RS9113 WiseConnect™ FIPS Software Programming Reference Manual

Version 2.0

January 2019

Table Of Contents

1 Overview

This document explains commands and their usage for RS9113 WiseConnect FIPS mode of use.

1.1 Enabling FIPS Mode

FIPS mode has to be enabled through bootloader. Once it is enabled it can't be disabled. FIPS mode can be disabled through FIPS bypass mode command. But this FIPS bypass command will not affect enable flag in the non volatile memory.

1.2 Power-up Self Tests

Firmware will do power-up self tests once FIPS approved mode is enabled. If any failure occurs in these self tests, firmware will indicate the same to host(through error message) and enters into error state. Firmware will do following self tests

- Boot-loader checksum
- Functional firmware checksum
- Store configuration checksum(critical functionality test)
- KAT tests for crypto routines

Firmware checksums will be indicated to host as well in card ready frame.

1.3 Runtime Self Tests

Firmware will do continuous random number test and enters into error state if any failure occurs.

1.4 Firmware up-gradation through Functional Firmware

Support for upgrading firmware through functional firmware from host interface is added. Firmware up-gradation through boot loader is disabled once FIPS approved mode is enabled. This was added mainly because the digital signature requirement for FIPS. Digital signature is added at the end of the firmware and magic number in 32byte rps header will indicate presence of Digital signature at the end. Digital signature is generated using RSA 2048 SHA256. RSA 2048 bit public key has to be loaded into the module which is used for validating the Digital signature during firmware upgradation.Signature can be inserted into the rps file using the scripts provided in the package.

1.5 Checksum for Store Configuration

SHA1 is used for computing the checksum for store configuration. Whenever store configuration structure is changed in non volatile memory, checksum also will be computed and stored in non volatile memory. At power-up, checksum will be computed as part of self tests and compared against stored checksum. Host can also query the checksum using checksum query command. Host can compute the checksum of the store configuration structure, hold the same and compare the result of the query command. While implementing one has to keep in mind that store configuration enable/disable commands will affect the checksum

1.6 Manual Key Reentry

Keys have to be entered twice. A new command has been added for taking the key second time. This key will be compared against the key entered through regular key entry commands

1.7 Key Zeroization

All the keys volatile and non volatile memory will be zeroized on this command. All session keys generated during command exchanges also will be zeroized.

And also module sets its GPIO 21 pin to low once key zeroization is finished. After key zeroization is issued host has to wait for GPIO 21 pin to become low and then hard reset should be issued. Module will not accept any command after key zeroization.

1.8 Wireless configuration

An error will be thrown to the browser if unallowed configuration is chosen from wireless configuration

1.9 Allowed Modes of Operation

- Only client mode is allowed
- Only WPA2(security mode 2) is allowed in FIPS approved mode
- Only allowed inner methods are “CHAP”, “PAP” in case of EAP-WPA2
- PSK is not allowed, only PMK is allowed
- ZB/BT coexistence modes are not allowed
- Only following TLS cipher suites are support in enterprise mode

  DHE-RSA-AES256-SHA256

  DHE-RSA-AES128-SHA256

  DHE-RSA-AES256-SHA

  DHE-RSA-AES128-SHA

  AES256-SHA256

  AES128-SHA256

  AES256-SHA

  AES128-SHA

NOTE: RS9113 Module uses hardware accelerator for DH computation and module supports up to 2048 bit.

1.10 DRBG Random Number Generation

On issuing generate DRBG random command, 256 bit random number will be generated from DRBG and given to the host. This generated random number can be used as PMK. In order to use this as PMK user has to enter this random number through PMK command.

2 Bootloader

1. Boot loader version 1.5 or higher is mandatory for using FIPS mode
2. Enable FIPS mode:
   I. In AT command mode select ‘F’(hidden option) as a boot up option to enable FIPS mode
   II. In Binary mode to enable FIPS mode, write value RSI_ENABLE_FIPS_MODE(0xab46) in HOST_INTERACT_REG_IN(0x41050034) register. rsi_select_option() API can be used with RSI_ENABLE_FIPS_MODE as parameter to enable FIPS approved mode

   NOTE: FIPS mode can’t be disabled once it is enabled

3. Disabling Firmware upgradation through bootloader :
   I. In AT command mode select ‘T’ (hidden option) as a bootup option to disable firmware upgrade through bootloader
   II. In Binary mode to disable Firmware upgradation through bootloader, write value RSI_DISABLE_FW_UPGRADATION_MODE(0xab54) in HOST_INTERACT_REG_IN(0x41050034) register. rsi_select_option() API can be used with RSI_DISABLE_FW_UPGRADATION_MODE as parameter to disable firmware upgradation through bootloader

   NOTE: Once Firmware upgradation through bootloader is disabled, it can’t be enabled again.

4. Firmware up-gradation and boot loader up-gradation is not allowed when FIPS mode is enabled or Firmware upgradation through bootloader is disabled. Firmware up-gradation has to be done from host through functional firmware or through wireless firmware up-gradation.
5. FIPS self tests should trigger without any user intervention. So once FIPS mode is enabled, Module has to configure in boot loader bypass mode.

3 AT Commands

This section explains AT commands and their usage. These are applicable only for UART or USB-CDC host interfaces.

3.1 Enable Non-FIPS Mode

Description:

By default FIPS approved mode is enabled at power up once it is enabled through boot loader. This command can be used to enable Non-FIPS mode. Once Non-FIPS mode is enabled user has to issue key zeroization command.

Command:

at+rsi_fips_mode

Usage:

at+rsi_fips_mode=mode\r\n

Parameters:

Mode(ASCII) – 0 – To Enable Non-FIPS mode

Response:

Result Code	Description
OK	Successful execution of the command
ERROR<Error code>	Failure, Possible error codes are 0x0021 – Command issued in wrong state 0xFFF8 – Error in command

Relevance:

This command is relevant only when FIPS approved mode is enabled through boot loader. This command has to be given immediately after init command.

Example:

Command:

at+rsi_fips_mode=0\r\n
……
0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x66 0x69 0x70 0x73 0x5F 0x6D 0x6F 0x64 0x65 0x3D 0x30 0x0D 0x0A

Response:

OK\r\n
……
0x4F 0x4B 0x0D 0x0A

3.2 Key Zeroization

Description:

This command is used to reset all the keys stored in module's RAM/Flash. Once this command is given module resets keys information.

And also module sets its GPIO 21 pin to low once key zeroization is finished. After key zeroization is issued host has to wait for GPIO 21 pin to become low and then hard reset should be issued.

Module will not accept any command after key zeroization.

This command can be given at any time after operating mode command.

Command:

at+rsi_key_zeroization

Usage:

at+rsi_key_zeroization\r\n

Parameters:

None

Response:

Module will not return any response for this command.

Relevance:

This command is relevant only when FIPS approved mode is enabled through bootloader.

Example:

Command:

at+rsi_key_zeroization\r\n
……..
0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x6B 0x65 0x79 0x5F 0x7A 0x65 0x72 0x6F 0x69 0x7A 0x61
0x74 0x69 0x6F 0x6E 0x0D 0x0A

Response:

3.3 Recheck Key

Description:

This command is used to

- Re-check the key(PMK/EAP PASSWORD/Encrypted private key) entered through normal key entry functions(at+rsi_psk & at+rsi_eap).
- Re-check the key(PMK/EAP PASSWORD/Encrypted private key) entered through normal key entry functions(at+rsi_psk & at+rsi_eap) and store the key into non-volatile memory if re-check is successful.

If the key in this command doesn't match with key entered through regular key entry function, an error will be given.

Command:

at+rsi_recheck_key

Usage:

at+rsi_recheck_key=type,key_store,key\r\n

Parameters:

type (1 character, ASCII):

This parameter is used to inform the key type.

Possible values for this parameter are ‘0’, ‘1’ and ‘2’.

0 - Entered key is PMK.

1 - Entered key is EAP password

2 – Entered key is encrypted private key

key_store(1 character, ASCII):

This parameter is used to store the key into non-volatile memory.

Possible values for this parameter are ‘0’ and ‘1’

0 – Don’t store key into non-volatile memory

1- Store key into non-volatile memory

Key will be stored only after recheck is successful.

key (maximum of 128 characters, ASCII): This field will contain the PMK/EAP password/Encrypted-private-key to be rechecked.

Response:

Result Code	Description
OK	Successful execution of the command.
ERROR<Error code>	Failure, Possible error codes are 0xFF3A – If re check command is issued before giving EAP/PMK command 0xFF3B – If EAP password/PMK/encrypted-private-key mismatches 0x0039 – If PMK length is not equal to 64 ASCII characters 0xFFF8 – Error in command

Relevance:

This command is relevant when FIPS approved mode is enabled through boot loader. This command should be issued before join and after giving the PMK/EAP PASSWORD/Encrypted-private-key.

Example:

1. To recheck key without storing key

at+rsi_recheck_key=1,0,test123\r\n

……..

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x72 0x65 0x63 0x68 0x65 0x63 0x6B 0x5F 0x6B 0x65 0x79 0x3D 0x31 0x2C 0x30

0x2C 0x74 0x65 0x73 0x74 0x31 0x32 0x33 0x0D 0x0A

Response:

OK\r\n

……..

0x4F 0x4B 0x0D 0x0A

2. To recheck key with storing key

at+rsi_recheck_key=1,1,test123\r\n

……..

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x72 0x65 0x63 0x68 0x65 0x63 0x6B 0x5F 0x6B 0x65 0x79 0x3D 0x31 0x2C 0x31

0x2C 0x74 0x65 0x73 0x74 0x31 0x32 0x33 0x0D 0x0A

Response:

OK\r\n

……

0x4F 0x4B 0x0D 0x0A

3.4 Store Configuration Checksum Query

Description:

This command is used for querying the store configuration checksum in FIPS mode. When this command is issued, firmware will compute the checksum using SHA1 algorithm on the current user store configuration structure present in flash and returns the same in response frame.

Command:

at+rsi_get_cfg_checksum?

Usage:

at+rsi_get_cfg_checksum?\r\n

Response:

Result Code	Description
OK<checksum>	Size of the checksum is 20 bytes.
ERROR<Error code>	Failure,

Result Code	Description
	Possible error codes are 0xFFF8 – Error in command

Relevance:

This command is relevant when the FIPS approved mode is enabled through boot loader. This command should be issued after opermode command.

Example:

Command:

at+rsi_get_cfg_checksum?

……..

0x61 0x74 0x2B 0x72 0x73 0x69 0x5F 0x67 0x65 0x74 0x5F 0x63 0x66 0x67 0x5F 0x63 0x68 0x65 0x63 0x6B 0x73 0x75 0x6D 0x3F 0x0D 0x0A

Response:

OK<checksum>\r\n

……..

0x4F 0x4B 0x12 0x23 0x34 0x45 0x56 0x67 0x78 0x89 0x90 0x00 0x12 0x23 0x34 0x45 0x56 0x67 0x78 0x89 0x90 0x00 0x0D 0x0A

3.5 Firmware Up-gradation From Host

Description:

This command is used for sending the firmware image(rps file) for firmware up-gradation from host interface.

NOTE: In FIPS approve mode, module can able upgrade only DSA signatured firmware. Please refer section Digital Signature Generation and Validation

Command:

at+rsi_upgrade_fw

Usage:

at+rsi_upgrade_fw=pkt_info,payload

Parameters:

pkt_info (4 bytes, ASCII):

This field contains the packet information. Packet information will contain the type of packet and length of packet. First two bytes of pkt_info will contain the type of packet(up-gradation request(1) or payload(0)). last two bytes will contain the length of the packet.

payload (maximum of 1024 bytes, Hex): This field will contain the actual payload. If packet type is firmware up-gradation request (1) this field will contain rps file header(the first 32 bytes of rps file). If packet type is payload (0), this field will contain the actual payload of the rps file (32 bytes onwards). Following shows the pseudo code of

```
send up-gradation request
image_size = size of the file – header size (32bytes)
i = 0
while(image_size)
{
    if(image_size >= 1024)
    {
        send payload message(1024 bytes from 32+i*1024 offset in   rps file)
        image_size = image_size – 1024
}
else
{
        send payload message(image_size bytes from 32+i*1024 offset in rps file)
        image_size = 0
    }

    i++
}
```

Response:

Result Code	Description
OK or 0x0003	OK will come on successful execution of the commands. For the last command instead of OK, 0x0003 will come as response
ERROR<Error code>	Failure, Possible error codes are 0x0015, 0xFF3D – Memory allocation failure 0xFF3E – Insufficient flash memory to hold image 0xFF3F – Check sum mismatch 0xFF40 – Packet info parameters are wrong 0xFF41 – invalid length 0xFFF8 – Error in command 0xFF47 – DSA mismatch

Relevance:

This command should be issued after opermode command.

3.6 Generate DRBG Random Number

Description:

On issuing generate DRBG random command, 256 bit random number will be generated from DRBG and given to the host. This generated random number can be used as PMK. In order to use this as PMK user has to enter this random number through PMK command.

Command:

at+rsi_generate_drbg_rand

Usage:

at+rsi_generate_drbg_rand\r\n

Parameters:

No parameters

Response:

Result Code	Description
OK<Random Number>	Successful execution of the command Random Number- 32 bytes random number
ERROR<Error code>	Failure, Possible error codes are 0x0021 – Command issued in wrong state 0xFFF8 – Error in command

Relevance:

This command has to be given after init command.

Example:

Command:

at+rsi_generate_drbg_rand\r\n

……

0x61 0x74 0x2B 0x72 0x73 0x69 0x2D 0x67 0x65 0x6E 0x65 0x72 0x61 0x74 0x65 0x5F 0x64 0x72 0x62 0x67 0x5F 0x72 0x61 0x6E 0x64 0x0D 0x0A

Response:

OK<Random Number>\r\n

3.7 FIPS Failure Indication

- Module will perform FIPS power up self tests during every boot up. If any of the power up self tests fails, then module indicates to host by giving message “FIPS Self Test Failed\r\n” instead of giving “Loading Done\r\n” message.
- If any continuous random generation tests are fail, module sends an asynchronous message “AT+RSI_FIPS_FAILED\r\n” to host.
- If any of the above failure occurs, module enters a error state. In this state module won’t accept/send any commands from/to host. Also module won’t accept/send any packets on wireless
- Host has to give hard reset to recover from this state

3.8 Load Firmware Up-gradation Key

Description:

This command is used to load Firmware Up-gradation Key to the module. This key is used to validate the DSA signature of firmware. This command can be given after operating mode command.

NOTE: This Command is allowed only one time. Once Firmware upgradation key loaded, module will not allow to change the key.

Command:

at+rsi_load_fwupgradation_key

Usage:

at+rsi_load_fwupgradation_key=<n-size><e-size><n-key><e-key>\r\n

Parameters:

fwupgradation_key :

This parameter is used to send the firmware up-gradation key to the module. Firmware upgradation key should be send in binary format. Firmware upgradation key contains following fields:

n-size (2bytes, Binary) : length of the modulus

e-size (2bytes, Binary) : length of the exponent

n-key : RSA 2048 bit Public key modulus

e-key: RSA 2048 bit Public key exponent

Response:

Result Code	Description
OK	Successful execution of the command
ERROR<Error code>	Failure, Possible error codes are

Result Code	Description
	0x0015- Memory allocation failure 0xFF35 – Key load failed 0xFFF8 – Error in command 0xFF44 – Wrong key length 0xFF45 – Key already present

Relevance:

This command has to be given after opermode command.

Example:

Command:

at+rsi_load_fwupgradation_key=
00010300B37EF81FFAAE2904E2862559BDD2A0B48A2A0841ED63108A26E172D782F5BDA9C53A649810AC1
052A33D2CFFF4D3FD9EF0341C5BB2A2CB2BE3292583156E31456EB1EA26A8CDA8132F0FC1ABA3F6E9FF0
FD51E9F7344EB3B07DA57654519B188F3170BECFC8592FD62480AF4D867BF4D3AD0F6B0B0BE6B4EDA19C
16C746FA5A1ED0D33D32905FD6ABE53D13D73279C5348D30C99B6D0CF9E99662240DB0D16BA707BB4C37
D53B58FED12D5EF00025B8F11CF065B64867CDC016EE79DF24FD1DF9F980CD0BCAF2CE4FDA0F7E82DD2
AF6A7A4F8EE1AE9013B34033B36B60CD1E7AA957BADA09B78E2E223B46B271E9B965A6E7F8A616C5122D
967608809F36FA61010001 \r\n

Response:

OK\r\n

3.9 Get Firmware Up-gradation Key

Description:

This command is used to retrieve already loaded Firmware Up-gradation Key from the module.This command can be given after operating mode command.

Command:

at+rsi_get_fwupgradation_key

Usage:

at+rsi_get_fwupgradation_key\r\n

Parameters:

No parameters required

Response:

Result Code	Description
OK<n-size><e-size><n-key><e-key>\r\n	Successful execution of the command returns stored firmware upgradation key in binary format, which contain following fields: n-size (2bytes, Binary) : length of the

	modulus e-size (2bytes, Binary) : length of the exponent n-key : RSA 2048 bit Public key modulus e-key: RSA 2048 bit Public key exponent
ERROR<Error code>	Failure, Possible error codes are 0xFFF8 – Error in command 0xFF46 – Key not present

Relevance:

This command has to be given after operating mode command.

Example:

Command:

at+rsi_get_fwupgradation_key\r\n

Response:

OK0001 0300B37EF81FFAAE2904E2862559BDD2A0B48A2A0841ED63108A26E172D782F5BDA9C53A649810A
C1052A33D2CFFF4D3FD9EF0341C5BB2A2CB2BE3292583156E31456EB1EA26A8CDA8132F0FC1ABA3F6E9F
F0FD51E9F7344EB3B07DA57654519B188F3170BECFC8592FD62480AF4D867BF4D3AD0F6B0B0BE6B4EDA1
9C16C746FA5A1ED0D33D32905FD6ABE53D13D73279C5348D30C99B6D0CF9E99662240DB0D16BA707BB4C
37D53B58FED12D5EF00025B8F11CF065B64867CDC016EE79DF24FD1DF9F980CD0BCAF2CE4FDA0F7E82D
D2AF6A7A4F8EE1AE9013B34033B36B60CD1E7AA957BADA09B78E2E223B46B271E9B965A6E7F8A616C512
2D967608809F36FA61010001 \r\n

4 Binary Commands

This section explains commands and their usage in FIPS mode. These are applicable for SPI, UART, USB, USB-CDC host interfaces.

4.1 Enable Non-FIPS Mode

Description:

By default FIPS approved mode is enabled at power up once it is enabled through boot loader. This command can be used to enable Non-FIPS mode. Once Non-FIPS mode is enabled user has to issue key zeroization command.

Payload Structure:

```
struct {
        uint8           mode[4];
} fipsFrameSnd;
```

Parameters:

mode: 0– To Enable Non-FIPS mode

Response Payload:

There is no response payload for this command.

Request Frame Type:

0xFE

Response Frame Type:

0xFE

Possible Error Codes:

0x0021 – Command given in wrong state

Relevance:

This command is relevant only when FIPS approved mode is enabled through boot loader. This command has to be given immediately after init command.

4.2 Key Zeroization

Description:

This command is used to reset all the keys stored in module's RAM/Flash. Once this command is given module resets keys information.

And also module sets its GPIO 21 pin to low once key zeroization is finished. After key zeroization is issued host has to wait for GPIO 21 pin to become low and then hard reset should be issued.

Module will not accept any command after key zeroization.

This command can be given at any time after operating mode command.

Payload Structure:

No payload required.

Parameters:

No parameters.

Response Payload:

There is no response payload for this command.

Request Frame Type:

0xFD

Response Frame Type:

No response frame

Possible Error Codes:

0x0021 – Command given in wrong state

Relevance:

This command is relevant only when FIPS approved mode is enabled through bootloader.

4.3 Recheck Key

Description:

This command is used to

- Re-check the key(PMK/EAP PASSWORD/Encryptyed-private-key) entered through normal key entry functions(at+rsi_psk & at+rsi_eap).
- Re-check the key(PMK/EAP PASSWORD/Encrypted-private-key) entered through normal key entry functions(at+rsi_psk & at+rsi_eap) and store the key into non-volatile memory if re-check is successful.

If the key in this command doesn't match with key entered through regular key entry function, an error will be given.

Payload Structure:

```
Struct {
    uint8 type;
    uint8 key_store;
    uint8 key[128];
 }RecheckKeyFrameSnd;
```

Parameters:

type : 0 – PMK,

1 – EAP PASSWORD,

2 – Encrypted private key

Key_store:

This parameter is used to store the key into non-volatile memory.

Possible values for this parameter are '0' and '1'

0 – Don't store key into non-volatile memory

1 – Store key into non-volatile memory

Key will be stored only after recheck is successful.

Key: PMK/EAP password/Encrypted-private-key.

If type is 0 then this field should contain PMK . Only 32 bytes(Hex) are valid in this case.

if type is 1 then this field should contains EAP password in string format.

If type is 2 then this field should contains Encrypted private key.

Response Payload:

There is no response payload for this command.

Request Frame Type:

0xFB

Response Frame Type:

0xFB

Possible Error Codes:

0xFF3A – If re check command is issued before giving EAP/PMK command

0xFF3B – If EAP password/PMK/Encrypted-private-key mismatches

0x0039 – If PMK length is not equal to 64 ASCII characters

Relevance:

This command is relevant when the FIPS approved mode is enabled through bootloader.

4.4 Store Configuration Checksum Query

Description:

This command is used for querying the store configuration checksum in fips mode. When this command is issued, firmware will compute the checksum using SHA1 algorithm on the current user store configuration structure present in flash and returns the same in response frame.

Payload Structure:

No payload required.

Parameters:

No parameters.

Response Payload:

Response of this command will contain the 20 byte checksum of the store configuration structure.

```
Typedef struct sore_config_chksum_s{
    uint8 checksum[20];
} rsi_store_config_checksumRsp;
```

Request Frame Type:

0xFC

Response Frame Type:

0xFC

Relevance:

This command is relevant when the FIPS approved mode is enabled through bootloader. This command should be issued after opermode command.

4.5 Firmware Up-gradation From Host

Description:

This command is used for sending the firmware image(rps file) for firmware up-gradation from host interface.

NOTE: In FIPS approve mode, module can able upgrade only DSA signatured firmware. Please refer section Digital Signature Generation and Validation

Payload Structure:

```
typedef struct rsi_fw_up_frm_host_s {
    uint8 packet_info[4];
    uint8 payload[1024];
    }rsi_fw_up_t;
```

Parameters:

Packet_info :

This field contains the packet information. Packet information will contain the type of packet and length of packet. first two bytes of pkt_info will contain the type of packet(up-gradation request(1) or payload(0)). last two bytes will contain the length of the packet.

Payload (maximum size 1024 bytes): This field will contain the actual payload. If packet type is firmware up-gradation request (1) this field will contain rps file header(the first 32 bytes of rps file). If packet type is payload (0), this field will contain the actual payload of the rps file (32 bytes onwards). Following shows the pseudo code of

```
send up-gradation request
image_size = size of the file – header size (32bytes)
i = 0
while(image_size)
{
    if(image_size >= 1024)
   {
            send payload message(1024 bytes from 32+i*1024 offset in   rps file)
            image_size = image_size – 1024
}
else
{
            send payload message(image_size bytes from 32+i*1024 offset in rps file)
            image_size = 0
 }

 i++
}
```

Response Payload:

There is no response payload for this command.

Request Frame Type:

0x99

Response Frame Type:

0x99

Possible Error Codes:

0x0015, 0xFF3D – Memory allocation failure

0xFF3E – Insufficient flash memory to hold image

0xFF3F – Checksum mismatch

0xFF40 – Packet info parameters are wrong

0xFF41 – invalid length

0xFF47 – DSA mismatch

Relevance:

This command should be issued after the init command.

Note: For this command response status will come as success(0) for all packets except the last packet. For last packet 0x0003 response will come. Which will indicate firmware upgrade as success.

4.6 Generate DRBG Random Number

Description:

On issuing generate DRBG random command, 256 bit random number will be generated from DRBG and given to the host. This will not be used by the module. This has to be reentered into the module. This generated random number can be used as PMK.

Payload Structure:

No payload required.

Parameters:

No parameters.

Response Payload:

```
typedef struct generate_drbg_rand_s{
    uint8 rand_number[32];
} rsi_generate_DRBGRandRsp;
```

Request Frame Type:

0xF8

Response Frame Type:

0xF8

Possible Error Codes:

0x0021 – Command given in wrong state

Relevance:

This command has to be given after init command.

4.7 FIPS Failure Indication

- Module will perform FIPS power up self tests during every boot up. If any of the power up self tests fail, or if any continuous random generation tests fail, then module sends an asynchronous frame with type 0xFA, which indicates FIPS failure.
- If any of the above failure occurs, module enters error state. In this state module won't accept/send any commands from/to host. Also module won't accept/send any packets on wireless
- Host has to give hard reset to recover from this state
-

4.8 Load Firmware Up-gradation Key

Description:

This command is used to load Firmware Up-gradation Key into the module. This key is used to validate the DSA signature of firmware. This command can be given after operating mode command.

NOTE: This Command is allowed only one time. Once Firmware upgradation key loaded, module will not allow to change the key.

Payload Structure:

```
typedef struct load_fwupgadation_key_s{
    uint8 n-size[2];
    uint8 e-size[2];
        uint8 publickey[KEY_LENGTH];
} load_ fwupgradation_key_t;
```

NOTE: Size of the publickey (KEY_LENGTH) is sum of n-size and e-size.

Parameters:

n-size (2 bytes): length of the modulus

e-size (2 bytes): length of the exponent

public_key : RSA 2048 bit Public key modulus followed by RSA 2048 bit Public key exponent

Response Payload:

There is no response payload for this command.

Request Frame Type:

0xF9

Response Frame Type:

0xF9

Possible Error Codes:

0x0015 – Memory allocation failure

0xFF35 – Key load failed

0xFF44 – Wrong key length

0xFF45 – Key already present

Relevance:

This command has to be given after operating mode command.

4.9 Get Firmware Up-gradation Key

Description:

This command is used to retrieve already loaded Firmware Up-gradation Key from the module.This command can be given after operating mode command.

Payload Structure:

No payload required.

Parameters:

No parameters.

Response Payload:

```
typedef struct get_fwupgadation_key_s{
    uint8 n-size[2];
    uint8 e-size[2];
        uint8 publickey[KEY_LENGTH];
} get_fwupgradation_key_t;
```

NOTE: Size of the publickey (KEY_LENGTH) is sum of n-size and e-size.

Parameters:

n-size (2 bytes): length of the modulus

e-size (2 bytes): length of the exponent

public_key : RSA 2048 bit Public key modulus followed by RSA 2048 bit Public key exponent

Request Frame Type:

0xF6

Response Frame Type:

0xF6

Possible Error Codes:

0xFF46 – Key not present

Relevance:

This command has to be given after opermode command.

4.10 Commands Request and response IDs

Below table explains request and response IDs for FIPS commands

Command	Request ID	Response ID
Get firmware upgradation key	0xF6	0xF6
Generate PMK	0xF8	0xF8
Load firmware upgrade key	0xF9	0xF9
FIPS failure	-	0xFA
Re-Check key	0xFB	0xFB
Store Configuration Checksum Query	0xFC	0xFC
Key Zeroization	0xFD	-
Enable Non-FIPS Mode	0xFE	0xFE

5 Error Codes

Error Code	Description
0x0015	Memory allocation failed
0x0021	Command given in wrong state
0x0039	If PMK length is not equal to 64 ASCII characters
0xFF35	Firmware upgradation key load failed
0xFF3A	Re check command is issued before giving EAP/PMK command
0xFF3B	PMK/EAP password mismatches
0xFF3C	Re-key is not done
0xFF3E	Insufficient flash memory to hold image
0xFF3F	Checksum mismatch
0xFF40	Packet info parameters are wrong
0xFF41	invalid length
0xFF43	Invalid configuration for store configuration save
0xFF6E	Invalid operating mode
0xFF37	Key is not stored
0xFF44	Wrong key length
0xFF45	Firmware upgration key already present
0xFF46	Firmware up-gradation key not present
0xFF47	DSA mismatch
0xFF48	DSA not present in RPS header of firmware file
0xFFF8	Error in Command

6 Differences in existing commands Usage in FIPS mode

6.1 Set Operating Mode

Description:

This is the first command that needs to be sent from the Host. This command configures the module in different functional modes.

Payload Structure:

The structure of the payload is give below

```
struct
{
  uint32 oper_mode;
  uint32  feature_bit_map;
  uint32  tcp_ip_feature_bit_map;
  uint32  custom_feature_bit_map;
  uint32  ext_custom_feature_bit_map;
  uint32  ble_feature_bit_map;
  uint32  ext_tcpip_feature_bit_map
} operModeFrameSnd;
```

Changes:

- Only normal client mode(opermode 0) and enterprise client modes(opemode 2) are allowed

6.2 Join

Description:

This command is used to associate to an access point (operating mode = 0 or 2 for FIPS enabled mode)

Payload Structure:

```
Struct {
                uint8     reserved1;
                uint8     security_type;
                uint8     dataRate;
                uint8     powerLevel;
                uint8     psk[64];
                uint8     ssid[34];
                uint8    join_feature_bitmap;
                uint8     reserved2[2];
                uint8     ssid_len;
                uint32   listen_interval;
                uint8     vap_id;
                uint8     join_bssid[6];
        } joinFrameSnd;
```

Changes:

- Only WPA2 security type(2) is allowed. Even for enterprise client security mode only this option has to be given.
- PSK is not allowed

- Following cipher suits will not be present in client hello message when module is operating in enterprise client mode

  SSL_RSA_WITH_RC4_128_CBS_SHA (5)

  SSL_RSA_WITH_RC4_128_CBS_MD5 (4)

  SSL_RSA_WITH_3DES_EDA_CBS_SHA (10)

Possible error codes:

0xFF3C – Re-key is not done

0xFF42 – Security mode is not allowed

6.3 PSK/PMK

Description:

This command is used to give the PMK to the module.

Payload Structure:

```
typedef union{
   struct {
      uint8 TYPE;
      uint8 psk_or_pmk [RSI_PSK_LEN];
      uint8 ap_ssid [RSI_SSID_LEN];
        }
      uint8 uPskBuf[1 + RSI_PSK_LEN + RSI_SSID_LEN];
       } rsi_uPsk;
```

Changes:

- Only type value 2 is allowed
- If psk_or_pmk is given with all zeros then module checks whether PMK is already stored in non-volatile memory or not.

  If PMK is already stored, then module will use PMK key from non-volatile memory. In this case re-check key is not required.

  If PMK is not stored before then module will throws an error (0xFF37).

6.4 Card Ready Frame

- Module will give card ready frame with 1's complement checksums of boot loader and firmware.

Response Payload:

```
typedef struct card_ready_s{
        uint8 bootloader_checksum[4];
        uint8 firmware_checksum[4];
} rsi_card_readyRsp;
```

bootloader_checksum : Contains boot loader's 1's complement additive checksum.

Firmware_checksum : Contains functional firmware's 1's complement additive checksum.

- Last byte(16th byte) of the card ready frame descriptor gives following status bits

BIT(0) :

0 – Trigger auto configuration is disabled

1- Trigger auto configuration is enabled

BIT(1):

0 – Auto-Join is disabled

1- Auto-Join is enabled

BIT(2):

0 – FIPS mode is disabled

1- FIPS mode is enabled

BIT(3):

0 – Keys stored in non-volatile memory

1 – Keys not stored in non-volatile memory

BIT(4):

0 – Firmware up-grade key is not present in non-volatile memory

1 – Firmware up-grade key is present in non-volatile memory

BIT(5):

0 – Firmware up-gradation through bootloader is not disabled

1 – Firmware up-gradation through bootloader is disabled

- 12th byte of the card ready frame descriptor describes about the type of the key stored.

  This byte contains valid value only if BIT(3) in the last byte is set.

  This byte contains following bit values.

  BIT(0) – PMK stored

  BIT(1) – EAP password stored

  Both BIT(0) and BIT(1) will be set if both keys are stored

6.5 EAP Configuration Frame

Description:

This command is used to set EAP configuration parameters to the module.

Payload Structure:

```
typedef union {
   struct {
      uint8   eapMethod[32];   //@ EAP method
```

```
    uint8   innerMethod[32]; //@ Inner method
    uint8   userIdentity[64];//@ user name
    uint8   password[128];   //@ Password
    int8    okc_enable[4];   //@ Opportunistic Key Caching
    uint8   private_key_password[82]; //@Encrypted private key password
    uint8   is_ca_required; //@ is CA certificate required for PEAP or not.
    uint8  tls_version; //@ tls version used for connectivity
  }setEapFrameSnd;
  uint8   uSetEapBuf[256 + 4 + 82];
}rsi_uSetEap;
```

Changes:

- Only allowed inner methods are “CHAP”, “PAP”
- If password is given with NULL, then module checks whether EAP password is already stored in non-volatile memory or not.

  If EAP password is already stored, then module will use EAP password from non-volatile memory. In this case re-check key is not required.

  If EAP password is not stored before then module will throws an error (0xFF37).

6.6 Store Configuration from User

Description:

This command is used to give the configuration values which are supposed to be stored in the module’s memory and is used in auto-join or auto-create modes on next boot up

Changes in Payload Structure:

```
 #define IP_ADDRESS_SZ   4
#define RSI_SSID_LEN    34
#define WISE_PMK_LEN    32
typedef struct {
        uint8   channel_no[2];
        uint8   ssid[RSI_SSID_LEN];
        uint8   security_type;
        uint8   encryp_mode;
        uint8   psk[RSI_PSK_LEN];
        uint8   beacon_interval[2];
        uint8   dtim_period[2];
        uint8   ap_keepalive_type;
        uint8   ap_keepalive_period;
        uint8   max_sta_support[2];
}rsi_apconfig;
```

```
typedef struct {
        uint8   index[2];
        uint8   key[4][32];
}rsi_wepkey;

typedef union {
        struct {
                uint8   roam_enable[4];
                uint8   roam_threshold[4];
                uint8   roam_hysteresis[4];
        }roamParamsFrameSnd;
        uint8   uRoamParamsBuf[12];
}rsi_uRoamParams;

typedef struct rsi_rejoin_params_s{
        uint8   rsi_max_try[4];
        int8    rsi_scan_interval[4];
        int8    rsi_beacon_missed_count[4];
        uint8   rsi_first_time_retry_enable[4];
} rsi_rejoin_params_t;

typedef struct {
        uint8   cfg_enable;
        uint8   opermode[4];
        uint8   feature_bit_map[4];
        uint8   tcp_ip_feature_bit_map[4];
        uint8   custom_feature_bit_map[4];
        uint8   band;
        uint8   scan_feature_bitmap;
        uint8   join_ssid[RSI_SSID_LEN];
        uint8   uRate;
        uint8   uTxPower;
        uint8   join_feature_bitmap;
        uint8   reserved_1;
        uint8   scan_ssid_len;
        uint8   keys_restore;
        uint8   csec_mode;
        uint8   psk[RSI_PSK_LEN];
        uint8   scan_ssid[RSI_SSID_LEN];
        uint8   scan_cnum;
```

```
uint8   dhcp_enable;
uint8   ip[IP_ADDRESS_SZ];
uint8   sn_mask[IP_ADDRESS_SZ];
uint8   dgw[IP_ADDRESS_SZ];
uint8   eap_method[32];
uint8   inner_method[32];
uint8   user_identity[64];
uint8   passwd[128];
uint8   go_intent[2];
uint8   device_name[64];
uint8   operating_channel[2];
uint8   ssid_postfix[64];
uint8   psk_key[64];
uint8   pmk[WISE_PMK_LEN];
rsi_apconfig apconfig;
uint8   module_mac[6];
uint8   antenna_select[2];
uint8   fips_bypass_mode[2];
rsi_wepkey wep_key;
uint8   dhcp6_enable[2];
uint8   prefix_length[2];
uint8   ip6[16];
uint8   dgw6[16];
uint8   tcp_stack_used;
uint8   bgscan_magic_code[2];
uint8   bgscan_enable[2];
uint8   bgscan_threshold[2];
uint8   rssi_tolerance_threshold[2];
uint8   bgscan_periodicity[2];
uint8   active_scan_duration[2];
uint8   passive_scan_duration[2];
uint8   multi_probe;
uint8   chan_bitmap_magic_code[2];
uint8   scan_chan_bitmap_stored_2_4_GHz[4];
uint8   scan_chan_bitmap_stored_5_GHz[4];
uint8   roam_magic_code[2];
rsi_uRoamParams  roam_params_stored;
uint8   rejoin_magic_code[2];
rsi_rejoin_params_t rejoin_param_stored;
uint8   region_request_from_host;
```

```
uint8    rsi_region_code_from_host;
uint8    region_code;
uint8     reserved_4[43];
uint8     multicast_magic_code[2];
uint8     multicast_bitmap[2];
uint8     powermode_magic_code[2];
uint8     powermode;
uint8     ulp_mode;
uint8     wmm_ps_magic_code[2];
uint8     wmm_ps_enable;
uint8     wmm_ps_type;
uint8     wmm_ps_wakeup_interval[4];
uint8     wmm_ps_uapsd_bitmap;
uint8     listen_interval[4];
uint8     listen_interval_dtim;
uint8     ext_custom_feature_bit_map[4];
uint8     private_key_password[82];
uint8     join_bssid[6];
uint8      fast_psp_enable;
uint8      monitor_interval[2];
uint8      timeout_value[2];
uint8      timeout_bitmap[4];
 uint8      request_timeout_magic_word[2];
 rsi_uHtCaps ht_caps;
 uint8      ht_caps_magic_word[2];
//! AP IP parameters in Concurrent mode
 UINT8     dhcp_ap_enable;
 UINT8     ap_ip[4];
 UINT8     ap_sn_mask[4];
 UINT8     ap_dgw[4];
 uint8     dhcp6_ap_enable[2];
 UINT8     ap_prefix_length[2;
 UINT8     ap_ip6[16];
 UINT8     ap_dgw6[16];
 uint8     ext_tcp_ip_feature_bit_map[4];
 uint8     http_credentials_avail;
 uint8     http_username[MAX_HTTP_SERVER_USERNAME];
 uint8     http_password[MAX_HTTP_SERVER_PASSWORD];

}rsi_user_store_config_t;
```

Changes:

- Added variables keys_restore and fips_bypass_mode in the structure which are used in the FIPS enabled mode
- keys_restore : If this variable value is 1, then module restore's PMK/EAP password/Encrypte-private-key from non-volatile memory in FIPS mode.
- fips_bypass_mode : set value 0 to enable FIPS bypass mode.

  Set value 1 to disable FIPS bypass mode.

7 APIs

Following APIs are available for FIPS mode. These are available in

FIPS/apis folder of the release directory

1. rsi_fips_mode.c

   This API is used to enable or disable FIPS mode

2. rsi_fips_key_zeroization.c

   This API is used to send key zeroization command

3. rsi_recheck_key.c

   This API is used to send recheck key command

4. rsi_config_checksum_query.c

   This API is used to query store configuration checksum value

5. rsi_fwup_frm_host.c

   This API is used to upgrade the firmware through host without boot loader interaction

6. rsi_generate_drbg_rand.c

   This API is used to generate the 32 byte random number from FIPS certified DRBG

7. rsi_get_fwupgradation_key.c

   This API is used to retrieve firmware upgradation key loaded in module

8. rsi_load_fwupgradation_key.c

   This API is used to load the firmware up-gradation key used in digital signature validation in firmware up-gradation

8 Boot loader Up-gradation

This section explains procedure to upgrade boot loader from lower versions (1.4 or before) to higher versions(1.5 or later) over UART/USB-CDC interfaces, following steps have to be followed

1. Perform ABRD(Auto Baud Rate Detection) with module
2. Select option '#' (hidden option) to upgrade boot loader
3. Module sends a message "ARE YOU SURE YOU WANT TO UPGRADE BOOTLOADER? (YES/NO)"
4. Send "YES" (In capital letters)
5. Then module asks to send boot loader file
6. Send boot loader rps file through Kermit protocol
7. Once up-gradation is successful module sends a successful message
8. Reboot the module and follow above steps to upgrade boot loader once again. Second time up-gradation is required to store the boot loader checksum on non-volatile memory.

9 Digital Signature Generation and Validation

Digital signature is attached at the end of firmware image (rps file) is used to authenticate and validate the firmware. Existence of Digital signature at the end of the firmware is represented by magic number in firmware image (rps file) header. As DSA is part of Firmware image, it will be sent to the module through firmware upgradation.

In order to use digital signature validation feature user should load the firmware up-gradation key (RSA 2048 bit Public key) in the module. Please note that the firmware up-gradation key is allowed to load only one time (i.e once loaded, module does not allow changing the firmware upgradation key).

1) Generate public and private key pair using the procedure described in section Procedure to generate Digital Signature.

   NOTE: Private key is a secret key and should not be disclosed out side the secure premises.

2) Load the Public key to the module using Load firmware upgradation key command
3) Add the digital signature to the firmware image using private key.

   Release package contain scripts to generate digital signature using open ssl commands and append generated Digital signature at the end of firmware image.Please refer Procedure to generate Digital Signature section for more details.

   NOTE: Should use corresponding private key pair for the public key loaded in step 2 to generate digital signature.

4) Burn the rps file into the module with the help of wireless firmware up-gradation or firmware up-gradation from host through functional firmware
5) After receiving complete firmware image, module will validate the digital signature of the received firmware image using public key loaded through load firmware upgrade key command (step 2) .
6) Firmware up-gradation successful if digital signature validation is successful.
7) Give soft reset or hard reset, so that boot loader will update the current image with up-graded image. For this update boot loader may take around 2 minutes.

9.1 Procedure to generate Digital Signature

This utility is used to add the digital signature to the provided firmware image.

1) Go the DSA utility directory in release package(FIPS/generate_dsa)

2) sh ./generate_key.sh

This script will generate following files using open ssl commands:

privkey-ID.pem : RSA 2048 bit private key, which is used to generate digital signature on firmware image.

pubkey-ID.pem : RSA 2048 bit public key, which is used to authenticate and validate the digital signature of firmware image.

fwupgrade_load_key : This file contain RSA 2048 bit public key modulus and RSA 2048 bit public key exponent values which are extracted from pubkey-ID.pem file. User required to load the content of this file to the module using load firmware upgrade key command.

NOTE: fwupgrade_load_key file content is in the format expected by load firmware upgrade key command. Content of this file has to be assigned to “RSI_FWUPGRADATION_KEY” as a string in rsi_config.h

3) sh ./generate_dsa.sh FIRMWARE_IMAGE_PATH privkey-ID.pem

First Argument (FIRMWARE_IMAGE_PATH): path of the firmware image in which we want to add DSA

Second Argument (privkey-ID.pem): RSA 2048 bit private key generated in step 2.

This script generates digital signature using RSA with SHA-256 and append at the end of the firmware file.This script also add magic number in firmware image (rps file) header to indicate Digital signature is present at the end of the firmware image.

4) New firmware image file is generated

Note: DSA signature is mandatory for firmware up-gradation even though FIPS bypass mode is enabled.

10 FIPS Enable Flags

Two types of flags are available for enabling or disabling FIPS mode.

1. FIPS approved mode flag:
   - This Flag can be enabled through boot loader.
   - This Flag can't be disabled once it is enabled.
   - This Flag will be checked in following scenarios:
     - FIPS Power-up Self Test
     - DSA validation for firmware upgrade
     - Loading firmware up-gradation key (Allowed only if this flag is disabled).
     - For restoring masked configuration fields(PMK/ EAP password)

2. FIPS bypass mode flag:
   - This Flag will be valid only if FIPS approved mode flag is set.
   - This Flag can be disabled using FIPS bypass mode command.
   - To enable FIPS bypass mode, this Flag should be set to 0 by using FIPS bypass mode command.
   - This flag also be stored using configuration save or user store configuration.

11 Flow Charts

This section explains different FIPS functionalities through flow charts

11.1 Enable FIPS mode through boot loader in AT mode

11.2 Enable FIPS mode through boot loader in Binary mode

11.3 FIPS Keys store/retrieve and connection establishment

11.4 Firmware upgrade through Firmware

11.5 Key zeroization

This command can be given at any time after operating mode command

12 Minimal Documentation Requirements for SP800-90A section 11.3

- Document the method for obtaining entropy input.

  Entropy input for DRBG is obtained by reading hardware random number register

- Document how the implementation has been designed to permit implementation validation and health testing.

  A set of known answer tests and continuous random number test are provided in the implementation to perform the above testing. Health tests are performed during power up and conditional tests are performed during normal operations.

- Document the type of DRBG mechanism (e.g., **CTR_DRBG**), and the cryptographic primitives used (e.g., AES-128 or SHA-256).

  SP800-90A DRBG SHA-256 HASH_DRBG .

- Document the security strengths supported by the implementation.

  Security strength supported is 256

- Document features supported by the implementation (e.g., prediction resistance, personalization string, additional input, etc.).
  Does not support any other features, only entropy is given as input.

- If DRBG mechanism functions are distributed, specify the mechanisms that are used to protect the confidentiality and integrity of the internal state or parts of the internal state that are transferred between the distributed DRBG mechanism sub-boundaries (i.e., provide documentation about the secure channel).

  N/A

- In the case of the **CTR_DRBG**, indicate whether a derivation function is provided. If a derivation function is not used, document that the implementation can only be used when full entropy input is available.

  N/A

- Document any support functions other than health testing.

  DRBG continuous random number tests and timer based periodic KAT.

- If periodic testing is performed for the generate function, document the intervals and provide a justification for the selected intervals (see Section 11.3.3).

  Periodic testing is done in the interval of 24 hours. Silicon Labs believes 24 hours is sufficient based on module activity and module usage of DRBG mechanism.

- Document whether the DRBG functions can be tested on demand.

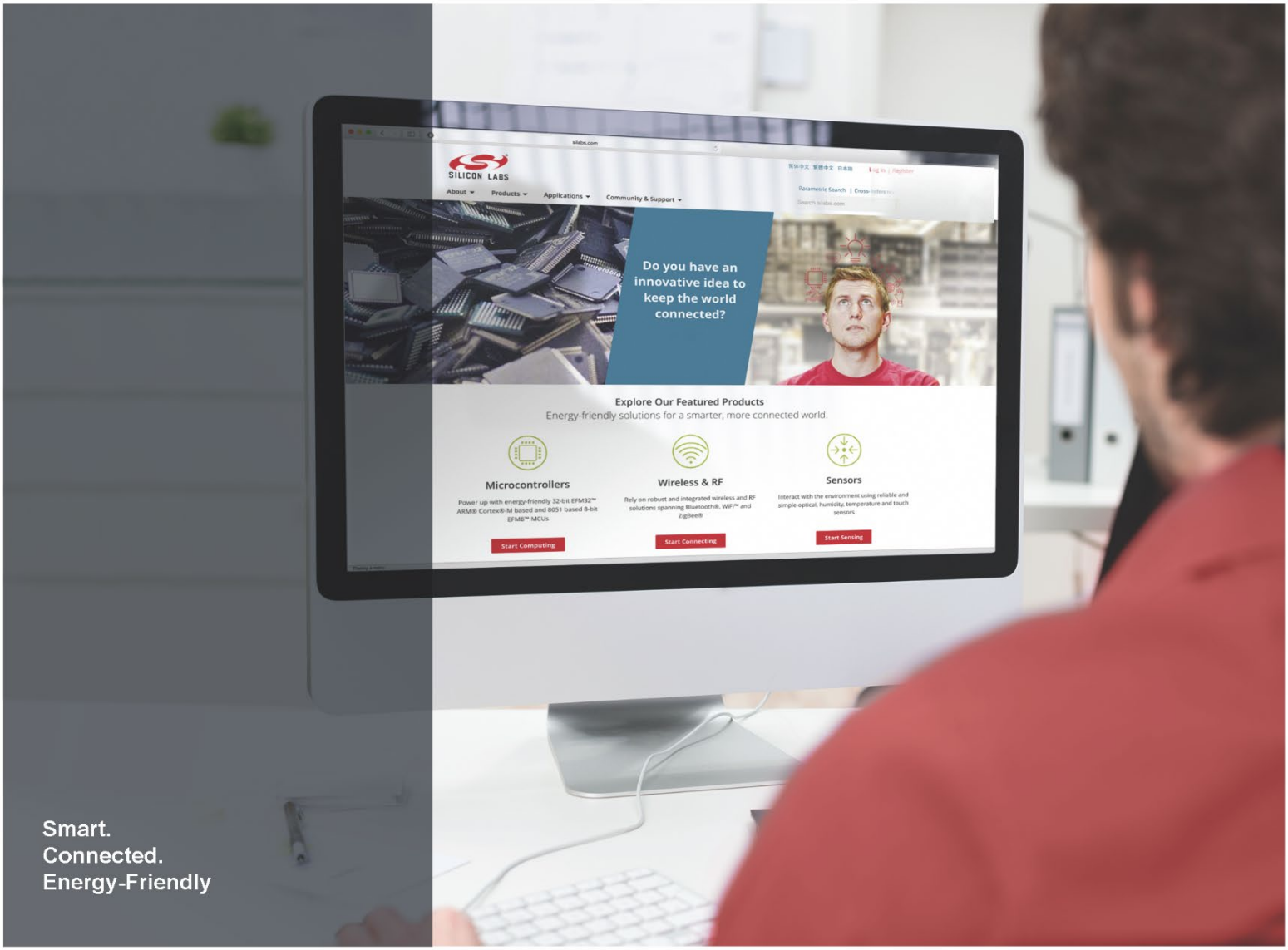
  DRBG functions can be tested on demand by doing power cycle to execute DRBG KAT.

- Document how the integrity of the health tests will be determined subsequent to implementation validation testing.
  Integrity of the firmware is verified by CRC verification of the firmware image subsequent to implementation validation.

Revision History

Revision No.	Version No.	Date	Changes
1	1.0	Oct 2014	Initial version
2	1.1	Oct 2014	1. Added description for PMK generation 2. Added description for DSA signature generation 3. Added description for DSA utility
3	1.2	Oct 2014	1. Added section 5.5(New Error Codes) 2. Added firmware upgradation error code 0xFF47 (DSA mismatch) 3. Changed generate pmk description to generate DRBG random number description
4	1.3	Nov 2014	1. Added examples for at commands 2. Changed FIPS mode enable description to FIPS bypass mode description for sections 3.1 and 4.1 3. Changed rsi_generate_pmk.c API name to rsi_generate_drbg_rand.c 4. Added section 5.5 EAP Configuration Frame 5. modified section 1.1 to indicate FIPS bypass mode usage 6. Added section 9 for describing FIPS enable flags
5	1.4	Dec 2014	1.Renamed FIPS bypass mode description to Non-FIPS mode 2.Added note about algorithm used in DSA signature 3.Added boot loader bypass mode option(5th point) in Boot loader section
6	1.5	June 2015	1. Added description for storing keys through re check key command 2. Added description for retrieving keys using PMK/EAP commands 3. Added status indication bits in card ready frame 4. Added flow charts for different commands 5. Added commands request and response IDs section 6. Remove newly added error codes section and added error code section 7. Updated user store configuration structure
7	1.6	Nov 2015	Added NOTE for DH key size support.
8	1.7	Nov 2015	1. Modified load firmware upgradation key command to load RSA 2048 bit public key. 2. Added new command to retrieve loaded firmware upgradation key from module. 3. Modified Digital signature generation procedure. 4. Added new error codes 0xFF35 and 0xFF3C. 5. updated description for error code 0xFF45 6.Updated Bootloader section with description to disable firmware upgrade

Revision No.	Version No.	Date	Changes
			through bootloader 7.Updated card ready frame (section 6.4) description by adding firmware upgradation through bootloader disable bit (BIT(5)) indication in 16th byte of card ready frame descriptor
9	1.8	Feb 2018	None.
10	1.9	April 2018	None
11	2.0	Aug 2018	1. Added rekeying description for encrypted private key. 2.Updated section 6. Differences in existing commands Usage in FIPS mode
12	2.0	Nov 2018	1.Added description of TLS_VERSION in EAP command



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com