THALES

www.thales-esecurity.com

Thales e-Security

paySheld 9000 Application Note:

Secure Host Communications

PWPR0536-002

1 September 2014

Contents

References

No.	Document	Publisher
1	1270A546 payShield 9000 Host Command Reference Manual. (*This document has restricted availability.*)	Thales
2	1270A544 payShield 9000 Console reference Manual. (*This document has restricted availability.*)	Thales
3	1270A596 Local HSM Manager's User Guide. (*This document has restricted availability.*)	Thales
4	1270A529 Remote HSM Manager's User Guide. (*This document has restricted availability.*)	Thales
5	1270A593 payShield 9000 General Information Manual. (*This document has restricted availability.*)	Thales
6	1270A545 payShield 9000 Security Operations Manual. (*This document has restricted availability.*)	Thales
7	Payment Card Industry (PCI) Data Security Standard - Requirements and Security Assessment Procedures Version 2.0	PCI SSC

Introduction

Background

The Thales family of payment HSMs has been protecting the issuing of payment cards and the processing of payments for over 25 years, and is the most widely used product range of its type. The payShield 9000 is the latest model in this product range, and is described later in this document.

During this period, the needs of users have evolved as their environment has changed, and the Thales payment HSMs have been developed to meet these changing needs. The rate of change has accelerated sharply since the latter part of the first decade of the 21st century. This is partly because of the use of the Internet for financial transactions, and the entry into the payments community of new types of organisation with approaches which are different to those of the traditional banking industry.

One of the areas that is changing is the way that HSMs are connected to the host computers that use them. In the past, this connection has been over a network which is dedicated to the host-HSM link, and where the host and HSM are in the same secure data center.

Now, however, HSMs may be located remotely from the host computers - for example where hosts are in "the cloud". Some users are looking to use networks which are shared with other traffic - and which may even be public networks. This introduces new requirements to maintain the security of the HSM environment, i.e.:

- Mutual authentication between the host application and the HSM - to provide confidence to each "end" that the communication partner is what they expect it to be; and
- Privacy of transmitted data - to prevent any interceptor of the traffic from being able to read it. (Most secret information (such as keys, PINs) is in fact already encrypted when outside of the HSM, although PINs may be communicated in the clear for the purpose of PIN mailer printing. Other information, such as account numbers, has traditionally been sent in the clear as it was not viewed as being secret.)
- Integrity of the transmitted data - to ensure that the data is not modified between sender and receiver.

These requirements are driven by best security practise. In addition, the PCI DSS standard requires cardholder data, such as account numbers, to be protected when transmitted over non-private communication interfaces.

payShield 9000 Secure Host Communications

To meet this emerging requirement for secure host communications, payShield 9000 software v2.3 supports the use of TLS to secure traffic between host applications and HSM. SSL is also supported for communication with applications which do not support TLS, but TLS is the preferred protocol.

This capability is available for Ethernet connections: it is not appropriate to Asynchronous or FICON interfaces.

This document describes the implementation of secure host communications on the payShield 9000.

The Thales payShield 9000

The Thales payShield 9000 is a payment hardware security module (HSM) with functionality dedicated to the credit and debit card payments market. It connects to a host application server via an asynchronous, Ethernet or FICON interface and is available in various performance levels. Unlike many other HSMs, it does not require any software libraries to be installed on the host server.

Its main purpose is to keep all cryptographic keys and sensitive data (such as customer PINs) secure. This is achieved by processing all keys and data in a highly secure cryptographic sub-system which supports the recognized security standards for the banking industry.

The payShield 9000 is the latest model of payment HSMs from Thales.

Certifications and standards

The payShield 9000 complies with all the standards required by the payments industry. Formal certifications include:

- PCI HSM
- FIPS 140-2 Level 3 for the secure cryptographic device that the payShield 9000 is built around.

payShield 9000 Host Commands

The way that the host system and HSM exchange data is by exchanging messages: the host sends a command to the HSM, and the HSM sends a response. No payShield 9000-specific API software needs to be installed on the host: the HSM can be used by any host which supports one of the following communication methods:

- Ethernet - TCP/IP
- Ethernet - UDP
- Ethernet - TCP/IP through TLS (or SSL)
- FICON (IBM proprietary fiber-optic networking)
- Asynchronous serial

The commands sent by the host are identified by a 2-character identifier (e.g. "A2", "GK"), and the structure of each command (and its response) is different to reflect the different parameters and output fields required. In this document, the commands are referred to by their 2-character identifier: the detailed command structure is provided in *Reference 1*.

The payShield 9000 Console

The traditional method of managing Thales payment HSMs was by the use of a console - an 80-line x 24 character text display running over a serial asynchronous communication link, or "dumb terminal".

The console is used by entering command codes consisting of one or more alphanumeric characters to initiate a command, and then entering parameters specific to the desired command in response to prompts.

The detailed operation of the commands can be found in *Reference 2*.

Local HSM Manager

Local HSM Manager is designed as a direct replacement for the console (and must be located next to the payShield 9000), offering a more modern graphical user interface than the console.

Local HSM Manager is used by selecting options from menus and then entering data into dialog boxes. This document explains how these options have been modified for AES. Detailed operation of Local HSM Manager is described in *Reference 3*.

Remote HSM Manager

Remote HSM Manager offers a similar user interface to that provided by Local HSM Manager, but allows the payShield 9000 to be managed remotely over a TCP/IP link from anywhere in the world.

Remote HSM Manager is an extension of Local HSM Manager, and its operation is described in *Reference 4*.

Local Master Keys (LMKs)

LMKs are the critical secrets in the payShield 9000, and are automatically deleted if an attempt is made to attack the HSM. The LMKs are used to encrypt all operational keys (including other master keys).

The payShield 9000 supports 2 types of LMK:

- **Variant LMK**: this is the traditional type of LMK and still in most widespread use. Key separation is provided by creating a variant of the LMK to encrypt different types of key or data. Variant LMKs are double-length TDES keys.
- **Keyblock LMK**: this is a more modern and more secure type of LMK. Keys encrypted using the LMK are incorporated into a keyblock which includes metadata defining aspects such as exportability of the key and what it can be used for. The keyblock structure used for this is a proprietary Thales design, based on the TR-31 keyblock described below. Keyblock LMKs are triple-length TDES keys or 256-bit AES keys.

Variant and Keyblock LMKs are described in *Reference 5*.

A single payShield 9000 can have multiple LMKs installed, with each LMK managed by a different security team. This allows multiple applications or clients to be securely segregated on the same hardware unit. The LMKs installed on a payShield 9000 can be a mix of variant and keyblock types, and TDES or AES algorithms.

Use of Keyblocks for exporting keys

An important capability of the payShield 9000 is its support for TR-31 keyblocks when exporting keys.

Traditionally, key management specified in ANSI X9.17 has been used to exchange keys in a multi-vendor environment, where keys are exchanged between products sourced from different manufacturers such that proprietary exchange mechanisms cannot be used. X9.17 key exchange is supported by the payShield 9000.

However, X9.17 key exchange has a number of well-recognised security deficiencies, such as being unable to prevent attacks based on key masquerading and not providing mechanisms for detecting modifications made to the transmitted key.

The deficiencies in X9.17 are addressed by the use of TR-31 keyblocks for key exchange. The TR-31 specification (produced by the ANSI X9 committee) meets the requirements of the X9.24 Part 1 specification. It defines how the key being exchanged is formatted within a block which includes restrictions on how that key can be used and managed, and uses a MAC to prevent modifications to the data from going undetected.

TLS and SSL

History

SSL (Secure Sockets Layer) is a protocol originally developed by Netscape as a means of protecting traffic between internet browsers and servers, and first released in 1995.

In 1999 the IETF (Internet Engineering Task Force) adopted the principles of SSL in their definition of TLS (Transport Layer Security). TLS v1.0 was developed from, and is very similar to, SSL v3.0, but the two protocols are incompatible.

Protocol Versions

SSL v1.0 was not issued as a general release, and SSL v2.0 is considered to have a number of security shortcomings.

The protocols in current use are SSL v3.0, TLS v1.0, TLS v1.1, and TLS v1.2. Although the protocols are not compatible, TLS includes mechanisms to downgrade the connection to SSL or older versions of TLS.

The following table provides a summary of the differences between these protocols:

Comparison	Summary of Differences
TLS v1.0 vs. SSL v3.0	Modifies initial handshake protocol. More secure master key derivation, combining use of MD5 and SHA-1. Application Data record sequence numbering. Enhanced MAC generation, using a combination of MD5 and SHA-1. Encrypted MAC. Handshake "Finished" message includes a hash of all preceding messages.
TLS v1.1 vs. TLS v1.0	Enhanced protection against CBC attacks. Support for registration of parameters with Internet Assigned Numbers Authority (IANA) Protection against the BEAST vulnerability.
TLS v1.2 vs. TLS v1.1	MD5/SHA-1 combinations replaced by SHA-256 or just SHA-1. Enhancements in cipher suite negotiation. Extended support for authenticated encryption ciphers such as GCM and CCM. Support for AES Withdrawal of ability to negotiate sessions with SSL v2.0

What TLS/SSL Provides

TLS/SSL provides a high level of security for sessions between an Ethernet connected client application and server application with no prior knowledge of each other and without any prior exchange of encryption keys. A mutually trusted third party (the CA, or Certificate Authority) is used to certify that the client and server are the owners of their respective private and public key pairs used in establishing the communication session.

This trusted environment provides:

- Authentication - the server may authenticate itself to the client (typically a browser), or the client and server may mutually authenticate themselves to each other.
- Privacy - the communications traffic is encrypted.
- Integrity assurance - using hash and signature algorithms.

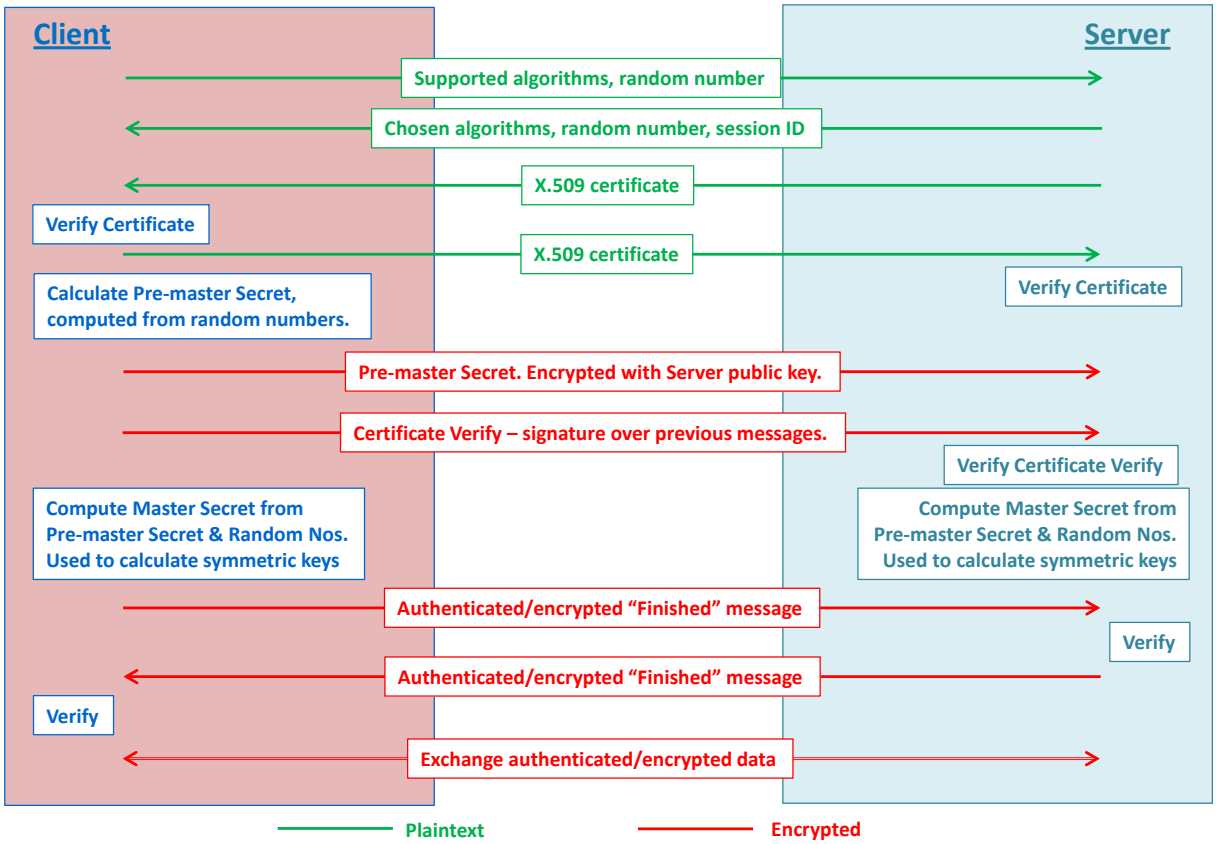
Note that TLS/SSL works between applications. This means that both communicating ***applications*** must be TLS/SSL-enabled, rather than the host and client ***devices***. Proxies can be implemented to allow non-TLS/SSL-enabled applications to be used over a TLS/SSL-protected link: here, the authentication is from/to the proxy rather than the application.

How TLS/SSL Works

The process for setting up a TLS/SSL session can be summarised as follows:

- Both the client and server have their own private (secret) and public keys.
    - The public keys are certified by Certificate Authorities (CAs).
    - The public key certificates can include multiple, chained CA hierarchies - e.g. the public key can be certified by one CA (e.g. operated by the organisation owning the key), and this CA certificate is then certified by a higher-level CA (e.g. a third-party CA trusted by both the key owner and the key user).
    - The client and server can use different CAs.
- The client and server applications negotiate which cipher suite they will use, and exchange some information (but not keys) that will be needed to establish the session. The cipher suite defines the algorithms and key lengths that will be used to establish and protect the session.
- The client and server applications exchange certificates (including their public keys).
- The client and server validate each other's certificate and extract the public key. The validation may be performed by contacting the CA online or by using previously stored CA materials.
- The client application sends an encrypted "Pre-master" secret to the Server application.
- Server and client applications both independently compute a Master secret from the Pre-master secret and use this to calculate the symmetric keys to be used to protect the exchanged data. The keys therefore do not need to be exchanged.
- Following a successful client-server handshake, the application data is exchanged in records (see *SSL/TLS Record Structure*), with the data encrypted using the independently computed keys and MACed using the hashing algorithm in the agreed cipher suite.

The following diagram illustrates this, with some additional detail:


Client
Server
Supported algorithms, random number
Chosen algorithms, random number, session ID
X.509 certificate
Verify Certificate
X.509 certificate
Verify Certificate
Calculate Pre-master Secret, computed from random numbers.
Pre-master Secret. Encrypted with Server public key.
Certificate Verify – signature over previous messages.
Verify Certificate Verify
Compute Master Secret from Pre-master Secret & Random Nos. Used to calculate symmetric keys
Compute Master Secret from Pre-master Secret & Random Nos. Used to calculate symmetric keys
Authenticated/encrypted "Finished" message
Verify
Authenticated/encrypted "Finished" message
Verify
Exchange authenticated/encrypted data
Plaintext
Encrypted

SSL/TLS Record Structure

All records exchanged during an SSL or TLS session have the same high-level structure:

Field	Length (Bytes)	Description
0	1	Content Type: 0x14 = Change Cipher Specification 0x15 = Alert 0x16 = Handshake 0x17 = Application Data
1	2	Protocol Version: "30" = SSL v3.0 "31" = TLS v1.0 "32" = TLS v1.1 "33" = TLS v1.2
3	2	The length of the Protocol Messages. For application data, this includes the MAC and any padding.
4	Variable	The message(s) required for the content type. This may be structured as a multiple fields (e.g. for handshaking) or may be the encrypted application data.
5	Variable (32 for SHA-256; 20 for SHA-1; 16 for MD5)	MAC over the Protocol Messages. May not be present and may be encrypted or unencrypted, depending on the stage of the session.
6	Variable	Padding, where block ciphers are being used. The length of the padding is given in the last byte of the padding.

Overview of TLS/SSL on the payShield 9000

Working with IBM z/OS Mainframes

Users who have payShield 9000 units working with IBM z/OS host systems should make use of the AT-TLS feature in z/OS. Guidance on how to configure z/OS system can be found in the Thales document *1270A640 Configuring z/OS and SRM for HSM SSL/TLS Support*, available from Thales support. This document also covers the use of the Thales SRM for z/OS.

Relevant information is also available in the IBM publication "*System Secure Sockets Layer Programming*" for z/OS at chapter 3 on "*Using Cryptographic Features with System SSL*".

TLS/SSL Support

payShield 9000 supports the following protocol versions:

- TLS v1.2
- TLS v1.1
- TLS v1.0
- SSL v3.0. (This protocol is not recommended, and is disabled by default.)

The payShield 9000 can simultaneously support TLS/SSL and non-secured (TCP or UDP) traffic. It is possible to disable TLS/SSL or non-secured traffic. If TLS/SSL is enabled, it is possible to specify that only TLS is accepted (i.e. SSL is disabled).

Optional License HSM9-LIC036 must be installed on the payShield 9000 to enable use of TLS or SSL.

Supported Cipher Suites

The payShield 9000 supports the following TLS/SSL cipher suites.

Cipher Suite Name	Protocols	Algorithms		
		Asymm	Symm	Hash
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS v1.2	ECDSA	AES 256-bit GCM	SHA-384
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS v1.2	ECDSA	AES 256-bit CBC	SHA-384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS v1.2	ECDSA	AES 128-bit GCM	SHA-256

Cipher Suite Name	Protocols	Algorithms		
		Asymm	Symm	Hash
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS v1.2	ECDSA	AES 128-bit CBC	SHA-256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	TLS v1.2 TLS v1.1 TLS v1.0 SSL V3.0	ECDSA	AES 256-bit CBC	SHA-1
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	TLS v1.2 TLS v1.1 TLS v1.0 SSL v3.0	ECDSA	AES 128-bit CBC	SHA-1
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	TLS v1.2	RSA 2048-bit	AES 256-bit GCM	SHA-384
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	TLS v1.2	RSA 2048-bit	AES 256-bit CBC	SHA-256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	TLS v1.2	RSA 2048-bit	AES 128-bit GCM	SHA-256
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	TLS v1.2	RSA 2048-bit	AES 128-bit CBC	SHA-256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	TLS v1.2 TLS v1.1 TLS v1.0 SSL v3.0	RSA 2048-bit	AES 256-bit CBC	SHA-1
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	TLS v1.2 TLS v1.1 TLS v1.0 SSL v3.0	RSA 2048-bit	AES 128-bit CBC	SHA-1
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA *	TLS v1.2 TLS v1.1 TLS v1.0 SSL v3.0	RSA 2048-bit	Triple DES CBC	SHA-1
SSL_RSA_WITH_3DES_EDE_CBC_SHA *	TLS v1.2 TLS v1.1 TLS v1.0 SSL v3.0	RSA 2048-bit	Triple DES CBC	SHA-1

** These cipher suites are available only if SSL v3.0 has been enabled on the payShield 9000.*

Cipher Suite Negotiation

The Cipher Suites in the table above are listed in decreasing order of preference by the payShield 9000. When negotiating Cipher Suites, the HSM's preferences will take precedence over the client's preferences.

Ephemeral key cipher suites are preferred by the payShield 9000. When selected, every new handshake will require new ephemeral keys be generated; this provides perfect forward secrecy such that if an attacker should ever break the cryptography being used

for a connection then this will be of no use to the attacker in a subsequent connection. All of the cipher suites except the last in the above table use ephemeral keys.

When performing a renegotiation of an existing connection, the payShield 9000 will always force a new session to be negotiated; this protects against a known renegotiation vulnerability.

Data compression

Connections will not use data compression, protecting against the CRIME vulnerability.

The HSM Master Key (HMK)

The HMK is used to encrypt the HSM's private key used by the HSM in establishing the TLS/SSL session.

The HMK-encrypted private key is held outside of the tamper-protected memory such that if the HSM detects a tamper event it is ***not*** lost: the unencrypted private key used during live running is held in tamper-protected memory and ***is*** lost if the HSM detects a tamper event.

The private key can therefore be recovered after a tamper event, once the HMK is installed, by decrypting the encrypted version.

The HMK is generated by the HSM using 2 passphrases entered by security officers. These passphrases must be provided to reconstitute the HMK when recovering the private key after a tamper event. It is held in tamper-protected memory such that it is automatically erased if the HSM detects an attempted tamper.

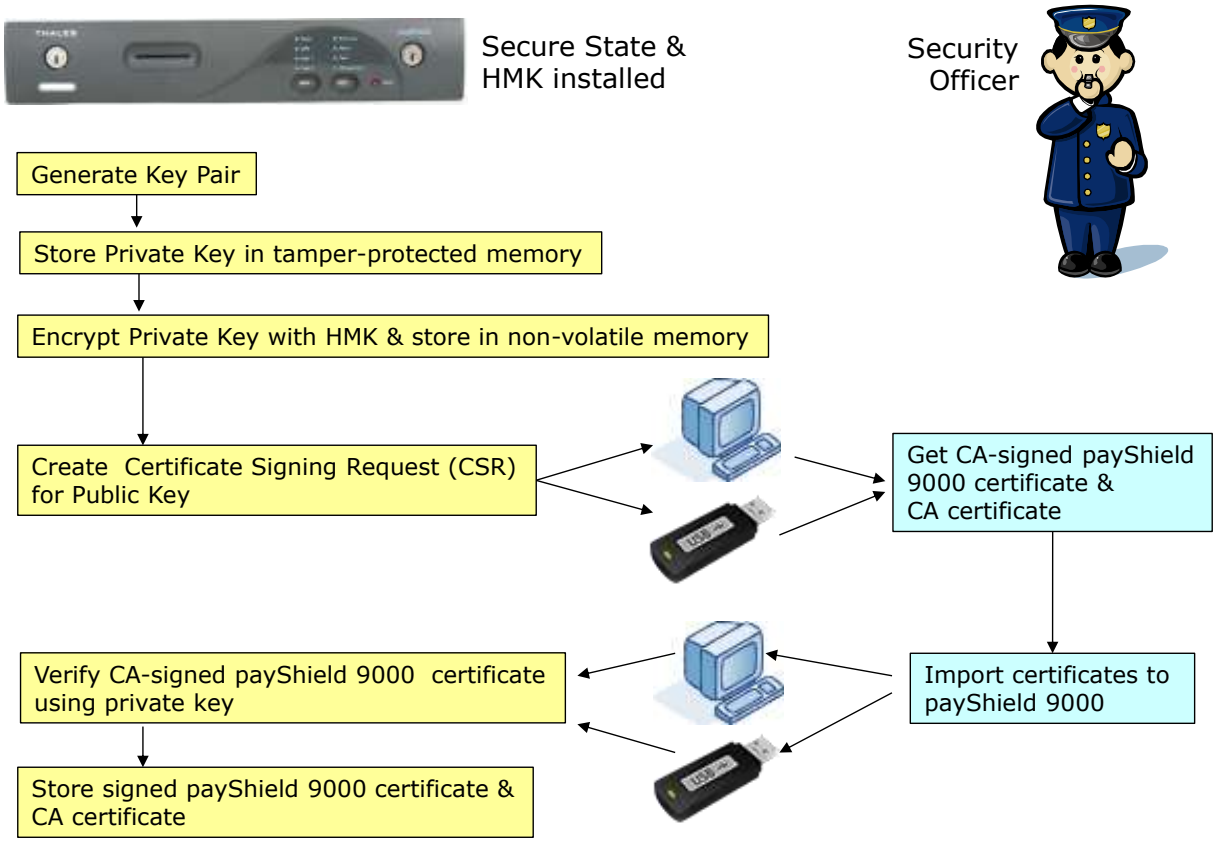
The HMK also performs the role previously played by the RMK (Recovery Master Key) in recovering the private key for the Remote HSM Manager CA.

HSM Certificates

The key pair for the HSM is created by the HSM.

An unencrypted copy of the private key is held in tamper-protected memory, and the HMK-encrypted copy is held in non-volatile memory.

The HSM generates a Certificate Signing Request (CSR) containing its public key in PKCS#10 format, and outputs this to the console or HSM Manager screen and, if attached, to a USB memory device. A Security Officer will generate a certificate signed by their chosen CA and return the public key certificate and CA's certificate chain to the HSM.



If the host computer applications are using a different CA to the HSM, the HSM's CA certificate also needs to be imported by the security officers to the applications.

Intermediate CA certificates can be included to a maximum certificate chain depth of 6, and must include the X509 v3 optional extensions of Subject Key Identifier and Authority Key Identifier. The signed HSM (server) certificate must include the Authority Key Identifier extension. (Examples of certificates are given at *Appendix A*).

Application Certificates

Each application that wishes to establish a secure communications session using TLS or SSL will need to provide to the payShield 9000 a public key in the form of a certificate signed by a CA (or by a hierarchy of CAs). The way that certificate is obtained will depend on the standard procedures of the organisation and its selected CA mechanism.

The application certificates and their associated CA chain certificates are imported by the security officer into the HSM using the console or HSM Manager, or a USB memory device.

The set of client endpoint certificates forms an effective "White List" of applications that are entitled to use the HSM through Secure Host Communications. This is used by the payShield 9000 to mitigate against "man-in-the-middle" attacks.

Intermediate CA certificates can be included to a maximum certificate chain depth of 6, and must include the X509 v3 optional extensions of Subject Key Identifier and Authority Key Identifier. The signed application (client) certificate must include the Authority Key Identifier extension. (Examples of certificates are given at *Appendix A*).

Out-of-Date Certificates

If an attempt to establish a Secure Host Communications session is made using an out-of-date (i.e. expired or not yet valid) certificate, the connection will fail. As a result it is important for users to have suitable processes in place to manage certificate introduction and expiry.

As an option, users can audit attempts to use out-of-date certificates.

Performance considerations

For the majority of payShield 9000 functions, users will not notice a difference in performance when implementing Secure Host Communications.

Where a performance change is likely to be encountered, it is difficult to provide comprehensive guidance on the effect because this will depend on a number of factors, in particular:

- the payShield 9000 performance model
- the TLS/SSL protocol being used
- the cipher suite that has been negotiated
- key lengths being used
- the host commands being used
- the size of the payload where host commands can have different payload sizes
- whether the RSA Booster Licence has been installed

Host commands which do not have large payloads (e.g. PIN Block translation (CA), key import (A6), RSA key generation (EI)) will normally run at the same speed as when using a standard Ethernet interface. However, some combinations of protocol and cipher suite can result in a small performance drop on the highest payShield 9000 performance models.

Host commands processing large data blocks (e.g. encryption/decryption of data blocks (M0/M2), translation of data blocks (M4), MACing (M6, M8)) can experience a more significant performance impact as the payload size increases – but again this is dependent on the protocol/cipher suite combination and is limited to the highest payShield 9000 performance models.

If the RSA Booster Licence (HSM9-LIC033) is installed on a payShield 9000 which is implementing Secure Host Communications, the full benefits of the RSA Booster licence will not be achieved for signature generation with short key lengths or for signature verification.

Users who want to assess the likely impact on performance when implementing Secure Host Communications in their specific environment should contact Thales for guidance.

Security considerations

TLS and SSL can only provide a secure environment when implemented correctly. When implementing TLS/SSL on the payShield 9000, the guidance in *Reference 6* should be followed.

Support for USB Memory Sticks

USB memory sticks are used to transfer material such as certificates in and out of the payShield 9000. The Operating System used in the payShield 9000 supports most types of USB memory stick, but may not have the drivers for some of the newer types. If difficulties are experienced when trying to read from or write to a USB device, an alternative memory stick should be used.

Configuring TLS/SSL on the payShield 9000

This chapter discusses the use of the following console commands and HSM Manager actions to configure the payShield 9000 to use TLS or SSL. The payShield 9000 must have license LIC036 installed.

Operation	Console Command	HSM Manager Action
Check Availability of TLS/SSL	*VR*	*View / HSM Information*
Configure Ethernet host ports	*CH*	*Edit / Host Interface*
Generate HMK	*SK*	*Tools / Secure Host Communications / Generate HMK*
Change HMK Passphrase	*SP*	*Tools / Secure Host Communications / Change HMK Passphrase*
Restore HMK	*SL*	*Tools / Secure Host Communications / Recover HMK*
Generate/Export SSL Server Certificate Signing Request	*SG*	*Tools / Secure Host Communications / Generate Certificate Server Request*
Export Server (HSM) CA	*SE*	*Tools / Secure Host Communications / Export HSM CA Certificate*
Import Signed Certificate	*SI*	*Tools / Secure Host Communications / Import Signed Certificate*
Viewing Certificates held on the HSM	*SV*	*Tools / Secure Host Communications / View Certificates*
Delete Certificate	*SD*	*Tools / Secure Host Communications / Delete Certificates*

In the following example output, text in **bold+underscore** represents inputs made by the user.

Checking availability of TLS/SSL

The *VR* console command will report on the availability of TLS/SSL on the payShield 9000. The start of the output will consist of the following information:

Row	Dialogue
1	**VR**
2	Base release: 2.3b
3	Revision: 1346-0913
4	Build Number: 0002
5	HSM Core API Version: 7.0.18
6	Serial Number: A4665302078G
7	Unit info: Licenced
8	Host Configuration: Async,Ethernet,(optional) TLS/SSL
9	Licence Issue No: 5
10	Performance: 220 TPS
11	Base Software: Version 2
12	Ship Counter: 1
13	Crypto: 3DES,AES,RSA
14	LMKs Enabled: 10 LMKs
15	Press "Enter" to view additional information...

The following elements are relevant to configuring TLS/SSL:

- Row 8: the availability of TLS/SSL is indicated in the "Host Configuration" information.

Equivalent changes have been made in HSM Manager to the *View / HSM Information* function.

Configuring the Ethernet Ports

The *CH* (Configure Host) console command has been modified to allow the Ethernet ports to be configured for TLS/SSL. The following example shows a payShield 9000 being configured for a single physical Ethernet port:

Row	Dialogue
1	**CH**
2	Please make a selection. The current setting is in parentheses.
3	Message header length [1-255] (4):

Row	Dialogue
4	Host interface [[A]sync, [E]thernet] (E):
5	Enter Well-Known-Port (1500):
6	Enter Well-Known-TLS-Port (2500):
7	UDP [Y/N] (Y):
8	TCP [Y/N] (Y):
9	TLS/SSL [Y/N] (Y):
10	TLS Enforced [Y/N] (N):
11	Number of connections [1-64] (5):
12	Enter TCP keep alive timeout [1-120 minutes] (120):
13	Number of interfaces [1/2] (1):
14	Interface Number [1/2] (1):
15	Interface Number 1:
16	Enter IP Address (193.240.100.216): **193.240.100.215**
17	Enter subnet mask (255.255.255.0):
18	Enter Default Gateway Address (193.240.100.1):
20	Save HOST settings to smart card? [Y/N]: **n**

The following elements are relevant to configuring TLS/SSL:

- Row 6: a well-known port must be specified for TLS/SSL - the default is 2500. This is analogous to the well-known port for unsecured host traffic (Row 5), and can be used in the same way to identify the LMK required for the host command, i.e.:
  - 2500 = default LMK
  - 2501 = LMK 0
  - 2502 = LMK 1
  - etc.
- Row 7 and 8: enable or disable non-secured host traffic (TCP or UDP).
- Row 9: enable or disable secured host traffic.
- Row 10: forces TLS (rather than SSL) to be used to protect secure host traffic.

- Row 11: specifies the number of connections/threads for each port. If a value of 5 was entered and both Ethernet ports were enabled, a total of 10 connections/threads would be available. These connections/threads are shared between secured and non-secured traffic.

The *QH* (Query Host) console command used to view the port settings has been modified to display the new parameters.

Equivalent changes have been made in HSM Manager to the *Edit / Host Interface* function.

Managing the HMK

A number of new console commands and equivalent HSM Manager functions have been introduced to create and manage the new HMK.

Generate HMK

The HMK is generated using the *SK* console command while the HSM is in Secure state:

Row	Dialogue
1	**SK**
2	**** NOTE ****
3	Passphrase rules as follows:
4	1 - Must be between 8 and 30 characters long.
5	2 - Can contain spaces
6	3 - Must be comprised of (at a minumum):
7	2 digits
8	2 uppercase characters
9	2 lowercase characters
10	2 symbols (ex. !/?.#:')
11	Enter administrator 1 passphrase: **********
12	Re-enter administrator 1 passphrase: **********
13	Enter administrator 2 passphrase: **********
14	Re-enter administrator 2 passphrase: **********
15	Creating HMK. Please, wait ... DONE
16	HMK generated successfully

Row	Dialogue
17	Key synchronisation complete

Notes:

- Rows 11-14: two different passphrases are required, each entered by a different security officer. These passphrases must be stored securely (in the same way as key components) to allow subsequent HMK recovery in the event that the HSM enters a tampered state..
- Rows 3-10: the passphrases must be of an acceptable complexity. Spaces are allowed.

When using HSM Manager, the *Tools / Secure Host Communications / Generate HMK* function should be used.

Change HMK Passphrase

The HMK passphrase should be changed regularly as best security practise, and will need to be changed if a security officer is replaced by another person. This is accomplished using the *SP* console command while the HSM is in Secure state:

Row	Dialogue
1	**SP**
2	**** NOTE ****
3	Passphrase rules as follows:
4	1 - Must be between 8 and 30 characters long.
5	2 - Can contain spaces
6	3 - Must be comprised of (at a minimum):
7	2 digits
8	2 uppercase characters
9	2 lowercase characters
10	2 symbols (ex. !/?.#:')
11	4 - Cannot use the same passphrase that was used within the past 10 previous attempts
12	Select administrator password to change [1,2]: 1
13	Enter administrator 1 current passphrase: **********

Row	Dialogue
14	Enter administrator 1 new passphrase: **********
15	Re-enter administrator 1 new passphrase: **********
16	Changing passphrases. Please, wait ... DONE
17	HMK generated successfully

Notes:

- Rows 11-14: two different passphrases are required, each entered by a different security officer. These passphrases must be stored securely (in the same way as key components) to allow subsequent HMK recovery in the event that the HSM enters a tampered state.
- Rows 3-10: the passphrases must be of an acceptable complexity. Spaces are allowed.
- A Passphrase cannot be re-used until at least 10 generations of passphrase changes have been made.

When using HSM Manager, the *Tools / Secure Host Communications / Change HMK Passphrase* function should be used.

Restore HMK

If the HSM detects a tamper event, its private key used to establish TLS/SSL sessions is deleted. An HMK-encrypted copy of the private key is held in non-volatile memory, and the key itself can be recovered and restored to tamper-protected memory by entering the passphrases used at HMK generation into the *SL* console command while the HSM is in Secure state:

Row	Dialogue
1	SL
2	Enter administrator 1 passphrase: **********
3	Enter administrator 2 passphrase: **********
4	Recovering HMK. Please, wait ... DONE
5	HMK recovered successfully
6	Key synchronization complete

When using HSM Manager, the *Tools / Secure Host Communications / Recover HMK* function should be used.

Managing Certificates

New console commands and HSM Manager functions are available to manage the HSM, application, and CA certificates required to establish TLS/SSL sessions.

Generate and Export SSL Server Certificate Signing Request

This process generates the HSM key pair, stores the private key in tamper-protected memory and (in HMK-encrypted form) in non-volatile memory, and creates a Certificate Signing Request for the public key.

On the console, this is done by using the *SG* command while the HSM is in Secure state:

Row	Dialogue
1	**SG**
2	Please enter the Subject Information for the Certificate Request:
3	Country Name (2 letter code) [US]: **UK**
4	State or Province Name (full name) []: **Buckinghamshire**
5	Locality Name (eg, city) []: **Long Crendon**
6	Organization Name (eg, company) []: **Mega Banking Corporation.**
7	Organizational Unit Name (eg, section) []: **CorpSecurity**
8	Common Name (e.g. server FQDN or YOUR name) []: **HSM1**
9	Email Address []: **michael.mouse@megabank.com**
10	Select key type:
11	1 - RSA
12	2 - ECDSA P-256
13	3 - ECDSA P-384
14	4 - ECDSA P-521
15	Type [4]: **1**
16	Generating key pair
17	.....+++
18	...+++
20	DONE
21	Do you wish to save to a file [Y/N]:**y**

Row	Dialogue
22	Enter filename: **HSM1**
23	File exists - replace? [Y/N]:**Y**
23	-----BEGIN CERTIFICATE REQUEST-----
25	MIIC9zCCAd8CAQAwgbExCzAJBgNVBAYTAlVLMRgwFgYDVQQIEw9CdWNraW5naGFt . . . . ehNJSXN703d6vP2aktV3VRHkMvbrkjqT37dU3chCaqkOYOqEMhnCEMdVGw==
26	-----END CERTIFICATE REQUEST-----

Notes:

- Rows 10-15: the key type is selected. If RSA is used, the key will be of 2048-bit length.
- Rows 11: The RSA is 2048-bit key length.
- Rows 11-14: ***the client certificate must use the same key type as is used in the HSM's Certificate Signing Request***.
- Rows 21-23: this allows the certificate signing request to be filed (e.g. on a USB memory device).
- Rows 23-25: the certificate signing request is also displayed on the screen.

When using HSM Manager, the *Tools / Secure Host Communications / Generate Certificate Signing Request* function should be used.

Export Server (HSM) CA

The CA certificate used by the HSM must be made available to the host applications. It can be exported using the *SE* console command while the HSM is in Secure state:

Row	Dialogue
1	**SE**
2	Do you wish to save to a file [Y/N]:**Y**
3	Enter filename: **CACertpS9000.crt**
4	payShield Certificate
5	MIID+TCCAuGgAwIBAgIJAJyPxxP6oxAQMA0GCSqGSIb3DQEBBQUAMIGyM . . . X4FkYiQv2CJb7J/vAw==

Row	Dialogue
6	-----END CERTIFICATE-----

Notes:

- Rows 2-3: the certificate can be saved to a file - e.g. on a USB memory device - to allow it to be exported.

When using HSM Manager, the *Tools / Secure Host Communications / Export HSM CA Certificate* function should be used.

Import Signed Certificate

Following the certificate signing request, the signed certificate for the HSM's public key will need to be imported. It will also be necessary to import:

- all signed certificates for applications
- Self-signed certificate for the root CA
- Where a chained CA hierarchy is being used, certificates for each intermediate CA signed by the next CA up in the hierarchy.

On the console, this is done using the *SI* command while the HSM is in Secure state:

Row	Dialogue
1	**SI**
2	Select File
3	1 - RsaServerRootCA.crt
4	2 - TCPUDPSIM.crt
5	3 - RsaClientRootCA.crt
6	4 - hsm1.crt
7	File: **1**
8	Imported Trusted CA Certificate
9	Issued to: payShield Certificate, Issued by: payShield Certificate
10	Validity : May 9 10:59:22 2013 GMT to May 7 10:59:22 2023 GMT
11	Unique ID: 9C8FC713FAA31010 - AC03FAD5 (Root)
12	Do you wish to import another certificate? **n**

Notes:

- Rows 3-6 in this example identify files available on a USB memory device attached to the HSM0 The user identifies which of these is to be imported at Row 7.

When using HSM Manager, the *Tools / Secure Host Communications / Import Signed Certificate* function should be used.

Viewing certificates held on the HSM

The payShield 9000 will establish TLS/SSL sessions only with applications for which it has stored a copy of their certificate. All stored certificates including the payShield's own certificate and CA certificates) can be viewed using the *SV* console command:

Row	Dialogue
1	sv
2	HSM Private Key installed: Yes
3	HSM Certificate installed:
4	1 - Issued to: HSM-0002, Issued by: Bank XYZ
5	Validity : May 21 15:05:51 2013 GMT to May 21 15:05:51 2014 GMT
6	Unique ID: 2050 - AC03FAD5
7	Client certificate(s) installed:
8	2 - Issued to: Client Certificate, Issued by: Client Certificate
9	Validity : May 7 09:37:18 2013 GMT to May 7 09:37:18 2014 GMT
10	Unique ID: 2016 - D221289A
11	CA Certificate(s) installed:
12	3 - Issued to: Client Certificate, Issued by: Client Certificate
13	Validity : May 7 09:24:10 2013 GMT to May 5 09:24:10 2023 GMT
14	Unique ID: C14FF9DE78FB441A - D221289A (Root)
15	4 - Issued to: payShield Certificate, Issued by: payShield Certificate
16	Validity : May 9 10:59:22 2013 GMT to May 7 10:59:22 2023 GMT
17	Unique ID: 9C8FC713FAA31010 - AC03FAD5 (Root)

Row	Dialogue
18	Chain of Trust validated:
19	payShield Certificate (Root)
20	Select an item to view: **1**
21	Certificate:
22	Data:
23	Version: 3 (0x2)
24	Serial Number: 8273 (0x2051)
25	Signature Algorithm: sha1WithRSAEncryption
26	Issuer: C=UK, ST=Greater London, L=London, O=Bank XYZ, OU=RootCA, CN=Bank XYZ/emailAddress=root@bankxyz.com
27	Validity
28	Not Before: May 21 15:05:51 2013 GMT
29	Not After : May 21 15:05:51 2014 GMT
30	Subject: C=UK, ST=Greater London, O=Bank XYZ, OU=Operations, CN=HSM-0002/emailAddress=bill@bankxyz.com
31	Subject Public Key Info:
32	Public Key Algorithm: rsaEncryption
33	Public-Key: (2048 bit)
34	Modulus:
35	00:aa:31:e6:90:46:fe:e9:26:8b:93:39:5a:8c:be:
36	…
37	3d:39:2b:d7:06:47:04:6a:54:d2:12:4e:ac:9a:a3:
38	5b:49
39	Exponent: 65537 (0x10001)
40	X509v3 extensions:
41	X509v3 Basic Constraints:
42	CA:FALSE
43	X509v3 Key Usage:
44	Digital Signature, Non Repudiation, Key Encipherment

Row	Dialogue
45	Signature Algorithm: sha1WithRSAEncryption
46	b8:e9:e9:8f:2e:f9:50:93:a1:8b:8d:0b:e5:fd:ef:6f:6c:05:
47	…
48	59:0d:df:85:b7:48:c6:02:d9:16:f9:80:e5:c9:c2:69:7f:06:
49	2b:ba:18:9f
50	Do you wish to view another certificate? **N**

Notes:

- Row 22: the number for the required certificate as offered in rows 4, 8, 12, and 15 should be entered.

When using HSM Manager, the *Tools / Secure Host Communications / View Certificates* function should be used.

Delete Certificate

Where it is no longer required to establish TLS/SSL connections with an application or where a certificate has been withdrawn or updated, it will be necessary to delete the certificate stored on the payShield 9000. This can be achieved using the *SD* console command while the HSM is in Secure state:

Row	Dialogue
1	**SD**
2	HSM Private Key installed: Yes
3	HSM Certificate installed:
4	1 - Issued to: HSM1, Issued by: payShield Certificate
5	Validity : May 20 08:51:27 2013 GMT to May 20 08:51:27 2014 GMT
6	Unique ID: 204D - AC03FAD5
7	Client certificate(s) installed:
8	2 - Issued to: Client Certificate, Issued by: Client Certificate
9	Validity : May 7 09:37:18 2013 GMT to May 7 09:37:18 2014 GMT
10	Unique ID: 2016 - D221289A
11	CA Certificate(s) installed:

Row	Dialogue
12	3 - Issued to: Client Certificate, Issued by: Client Certificate
13	Validity : May 7 09:24:10 2013 GMT to May 5 09:24:10 2023 GMT
14	Unique ID: C14FF9DE78FB441A - D221289A (Root)
15	4 - Issued to: payShield Certificate, Issued by: payShield Certificate
16	Validity : May 9 10:59:22 2013 GMT to May 7 10:59:22 2023 GMT
17	Unique ID: 9C8FC713FAA31010 - AC03FAD5 (Root)
18	Chain of Trust validated:
20	payShield Certificate (Root)
21	5 - HSM Private Key
22	Select an item to delete (6 for ALL): **6**
23	Do you wish to delete another certificate? **n**

When using HSM Manager, the *Tools / Secure Host Communications / Delete Certificates* function should be used.

Auditing attempted use of out-of-date certificates.

Users can choose to record in the payShield 9000's Audit Log any attempts made to establish a Secure Host Communications session using an out-of-date certificate. This option is enabled/disabled using the AUDITOPTIONS console command:

Row	Dialogue
1	**AUDITOPTIONS**
2	Audit User Actions: NO
3	Audit Error Responses to Host Commands: NO
4	Audit utilization data resets: NO
5	Audit diagnostic self tests: NO
6	Audit ACL connection failures: YES
7	Audit out-of-date Certificates for Secure Host Sessions: NO
8	Audit Counter Value: 0000005A

Row	Dialogue
9	List of Audited Console Commands:
10	List of Audited Host Commands:
11	Audit User Actions? [Y/N]: **n**
12	Audit Error Responses to Host Commands? [Y/N]: **n**
13	Audit Utilization Data Resets? [Y/N]: **n**
14	Audit Automatic Self Testing? [Y/N]: **n**
15	Audit ACL connection failures? [Y/N]: **n**
16	Audit out-of-date Certificates for Secure Host sessions? [Y/N]: **y**
17	Current Audit Counter value is: 0000005A
18	Enter new value or <Return> for no change:
20	Modify Audited Command List? [Y/N]: **n**
21	Audit User Actions: NO
22	Audit Error Responses to Host Commands: NO
23	Audit utilization data resets: NO
24	Audit diagnostic self tests: NO
25	Audit ACL connection failures: NO
26	Audit out-of-date Certificates for Secure Host Sessions: YES
27	Audit Counter Value: 0000005A
28	List of Audited Console Commands:
29	List of Audited Host Commands:
30	Save Audit Settings to Smartcard? [Y/N]: n

When using HSM Manager, the *Misc* tab of the *Edit / Auditing* function should be used.

The resulting Audit Log entries will be of the following formats:

```
0000002F  10:25:22  02/Jul/2016  Certificate has expired.
Unique ID: A8F66A587213303F - 2BA3B089

00000027  22:24:33  02/Jul/2014  Certificate not yet valid.
Unique ID: A8F66A587213303F - 2BA3B089
```

Abbreviations

Abbreviation	Meaning
3DES	Triple DES
AES	Advanced Encryption Standard (algorithm)
ANSI	American National Standards Institute
CA	Certificate Authority
CBC	Cipher Block Chaining
CBC3	Cipher Block Chaining using TDES
CCM	Counter with CBC-MAC
CSR	Certificate Signing Request
DES	Data Encryption Standard (algorithm)
DHE	Diffie-Hellman Ephemeral mode for key exchange (TLS terminology)
DSS	(PCI) Data Security Standard
ECDHE	Diffie-Hellman Ephemeral mode for key exchange using elliptic curve cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EDH	Diffie-Hellman Key Exchange (SSL terminology)
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
HMK	HSM Master Key
HSM	Hardware security Module
IETF	Internet Engineering Task Force
IP	Internet Protocol
LMK	Local Master Key
PCI	Shorthand for PCI SSC
PCI SSC	Payment Card Industry Security Standards Council
PIN	Personal Identification Number
RMK	Recovery Master Key
RSA	Rivest Shamir Adleman (algorithm)
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol

Abbreviation	Meaning
TDES	Triple DES
TLS	Transport Layer Security
UDP	User Datagram Protocol

App A - Certificate Examples

Intermediate CA Certificate

```
Intermediate CA Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 762114 (0xba102)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=Florida, L=Plantation,
               CN=RsaClientRootCA.thalesesec.com
        Validity
            Not Before: Jul 10 18:55:49 2013 GMT
            Not After : Jul 10 18:55:49 2014 GMT
        Subject: C=US, ST=Florida, CN=RsaClientIntCA1.thalesesec.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:b0:2d:f9:ce:ba:b1:40:f5:c2:43:1d:3f:bd:a1:
                    2e:e6:5b:73:f3:0c:ed:ce:de:71:80:2a:dc:ca:3a:
                    0e:b7:8a:82:56:86:80:1c:65:b2:47:6f:0d:77:d7:
                    78:db:d3:51:e4:32:50:5e:cf:ae:05:7b:a4:5f:c2:
                    d2:90:d5:70:63:b4:6a:56:a4:c5:4c:2e:1d:47:f0:
                    59:b6:10:f4:c9:46:1e:9c:db:43:43:80:76:aa:40:
                    78:fe:23:73:ab:c1:1e:15:8b:7a:e1:66:b5:57:3b:
                    bf:d0:3a:e6:7d:ed:32:c2:21:fc:57:7b:b1:62:51:
                    bc:d7:38:8f:4e:df:76:cc:5a:c3:5a:ca:75:2c:86:
                    e6:fc:82:b6:5e:fd:c8:14:ca:f2:c6:9b:c8:33:58:
                    9b:fd:90:ea:ec:b6:77:0e:fe:12:35:be:89:b3:68:
                    6e:69:46:5c:03:8c:41:5f:c3:d3:99:58:d5:35:7a:
                    88:41:ce:50:7e:5a:a2:ff:28:36:73:86:61:94:23:
                    24:69:86:5c:73:31:60:ee:b8:ad:d4:fe:3c:b8:65:
                    50:35:49:6d:08:9a:2b:d4:26:b6:97:1c:ba:d1:c2:
                    c3:fe:4b:bf:4b:27:be:d6:57:d7:97:37:10:23:f1:
                    4d:33:5b:41:d7:8e:55:bf:9a:76:05:50:5d:8f:f0:
                    ef:43
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Key Identifier:
                DE:C0:02:1B:48:37:6B:C3:34:F5:9E:D0:8A:32:12:5E:ED:1C:50:2C
            X509v3 Authority Key Identifier:
                 keyid:08:20:EB:E6:51:CF:7F:08:D3:9D:33:A4:DC:48:AE:2E:5D:6C:
                       F3:EC

    Signature Algorithm: sha1WithRSAEncryption
         4b:07:e2:e2:90:60:0a:dc:29:56:bb:65:8b:9a:62:3c:a0:70:
         22:0c:8c:fe:2f:7b:9f:46:9a:ac:fb:6b:f7:e4:4a:d5:54:b4:
         c9:46:97:e3:82:d7:66:ed:5d:e6:24:e8:8c:b8:8b:86:0c:82:
         bf:00:e3:6c:73:bc:27:0b:aa:02:07:f9:10:1d:9a:fc:2e:7e:
         34:6d:74:6a:90:73:14:8a:ba:81:77:74:66:01:e5:da:4d:54:
         ed:18:c5:f7:7d:e4:62:24:ec:f6:80:86:b3:56:43:ec:d5:48:
         90:fd:a4:28:e5:89:7f:60:a9:a9:a7:67:3c:cd:f1:22:7b:0e:
         dd:16:a8:09:a8:6e:0e:97:a8:26:cc:94:fb:95:a2:1f:8e:83:
         ee:6c:6d:f7:f4:ec:fe:2b:bd:bf:ce:a8:5f:f2:6f:92:89:80:
         f0:41:83:89:56:3e:9a:47:e0:24:28:6a:fd:69:05:a7:6a:fd:
         66:4e:2d:35:69:54:4c:00:8b:52:3c:26:f1:8a:35:82:e3:d4:
         b8:f8:09:e6:0e:6f:3b:3a:c6:9c:f5:23:c6:5e:9c:00:fa:90:
         21:aa:4c:fa:fd:84:bf:87:55:b9:a1:0e:d8:82:92:07:79:08:
         9b:49:fd:89:3d:c1:f4:61:ba:c0:9a:ac:e3:d5:75:ec:3a:51:
         c5:70:59:23
```

Client Certificate

```
Client Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 762369 (0xba201)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=Florida, L=Plantation,
               CN=RsaClientRootCA.thalesesec.com
        Validity
            Not Before: Jul 16 21:34:50 2013 GMT
            Not After : Jul 16 21:34:50 2014 GMT
        Subject: C=US, ST=Florida, CN=RSAClient.thalesesec.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:f6:26:4d:61:3d:62:22:62:44:59:57:f0:60:a4:
                    a0:63:c0:2a:24:a3:45:d5:2c:c2:1b:2c:70:f7:2d:
                    29:da:48:af:cd:17:b2:5b:05:08:dd:b6:27:5e:d6:
                    f7:e6:a7:56:df:d5:e6:07:e6:dd:4f:2a:68:58:60:
                    4d:e2:08:8d:ee:04:e0:d2:29:35:36:0c:8b:38:88:
                    f3:ea:9a:2e:35:f6:3d:b9:73:99:53:f1:0f:76:10:
                    74:10:19:9d:02:71:49:0b:0c:29:e4:af:91:f5:ac:
                    73:0c:d2:e0:7c:d8:b1:d3:0b:72:94:6b:6b:9b:f1:
                    c1:6e:22:5e:ee:77:d0:40:3c:cd:2a:cc:82:83:a6:
                    af:c3:b6:d9:b5:9b:85:c3:b3:00:64:f1:50:5a:f1:
                    88:c0:3b:f2:c7:d0:c2:d2:76:bf:9f:5c:7a:f4:a4:
                    7c:8e:c7:ab:a0:2b:dc:69:c3:95:51:f4:73:ad:ac:
                    a3:32:85:a1:15:79:d6:d0:e1:be:a7:00:33:60:1c:
                    21:90:7c:2b:9e:e1:09:13:fa:de:fd:31:90:db:6d:
                    89:f8:f4:e7:a4:b0:0b:c4:d5:e4:f3:67:20:e1:17:
                    4a:65:3a:f0:08:57:4b:85:a2:3c:1f:17:cd:a3:3a:
                    01:3c:e0:d2:39:27:de:38:53:3e:e7:43:09:58:21:
                    95:f7
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Key Agreement
            X509v3 Authority Key Identifier:
                       keyid:08:20:EB:E6:51:CF:7F:08:D3:9D:33:A4:DC:48:AE:2E:5D:
                       6C:F3:EC

    Signature Algorithm: sha1WithRSAEncryption
         78:e5:d2:5f:cd:03:8e:65:e3:40:7d:9c:25:15:41:ed:da:67:
         7d:d4:86:cb:7c:84:75:e4:3a:36:a0:ec:a1:28:ba:7d:37:ba:
         b3:2f:48:f4:0e:24:56:e1:df:0e:f9:cf:8c:7f:b7:bc:57:99:
         6c:de:c9:30:3b:47:12:d0:be:c9:f9:d4:c3:87:c0:e6:36:b4:
         7a:e1:fa:8e:51:32:fb:6a:fd:08:87:93:ab:3b:67:42:a7:1d:
         a8:46:07:04:4a:5a:ad:a6:b6:60:89:7a:50:e1:8d:48:97:af:
         8b:e1:98:8b:f2:3a:26:a6:cf:c6:a7:18:36:ff:9c:05:95:3a:
         b2:a3:88:61:02:d8:31:df:bc:97:77:9c:e7:ce:33:65:20:f0:
         27:0c:2e:db:b5:d2:ab:82:3d:c3:d4:c5:2b:29:82:b4:21:d1:
         48:ed:eb:ff:56:14:34:c9:62:99:cd:7b:73:c9:93:01:3d:d2:
         a8:37:5f:0d:4a:2f:56:1d:d0:57:95:f8:7c:aa:f7:5e:bb:09:
         1e:7c:74:81:be:b4:1e:03:a3:e5:1a:bc:ba:7a:04:02:57:b5:
         00:1f:f8:32:29:74:1b:5d:f1:96:b8:f9:3e:f3:02:bb:dc:de:
         4e:35:43:cd:4e:80:a3:60:69:a2:47:97:7a:2e:e4:0f:f3:d3:
         b1:22:76:40
```

OpenSSL Configuration File

Where OpenSSL is being used to provide TLS/SSL support on the host system, the configuration file must contain the following:

```
[ v3_client ]

basicConstraints       = CA:FALSE
#extendedKeyUsage      = clientAuth
keyUsage               = keyAgreement, digitalSignature
authorityKeyIdentifier = keyid,issuer

[ v3_server ]

basicConstraints       = CA:FALSE
extendedKeyUsage       = serverAuth
keyUsage               = keyAgreement, keyEncipherment,
                                    digitalSignature, nonRepudiation
authorityKeyIdentifier = keyid,issuer

[ v3_ca ]

basicConstraints       = CA:true
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid,issuer
```

Notices

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (*http://www.openssl.org/*)

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)

THALES

www.thales-esecurity.com

About Thales e-Security

Thales e-Security is a leading global provider of data encryption and cyber security solutions to the financial services, high technology manufacturing, government and technology sectors. With a 40-year track record of protecting corporate and government information, Thales solutions are used by four of the five largest energy and aerospace companies, 22 NATO countries, and they secure more than 80 percent of worldwide payment transactions. Thales e-Security has offices in Australia, France, Hong Kong, Norway, United Kingdom and United States. For more information, visit www.thales-esecurity.com

Follow us on:

Americas – Thales e-Security Inc. 900 South Pine Island Road, Suite 710, Plantation, FL 33324 USA • Tel: +1 888 744 4976 or +1 954 888 6200 • Fax: +1 954 888 6211 • E-mail: sales@thalesesec.com

Asia Pacific – Unit 4101 41/F 248, Queen's Road East, Wanchai, Hong Kong, PRC • Tel: +852 2815 8633 • Fax: +852 2815 8141 • E-mail: asia.sales@thales-esecurity.com

Europe, Middle East, Africa – Meadow View House, Long Crendon, Aylesbury, Buckinghamshire HP18 9EQ • Tel: +44 (0)1844 201800 • Fax: +44 (0)1844 208550 • E-mail: emea.sales@thales-esecurity.com