;

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		91

```

// getAttr returns 0 on success, -errno on failure
int getAttr(efs_stat *stbuf, void *statCache) const;
efs_off_t getSize() const;

ssize_t read(efs_off_t offset, unsigned char *data, ssize_t size) const;
bool write(efs_off_t offset, unsigned char *data, ssize_t size);

// truncate the file to a particular size
int truncate( efs_off_t size );

// datasync or full sync
int sync(bool dataSync);
private:

// doing locking at the FileNode level isn't as efficient as at the
// lowest level of RawFileIO, since that means locks are held longer
// (held during CPU intensive crypto operations!). However it makes it
// easier to avoid any race conditions with operations such as
// truncate() which may result in multiple calls down to the FileIO
// level.
mutable boost::mutex mutex;

FSConfigPtr fsConfig;

shared_ptr<FileIO> io;
std::string _pname; // plaintext name
std::string _cname; // encrypted name
DirNode *parent;

private:
FileNode(const FileNode &src);
FileNode &operator = (const FileNode &src);

};

#endif

```

Файл « Interface.cpp»

```

#include "Interface.h"

#include "ConfigVar.h"

#include <rlog/rlog.h>
#include <rlog/RLogChannel.h>

using namespace rel;
using namespace rlog;

static RLogChannel * Info = DEF_CHANNEL( "info/iface", Log_Info );

Interface::Interface(const char *name_, int Current, int Revision, int Age)
: _name( name_ )
, _current( Current )
, _revision( Revision )
, _age( Age )
{
}

Interface::Interface(const std::string &name_, int Current,

```

									Лист
									92
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 2.090301.116.001 ПЗ				

```

        int Revision, int Age)
    : _name( name_ )
    , _current( Current )
    , _revision( Revision )
    , _age( Age )
{
}

Interface::Interface(const Interface &src)
    : _name( src._name )
    , _current( src._current )
    , _revision( src._revision )
    , _age( src._age )
{
}

Interface::Interface()
    : _current( 0 )
    , _revision( 0 )
    , _age( 0 )
{
}

Interface &Interface::operator = (const Interface &src)
{
    _name = src._name;
    _current = src._current;
    _revision = src._revision;
    _age = src._age;
    return *this;
}

const std::string & Interface::name() const
{
    return _name;
}

std::string & Interface::name()
{
    return _name;
}

int Interface::current() const
{
    return _current;
}

int &Interface::current()
{
    return _current;
}

int Interface::revision() const
{
    return _revision;
}

int &Interface::revision()
{
    return _revision;
}

int Interface::age() const
{

```

```

        return _age;
    }

    int &Interface::age()
    {
        return _age;
    }

    bool operator == (const Interface &A, const Interface &B)
    {
        return ( A.name() == B.name()
                && A.current() == B.current()
                && A.revision() == B.revision()
                && A.age() == B.age() );
    }

    bool operator != (const Interface &A, const Interface &B)
    {
        return ( A.name() != B.name()
                || A.current() != B.current()
                || A.revision() != B.revision()
                || A.age() != B.age() );
    }

    // zero branch method of getting comparison sign..
    // tricky.. makes assumptions
    #if 0
    static int sign( int a, int b )
    {
        unsigned int ab = ((unsigned int)(a - b)) >> 31;
        unsigned int ba = ((unsigned int)(b - a)) >> 31;

        return 1 + ba - ab;
    }
    #else
    // simple, easy to check, unlikely to break due to unforeseen events..
    static int sign( int a, int b )
    {
        if(a < b)
            return 0;
        else if(a == b)
            return 1;
        else
            return 2;
    }
    #endif

    static int diffSum( const Interface &A, const Interface &B )
    {
        int cS = sign( A.current() , B.current() );
        int aS = sign( A.age(), B.age() );
        int rS = sign( A.revision(), B.revision() );

        return (cS * 3 + aS) * 3 + rS;
    }

    const int EqualVersion = (1 * 3 + 1) * 3 + 1;

    bool Interface::implements(const Interface &B) const
    {
        rLog(Info, "checking if %s(%i:%i:%i) implements %s(%i:%i:%i)",
            name().c_str(), current(), revision(), age(),
            B.name().c_str(), B.current(), B.revision(), B.age());
    }

```

```

    if( name() != B.name() )
        return false;

    int currentDiff = current() - B.current();
    return ( currentDiff >= 0 && currentDiff <= age() );
}

bool operator < (const Interface &A, const Interface &B)
{
    if( A.name() == B.name() )
    {
        return ( diffSum(A,B) < EqualVersion );
    } else
        return A.name() < B.name();
}

bool operator > (const Interface &A, const Interface &B)
{
    if( A.name() == B.name() )
    {
        return ( diffSum(A,B) > EqualVersion );
    } else
        return A.name() < B.name();
}

bool operator <= (const Interface &A, const Interface &B)
{
    if( A.name() == B.name() )
    {
        return ( diffSum(A,B) <= EqualVersion );
    } else
        return A.name() < B.name();
}

bool operator >= (const Interface &A, const Interface &B)
{
    if( A.name() == B.name() )
    {
        return ( diffSum(A,B) >= EqualVersion );
    } else
        return A.name() < B.name();
}

ConfigVar & operator << (ConfigVar &dst, const rel::Interface &iface)
{
    dst << iface.name() << iface.current() << iface.revision() << iface.age();
    return dst;
}

const ConfigVar & operator >> (const ConfigVar &src, Interface &iface)
{
    src >> iface.name();
    src >> iface.current();
    src >> iface.revision();
    src >> iface.age();
    return src;
}

```

Файл « CreateNewEncFSDialog.cpp»

```
#include "CommonIncludes.h"

// Unfortunately, my version of libencfs is a tiny bit newer than 1.7.4 on Linux.
// This define restores full compatibility
#define EFS_COMPATIBILITY_WORKAROUND 1

#include "CreateNewEncFSDialog.h"
#include "MountList.h"

#if wxCHECK_VERSION(2, 9, 0) && defined(_WIN32)
#define HAVE_MODE_T // Workaround for double defined mode_t on Windows
#endif

// libencfs
#include "config.h"
#include "FileUtils.h"
#include "ConfigReader.h"
#include "FSConfig.h"

#include "DirNode.h"
#include "Cipher.h"
#include "StreamNameIO.h"
#include "BlockNameIO.h"
#include "NullNameIO.h"
#include "Context.h"

CreateNewEncFSDialog::CreateNewEncFSDialog(wxWindow* parent)
: CreateNewEncFSDialogBase(parent, pMountList_(NULL),
driveLetter_(L'?'), useExternalConfigFile_(false), isLocalDrive_(true),
cachingEnabled_(false), cipherKeySize_(256),
cipherBlockSize_(4096), keyDerivationDuration_(500),
autoChoice_(0), noneChoice_(0)
{
    pUseExternalConfigFileCheckBox_>SetValue(useExternalConfigFile_);
    pExternalConfigFileNamePickerCtrl_>Enable(false);
    worldWritable_ = false;
    storePassword_ = false;
    pWorldWritableCheckBox_>SetValue(false);
    pStorePasswordCheckBox_>SetValue(false);
    pLocalDriveCheckBox_>SetValue(true);
    pEnableCachingCheckBox_>SetValue(false);

    #if defined(EFS_WIN32)
        pDriveLetterChoice_>Clear();
        autoChoice_ = pDriveLetterChoice_>Append(wxT("Auto"));
        noneChoice_ = pDriveLetterChoice_>Append(wxT("None"));
        pDriveLetterChoice_>Select(autoChoice_);

        // Find out unused drive letters
        DWORD usedLogicalDrives = GetLogicalDrives();
        wxCSConv conv(wxT("C")); // Use "C" locale
        for(char i = 'A'; i <= 'Z'; i++)
        {
            if((usedLogicalDrives & 0x01) == 0)
            {
                // current drive letter is unused -> add it to combo box
                size_t outLen;
                wxString driveLetter(conv.CMB2WC(&i, 1, &outLen));
                driveLetter.Append(wxT(":"));
                pDriveLetterChoice_>Append(driveLetter);
            }
        }
    #endif
}
```

										Лист
										96
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 2.090301.116.001 ПЗ					

```

        }
        usedLogicalDrives >>= 1;
    }

    pWorldWritableStaticText_>Hide();
    pWorldWritableCheckBox_>Hide();
    pFlexGridSizer_>AddGrowableRow(6);
#else
    pDriveLetterStaticText_>Hide();
    pDriveLetterChoice_>Hide();
    pFlexGridSizer_>AddGrowableRow(5);
    pLocalDriveStaticText_>Hide();
    pLocalDriveCheckBox_>Hide();
    pFlexGridSizer_>AddGrowableRow(7);
#endif

    pCipherAlgorithmChoice_>Clear();
    Cipher::AlgorithmList algorithms = Cipher::GetAlgorithmList();
    Cipher::AlgorithmList::const_iterator it;
    for(it = algorithms.begin(); it != algorithms.end(); ++it)
    {
        pCipherAlgorithmChoice_>Append(wxString(it->name.c_str(),
*wxConvCurrent));
    }
    pCipherAlgorithmChoice_>Select(0);

    pNameEncodingChoice_>Clear();
    wxString toolTipStr;
    NameIO::AlgorithmList nmalgorithms = NameIO::GetAlgorithmList();
    NameIO::AlgorithmList::const_iterator nmit;
    for(nmit = nmalgorithms.begin(); nmit != nmalgorithms.end(); ++nmit)
    {
        #if defined(EFS_COMPATIBILITY_WORKAROUND)
            if(nmit->name == std::string("Block32"))
                continue;
        #endif

        if(toolTipStr.Length() > 0)
            toolTipStr.Append(wxT("\n"));
        pNameEncodingChoice_>Append(wxString(nmit->name.c_str(), *wxConvCurrent));
        toolTipStr.Append(wxString(nmit->name.c_str(), *wxConvCurrent)
            + wxT(": ") + wxString(nmit->description.c_str(), *wxConvCurrent));
    }
    pNameEncodingChoice_>Select(0);
    pNameEncodingChoice_>SetToolTip(toolTipStr);

    pKeyDerivationDurationChoice_>Clear();
    wxArrayString timeStrings;
    for(int i = 500; i <= 3000; i+= 500)
    {
        wxString timeStr = wxString::Format(wxT("%d"), i);
        timeStrings.Add(timeStr);
    }
    pKeyDerivationDurationChoice_>Append(timeStrings);
    pKeyDerivationDurationChoice_>Select(0);

    pEncFSConfigurationRadioButton_>Select(0);
    wxCommandEvent dummyEvent;
    OnEncFSConfigurationRadioBox(dummyEvent);

    Layout();
    Fit();
}

CreateNewEncFSDialog::~CreateNewEncFSDialog()

```

```

{
}

void CreateNewEncFSDialog::setMountList(MountList *pMountList)
{
    pMountList_ = pMountList;
}

void CreateNewEncFSDialog::OnUseExternalConfigFileCheckBox( wxCommandEvent& event )
{
    bool isChecked = event.IsChecked();
    if(isChecked)
    {
        pExternalConfigFileNamePickerCtrl_>Enable(true);
    }
    else
    {
        pExternalConfigFileNamePickerCtrl_>Enable(false);
    }
}

void CreateNewEncFSDialog::OnEncFSConfigurationRadioBox( wxCommandEvent& event )
{
    bool enableDetailWidgets = false;
    int selection = pEncFSConfigurationRadioButton_>GetSelection();
    if(selection == 0)
    {
        // Standard configuration: AES, 192, 1024, Block, false, true, true, false,
500
        pCipherAlgorithmChoice_>SetStringSelection(wxT("AES"));

        wxCommandEvent dummyEvent;
        OnCipherAlgorithmChoice(dummyEvent);

        pCipherKeysizeChoice_>SetStringSelection(wxT("192"));
        pCipherBlocksizeChoice_>SetStringSelection(wxT("1024"));
        pNameEncodingChoice_>SetStringSelection(wxT("Block"));
        pPerBlockHMACCheckBox_>SetValue(false);
        pUniqueIVCheckBox_>SetValue(true);
        pChainedIVCheckBox_>SetValue(true);
        pExternalIVCheckBox_>SetValue(false);
        pKeyDerivationDurationChoice_>SetStringSelection(wxT("500"));
    }
    else if(selection == 1)
    {
        // Paranoia configuration: AES, 256, 1024, Block, true, true, true, true,
3000
        pCipherAlgorithmChoice_>SetStringSelection(wxT("AES"));

        wxCommandEvent dummyEvent;
        OnCipherAlgorithmChoice(dummyEvent);

        pCipherKeysizeChoice_>SetStringSelection(wxT("256"));
        pCipherBlocksizeChoice_>SetStringSelection(wxT("1024"));
        pNameEncodingChoice_>SetStringSelection(wxT("Block"));
        pPerBlockHMACCheckBox_>SetValue(true);
        pUniqueIVCheckBox_>SetValue(true);
        pChainedIVCheckBox_>SetValue(true);
        pExternalIVCheckBox_>SetValue(true);
        pKeyDerivationDurationChoice_>SetStringSelection(wxT("3000"));
    }
    else
    {
        // Expert configuration

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		98

```

        enableDetailWidgets = true;
    }

    pCipherAlgorithmChoice_->Enable(enableDetailWidgets);
    pCipherKeysizeChoice_->Enable(enableDetailWidgets);
    pCipherBlocksizeChoice_->Enable(enableDetailWidgets);
    pNameEncodingChoice_->Enable(enableDetailWidgets);
    pPerBlockHMACCheckBox_->Enable(enableDetailWidgets);
    pUniqueIVCheckBox_->Enable(enableDetailWidgets);
    pChainedIVCheckBox_->Enable(enableDetailWidgets);
    pExternalIVCheckBox_->Enable(enableDetailWidgets);
    pKeyDerivationDurationChoice_->Enable(enableDetailWidgets);
}

void CreateNewEncFSDialog::OnCipherAlgorithmChoice( wxCommandEvent& event )
{
    wxString selectedCipherAlgorithm = pCipherAlgorithmChoice_->GetString(
pCipherAlgorithmChoice_->GetSelection() );

    pCipherKeysizeChoice_->Clear();
    pCipherBlocksizeChoice_->Clear();

    Cipher::AlgorithmList algorithms = Cipher::GetAlgorithmList();
    Cipher::AlgorithmList::const_iterator it;
    int optNum = 1;
    for(it = algorithms.begin(); it != algorithms.end(); ++it, ++optNum)
    {
        wxString algoName(it->name.c_str(), *wxConvCurrent);
        if(algoName == selectedCipherAlgorithm)
        {
            wxArrayString keySizes;
            int keySizeMin = it->keyLength.min();
            int keySizeMax = it->keyLength.max();
            int keySizeInc = it->keyLength.inc();
            for(int keySize = keySizeMin; keySize <= keySizeMax; keySize +=
keySizeInc)
            {
                wxString keySizeStr = wxString::Format(wxT("%d"), keySize);
                keySizes.Add(keySizeStr);
            }
            pCipherKeysizeChoice_->Append(keySizes);
            pCipherKeysizeChoice_->Select(keySizes.GetCount() - 1);

            wxArrayString blockSizes;
            int blockSizeMin = it->blockSize.min();
            int blockSizeMax = it->blockSize.max();
            int blockSizeInc = it->blockSize.inc();
            for(int blockSize = blockSizeMin; blockSize <= blockSizeMax;
blockSize += blockSizeInc)
            {
                wxString blockSizeStr = wxString::Format(wxT("%d"),
blockSize);
                blockSizes.Add(blockSizeStr);
            }
            pCipherBlocksizeChoice_->Append(blockSizes);
            pCipherBlocksizeChoice_->Select(blockSizes.GetCount() - 1);
        }
    }
}

void CreateNewEncFSDialog::OnCancel( wxCommandEvent& event )
{
    EndModal(wxID_CANCEL);
}

```

```

void CreateNewEncFSDialog::OnOK( wxCommandEvent& event )
{
    if (Validate() && TransferDataFromWindow())
    {
        // Make sure mountName_ is unique
        if(pMountList_>findEntryByName(mountName_) != NULL)
        {
            wxMessageBox(wxT("The mount name is not unique! Please choose another
name."),
                        wxT("Error"), wxOK | wxICON_ERROR);
            return;
        }

#ifdef EFS_WIN32
        if(pDriveLetterChoice_>GetSelection() == autoChoice_)
        {
            driveLetter_ = L'?' ; // Auto
        }
        else if(pDriveLetterChoice_>GetSelection() == noneChoice_)
        {
            driveLetter_ = L'-' ; // None (treated by EncFSMP, not PFM)
        }
        else
        {
            wxString selectedDriveLetter = pDriveLetterChoice_>GetStringSelection();
            driveLetter_ = selectedDriveLetter.c_str()[0];
        }
#else
        driveLetter_ = L'?' ;
#endif

        encFSPath_ = pEncFSPathDirPicker_>GetPath();
        if(!wxFileName::DirExists(encFSPath_))
        {
            wxMessageBox(wxT("EncFS path does not exist! Please enter a valid
path"),
                        wxT("Error"), wxOK | wxICON_ERROR);
            return;
        }
        wxDir encDir(encFSPath_);
        if(encDir.HasFiles()
           || encDir.HasSubDirs())
        {
            wxMessageBox(wxT("EncFS path is not empty! Please select an empty
path"),
                        wxT("Error"), wxOK | wxICON_ERROR);
            return;
        }

        useExternalConfigFile_ = pUseExternalConfigFileCheckBox_>GetValue();
        if(useExternalConfigFile_)
        {
            externalConfigFileName_ = pExternalConfigFileNamePickerCtrl_>GetPath();

            wxFileName externalConfigFN(externalConfigFileName_);
            if(externalConfigFN.FileExists())
            {
                wxMessageBox(wxT("External config file already exists!"),
                            wxT("Error"), wxOK | wxICON_ERROR);
                return;
            }
        }
    }
}

```

```

        wxFileName externalConfigPath(externalConfigFN);
        externalConfigPath.SetFullName(wxEmptyString);
        if(!externalConfigPath.IsDirWritable())
        {
            wxMessageBox(wxT("Path of external config file is not
writable!"),
                        wxT("Error"), wxOK | wxICON_ERROR);
            return;
        }

        isLocalDrive_ = pLocalDriveCheckBox->GetValue();
        cachingEnabled_ = pEnableCachingCheckBox->GetValue();

        if(password_.Length() == 0)
        {
            wxMessageBox(wxT("Empty password is not allowed!"),
                        wxT("Error"), wxOK | wxICON_ERROR);
            return;
        }
        if(password_ != passwordRetype_)
        {
            wxMessageBox(wxT("The retyped password is not equal to the first
one!"),
                        wxT("Error"), wxOK | wxICON_ERROR);
            return;
        }

        cipherAlgorithm_ = pCipherAlgorithmChoice->GetStringSelection();
        pCipherKeysizeChoice->GetStringSelection().ToLong(&cipherKeySize_);
        pCipherBlocksizeChoice->GetStringSelection().ToLong(&cipherBlockSize_);
        nameEncoding_ = pNameEncodingChoice->GetStringSelection();
        pKeyDerivationDurationChoice->GetStringSelection().ToLong(&keyDerivationDuration_);

        EndModal(wxID_OK);
    }
}

BEGIN_EVENT_TABLE( CreateNewEncFSDialog, CreateNewEncFSDialogBase )
END_EVENT_TABLE()

```

Файл « CreateNewEncFSDialog.h»

```

#ifndef CREATENEWENCFSIALOG_H
#define CREATENEWENCFSIALOG_H

class MountList;

#include "EncFSMPMainFrameBase.h"

class CreateNewEncFSDialog: public CreateNewEncFSDialogBase
{
public:
    CreateNewEncFSDialog(wxWindow* parent);
    virtual ~CreateNewEncFSDialog();

    void setMountList(MountList *pMountList);

    wchar_t getDriveLetter() const { return driveLetter_; }
    wxString getEncFSPath() const { return encFSPath_; }
    bool getUseExternalConfigFile() const { return useExternalConfigFile_; }
    wxString getExternalConfigFileName() const { return externalConfigFileName_; }

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		101

```

    bool getIsLocalDrive() const { return isLocalDrive_; }
    bool getCachingEnabled() const { return cachingEnabled_; }
    wxString getCipherAlgorithm() const { return cipherAlgorithm_; }
    long getCipherKeySize() const { return cipherKeySize_; }
    long getCipherBlockSize() const { return cipherBlockSize_; }
    wxString getNameEncoding() const { return nameEncoding_; }
    long getKeyDerivationDuration() const { return keyDerivationDuration_; }

protected:
    virtual void OnUseExternalConfigFileCheckBox( wxCommandEvent& event );
    virtual void OnEncFSConfigurationRadioBox( wxCommandEvent& event );
    virtual void OnCipherAlgorithmChoice( wxCommandEvent& event );
    virtual void OnCancel( wxCommandEvent& event );
    virtual void OnOK( wxCommandEvent& event );

private:
    MountList *pMountList_;

    wchar_t driveLetter_;
    wxString encFSPath_;
    bool useExternalConfigFile_;
    wxString externalConfigFileName_;
    bool isLocalDrive_;
    bool cachingEnabled_;
    wxString cipherAlgorithm_;
    long cipherKeySize_;
    long cipherBlockSize_;
    wxString nameEncoding_;
    long keyDerivationDuration_;

    int autoChoice_, noneChoice_;

    DECLARE_EVENT_TABLE()
};

#endif

```

Файл « EncFSMPLogger.cpp »

```

#include "CommonIncludes.h"
#include "EncFSMPLogger.h"
#include "EncFSMPErrorLog.h"

#if wxCHECK_VERSION(2, 9, 0) && defined(_WIN32)
#define HAVE_MODE_T // Workaround for double defined mode_t on Windows
#endif
#include "fs_layer.h"

// rlog
#include "rlog/Error.h"

// Static member
EncFSMPLogger EncFSMPLogger::instance_;

EncFSMPLogger::EncFSMPLogger()
    : pEncFSMPErrorLog_(NULL)
{
}

EncFSMPLogger::~EncFSMPLogger()
{
}

```

									Лист
									102
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 2.090301.116.001 ПЗ				

```

void EncFSMPLogger::setErrorLog(EncFSMPErrorLog *pEncFSMPErrorLog)
{
    instance_.pEncFSMPErrorLog_ = pEncFSMPErrorLog;
}

void EncFSMPLogger::log(const std::wstring &errStr, const std::string &fn, rlog::Error
*pErr)
{
    if(instance_.pEncFSMPErrorLog_ != NULL)
    {
        wxString text(errStr);
        if(!fn.empty())
        {
            boost::filesystem::path path(fs_layer::stringToFSPath(fn));
            wxString pathWx(path.wstring());
            if(!text.empty())
                text.Append(wxT(", "));
            text.Append(wxT("file: "));
            text.Append(pathWx);
        }
        if(pErr != NULL)
        {
            text.Append(wxT(", details: "));
            text.Append(wxString(pErr->what(), *wxConvCurrent));
        }
        text.Append(wxT("\n"));
        instance_.pEncFSMPErrorLog_->addText(text);
    }
}

```

Файл « EncFSMPLogger.h»

```

#ifndef ENCFSMPLLOGGER_H
#define ENCFSMPLLOGGER_H

// Forward declarations
class EncFSMPErrorLog;
namespace rlog
{
    class Error;
};

class EncFSMPLogger
{
public:

    static void setErrorLog(EncFSMPErrorLog *pEncFSMPErrorLog);

    static void log(const std::wstring &errStr, const std::string &fn, rlog::Error
*pErr);

private:
    EncFSMPLogger();
    virtual ~EncFSMPLogger();

    static EncFSMPLogger instance_;

    EncFSMPErrorLog *pEncFSMPErrorLog_;
};

#endif

```

					ПГУ 2.090301.116.001 ПЗ	Лист
						103
Изм.	Лист	№ докум.	Подпись	Дата		

Файл « EncFSMPMainFrame.cpp»

```
#include "CommonIncludes.h"
#include "version.h"

#include "EncFSMPMainFrame.h"
#include "PFMPProxy.h"
#include "PFMHandlerThread.h"
#include "OpenExistingFSDialog.h"
#include "CreateNewEncFSDialog.h"
#include "ChangePasswordDialog.h"
#include "EncFSUtilities.h"
#include "EncFSMPStrings.h"
#include "EncFSMPTaskBarIcon.h"
#include "EncFSMPErrorLog.h"
#include "EncFSMPLogger.h"
#if defined(EFS_WIN32)
#    include "EncFSMPIPCWin.h"
#else
#    include "EncFSMPIPCPosix.h"
#endif

#include <wx/dirdlg.h>
#include <wx/busyinfo.h>
#include <wx/config.h>

#if defined(__WXMAC__) || defined(__WXOSX__) || defined(__WXOSX_COCOA__)
#    include "osx/MacMenuWorkaroundBridge.h"
#endif

#if defined(EFS_WIN32)
#    include "Win32Utils.h"
#    include <Shellapi.h>
#endif

// EncFS
#if wxCHECK_VERSION(2, 9, 0) && defined(_WIN32)
#define HAVE_MODE_T // Workaround for double defined mode_t on Windows
#endif
#include "config.h"

// OpenSSL
#include "openssl/ssl.h"

// boost
#include <boost/version.hpp>

#include <sstream>
#include <memory>

enum
{
    ID_TIMER = 2000,
    ID_NEW_MOUNT_EVENT,
    ID_ENCFD_MOUNT_ERROR_EVENT,
    ID_UAC,
    ID_MINIMIZE_TOTRAYMENUITEM,
    ID_CTXMOUNT,
    ID_CTXREMOVE,
    ID_CTXEDIT,
    ID_CTXBROWSE,
    ID_CTXSHOWINFO,
    ID_CTXCHANGEPSWD,

```

									Лист
									104
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 2.090301.116.001 ПЗ				

```

        ID_CTXEXPORT,
        ID_ENCF5_COMMAND
};

const wxEventType myCustomEventType = wxNewEventType();

EncFSMPMainFrame::EncFSMPMainFrame(wxWindow* parent)
    : EncFSMPMainFrameBase(parent),
      aTimer_(this, ID_TIMER), minimizeToTray_(false),
      disableUnmountDialogOnExit_(false), savePasswordsInRAM_(false),
      pTaskBarIcon_(NULL), pMountsListPopupMenu_(NULL),
      firstTimeOnTimer_(false), pEncFSMPErrorLog_(NULL)
{
    pMountsListCtrl_>ClearAll();
    pMountsListCtrl_>InsertColumn(0, wxT("Mounted"));
    pMountsListCtrl_>InsertColumn(1, wxT("Name"));
    pMountsListCtrl_>InsertColumn(2, wxT("EncFS path"));
#ifdef EFS_WIN32
    pMountsListCtrl_>InsertColumn(3, wxT("Drive"));
#endif

    SetIcon(getIcon());

#ifdef wxCHECK_VERSION(2, 9, 0)
    if(wxTaskBarIcon::IsAvailable())
#endif
    {
#ifdef EFS_MACOSX
        pMinimizeToTrayMenuItem_ = pToolsMenu_>Append(ID_MINIMIZE_TOTRAYMENUITEM,
            wxT("Minimize to menu icon"), wxEmptyString, wxITEM_NORMAL);
#else
        pMinimizeToTrayMenuItem_ = pOptionsMenu_>Append(ID_MINIMIZE_TOTRAYMENUITEM,
            wxT("Minimize to tray"), wxEmptyString, wxITEM_CHECK);
#endif
    }

    // Workaround for OS X: Hide empty menus
#ifdef __WXMAC__ || defined(__WXOSX__) || defined(__WXOSX_COCOA__)
    wxMenuItem *tempItem = pToolsMenu_>Append(ID_INFOBUTTON, wxT("Temp item"));
    CocoaBridge::hideEmptyMenus();
    pToolsMenu_>Remove(tempItem);
#endif

    pEncFSMPErrorLog_ = new EncFSMPErrorLog(this);
    pEncFSMPErrorLog_>Hide();
    EncFSMPLogger::setErrorLog(pEncFSMPErrorLog_);

    loadWindowLayoutFromConfig();
    mountList_.loadFromConfig();
    updateMountListCtrl();
    updateButtonStates();

    aTimer_.Start(50);
}

EncFSMPMainFrame::~EncFSMPMainFrame()
{
    aTimer_.Stop();
    if(pTaskBarIcon_ != NULL)
    {
        delete pTaskBarIcon_;
        pTaskBarIcon_ = NULL;
    }
    if(pMountsListPopupMenu_ != NULL)

```

					ПГУ 2.090301.116.001 ПЗ	Лист
						105
Изм.	Лист	№ докум.	Подпись	Дата		

```

        {
            delete pMountsListPopupMenu_;
            pMountsListPopupMenu_ = NULL;
        }
        EncFSMPLogger::setErrorLog(NULL);
        if(pEncFSMPErrorLog_ != NULL)
        {
            delete pEncFSMPErrorLog_;
            pEncFSMPErrorLog_ = NULL;
        }
    }

/**
 * This method gets called by the PFMMonitorThread in case of an mount/unmount event.
 */
void EncFSMPMainFrame::addNewMountEvent(bool isMountEvent, bool isError,
    const std::wstring &mountName, wchar_t driveLetter, const std::wstring
&mountPoint)
{
    {
        wxMutexLocker lock(mountEventsMutex_);
        MountEvent evt;
        evt.isMountEvent_ = isMountEvent;
        evt.isError_ = isError;
        evt.mountName_ = mountName;
        evt.driveLetter_ = driveLetter;
        evt.mountPoint_ = mountPoint;
        mountEvents_.push_back(evt);
    }

    #if wxCHECK_VERSION(2, 9, 0)
        wxCommandEvent* newEvent = new wxCommandEvent(myCustomEventType,
ID_NEW_MOUNT_EVENT);
        wxQueueEvent(this, newEvent);
    #else
        wxCommandEvent newEvent(myCustomEventType, ID_NEW_MOUNT_EVENT);
        AddPendingEvent(newEvent);
    #endif
}

/**
 * This method gets called by the PFMHandlerThread in case of an error.
 */
void EncFSMPMainFrame::reportEncFSError(const wxString &error, const wxString &mountName)
{
    {
        wxMutexLocker lock(encFSErrorMutex_);
        encFSErrorList_.push_back(error);
    }

    std::wstring mountNameStdStr(mountName.c_str());
    addNewMountEvent(false, true, mountNameStdStr, L' ', L"");

    #if wxCHECK_VERSION(2, 9, 0)
        wxCommandEvent* newEvent = new wxCommandEvent(myCustomEventType,
ID_ENCFS_MOUNT_ERROR_EVENT);
        wxQueueEvent(this, newEvent);
    #else
        wxCommandEvent newEvent(myCustomEventType, ID_ENCFS_MOUNT_ERROR_EVENT);
        AddPendingEvent(newEvent);
    #endif
}

void EncFSMPMainFrame::sendCommand(const wxString &command,

```

```

        const wxString &mountName,
        const wxString &password)
    {
        wxString args = EncFSMPIPC::marshalArguments(command, mountName, password);
        sendCommand(args);
    }

void EncFSMPMainFrame::sendCommand(const wxString &arg)
{
    #if wxCHECK_VERSION(2, 9, 0)
        wxCommandEvent* newEvent = new wxCommandEvent(myCustomEventType,
        ID_ENCFS_COMMAND);
        newEvent->SetString(arg);
        wxQueueEvent(this, newEvent);
    #else
        wxCommandEvent newEvent(myCustomEventType, ID_ENCFS_COMMAND);
        newEvent.SetString(arg);
        AddPendingEvent(newEvent);
    #endif
}

void EncFSMPMainFrame::unmountAllAndQuit()
{
    // Unmount
    std::list<MountEntry> &mountList = mountList_.getList();
    std::list<MountEntry>::iterator iter = mountList.begin();
    while(iter != mountList.end())
    {
        MountEntry &cur = (*iter);
        if(cur.mountState_ == MountEntry::MSMounted)
        {
            PFMPProxy::getInstance().unmount(cur.name_);
            cur.mountState_ = MountEntry::MSNotMounted; // Make sure
            OnMainFrameClose does not ask again...
        }

        iter++;
    }

    Close();
}

void EncFSMPMainFrame::OnMainFrameClose( wxCloseEvent& evt )
{
    // Check whether there are active or pending mounts
    int mountCount = 0, pendingCount = 0;
    std::list<MountEntry> &mountList = mountList_.getList();
    std::list<MountEntry>::iterator iter = mountList.begin();
    while(iter != mountList.end())
    {
        MountEntry &cur = (*iter);
        if(cur.mountState_ == MountEntry::MSMounted)
            mountCount++;
        else if(cur.mountState_ == MountEntry::MSPending)
            pendingCount++;

        iter++;
    }
    if(pendingCount > 0)
    {
        wxString msgText = wxString::Format(wxT("There are still %d pending
mounts.\n"))
            wxT("Are you sure you want to exit the program?\nData loss may
occur!"),

```

```

        pendingCount);

        int retVal = wxMessageBox(msgText, wxT("Confirm"), wxYES_NO | wxCANCEL,
this);
        if(retVal == wxCANCEL
            || retVal == wxNO)
        {
            evt.Veto();
            return;
        }
    }
    if(mountCount > 0
        && !disableUnmountDialogOnExit_)
    {
        wxString msgText = wxString::Format(wxT("There are still %d active
mounts.\n"),
            wxT("Are you sure you want to exit the program?"),
            mountCount);

        int retVal = wxMessageBox(msgText, wxT("Confirm"), wxYES_NO | wxCANCEL,
this);
        if(retVal == wxCANCEL
            || retVal == wxNO)
        {
            evt.Veto();
            return;
        }
    }

    // Point of no return
    PFMPProxy::getInstance().closingDown();

    // Unmount
    iter = mountList.begin();
    while(iter != mountList.end())
    {
        MountEntry &cur = (*iter);
        if(cur.mountState_ == MountEntry::MSMounted)
        {
            PFMPProxy::getInstance().unmount(cur.name_);
        }

        iter++;
    }

    saveWindowLayoutToConfig();

    aTimer_.Stop();
    Destroy();
}

void EncFSMPMainFrame::OnMainFrameIconize( wxIconizeEvent& event )
{
    if(
#ifdef wxCHECK_VERSION(2, 9, 0)
        wxTaskBarIcon::IsAvailable() &&
#endif
        minimizeToTray_)
    {
        if(event.IsIconized())
        {
            // Hide window, remove from task bar
            this->Hide();
            // Add tray icon

```

```

        if(pTaskBarIcon_ == NULL)
        {
            pTaskBarIcon_ = new EncFSMPTaskBarIcon();
            wxIcon icon = getIcon();

#ifdef EFS_MACOSX
            // On OS X, menu icons (NSStatusMenus) should be black and
            white

            wxBitmap iconBM;
            iconBM.CopyFromIcon(icon);
            wxImage img = iconBM.ConvertToImage();
            wxImage imgGrey = img.ConvertToGreyscale();
            wxBitmap iconBMGrey(iconBMGrey);
            icon.CopyFromBitmap(iconBMGrey);
#endif

            pTaskBarIcon_>SetIcon(icon, EncFSMPStrings::configAppName_);
            pTaskBarIcon_>setMainFrame(this);
        }
    }
    else
    {
        // Show window, add to task bar
#ifdef wxCHECK_VERSION(2, 9, 0)
        pTaskBarIcon_>Destroy();
#endif
        pTaskBarIcon_ = NULL;
    }
}

void EncFSMPMainFrame::OnExitMenuItem( wxCommandEvent& WXUNUSED(event) )
{
    Close();
}

void EncFSMPMainFrame::OnExportMenuItem( wxCommandEvent& event )
{
    MountEntry *pMountEntry = getSelectedMount();
    if(pMountEntry != NULL)
    {
        // Choose export directory
        wxDirDialog dlg(this, wxT("Please choose the export path:"));
        if(dlg.ShowModal() != wxID_OK)
            return;

        wxString exportPath = dlg.GetPath();

        // Enter password, if necessary
        wxString password = pMountEntry->password_;
        if(password.IsEmpty())
            password = pMountEntry->volatilePassword_;
        if(password.IsEmpty())
        {
            wxPasswordEntryDialog dlg(this, wxT("Please enter the password:"));
            int retVal = dlg.ShowModal();
            if(retVal == wxID_CANCEL)
                return;
            password = dlg.GetValue();
            if(savePasswordsInRAM_)
                pMountEntry->volatilePassword_ = password;
        }

        // TODO: Show progress bar, start separate thread?
        wxString errorMsg;
        bool isOK = true;
    }
}

```

```

        {
            wxWindowDisabler disableAll;
            wxBusyInfo wait(wxT("Exporting EncFS files..."));
            isOK = EncFSUtilities::exportEncFS(pMountEntry->encFSPath_,
                pMountEntry->externalConfigFileName_, pMountEntry-
>useExternalConfigFile_,
                password, exportPath, errorMsg);
        }
        if(isOK)
        {
            wxMessageBox(wxT("Exporting EncFS finished."), wxT("Export EncFS"),
                wxICON_INFORMATION | wxOK, this);
        }
        else
        {
            wxMessageBox(errorMsg,
                wxT("Export EncFS"), wxICON_ERROR | wxOK, this);
        }
    }

}

void EncFSMPMainFrame::OnShowErrorLogMenuItem( wxCommandEvent& event )
{
    if(pEncFSMPErrorLog_ != NULL)
    {
        pEncFSMPErrorLog_->toggleVisibleState();
    }
}

void EncFSMPMainFrame::OnShowErrorLogMenuItemUpdate( wxUpdateUIEvent& event )
{
    event.Enable(true);
    if(pEncFSMPErrorLog_ != NULL)
    {
        event.Check(pEncFSMPErrorLog_->isWindowShown());
    }
}

void EncFSMPMainFrame::OnShowErrorLogOnErrMenuItem( wxCommandEvent& event )
{
    if(pEncFSMPErrorLog_ != NULL)
        pEncFSMPErrorLog_->setShowErrorLogOnErr(event.IsChecked());
    saveWindowLayoutToConfig();
}

void EncFSMPMainFrame::OnDisableUnmountDialogOnExitMenuItem( wxCommandEvent& event )
{
    disableUnmountDialogOnExit_ = event.IsChecked();
    saveWindowLayoutToConfig();
}

void EncFSMPMainFrame::OnSavePasswordsInRAMMenuItem( wxCommandEvent& event )
{
    savePasswordsInRAM_ = event.IsChecked();
    saveWindowLayoutToConfig();
}

void EncFSMPMainFrame::OnAboutMenuItem( wxCommandEvent& WXUNUSED(event) )
{
    wxAboutDialogInfo info;
    info.SetName(wxT(ENCFSMP_NAME) wxT(" ") wxT(ENCFSMP_ARCHITECTURE_STRING));
    info.SetVersion(wxString(ENCFSMP_VERSION_STRING, *wxConvCurrent));
    info.SetIcon(getIcon());
}

```

```

wxString descrString;
descrString = wxT("Mount EncFS folders - multiplatform style.\n\n");
descrString.Append(wxT("Built with: ") wxT(ENCFSMP_COMPILER) wxT(" ")
wxT(ENCFSMP_COMPILER_VERSION) wxT("\n"));
descrString.Append(wxT("Linked with:\n - ") wxVERSION_STRING wxT("\n - "));
descrString.Append(wxT("EncFS ") wxT(EFS_STRINGIFY(VERSION)) wxT("\n - "));
descrString.Append(wxString(wxT(OPENSSEL_VERSION_TEXT)));

wxString boostVersion;
boostVersion.Printf(wxT("%d.%d.%d"), BOOST_VERSION / 100000, BOOST_VERSION / 100 %
1000, BOOST_VERSION % 100);
descrString.Append(wxT("\n - boost "));
descrString.Append(boostVersion);
descrString.Append(wxT("\n\n"));

if(PFMPProxy::getInstance().isPFMPPresent())
    descrString.Append(wxString(wxT("Pismo File Mount version found: "))
+ wxString(PFMPProxy::getInstance().getPFMVersionString().c_str(),
*wxConvCurrent) );
else
    descrString.Append(wxT("No Pismo File Mount installation found"));

descrString.Append(wxT("\nIcon \u00A9 by Oxygen\n"));

info.SetDescription(descrString);
info.SetCopyright(wxT("\u00A9 ") wxT(ENCFSMP_COPYRIGHT_YEAR) wxT(" ")
wxT(ENCFSMP_COPYRIGHT_NAME));

wxAboutBox(info);
}

void EncFSMPMainFrame::OnCreateMountButton( wxCommandEvent& event )
{
    CreateNewEncFSDialog dlg(this);
    dlg.setMountList(&mountList_);
    if(dlg.ShowModal() == wxID_OK)
    {
        if(!EncFSUtilities::createEncFS(dlg.getEncFSPath(),
            dlg.password_, dlg.getExternalConfigFileName(),
            dlg.getUseExternalConfigFile(),
            dlg.getCipherAlgorithm(), dlg.getCipherKeySize(),
            dlg.getCipherBlockSize(), dlg.getNameEncoding(),
            dlg.getKeyDerivationDuration(), dlg.perBlockHMAC_,
            dlg.uniqueIV_, dlg.chainedIV_, dlg.externalIV_))
        {
            wxMessageBox(wxT("Creation of new EncFS failed"), wxT("Error"),
                wxOK | wxICON_ERROR);
            return;
        }

        wxString password;
        if(dlg.storePassword_)
            password = dlg.password_;
        mountList_.addMount(dlg.mountName_, dlg.getEncFSPath(),
            dlg.getExternalConfigFileName(), dlg.getDriveLetter(),
            password, dlg.getUseExternalConfigFile(), dlg.getCachingEnabled(),
            dlg.worldWritable_, dlg.getIsLocalDrive(), false);

        mountList_.storeToConfig();

        updateMountListCtrl();
    }
}

```

```

}

void EncFSMPMainFrame::OnOpenExistingEncFSButton( wxCommandEvent& event )
{
    OpenExistingFSDialog dlg(this);
    dlg.setMountList(&mountList_);
    if(dlg.ShowModal() == wxID_OK)
    {
        // Check whether path is EncFS path
        EncFSUtilities::EncFSInfo info;
        bool encFSFound = EncFSUtilities::getEncFSInfo(dlg.getEncFSPath(),
            dlg.getExternalConfigFileName(), dlg.getUseExternalConfigFile(),
info);
        if(!encFSFound)
        {
            wxString msg(wxT("No supported encrypted filesystem found.\n"));
            if(!info.configVersionString.empty())
                msg.Append(info.configVersionString);
            wxMessageBox(msg, wxT("Open existing EncFS failed"), wxOK |
wxICON_ERROR);
            return;
        }
        wxString password;
        if(dlg.storePassword_)
            password = dlg.password_;

        mountList_.addMount(dlg.mountName_, dlg.getEncFSPath(),
            dlg.getExternalConfigFileName(),
            wxString(dlg.getDriveLetter()), password,
            dlg.getUseExternalConfigFile(), dlg.getCachingEnabled(),
            dlg.worldWritable_, dlg.getIsLocalDrive(), false);

        mountList_.storeToConfig();
        updateMountListCtrl();
    }
}

void EncFSMPMainFrame::OnMountsListColClick( wxListEvent& event )
{
}

void EncFSMPMainFrame::OnMountsListItemDeselected( wxListEvent& event )
{
    updateButtonStates();
}

void EncFSMPMainFrame::OnMountsListItemRightClick( wxListEvent& event )
{
    if(pMountsListPopupMenu_ != NULL)
    {
        delete pMountsListPopupMenu_;
        pMountsListPopupMenu_ = NULL;
    }

    bool isPFMPresent = PFMPProxy::getInstance().isPFMPresent();
    MountEntry *pMountEntry = getSelectedMount();
/*
    ID_CTXMOUNT,
    ID_CTXREMOVE,
    ID_CTXEDIT,
    ID_CTXBROWSE,
    ID_CTXSHOWINFO,
    ID_CTXCHANGEPASSWD,
    ID_CTXEXPORT

```

```

*/

// Don't show popup menu if no selected entry was found or mount is pending
if(pMountEntry == NULL
    || pMountEntry->mountState_ == MountEntry::MSPending)
    return;

pMountsListPopupMenu_ = new wxMenu();
if(pMountEntry->mountState_ == MountEntry::MSMounted)
{
    if(isPFMPresent)
        pMountsListPopupMenu_->Append(ID_CTXMOUNT, wxT("Unmount"));
    pMountsListPopupMenu_->Append(ID_CTXBROWSE, wxT("Browse"));
}
else
{
    if(isPFMPresent)
        pMountsListPopupMenu_->Append(ID_CTXMOUNT, wxT("Mount"));
    pMountsListPopupMenu_->Append(ID_CTXREMOVE, wxT("Remove"));
    pMountsListPopupMenu_->Append(ID_CTXEDIT, wxT("Edit"));
    pMountsListPopupMenu_->Append(ID_CTXSHOWINFO, wxT("Show Info"));
    pMountsListPopupMenu_->Append(ID_CTXCHANGEPASSWD, wxT("Change password"));
    pMountsListPopupMenu_->Append(ID_CTXEXPORT, wxT("Export"));
}

PopupMenu(pMountsListPopupMenu_);
}

void EncFSMPMainFrame::OnMountsListItemSelected( wxListEvent& event )
{
    updateButtonStates();
}

void EncFSMPMainFrame::OnMountsListKeyDown( wxListEvent& event )
{
    int keyCode = event.GetKeyCode();
    if(keyCode == W XK_DELETE || keyCode == W XK_BACK)
    {
        // Delete selected item
        MountEntry *pMountEntry = getSelectedMount();
        if(pMountEntry != NULL)
        {
            std::list<MountEntry> &mountList = mountList_.getList();
            std::list<MountEntry>::iterator iter = mountList.begin();
            while(iter != mountList.end())
            {
                MountEntry &cur = (*iter);
                if(pMountEntry->name_ == cur.name_
                    && pMountEntry->mountState_ ==
MountEntry::MSNotMounted)
                {
                    mountList.erase(iter);
                    mountList_.storeToConfig();
                    updateMountListCtrl();
                    return;
                }
                iter++;
            }
        }
    }
}

void EncFSMPMainFrame::OnMountButton( wxCommandEvent& event )
{

```

```

MountEntry *pMountEntry = getSelectedMount();
if(pMountEntry != NULL)
{
    if(pMountEntry->mountState_ == MountEntry::MSMounted)
    {
        PFMProxy::getInstance().unmount(pMountEntry->name_);
        pMountEntry->mountState_ = MountEntry::MSPending;
    }
    else if(pMountEntry->mountState_ == MountEntry::MSNotMounted)
    {
        // Mount: Start a PFMHandlerThread
        wxString password = pMountEntry->password_;
        if(password.IsEmpty())
            password = pMountEntry->volatilePassword_;
        if(password.IsEmpty())
        {
            wxPasswordEntryDialog dlg(this, wxT("Please enter the
password:"));

            int retVal = dlg.ShowModal();
            if(retVal == wxID_CANCEL)
                return;
            password = dlg.GetValue();
            if(savePasswordsInRAM_)
                pMountEntry->volatilePassword_ = password;
        }
        bool isWorldWritable = pMountEntry->isWorldWritable_;

        PFMHandlerThread *pPFMHandlerThread = new PFMHandlerThread();
        pPFMHandlerThread->setParameters(pMountEntry->name_,
            pMountEntry->encFSPPath_, pMountEntry->externalConfigFileName_,
            pMountEntry->driveLetter_, password,
            pMountEntry->useExternalConfigFile_, pMountEntry->
>enableCaching_,
                isWorldWritable, pMountEntry->isLocalDrive_, false);

        pPFMHandlerThread->Create();
        pPFMHandlerThread->Run();
        pMountEntry->mountState_ = MountEntry::MSPending;
    }

    updateMountListCtrl();
}

}

void EncFSMPMainFrame::OnRemoveButton( wxCommandEvent& event )
{
    MountEntry *pMountEntry = getSelectedMount();
    if(pMountEntry != NULL)
    {
        std::list<MountEntry> &mountList = mountList_.getList();
        std::list<MountEntry>::iterator iter = mountList.begin();
        while(iter != mountList.end())
        {
            MountEntry &cur = (*iter);
            if(pMountEntry->name_ == cur.name_)
            {
                mountList.erase(iter);
                mountList_.storeToConfig();
                updateMountListCtrl();
                return;
            }
            iter++;
        }
    }
}

```

```

}

void EncFSMPMainFrame::OnEditButton( wxCommandEvent& event )
{
    MountEntry *pMountEntry = getSelectedMount();
    if(pMountEntry != NULL)
    {
        OpenExistingFSDialog dlg(this);
        dlg.setMountList(&mountList_);
        dlg.setEditMode(pMountEntry->name_,
            pMountEntry->encFSPath_,
            pMountEntry->useExternalConfigFile_, pMountEntry->
>externalConfigFileName_,
            pMountEntry->driveLetter_[0], pMountEntry->isLocalDrive_,
            pMountEntry->password_, pMountEntry->isWorldWritable_,
            pMountEntry->enableCaching_);
        if(dlg.ShowModal() == wxID_OK)
        {
            wxString password;
            if(dlg.storePassword_)
                password = dlg.password_;
            else
                password = wxEmptyString;

            pMountEntry->driveLetter_ = wxString(dlg.getDriveLetter());
            pMountEntry->password_ = password;
            pMountEntry->useExternalConfigFile_ = dlg.getUseExternalConfigFile();
            pMountEntry->externalConfigFileName_ =
dlg.getExternalConfigFileName();
            pMountEntry->enableCaching_ = dlg.getCachingEnabled();
            pMountEntry->isLocalDrive_ = dlg.getIsLocalDrive();
            pMountEntry->isWorldWritable_ = dlg.worldWritable_;

            mountList_.storeToConfig();
            updateMountListCtrl();
        }
    }
}

void EncFSMPMainFrame::OnBrowseButton( wxCommandEvent& event )
{
    MountEntry *pMountEntry = getSelectedMount();
    if(pMountEntry != NULL)
    {
        #if defined(EFS_WIN32)
            wxString path = pMountEntry->assignedDriveLetter_;
            if(!path.IsEmpty())
                path.Append(wxT(":\\"));
            else
                path = pMountEntry->assignedMountPoint_;

            if(!wxFileName::DirExists(path))
                return;

            wxString winDir = wxGetOSDirectory();
            wxFileName fn(winDir, wxT("Explorer.exe"));
            if(fn.FileExists())
            {
                wxString cmd;
                cmd = fn.GetFullPath();
                cmd.Append(wxT(" "));
                cmd.Append(path);
                wxExecute(cmd, wxEXEC_ASYNC);
            }
        #endif
    }
}

```

```

        else
        {
            wxMessageBox(wxT("Explorer.exe not found!"),
                wxT("Serious error"), wxICON_ERROR | wxOK, this);
        }
    #elif defined(EFS_MACOSX)
        wxString path = pMountEntry->assignedMountPoint_;
        wxString cmd(wxT("/usr/bin/open \""));
        cmd.Append(path);
        cmd.Append(wxT("\\\""));
        wxExecute(cmd, wxEXEC_ASYNC);
    #endif
}

void EncFSMPMainFrame::OnInfoButton( wxCommandEvent& event )
{
    MountEntry *pMountEntry = getSelectedMount();
    if(pMountEntry != NULL)
    {
        wxString path = pMountEntry->encFSPath_;
        EncFSUtilities::EncFSInfo info;
        bool retVal = EncFSUtilities::getEncFSInfo(path,
            pMountEntry->externalConfigFileName_, pMountEntry->
            useExternalConfigFile_, info);
        if(retVal)
        {
            ShowEncFSInfoDialog dlg(this);
            dlg.mountName_ = pMountEntry->name_;
            dlg.encFSPath_ = pMountEntry->encFSPath_;
            dlg.configVersion_ = info.configVersionString;
            dlg.cipherAlgorithm_ = info.cipherAlgorithm;
            dlg.cipherKeySize_ = wxString::Format(wxT("%i"), info.cipherKeySize);
            dlg.cipherBlocksize_ = wxString::Format(wxT("%i"),
            info.cipherBlockSize);
            dlg.nameEncoding_ = info.nameEncoding;
            dlg.keyDerivIterations_ = wxString::Format(wxT("%i"),
            info.keyDerivationIterations);
            dlg.saltSize_ = wxString::Format(wxT("%i"), info.saltSize);
            dlg.perBlockHMAC_ = info.perBlockHMAC;
            dlg.pPerBlockHMACCheckBox_->Enable(false);
            dlg.uniqueIV_ = info.uniqueIV;
            dlg.pUniqueIVCheckBox_->Enable(false);
            dlg.chainedIV_ = info.chainedIV;
            dlg.pChainedIVCheckBox_->Enable(false);
            dlg.externalIV_ = info.externalIV;
            dlg.pExternalIVCheckBox_->Enable(false);
            dlg.SetExtraStyle( dlg.GetExtraStyle() |
            wxWS_EX_VALIDATE_RECURSIVELY);
            dlg.ShowModal();
        }
        else
        {
            wxMessageBox(info.configVersionString,
                wxT("Show EncFS info"), wxICON_ERROR | wxOK, this);
        }
    }
}

void EncFSMPMainFrame::OnChangePasswordButton( wxCommandEvent& event )
{
    MountEntry *pMountEntry = getSelectedMount();
    if(pMountEntry != NULL)
    {

```

					ПГУ 2.090301.116.001 ПЗ	Лист
						116
Изм.	Лист	№ докум.	Подпись	Дата		

```

wxString path = pMountEntry->encFSPath_;

ChangePasswordDialog dlg(this);
if(dlg.ShowModal() == wxID_OK)
{
    wxString errorMsg;
    bool isOK = EncFSUtilities::changePassword(path,
        pMountEntry->externalConfigFileName_, pMountEntry-
>useExternalConfigFile_,
        dlg.oldPassword_, dlg.newPassword_, errorMsg);
    if(isOK)
    {
        if(dlg.storeNewPassword_)
            pMountEntry->password_ = dlg.newPassword_;
        else
            pMountEntry->password_ = wxEmptyString;
        pMountEntry->volatilePassword_.Empty();
        mountList_.storeToConfig();

        wxMessageBox(errorMsg,
            wxT("Change password successful"), wxICON_INFORMATION |
wxOK, this);
    }
    else
    {
        wxMessageBox(errorMsg,
            wxT("Change password error"), wxICON_ERROR | wxOK,
this);
    }
}

}

}

void EncFSMPMainFrame::OnTimer( wxTimerEvent & WXUNUSED(event) )
{
    if(firstTimeOnTimer_)
    {
        firstTimeOnTimer_ = false;

        updateButtonStates();
    }
}

void EncFSMPMainFrame::OnMountEvent( wxCommandEvent &event )
{
    wxMutexLocker lock(mountEventsMutex_);

    bool updateListCtrl = false;
    while(!mountEvents_.empty())
    {
        const MountEvent &evt = mountEvents_.front();

        // Update mount list
        MountEntry *pMountEntry = mountList_.findEntryByName(evt.mountName_);
        if(pMountEntry != NULL)
        {
            if(evt.driveLetter_ == 0)
                pMountEntry->assignedDriveLetter_ = wxEmptyString;
            else
                pMountEntry->assignedDriveLetter_ =
wxString(evt.driveLetter_);
            pMountEntry->assignedMountPoint_ = wxString(evt.mountPoint_.c_str());
            if(evt.isMountEvent_)

```

```

        pMountEntry->mountState_ = MountEntry::MSMounted;
    else
        pMountEntry->mountState_ = MountEntry::MSNotMounted;
    if(evt.isError_)
        pMountEntry->volatilePassword_.Empty(); // Clear volatile
password
        updateListCtrl = true;
    }

    mountEvents_.pop_front();
}

if(updateListCtrl)
{
    updateMountListCtrl();
    updateButtonStates();
}
}

void EncFSMPMainFrame::OnEncFSMountErrorEvent( wxCommandEvent &event )
{
    std::list<wxString> encFSErrorListCopy;
    {
        // Copy the list locally, so that the mutex is only locked for a short time
        wxMutexLocker lock(encFSErrorMutex_);
        encFSErrorListCopy = encFSErrorList_;
        encFSErrorList_.clear();
    }

    std::list<wxString>::iterator iter = encFSErrorListCopy.begin();
    while(iter != encFSErrorListCopy.end())
    {
        wxString &errMsg = (*iter);
        wxMessageBox(errMsg, wxT("EncFS error"), wxOK | wxICON_ERROR);
        iter++;
    }
}

void EncFSMPMainFrame::OnUAC( wxCommandEvent& event )
{
#ifdef EFS_WIN32
    if(Win32Utils::restartAsAdmin((HWND)this->GetHWND()))
        Close();
#endif
}

void EncFSMPMainFrame::OnMinimizeToTrayMenuItem( wxCommandEvent& event )
{
#ifdef EFS_MACOSX
    wxIconizeEvent dummyEvent(0, true);
    dummyEvent.SetEventObject(this);
    OnMainFrameIconize(dummyEvent);
#else
    minimizeToTray_ = (event.GetInt() != 0);
    saveWindowLayoutToConfig();
#endif
}

void EncFSMPMainFrame::OnContextMenuMount( wxCommandEvent& event )
{
    OnMountButton(event);
}

```

```

void EncFSMPMainFrame::OnContextMenuRemove( wxCommandEvent& event )
{
    OnRemoveButton(event);
}

void EncFSMPMainFrame::OnContextMenuEdit( wxCommandEvent& event )
{
    OnEditButton(event);
}

void EncFSMPMainFrame::OnContextMenuBrowse( wxCommandEvent& event )
{
    OnBrowseButton(event);
}

void EncFSMPMainFrame::OnContextMenuShowInfo( wxCommandEvent& event )
{
    OnInfoButton(event);
}

void EncFSMPMainFrame::OnContextMenuChangePassword( wxCommandEvent& event )
{
    OnChangePasswordButton(event);
}

void EncFSMPMainFrame::OnContextMenuExport( wxCommandEvent& event )
{
    OnExportMenuItem(event);
}

void EncFSMPMainFrame::OnEncFSCommand( wxCommandEvent &event )
{
    wxString command, mountName, passwordCmd;
    if(EncFSMPIPC::unmarshalArguments(event.GetString(), command,
        mountName, passwordCmd))
    {
        bool isMountCommand = (command.IsSameAs(EncFSMPStrings::commandMount_,
false));
        bool isUnmountCommand = (command.IsSameAs(EncFSMPStrings::commandUnmount_,
false));
        if(!isMountCommand && !isUnmountCommand)
        {
            wxMessageBox(wxString(wxT("Unknown command \"") + command
+ wxString(wxT("\" received")),
wxT("Unknown command"), wxOK | wxICON_ERROR));
            return;
        }
        MountEntry *pMountEntry = mountList_.findEntryByName(mountName);
        if(pMountEntry != NULL)
        {
            if(pMountEntry->mountState_ == MountEntry::MSMounted)
            {
                if(isUnmountCommand)
                {
                    PFMPProxy::getInstance().unmount(pMountEntry->name_);
                    pMountEntry->mountState_ = MountEntry::MSPending;
                }
                else
                {
                    wxMessageBox(wxT("EncFS folder is already mounted"),
wxT("Mount command"), wxOK | wxICON_ERROR);
                    return;
                }
            }
        }
    }
}

```



```

        wxBitmap myimage_bmp(myimage_img);
        wxIcon icon;
        icon.CopyFromBitmap(myimage_bmp);
    #endif

    return icon;
}

void EncFSMPMainFrame::updateMountListCtrl()
{
    std::list<MountEntry> &mountList = mountList_.getList();
    int listCtrlItemCount = pMountsListCtrl_>GetItemCount();
    int mountsCount = static_cast<int>(mountList.size());

    std::list<MountEntry>::iterator iter = mountList.begin();
    //while(iter != mountList.end())
    for(int i = 0; i < mountsCount; i++, iter++)
    {
        MountEntry &curEntry = (*iter);
        wxString mountedString(wxT("No"));
        if(curEntry.mountState_ == MountEntry::MSMounted)
            mountedString = wxT("Yes");
        else if(curEntry.mountState_ == MountEntry::MSPending)
            mountedString = wxT("Pending");

        if(i >= listCtrlItemCount)
        {
            pMountsListCtrl_>InsertItem(i, mountedString);
        }
        else
        {
            pMountsListCtrl_>SetItem(i, 0, mountedString);
        }
        pMountsListCtrl_>SetItem(i, 1, curEntry.name_);
        pMountsListCtrl_>SetItem(i, 2, curEntry.encFSPath_);
    #if defined(EFS_WIN32)
        wxString driveLetterStr(wxT("Auto"));
        if(curEntry.mountState_ == MountEntry::MSMounted)
        {
            if(!curEntry.assignedDriveLetter_.IsEmpty())
                driveLetterStr = curEntry.assignedDriveLetter_ + wxT(":");
            else
                driveLetterStr = curEntry.assignedMountPoint_;
        }
        else
        {
            if(curEntry.driveLetter_ == wxT("-"))
                driveLetterStr = wxT("None");
            else if(curEntry.driveLetter_ != wxT("?"))
                driveLetterStr = curEntry.driveLetter_ + wxT(":");
        }
        pMountsListCtrl_>SetItem(i, 3, driveLetterStr);
    #endif

        //iter++;
    }

    // Remove superfluous entries in list control
    for(int i = mountsCount; i < listCtrlItemCount; i++)
    {
        pMountsListCtrl_>DeleteItem(i);
    }
}

```

```

MountEntry *EncFSMPMainFrame::getSelectedMount()
{
    int selectedItem = -1;
    int stateMask = wxLIST_STATE_SELECTED;
    int itemCount = pMountsListCtrl_>GetItemCount();
    for(int i = 0; i < itemCount; i++)
    {
        int itemState = pMountsListCtrl_>GetItemState(i, stateMask);
        if(itemState != 0)
        {
            selectedItem = i;
        }
    }

    if(selectedItem >= 0)
    {
#ifdef wxCHECK_VERSION(2, 9, 0)
        wxString selectedMountName = pMountsListCtrl_>GetItemText(selectedItem,
1);
#else
        wxListItem info;
        info.SetMask(wxLIST_MASK_TEXT);
        info.SetId(selectedItem);
        info.SetColumn(1);
        wxString selectedMountName;

        if(pMountsListCtrl_>GetItem(info))
            selectedMountName = info.GetText();
#endif

        MountEntry *pMountEntry = mountList_.findEntryByName(selectedMountName);
        return pMountEntry;
    }

    return NULL;
}

void EncFSMPMainFrame::updateButtonStates()
{
    bool isPFMPresent = PFMPProxy::getInstance().isPFMPresent();
    MountEntry *pMountEntry = getSelectedMount();
    if(pMountEntry != NULL)
    {
        if(pMountEntry->mountState_ == MountEntry::MSMounted)
        {
            pMountButton_>Enable(isPFMPPresent);
            pMountButton_>SetLabel(wxT("Unmount"));
            pRemoveButton_>Enable(false);
            pEditButton_>Enable(false);
            pBrowseButton_>Enable(isPFMPPresent);
            pInfoButton_>Enable(false);
            pChangePasswordButton_>Enable(false);
            pExportMenuItem_>Enable(false);
        }
        else if(pMountEntry->mountState_ == MountEntry::MSNotMounted)
        {
            pMountButton_>Enable(isPFMPPresent);
            pMountButton_>SetLabel(wxT("Mount"));
            pRemoveButton_>Enable(true);
            pEditButton_>Enable(true);
            pBrowseButton_>Enable(false);
            pInfoButton_>Enable(true);
            pChangePasswordButton_>Enable(true);
            pExportMenuItem_>Enable(true);
        }
    }
}

```

```

        else // pending
        {
            pMountButton_->Enable(false);
            pMountButton_->SetLabel(wxT("Unmount"));
            pRemoveButton_->Enable(false);
            pEditButton_->Enable(false);
            pBrowseButton_->Enable(false);
            pInfoButton_->Enable(false);
            pChangePasswordButton_->Enable(false);
            pExportMenuItem_->Enable(false);
        }
    }
    else
    {
        pMountButton_->Enable(false);
        pRemoveButton_->Enable(false);
        pEditButton_->Enable(false);
        pBrowseButton_->Enable(false);
        pInfoButton_->Enable(false);
        pChangePasswordButton_->Enable(false);
        pExportMenuItem_->Enable(false);
    }
}

void EncFSMPMainFrame::saveWindowLayoutToConfig()
{
    std::auto_ptr<wxConfig> config(new wxConfig(EncFSMPStrings::configAppName_,
        EncFSMPStrings::configOrganizationName_));

    {
        wxPoint pt = GetPosition();
        wxSize sz = GetSize();

        std::ostringstream ostr;
        ostr << pt.x << L" " << pt.y << L" " << sz.x << L" " << sz.y;
        wxString dimensions(ostr.str().c_str());

        config->Write(EncFSMPStrings::configWindowDimensions_, dimensions);
    }

    {
        std::ostringstream ostr;
        int columnCount = pMountsListCtrl_->GetColumnCount();
        for(int i = 0; i < columnCount; i++)
        {
            int width = pMountsListCtrl_->GetColumnWidth(i);
            ostr << width << " ";
        }
        wxString columnWidths(ostr.str().c_str());

        config->Write(EncFSMPStrings::configColumnWidths_, columnWidths);
    }

    config->Write(EncFSMPStrings::configMinimizeToTray_, minimizeToTray_);
    config->Write(EncFSMPStrings::configDisableUnmountDialogOnExit_,
        disableUnmountDialogOnExit_);
    config->Write(EncFSMPStrings::configSavePasswordsInRAM_, savePasswordsInRAM_);

    if(pEncFSMPErrorLog_ != NULL)
        config->Write(EncFSMPStrings::configShowErrorLogOnErr_, pEncFSMPErrorLog_-
            >getShowErrorLogOnErr());
}

void EncFSMPMainFrame::loadWindowLayoutFromConfig()

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		123

					ПГУ 2.090301.116.001 ПЗ	Лист
						124
Изм.	Лист	№ докум.	Подпись	Дата		

```

EVT_MENU( ID_CTXREMOVE, EncFSMPMainFrame::OnContextMenuRemove )
EVT_MENU( ID_CTXEDIT, EncFSMPMainFrame::OnContextMenuEdit )
EVT_MENU( ID_CTXBROWSE, EncFSMPMainFrame::OnContextMenuBrowse )
EVT_MENU( ID_CTXSHOWINFO, EncFSMPMainFrame::OnContextMenuShowInfo )
EVT_MENU( ID_CTXCHANGEPASSWD, EncFSMPMainFrame::OnContextMenuChangePassword )
EVT_MENU( ID_CTXEXPORT, EncFSMPMainFrame::OnContextMenuExport )
EVT_COMMAND(ID_ENCFSS_COMMAND, myCustomEventType, EncFSMPMainFrame::OnEncFSCommand)
END_EVENT_TABLE()

```

Файл « EncFSMPMainFrame.h»

```

#ifndef ENCFSMPTASKBARICON_H
#define ENCFSMPTASKBARICON_H

class EncFSMPTaskBarIcon;
class wxMenu;
class EncFSMPErrorLog;

#include <list>
#include "EncFSMPMainFrameBase.h"
#include "MountList.h"

class EncFSMPMainFrame: public EncFSMPMainFrameBase
{
public:
    EncFSMPMainFrame(wxWindow* parent);
    virtual ~EncFSMPMainFrame();

    void addNewMountEvent(bool isMountEvent, bool isError,
        const std::wstring &mountName, wchar_t driveLetter, const std::wstring
&mountPoint);
    void reportEncFSError(const wxString &error, const wxString &mountName);
    void sendCommand(const wxString &command,
        const wxString &mountName,
        const wxString &password);
    void sendCommand(const wxString &arg);

    void unmountAllAndQuit();

    virtual void OnMainFrameIconize( wxIconizeEvent& event );

protected:
    virtual void OnMainFrameClose( wxCloseEvent& event );
    virtual void OnExitMenuItem( wxCommandEvent& event );
    virtual void OnExportMenuItem( wxCommandEvent& event );
    virtual void OnShowErrorLogMenuItem( wxCommandEvent& event );
    virtual void OnShowErrorLogMenuItemUpdate( wxUpdateUIEvent& event );
    virtual void OnShowErrorLogOnErrMenuItem( wxCommandEvent& event );
    virtual void OnDisableUnmountDialogOnExitMenuItem( wxCommandEvent& event );
    virtual void OnSavePasswordsInRAMMenuItem( wxCommandEvent& event );
    virtual void OnAboutMenuItem( wxCommandEvent& event );
    virtual void OnCreateMountButton( wxCommandEvent& event );
    virtual void OnOpenExistingEncFSButton( wxCommandEvent& event );
    virtual void OnMountsListColClick( wxListEvent& event );
    virtual void OnMountsListItemDeselected( wxListEvent& event );
    virtual void OnMountsListItemRightClick( wxListEvent& event );
    virtual void OnMountsListItemSelected( wxListEvent& event );
    virtual void OnMountsListKeyDown( wxListEvent& event );
    virtual void OnMountButton( wxCommandEvent& event );
    virtual void OnRemoveButton( wxCommandEvent& event );
    virtual void OnEditButton( wxCommandEvent& event );
    virtual void OnBrowseButton( wxCommandEvent& event );
    virtual void OnInfoButton( wxCommandEvent& event );

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		125

```

virtual void OnChangePasswordButton( wxCommandEvent& event );

virtual void OnTimer( wxTimerEvent &event );

virtual void OnMountEvent( wxCommandEvent &event );
virtual void OnEncFSMountErrorEvent( wxCommandEvent &event );
virtual void OnUAC( wxCommandEvent& event );
virtual void OnMinimizeToTrayMenuItem( wxCommandEvent& event );
virtual void OnContextMenuMount( wxCommandEvent& event );
virtual void OnContextMenuRemove( wxCommandEvent& event );
virtual void OnContextMenuEdit( wxCommandEvent& event );
virtual void OnContextMenuBrowse( wxCommandEvent& event );
virtual void OnContextMenuShowInfo( wxCommandEvent& event );
virtual void OnContextMenuChangePassword( wxCommandEvent& event );
virtual void OnContextMenuExport( wxCommandEvent& event );
virtual void OnEncFSCommand( wxCommandEvent &event );

wxIcon getIcon();
void updateMountListCtrl();
MountEntry *getSelectedMount();
void updateButtonStates();
void saveWindowLayoutToConfig();
void loadWindowLayoutFromConfig();

private:
wxTimer aTimer_;
wxMenuItem *pMinimizeToTrayMenuItem_;
bool minimizeToTray_, disableUnmountDialogOnExit_;
bool savePasswordsInRAM_;
EncFSMPTaskBarIcon *pTaskBarIcon_;
wxMenu *pMountsListPopupMenu_;

bool firstTimeOnTimer_;

class MountEvent
{
public:
MountEvent() : isMountEvent_(true), isError_(false), driveLetter_(L'A') { }
MountEvent(const MountEvent &o) { copy(o); }
virtual ~MountEvent() { }
MountEvent &copy(const MountEvent &o)
{
    isMountEvent_ = o.isMountEvent_;
    isError_ = o.isError_;
    mountName_ = o.mountName_;
    driveLetter_ = o.driveLetter_;
    mountPoint_ = o.mountPoint_;
    return *this;
}
MountEvent & operator=(const MountEvent & o)
{
    return copy(o);
}

bool isMountEvent_;           // true: mount, false: unmount
bool isError_;               // Some error occurred during mount
std::wstring mountName_, mountPoint_;
wchar_t driveLetter_;
};

wxMutex mountEventsMutex_;
std::list<MountEvent> mountEvents_;

```

```

wxMutex encFSErrorMutex_;
std::list<wxString> encFSErrorList_;

MountList mountList_;

EncFSMPErrorLog *pEncFSMPErrorLog_;

DECLARE_EVENT_TABLE()
};

#endif

```

Файл « MountList.cpp»

```

#include "CommonIncludes.h"

#include "MountList.h"
#include "EncFSMPStrings.h"

#include <wx/config.h>
#include <memory>

MountList::MountList()
{
}

MountList::~MountList()
{
}

bool MountList::addMount(wxString name, wxString encFSPath, wxString
externalConfigFileName,
                        wxString driveLetter, wxString password, bool useExternalConfigFile,
                        bool enableCaching, bool isWorldWritable, bool isLocalDrive, bool
isMounted)
{
    wxString correctedName = name;

    // Remove any paths in front of name
    wxFileName nameFN(correctedName);
    correctedName = nameFN.GetFullName();

    // Check for double entry
    if(findEntryByName(correctedName) != NULL)
        return false;

    MountEntry newEntry;
    newEntry.name_ = correctedName;
    newEntry.encFSPath_ = encFSPath;
    newEntry.externalConfigFileName_ = externalConfigFileName;
    newEntry.driveLetter_ = driveLetter;
    newEntry.password_ = password;
    newEntry.useExternalConfigFile_ = useExternalConfigFile;
    newEntry.isWorldWritable_ = isWorldWritable;
    newEntry.isLocalDrive_ = isLocalDrive;
    newEntry.mountState_ = (isMounted ? MountEntry::MSMounted :
MountEntry::MSNotMounted);

    mountEntries_.push_back(newEntry);

    return true;
}

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		127

```

MountEntry *MountList::findEntryByName(wxString name)
{
    wxString correctedName = name;

    // Remove any paths in front of name
    wxFileName nameFN(correctedName);
    correctedName = nameFN.GetFullName();

    std::list<MountEntry>::iterator iter = mountEntries_.begin();
    while(iter != mountEntries_.end())
    {
        MountEntry &cur = (*iter);

        if(cur.name_ == correctedName)
            return &cur;

        iter++;
    }

    return NULL;
}

bool MountList::storeToConfig()
{
    std::auto_ptr<wxConfig> config(new wxConfig(EncFSMPStrings::configAppName_,
    EncFSMPStrings::configOrganizationName_));
    config->DeleteGroup(EncFSMPStrings::configMountListPath_);
    config->SetPath(EncFSMPStrings::configMountListPath_);

    int i = 0;
    std::list<MountEntry>::iterator iter = mountEntries_.begin();
    while(iter != mountEntries_.end())
    {
        MountEntry &cur = (*iter);

        wxString grpName = wxString::Format(wxT("MountEntry%d"), i);
        config->SetPath(grpName);

        if(!config->Write(EncFSMPStrings::configNameKey_, cur.name_))
            return false;
        config->Write(EncFSMPStrings::configEncFSPathKey_, cur.encFSPath_);
        config->Write(EncFSMPStrings::configExternalConfigFileKey_,
cur.externalConfigFileName_);
        config->Write(EncFSMPStrings::configDriveLetterKey_, cur.driveLetter_);
        if(cur.password_.Length() > 0)
            config->Write(EncFSMPStrings::configPasswordKey_, cur.password_);
        config->Write(EncFSMPStrings::configUseExternalConfigFileKey_,
cur.useExternalConfigFile_);
        config->Write(EncFSMPStrings::configEnableCachingKey_, cur.enableCaching_);
        config->Write(EncFSMPStrings::configIsWorldWritableKey_,
cur.isWorldWritable_);
        config->Write(EncFSMPStrings::configIsLocalDriveKey_, cur.isLocalDrive_);

        config->SetPath(wxT(".."));

        iter++;
        i++;
    }

    return true;
}

bool MountList::loadFromConfig()

```

					ПГУ 2.090301.116.001 ПЗ	Лист
						128
Изм.	Лист	№ докум.	Подпись	Дата		

```

{
    long index = 0;
    wxString str;

    std::auto_ptr<wxConfig> config(new wxConfig(EncFSMPStrings::configAppName_,
EncFSMPStrings::configOrganizationName_));
    config->SetPath(EncFSMPStrings::configMountListPath_);
    bool isOK = config->GetFirstGroup(str, index);
    while(isOK)
    {
        config->SetPath(str);
        MountEntry cur;

        if(!config->Read(EncFSMPStrings::configNameKey_, &cur.name_))
            return false;
        if(!config->Read(EncFSMPStrings::configEncFSPathKey_, &cur.encFSPath_))
            return false;
        config->Read(EncFSMPStrings::configExternalConfigFileKey_,
&cur.externalConfigFileName_);
        config->Read(EncFSMPStrings::configDriveLetterKey_, &cur.driveLetter_);
        config->Read(EncFSMPStrings::configPasswordKey_, &cur.password_);
        config->Read(EncFSMPStrings::configUseExternalConfigFileKey_,
&cur.useExternalConfigFile_, false);
        config->Read(EncFSMPStrings::configEnableCachingKey_, &cur.enableCaching_,
false);
        config->Read(EncFSMPStrings::configIsWorldWritableKey_,
&cur.isWorldWritable_);
        config->Read(EncFSMPStrings::configIsLocalDriveKey_, &cur.isLocalDrive_,
true);

        cur.assignedDriveLetter_ = wxEmptyString;
        cur.mountState_ = MountEntry::MSNotMounted;

        mountEntries_.push_back(cur);

        config->SetPath(wxT("../"));
        isOK = config->GetNextGroup(str, index);
    }

    return true;
}

```

Файл « MountList.h»

```

#ifndef MOUNTLIST_H
#define MOUNTLIST_H

#include <list>

/**
 * Data-only class for holding information about a mount.
 */
class MountEntry
{
public:

    enum MountState
    {
        MSNotMounted = 0, MSPending, MSMounted
    };

    MountEntry()

```

					ПГУ 2.090301.116.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		129

```

        : isWorldWritable_(false), isLocalDrive_(true),
        mountState_(MSNotMounted)
    { }
    MountEntry(const MountEntry &o) { copy(o); }
    virtual ~MountEntry() { }
    MountEntry &copy(const MountEntry &o)
    {
        name_ = o.name_;
        encFSPath_ = o.encFSPath_;
        externalConfigFileName_ = o.externalConfigFileName_;
        driveLetter_ = o.driveLetter_;
        assignedDriveLetter_ = o.assignedDriveLetter_;
        password_ = o.password_;
        assignedMountPoint_ = o.assignedMountPoint_;
        volatilePassword_ = o.volatilePassword_;
        useExternalConfigFile_ = o.useExternalConfigFile_;
        enableCaching_ = o.enableCaching_;
        isWorldWritable_ = o.isWorldWritable_;
        isLocalDrive_ = o.isLocalDrive_;
        mountState_ = o.mountState_;
        return *this;
    }
    MountEntry & operator=(const MountEntry & o)
    {
        return copy(o);
    }

    wxString name_, encFSPath_, externalConfigFileName_, driveLetter_,
    assignedDriveLetter_, password_;
    wxString assignedMountPoint_, volatilePassword_; // Not persistent attributes
    bool useExternalConfigFile_, enableCaching_, isWorldWritable_, isLocalDrive_;
    MountState mountState_;
};

class MountList
{
public:
    MountList();
    virtual ~MountList();

    bool addMount(wxString name, wxString encFSPath, wxString externalConfigFileName,
        wxString driveLetter, wxString password, bool useExternalConfigFile,
        bool enableCaching, bool isWorldWritable, bool isLocalDrive, bool
isMounted);

    std::list<MountEntry> &getList() { return mountEntries_; }

    MountEntry *findEntryByName(wxString name);

    bool storeToConfig();
    bool loadFromConfig();

private:
    std::list<MountEntry> mountEntries_;
};

#endif

```

Файл « EncFSMPStrings.cpp»

```
#include "CommonIncludes.h"
#include "EncFSMPStrings.h"

const std::wstring EncFSMPStrings::formatterName_(L"encFSMP");
const std::string EncFSMPStrings::formatterName8_("encFSMP");

const wxString EncFSMPStrings::configAppName_(wxT("EncFSMP"));
const wxString EncFSMPStrings::configOrganizationName_(wxT("hiesti.ch"));
const wxString EncFSMPStrings::configMountListPath_(wxT("/MountList"));

const wxString EncFSMPStrings::configNameKey_(wxT("Name"));
const wxString EncFSMPStrings::configEncFSPathKey_(wxT("EncFSPath"));
const wxString EncFSMPStrings::configExternalConfigFileKey_(wxT("ExternalConfigFile"));
const wxString EncFSMPStrings::configDriveLetterKey_(wxT("DriveLetter"));
const wxString EncFSMPStrings::configPasswordKey_(wxT("Password"));
const wxString
EncFSMPStrings::configUseExternalConfigFileKey_(wxT("UseExternalConfigFile"));
const wxString EncFSMPStrings::configEnableCachingKey_(wxT("EnableCaching"));
const wxString EncFSMPStrings::configIsWorldWritableKey_(wxT("IsWorldWritable"));
const wxString EncFSMPStrings::configIsLocalDriveKey_(wxT("IsLocalDrive"));
const wxString EncFSMPStrings::configWindowDimensions_(wxT("WindowDimensions"));
const wxString EncFSMPStrings::configColumnWidths_(wxT("ColumnWidths"));
const wxString EncFSMPStrings::configMinimizeToTray_(wxT("MinimizeToTray"));
const wxString
EncFSMPStrings::configDisableUnmountDialogOnExit_(wxT("DisableUnmountDialogOnExit"));
const wxString EncFSMPStrings::configShowErrorLogOnErr_(wxT("ShowErrorLogOnErr"));
const wxString EncFSMPStrings::configSavePasswordsInRAM_(wxT("SavePasswordsInRAM"));

const wxString EncFSMPStrings::commandMount_(wxT("mount"));
const wxString EncFSMPStrings::commandUnmount_(wxT("unmount"));
const wxString EncFSMPStrings::commandDDETopic_(wxT("EncFSMP_Command"));
const wxString EncFSMPStrings::commandDDEServerName_(wxT("EncFSMP_DDEServer"));
```

Система хранения облачных данных

Автор работы: студент группы 15BB1 Соустин К.Р

Научный руководитель: Дорошенко Иван Николаевич

Цели и задачи работы

Цель работы: Разработать систему хранения зашифрованных данных в облаке. Разработать и отладить взаимодействие модулей приложения. Система должна быть представлена в виде установочного файла программы.

Достижение поставленной цели требует решения следующих задач:

- 1) формирование требований к проектируемой системе и определение её функциональных возможностей;
- 2) изучить способы обеспечения безопасности данных в облаке;
- 3) изучить методы шифрования данных;
- 4) разработка необходимых программных модулей, для работы с облачными данными.

Облачные хранилища



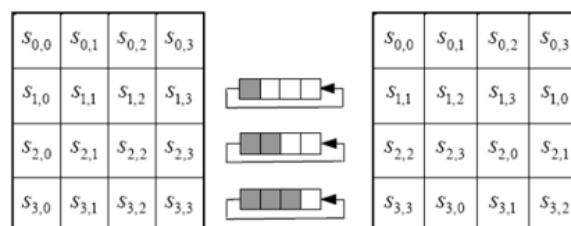
Алгоритм шифрования

Таблица подстановок

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	e0	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	ce	55	28	df	
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

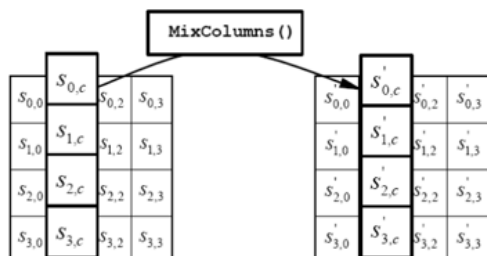
$$\begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Преобразование ShiftRows

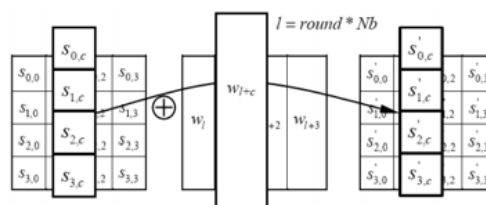


Алгоритм шифрования

Преобразование MixColumns



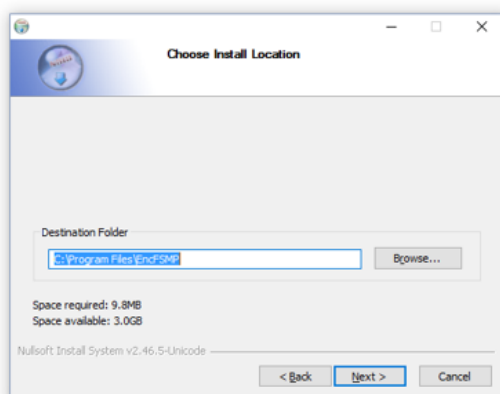
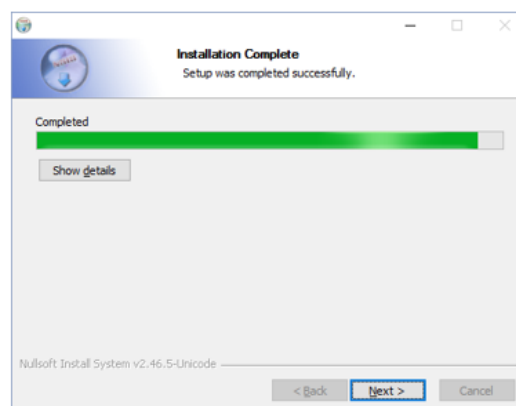
Преобразование AddRoundKey



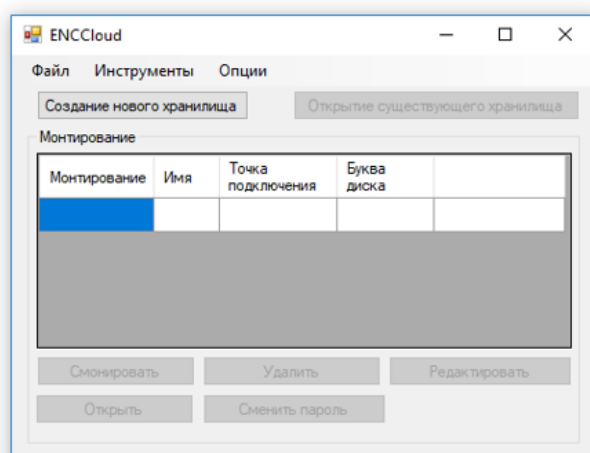
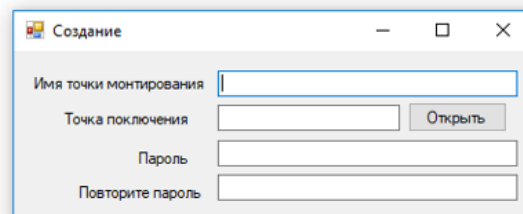
Умножение производится в поле Галуа

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Установка приложения



Работа с приложением



Результаты работы

- Разработан механизм подключения к облачным хранилищам
- Реализован механизм шифрования AES
- В процессе работы проводился анализ, сравнение существующих программ шифрования

В результате выполнения работы была создана система хранения облачных данных.

В дальнейшем планируется добавлять алгоритмы шифрования данных пользователя, а также улучшать пользовательский интерфейс, для повышения безопасности и оптимизации скорости работы программы.

Спасибо за внимание

		.			<i>ПГУ 2.090301.116.001 ПЗ</i>	<i>Лист</i>
						136
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		