AMSS source code distibutions (modem/baseband, wifi, bootloaders, trustzone, rpm)

NS-55[1]

[1]**Haddad Rafik -LCS**

ABSTRACT

AMSS source code distibutions (modem/baseband, wifi, bootloaders, trustzone, rpm)

-view code source of some AMSS MDM 89XY.

-debug and read code modemProc .

-review source code including source code of qualcomm proprietary modules.

-find Some function AUTH-BS

-modem Proc

MMGSDI SESSION BS CHAL

path : / modem proc / uim /api / mmgsdisessionlib.h MMGSDI_SESSION_BS_CHAL :related to the management of communication sessions and the generation of security challenges in the context of MMGSDI (Mobile Multimedia Global System for Mobile Communications (GSM) Subscriber Identity Module (SIM) Directory Interface).

- MMGSDI (Mobile Multimedia Global System for Mobile Communications Subscriber Identity Module Directory Interface) : This is a software interface that enables applications and mobile devices to communicate with the SIM (Subscriber Identity Module) or USIM (Universal Subscriber Identity Module) card of a mobile phone. The SIM card stores subscriber identification information, such as security keys, authentication information.
- Session:refers to a period of communication between the mobile device and the SIM card. A session is established when the mobile device needs to access information stored on the SIM card or perform security operations.
- "BS_CHAL: 'BS_CHAL' could mean 'Base Station Challenge' or 'Broadcast Service Challenge.' It may refer to a security process where a base station sends a challenge to a mobile device to verify its identity and ensure the security of communication.

Performs a challenge for an OTASP base station.A challenge is initiated by the ME prior to responding to an SSD Update command confirming that the SSD update process is being initiated by a legitimate source.

Description
session id : Session ID of the caller.
randseed : Random number generator seed.
response cb ptr : Pointer to the response callback.
client ref : User data returned upon completion of this command.
@return
MMGSDI_SESSION_GET_APP_CAPABILITIES Extracts all the FDN, BDN, ACL, and IMSI status provisioning application capabilities, and the phone book-related capabilities.
in file header : mmgsdilib_common.h
this is an enum for the config of function responses generate for authentication.

/** Bit masks to be used to indicate network type associated with MNC/MCC */

```
#define MMGSDI_PLMN_NETWK_TYPE_GSM_900_MASK          0x00000001
#define MMGSDI_PLMN_NETWK_TYPE_DCS_1800_MASK         0x00000002
#define MMGSDI_PLMN_NETWK_TYPE_PCS_1900_MASK         0x00000004
#define MMGSDI_PLMN_NETWK_TYPE_GSM_SAT_MASK          0x00000008
#define MMGSDI_PLMN_NETWK_TYPE_UMTS_MASK             0x00000010
#define MMGSDI_PLMN_NETWK_TYPE_LTE_MASK              0x00000020
```

for GSM and MMGSDI IMSI (important

```
    /* 110 */
  MMGSDI_GSM_IMSI,           /**< International Mobile Subscriber Identity. */
  MMGSDI_GSM_KC,             /**< Key Cipher. */
  MMGSDI_GSM_PLMN,           /**< Public Land Mobile Network selector. */
  MMGSDI_GSM_HPLMN,          /**< Home Public Land Mobile Networks search
                                  period. */
  MMGSDI_GSM_ACM_MAX,        /**< Accumulated Call Meter maximum value. */
  MMGSDI_GSM_SST,            /**< SIM Service Table. */
  MMGSDI_GSM_ACM,            /**< Accumulated Call Meter. */
  MMGSDI_GSM_GID1,           /**< Group Identifier Level 1. */
  MMGSDI_GSM_GID2,           /**< Group Identifier Level 2. */
  MMGSDI_GSM_SPN,            /**< Service Provider Name. */
```

we need to debug somme variable
path: / modem_proc / uim /api / mmgsdisessionlib

```
/*==================================================================
FUNCTION:      MMGSDI_SESSION_OTASP_MS_KEY_REQ
=================================================================*/
/************************************************************************************/

  Performs an OTASP MS Key Request command. The network sends the MS Key
  Request message to the ME.

  @param[in] session_id       Session ID of the caller.
  @param[in] randseed         Seed used to generate a true random number; sent
                              to the RUIM by the ME.
  @param[in] a_key_p_rev      A_KEY_P_REV variable sent by the base station,
                              indicating the Protocol Revision and
                              type of A-key supported by the base station.
  @param[in] param_p          Parameter modulus P is sent to the RUIM from the
                              network. The RUIM uses this with parameter
                              generator G to calculate the A-key for the SSD
              update.
  @param[in] param_g          Parameter generator G is sent to the RUIM from
                              the network. The RUIM uses this with parameter
                              modulus P to calculate the A-key for the SSD
                              update.
  @param[in] response_cb_ptr  Pointer to the response callback.
  @param[in] client_ref       User data returned upon completion of this
                              command.

  @return
```

```
  MMGSDI_SUCCESS          -- Command structure was properly generated and sent
                             to the MMGSDI command queue.\n
  MMGSDI_INCORRECT_PARAMS -- Parameters are not within the appropriate ranges.\n
  MMGSDI_CMD_QUEUE_FULL   -- Command was not queued to the MMGSDI task because
                             the maximum number of commands are already queued.

  @dependencies
  A valid 1X session ID for a provisioning application is required.
/*************************************************************************************/
```

—A valid 1X session ID —

1X/API/PUBLIC/AUTH_V

with same class UIM / used into folder 1x : used for the first step auth_v.h .
This contains all the declarations for the Authentication Task.
The Authentication Task optionally sends a report when it completes a command which produces a result which would be useful to the task which issued the command.
Other enum used to do somme results firmware and status Status values for a command . using **rendseed** AUTH_BS_CHAL_F

```
/*************************************************************************************/

    typedef struct {                  /* AUTH_BS_CHAL_F */
  auth_hdr_type   hdr;             /* Command header */
  dword           randseed;          /* Random number generator seed */
  void            (*rpt_function)( auth_rpt_type * );
                                   /* Pointer to Function to send report */
} auth_bs_chal_type;
/*************************************************************************************/
```

on code auth.c (path 1x/cp/src/auth.c)

- GENERAL DESCRIPTION This module performs all authentication functions required by IS-95A and is also the server for the R-UIM card (Removable User Identity Module) when FEATURE_UIM_RUIM is defined.
- In the presence of FEATURE_UIM_RUIM, the Authentication is performed by the RUIM card and this task acts as the server for the RUIM.
- EXTERNALIZED FUNCTIONS auth_validate_a_key

Determines if a given A-key is valid.

AKAalgo

- AKA: AKA stands for "Authentication and Key Agreement." It's a security protocol used in cellular networks to authenticate and establish a secure connection between a mobile device and the network. AKA is commonly used in 3G and 4G (LTE) networks to protect user privacy and the integrity of data transmissions.
- AlgoKeyGen: "AlgoKeyGen" likely refers to an algorithm or process for generating encryption keys in the context of AKA. Key generation is a crucial part of security protocols as it involves creating secret keys that will be used for secure communication.

XRES /SRES

```
/*************************************************************************************/

  User authentication function.
```

```
  The f2 function is used to compute the challenge response returned from the UIM
  when an authentication vector is processed.
     @code
   f2:  (K; RAND)-> RES (or XRES)
     @endcode
  f2 is a MAC function. It is computationally infeasible to derive K from
  knowledge of RAND and RES (or XRES).

  @param[in] K[]      Subscriber authentication key. \n
                      @code
                      K[0], K[1], ... K[127]
                      @endcode
                      The length of K is 128 bits. The subscriber
                      authentication key K is a long term secret key stored
                      in the USIM and the AuC.

  @param[in] fi       Type identifier.

  @param[in] RAND[]   Random challenge. \n
                      @code
                      RAND[0], RAND[1], ... RAND[127]
                      @endcode
   The length of RAND is 128 bits.

  @param[in] Fmk[]    Family key.
  @param[out] RES[]   User response. \n
                      @code
                      RES[0], RES[1], ... RES[n 1]
                      @endcode
                      The length n of RES and XRES is at most 128 bits and at
                      least 32 bits, and shall be a multiple of 8 bits. RES
                      and XRES constitute to entity authentication of the user
  to the network.

  @param[in] l_res[] Length of RES.
/*************************************************************************************/
```

CK GENERATOR

```
/*************************************************************************************/

  Cipher key (CK) derivation function.

  The f3 function is a pseudo random function used to generate a ciphering key.
  The output can be used as the CK key in an AKA authentication vector, or for
  other purposes.
     @code
   f3:  (K; RAND) -> CK
     @endcode
  f3 is a key derivation function. It is computationally infeasible to derive K
  from knowledge of RAND and CK.

  @param[in] K[]      Subscriber authentication key. \n
                      @code
                      K[0], K[1], ... K[127]
```

```
                    @endcode
                    The length of K is 128 bits. The subscriber
                    authentication key K is a long term secret key stored in
                    the USIM and the AuC.

  @param[in] fi      Type identifier.

  @param[in] RAND[]  Random challenge. \n
                    @code
                    RAND[0], RAND[1], ... RAND[127]
                    @endcode
   The length of RAND is 128 bits.

  @param[in] Fmk[]   Family key.

  @param[out] CK[]   Cipher key. \n
                    @code
                    CK[0], CK[1], ... CK[127]
                    @endcode
                    The length of CK is 128 bits. If the effective key length
                    is to be smaller than 128 bits, the most significant bits
                    of CK will carry the effective key information, whereas
                    the remaining least significant bits will be set to zero.
/***************************************************************************************/
```

KI

```
/***************************************************************************************/

  Integrity key derivation function.

  The f4 function is a pseudo random function used to generate the integrity key
  (IK).
     @code
   f4:  (K; RAND) -> IK
     @endcode
  f4 is a key derivation function. It is computationally infeasible to derive
  K from knowledge of RAND and IK.

  @param[in] K[]     Subscriber authentication key. \n
                    @code
                    K[0], K[1], ... K[127]
                    @endcode
                    The length of K is 128 bits. The subscriber authentication
                    key K is a long-term secret key stored in the USIM and
                    the AuC.

  @param[in] fi      Type identifier.

  @param[in] RAND[]  Random challenge. \n
                    @code
                    RAND[0], RAND[1], ... RAND[127]
                    @endcode
   The length of RAND is 128 bits.
```

```
  @param[in] Fmk[]   Family key.

  @param[out] IK[]   Integrity key. \n
                     @code
                     IK[0], IK[1], ... IK[127]
                     @endcode
                     The length of IK is 128 bits. If the effective key
                     length is to be smaller than 128 bits, the
                     most significant bits of IK will carry the effective key
                     information, whereas the remaining least significant
                     bits will be set zero.
/*************************************************************************************/
```

there are many algo like f1,f2 ,AKAmilenage_f1 ....
for example Void F1 description and F2345:

```
/*************************************************************************************/

  Algorithm f1.
Computes network authentication code MAC-A from key K, random
challenge RAND, sequence number SQN and authentication management
field AMF.

@param[in] K 128 bits. Subscriber key \n
@param[in] RAND 128 bits. Random challenge (RAND). \n
@param[in] SQN 48 bits. Sequence number. \n
@param[in] AMF 16 bits. Authentication management field. \n
@param[out] MAC-A 64 bits. network authentication code. \n
@param[in] OP 128 bits. Operator Variant Algorithm Configuration Field. \n

@return

f2345
  Algorithm f2345.
  Takes key K and random challenge RAND, and returns response RES,
  confidentiality key CK, integrity key IK and anonymity key AK.

@param[in] K 128 bits. Subscriber key \n
@param[in] RAND 128 bits. Random challenge (RAND). \n
@param[out] RES 48 bits. Sequence number. \n
@param[out] CK 128 bits. Authentication management field. \n
@param[out] IK   128 bits. network authentication code. \n
@param[out] AK   48 bits. Anonymity key. \n
@param[in] OP 128 bits. Operator Variant Algorithm Configuration Field. \n
@param[in] encrypt_xor indicate use OP or OPc (encrypted OP) as components of f function. \n


/*************************************************************************************/
```

we need to learn function necessary to get authentification methode on modem firmware and get PoC modem to test and debug.