УДК 004.9

АНАЛИЗ КРИПТОГРАФИЧЕСКИХ СРЕДСТВ В WINDOWS 7

Первусяк А.И., Иванов А.Ю.
Кафедра компьютерной инженерии,
Донецкий национальный технический университет, Украина
tyran-cool@mail.ru

Рассмотрены функции подсистемы защиты ОС. Исследованы модули, реализующие криптографическую функцию подсистемы защиты. Исследованы возможности системы шифрования EFS и BitLocker Windows 7. Исследован алгоритм шифрования DESX. Проведен сравнительный анализ способов запуска програмного модуля BitLocker Windows 7.

Общая постановка проблемы

При создании защищенных операционных систем используется два основных подхода – фрагментарный и комплексный. При фрагментарном подходе организуется поэтапная защита от последовательности угроз, при комплексном - защитные функции вносятся в операционную систему на этапе проектирования архитектуры операционной системы и являются ее неотъемлемой частью. В операционнай системе Windows 7 реализован комплексный подход. Отдельные элементы подсистемы защиты тесно взаимодействуют друг с другом и конфликты между компонентами практически невозможны. Одновременно с этим подсистема устроена таким образом, что при фатальных сбоях в функционировании ключевых элементов она вызывает крах операционной системы. Это не позволяет злоумышленнику отключать защитные функции системы. Подсистемой допускается замена отдельных ее элементов.

Подсистема защиты ОС Windows 7 выполняет следующие основные функции: идентификация и аутентификация, разграничение доступа, аудит, управление политикой безопасности, криптографические функции и сетевые функции. Каждая из перечисленных функций подсистемы защиты решается одним или несколькими программными модулями (для криптографической – BitLocker Windows 7). Некоторые функции встраиваются непосредственно в ядро ОС (для криптографической – система шифрования файлов EFS). Установлен определенный интерфейс, используемый для взаимодействия модулей при решении задач [1].

Встроенный непосредственно в ядро ОС модуль, реализующий криптографическую функцию подсистемы защиты ОС Windows 7

Функции системы шифрования файлов EFS (Encrypting File System) обеспечивают гибкость для корпоративных пользователей при шифровании файлов с данными. В основе технологии шифрования EFS используют архитектуру Windows CryptoAPI. При этом используется шифрование с открытым ключом. Для шифрования каждого файла случайным образом генерируется ключ шифрования файла. Для шифрования файла может применяться любой симметричный алгоритм шифрования. В настоящее же время в EFS используется один алгоритм (DESX), являющийся модификацией стандарта DES. Ключи шифрования EFS хранятся в резидентном пуле памяти (сама EFS расположена в ядре Windows 7), что исключает несанкционированный доступ к ним через файл подкачки [3].

Crypto API ориентирован на решение задач:

- надежное сокрытие данных;
- возможность передачи сокрытых данных третьим лицам;
- надежная система проверки достоверности пришедшей от третьих лиц информации;
- расшифровывание полученных конфиденциальных данных;

– работа с «идентификационными удостоверениями» третьих лиц;
– обеспечение работы с признанными криптографическими стандартами;
– возможность расширения и работы с пока еще неизвестными алгоритмами.

В Crypto API для решения задачи расширения реализация всех алгоритмов (шифрования, цифровой подписи и т.п.) полностью выведена из состава самого Crypto API и реализуется в отдельных, независимых динамических библиотеках – «криптопровайдерах» (Cryptographic Service Provider – CSP). Сам же Crypto API предъявляет определенные требования к набору функций криптопровайдера, предоставляя конечному пользователю унифицированный интерфейс работы с CSP. Конечному пользователю для полноценного использования всех функций криптопровайдера достаточно знать его строковое имя и номер типа.

Общепризнанным решением в однозначной идентификации передающей/принимающей стороны в протоколе передачи данных является использование механизма сертификатов.

Crypto API, как основная библиотека для обеспечения работы с криптографическими данными поддерживает данные стандарты и позволяет формировать криптографические приложения, которые могут быть обработаны в дальнейшем любыми программными продуктами.[5]

Суть алгоритма DESX состоит в том, что перед выполнением однократного DES и после него операцией XOR накладываются различные 64-битные фрагменты ключа на данные:

$$C = k_3 \text{ XOR } DES_{k1} (k_2 \text{ XOR } M).$$

Операция наложения фрагмента ключа на вход и/или выход алгоритма называется отбеливанием. Однако в алгоритме DESX найдено несколько уязвимостей:

– каждый ключ DESX имеет эквивалентный ключ, подключи которого комплементарны подключам исходного ключа;
– существует атака, позволяющая раскрыть ключ алгоритма; атака основана на связанных ключах и при наличии 2^n выбранных открытых текстов требует выполнения 2^{120-n} шифрований;
– дифференциальный криптоанализ позволяет вскрыть алгоритм при наличии 2^{61} выбранных открытых текстов; линейный криптоанализ успешен при наличии 2^{60} известных открытых текстов.

Эти проблемы не проявляются только в варианте DESX, использующем алгоритм хэширования SHA-1 для расширения ключа алгоритма [2].

Процесс шифрования файла при помощи EFS можно разделить на следующие этапы.

1. Загрузка профиля. Если профиль пользователя еще не загружен (например, шифрование или дешифрование файла происходит при помощи программы командной строки runas.exe), тогда профиль пользователя загружается.

2. Создание файла журнала. На этом этапе создается файл журнала шифрования с именем формата efsX.log, где X определяет номер файла, который был зашифрован в текущий сеанс шифрования процессом lsasrv.exe. Данный файл создается в каталоге System Volume Information.

3. Генерация FEK. Выполняется генерация случайного 128-битного числа, используемого в качестве FEK. FEK используется операционной системой для шифрования содержимого файла по определенному алгоритму, после чего значение FEK добавляется к самому зашифрованному файлу. Однако перед тем как добавить FEK к файлу, операционная система выполняет шифрование FEK при помощи алгоритма DESX, используя при этом открытый пользовательский ключ. После выполнения всех этих операций файл считается зашифрованным. Чтобы просмотреть файл, нужно расшифровать его при помощи FEK, а FEK можно расшифровать только при помощи ключей, которые можно получить только после загрузки учетной записи пользователя, который зашифровал файл.

4. Получение открытого и закрытого пользовательского ключа. Если прежде пользователь никогда не выполнял шифрование файлов, тогда генерируется пара пользовательских ключей (закрытый и открытый пользовательские ключи). В противном случае открытый пользовательский ключ берется из параметра «REG_BINARY» типа «CertificateHash», расположенного в ветви реестра «HKCU\Software\Microsoft\Windows NT\CurrentVersion\EFS\CurrentKeys». На основе данных этой

ветви реестра создается сигнатура открытого ключа пользователя.

Закрытые ключи находятся в каталоге «%userprofile%\AppData\Roaming\ Microsoft\Crypto\ RSA. Содержимое этого каталога зашифровано на основе симметричного ключа, который называется мастер-ключом пользователя.

Мастер ключ содержится в каталоге «%userprofile%\AppData\Roaming\Microsoft\ Protect». Он также зашифрован — с помощью алгоритма DESX и пароля пользователя.

5. Создание DDF. Для шифруемого файла создается связка ключей DDF (совокупность нескольких DDF, определяющих пользователей, которые могут расшифровать данный файл). DDF содержит информацию о пользователе, который может открыть данный файл, а также о его правах доступа к файлу. В эту информацию входит SID-пользователя, имя контейнера, имя провайдера, зашифровавшего файл, хеш сертификата EFS, используемый при расшифровке, а также зашифрованный FEK.

6. Создание DRF. Для шифруемого файла создается связка ключей DRF. DRF содержит информацию обо всех агентах восстановления, которые могут открыть данный зашифрованный файл. Информация, помещаемая в DRF, аналогична информации, помещаемой в DDF. Агенты восстановления представляют собой учетные записи пользователей, которые могут открыть зашифрованный файл, даже если они не входят в ту группу пользователей, которым разрешен доступ к файлу. Как правило, агентом восстановления является администратор.

7. Шифрование. Создается резервная копия шифруемого файла с именем efs0.tmp. В дальнейшем шифрование будет применяться именно к резервному файлу, после чего его содержимое будет скопировано в исходный файл. И в самом конце произойдет удаление резервного файла, а также файла журнала шифрования [3].

Программный модуль подсистемы защиты Windows 7, реализующий криптографическую функцию *BitLocker Windows 7*

Когда операционная система находится в активном состоянии, ее можно защитить при помощи локальных политик безопасности, антивирусного программного обеспечения и брандмауэров с межсетевыми экранами, а вот защитить том операционной системы на жестком диске вы можете средствами шифрования BitLocker. Для того чтобы воспользоваться всеми преимуществами шифрования BitLocker и проверки подлинности системы, компьютер должен позволять сохранять определенный ключ на съемном носителе для запуска системы. Помимо модуля TPM, в базовой системе ввода-вывода (BIOS) должна быть установлена спецификация группы Trusted Computing Group (TCG), которая перед загрузкой операционной системы создает цепочку доверий для действий и включает поддержку статического корневого объекта изменения уровня доверия. Не все материнские платы оснащены таким модулем как TPM, но даже без этого модуля операционная система позволяет воспользоваться данной технологией шифрования при наличии запоминающих устройств USB с поддержкой команд UFI, а также в том случае, если жесткий диск разбит на два и более тома. Все тома должны быть отформатированы в файловой системе NTFS.

Архитектура шифрования BitLocker обеспечивает управляемые и функциональные механизмы, как в режиме ядра, так и в пользовательском режиме. На высоком уровне, к основным компонентам BitLocker можно отнести:

- Драйвер Trusted Platform Module (%SystemRoot%System32DriversTpm.sys) – драйвер, который обращается к чипу TPM в режиме ядра;
- Основные службы TPM, которые включают пользовательские службы, предоставляющие доступ к TPM в пользовательском режиме (%SystemRoot%System32tbssvc.dll), поставщика WMI, а также оснастку MMC (%SystemRoot%System32Tpm.msc);
- Связанный код BitLocker в диспетчере загрузки (BootMgr), который аутентифицирует доступ к жесткому диску, а также позволяет восстанавливать и разблокировать загрузчик;
- Драйвер фильтра BitLocker (%SystemRoot%System32DriversFvevol.sys), который позволяет шифровать и расшифровывать тома на лету в режиме ядра;

– Поставщик WMI BitLocker и управление сценариями, которые позволяют настраивать и управлять сценариями интерфейса BitLocker.

Шифрование BitLocker поддерживает пять режимов проверки подлинности в зависимости от аппаратных возможностей компьютера и требуемого уровня безопасности. Если аппаратная конфигурация поддерживает технологию доверенного платформенного модуля (TPM), то можно сохранять VMK как в TMP, так и в TPM и на устройстве USB и при загрузке системы вводить PIN. А для платформ, которые не совместимы с технологией TPM, можно хранить ключ на внешнем USB устройстве [4].

При загрузке операционной системы с включенным шифрованием BitLocker, выполняется последовательность действий, которая зависит от набора средств защиты тома. Эти действия включают в себя проверку целостности системы и другие шаги по проверке подлинности перед снятием блокировки с защищённого тома. В приведенной таблице (таблица 1) приведены способы, которые можно использовать для шифрования тома:

Следует также отметить, что криптографические функции подсистемы защиты в Windows 7 поддерживают резервное копирование и восстановление зашифрованных файлов без их расшифровки.

Таблица 1. Сравнительная характеристика способов запуска BitLocker Windows 7

Источник	Безопасность	Действия пользователя
Только TPM	Защищает от программных атак, но уязвим к аппаратным атакам	Никаких
TPM + PIN	Добавляет защиту от аппаратных атак	Пользователь должен вводить PIN-код при каждом запуске ОС
TPM + ключ USB	Полная защита от аппаратных атак, но уязвима к потере ключа USB	Пользователь должен использовать ключ USB при каждом запуске ОС
TPM + ключ USB + PIN	Максимальный уровень защиты	При каждом запуске ОС пользователь должен вводить PIN-код и использовать ключ USB
Только ключ USB	Минимальный уровень защиты для компьютеров, не оснащенных TPM + есть риск потери ключа	Пользователь должен использовать ключ USB при каждом запуске ОС

Экспериментальная часть

В экспериментальной части было проведено исследование системы шифрования EFS. Выполнялось шифрование/расшифрование различных типов файлов (doc, xls, exe, txt, mpx, bmp,avi и т.п.). Результаты отбражены в таблице 2.

Таблица 2. Характеристики шифрования/расшифрования различных типов файлов.

Тип файла	Размер	Шифрование		Расшифрование	
		Время	Коэффициент сжатия	Время	Коэффициент сжатия
*.doc	934 Кб	760 мс	0,998	420 мс	0,998
*.xls	34 Кб	600 мс	0,945	260 мс	0,945
*.exe	39 404 Кб	2340 мс	1	2290 мс	1
*.txt	1 Кб	200 мс	0,019	220 мс	0,019
*.mp3	3 738 Кб	500 мс	0,9993	460 мс	0,9993
*.bmp	2 917 Кб	550 мс	0,9986	550 мс	0,9986
*.wmv	25 631 Кб	1740 мс	1	1460 мс	1

Основываясь на полученных результатах, пришли к выводам, что в результате шифрования файлов их сжатие не производится, и процесс расшифрования в среднем на 10-20% быстрее. Это объясняется структурой алгоритма шифрования и расшифрования.

Выводы

Проведенный анализ показал, что криптографические функции подсистемы защиты в Windows 7 надежно выполняют функции защиты и предусматривают многочисленные меры предосторожности для обеспечения безопасности восстановления данных, а также защиту от утечки и потери данных в случае фатальных сбоев системы.

Литература

[1] Шаньгин В. Ф. Защита компьютерной информации. Эффективные методы и средства. – М.: ДМК Пресс, 2008. – 544 с. : ил.

[2] Панасенко С. П. Алгоритмы шифрования. Специальный справочник. – СПб. : БХВ-Петербург, 2009. – 576с. : ил.

[3] Станислав Коротыгин. Шифрующая файловая система (EFS). Электронный ресурс. Режим доступа: http://www.infocity.kiev.ua.

[4] Дмитрий Буланов. Windows 7 BitLocker или шифрование данных. Часть 1. Электронный ресурс. Режим доступа: http://dimanb.wordpress.com.

[5] Юрий Николаев. Использование Crypto API. Электронный ресурс. Режим доступа: http://www.insidepro.com/rus/index.shtml.