Приведенное выше убеждает нас, что при условии, если $Q$ или $G$ объявить конфиденциальными параметрами, то можно обеспечить сложность криптоанализа, превышающую сложность грубого перебора для симметричного алгоритма.

## Выводы

Практическая реализация такой схемы может быть следующая. Очевидно, что обеспечить конфиденциальность базовой точки $G$, как общесетевого параметра, сложно. Хотя есть случай рассылки $G$ по защищенному каналу с записью в качестве ключа. На практике вполне реализуем случай, когда $Q$ хранится как конфиденциальный ключ. Ключи и параметры хранятся и пересылаются в защищенном виде, реальная стойкость такой системы может быть оценена по $I_1$ и может совпадать со стойкостью симметричных систем. Естественно, что для рассмотренного случая необходимо построить соответствующие протоколы управления параметрами $d$, $Q$, при которых они останутся конфиденциальными, аутентичными и целостными.

*Литература:1. Бондаренко М. Ф., Горбенко И. Д., Качко Е. Г. Свинарев А. В., Гриненко Т. А. Сущность и результаты исследований свойств перспективных стандартов цифровой подписи X9.62-1998 и распределение ключей X9/63-199X на эллиптических кривых //Радиотехника 2000.- № 114- С. 15-24. 2. Горбенко И. Д., Збитнев С. И. Расширенное поле Галуа GF(2M). Вычислительная сложность простейших операций над расширенным полем GF(2M) // Радиотехника 2000.- 114- С. 80-89. 3. ANSI X9.63-199x: Elliptic curve key agreement and transport protocols, draft. 4. R. Gallent, R. Lambert And S. Vatsone Improving the paralelized Pollard lambda search on binary anomalous curves, to appear in Mathematics of Computation.*

**УДК 681.3.06**

# ТЕХНОЛОГИЯ БЕЗОПАСНОЙ ПАМЯТИ В СИСТЕМАХ ЗАЩИТЫ ИНФОРМАЦИИ

***Даниил Меалковский, Юрий Горбенко, Виталий Вервейко, Сергей Полчанинов***
*Харьковский государственный технический университет радиоэлектроники*

*Аннотация:* **Рассматривается задача построения и реализации модели безопасной памяти, создания программных компонентов работы с безопасной памятью в системах защиты информации.**

*Summary:* **This article describes a safe memory model construction and realization, creation a software component for working with safe memory in information security system.**

*Ключевые слова:* **Системы защиты информации, безопасная память, объект безопасности.**

## Введение

Основной особенностью современных операционных систем является их объектная ориентированность. Преимущества использования системой объектов (objects) для регулирования доступа к системным ресурсам можно объяснить двумя основными причинами. Во-первых, использование объектов позволяет разработчику обновлять систему функционально, так как определенный интерфейс поддерживается. Во-вторых, использование объектов позволяет использовать возможность защиты функционирования системы на уровне защиты функционирования отдельных объектов. Преимущество использования объектов проявляется также на этапе разработки, так как позволяет упростить разработку сложных многокомпонентных систем и информационных технологий.

Для применения технологии объектов и дескрипторов при реализации систем защиты информации необходимо создать сервисы, обеспечивающие защищенность объекта безопасности от внешних компонентов системы, например, реализации модели безопасной памяти (secure memory model или SMM) и менеджеров объектов безопасности (secure objects или SO) для безопасного функционирования объектов в системе. Построение и реализация модели SMM являлись целью данной работы. Созданные программные компоненты работы с SMM встраиваются в платформы Microsoft® Win32® и используются Cryptographic Service Provider (CSP).

## Дескрипторы и объекты

Объекты - это структуры данных, представляющие системный ресурс. Приложения не имеют прямого доступа к данным объекта или системному ресурсу, который представляет объект. Вместо этого приложения получают дескрипторы объектов, которые могут быть использованы для изменения системного ресурса.


*ISBN 966-622-022-9*
*Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, вип. 2, 2001 р.*

Каждый дескриптор имеет вход во внутреннюю таблицу объектов. Входы содержат адреса ресурсов о средствах, определяющих тип ресурса.

Созданием и управлением использования объектов занимается менеджер объектов (objects manager). Объекты имеют одинаковую структуру, поэтому один менеджер объектов поддерживает все объекты.

Заголовок объекта содержит имя объекта (так что остальные процессы могут ссылаться на объект по имени) и дескрипторы безопасности (security descriptors) (так что менеджер объектов может управлять доступом процессов к системному ресурсу).

Основными задачами менеджера объектов являются:

- создание объектов;
- проверка прав процесса на возможность использования объекта;
- создание дескрипторов объектов и передача их вызвавшему процессу;
- поддержка ссылок ресурса;
- создание копии объекта;
- закрытие дескрипторов объектов.

Microsoft® Win32® интерфейс программирования приложений (application programming interface или API) предоставляет функции, выполняющие следующие задачи:

- создание объектов;
- получение дескрипторов объектов;
- получение информации об объекте;
- установка информации об объекте;
- закрытие дескриптора объекта;
- уничтожение объекта.

Некоторые из этих задач не обязательны для каждого объекта, некоторые скомбинированы для определенных объектов. Одно приложение может создавать объект. Другие приложения могут открывать объект, используя дескриптор объекта. Но по завершению каждое приложение, использующее объект, закрывает дескриптор объекта. Когда не остается открытых дескрипторов объекта, система уничтожает объект. Приложение может получить дескриптор существующего объекта. Если объект больше не нужен, приложение должно уничтожить объект, что делает не действительным его дескрипторы.

Иногда объекты остаются в памяти после того, как все дескрипторы объектов закрыты. Возникновение таких ситуаций отслеживается системой, и после завершения всех операций над объектом система удаляет объект из памяти.

Дескрипторы и объекты расходуют память. Поэтому для сохранения ресурсов системы приложения должны закрывать дескрипторы и уничтожать объекты по мере того, как они становятся не нужными. Если не делать этого, то значительный расход системных ресурсов приведет к неэффективной работе системы.

По завершению процесса операционные системы Microsoft® Win32® автоматически закрывают дескрипторы и удаляют объекты, созданные процессом. Однако при завершении потока система обычно не закрывает дескрипторы и не удаляет объекты. Использование внутренних объектов позволяет закрывать дескрипторы и удалять объекты перед завершением процесса.

Объектная технология позволяет динамической библиотеке CSP передавать вызывающему процессу дескрипторы создаваемых объектов или компонент. Это позволяет хранить экземпляры созданных объектов во внутренней памяти. Ресурсы существующих объектов не доступны вызывающим процессам напрямую. Все операции с объектом доступны только через механизм дескрипторов объектов.

В технологии менеджера объектов безопасности отсутствуют средства сериализации объектов. Объект безопасности существует только в памяти и не сохраняется на жесткий диск. Область памяти, занимаемая объектом безопасности, исключается из механизма сохранения страниц памяти в страничный файл подкачки, используемый в операционных системах Microsoft® Win32®.

## Виртуальное адресное пространство в Win32®

Каждый процесс имеет собственное 32-битное виртуальное адресное пространство. Это позволяет адресовать до 4Гбайт памяти.

Виртуальные адреса, используемые процессами, не отображают действительного расположения объекта в памяти. Вместо этого система поддерживает карту страниц для каждого процесса, представляющую собой внутреннюю структуру данных, используемую для преобразования виртуальных адресов в физические.

Виртуальное адресное пространство каждого процесса значительно больше общей физической памяти, доступной всем процессам. Для увеличения объема физической памяти система использует диск для дополнительного хранения. Общее количество памяти, доступной всем исполняемым процессам, состоит из

физической памяти и свободного пространства на диске, доступного для страничного файла. Страничный файл используется системой для увеличения объема пространства физической памяти. Физическая память и виртуальное адресное пространство организуются в страницы. Размер страниц зависит от процессора. Для процессоров Intel® x86 минимальный размер страницы 4 Кбайта.

Для увеличения гибкости в управлении памятью система может сбрасывать страницы из физической памяти в страничный файл на диске и восстанавливать страницы в физическую память с диска. Когда страница перемещается в физическую память, система обновляет карту страниц затронутых процессов. Когда системе требуется физическая память, она перемещает наиболее давно использованные страницы в страничный файл. Обращения системы к физической памяти полностью прозрачны для приложений, использующих свое виртуальное адресное пространство.

## Модель безопасной памяти (SMM) CSP

Основными компонентами модели безопасной памяти (SMM) являются:

- объекты и дескрипторы безопасности;
- безопасная область памяти (SM);
- элементы и сервисы управления компонентами SM и объектами безопасности.

К объектам безопасности SMM CSP относятся:

- объекты криптопровайдер;
- объекты ключей;
- хеш-объекты.

Объекты безопасности – это структуры данных, представляющие криптографический компонент. Физически каждый объект является внутренней структурой данных, используемой CSP для представления криптографических и системных компонентов. Криптографические компоненты являются основой построения системных компонентов, использующих криптографические функции и сервисы.

Элементы объектов безопасности содержат криптографические ключи или другие криптографические данные. Криптографические ключи являются центральным компонентом криптографических операций. Они должны держаться в секрете, потому что любой процесс, получивший ключ, имеет доступ к любым данным, связанным с этим ключом.

Существует два основных типа ключей, используемых в CSP: сеансовые ключи и ключи обмена. Каждый CSP имеет базу данных ключей обмена, в которой он сохраняет криптографические ключи. Каждая база данных содержит один и более контейнеров ключей, содержащих все ключевые пары пользователя.

CSP использует ключевые контейнеры с содержащимися в них открытыми и секретными ключами от сеанса к сеансу. Сеансовые ключи изменяются и используются один раз. Поэтому для повторного использования в обратных операциях сеансовые ключи должны быть экспортированы из безопасного окружения CSP в пространство данных приложения. Экспортируемые и сохраняемые сеансовые ключи преобразуются в зашифрованные ключевые блобы. Для обеспечения большей безопасности сеансовые ключи зашифровываются и подписываются на ключе обмена перед сохранением.

Сеансовые ключи и ключи обмена недоступны в открытом виде вне CSP. Ключи обмена хранятся в виде контейнеров с ключевыми парами или в виде сертификатов ключей. Сеансовые ключи экспортируются из CSP в виде зашифрованных блобов. Однако ключи хранятся в открытом виде во внутренней области памяти CSP. Безопасность функционирования объектов ключей и других криптографических объектов во внутренней памяти CSP определяется безопасностью области памяти, которую занимают объекты. Безопасность внутренней области памяти CSP от внешних компонентов системы представляет основной элемент безопасного окружения CSP.

Объекты криптопровайдеров состоят из криптографических и системных элементов. Эти объекты содержат информацию о криптопровайдере и его состоянии в системе. Объекты ключей содержат ключевые данные и информацию о состоянии используемого ключа. Хеш-объекты содержат временные данные и информацию о состоянии криптографической операции.

Приложения не имеют прямого доступа к данным объекта безопасности. Вместо этого приложения получают дескрипторы объектов, которые могут быть использованы для обращения к данным объекта безопасности. Каждый дескриптор имеет вход во внутреннюю таблицу объектов. Входы содержат адреса структур данных объектов безопасности.

Созданием и управлением использования объектов безопасности занимается менеджер объектов безопасности (secure objects manager). Отдельные фиксированные типы объектов имеют одинаковую структуру, поэтому один менеджер объектов поддерживает все объекты.

Основными задачами менеджера объектов безопасности являются:

- Создание объектов безопасности
- Создание дескрипторов объектов безопасности и передача их вызвавшему процессу
- Поддержка ссылок ресурса
- Создание копии объекта безопасности
- Закрытие дескрипторов объектов безопасности

CSP использует менеджеры, выполняющие следующие задачи:

- Создание объектов безопасности
- Получение дескрипторов объектов безопасности
- Получение информации об объекте безопасности
- Установка информации об объекте безопасности
- Закрытие дескриптора объекта безопасности
- Уничтожение объекта безопасности

Объектная технология дает возможность динамической библиотеке CSP передавать вызывающему процессу дескрипторы безопасности создаваемых объектов криптопровайдеров, ключей или хеш-объектов. Это позволяет хранить экземпляры созданных объектов безопасности в безопасной памяти. Ресурсы существующих объектов не доступны вызывающим процессам напрямую. Все операции с объектом безопасности доступны только через механизм дескрипторов безопасности. Это позволяет ограничить прямой доступ к объектам безопасности процессов и других компонентов системы.

Основой технологии менеджера объектов безопасности является SMM и набор основных сервисов управления объектами и дескрипторами. Сервисы управления объектами и дескрипторами включают в себя следующие функции:

- Создание объектов и дескрипторов безопасности,
- Блокировка объектов безопасности,
- Разблокировка объектов безопасности,
- Проверка доступности объектов безопасности,
- Удаление объектов и дескрипторов безопасности.

Объекты безопасности создаются функцией диспетчера объектов безопасности. Функция поддерживает создание объектов криптопровайдера, объектов ключей и хеш-объектов. При создании объекта криптопровайдера выделяется память под объект криптопровайдера. Ссылка объекта записывается в глобальную таблицу дескрипторов существующих объектов криптопровайдеров. Дескриптор ссылки созданного объекта криптопровайдера возвращается вызывающей функции. При создании объектов ключей и хеш-объектов ссылки объектов записываются в локальную таблицу дескрипторов указанного криптопровайдера. Дескриптор ссылки созданного объекта безопасности возвращается вызывающей функции. Создание объектов безопасности не инициализирует элементов создаваемых объектов. При создании устанавливается только начальное состояние объекта.

Объекты безопасности блокируются набором сервисных функций. Блокировка объектов безопасности не дает возможности использовать объект одновременно двумя потоками. Заблокированный объект находится в использовании с установленным состоянием занятости. Состояние объекта безопасности проверяется набором сервисных функций. Проверка объектов безопасности включает проверку состояния и элементов объекта. Объекты безопасности разблокируются набором сервисных функций. Разблокировка объектов безопасности снимает состояние занятости объектов. Разблокированный объект может использоваться другим потоком.

Объекты безопасности удаляются функцией диспетчера объектов безопасности. Функция поддерживает удаление объектов криптопровайдера, объектов ключей и хеш-объектов. При удалении объекта криптопровайдера освобождается память объекта криптопровайдера. Ссылка объекта удаляется из глобальной таблицы дескрипторов существующих объектов криптопровайдеров. Дескриптор ссылки удаляемого объекта криптопровайдера получается из вызывающей функции. При удалении объектов ключей и хеш-объектов ссылки объектов удаляются из локальной таблицы дескрипторов указанного криптопровайдера. Дескриптор ссылки удаляемого объекта безопасности получается из вызывающей функции.

Область памяти глобальной таблицы дескрипторов выделяется динамически при первом подключении DLL криптопровайдера к процессу. Область памяти локальных таблиц дескрипторов выделяется в объекте криптопровайдера.

## Драйверы блокировки страниц

Создание объектов безопасности осуществляется в безопасной памяти. Область памяти, выделяемая под

объект безопасности, исключается из механизма сохранения страниц памяти в страничный файл подкачки, используемый в Win32®. Блокировка выделенных страниц области защищенной памяти осуществляется с использованием драйвера ядра блокировки страниц.

Функциями управления виртуальной памятью (менеджером виртуальной памяти) выделяется область памяти объекта безопасности в виртуальном адресном пространстве текущего процесса. Адрес в виртуальном адресном пространстве и размер выделенной области памяти используется драйвером для блокировки диапазона виртуальных адресов, выровненных на границу страницы памяти.

Информация о заблокированных страницах виртуального адресного пространства процессов динамически обновляется во внутренних контейнерах объектов памяти. Блокировка страниц добавляет информацию о заблокированных страницах памяти в элемент контейнера объектов памяти. Выделение дополнительных контейнеров выполняется в процессе добавления информации о заблокированных страницах. Контейнеры объектов памяти с элементами информации о заблокированных страницах связываются в список. Поэтому информация о заблокированной области памяти объекта может содержаться в нескольких элементах контейнеров объектов памяти.

Блокируемый диапазон страниц связывается с их виртуальным адресом, числом страниц в диапазоне, дескриптором процесса и дескрипторами объекта драйвера. Информация о заблокированном диапазоне страниц используется при разблокировке области памяти.

Удаление объектов безопасности из безопасной памяти выполняется сервисной функцией менеджера. Разблокировка страниц освобождаемой области памяти, занимаемой объектом безопасности, осуществляется с использованием драйвера блокировки страниц. Функциями управления виртуальной памятью освобождается область памяти объекта безопасности в виртуальном адресном пространстве текущего процесса.

Адрес в виртуальном адресном пространстве и размер выделенной области памяти используется драйвером для разблокировки диапазона виртуальных адресов, выровненных на границу страницы памяти. Информация о диапазоне страниц и вызывающем процессе сопоставляется с информацией в элементах контейнеров объектов памяти. Последовательным поиском в связанном списке определяется информация для сервисов разблокировки страниц. После разблокировки диапазона страниц информация элемента контейнера удаляется из связанного списка элементов контейнеров. Динамически обновляются и контейнеры объектов памяти.

Создание начального контейнера и инициализация внутренних элементов выполняется при загрузке драйвера. Освобождение занимаемых ресурсов системы драйвер выполняет при отключении. Все не разблокированные страницы и не удаленные объекты памяти удаляются из списка контейнеров и элементов после разблокировки диапазона страниц.

Обращение к сервисам блокировки страниц, создание объектов памяти новых контейнеров и элементов выполняется в блокируемом коде (в критических секциях).

Механизмы блокировки страниц и сервисы управления контейнерами объектов памяти отличаются в платформах Win32® и WinNT®. Отличия проявляются в построении драйверов, связанных с архитектурой ядра системы и различиями сервисов менеджеров и диспетчеров управления памятью.

## Компонентная модель управления SM

Модель управления безопасной памятью включает следующие компоненты и сервисы:

- Драйверы блокировки страниц или памяти для разных платформ Win32®,
- Сервисы управления и использования драйверов,
- Сервисы блокировки страниц пользовательского режима,
- Сервисы управления безопасной памятью.

Сервисы управления безопасной памяти включают функции выделения и освобождения безопасной памяти. Выделение памяти выполняется функцией выделения виртуальной памяти. Выделенная память блокируется с помощью сервисов блокировки страниц или памяти. При освобождении памяти она разблокируется и освобождается функцией освобождения виртуальной памяти. Сервисы блокировки страниц или памяти включают функции блокировки и разблокирования страниц или памяти. Загрузка и запуск драйверов перед использованием выполняется в этих функциях. Эти функции направляют запросы драйверам. Останов и закрытие драйверов после использования также выполняется в этих функциях.

## Заключение

Разработанная модель безопасной памяти (SMM), а также реализованные механизмы управления объектами безопасности и блокировки страниц в операционных системах Win32® позволяют надежно и

эффективно управлять объектами безопасности. Компонентная технология модели позволяет интегрировать сервисы управления объектами безопасности в исходный код CSP с использованием драйверов (kernel-mode drivers) блокировки страниц виртуальной памяти для платформ Microsoft® Win32®.

Разработка применяется в пользовательских системах защиты информации и системах защиты сетевого трафика для обеспечения конфиденциальности и целостности передаваемой и хранимой информации, а также в пользовательских системах, обрабатывающих и управляющих критической информацией, с целью защиты информации от возможных каналов утечки, связанных с особенностью управления памятью ОС Microsoft® Win32®. Однако разработанная технология не учитывает другие каналы возможной утечки информации, поэтому используется с другими системными компонентами.

*Литература: 1. Microsoft Developer Network Library – January 2000. Platform SDK (Base Services). Microsoft Corporation. 2. Windows 2000 Driver Development Kit. Kernel-mode drivers. Microsoft Corporation. 3. Windows 98 Driver Development Kit. Virtual device drivers. Microsoft Corporation.*

**УДК 681.142.35**

# АЛГОРИТМЫ ШИФРОВАНИЯ ДЛЯ ПЕРЕДАЧИ ДАННЫХ В ОТКРЫТЫХ СЕТЯХ

***Сергей Макаренко, Андрей Брусникин***
*Национальный технический университет Украины "КПИ"*

*Анотація*: **Надано стислий перелік вимог до алгоритмів кодування, що використовуються у відкритих мережах.**

*Summary*: **The list of requirements is given below for the security algorithms implemented for use in open network.**

*Ключевые слова:* **Шифр, RSA, DES, IDEA, ГОСТ 28147-89.**

## I Введение

Бурное развитие электронной коммерции и все возрастающее число пользователей заставляют серьезно задуматься о проблемах безопасности открытых каналов передачи данных. Отсутствие руководящих органов привело к тому, что каждый член сети должен заботиться о своей безопасности сам.

Подключение к открытым (глобальным) сетям, таким как Интернет, существенно увеличивает эффективность работы и открывает множество новых возможностей. В то же время необходимо позаботится о создании системы защиты информационных ресурсов от желающих их использовать, модифицировать или просто уничтожить. Защита информации подразумевает поддержание целостности, доступности и, если необходимо, конфиденциальности информации и ресурсов, используемых для ввода, хранения, обработки и передачи данных. Для решения комплексной проблемы защиты необходимо сочетание законодательных, организационных и программно - технических мер [1].

Данная работа посвящена влиянию специфики открытых сетей на решение задач обеспечения безопасности передачи данных и создания библиотек на основе лучших алгоритмов. Показано, как особенности данных отразились на развитии современных алгоритмов их шифрования.

## II Общие положения [1–5]

**1**. Абсолютно стойкие алгоритмы уже давно существуют, но до сих пор не нашли широкого применения в открытых сетях.

Первым абсолютно стойким алгоритмом был алгоритм Вернама, а его теоретическое обоснование разработано Шенноном.

Максимально возможная неопределенность блока данных фиксированного размера достигается, когда все возможные значения этого блока равновероятны. В этом случае она равна размеру блока в битах. Таким образом, неопределенность ключа K не превышает его длины:

$$H(K) \le K.$$

Условие абсолютной стойкости для шифров, удовлетворяющих принципу Кирхгофа,

$$|K| \ge H(K) = H(E) \ge H(T) = |T|,$$

где E – шифр, а T – исходные данные.

Для того чтобы шифр, построенный по принципу Кирхгофа, был абсолютно стойким, необходимо, чтобы