ARM XMC - experiment No. 11

Linux openSSL -cryptographic mechanisms to support web server authentication

Overview and purpose:

The experiment explore Linux OpenSSL command to realise the main cryptographic operations like symmetric encryption, public-key encryption, digital signature, hash functions.

The main OpenSSL command used :

ca - Create certificate authorities.

dgst - Compute hash functions.

enc - Encrypt/decrypt using secret key algorithms. It is possible to generate using a password or directly a secret key stored in a file.

genrsa - Generate a pair of public/private key for the RSA algorithm.

password Generation of Ŧ hashed passwords ƫ.

pkcs12 - Manage information according to the PKCS #12 standard. pkcs7 Manage information according to the PKCS #7 standard.

rand Generation of pseudo-random bit strings.

rsa - RSA data management.

rsautl Encrypt/decrypt or sign/verify signature with RSA. verify Checkings for X509. x509 Data managing for X509.

Resources

Hardware: Raspberry pi 2 with Debain Linux, Internet acces,
Software: OpenSSL ,IBM TPM simulator
Chip : SLB9645TT1.2 PG-TSSOP-28

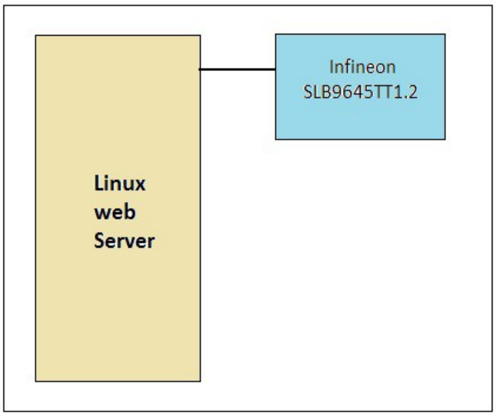

Fig.11.1 Educational Trusted Web Server

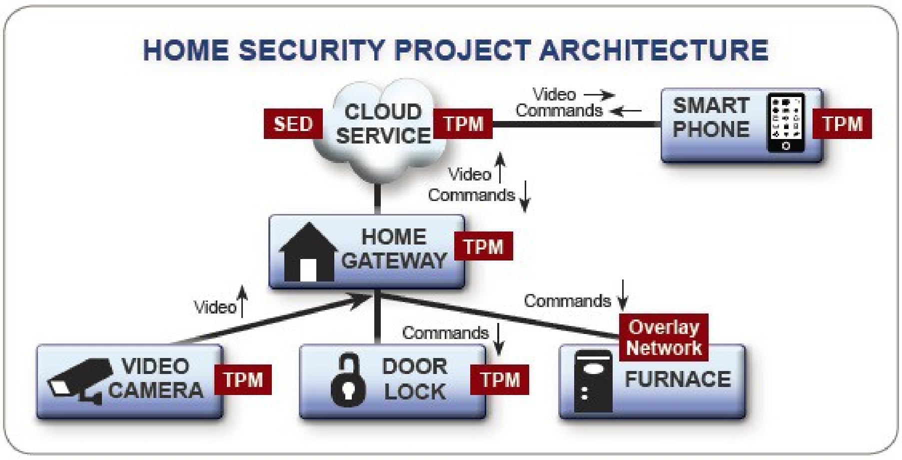

Fig.2 Trusted Computer Group – HS Project Architecture

Material and methode

Install OpenSSL using next comand line:

> sudo apt-get install OpenSSL View openssl version using command line:

> openssl version View openssl command using command line line:
>openssl list-standard-commands

```
asn1parse
ca
ciphers
cms
crl
crl2pkcs7
dgst
dh
dhparam
dsa
dsaparam
ec
ecparam
enc
engine
errstr
gendh
gendsa
genpkey
genrsa
nseq
ocsp
passwd
pkcs12
pkcs7
pkcs8
pkey
pkeyparam
pkeyutl
prime
rand
req
rsa
rsautl
s_client
s_server
s_time
sess_id
smime
speed
spkac
srp
ts
verify
version
x509
```

View list of secret key algorithms using command line:
>openssl list-cipher-commands

Use base64 algorithm witch allows to code binary information with alfanumeric character :

```
root@raspberrypi:~# touch n1.txt
root@raspberrypi:~# echo "abcdefgh" > n1.txt
root@raspberrypi:~# openssl enc -base64 -in n1.txt
YWJjZGVmZ2gK
root@raspberrypi:~#
```

Use secret key algorithm AES to encrypt a file using encryption password:

```
root@raspberrypi:~# openssl enc -aes-256-cbc -in t1.txt -out t1_cripted.bin
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
root@raspberrypi:~# ls -l
total 72
-rw-r--r-- 1 root root   48 Jan 26 09:23 t1_cripted.bin
-rw-r--r-- 1 root root   30 Jan 26 09:20 t1.txt
drwxr-xr-x 5  508  508 4096 Apr  2  2014 tpm4720
```

Decript the encripted file :

```
root@raspberrypi:~# openssl enc -aes-256-cbc -d -in t1_cripted.bin
enter aes-256-cbc decryption password:
Acest mesaj este confidential
```

Test RSA with private and public key
a. Generate a pair of public/private key RSA 1024 bits:

> openssl genersa -out duokey.pem 1024

```
root@raspberrypi:~# openssl genrsa -out duokey.pem 1024
Generating RSA private key, 1024 bit long modulus
...........++++++
..............++++++
e is 65537 (0x10001)
```

The duokey.pem file include both public and private key.
The private key is coded using the Privacy Enhanced Email (PEM) standard.
RSA private key can bee examine with:

> cat duokey.pem
More information about generated file ca be done with:
> openssl rsa -in key.pem -text -noout

```
root@raspberrypi:~# openssl rsa -in duokey.pem -text -noout
Private-Key: (1024 bit)
modulus:
    00:c3:5b:a3:18:7c:05:c3:e9:06:05:6c:99:50:1b:
    55:cd:1d:ca:a9:2a:fc:0d:be:3d:e0:e6:14:bb:8d:
    21:95:1a:ca:f6:7a:c0:e4:d5:d4:f1:20:6c:24:64:
    a0:a0:fd:e1:4a:0d:d2:b6:e2:1b:79:8d:cd:d6:55:
    05:fb:75:4c:7a:bc:a3:a7:f9:d7:ea:2a:71:07:e2:
    c6:90:57:56:3c:8a:59:d8:e8:8c:5e:72:73:d5:dc:
    5f:96:d2:d5:8d:74:8d:9e:8a:f5:0d:9e:86:7b:83:
    71:59:56:93:05:81:2c:e8:3c:27:9e:90:87:ce:4e:
    de:81:f1:00:a3:92:00:a7:85
publicExponent: 65537 (0x10001)
privateExponent:
    21:a2:e1:8e:11:ac:9c:72:be:ca:a8:4c:2d:72:c7:
    6f:2d:b5:fb:32:9f:7b:6e:4a:11:33:0c:56:ce:91:
    fd:ae:43:4b:f5:0e:c9:57:d7:f6:3c:72:e2:41:41:
    36:f6:ff:97:54:91:f7:53:2f:0f:da:ce:9a:1c:c1:
    8e:ee:3f:60:85:34:1a:4b:c9:9e:c5:fa:f9:6b:ff:
    59:bc:5e:31:88:31:06:f2:60:71:b6:a7:e0:c6:69:
    02:3d:72:a4:93:f2:13:fe:11:62:ea:cb:d3:0d:ce:
    d0:32:45:e8:5c:61:41:2a:02:28:20:fe:cd:3c:f6:
    83:2e:b1:15:be:f4:3f:81
prime1:
```

```
prime1:
    00:f8:40:63:d1:ed:73:9f:f3:ab:f6:ea:14:24:03:
    1a:42:11:3b:cc:7e:67:01:30:ab:f9:4d:c9:02:80:
    43:84:31:20:03:0f:d8:79:f9:13:29:4d:6b:35:92:
    b1:fc:0c:f5:57:d4:f0:ee:2b:0c:5f:c9:ae:af:e4:
    18:87:43:a8:e9
prime2:
    00:c9:74:9c:46:51:7e:a0:69:3a:01:42:a0:c7:3e:
    da:a0:cb:ac:5e:67:b6:d9:ad:5a:05:b1:de:5d:6d:
    1c:0d:6e:5f:7d:e1:bc:a0:16:c2:10:97:96:9f:8c:
    04:b2:ea:76:95:7a:77:d7:05:39:d1:3b:1e:18:16:
    5c:c2:75:28:3d
exponent1:
    00:a3:30:68:ad:d2:02:c4:ed:c0:68:52:9d:a4:c2:
    a9:5b:2e:ca:f9:75:4a:2e:dd:18:df:8c:43:8b:b2:
    57:2f:a9:bf:5a:63:eb:30:db:0b:be:85:d6:e8:e3:
    a2:be:86:a2:2c:f7:9c:dd:63:4d:02:16:a9:0f:94:
    c8:8e:fa:6a:29
exponent2:
    7f:5b:07:dc:70:72:a3:c8:42:12:3f:e3:d2:72:0a:
    d6:b2:4b:c2:d6:c0:42:b6:93:7d:9c:27:9e:5a:76:
    ec:8a:c5:35:98:7a:9d:9e:25:8b:45:b9:c4:1d:49:
    6f:2e:48:4f:51:3f:e7:9f:f7:20:2d:c6:65:a5:e5:
    78:4d:86:65
```


```
    18:87:43:a8:e9
prime2:
    00:c9:74:9c:46:51:7e:a0:69:3a:01:42:a0:c7:3e:
    da:a0:cb:ac:5e:67:b6:d9:ad:5a:05:b1:de:5d:6d:
    1c:0d:6e:5f:7d:e1:bc:a0:16:c2:10:97:96:9f:8c:
    04:b2:ea:76:95:7a:77:d7:05:39:d1:3b:1e:18:16:
    5c:c2:75:28:3d
exponent1:
    00:a3:30:68:ad:d2:02:c4:ed:c0:68:52:9d:a4:c2:
    a9:5b:2e:ca:f9:75:4a:2e:dd:18:df:8c:43:8b:b2:
    57:2f:a9:bf:5a:63:eb:30:db:0b:be:85:d6:e8:e3:
    a2:be:86:a2:2c:f7:9c:dd:63:4d:02:16:a9:0f:94:
    c8:8e:fa:6a:29
exponent2:
    7f:5b:07:dc:70:72:a3:c8:42:12:3f:e3:d2:72:0a:
    d6:b2:4b:c2:d6:c0:42:b6:93:7d:9c:27:9e:5a:76:
    ec:8a:c5:35:98:7a:9d:9e:25:8b:45:b9:c4:1d:49:
    6f:2e:48:4f:51:3f:e7:9f:f7:20:2d:c6:65:a5:e5:
    78:4d:86:65
```


```
coefficient:
    00:aa:9b:cf:e1:68:63:b7:e2:6f:1a:c4:23:c6:b6:
    d7:6a:58:be:74:de:fe:58:60:dc:73:42:6e:e7:86:
    ee:b5:18:61:2f:78:e9:81:17:55:bf:8d:85:3f:9d:
    78:d3:06:31:fb:d6:2a:4e:9c:6e:08:18:bb:8d:b3:
    76:07:ff:c4:af
```

Encript private key can using next command line:

```
> openssl rsa -in duokey.pem -des3 -out enc_private.pem
```

Extract public key from generated duokey file :

> openssl rsa -in duokey.pem -pubout -out public.pem

More helpful information:

1. http://www.trustedcomputinggroup.org/resources/tcg_byod_architects_guide
2.
3. http://www.trustedcomputinggroup.org/developers/virtualized_platform
4. http://www.trustedcomputinggroup.org/developers/virtualized_platform
5. http://www.trustedcomputinggroup.org/resources/endpoint_security_hardware_roots_of_tru st

6. http://www.sans.org/reading_room/whitepapers/services/analysis-building-blocks-attack-vectors-unifiedextensible-
7. firmware_34215