Андрей Самоделов (г. Москва)

КРИПТОГРАФИЯ В ОТДЕЛЬНОМ БЛОКЕ: КРИПТОГРАФИЧЕСКИЙ СОПРОЦЕССОР СЕМЕЙСТВА STM32F4XX

*Компания **STMicroelectronics** разгрузила **ARM-ядро** своего нового семейства **микроконтроллеров STM32F4xx,** введя в эти контроллеры отдельный блок **криптографического ускорителя** для решения задач **шифрования данных.** Автор статьи, сотрудник ОАО «ИнфоТеКС», подробно рассматривает работу этого ускорителя.*

Многие производители ARM-процессоров добавляют в свои изделия специальные аппаратные блоки – криптографические ускорители, которые работают отдельно от основного ARM-ядра. Таким образом, ядро ARM может практически не участвовать в криптографических процессах, сохраняя свои ресурсы для выполнения тех задач, с которыми оно справляется наилучшим образом. К таким задачам можно отнести обслуживание обмена с периферийными устройствами, обработку данных, реализацию беспроводного обмена с другими устройствами, управляющие и другие прикладные алгоритмы.

Компания STMicroelectronics, следуя мировым тенденциям, добавила в свои новейшие микроконтроллеры **STM32F415/STM32F417** с 32-разрядным ядром ARM Cortex-M4F криптографический ускоритель, проверенный на микроконтроллерах **STM32F215/STM32F217** с ядром ARM Cortex-M3. Ускоритель позволяет шифровать данные по алгоритмам DES/TDES/AES, вычислять хеш-функции SHA-1/MD5/HMAC и генерировать случайные числа. Повышение максимальной тактовой частоты со 120 МГц (для STM32F2xx) до 168 МГц (для STM32F4xx) позволило повысить и производительность криптографического ускорителя.

Мир все сильнее и сильнее опутывается сетью коммуникаций. Список угроз безопасности передачи данных продолжает расти. В него постоянно добавляются новые способы взлома, мошенничества, новые вредоносные программы и новые типы вирусов. В сегодняшнем, перегруженном коммуникациями, мире возможности для жульничества или воровства часто реализуются одним кликом мышки или одним касанием сенсорного экрана портативного компьютера. Важная персональная и конфиденциальная информация, расположенная в сети интернет, ежедневно передается через беспроводные соединения миллионами людей по всему миру. Кроме того через беспроводные соединения передается информация от большого количества датчиков телеметрических систем сбора и обработки информации, а также сигналы управления промышленными объектами и процессами и транспортными средствами.

Многие производители перемещают криптографическую обработку данных в аппаратные блоки своей продукции. Ускоренная аппаратная криптографическая обработка вместо программного выполнения этих же алгоритмов позволяет разгрузить центральный процессор для выполнения алгоритмов управления и поддержки пользовательского интерфейса.

Основы криптографии

Криптография имеет дело с кодированием или **шифрованием** обмена данными, чтобы сделать сообщения скрытыми от всех лиц, за исключением авторизованных для декодирования или **расшифровывания**. С помощью криптографии реализуется ряд защищенных протоколов обмена данными: с одной стороны канала данные перед их передачей зашифровываются; на другой стороне происходит расшифровывание данных, при этом они преобразуются обратно в понятную форму. Шифрование происходит с использованием специального блока данных, который называется **ключом,** и длина которого зависит от используемого **алгоритма шифрования.** Исходные данные, подлежащие зашифровыванию, называются **открытым текстом,** данные после криптографической обработки называются криптограммой или **шифротекстом.** В современных системах с помощью криптографии решаются четыре основные задачи:

1. Обеспечение конфиденциальности сообщений;
2. Аутентификация;
3. Обеспечение целостности данных;
4. Невозможность отказаться от авторства.

Эти задачи защиты данных реализованы в большом количестве пользовательских приложений, постоянно разворачиваемых во всем мире, включая просмотр Web-страниц, электронную коммерцию, защищенные беспроводные сети, виртуальные защищенные сети (virtual private networks, VPN) и многое другое.

Используются четыре основных вида криптографических преобразований:

1. Криптография с симметричным (или закрытым) ключом;
2. Криптография с асимметричным (или открытым) ключом;
3. Вычисление хэш-функций;

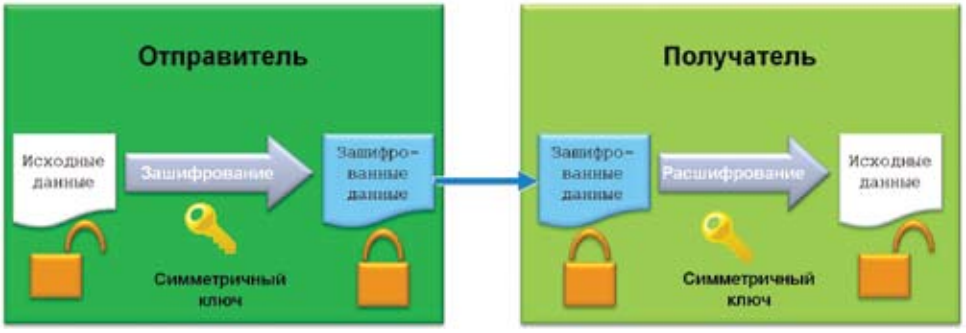


Рис. 1. **Криптография с симметричным ключом**

Таблица 1. **Сравнение алгоритмов шифрования DES/TDES/AES**

Алгоритм/Размер ключа	AES			DES	TDES		
	128	192	256	64[1]	64[2]	128[3]	192[4]
Размер блока, бит	128			64	64		
Время обработки одного блока, тактов HCLK	14	16	18	16	48		
Тип шифра	Блочный			Блочный	Блочный		
Тип сети	Подстановочно-перестановочная			Фейстеля	Фейстеля		
Первая публикация (стандарт)	1998			1977 (январь 1979)	1998 (ANS X9.52)		

Примечание: [1] 8 бит четности; [2] 8 бит четности. Тип ключа 1; [3] 16 бит четности. Тип ключа 2; [4] 24 бита четности. Тип ключа 3.

4. Генерация случайных и псевдослучайных (с заданным алгоритмом формирования последовательности) чисел.

Криптография с симметричным ключом

В криптографии с симметричным ключом оба участника обмена имеют один и тот же ключ, называемый «секретным» (или закрытым), который используется как для зашифровывания, так и для расшифровывания сообщений (рис. 1). Этот закрытый ключ необходимо держать в секрете, чтобы предотвратить возможность расшифровать и прочитать передаваемые сообщения со стороны третьих лиц (называемых нарушителями). Термин «симметричный» применяется постольку, поскольку пользователи на обеих сторонах канала обмена данными используют один и тот же ключ. Симметричный ключ также называется парным, поскольку служит для шифрования сообщений, передаваемых между двумя пользователями. Основной проблемой в этой схеме является защищенная передача секретного ключа. Канал, с помощью которого осуществляется такая передача, называется доверенным каналом. Дополнительную защиту сообщения при использовании симметричной криптографии можно получить за счет использования имитовставки, которая вычисляется на основе исходных открытых данных по специальному алгоритму и гарантирует неизменность передаваемых данных.

Криптографические алгоритмы симметричного шифрования

В зарубежных микроконтроллерах для шифрования на симметричном ключе наиболее часто используют следующие криптографические алгоритмы:

- **Data Encryption Standard (DES) –** Алгоритм шифрования DES разработан в 1970-е годы. Хотя он широко использовался все эти годы, из-за малой длины ключа сейчас он не обладает достаточной криптостойкостью и заменен другими алгоритмами шифрования;

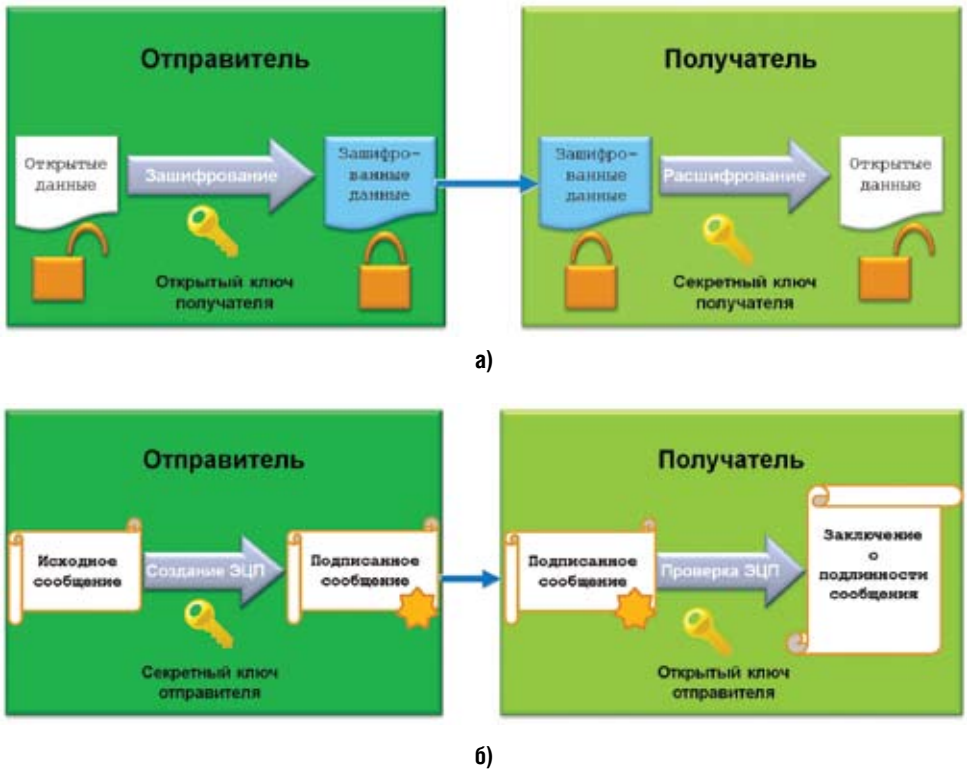


Рис. 2. **Криптография с асимметричным ключом – шифрование**

- **Triple Data Encryption Standard (TDES) –** TDES, для усиления защиты данных по сравнению с алгоритмом DES, выполняет трехкратное шифрование по алгоритму DES с тремя ключами;
- **Advanced Encryption Standard (AES) –** AES является одним наиболее широко используемых современных алгоритмов шифрования данных. На данный момент алгоритм AES является стандартом шифрования данных.

В таблице 1 дана сравнительная характеристика алгоритмов шифрования DES/TDES/AES.

В статье будет рассмотрен блок CRYP, позволяющий на аппаратном уровне шифровать данные и сообщения с использованием алгоритмов DES/TDES/AES.

Криптография с асимметричным ключом

Криптография с асимметричным ключом использует пару ключей, находящихся в такой математической зависимости, что информацию, зашифрованную одним ключом, можно расшифровать только другим ключом. Один из этих ключей называется **открытым** или **публичным,** а второй – **закрытым** или **секретным.** Оба ключа создаются одновременно по особому алгоритму, который гарантирует, что по одному ключу крайне трудно получить другой ключ. Секретность шифрования обеспечивается исключительно надежностью хранения закрытого ключа, которая может быть обеспечена относительно простыми способами. Открытый же ключ может распространяться свободно.

Для криптографии с асимметричным ключом существует два основных применения: шифрование (рис. 2а) и аутентификация (рис. 2б). Зашифрованное открытым ключом сообщение передается по открытому каналу владельцу секретного ключа. Только пользователь закрытого ключа, созданного в паре с открытым, которым было зашифровано сообщение, сможет его прочитать. Аутентификация сообщений чаще всего реализуется с использованием электронной цифровой подписи (ЭЦП), которая

обычно формируется на основе **хэш-функции** сообщения с помощью секретного ключа отправителя. Получатель, имеющий соответствующий открытый ключ отправителя, сможет проверить подпись и узнать, что сообщение является аутентичным.

Функции хэширования (hash function)

Одним из типов криптографических алгоритмов, использующихся в асимметричной криптографии, являются **функции хэширования** или **хэш-функции** (рис. 3). Функция хеширования применяется к данным для получения последовательности битов фиксированной длины, уникальной для каждого сообщения, или блока данных — значения хэша (*hash value*). Даже незначительные несанкционированные или нежелательные изменения данных изменяют значение хэш-функции.

Хэширование используется для формирования электронной цифровой подписи, обеспечения целостности данных, невозможности отказа от авторства, аутентификации сообщений и других видов аутентификации.

Некоторые алгоритмы хэширования были стандартизованы и получили широкое распространение, включая следующие:

- **Message Digest Algorithm (MD5) –** Хотя эта функция хэширования была широко распространена, она имеет уязвимости и не используется в критичных приложениях;
- **Secure Hash Algorithm (SHA) –** SHA имеет несколько модификаций, наиболее криптостойкой считается 256-битная;
- **Hash-based Message Authentication Code (HMAC) –** Основанный на хэш-функциях алгоритм аутентификации.

Рассмотренный в статье блок **HASH** позволяет на аппаратном уровне вычислять хэш-функции **MD5** и **SHA.** Для аутентификации можно использовать реализованный в блоке алгоритм **HMAC.**

Шифрование и аутентификация могут использоваться совместно. Отправитель может вначале зашифровать сообщение открытым ключом получателя, затем подписать собственным закрытым ключом. Получатель вначале будет использовать открытый ключ отправителя для проверки подлинности сообщения, и затем собственный закрытый ключ для расшифровки сообщения.

Проверенный специальным **центром сертификации** открытый ключ помещается в состав специальной структуры данных (**сертификата**), которая кроме самого ключа содержит данные о владельце ключевой пары, об организации, выпустившей сертификат, сроке его действия и некоторые другие. Наиболее известным мировым сервисом сертификации открытых ключей является Verisign. Следует упомянуть, что сертификаты открытых ключей можно проверить с использованием третьей стороны (которая называется **удостоверяющим центром**), обеспечивающей подтверждение того, что ключ принадлежит именно пользователю, подписавшему сообщение. Глобальная инфраструктура, возникающая при использовании асимметричных ключей, называется **инфраструктурой открытого ключа (Public Key Infrastructure, PKI).**

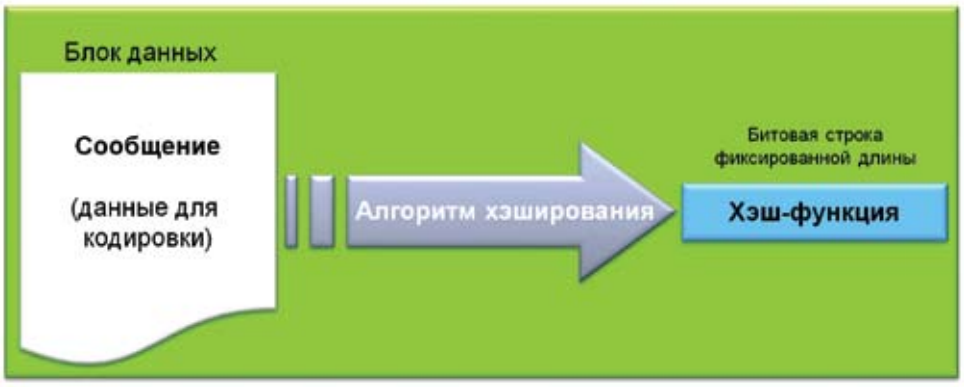


Рис. 3. **Хэш-функция**

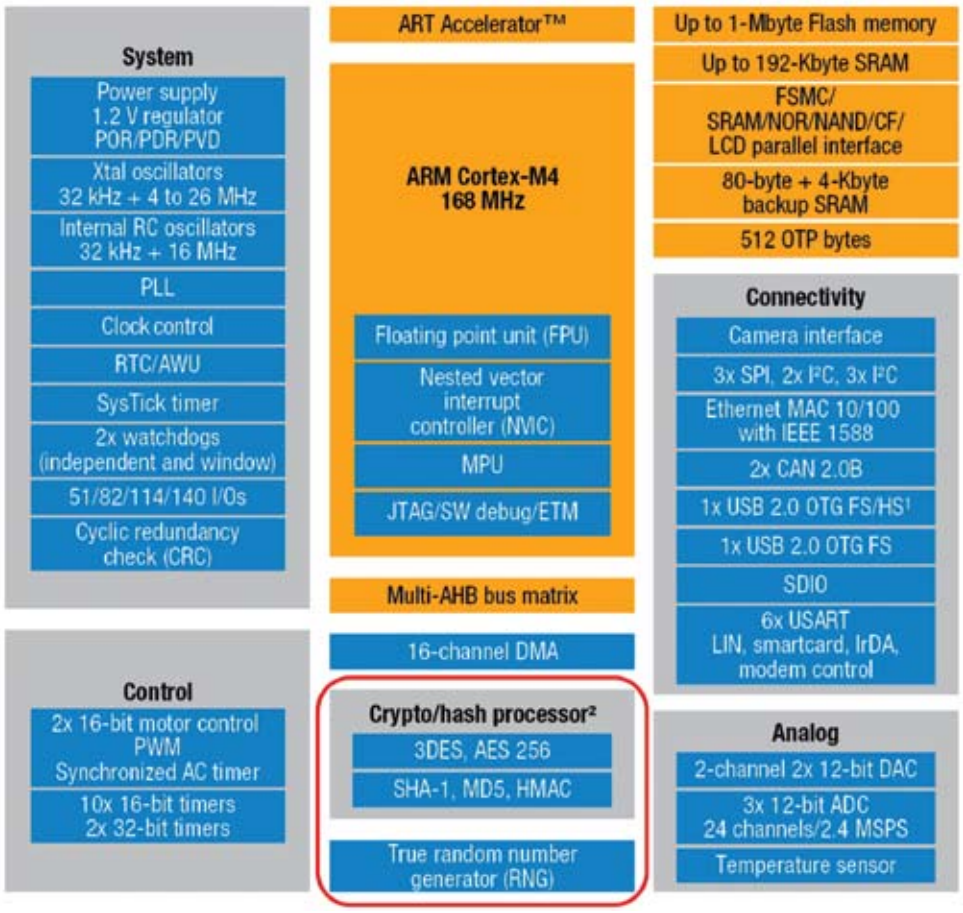


Рис. 4. **Блок-схема микроконтроллеров семейства STM32F4xx**

Генерация случайных чисел

Важной составной частью многих криптографических приложений является **генератор (датчик) случайных чисел – ДСЧ.** Случайные числа используются в таких важных криптографических задачах, как генерация ключей и вычисление **гаммы** для наложения на поток данных с целью их зашифровывания/расшифровывания. Для вычисления гаммы используются случайные числа, генерируемые программным способом. Хотя они не являются истинно случайными числами, но программная реализация при использовании одинаковых начальных значений (которые называются **синхропосылки,** или **вектора инициализации – IV**) позволяет формировать достаточно длинные одинаковые последовательности случайных

Таблица 2. **Основные характеристики микроконтроллеров семейства STM32F4xx**

Технические характеристики/Наименование		F4x5RG	F4x5VG	F4x5ZG	F4x7Vx		F4x7Zx		F4x7Ix	
Объем Flash памяти, кбайт		1024			512	1024	512	1024	512	1024
Объем ОЗУ, кбайт	**Системное**	192 (112+16+64)								
	Резервное	4								
Контроллер FSMC памяти		Нет	Да							
Интерфейс Ethernet		Нет			Да					
Таймеры	**Общего назначения**	10								
	С расширенным управлением	2								
	Базовые	2								
Генератор случайных чисел		Да								
Интерфейсы обмена данными	**SPI / I²S**	3/2 (полный дуплекс)								
	I²C	3								
	USART/UART	4/2								
	USB OTG FS	Да								
	USB OTG HS	Да								
	CAN	2								
Интерфейс видеокамеры		Нет			Да					
Крипто-ядро		Да (только для F415/17)								
Линии GPIO		51	82	114			82		114	140
12-разрядный АЦП		3								
Количество каналов		16	16	24			16		24	24
12- разрядный ЦАП		Да								
Количество каналов		2								
Максимальная частота CPU, МГц		168								
Рабочее напряжение питания, В		1,8…3,6								
Рабочая температура, °С	**Окружающей среды**	-40…85/-40…105								
	Кристалла	-40…125								
Тип корпуса		LQFP64	LQFP100	LQFP144	LQFP100		LQFP144		UFBGA176 LQFP176	

чисел. Случайные числа, генерируемые аппаратным способом, больше приближаются к истинно случайным числам и обычно используются для генерации ключей или начальных значений для программных генераторов случайных чисел (синхропосылок).

Для аппаратной генерации случайных чисел предназначен рассмотренный в статье блок **RNG.**

Перенос криптографии от программной к аппаратной реализации

Загрузка основного процессора системы выполнением криптографического кода с высокой интенсивностью вычислений может увести далеко от выполнения кода пользовательского приложения и значительно замедлить производительность системы на прикладных задачах. Современные встраиваемые процессоры все чаще оборудуются аппаратными ускорителями для выполнения криптографических функций, которые разгружают процессор от этих функций и позволяют больше времени уделять выполнению прикладных алгоритмов. В результате происходит оптимизация производительности системы.

Переход к более эффективным методам криптографической обработки был выполнен плавно и абсолютно прозрачно для разработчиков. В прошлом при выполнении алгоритмов защиты ядро ARM вызывало специальные функции API, и необходимые алгоритмы выполнялись непосредственно ядром ARM. В настоящее время при наличии аппаратных криптографических ускорителей ядро ARM вызывает те же самые функции криптографического API, но сами алгоритмы теперь выполняются специализированным криптографическим модулем, а не ядром ARM. Перенос криптографической обработки с ядра ARM на отдельный аппаратный модуль значительно уменьшил влияние задач защиты на выполнение других прикладных задач.

Микроконтроллеры STM32F4xx

Микроконтроллеры семейства STM32F4xx компании STMicroelectronics представляют собой высокопроизводительные встраиваемые процессоры с 32-разрядным ядром ARM Cortex-M4F, сопроцессором арифметики с плавающей точкой одинарной точности FPU и богатым набором периферийных модулей. Блок-схема микроконтроллеров STM32F4xx приведена на рисунке 4, а основные характеристики – в таблице 2.

На рисунке 4 красной рамкой обведены блоки криптопроцессора, которые и будут подробно рассмотрены далее. К этим блокам относятся:

1. Криптографический процессор **CRYP,** реализующий на аппаратном уровне алгоритмы **DES/TDES/AES;**
2. Процессор вычисления хэш-функций **HASH,** позволяющий вычислять хэш-функции по алгоритмам **SHA-1/MD5** и коды аутентификации сообщений (имитовставки) на **HMAC** основе хэш-функций;
3. Генератор (датчик) случайных чисел **RNG,** позволяющий на основе аналоговых генераторов шума получать **32-разрядные случайные числа.**

Криптографический процессор (CRYP)

Криптографический процессор (CRYP) расположен на 32-разрядной высокоскоростной шине AHB2 и предназначен для зашифровывания/расшифровывания данных в режимах электронного одноразового блокнота (Electronic Codebook, ECB) или зацепления по шифротексту (Cipher block chaining, CBC) для алгоритмов DES, Triple-DES и в режиме со счетчиком (Counter mode, CTR) для алгоритма

Таблица 3. **Производительность криптоускорителя в Мбайт/сек***

Алгоритм/Размер ключа	AES-128		AES-192		AES-256		DES		TDES	
Тип ядра	CM3	CM4	CM3	CM4	CM3	CM4	CM3	CM4	CM3	CM4
Аппаратная (теоретическая)	137,14	192,00	120,00	168,00	106,67	149,33	60,00	84,00	20,00	28,00
Аппаратная без DMA	51,89	72,64	51,89	72,64	44,65	62,51	30,97	43,35	11,43	16,00
Аппаратная с DMA	128,00	192,00	120,00	168,00	106,67	149,33	60,00	84,00	20,00	28,00
Чисто программная	0,99	1,38	0,82	1,14	0,69	0,96	0,53	0,74	0,18	0,25

* – при тактовой частоте ядра Cortex-M3 (CM3) 120 МГц и частоте ядра Cortex-M4F (CM4) 168 МГц.

AES. Для алгоритма DES длина ключа составляет 64 бита, для TDES – 64, 128 или 192 бит, для AES – 128, 192 или 256 бит. В режимах CBC и CTR используется синхропосылка или вектор инициализации (Initialization Vector, IV) длиной в четыре 32-разрядных слова.

Модуль CRYP имеет аппаратную реализацию перечисленных алгоритмов, полностью совместимую со следующими стандартами:

- Стандарт шифрования данных DES и Triple-DES (), разработанный IBM Inc., как определено в публикации FIPS PUB 46-3 от 25 октября 1999 г. и предыдущем стандарте ANSI X9.52 от 1998 г;
- Современный стандарт шифрования данных AES, как определено в публикации FIPS PUB 197 от 26 ноября 2001 г.

Алгоритмы TDES и AES являются блочными шифрами, поэтому неполные блоки входных данных перед зашифровыванием необходимо дополнять некоторыми данными (в конец последнего блока сообщения необходимо записывать некоторые данные). После расшифровывания дополнение необходимо отбрасывать. Аппаратный блок не может управлять операцией добавления/отбрасывания данных, поэтому этим должно заниматься программное обеспечение.

Модуль CRYP обеспечивает автоматический контроль потока данных с поддержкой прямого доступа к памяти (DMA) (используется два канала: один – для приема входных данных, другой – для выдачи обработанных данных) и имеет входной (IN) и выходной (OUT) буферы FIFO, глубиной восемь слов каждый, соответствующие четырем блокам DES/TDES или двум блокам AES. Логика перестановки данных обеспечивает работу с 1-, 8-, 16- или 32-разрядными данными.

В таблице 3 приведены данные по производительности криптоускорителя.

Функциональное описание модуля CRYP

Криптографический процессор состоит из ядра, реализующего алгоритмы DES/TDES/AES буферов входных/выходных данных, регистров хранения ключей/синхропосылки, регистров состояния и регистров управления.

На рисунке 5 показана блок-схема криптографического процессора, на рисунке 6 – флаги и прерывания, а в таблице 4 – описание регистров.

Перейдем к подробному описанию алгоритмов шифрования.

Шифрование по алгоритму DES/TDES

Криптографическое ядро для алгоритма DES/Triple-DES состоит из трех компонентов:

- Блока реализации алгоритма DES (DEA);
- Хранилища ключей. Ключ = [K1] для DES, и Ключ = [K3 K2 K1] для TDES. Ключ [K0] не используется;
- Генератора синхропосылки (для режима CBC).

В режиме TDES выполняется следующий порядок действий:

- Входной блок считывается DEA и зашифровывается с первым ключом, K1;
- Выходной блок расшифровывается с использованием второго ключа, K2, и зашифровывается с использованием третьего ключа, K3.

Результирующий выходной блок используется как шифротекст/открытый текст.

Следует отметить, что выходы промежуточных ступеней DEA недоступны за пределами криптографического ядра.

В режиме TDES допустимы три различных варианта использования ключей (таблица 5).

Документ FIPS PUB 46-3 – 1999 (и ANSI X9.52-1998) дает полное описание режимов обработки информации, реализованных в TDEA. Рассмотрим подробно работу DEA в каждом из этих режимов.

Режим электронного одноразового блокнота DES-ECB/TDES-ECB

Алгоритмы зашифровывания и расшифровывания в режиме DES-ECB/TDES-ECB совпадают и отличаются только порядком применения ключей шифрования K1…K3 (рисунок 7 и рисунок 8).

Зашифровывание/расшифровывание происходит следующим образом:

- Входной 64-битный блок открытого текста (P)/шифротекста (C) считывается из входного буфера IN FIFO;
- После перестановки битов/байтов/полуслов полученный входной блок (I) обрабатывается DEA в режиме зашифрования с ключом K1/K3;
- Блок данных с выхода DEA (O) возвращается на его вход и обрабатывается в режиме расшифровывания с ключом K2;
- Блок данных с выхода DEA (O) возвращается на его вход и обрабатывается в режиме зашифровывания с ключом K3/K1;

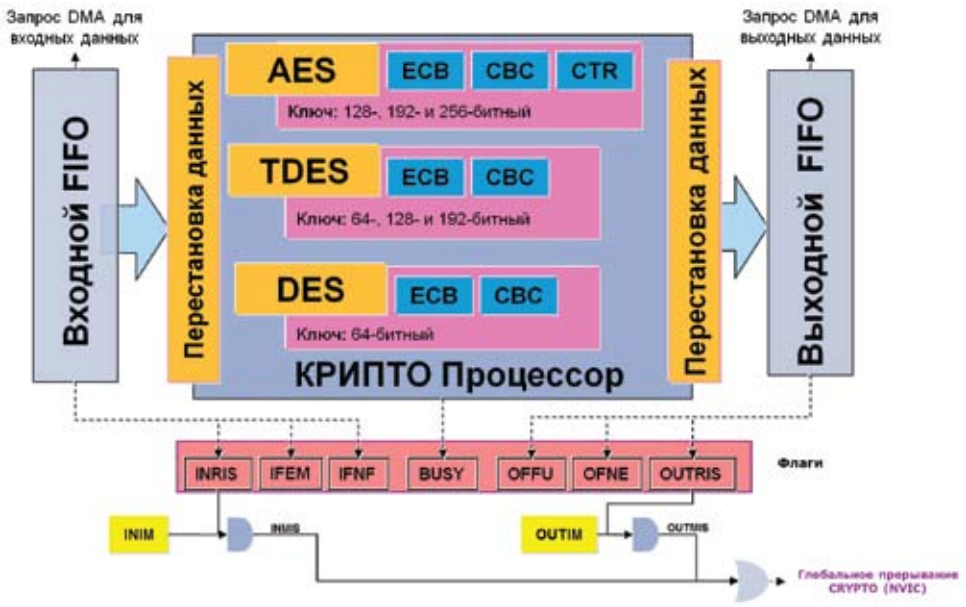


Рис. 5. **Блок-схема криптографического процессора**

Таблица 4. **Краткое описание регистров криптографического процессора**

Обозначение	Функция
CRYP_DIN	Регистр ввода данных
IN FIFO	Входной буфер размером восемь 32-разрядных слов
Swapping logic	Логика перестановки битов/байтов/полу-слов
DES/TDES/AES Processor core	Криптографическое ядро, реализующее алгоритмы шифрования
OUT FIFO	Выходной буфер размером восемь 32-разрядных слов
CRYP_DOUT	Регистр вывода данных
Key CRYP_K0…K3	Регистры ключей (64-разрядные)
Initialization Vectors CRYP_IV0….IV1	Регистры синхропосылки (вектора инициализации)
CRYP_SR	Регистр состояния
CRYP_DMACR	Регистр управления контроллером прямого доступа к памяти
CRYP_IMSCR	Регистр установки/очистки маски прерываний
CRYP_RIS	Регистр состояния сырых прерываний
CRYP_MISR	Регистр состояния маскированных прерываний
CRYP_CR	Регистр управления

Таблица 5. **Варианты использования ключей в режиме TDES**

Три независимых ключа	Ключи K1, K2 и K3 независимы. FIPS PUB 46-3 – 1999 (и ANSI X9.52 – 1998) ссылается на этот режим как на «Keying Option 1», или TDES с тремя ключами.
Два независимых ключа	Ключи K1 и K2 независимы, а K3 = K1. FIPS PUB 46-3 – 1999 (и ANSI X9.52 – 1998) ссылаются на этот режим как на «Keying Option 2» или TDES с двумя ключами.
Три одинаковых ключа	Ключи K1, K2 и K3 одинаковы, т.е., K1 = K2 = K3. FIPS PUB 46-3 – 1999 (и ANSI X9.52 – 1998) ссылается на третий режим как на «Keying Option 3». Этот «одноключевой» TDES эквивалентен режиму DES.

Таблица 6. **Особенности реализации алгоритма AES в зависимости от длины ключа**

Алгоритм	Длина ключа, бит (Nk, слов)	Размер блока, бит (Nb, слов)	Количество раундов, Nr	Время обработки 1 блока, тактов HCLK
AES-128	128 (4)	128 (4)	10	14
AES-192	192 (6)	128 (4)	12	16
AES-256	256 (8)	128 (4)	14	18

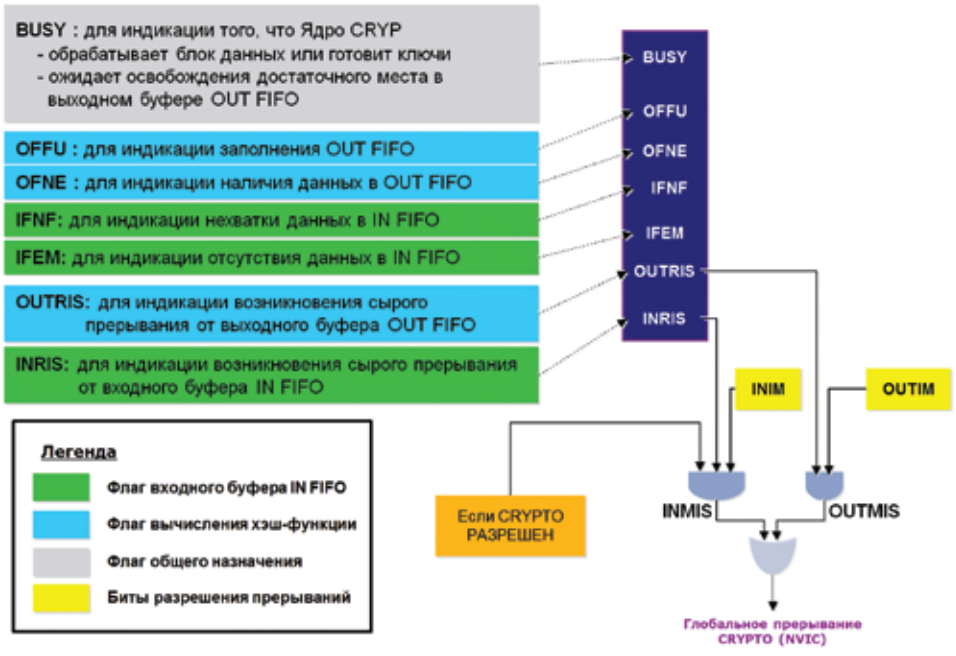


Рис. 6. **Прерывания и флаги криптографического процессора**

• Блок данных с выхода DEA (O) после обратной перестановки битов/байтов/полуслов образует шифротекст (C)/открытый текст (P) и пересылается в выходной буфер OUT FIFO.

Режим с зацеплением по шифротексту DES-CBC/TDES-CBC

Зашифровывание в режиме DES-CBC/TDES-CBC

Зашифровывание в режиме DES-CBC/TDES-CBC происходит следующим образом (рисунок 9):

• Первый 64-битный блок открытого текста (P) считывается из входного буфера IN FIFO и после перестановки, образует блок (Ps). Блок (Ps) складывается по модулю 2 с 64-разрядной синхропосылкой (IV), образуя входной блок (I = IV ⊕ Ps) для шифрования ядром DEA;

• Входной блок (I) обрабатывается ядром DEA в режиме зашифровывания, как это было описано выше;

• Блок данных с выхода DEA (O) записывается на место исходной синхропосылки (IV) и после перестановки образует шифротекст (C). Новое значение синхропосылки (IV) используется для зашифрования следующего блока. Шифротекст (C) пересылается в выходной буфер OUT FIFO;

• Блок данных с выхода DEA (O) при зашифровании второго блока записывается на место синхропосылки (IV) и используется для зашифрования третьего блока и т.д.

Если сообщение содержит неполное количество блоков, то последний блок зашифровывается способом, определяемым конкретным приложением.

Расшифровывание в режиме DES-CBC/TDES-CBC

Расшифровывание в режиме DES-CBC/TDES-CBC происходит следующим образом (рисунок 10):

• Первый 64-битный блок шифротекста (C) после перестановки используется непосредственно в качестве входного блока (I);
• Входной блок (I) обрабатывается ядром DEA в режиме расшифровывания, как это было описано выше;
• Блок обработанных данных с выхода DEA (O) после сложения по модулю 2 с начальной синхропосылкой (IV) образует блок переставленных открытых данных (Ps = IV ⊕ O);
• После перестановки блок исходных открытых данных (P) пересылается в выходной буфер OUT FIFO;
• При расшифровывании второго блока в качестве синхропосылки (IV) используется переставленный блок (I) шифротекста (C) первого блока и т.д.

Шифрование по алгоритму AES

Криптографическое ядро для алгоритма AES состоит из трех компонентов:

• Блока реализации алгоритма AES (AEA);
• Хранилища ключей [K3, K2, K1, K0]. Ключ = [K3, K2] для 128-битного ключа; Ключ = [K3, K2, K1] для 192-битного ключа; Ключ = [K3, K2, K1, K0] для 256-битного ключа;
• Генератора синхропосылки (IV) для режима CBC.

Ядро AES использует ключи трех возможных размеров: 128, 192 или 256 (таблица 6).

В режиме AES выполняется следующий порядок действий:

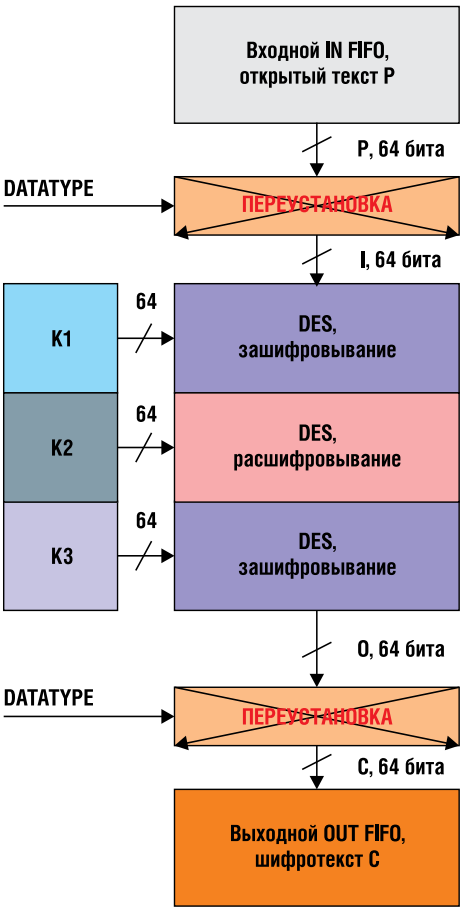
• Входной 128-битный блок считывается AEA и обрабатывается с использованием ключа (K0…3).
• В соответствии с реализованным режимом полученный выходной блок используется для вычисления шифротекста.

Особенностью алгоритма AES является то, что исходный ключ преобразуется в массив четырехбайтных слов [wi], где 0 ≤ i< Nb · (Nr+1). Это приводит к необходимости при расшифровывании производить полную генерацию ключевой последовательности, чтобы в качестве первого раундового ключа при расшифровывании можно было использовать последний раундовый ключ, использованный при зашифровывании.

Полное описание алгоритмов, используемых ядром AES, дается в публикации FIPS PUB 197 (26 ноября 2001). Дадим краткое описание возможных режимов шифрования.

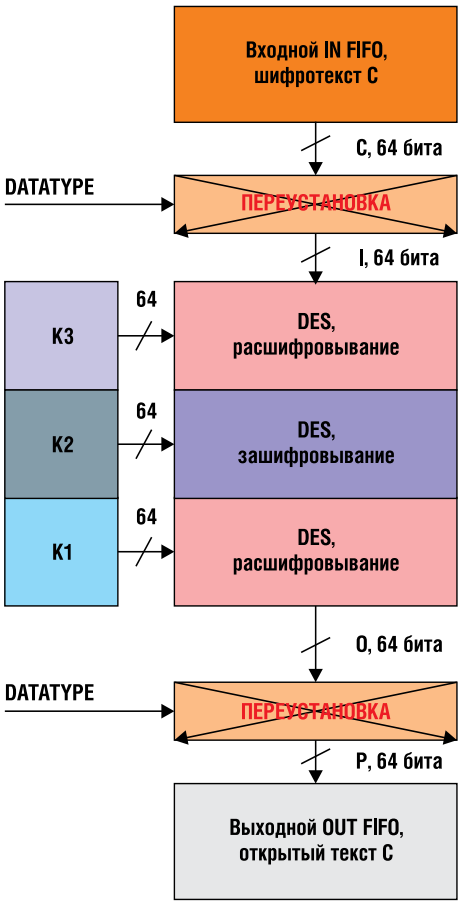
Режим одноразового электронного блокнота AES-ECB

Алгоритмы зашифрования и расшифрования в режиме AES-ECB совпадают (рисунок 11 и рисунок 12) и различаются только необходимостью полной



K: ключ; **C:** шифротекст; **I:** входной блок; **O:** выходной блок; **P:** открытый текст.

Рис. 7. **Зашифровывание в режиме DES-ECB/TDES-ECB**



K: ключ; **C:** шифротекст; **I:** входной блок; **O:** выходной блок; **P:** открытый текст.

Рис. 8. **Режим расшифрования DES-ECB/TDES-ECB**

генерации раундовых ключей (которая выполняется как отдельная операция) перед выполнением операции расшифрования.

Зашифровывание/расшифровывание происходит следующим образом:

• 128-разрядный блок данных открытого текста (P)/шифротекста (C) после перестановки используется в качестве входного блока (I);
• Входной блок (I) обрабатывается AEA в режиме зашифровывания/расшифровывания. Полученный 128-битный выходной блок (O) используется после перестановки как шифротекст (C)/открытый текст (P) и пересылается в выходной буфер OUT FIFO.

Режим зацепления по шифротексту AES-CBC

Зашифровывание в режиме AES-CBC

Зашифровывание в режиме AES-CBC происходит следующим образом (рисунок 13):

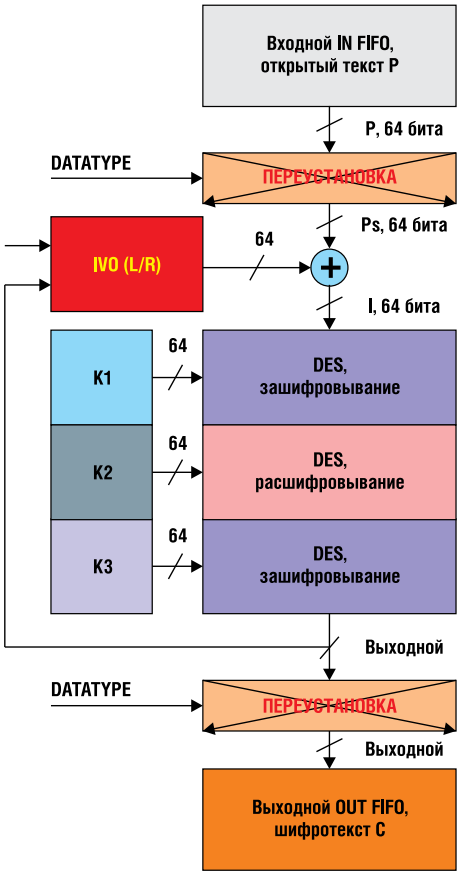
• Первый 128-битный блок открытого текста (P) считывается из входного буфера IN FIFO и после перестановки образует блок (Ps). Блок (Ps) складывается по модулю 2 со 128-разрядной синхропосылкой (IV), образуя входной блок (I = IV ⊕ Ps) для шифрования ядром AEA;
• Входной блок (I) обрабатывается ядром AEA в режиме зашифрования, как это было описано выше;
• Блок данных с выхода AEA (O) записывается на место исходной синхропосылки (IV) и после перестановки образует шифротекст (C). Новое значение синхропосылки (IV) используется для зашифровывания следующего блока. Шифротекст (C) пересылается в выходной буфер OUT FIFO;
• Блок данных с выхода AEA (O) при зашифровывании второго блока записывается на место синхропосылки (IV) и используется для зашифровывания третьего блока и т.д.

Расшифровывание в режиме AES-CBC

При расшифровывании в режиме AES-CBC, так же как при расшифровывании в режиме AES-EBC, необходимо выполнять операцию подготовки ключа, как это было описано выше.

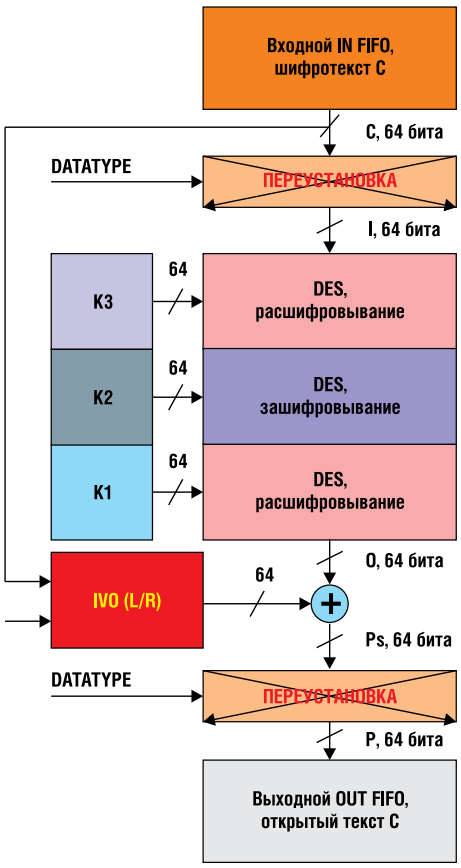
Расшифровывание в режиме AES-CBC происходит следующим образом (рисунок 14):

• Осуществляется подготовка ключа;



K: ключ; **C:** шифротекст; **I:** входной блок; **O:** выходной блок; **Ps:** открытый текст до/после перестановки (при расшифровывании) или после перестановки (при зашифровывании); **Cs:** шифротекст после перестановки (при расшифровывании) или перед перестановкой (при зашифровывании); **P:** открытый текст; **IV:** синхропосылка.

Рис. 9. **Зашифровывание в режиме DES-CBC/TDES-CBC**



K: ключ; **C:** шифротекст; **I:** входной блок; **O:** выходной блок; **Ps:** открытый текст до/после перестановки (при расшифровывании) или после перестановки (при зашифровывании); **Cs:** шифротекст после перестановки (при расшифровывании) или перед перестановкой (при зашифровывании); **P:** открытый текст; **IV:** синхропосылка.

Рис. 10. **Расшифровывание в режиме DES-CBC/TDES-CBC**

- Первый 128-битный блок шифротекста (C) после перестановки используется непосредственно в качестве входного блока (I);
- Входной блок (I) обрабатывается ядром AEA в режиме расшифровывания, как это было описано выше;
- Блок обработанных данных с выхода AEA (O) после сложения по модулю 2 с начальной синхропосылкой (IV) образует блок переставленных открытых данных (Ps = IV ⊕ O);
- После перестановки блок исходных открытых данных (P) пересылается в выходной буфер OUT FIFO;
- При расшифровании второго блока в качестве синхропосылки (IV) используется переставленный блок (I) шифротекста (C) первого блока и т.д.

Если сообщение содержит нецелое количество блоков, то последний блок обрабатывается в соответствии с особенностями приложения.

Режим счетчика AES-CTR

Зашифровывание и расшифровывание в режиме AES-CTR выглядит одинаковым образом (рисунок 15 и рисунок 16).

В режиме AES-CTR блок AES используется как генератор потока ключей, которые для получения шифротекста складываются по модулю 2 с открытым входным текстом. При обработке каждого следующего блока значение одного из полей счетчика увеличивается на 1. Начальное значение счетчика (Initial Counter Block, ICB) выполняет ту же самую функцию, что и синхропосылка IV в режиме CBC. Для расшифровывания используется то же самое значение ICB, что и для зашифровывания.

В таблице 7 показана структура блока IV в режиме AES-CTR.

Флаг «Занят» («BUSY») модуля CRYP

При достаточном количестве данных во входном буфере IN FIFO (не менее двух слов для DES/TDES или четырех слов для AES), достаточном количестве места в выходном буфере OUT FIFO (не менее двух слов для DES/TDES или четырех слов для AES) и CRYP_CR.CRYPEN = 1, криптографический процессор автоматически начинает процесс зашифровывания/расшифровывания (CRYP_CR.ALGODIR).

При выполнении обработки CRYP_SR.BUSY = 1. После завершения обработки два (DES/TDES) или четыре (AES) слова записываются ядром CRYP в выходной буфер OUT FIFO, и CRYP_SR.BUSY = 0.

При обработке данных (CRYP_SR.BUSY = 1) операция записи в регистры ключа, синхропосылки или биты [9:2] регистра CRYP_CR игнорируется, и регистры не модифицируются. Это делает невозможным изменение конфигурации криптопроцессора во время обработки блока данных. Однако, при CRYP_SR.BUSY = 1 можно очистить бит CRYPEN. В таком случае завершается непрерывная обработка данных, после окончания обработки текущего блока два/четыре полученных слова записываются в выходной буфер FIFO, и только затем CRYP_SR.BUSY устанавливается в 0.

Свопинг контекста

Иногда требуется свопинг контекста: из-за того, что новая высокоприоритетная задача, запущенная ОС, требует ресурсы криптопроцессора, для полного восстановления контекста необходимо выполнить следующие задачи (пример с использованием DMA):

Случай AES и DES

1. Сохранить текущую конфигурацию (биты [9:2] регистра CRYP_CR) и, если не в режиме ECB, синхропосылку. Значения ключей должны быть уже доступны в памяти. При необходимости сохранить состояние DMA (указатели на входное (IN) и выходное (OUT) сообщения, количество оставшихся для обработки байтов, и т.п.).
2. Сконфигурировать и выполнить высокоприоритетную обработку.
3. Восстановить контекст.

Случай TDES

Свопинг контекста в режиме TDES происходит аналогично выполнению этой операции в режиме AES. Но, поскольку входной FIFO может содержать до 4 необработанных блоков и время обработки блока значительно больше, в некоторых случаях может оказаться более быстрым прервать выполнения обработки, не дожидаясь, пока буфер IN FIFO опустеет.

1. Сохранить текущую конфигурацию (биты [9:2] регистра CRYP_CR) и, если не в режиме ECB, синхропосылки. Значения ключей должны быть уже доступны в памяти. При необходимости, сохранить состояние DMA (указатели на входное (IN) и выходное (OUT) сообщения, количество оставшихся для обработки байтов, и т.п.). Прочитать данные, загруженные в IN FIFO, которые не успели обработаться, и сохранять их в памяти, пока FIFO не опустеет.

2. Сконфигурировать и выполнить высокоприоритетную обработку.

3. Восстановить сохраненную конфигурацию

Прерывания CRYP

Имеется два независимых маскируемых источника прерываний, генерируемых CRYP. Они объединяются по ИЛИ в один вектор прерывания, который является единственным запросом на прерывание от CRYP в NVIC (контроллер вложенных векторных прерываний). Это комбинированное прерывание выставляется в случае, если возникло одно из описанных выше прерываний, и оно разрешено.

Источники прерываний можно независимо разрешать и запрещать, изменяя значение битов маски в регистре CRYP_IMSCR. Установка соответствующего бита маски в 1 разрешает прерывание.

Состояние каждого источника прерывания можно прочитать либо из регистра CRYP_RISR для статуса необработанного прерывания, либо из регистра CRYP_MISR для замаскированного прерывания.

Прерывание от выходного буфера OUT FIFO – OUTMIS

Сигнал прерывания от выходного FIFO выставляется, когда в выходном FIFO оказывается один или более элементов данных (32-разрядных слов). Это прерывание очищается чтением данных из выходного FIFO до тех пор, пока в нем не остается ни одного (32-разрядного) слова (т.е., прерывание следует за установкой флага OFNE – выходной FIFO не пустой).

Сигнал прерывания OUTMIS от выходного FIFO разрешен даже при очистке бита разрешения работы CRYP. Соответственно, запрещение работы CRYP не приведет к принудительной установке сигнала OUTMIS в низкий уровень, если выходной FIFO не пуст.

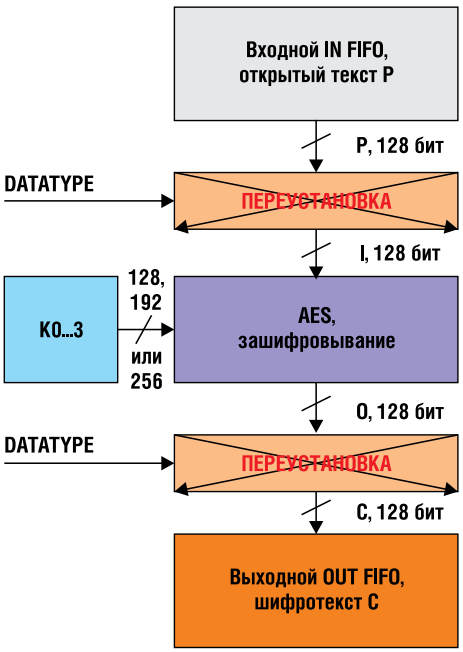
Прерывание от входного буфера IN FIFO – INMIS

Сигнал прерывания от входного буфера FIFO выставляется, если в нем присутствуют менее четырех слов. Он очищается в результате записи во входной буфер FIFO после того, как в нем окажется четыре или более слов.

Сигнал прерывания INMIS от входного буфера FIFO разрешен, только если разрешена работа CRYP. При запрещении работы CRYP сигнал INMIS принудительно устанавливается в 0, даже если входной FIFO пуст.

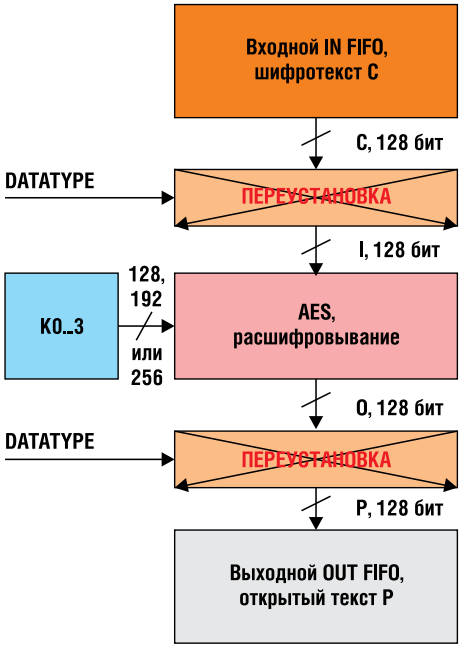
Интерфейс прямого доступа к памяти (DMA) CRYP

Криптографический процессор обеспечивает интерфейс взаимодействия с контроллером прямого доступа к памяти (DMA). Работа DMA определяется регистром управления контроллера DMA модуля CRYP, CRYP_DMACR.



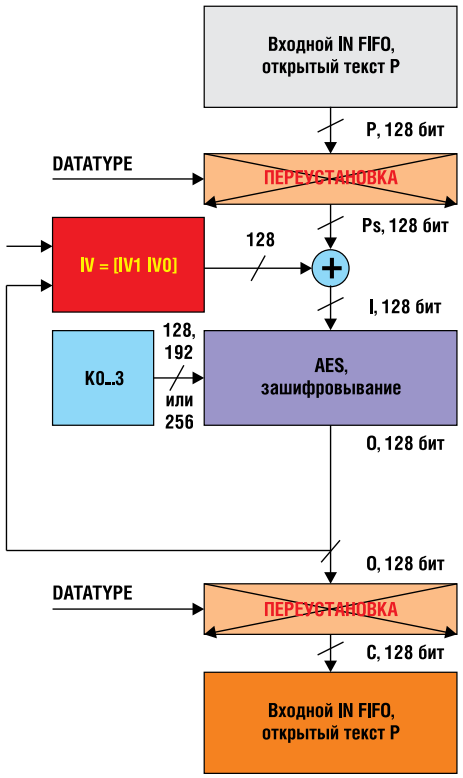
K: ключ; **C:** шифротекст; **I:** входной блок; **O:** выходной блок; **P:** открытый текст.

Рис. 11. **Зашифровывание в режиме AES-ECB**



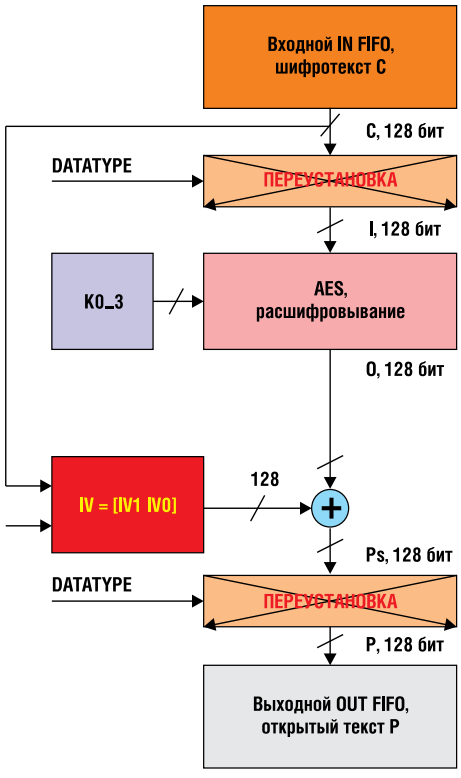
K: ключ; **C:** шифротекст; **I:** входной блок; **O:** выходной блок; **P:** открытый текст.

Рис. 12. **Расшифровывание в режиме AES-ECB**



K: ключ; **C:** шифротекст; **I:** входной блок; **O:** выходной блок; **Ps:** открытый текст до/после перестановки (при расшифровывании) или после перестановки (при зашифровывании); **Cs:** шифротекст после перестановки (при расшифровывании) или перед перестановкой (при зашифровывании); **P:** открытый текст; **IV:** синхропосылка.

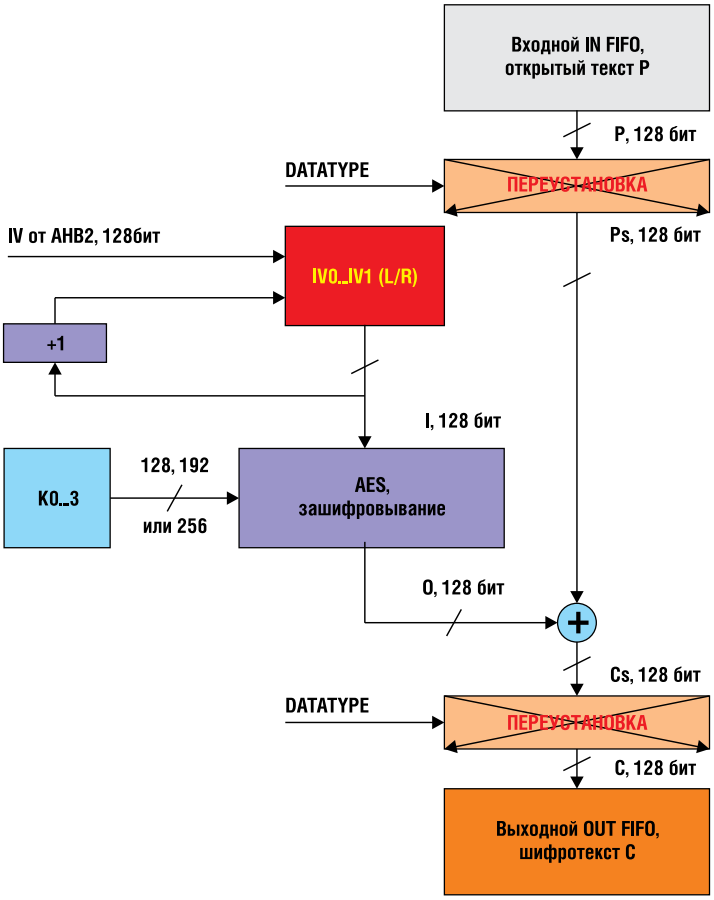
Рис. 13. **Зашифровывание в режиме AES-CBC**



K: ключ; **C:** шифротекст; **I:** входной блок; **O:** выходной блок; **Ps:** открытый текст до/после перестановки (при расшифровывании) или после перестановки (при зашифровывании); **Cs:** шифротекст после перестановки (при расшифровывании) или перед перестановкой (при зашифровывании); **P:** открытый текст; **IV:** синхропосылка.

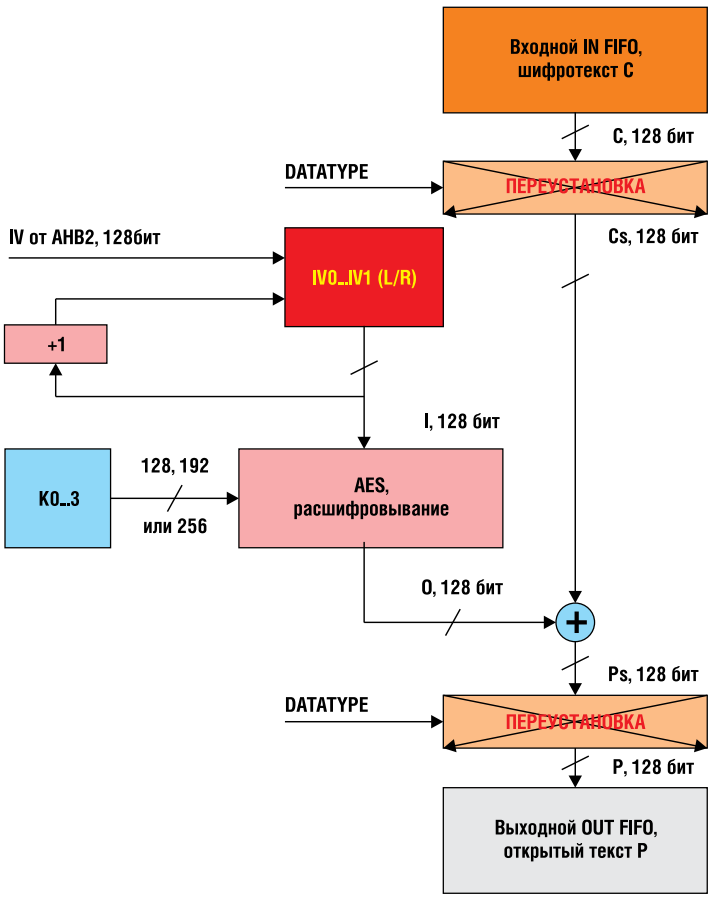
Рис. 14. **Зашифровывание в режиме AES-CBC**

Сигналы запроса на пакетный и одиночный обмен данными не являются взаимоисключающими. Оба они могут подтверждаться одновременно. Например,



K: ключ; **C:** шифротекст; **I:** входной блок; **O:** выходной блок; **Ps:** открытый текст до/после перестановки (при расшифровывании) или после перестановки (при зашифровывании); **Cs:** шифротекст после перестановки (при расшифровывании) или перед перестановкой (при зашифровывании); **P:** открытый текст; **IV:** синхропосылка.

Рис. 15. **Зашифровывание в режиме AES-CTR**



K: ключ; **C:** шифротекст; **I:** входной блок; **O:** выходной блок; **Ps:** открытый текст до/после перестановки (при расшифровывании) или после перестановки (при зашифровывании); **Cs:** шифротекст после перестановки (при расшифровывании) или перед перестановкой (при зашифровывании); **P:** открытый текст; **IV:** синхропосылка.

Рис. 16. **Рисунок 16 Расшифровывание в режиме AES-CTR**

если в выходном OUT FIFO доступны шесть слов, могут подтверждаться запросы на пакетный и одиночный обмен данными. После пакетной передачи четырех слов, для передачи оставшихся двух слов может использоваться только режим одиночного обмена. Это удобно в ситуациях, когда оставшееся для передачи количество слов меньше, чем заданная в конфигурации контроллера DMA длина пакета.

Каждый сигнал запроса остается активным до тех пор, пока DMA не активирует сигнал очистки запроса. После снятия сигнала очистки запроса сигнал запроса снова может стать активным в зависимости от описанных выше условий. Все сигналы запросов игнорируются, если запрещена работа CRYP или очищен бит разрешения DMA (бит CRYP_DMACR.DIEN для IN FIFO и бит CRYP_DMACR.DOEN для OUT FIFO).

Генератор случайных чисел, ГСЧ (Random Number Generator, RNG)

Модуль **RNG** является генератором случайных чисел, основанным на непрерывном аналоговом шуме. Модуль предоставляет базовой подсистеме 32-разрядные случайные числа при выполнении операции чтения из регистра данных RNG_DR. При проведении тестирования в рабочем диапазоне эксплуатации микроконтроллера ГСЧ более чем на 85% удовлетворяет тестам NIST SP800-22 согласно FIPS 140-2 для последовательности из 20000 бит.

Интервал между выработкой двух последовательных случайных чисел составляет 40 периодов тактовой частоты PLL48CLK. Для сигнализации об анормальном поведении (последовательная генерация постоянного значения или повторяющейся последовательности случайных чисел) осуществляется постоянный мониторинг энтропии на выходе ГСЧ. С целью уменьшения общей потребляемой мощности модуль можно отключить.

Функциональное описание ГСЧ

Генератор случайных чисел реализован на базе аналоговой схемы, которая генерирует начальные значения (вектора инициализации) для загрузки в сдвиговый регистр (RNG_LFSR) со

Таблица 7. **Начальная структура блока синхропосылки (IV) в режиме AES-CTR**

Поле	Размер	Описание
Метка времени (Nonce)	32 бита	Однократно используемое значение. При каждом сеансе обмена данными необходимо устанавливать новое значение для этого поля.
Синхропосылка (initialization vector, IV)	64 бита	Случайное значение. Стандартом определено, что производящий зашифровывание должен выбирать IV таким образом, чтобы в паре в данным ключом оно использовалось только один раз.
Счетчик (Counter)	32 бита	Целое в формате big-endian, которое инкрементируется каждый раз после обработки очередного блока. Начальное значение должно быть равно 1.

специальным образом подобранными обратными связями. На выходе RNG_LFSR образуется последовательность 32-разрядных случайных чисел.

Аналоговая схема состоит из нескольких генераторов, выходы которых для получения векторов инициализации с более равномерной энтропией соединены по схеме «ИСКЛЮЧАЮЩЕЕ ИЛИ». Регистр RNG_LFSR тактируется отдельным тактовым сигналом (PLL48CLK) с постоянным значением частоты, что обеспечивает независимость стабильности генерации случайных чисел от частоты основного сигнала синхронизации HCLK микроконтроллера. Содержимое регистра RNG_LFSR преобразуется в содержимое выходного регистра данных ГСЧ (RNG_DR) после загрузки в регистр RNG_LFSR большого количества начальных значений.

Параллельно с генерацией случайных чисел происходит мониторинг начальных значений для сдвигового регистра и значения тактовой частоты PLL48CLK. Биты состояния (в регистре RNG_SR) отображают появление анормальной последовательности начальных значений или выход значения тактовой частоты PLL48CLK за пределы допустимого диапазона. При обнаружении ошибок генерации случайных чисел вырабатывается прерывание.

Принцип работы ГСЧ

Для запуска ГСЧ необходимо выполнить следующие действия:

1. Разрешить прерывания (если это необходимо), установив RNG_CR.IE = 1. Прерывания будут возникать при готовности очередного случайного числа или возникновении ошибок при его генерации.

2. Разрешить работу ГСЧ (RNG_CR.RNGEN = 1). Это активирует аналоговую часть, регистр RNG_LFSR и систему обнаружения ошибок.

3. При обработке прерывания от ГСЧ проверяется, не возникло ли ошибок (биты RNG_SR.SEIS = 0 и RNG_SR.CEIS = 0) и готово ли к выдаче случайное число (RNG_SR.DRDY = 1). Если все эти условия выполнены, то из регистра RNG_DR можно считывать очередное случайное число.

В соответствии с требованиями публикации FIPS PUB 140-2 первое случайное число, сгенерированное после инициализации ГСЧ установкой RNG_CR.RNGEN = 1, не должно использоваться. Его необходимо запомнить для сравнения со следующим сгенерированным случайным числом. Каждое следующее сгенерированное случайное число сравнивается с предыдущим. Проверка выдает ошибку, если любые два последовательно сгенерированных случайных числа оказываются одинаковыми

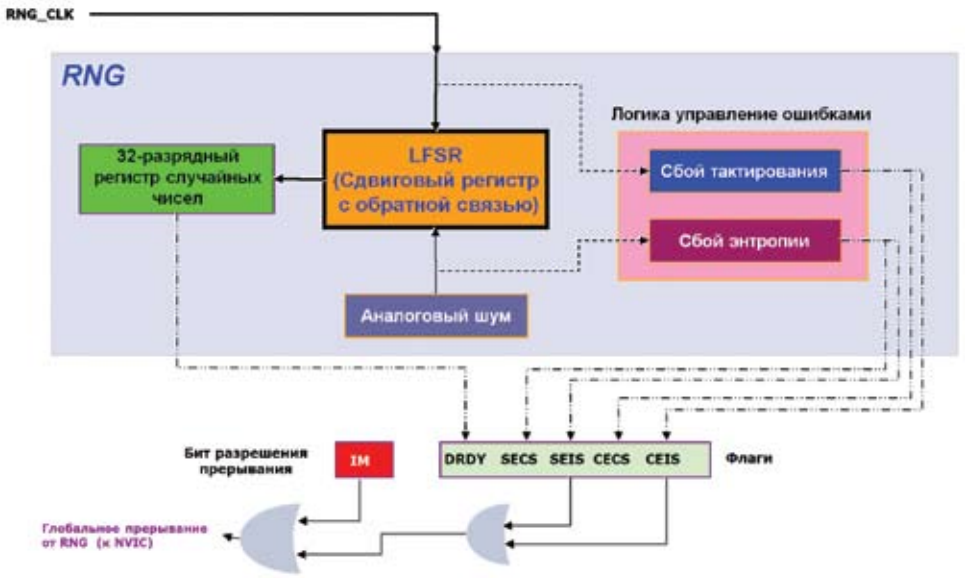


Рис. 17. **Блок-схема модуля RNG**

(непрерывный контроль генератора случайных чисел).

Управление ошибками RNG

Ошибка тактирования (RNG_SR.CEIS = 1)

В этом случае ГСЧ не может дальше генерировать случайные числа. Необходимо проверить корректность конфигурации контроллера синхронизации и значение бита RNG_SR.CEIS. ГСЧ может работать только при RNG_SR.CECS = 0.

Ошибка тактирования не влияет на сгенерированное ранее случайное значение, и содержимое регистра RNG_DR может использоваться в качестве очередного случайного числа.

Ошибка начальных значений (энтропии) (RNG_SR.SEIS = 1)

В случае ошибки выработки начальных значений генерация случайных чисел прерывается на время, пока RNG_SR. SECS = 1. Если в регистр RNG_DR уже записано случайное число, то его нельзя использовать, поскольку оно может не иметь достаточной энтропии. В таком случае необходимо установить RNG_SR.SEIS = 0, и переинициализировать и перезапустить ГСЧ, установив вначале RNG_CR.RNGEN = 0, RNG_CR.RNGEN = 1.

Хэш-процессор (HASH)

HASH-процессор, расположенный на шине AHB, представляет собой блок с аппаратной реализацией алгоритмов хеширования SHA-1 (Secure Hash Algorithm), MD5 (Message-Digest Algorithm 5) и HMAC (Keyed-Hash Message Authentication Code). Он вычисляет хэш-функцию сообщения (160 бит для SHA-1; 128 бит для MD5) для сообщений длиной до 264 – 1 бит. Алгоритм HMAC предоставляет способ подтверждения подлинности сообщений и состоит из последовательного вычисления хэш-функций SHA-1 или MD5 с использованием специального числового значения (ключа).

HASH-процессор имеет 32-битную длину входных данных и поддерживает форматы битовой строки «слово», «полуслово», «байт» и «бит» с порядком следования битов little-endian.

Автоматически переставляет входные little-endian-строки для совместимости с big-endian-стандартом вычисления SHA1 и дополняет входную битовую строку до длины, кратной 512 (16x32 бит). Обеспечивает автоматический контроль целостности потока данных с поддержкой прямого доступа к памяти (DMA). На рисунке 18 показана блок-схема хэш процессора.

Сообщение или файл данных, обрабатываемый хэш-процессором, необходимо рассматривать как битовую строку. Длиной сообщения является количество битов в сообщении (пустое сообщение имеет длину 0). В стандарте FIPS PUB 180-1 используется соглашение, что нумерация битов в битовой строке возрастает слева направо, и биты могут группироваться в байты (8 бит) или слова (32 бита). Некоторые реализации используют также полуслова (16 бит) с порядком следования байтов (полуслов) big-endian. Это соглашение в основном важно при создании дополнения неполного блока до стандартного размера.

На непосредственное вычисление хеш-функции одного блока сообщения требуется:

- 66 циклов тактовой частоты HCLK для алгоритма SHA-1;
- 50 циклов тактовой частоты HCLK для алгоритма MD5.

Для получения значения полного времени обработки блока необходимо

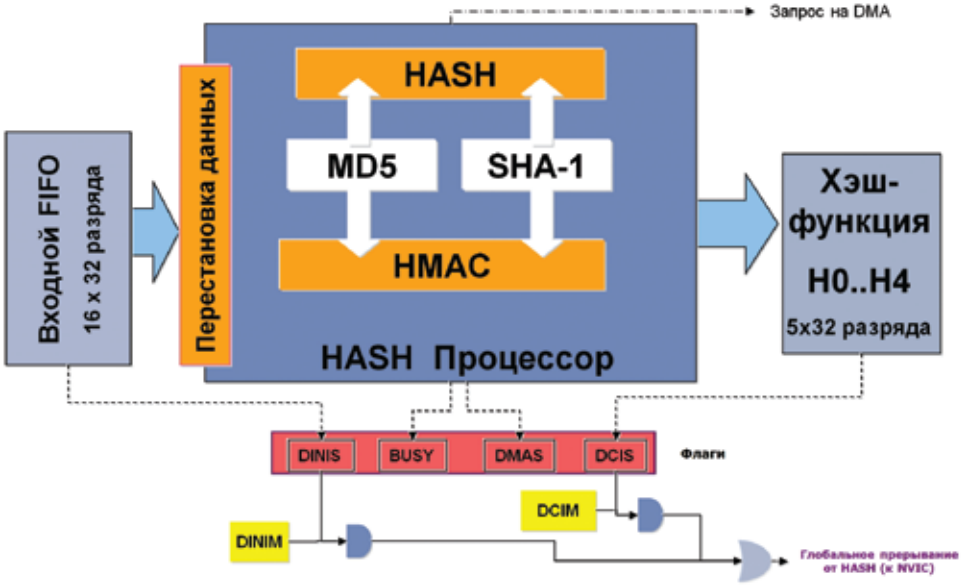


Рис. 18. **Блок-схема HASH-процессора**

добавить время загрузки 16 слов блока (16 циклов тактовой частоты для 512-битного блока).

Время обработки последнего блока сообщения (или ключа в режиме HMAC) может быть более длительным, поскольку включает в себя операцию дополнения блока. Оно зависит от длины последнего блока и размера ключа (в режиме HMAC).

По сравнению со временем обработки средних блоков, оно может увеличиваться:

- в 1…2,5 раза при вычислении хэш-функции сообщения;
- в 2,5 раза для входного ключа в режиме HMAC;
- в 1…2,5 раза при обработке сообщения в режиме HMAC;
- в 2,5 раза для выходного ключа в режиме HMAC в случае короткого ключа;
- в 3,5…5 раз для выходного ключа в режиме HMAC в случае длинного ключа.

В таблице 8 приведены значения производительности HASH-процессора для различных вариантов реализации алгоритмов

Хэш-функция сообщения

Подготовленное для обработки сообщение контроллером прямого доступа к памяти (DMA) или центральным процессором (CPU) последовательно вводится в HASH процессор блоками по шестнадцать 32-разрядных слов (512 бит), записываемых в регистр HASH_DIN. После введения последнего блока автоматически начинается вычисление хэш-функции. Всякий раз, когда в регистр HASH_DIN записываются новые данные, его текущее содержимое передается во входной буфер IN FIFO. Регистр HASH_DIN и IN FIFO формируют буфер FIFO размером 17 слов.

Имеются следующие условия начала обработки:

- Прямой доступ к памяти (DMA) не используется:

– При вычислении промежуточной хэш-функции в регистр HASH_DIN записывается дополнительное слово (первое слово следующего блока). После этого программа перед записью новых данных в HASH_DIN должна ожидать флаг готовности HASH-процессора (HASH_SR.DINIS = 1).

– В случае вычисления окончательной хэш-функции (введен последний блок) это производится путем установки бита HASH_STR.DCAL в 1.

- При использовании прямого доступа к памяти (DMA):

– Содержимое регистра HASH_DIN интерпретируется автоматически после завершения передачи информации контроллером DMA.

Процесс ввода данных и вычисление частичной хэш-функции повторяется до тех пор, пока в регистр HASH_DIN не будут записаны последние биты исходного сообщения. Поскольку длина сообщения может быть любой, последнее слово, записываемое в HASH процессор, может иметь любую длину от 1 до 32 бит. Это число валидных битов последнего слова, NBLW, необходимо записать в регистр HASH_STR, чтобы перед окончательным вычислением хэш-функции было корректно проведено дополнение сообщения. Обработка последнего введенного блока запускается установкой бита HASH_STR.DCAL в 1. При этом сообщение автоматически дополняется, и вычисляется окончательная хэш-функция.

При разрешенном прямом доступе к памяти (DMA) информация поступает в хэш-процессор после передачи последнего слова данных. Дополнение и вычисление хэш-функции выполняется автоматически при установке бита HASH_STR.DCAL в 1.

Дополнение сообщения

Дополнение сообщения состоит в следующем:

1. В последнем слове, введенном в регистр HASH_DIN, бит с номером, задаваемым значением битового поля HASH_STR.NBLW, устанавливается в 1.

2. Все биты, следующие за заданным в поле HASH_STR.NBLW и до конца блока (размером 512 бит), устанавливаются в 0.

Принцип работы HASH-процессора

Для вычисления хэш-функции (SHA-1, MD5) бит HASH_CR.INIT устанавливается в 1, а бит HASH_CR.MODE устанавливается в 0. Тип алгоритма (SHA-1 или MD5) выбирается битом HASH_CR.ALGO. Обработка данных запускается установкой бита HASH_CR.INIT в 1. HASH-процессор остается занятым в течение 66 циклов тактовой частоты для алгоритма SHA-1 или 50 циклов для алгоритма MD5. Значение хеш-функции можно прочитать из регистров HASH_H0…HASH_H4 (для алгоритма MD5 содержимое регистра HASH_H4 не имеет значения).

Вычисление HMAC

Алгоритм HMAC используется для аутентификации сообщения, путем вычисления односторонней функции от сообщения с использованием выбранного пользователем ключа в соответствии с

Таблица 8. **Производительность HASH-процессора в Мбит/сек для различных вариантов реализации алгоритмов SHA-1 и MD5**

Тип производительности	MD5		SHA1	
	CM3	CM4	CM3	CM4
Теоретическая аппаратная	116,36	162,9	93,66	131,12
Аппаратная без DMA	55,25	77,35	51,20	71,68
Аппаратная с DMA	75,29	105,4	65,08	91,11
Программная	8,23	1152	3,68	5,15