9 Security System

9.1 Crypto Engine (CE)

9.1.1 Overview

The Crypto Engine (CE) module is one encryption/decryption algorithms accelerator. It supports kinds of symmetric, asymmetric, HASH, and RBG algorithms. There are two software interfaces for secure and non-secure world each. Algorithm control information is written in memory by task descriptor, then CE automatically reads it when executing request. It supports parallel requests from 4 channels each world and 4 different types of algorithms simultaneously. This module also has an internal DMA controller to transfer data between CE and memory. It supports parallel running for symmetric, HASH, asymmetric algorithms.

The CE has the following features:

- Symmetrical algorithm:
  - AES symmetrical algorithm
    - ➢ Key size: 28/192/256 bits
    - ➢ CFB mode includes: CFB1, CFB8, CFB64, and CFB128
    - ➢ CTR mode includes: CTR16, CTR32, CTR64, and CTR128
    - ➢ Supports ECB, CBC, CTS, OFB, CBC-MAC, and GCM modes
  - DES symmetrical algorithm
    - ➢ CTR mode, includes: CTR16, CTR32, and CTR64
    - ➢ Supports ECB, CBC, and CBC-MAC mode
  - Supports 3DES
  - SM4 symmetrical algorithm supports ECB and CBC mode
- Hash algorithms
  - Support MD5, SHA1, SHA224, SHA256, SHA384, SHA512, and SM3
  - Support HMAC-SHA1, HMAC-SHA256
  - Support multi-package[1] mode for these ones
  - Support hardware padding
- Random bit generator algorithms
  - Support PRNG, 175 bits seed width, and output with multiple of 5 words

[1] If not last package, input should aligned with computation block, namely 512bits or 1024bit

  - Support TRNG, post-process by hardware with SHA256, output with multiple of 8 words
  - Support Instantiate/Reseed/Generate/Uninstantiate 4 process
  - Support prediction resistance requests
  - Support 8 separate suits of Internal State
  - Maxim 2^32 BYTE length of Entropy input, Nonce, Personalization, Additional input. And length is multiple of word

- Public key algorithms
  - Supports RSA public key algorithms: 512/1024/2048/3072/4096-bit width
  - Supports ECC public key algorithms: 160/224/256/384/521-bit width
  - Supports SM2 algorithms

- Security Strategy and System Feature
  - Symmetric, asymmetric, HASH/RBG ctrl logics are separate, can handle task simultaneously. Symmetric logic can select instantiate 2 suits at implementation time.
  - Support task chain mode for each request. Task or task chain are executed at request order.
  - multi- scatter group(sg) are supported for both input and output data
  - Support secure and non-secure interfaces respectively, each world issues task request through its own interface, don’t know each other’s existence.
  - Each world has 4 channels for software request, each channel has an interrupt control and status bit, and channels are independent with each other.
  - Supports byte-aligned address for all configurations

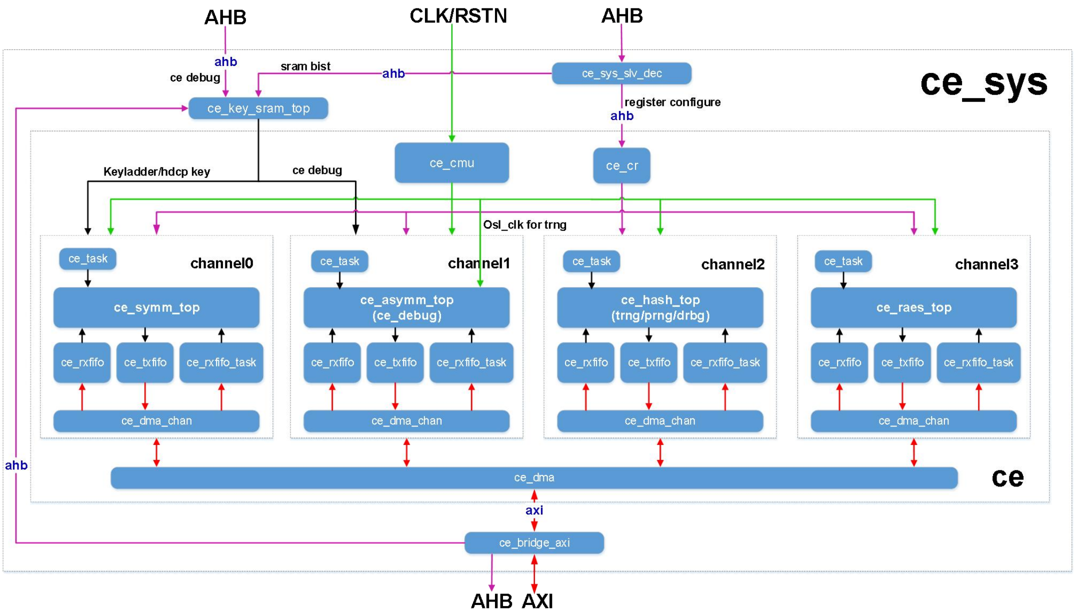
NOTE

The total length of data_length in the CBC and ECB modes of the symmetric channels needs to be aligned according to the algotithm granularity. For example, in the AES-128 algorithm, the total length of data_length needs to be an integer multiple of 128 bits

9.1.2 Block Diagram

The following figure shows a block diagram of CE.

Figure 9-1 CE Block Diagram

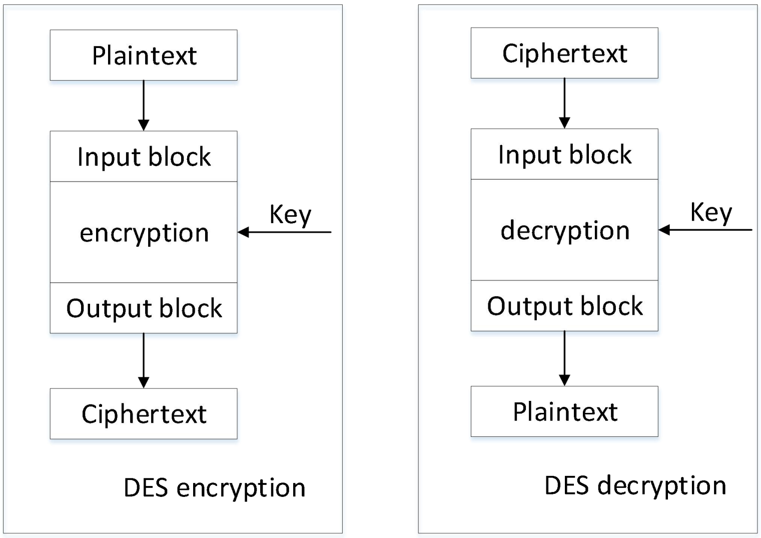


9.1.3 Functional Description

9.1.3.1 DES Algorithm

The following figure shows the DES encryption and decryption operation.

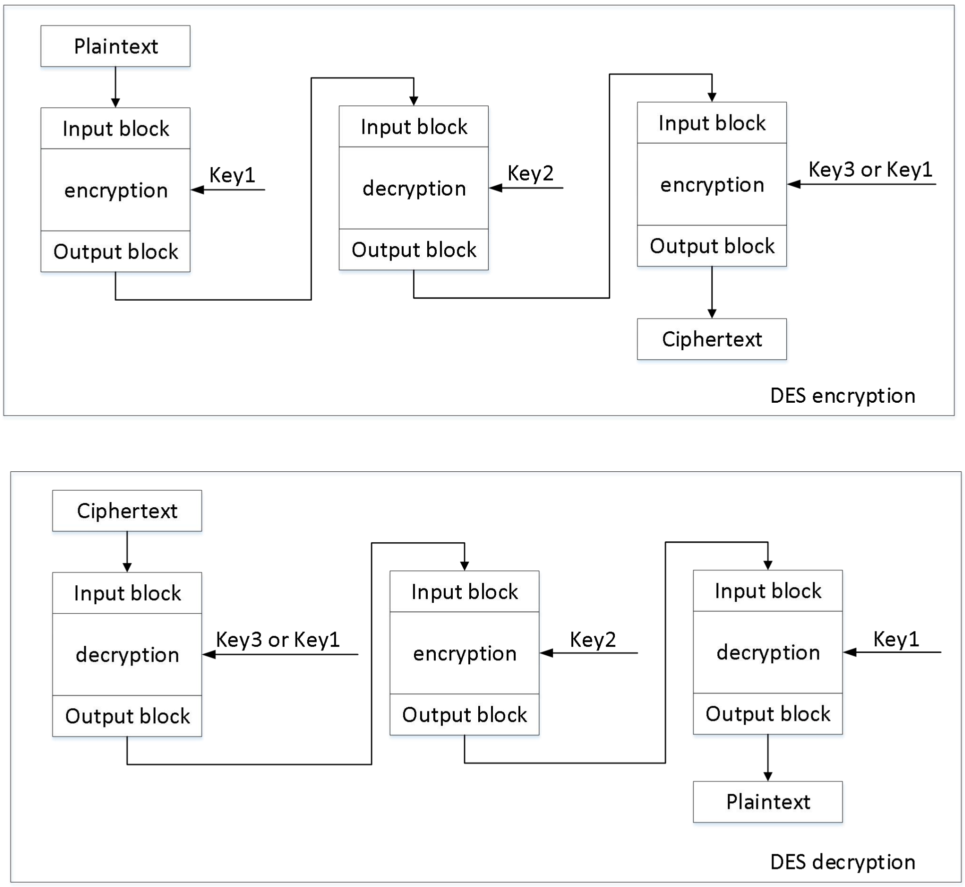
Figure 9-2 DES Encryption and Decryption



9.1.3.2 3DES Algorithm

The 3DES algorithm supports both 3-key and 2-key operations. A 2-key operation can be regarded as a simplified 3-key operation. To be specific, key 3 is represented by key 1 in a 2-key operation. The following figure shows the 3DES encryption and decryption operation of a 3-key operation and a 2-key operation.

Figure 9-3 3DES Encryption and Decryption of a 3-key Operation and a 2-key Operation

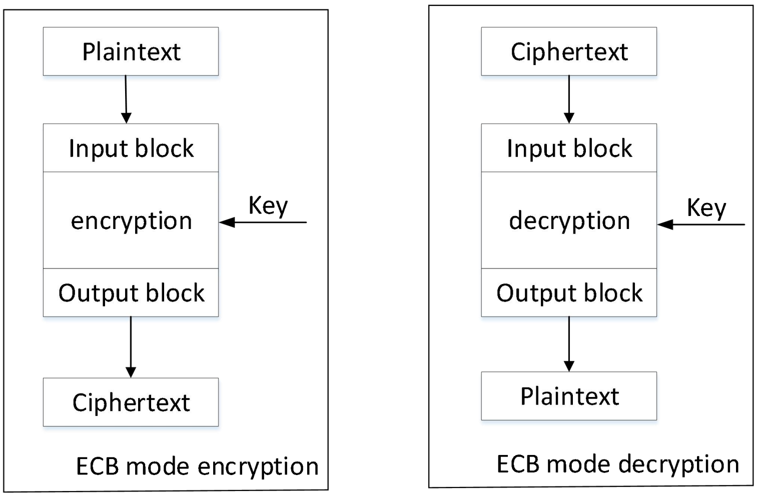


9.1.3.3 ECB Mode

The ECB mode is a confidentiality mode that features, for a given key, the assignment of a fixed ciphertext block to each plaintext block, analogous to the assignment of code words in a codebook.

In ECB mode, encryption and decryption algorithms are directly applied to the block data. The operation of each block is independent, so the plaintext encryption and ciphertext decryption can be performed concurrently.

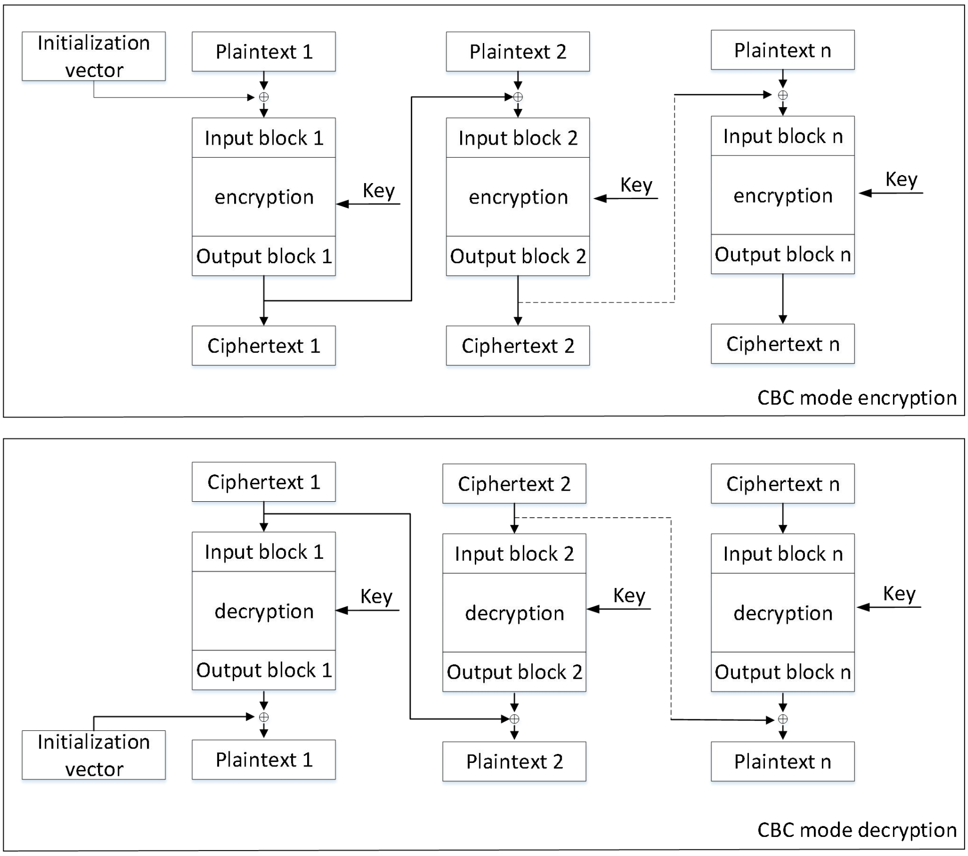
Figure 9-4 ECB Mode Encryption and Decryption



9.1.3.4 CBC Mode

The CBC mode is a confidentiality mode whose encryption process features the combining of the plaintext blocks with the previous ciphertext blocks. The CBC mode requires an initialization vector (IV) to combine with the first plaintext block. The encryption process of each plaintext block is related to the block processing result of the previous ciphertext blocks, so encryption operations cannot be concurrently performed in CBC mode. The decryption operation is independent of output plain text of the previous block, so decryption operations can be performed concurrently.

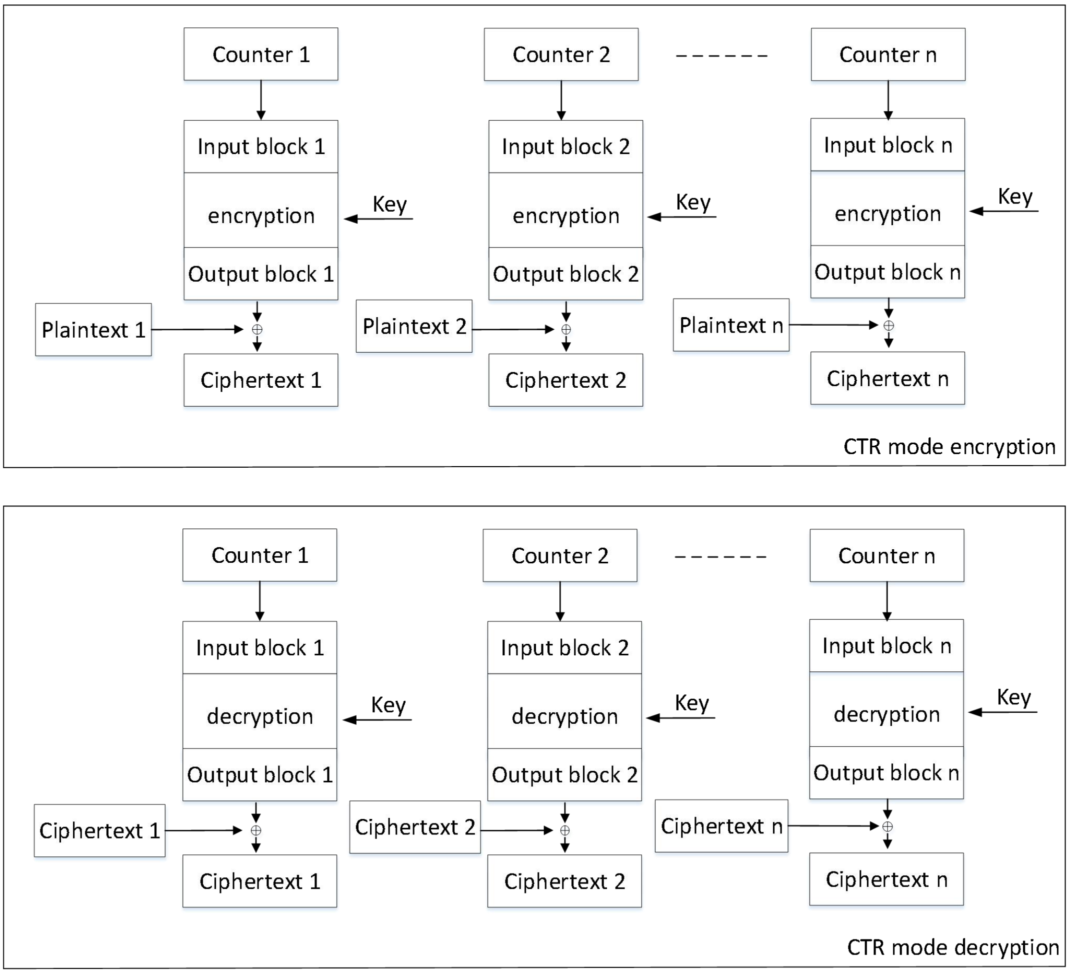
Figure 9-5 CBC Mode Encryption and Decryption



9.1.3.5 CTR Mode

The CTR mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. All of the counters must be distinct.

Figure 9-6 CTR Mode Encryption and Decryption

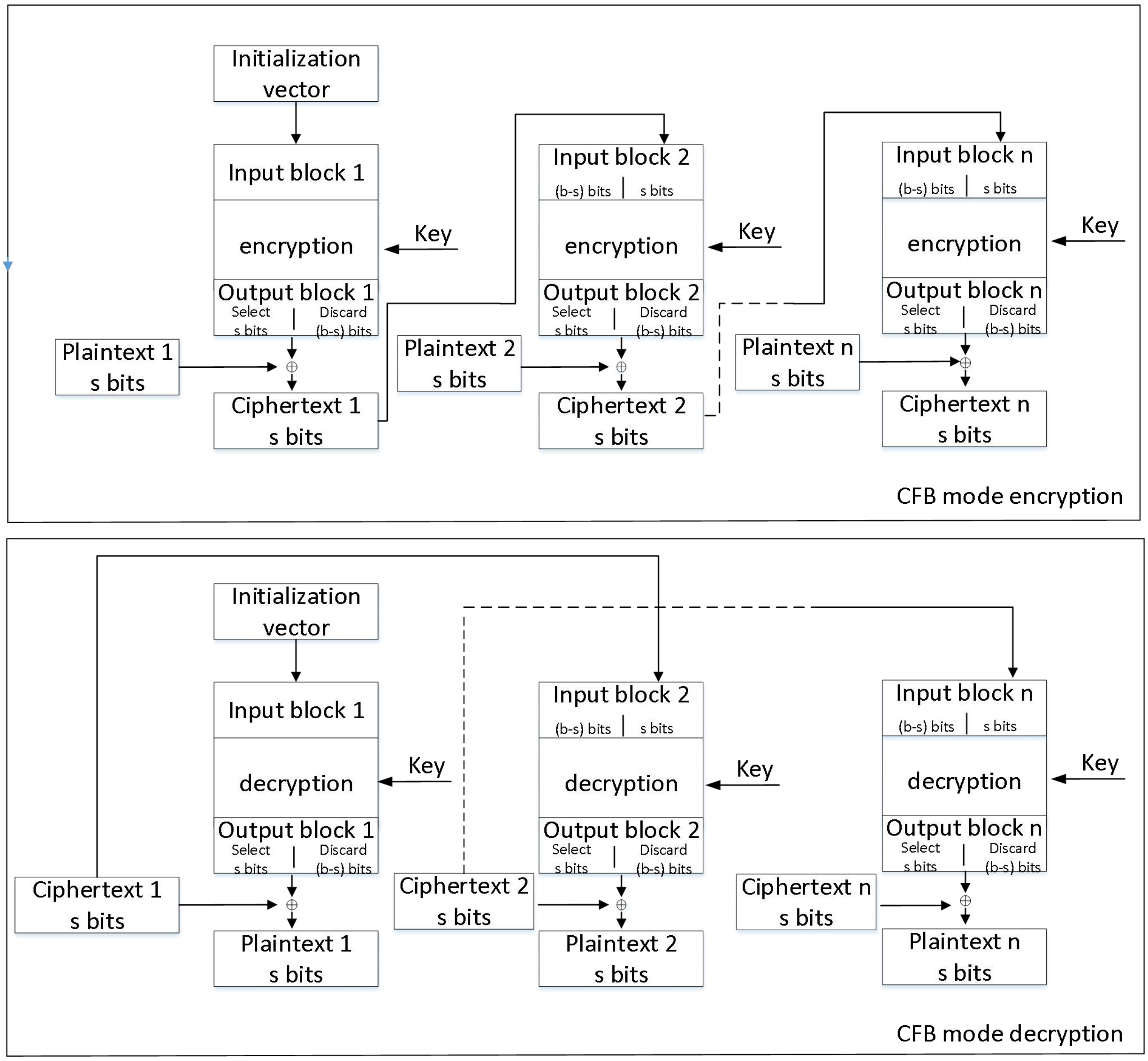


9.1.3.6 CFB Mode

The CFB mode is a confidentiality mode that features the feedback of successive ciphertext segments into the input blocks of the forward cipher to generate output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The CFB mode requires an IV as the initial input block, and the forward cipher operation is applied to the IV to produce the first output block. The first ciphertext segment is produced by exclusive-ORing the first plaintext segment with the s most significant bits of the first output block. The value of s is 1 bit, 8 bits, 64 bits, or 128 bits.

The following figure shows the s-bit CFB mode of the AES algorithms.

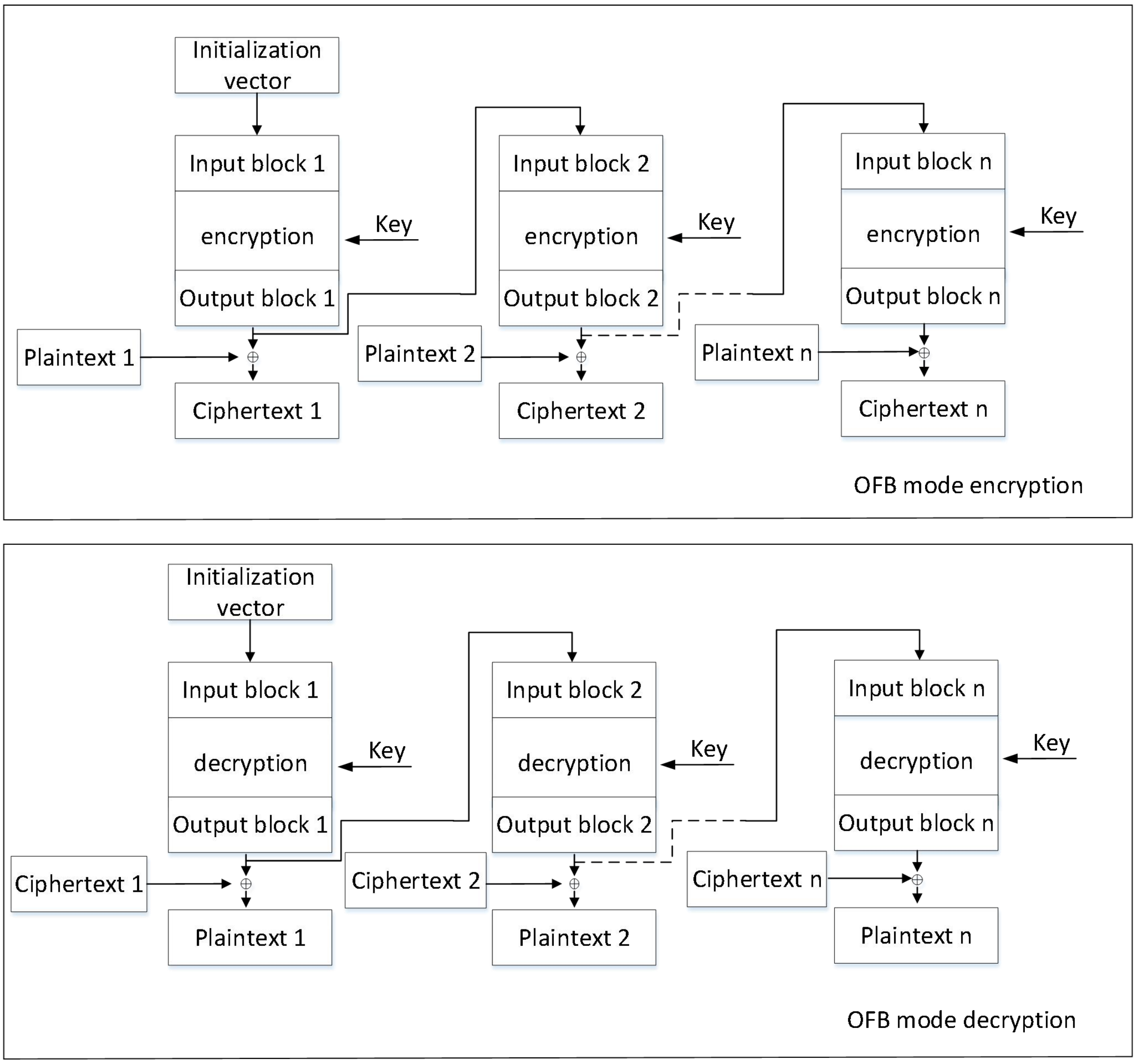
Figure 9-7 CFB Mode Encryption and Decryption



9.1.3.7 OFB Mode

The OFB mode is a confidentiality mode that features the iteration of the forward cipher on an IV to generate a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. If a same key is used, different IVs must be used to ensure operation security.

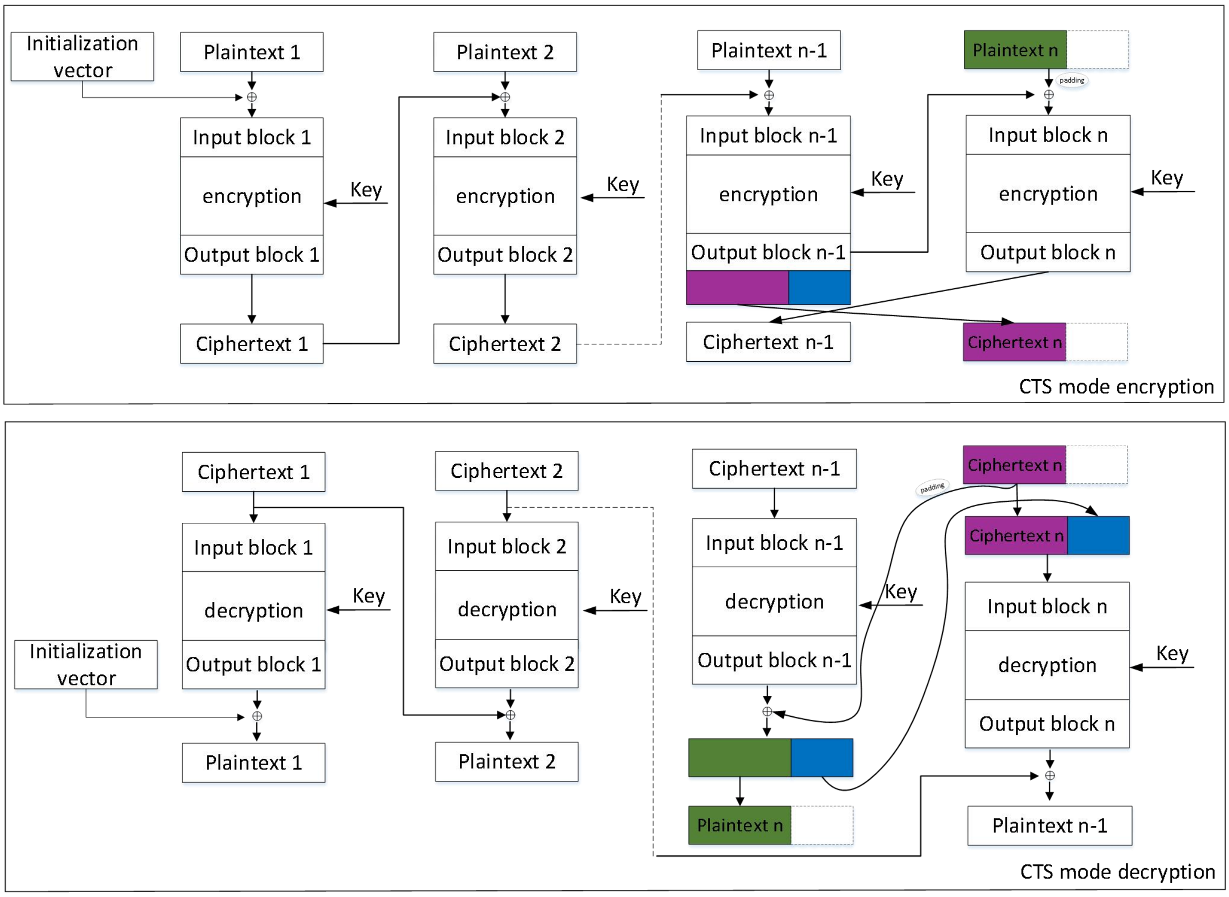
Figure 9-8 OFB Mode Encryption and Decryption



9.1.3.8 CTS Mode

The CTS mode is a confidentiality mode that accepts any plaintext input whose bit length is greater than or equal to the block size but not necessarily a multiple of the block size. Below are the diagrams for CTS encryption and decryption.

Figure 9-9 CTS Mode Encryption and Decryption



9.1.3.9 HASH Algorithm

The hash algorithms support MD5, SHA1, SHA224, SHA256, SHA384, SHA512, HMAC-SHA1, and HMAC-SHA256. All algorithms are iterative, one-way hash functions that can process a message to produce a condensed representation called a message digest. When a message is received, the message digest can be used to verify whether the data has changed, that is, to verify its integrity.

The hash algorithm of the CE supports block-aligned total length of the input data (padded by software), that is, a multiple of 64 bytes. The message length after padding by software is used as the configured data length for the hash algorithm.

9.1.3.10 RSA Algorithm

The RSA is a public key encryption/decryption algorithm implemented through the modular exponentiation operation.

The ciphertext is obtained as follows: C = ME mod N. The plaintext is obtained as follows: M = CD mod N.

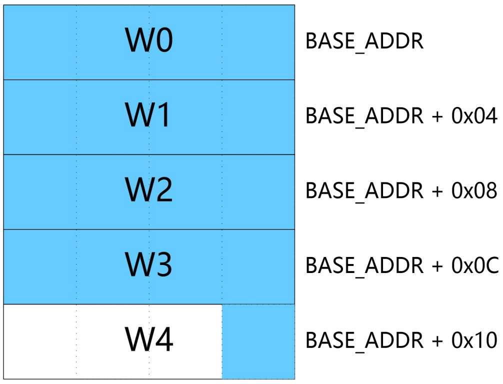
M indicates the plaintext, C indicates the ciphertext, (N, E) indicates the public key, and (N, D) indicates the private key.

9.1.3.11 Storing Message

In the application, a message may not be stored contiguously in the memory, but divided into multiple segments. Or a piece of continuously stored messages can be artificially split into multiple pieces as needs. Then each segment corresponds to a set of the source address and source length in the descriptor. Multiple segments correspond to groups 0-7 source address/source length in sequence.

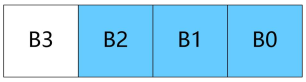
Each task supports up to 8 message segments, and the data volume of each message segment supports up to 4 GWord (AES-CTS is 1 GByte). The total amount of all segments in a task (that is a package) supports up to 4 GWord (AES-CTS is 1 GByte). If a message is divided into multiple packages, all others are required to be whole words; when the last package of AES-CTS is less than one word, 0 needs to be padded, and those less than one word are counted as one word. The following figure shows the address order structure.

Figure 9-10 Word Address of Message



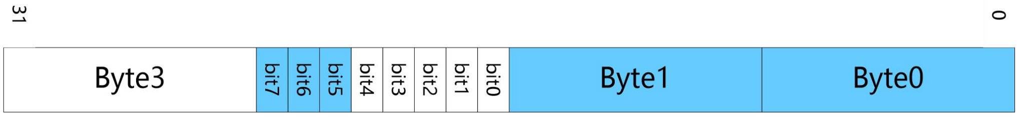
Byte order: low byte first, high byte last. When the data is less than one word, the low byte is filled first. The following figure shows the byte order structure (blue means it is filled by the message).

Figure 9-11 Byte Order



Bit order: high bit first, low bit last. When the data is less than one Byte, the high bit is filled first. The following figure shows the bit order structure.

Figure 9-12 Bit Order



9.1.3.12 Storing Key

The length of KEY must be an integer multiple of word.

9.1.3.13 Storing IV

For different algorithms, the length of IV is different. But they are integer multiples of word. To keep the byte order of IV and HASH digest output consistent, the byte order of IV is different from that of the message. For the multi-packet operation, the first address of the digest output result of the previous HASH can be directly configured to the first address of the next IV, and the software does not need to do any processing on the digest.

The following figure shows the storage method of 32-bit IV value.

Figure 9-13 The Storage Method of 32-bit IV

IV0[31:0]	BASE_ADDR
IV1[31:0]	BASE_ADDR + 0x04
……	……
IV7[31:0]	BASE_ADDR + 0x1C

The following figure shows the storage method of 64-bit IV value.

Figure 9-14 The Storage Method of 64-bit IV

IV0[63:32]	BASE_ADDR
IV0[31:00]	BASE_ADDR + 0x04
IV1[63:32]	BASE_ADDR + 0x08
IV1[31:00]	BASE_ADDR + 0x0C
……	……
IV7[63:32]	BASE_ADDR
IV7[31:00]	BASE_ADDR + 0x3C

9.1.3.14 Task Descriptor of Hash Algorithms and RBG Algorithms

The task descriptor is data written by software to a contiguous space in memory. The data describes the various properties of a task, such as algorithm type, mode, subcommand, key address, data source address, the data size read from data source, abstract destination address, the written destination data size, and the information of other tasks. First, we configure the task descriptor by software; then we operates the registers of CE to start this task. After the task starts, CE will read task descriptor based on the address of the task descriptor configured in register, and perform the task one time based on the described properties.

In applications, the “NEXT TASK ADDR” field can be configured as the starting address of the next task descriptor, to concatenate multi task descriptors into a task chain. After starting the first task, CE will perform every task in order until the “NEXT TASK ADDR” field is invalid (that is 0).

The HASH/RBG algorithms and Symmetrical/Asymmetrical algorithms use the different descriptor structure, separately.

Figure 9-15 Task Chaining of Hash Algorithms and Random Bit Generator Algorithms

0		IE		DLAV	LPKG		IVE		CHN
1	SUB CMD		RBG SEL		HME	HASH SEL			
2	Data Total Length (Address)(byte3-0)								
3	HMAC/PRNG KEY Address(byte2-0)	Data Total Length (Address)(byte4)							
4	IV Address(byte1-0)	HMAC/PRNG KEY Address(byte4-3)							
5		IV Address(byte4-2)							
6	SG0_word0								
7	SG0_word1								
8	SG0_word2								
9	SG0_word3								
10	SG0_word4								
	……								
41	SG7_word0								
42	SG7_word1								
43	SG7_word2								
44	SG7_word3								
45	SG7_word4								
46	Next SG Address(byte3-0)								
47	Next Task Address(byte2-0)	Next SG Address (byte4)							
48		Next Task Address(byte4-3)							
49	Reserved								
50	Reserved								
51	Reserved								

The detail structures are as follows.

No.	Descriptor	Name	Width	Description
0	CTRL	CHN	[1:0]	Channel ID
		IVE	[8]	IV mode enable, active high
		LPKG	[12]	1: Multi-SG enable. This bit needs to be fixed as 1.
		DLAV	[13]	Data length valid For last package, the bit needs be configured. For non last package, the bit needs not be configured. (Please configure it as 0 in PRNG/TRNG) 1: DLA means the WORD address where data total length (by bits) is saved. 0: DLA means the value of message total length (by bits).
		IE	[16]	Interrupt enable for current task, active high
1	CMD	HASH SEL	[3:0]	Hash algorithms select 0: MD5 1: SHA1 2: SHA224 3: SHA256 4: SHA384 5: SHA512 6: SM3 Other: Reserved
		HME	[4]	HMAC mode enable, active high
		RGB SEL	[11:8]	RGB algorithms select 0: No RGB use 1: PRNG 2: TRNG Other: Reserved
		SUB CMD	[31:16]	Sub-command in a specific algorithms When using PRNG, sub_cmd[15] means PRNG seed reload；sub_cmd[14:0] means PRNG linearly shifted seed
2	DLA	DLA	[31:0]	Data length OR its address. For last package, the field needs be configured. For non last package, the field needs not be configured. (Not used in PRNG/TRNG) When DLAV=1, here is the WORD address where data total length (by bits) is saved.

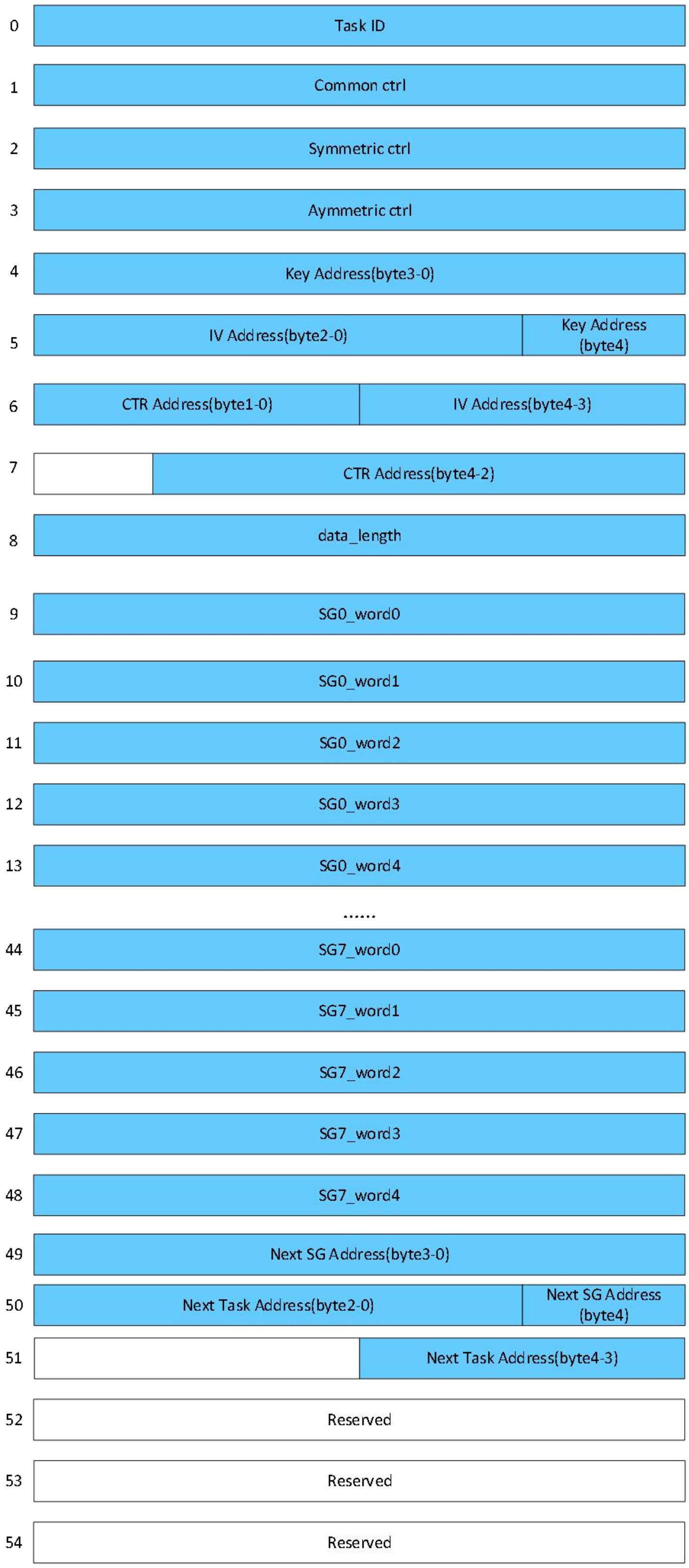
No.	Descriptor	Name	Width	Description
				When DLAV=0, here is the value of message total length (by bits)
3	DLA	DLA	[7:0]	When DLAV=1, here is the byte address bit[39:32] where data total length (by bits) is saved.
	KA	KA	[31:8]	KEY Address: The byte address bit[23:0].where HMAC KEY or PRNG KEY is saved.
4	KA	KA	[15:0]	KEY Address: The byte address bit[39:24].where HMAC KEY or PRNG KEY is saved.
	IVA	IVA	[31:16]	IV Address: The byte address bit[15:0] where IV is saved.
5	IVA	IVA	[23:0]	IV Address: The byte address bit[39:16] where IV is saved.
	Reversed	Reversed	[31:24]	/
6+5*x	SGx_W0	SGx_WORD0	[31:0]	Source Data Address x: The byte address bit[31:0] where Source Datax is saved.
7+5*x	SGx_W1	SGx_WORD1	[7:0]	Source Data Address x: The byte address bit[39:32] where Source Datax is saved.
			[31:8]	Output Data Address x: The byte address bit[23:0]where Output Datax to be saved.
8+5*x	SGx_W2	SGx_WORD2	[15:0]	Output Data Address x: The byte address bit[39:24]where Output Datax to be saved.
	Reversed	Reversed	[31:16]	/
9+5*x	SGx_W3	SGx_WORD3	[31:0]	Source Data length x: The Length (by bytes) of Source Datax.
10+5*x	SGx_W4	SGx_WORD4	[31:0]	Output Data length x: The Length (by bytes) of output Datax.
46	NSA	NSA	[31:0]	Next SG Address: The byte address bit[31:0].where the descriptor of the next 8 sg in a task is saved. If this is the only one group sg or the last group of a task, NSA must be 32'h0.
47	NSA	NSA	[7:0]	Next SG Address:

No.	Descriptor	Name	Width	Description
				The byte address bit[39:32].where the descriptor of the next 8 sg in a task is saved. If this is the only one group sg or the last group of a task, NSA must be 8’h0. The [38] indicate whether the next set of source sg exists.[39] bit indicate whether the next set of output（dst）sg exists.
	NTA	NTA	[31:8]	Next task Address: The byte address bit[23:0] where the descriptor of the next task in a task-chain is saved. If this is the only task or the last task of a task-chain, NTA must be 24’h0.
48	NTA	NTA	[7:0]	Next task Address: The byte address bit[39:24] where the descriptor of the next task in a task-chain is saved. If this is the only task or the last task of a task-chain, NTA must be 16’h0.
	Reversed	Reversed	[31:8]	/
49	Reversed	Reversed	/	/
50	Reversed	Reversed	/	/
51	Reversed	Reversed	/	/

9.1.3.15 Other Algorithms Task Descriptor

Software make request through task descriptor, providing algorithm type, mode, key address, source/destination sg address and size, etc. The task descriptor is as follows.

Figure 9-16 Task Chaining of Other Algorithms



0	Task ID	
1	Common ctrl	
2	Symmetric ctrl	
3	Aymmetric ctrl	
4	Key Address(byte3-0)	
5	IV Address(byte2-0)	Key Address (byte4)
6	CTR Address(byte1-0)	IV Address(byte4-3)
7		CTR Address(byte4-2)
8	data_length	
9	SG0_word0	
10	SG0_word1	
11	SG0_word2	
12	SG0_word3	
13	SG0_word4	
	......	
44	SG7_word0	
45	SG7_word1	
46	SG7_word2	
47	SG7_word3	
48	SG7_word4	
49	Next SG Address(byte3-0)	
50	Next Task Address(byte2-0)	Next SG Address (byte4)
51		Next Task Address(byte4-3)
52	Reserved	
53	Reserved	
54	Reserved	

Channel id supports 0-3 for each world.

- Common ctrl

Bit	Read/Write	Default	Description
31	R/W	0	interrupt enable for current task 0: disable interrupt 1: enable interrupt
30:25	/	/	/
24:17	R/W	0	cbc_mac_len the outcome bit length of CBC-MAC when in CBC-MAC mode. The part also be used as gcm/ocb mode tag_len.
16:9	/	/	/
8	R/W	0	OP DIR Algorithm Operation Direction 0: Encryption 1: Decryption
7	/	/	/
6:0	R/W	0	Algorithm type 0x0: AES 0x1: DES 0x2: Triple DES (3DES) 0x3：SM4 others: reserved 0x20: RSA 0x21: ECC 0x22: SM2 others: reserved 0x30: RAES Others: reserved

- Symmetric ctrl

Bit	Read/Write	Default	Description
31:30	/	/	/
29:28	R/W	0	SCK_SEL 0: use sck0/maskkey0 1: use sck1/maskkey1 2: use sck2/maskkey2 3: reserved
27:24	/	/	/
23:20	R/W	0	KEY Select key select for AES/SM4/TDES（TDES only configured as

Bit	Read/Write	Default	Description
			0/8-15) 0: Select input CE_KEYx (Normal Mode) 1: Select {SSK} 2: Select {HUK} 3: Select {RSSK}, used for decrypt EK, BSSK 4-7: Reserved 8-15: Select internal Key n (n from 0 to 7)
19:18	R/W	0	cfb_width For AES-CFB width 0: CFB1 1: CFB8 2: CFB64 3: CFB128
17	/	/	/
16	R/W	0	AES CTS last package flag When set to '1' , it means this is the last package for AES-CTS mode(the size of the last package >128bit). The part also be used as gcm/ocb mode gcm_last/ocb_last .
15:12	/	/	/
11:8	R/W	0	AES/DES/3DES/RAES modes. DES/3DES only supports ECB/CBC/CTR. RAES only supports ECB/CBC operation mode for symmetric 0: Electronic Code Book (ECB) mode 1: Cipher Block Chaining (CBC) mode 2: Counter (CTR) mode 3: CipherText Stealing (CTS) mode 4: Output feedback (OFB)mode 5: Cipher feedback (CFB)mode 6: CBC-MAC mode 7: OCB mode 8: GCM mode 9: Reserved Other: reserved
7:6	/	/	/
5:4	R/W	0	gcm_iv_mode[1:0] gcm_iv_mode[0]:value 1 show the last req for iv calculate gcm_iv_mode[1]: 0 :no GHASH calculate mode 1: GHASH calculate mode gcm_iv_mode[1:0]: 00: IDLE state ,this calculate do not have the process from iv to J0.

Bit	Read/Write	Default	Description
			01: by iv padding generating J0. On the mode ,iv padding is 96 bits, so iv_length will be 96bits. 10: by GHASH calculate for iv generating J0, and this is not the last req for iv calculate. 11: by GHASH calculate for iv generating J0, and this is the last req for iv calculate
3:2	R/W	0	CTR Width Counter Width for CTR Mode 0: 16-bits Counter 1: 32-bits Counter, gcm mode always use this setting without software 2: 64-bits Counter 3: 128-bits Counter
1:0	R/W	0	AES Key Size 0: 128-bits 1: 192-bits 2: 256-bits 3: Reserved

- Asymmetric ctrl

Bit	Read/Write	Default	Description
31:21	/	/	/
20:16	R/W	0	PKC algorithm mode. For modular computation: 00000: modular exponent(RSA) 00001: modular add 00010: modular minus 00011: modular multiplication others: reserved For ECC: 00000: point add 00001: point double 00010: point multiplication 00011: point verification 00100: encryption 00101: decryption 00110: sign 00111: sign verify others: reserved For SM2： 00000: encryption

Bit	Read/Write	Default	Description
			00001: decryption 00010: sign 00011: sign verify 00100: key exchange
15:8	/	/	/
7:0	R/W	0	Asymmetric algorithms operation width field. It indicates how much width this request apply, as words.

key addr field is address for each algorithm's key, also for extension feature micro codes address. (By byte)

ctr addr is address for next block's IV. (By byte)

src/dst sgX addr field indicate 40bits address for source and destination data. (By byte)

src/dst sgX size field indicates size for each sg respectively(by byte)

For SG, the detail as flow:

byte3	byte2	byte1	byte0	
SRC_ADDR0 [B3]	SRC_ADDR0 [B2]	SRC_ADDR0 [B1]	SRC_ADDR0 [B0]	SG_WORD0
DST_ADDR0 [B2]	DST_ADDR0 [B1]	DST_ADDR0 [B0]	SRC_ADDR0 [B4]	SG_WORD1
		DST_ADDR0 [B4]	DST_ADDR0 [B3]	SG_WORD2
SRC_SIZE0 [B3]	SRC_SIZE0 [B2]	SRC_SIZE0 [B1]	SRC_SIZE0 [B0]	SG_WORD3
DST_SIZE0 [B3]	DST_SIZE0 [B2]	DST_SIZE0 [B1]	DST_SIZE0 [B0]	SG_WORD4

1 group SG has 8 sg, each sg has 5 words, the ADDR is 40 bits and byte-addr; the SIZE is 32bits nad byte-unit. We will support unlimited SG number,but the 1860 just use for test. This can has many group SG in a task, using the next_sg_addr to create the new SG information in the task.

Next sg field should be set to 0 when no next group sg, else set to next sg's descriptor.

next task field should be set to 0 when no next task, else set to next task's descriptor.

9.1.3.16 PKC Microcode

PKC module supports RSA, ECC, SM2 algorithms in the form of microcode. It implements basic modular add, minus, multiplication, point add, point double, and logic computing, etc. Complete RSA/ECC/SM2 encryption, decryption, sign, verify are implemented with these microcode.

Asymmetric algorithms RSA/ECC/SM2 are implemented as microcode in PKC module. The encryption, decryption, sign, verify operations of asymmetric algorithms are composed with certain fixed microcode with hardware.

9.1.3.17 PKC Configuration

Before starting PKC, task description must be configured. Parameters to PKC are assigned to source sg, outcome is put to destination sg.

For RSA, parameters should be at the order of key, modulus, plaintext.

For ECC point add P2 = P0 + P1, parameters should be at the order of p, P0x, P0y, P1x, P1y. Output is at the order of P2x, P2y.

For ECC point double P2 = 2*P0, parameters should be at the order of p, a, P0x, P0y. Output is at the order of P2x, P2y.

For ECC point multiplication P2 = k*P0, parameters should be at the order of p, k, a, P0x, P0y. Output is at the order of P2x, P2y.

For ECC point verification, parameters should be at the order of p, a, P0x, P0y, b. Output is 1 or 0.

For ECC encryption, parameters should be at the order of random k, p, a, Gx, Gy, Qx, Qy, m. Output is at the order of Rx, Ry, c.

For ECC decryption, parameters should be at the order of random k, p, a, Rx, Ry, c. Output is m.

For ECC signature, parameters should be at the order of random k, p, a, Gx, Gy, n, d, e. Output is at the order of r, s.

For ECC signature verification, parameters should be at the order of n, s, e, r, p, a, Gx, Gy, Qx, Qy, n, r. Output is 1 or 0.

9.1.3.18 Error Check

After CE reads the task descriptor, CE can monitor error during algorithm operation. When the error is monitored, CE will do the following operations:

The task will pause immediately

Generates interrupt

The corresponding channel of the task status register is Fail

The corresponding channel bit of error status register can be read error number

The error number has the following types.

Code	Name	Description	Algorithms Type
0x01	algorithm not support	The algorithm type is not supported.	All
0x11	KEYSRAM access error	In AES decryption task, RSSK is used as plaintext, the DST address is not in	AES decryption

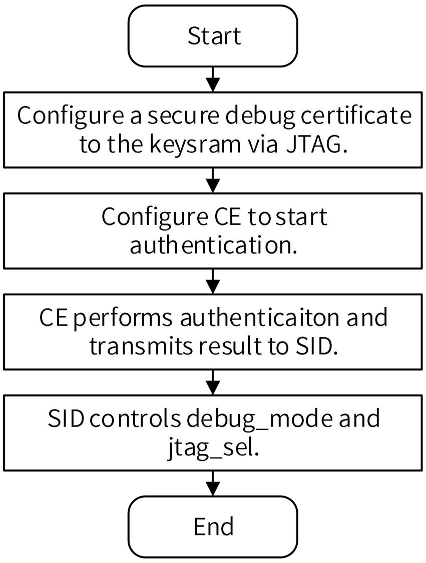
Code	Name	Description	Algorithms Type
		KEYSRAM space.	
0x21	key ladder configuration error	/	KL
0x31	data length error	Input size or output size configuration size error.	All

9.1.4 Programming Guidelines

9.1.4.1 Processing Secure Debug

The following figure shows the secure debug process.

Figure 9-17 Secure Debug Process



Start
Configure a secure debug certificate to the keysram via JTAG.
Configure CE to start authentication.
CE performs authenticaiton and transmits result to SID.
SID controls debug_mode and jtag_sel.
End

In secure debug process, CE mainly performs the following operatioins:

- Signature authentication
- Comparision of hash values of public key and chip_id
- Transmission of debug mode and transmission of authentification result

9.1.5 Register List

There are tree groups of registers in

Module Name	Base Address	Comments
CE_NS	0x03040000	Non-Security CE
CE_S	0x03040800	Security CE
SECURE_DEBUG_CFG	0x03042000	Secure Debug Configuration