atsha204a 加密芯片使用攻略——配置篇

大家好，这是接上一篇使用篇的博客，本篇文章主讲 atsha204a 加密芯片的配置方式，前面讲到，atsha204a 加密芯片内部 rom 分三个区域，一个是 config zone，一个是 slot zone，还有一个是 OTP zone，本篇不只讲解 config zone 的配置，还讲解 slot 和 OTP 区域的配置方法，并且结合官方的库来演示一下代码。

首先约定一下，本文所有的通信接口均使用 I2C 协议，但所有的功能都能在单总线上实现。

开始，我想先讲几个细节，这是刚开始对于接触这款芯片的我的一些困扰。

1、芯片的 config 区和 data 区一旦锁定，没有办法解锁，锁的方法只能通过 lock command 来进行锁定。

2、config 区在没锁定的时候，可以使用 write command 来进行写操作，但注意，0x00-0x03 地址(word 地址，详情请参考手册)不能被写，0x15word 地址不能使用 write command 来写。

3、在 config 锁定前，data 区(包括 slot 区和 OTP 区)既不能写也不能读。而在 config 锁定后，data 区锁定前，data 区只能写不能读，在 data 区锁定后，可以根据 config 中的配置来进行读写。

我之所以总结出这几点，是能让大家好理解这些区的读写属性，接下来进入正题。

1、通信包的格式

atsha204a 的所有发送的数据包均是以下格式.

1Byte	1byte	N byte	2byte
Word address	count	data	CRC16

word address 指示这个数据包是为何作用，有如下值

Table 6-2. Word Address Values

Name	Value	Description
Reset	0x00	Reset the address counter. The next read or write transaction will start with the beginning of the I/O buffer.
Sleep (Low Power)	0x01	The ATSHA204A goes into the low-power sleep mode and ignores all subsequent I/O transitions until the next Wake flag. The entire volatile state of the device is reset.
Idle	0x02	The ATSHA204A goes into the idle state and ignores all subsequent I/O transitions until the next Wake flag. The contents of TempKey and RNG Seed registers are retained.
Command	0x03	Write subsequent bytes to sequential addresses in the input command buffer that follow previous writes. This is the normal operation.
Reserved	0x04 – 0xFF	These addresses should not be sent to the device.

最常用的就是 0x03 命令数据包了

count 是数据包的长度，包括 count 本身，data 和 CRC16。

data 是数据内容，根据命令的不同而不同。

CRC16 是数据包的校验码，校验内容从 count 到 data。

再来看看配置区 config 的内容

Table 2-2. Configuration Zone

Word	Byte 0	Byte 1	Byte 2	Byte 3	Default	Write Access	Read Access
0x00	Serial Number[0:3]				01 23 xx xx	Never	Always
0x01	Revision Number				xx xx xx xx	Never	Always
0x02	Serial Number[4:7]				xx xx xx xx	Never	Always
0x03	SN[8]	Reserved	I2C Enable	Reserved	EE 55 xx 00	Never	Always
0x04	I2C Address	CheckMacConfig	OTP Mode	Selector Mode	C8 00 55 00	If Config is unlocked	Always
0x05	Slot Configuration 0		Slot Configuration 1		8F 80 80 A1	If Config is unlocked	Always
0x06	Slot Configuration 2		Slot Configuration 3		82 E0 A3 60	If Config is unlocked	Always
0x07	Slot Configuration 4		Slot Configuration 5		94 40 A0 85	If Config is unlocked	Always
0x08	Slot Configuration 6		Slot Configuration 7		86 40 87 07	If Config is unlocked	Always
0x09	Slot Configuration 8		Slot Configuration 9		0F 00 89 F2	If Config is unlocked	Always
0x0A	Slot Configuration 10		Slot Configuration 11		8A 7A 0B 8B	If Config is unlocked	Always
0x0B	Slot Configuration 12		Slot Configuration 13		0C 4C DD 4D	If Config is unlocked	Always
0x0C	Slot Configuration 14		Slot Configuration 15		C2 42 AF 8F	If Config is unlocked	Always
0x0D	Use Flag 0	Update Count 0	Use Flag 1	Update Count 1	FF 00 FF 00	If Config is unlocked	Always
0x0E	Use Flag 2	Update Count 2	Use Flag 3	Update Count 3	FF 00 FF 00	If Config is unlocked	Always
0x0F	Use Flag 4	Update Count 4	Use Flag 5	Update Count 5	FF 00 FF 00	If Config is unlocked	Always
0x10	Use Flag 6	Update Count 6	Use Flag 7	Update Count 7	FF 00 FF 00	If Config is unlocked	Always
0x11	Last Key Use 0	Last Key Use 1	Last Key Use 2	Last Key Use 3	FF FF FF FF	If Config is unlocked	Always
0x12	Last Key Use 4	Last Key Use 5	Last Key Use 6	Last Key Use 7	FF FF FF FF	If Config is unlocked	Always
0x13	Last Key Use 8	Last Key Use 9	Last Key Use 10	Last Key Use 11	FF FF FF FF	If Config is unlocked	Always
0x14	Last Key Use 12	Last Key Use 13	Last Key Use 14	Last Key Use 15	FF FF FF FF	If Config is unlocked	Always
0x15	User Extra	Selector	Lock Data	Lock Config	00 00 55 55	Via Update Extra Command Only	Always

对 atsha204a 的所有地址操作都是基于 word 地址进行的，如上图所示。

好了，现在我们先来看下读命令格式(也就是数据包 data 域里面的具体内容)

Table 8-33. Input Parameters

	Name	Size	Notes
Opcode	READ	1	0x02
Param1	Zone	1	Bits 0 and 1: Select among config, OTP, or data. See Section 8.5.3, "Zone Encoding". Bits 2-6: Must be zero. Bit 7: If one, 32 bytes are read; otherwise four bytes are read. Must be zero if reading from OTP zone.
Param2	Address	2	Address of first word to be read within the zone. See Section 8.5.4, "Address Encoding".
Data	—	0	—

第一个字节是操作码，read command 的操作码是 0x02，

第二个字节的最高位 bit7 指示是读取 32 字节还是 4 字节，1 为读取 32 字节。bit0 指示读哪个区。

第三个字节是 Address 地址。

关于这里的地址，datasheet 手册上写着如下

8.5.4 Address Encoding

Param2 includes a single address that indicates the memory to be accessed. All Reads and Writes are in units of Words (4-byte). The most-significant byte of a legal ATSHA204A address is always zero. All unused address bits should always be set to zero. The least-significant bits in the address describe the offset to the first word to be accessed within the Block/Slot, while the upper bits specify the Slot number per the table below:

Table 8-6. Address Encoding (Param2)

	Byte 0 (First Byte on the Bus)								Byte 1							
Zone	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Data	0	Block				Offset			0	0	0	0	0	0	0	0
Config	0	0	0	Block		Offset			0	0	0	0	0	0	0	0
OTP	0	0	0	0	Block	Offset			0	0	0	0	0	0	0	0

读写命令中 Param2 的这个 Address 如上，高 8 位为 0，低 8 位的前 3 位是偏移量，高 5 位是 block，一个 block 为 32 字节，所以 config 分 3 个 block(0-3)，slot 分 16 个 block(0-15)，otp 分 2 个 block(0-1)，如果是使用 32 字节进行读写的话，则忽略前 3 位 offset。我们举个例子，比如说，我们要读 config 区的 word 地址 0x09 那 4 个字节，因为

0x09 是属于 block1 的 ,偏移是 1(config 的 block 划分是,word 0x00 -0x07 是 block,0，0x08-0x0F 为 block1，0x10-0x15 是 block2，反正就是 32 个字节为 1 个 block，slot 和 otp 相同）

Address[4:3] = 1;

如果是 4 字节读取，那么

Address[2:0] = 1；

如果是 32 字节读取

则忽略 Address[2:0] 。

现在我们来举个例子，说说通信的数据包。

假设我们要读取 config 的 word0 地址那 4 个字节，那么数据包格式如下：

Word Address	Count	Opcode	Param1	Param2	CRC16
0x03	0x06	0x02	0x00	0x00	2 字节

如果通信正常，那么芯片就会返回数据，返回的数据包格式如下

1byte	N byte	2 byte
Count	data	CRC16

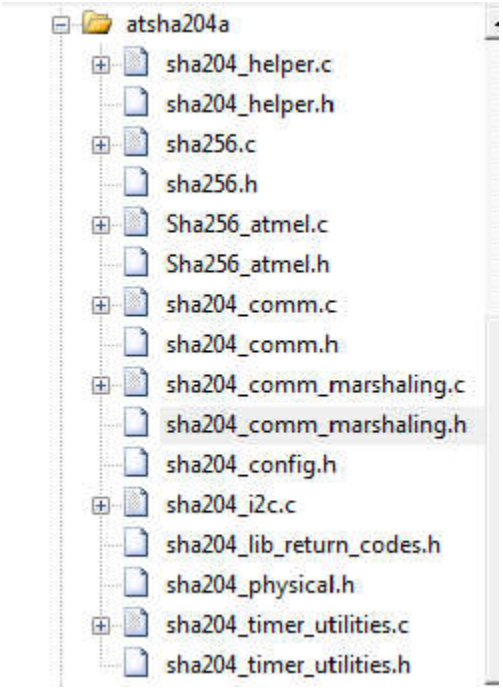
各数据域的意思与发送的数据一样。

根据刚才的例子，返回的数据就是如下:

Count	Byte0	Byte1	Byte2	Byte3	CRC16
0x07	0x12	0x23	xx	xx	2 字节

因我们读的是芯片序列号，xx 是每块芯片的唯一序列号一部分。

这里仅仅是演示了读的命令，如果是其他命令，则数据域会更加复杂，所幸 ATMEL 有专门的库来封装这些操作，官方库文件如下列表：



sha204_comm_marshling.h 和 sha204_comm_marshling.c 这两个文件里面的

uint8_t sha204m_execute(

uint8_t op_code, uint8_t param1, uint16_t param2,

uint8_t datalen1, uint8_t *data1, uint8_t datalen2,

uint8_t *data2, uint8_t datalen3, uint8_t *data3,

uint8_t tx_size, uint8_t *tx_buffer, uint8_t rx_size,

uint8_t *rx_buffer);

这个函数封装了所有命令 ,使用起来非常方面 ,具体实现的话可以去下载库的源码进行查看，而移植性的文件则是 sha204_i2c.c、sha204_physical.h、sha_timer_utilities.h 和 sha_timer_utilities.c 这几个文件，做移植只需要简单修改里面几个关于 i2c 通信和延时的函数即可。

本节以一个简单的例子来讲述了命令的基本格式，并简单的介绍了官方库文件的结构，有问题的可以私信。

config 区的配置

这节是这篇博客的重点，只有正确配置了 config，atsha204a 芯片才能按自己的意图办事。

我再放一下 config zone 的配置表

Table 2-2. Configuration Zone

Word	Byte 0	Byte 1	Byte 2	Byte 3	Default	Write Access	Read Access
0x00	Serial Number[0:3]				01 23 xx xx	Never	Always
0x01	Revision Number				xx xx xx xx	Never	Always
0x02	Serial Number[4:7]				xx xx xx xx	Never	Always
0x03	SN[8]	Reserved	I2C Enable	Reserved	EE 55 xx 00	Never	Always
0x04	I2C Address	CheckMacConfig	OTP Mode	Selector Mode	C8 00 55 00	If Config Is unlocked	Always
0x05	Slot Configuration 0		Slot Configuration 1		8F 80 80 A1	If Config Is unlocked	Always
0x06	Slot Configuration 2		Slot Configuration 3		82 E0 A3 60	If Config Is unlocked	Always
0x07	Slot Configuration 4		Slot Configuration 5		94 40 A0 85	If Config Is unlocked	Always
0x08	Slot Configuration 6		Slot Configuration 7		86 40 87 07	If Config Is unlocked	Always
0x09	Slot Configuration 8		Slot Configuration 9		0F 00 89 F2	If Config Is unlocked	Always
0x0A	Slot Configuration 10		Slot Configuration 11		8A 7A 0B 8B	If Config Is unlocked	Always
0x0B	Slot Configuration 12		Slot Configuration 13		0C 4C DD 4D	If Config Is unlocked	Always
0x0C	Slot Configuration 14		Slot Configuration 15		C2 42 AF 8F	If Config Is unlocked	Always
0x0D	Use Flag 0	Update Count 0	Use Flag 1	Update Count 1	FF 00 FF 00	If Config Is unlocked	Always
0x0E	Use Flag 2	Update Count 2	Use Flag 3	Update Count 3	FF 00 FF 00	If Config Is unlocked	Always
0x0F	Use Flag 4	Update Count 4	Use Flag 5	Update Count 5	FF 00 FF 00	If Config Is unlocked	Always
0x10	Use Flag 6	Update Count 6	Use Flag 7	Update Count 7	FF 00 FF 00	If Config Is unlocked	Always
0x11	Last Key Use 0	Last Key Use 1	Last Key Use 2	Last Key Use 3	FF FF FF FF	If Config Is unlocked	Always
0x12	Last Key Use 4	Last Key Use 5	Last Key Use 6	Last Key Use 7	FF FF FF FF	If Config Is unlocked	Always
0x13	Last Key Use 8	Last Key Use 9	Last Key Use 10	Last Key Use 11	FF FF FF FF	If Config Is unlocked	Always
0x14	Last Key Use 12	Last Key Use 13	Last Key Use 14	Last Key Use 15	FF FF FF FF	If Config Is unlocked	Always
0x15	User Extra	Selector	Lock Data	Lock Config	00 00 55 55	Via Update Extra Command Only	Always

以下我讲的地址全部是 word 地址

配置区地址 0x00-0x03 是不允许写的，只能读。

从 0x04 开始讲起，首先是 I2C Address，可以设置 I2C 的器件地址，如果 I2C 总线上挂了好几个 atsha204a，可以修改一下这个地址，然后是 CheckMacConfig，文档介绍如下

2.1.2.3 CheckMacConfig

This byte applies only to the `CheckMac`, `Read`, and `Write` commands:

- **Read and Write**
  Bit 0 controls Slots 0 and 1, bit 1 controls Slots 2 and 3, and so on. Any encrypted Read/Write command will fail if the value in TempKey.SourceFlag does not match the corresponding bit in this byte. This byte is ignored for clear text reads and writes.
- **CheckMac**
  Bit 0 controls slot 1, bit 1 controls Slot 3 and so on. The copy function will only be enabled if the CheckMacSource value corresponding to the target slot matches the value of Mode bit 2 of the CheckMac command. The command will fail if Mode bit 2 does not match TempKey.SourceFlag, so this is equivalent to requiring the corresponding bit in this byte to match TempKey.SourceFlag.

这个字节理解起来挺费劲的，这个字节只影响读写命令和 CheckMac 命令，对于 Read 和 Write 命令，在进行加密读\写的时候要匹配 TempKey 中的 SourceFlag 相应的位，否则加密读\写将会失败，什么是加密读写？在后面会介绍到，而对于 CheckMac 命令也是类似，可以参照一下文档具体使用。

接下来是 OTP Mode

文档描述如下

2.1.2.4 OTP Mode

0xAA (Read-only mode) = When OTP zone is locked, writes are disabled and reads of all words are permitted.

0x55 (Consumption mode) = Writes to the OTP zone when the OTP zone is locked will causes the bits to transition only from a one to a zero. Reads of all words are permitted.

0x00 (Legacy mode) = When OTP zone is locked, writes are disabled, reads of Words 0 and 1, and 32-byte reads are disabled.

All other modes are reserved.

OTP Mode = 0xAA 只读模式，在 OTP 锁定后，只能读取 OTP 区中的内容

OTP Mode = 0x55 消耗模式，在 OTP 锁定后，写操作只能使位 1 变 0，不能再变为 1，比如原先的内容是 0xFF ,写入 0xFC ,就变为 0xFC ,再写 0xFF ,则仍然是 0xFC ,写入 0x00 则立即清零并且不能再写其他值(这个过程有点像 flash 的烧写)。

OTP Mode = 0x00 传统模式，在 OTP 锁定后，不能读 OTP0 和 OTP1，也不能以 32 字节读取。

其他值不被允许。

Selector Mode，这个字节定义 0x15 地址上的 Selector 更新方式，

Selector Mode = 0x00 ，使用 UpdateExtra 命令可以随意更新 Selector 中的值，

Selector Mode = 其他值，只有当 Selector 中的值为 0 时，才能被 UpdateExtra 命令命令更新。

这里顺便说一下 Selector，当执行 Pause Command 命令的时候，Pause Command 命令参数中指定的 selector 与配置中的 selector 相等时，器件将不会进入空闲模式。

对于一般用户，0x04 地址的 4 个配置按默认值设置即可。

接下来是地址 0x05-0x0C 关于 slot 的配置了，这里才是这节的重点，也是需要注意最多的地方。

每个 slot config 占用 2 个字节，每个位具体配置如下

Table 2-3. SlotConfig Bits (Per Slot)

Bit	Name	Description
0 – 3	ReadKey	Slot of the key to be used for encrypted reads. If 0x0, then this slot can be used as the source slot for the CheckMac Copy Command.
4	CheckOnly	0 = This slot can be used for all crypto commands. 1 = This slot can only be used for CheckMac and GenDig followed by CheckMac Commands.
5	SingleUse	0 = No limit on the number of time the key can be used. 1 = Limit on the number of time the key can be used based on the UseFlag (or LastKeyUse) for the slot.
6	EncryptRead	0 = Clear reads are permitted. 1 = Requires the slot to be Secret and encrypted read to access.
7	IsSecret	0 = The slot is not secret and allows clear read, clear write, no MAC check, and no Derivekey Command. 1 = The slot is secret. Reads and writes if allowed, must be encrypted.
8 – 11	WriteKey	Slot of the key to be used to validate encrypted writes.
12 – 15	Write Config	See detailed function definition for use.

Table 2-4. Write Configuration Bits — Derivekey Command

Bit 15	Bit 14	Bit 13	Bit 12	Source Key(1)	Description
0	X	1	0	Target	DeriveKey command can be run without authorizing MAC (Roll).
1	X	1	0	Target	Authorizing MAC required for DeriveKey command (Roll).
0	X	1	1	Parent	DeriveKey command can be run without authorizing MAC (Create).
1	X	1	1	Parent	Authorizing MAC required for DeriveKey command (Create).
X	X	0	X	—	Slots with this value in the WriteConfig field may not be used as the target of the DeriveKey command.

Note: 1. The source key for the computation performed by the DeriveKey command can either be the key directly specified in Param2 (the "Target") or the key at slotConfig[Param2].WriteKey (the "Parent"). See Section 13.3, "Key Values" for more details.

Table 2-5. Write Configuration Bits — Write Command

Bit 15	Bit 14	Bit 13	Mode Name	Description
0	0	0	Always	Clear text writes are always permitted on this slot. Slots set to "always" should never be used as key storage. Either 4 or 32 bytes may be written to this slot.
X	0	1	Never	Writes are never permitted on this slot using the Write command Slots set to "never" can still be used as key storage.
1	0	X	Never	Writes are never permitted on this slot using the Write command Slots set to "never" can still be used as key storage.
X	1	X	Encrypt	Writes to this slot require a properly computed MAC, and the input data must be encrypted by the system with WriteKey using the encryption algorithm documented in the Write command description (Section 8.5.17, "UpdateExtra Command"). 4-byte writes to this slot are prohibited.

Table 2-4. Write Configuration Bits — Derivekey Command

Bit 15	Bit 14	Bit 13	Bit 12	Source Key(1)	Description
0	X	1	0	Target	DeriveKey command can be run without authorizing MAC (Roll).
1	X	1	0	Target	Authorizing MAC required for DeriveKey command (Roll).
0	X	1	1	Parent	DeriveKey command can be run without authorizing MAC (Create).
1	X	1	1	Parent	Authorizing MAC required for DeriveKey command (Create).
X	X	0	X	—	Slots with this value in the WriteConfig field may not be used as the target of the DeriveKey command.

Note: 1. The source key for the computation performed by the DeriveKey command can either be the key directly specified in Param2 (the "Target") or the key at slotConfig[Param2].WriteKey (the "Parent"). See Section 13.3, "Key Values" for more details.

Table 2-5. Write Configuration Bits — Write Command

Bit 15	Bit 14	Bit 13	Mode Name	Description
0	0	0	Always	Clear text writes are always permitted on this slot. Slots set to "always" should never be used as key storage. Either 4 or 32 bytes may be written to this slot.
X	0	1	Never	Writes are never permitted on this slot using the Write command Slots set to "never" can still be used as key storage.
1	0	X	Never	Writes are never permitted on this slot using the Write command Slots set to "never" can still be used as key storage.
X	1	X	Encrypt	Writes to this slot require a properly computed MAC, and the input data must be encrypted by the system with WriteKey using the encryption algorithm documented in the Write command description (Section 8.5.17, "UpdateExtra Command"). 4-byte writes to this slot are prohibited.

一位一位的说

bit0-3：指定加密读的 slot 区，这里的意思是，如果你把 slotX 设置成可以加密读取，则这里要指定加密读取的密钥存放的 slot，简单来说，你要读一个 slotX，而 slot config[X]已经被配置成可以加密读，但是需要知道 slot config[X].ReadKey 中指定的 slot 区的密钥。

bit4：0：这个 slot 区可以用于所有加密命令，1：这个 slot 只能用于 CheckMac 命令和 GenDig 命令。

bit5：0：此 slot 中存放的密钥可以无限使用，1：此 slot 存放的密钥有限使用，次数根据 UseFlag 或 LastKeyFlag 指定。

bit6：0:可以明文读取，1：必须加密读取。

bit7：0:此 slot 不是密钥区，可以随意读写，1:此 slot 作为密钥区，如果能读写则必须是加密进行。

说到这里大家先看一张表格

Table 8-35. Read Operation Permission

IsSecret	EncryptRead	Description
0	0	Clear text reads are always permitted from this slot. Slots set to this state should never be used as key storage. Either 4 or 32 bytes may be read at a time.
0	1	Prohibited. No security is guaranteed for slots using this code.
1	0	Reads are never permitted from this slot. Slots set to this state can still be used for key storage.
1	1	Reads from this slot are encrypted using the encryption algorithm documented in the Read command description (See Section 8.5.15, "Read Command"). The encryption key is in the slot specified by ReadKey. 4-byte reads and writes are prohibited.

可以清楚看到 IsSecret 和 EncryptRead 位的配置对 slot 读操作的影响。

IsSecret = 1；EncryptRead = 0；slot 不能被读。

IsSecret = 1；EncryptRead = 1；slot 能被加密读。

IsSecret = 0；EncryptRead = 0；随意读。

说回 slot config 配置，bit8-bit11，指定加密写操作的 slot 区密钥，跟 bit0-3 类似，只不

过这里是写操作，使用 write command 命令。

bit12-15 Write Config 表，上述 Write Config 那张表看起来有点费劲，bit12-15 对于 DeriveKey Command 和 Write command 命令有不同的意思，对于 DeriveKey Command，就是决定生成的新的密钥要通过什么途径来获取(Target?,Parent?)，具体大家可以看我上一篇使用篇中对于 DeriveKey Command 的描述。而对于 Write Command，定义如下

bit15 = 0;bit14 = 0;bit13 =0; 随意写

bit15 = x;bit14 = 0;bit13 =1; 不能写

bit15 = 1;bit14 = 0;bit13 =x; 不能写

bit15 = x;bit14 = 1;bit13 =x; 加密写

到此大致介绍完了 slot 区的配置，这里举个最简单的例子，如下

现在要把 slot0 和 slot1 设置成密钥区，不能进行任何读写，可以执行所有加密命令，无限次使用，则地址 word0x04 的配置应该是 0x80 0xA0 0x80 0xA0，

根据手册，得知 write command 命令如下

Table 8-40. Input Parameters

	Name	Size	Notes
Opcode	Write	1	0x12
Param1	Zone	1	Bits 0 and 1: Select among config, OTP or data. See Section 8.5.3, "Zone Encoding". Bits 2-5: Must be zero. Bit 6: If one, the input data must be encrypted. Must be zero if Data/OTP zones are locked. Bit 7: If one, 32 bytes will be written; otherwise, four bytes are written.
Param2	Address	2	Address of first word to be written within the zone. See Section 8.5.4, "Address Encoding".
Data_1	Value	4 or 32	Information to be written to the zone; may be encrypted.
Data_2	Mac	0 or 32	Message authentication code to validate address and data.

我们要写 config zone，地址是 0x04，4 字节写入，所以填充如下

opcode = 0x12,

zone = 0x00,

address = 0x04,

value = 0x80,0xA0,0x80,0xA0

数据包为如下格式

Word address	Count	Opcode	Param1	Param2	Data_1	CRC16
0x03	0x0A	0x12	0x00	0x05	0x80,0xA0,0x80,0xA0	2 字节

Data_2 的 MAC 域在进行加密写操作的时候才会有内容，这里不进行加密写操作，所以上述数据包没有这个字段，大家如果使用官方封装的库，可以使用 sha204m_execute 函数

函数原型如下：

uint8_t sha204m_execute(uint8_t op_code, uint8_t param1, uint16_t param2,

uint8_t datalen1, uint8_t *data1, uint8_t datalen2, uint8_t *data2, uint8_t datalen3,

uint8_t *data3,

uint8_t tx_size, uint8_t *tx_buffer, uint8_t rx_size, uint8_t *rx_buffer);

这个函数参数很多，主要是考虑到各种命令的各种不同内容，按上面的描述则调用这个函数时，具体内容为：

state = sha204m_execute(

0x12,

0x00,

0x05,

4,*yourbuffer,

0,0,0,0,

SHA204_CMD_SIZE_MIN,

txbuf,

WRITE_RSP_SIZE,

rxbuf

);

当然这是一段伪代码，yourbuffer是data_1:value的存放地方，SHA204_CMD_SIZE_MIN 和 READ_32_RSP_SIZE 在库中都有定义，实际上在这个函数实现内部 SHA204_CMD_SIZE_MIN 和 WRITE_RSP_SIZE 指定的大小均不起作用，在函数内部自动根据命令来调节收发数据包的大小，而用户只需要给定足够大的 txbuf 和 rxbuf 即可。

接下来是 UseFlag，这里 UseFlag 里面的值指明了对应的 slot 区(只限 slot0-7)中的 key 还能使用多少次，但前提是 slot config.SingleUse = 1，UseFlag 才会有用。在原 datasheet 的 78 页里面详细介绍了 single-use key 的用法，我截个图给大家看一下

13.3.4 Single-use Keys

For the SlotID values corresponding to slots 0 thru 7 in the data section of the EEPROM, repeated usage of the key stored in the slot can be strictly limited. This feature is enabled if the SingleUse bit is set in the SlotConfig field. The SingleUse bit is ignored for slots 8 thru 14. The number of remaining uses is stored as a bit map in the UseFlag byte corresponding to the slot in question.

Prior to execution of any cryptographic command that uses this slot as a key, the following takes place:

- If SlotConfig[SlotID].SingleUse is set and UseFlag[SlotID] is `0x00`, the device returns an error.
- Starting at bit 7 of UseFlag[SlotID], clear to zero the first bit that is currently a one.

In practice, this procedure permits SingleUse keys to be used eight times between “refreshes” using the `DeriveKey` command. If power is lost during the execution of any command referencing a key that has this feature enabled, one of the use bits in UseFlag may still be cleared even though the command did not complete. For this reason, Atmel recommends that the key be used a single time only, with the other bits providing a safety margin for errors.

Under normal circumstances, all eight UseFlag bytes should be initialized to `0xFF`. If it is the intention to permit fewer than eight uses of a particular key, these bytes should be initialized to `0x7F` (seven uses), `0x3F` (six uses), `0x1F` (five uses), `0x0F` (four uses), `0x07` (three uses), `0x03` (two uses), or `0x01` (one use). Initialization to any other value besides these values or 0xFF is prohibited.

大概的意思就是，slot0-7 才有 useflag 这功能，slot8-14 没有（slot15 比较特殊）。如果相应 slot config.singleuse = 1，那么当执行一些需要用到此 slot 的加密命令时而 UseFlag[SlotID] = 0x00，那么这些命令就会返回错误。UseFlag 的计数方式简单的说就是这个 8 位数位为 1 的个数，比如 0xFF 是八次，0x7F 是 7 次，0x01 是 1 次，如此类推。当执行了一些加密命令时，UseFlag 就会从最高位开始置零，DeriveKey 命令可以更新 UseFlag 中的值，关于 DeriveKey 命令，大家可以看一下我上一篇使用篇的博客，因篇幅问题，不再讨论。

UpdateCount 是对应的 slot 使用 DeriveKey 更新了多少次，到了 0xFF 就回滚至 0x00。

然后是 LastKeyUse0-15，首先注意，这个”Last”不是上一个的意思，而是最后一个的意思，也就是说，这 LastKeyUse0-15 都是为 slot15 服务的，真是厉害，其实 LastKeyUse 使用起来和 UseFlag 差不多，也是要 slot15 config.singleuse = 1 才有效，而计算剩余使用次数的方法是，所有 LastKeyUse 位 1 的个数加起来，那么总共就有 16*8 = 128 次了，不过用完后，slot15 就彻底完了，不能再使用，无法被 DeriveKey 更新，这也是 slot15 比较特殊的地方。

地址 0x15 不能被 write command 写，只能通过其他命令，关于 UserExtra，官方源文档中说了这个可以被 UpdateExtra Command 更新，它的用途主要是给用户使用的，其他用途不明，如果有其他人知道这个字节有什么用，也希望告知，谢谢。

Selector 前文中提了一下：执行 Pause Command 命令的时候，Pause Command 命令参数中指定的 selector 与配置中的 selector 相等时，器件将不会进入空闲模式。也可以被 UpdateExtra Command 更新，

UpdateExtra Command 命令格式如下：

Table 8-38. Input Parameters

	Name	Size	Notes
Opcode	UPDATEEXTRA	1	0x20.
Param1	Mode	1	Bit 0: If zero, update Config byte 84. If one, update Config byte 85. Bit 1: If one, ignore bit 0 and decrement the limited use counter associated with the key in slot "NewValue". If zero, update Config byte 84 or 85. Bits 2 – 7: Must be zero.
Param2	NewValue	2	LSB: Value to optionally be written to location 84 or 85 in Configuration zone. MSB: Must be 0x00.
Data	—	0	—

Table 8-39. Output Parameter

Name	Size	Notes
Success	1	If the memory byte was updated, this command returns a value of 0x00; otherwise, it returns an Execution error.

参数一目了然，不再多说，这里只需要注意，

1、Mode.bit1 = 1 时，这条命令做的操作是，减少 NewValue 中指定的 SlotID 的使用次数。

2、UserExtra 中的值不为 0，那么此条命令如果更新 UserExtra 的话将会返回失败。

3、如果 SelectorMode 不为 0 ,而且 Selector 不为 0 ,那么那么此条命令如果更新 Selector 的话将会返回失败。

Lock Data 和 Lock Config 是控制锁定数据区和配置区的两个字节，

LockData = 0x55，data 区没有锁定可以进行写操作。

LockData = 0x00，data 区进行锁定，读写需要根据配置来进行。

LockConfig =0x55，config 区没有锁定可以进行读写操作。

LockConfig = 0x00，config 区不能写。

这两个字节必须使用 LockCommand 来设置，LockCommand 如下：

Table 8-20. Input Parameters

	Name	Size	Notes
Opcode	LOCK	1	0x17.
Param1	Zone	1	Bit 0: Zero for Configuration zone, one for Data and OTP zones. Bits 1-6: Must be zero. Bit 7: If one, the check of the zone CRC is ignored and the zone is locked, regardless of the state of the memory. Atmel does *not* recommend using this mode.
Param2	Summary	2	Summary of the designated zones, or should be 0x0000 if Zone[7] is set.
Data	—	0	—

Table 8-21. Output Parameter

Name	Size	Notes
Success	1	Upon successful execution, the ATSHA204A returns a value of zero.

Zone.bit0 决定的是锁 config 还是 OTP 区，Zone.bit7 = 1，则不计算需要锁定区域的 CRC16 值，否则需要输入一个 CRC16 值来校验。

Summary 如果 Zone.bit7=0，那么就这里就需要填入相应锁定区域的 CRC16 值，否则填 0x0000。如果命令成功就返回 0。

本节介绍了 config zone 的配置，用户可以根据自己的需要来对 config 进行配置，强烈建议用户去多次阅读 datasheet 才能深刻理解。

slot 区的配置

此区的配置没那么复杂，slot 主要意图是存放密钥，当然也可以设置成 eeprom 来使用，但一般用户都不会这么做。

slot 区一共分 16 部分，slot0-15，每部分可以存放 32 字节密钥，相信足够使用。

对 slot 区的配置无外乎读写内容，在 config 区锁定前，所有对 slot 区的读写操作均返回错误。在 config 区锁定后，data 区锁定前，只能写 slot 区而不能读取，在二者都锁定后，可以根据上节所讲的 slot config 来进行读写。

假设我们配置完了 config 区，现在我们来写 slot 区，很简单，在 data 区锁定前可以随意读写，但强烈建议使用 32 字节来写操作，采用 4 字节写操作，会有很多的限制条件。

假如我们对 slot0 写入 0x11,0x11,.....0x11（共 32 个 0x11），使用 write command，参数如下（这里我们暂时先不考虑各种加密读写，会在本文最后介绍加密读写）

opcode = 0x12,

zone = 0x80|0x02; //32 字节写，写 slot 区。

address = 0x00;

data_1 = 32 个字节 0x11;

data_2 不填

如果写入成功，atsha204a 将会返回 0。

OTP 区配置

OTP 区和 slot 区配置类似，在 atsha204a 里，OTP 分 16 部分，每部分 4 字节，默认值都是 0xffffffff，OTP 区在 config 区锁定前无法进行任何读写操作，在 config 锁定后，data 区锁定前，不能读，只能写，与 slot 区不同的是，OTP 区并没有加密读写这一概念，另外也强烈建议使用 32 字节写，4 字节写入会有很多限制，在 data 区锁定后，OTP 区的读写操作由 OTP Mode 决定，在 config 区那节已经进行描述，这里不再论述。

关于 OTP 区的作用，一是用户可以自定义用途，二是 OTP 区要参与一些加密命令的 MAC 摘要计算，如果还有其他用途，欢迎读者补充。

加密读写

本人觉得加密读写是高级用户需要的功能，既然 atsha204a 提供这个功能，那想必此功能

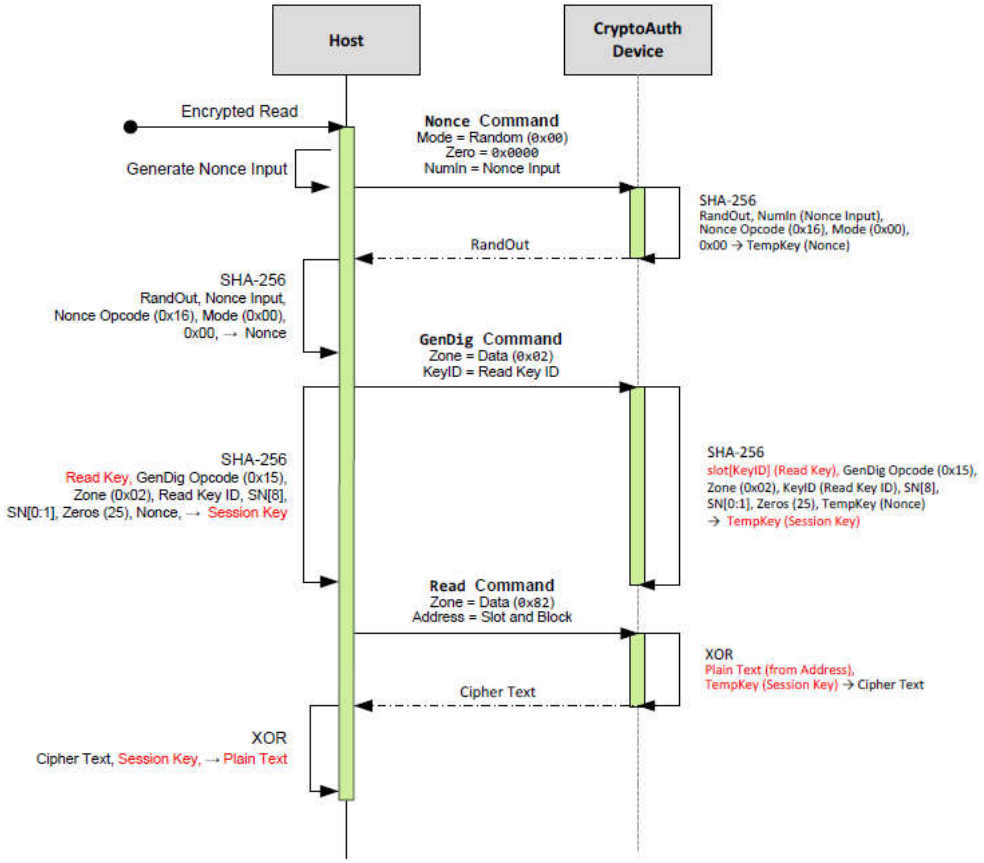
也十分有用，废话不多说，进入正题。

加密读，只有当 slot config.EncryptRead = 1 和 slot config.IsSecret = 1 时才能进行。

加密读是为了防止别人在总线上嗅探到密钥 ,这里我借用官方文档 Atmel-8981 的一幅图，

2.1 Standard Encrypted Read

Figure 2-1. Standard Encrypted Read Flow Diagram



这个流程图其实非常明了，主要步骤如下：

1、首先发送 Nonce Command 命令，更新 TempKey 中的值，主机这边根据命令返回的 randout 来使用 SHA-256 算法计算出 TempKey 中的值。

2、发送 GenDig Command ,Gen Dig 命令中的参数 KeyID 就是 slot config.ReadKey，比如说，我们要读取 slot1 的密钥，而 slot config[0].ReadKey 设置为 0，那么我们就需要知道 slot0 中存放的密钥才能读取 slot1 的密钥 ,发送完这个命令后如果执行正确 ,TempKey 中的值会被重新计算，而主机需要根据 Read Key 等一些列参数计算 TempKey 中的值。

3、执行读命令，读取 slot 区中的值，这时候返回来的内容是经过异或加密的，密钥是第 2 步计算出来的摘要，也就是说：读回来的值^第二步计算出来的摘要 = 原文。

这里说下 GenDig Command，这个命令在我上一篇使用篇的博客中没有提到，这里说一下，这条命令作用如下（datasheet 原文）：

8.5.8 GenDig Command

The GenDig command uses SHA-256 to combine a stored value with the contents of TempKey, which must have been valid prior to the execution of this command. The stored value can come from one of the data slots, either of the OTP pages, either of the first two pages of the Configuration zone, or retrieved from the hardware transport key array. The resulting digest is retained in TempKey, and can be used in one of three ways as follows:

1. It can be included as part of the message used by the MAC, CheckMac, or HMAC commands. Because the MAC response output incorporates both the data used in the GenDig calculation and the secret key from the MAC command, it serves to authenticate the data stored in the Data and/or OTP zones.
2. A subsequent Read or Write command can use the digest to provide authentication and/or confidentiality for the data, in which case it is known as a data protection digest.
3. This command can be used for secure personalization by using a value from the transport key array. The resulting data protection digest would then be used by the Write Command.

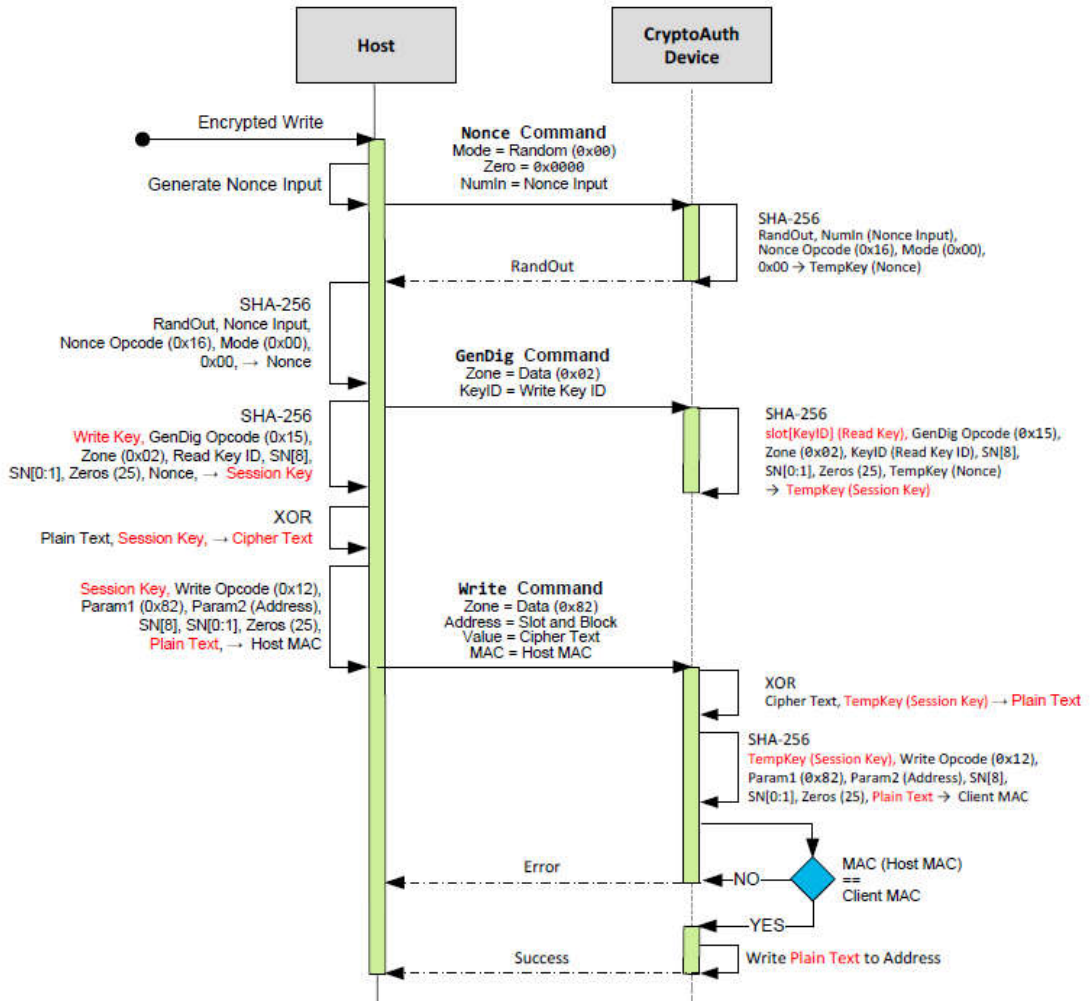
GenDig command 使用 SHA-256 算法结合 TempKey 中的值去更新 TempKey，在执行这条命令前，TempKey 中的值必须是合法的。

因为一些加密命令如 MAC，HMAC 命令都需要用到 TempKey 值，使用这条命令可以多增加一些 slot 区的使用验证，从而达到更加安全的验证。而在加密读的过程中，这条命令必须被正确执行，相应的区才能被正确的加密读取。

关于 Nonce 命令和 Read 命令，分别在我上一篇配置篇和上文中都提到了，这里不再讨论。

加密写，加密写稍微复杂一点，这里还是放 Atmel-8981 文档的图

Figure 3-1. Standard Encrypted Write Flow Diagram



同样，加密写也是为了防止别人在总线上获取密钥。与加密读过程类似，不过主机这里还要计算一个 MAC 发送给 atsha204a。

1、发送 Nonce Command 命令 ,更新 TempKey 中的值 ,主机这边根据命令返回的 randout 来使用 SHA-256 算法计算出 TempKey 中的值。

2、发送 GenDig command，KeyID 是所要写的 slot config.writekey 中指定的 ID，比如我们要写 slot1，而 slot config[1].writekey 设置为 0，那这个 keyID 就是 0。然后主机这边根据 Slot0 中的密钥和其他参数计算出一个摘要(Session Key)，当然 atsha204a 那边也会进行同样的计算并存在 TempKey 中。

3、我们要写的密钥和第二步计算出来的摘要进行异或运算等到加密文。

4、再根据第二步生成的摘要和明文还有其他参数(参照流程图中指示)计算出一个 Host MAC。

5、执行写命令，写命令中参数 value 是加密文，而参数 MAC 是 Host MAC(关于 Write command 的格式请回看上文)。

6、atsha204a 收到这个命令后，会对 value 解密(就是和自己的 TempKey 做异或运算)，然后再执行类似第 4 步的计算过程得到 Client MAC ,比较 Host MAC 和 Client MAC 就知道写入的数据是否正确了。

这样，整个加密写过程就是如此，在这整个过程中，无论是密钥，还是 Write Key，都没有明文在总线上传输，嗅探者也就不能截获我们设置的密钥了。

总结

本文介绍了 atsha204a 这款芯片的一系列配置方法，这是本人多日来研究所得，不过总结的最全的还是原 datasheet ,本人写的博客可能没有彻彻底底的把这款芯片的所有功能以及用途介绍清楚，开发者根据这么多的配置项，完全可以配置成自己所要的功能(如防抄板、产品匹对，密钥验证等)。另外本文中的不足之处，恳请指正。

附录

鉴于官方的烧录器价格非常感人 ,而某宝上也没有便宜的烧录器 ,本人亲自做了两种烧录器，分别是在线烧录并可调试，另外一种就是脱机带屏显示烧录器，可以进行小批量的烧录，下面是烧录器的样式

有需要的朋友可以到以下链接看看

https://item.taobao.com/item.htm?spm=a230r.1.14.22.oblrCT&id=550477591352&ns=1&abbucket=10#detail

另外一款脱机烧录带 oled 屏显示，支持一键下载，一键烧写，一键锁定，烧录高效，有兴趣的朋友可以到以下链接看看

https://item.taobao.com/item.htm?spm=a230r.1.14.44.oblrCT&id=553579727124&ns=1&abbucket=10#detail