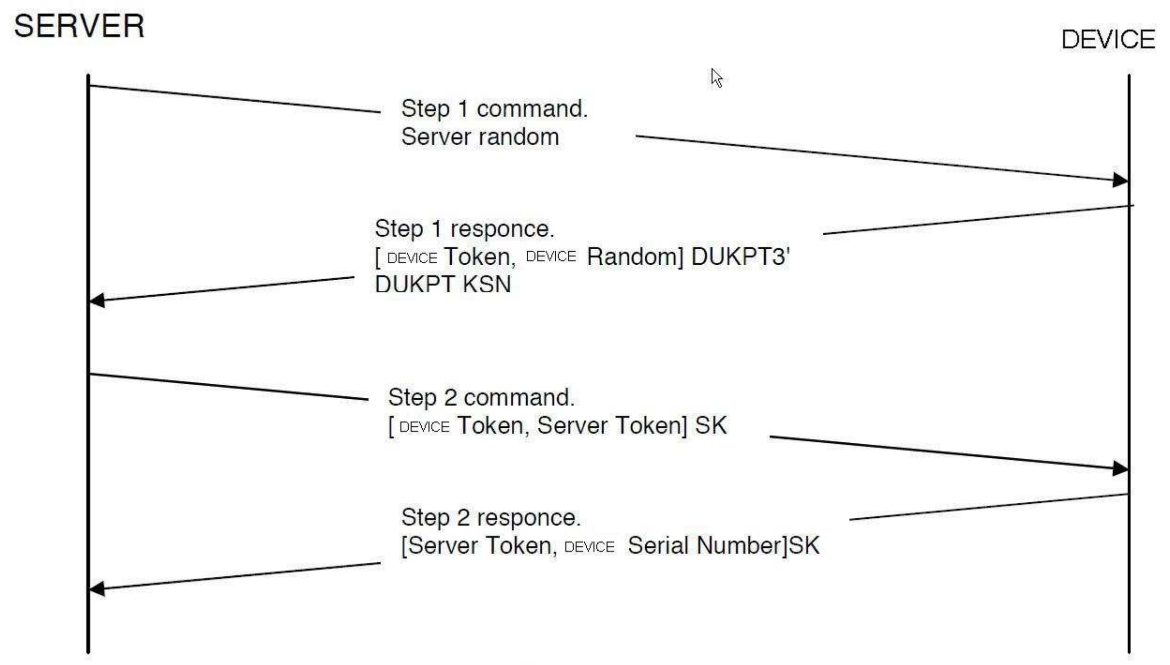
System Authenticate flow.


SERVER
DEVICE
Step 1 command.
Server random
Step 1 responce.
[DEVICE Token, DEVICE Random] DUKPT3'
DUKPT KSN
Step 2 command.
[DEVICE Token, Server Token] SK
Step 2 responce.
[Server Token, DEVICE Serial Number]SK

System Authentication Flow

Step 1.
- Server sends 16 bytes of Server random.
- Device stores the Server random and sends the response to the server.
- Device response includes 32 bytes device data block and KSN of the DUKPT engine 3 current key.
Device data block includes 16 bytes of device token and 16 bytes of device random. Device data block is encrypted by CBC method with zero ini vector.
3DES key for encryption based on the current key of DUKPT engine 3. After encryption DUKPT engine 3 make new keys.
The encryption key create process see in ANSI X29 24.
Derived key is the current 3DES key from DUKPT engine 3. Variant constant key is:
00 00 00 00 00 FF 00 00 00 00 00 00 00 FF 00 00.
- Server receives the device response, decrypts the device data block and stores device token and device random. For decryption of the device data block the server creates the key by ANSI X29 24.
Derived key is the current 3DES key by KSN from device response. Variant constant key is:
00 00 00 00 00 FF 00 00 00 00 00 00 00 FF 00 00.
Step 2.
- Server and device prepare 16 bytes 3DES session key (SK) as XOR of server random and device random.
- Server sends 32 bytes data block, which includes 16 bytes of device Token and 16 bytes of Server Token.
The data block encrypted with 3DES session key (SK).
- Device receives the step 2 command, decrypts the data block with 3DES session key (SK) and compares the received device token with the device Token, which sent to the server in the step 1 response. If they are same, the server is authenticated and device continues the step 2. Else authenticate is stopped.
- Device sends the response to the server. The response is 32 bytes data block, which includes 16 bytes of Server Token and 16 bytes of device Serial Number. The data block encrypted with 3DES session key (SK).

- Server receives the step 2 response, decrypts the data block with 3DES session key (SK) and compares the received Server Token with the Server Token, which sent to the device in the step 2 command. If they are same, the device is authenticated and the server can device Serial Number, if need. Else authenticate is stopped.

Message MAC

For secure message integrity (MAC) the device uses the CMAC algorithm. The key used is the Session Key (SK) derived during the System Authentication. Hence System Authentication is required for CMAC support. For messages that have both encryption and MAC the MAC process is applied to the cleartext data not the encrypted data.

http://en.wikipedia.org/wiki/CMAC

To generate an ℓ-bit CMAC tag (t) of a message (m) using a b-bit block cipher (E) and a secret key (k), one first generates two b-bit sub-keys (k_1 and k_2) using the following algorithm (this is equivalent to multiplication by x and x^2 in a finite field $GF(2^b)$). Let $\ll$ signify a standard left-shift operator:

1. Calculate a temporary value $k_0 = E_k(0)$.
2. If $\text{msb}(k_0) = 0$, then $k_1 = k_0 \ll 1$, else $k_1 = (k_0 \ll 1) \oplus C$; where C is a certain constant that depends only on b. (Specifically, C is the non-leading coefficients of the lexicographically first irreducible degree-b binary polynomial with the minimal number of ones.)
3. If $\text{msb}(k_1) = 0$, then $k_2 = k_1 \ll 1$, else $k_2 = (k_1 \ll 1) \oplus C$.
4. Return keys $(k_{1,2})$ for the MAC generation process.

As a small example, suppose $b = 4$, $C = 0011_2$, and $k_0 = E_k(0) = 0101_2$. Then $k_1 = 1010_2$ and $k_2 = 0100 \oplus 0011 = 0111_2$.

The CMAC tag generation process is as follows:

1. Divide message into b-bit blocks $m = m_1 \parallel \ldots \parallel m_{n-1} \parallel m_n'$ where $m_1, \ldots, m_{n-1}$ are complete blocks. (The empty message is treated as 1 incomplete block.)
2. If m_n' is a complete block then $m_n = k_1 \oplus m_n'$ else $m_n = k_2 \oplus (m_n' \parallel 10\ldots0_2)$.
3. Let $c_0 = 00\ldots0_2$.
4. For $i = 1,\ldots, n$, calculate $c_i = E_k(c_{i-1} \oplus m_i)$.
5. Output $t = \text{msb}_\ell(c_n)$.

The verification process is as follows:

1. Use the above algorithm to generate the tag.
2. Check that the generated tag is equal to the received tag.

Test cases

Host random: C9E61721D09D0AC4BE3E52CD2841621C
Host token: A52580B2B7B130ECC32D68E2EEB6D77A
Slave random: A59ABAFB00C2EA52DF27534F9E1A80A0
Slave token: 442E37C763DEE707A05ED7065D70EC64
Symmetric Key: 6C7CADDAD05FE09661190182B65BE2BC

Test 1, Symmetic key calculation:
Slave Random XOR'ed with Host Random should yield Symmetric Key

Test 2, Decrypt with symmetric key (3DES):
45E6E5CA64E355B3C1549138912248166A65B2AB9C13889E6EFB25B2D8EAD8B5 should yield 32 bytes
the first 16 bytes should equal to the host token
the last 16 bytes (serial number) should be equal to
[0] = 0x33
[1] = 0x32
[2] = 0x34
[3] = 0x38
[4] = 0x30
[5] = 0x30
[6] = 0x33
[7] = 0x32
[8] = 0x32
[9] = 0xFF
[10] = 0xFF
[11] = 0xFF
[12] = 0x0
[13] = 0x0
[14] = 0x0
[15] = 0x0

Test 3:
Given input
F0369F3501229F1A0202809F3303E0F8C89F4005F000B0A0015F2A0209785F360102DF67080102000000000
0009F1E083132333435363738
Generate with the symmetric key a CMAC (3DES): F874E98BDFC1F12D