УДК 004.2

Архитектура криптографического сопроцессора на ПЛИС

А. Ю. Родионов
Закрытое акционерное общество "ОКБ САПР", Москва, Россия

Дано подробное описание архитектуры криптографического сопроцессора на основе программируемых логических интегральных схем (ПЛИС) как наиболее подходящей аппаратной платформы для его реализации. Приведен перечень минимально необходимых элементов системы, указаны их взаимосвязи между собой. Освещены особенности аппаратной реализации в аспекте обеспечения высокой производительности системы. Большое внимание уделяется вопросу безопасности архитектуры с описанием возможных атак и соответствующих мер противодействия.

Ключевые слова: ПЛИС, СКЗИ, HSM, параллельные вычисления, ключевая информация.

Непрерывное увеличение количества пользователей и устройств, подключенных к телекоммуникационным сетям, увеличение пропускной способности каналов связи и вместе с этим объемов передаваемых данных повышают требования к вычислительным мощностям поставщиков дистанционных информационных услуг. Независимо от рода предоставляемых услуг некоторое количество ресурсов выполняет функции по обеспечению информационной безопасности, в частности криптографические функции. Высокие требования к производительности серверов для работы с криптографией вынуждают использовать для этой задачи отдельные аппаратные модули. Автором описана архитектура криптографического сопроцессора, проектировавшегося с учетом современного состояния аппаратных и программных решений, отечественных криптографических алгоритмов, обеспечивающая потенциально высокое быстродействие и высокий класс защиты.

Исходя из области применения криптографического сопроцессора можно сформулировать ряд требований, которым он должен соответствовать.

1. Гибкая настройка и возможность обновления аппаратной реализации криптографических алгоритмов. Выполнение данного требования значительно упрощает поддержку произведенных сопроцессоров, позволяя исправлять ошибки, оптимизировать и добавлять новые реализации криптоалгоритмов. Пользователи смогут динамически менять конфигурацию сопроцессора под текущие нужды системы. Например, при необходимости зашифровать большое количество файлов можно переконфигурировать сопроцессор, уменьшив количество блоков вычисления хэш-функции и увеличив количество блоков шифрования.

2. Преобладание значимости количества одновременных операций над скоростью их выполнения. Отказ в обслуживании из-за недостатка свободных ресурсов критичнее, чем более продолжительное ожидание ответа.

3. Безопасное хранение ключевой информации.

Современные аппаратные решения предлагают несколько альтернатив в качестве платформы для реализации сопроцессора.

- Интегральные схемы специального назначения (ASIC). Разрабатываются для выполнения строго определенных функций, например криптоалгоритмов. Из-за своей узкой специализации имеют высокое быстродействие, но при этом отсутствует возможность динамического аппаратного конфигурирования, что приводит к замене устройства при необходимости использования других криптоалгоритмов. Дороги в разработке и мелкосерийном и единичном производстве.
- Микроконтроллеры и процессоры. Возможна лишь программная реализация алгоритмов, позволяющая легко и быстро разрабатывать и переконфигурировать устройство, но обладающая меньшим быстродействием по сравнению с аппаратной реализацией.
- Программируемая логическая интегральная схема (FPGA). В отличие от цифровых микросхем логика работы задается не на производстве, а при помощи заранее подготовленной прошивки при подаче электропитания, что позволяет динамически переконфигурировать схему. Сочетание этого свойства с быстродействием аппаратной реализации делает ПЛИС наилучшим выбором в соответствии с приведенными требованиями, поэтому дальнейшее описание архитектуры будет основано на ее применении [1].

Родионов Андрей Юрьевич, программист 1-й категории группы программирования аппаратных средств защиты информации.
E-mail: camelot@okbsapr.ru

Статья поступила в редакцию 26 июня 2016 г.

Помимо высокого быстродействия, аппаратная реализация криптографических алгоритмов в виде отдельных функциональных блоков позволяет воспользоваться таким преимуществом, как распараллеливание вычислений. Таким образом, на одном сопроцессоре в один момент времени независимо друг от друга могут выполняться вычисления в соответствии с действующими ГОСТ: ГОСТ 28147-89, ГОСТ 34.12-2015, ГОСТ Р 34.11-94, ГОСТ Р 34.11-2012, ГОСТ Р 34.10-2001, ГОСТ Р 34.10-2012. Естественно, для эффективной работы параллельных вычислений необходима соответствующая поддержка на высоком абстрактном уровне, например организация программных процессов/потоков в ОС хоста и виртуальных каналов передачи данных.

В целом сопроцессор можно представить в виде взаимодействующих между собой и хостом функциональных блоков. Реализация сложных взаимосвязей между ними и распределение нагрузки возлагаются на центральный управляющий элемент. В отличие от функциональных блоков, реализующих ресурсоемкие криптоалгоритмы с неизменной структурой, управляющий элемент большую часть времени находится в режиме ожидания результатов их работы. Кроме того, логика его работы сложна в отладке и может претерпевать изменения. По этим причинам целесообразно реализовать его в виде soft-процессора (процессор, созданный с помощью логического синтеза), выполняющего программный код под управлением многозадачной операционной системы, необходимой для поддержки абстракции параллельных вычислений [2]. Исходя из типичного сценария работы управляющего программного потока, состоящего из ожидания запроса на обработку данных, инициализации соответствующего функционального блока и перехода в режим ожидания результатов обработки или нового запроса, оптимальным вариантом является кооперативная ОС. При данном подходе текущий поток исполняется до тех пор, пока явно не передаст управление следующему потоку, что гарантирует постановку задачи функциональному блоку без накладных расходов на переключение контекста, которые могли бы проявляться при использовании ОС с вытесняющей многозадачностью. Кроме того, отсутствует необходимость защиты разделяемых ресурсов критическими секциями, мьютексами, что значительно упрощает программирование [3].

Кроме уже рассмотренных составляющих элементов сопроцессора, для его функционирования необходимы:

- ПЗУ для хранения ключевой информации, прошивки ПЛИС, кода программ и другой служебной информации;
- ОЗУ;
- физический датчик случайных чисел для выработки ключевого материала и случайных последовательностей.

Наиболее безопасным решением было бы размещение всех вышеописанных компонентов внутри одного кристалла. К сожалению, внутренних ресурсов ПЛИС может быть недостаточно для решения задач высоконагруженных серверов. Универсальное решение состоит в задействовании внешних по отношению к ПЛИС аппаратных компонентов для расширения функциональных возможностей сопроцессора. Однако с увеличением количества элементов системы возрастают ее сложность и количество уязвимостей. Рассмотрим возможные атаки на информационную систему с использованием средств криптографической защиты информации (СКЗИ) в контексте их направленности на проектируемый сопроцессор, а также архитектурные решения, противодействующие им.

1. Атака, направленная на изменение прошивки ПЛИС или кода программ в ПЗУ на этапе производства или обновления СКЗИ. Хранение данной информации в открытом виде позволяет нарушителю успешно провести эту атаку, будучи необнаруженным. Защититься можно добавлением к хранящейся информации имитовставки, вычисленной на секретном ключе производителя, зашиваемым в сопроцессор на этапе производства. Поскольку ПЛИС не может проверить свою собственную прошивку, необходимо добавить элемент, который вычислял бы имитовставку и инициализировал ПЛИС. В качестве такого элемента может выступать микроконтроллер, в котором хранится ключ производителя. Это решение позволяет выявить несанкционированные модификации как при старте устройства микроконтроллером, так и в любой момент времени специальным ПО в ОС хоста.

2. Компрометация ключевой информации, хранящейся в открытом виде в ПЗУ, нарушителем, имеющим физический доступ к СКЗИ. Ее конфиденциальность может быть обеспечена шифрованием на мастер-ключе, не покидающем кристалл ПЛИС, и расшифрованием в ПЛИС непосредственно перед использованием в функциональном блоке.

3. Компрометация ключевой информации, передаваемой между ПЗУ и ПЛИС в открытом виде по каналам побочных излучений и наводок. Описанный ранее метод по ее зашифрованию обеспечивает защиту от данной атаки.

4. Компрометация ключевой информации, хранящейся в открытом виде в оперативном записывающем устройстве (ОЗУ), нарушителем, имеющим физический доступ к СКЗИ. При этом ОЗУ используется для размещения кода программ, кучи

и стэка центрального управляющего элемента. Следовательно управляющий элемент не должен иметь доступа к ключевой информации в открытом виде во избежание возможности ее попадания в ОЗУ. Таким образом, функция шифрования, расшифрования и передачи ключевой информации в открытом виде в функциональные блоки по шине данных, не соединяющейся с управляющим элементом и ОЗУ, возлагается на отдельный элемент в ПЛИС. Из-за высокой частоты использования секретных ключей сопроцессором работу по зашифрованию/расшифрованию целесообразно оптимизировать, объединив ее с функцией кэширования в том же блоке.

5. Несанкционированный доступ к СКЗИ. В случае кражи СКЗИ нарушитель может, пользуясь штатными функциями СКЗИ, выполнять криптографические преобразования данных с использованием секретных ключей. Для исключения этой возможности достаточно разделить мастер-ключ, на котором зашифрованы секретные ключи, на две части, одна из которых — зашита в микроконтроллере сопроцессора, а другая, необходимая к предъявлению перед началом работы, записана на отчуждаемом ключевом носителе.

Итоговая схема криптографического сопроцессора представлена на рисунке.

Архитектура криптографического сопроцессора

Литература

1. *Hasegawa Y., Abe Sh., Matsutani H. et al*. An Adaptive Cryptographic Accelerator for IPsec on Dynamically Reconfigurable Processor. In Proceedings of IEEE International Conference on Field Programmable Technology (FPT2005, 2005).

2. *Biedermann A., Molter H. G*. Design Methodologies for Secure Embedded Systems. Berlin, Heidelberg: — Springer, 2011.

3. *Tanaka K.* Embedded Systems — Theory and Design Methodology // InTech, Chapters published March 02, 2012.

Architecture of cryptographic coprocessor based on FPGA

A. Yu. Rodionov
Closed Joint Stock Company "OKB SAPR", Moscow, Russia

The paper proposes a detailed description of cryptographic coprocessor architecture based on FPGA as the most suitable hardware platform for implementation, provides the list of minimum required system elements and their interconnection. Features of hardware implementation are reviewed in terms of providing high performance. Considerable attention is given to security of architecture, the possible attacks provided with corresponding countermeasures.

Keywords: FPGA, CIPF, HSM, parallel computing, cryptographic keys.

Bibliography — 3 references. *Received June 26, 2016*