osnma_core

Release 0.0.1

Aleix Galan

Aug 31, 2020

Table of Contents

Chapter 1

OSNMACore Class

Module that implements the basic OSNMA actions

class osnma_core.**OSNMACore** (*svid=1*, *pubk_path=None*)

Class that handle all the atributes and methods related to the OSNMA protocol. It stores data fields, process messages and perform verification of the different OSNMA structures. It also provides information about the internal stucture of the OSNMA message and auxiliat data such as bitmasks and tables in case the receiver wants to implement them by itself.

dms_kroot_process (*dms_kroot*, *pubk_path=None*, *nma_header=None*, *ds_length=None*)

Process the message from the OSNMA DMS-KROOT. Reads and disfragment the fields from the dms_kroot message and then proceeds with the KROOT verification calling self.kroot_verification(). Allows to load custom NMA Header, DS length and pubk path in case they are not already saved in the object.

:param dms_kroot Bits from the DMS-KROOT OSNMA message. :type dms_kroot BitArray; formated String for bin, oct or hex

:param pubk_path Path to the public key that will be used :type pubk_path String

:param nma_header NMA Header to be used in the verification :type nma_header BitArray; formated String for bin, oct or hex

:param ds_length Length of the DS field in DMS-KROOT message :type ds_length int

dms_pkr_process (*dms_pkr*)

Fragment the dms_pkr message in its fields and autenticates the new pkr key calling to self.pkr_verification

:param dms_pkr Raw DMS-PRK message :type dms_pkr BitArray; formated String for bin, oct or hex

filter_nav_data_by_adkd (*nav_data*, *adkd*)

Filters nav_data depending on the adkd parameter. Return all the nav_data to verifiy concatenated.

:param nav_data List with 15 BitArray objects containing full pages of the sub frame :type nav_data list

:param adkd Authentication Data and Key Delay that indicates the data to authenticate :type int

format_mack_data (*raw_mack_data*)

Returns a list of list with the mack blocks and each entry.

:param raw_mack_data Subframe mack data with keys :type BitArray

kroot_verification (*pub_key=None*, *hash_name=None*)

Authenticates the saved KROOT with the current Public Key or the path for the one passed as

parameter.
:param pub_key Path to the pem file with the pub_key used for the authentication :type pub_key String
:param hash_name OpenSSL hash name to override the loaded one :type hash_name String

load (*field_name*, *data*)
Load data to the OSNMa Field indicated. Also triggers secondary actions related to certain fields that modify the size of other fields or the use of certain funcions.
:param field_name Name of the OSNMA Field :type field_name String
:param data Data of the field :type data BitArray; formated String for bin, oct or hex

load_batch (*data_dict*)
Load a dictionary with OSNMA Fields to the object.
:param data_dict Dictionary with key = field_name and data the data for the field :type dict

load_floating_key (*index*, *gst_WN*, *gst_TOW*, *key*)
Loads a floating key of the chain to speed up chain authentication

mac0_verification (*mac_entry*, *nav_data*, *key*)
Compute the mac0 verification from it's entry in the first mack block, the navigation data of the subframe and it's correspondant key.
:param mac_entry First MAC entry from the first mack block. :type mac_entry BitArray
:param nav_data List with 15 BitArray objects containing full pages of the sub frame :type nav_-data list
:param key Key from the first mack block. :type key BitArray

mac_seq_verification (*mack_block*, *key*)
Verify mac seq field of the first mac entry on the first mack block.
:param mack_block List with mac entries as BitArray :type mack_block list
:param key Key of the first mack_block. :type BitArray

mac_verification (*mac_entry*, *nav_data*, *key*, *counter*)
Not implemented yed.

mack_verification (*tesla_keys*, *mack_subframe*, *nav_data*, *gst_wn=None*, *gst_tow=None*)
Authenticates a full MACK message with the correspondant keys. Allows the authentication of past MACK messages with the parameters gst_wn and gst_tow. Note: Current version does not support cross-authentication.
:param tesla_keys List with the tesla keys for the MACK message in the same order as macs. :type list
:param mack_subframe Raw MACK message to be authenticated in BitArray format. :type mack-_subframe BitArray
:param nav_data Navigation data of current satellite. Sorted in a list of 15 entries (one for each page) in BitArray format. :type nav_data list
:param gst_wn Galileo Satellite Time Week Number to overwrite the current one only for this MACK. :type gst_wn BitArray
:param gst_tow Galileo Satellite Time Time of Week to overwrite the current one only for this MACK. :type gst_tow BitArray

pkr_verification ( )
Craft and authenticates the new public key message with the saved merkle root

set_merkle_root (*merkle_root*)

Change the value of the Merkle Tree root node.

:param merkle_root The hash correspondant to the root node of the Merkle Tree :type merkle_-root BitArray

tesla_key_verification (*key*, *gst_wn*, *gst_tow*, *position*, *svid=None*)

Authenticates a TESLA key with the gst_wn and gst_tow from when it has been received. It also needs the position of the key in the mack block. The rest of the necessary data must be uploaded to the object before

:param key TESLA key to be authenticated :type key BitArray

:param gst_wn Galileo Satellite Time Week Number of the TESLA key :type gst_wn BitArray

:param gst_wn Galileo Satellite Time Time of Week of the TESLA key :type gst_wn BitArray

:param position Position of the TESLA key inside the MACK keys :type position int

:param svid Override the current svid :param svid int

- *Index*
- *Module Index*
- *Search Page*

O