Симметричное шифрование

Электронные ключи работают с аппаратными алгоритмами по следующей общей схеме: от защищенного приложения к ключу поступает блок данных (вопрос к ключу), и он при помощи симметричного аппаратного алгоритма преобразует (кодирует или декодирует) эту последовательность. Таким образом, получается ответ ключа, посылаемый защищенной программе.
Чаще всего разработчик системы защиты для какого-либо приложения ограничивается весьма небольшим количеством возможных вопросов-ответов, которые используются на протяжении недолгого периода времени. Это сильно облегчает задачу построения табличных эмуляторов, поскольку для отслеживания всех запросов и ответов требуется то самое недолгое время. Это одна из самых распространенных ошибок при разработке систем защиты.
Для эффективной борьбы с созданием табличных эмуляторов число различных запросов и ответов должно быть как можно большим, а время, за которое они будут все использованы, должно измеряться месяцами.
Алгоритмы типа GSII64, с длиной ключа 128 или 256 бит, а также AES, с длиной ключа 128 бит, реализованные в ключах Guardant, симметрично кодируют и декодируют информацию в самом электронном ключе. Это позволяет значительно расширить возможности электронных ключей Guardant и повысить защищенность программного обеспечения за счет того, что данные для кодирования могут динамически изменяться.

Симметричные алгоритмы семейства GSII64

Важная информация
Аппаратный алгоритм GSII64 есть только в ключах Guardant Sign/Time и их разновидностях. Он не реализован в ключах с загружаемым кодом (Guardant Code и его модификации).
GSII64 – блочный симметричный алгоритм, устойчивый к криптоанализу. Длина секретного ключа GSII64 может составлять 16 или 32 байта (128 или 256 бит). Алгоритм может преобразовать за один цикл блок данных длиной 8 байт, имеет возможность преобразования блоков данных длиной, кратной 8 байтам, а также имеет режимы для преобразования блоков данных произвольной длины.
Алгоритм является симметричным, его можно использовать как для кодирования данных, так и для декодирования.
К семейству GSII64 относятся еще два алгоритма: GSII64_ ENCRYPT, GSII64_DECRYPT. Эти алгоритмы по своим свойствам идентичны GSII64, за исключением того, что первый может выполнять только прямое преобразование (зашифрование), а второй – обратное (расшифрование).

Симметричные алгоритмы семейства AES

Алгоритм AES (Advanced Encryption Standard) – это блочный симметричный алгоритм шифрования, принятый в качестве стандарта шифрования США. Длина секретного ключа AES составляет 16 байт (128 бит). Минимальная длина блока данных, преобразуемых алгоритмом за один цикл, – 16 байт. У алгоритма имеются режимы, позволяющие шифровать блоки данных, кратные по длине 16 байтам, и блоки произвольной длины.
Симметричность алгоритма означает использование одного и того же секретного ключа шифрования, как для прямого преобразования, так и для обратного.
Подробное описание алгоритма AES можно найти на сайте NIST: http://csrc.nist.gov/archive/aes/index.html
К разновидностям AES, реализованным в ключах Guardant, относятся еще два алгоритма: AES128_ENCRYPT, AES128_DECRYPT. Эти алгоритмы по своим свойствам идентичны AES128, за исключением того, что первый может выполнять только прямое преобразование (зашифрование), а второй – обратное (расшифрование).
Кроме того, в Guardant API реализован также алгоритм AES с длиной ключа 256 бит. В отличие от аппаратного AES128, он реализован программно, то есть выполняется в оперативной памяти и на процессоре компьютера, а не электронного ключа.

Программный алгоритм AES256

Для аппаратных алгоритмов существуют и ограничения, связанные с тем, что скорость аппаратных преобразований, выполняемых электронным ключом, сравнительно невысока. Преобразование больших объемов данных – от десятков килобайт до сотен мегабайт (разного рода базы данных, текстовые и графические файлы и т. п.), повлечет очень существенные задержки в работе защищенного приложения: все же мощность процессоров современных компьютеров во много раз выше, чем процессора электронного ключа.
Это ограничение отчасти можно преодолеть, используя программно реализованные алгоритмы шифрования. Такие алгоритмы используют всю вычислительную мощность компьютера, а потому работают на порядки быстрее, чем алгоритмы электронного ключа. В настоящее время можно легко найти достаточно большое количество реализаций различных алгоритмов. В состав Guardant API включена реализация симметричного алгоритма AES с длиной ключа 256 бит. Этот стойкий алгоритм с успехом можно использовать для надежного кодирования больших объемов данных.
Аппаратное преобразование в этом случае можно эффективно применять для кодирования или генерации ключей шифрования, используемых программно реализованными алгоритмами.

Режимы работы алгоритмов AES и GSII64

Режим ECB

Режим «электронной кодовой книги». Это простейший режим работы блочного симметричного алгоритма. В режиме ECB каждый блок открытого текста, подаваемый на вход алгоритма, преобразуется с одним и тем же определителем в блок шифртекста. Поэтому преобразование 2-х одинаковых блоков даст идентичный результат.
Если длина блока данных превышает минимальную длину блока (16 байт для AES и 8 байт для GSII64), он должен быть разбит на блоки, равные минимальной длине блока. При необходимости, к последнему блоку нужно добавить недостающие байты. Сильно желательно, чтобы байты-заполнители не были постоянными. В качестве байтов-заполнителей можно использовать случайные числа. В этом случае последний закодированный блок требуется хранить полностью, вместе с зашифрованными байтами-заполнителями (а не отбрасывать эти байты). Иначе полезные байты данных из этого блока невозможно будет расшифровать.
Режим ECB подходит для шифрования небольших объемов данных, например, векторов инициализации, используемых в других режимах алгоритма или ключей шифрования других алгоритмов.

Режим CBC

Режим сцепления блоков по шифртексту. В режиме CBC, как и в ECB, каждый блок открытого текста преобразуется в блок шифртекста той же длины. Преобразование в режиме CBC для всех блоков осуществляется с одним и тем же ключом. Режим CBC чаще используется и лучше подходит для преобразования блоков данных, превышающих по длине минимальную длину блока.
Однако в отличие от ECB, преобразование двух одинаковых блоков открытого текста, находящихся в разных позициях исходного блока данных, не даст идентичного результата. Это осуществляется благодаря тому, что на каждом следующем шаге шифруется не сам блок, а его сумма по модулю 2 с предыдущим блоком шифртекста. Для получения первого зашифрованного блока используется сумма по модулю 2 первого зашифрованного блока и некоторого вектора инициализации IV. Значение IV должно быть сохранено для корректного обратного преобразования, но желательно, если оно будет защищено (например, зашифровано в режиме ECB).
Преобразование получается позиционно-зависимым, поскольку результат шифрования зависит не только от самого блока открытого текста, но и от предшествующего ему.
Обратное преобразование также производится поблочно.
Суммарная длина исходного набора данных должна быть кратна минимальной длине блока. В противном случае, к последнему блоку нужно добавить байты-заполнители, так же, как и в режиме ECB.
Режим CBC можно использовать для вычисления надежных контрольных сумм, аутентификации и проверки подлинности данных. В качестве такой контрольной суммы используется последний блок шифротекста. Этот блок зависит от всех предыдущих блоков, а также от вектора инициализации, и вычисляется на основе секретного ключа алгоритма. Он не дает информации об исходных данных, но идентифицирует их практически однозначно. Подделать этот блок так же трудно, как подобрать ключ алгоритма.

Режим CFB

Режим с кодированной обратной связью. Режим CFB позволяет преобразовывать блоки данных произвольного размера, не обязательно кратного минимальной длине блока. Это избавляет от необходимости дополнять исходные данные до целого количества 8 блоков. В этом режиме длина зашифрованной последовательности будет равна длине исходной.
В режиме CFB, подобно режиму CBC, происходит сцепление блоков исходных данных, поэтому каждый закодированный блок будет зависеть от всех предыдущих блоков исходных данных, поскольку для зашифрования каждого следующего блока исходных данных используется предыдущий блок шифртекста.
В этом режиме для преобразования используется вектор инициализации IV (см. режим CBC).
Для однонаправленных алгоритмов, выполняющих только зашифрование или только расшифрование этот режим недоступен, поскольку математически прямое и обратное преобразования в этом режиме эквивалентны.
Важная информация
Если при декодировании указан неверный вектор инициализации, все данные, кроме первого блока, все равно расшифруются правильно. Если это критично для приложения, предпочтительно использовать режим OFB.

Режим OFB

Режим с обратной связью по выходу. Этот режим имеет много общего с режимом CFB.
Главное отличие состоит в том, что для зашифрования следующего блока используется не предыдущий блок шифртекста, а результат преобразования вектора инициализации.
Преимущество данного режима заключается в том, что при передаче зашифрованных данных снижается зависимость от искажений предыдущих блоков. То есть при повреждении одного блока, остальные блоки при расшифровании не пострадают.
У этого преимущества есть и обратная сторона – режим OFB обеспечивает меньшую защиту от искажения данных, поскольку при изменении одного бита шифртекста в расшифрованных данных будет изменен тот же бит. Для проверки подлинности данных в таком случае нужно пользоваться надежной контрольной суммой.
В этом режиме, так же как и в 2-х предыдущих, для преобразования используется вектор инициализации IV (см. режимы CBC и CFB).
Для однонаправленных алгоритмов, выполняющих только зашифрование или только расшифрование этот режим недоступен, поскольку математически прямое и обратное преобразования в этом режиме эквивалентны.

Рекомендации по работе с вектором инициализации IV

Для корректного преобразования данных симметричными алгоритмами GSII64 и AES необходимо принимать во внимание что:

- Вектор инициализации IV должен иметь одинаковые значения перед началом зашифрования и перед началом расшифрования
- Необходимо сохранять значение вектора инициализации IV между обращениями к GrdCrypt (GrdCryptEx) или GrdTrans-form при продолжении зашифрования/расшифрования больших блоков (больше 248 байт для ECB и CBC и 255 байт для CFB и OFB для алгоритма GSII64)
- При шифровании данных типа разных записей БД или секторов диска, инициализировать IV этим номером записи/ сектора для того, чтобы каждая из них кодировалась всегда одинаково, а разные записи с одинаковыми значениями кодировались по-разному.