Приемы работы с аппаратными алгоритмами

Очень важно использовать аппаратные алгоритмы творчески. Ведь преобразовывать информацию можно целым рядом способов, преследуя при этом самые разные цели. От того, насколько оригинально и необычно сконструированная защита использует аппаратные алгоритмы и преобразуемые ими данные, напрямую зависит надежность защиты.

Несколько аппаратных алгоритмов в одном ключе

В одном ключе Guardant можно создавать дескрипторы аппаратных алгоритмов, число которых ограничивается только размером памяти ключа. В любой момент времени можно использовать любой из активированных дескрипторов. Для защиты одного и того же приложения можно использовать все созданные дескрипторы: например, какая-то часть данных кодируется одним алгоритмом, какая-то – другим и т. д.
Конструирование и запись в элетронный ключ дескрипторов алгоритмов происходит при помощи утилиты программирования **GrdUtil.exe**. Также возможна разработка собственных альтернативных утилит, выполняющих аналогичные функции, но программирующей ключи согласно технологии, принятой разработчиками для конкретной компании.
Возможность активации и деактивации аппаратных алгоритмов позволяет записывать их дескрипторы «впрок», для последующего использования в будущих версиях приложения. До выхода новой версии эти алгоритмы могут находиться в деактивированном состоянии, а при переходе на новую версию – активироваться.
Если компания производит несколько программных продуктов, можно и нужно каждый из них защищать с использованием своих уникальных дескрипторов аппаратных алгоритмов. Также возможен вариант, когда в каждом электронном ключе создаются свои уникальные дескрипторы – и тогда каждая копия приложения будет защищена уникальным методом. Этим исключается опасность создания универсальных эмуляторов для разных программных продуктов (или для разных версий одного продукта).

Комбинация аппаратных и программных алгоритмов

Этот метод рекомендуется использовать при необходимости работы с большим объемом данных, которые использует защищенное приложение. При этом аппаратный алгоритм должен декодировать или преобразовывать ключ программного алгоритма AES256, который будет использоваться для кодирования основного объема данных.
Смысл преобразования ключа перед его использованием – повысить уровень защищенности кодированных данных. В этом случае данные преобразуются не по паролю, вводимому пользователем, а по его видоизмененной форме, которая не хранится нигде.

Случайные числа и анализ вероятностной функции

На вероятностном принципе можно строить надежные схемы защиты. Например, создается алгоритм, который возвращает случайную последовательность размером 100 байт. Она разбивается на 50 чисел по 2 байта, рассчитывается математическое ожидание, среднеквадратичное отклонение и определяется, действительно ли вероятностная функция имеет равномерное распределение. Если это не так, значит, хакер пытается эмулировать защиту от копирования, т. к. ему сложно (да и негде) взять столько правильных случайных последовательностей с равномерным распределением. Если приложение периодически обращается к ключу, то статистика может накапливаться и уточняться при каждом обращении, что будет давать с каждым разом все более точную статистическую картину.
Т. о., чтобы взломать защиту, надо написать весьма громоздкую программу-надстройку, которую даже неясно как внедрять в приложение. Либо придется анализировать всю систему защиты в совокупности, на что может уйти очень много времени, если система сбора и анализа статистики имеет распределенный по приложению характер.