Thistle Type A Interface for Encryption Version 1

CML00079-01

Code Magus Limited (England reg. no. 4024745)
Number 6, 69 Woodstock Road
Oxford, OX2 6EY, United Kingdom
www.codemagus.com

August 16, 2016

Contents

1 Thistle Interface

1.1 Introduction

The *Thistle* Type A Interface `cryptotai` is an interface which allows *Thistle* scripts (packages and usecases) to encrypt and decrypt values using the DES encryption algorithm as implemented in the `OpenSSL` Library as well as the Code Magus Limited `cmlcrypto` library.

The *Thistle* Type A Interface `cryptotai` currently supports only the DES ecb3 routine for encryption and decryption. It is expected that support for more routines will be added in the future.

This document shows how the *Thistle* Type A Interface `cryptotai` can be used by *Thistle* scripts to perform key based encryption and decryption using the DES ecb3 routine.

1.2 *Thistle* Interface Definition for `cryptotai`

The *Thistle* Interface Definition for `cryptotai` is a normal Type A Interface to *Thistle*.

```
interface CRYPTO;

{
  -- File: CRYPTO.tid
  --
  -- This is the Thistle Interface Definition for the Type A Interface to the
  -- Code Magus Limited Encryption Library. This interface makes the
  -- Encryption Methods available to Thistle Scripts.
  --
  -- Copyright (c) 2010 by Code Magus Limited. All rights reserved.
  --
  -- Author: Stephen R. Donaldson [stephen@codemagus.com].
  --

  -- $Author: justin $
  -- $Date: 2011/01/17 10:52:24 $
  -- $Id: CRYPTO.tid,v 1.3 2011/01/17 10:52:24 justin Exp $
  -- $Name:  $
  -- $Revision: 1.3 $
  -- $State: Exp $
  --
}

     type : typea;
     module : "C:\Program Files\CodeMagusLimited\eresiavte\lib\cryptotai.dll";
     init : cryptotai_init;
end.
```

2 `cryptotai` Type A Interface reference

2.1 `cryptotai` Type A Interface Methods

There are currently five methods exposed by the `cryptotai` Type A Interface. They are shown here together with the names of the parameters and with the associate library function name.

2.1.1 `DESKeyLoad()`

`DESKeyLoad()` Function `DESKeyLoad()` is the Type A Interface wrapper function for the `cmlcrypto` library function `cmlcrypto_deskeys_load()`. For a description of the method, refer to the corresponding `cmlcrypto` Library documentation.

This function maps between the *Thistle* called function and the cmlcrypto library function. If the cmlcrypto library operation fails then this writes the error message to the log and reports the failure to the executing script.

If function DESKeyLoad() successfully loads a *key set* from the *key file*, then the function returns the architecture dependent *DES key schedule* for which all subsequent calls for encryption and decryption in *Thistle* using the *key set* will be required.

Parameters for Function DESKeyLoad()

keyfilename The keyfilename is the file name string supplied to the DESKeyLoad() function. It is a text string in the format as required by the first parameter of the DESKeyLoad() function. This string names the path and file name of the key file to be opened.

keyset The keyset parameter is required and names the DES key set to be used from the file named in the keyfilename parameter.

2.1.2 ECBTDESEncrypt()

ECBTDESEncrypt() Function ECBTDESEncrypt() is the Type A Interface wrapper function for the OpenSSL library function DES_ecb3_encrypt(). For a description of the method, refer to the corresponding OpenSSL Library documentation.

This function maps between the *Thistle* called function and the OpenSSL library function. If the OpenSSL library operation fails then this writes the error message to the log and reports the failure to the executing script.

If function ECBTDESEncrypt() successfully encrypts the encryptvalue using the keyschedule, the encrypted value will be returned as a blob for subsequent use.

Parameters for Function ECBTDESEncrypt()

keyschedule The keyschedule is the the architecture dependent *DES key schedule* obtained as a result of a successful call to DESKeyLoad(). This parameter is required and is expected to be a blob.

encryptvalue The encryptvalue parameter is required and is expected to be a blob. It is the value intended to be encrypted. This value must have a length of modular 8[1]. This is facilitated by the function PadLowValues().

[1] The length must be a multiple of 8.

2.1.3 ECBTDESDecrypt()

ECBTDESDecrypt() Function ECBTDESDecrypt() is the Type A Interface wrapper function for the OpenSSL library function DES_ecb3_encrypt(). For a description of the method, refer to the corresponding OpenSSL Library documentation.

This function maps between the *Thistle* called function and the OpenSSL library function. If the OpenSSL library operation fails then this writes the error message to the log and reports the failure to the executing script.

If function ECBTDESDecrypt() successfully decrypts the decryptvalue using the keyschedule, the decrypted value will be returned as a blob for subsequent use.

The blob returned by this function will always have a length of modular 8[2]. The blob can be modified by the function TrimLowValues() in order to remove any padded low values.

Parameters for Function ECBTDESDecrypt()

keyschedule The keyschedule is the the architecture dependent *DES key schedule* obtained as a result of a successful call to DESKeyLoad(). This parameter is required and is expected to be a blob. The *DES key schedule* must obtained from the same DES key set as the encrypted value, otherwise decryption will not be successful.

decryptvalue The decryptvalue parameter is required and is expected to be a blob. It is the value intended to be decrypted.

2.1.4 KeyManLoadZoneKeys()

anLoadZoneKeys() Function KeyManLoadZoneKeys() is the Type A Interface wrapper function for the keyman library function keyman_load_zone_keys(). For a description of the method, refer to the corresponding keyman Library documentation.

This function maps between the *Thistle* called function and the keyman library function. If the keyman library operation fails then this writes the error message to the log and reports the failure to the executing script.

If function KeyManLoadZoneKeys() fails to load the supplied zone keys, an error is reported.

There is no return value for this function.

[2]The length must be a multiple of 8.

Parameters for Function `KeyManLoadZoneKeys()`

`ZoneKeys` Required. The zone keys to be loaded.

2.1.5 `KeyManGetZoneKeys()`

`ManGetZoneKeys()` Function `KeyManGetZoneKeys()` is the Type A Interface wrapper function for the `keyman` library function `keyman_display_zone_keys()`. For a description of the method, refer to the corresponding `keyman` Library documentation.

This function maps between the *Thistle* called function and the `keyman` library function. If the `keyman` library operation fails then this writes the error message to the log and reports the failure to the executing script.

It will return the display version of the zone master keys.

If function `KeyManGetZoneKeys()` fails, an error is reported.

There are no input parameters for this function.

2.1.6 `KeyManLoadSessionKeys()`

`oadSessionKeys()` Function `KeyManLoadSessionKeys()` is the Type A Interface wrapper function for the `keyman` library function `keyman_load_session_keys()`. For a description of the method, refer to the corresponding `keyman` Library documentation.

This function maps between the *Thistle* called function and the `keyman` library function. If the `keyman` library operation fails then this writes the error message to the log and reports the failure to the executing script.

If function `KeyManLoadSessionKeys()` fails to load the supplied session keys, an error is reported.

There is no return value for this function.

Parameters for Function `KeyManLoadSessionKeys()`

`SessionKeys` Required. The session keys to be loaded.

2.1.7 `KeyManGetSessionKeys()`

`GetSessionKeys()` Function `KeyManGetSessionKeys()` is the Type A Interface wrapper function for the `keyman` library function `keyman_display_session_keys()`.

For a description of the method, refer to the corresponding keyman Library documentation.

This function maps between the *Thistle* called function and the keyman library function. If the keyman library operation fails then this writes the error message to the log and reports the failure to the executing script.

It will return the display version of the session keys.

If function KeyManGetSessionKeys() fails, an error is reported.

There are no input parameters for this function.

2.1.8 EncryptPinSessionKeyDecoded()

sionKeyDecoded() Function EncryptPinSessionKeyDecoded() is the Type A Interface wrapper function for the keyman library function keyman_encrypt_pin_ses_key(). For a description of the method, refer to the corresponding keyman Library documentation.

This function maps between the *Thistle* called function and the keyman library function. If the keyman library operation fails then this writes the error message to the log and reports the failure to the executing script.

This function encrypts the pin from the supplied parameters pan and pin into parameter pinblock, using the keys loaded by the last call to KeyManLoadSessionKeys(), and returns the decrypted pin block;

If function EncryptPinSessionKeyDecoded() fails, an error is reported.

Parameters for Function EncryptPinSessionKeyDecoded()

PAN Required. The plastic number.

PIN Required. The PIN number.

2.1.9 EncryptPinSessionKeyEncoded()

sionKeyEncoded() Function EncryptPinSessionKeyDecoded() is the Type A Interface wrapper function for the keyman library function keyman_encrypt_pin_ses_key(). For a description of the method, refer to the corresponding keyman Library documentation.

This function maps between the *Thistle* called function and the keyman library function. If the keyman library operation fails then this writes the error message to the log and reports the failure to the executing script.

This function encrypts the pin from the supplied parameters pan and pin into parameter pinblock, using the keys loaded by the last call to KeyManLoadSessionKeys(), and returns the encrypted pin block;

If function `EncryptPinSessionKeyEncoded()` fails, an error is reported.

Parameters for Function `EncryptPinSessionKeyEncoded()`

`PAN` Required. The plastic number.

`PIN` Required. The PIN number.

2.1.10 `KeyManDecryptPinBlockKey()`

`yptPinBlockKey()` Function `KeyManDecryptPinBlockKey()` is the Type A Interface wrapper function for the `keyman` library function `keyman_decrypt_pin_block()`. For a description of the method, refer to the corresponding `keyman` Library documentation.

This function maps between the *Thistle* called function and the `keyman` library function. If the `keyman` library operation fails then this writes the error message to the log and reports the failure to the executing script.

This function decrypts the pin block using the keys loaded by the last call to KeyManLoadSessionKeys() and returns the encoded key.

If function `KeyManDecryptPinBlockKey()` fails, an error is reported.

Parameters for Function `KeyManDecryptPinBlockKey()`

`PAN` Required. The plastic number.

`PIN` Required. The PIN number.

2.1.11 `KeyManDecryptPinBlockPin()`

`yptPinBlockPin()` Function `KeyManDecryptPinBlockPin()` is the Type A Interface wrapper function for the `keyman` library function `keyman_decrypt_pin_block()`. For a description of the method, refer to the corresponding `keyman` Library documentation.

This function maps between the *Thistle* called function and the `keyman` library function. If the `keyman` library operation fails then this writes the error message to the log and reports the failure to the executing script.

This function decrypts the pin block using the keys loaded by the last call to KeyManLoadSessionKeys() and returns the PIN.

If function `KeyManDecryptPinBlockPin()` fails, an error is reported.

Parameters for Function `KeyManDecryptPinBlockPin()`

`EncodedPinBlock` Required. The encoded PIN block.

`PAN` Required. The plastic number.

2.1.12 `KeyManParseNipsSvcMsgKeys()`

`NipsSvcMsgKeys()` Function `KeyManParseNipsSvcMsgKeys()` is the Type A Interface wrapper function for the `keyman` library function `keyman_parse_nips_svc_msg()`. For a description of the method, refer to the corresponding `keyman` Library documentation.

This function maps between the *Thistle* called function and the `keyman` library function. If the `keyman` library operation fails then this writes the error message to the log and reports the failure to the executing script.

This function parses the BICISO NIPS Service Message. This is the field S-123 Cryptographic Service Message received on the key exchange.

If function `KeyManParseNipsSvcMsgKeys()` fails, an error is reported, otherwise the session keys are returned.

Parameters for Function `KeyManParseNipsSvcMsgKeys()`

`ServiceMessage` Required. The service message to parse.

2.1.13 `KeyManParseNipsSvcMsgResponse()`

`SvcMsgResponse()` Function `KeyManParseNipsSvcMsgResponse()` is the Type A Interface wrapper function for the `keyman` library function `keyman_parse_nips_svc_msg()`. For a description of the method, refer to the corresponding `keyman` Library documentation.

This function maps between the *Thistle* called function and the `keyman` library function. If the `keyman` library operation fails then this writes the error message to the log and reports the failure to the executing script.

This function parses the BICISO NIPS Service Message. This is the field S-123 Cryptographic Service Message received on the key exchange.

If function `KeyManParseNipsSvcMsgResponse()` fails, an error is reported, otherwise the formatted response of the service message is returned.

Parameters for Function `KeyManParseNipsSvcMsgResponse()`

`ServiceMessage` Required. The service message to parse.

2.1.14 `VisaGenerateCVV()`

`isaGenerateCVV()` Function `VisaGenerateCVV()` is the Type A Interface wrapper function for the `visacvv` library function `visacvv_computecvv()`. For a description of the method, refer to the corresponding `visacvv` Library documentation.

This function maps between the *Thistle* called function and the `keyman` library function. If the `keyman` library operation fails then this writes the error message to the log and reports the failure to the executing script.

This function uses the VISA method to to compute the CVV for a given set of keys and card details.

If function `VisaGenerateCVV()` fails, an error is reported.

Parameters for Function `VisaGenerateCVV()`

`DesKey1` Required. DES key.

`DesKey2` Required. DES key.

`Plastic` Required. PAN or Plastic number.

`ExpiryDate` Required. Expiry date of the card.

`ServiceCode` Required. The service code of the card.

2.1.15 `AmexGenerateCSC3()`

`exGenerateCSC3()` Function `AmexGenerateCSC3()` is the Type A Interface wrapper function for the `amexcsc` library function `amexcsc_computecvv_values()`. For a description of the method, refer to the corresponding `amexcsc` Library documentation.

This function maps between the *Thistle* called function and the `keyman` library function. If the `amexcsc` library operation fails then this writes the error message to the log and reports the failure to the executing script.

This function uses the AMEX Network Specification of 2008 to compute the three character CSC value.

If function `AmexGenerateCSC3()` fails, an error is reported.

Parameters for Function `AmexGenerateCSC3()`

`DesKey1` Required. DES key.

`DesKey2` Required. DES key.

`Plastic` Required. PAN or Plastic number.

`ExpiryDate` Required. Expiry date of the card.

2.1.16 `AmexGenerateCSC4()`

`exGenerateCSC4()` Function `AmexGenerateCSC4()` is the Type A Interface wrapper function for the `amexcsc` library function `amexcsc_computecvv_values()`. For a description of the method, refer to the corresponding `amexcsc` Library documentation.

This function maps between the *Thistle* called function and the `keyman` library function. If the `amexcsc` library operation fails then this writes the error message to the log and reports the failure to the executing script.

This function uses the AMEX Network Specification of 2008 to compute the four character CSC value.

If function `AmexGenerateCSC4()` fails, an error is reported.

Parameters for Function `AmexGenerateCSC4()`

`DesKey1` Required. DES key.

`DesKey2` Required. DES key.

`Plastic` Required. PAN or Plastic number.

`ExpiryDate` Required. Expiry date of the card.

2.1.17 `AmexGenerateCSC5()`

`exGenerateCSC5()` Function `AmexGenerateCSC5()` is the Type A Interface wrapper function for the `amexcsc` library function `amexcsc_computecvv_values()`. For a description of the method, refer to the corresponding `amexcsc` Library documentation.

This function maps between the *Thistle* called function and the keyman library function. If the amexcsc library operation fails then this writes the error message to the log and reports the failure to the executing script.

This function uses the AMEX Network Specification of 2008 to compute the five character CSC value.

If function AmexGenerateCSC5() fails, an error is reported.

Parameters for Function AmexGenerateCSC5()

DesKey1 Required. DES key.

DesKey2 Required. DES key.

Plastic Required. PAN or Plastic number.

ExpiryDate Required. Expiry date of the card.

2.1.18 PadLowValues()

PadLowValues() Function PadLowValues() is exposed by the Type A Interface in order to facilitate the padding of blob's which are not of length modular 8.

The blob is padded to the nearest multiple of 8, where the padded length is always greater than or equal to the original length.

A blob is returned which is a concatenation of the input blob and the padded low values[3].

Parameters for Function PadLowValues()

padvalue The padvalue parameter is required and is expected to be a blob. It is the value intended to be padded with low values.

2.1.19 TrimLowValues()

TrimLowValues() Function TrimLowValues() is exposed by the Type A Interface in order to facilitate the trimming of blob's which have trailing low values.

A blob is returned which is stripped of all trailing low values present in the input blob.

[3]Low values in the context of this document are NULL's or 0x00.

Parameters for Function `TrimLowValues()`

`trimvalue` The `trimvalue` parameter is required and is expected to be a blob. It is the value intended to be stripped of all trailing low values.

2.2 Example *Thistle* Script Using `cryptotai`

The module `cryptotai` wraps the `OpenSSL` and `deskeys` library interfaces and exposes them to the *Thistle* environment as a Type A Interface. The following Thistle Interface Definition defines this interface to the *Thistle* environment:

```
interface CRYPTO;

{
  -- File: CRYPTO.tid
  --
  -- This is the Thistle Interface Definition for the Type A Interface to the
  -- Code Magus Limited Encryption Library. This interface makes the
  -- Encryption Methods available to Thistle Scripts.
  --
  -- Copyright (c) 2010 by Code Magus Limited. All rights reserved.
  --
  -- Author: Stephen R. Donaldson [stephen@codemagus.com].
  --

  -- $Author: justin $
  -- $Date: 2011/01/17 10:52:24 $
  -- $Id: CRYPTO.tid,v 1.3 2011/01/17 10:52:24 justin Exp $
  -- $Name:  $
  -- $Revision: 1.3 $
  -- $State: Exp $
  --
}

     type : typea;
     module : "C:\Program Files\CodeMagusLimited\eresiavte\lib\cryptotai.dll";
     init : cryptotai_init;
end.
```

The *Thistle* run-time locates the named interface definition using the standard *Thistle* external pathing convention. For example, the following in the preamble of a script

```
interface crypto : CodeMagus.CRYPTO;
```

introduces `cryptotai` as an internal local name of the interface.

The *Thistle* interface definition file shown above is installed with the Eresia Visual Test Environment to the directory listed below.

```
C:\Program Files\CodeMagusLimited\eresiavte\components\CRYPTO.tid
```

The following script shows how the `cryptotai` interface could be used for encryption and decryption.

```
package CryptoSamplePackage;

{ preamble }

   created by 'Stephen Donaldson';
   description 'Demonstrate usage of Encryption Type A Interface';
   date 2007-05-10T15:28:49;
   target 'Eresia File Portal';
   interface encrypt : CodeMagus.CRYPTO;

begin
  {
   set up path for keys as well as which keyset is required.
  }
  keyfile   := "C:\\build\\TypeA_Encryption\\cmlcrypto\\example.keys";
  keyset    := "testkey";


  {Connect to the interface.}
  crypto := encrypt.Connect();

  {
   Load the Keyfile and Keyset.
   This returns a keyschedule to be used for encryption/decryption.
  }
  keyschedule := crypto.DESKeyLoad(keyfile,keyset);

  {Part 1.  Decrypt a previously encrypted password}

  {Previously encrypted password = "testing06}
  encrypted_password  := "BF52F2C8A9A38B2356F1AF4500EA4B52";

  {
   Call the Des ECB Decrypt Funtion which returns a blob
   decrypted password.
  }
  decrypted_password := crypto.ECBTDESDecrypt(keyschedule, System.BinaryPack(encrypte

  System.DumpScope(decrypted_password);

  {Trim of trailing nulls, if they exist.}
  trimmed_decrypted_password := crypto.TrimLowValues(decrypted_password);

  {Translate the decrypted password into a printable string}
  string_password := System.TranslateFromASCIIToString(trimmed_decrypted_password);

  System.WriteLn("Translate of " # encrypted_password # ": " # string_password);


  {Part 2.  Encrypt and Decrypt a string.}
```

```
  teststring   := "testingString04";

  {Get the string as a ascii blob.}
  asciitstring := System.TranslateFromStringToASCII(teststring);

  {Pad with low values to a mod 8 length}
  padded_asciistring := crypto.PadLowValues(asciitstring);

  {Encrypt the String.}
  encrypted_password := crypto.ECBTDESEncrypt(keyschedule, padded_asciistring);

  {Decrypt the encrypted string}
  decrypted_password := crypto.ECBTDESDecrypt(keyschedule, encrypted_password);

  {Trim of trailing nulls, if they exist.}
  trimmed_decrypted_password := crypto.TrimLowValues(decrypted_password);

  System.WriteLn("Decrytped String is: " # System.TranslateFromASCIIToString(trimmed_


  {Part 3. Test Padding and Trimming}

  tstring    := "1234";
  bitstring  := System.TranslateFromStringToASCII(tstring);


  padstring := crypto.PadLowValues(bitstring);

  System.DumpScope(padstring);

  trimstring := crypto.TrimLowValues(padstring);

  System.DumpScope(trimstring);

 end.
```

Running this script produces the expected (but truncated here) output:

```
-- Code Magus Limited Eresia Visual Test Environment V2.1.
-- Copyright 2002, 2005 and 2008 Code Magus Limited, All Rights Reserved.
-- Built on: 2010-11-26-16.53.44.

20110113-160423 Justin@SETEBOS Job J0001477_D20110113_T160423_CryptoSamplePackage sta
Compiled C:\build\TypeA_Encryption\cryptotai\CryptoSamplePackage.pts successfully.
Thistle configuration locate starts.
Environment variable THISTLE_CONFIG not set.
Thistle configuration using C:\build\TypeA_Encryption\cryptotai\CryptoSamplePackage.t
Thistle configuration starting.
CodeMagusExtras=C:\CodeMagus\CodeMagus\bin\
Thistle configuration completed successfully.
20110113-160423 Justin@SETEBOS Package CryptoSamplePackage has started.
Compiled C:\Program Files\CodeMagusLimited\eresiavte\components\CRYPTO.tid successful
```

```
$Id: output.txt,v 1.1 2011/01/13 16:05:16 justin Exp $
Copyright (c) 2000-2003 by Stephen Donaldson. All rights reserved.
Copyright (c) 2001-2004 by Code Magus Limited. All rights reserved.

Support: mailto:stephen@codemagus.com, http://www.codemagus.com
Translate of BF52F2C8A9A38B2356F1AF4500EA4B52: testing06
Decrytped String is: testingString04
20110113-160423 Justin@SETEBOS Package CryptoSamplePackage has completed with highest
Interface cryptotai cleanup complete.
$Id: output.txt,v 1.1 2011/01/13 16:05:16 justin Exp $
20110113-160423 Justin@SETEBOS Job J0001477_D20110113_T160423_CryptoSamplePackage com
```