HMAC DISCUSSIONS

* Are the HMAC/HPAN Keys stored and handled in a P2PE SRED compliant way?  
    
  This is a bit broad question, my answer may be not exactly answering it.  
  P2PE keys are stored and processed by ADE engine, and ADE engine is designed for P2PE. HMAC/HPAN keys are stored and processed by VSS engine, that meets PCI requirements (it is usually used for Online PIN processing), but it is not used for P2PE SRED.
* Are the HMAC/HPAN Keys exposed outside of the SRED module?  
    
  HMAC/HPAN keys are stored in VSS area, SRED keys are stored in ADE. Keys used by HMAC/HPAN are not exposed outside VSS, VIPA requests HMAC calculation from secure VSS module (part of OS).
* Is the HMAC/HPAN function executed within the SRED module?  
    
  No, this is independent module. In particular calculation of HMAC is possible also in the terminals which are not ADE-enabled.

Request to improve the security of VIPA Update Key command and prevent malicious HMAC secret loading.

New parameter **HMAC\_key\_signing**, default **false** (for backward compatibility) will be added to signed VIPA mapp\_prot.cfg file (this is standard file for security related settings).

When new HMAC VSS is loaded it has zero key set by default. Up to now VIPA allowed loading any HMAC key using Update Key command, without any additional validation of the new key.

When the above new parameter is set to **true** VIPA will require sending additional data in tag **DFED15** in Update Key command. That tag needs to provide HMAC of the new key calculated using existing HMAC key.

If the provided HMAC of the new key does not match HMAC calculated by VIPA internally the update of the key will be rejected with error code set in tag DFEC30 in VIPA response.

The current key “KSN” can be checked using Calculate HMAC command with zero data, or using Get Security Configuration VIPA command.

To reset HMAC key it is necessary to either reload VSS script (which is protected by VSR) or change the above parameter in mapp\_prot.cfg temporarily to allow plain text loading.

* You mentioned that the VSS script is protected by VSR. Can you elaborate on the protection that VSR provides?  
    
  VSR (Verishield Retain) protection is the requirement to RSA-sign any software package by the sponsor certificate installed in the terminal.  This is fundamental security mechanism that prevents installation of any unauthorized software in the terminal.  Each VSS script is VSR-signed, as in the attached. The signature is validated at OS level.
* Is it possible to restrict altering the mapp\_prot.cfg file to disable HMAC\_key\_signing after it has been enabled? My concern is that any properly signed bundle containing a mapp\_prot.cfg without the HMAC\_key\_signing value (or with the value set to false) could be used to disable this protection. If this is not possible, can you please describe what additional protection would protect against loading the default mapp\_prot.cfg file from, for instance, the default firmware load?

The same applies to mapp\_prot.cfg file, though in this case it is VIPA checking if mapp\_prot.cfg passed VSR validation.  If you (TrustCommerce) have your own Sponsor Certificate installed in the terminal only the holder of the signing card will be able to change VIPA binaries, VSS or secure config files like mapp\_prot.cfg.

Request to improve the security of VIPA Update Key command and prevent malicious HMAC secret loading.

New parameter **HMAC\_key\_signing**, default **false** (for backward compatibility) will be added to signed VIPA mapp\_prot.cfg file (this is standard file for security related settings).

When new HMAC VSS is loaded it has zero key set by default. Up to now VIPA allowed loading any HMAC key using Update Key command, without any additional validation of the new key.

When the above new parameter **HMAC\_key\_signing** is set to **true** VIPA will require sending additional data in tag **DFED15** in Update Key command. That tag needs to provide HMAC of XOR of the existing (old) key and new key, calculated using existing (old) HMAC key.  In other words it should be HMAC\_old\_key( old\_key XOR new\_key ).

If the provided HMAC value does not match HMAC calculated by VIPA internally the update of the key will be rejected with error code set in tag DFEC30 in VIPA response.

The current key “KCV” can be checked using Calculate HMAC command with zero data, or using Get Security Configuration VIPA command.

To reset HMAC key it is necessary to either reload VSS script (which is protected by VSR) or change parameter **HMAC\_key\_signing** in mapp\_prot.cfg to **false** temporarily to allow plain text HMAC key loading.

1. Used with HMAC clear text key update "Update Key [C4, 0A]"
2. When the parameter HMAC\_key\_signing is set to 1, tag DFED15 in Update Key command must be sent.
3. Tag DFED15 needs to provide HMAC of XOR of the existing (old) key and new key, calculated using existing (old) HMAC key. In other words it should be HMAC\_old\_key( old\_key XOR new\_key ).
4. HMAC\_key\_signing = 0

allowed\_direct\_vss\_execution

this is additional security feature that requires listing of scripts and macros that can be called 'directly', via **Execute VSS Script** [C4, 13] command.  
In nearly all cases that call should be blocked, and if it is not blocked then VIPA integrator should allow for execution of only these whitelisted macros.

Here is a snippet from the latest VIPA manual

*Macros that are allowed to be executed for particular VSS script are configured in mapp\_prot.cfg under section [allowed\_direct\_vss\_execution].*

*Example 44. "VSS script slot id"="[List of allowed macros]"*  
*[allowed\_direct\_vss\_execution]*  
*00=[10,11,FE,FF]*  
*3F=[10,12,FF]*

Each line corresponds to VSS script, and then whitelisted macros follow

SIGNED BUNDLES

#whitelist.dat:  
Add bin for Conduent 'Maintenance' card (passed unencrypted)  
Remove default bin ranges (all other cards are masked/encrypted)

mapp\_prot.cfg (both bundles):  
Enable SRED tokenization and KCV with VSS 06, 07  
hmac\_config\_ids=0x06, 0x07  
kcv=1

Allow numeric data in keypad status (to allow pinpad use as general user input device outside of transaction workflows)  
[keyboard\_status]  
allow\_numeric = 1

Enable new HMAC key signing feature:  
HMAC\_key\_signing = 1

mapp\_prot.cfg (second bundle only):  
Disable Update Key [C4, 0A] and Execute VSS Script [C4, 13] commands for modes that disable pin, as well as for lock0 (normal/non-SRED and lock7 (normal/SRED) VIPA modes  
[disable\_pin]  
LockedCommands=C40A,C413,DED1,DED2,DED5,DED7,DED6,11\*\*

mapp\_vsd\_prot.cfg:  
Enable pan\_cache\_timeout default of 300 seconds (to allow use of Online PIN [and Generate HMAC] on SRED device)  
pan\_cache\_timeout=300

Alter default track2 masking\_scheme to include first 6 and last 4 digits (MF6). Note: track1 always completely masked

# Values in this section override defaults, set in [default] section  
masking\_scheme=MF6

Set default section masking\_scheme for track2 to include first 6 and last 4 digits (MF6) of PAN. Note: track1 always completely masked

# Masking schemes  
masking\_scheme=MF6

TESTING BUNDLES

I loaded ['dl.bundle.Sphere\_config20191210a.tar'](file:///C:\Python\upload\SphereConfig\hmac), rebooted the device, the attempted to load the HMAC secrets (using Update Key [C4, 0A]) without sending the DFED15 tag. The response returned (via the Test Harness) was :

IN: b'dfed159f27'

Was this the expected error? The documentation mentioned tag DFEC30 would be used on error, but I did not see that in the response.

TEST SCRIPT: C:\Python\TC\_load\_hmac\_keys.py

In this case tag DFED15 (key signature) is mandatory so 9F27 is correct VIPA response indicating missing data.

After loading the 'dl.bundle.Sphere\_config20191210a.tar', if I send an Update Key [C4, 0A] with the DFED15 tag, but with an incorrect value, I do see the response:

IN: b'dfdf30012a9f0d'

Are the response value '2A', and the status '9F0D', expected response values for using an incorrect DFED15 value?

'2A' is most like our security middleware error code indicating some key-related problem - we are investigating.

**If the signature was not correct then the expected error code would be 29.**

Security related error codes are described in Appendix E: "Security Application Error Codes".