

КАК ЗАЩИТИТЬ ПРОШИВКУ И ДАННЫЕ ВО ВСТРАИВАЕМЫХ УСТРОЙСТВАХ

СКОТ ДЖЕЙМС (SCOTT JAMES), КРИСТОФ ТРЕМЛЕТ (CRISTOPHE TREMLET)
И МАЙКЛ ДЖЕКсон (MICHAEL JACKSON), Maxim Integrated Products

Чтобы быть уверенными в том, что конкретное встраиваемое устройство работает под управлением только авторизованного программного обеспечения или использует только авторизованные конфигурационные данные, необходимо иметь возможность убедиться в подлинности и целостности информации. С этой целью применяется криптографическая цифровая подпись, которая выполняет те же функции, что и заверяющая подпись или печать под текстом бумажного письма.

По мере все большего проникновения IoT-устройств в повседневную жизнь постоянно растет и число попыток их взлома для получения несанкционированного контроля над ними. Чтобы их пресечь, требуется принять специальные меры по защите устройств. В качестве примера такой угрозы рассмотрим модификацию злоумышленником прошивки или конфигурационных данных IoT-устройств. В общем случае можно считать, что на этапе производства подлинность и целостность кода и данных, используемых этими устройствами, обеспечивается по умолчанию. Однако в процессе эксплуатации устройства могут оказаться в руках хакера. Кроме того, периодически устройствам требуется обновление микропрограммного ПО (прошивки) или конфигурационных данных. В результате у злоумышленника имеется возможность повлиять на функционирование устройства или, что еще хуже, получить над ним полный контроль. Одна из атак такого рода называется «внедрение вредоносного кода» (malware injection). Как следует из названия, она заключается в интеграции вредоносного кода в источник обновления ПО. В случае успешного внедрения стороннего фрагмента в прошивку неавторизованная конфигурация становится причиной:

- извлечения конфиденциальных и секретных данных. Внедрение вредоносного кода в медицинское оборудование, например в носимое устройство для мониторинга состояния пациента, может заставить устройство передавать закрытую медицинскую информа-

цию посторонним лицам. В худшем случае вредоносное ПО может раскрыть ключ шифрования;

- некорректного функционирования устройства. Известный пример такого рода – вирус Stuxnet, который, внедрившись в программируемые логические контроллеры (ПЛК), вызвал вращение центрифуг с такими скоростями, что произошло разрушение оборудования;
- непредсказуемого поведения устройства, в т. ч. представляющего опасность для жизни людей.

АУТЕНТИФИКАЦИЯ И ПОДТВЕРЖДЕНИЕ ЦЕЛОСТНОСТИ ПРОШИВКИ

Чтобы быть уверенными в том, что конкретное встраиваемое устройство работает под управлением только авторизованного программного обеспечения или использует только авторизованные конфигурационные данные, необходимо убедиться в подлинности и в целостности информации, т. е. иметь гарантию, что данные получены из доверенного источника и в неизменном виде. С этой целью применяется криптографическая цифровая подпись, выполняющая те же функции, что и заверяющая подпись или печать внизу бумажного письма.

В этом случае прошивка или конфигурационные данные, загружаемые в устройство на этапе производства, а также все последующие обновления сопровождаются цифровой подписью. Она гарантирует подлинность информации в течение всего срока эксплуатации устройства. Для создания

устойчивой цифровой подписи используются криптографические алгоритмы. Чтобы обеспечить максимальный уровень безопасности, эти алгоритмы должны быть открытыми и апробированными. В этой статье мы рассмотрим асимметричные криптографические алгоритмы, а именно алгоритм цифровой подписи на эллиптических кривых FIPS 186 (ECSDA).

АСИММЕТРИЧНЫЕ КРИПТОГРАФИЧЕСКИЕ АЛГОРИТМЫ

В асимметричной криптографии (с открытым ключом) применяется пара ключей (открытый и закрытый), компоненты которой связаны определенными математическими соотношениями. Как следует из их названия, открытый ключ может быть известен кому угодно, что не представляет какой-либо угрозы безопасности. Закрытый ключ, напротив, является чрезвычайно конфиденциальной информацией, которая ни в коем случае не должна попасть в чужие руки. Основным принцип безопасной загрузки с использованием асимметричной криптографии заключается в том, что разработчик прошивки подписывает ее с помощью закрытого ключа, в то время как встраиваемое устройство хранит открытый ключ, который и применяется для верификации ПО. Таким образом, при асимметричной криптографии, в отличие от симметричной, конфиденциальный элемент (в данном случае закрытый ключ, используемый для подписи) никогда не хранится во встраиваемом устройстве. Следовательно, если мы задействуем алгоритм ECSDA, у злоумышленника не будет возможно-

сти узнать закрытый ключ, использованный для подписи прошивки и данных, даже если он разберет наше устройство по винтикам. Он сможет узнать только открытый ключ, а алгоритм ECSDA не имеет возможности получить закрытый ключ из открытого. Это фундаментальное свойство асимметричной криптографии.

На рисунке 1 показан процесс безопасной загрузки и зачатки, основанный на алгоритме ECSDA, который обеспечивает высокий уровень доверия при достаточной длине ключа (как правило, не менее 256 бит). Как видно из рисунка, существуют два аспекта этого решения. В отделе проектирования, где разрабатывается прошивка и формируются конфигурационные данные, создается пара ключей ECSDA – закрытый и открытый. Прошивка или данные, подлежащие защите, подписываются в доверенном окружении с помощью закрытого ключа. Как видно из рисунка 2, в процессе вычисления подписи применяется алгоритм FIPS 180 SHA-256: подпись ECSDA вычисляется для хэша SHA-256 исходного образа прошивки или файла данных. На практике подпись вычисляется и добавляется к коду или данным в отделе разработки (см. рис. 1). Именно эта подпись хэша SHA-256 позволяет конечному приложению проверить не только подлинность, но и целостность файла прошивки или данных. Очевидно, что процессор конечного устройства должен обладать внутренними или внешними ресурсами, которых достаточно, во-первых, для вычисления хэша SHA-256 файла прошивки или данных, и, во-вторых, для последующей проверки корректности прикрепленной подписи ECSDA с использованием вычисленного значения хэша и хранящегося в устройстве открытого ключа (см. рис. 3). В случае успешной проверки можно гарантировать, что полученный файл прошивки или данных – подлинный и не подвергался каким-либо изменениям.

ПРОБЛЕМЫ

Очевидно, что правильно реализованный процесс безопасной загрузки или зачатки гарантирует, что во встраиваемом устройстве будет выполняться только авторизованное/подлинное программное обеспечение, поскольку такой процесс предотвращает внедрение вредоносного кода даже при обновлении прошивки. При этом имеются следующие проблемы.

– **Хэш SHA-256** – в случае программной реализации алгоритма для вычисления хэша большого объема данных может потребоваться много времени.

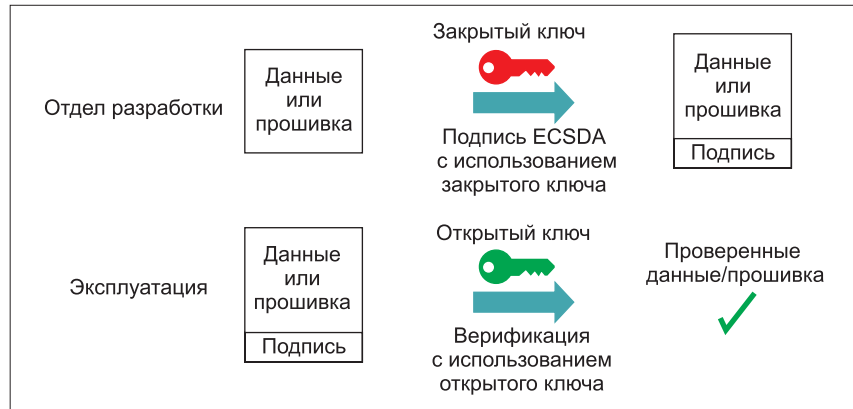


Рис. 1. Использование алгоритма ECSDA для безопасной загрузки и безопасной зачатки

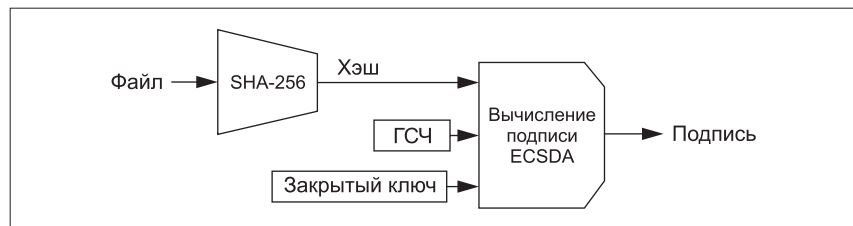


Рис. 2. Формирование подписи ECSDA для файла прошивки/данных

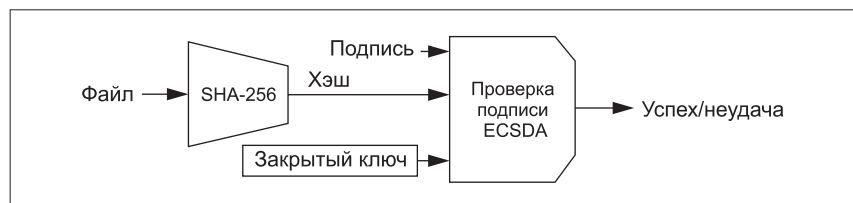


Рис. 3. Верификация подписи ECSDA полученного файла прошивки/данных

- **Верификация подписи ECSDA** представляет собой сложную вычислительную задачу; во встраиваемых устройствах эта операция выполняется, как правило, с использованием подходящих ресурсов, ускоряющих математические расчеты.
- **Реализация** – крайне важно грамотно реализовать криптографические алгоритмы, чтобы исключить появление дыр в безопасности, которые впоследствии сможет обнаружить и использовать злоумышленник.

БЕЗОПАСНАЯ ЗАГРУЗКА И ЗАКАЧКА С ПОМОЩЬЮ DS28C36

Для встраиваемых систем, не имеющих в своем составе защищенного микроконтроллера, который способен выполнять все вычисления по проверке подлинности и целостности загруженного программного обеспечения, компания Maxim Integrated предлагает недорогое решение в виде специализированной микросхемы защищенной аутентификации DS28C36 семейства Deepcover. На рисунке 4 показано, как DS28C36 подключается к основному процессору устройства, и указаны выполняемые ею операции.

- Как упоминалось, ключевая пара из открытого и закрытого ключей,

необходимых для реализации процесса безопасной загрузки или зачатки, формируется в отделе разработки. Закрытый ключ пары используется для генерации подписи прошивки или файла данных, которая, в итоге, проверяется микросхемой DS28C36 в конечном устройстве. Еще раз заметим, что закрытый ключ никогда не выходит за пределы контролируемого окружения отдела разработки. Открытый ключ пары загружается в соответствующий регистр DS28C36 с атрибутом «ключ авторизации» – это конфигурируемая настройка DS28C36.

- Закрытый ключ используется для вычисления цифровой подписи прошивки или файла данных. Как уже упоминалось (см. рис. 2), подпись вычисляется для хэша SHA-256 исходного файла.
- Микросхема DS28C36 с заранее записанным открытым ключом (значение с атрибутом «ключ авторизации») устанавливается на системную плату устройства и подключается к основному процессору.
- Если процессору необходимо выполнить фрагмент кода прошивки или воспользоваться конфигурационными данными, они сначала

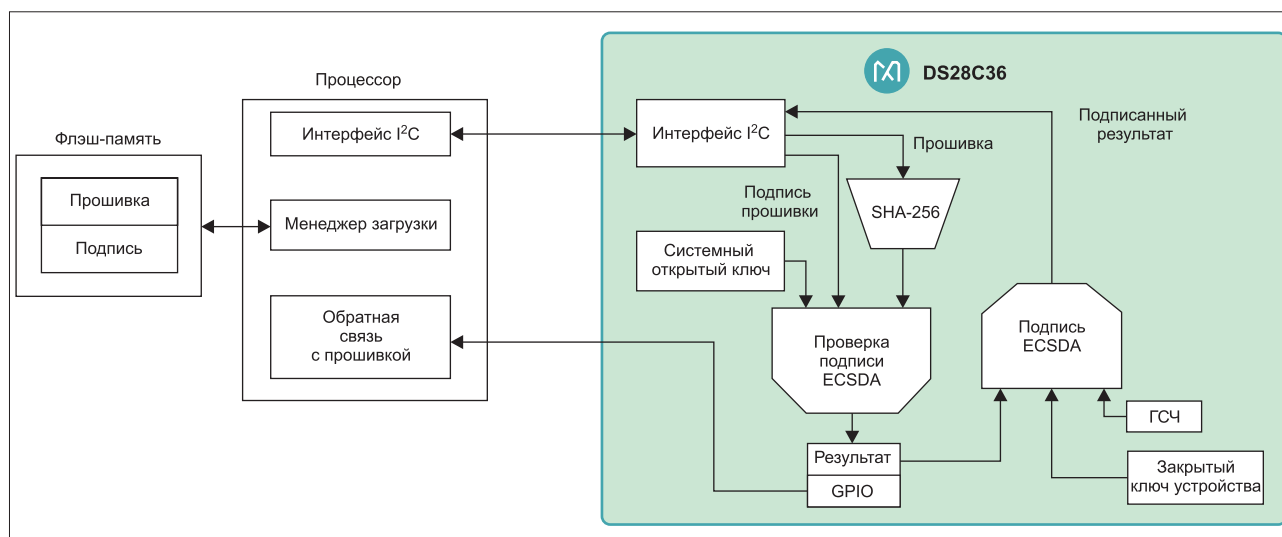


Рис. 4. Взаимодействие основного процессора с DS28C36

Таблица 1. Алгоритм безопасной загрузки с использованием DS28C36

Этап	Основной микроконтроллер	Направление данных	DS28C36
1	Прошивка или файл данных	→	Вычисляет хэш SHA-256, используя функцию вычисления хэша для последовательности блоков
2	Подпись ECSDA прошивки или данных	→	Сравнивает подписи ECSDA для прошивки и для вычисленного значения хэша. Изменяет состояние вывода в случае корректности
3			Результат проверки может быть получен процессором как аппаратно (чтением состояния вывода), так и программно (чтением бита результата)
4	Случайный запрос	→	Вычисляет подпись ECSDA для содержимого регистра, в котором хранится результат проверки допустимости безопасной загрузки
5		←	Результат вычисления ECSDA
6	Проверяет результат вычисления ECSDA для запроса. На данном этапе проверяется соответствие логического уровня на выходе DS28C36 переданному значению для обратной связи		
7	Выполняет код прошивки после успешного завершения безопасной загрузки		

считываются из флэш-памяти менеджером загрузки и передаются в DS28C36 в виде последовательности 64-байт блоков для вычисления хэша SHA-256.

- После того как DS28C36 завершит вычисление хэша, процессор передает ей подпись ECSDA, которая была вычислена на этапе разработки и присоединена к файлу.
- После того, как DS28C36 получит подпись ECSDA, процессор передает команды проверки подписи с использованием предварительно записанного открытого ключа.
- В случае успешной проверки на выходе DS28C36 появляется логический 0, а процессору передается результат верификации. Состояние вывода и значение переданного бита служат сигналом для процессора, что он может безопасно исполнять код или использовать конфигурационный файл.
- Для обеспечения дополнительного уровня безопасности и противодействия принудительному изменению

состояния выхода или подмене бита результата микросхема DS28C36 может опционально сгенерировать подпись ECSDA для внутреннего состояния, которое показывает, разрешено или нет выполнять безопасную загрузку или загрузку. В таблице 1 представлены основные этапы реализации алгоритма безопасной загрузки с использованием DS28C36.

БЕЗОПАСНАЯ ЗАГРУЗКА И ЗАКАЧКА С ПОМОЩЬЮ MAXQ1061

ИС MAXQ1061 представляет собой криптографический контроллер, работающий под управлением собственной прошивки, которая поддерживает:

- безопасную загрузку и безопасную загрузку;
- безопасный обмен данными с использованием протокола TLS;
- безопасное хранение ключей;
- шифрование и вычисление цифровой подписи.

Микросхема MAXQ1061 предназначена для использования в качестве ядра доверия встраиваемых систем. Она отве-

чает всем перечисленным выше требованиям. Ее аппаратные блоки ускорения обеспечивают быстрые вычисления алгоритмов SHA и ECSDA, разгружая тем самым основной процессор. Кроме того, MAXQ1061 предоставляет надежную инфраструктуру для автономного хранения открытых ключей; при этом сертификаты открытых ключей могут быть либо неизменяемыми, либо допускать обновление только уполномоченными лицами. Контроллер MAXQ1061 гарантирует невозможность замены открытого ключа поддельным, что позволяет защитить конечное устройство от атак, заключающихся во внедрении открытого ключа, который обеспечил бы успешную проверку недоверенного программного обеспечения.

Последовательность операций для MAXQ1061 во многом аналогична приведенной ранее для DS28C36. На рисунке 5 показано взаимодействие основного процессора с MAXQ1061.

- Как уже неоднократно упоминалось, пара ключей для обеспечения безопасной загрузки или

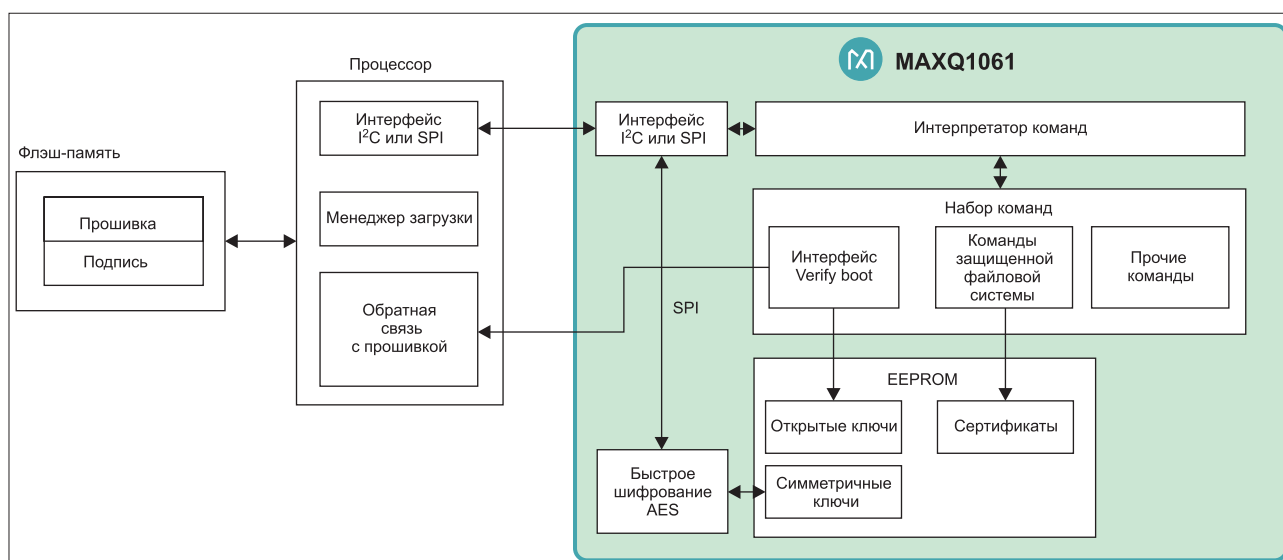


Рис. 5. Взаимодействие основного процессора с MAXQ1061

Таблица 2. Алгоритм безопасной загрузки с использованием MAXQ1061

Этап	Основной микроконтроллер	Направление данных	MAXQ1061
1	Прошивка или файл данных, снабженные подписью ECSDA	→	Вычисляет хэш полученных данных и проверяет подпись ECSDA
2	Подпись ECSDA прошивки или файла данных	←	Возвращает статус команды «VERIFY BOOT»: успех или неудача
3	Сбрасывается или принимает сигнал прерывания	←	Опционально активируется выход RESET_OUT
4		→	В состоянии «SECURE BOOT» предоставляется доступ к определенным объектам. Например, к ключу шифрования прошивки
5	Опционально передает прошивку в MAXQ1061 для расшифровки	→	
6		←	Расшифровывает прошивку и передает результат обратно в основной микроконтроллер
7	Выполняет код прошивки после успешного завершения безопасной загрузки		

закачки формируется в отделе разработки. В случае использования MAXQ1061 длина ключей ECSDA может составлять 256, 384 или 512 бит. Закрытый ключ пары используется для подписи прошивки или файла данных, проверкой которых в конечном устройстве занимается MAXQ1061. Этот закрытый ключ никогда не покидает пределов контролируемого окружения отдела разработки. Открытый ключ пары загружается в MAXQ1061.

- Как видно из рисунка 2, закрытый ключ используется при вычислении цифровой подписи. Она вычисляется для хэша SHA-х файла прошивки или данных и добавляется к нему.
- Основной процессор передает MAXQ1061 команду VERIFY BOOT вместе с проверяемыми данными и их цифровой подписью.
- MAXQ1061 возвращает результат проверки в виде флага успеха или кода ошибки. Опционально может формироваться сигнал на выходе RESET_OUT. Этот сигнал можно использовать для генерации прерывания основного процессора или для его сброса.

- В случае успешной проверки микросхема переходит в состояние Secure Boot. Благодаря наличию защищенной файловой системы пользователь MAXQ1061 может конфигурировать доступ к определенным объектам в случае успешной проверки программного обеспечения. Доступ к таким объектам, если он не заблокирован, предоставляется только в указанном состоянии. Данная функция может использоваться, в частности, для хранения в MAXQ1061 ключа шифрования прошивки – он требуется для расшифровки прошивки только в случае успешной проверки ее подписи.
- Опционально прошивка передается для расшифровки в аппаратный модуль AES-SPI.

В таблице 2 представлены основные этапы реализации алгоритма безопасной загрузки с использованием MAXQ1061.

Выводы

Способность контролировать целостность и оригинальность прошивки или файла с конфигурационными данными, которые установлены во встраиваемое

устройство или загружены в него при эксплуатации, обозначается термином «безопасная загрузка» или «безопасная загрузка». Это проверенное решение для защиты IoT-устройств от атак различного рода. Успешно реализованные безопасная загрузка и безопасная загрузка в устройстве:

- гарантируют подлинность и неизменность загруженного файла данных или прошивки;
- предотвращают внедрение в устройство модифицированных данных или кода;
- повышают безопасность промышленного и медицинского оборудования;
- контролируют разблокирование функциональных возможностей устройства.

Микросхемы DSC28C36 и MAXQ1061 производства компании Maxim Integrated представляют собой простые аппаратные решения, которые позволяют добавлять во встраиваемые системы функции безопасной загрузки прошивки или безопасной загрузки данных для их использования не только на этапе производства, но и во время эксплуатации. ▢

КОММЕНТАРИЙ СПЕЦИАЛИСТА

Николай Федотов, руководитель направления Maxim Integrated, «Симметрон»

На сегодняшний день вопрос информационной безопасности можно назвать самым «горячим» на нашем рынке. Столь большому вниманию данная тема обязана проблемам последнего времени в сфере информационной безопасности – общая история массовых атак пополняется новыми примерами, которые активно обсуждаются всеми мировыми СМИ. Такое внимание, в конечном счете, отражается и на различных группах разработчиков, поскольку обеспокоенные заказчики все чаще начинают интересоваться уровнем безопасности предлагаемых или разрабатываемых систем. Например, это особенно актуально для сферы промышленной автоматизации, поскольку целостность и проверка корректности данных, получаемых от промышленного логического контроллера или конечных измерительных узлов, являются залогом правильного выбора алгоритма для контроля и настройки определенного процесса. Иными словами, без корректных данных об актуальном состоянии системы ею просто-напросто невозможно управлять или анализировать. Мы с вами прекрасно понимаем, что имеются критически важные элементы инфраструктуры, которые недостаточно только перевести на автономное управление без доступа в сеть. Автономность уже давно не является стопроцентной гарантией защиты от киберугроз. Все это вынуждает разработчика посвящать больше времени вопросу безопасности разрабатываемых систем, подыскивать подходящие его проекту способы защиты данных и выстраивать комплексную защиту с помощью разных технологий (аутентификации и потокового шифрования данных), но только там, где это действительно необходимо.

Для задач аутентификации компания Maxim Integrated предлагает обширную линейку аутентификаторов и сопроцессоров к ним, реализующих алгоритмы SHA-1, SHA-256 и ECDSA. Для шифрования разработана серия защищенных микроконтроллеров с безопасной загрузкой и с аппаратными ускорителями SHA, RSA, ECDSA и AES. Обе группы продукции максимально адаптированы для сокращения времени разработки на стороне заказчика. На сегодняшний день все эти решения обеспечивают самый легкий, быстрый и в то же время надежный вариант для перехода на принципиально новый уровень информационной безопасности.

Если говорить несколько подробнее об аппаратной аутентификации в России и СНГ, то можно отметить, что рынок в большей степени знаком с решениями от Maxim Integrated старого поколения на основе алгоритма криптографического хеширования SHA-1 (ключ DS1961S и сопроцессор DS2460 к нему). Эти решения начали активно применяться на нашем рынке только последние пару лет. Ранее применение ограничивалось только небольшой серией точечных проектов. В настоящее время ключ DS1961S стал основой для многих крупных проектов систем домофонии и СКУД. Но и с этим отработанным/проверенным решением могут возникать некоторые недоразумения, причиной которых служит недостаточной уровень компетенции конечного заказчика и, как следствие, составление непродуманных технических заданий. Поскольку при аутентификации на основе симметричной криптографии используется один и тот же секретный ключ как со стороны аутентификатора, так и стороны хоста, для корректного управления такой системой безопасности необходимо сохранять право на создание или изменение

секретного ключа только конечному потребителю. В противном случае, конечный потребитель попадает в ситуацию, когда сторонний разработчик записывает в память аутентификатора и в хост секретный ключ, который знает только

сам разработчик. При этом оценить вероятность утечки секретного ключа со стороны разработчика зачастую невозможно, что неприемлемо, например, для банковского сегмента. Такой подход с поставкой уже запрограммированных ключей объясняется достаточно просто – в основном, это вызвано желанием компании, разработавшей систему, сохранить за собой монопольное право на все последующие поставки аутентификаторов. При этом заказчик не обязательно уведомляется о такой специфике работы и даже не подозревает, что в дальнейшем купить дополнительное количество аутентификаторов для своей системы через другую компанию он просто не сможет. Кроме того, конечный заказчик может размещать запросы на тендерных площадках и пребывать в полной уверенности, что найти оригинальный ключ DS1961S производства Maxim Integrated не составит большого труда, но это совершенно не так.

К счастью, у данной проблемы имеется достаточно элегантно решение, которое может удовлетворить обе стороны. Поскольку в процессе вычисления цифровой подписи участвуют данные из разных блоков памяти, можно одну составную часть заранее запрограммировать на стороне компании-разработчика, а вторую часть сгенерирует сам заказчик в ходе установки системы.

Всем заинтересованным потребителям систем на базе «некопируемых ключей» рекомендуем рассматривать только два варианта: применение оригинальных незапрограммированных аутентификаторов (с сохранением возможности купить их в разных местах) либо применение аутентификаторов с частичным программированием со стороны разработчика (их покупка возможна только у данного разработчика, но утечка ключа с его стороны уже не будет представлять серьезной угрозы). В любом случае, партнерам необходимо однозначно определиться, какой вариант программирования секретных ключей требуется заказчику и какую реализацию готов предложить сам разработчик.

От редакции

Уважаемые читатели, мы хотим еще раз обратить ваше внимание на возможность публикации в журнале ваших отзывов к статьям, посвященным тем или иным компонентам и решениям. Вы нередко используете в своих проектах элементную базу одного и того же производителя и, конечно, хорошо знаете достоинства и недостатки продукции. Приглашаем вас на страницах нашего журнала поделиться с коллегами своим опытом. Думаем, у вас наладится взаимовыгодный обмен мнениями и опытом.

