cpl.thalesgroup.com

payShield® 10K

Host Programmer's Manual

007-001518-022

Date: February 2025

Rev: A2

Doc. Number: 007-001518-022

The information contained in this document is provided "AS IS" without any warranty of any kind. Unless otherwise expressly agreed in writing, Thales makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Thales reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Thales hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Thales be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Thales does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Thales be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Thales products. Thales disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

Contents

Revision Status **12**

References **17**

1 Introduction **18**

1.1 General Information 18
1.2 Printer Requirements 19
1.3 Command Message Format 19
1.4 Response Message Format 20
1.5 Data Representation 21
1.5.1 ASCII Character Codes 22
1.5.2 EBCDIC Character Codes 22
1.6 Input/Output Flow Control 24
1.7 Error Handling 24
1.8 Error Logs 25
1.9 Multiple HSMs 26
1.10 User Storage 26

2 Host Connections **27**

2.1 TCP/IP Protocol 27
2.1.1 Port Addresses 27
2.1.2 Sending Commands and Returning Responses 27
2.1.3 Example of Sending Commands and Returning Responses 28
2.2 UDP Protocol 30
2.2.1 Sending Commands 30
2.2.2 Returning Responses 30
2.3 payShield 10K PS10-F (FICON) 31
2.3.1 Connections 32
2.3.2 Configuration Settings 32

3 PIN Printing and Solicitation **33**

3.1 PIN Mailers 33
3.1.1 Setting up the PIN Mailer form 34
3.1.2 Generate an Encrypted PIN 35
3.1.3 Sending PIN Mailer Data for Printing at the HSM 36
3.1.4 PINs Created Outside of the payShield 10K 36
3.1.5 Verify PIN Data Cryptography 37
3.2 PIN Solicitation 37

3.2.1 Validation Algorithm ... 38
3.2.2 Stages in PIN Solicitation ... 40
3.3 Printer Support ... 43
3.3.1 Impact Printers ... 43
3.3.2 Directly-attached Laser or Ink Jet ... 44
3.3.3 High-volume "Print Factories" ... 45

4 Transaction Key Schemes ... 48
4.1 Racal Transaction Key Scheme (RTKS) ... 48
4.1.1 RTKS Functions ... 49
4.2 Simultaneous use of both Racal and Australian Transaction Key Schemes ... 50
4.3 Derived Unique Key Per Transaction (DUKPT) ... 52
4.3.1 General Description ... 52
4.3.2 The Role of the payShield 10K in DUKPT ... 59
4.4 German Banking Industry Committee (GBIC) Key Derivation ... 68
4.5 Italian Key Derivation ... 71
4.5.1 Introduction ... 71
4.5.2 Summary ... 71
4.5.3 Key Management ... 72

5 RSA and ECC Cryptosystem ... 77
5.1 RSA Cryptosystem ... 77
5.1.1 RSA Cryptosystem Functions ... 77
5.1.2 Even Public Exponents ... 78
5.2 ECC Cryptosystem ... 78
5.2.1 Functionality Supported ... 78
5.2.2 Host Commands Supporting ECC ... 79
5.2.3 ECC General Information ... 79
5.3 General Information ... 80
5.3.1 HSM Buffer Sizes ... 80
5.3.2 Data Formats ... 80
5.4 Common Parameters ... 80
5.4.1 DES Key Type ... 80
5.4.2 Signature Algorithm ... 80
5.4.3 ECC Curve Reference ... 80
5.4.4 Encryption Identifier ... 81
5.4.5 Hash Identifier ... 81
5.4.6 Pad Mode Identifier ... 83
5.4.7 Key Data Block Type ... 85
5.4.8 Public Key Encoding ... 88
5.4.9 Signature Format & Signature Output Format (ECC Only) ... 90

5.5 Worked Examples 90

5.5.1 Sample Data Definitions (1024-bit RSA keys) 90

5.5.2 Sample Data Generation Procedure (1024-bit RSA keys) 92

5.5.3 Sample Data Calculation (1024-bit RSA keys) 93

5.5.4 Sample Data Definitions (2048-bit RSA keys) 95

5.5.5 Sample Data Generation Procedure (2048-bit RSA keys) 98

5.5.6 Sample Data Calculation (2048-bit RSA keys) 100

6 Local Master Keys (LMKs) 103

6.1 Introduction 103

6.2 Multiple LMKs 103

6.3 Key Block & Variant Key Comparison Table 103

6.4 Converting Key Names 105

7 Variant LMK Key Scheme 106

7.1 How the Variant Scheme Works 106

7.2 Local Master Key (LMK) Variants 107

7.2.1 Double-length Variant LMK 108

7.2.2 Triple-length Variant LMK 108

7.2.3 Local Master Key Triple DES Variant scheme 109

7.3 Test Variant LMK 112

7.3.1 Double-length Variant LMK 113

7.3.2 Triple-length Variant Test LMK 114

7.4 Variant Key Type Codes 115

7.5 Key Type Table 118

7.5.1 Non PCI-HSM Compliant (for backwards compatibility) 119

7.5.2 PCI Compliant 120

8 Key Block LMK Key Scheme 122

8.1 Introduction 122

8.2 Thales Key Blocks 122

8.3 Thales Key Block & PCI Compliance 122

8.4 Support for Thales Key Blocks 122

8.5 Key Block Format 123

8.5.1 Key Block Header 123

8.5.2 Optional Header 131

8.5.3 Permitted Key Status transitions 133

8.6 Encrypted Key Data 133

8.7 Authenticator 134

8.8 Key Block Local Master Keys (LMKs) 134

8.8.1 3DES Key Block Test LMK 135

8.8.2 AES Key Block Test LMK ... 135
8.9 Console Command examples ... 135
8.9.1 CK Console Command (Generate Check Value) ... 135
8.9.2 KG Console Command (Generate Key) ... 135
8.10 Host Commands ... 137
8.10.1 Host Command Examples ... 137
8.10.2 Error Codes ... 137

9 Multiple LMKs ... 138
9.1 Introduction ... 138
9.2 Need for Multiple LMKs ... 138
9.2.1 About LMKs ... 138
9.2.2 Multiple LMKs ... 138
9.3 Multiple LMK Licensing ... 139
9.4 Managing Multiple LMKs ... 139
9.4.1 LMK Component Generation ... 139
9.4.2 Migration from Old to New LMK ... 140
9.5 Authorization ... 141
9.6 LMK Table ... 142
9.7 Deleting LMKs ... 143
9.8 Identifying the Required LMK ... 144
9.8.1 Console Commands ... 144
9.8.2 Host Commands ... 144
9.9 Settings per LMK ... 146
9.9.1 Summary ... 146
9.9.2 Configuration of Settings per LMK ... 147
9.9.3 Security Settings ... 147
9.9.4 Security Settings and PCI HSM Compliance ... 148
9.9.5 PIN Block Formats ... 148
9.9.6 Enabling and Disabling Host Commands ... 148
9.9.7 Saving Configuration to Smart Card ... 149
9.9.8 Adding Additional Multiple LMK Licenses ... 149
9.9.9 Use of Default and Management LMK ID ... 149
9.9.10 LMK Identifier Security Settings ... 150
9.9.11 Disabling Settings per LMK ... 150
9.9.12 Additional Information ... 151

10 Migrating LMKs ... 152
10.1 Introduction ... 152
10.2 Multiple LMKs ... 152
10.3 LMK Migration Process Overview ... 152

10.3.1 Generating New LMK Component Smartcards ... 153
10.3.2 Types of LMK Component Cards ... 154
10.4 Formatting LMK Smart Cards ... 154
10.4.1 HSM LMK Cards ... 154
10.4.2 payShield Manager LMK Cards ... 154
10.5 Generating LMK component cards ... 154
10.5.1 HSM LMK Cards ... 154
10.6 Creating Copies of LMK Component Cards ... 155
10.6.1 HSM LMK cards ... 155
10.6.2 payShield Manager RLMK Card ... 155
10.7 Loading the New LMK ... 155
10.7.1 Using the Console ... 156
10.7.2 Load Using payShield Manager ... 156
10.7.3 Checking the LMK ... 157
10.8 Loading the Old LMK ... 157
10.8.1 Using the Console ... 157
10.8.2 Load Using payShield Manager ... 157
10.9 Migrating Keys Between Variant LMKs ... 157
10.9.1 The BW Host Command ... 158
10.9.2 The BX Response to the Host ... 160
10.10 Migrating Keys from Variant to Key Block LMKs ... 160
10.10.1 The BW Host Command ... 161
10.10.2 The BX Response to the Host ... 162
10.11 Migrating Keys Between Key Block LMKs ... 163
10.11.1 The BW Host Command ... 163
10.11.2 The BX Response to the Host ... 164
10.12 Migrating Keys from Key Block to Variant LMKs ... 165
10.13 Migrating Keys for PCI HSM Compliance ... 165
10.14 Re-encrypting PINs ... 165
10.14.1 The BG Host Command ... 165
10.14.2 The BH Response ... 166
10.15 Re-encrypting Decimalization Tables ... 166
10.16 Switching to the New LMK ... 168
10.17 Taking Advantage of Multiple LMKs ... 168
10.18 Tidying Up After Migration to a New LMK ... 169
10.18.1 Deleting the Old LMK from Key Change Storage ... 169
10.18.2 Deleting the New LMK ... 171

11 X9.143/TR-31 Key Blocks ... 172

11.1 X9.143/TR-31 Key Block Structure ... 172

11.2 Key Block Header 173
11.2.1 Version ID (Byte 0) 173
11.2.2 Key Block Length (Bytes 1-4) 174
11.2.3 Key Usage (Bytes 5-6) 174
11.2.4 Algorithm (Byte 7).. 175
11.2.5 Mode of Use (Byte 8).. 176
11.2.6 Key Version Number (Bytes 9-10) 176
11.2.7 Exportability (Byte 11) 176
11.2.8 Number of Optional Blocks (Bytes 12-13).. 177
11.2.9 Example X9.143/TR-31 Key Block Header 177
11.3 Optional Header.. 178
11.3.1 Standard Optional Header.. 178
11.3.2 Extended Numeric Optional Blocks 180
11.4 Using X9.143/TR-31 Key Blocks 180
11.4.1 Key Scheme Tags 180
11.4.2 Optional Blocks 181
11.5 GISKE Key Block 181

12 DES Key Support 183
12.1 DES Keys.. 183
12.2 Key Usage 183
12.3 Key Encryption Schemes 183
12.3.1 ANSI X9.17 Method.. 183
12.4 Key Generate, Import and Export.. 184
12.5 Rejection of Weak, Semi-Weak, & Possibly Weak Keys.. 185
12.5.1 DES Weak Keys.. 185
12.5.2 DES Semi-Weak Keys.. 185
12.5.3 DES Possibly Weak Keys.. 186

13 User Storage.. 188
13.1 Introduction 188
13.2 User Storage Area 188
13.3 Defining Block Size 189
13.4 User Storage when using Fixed Block Sizes 189
13.4.1 Loading Data into User Storage with Fixed Block Sizes 189
13.4.2 Reading Data from User Storage with Fixed Block Sizes 190
13.4.3 Key Block LMKs 191
13.4.4 Storing General Data.. 192
13.4.5 PIN Solicitation Data.. 192
13.4.6 PIN Blocks.. 193
13.4.7 Diebold Table 193

13.4.8 Decimalization Tables ... 193
13.4.9 Storing a Mix of Data Types ... 193
13.5 User Storage when using Variable Block Sizes ... 194
13.5.1 Enabling the Enhanced Functionality ... 195
13.5.2 New Data Structure ... 195
13.5.3 Loading Data into User Storage ... 195
13.5.4 Storing a Diebold Table ... 196
13.5.5 Reading Data From User Storage ... 196
13.5.6 Referencing Keys Stored in User Storage ... 196
13.6 Storing RSA and ECC Keys Using Key Indexes ... 197
13.6.1 Storing the RSA and ECC Private Key ... 197
13.6.2 Using RSA and ECC Private Keys Held in the payShield 10K ... 197
13.7 Host Command Summary ... 198

14 PIN Block Formats ... 199
14.1 General ... 199
14.1.1 PCI HSM Compliance ... 199
14.2 PIN Block Translations and PCI ... 199
14.2.1 Permitted PIN Block Translations – Not PCI Compliant ... 200
14.2.2 Permitted PIN Block Format Translations – PCI Compliant ... 201
14.3 Thales PIN Block Formats ... 202
14.3.1 Format 01 ... 202
14.3.2 Format 02 ... 203
14.3.3 Format 03 ... 203
14.3.4 Format 04 ... 203
14.3.5 Format 05 ... 204
14.3.6 Format 34 ... 204
14.3.7 Format 35 ... 205
14.3.8 Format 41 ... 206
14.3.9 Format 42 ... 206
14.3.10 Format 46 ... 207
14.3.11 Format 47 ... 207
14.3.12 Format 48 ... 208
14.4 Support for Deprecated Proprietary PIN Block Format Tag J ... 209
14.4.1 Overview ... 209
14.4.2 Miscellaneous Host Commands ... 210
14.4.3 PIN Verification Host Commands ... 211
14.4.4 Clear PIN and Remaining PIN Generation Host Commands ... 212
14.4.5 PIN Translation from Encryption under an LMK Host Commands ... 213
14.4.6 PIN Translation to Encryption under an LMK Host Commands ... 214

15 Key Component Printing 216
15.1 Introduction 216
15.2 Process Description 216
15.2.1 Overview 216
15.2.2 Directly Attached Printers 217
15.2.3 Operation Without a Directly Attached Printer 218
15.2.4 Setting Up Key Component Print Forms 219
15.2.5 Security Considerations 221
15.2.6 Forming the KEK at the Issuer System 221
15.2.7 Forming the KEK at the Recipient System 222

16 Moving to PCI HSM Compliance 224
16.1 Introduction 224
16.2 PIN Blocks 224
16.2.1 Using PIN Block Formats in a Compliant Manner 224
16.2.2 Permitted PIN Block Format Translations 225
16.2.3 PIN Block Format for Values Derived from PIN & PAN 225
16.2.4 PIN Translations to/from LMK Encryption 226
16.2.5 Summary of Required Actions 226
16.3 Updating Key Type 002 in Host Commands 226
16.3.1 Example 227
16.3.2 Interoperability with Older HSMs 228
16.3.3 Summary of Required Actions 228
16.4 Migrating keys from Key Type 002 228
16.4.1 Migrating Keys WITHOUT a Change of LMK 229
16.4.2 Migrating Keys WITH a Change of LMK 230
16.4.3 Using Disaster Recovery (DR) Sites 231
16.4.4 Examples of Using the BW Host Command 231
16.5 Diebold Table Re-encryption 232
16.5.1 Summary of Actions Required 232

Appendix A - Key Scheme Table 233

Appendix B - Reduced Character Sets 236

Appendix C - Print Formatting Symbols 237

Appendix D - Example Laser Printer Formatting Control Codes 240

Appendix E - List of Authorized Activities 242

Appendix F - Printer Settings 247

Appendix G – SNMP Security Setting 248

Technical Support Contacts 251

Revision Status

Document Number	Revision	Date	Summary of changes
PUGD0541-001	001	April 2019	Initial issue
PUGD0541-002	002	July 2019	Editorial updates
PUGD0541-003	003	August 2020	Updates for Software Version v1.1a: Appendix G - Printer Settings
PUGD0541-004	004	January 2021	Updates for Software Version v1.2a including: Details on the support provided for Elliptic Curve Cryptography (ECC) in v1.2a included References to the Trusted Management Device (TMD) added Update to Section 1.10 User Storage A number of other updates and corrections have been included
PUGD0541-005	005	September 2021	Editorial updates Clarifications added to 5.4.7 Key Data Block Type
007-001518-005	005	February 2022	New document number Update to table in Section 6.3 Key Block & Variant Key Comparison Table regarding: ZKA - mode of use must be X to generate a keyblock with key usage 53
007-001518-006	006	June 2022	Updated MIB Added: Section 14.1.1 Security Guidelines for SNMP Configuration Updates to Section 4.3 Derived Unique Key Per Transaction (DUKPT) Updates to tables in Sections: 15.2.1 Permitted PIN Block Translations – Not PCI Compliant 15.2.2 Permitted PIN Block Format Translations – PCI Compliant 15.3 Thales PIN Block Formats

Document Number	Revision	Date	Summary of changes
			to indicate that Thales Format 48 is used to encrypt a PIN Block under an AES Key Block LMK
			Addition of Section 4.3.9 Italian DUKPT Standard Update to Section 4.4 German Banking Industry Committee (GBIC) Key Derivation Addition of Section 4.5 Italian Key Derivation Update to Section 6.3 Key Block & Variant Key Comparison Table for Italian Key Update to Section 8.5.1.2 Key Usage (Bytes 5-6) new key (‘57’ / Master MKPOS/MKSER) added Update to Section 8.5.1.8 LMK Identifier (Bytes 14-15) to clarify value “FF” Update to Section 11.3 Optional Header (to delete placeholder text) Addition of Section 11.4.2 Optional Blocks Update to Section 13.2 Support for AES Italian Standard and GBIC Update to Appendix C Thales Key Block / TR 31 Key Usage Conversion
007-001518-007	007	November 2022	Updates to Section 11 X9.143/TR-31 Key Blocks regarding X9.143/TR-31 Updates to the following sections for the AES ZKA feature: Section 4.4 German Banking Industry Committee (GBIC) Key Derivation Section 6.3 Key Block & Variant Key Comparison Table Section 8.5.1.2 Key Usage (Bytes 5-6) Section 8.5.2 Optional Header Section 11.3.1 Standard Optional Header Section 11.2.3 Key Usage (Bytes 5-6) Section 13.2 Support for AES Appendix C: Thales Key Block/ TR 31 Key Update to: Appendix A - Key Scheme Table for AES CBC Export feature

Document Number	Revision	Date	Summary of changes
007-001518-007	007 Rev. B	December 2022	Update to description of Thales Key Block in Chapter 8 Key Block LMK Key Scheme
007-001518-007	007 Rev. C	March 2023	Update to Appendix F - List of Authorized Activitiesto include ECC
007-001518-008	008	May 2023	Update to Section 2.1 TCP/IP Protocol to reflect 128 TCP sockets Appendix G - Printer Settings Updated MIB in Appendix E - SNMP MIB Update to the following sections for the X9.143/TR-31 HMAC Feature: Section 11.2.4, Algorithm (Byte 7) Section 11.3.1 Standard Optional HeaderStandard Optional Header Addition of Section 15.4, Support for Deprecated Proprietary PIN Block Format Tag J Update to Section 4.3.10 AES DUKPT for clarification
007-001518-008	008 Rev. B	May 2023	Update to Appendix F - List of Authorized Activities for ECC
007-001518-009	009	September 2023	Addition of: Section 4.5 Italian Key Derivation Section 14.5.13, Security Settings and Appendix I, SNMP Security Setting Reference Numbers Update to table entries to reflect "K1" Key Block Protection Key; see sections: 8.5.1.2 Key Usage (Bytes 5-6), 11.2.3 Key Usage (Bytes 5-6) and 13.2 Support for AES
007-00118-009	009 Rev. B	January 2024	Clarifications added to Appendix H – SNMP Security Setting Addition of Section 1.2 Printer Requirements Update to Section 4.4, German Banking Industry Committee (GBIC) Key Derivation Update to Appendix A - Key Scheme Table • to include new Scheme Key tags: N, O, and M

Document Number	Revision	Date	Summary of changes
			Update to Section 4.3.3, The Base Derivation Key (BDK) to reflect support of triple-length DES Removed *Chapter 13 - AES Key Support* as this duplicatedinformation provided elsewhere in this document. Removed *Appendix C - Thales Key Block / X9.143/TR-31 Key Usage Conversion* as this information is provided in section 8.5.1.2.
007-00118-009	009 Rev. C	February 2024	Update to Appendix H – SNMP Security Settings
007-00118-020	020	April 2024	Chapter title update and additional sections added to Chapter 9 Settings per LMK Addition of Section 9.9, Settings per LMK Addition of Chapter 15, Remote Syslog Update to Appendix H – SNMP Security Settings
007-00118-020	020 Rev. B1	August 2024	Update to titles in Sections 2.1.2 and 2.1.3 Update to Section 4.3, Derived Unique Key Per Transaction (DUKPT) Update to Section 8.5.1.2, Key Usage (Bytes 5-6) to reflect support of AES for additional keys (Updates to Keys: 11, 12, 13, E1, E2, E3, 35, K1, 53, 55, M4, M5, and V2) Update to Section 11.2.3, Key Usage (Bytes 5-6) to reflect addition of "24" for value K0 Update to Appendix A - Key Scheme Table to include 128-bit and 256-bit AES key in EBC mode Updates to "Notes" for Key Scheme Tags: N, P, Q, W Deletion of Key Schemes PG and WG Update to Appendix F - List of Authorized Activities; follow this link: *SYMMETRIC KEY EXPORT*
007-00118-021	021 Rev A	October 2024	Update to Section 11, X9.143/TR-31 Key Blocks to include references to X9.143. Update to Key Usage table in Section 8.5.1.2, Key Usage (Bytes 5-6)

Document Number	Revision	Date	Summary of changes
			- Updated Thales KB Key Usage entries: 13, E1, E3, V2, D, C0 Update to Appendix A - Key Scheme Table - Updated Key Scheme Tags: XI, PG, PI, P, WG, WI, W Addition of Section 5.4.6.3, 04 = PKCS#1 v2.1 method (EMSA-PSS) Update to Section 11.2.3, Key Usage (Bytes 5-6) to include M7 The SNMP and Remote Syslog chapters and the table containing the SNMP MIB Security Settings (previously Appendix E) are now found in the payShield 10K Installation and User Guide
007-00118-022	022 Rev A	January 2025	Update to Section 4.4, German Banking Industry Committee (GBIC) Key Derivation Update to Section 14.2.2, Permitted PIN Block Format Translations – PCI Compliant
007-00118-022	022 Rev A2	February 2025	Update to Appendix G – SNMP Security Setting Update to Section 14.2.2, Permitted PIN Block Format Translations – PCI Compliant for clarification regarding the GBIC Scheme and the destination key Update to Section 4.4, German Banking Industry Committee (GBIC) Key Derivation to include additional clarification

References

The following documents are referenced within this manual.

Reference Number	Title
1	payShield 10K Installation and User Guide
2	payShield 10K Core Host Commands reference manual
3	PKCS#1: RSA Cryptography Standard – Version 2.2 October 2012
4	Visa Integrated Circuit Card Specification, Version 1.5 May 2009
5	MasterCard M/Chip 4 Security and Key Management 1.0 October 2002
6	ASC X9 TR-31, Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms, 2018.
7	Global Interoperable Secure Key Exchange Key Block Specification, version 2.3, written by ACI Worldwide, HP Atalla, Diebold, Thales e-Security and VeriFone Inc. 2002.
8	payShield Trusted Management Device (TMD) User Guide
9	ANSI X9.143-2022 Retail Financial Services Interoperable Secure Key Block Specification

1 Introduction

The payShield 10K hardware security module (HSM) is an external device that provides secure cryptographic processing for a host computer in a physically protected environment. It receives commands via a data link and fulfills processing on behalf of the host.

Among its functions, the HSM manages encryption keys for secure communication between devices or systems. It is responsible for generating and safeguarding these keys to prevent unauthorized access. Additionally, the HSM performs critical cryptographic functions, such as validating PIN codes and creating MAC codes.

The Personal Identification Number (PIN) serves as a confidential code used for user authentication, and the HSM can verify this code to establish the user's authorization to access the system. Furthermore, a Message Authentication Code (MAC) is a unique code integrated into a message to ensure its integrity and authenticity. The HSM has the capability to produce and authenticate MAC codes as needed, ensuring that messages remain unaltered during transmission.

This manual contains programming notes to assist the application programmer. Refer to the payShield 10K Core Host Commands reference manual for a full discussion of Core Host Commands.

Note: All Host Commands are disabled by default.

1.1 General Information

The application program sends commands to an HSM and receives responses from it. Each command and response consists of a variable number of fields.

Versions of the payShield 10K HSM can be configured to support TCP/IP and UDP communications protocols. The HSM has no flow control support so the programmer must ensure that the HSM input buffer is not exceeded.

The HSM returns an error code to the Host as part of the response message. The programmer must ensure that a suitable response is made to each type of error.

In a typical system, a minimum of two HSMs are connected to the Host via separate Host ports. The HSM units are independent, and the programmer should make maximum use of all the HSM units to increase throughput, using one HSM if another is already processing data or is faulty. Also, it is useful to ensure that the program allows for additional HSM units to be subsequently added as throughput requirements increase.

Each HSM has a user storage area reserved for use by the programmer to store data required during processing. Typically, it is used to store keys and tables. Instructing an HSM to access data from user storage reduces the amount of data necessary in each command, and thus reduces the communications time.

There is a facility to print data (e.g., account holder PINs) at a printer connected to a payShield 10K HSM.

The HSM must have format information for the data before sending it to the printer. The program must send a print format command to an HSM before print commands can be issued.

1.2 Printer Requirements

- Both the printer and the HSM should be located in the same secure room
- When printing PINs, the printer should not be left unattended
- The cable connecting the HSM to the printer should not exceed 2.5 meters
- Converters/extensions are not supported
- Refer to the payShield 10K Console Manual (CP command) for additional information on configuring the printer, e.g., adjusting timeout and delay values, etc.
- Printer must support ASCII

Note For additional information on configuring the printer, e.g., setting I/F mode, adjusting timeout and delay values, etc., refer to:

  - *Appendix F - Printer Settings*
  - payShield 10K Installation and User Guide (Section 9.10.3.5) or the
  - payShield 10K Console Manual (CP command)

1.3 Command Message Format

To give an HSM an instruction, the Host application must assemble a message containing all the necessary information and send it to an HSM as a sequence of characters on the communications link. In general, each command consists of the following fields:

- Message Header

The message header field can be any length from 1 to 255 characters and it is configured at installation. It can contain any printable characters and an HSM returns them unmodified in the response message.

It can be used to label commands and their responses for systems that implement batch queues or which multi-thread commands.

- Command Code

Every command has a unique two-character command code. The command codes are detailed in the *payShield 10K Core Host Commands reference manual*.

- Data

Most HSM commands require data, often including cryptographic keys. Details of the data for each command can be found in the *payShield 10K Core Host Commands reference manual*.

- Message Trailer

The message trailer is an additional variable-length field (to a maximum of 32 characters), which can be used to pass additional details required by the Host for further processing. The field should always be preceded by the EM control character; ASCII and EBCDIC value is X'19.

The data in this field can be any printable character, and it is returned in the response message unchanged when:

  - the "no error" error code 00 is returned; or

- a "warning" error code is returned (as detailed in the host command's response message description).

For all other error codes, the message trailer is not returned.

1.4 Response Message Format

To inform the Host of the results of processing, an HSM sends a message containing all the necessary information as a sequence of characters on the communications link. A response message is generated for each of the following:

- In response to a command
- As a second response to a print command after an HSM has finished sending the print data to the printer.
- In response to the entry of PIN solicitation data at the console (but only after the Host has enabled this function).

Each response from the HSM consists of the following:

- Message Header

The message header field is a copy of the field received in the command message from the Host. The data is returned to the Host unchanged.

It can be used to label commands and their responses for systems that implement batch queues or which multi-thread commands.

- Response Code

Every response has a unique two-character code. Normally this code has the same first character as the command to which it is a response, and the second character is one greater than the second character of the command (e.g., if the command code is AA, the response code is AB). The value of each code is detailed in the *payShield 10K Core Host Commands reference manual*.

- Error Code

The two-character error code field is used by an HSM to report errors detected during processing. The values are alphanumeric and the value 00 indicates that no errors have been found. If an error (other than 00) is returned, subsequent fields, with the exception of the end of text character, are not returned by an HSM. Error codes specific to a command or frequently returned with a command are listed with the command code in the payShield 10K Core Host Commands reference manual. A list of global errors is also included in the *payShield 10K Core Host Commands reference manual*.

- Datas

Many payShield 10K HSM commands return data as a result of the processing. Details of the contents of the returned data are given in the *payShield 10K Core Host Commands reference manual*. Generally, data is not returned for error codes other than 00. There are some exceptions to this rule, for example, the Key Import command (A6) returns error code 01 to advise that the key being imported does not have odd parity.

- Message Trailer

The message trailer field is returned only if it was present in the command message and when:

- the "no error" error code 00 is returned; or
- a "warning" error code is returned (as detailed in the host command's response message description).

For all other error codes, the message trailer is not returned.

1.5 Data Representation

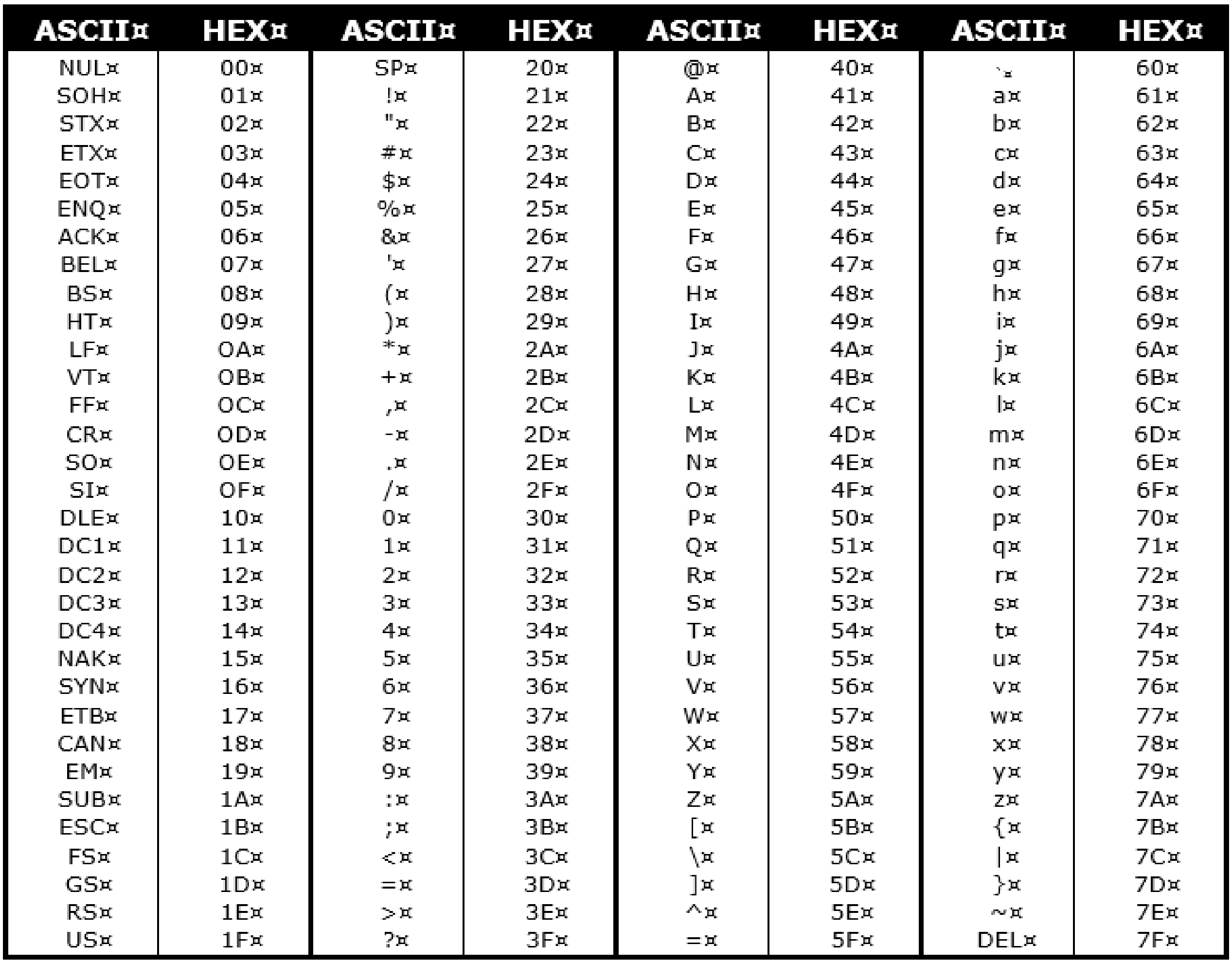
When sending data to an HSM, other than data that is already in character format, encode each digit (0-9, A-F) as a character (e.g., to send the hexadecimal value 1234ABCD to an HSM requires eight characters).

For Ethernet communications, the HSM accepts certain fields in binary format. Refer to individual host commands for full details.

Note: The payShield 10K HSM automatically detects whether an incoming command message uses ASCII or EBCDIC characters, and processes the command accordingly, returning the result in the same format.

1.5.1 ASCII Character Codes

The table below shows the ASCII characters and their hexadecimal values.



ASCII¤	HEX¤	ASCII¤	HEX¤	ASCII¤	HEX¤	ASCII¤	HEX¤
NUL¤	00¤	SP¤	20¤	@¤	40¤	`¤	60¤
SOH¤	01¤	!¤	21¤	A¤	41¤	a¤	61¤
STX¤	02¤	"¤	22¤	B¤	42¤	b¤	62¤
ETX¤	03¤	#¤	23¤	C¤	43¤	c¤	63¤
EOT¤	04¤	$¤	24¤	D¤	44¤	d¤	64¤
ENQ¤	05¤	%¤	25¤	E¤	45¤	e¤	65¤
ACK¤	06¤	&¤	26¤	F¤	46¤	f¤	66¤
BEL¤	07¤	'¤	27¤	G¤	47¤	g¤	67¤
BS¤	08¤	(¤	28¤	H¤	48¤	h¤	68¤
HT¤	09¤	)¤	29¤	I¤	49¤	i¤	69¤
LF¤	0A¤	*¤	2A¤	J¤	4A¤	j¤	6A¤
VT¤	0B¤	+¤	2B¤	K¤	4B¤	k¤	6B¤
FF¤	0C¤	,¤	2C¤	L¤	4C¤	l¤	6C¤
CR¤	0D¤	-¤	2D¤	M¤	4D¤	m¤	6D¤
SO¤	0E¤	.¤	2E¤	N¤	4E¤	n¤	6E¤
SI¤	0F¤	/¤	2F¤	O¤	4F¤	o¤	6F¤
DLE¤	10¤	0¤	30¤	P¤	50¤	p¤	70¤
DC1¤	11¤	1¤	31¤	Q¤	51¤	q¤	71¤
DC2¤	12¤	2¤	32¤	R¤	52¤	r¤	72¤
DC3¤	13¤	3¤	33¤	S¤	53¤	s¤	73¤
DC4¤	14¤	4¤	34¤	T¤	54¤	t¤	74¤
NAK¤	15¤	5¤	35¤	U¤	55¤	u¤	75¤
SYN¤	16¤	6¤	36¤	V¤	56¤	v¤	76¤
ETB¤	17¤	7¤	37¤	W¤	57¤	w¤	77¤
CAN¤	18¤	8¤	38¤	X¤	58¤	x¤	78¤
EM¤	19¤	9¤	39¤	Y¤	59¤	y¤	79¤
SUB¤	1A¤	:¤	3A¤	Z¤	5A¤	z¤	7A¤
ESC¤	1B¤	;¤	3B¤	[¤	5B¤	{¤	7B¤
FS¤	1C¤	<¤	3C¤	\¤	5C¤	|¤	7C¤
GS¤	1D¤	=¤	3D¤	]¤	5D¤	}¤	7D¤
RS¤	1E¤	>¤	3E¤	^¤	5E¤	~¤	7E¤
US¤	1F¤	?¤	3F¤	=¤	5F¤	DEL¤	7F¤

1.5.2 EBCDIC Character Codes

The table on the following page shows the EBCDIC characters and their hexadecimal values.

EBCDIC	HEX	EBCDIC	HEX	EBCDIC	HEX	EBCDIC	HEX
NUL	00	SP	40		80		C0
SOH	01		41	a	81	A	C1
STX	02		42	b	82	B	C2
ETX	03		43	c	83	C	C3
	04		44	d	84	D	C4
HT	05		45	e	85	E	C5
	06		46	f	86	F	C6
DEL	07		47	g	87	G	C7
	08		48	h	88	H	C8
	09		49	i	89	I	C9
	0A		4A		8A		CA
VT	0B	.(period)	4B	{	8B		CB
FF	0C	<	4C		8C		CC
CR	0D	(	4D		8D		CD
SO	0E	+	4E		8E		CE
SI	0F	|	4F		8F		CF
DLE	10	&	50		90		D0
DC1	11		51	j	91	J	D1
DC2	12		52	k	92	K	D2
DC3	13		53	l	93	L	D3
	14		54	m	94	M	D4
	15		55	n	95	N	D5
BS	16		56	o	96	O	D6
	17		57	p	97	P	D7
CAN	18		58	q	98	Q	D8
EM	19		59	r	99	R	D9
	1A	!	5A		9A		DA
	1B	$	5B	}	9B		DB
	1C	*	5C		9C		DC
	1D	)	5D		9D		DD
	1E	;	5E		9E		DE
	1F		5F		9F		DF
	20	- (minus)	60		A0	\	E0
	21	/	61	~ (tilde)	A1		E1
FS	22		62	s	A2	S	E2
	23		63	t	A3	T	E3
	24		64	u	A4	U	E4
LF	25		65	v	A5	V	E5
ETB	26		66	w	A6	W	E6
ESC	27		67	x	A7	X	E7
	28		68	y	A8	Y	E8
	29		69	z	A9	Z	E9
	2A		6A		AA		EA
	2B	,(comma)	6B		AB		EB
	2C	%	6C		AC		EC
ENQ	2D	_(underscore)	6D	[	AD		ED
ACK	2E	>	6E		AE		EE
BEL	2F	?	6F		AF		EF
	30		70		B0	0	F0
	31		71		B1	1	F1
SYN	32		72		B2	2	F2
	33		73		B3	3	F3
	34		74		B4	4	F4
	35		75		B5	5	F5

EBCDIC	HEX	EBCDIC	HEX	EBCDIC	HEX	EBCDIC	HEX
	36		76		B6	6	F6
EOT	37		77		B7	7	F7
	38		78		B8	8	F8
	39	`(grave)	79		B9	9	F9
	3A	:	7A		BA		FA
	3B	#	7B		BB		FB
DC4	3C	@	7C		BC		FC
NAK	3D	'	7D	]	BD		FD
	3E	=	7E		BE		FE
SUB	3F	"	7F		BF		FF

1.6 Input/Output Flow Control

There is no flow control provided by a payShield 10K HSM. It is the responsibility of the application to ensure that the input buffer in the HSM, which is 32K bytes per connection, is not exceeded.

1.7 Error Handling

There are four types of errors generated by a payShield 10K HSM:

- Fatal errors
- Non-recoverable errors
- Recoverable errors
- Programming errors

Fatal errors indicate a hardware fault in the equipment. Such an error should be logged and reported for user action to be taken (e.g., report to supervisor). Fatal errors are normally reported on the console and are not seen by the host application. The host application usually times out if a fatal error occurs.

Non-recoverable errors cannot be rectified by the program and need user intervention (e.g., with an HSM set into the Authorized state). Such errors should also be logged and reported for user action to be taken (e.g., report to supervisor). This type of error does not mean that an HSM cannot action other types of commands.

Recoverable errors may be the result of data corruption or indicate that an HSM cannot process a command because some other action is required first. The application should attempt to recover by re-issuing the command, attempting to clear the corruption or by implementing the missing action (e.g., an HSM reports that the print format definition is not loaded, so the program should load it and re-issue the failed command).

Programming errors are normally found during testing, but if they occur at other times, they are probably non- recoverable.

Additionally, the application should monitor an HSM for timeouts on the interface.

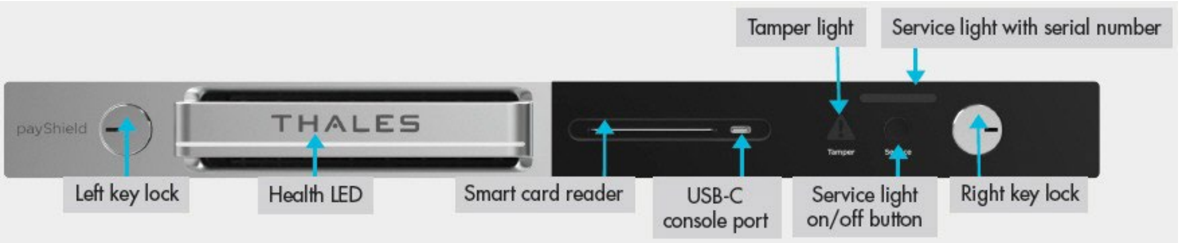
In any of the above events, the application should try to continue processing by using another HSM to action the command. Continued failure may indicate a catastrophic failure of all HSM units (unlikely), a power failure or a program error.

The application should monitor usage of all HSM units and mark any unit as “out of service” if it has given a fatal error, or where a unit repeatedly reports non-recoverable errors.

1.8 Error Logs

Hardware failures, software errors and alarm events are recorded in the Error Log. This has 100 slots, with fields for error code, sub-code, date, time and severity level. When an error is recorded for a particular error code, any subsequent error with the same code updates the date and time for that code, thus each error type remains in the log until it is cleared.

The status of the payShield 10K is displayed via the Health LED on the front of the unit.



- Solid white indicates no errors or no unread errors in the error log
- Solid red indicates unread error(s) in the error log

Error log maintenance can be performed two ways: via payShield Manager and via console command.

- Refer to the *payShield 10K Installation and User Guide*, section titled “Using payShield Manager.
- Refer to the *payShield 10K Console Guide* for discussion of the ERRLOG and CLEARERR commands.

An example of the ERRLOG console command follows:

Entries in the HSM error log have a hash-based integrity check using HMAC. In this example the verification of integrity of the entry failed. A message indicates that an error happened during the verification process and the entry is shown as Unparsed.

```
Offline> ERRLOG <Return>
Error Log (3 entries)
--------------------------
973: May 31 15:17:35 ERROR: [FAN 1 is now present] (Severity: 3, Code = 0x00000003, Sub-Code =
0x00000018)
Error hmac missmatch - Unable to verify text integrity
 974: UNPARSED [[FAN1 is missing, setting FAN??? speed to 16000 RPM] (Severity: 3, Code =
0x00000003, Sub-Code = 0x00000018]
 975: May 31 17:33:14 ERROR: [FAN 1 is now NOT present] (Severity: 3, Code = 0x00000003, Sub-
Code = 0x00000018)
Please copy this log to a text file and send it to your regional Thales E-Security Support
center.
Confirm error log has been read and error light should be extinguished? [Y/N]: Y <Return>

Offline>
```

1.9 Multiple HSMs

A typical system has two or more HSM units connected as 'live' units. This provides increased capability where the processing requires more than one HSM, and provision for backup in the event of an HSM failure.

Each HSM is normally connected to the Host via a separate Host port, although a port- sharing unit can be used if the number of Host ports available is limited. The sharing configuration is not capable of providing backup if the port or the port-sharing unit becomes faulty.

Optionally it is possible to have a backup unit not connected to the Host but ready for connection in place of a faulty unit. This is not the preferred practice because the unit may remain idle for a long time and may itself have developed a fault.

In addition to the 'live' units, a typical system contains at least one HSM connected to a test or development computer system. This allows changes in the environment to be tested, without disturbing the live system.

1.10 User Storage

It is possible to store keys and other data securely inside the payShield 10K using the User Storage facility. Follow this link to: Section 13 User Storage.

2 Host Connections

The following communication connections are possible between the payShield 10K HSM and the Host computer:

- Ethernet TCP/IP
- Ethernet UDP

2.1 TCP/IP Protocol

IP addresses can be configured manually or obtained automatically by using DHCP. DNS is also supported to allow the HSM to be addressed by name rather than by its IP address.

The HSM employs TCP for the transfer of data. It acts as a TCP server supporting multiple TCP clients configurable via the CH command. The maximum number of TCP sockets that can be supported is 128. If a TCP client attempts to establish a connection with an HSM that already has the maximum number of configured sockets active, the TCP client's request is rejected.

The HSM supports the TCP Push function. To improve the efficiency of data transfer the TCP protocol software can buffer data into larger blocks, or divide the data into smaller blocks. This is useful for time-critical applications, such as transaction processing systems, where response time is more important than Ethernet utilization efficiency.

The HSM always returns a response to a command using the Push function.

The payShield 10K Secure Host Communications facility supports the use of TLS to protect the TCP/IP link between the HSM and the host. The payShield 10K also allows whitelists of acceptable host IP addresses to be configured using its Access Control Lists (ACL) facility.

Refer to the *payShield 10K Security Manual* for additional information regarding TLS. Refer to the *payShield 10K Console Guide* for additional information regarding the CH console command.

2.1.1 Port Addresses

When the port is configured using the Console or payShield Manager, a well-known port address is assigned (default value 1500). Refer to the *payShield 10K Core Host Commands reference manual*.

2.1.2 Sending Commands and Returning Responses

The sections that follow, define host command formats.

2.1.2.1 Sending commands

The HSM expects a command to be sent in the form defined in the following table.

Field	Size	Format	Description
LENGTH	2	Byte	Length of the COMMAND field
COMMAND	n	Byte	HSM command

Single or Multiple commands can be sent to an HSM within one TCP transmission. Each should be of the form defined in the above table.

2.1.2.2 Returning Responses

When an HSM receives a command from a TCP client, the command is processed and the response returned to the TCP client. The response is of the form defined in the table below.

Field	Size	Format	Description
LENGTH	2	Byte	Length of the RESPONSE field
RESPONSE	n	Byte	HSM response

The result of each command sent to an HSM is returned as a separate response to the TCP client. This also operates when multiple commands are sent to an HSM in a single TCP transmission.

All HSM responses are returned to the TCP client using the TCP Push function.

2.1.3 Example of Sending Commands and Returning Responses

The following sections provides examples of sending a single host command or multiple host commands in one TCP transmission.

2.1.3.1 TCP Transmission Containing One Request

The command format for a diagnostics command (NC) is:

One host command: X'00 X'06 X'31 X'32 X'33 X'34 X'4E X'43

length byte 1	length byte 2	Command code byte 1	Command code byte 2	Command code byte 3	Command code byte 4	Command code byte 5	Command code byte 6
X'00	X'06	X'31	X'32	X'33	X'34	X'4E	X'43

where the HSM message header length is set to 04, a message header of 1234 is used, and character representation is ASCII.

2.1.3.2 TCP Transmission Containing Two Requests

The response format from a diagnostics command 'NC' is:

X'00	X'21	X'31	X'32	X'33	X'34	X'4E	X'44	X'30	X'30	X'32	X'36
X'38	X'36	X'30	X'34	X'37	X'34	X'34	X'34	X'39	X'31	X'32	X'34
X'32	X'32	X'30	X'30	X'30	X'37	X'2D	X'45	X'30	X'30	X'30	

where the HSM message header length is set to 04, a message header of 1234 is used, and the character representation is ASCII.

The example shows the error code returned was 00 and the LMK check value returned was 2686047444912422 and the firmware installed is 0007-E000.

2.1.3.3 One TCP transmission Contains Two Requests

The command format for a diagnostics command (NC) is:

First host command: X'00 X'06 X'31 X'32 X'33 X'34 X'4E X'43

Second host command: X'00 X'06 X'31 X'32 X'33 X'34 X'4E X'43

Following table shows two host commands in one TCP transmission:

length byte 1	length byte 2	Header byte 2	Header byte 1	Header byte 1	Header byte 1	Command code byte 1	Command code byte 2
X'00	X'06	X'31	X'32	X'33	X'34	X'4E	X'43
length byte 1	length byte 2	Header byte 2	Header byte 1	Header byte 1	Header byte 1	Command code byte 1	Command code byte 2
X'00	X'06	X'31	X'32	X'33	X'34	X'4E	X'43

where the HSM message header length is set to 04, a message header of 1234 is used, and character representation is ASCII.

2.1.3.4 Two TCP Transactions are Received and Each Packet Contains Response

The response format from a diagnostics command 'NC' is:

First response packet:

X'00	X'21	X'31	X'32	X'33	X'34	X'4E	X'44	X'30	X'30	X'32	X'36
X'38	X'36	X'30	X'34	X'37	X'34	X'34	X'34	X'39	X'31	X'32	X'34
X'32	X'32	X'30	X'30	X'30	X'37	X'2D	X'45	X'30	X'30	X'30	

where the HSM message header length is set to 04, a message header of 1234 is used, and the character representation is ASCII.

The example shows the error code returned was 00 and the LMK check value returned was 2686047444912422 and the firmware installed is 0007-E000.

Second response packet:

X'00	X'21	X'31	X'32	X'33	X'34	X'4E	X'44	X'30	X'30	X'32	X'36
X'38	X'36	X'30	X'34	X'37	X'34	X'34	X'34	X'39	X'31	X'32	X'34
X'32	X'32	X'30	X'30	X'30	X'37	X'2D	X'45	X'30	X'30	X'30	

where the HSM message header length is set to 04, a message header of 1234 is used, and the character representation is ASCII.

The example shows the error code returned was 00 and the LMK check value returned was 2686047444912422 and the firmware installed is 0007-E000.

2.2 UDP Protocol

The HSM client expects all UDP connections to be made on the Well-Known-Port at the IP address. The IP address and Well-Known-Port address are defined to the HSM when configuring the software settings with the Console 'CH' command.

All UDP host clients sending data to an HSM send the datagrams to the Well-Known-Port at the IP address. The HSM (UDP server) processes the datagram and returns a datagram response to the originating UDP host client.

UDP is a connection-less protocol. If an HSM detects an error in a received datagram it is discarded. The UDP host client should support a time-out mechanism whereby if a response is not received within the time-out period the original request is re-sent.

2.2.1 Sending Commands

The payShield 10K HSM expects a command to be sent in the form defined in the table.

Field	Size	Format	Description
LENGTH	2	Byte	Length of the COMMAND field
COMMAND	n	Byte	HSM command

Only a single command can be sent to an HSM in one UDP transmission (packet).

Example:

The command format for a diagnostics command (NC) is:

X’00 X’06 X’31 X’32 X’33 X’34 X’4E X’43

where the HSM message header length is set to 04, a message header of 1234 is used, and character representation is ASCII.

2.2.2 Returning Responses

When an HSM receives a command from a UDP client the command is processed and the response returned to the UDP client. The response is of the form defined in the table.

Field	Size	Format	Description
LENGTH	2	Byte	Length of the RESPONSE field
RESPONSE	n	Byte	HSM command

The result of each command sent to an HSM is returned as a separate response to the UDP client.

Example:

The response format from a diagnostics command 'NC' is:

```
X’00  X’21  X’31  X’32  X’33  X’34  X’4E  X’43  X’30  X’30  X’32  X’36
X’38  X’36  X’30  X’34  X’37  X’34  X’34  X’34  X’39  X’31  X’32  X’34
X’32  X’32  X’30  X’30  X’30  X’37  X’2D  X’45  X’30  X’30  X’30
```

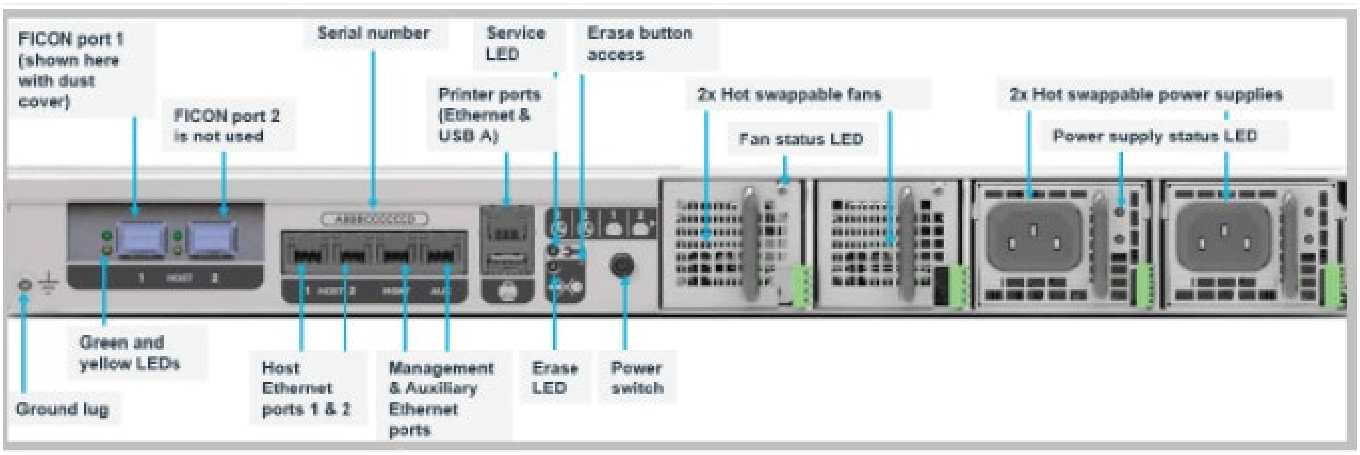
where the HSM message header length in set to 04, a message header of 1234 is used, and the character representation is ASCII.

The example shows the error code returned was 00 and the LMK check value returned was 2686047444912422 and the firmware installed is 0007-E000. The *payShield 10K Installation and User Guide* provides extensive information about setting up FICON on the payShield 10K.

2.3 payShield 10K PS10-F (FICON)

FICON is the system for fiber-optic connections to IBM mainframes, and replaces the older ESCON system.

The payShield 10K can be ordered with an auto-sensing factory-fitted FICON (Fiber Connection) interface. This provides a port for connection to an IBM mainframe host computer to allow host commands and responses to be transmitted using a FICON fiber optic interface.



The HSM’s FICON interface supports speeds of 32 Gbps, with the option of 8 Gbps or 16 Gbps, if available. If using a switched fabric, the connecting switch must have the connecting port type set to fabric port (F_port).

The FICON interface can be ordered with a choice of transceivers to support. One Transceiver MUST be ordered with each payShield 10K FICON Platform:

Part Number	Model Number	Transceiver type
971-000074-001	PS10-F-XCV-L	payShield 10K FICON Long Wave Transceiver
971-000075-001	PS10-F-XCV-S	payShield 10K FICON Short Wave Transceiver

The FICON interface must be specified when the payShield 10K is ordered. It is installed in the factory and cannot be added to an existing payShield 10K.

The only package and performance license available for the payShield 10K FICON is the following, which MUST be ordered together with the Hardware Platform, Transceiver, any optional licenses, and hardware accessories as required:

Part Number	Model Number	Transceiver type
971-701630-001	PS10-PRM-X	Premium package - 2500 cps

Support for this model is provided in base software version v1.3a and above.

2.3.1 Connections

payShield Manager provides a secure GUI interface with an authenticated, encrypted connection allowing a full remote or local management of the payShield 10K. Remote payShield Manager requires an Ethernet cable from the Management Port into your network. If you are not using DHCP, then you may need to use the console to set up the static IP address for payShield Manager.

(Refer to the *payShield Manager Quick Start Guide*. For the Console, refer to the *payShield 10K Console Guide*.)

2.3.2 Configuration Settings

The FICON interface can be configured using the CH console command or by using payShield Manager. Refer to the *payShield 10K Installation and User Guide*, section titled "*payShield 10K FICON Platform Variant*".

3 PIN Printing and Solicitation

This chapter describes how the HSM can:

- Use directly-attached impact printers to print PIN mailers which are used to notify cardholders of their PINs when new cards are issued, or
- Use directly-attached impact printers to print PIN Solicitation mailers which invite users to submit their desired PINs, and how to process the PIN submitted by the user.
- Use directly-attached laser printers for PIN mailers or PIN Solicitation mailers.
- Support PIN mailer or PIN Solicitation mailer printing in high-volume "print factories".

The focus here is on the role of the payShield 10K in the use of printed matter to engage with cardholders when setting up PINs. Some organizations are moving to using electronic means (in particular, the telephone system or the Internet) to set up PINs.

3.1 PIN Mailers

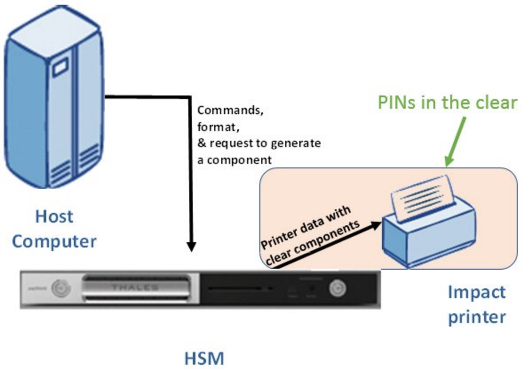
The principle of PIN mailer printing is straightforward - the card issuer's computer systems generate a PIN when a new card is issued or a PIN change is requested, print it in tamper-evident PIN mailer stationery through which the PIN cannot be viewed, and post it to the cardholder separately from the card itself.

The need for security, to prevent the PIN being disclosed within the card issuer's data center, means that an HSM should be used to prevent the PIN being available in the clear. Using the methodology described in this chapter, the PIN is in the clear only on the cable connecting the printer to the HSM (and, of course, inside the PIN mailer).

The stages in providing a secure method of printing PIN mailers are as follows:

- Set up the PIN Mailer form at the HSM
- Generate an Encrypted PIN
- Send PIN Mailer data for printing into the mailer at the HSM
- Verify PIN data cryptography

The following sections assume use of an impact printer directly attached to the HSM and the use of specialized multi- part stationery requiring the use of impact printers.



3.1.1 Setting up the PIN Mailer form

Forms for conveying and protecting PINs can be produced on a printer attached to a payShield 10K HSM. The documents are printed at the terminal in response to a command from the Host.

A form definition needs to be loaded into the payShield 10K so that it can format the output sent to the printer. The HSM can hold a single form definition at a time: this definition formats the printed output. Form definitions may be used for other purposes such as key component printing, and so a PIN Mailer form may need to be loaded before each PIN mailer print run. The form definition data is loaded onto the HSM as text strings. This data consists of:

- Formatting symbols to format the data (See *Appendix C* - Print Formatting Symbols)
- Constants (text strings)
- Variable print field markers, which will be populated with data when the PIN mailers are to be printed.

The maximum length of a form definition is 299 symbols and characters of constant data.

This form definition is loaded onto the HSM using payShield 10K Host commands. The definition data is stored in an HSM until power is removed, or until a reset is performed.

The form definition allows for PIN digits to be printed as words (using the ^V and ^W symbols). Where words other than the standard English words for numerals (“ZERO”, “ONE”, “TWO”, …) are to be used, these can also be loaded to the HSM by using a Host command. The required host commands are as follows. (See the *payShield 10K Core Host Commands reference manual* for full details.)

payShield 10K Host Command	
Command	**Command Description**
PA	Load Formatting Data to HSM - to load the first part of the set of symbols and constant characters.
PC	Load Additional Formatting Data to HSM - to load any continuation of the symbols and constant characters. (PC is a continuation of PA, and must be preceded by a PA command.)
LI	Load a PIN Text String (to allow PIN digits to be printed as words in languages other than English).

It is possible to format PIN mailers to be "one-up" (i.e. the PIN mailers are printed one after another on the PIN mailer stationery) or "two-up" (i.e. the PIN mailers are printed in side-by-side pairs).

Example of a two-up formatting string:

Desired result:

1	THOMAS M SMITH	JOHN R JONES
2	APT 4B 1782	427 WEST 9TH ST 3690
3	39 ELM DR	WAYNE PA 19132
4	MEDIA PA 19063	
5		
6	NEVER DISCLOSE YOUR PIN	NEVER DISCLOSE YOUR PIN
7		

The formatting string required to generate the above form is:

```
>L>003^0>033^4>L>003^1>023^P>033^5>053^Q>L>003^2>033^6>L>003^3>033^7>L>L>003NEVER DISCLOSE YOUR PIN>033 NEVER DISCLOSE YOUR PIN >L>F
```

In this example, the following print formatting symbols are used:

>L means carriage return + line feed (CR/LF)

>nnn means skip to output column nnn

^n means insert variable print field n - the actual value will be provided at print time

^P means insert clear PIN for mailer 1 of the side-by-side pair

^Q means insert clear PIN for mailer 2 of the side-by-side pair

>F means Form Feed (FF)

The full set of print formatting symbols is defined in “*Appendix C* - Print Formatting Symbols”.

Note that the PIN is not printed on the mailer, although the last 6 digits can be printed for the cardholder’s convenience.

Example of a one-up formatting string:

Desired result:

	THOMAS M SMITH	
2	APT 4B	1782
3	39 ELM DR	
4	MEDIA PA 19063	
5	YOUR FULL SERVICE BANK	
6		
1	JOHN R JONES	
2	427 WEST 9TH ST	3690
3	WAYNE PA 19132	
4		
5	YOUR FULL SERVICE BANK	
6		

The formatting string required to generate the above form is:

```
>L>013^0>L>013^1>041^P>L>013^2>L>013^3>L>013 YOUR FULL SERVICE BANK>L>F>
```

See the previous example to understand the meanings of the formatting codes.

3.1.2 Generate an Encrypted PIN

A PIN must be generated at the time that the card data is created in preparation for the issuing of the card. For security, this PIN must be encrypted whenever it is associated with the card data, such that no individual or application has access to both the card’s PAN and clear-text PIN.

The payShield 10K offers various host commands for generating a PIN and returning it, encrypted using the HSM’s LMK, to the Host application:

payShield 10K Host Command	
ID	**Command Description**
JA	Generate a random PIN. This command can prevent the generation of PINs considered to be weak.
EE	Derive a PIN from the PAN, using the IBM method.
GA	Derive a PIN from the PAN using the Diebold method.

The PIN is stored encrypted under the LMK, and is passed to the HSM in this encrypted form at print time. Therefore, the PIN is never available in plain text to the host computer.

3.1.3 Sending PIN Mailer Data for Printing at the HSM

When it is required to print the PIN Mailer, the following Host command is used to send the variable PIN mailer data to the HSM.

payShield 10K Host Command	
ID	**Command Description**
PE	Print PIN/PIN and Solicitation Data

A PE command is issued for each PIN mailer record. The parameters for this command include:

- Whether this record is for one-up printing, or whether it is record 1 or 2 for two-up printing
- The PAN
- The PIN, encrypted under the LMK

The values for the Print Fields identified in the form definition.

The HSM will decrypt the PIN, and then output the record in the defined format to the printer. The only time the PIN is in the clear is on the cable linking the HSM to the printer.

3.1.4 PINs Created Outside of the payShield 10K

The process described above assumes that the PIN was originally created using the payShield 10K and is encrypted using the payShield 10K LMK.

It may be, however, that the PIN derives from a different source, which cannot encrypt the PIN under the payShield 10K LMK. In this case, the PIN will be available as a PIN Block encrypted using a Zone PIN Key (ZPK) and the following payShield 10K Host command should be used to obtain the LMK-encrypted PIN which is required by the PE command:

payShield 10K Host Command	
ID	**Command Description**
JE	Translate PIN from ZPK to LMK

3.1.5 Verify PIN Data Cryptography

The Host application cannot "see" the data that the HSM has sent to the printer, and therefore a mechanism is required to enable the Host application to verify that the cryptographic processing for the PIN mailer printing was performed correctly.

After the HSM has processed the PE Host command, it sends 2 responses to the Host application:

payShield 10K Host Command	
ID	**Command Description**
PF	Before the PIN mailer is printed. This includes a cryptographic PIN check value calculated using the LMK.
PZ	After the PIN mailer is printed. This confirms that the physical printing activity completed successfully.

The Host application should confirm the check value returned in the PF response. It does this by sending the following Host command - preferably to a different HSM from the one that printed the mailer (but which is using the same LMK).

payShield 10K Host Command	
ID	**Command Description**
PG	Verify PIN/PIN and Solicitation Mailer Cryptography

This command contains the same PAN and encrypted PIN as was used in the PE Host command, and the check value contained in the PF response.

The PH response to the PG Host command will indicate whether there is a mismatch between the encrypted PIN and the check value.

3.2 PIN Solicitation

The objective of PIN solicitation is to send a mailer to the cardholder to invite them to:

- Select their desired PIN when a new card is about to be issued and return the desired PIN to the card issuer
- Select a new PIN for a previously issued card

The PIN may be returned by various methods depending on how the card issuer's applications have been designed, such as:

- Returning all or part of the PIN Solicitation mailer onto which the PIN has been entered
- Entry at an ATM
- Entry via the Internet

- Entry by telephone (Interactive Voice Recognition)

Once the PIN is returned, it is then associated with the card data in the card issuer's application.

The crucial factor in making this secure is that the mailer and returned data do not include the PAN. Instead, they include a reference number which is a cryptographic representation of the last 10 digits of the account number (excluding the account number check digit). This reference number is a 10-digit number, followed by two check digits.

Knowledge of the PIN and reference number is of no value to a fraudster. The relationship between PAN and reference number is determined only within the HSM, and the PAN and PIN are never presented together.

Because the reference number is the only link to the cardholder's PIN, there must be a means of validating the data that is manually entered. There is no way to validate the PIN except through dual entry procedures or through the visual comparison of the value entered and the value recorded on the mailer form.

The 12-digit reference number, unlike the PIN, can be validated by a Host program - see Section 3.2.1. The check digits can be validated during or after data entry.

PIN solicitation data is batch processed using Host commands. The number of records entered must be greater than or equal to the minimum batch size specified in the payShield 10K's security settings. Each batch consists of at least one logical record. Each logical record contains the 12-digit reference number in the returned solicitation mailer and the cardholder-selected PIN.

When the batch has been loaded to the payShield 10K internal memory, the HSM encrypts the PINs under LMK pair 02-03, and decrypts the reference numbers, yielding a value which contains the 10 right-most digits of the account number (excluding the check digit). The encrypted PIN and 10 digits of the account number are returned to the Host. The Host can match the account number digits and store the encrypted PIN for subsequent processing (for verification purposes or the creation of PIN offsets etc.).

3.2.1 Validation Algorithm

The algorithm for validating the two check digits of a reference number is as follows:

The first of the two check digits is calculated as:

MOD 10 [10 - MOD 10 (Y)]

where Y is the sum of the products obtained by multiplying the 3rd to the 10th digits of the reference number by the following weights:

Digit	Weight
3	9
4	7
5	8
6	6
7	7
8	9
9	6
10	8

Second Check Digit:

The second check digit is calculated as:

MOD 10 [10 - MOD 10 (Z)]

where Z is the sum of the following:

f(digit 1) + digit 2 + f(digit 3) + digit 4 +f(digit 5) + digit 6 + f(digit 7) + digit 8 + f(digit 9) + digit 10 + f(first check digit)

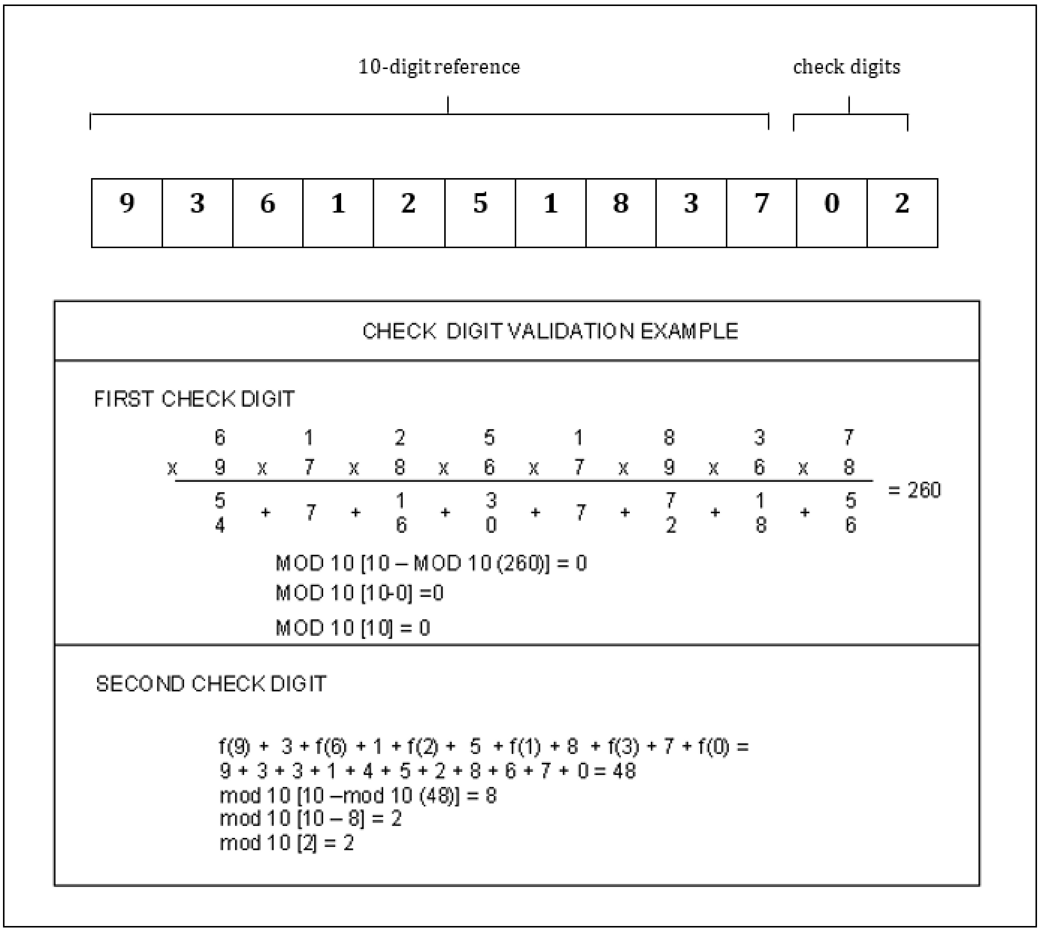
The value of f(digit n) is determined as follows:

Digit	f(digit n)
0	0
1	2
2	4
3	6
4	8
5	1
6	3
7	5
8	7
9	9

The MOD 10 (n) operation yields a value that is the remainder after dividing n by 10. This remainder is the same as the low-order digit on n.

Example:

The following example illustrates the validation of the reference number 936125183702, where 0 is the first check digit and 2 is the second check digit.



CHECK DIGIT VALIDATION EXAMPLE

FIRST CHECK DIGIT

```
     6       1       2       5       1       8       3       7
x    9   x   7   x   8   x   6   x   7   x   9   x   6   x   8
    ---------------------------------------------------------------
     5       7       1       3       7       7       1       5
     4   +       +   6   +   0   +       +   2   +   8   +   6    = 260
```

MOD 10 [10 – MOD 10 (260)] = 0
MOD 10 [10-0] =0
MOD 10 [10] = 0

SECOND CHECK DIGIT

f(9) + 3 + f(6) + 1 + f(2) + 5 + f(1) + 8 + f(3) + 7 + f(0) =
9 + 3 + 3 + 1 + 4 + 5 + 2 + 8 + 6 + 7 + 0 = 48
mod 10 [10 –mod 10 (48)] = 8
mod 10 [10 – 8] = 2
mod 10 [2] = 2

3.2.2 Stages in PIN Solicitation

The stages in providing a secure method of PIN Solicitation are as follows:

The stages in providing a secure method of PIN Solicitation are as follows:

- Set up PIN Solicitation Mailer form at the HSM
- Generate an Encrypted PIN (optional)
- Send PIN Solicitation Mailer data to the HSM for printing on a printer directly attached to the HSM.
- Verify the PIN data cryptography
- Processing the selected PIN returned by the cardholder.

As for PIN printing, this section assumes use of an impact printer directly attached to the HSM and the use of specialized multi-part stationery requiring the use of impact printers.

As for PIN printing, this section assumes use of an impact printer directly attached to the HSM and the use of specialized multi-part stationery requiring the use of impact printers.

3.2.2.1 Setting Up PIN Solicitation Mailer

This is the same process as described earlier for setting up the PIN Mailer form.

The available symbols to control printing (sent to the HSM using the PA and PC Host commands) include:

^R for Reference Number 1

^Q for Reference Number 2.

See *Appendix C - Print Formatting Symbols* for the full set of printing control symbols.

The mailer can be designed to print the Reference Number but not the PIN in situations where the cardholder is required to provide a PIN. Alternatively, both the reference Number and the PIN can be printed for situations where the cardholder is invited to change the default PIN that the issuer has provided and will use in the absence of a customer-specified PIN.

Note that the PAN is not printed on the mailer, although the last 6 digits can be printed for the card holder's convenience.

3.2.2.2 Generating an Encrypted PIN

An encrypted PIN is required only where the card issuer is providing a default PIN which the cardholder is invited to change. This is not necessary where the card issuer always requires the cardholder to select their own PIN.

Where an encrypted PIN is required, the same Host commands are used as described previously.

3.2.2.3 Sending PIN Solicitation Mailer Data for Printing

Where the mailer is to include both the PIN and the Reference Number, the same PE Host command as described earlier should be used.

Where only the Reference Number (without the PIN) is to be printed on the mailer, the OA Host command should be used.

payShield 10K Host Command	
ID	**Command Description**
OA	Print a PIN Solicitation Mailer

This provides the same information as the PE Host command, except that it does not provide an encrypted PIN.

The Reference Number is not part of the PE or OA command, but is calculated by the HSM from the last 10 digits of the PAN (excluding the check digit) prior to printing the mailer.

3.2.2.4 Verify PIN Solicitation Data Cryptography

The Host application cannot "see" the data that the HSM has sent to the printer, and therefore a mechanism is required to enable the Host application to verify that the cryptographic processing for the PIN Solicitation mailer printing was performed correctly.

After the HSM has processed the PE or OA Host command, it sends 2 responses to the Host application:

payShield 10K Responses to PE Command	
ID	**Command Description**
PF	Before the PIN mailer is printed. This includes a cryptographic PIN check value calculated using the LMK.
PZ	After the PIN mailer is printed. This confirms that the physical printing activity completed successfully.

payShield 10K Responses to OA Command	
ID	**Command Description**
OB	Before the PIN Solicitation mailer is printed. This includes a cryptographic Reference Number check value calculated using the LMK.
OZ	After the PIN Solicitation mailer is printed. This confirms that the physical printing activity completed successfully.

The Host application should confirm the check value returned in the PF or OB response. It does this by sending the following Host commands - preferably to a different HSM (but with the same LMK).

payShield 10K Host Command	
ID	**Command Description**
PG	Verify PIN/PIN and Solicitation Mailer Cryptography (for a PF response)
RC	Verify Solicitation Mailer Cryptography (for an OB response)

These commands contain the same PAN and encrypted PIN as was used in the PE or OA Host command, and the check value contained in the PF or OB response.

The PH or RD response to the PG or RC Host command will indicate whether there is a mismatch between the encrypted PIN and the check value.

3.2.2.5 Processing the Selected PIN Returned by the Cardholder

The cardholder will return their selected PIN together with the Reference Number. This data needs to be processed as follows prior to the card being issued.

3.2.2.5.1 Entry of PIN and Reference Number into a Host Application

The PIN and Reference Number need to be entered into a host application. The reference Number consists of 10 digits of data plus 2 check digits. The Host application should confirm that the Reference Number has been entered correctly by validating the check digits. This should be done in the host application, using the algorithm documented elsewhere in this manual.

3.2.2.5.2 Recovery of PAN and Encrypted PIN

The Reference Number and plain-text selected PIN must now be converted into a plain-text PAN and encrypted PIN, which can be passed to the card issuing system. This is accomplished using the HSM.

Up to 2,520 records (consisting of Reference Number + Selected PIN) can be loaded into the HSM at a time. This data goes into the HSM's user storage area, and overwrites any data that is already there: any user storage area data that is overwritten in this way will need to be reloaded after this process is completed.

These records are loaded using the following Host commands:

payShield 10K Host Command	
ID	**Command Description**
QC	Final Load of Solicitation Data to User Storage - one QC command is required for the batch, and can contain up to 1,260 records.
QA	Load Solicitation Data to User Storage - if more than 1,260 records is needed for the batch. Each QA command can contain 25 records. Any QA command(s) must precede the QC command.

The QB responses to the QA command are simple acknowledgements. The actual returned data for all the QA and QC commands is returned in one or two QD responses to the QC command. (If it is required to ensure that only a single QD response is returned, the batch size must not exceed 1,260 records.)

For each of these records, the QD response(s) provide(s) the 10 rightmost digits of the PAN in plain-text and the PIN encrypted under the LMK. This data will then be used by the host applications to generate the data for the card to be issued.

In order to prevent the plain text PIN in the QA/QC commands from being matched with the account number in the QD response(s), the order of the records in the QD response(s) is randomized. To provide additional security, the minimum batch size (as defined in the CS Console command or in the payShield Manager Configuration / Security Settings / Initial dialogue box) should be made as large as possible.

3.3 Printer Support

3.3.1 Impact Printers

The payShield 10K allows direct attachment of printers with the following interfaces:

- Serial
- Parallel - D25 connectors
- Parallel - Centronics
- USB

These printers are connected to one of the USB ports on the payShield 10K. Serial and parallel printers are attached using the appropriate serial or parallel adapter cable from Thales: these adapter cables are intelligent and include microprocessors, the drivers for which are included with the HSM software. (USB adapters from other sources should not be used as these may not work with the drivers installed on the HSM.)

Cable length limitations conform to appropriate standards. (Note that for serial printers, the maximum cable lengths are dependent on the baud rate of the interface and the physical characteristics of the type of cable used.)

Printers must support the ASCII character set.

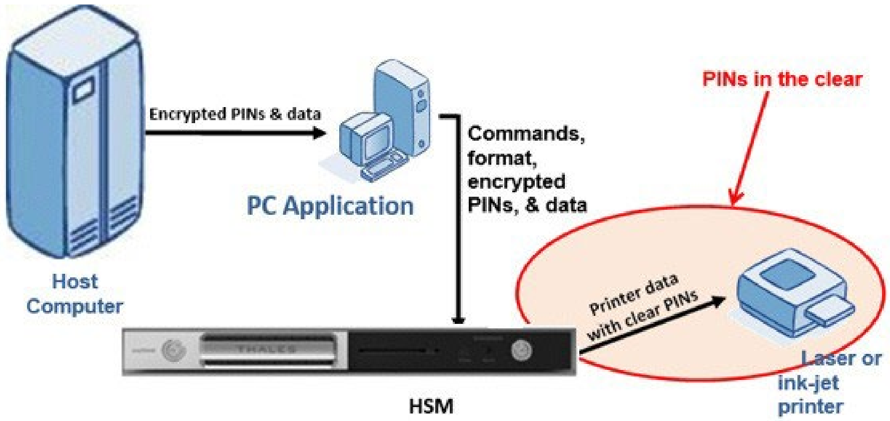
3.3.2 Directly-attached Laser or Ink Jet

There is a gradual move away from character-based impact printers to laser (or ink-jet) printers for the production of PIN or PIN Solicitation mailers. These offer benefits in terms of performance, quality of output, and use of a wider range of stationery.

A PC application is required to manage and drive the laser printer, providing very accurate positioning of printing. Additional fonts are required, and the laser printer may require additional memory to accommodate these.

As with the use of directly-attached impact printers, this arrangement offers a high degree of security because the clear PINs are only present on the printer cable and in the printer.

For convenience, this document will refer just to laser printers. However, ink-jet printers can also be used.



3.3.2.1 Stationery for Laser-Printed Mailers

Mailer printing using laser printers cannot use the traditional multi-part stationery, which requires an impact printer to cause the secret data to be printed inside the envelope.

Instead specialized laser printer stationery is used, with a plastic panel where the PIN (or secret data) will be printed. The panel has a printed mask on it such that the PIN printed on the panel cannot be read unless the panel is removed and held up to the light or the mask is rubbed off: if these actions are performed, it is evident that the panel has been tampered with.

Special fonts, provided by the stationery manufacturer, are required to allow the PIN to be printed over the mask. Accurate positioning is essential to align the mask and printed PIN.

Examples of stationery made for this purpose are:

- Bastione PIN-TAB
- Hyadalam Laser pin
- Page International ICS V3

3.3.2.2 Security Concerns

The use of laser printers for printing of mailers is growing in popularity because of the benefits in terms of performance, reduced noise, and quality of output. However, some concern has been expressed about the security of this approach.

Anyone contemplating implementing laser printing technology should research the concerns and implement any additional security measures that they feel are appropriate.

3.3.2.3 Modifications for this Solution

The principles described previously relating to the use of character impact printers apply equally to the use of laser printers, but there are some additional considerations.

3.3.2.3.1 Fonts

The laser printer must be capable of allowing the specialized fonts needed for the stationery to be loaded. This may require additional memory to be installed in the printer.

3.3.2.3.2 PC Application

- A PC application is necessary to enable this approach to work. This must be able to:
- Communicate with the payShield 10K - sending the appropriate host commands and data (including encrypted keys) and processing responses.
- Design output forms.

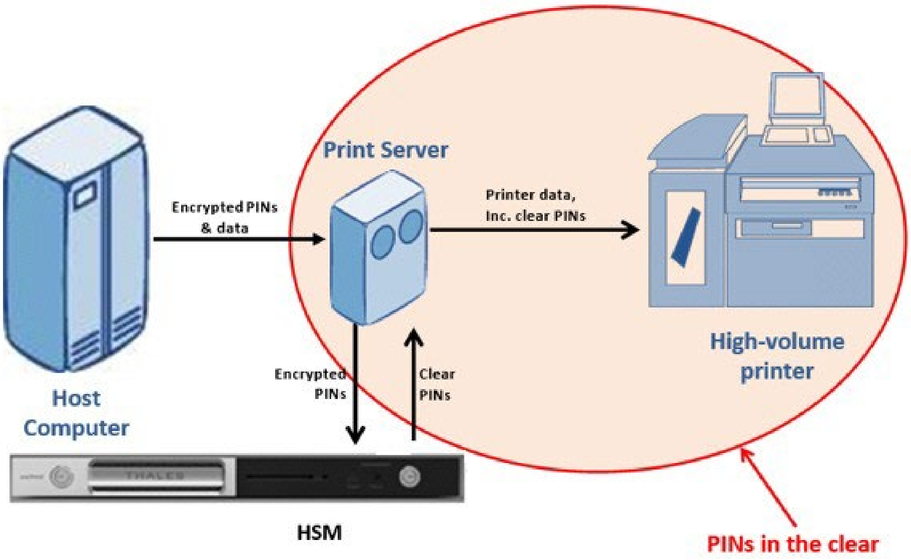
Pass printer control data to the HSM, using the 'PA' and 'PC' host commands as described previously. Particularly important is the ability to accurately position the (notional) print head - the standard column/row positioning provided by the standard 'PA' / 'PC' formatting commands is not precise enough, and the laser printer chosen must allow print head positioning by user-defined data. In the 'PA' / 'PC' command the control string required by the printer to do this would be passed by the application to the HSM using the '|' Symbol ('6A' in EBCDIC, '7C' in ASCII). See *Appendix D - Example laser printer formatting control codes* for an example.

3.3.3 High-volume "Print Factories"

3.3.3.1 Overview

Where high-volume PIN Mailer printing is required, greater throughput can be achieved by using a print server driving one or more high-volume laser printers, possibly providing other functions such as collating, stapling, stuffing into envelopes, and so on.

In this case, the printer is not directly connected to the HSM but is connected to the print server. The print server can make use of one or more HSMs to decrypt the PINs provided by the host application. It then incorporates the clear PINs returned by the HSM in the data sent to the printer.



Commercial products implementing this architecture are available from various vendors such as Red Titan and Phalanx e-Solutions.

3.3.3.2 Security Considerations

Whereas in the previously discussed cases of directly attached printers the clear PINs are only present in the printer and the cable connecting the printer to the HSM, in the print factory scenario the PINs are also in the clear in the print server and on the network connecting the print server to the HSM. Furthermore, the print server is also connected to the host system and potentially to the institution's network.

There is therefore a wider range of attack opportunities than with directly attached printers. Organizations implementing this printing architecture should satisfy themselves that they have implemented suitable security and system design measures - which might include:

- Physical access controls
- Network isolation
- Software design preventing PINs returned by the HSM from being accessed more than once
- Use of different print engines used for sensitive and non-sensitive data
- Automatic memory flushing.

Solutions of this type will use similar stationery to that used with directly attached laser or ink-jet printers, and so the Caution described in that section also applies here.

However, the Phalanx PIN Production System using PIN-TAB mailers meets the UK Cards Association Standard 71, Security level 4 (Formerly APACS PIN standard).

3.3.3.3 Relevant payShield 10K Commands

The following payShield 10K Host commands allow:

- a PIN encrypted under the LMK to be decrypted, and
- a PIN encrypted under a ZPK to be translated to encryption under the LMK (so that it can then be decrypted).

payShield 10K Host Command	
ID	**Command Description**
NG	Decrypt a PIN encrypted with LMK.
JE	Translate a PIN from ZPK to LMK Encryption

The response to the NG command includes both the clear PIN for PIN Mailer printing and a Reference Number that can be used for PIN Solicitation Mailer printing.

4 Transaction Key Schemes

The transaction key scheme is a technique in which data-encrypting keys change with each transaction in a manner that cannot be followed by a third party. This is typically of use in Electronic Fund Transfer at Point of Sale (EFTPOS) systems where fund transfer requests and responses are exchanged between a retailer (EFTPOS terminal) and an acquirer, and then, optionally, between the acquirer and the card issuer.

The payShield 10K HSM supports three techniques:

- Racal (or APACS) Transaction Key Scheme (RTKS).
- Derived Unique Key Per Transaction (DUKPT).
- Australian (AS 2805) Transaction Key Scheme.

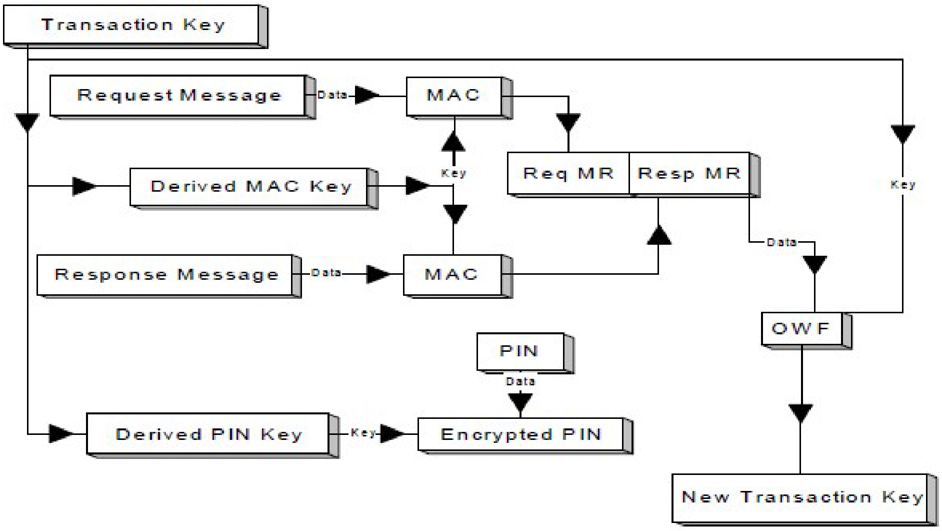
The payShield 10K also provides functionality to enable it to participate in systems meeting the requirements of GBIC/ZKA: this is discussed later in this chapter.

4.1 Racal Transaction Key Scheme (RTKS)

The Racal Transaction Key Scheme (RTKS) is a key management technique that is closely coupled with message authentication. The functions provided by an HSM include key management in addition to MAC generation and verification.

In the Racal Transaction Key Scheme the TPK and the TAK are updated after each EFT transaction using an algorithm that depends on the current key and the details of the transaction (which are known to both communicating parties but do not form part of the transmitted message, and so would not be known to a third party).

This affords a very high degree of protection for the cryptographic keys. Even if a third party were able to discover the value of the cryptographic key in use at a particular time, this would not facilitate discovery of the keys used on previous transactions (i.e., the scheme has good break-backward protection). Also, if some card data were not transmitted, the third party would not be able to discover the new value of the keys for the subsequent transaction (break-forward protection).



The key update algorithm used in the Racal Transaction Key Scheme is based on a One- Way Function (OWF) involving the current key value and the Message Authentication Code (MAC) residues from the request and response messages from the transaction. The use of the MAC residues is important, as they are not part of the transmitted data.

In this scheme, the MACs are calculated using a key derived from the transaction key, and not the transaction key itself. This also applies to the PIN encrypting key.

For more details of the Racal Transaction Key Scheme, see Racal-Transcom Publication RRL4 Secure Key Management for Pin Encryption and Message Authentication.

4.1.1 RTKS Functions

The following functions are all for use at an acquirer site:

- Transaction request with PIN (T/AQ key). Used to receive a cardholder request message from a terminal with a PIN encrypted under the T/AQ key.
- Transaction request without PIN. Used to receive a cardholder request message from a terminal with no PIN.
- Transaction request with PIN (T/CI key). Used to receive the request from the terminal when the PIN key cannot be determined by the acquirer.
- KEYVAL translation. Used to pass KEYVAL to the card issuer (required to derive the PIN key) when the PIN key cannot be determined by the acquirer.
- Administration request. Used to receive an administration request message (such as a reconciliation request).
- Transaction response originating at the card issuer. Used when authorization is generated at the card issuer.
- Transaction response originating at the acquirer. Used when authorization is generated by the acquirer.
- Verify confirmation message from terminal. Used to verify the MAC on a confirmation message from the terminal.

The host commands RI, RK, RM, RO, RQ, RS and RU are only available when Racal Transaction Key Scheme is selected in the payShield 10K security settings.

The existing Racal Transaction key commands have been modified to support longer messages. The new commands are backward compatible with existing systems.

The details of the modifications are as follows:

Old style:	Pointer (not all functions) Message Length	2 H 2 H	
	Message Text	n A	
New style:	Pointer (if required)	2 H	
	Message Length	2 H	
	Message Text	n A	
	Delimiter	1 C	Optional, only if original length is 0
	Extended Message Pointer(s)	4 H	Optional, only if original length is 0 and function requires one or two pointers
	Extended Message Length	4 H	Optional, only if original length is 0
	Extended Message	n A	Optional, only if above field is non zero

To use the extended message length option, the calling application has to set the Message Length field to zero, whereupon the Message Text field will be of zero length, i.e. not present. The zero Message Length enables an HSM to check for the optional Delimiter, any Extended Message Pointer(s), and the Extended Message Length field which defines the length of the Extended Message.

Some of the functions do not include a pointer to items included in the message, while other functions include either one or two pointers. If a function does include one or two pointers, one or two Extended Message Pointers are included after the Delimiter as appropriate. The original pointer(s) in the function are ignored when extended messages are used, however the 2 hex digit placeholder(s) for the original pointer(s) must still be supplied.

While the extended commands allow for message sizes up to 65537 characters long (hex FFFF), in practice the limit is imposed by the maximum size of the HSM input buffer. The HSM has a much larger input buffer (32K) per connection although the interface option in use may impose limits, which are smaller than this. The HSM will check that the message lengths (and the pointers) are within sensible limits for an HSM platform executing the function.

Users may, if they wish, use the Extended Message Length scheme for small messages (i.e. less than 160 bytes).

4.2 Simultaneous use of both Racal and Australian Transaction Key Schemes

The original design of the payShield assumed that only one (or none) of these schemes would be used on any one HSM. The desired TKS was selected using a security setting, e.g., in the CS Console command:

```
Transaction Key Scheme: Racal, Australian or None [R/A/N]:
```

Because the two TKSs were mutually exclusive, the same Host command code was used to identify two different commands, one applicable to the Racal TKS and the other applicable to the Australian TKS. The command codes affected by this are:

Code	Racal TKS	Australian TKS
RI	Transaction Request With a PIN (T/AQ Key)	Verify a Transaction Request, with PIN, when CD Field not Available
RK	Transaction Request Without a PIN	Generate Transaction Response, with Auth Para Generated by Acquirer
RM	Administration Request Message	Generate Transaction Response with Auth Para Generated by Card Issuer
RO	Transaction Response with Auth Para from Card Issuer	Translate a PIN from PEK to ZPK Encryption
RQ	Generate Auth Para and Transaction Response	Verify a Transaction Completion Confirmation
RS	Confirmation	Generate a Transaction Completion Response
RU	Transaction Request With a PIN (T/CI Key)	Generate Auth Para at the Card Issuer
RW	Translate KEYVAL	Generate an Initial Terminal Key

The functionality delivered when any of these Host commands was used was dependent on the security setting mentioned above.

More recently, the need has arisen for both TKSs to be used on the same payShield 10K, and therefore to have access to both versions of these 'R*' commands. In order to allow users to have access to both versions of the commands, the 'R*' commands have been "aliased" as 'H*' commands - e.g., for each of the two 'RI' commands there is an 'HI' command which is identical to the 'RI' command except for the Command code ('HI' instead of 'RI') and Response code ('HJ' instead of 'RJ').

Where the user wishes to use the version of the command that is appropriate to the TKS selected in the security setting they should use the R* variant of the command.

For example:

- if the TKS is set to "Racal" and the user wants the "Transaction Request Without a PIN" functionality then they would use the 'RK' command.
- if the TKS is set to "Australian" and the user wants the "Generate Transaction Response, with Auth Para Generated by Acquirer" functionality then they would use the 'RK' command.

  This is exactly as for earlier versions of payShield 10K software, and so provides backwards compatibility for existing applications.
  On the other hand, if the user wishes to use the version of the command that is not appropriate to the selected TKS, they should use the 'H*' variant of the command.
  For example:

- if the TKS is set to "Australian" and the user wants the "Transaction Request Without a PIN" functionality then they would use the 'HK' command.

- if the TKS is set to "Racal" and the user wants the "Generate Transaction Response, with Auth Para Generated by Acquirer" functionality then they would use the 'HK' command.

  In summary:

	If Transaction Key Scheme = Racal	If Transaction Key Scheme = Australian
You want to process Racal Transaction Key commands	Use the 'R*' variant of the command	Use the 'H*' variant of the command
You want to process Australian Transaction Key commands	Use the 'H*' variant of the command	Use the 'R*' variant of the command

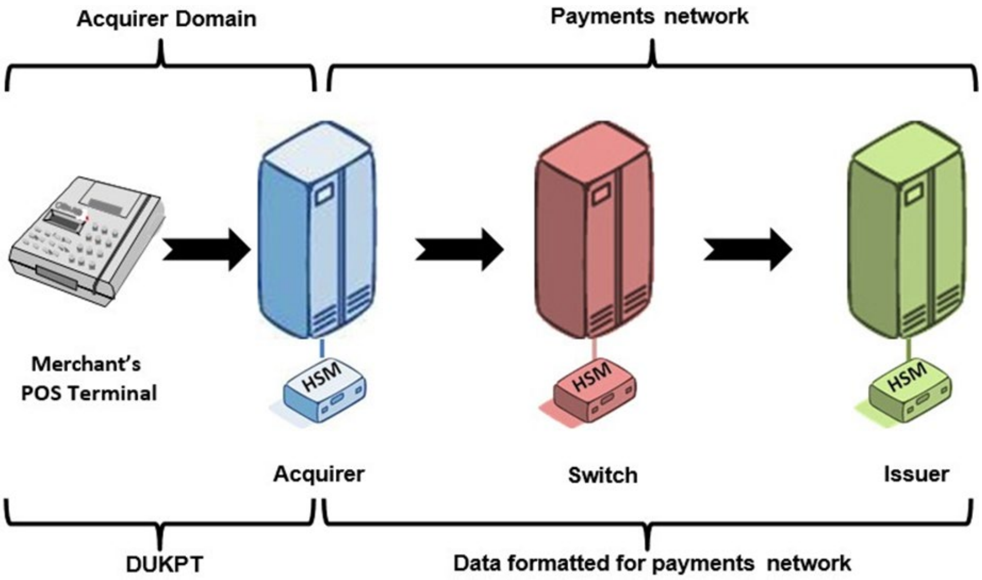
4.3 Derived Unique Key Per Transaction (DUKPT)

4.3.1 General Description

4.3.1.1 Introduction

DUKPT (Derived Unique Key Per Transaction) is a scheme to manage 3DES (also referred to a Triple-DES or TDES) and AES encryption keys in a card payment environment. It was originally used with POS (Point of Sale) terminals in North America, but has been adopted in other regions and for other applications - and so DUKPT is becoming increasingly important in the payments space. The 3DES DUKPT scheme was originally specified in ANSI X9.24-1:2009 and the AES DUKPT scheme together with the 3DES scheme is specified in ANSI X9.24-3:2017.

DUKPT is used for encrypting PIN blocks, encrypting other data, and message authentication (MACing). DUKPT is typically used between merchant's POS terminal and their Acquirer. The Acquirer may need to repackage the transaction data for passage through the payments network using Master/Session key management before passing it to a payments switch.



The strength of DUKPT lies in the fact that a new, unique key is generated for each transaction, such that if a transaction key is compromised this cannot be used to attack transactions at that or any other terminal. In addition, key management in the DUKPT environment is simplified by having a single master key which can manage an entire estate of terminals.

This technique involves the use of a non-secret "Key Serial Number" and a secret "Base Derivation Key", or BDK. On each transaction, the POS terminal uses a unique key based on a previous key and the key serial number, which contains a transaction counter. It encrypts the PIN with this key, then supplies both the encrypted PIN and the key serial number to the acquirer. In an HSM, the key generated by the POS terminal is derived dynamically and independently of the POS terminal, using the original BDK together with the key serial number supplied.

The Initial Key (IKEY) is securely installed into each POS terminal before use and is used to derive the first key used in the transaction. This is then discarded and as noted above, the unique key for each transaction is then derived from the previous key used. The IKEY is also derived from the BDK and the Key Serial Number.

As well as providing PIN encryption, the standard allows keys to be derived for data encryption and also for MACing.

The same BDK can be used by thousands of POS terminals because each POS terminal has a unique serial number. Therefore, each POS terminal produces a unique key for every transaction and a successful cryptographic attack on one POS terminal will have no effect on any other. The acquirer only has to manage a relatively small number of BDKs, and the algorithm to derive a given transaction key is designed in such a way as to require very little overhead in an HSM.

The Host has the responsibility for maintaining the BDKs. For each transaction, the Host verifies that the serial number supplied by the POS terminal is valid and uses the appropriate encrypted BDK identified by the left-most portion of the serial number. The Host also controls BDK generation.

The DUKPT standard supports use of both a 3DES and an AES DUKPT, but AES DUKPT has a different derivation function when compared to 3DES DUKPT. The main changes included in AES DUKPT when compared to 3DES DUKPT are as follows:

1. A different key derivation function is used with AES DUKPT.
2. In AES DUKPT, all keys are derived using the same key derivation function – 3DES DUKPT uses 4 different functions for key derivation.
3. The Initial Key ID (IKID) is 64 bits in AES DUKPT – whereas in 3DES it is 59 bits.
4. The Transaction Counter (TC) is 32 bits rather than 21 bits, and up to 16 one bits are allowed instead of 10.
5. The Key Serial Number (KSN) is 96 bits rather than 80 bits. The KSN is made up of the Initial Key ID plus the Transaction Counter.
6. The algorithm supports transaction-originating devices with 21 or 32 key registers.
7. The algorithm includes support for loading a new initial key under an existing key.

4.3.1.2 DUKPT Keys

4.3.1.2.1 Introduction

Three levels of keys are employed in DUKPT:

- Base Derivation Key (BDK) - a 3DES or AES master key owned by the acquirer. The BDK is used for a large number of terminals - perhaps all the terminals that the provider ships, or to a model of terminals, or to a serial number range.
- Initial Key (IKEY, IK, or IPEK) - a 3DES or AES key that is unique to a terminal. The IKEY is used to initiate the sequence of transaction keys and is then discarded by the terminal.
- Transaction key - generated within the terminal. Keys for PIN encryption, data encryption, and MACing are derived from the transaction key. Each transaction is provided with a unique key to protect its data. When the encrypted data is received by the Acquirer, the Acquirer will derive the same transaction key using the same process that the terminal used to derive the encryption key.

It can be seen from this process that there is no requirement for the Acquirer and terminal to exchange keys - except in the unlikely event of a terminal generating a million transaction keys and therefore requiring a new IKEY (for 3DES DUKPT) or an Update Key (for AES DUKPT).

4.3.1.2.2 The Base Derivation Key (BDK)

The BDK is a double-length or triple-length 3DES or AES key, which is usually generated and owned by the Acquirer, and is used for a large number of terminals. (If the BDK is owned by an organization other than the Acquirer, it will need to be distributed to the Acquirer to enable them to process transactions. It will also need to be distributed to any other organization involved in generating IKEYs.) Multiple BDKs will generally be held to allow for different terminal families or groups. BDK distribution can be done:

- electronically, with the BDK protected by a Zone Master Key (ZMK), or
- in the form of printed components, with separate component holders coming together to enable the BDK to be formed from their components.

A stored BDK must be protected by encryption using an appropriate Key Encryption Key (KEK) whenever it exists outside of an HSM.

The DUKPT standard defines two different methods for deriving data authentication and encryption keys:

- The bidirectional method uses a single key to protect terminal-to-host data and host-to-terminal data.
- The unidirectional method uses two keys: one key to protect terminal-to-host data, and another key to protect host-to-terminal data.

4.3.1.2.3 The Initial Key (IKEY or IK)

The IKEY (originally referred to as an IPEK - Initial PIN Encryption Key) is unique to each terminal. The IKEY is calculated from:

- the BDK
- the Key Serial Number (KSN) which is unique to the terminal - see below.

Once created, the IKEY is installed into the terminal. (The IKEY is also recreated transiently by the Acquirer when processing transactions from the terminal to derive the same transaction key that the terminal used.)

4.3.1.2.4 Transaction Keys

When the IKEY is installed in the terminal, it calculates up to 21 “Future Keys” (for 3DES DUKPT) or 32 “Future Keys” for AES DUKPT. These Transaction Keys are the keys that will be used in the encryption of future transactions. The calculation of these keys involves the value of the transaction counter, which increments for each transaction.

When the initial batch of Future Keys has been derived, the IKEY is no longer required, and is deleted by the terminal.

When a transaction is being processed, the next available Transaction Key is used. The keys used for PIN block encryption, MACing, and data encryption are derived from this transaction key.

The Key Serial Number (KSN) is also modified by incrementing the Transaction Counter.

The DUKPT terminal sends its encrypted data and the KSN, together with other transaction data, to the Acquirer.

As each Transaction Key is used, it is deleted by the terminal and replaced by a new Future Transaction Key. This means that even if the security of the terminal is compromised in any way and its keys extracted, they cannot be used to attack a previous transaction from this or any other terminal because the key for that transaction has already been deleted and each terminal generates different keys.

4.3.1.3 Key Serial Number (KSN)

4.3.1.3.1 Legacy 3DES DUKPT KSN

For 3DES DUKPT, the KSN has a length of 80 bits. The structure of the KSN is not fully defined in the standards so it is up to the system implementor to define the lengths of the fields with the following constraints:

- The left most 59 bits are the Initial Key ID (Initial Key Serial Number)
    - The right most 43 bits of these must be unique for each device.
- The right most 21 bits are used as a counter that is increased for each derived key.

Element	Contents	
Key Serial Number (KSN) - 80 bits	Initial Key ID (IKID) - 59 bits	Base Derivation Key ID (part) - 16 bits
		Base Derivation Key & Device ID - 43 bits
	Key Counter - 21 bits	

The Initial Key ID does not change during the life of the terminal (unless a new IKEY is loaded).

Note that in earlier versions of the standards:

- Initial Key ID was also known as the Initial Key Serial Number
- The Key Counter was also referred to as the Encryption Counter

4.3.1.3.2 AES DUKPT KSN

For AES DUKPT, the Key Serial Number (KSN) has a length of 96 bits and follows a specific structure outlined in the ANSI X9.24-3:2017 standard:

Element	Contents	
Key Serial Number (KSN) - 96 bits	Initial Key ID (IKID) - 64 bits	Base Derivation Key ID (BDK ID) - 32 bits
		Derivation ID (DID) - 32 bits
	Transaction Counter (TC) - 32 bits	

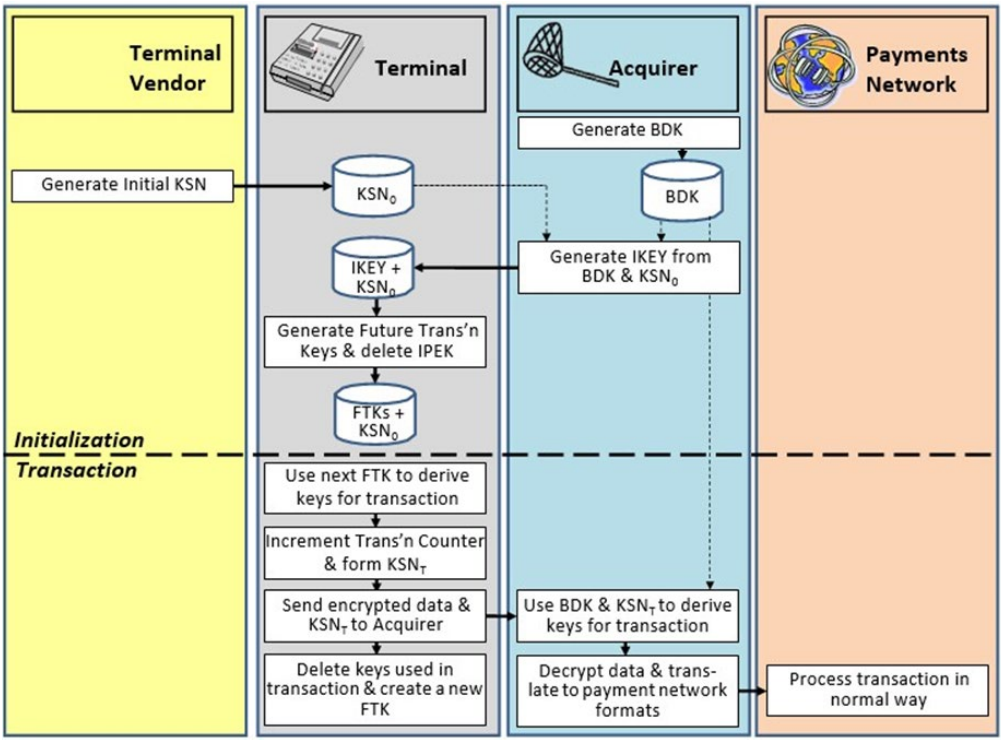
4.3.1.4 Processing by the Acquirer

When the encrypted data is sent by the terminal to the Acquirer, the KSN (including the transaction counter) is also sent. The Acquirer can reconstruct the Transaction Key used by the terminal from the KSN and the appropriate BDK (as identified in the KSN's Key Set Identifier).

The Acquirer needs to re-package the data received from the terminal into the standard formats used by the payments network. This will include actions such as:

- Translating a DUKPT PIN Block encrypted with the DUKPT transaction key into one of the standard PIN Block formats that the payments network uses encrypted with a Zone PIN Key (ZPK).
- Verifying and translating MACs.

4.3.1.5 Summary of DUKPT Operations


Terminal Vendor
Terminal
Acquirer
Payments Network
Generate BDK
Generate Initial KSN
KSN$_0$
BDK
IKEY + KSN$_0$
Generate IKEY from BDK & KSN$_0$
Generate Future Trans’n Keys & delete IPEK
FTKs + KSN$_0$
Initialization
Transaction
Use next FTK to derive keys for transaction
Increment Trans’n Counter & form KSN$_T$
Send encrypted data & KSN$_T$ to Acquirer
Use BDK & KSN$_T$ to derive keys for transaction
Delete keys used in transaction & create a new FTK
Decrypt data & translate to payment network formats
Process transaction in normal way

Notes:

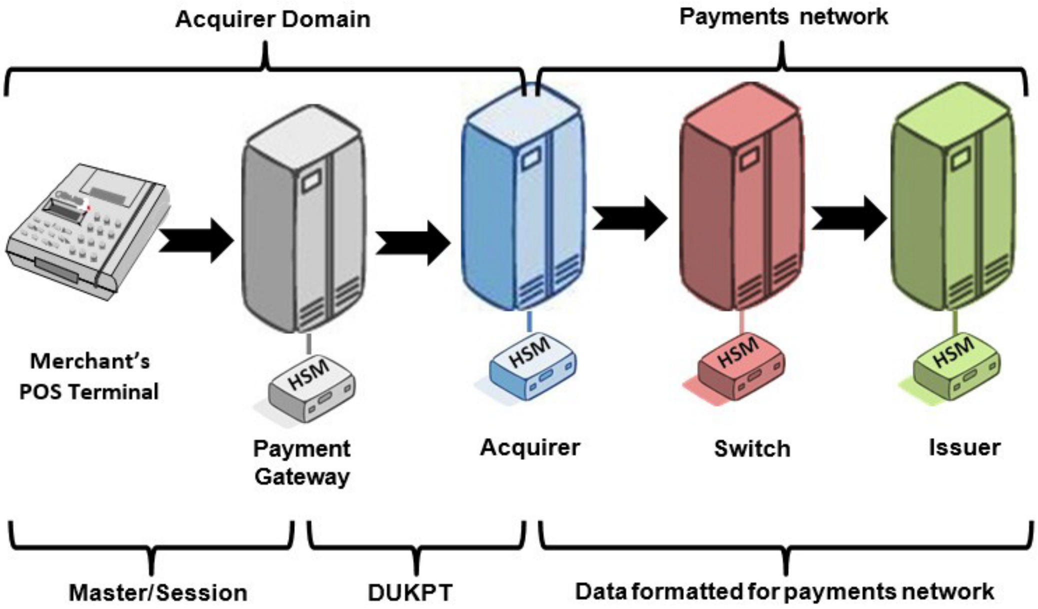
- KSN_0 = Initial Key Serial Number (with Transaction Counter = 0). May be modified by Acquirer before generating IKEY.
- KSN_T = Key Serial Number for the transaction, with Transaction Counter incremented)
- The BDKs held by the acquirer will be protected using an HSM.
- The Acquirer operations shown here will involve the use of an HSM for various cryptographic functions.

4.3.1.6 Variations on DUKPT

4.3.1.6.1 Data Translation to DUKPT

In the diagram above, the data flow is from the POS terminal to the Acquirer and then into the standard payments network. As mentioned already, this will involve translating DUKPT data to formats used by the payments network.

However, consider the following environment:



Here we have an additional link in the chain - the Payment Gateway. This is an intermediary between the merchant and the Acquirer. The Payment Gateway provides various services to the Merchant and might be needed because the merchant is too small to have a direct relationship with an Acquirer. For the present discussion we shall assume that the Payment Gateway is required because the Acquirer provides support for only DUKPT terminals, but the merchant does not support DUKPT.

In such an environment, the Payment Gateway will implement master/session key management with the merchant POS terminals, using the same data and PIN Block formats as the payments network. However, because the Acquirer supports only DUKPT key management and data formats on the terminal side, the Payment Gateway must have the capability to translate data to DUKPT formats.

4.3.1.6.2 Point-to-Point Encryption (P2PE)

Traditionally in card payment processing, only the PIN is encrypted (in the form of a PIN Block) - other data such as the PAN has generally not been encrypted: PCI DSS covers the Acquirer domain and requires encryption of cardholder data such as the PAN when the data passes over a public network, but not where the data is on a secure private network.

Encrypting cardholder data at the merchant, at the Acquirer, and on the network connecting them - whether this is a private network or not - takes the data out of scope of PCI DSS compliance and audit. This is very attractive to merchants and Acquirers because of the cost of PCI DSS compliance and audits, and so many customers are keen to encrypt all cardholder data in the Acquirer domain - this is referred to as P2PE.

A wide variety of P2PE implementations are available, with proprietary designs by their vendors. These implementations may involve a Payment Gateway.

DUKPT is often used as the key management method for a P2PE solution because it provides a standardized way of exchanging encrypted data (not just PINs) and high levels of security.

4.3.1.6.3 Mobile Acceptance (or Mobile Point of Sale - mPOS)

Traditional card payment terminals are beyond the reach of small merchants or mobile merchants (such as market stall operators or plumbers) because of the cost of the terminals and the fixed-line communications infrastructure needed to support them.

Mobile Acceptance is a technology that addresses this issue and has been adopted widely.

With Mobile Acceptance, the terminal consists of low-cost, secure card readers and PIN entry devices attached to intelligent mobile communications devices (such as tablets or standard smartphones). This brings the technology within financial reach of any merchant or service provider and removes the dependence on a fixed communications infrastructure.

P2PE must be employed between the mobile terminal and the Acquirer (or a Payment Gateway providing a mobile transaction service to merchants and passing the transactions to an Acquirer).

Again, DUKPT is often employed as the key management scheme in this environment.

4.3.1.6.4 Italian DUKPT Standard

The DUKPT standard is used in Italy is identical to ANSI X9.24-3:2017 when using bi-directional 3DES keys apart from the derivation of the initial key. For this scheme, the initial key is derived using a proprietary method defined in the Italian Standard Key Derivation Method SPE-DEF-041-112.

4.3.2 The Role of the payShield 10K in DUKPT

4.3.2.1 Introduction

The payShield 10K provides functionality supporting the ANSI X9.24-3:2017 standard and this is described in this section. The key types, the format of the Key Serial Number and the host commands supporting this functionality are covered in this section.

4.3.2.2 Thales Key Types Supporting DUKPT

The host commands, in the sections that follow, use the following Thales key types relating to DUKPT:

Thales Key Type	Key Type Code (Variant LMKs)	Key Usage (Key Block LMKs)	Description
BDK Type 1 ("BDK-1")	009	B0	Used to secure transactions sent between a terminal and an acquirer. Three types of bidirectional keys may be derived from a BDK-1: • PIN encryption keys • Data authentication (MAC) keys • Data encryption keys The same key is used in both terminal-to-acquirer messages and acquirer-to-terminal messages. Both 3DES & AES are supported. This type of BDK meets the requirements of ANSI X9.24-3:2017.
BDK Type 2 ("BDK-2")	609	41	This type of BDK is used to secure transactions sent between a terminal and an acquirer. Five types of unidirectional keys may be derived from a BDK-2: • PIN encryption keys (for terminal-to-acquirer messages) • Data authentication (MAC) keys (for terminal-to-acquirer messages) • Data authentication (MAC) keys (for acquirer-to-terminal messages) • Data encryption keys (for terminal-to-acquirer messages) • Data encryption keys (for acquirer-to-terminal messages) Different keys are used for terminal-to-acquirer messages and acquirer-to-terminal messages. Both 3DES & AES are supported. This type of BDK meets the requirements of ANSI X9.24-3:2017

Thales Key Type	Key Type Code (Variant LMKs)	Key Usage (Key Block LMKs)	Description
BDK Type 3 ("BDK-3")	809	42	This type of BDK is used to secure portions of the data portions (not PINs or MACs) of transactions sent between a terminal and an acquirer. One type of bidirectional key may be derived from a BDK-3: • Data encryption key Note: The data encryption key is derived from the BDK-3 using the "PIN verification" variant but can be used only for data encryption / decryption purposes. The same key is used in both terminal-to-acquirer messages and acquirer-to-terminal messages. This type of BDK does not meet the requirements of ANSI X9.24-3:2017 but is used in some proprietary terminal solutions. Only 3DES is supported. Note: this method cannot be used to derive a PIN encryption key, or a data authentication (MAC) key.
BDK Type 4 ("BDK-4")	909	43	This type of BDK is used to secure transactions between a Payment Service Provider (PSP) emulating a DUKPT terminal, and an upstream acquirer. Five types of unidirectional keys may be derived from a BDK-4: • PIN encryption keys (for PSP-to-acquirer messages) • Data authentication (MAC) keys (for PSP-to-acquirer messages) • Data authentication (MAC) keys (for acquirer-to-PSP messages) • Data encryption keys (for PSP-to-acquirer messages) • Data encryption keys (for acquirer-to-PSP messages) Different keys are used for PSP-to-acquirer messages and acquirer-to-PSP messages. Both 3DES & AES are supported. This type of BDK meets the requirements of ANSI X9.24-3:2017.
BDK Type 5 ("BDK-5")	-	44	This type of BDK is identical to a BDK-1, apart from the derivation of the IKEY, which uses a proprietary method implementing the Italian Standard Key Derivation Method SPE-DEF-041-112. Only 3DES is supported.
IKEY	302	B1	Initial key injected into POS terminal from which future transaction keys are generated.

Thales Key Type	Key Type Code (Variant LMKs)	Key Usage (Key Block LMKs)	Description
			Both 3DES & AES are supported.

Selection of the correct BDK type to use:

- When acquiring transactions (i.e., receiving 'request' transaction data from a terminal, and (optionally) transmitting 'response' transaction data to the terminal):
  - Use a BDK of type BDK-1 when *bidirectional* terminal-to-acquirer keys are required.
  - Use a BDK of type BDK-2 when *unidirectional* terminal-to-acquirer keys are required.
- For Payment Service Providers (PSPs) that emulate the function of a terminal (i.e., by transmitting 'request' transaction data to an acquirer, and (optionally) receiving 'response' transaction data from an acquirer):
  - When *bidirectional* keys are required, the PSP and the acquirer should use the same BDK, of type BDK-1.
  - When *unidirectional* keys are required, the PSP should use a BDK of type BDK-4, while the acquirer should use the same BDK, but of type BDK-2.
- (Note that a BDK-3 uses the 'PIN encryption' variant to derive the data encryption key, and therefore can only be used to perform data encryption operations. A BDK-3 cannot be used to perform PIN or MAC related operations.)
- Use BDK-5 to derive the appropriate keys using the Italian Standard Derivation Method SPE-DEF-041-112. BDK-5 is equivalent to BDK-1 in all aspects apart from derivation of the IKEY which uses the Italian Standard Key Derivation Method.

Additional notes for unidirectional keys:

- BDK-2 is used when the HSM is behaving like a traditional acquirer, receiving “request” messages from the terminal (which need to be decrypted and validated), and generating “response” messages for the terminal (which need to be encrypted and MAC'd).
- BDK-4 is used when the HSM is behaving like a terminal, generating “request” messages for an acquirer (which need to be encrypted and MAC'd), and receiving “response” messages from an acquirer (which need to be decrypted and validated).

Additional note when using AES DUKPT:

- The key lengths supported for the AES BDK, Initial Key (IKEY) and Update Key (UK) are 128/192/256-bit.
- The derived working keys (PIN, Data, MAC keys) will always be the same size as the AES BDK.
- Note that X9.24-3 also allows for the derivation of 3DES and HMAC keys, but this is not supported currently.
- All PIN blocks protected by an AES PIN key will use ISO format 4 (HSM format 48).

Additional note when using the Italian DUKPT Standard:

- Support for the DUKPT standard used in Italy is provided. This is identical to ANSI X9.24-3:2017 when using bi-directional 3DES keys apart from the derivation of the initial key, which uses a proprietary method. The method is defined in the Italian Standard Key Derivation Method SPE-DEF-041-112.

- BDK-5 is used for the Italian DUKPT standard and so as implied above, a BDK-5 is used in an identical way to BDK-1, the main difference being the method used to derive the initial key. Please note:
    - The Italian standard only supports 3DES, so BDK-5 is restricted to 3DES only.
    - The length of the Key Serial Number (KSN) supplied when using the host command 'A0' is 16H when using BDK-5. This differs to the length of the KSN required when using BDK-1 to 4, which requires a length of 15 H when using 3DES DUKPT.

4.3.2.3 Key Serial Number (KSN)

4.3.2.3.1 Legacy 3DES DUKPT KSN

The format of the KSN Field for Host Commands for 3DES DUKPT is described in this section.

As noted previously, the standards for 3DES DUKPT do not fully define the contents of the KSN, so there is some flexibility allowed in the length of the elements in the KSN when supplied as an input field to the host commands.

The first step required is to derive the Initial Key IKEY) from the BDK and the KSN for each terminal. Host Command 'A0' (Derive Key and Optionally Export) is used for this with 'Derive Mode' = '0'.

The derivation of the IKEY doesn't require the counter element of the KSN to be supplied, so this is omitted from the data supplied in the KSN field. The only complication is that the counter is 21 bits (represented by 4 hex characters plus one additional bit) so the right most bit of the KSN supplied in this field must be '0' – this implies the right most hex character must be even.

If the Key Set ID and the Device ID (plus 1 bit of the counter element) are represented by less than the maximum number of hex characters, then the KSN is padded to the left with 'F'. For BDK 1 to 4, the field length is 15 hex characters, and for BDK 5 the field length is 16 hex characters.

When processing a command supplied by the terminal, the entire KSN is supplied to the relevant host command, including the transaction counter. A three-character KSN Descriptor is supplied which defines the length (in characters) of each of the first 3 fields in the KSN. The KSN descriptor consists of:

- Left character:
    - Key Set ID length (lengths of 5 to 10 hex characters supported, with 10 decimal represented by 'A')
- Middle character:
    - Sub-key ID length (currently always 0).
- Right character:
    - Device ID length (lengths of 2 to 5 hex characters supported)

The KSN field consists of the following:

Element	Represented by	Description
Key Set ID (KSI)	5-10 Hex characters	Identifies the BDK to be used for this terminal.
Sub-key ID	-	Not used
Device ID	2-5 Hex characters	Unique identifier (i.e., serial number) for this terminal. Note the right most bit is the most significant bit of the next field – i.e., the Transaction Counter
Transaction Counter	5 Hex characters	Counter of the number of PIN encryptions since terminal was initialized. Note that as the Transaction Counter is 21 bits in length, the left most bit of the Transaction Counter is the most significant bit of the previous field – i.e., the Transaction Counter

The first 3 elements in the table above form the Initial Key ID, and do not change during the life of the terminal (unless a new IKEY is loaded).

Often only 64 bits of the KSN are used, with the KSN padded with "F" Hex characters to the left. In this scheme, the KSN would have the following structure:

- Padding - 4 "F" Hex characters, 16 bits.
- Key Set ID - 6 Hex characters, 24 bits.
  - This allows for about 16 million different BDKs.
- Device ID - 5 Hex characters, 20 bits.
  - This includes one bit of the Transaction Counter, leaving 19 bits for the actual Device Identifier. This means that about half a million different devices can be managed by the Device Identifier.
  - No two terminals with the same base derivation key and sub- key identifiers may be given the same device identifier.
  - Note that the terminal packs the left-most bit of the Transaction Counter as the right-most bit of the Device ID.
- Transaction Counter - 5 Hex characters, 20 bits (plus the bit included in the Device ID).
  - The transaction counter is supplied by the terminal to identify a particular transaction. It is used by an HSM to derive the required key.
  - The left-most bit is supplied as the right-most bit of the device identifier, so the length of this field is 20 bits (5 hex digits).
  - This allows for about 1 million transactions before a new IKEY would be required - a limit which is unlikely to be reached.

The terminal cannot accept a KSN longer than 20 hex characters, so the Host ensures that the total length of the first three fields does not exceed 15 characters.

4.3.2.3.2 AES DUKPT KSN

The format of the KSN Field for Host Commands for AES DUKPT is described here.

As noted previously, the Key Serial Number (KSN) for AES DUKPT follows a specific structure outlined in the ANSI X9.24-3:2017 standard and has a length of 96 bits.

The first step required is to derive the Initial Key IKEY) from the BDK and the KSN for each terminal. Host Command ‘A0’ (Derive Key and Optionally Export) is used for this with ‘Derive Mode’ = ‘0’.

The derivation of the IKEY doesn’t require the counter element of the KSN to be supplied, so this is omitted from the data supplied in the KSN field.

The Key Set ID and the Device ID are represented by 16 hex characters, and this is supplied in the KSN Field.

When processing a command supplied by the terminal, the entire KSN is supplied to the relevant host command, including the transaction counter. 24 hex characters are supplied in the KSN Field as specified in the table below:

Element	Contents	
Key Serial Number (KSN) - 96 bits (24 hex characters)	Initial Key ID (IKID) - 64 bits (16 hex characters)	Base Derivation Key ID (BDK ID) - 32 bits (8 hex characters)
		Derivation ID (DID) - 32 bits (8 hex characters)
	Transaction Counter (TC) - 32 bits (8 hex characters)	

Note that for AES DUKPT the KSN Descriptor Field is not used and is set to ‘000’.

4.3.2.4 Host Commands used for DUKPT Scheme

This section provides an overview of the host commands used for DUKPT.

4.3.2.4.1 Key Management Host Commands

payShield 10K Host Command	
ID	**Command Use**
A0	Generate a BDK. Use Mode = 0 where the key is to be used only by the HSM at this time, or Mode = 1 if it is also required to be exported to another device, encrypted under a ZMK. The key type is specified in the command as being the appropriate type of BDK.
A0	Generate an IKEY. Use Mode = A where the key is to be used only by the HSM at this time, or Mode = B if it is also required to be exported to another device, encrypted under a ZMK or TMK. The BDK key type is specified in the command. Mode = B can also be used to generate an Update IKEY and export under the Update Key derived from the current BDK. In this case the current BDK is supplied in place of the ZMK or TMK. This is only supported when using an AES BDK.
A6	Import a BDK encrypted under a ZMK. This command would only be used by the Acquirer if for any reason the BDK was generated by a third party and passed to the Acquirer.
A8	Export an IKEY encrypted under a TMK for importing into the terminals. Export a BDK or IKEY encrypted under a ZMK. This may be used by a third party that generates the BDK/IKEY and passes it to the Acquirer.
B8	Export an IKEY encrypted under an RSA public key in TR-34 format for importing into the terminals.
BW	Translate a BDK or IKEY from an 'old' LMK to a 'new' LMK.
GK	Export an IKEY encrypted under an RSA public key for importing into the terminals.

Note that the Console Commands provided for Key Management can also be used for generating, importing and exporting the BDK.

4.3.2.4.2 PIN Processing Host Commands

payShield 10K Host Command	
ID	**Command Use**
CA	The CA command can translate a PIN Block from encryption under a Terminal PIN Key (TPK) to encryption under DUKPT. Note that this command currently only supports 3DES DUKPT. This allows non-DUKPT terminals to work with a DUKPT-enabled Acquirer.
G0	Translate a DUKPT PIN Block to encryption under a ZPK. This allows the PIN to be passed into the payments network. The host provides the appropriate BDK-1, BDK-2, BDK-4 or BDK-5 as identified in the KSN sent by the terminal. G0 can also translate PINs between 2 DUKPT schemes – for example, where a DUKPT-enabled service provider exists between DUKPT terminals and a DUKPT-enabled Acquirer.
GO	Verify a PIN Using the IBM Method, with optional MAC verification. The host provides the appropriate BDK-1, BDK-2 or BDK-5 as identified in the KSN sent by the terminal. This command would only be used where the Acquirer is the issuing bank, and the transaction is not being passed through the payments network.
GQ	Verify a PIN Using the Visa PVV Method, with optional MAC verification. The host provides the appropriate BDK-1, BDK-2 or BDK-5 as identified in the KSN sent by the terminal. This command would only be used where the Acquirer is the issuing bank, and the transaction is not being passed through the payments network.
GS	Verify a PIN Using the Diebold Method, with optional MAC verification. The host provides the appropriate BDK-1, BDK-2 or BDK-5 as identified in the KSN sent by the terminal. This command would only be used where the Acquirer is the issuing bank, and the transaction is not being passed through the payments network.
GU	Verify a PIN Using the Encrypted PIN Method, with optional MAC verification. The host provides the appropriate BDK-1, BDK-2 or BDK-5 as identified in the KSN sent by the terminal. This command would only be used where the Acquirer is the issuing bank, and the transaction is not being passed through the payments network.

4.3.2.4.3 MACing Host Commands

payShield 10K Host Command	
ID	**Command Use**
GW	Generate a MAC on a message to be sent to a DUKPT terminal. The host provides the appropriate BDK- 1, BDK-2, or BDK-4 and the KSN for the terminal. Use modes 4-6 (for BDK-1), D-F (for BDK-2), or J-L (for BDK-4): Mode 4 or D: Generate an 8-byte MAC Mode 5 or E: Generate Approval MAC (4 leftmost bytes of MAC) Mode 6 or F: Generate Decline MAC (4 rightmost bytes of MAC)
GW	Verify a MAC on a message sent from a DUKPT terminal. The host provides the appropriate BDK-1, BDK- 2, or BDK-4 and the KSN for the terminal. Use modes 1-3 (for BDK-1), A-C (for BDK-2), or G-I (for BDK-4): Mode 1 or A: Verify an 8-byte MAC Mode 2 or B: Verify Approval MAC (4 leftmost bytes of MAC) Mode 3 or C: Verify Decline MAC (4 rightmost bytes of MAC)

4.3.2.4.4 Data Encryption and Decryption Host Commands

payShield 10K Host Command	
ID	**Command Use**
M0	Encrypt a block of data to be sent to a DUKPT terminal. The host provides to the HSM the appropriate BDK-1, BDK-2, BDK-3, of BDK-4 and the KSN for the terminal.
M2	Decrypt a block of data sent from a DUKPT terminal. The host provides to the HSM the appropriate BDK- 1, BDK-2, BDK-3, or BDK-4 and the KSN for the terminal.
M4	Translate a block of data from encryption under a BDK-1, BDK-2, or BDK-3 to encryption under a ZEK, DEK, or TEK. The host provides to the HSM the appropriate BDK-1, BDK-2, or BDK-3 and the KSN for the terminal. This allows data from a DUKPT terminal to be passed into the payments network or used for other purposes.
M4	Translate a block of data from encryption under a ZEK, DEK, or TEK to encryption under a BDK-1, BDK- 2, or BDK-3. The host provides to the HSM the appropriate BDK-1, BDK-2, or BDK-3 and the KSN for the terminal. This allows data from payments network or other sources to be passed to a DUKPT terminal.

4.4 German Banking Industry Committee (GBIC) Key Derivation

GBIC (or DK - Die Deutsche Kreditwirtschaft - in German) sets security standards for German domestic card payments. Until 2011, GBIC was known as ZKA - Zentraler Kreditausschuss - and this name is still in use.

payShield 10K has now been evaluated by GBIC and is now approved for use with a in the Girocard Network meeting the latest requirements as follows:

- When using either a TDES or AES Master Key, payShield 10K Software v2.2a or above is required.
- There are also a number of specific conditions that must be met, and these are listed in Section 9.1.6 in the Security Manual. Please note that an important additional condition has now been added in v2.2a as follows:
  - When deriving "send" keys for the purposes of PIN translation, host command 'G0' must be used instead of host command ‘A0’, as ‘G0’ now supports the generation of a random RNDI for each PIN translation operation.

payShield 10K supports the derivation of the PIN Encryption, Message Authentication and Data Encryption keys from the Master Key, which is referred to as the ZKA in the Thales Documentation as follows.

- When using a TDES, customers operating as a “Network Operator” (NSP) in the GBIC scheme are supported.
- When using an AES, customers operating as either a “Network Operator” (NSP) or an “Acquirer” in the GBIC scheme are supported.

The derivation of these keys uses a 2-step process as follows:

1. The first step is to derive the Master Communications Link Key from the Acquirer Master Key. The Network Operator ID_{NO} is used as input into the derivation process.
2. The second step is to derive the PIN Encryption, Message Authentication and Data Encryption keys from the Master Communications Link Key. A random number (the RNDI) is used as input into this step.

The GBIC Key Administration Centre provides the Master Keys as follows:

- The Acquirer Master Key is provided to the Acquirer. The Acquirer communicates to multiple NSP and so both derivations steps given above need to be undertaken to derive the correct PIN Encryption, Message Authentication and Data Encryption keys for each NSP. For this reason, the Network Operator ID_{NO} is required to be supplied when deriving the PIN Encryption, Message Authentication and Data Encryption keys.
- The Master Communication Link Key is provided to the Network Operator (NSP). The first step in the derivation process has already been undertaken by the Key Administration Centre in this case so the Network Operator ID_{NO} is not required to be provided when deriving the PIN Encryption, Message Authentication and Data Encryption keys.

An overview of the communication flow is shown below:

NSP -------------------------request-------------------- →	**Acquirer**
MAC Generation	MAC Verification
DATA Encryption	DATA Decryption
PIN Encryption	PIN Decryption

NSP ← -----------------------response-------------------	**Acquirer**
MAC Verification	MAC Generation
DATA Decryption	DATA Encryption

The derivation of these keys is supported in the A0 (Derive a Key and Derive & Export a Key) and the G0 (Translate a PIN from BDK to ZPK) host command. The host commands are used as follows:

- PIN Encryption and Decryption:
  - NSP uses the host commands as follows:
    - G0 to translate the PIN from encryption under a BDK to encryption using the "send" or "send request" PIN Encryption Key derived from the ZKA Communication Link Key using the random RNDI which is returned in the command.
    - Both 3DES and AES Communication Link Keys are supported:
      - For 3DES, translation to ISO PIN Block Format 0 (Thales PIN Block Format 01) and ISO PIN Block Format 1 (Thales PIN Block Format 05) are supported. Note that translation to ISO PIN Block Format 1 is allowed in PCI Mode, as the command enforces the PCI requirement that the PIN Block is encrypted using a unique-key-per-transaction.
      - For AES, translation to ISO PIN Block 4 (Thales PIN Block 48) is supported.
    - The encrypted PIN Block and the random RNDI are then supplied to the Acquirer.
  - Acquirer uses the host commands as follows (only AES is supported for the Acquirer):
    - A0 to derive the "receive request" PIN Encryption Key (ZPK) which can be used to decrypt the PIN Block only. This is derived from the Acquirer Master Key, the Network Operator ID_{NO} and the RNDI supplied by the NSP. ZKA Option 2 is used as support for the Acquirer when using 3DES is not supported.
- MAC Generation and Verification and Data Encryption and Decryption:
  - NSP uses the host commands as follows when sending a message to the Acquirer:
    - A0 to derive the "send" / "send request" MAC Generation Key (ZAK) or Data Encryption Key (ZEK). These are derived from the Communication Link Key and the random RNDI which is generated and returned by the command.
    - The resultant ZAK or ZEK can then be used in the relevant host commands as required to generate a MAC on the message or to encrypt the message.
    - Both 3DES and AES Communication Link Keys are supported:
    - The resultant MAC and encrypted message together with the random RNDI are then supplied to the Acquirer.

  - Acquirer uses the host commands as follows to receive a message from the NSP (only AES is supported for the Acquirer):
    - A0 to derive the "receive request" MAC Verification Key (ZAK) or Data Decryption Key (ZEK). This is derived from the Acquirer Master Key, the Network Operator ID_{NO} and the RNDI supplied by the NSP. ZKA Option 2 is used as support for the Acquirer when using AES - 3DES is not supported.
    - The resultant ZAK or ZEK can then be used in the relevant host commands as required to verify the MAC on the message or to decrypt the message.
  - Acquirer uses the host commands as follows to send a response to the NSP (only AES is supported for the Acquirer):
    - A0 to derive the "send response" MAC Verification Key (ZAK) or the "send response" Data Decryption Key (ZEK). These are derived from the Acquirer Master Key, the Network Operator ID_{NO} and the random RNDI generated by the command. ZKA Option 5 is used using AES as 3DES is not supported.
    - The resultant ZAK or ZEK can then be used in the relevant host commands as required to generate a MAC on the message or to encrypt the message.
  - NSP uses the host commands as follows when receiving a message to the Acquirer:
    - A0 to derive the "receive response" MAC Generation Key (ZAK) or the "receive response" Data Encryption Key (ZEK). These is derived from the Communication Link Key and the random RNDI which is supplied by the Acquirer.
    - The resultant ZAK or ZEK can then be used in the relevant host commands as required to verify the MAC on the message or to decrypt the message.
    - Both 3DES and AES Communication Link Keys are supported

Please note that generation, import and export of the AES ZKA (key usage 53) is only supported in the following host commands:

- A0 (Generate Key – generate and export is not supported),
- A6 (Import Key),
- A8 (Export Key),
- BW (Translate Keys from Old LMK to New LMK) and
- BU (Generate Key Check Value).

The AES ZKA is imported and exported using ANSI X.9.143 key block format only. The AES ZKA (key usage 53) is supported in ANSI X.9.143 as key usage '11' as defined in the GBIC standards. When importing, the key usage will be converted from key usage '11' in ANSI X.9.143 to key usage '53' for Thales Key Block.

Also note that when importing, the Key Set Identifier (KS), Optional Block is ignored and is not imported into payShield 10K – this is because the contents of the KS Optional Block provided by the GBIC Key Administration Centre is not compatible with ANSI X.9.143 or payShield 10K. When exporting, the key usage will be converted from '53' in Thales Key Block to '11' in ANSI X.9.143. The 3DES ZKA is not supported in ANSI X.9.143.

Note that authorized evaluation labs. must also evaluate the complete system against the security standard and recommend approval by GBIC. Each system has to be individually approved in addition to the HSM.

4.5 Italian Key Derivation

4.5.1 Introduction

Support for the key management of the PIN Verification Key (PVK) to the Italian Standard using a Key Block LMK is now provided. The PVK is then used with the standard host commands supporting the IBM 3624 PIN Offset method to manage PINs.

Host Command 'A0' (Derive Key) now allows derivation of a Session PIN Key Card key (SPKC) from the Master PIN Key Card key (MPKC) using the Italian Standard Key Derivation Method.

The PVK is then imported or exported encrypted under an SPKC in ANSI X9.17 format using Host Command 'A6' (Import Key) and 'A8' (Export Key).

The MPKC is imported or exported encrypted under an RSA key using Host Commands 'GI' (Import Key under an RSA Key) and 'GK' (Export Key under an RSA Key).

Further information is given below and also in the Section describing Host Command 'A0' (Derive Key) in the *payShield 10K Core Host Commands Manual* and Section 8 in the *payShield 10K Applications Manual*.

4.5.2 Summary

The derivation of two specific keys to the Italian Standard is supported by payShield 10K. Host Command 'A0' (Derive Key) is used to derive these keys from the appropriate Master Keys. The details of the keys supported are given below:

- The Master PIN Key Card key (MPKC) is used to derive the Session PIN Key Card key (SPKC).
- ○ The SPKC is then used to encrypt the PIN Key Card key (PKC) for import and export operations.
- ○ The PKC in Thales Terminology is the PIN Verification Key (PVK) (IBM Offset Method).
- ○ The PVK is used in the standard commands for PIN Generation, Verification and Change using the IBM 3624 Offset Method. Note that when using these commands, the PIN Validation Data defined in the Italian Standard must be used.
- The Master MKPOS/MKSER is used to derive the standard Thales data encryption key – the Terminal Encryption Key (TEK).
- ○ The TEK is then used to encrypt and decrypt data to/from the Terminal to the Host

Please note:

- Support is provided when using a Key Block LMK only
- Equivalent functionality is provided in the Custom Italian Commands software when using a Variant LMK using Custom Host Command TG (Derive Key)
- The Italian standard only supports single, double or triple length DES keys for the MKPOS/MKSER and the TEK. AES is not supported.
- The length of the master key can be single, double or triple length.
- The length of the derived key can be single, double or triple length, but must not be greater the length of the master key.

- The length of the Derivation Data fields required to be supplied by the application depend on the length of the key being derived and the Derivation Mode – see the table given for host command A0 (Derive a Key)
- Host Command A0 only supports the derivation of the TEK and the SPKC – the option to Derive and Export is not provided as it is not required.

4.5.3 Key Management

The following key management host commands are provided to manage the keys used for the Italian Standard.

4.5.3.1 A0 - Generate a Key

This command supports generation of the following Italian keys:

Usage	Algorithms Supported	Modes of Use Supported	Key Name
'56'	'D', 'T'	'X' , 'N'	Master PIN Key Card (MPKC) – Italian Standard
'57'	'D', 'T'	'X' , 'N'	Master MKPOS/MKSER – Italian Standard

Generate and export results in a standard error code being returned and processing terminating.

There is no requirement for the SPKC to be generated as this is derived from the MPKC.

4.5.3.2 A0 - Derive a Key

This command supports derivation of the following Italian keys:

Usage	Algorithms Supported	Modes of Use Supported	Key Name
'58'	'D', 'T'	'B', 'D', 'E'	Session PKC Key Encryption Key (SPKC) – Derived from the MPKC
'23'	'D', 'T'	'B', 'D', 'E'	Terminal Encryption Key (TEK) – derived from the MKPOS/MKSER

4.5.3.3 A6 - Import a Key - Import Master Derivation Keys

The following Italian keys can be imported using both ANSI X9.17 and Thales Key Block using the ZMK. This is to allow migration from Variant to Key Block as described in the *payShield 10K Applications Manual,* Section 8.8:

Usage	Algorithms Supported	Modes of Use Supported	Key Name
'56'	'D', 'T'	'X' , 'N'	Master PIN Key Card (MPKC) – Italian Standard
'57'	'D', 'T'	'X' , 'N'	Master MKPOS/MKSER – Italian Standard

Security Settings:

- To import in ANSI X9.17 format, the Enable X9.17 for Import security setting must be set to Yes

Authorization:

- Authorization is not required when importing using Thales Key Block format.
- Authorization is required when importing from non-key block format – Host Command A6 must be Authorized to import the MPKC and MKPOS/MKSER keys.

Notes:

- The ZMK must be equal or greater in length than the keys being imported.

4.5.3.4 A6 – Import a Key - Import PVK Encrypted under a SPKC in ANSI X.19 Format

The import of a PVK encrypted under the SPKC in ANSI X9.17 format or Thales Key Block Format is supported in this Host Command. This is addition to the functions already provided to support import of the PVK

The PVK is specified below:

Usage	Algorithms Supported	Modes of Use Supported	Key Name
'V1'	'D', 'T'	'C', 'G', 'N', 'V'	PVK - PIN Verification Key (IBM Offset Method)

The SPKC used to encrypt the PVK is specified below:

Usage	Algorithms Supported	Modes of Use Supported	Key Name
'58'	'D', 'T'	'B', 'D'	Session PKC Key Encryption Key (SPKC)

Security Settings:

- To import in ANSI X9.17 format, the Enable X9.17 for Import security setting must be set to Yes.

Authorization:

- Authorization is not required when importing using Thales Key Block format.
- Authorization is required when importing from a non-key block format. Host Command A6 must be authorized to import the PVK.

Notes:

- The SPKC cannot be used in any other export operation with any other keys.
- The encrypting key (SPKC) must be equal or greater in length than the PVK.

4.5.3.5 A8 – Export a Key

The export of the PVK encrypted under the SPKC in ANSI X9.17 format or Thales Key Block Format using the SPKC is supported in this Host Command as follows. This is addition to the functions already provided to support the export of the PVK.

The PVK is specified below:

Usage	Algorithms Supported	Modes of Use Supported	Key Name
'V1'	'D', 'T'	'C', 'G', 'N', 'V'	PIN Verification Key (IBM Offset Method)

The key used to encrypt the PVK is as follows:

Usage	Algorithms Supported	Modes of Use Supported	Key Name
'58'	'D', 'T'	'B', 'E'	Session PKC Key Encryption Key (SPKC)

Security Settings:

- To export in ANSI X9.17 format, the Enable X9.17 for Export security setting must be set to Yes.
- If security setting Enforce PCI HSMv3 Key Equivalence for Key Wrapping is set to Yes, the encrypting key (SPKC) must be equal or greater in length than the key encrypted (PVK).

Authorization:

- Authorization is required – Host Command A8 must be authorized to export the PVK.

Note:

- The SPKC cannot be used in any other export operation with any other keys.

4.5.3.6 GI/GJ - Import Key or data under an RSA Public Key

Import of the following Italian key is supported in this command:

Usage	Algorithms Supported	Modes of Use Supported	Key Name
'56'	'D', 'T'	'X' , 'N'	Master PIN Key Card (MPKC) – Italian Standard

Security Settings:

- Authorized State is required when the security setting Authorized State required when importing DES key under RSA key is set to Yes.

Authorized State:

- If the above Security Setting is set to Yes, Authorized State is required by authorizing this command.

4.5.3.7 GK/GL - Export Key or data under an RSA Public Key

Export of the following Italian keys is supported in this command:

Usage	Algorithms Supported	Modes of Use Supported	Key Name
'56'	'D', 'T'	'X' , 'N'	Master PIN Key Card (MPKC) – Italian Standard
'57'	'D', 'T'	'X' , 'N'	Master MKPOS/MKSER – Italian Standard

Security Settings:

- If security setting Enforce PCI HSMv3 Key Equivalence for Key Wrapping is set to Yes, the encrypting key (SPKC) must be equal or greater in length than the key encrypted (PVK).

4.5.3.8 BU – Generate Key Check Value (KCV)

Support is provided to generate a key check value for the Italian keys in the following commands:

Usage	Algorithms Supported	Modes of Use Supported	Key Name
'56'	'D', 'T'	'X' , 'N'	Master PIN Key Card (MPKC) – Italian Standard
'57'	'D', 'T'	'X' , 'N'	Master MKPOS/MKSER – Italian Standard
'58'	'D', 'T'	'B', 'D', 'E'	Session PKC Key Encryption Key (SPKC)

4.5.3.9 BW/BX - Translate Keys from Old LMK to New LMK and Migrate to New Key Type

Translation for the following Italian keys is provided:

Usage	Algorithms Supported	Modes of Use Supported	Key Name
'56'	'D', 'T'	'X' , 'N'	Master PIN Key Card (MPKC) – Italian Standard
'57'	'D', 'T'	'X' , 'N'	Master MKPOS/MKSER – Italian Standard
'58'	'D', 'T'	'B', 'D', 'E'	Session PKC Key Encryption Key (SPKC)

The command performs the following translations on keys with key usages 56, 57 and 58:

- Key Block LMK > Key Block LMK
- ○ Translate the keys from encryption under the old Key Block LMK to encryption under the new Key Block LMK
- Variant LMK > Key Block LMK
- ○ As these keys are not supported in base when using a Variant LMK, there is no requirement for BW to support translation from a Variant LMK to a Key Block LMK. See the *payShield 10K Applications Manual*, for further information.
- Additional Notes:
- ○ The SPKC can only be derived from the MPKC – there are no other commands that generate, import or export the SPKC.
- ○ MKPOS/MKSER (key usage 57) are non-standard scheme specific keys, and so are not supported in ANSI X.9.143.

5 RSA and ECC Cryptosystem

This chapter provides information on the functionality provided to support both the RSA and ECC cryptographic algorithms.

RSA functionality has been supported in payShield for a number of years. Support for Elliptic Curve Cryptography (ECC) is introduced for the payShield 10K with the release of software version v1.2a.

Both RSA and ECC are "asymmetric" cryptographic algorithms, where the encryption and decryption keys are different. With both algorithms, it is computationally infeasible to deduce the decryption key from the encryption key. Therefore, the encryption key may be made public and distributed in clear without compromising the security of the decryption key and both algorithms support Public Key Cryptography. In contrast, both the DES and AES cryptographic algorithms are "symmetric" where the keys used by each party to secure a message are the same and therefore must remain secret.

Both RSA and ECC algorithms are usually used in two ways:

- To digitally sign electronic messages to provide proof of the identity of the sender, and to protect the integrity of the contents of the messages.
- To automate and simplify the difficult problem of secret key distribution and management in large distributed networks, such as the Internet.

5.1 RSA Cryptosystem

The RSA public key algorithm was devised in 1979 by Rivest, Shamir and Adleman (hence the name). As noted above, it is an asymmetric cryptographic algorithm, which means that the encryption and decryption keys are different, and that it is computationally infeasible to deduce the decryption key from the encryption key. The RSA algorithm is implemented in all variants of the payShield 10K.

The length of the RSA keys used can be selected from 320 to 4096 bits.

Note: RSA keys longer than 2048 bits can only be used with AES Key Block LMKs: TDES LMKs have insufficient strength to protect these keys.

5.1.1 RSA Cryptosystem Functions

The RSA cryptosystem provides the following functions:

- Generation of variable-length RSA keys.
- Validation of public key certificates.
- Generation and validation of digital signatures.
- Secure DES key management using RSA public master keys
- Generation of hash values

These functions are implemented by the following host commands:

- Generate an RSA Key Set (EI)
- Load a Secret Key (EK)
- Translate a Secret Key from the Old LMK to a New LMK (EM)

- Generate a MAC on a Public Key (EO)
- Verify a MAC on a Public Key (EQ)
- Validate a Certificate and Generate a MAC on its Public Key (ES)
- Translate a MAC on a Public Key (EU)
- Generate a Signature (EW)
- Validate a Signature (EY)
- Import a DES Key (GI)
- Export a DES Key (GK)
- Hash a Block of Data (GM)

Note: Details of these commands can be found in the *payShield 10K Core Host Commands reference manual*.

5.1.2 Even Public Exponents

There is a variant of RSA (known as the "Rabin" variant) which utilizes an even Public Exponent. This variant cannot be used for unique encryption/decryption unless the associated data contains some redundant information which can be used to determine the correct result. Although the commands specified in this document, which use a Public Key, could be used with an even Exponent, there is no guarantee that the results produced by these commands will be correct. It is strongly recommended that the commands in this document are used only with odd Public Exponents. Note that it is not possible to use an HSM to generate an RSA Key Set that has an even Public Exponent.

5.2 ECC Cryptosystem

As noted in the introduction to this chapter, like RSA, Elliptic Curve Cryptography (ECC) is an asymmetric algorithm that supports Public Key cryptography. It is based on a branch of mathematics called elliptic curves and is an alternative technique to RSA.

Some of the advantages over RSA are the smaller key sizes required to provide an equivalent level of security, and an increase in speed of key generation. Therefore, it is considered the next generation implementation of public key cryptography and is becoming more widely used in the payments industry.

5.2.1 Functionality Supported

Functions are provided for:

- ECC Key management:
    - Key pair generation
    - Generation certificate signing request
    - Load public key certificate
    - Translate private and public keys when LMK changed
- ECC Signature Functions:
    - Generate and validate signatures on a message using ECDSA

- ECC Key Derivation Functions:
    - Key Derivation Using Key Agreement: Derives keys using an Elliptic Curve Key Agreement Algorithm (ECKA).
    - Either ECKA-EG (El-Gamal) using ephemeral/static keys
    - Or ECKA-DH (Diffie-Hellman) using ephemeral keys only

5.2.2 Host Commands Supporting ECC

5.2.2.1 ECC Key Management:

- 'FY' – Generate ECC Key Pair

  The Private key is returned encrypted under the LMK in key block format.
- 'QE' – Generate Certificate Request

  Generates a certificate signing request CSR by signing the subject and public key information with the private key to create a self-signed certificate in PKCS#10 format
- 'EO' – Import Public Key Certificate

  Imports a Public Key by creating the public key in key block format
- 'EK' – Load Private Key

  Load a private key (encrypted under the LMK) into the HSM's tamper-protected memory
- 'EM' – Translate a Private Key

  Translate a private key from encryption under an 'old' LMK to encryption under the 'new' LMK
- 'EU' – Translate a Public Key

  Translate a public key from protection under an 'old' LMK to protection under the 'new' LMK

5.2.2.2 ECC Signature Functions:

- 'EW' – Generate Signature

  Generate a signature on a message using a private key using ECDSA
- 'EY' – Validate Signature

  Validate signature on a message using a private key using ECDSA

5.2.2.3 ECC Key Derivation Functions:

- 'IG' – Key Derivation Using Key Agreement.

  Derives keys using an Elliptic Curve Key Agreement Algorithm (ECKA). The command supports either ECKA- EG using ephemeral/static keys or ECKA-DH using ephemeral keys only.

5.2.3 ECC General Information

This section includes some general information on the support for ECC provided in payShield 10K.

It is important to note that for security reasons functions using ECC public and private keys are only available when used with an AES Key Block LMK.

The ECC Prime Curves currently supported for payments are defined in FIPS 186-3 and are as follows:

- '00' – P-256
- '01' – P-384
- '02' – P-521

5.3 General Information

5.3.1 HSM Buffer Sizes

The payShield 10K HSM has a 32K-byte input buffer per connection, and it is the responsibility of the host application to ensure that the total amount of data sent in an HSM command does not cause a buffer overflow.

5.3.2 Data Formats

Certificates, signatures, encrypted key blocks and message data supplied in commands specified in this document are binary fields, with the leftmost byte being the most significant and the rightmost byte being the least significant. Note that the binary data may be right justified and padded to the left with zeros, if necessary, in order to make the data length (in bits) an exact multiple of eight.

5.4 Common Parameters

Within these functions certain common parameters are defined.

5.4.1 DES Key Type

The DES Key Type field is 4 digits. The first two digits indicate the LMK pair used to encrypt the key, the last two digits indicate the LMK variant.

For example:

- If the DES Key Type is 0600, LMK pair 06-07 is used (no variant).
- If the DES Key Type is 3007, variant 7 of LMK pair 30-31 is used.

5.4.2 Signature Algorithm

01	=	RSA
02	=	ECC

5.4.3 ECC Curve Reference

Prime Curves defined in FIPS 186-3:

00	=	P-256
01	=	P-384
02	=	P-521

5.4.4 Encryption Identifier

```
01 =   RSA
```

5.4.5 Hash Identifier

```
01 =   SHA-1, produces a 20 byte result.
02 =   MD5, produces a 16 byte result.
03 =   ISO 10118-2, produces a 16 byte result.
04 =   No hash.
05 =   SHA-224
06 =   SHA-256
07 =   SHA-384
08 =   SHA-512
```

5.4.5.1 01 = SHA-1 hashing algorithm

The ASN.1 DER object identifier for this hashing function is:

```
{iso(1) identified-organization(3) oiw(14) secsig(3) 2 26}
```

which encodes as:

```
2B 0E 03 02 1A
```

5.4.5.2 02 = MD5 hashing algorithm

The ASN.1 DER object identifier for this hashing function is:

```
{iso(1) member-body(2) US(840) rsadsi(113549) digest Algorithm(2) 5 }
```

which encodes as:

```
2A 86 48 86 F7 0D 02 05
```

5.4.5.3 03 = ISO 10118-2 hashing algorithm

The ASN.1 DER object identifier for this hashing function is:

```
2 10 67 4
```

which encodes as:

```
5A 43 04
```

5.4.5.4 04 = No hash

The no-hash option can be used when an HSM provides signature generation or validation, or certificate validation, on data that is hashed outside an HSM.

If the no-hash option is chosen, the data that is provided in the Validate a Certificate, Generate a Signature and Validate a Signature commands is not modified in any way by an HSM, so it must be precisely the data in the plain signature block (which depends on the pad mode selected by the Pad Mode Identifier). It is the responsibility of the Host application to ensure that the precise data to be included in the signature block is supplied in the command.

5.4.5.5 05 = SHA-224 hashing algorithm

The ASN1.DER object identifier for this hashing function is:

```
id-SHA224 OBJECT IDENTIFIER ::=

{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
nistalgorithm(4) hashalgs(2) sha224(4) }
```

which encodes as:

```
60 86 48 01 65 03 04 02 04
```

5.4.5.6 06 = SHA-256 hashing algorithm

The ASN1.DER object identifier for this hashing function is:

```
id-SHA256 OBJECT IDENTIFIER ::=

{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
nistalgorithm(4) hashalgs(2) sha256(1) }
```

which encodes as:

```
60 86 48 01 65 03 04 02 01
```

5.4.5.7 07 = SHA-384 hashing algorithm

The ASN1.DER object identifier for this hashing function is:

```
id-SHA384 OBJECT IDENTIFIER ::=

{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
nistalgorithm(4) hashalgs(2) sha384(2) }
```

which encodes as:

```
60 86 48 01 65 03 04 02 02
```

5.4.5.8 08 = SHA-512 hashing algorithm

The ASN1.DER object identifier for this hashing function is:

```
id-SHA512 OBJECT IDENTIFIER ::=

{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101) csor(3)
nistalgorithm(4) hashalgs(2) sha512(3) }
```

which encodes as:

60 86 48 01 65 03 04 02 03

Example:

If the SHA-1 algorithm is used to hash the data and the resultant hash value is:

0123456789ABCDEF0123456789ABCDEF01234567

and if the PKCS#1 pad mode is used, the data to be provided must be the complete ASN.1 DER encoded DigestInfo, which is:

30 21 300906052B0E03021A0500 04140123456789ABCDEF0123456789ABCDEF01234567

Note that when using the no-hash mode, the HSM checks that the DER encoded DigestInfo syntax is correct. If there is a digest info syntax error, the HSM returns error code 74.

5.4.6 Pad Mode Identifier

This section describes the padding methods supported by payShield 10K. Details can be found in the relevant RFC which are freely available online.

Please note that the padding method supported varies depending on the Host Command used as given below. Further details are given in the payShield 10K Core Host Commands Manual.

5.4.6.1 Pad Mode 1

Pad Mode 1 supports the following two padding schemes, depending on which host command is used:

- EME-PKCS1-v1_5
    - Specified in RFC 2437: PKCS #1: RSA Cryptography Specifications Version 2.0 Section 9.1.2.
    - Supported in Host Commands:
        - AQ (Translate an RSA-encrypted PIN to a ZPK or TPK-encrypted PIN)
        - GI (Import Key or data under an RSA Public Key)
        - GK (Export Key under an RSA Public Key)
- EMSA-PKCS1-v1_5
    - Specified in RFC 8017 PKCS #1: RSA Cryptography Specifications Version 2.2, Section 9.2
    - Supported in Host Commands:
        - ES (Validate a Certificate and Import the Public Key)
        - EW (Generate Signature)
        - EY (Validate Signature)
        - GI (Import Key or data under an RSA Public Key)
        - GK (Export Key under an RSA Public Key)
        - JW (JWT Encode)
        - QE (Generate a Certificate Request)
- RSAES-PKCS1-v1_5
    - Specified in RFC 8017 PKCS #1: RSA Cryptography Specifications Version 2.2, Section 7.2
    - Supported in Host Commands:
        - B8 (TR-34 Export)

When the PKCS#1 pad mode is used, the following validity checks are carried out: For a validation operation (Validate a Certificate, Validate a Signature):

- The length of the data to be validated is equal to the length (in bytes) of the modulus of the key to be used for the validation. If not, error code 76 is returned.

- The first byte of the clear data block is 00. If not, error code 77 is returned.
- The second byte of the clear data block is 01. If not, error code 77 is returned.
- Subsequent bytes consist of at least 8 bytes of binary 1s, followed by a zero byte. If not, error code 77 is returned.
- The hash algorithm object identifier corresponds to that of the identifier of the hash algorithm supplied in the command message. If not, error code 79 is returned.
- The digest is compared with the hash of the supplied data. If the two values are not equal, error code 02 is returned

For a generation operation (Generate a Signature):

- The length (in bytes) of the data block D is at most m-11 (where m is the length, in bytes, of the modulus of the key to be used). If not, error code 76 is returned.

For an import key operation (Import a DES Key):

- The length of the imported key block is equal to the length (in bytes) of the modulus of the secret key to be used to decrypt the block. If not, error code 76 is returned.
- The first byte of the clear data block is 00 and the second byte is 02. If not, error code 77 is returned.
- Subsequent bytes consist of at least 8 bytes of random non-zero bytes, followed by a zero byte. If not, error code 77 is returned.
- The data block D conforms to the ASN.1 encoding rules. If not, error code 77 is returned.

For an export key operation (Export a DES Key):

The length (in bytes) of the data block D is at most m-11 (where m is the length, in bytes, of the modulus of the key to be used). If not, error code 76 is returned.

5.4.6.2 Pad Mode 2

Pad Mode 2 supports the following padding schemes, in a selection of host commands:

- EME-OAEP-ENCODE
  - Specified in RFC 2437: PKCS #1: RSA Cryptography Specifications Version 2.0 Section 9.1.1.1
  - Supported in Host Command:
    - AQ (Translate an RSA-encrypted PIN to a ZPK or TPK-encrypted PIN)
    - GI (Import Key or data under an RSA Public Key)
    - GK (Export Key under an RSA Public Key)

Optimal Asymmetric Encryption Padding (OAEP) was introduced in PKCS#1 v2.0, as an improvement on the original, simple PKCS#1 v 1.5 method described above. OAEP requires four additional parameters:

- Mask Generation Function

`01` = MGF

- MGF Hash Function

`01` = SHA1

- OAEP Encoding Parameters Length

Specifies the length of the encoding parameters.

- OAEP Encoding Parameters

The host may optionally supply a set of OAEP encoding parameters. If OAEP padding is used, but no Encoding Parameters are required, then OAEP Encoding Parameters Length should be "00", and this field will be empty.

The OAEP fields are encoded according to PKCS#1. The HSM does not interpret or validate the contents of this field, it applies the Hash Algorithm to it and feeds the result into the OAEP process.

5.4.6.3 Pad Mode 4

Pad Mode 4 supports the following two padding schemes, depending on which host command is used:

- RSASSA-PSS
    - Specified in RFC 8017 PKCS #1: RSA Cryptography Specifications Version 2.2, Section 8.
    - Supported in Host Commands:
        - QE (Generate a Certificate Request)
        - EW (Generate Signature)
        - EY (Validate Signature)
- EMSA-PSS
    - Specified in RFC 8017 PKCS #1: RSA Cryptography Specifications Version 2.2, Section 9.
    - Supported in Host Command:
        - JW (JWT Encode)

5.4.7 Key Data Block Type

The Key Data Block Type field is used with the host commands to import and export keys encrypted under an RSA public key. The values supported are:

01 =	Standard Key Data Block Type
02 =	Key Data Block Template
03 =	Unformatted Key Data Block
04 =	ASN.1 Encoded Key Data Block

Key Data Block Types '01', '02', and '03' may be used for importing and exporting DES/AES keys. Key Data Block Types '02', '03', and '04' may be used for importing and exporting HMAC keys.

5.4.7.1 01 = Standard Key Data Block Type

This is the standard key data block format supported in earlier versions of payShield. The format is as shown in the PKCS#1 v1.5 padding scheme above, i.e.:

```
Key Data Block :: SEQUENCE {
key   OCTET STRING,
iv    OCTET STRING
}
```

Note: For a DES/3DES key, the 'key' may be 8, 16 or 24 bytes, and the 'iv' is 8 bytes; for an AES key, the 'key' may be 16, 24 or 32 bytes, and the 'iv' is 16 bytes.

5.4.7.2 02 = Key Data Block Template

This method supports any valid ASN.1 DER encoded Key Data Block format, which may consist of arbitrary encoded data with a Key Data Block field containing a plain-text DES Key of single, double or triple length.

The Host must supply a block of data, which conforms to ASN.1 DER encoding, with an indication of the position in which the key is located (DES Key Offset). The key data area of the template must be zero filled.

For key export, an HSM overlays the zero filled data with a DES or Triple DES key as appropriate.

For key import, an HSM verifies that the decrypted data conforms to the specified padding, then checks that the supplied template matches the decoded data. It then extracts the data at the position indicated by the DES Key Offset and use this as the key for import.

An example Key Data Block structure and template is shown below. This structure is used for Diebold Remote Key Transport.

Example Key Data Block Structure

```
RecipientInfo ::= SEQUENCE {
  version                        Version,
  issuerAndSerialNumber               IssuerAndSerialNumber,
  keyEncryptionAlgorithm              KeyEncryptionAlgorithmIdentifier,
  keyOrKeyBlock                  KeyOrKeyBlock
}

KeyOrKey block ::= CHOICE {
  encryptedKey                   EncryptedKey,
  EncryptedKeyBlock              encryptedKeyBlock
}

EncryptedKey ::= OCTET STRING

EncryptedKeyBlock ::= ENCRYPTED KeyBlock – a BIT STRING

Key block ::= SEQUENCE {
  version                        Version, -- 0
  originatorIssuerAndSerialNumber     IssuerAndSerialNumber,
  keyId                          KeyId,
  key                            Key,
  keyUsage [0]                   KeyUsage OPTIONAL
}
```

Example Key Data Block Template

A key data block template corresponding to the above structure is shown on the following page:

```
30 61                                                Key block
 02 01 00                                       version = 0
  30 47
originatorIssuerAndSerialNumber
   30 42                                        issuer
    31 10
     30 0E
```

```
      06 03 55 04 03                             attributeType =
commonName
       13 07 52 6F 6F 74 20 43 41                      attributeValue =
"Root CA"
    31 2E
     30 2C
      06 03 55 04 0A                             attributeType =
organizationName
       13 25                                     attributeValue =
"Initial Certificate
     Authority Company"
        49 6E 69 74 69 61 6C 20 43 65 72 74 69
        66 69 63 61 74 65 20 41 75 74 68 6F 72
        69 74 79 20 43 6F 6D 70 61 6E 79
    02 01 02                                     serialNumber = 2
  02 01 00                                       keyIdentifier = 0, A key
  04 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     key
```

The Key Data Block Template requires four additional parameters:

- Key Data Block Template Length

  The length of the key data block data
- Key Data Block Template

  The actual template, as shown in the example above
- DES Key Length

  The length of the DES key within the key data block.
- DES Key Offset

  Offset to the location of the DES key within the key data block.

  In the example above this points to the beginning of the block of zeros shown in bold italics and the offset is 83 (decimal) bytes.

Another two optional parameters support a check value. The Check Value is not required for the Diebold implementation, but provides flexibility to support applications that require a check value in the key data block.

- Check Value Length

  Length in bytes of the check value field. This field should be 0 if no check value is used.
- Check Value Offset

Offset to the location of the check value within the key data block.

5.4.7.3 03 = Unformatted Key Data Block

This is the format required for remote ATM key loading for NCR ATMs. It consists of only 8, 16 or 24 bytes of key data (for a single, double or triple length DES key), with no encoding or additional information.

5.4.7.4 04 = ASN.1 Encoded Key Data Block

04 = ASN.1 Encoded Key Data Block

This key data block is encoded in ASN.1 format as shown below:

```
Key Data Block :: SEQUENCE {
key   OCTET STRING,
iv    OCTET STRING
}
```

5.4.8 Public Key Encoding

`01` = DER encoding for ASN.1 public key (INTEGER uses unsigned representation)

`02` = DER encoding for ASN.1 public key (INTEGER uses 2's complement representation)

`03` = DER encoding for ASN.1 ECC public key in X9.62 format

`04` = Uncompressed encoding 05 = ISO 7816 format

`06` = Public key in key block format

An ASN.1 RSAPublicKey has the following definition:

```
RSAPublicKey :: =      SEQUENCE {
  modulus              INTEGER, -- n
  publicExponent       INTEGER  -- e
}
```

payShield 10K HSM base software supports two different representations for INTEGER values in the RSAPublicKey-unsigned INTEGER (Public Key Encoding 01) and 2's complement INTEGER (Public Key Encoding 02).

A public key Modulus represented in 2's complement form will always have a leading 00 byte, the most significant bit of the second byte will always be '1'. A public key modulus represented in unsigned form will never begin with a 00 byte, the most significant bit of the modulus will always be '1'.

To avoid interoperability problems with non-Thales host security modules, it is recommended to use Public Key Encoding 02.

Examples:

1024 bit modulus with an exponent of 03 using Public Key Encoding 01:

Sequence Identifier	Byte Length	Integer Identifier	Modulus Length	Modulus	Integer Identifier	Exponent Length	Exponent
X'30	X'81 X'86	X'02	X'81 X'80	128	X'02	X'01	X'03

Where:

- X'30 is the identifier specifying the start of a sequence.
- X'81 X'86 specifies the length of the following field in bytes:
  - If value is between X'01 and X'7F then this directly specifies length of following field in bytes (1byte to 127 bytes).
  - If value is greater than X'80 it defines the number of bytes to define the length of the next field in the above example X'81 therefore length i.e. 1 byte (X'86 - 134 bytes).
- X'02 is the identifier specifying the start of the integer.
- X'81 X'80 specifies the length of the following field in bytes using the same definition as above (128 Bytes).
- The modulus in this example is 1024 bits. For Public Key Encoding 01 this is represented in unsigned form. The resulting field is 128 bytes.
- X'02 is the identifier specifying the start of the second integer.
- X'01 specifies the length of the following field in bytes using the same definition as above (1 Byte).
- X'03 is the value of the exponent.

1024 bit modulus with an exponent of 03 using Public Key Encoding 02:

Sequence Identifier	Byte Length	Integer Identifier	Modulus Length	Modulus	Integer Identifier	Exponent Length	Exponent
X'30	X'81 X'87	X'02	X'81 X'81	129	X'02	X'01	X'03

Where:

- X'30 is the identifier specifying the start of a sequence.
- X'81 X'87 specifies the length of the following field in bytes. Using the same definition as described in example above, X'87 - 135 bytes.
- X'02 is the identifier specifying the start of the integer.
- X'81 X'81 specifies the length of the following field in bytes using the same definition as above (129 Bytes).
- The modulus in this example is 1024 bits. For Public Key Encoding 02 this is represented in 2's complement form. The resulting field is 129 bytes. The first byte will always be X'00.
- X'02 is the identifier specifying the start of the second integer.
- X'01 specifies the length of the following field in bytes using the same definition as above (1 Byte).

- X'03 is the value of the exponent.

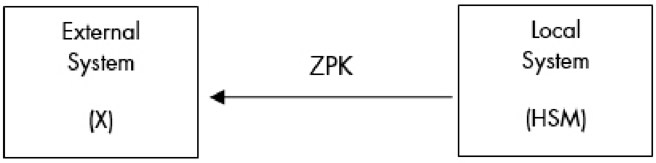
5.4.9 Signature Format & Signature Output Format (ECC Only)

`0` = Plain format

`1` = X9.62 ASN 1 encoded

5.5 Worked Examples

This section demonstrates the typical use of the GK (Export Key under RSA Public Key) host command, whereby a ZPK is securely transported from the local system (an HSM) to a trusted external system (X).



Both systems must previously have generated RSA key pairs. Additionally, both systems must trust each other's public keys. For the local system, this can be achieved using the EO (Generate a MAC on a Public Key) command, which requires Authorized State. How the external system achieves this is outside the scope of this example.

The ZPK is encrypted using X's public key, positioned within a block containing additional data, and then signed using an HSM's private key.

The examples provide values for all input and (anticipated) output parameters to the GK (Export Key under RSA Public Key) command when using both 1024-bit and 2048-bit RSA keys.

The first section defines a number of values that are used in producing the encrypted & signed ZPK (using 1024-bit RSA keys).

The second section *Sample Data Generation Procedure* (1024-bit RSA keys) shows the intermediate steps to producing the encrypted & signed ZPK (using 1024-bit RSA keys).

The third section *Sample Data Calculation* (1024-bit RSA keys) consists of a complete command message / response message sequence for this example (using 1024-bit RSA keys).

The fourth section *Sample Data Definitions* (2048-bit RSA keys) defines a number of values that are used in producing the encrypted & signed ZPK (using 2048-bit RSA keys).

The fifth section *Sample Data Generation Procedure* (2048-bit RSA keys) shows the intermediate steps to producing the encrypted & signed ZPK (using 2048-bit RSA keys).

The sixth and final section *Sample Data Calculation* (2048-bit RSA keys) consists of a complete command message / response message sequence for this example (using 2048-bit RSA keys).

5.5.1 Sample Data Definitions (1024-bit RSA keys)

```
DES Key to export = 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C

DES Key Type = ZPK

DES Key Check Value = 8B 15 36 D4 D6 0F EC 19
```

```
LMK Encrypted DES Key (as ZPK) in key scheme U = U 51 3E CF 7E 7D 0A 55
54 90 0F A7 09 B9 A5 F7 21

Encryption Padding mode = OAEP (MGF1, SHA-1, OAEP Params = "09 08 07 06")

Encryption Public Key (ASN.1 DER encoded) =
0000: 30 81 89 02 81 81 00 40 CE 04 F8 5B 70 30 49 C8  0□‰.□□.@Î.ø[p0IÈ
0010: 48 6C 84 C8 EB 40 54 EC B2 2A FD 6B 78 96 7D A2  Hl„Èë@Tì²*ýkx–}¢
0020: D8 CD F7 5E 5E B9 63 94 95 43 C6 8F 54 31 88 11  ØÍ÷^^¹c"•CÆ□T1ˆ.
0030: B3 A9 91 27 A8 C1 D0 9C B1 3C CD 70 05 80 8E 91  ³©‘'¨ÁÐœ±<Íp.€Ž‘
0040: 80 80 B0 AF 5A 58 C7 11 8B 44 3F C7 CD AE E4 D5  €€°¯ZXÇ.‹D?ÇÍ®äÕ
0050: A7 F6 3C C0 F3 59 6C 98 E3 7B 9F 97 9D 53 C4 4B  §ö<ÀóYl˜ã{Ÿ—□SÄK
0060: E1 0E C4 06 8F DD 7A 38 24 BD 20 34 F0 E5 EA 19  á.Ä.□Ýz8$½ 4ðåê.
0070: 8E FF 9B 36 B0 EF 65 90 BF D0 50 99 E2 8A E0 4D  Žÿ›6°ïe□¿ÐP™âŠàM
0080: 09 96 D2 E1 36 21 E9 02 03 01 00 01              .–Òá6!é.....

Encryption Public Key MAC = 0E FA 9C 3A

Encryption Public Key Authentication Data = "ABCDEFG"

Encryption Private Key (d) =
0000: 03 36 B4 44 64 B4 71 90 97 20 10 51 9D 6D 1D 29  .6´Dd´q□— .Q□m.)
0010: 98 FB 54 EA 70 53 F0 92 96 6A CD FC 00 70 0E 1D  ˜ûTêpSð’–jÍü.p..
0020: 84 16 CA DF A3 E7 F6 F4 DA 7B E0 62 D4 66 A8 05  „.Êß£çöôÚ{àbÔf¨.
0030: E2 5F 5F B6 88 61 9D 78 74 7A BC E7 06 2B 22 CF  â__¶ˆa□xtz¼ç.+"Ï
0040: DF A6 7B 5B A5 4D 72 BD E4 69 10 03 60 D2 06 37  ß¦{[¥Mr½äi..`Ò.7
0050: EC E7 09 A6 19 0C BF 33 D3 CE C7 F9 89 94 31 41  ìç.¦..¿3ÓÎÇù‰"1A
0060: 42 11 2F 5E 48 FA 05 C6 70 22 D6 05 7E 6D 10 78  B./^Hú.Æp"Ö.~m.x
0070: 78 BA 9C D4 F2 A2 63 9B A8 71 A4 4D F7 EE 8A 21  x°œÔò¢c›¨q¤M÷îŠ!

Signature Public Key (ASN.1 DER encoded) =
0000: 30 81 89 02 81 81 00 40 85 8A 70 C8 7F 3E AB 9E  0□‰.□□.@…ŠpÈ□>«ž
0010: 13 6D 2C 0C A3 B6 CC 47 ED 68 6E 3C 6F 31 46 C5  .m,.£¶ÌGíhn<o1FÅ
0020: FE 05 64 3B 4F EE F9 B6 9A 1C ED 6A EB D3 B9 15  þ.d;Oîù¶š.íjëÓ¹.
0030: 31 3C 8C D1 5C BF 26 FB AB D4 8C 6E 08 2A D0 F0  1<ŒÑ\¿&û«ÔŒn.*Ðð
0040: D5 FD 03 64 56 B6 CE A5 91 DF B7 F5 A4 30 B2 6B  Õý.dV¶Î¥‘ß·õ¤0²k
0050: EA 3A 8C E4 15 2C DC 50 DE AF 9D DF EF D8 AC 10  ê:Œä.,ÜPÞ¯□ßïØ¬.
0060: DB FF 2B 92 F1 97 C7 D6 D0 CF BC 1A 6D 85 06 CB  Ûÿ+’ñ—ÇÖÐÏ¼.m….Ë
0070: F8 1B 76 F1 5D 32 AC D0 72 ED 34 72 30 A9 24 F0  ø.vñ]2¬Ðrí4r0©$ð
0080: 6D B2 E3 8D 55 33 33 02 03 01 00 01              m²ã□U33.....

Signature Private Key (d) =
0000: 01 9f 3e 65 05 75 57 c0 b5 08 7f b4 d3 ee 3b 6b  .Ÿ>e.uWÀµ.□´Óî;k
0010: 39 4c 42 79 b1 d7 89 33 bd da c9 b1 e9 3c 62 33  9LBy±×‰3½ÚÉ±é<b3
0020: c3 7b 00 a6 4f e6 87 45 15 76 4e 6a 62 26 2d c0  Ã{.¦Oæ‡E.vNjb&-À
0030: 9d 90 86 72 af 9e 9a 5f 3d 7a 2a 92 53 66 b8 f6  □□†r¯žš_=z*’Sf¸ö
0040: 0d a9 89 e5 24 1d 38 b8 1b 01 0d 55 a6 59 c4 e5  .©‰å$.8¸...U¦YÄå
0050: 53 2f 2a a9 53 cf 68 cf 20 ae 4b cd 9e 26 f1 60  S/*©SÏhÏ ®KÍž&ñ`
0060: 05 4a 58 29 7a c8 43 7f d1 46 c8 e9 e0 66 ee ff  .JX)zÈC□ÑFÈéàfîÿ
0070: 64 93 6c 3d 12 cf 3b 78 42 3d 39 77 3e ef 42 e9  d"l=.Ï;xB=9w>ïBé
Signature Private Key (Encrypted under LMK) =
0000: b9 02 c5 03 07 7e e3 df ff 12 dc 4f 35 a9 96 22  ¹.Å..~ãßÿ.ÜO5©–"
0010: 11 58 a5 59 72 a7 cb 93 0c 39 93 c9 fd 70 07 6d  .X¥Yr§Ë".9"Éýp.m
0020: f6 3d 62 c1 bf 00 65 86 42 84 72 9b de ad 89 8a  ö=bÁ¿.e†B„r›Þ-‰Š
0030: 3f 58 94 ce b2 7e 66 e5 15 d7 e1 6a 91 6f f1 96  ?X"Î²~få.×áj‘oñ–
0040: 89 bf ac 5f 97 54 b1 ef 4a 36 2a 1a 53 6f 3f 82  ‰¿¬_—T±ïJ6*.So?‚
0050: aa 31 f9 f7 1a 95 ec e9 dd 70 76 ce 7e e5 1c b4  ª1ù÷.•ìéÝpvÎ~å.´
0060: 70 fe a8 a8 6d cc e8 25 e8 6e 4e d7 7a 0a 71 22  pþ¨¨mÌè%ènN×z.q"
0070: 25 5d 11 1c d2 da ee da fc 5a 92 93 39 39 2c 77  %]..ÒÚîÚüZ’"99,w
0080: 93 a2 c9 47 ed 0a f1 7c 4f 15 a6 0f c7 f4 36 e4  "¢ÉGí.ñ|O.¦.Çô6ä
0090: 60 d0 38 4f 5f b2 43 3c 01 13 57 44 9a ba 8d 94  `Ð8O_²C<..WDš°□"
00a0: 95 98 e0 a5 ee d3 8c d8 8f 29 93 af 62 e0 0b e2  •˜à¥îÓŒØ□)"¯bà.â
00b0: 12 b7 07 76 05 bc bb 0d d2 eb 87 b1 dc d8 b8 12  .·.v.¼».Òë‡±ÜØ¸.
00c0: 26 b3 ea 58 58 d1 1f 3b c1 66 dc 75 68 7c ef 64  &³êXXÑ.;ÁfÜuh|ïd
00d0: 4d 46 61 f8 c3 74 ac 76 b4 42 d1 9b d7 d4 63 dc  MFaøÃt¬v´BÑ›×ÔcÜ
00e0: c2 a0 e7 ca 7c 4e 7a 09 57 da fa 3e a6 c8 50 4e  Â çÊ|Nz.WÚú>¦ÈPN
00f0: bc 49 37 97 c1 89 67 26 07 2c 3c 1c 1b 8a 53 a4  ¼I7—Á‰g&.,<..ŠS¤
0100: ab f1 63 f1 9d 9e ed 59 ef 62 e3 75 22 13 bd 46  «ñcñ□žíYïbãu".½F
0110: 10 bc ed ce 38 78 03 72 f5 d1 2d 1c df 62 42 73  .¼íÎ8x.rõÑ-.ßbBs
0120: 52 53 2c 67 84 ae 7b a7 3b b3 ad 1c 33 ee 6e e4  RS,g„®{§;³-.3înä
0130: a0 f2 62 ae 82 de f1 80 7a 69 8c 68 27 fa 4a 45   òb®‚Þñ€ziŒh'úJE
0140: 73 e2 b4 7c 12 13 9c d3 bc 23 72 9a c3 42 91 20  sâ´|..œÓ¼#ršÃB‘
0150: 8b 1b 9a f3 1f 7d 37 9d                          ‹.šó.}7□
```

5.5.2 Sample Data Generation Procedure (1024-bit RSA keys)

1. DER encode ASN.1 format DES Key

```
30 12
      03 10
            7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C
```

2. Pad #1 using PKCS#1 OAEP

```
0000: 00 D8 BF 0E 30 FB 8A E3 98 9E 6D 47 CE A8 05 A9 .Ø¿.0ûŠã˜žmGÎ¨.©
0010: 22 D9 19 4A 2C 3B AC 74 11 CA DE CD C8 7F 78 A6 "Ù.J,;¬t.ÊÞÍÈ?x¦
0020: 64 E5 DC 33 CE 37 72 85 A7 62 7A 37 FE C8 8E 8C dåÜ3Î7r…§bz7þÈŽŒ
0030: 21 21 E0 F9 64 73 24 9C B2 07 7A A3 60 B4 DB ED !!àùds$œ².z£`´Ûí
0040: 52 EF 02 89 3B C8 E9 A9 C3 E6 E8 AB 29 EC 20 D1 Rï.‰;Èé©Ãæè«)ì Ñ
0050: F0 CA 0B AF 7D 6A D3 C7 90 D3 F3 73 75 63 AC A6 ðÊ.¯}jÓÇ?Óósuc¬¦
0060: D4 D4 91 FA D8 5F 23 6B 7C 2A 07 E1 09 C9 5D AC ÔÔ‘úØ_#k|*.á.É]¬
0070: 16 2C C9 70 12 B0 65 F0 40 A1 02 A2 8A 6A 4C 45 .,Ép.°eð@¡.¢ŠjLE
```

3. Encrypt #2 using Encryption Public Key

```
0000: 0F 7B 55 85 C7 26 FB 5F 81 45 9F CC 7C 2A CD FC .{U…Ç&û_?EŸÌ|*Íü
0010: B2 D4 62 8F CB 86 00 5C 62 47 F9 25 B0 7F FF 05 ²Ôb?Ë†.\bGù%°?ÿ.
0020: 2F D0 03 7A B6 8C 24 DF 98 BB 65 78 D3 40 1B 8B /Ð.z¶Œ$ß˜»exÓ@.‹
0030: 19 17 9F 5E 92 32 BB D0 87 C5 5F 52 A3 BF 3F E8 ..Ÿ^’2»Ð‡Å_R£¿?è
0040: 81 6A 75 C7 19 2D 5F 0D 74 39 23 DA 81 0B 96 FC ?juÇ.-_.t9#Ú?.–ü
0050: 43 B8 55 B2 FB 60 F3 E8 B3 45 3E 89 43 4E 40 17 C¸U²û`óè³E>‰CN@.
0060: DC 03 66 C8 8D C6 9E 17 A4 C5 89 54 0B 91 11 9A Ü.fÈ?Æž.¤Å‰T.‘.š
0070: 1D 0B 04 22 BC B3 55 29 8D BF 0B 80 AC C9 BD 1C ..."¼³U)?¿.€¬É½.
```

4. Insert #3 into data block for signing

```
0000: 59 59 59 59 59 59 59 59 59 59 59 59 0F 7B 55 85 YYYYYYYYYYYY.{U…
0010: C7 26 FB 5F 81 45 9F CC 7C 2A CD FC B2 D4 62 8F Ç&û_?EŸÌ|*Íü²Ôb?
0020: CB 86 00 5C 62 47 F9 25 B0 7F FF 05 2F D0 03 7A Ë†.\bGù%°?ÿ./Ð.z
0030: B6 8C 24 DF 98 BB 65 78 D3 40 1B 8B 19 17 9F 5E ¶Œ$ß˜»exÓ@.‹..Ÿ^
0040: 92 32 BB D0 87 C5 5F 52 A3 BF 3F E8 81 6A 75 C7 ’2»Ð‡Å_R£¿?è?juÇ
0050: 19 2D 5F 0D 74 39 23 DA 81 0B 96 FC 43 B8 55 B2 .-_.t9#Ú?.–üC¸U²
0060: FB 60 F3 E8 B3 45 3E 89 43 4E 40 17 DC 03 66 C8 û`óè³E>‰CN@.Ü.fÈ
0070: 8D C6 9E 17 A4 C5 89 54 0B 91 11 9A 1D 0B 04 22 ?Æž.¤Å‰T.‘.š..."
0080: BC B3 55 29 8D BF 0B 80 AC C9 BD 1C 5A 5A 5A 5A ¼³U)?¿.€¬É½.ZZZZ
0090: 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A                ZZZZZZZZZZZ
```

5. Generate hash over #4

```
0000: 91 6C E5 4D EB 72 0C 65 DE E0 32 3F 62 01 61 32 ‘låMër.eÞà2?b.a2
0010: E5 49 DC A4                                     åIÜ¤
```

6. DER encode ASN.1 format #5

```
30 21
   30 09
      06 05
         2B 0E 03 02 1A
      05 00
   04 14
      91 6C E5 4D EB 72 0C 65 DE E0 32 3F 62 01 61 32 E5 49 DC A4
Pad #6 using PKCS#1 v1.5
0000: 00 01 FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0050: FF FF FF FF FF FF FF FF FF FF FF FF 00 30 21 30 ÿÿÿÿÿÿÿÿÿÿÿÿ.0!0
0060: 09 06 05 2B 0E 03 02 1A 05 00 04 14 91 6C E5 4D ....+.......‘låM
0070: EB 72 0C 65 DE E0 32 3F 62 01 61 32 E5 49 DC A4 ër.eÞà2?b.a2åIÜ¤
```

7. Encrypt #7 using Signature Private Key:

```
0000: 08 60 4D 82 4C B2 A7 3F 9F E7 9A 73 5B AB 66 AC .`M,L²§?Ÿçšs[«f¬
0010: 3C A0 85 33 BB F7 77 31 AD 36 C0 01 88 2D 2C 27  < …3»÷w1-6À.ˆ-,'
0020: D6 5D 2C 52 D1 B6 27 CF 92 FF 67 64 4C E4 3A A6 Ö],RÑ¶'Ï’ÿgdLä:¦
0030: 43 22 E6 D6 92 93 C3 D0 C6 2C B6 B2 E5 2D 41 E1 C"æÖ’"ÃÐÆ,¶²å-Aá
0040: 90 77 F8 D8 AB 5F D8 2C 4C C9 CC F5 E8 48 7E 08 ?wøØ«_Ø,LÉÌõèH~.
0050: 4B 2B BA E5 F8 A8 EF 19 76 A5 3D 53 6E 00 8A 88 K+°åø¨ï.v¥=Sn.Šˆ
0060: 33 75 7B B5 66 0D D3 40 CB A2 EE 49 CB 5D 44 C4 3u{µf.Ó@Ë¢îIË]DÄ
0070: 1D 01 13 08 42 2E 2F A6 67 22 6E 84 D7 8D 0D DC B./¦g"n„×?.Ü
```

5.5.3 Sample Data Calculation (1024-bit RSA keys)

Field	Length & Type	Value
COMMAND MESSAGE		
Message Header	m A	
Command Code	2 A	"GK" – Export Key under RSA Public Key
Encryption Identifier	2 N	"01" – RSA Encryption
Pad Mode Identifier	2 N	"02" – OAEP (EME-OAEP-ENCODE)
Mask Generation Function	2 N	"01" – MGF1 as defined in PKCS#1 v2.2

Field	Length & Type	Value
MGF Hash Function	2 N	"01" – SHA-1
OAEP Encoding Parameters Length	2 N	"04"
OAEP Encoding Parameters	N B	"09080706"
OAEP Encoding Parameters Delimiter	1 A	";"
DES Key Type	4 N	"0600"
Signature Indicator	1 A	"="
Signature Hash Identifier	2 N	"01" – SHA-1
Signature Identifier	2 N	"01" – RSA
Signature Pad Mode Identifier	2 N	"01" – PKCS#1 v1.5 padding
Header Data Block Length	4 N	"0012"
Header Data Block	n B	"YYYYYYYYYYYY"
Delimiter	1 A	";"
Footer Data Block Length	4 N	"0015"
Footer Data Block	n B	"ZZZZZZZZZZZZZZZ"
Delimiter	1 A	";"
Private Key Flag	2 N	"99"
Private Key Length	4 N	"0344"
Private Key	n B	b902c503077ee3dfff12dc4f35a996221158a55972a7cb930c3993c9fd70076df63d62c1bf0065864284729bdead898a3f5894ceb27e66e515d7e16a916ff19689bfac5f9754b1ef4a362a1a536f3f82aa31f9f71a95ece9dd7076ce7ee51cb470fea8a86dcce825e86e4ed77a0a7122255d111cd2daeedafc5a929339392c7793a2c947ed0af17c4f15a60fc7f436e460d0384f5fb2433c011357449aba8d949598e0a5eed38cd88f2993af62e00be212b7077605bcbb0dd2eb87b1dcd8b81226b3ea5858d11f3bc166dc75687cef644d4661f8c374ac76b442d19bd7d463dcc2a0e7ca7c4e7a0957dafa3ea6c8504ebc493797c1896726072c3c1c1b8a53a4abf163f19d9eed59ef62e3752213bd4610bcedce38780372f5d12d1cdf62427352532c6784ae7ba73bb3ad1c33ee6ee4a0f262ae82def1807a698c6827fa4a4573e2b47c12139cd3bc23729ac34291208b1b9af31f7d379d
DES Key Flag	1 N	"1"
DES Key (LMK)	1A+32H	"U513ECF7E7D0A5554900FA709B9A5F721"
Check Value	16 H	"8B1536D4D60FEC19"
MAC	4 B	0EFA9C3A
Public Key	n B	30818902818100 40CE04F85B703049C8486C84C8EB4054ECB22AFD6B78967DA2D8CDF75E5EB963949543C68F54318811B3A99127A8C1D09CB13CCD7005808E918080B0AF5A58C7118B443FC7CDAEE4D5A7F63CC0F3596C98E37B9F979D53C44BE10EC4068FDD7A3824BD2034F0E5EA198EFF9B36B0EF6590BFD05099E28AE04D0996D2E13621E90203010001
Authentication Data	n A	"ABCDEFG"
Delimiter	1 A	";"
Key Data Block Type	2 N	"02" – Key Block Data Template
Key Data Block Template Length	4 N	"0020"
Key Data Block Template	n H	"3012031000000000000000000000000000000000"
Delimiter	1 A	";"
DES Key Offset	4 N	"0004"

Field	Length & Type	Value
Check Value Length	2 N	"00"
Check Value Offset	4 N	"0000"
End Message Delimiter	1 C	
Message Trailer	n A	
RESPONSE MESSAGE		
Message Header	m A	
Response Code	2 A	GL
Error Code	2 A	"00" – No error
Initialization Value	16 H	"????????????????"
DES Key Length	4 N	"0128"
DES Key (PK)	n B	0f7b5585c726fb5f81459fcc7c2acdfcb2d4628fcb86005c6247f925b07fff052fd0037ab68c24df98bb6578d3401b8b19179f5e9232bbd087c55f52a3bf3fe8816a75c7192d5f0d743923da810b96fc43b855b2fb60f3e8b3453e89434e4017dc0366c88dc69e17a4c589540b91119a1d0b0422bcb355298dbf0b80acc9bd1c
Signature Length	4 N	"0128"
Signature	n B	08604d824cb2a73f9fe79a735bab66ac3ca08533bbf77731ad36c001882d2c27d65d2c52d1b627cf92ff67644ce43aa64322e6d69293c3d0c62cb6b2e52d41e19077f8d8ab5fd82c4cc9ccf5e8487e084b2bbae5f8a8ef1976a53d536e008a8833757bb5660dd340cba2ee49cb5d44c41d011308422e2fa667226e84d78d0ddc
End Message Delimiter	1 C	
Message Trailer	n A	

Note: PKCS#1's OAEP padding involves a random input, and therefore is not predictable. The value of an encrypted DES key and its signature will be different each time this command is run, even if the DES key and RSA keys remain the same. The values in the table above should only be used to indicate how an HSM's internal processing works: do not expect to get the same encrypted key and signature as shown above!

5.5.4 Sample Data Definitions (2048-bit RSA keys)

```
DES Key to export = 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C

DES Key Type = ZPK

DES Key Check Value = 8B 15 36 D4 D6 0F EC 19

LMK Encrypted DES Key (as ZPK) in key scheme U = U 51 3E CF 7E 7D 0A 55
54 90 0F A7 09 B9 A5 F7 21

Encryption Padding mode = OAEP (MGF1, SHA-1, OAEP Params = "09 08 07 06")

Encryption Public Key (ASN.1 DER encoded) =
0000: 30 82 01 0A 02 82 01 01 00 40 F5 B5 3A 32 47 50  0,...,...@õµ:2GP
0010: 90 AD DE 8A 33 EF 6F 9F 95 EE 58 41 84 F1 FE 7D  □-ÞŠ3ïoŸ•îXA„ñþ}
0020: 31 AE 58 C6 CA E2 A0 62 DA B0 39 E7 45 03 9F 7A  1®XÆÊâ bÚ°9çE.Ÿz
0030: A4 3A 23 6A 45 40 AD 0A 3F 36 1F 90 5F 29 B1 A0  ¤:#jE@-.?6.□_)±
```

```
0040: E9 C1 65 B0 A6 56 D9 F8 18 C5 7E 05 D7 87 F1 F8   éÁe°¦VÙø.Å~.×‡ñø
0050: EE 0C 6B E9 B4 2A 83 2C FD BD 35 74 A5 E7 EF 86   î.ké´*ƒ,ý½5t¥çï†
0060: 3B 11 FF 3F 95 DC AC BB DC FD D9 0E A9 C7 D5 52   ;.ÿ?•Ü¬»ÜýÙ.©ÇÕR
0070: CF 0D 1B E4 71 53 2C F6 4E 02 9C CD D7 D9 AE C9   Ï..äqS,öN.œÍ×Ù®É
0080: 2F 90 C3 9D 1B E9 63 AE E6 F5 78 E6 A3 D8 3A C6   /□Ã□.éc®æõxæ£Ø:Æ
0090: 57 2E B5 52 5D 0A 81 79 BC BA 02 63 D8 2E BB 5D  W.µR].□y¼°.cØ.»]
00A0: 77 C3 35 56 16 06 B3 27 01 75 3A ED C4 2A AA E1   wÃ5V..³'.u:íÄ*ªá
00B0: 20 D9 23 3C B3 B9 EF 4D DE 0C 7B 07 02 25 3E 25    Ù#<³¹ïMÞ.{..%>%
00C0: 5C 1E DE 05 2D 9C 4F 7A 6C 7A FB F4 E4 08 BB 18   \.Þ.-œOzlzûôä.».
00D0: D9 53 54 C4 44 62 FF 35 2B FF 31 1E 8E 07 A3 83   ÙSTÄDbÿ5+ÿ1.Ž.£ƒ
00E0: 49 A7 88 08 16 4A 24 A0 8F D9 D3 54 9B C3 3D 58   I§ˆ..J$ □ÙÓT›Ã=X
00F0: DE C5 25 E9 7F FB FB F8 A6 D4 12 05 12 0A E8 D4   ÞÅ%é□ûûø¦Ô....èÔ
0100: AC 75 6F 3E 85 80 91 8D CB 02 03 01 00 01         ¬uo>…€‘□Ë.....

Encryption Public Key MAC = AA 5E 41 CC

Encryption Public Key Authentication Data = "ABCDEFG"

Encryption Private Key (d) =
0000: 07 B7 88 C7 78 98 9B 3C 0C C3 AE A4 5B D1 E5 61   .·ˆÇx˜›<.Ã®¤[Ñåa
0010: F0 D6 20 36 74 6F 18 9D 51 CA 6F 17 44 13 EC 9A   ðÖ 6to.□QÊo.D.ìš
0020: 71 2B F7 CA ED 92 C1 05 88 78 93 93 D5 8A 98 F8   q+÷Êí’Á.ˆx""ÕŠ˜ø
0030: 88 6B F8 81 2D 99 51 F5 E3 09 3B 12 8F A7 C6 3E   ˆkø□-™Qõã.;.□§Æ>
0040: DF 1B 49 03 61 3D 80 26 7B 68 48 73 A4 47 40 0D   ß.I.a=€&{hHs¤G@.
0050: 86 B0 36 82 CD 0A 59 E6 63 8D 70 96 D3 87 DB AB   †°6,Í.Yæc□p–Ó‡Û«
0060: 75 A6 97 04 D9 5E 00 BF E3 1D 48 A6 A3 CC 68 18   u¦—.Ù^.¿ã.H¦£Ìh.
0070: 3D 5C 36 61 E9 94 C7 86 B4 8A 60 7C 23 DE 39 35   =\6aé"Ç†´Š`|#Þ95
0080: 4D 19 5D DA 82 71 06 4F C7 73 82 A0 F9 AB 9F 10   M.]Ú,q.OÇs, ù«Ÿ.
0090: 25 3B 43 74 B6 5B 29 79 0C 34 C2 FF 82 2C D9 62   %;Ct¶[)y.4Âÿ,,Ùb
00A0: 45 1B 66 0E 43 67 56 A1 E1 9E 04 E8 D1 FD 23 72   E.f.CgV¡áž.èÑý#r
00B0: 4D 81 1D 61 54 22 95 2B 66 05 A5 A2 CD 20 6F 74   M□.aT"•+f.¥¢Í ot
00C0: 18 18 FA E4 BD 0F 6E D9 24 64 F6 A7 C6 16 5E 89   ..úä½.nÙ$dö§Æ.^‰
00D0: 46 85 00 12 9F 45 50 CA 6F C9 A9 3B AC E4 2F 39   F….ŸEPÊoÉ©;¬ä/9
00E0: BA B7 63 7E 8B BB 97 4C D7 A3 90 C8 39 62 85 AE   º·c~‹»—L×£□È9b…®
00F0: B1 AC 82 E0 70 F8 84 05 53 99 F6 7E 33 EA 76 F1   ±¬,àpø„.S™ö~3êvñ

Signature Public Key (ASN.1 DER encoded) =
0000: 30 82 01 0A 02 82 01 01 00 40 BD C2 92 8D CF A2   0,...,...@½Â’□Ï¢
0010: 96 A7 B3 F2 FE 3E 78 C7 24 21 F3 B2 89 A0 CA 5D   –§³òþ>xÇ$!ó²‰ Ê]
0020: 6E E5 9F 4F 14 76 59 95 64 32 D2 30 31 42 8D 9B   nåŸO.vY•d2Ò01B□›
0030: 83 99 99 1A DB E3 B7 10 17 4C 12 87 1F 53 49 05   ƒ™™.Ûã·..L.‡.SI.
0040: 82 46 F3 94 22 32 D2 2F 02 B5 AF D3 C0 E9 1A 28   ,Fó""2Ò/.µ¯ÓÀé.(
0050: 51 13 96 3A 66 05 98 D9 00 40 E8 CB 48 71 C4 48   Q.–:f.˜Ù.@èËHqÄH
0060: BA FA 17 46 38 DB 3C 35 BA F5 CA 5D C5 6D B9 3D   ºú.F8Û<5ºõÊ]Åm¹=
0070: E2 5A C2 A5 C3 D7 49 1F 5B A7 02 98 F0 42 3A DE   âZÂ¥Ã×I.[§.˜ðB:Þ
0080: 55 06 9B 0B 40 B4 38 78 E0 55 76 B0 8F 71 A8 EE   U.›.@´8xàUv°□q¨î
0090: 10 B1 F7 52 4D A7 DF 52 C4 4B 23 0E 31 E0 B1 E4   .±÷RM§ßRÄK#.1à±ä
00A0: B5 A1 E4 B4 7C 94 E9 E5 30 F6 DB 57 70 7E 53 5A   µ¡ä´|"éå0öÛWp~SZ
00B0: 5F BF B0 32 21 44 B5 04 B2 8D 22 B7 21 E4 D5 98   _¿°2!Dµ.²□"·!äÕ˜
00C0: 10 17 3B 2B 71 4A 70 3B B3 6F FC B8 9A D5 33 32   ..;+qJp;³oü¸šÕ32
00D0: 60 91 FB 41 A6 D4 FB 71 F4 12 5E 80 33 1D BC E7   `‘ûA¦Ôûqô.^€3.¼ç
00E0: 53 EC 13 4F 9B 3D A9 E7 77 FF 89 54 CC 25 A8 FF   Sì.O›=©çwÿ‰TÌ%¨ÿ
```

```
00F0: C7 D3 E0 2F C7 03 F0 CF 0F 18 C8 8D 1F 5D 35 32  ÇÓà/Ç.ðÏ..È□.]52
0100: BB 17 DB 94 63 03 77 34 73 02 03 01 00 01        ».Û"c.w4s.....

SIGNATURE PRIVATE KEY (D) =
0000: 01 A1 65 CD 90 11 BB 1C 05 34 34 79 EF B3 D5 FC  .¡eÍ□.»..44yï³Õü
0010: 14 78 D1 35 C3 1D 65 95 FD E5 71 B5 E7 B7 20 DA  .xÑ5Ã.e•ýåqµç· Ú
0020: 89 A7 1E 7C 97 1A FE E0 25 15 A4 86 06 29 9D 97  ‰§.|—.þà%.¤†.)□—
0030: A0 9C 54 D7 D6 9E 9F AB 64 C3 0C A7 81 D5 26 46   œT×ÖžŸ«dÃ.§□Õ&F
0040: F0 B1 71 69 49 D5 95 4F 59 69 6E A6 14 1D 01 D6  ð±qiIÕ•OYin¦...Ö
0050: 0E 4C 6E 96 2F FB 4C 03 9D 79 C9 94 73 FD 03 B3  .Ln–/ûL.□yÉ"sý.³
0060: 66 2E 47 07 4A 58 A0 74 DB 69 4C 88 6E 9B 12 55  f.G.JX tÛiLˆn›.U
0070: 9A 12 A8 2C 60 D6 9F B3 CF 7B 47 20 C5 89 28 8E  š.¨,`ÖŸ³Ï{G Å‰(Ž
0080: 23 4C D6 97 21 D0 19 DF 7E A3 05 AC F2 2E 0C E9  #LÖ—!Ð.ß~£.¬ò..é
0090: 59 7C 98 1B DF AA D7 09 69 D3 04 CA 09 87 4E 52  Y|˜.ßª×.iÓ.Ê.‡NR
00A0: C5 62 CE 85 39 2B 45 8E 6E E3 AA DD E1 85 BD 62  ÅbÎ…9+EŽnãªÝá…½b
00B0: 4F 0A 0F 75 29 32 30 4A 4B 07 7C AA 4A C8 50 48  O..u)20JK.|ªJÈPH
00C0: 0A EE 65 55 89 6B 2D 98 40 33 61 1D DA 24 F4 E8  .îeU‰k-˜@3a.Ú$ôè
00D0: 08 79 D5 AA 5F 4B F8 EB FB 98 DC B9 CB CD 09 AC  .yÕª_Køëû˜Ü¹ËÍ.¬
00E0: 78 A0 10 A0 96 5A 63 59 D5 03 2C F3 68 64 D8 B0  x . –ZcYÕ.,óhdØ°
00F0: 05 D2 74 6B 18 A5 0A FB F9 AF 84 31 DB 5B 41 39  .Òtk.¥.ûù¯„1Û[A9

Signature Private Key (Encrypted under LMK) =
0000: 14 F7 B6 45 05 43 83 DA 0C 52 C6 5F 91 86 ED 10  .÷¶E.CƒÚ.RÆ_‘†í.
0010: F6 88 11 F2 8D 0F 69 98 00 3F A6 48 A7 5C E5 12  öˆ.ò□.i˜.?¦H§\å.
0020: 34 59 53 EB 41 5B B1 DE C9 21 F5 DD B3 E1 5D 47  4YSëA[±ÞÉ!õÝ³á]G
0030: 5D 8A 27 E4 C8 A9 48 CC 36 F8 C2 5D 84 7B 42 BA  ]Š'äÈ©HÌ6øÂ]„{Bº
0040: 66 55 1F 66 DA 86 02 B1 8C B0 F9 F0 B4 90 0E 1C  fU.fÚ†.±Œ°ùð´□..
0050: 40 DB BA 12 D3 E2 E7 D1 44 AF 02 97 70 38 8A F9  @Ûº.ÓâçÑD¯.—p8Šù
0060: BC 88 D9 FC 29 6B 1F A8 88 44 47 6D 5E B8 FC 70  ¼ˆÙü)k.¨ˆDGm^¸üp
0070: 90 8D F0 FC B8 18 B7 BB 48 67 67 D1 CB E4 BA 92  □□ðü¸.·»HggÑËäº’
0080: 77 D7 50 08 84 4E EB FE 38 E6 39 1C DB C8 D4 1C  w×P.„Nëþ8æ9.ÛÈÔ.
0090: F2 F6 5C C4 B8 1A BD A0 6D 37 13 75 06 85 A3 0A  òö\Ä¸.½ m7.u.…£.
00A0: 24 62 99 72 02 E0 F5 FE D5 BF 2A 3A 0C 25 0E 3D  $b™r.àõþÕ¿*:.%.=
00B0: 64 09 2E 7D 2A 1B 72 3B D1 57 1A 71 92 68 C3 F0  d..}*.r;ÑW.q’hÃð
00C0: DF E6 24 8A 8F 3D D9 34 43 7B 34 25 3D 06 1C 40  ßæ$Š□=Ù4C{4%=..@
00D0: 14 2E B0 F3 83 25 DE 14 3C 6F 3A E3 56 15 FC 61  ..°óƒ%Þ.<o:ãV.üa
00E0: A4 A7 A2 0D E4 F6 BF 38 66 CC 6E 80 6B DD 28 C2  ¤§¢.äö¿8fÌn€kÝ(Â
00F0: AD 96 BE F3 B3 AE 81 EE 49 5D 0E 25 90 A3 32 34  –¾ó³®□îI].%□£24
0100: 73 5F 48 A8 53 57 00 DF 51 71 E3 9B 1D 6B 35 B7  s_H¨SW.ßQqã›.k5·
0110: 28 AD FC E9 4D B6 95 DC 8D 26 9B 86 40 82 30 AA  (üéM¶•Ü□&›†@‚0ª
0120: 26 B6 C1 CC 72 48 B3 36 D5 65 18 A2 AF 09 C4 D2  &¶ÁÌrH³6Õe.¢¯.ÄÒ
0130: EB 1A FB F5 4B 91 C6 C0 BD 13 AD 0A F8 D4 EA B2  ë.ûõK‘ÆÀ½..øÔê²
0140: 3F A2 AD 0F 8C CE 1E 4B BE 2F D8 52 4E ED E0 76  ?¢.ŒÎ.K¾/ØRNíàv
0150: 90 D9 B0 C3 67 0F 71 51 3B 78 BE 32 B0 DC 28 56  □Ù°Ãg.qQ;x¾2°Ü(V
0160: 71 70 49 C0 55 1F 4E FD 5F AC 8A 46 F8 E6 02 DC  qpIÀU.Ný_¬ŠFøæ.Ü
0170: A6 55 A8 2A F9 18 2D E3 90 7D F7 6F E4 8B B9 65  ¦U¨*ù.-ã□}÷oä‹¹e
0180: 85 F6 9B FE B0 D4 37 4A 36 E0 C3 D6 D5 2D 9F 61  …ö›þ°Ô7J6àÃÖÕ-Ÿa
0190: E9 A2 5E 5E 4A A1 E4 C3 CC 74 4C B4 9E 86 4D 8B  é¢^^J¡äÃÌtL´ž†M‹
01A0: 23 6B E5 15 B2 42 5D 6F 28 0C 74 3F 4B CF B0 64  #kå.²B]o(.t?KÏ°d
01B0: 95 82 E9 9A 06 FB 78 ED F7 5D AE 1F D9 D3 FA 66  •‚éš.ûxí÷]®.ÙÓúf
01C0: B9 40 7C F2 B4 4A 6F 6E 3F AB 21 49 0A 30 91 92  ¹@|ò´Jon?«!I.0‘’
01D0: 5B 0D DF F0 98 9D 62 58 9A 4F 23 06 1A 2C DB 41  [.ßð˜□bXšO#..,ÛA
```

```
01E0: CF 34 3D 89 46 EB 39 02 81 9A 81 57 AC B2 B2 BF  Ï4=‰Fë9.□š□W¬²²¿
01F0: 58 7F 84 CF D6 7B FE 91 DB 9A FB BF A3 01 FC B5  X□„ÏÖ{þ‘Ûšû¿£.üµ
0200: 7B C5 12 64 BC 3A 75 8E DF 57 04 B6 A1 69 18 A6  {Å.d¼:uŽßW.¶¡i.¦
0210: 6F E4 E8 67 78 42 B9 08 0E 5C D1 CF 7E E2 FD 7A  oäègxB¹..\ÑÏ~âýz
0220: A7 AC 3B 4E 41 39 3F A2 D5 14 B9 76 1A 22 A5 FD  §¬;NA9?¢Õ.¹v."¥ý
0230: 80 9A 21 06 D9 9B 40 6A 1C 5B E7 A7 F9 97 7A 98  €š!.Ù›@j.[ç§ù—z˜
0240: 8B AA 1A 37 54 1C D7 D3 8D B0 64 88 AE C8 0E 0A  ‹ª.7T.×Ó□°d^®È..
0250: 9C D3 18 17 7B C1 09 DB 87 68 EA 57 1F 38 4E 45  œÓ..{Á.Û‡hêW.8NE
0260: 7F 80 08 B8 05 5A 6B 4B 82 98 F8 83 B5 1D 72 40  □€.¸.ZkK,˜øƒµ.r@
0270: 12 A8 F5 DD 87 8D 58 50 40 BF E5 2C 5C 00 61 22  .¨õÝ‡□XP@¿å,\.a"
0280: B4 8B 14 E9 13 2E B2 61 75 77 7B 9F 74 B8 F7 3F  ´‹.é..²auw{Ÿt¸÷?
```

5.5.5 Sample Data Generation Procedure (2048-bit RSA keys)

1. DER encode ASN.1 format DES Key

```
30 12
   03 10
      7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C 7C
```

2. Pad #1 using PKCS#1 OAEP

```
0000: 00 DF F6 83 6C EA D3 47 BE C2 3C 06 6D 66 19 9E .ßöƒlêÓG¾Â<.mf.ž
0010: 14 83 36 95 7B 3B AC 74 11 CA DE CD C8 7F 78 A6 .ƒ6•{;¬t.ÊÞÍÈ?x¦
0020: 64 E5 DC 33 CE 37 72 85 A7 62 7A 37 FE C8 8E 8C dåÜ3Î7r…§bz7þÈŽŒ
0030: 21 21 E0 F9 64 73 24 9C B2 07 7A A3 60 B4 DB ED !!àùds$œ².z£`´Ûí
0040: 52 EF 02 89 3B C8 E9 A9 C3 E6 E8 AB 29 EC 20 D1 Rï.‰;Èé©Ãæè«)ì Ñ
0050: F0 CA 0B AF 7D 6A D3 C7 90 D3 F3 73 75 63 AC A6  ðÊ.¯}jÓÇ?Óósuc¬¦
0060: D4 D4 91 FA D8 5F 23 6B 7C 2A 07 E0 39 DB 5E BC ÔÔ‘úØ_#k|*.à9Û^¼
0070: 6A 50 B5 0C 6E CC 19 8C 3C DD 7E DE F6 16 30 39 jPµ.nÌ.Œ<Ý~Þö.09
0080: 25 5A DD 34 C9 BF 2B 53 98 43 9D B8 1E 0C AF E8  %ZÝ4É¿+S˜C?¸..¯è
0090: E0 38 6F F8 31 CE 18 5B 17 BA F0 47 83 EE D7 7C à8oø1Î.[.°ðGƒî×|
00A0: F9 CD 37 C9 9A AD D1 FF 12 8D 27 9F 03 7D F4 41  ùÍ7Éš-Ñÿ.?'Ÿ.}ôA
00B0: A8 3B 9C F8 C4 5E 80 F3 2D AF 6F 20 F1 81 7E 07  ¨;œøÄ^€ó-¯o ñ~?.
00C0: 74 E4 0C E8 C0 2A FA D7 BD 22 18 FF BA DB F1 E1 tä.èÀ*ú×½".ÿ°Ûñá
00D0: 5D AB 8A 0A CC A1 6C 72 FB 8A 5E 8A 13 5B EC 2F ]«Š.Ì¡lrûŠ^Š.[ì/
00E0: A8 8A 55 93 F4 56 CF A4 2E 84 23 1D 7B 8E 4F 54 ¨ŠU"ôVÏ¤.„#.{ŽOT
00F0: 4D AF 58 F9 FF FA 1F 9C 9D F4 A8 5D 2B 2E FA 12  M¯Xùÿú.œô?¨]+.ú.
```

3. Encrypt #2 using Encryption Public Key

```
0000: 32 27 D3 03 4C 27 B6 48 B7 AE 68 53 77 17 50 62 2'Ó.L'¶H·®hSw.Pb
0010: AF 19 22 24 71 72 E1 E7 51 5C 11 43 81 09 54 FC  ¯."$qráçQ\.C?.Tü
0020: D4 46 38 AC 96 98 DE 90 AC EC 0E 0A 97 77 93 CA  ÔF8¬–˜Þ?¬ì..—w"Ê
0030: 8C 35 41 7E 0C C9 2B 6A 32 AB C6 60 C7 34 AA 7D Œ5A~.É+j2«Æ`Ç4ª}
0040: FA F8 29 91 71 20 4F 13 7C F9 98 10 91 2B 34 44 úø)‘q O.|ù˜.‘+4D
0050: 9D 7A 39 30 E6 04 13 5D 12 64 D7 0E C5 68 78 E1  ?z90æ..].d×.Åhxá
0060: C2 1F 42 C8 EF DE 21 1C CC 78 1D 84 97 96 72 65 Â.BÈïÞ!.Ìx.„—–re
0070: 85 9C 3B E9 3A DF F0 B5 DC A7 9D 53 EF E8 6E A6  …œ;é:ßðµÜ§?Sïèn¦
0080: 14 61 9A FB C0 6A AD FF 66 C5 D6 BD E3 E0 A4 C0 .ašûÀj-ÿfÅÖ½ãà¤À
0090: B2 08 4D 30 2B 28 96 65 4E F9 36 40 46 45 22 70 ².M0+(–eNù6@FE"p
00A0: C5 11 AE 6B 03 B1 1B 94 4C 8E FC BE 12 40 E5 95 Å.®k.±."LŽü¾.@å•
00B0: E4 32 AE 8C 9B A7 BE 13 5E 6A 29 60 F9 65 77 F9 ä2®Œ›§¾.^j)`ùewù
00C0: 76 34 AC B1 C7 42 C6 28 35 58 BB 68 1A 61 2D FF v4¬±ÇBÆ(5X»h.a-ÿ
00D0: 90 82 48 F6 C9 D4 E8 04 76 B5 94 C7 2A E1 47 F0  ?,HöÉÔè.vµ"Ç*áGð
```

```
00E0: 02 2C CD FB 9B 63 63 1B 74 D1 99 AF B3 85 26 7C .,Íû›cc.tÑ™¯³…&|
00F0: 22 3C EF FD 7C EA E8 D1 E0 4A 9E A1 27 FD CB 58 "<ïý|êèÑàJž¡'ýËX
```

4. Insert #3 into data block for signing

```
0000: 59 59 59 59 59 59 59 59 59 59 59 59 32 27 D3 03 YYYYYYYYYYYY2'Ó.
0010: 4C 27 B6 48 B7 AE 68 53 77 17 50 62 AF 19 22 24 L'¶H·®hSw.Pb¯."$
0020: 71 72 E1 E7 51 5C 11 43 81 09 54 FC D4 46 38 AC qráçQ\.C.?TüÔF8¬
0030: 96 98 DE 90 AC EC 0E 0A 97 77 93 CA 8C 35 41 7E  –˜Þ?¬ì..—w"ÊŒ5A~
0040: 0C C9 2B 6A 32 AB C6 60 C7 34 AA 7D FA F8 29 91 .É+j2«Æ`Ç4ª}úø)‘
0050: 71 20 4F 13 7C F9 98 10 91 2B 34 44 9D 7A 39 30  q O.|ù˜.‘+4D?z90
0060: E6 04 13 5D 12 64 D7 0E C5 68 78 E1 C2 1F 42 C8 æ..].d×.ÅhxáÂ.BÈ
0070: EF DE 21 1C CC 78 1D 84 97 96 72 65 85 9C 3B E9 ïÞ!.Ìx.„—–re…œ;é
0080: 3A DF F0 B5 DC A7 9D 53 EF E8 6E A6 14 61 9A FB  :ßðµÜ§?Sïèn¦.ašû
0090: C0 6A AD FF 66 C5 D6 BD E3 E0 A4 C0 B2 08 4D 30 Àj-ÿfÅÖ½ãà¤À².M0
00A0: 2B 28 96 65 4E F9 36 40 46 45 22 70 C5 11 AE 6B +(–eNù6@FE"pÅ.®k
00B0: 03 B1 1B 94 4C 8E FC BE 12 40 E5 95 E4 32 AE 8C .±."LŽü¾.@å•ä2®Œ
00C0: 9B A7 BE 13 5E 6A 29 60 F9 65 77 F9 76 34 AC B1 ›§¾.^j)`ùewùv4¬±
00D0: C7 42 C6 28 35 58 BB 68 1A 61 2D FF 90 82 48 F6  ÇBÆ(5X»h.a-ÿ?‚Hö
00E0: C9 D4 E8 04 76 B5 94 C7 2A E1 47 F0 02 2C CD FB ÉÔè.vµ"Ç*áGð.,Íû
00F0: 9B 63 63 1B 74 D1 99 AF B3 85 26 7C 22 3C EF FD ›cc.tÑ™¯³…&|"<ïý
0100: 7C EA E8 D1 E0 4A 9E A1 27 FD CB 58 5A 5A 5A 5A |êèÑàJž¡'ýËXZZZZ
0110: 5A 5A 5A 5A 5A 5A 5A 5A 5A 5A 5AZZZZZZZZZZZ
```

5. Generate hash over #4

```
0000: 27 E2 34 FF D1 99 2F 8A A8 56 F0 54 B5 A7 6D 81 'â4ÿÑ™/Š¨VðTµ§m?
0010: A5 80 FA 42             ¥€úB
```

6. DER encode ASN.1 format #5

```
30 21
   30 09
      06 05
         2B 0E 03 02 1A
      05 00
   04 14
      27 E2 34 FF D1 99 2F 8A A8 56 F0 54 B5 A7 6D 81 A5 80 FA 42
```

7. Pad #6 using PKCS#1 v1.5

```
0000: 00 01 FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0080: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0090: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00D0: FF FF FF FF FF FF FF FF FF FF FF FF 00 30 21 30 ÿÿÿÿÿÿÿÿÿÿÿÿ.0!0
00E0: 09 06 05 2B 0E 03 02 1A 05 00 04 14 27 E2 34 FF ...+........'â4ÿ
00F0: D1 99 2F 8A A8 56 F0 54 B5 A7 6D 81 A5 80 FA 42 Ñ™/Š¨VðTµ§m?¥€úB
```

8. Encrypt #7 using Signature Private Key:

```
0000: 19 5F 27 EF 10 CA F3 8D 81 98 30 06 38 B1 B4 7C ._'ï.Êó?˜?0.8±´|
0010: 53 DA 8C E1 96 31 0A A3 0C 43 2A EB 87 F9 4A CE SÚŒá–1.£.C*ë‡ùJÎ
0020: 67 E1 5B 0F 1E 6D 88 DB 59 05 EC AE F9 A0 35 61 gá[..mˆÛY.ì®ù 5a
0030: 8B D8 C4 AC C6 37 36 03 2B 6E C2 5A D7 87 5B 12 ‹ØÄ¬Æ76.+nÂZ×‡[.
0040: 89 B0 35 A5 F9 FD 77 46 BD CB 69 D2 15 4A C5 99 ‰°5¥ùýwF½ËiÒ.JÅ™
0050: DD A5 EE 91 DB B3 22 CD D2 83 FB 38 E8 F8 0E 37 Ý¥î‘Û³"ÍÒƒû8èø.7
0060: 83 C3 BE 3B 90 7B C6 49 59 EC 82 1B D2 9E 63 A8 ƒÃ¾;?{ÆIYì,.Òžc¨
0070: FD 49 0D 4B 1B ED 2E 29 71 C8 EA 87 3D EC 40 3A ýI.K.í.)qÈê‡=ì@:
0080: B1 6C 1F 2B 83 AD 54 8E 81 5A CE EA 26 1B AB 6E ±l.+ƒ–TŽ?ZÎê&.«n
0090: FC FA 8C B0 79 17 6C D7 10 4A 1E 97 11 4C B6 30 üúŒ°y.l×.J. L¶0
00A0: 06 E7 1C B2 C0 5C AD B2 7B 8B 79 31 F7 7D 20 75 .ç.²À\-²{‹y1÷} u
00B0: 6B 8C 85 07 6D E3 D5 66 95 24 18 95 6A D8 62 66 kŒ….mãÕf•$.•jØbf
00C0: 26 68 37 E9 66 8D E1 E6 58 63 BD 95 B1 2F 1D F7 &h7éf?áæXc½•±/.÷
00D0: E9 B9 41 CC DD E6 55 6F 22 DF 21 14 65 5E 29 0E é¹AÌÝæUo"ß!.e^).
00E0: 89 06 54 B1 11 EF 9A D8 8E 80 67 78 73 F5 73 84 ‰.T±.ïšØŽ€gxsõs„
00F0: 5C DE A5 23 52 8A C8 99 97 07 5B 8C 00 68 F7 A1 \Þ¥#RŠÈ™—.[Œ.h÷¡
```

5.5.6 Sample Data Calculation (2048-bit RSA keys)

Field	Length & Type	Value
COMMAND MESSAGE		
Message Header	m A	
Command Code	2 A	"GK" – Export DES Key
Encryption Identifier	2 N	"01" – RSA Encryption
Pad Mode Identifier	2 N	"02" – OAEP (EME-OAEP-ENCODE)
Mask Generation Function	2 N	"01" – MGF1 as defined in PKCS#1 v2.2
MGF Hash Function	2 N	"01" – SHA-1
OAEP Encoding Parameters Length	2 N	"04"
OAEP Encoding Parameters	N B	"09080706"
OAEP Encoding Parameters Delimiter	1 A	";"
DES Key Type	4 N	"0600"
Signature Indicator	1 A	"="
Signature Hash Identifier	2 N	"01" – SHA-1
Signature Identifier	2 N	"01" – RSA
Signature Pad Mode Identifier	2 N	"01" – PKCS#1 v1.5 padding
Header Data Block Length	4 N	"0012"
Header Data Block	n B	"YYYYYYYYYYYY"
Delimiter	1 A	";"
Footer Data Block Length	4 N	"0015"
Footer Data Block	n B	"ZZZZZZZZZZZZZZZ"
Delimiter	1 A	";"
Private Key Flag	2 N	"99"
Private Key Length	4 N	"0656"
Private Key	n B	14F7B645054383DA0C52C65F9186ED10F68811F28D0F6998003F A648A75CE512345953EB415BB1DEC921F5DDB3E15D475D8A27E 4C8A948CC36F8C25D847B42BA66551F66DA8602B18CB0F9F0B49

Field	Length & Type	Value
COMMAND MESSAGE		
		00E1C40DBBA12D3E2E7D144AF029770388AF9BC88D9FC296B1FA88844476D5EB8FC70908DF0FCB818B7BB486767D1CBE4BA9277D75008844EEBFE38E6391CDBC8D41CF2F65CC4B81ABDA06D3713750685A30A2462997202E0F5FED5BF2A3A0C250E3D64092E7D2A1B723BD1571A719268C3F0DFE6248A8F3DD934437B34253D061C40142EB0F38325DE143C6F3AE35615FC61A4A7A20DE4F6BF3866CC6E806BDD28C2AD96BEF3B3AE81EE495D0E2590A33234735F48A8535700DF5171E39B1D6B35B728ADFCE94DB695DC8D269B86408230AA26B6C1CC7248B336D56518A2AF09C4D2EB1AFBF54B91C6C0BD13AD0AF8D4EAB23FA2AD0F8CCE1E4BBE2FD8524EEDE07690D9B0C3670F71513B78BE32B0DC2856717049C0551F4EFD5FAC8A46F8E602DCA655A82AF9182DE3907DF76FE48BB96585F69BFEB0D4374A36E0C3D6D52D9F61E9A25E5E4AA1E4C3CC744CB49E864D8B236BE515B2425D6F280C743F4BCFB0649582E99A06FB78EDF75DAE1FD9D3FA66B9407CF2B44A6F6E3FAB21490A3091925B0DDFF0989D62589A4F23061A2CDB41CF343D8946EB3902819A8157ACB2B2BF587F84CFD67BFE91DB9AFBBFA301FCB57BC51264BC3A758EDF5704B6A16918A66FE4E8677842B9080E5CD1CF7EE2FD7AA7AC3B4E41393FA2D514B9761A22A5FD809A2106D99B406A1C5BE7A7F9977A988BAA1A37541CD7D38DB06488AEC80E0A9CD318177BC109DB8768EA571F384E457F8008B8055A6B4B8298F883B51D724012A8F5DD878D585040BFE52C5C006122B48B14E9132EB26175777B9F74B8F73F
DES Key Flag	1 N	"1"
DES Key (LMK)	1A+32H	"U513ECF7E7D0A5554900FA709B9A5F721"
Check Value	16 H	"8B1536D4D60FEC19"
MAC	4 B	AA 5E 41 CC
Public Key	n B	3082010A028201010040F5B53A32475090ADDE8A33EF6F9F95EE584184F1FE7D31AE58C6CAE2A062DAB039E745039F7AA43A236A4540AD0A3F361F905F29B1A0E9C165B0A656D9F818C57E05D787F1F8EE0C6BE9B42A832CFDBD3574A5E7EF863B11FF3F95DCACBBDCFDD90EA9C7D552CF0D1BE471532CF64E029CCDD7D9AEC92F90C39D1BE963AEE6F578E6A3D83AC6572EB5525D0A8179BCBA0263D82EBB5D77C335561606B32701753AEDC42AAAE120D9233CB3B9EF4DDE0C7B0702253E255C1EDE052D9C4F7A6C7AFBF4E408BB18D95354C44462FF352BFF311E8E07A38349A78808164A24A08FD9D3549BC33D58DEC525E97FFBFBF8A6D41205120AE8D4AC756F3E8580918DCB0203010001
Authentication Data	n A	"ABCDEFG"
Delimiter	1 A	";"
Key Data Block Type	2 N	"02" – Key Data Block Template
Key Data Block Template Length	4 N	"0020"
Key Data Block Template	n H	"3012031000000000000000000000000000000000"
Delimiter	1 A	";"
DES Key Offset	4 N	"0004"
Check Value Length	2 N	"00"
Check Value Offset	4 N	"0000"
End Message Delimiter	1 C	
Message Trailer	n A	
RESPONSE MESSAGE		
Message Header	m A	
Response Code	2 A	GL
Error Code	2 A	"00" – No error
Initialization Value	16 H	"????????????????"
DES Key Length	4 N	"0256"
DES Key (PK)	n B	3227D3034C27B648B7AE685377175062AF1922247172E1E7515C1143810954FCD44638AC9698DE90ACEC0E0A977793CA8C35417E0CC92B6A32ABC660C734AA7DFAF8299171204F137CF99810912B

Field	Length & Type	Value
COMMAND MESSAGE		
		34449D7A3930E604135D1264D70EC56878E1C21F42C8EFDE211C CC781D8497967265859C3BE93ADFF0B5DCA79D53EFE86EA6146 19AFBC06AADFF66C5D6BDE3E0A4C0B2084D302B2896654EF936 4046452270C511AE6B03B11B944C8EFCBE1240E595E432AE8C9B A7BE135E6A2960F96577F97634ACB1C742C6283558BB681A612D FF908248F6C9D4E80476B594C72AE147F0022CCDFB9B63631B74 D199AFB385267C223CEFFD7CEAE8D1E04A9EA127FDCB58
Signature Length	4 N	"0256"
Signature	n B	195F27EF10CAF38D8198300638B1B47C53DA8CE196310AA30C43 2AEB87F94ACE67E15B0F1E6D88DB5905ECAEF9A035618BD8C4A CC63736032B6EC25AD7875B1289B035A5F9FD7746BDCB69D215 4AC599DDA5EE91DBB322CDD283FB38E8F80E3783C3BE3B907BC 64959EC821BD29E63A8FD490D4B1BED2E2971C8EA873DEC403A B16C1F2B83AD548E815ACEEA261BAB6EFCFA8CB079176CD7104 A1E97114CB63006E71CB2C05CADB27B8B7931F77D20756B8C85 076DE3D566952418956AD86266266837E9668DE1E65863BD95B 12F1DF7E9B941CCDDE6556F22DF2114655E29E890654B111EF9A D88E80677873F573845CDEA523528AC89997075B8C0068F7A1
End Message Delimiter	1 C	
Message Trailer	n A	

6 Local Master Keys (LMKs)

6.1 Introduction

LMKs are used to encrypt operational keys used for encryption, MACing, digital signing, etc. LMKs are secret, internal to the HSM, and do not exist outside of the HSM except as components or shares held in smart cards. Each HSM can have a unique LMK, or an organization can install the same LMKs on multiple HSMs within a logical system.

LMKs provide separation between different types of keys to ensure that keys can be used only for their intended purpose. Thales payment HSMs support two types of LMK, both of which provide key separation:

- Variant LMKs. These are double- or triple-length Triple-DES keys and provide key separation by encrypting different types of key with different variants of the LMK. Double-length Variant LMKs have been in use for many years, and are the most widely used type of LMK. Triple-length Variant LMKs were introduced for later versions of the payShield 10K. See Chapter 7, “*Variant LMK Key Scheme*”.
- Key Block LMKs. These are either triple-length Triple-DES keys, or 256-bit AES keys, and key separation is provided by parameters in the key block which govern characteristics such as usage and exportability of the protected key. Key Block LMKs are newer technology than Variant LMKs and so are still less widely used, but provide security benefits. Chapter 8, “*Key Block LMK Key Scheme*” describes Key Block LMKs in more detail.
- It is possible to install multiple LMKs within a single payShield 10K. See Chapter 9, “*Multiple LMKs*”.
- Refer to the *payShield 10K Core Host Commands reference manual* for information on how the required LMK can be specified in Host Commands.

6.2 Multiple LMKs

The table below shows the possible values in the key block header fields when creating the standard HSM keys. (This table uses the same pink/blue shading as is used in the *payShield 10K Core Host Commands reference manual* to distinguish between information relating to key block and variant keys.)

6.3 Key Block & Variant Key Comparison Table

The table below shows the possible values in the key block header fields when creating the standard HSM keys.

Key Name	Variant LMK		Key Block LMK
	Key Type	LMK Pair/Variant	Key Usage
BDK-1	009	28-29/0	'B0'
BDK-2	609	28-29/6	'41'
BDK-3	809	28-29/8	'42'
BDK-4	909	28-29/9	'43'
BDK-5	-	-	'44'
CK-DEK	50D	36-37/5	'39'
CK-ENC	30D	36-37/3	'37'

Key Name	Variant LMK		Key Block LMK
	Key Type	LMK Pair/Variant	Key Usage
CK-MAC	40D	36-37/4	'38'
CSCK	402	14-15/4	'C0', '11'
CTRDEK	-	-	'25'
CVK	402	14-15/4	'C0', '12', '13'
DEK	00B	32-33/0	'D0', '21'
HMAC	10C	34-35/1	'61', '62', '63', '64', '65'
IKEY[□]	302	14-15/3	'B1'
KEK	107	24-25/1	'54'
KEK (Transport Key)	-	-	'24'
KMC	207	24-25/2	'E7'
KML	200	04-05/2	'E6',
			'31'
MK-AC	109	28-29/1	'E0'
MK-CVC3	709	28-29/7	'32'
			'E6'
MK-DAC	409	28-29/4	'E3'
MK-DN	509	28-29/5	'E4'
MK-SMC	309	28-29/3	'E1'
MK-SMI	209	28-29/2	'E2'
M_KEY_CONF	-	-	'33'
M_KEY_MAC	-	-	'34'
MS_KEY_CONF	-	-	'35'
MS_KEY_MAC	-	-	'36'
PSK	507	24-25/5	'40'
PVK	002	14-15/0	'V0', 'V1', 'V2'
RSA Private Key	00C	34-35/0	'03', '04', '05', '06'
RSA Public Key	00D	36-37/0	'02'
ECC Private Key	-	-	'03'
ECC Public Key	-	-	'02'
SK-DEK	507	24-25/5	'49'
SK-ENC	307	24-25/3	'47'
SK-MAC	407	24-25/4	'48'
SK-RMAC	008	26-27/0	'48'
TAK	003	16-17/0	'M0', 'M1', 'M3', 'M5', 'M6'
TEK	30B	32-33/3	'D0', '23'
TKR	002 or 90D	14-15/0 or 36-37/9	'P0', '73'
TMK	002 or 80D	14-15/0 or 36-37/8	'K0', '51'
ZKA	607	24-25/6	‘53’
MKPOS/MKSER	-	-	‘57’

[ɸ] IKEY is also known as IPEK.

6.4 Converting Key Names

The table below shows some of the conversions between Thales and other organizations’ key names:

Organization	Key Description	Thales Key Description	Thales Key Name
Mastercard	Issuer MK	Master Key for Authentication Cryptograms	MK-AC
		Master Key for Secure messaging Integrity	MK-SMI
		Master Key for Secure Message Confidentiality	MK-SMC
		Master Key for Data Authentication Codes	MK-DAC
		Master Key for Dynamic Numbers	MK-DN
Mastercard	ICC MK	Derived Key for Authentication Cryptograms	DK-AC
		Derived Key for Secure Messaging Integrity	DK-SMI
		Derived Key for Secure Messaging Confidentiality	DK-SMC
		Derived Key for Dynamic Numbers	DK-DN
Visa	AWK (Acquirer Working Key)	Zone PIN Key	ZPK
Visa	C2KA (Card Verification Key for generation of CVV2)	CVKA (1st half of double-length CVK)	CVK
Visa	C2KB (Card Verification Key for generation of CVV2)	CVKA (2nd half of double-length CVK)	CVK
Visa	CAKA (Card Verification Key (for generation of CAVV)	CVKA (1st half of double-length CVK)	CVK
Visa	CAKB (Card Verification Key (for generation of CAVV)	CVKA (2nd half of double-length CVK)	CVK
Visa	DMK-AC	Master Key for Authentication Cryptograms	MK-AC
Visa	DMK-MAC	Master Key for Secure messaging Integrity	MK-SMI
Visa	DMK-ENC	Master Key for Secure Message Confidentiality	MK-SMC
Visa	IWK (Issuer Working Key)	Zone PIN Key	ZPK

7 Variant LMK Key Scheme

A Variant LMK is a set of 20 double- or triple-length TDES keys, with different "pairs" (and variants of those pairs) being used to encrypt different types of keys. The Double- length Variant LMK is the original LMK format supported in all versions of Racal/Thales payment HSM firmware.

Note: The term "Variant LMK" refers to the 'variant' method of encrypting keys; a Variant LMK is not itself a variant of any other key.

Keys encrypted under a Variant LMK have an associated (3-digit) Key Type Code, which is used to enforce key separation between different types of keys. The different types of symmetric keys available are included in the table below.

Note: Variant LMKs cannot be used to protect AES keys or RSA keys longer than 2048 bits, or ECC keys.

The tables below use the same pink shading as in the *payShield 10K Core Host Commands reference manual* to indicate information relating to variant keys.

7.1 How the Variant Scheme Works

Each key of a double- or triple-length TDES key set is encrypted separately using the ECB mode of encryption. For the second key, a variant is applied to the encryption key. There are five variants to enable the encryption of each key distinctly. This application of variants enforces the key use as a double- or triple-length key and the key order. This scheme is available for encryption of keys under the Local Master Key and for import and export of keys.

Variant Local Master Keys can be either:

- Double-length TDES keys, consisting of a left and right half. Each half consists of 16 hexadecimal characters.
- Triple-length TDES keys, consisting of a left, middle and right part. Each part, as for double-length keys, consists of 16 hexadecimal characters.

Other key encryption keys, such as ZMKs, can be double- or triple-length TDES keys.

The variant is applied to the right half of double-length encrypting keys, and to the middle part of triple-length encrypting keys.

The tags for this scheme are as follows:

U - Double-length DES keys

T - Triple-length DES keys

The following variants are used for this purpose:

- Double-length key:

Left part - `6A`

Right part - `5A`

- Triple-length key:

Left part - `6A`

Middle part - DE

Right part - 2B

Example 1:

Given a double-length encrypting key of: XXXX XXXX XXXX XXXX YYYY YYYY YYYY YYYY, and a double-length key of: AAAA AAAA AAAA AAAA BBBB BBBB BBBB BBBB:

- The variant A6 is applied to the first two hex characters of Y to encrypt A.
- The variant 5A is applied to the first two hex characters of Y to encrypt B.

Example 2:

Given a double-length encrypting key of: XXXX XXXX XXXX XXXX YYYY YYYY YYYY YYYY, and a triple-length key of: AAAA AAAA AAAA AAAA BBBB BBBB BBBB BBBB CCCC CCCC CCCC CCCC:

- The variant 6A is applied to the first two hex characters of Y to encrypt A.
- The variant DE is applied to the first two hex characters of Y to encrypt B
- The variant 2B is applied to the first two hex characters of Y to encrypt C

Variants are applied by "Exclusive ORing" (XOR) the first two characters of Y with the Variant.

7.2 Local Master Key (LMK) Variants

To protect key usage, variants of the Local Master Key are used for encryption of defined keys or key components. The Key Type Table (see later in this chapter) defines the LMK pairs and variants that are used to protect various types of keys. For example, an MK- SMI is encrypted using LMK 28-29 variant 2.

In order to create an LMK variant, a non-secret fixed value (known as a variant) is XOR'ed with the first byte of the LMK. The variants used by the HSM are:

Variant 1 : A6

Variant 2 : 5A

Variant 3 : 6A

Variant 4 : DE

Variant 5 : 2B

Variant 6 : 50

Variant 7 : 74

Variant 8 : 9C

Variant 9 : FA

The example below demonstrates how a Local Master Key variant is calculated for key type MK-SMI:

1. Refer to the *Key Type Table* to select the appropriate LMK pair for the type of key that you wish to encrypt. For example, for key type MK-SMI, LMK 28-29 is used:

Test LMK 28-29: 1A1A 1A1A 1A1A 1A1A 1C1C 1C1C 1C1C 1C1C

2. Identify from the Key Type Table which Variant of the LMK is required. For key type MK-SMI variant 2 is used:

Variant 2: 5A

3. Exclusive-OR the selected variant with the first byte of the LMK pair:

1A **XOR** 5A = 40

4. Replace the left-most byte of the LMK pair with the result of Step 3 and use the resulting key to encrypt the MK- SMI:

LMK Variant 2 = **40**1A 1A1A 1A1A 1A1A 1C1C 1C1C 1C1C 1C1C

When the Variants are applied to the standard Double-length Variant Test LMK or Triple-length Variant Test LMK (see later in this chapter), the left-most bytes of the sets are as follows:

7.2.1 Double-length Variant LMK

LMK Pair	First byte of LMK								
	1	2	3	4	5	6	7	8	9
00-01	A7	5B	6B	DF	2A	51	75	9D	FB
02-03	86	7A	4A	FE	0B	70	54	BC	DA
04-05	E6	1A	2A	9E	6B	10	34	DC	BA
06-07	C7	3B	0B	BF	4A	31	15	FD	9B
08-09	26	DA	EA	5E	AB	D0	F4	1C	7A
10-11	07	FB	CB	7F	8A	F1	D5	3D	5B
12-13	67	9B	AB	1F	EA	91	B5	5D	3B
14-15	46	BA	8A	3E	CB	B0	94	7C	1A
16-17	BA	46	76	C2	37	4C	68	80	E6
18-19	A7	5B	6B	DF	2A	51	75	9D	FB
20-21	A4	58	68	DC	29	52	76	9E	F8
22-23	A1	5D	6D	D9	2C	57	73	9B	FD
24-25	B5	49	79	CD	38	43	67	8F	E9
26-27	B0	4C	7C	C8	3D	46	62	8A	EC
28-29	BC	40	70	C4	31	4A	6E	86	E0
30-31	85	79	49	FD	08	73	57	BF	D9
32-33	80	7C	4C	F8	0D	76	52	BA	DC
34-35	8C	70	40	F4	01	7A	5E	B6	D0
36-37	89	75	45	F1	04	7F	5B	B3	D5
38-39	A7	5B	6B	DF	2A	51	75	9D	FB

7.2.2 Triple-length Variant LMK

LMK Pair	First byte of LMK								
	1	2	3	4	5	6	7	8	9
00-01	75	89	B9	0D	F8	83	A7	4F	29
02-03	2C	D0	E0	54	A1	DA	FE	16	70
04-05	9B	67	57	E3	16	6D	49	A1	C7

LMK Pair	First byte of LMK								
	1	2	3	4	5	6	7	8	9
06-07	A7	5B	6B	DF	2A	51	75	9D	FB
08-09	13	EF	DF	6B	9E	E5	C1	29	4F
10-11	CD	31	01	B5	40	3B	1F	F7	91
12-13	F7	0B	3B	8F	7A	01	25	CD	AB
14-15	2F	D3	E3	57	A2	D9	FD	15	73
16-17	15	E9	D9	6D	98	E3	C7	2F	49
18-19	6E	92	A2	16	E3	98	BC	54	32
20-21	29	D5	E5	51	A4	DF	FB	13	75
22-23	6B	97	A7	13	E6	9D	B9	51	37
24-25	BC	40	70	C4	31	4A	6E	86	E0
26-27	C1	3D	0D	B9	4C	37	13	FB	9D
28-29	68	94	A4	10	E5	9E	BA	52	34
30-31	58	A4	94	20	D5	AE	8A	62	04
32-33	32	CE	FE	4A	BF	C4	E0	08	6E
34-35	26	DA	EA	5E	AB	D0	F4	1C	7A
36-37	1A	E6	D6	62	97	EC	C8	20	46
38-39	CE	32	02	B6	43	38	1C	F4	92

Note: the term "LMK Pair" and the associated numbering scheme is retained for historical reasons, even though for triple-length LMKs each key is actually a triplet.

7.2.3 Local Master Key Triple DES Variant scheme

The Local Master Key Variants described in the previous section are used only to protect key usage. The HSM can also use the variant technique to provide additional protection to Triple-DES keys:

- To ensure that the Left and Right parts of a double-length Triple-DES key can only be used as such.
- To ensure that the Left, Middle and Right parts of a triple-length Triple-DES key can only be used as such.

The following variants are used for this purpose:

- Double-length key:

  Left part - A6

  Right part - 5A

- Triple-length key:

  Left part - 6A

  Middle part - DE

  Right part - 2B

Key Scheme tags are used to identify the technique used to encrypt keys.

- 'U' is used to identify double-length Triple DES keys that are encrypted using the Triple-DES variant Scheme
- 'T' is used to identify triple-length Triple DES keys that are encrypted using the Triple-DES variant scheme.

The example below demonstrates how a double-length MK-SMI is encrypted using this method. The test key to be encrypted is:

Test MK-SMI = `F1F1 F1F1 F1F1 F1F1  C1C1 C1C1 C1C1 C1C1`

1. Refer to the Key Type Table to select the appropriate LMK pair and variant for the type of key that you wish to encrypt. For example, for key type MK-SMI, LMK 28-29 Variant 2 is used (this is the LMK variant that was calculated in the example in section 15):

LMK 28-29 Variant 2 = `401A 1A1A 1A1A 1A1A  1C1C 1C1C 1C1C 1C1C`

2. Select the appropriate variants to be applied to the encrypting key. In this case the MK-SMI is a double-length Triple-DES key, so the following variants should be used:

- To encrypt the left part of the MK-SMI - `A6`
- To encrypt the right part of the MK-SMI - `5A`

3. To create the key with which to encrypt the left part of MK-SMI, Exclusive-OR A6 with the first byte of the right part of the LMK pair:

`1C` **XOR** `A6` = `BA`

Key with which to encrypt left part of MK-SMI

= 01A 1A1A 1A1A 1A1A **BA**1C 1C1C 1C1C 1C1C

4. Use the key calculated in step 3 to encrypt the left part of the MK-SMI:

Key with which to encrypt left part of MK-SMI

= 401A 1A1A 1A1A 1A1A **BA**1C 1C1C 1C1C 1C1C

Left part of MK-SMI = `F1F1 F1F1 F1F1 F1F1`

Result of Triple-DES encryption is: `5178 C9D3 D105 2B15`

5. To create the key with which to encrypt the right part of MK-SMI, Exclusive-OR 5A with the first byte of the right part of the LMK pair

`1C` **XOR** `5A` = `46`

Key with which to encrypt left part of MK-SMI

= 401A 1A1A 1A1A 1A1A **46**1C 1C1C 1C1C 1C1C

6. Use the key calculated in step 5 to encrypt the right part of the MK-SMI:

Key with which to encrypt right part of MK-SMI

= 401A 1A1A 1A1A 1A1A **46**1C 1C1C 1C1C 1C1C+

Right part of MK-SMI = `C1C1 C1C1 C1C1 C1C1`

Result of Triple-DES encryption is:

BF6A EC45 8B4A 4564

The encrypted MK-SMI is the result of step 4 concatenated with the result of step 6:

```
5178 C9D3 D105 2B15  BF6A EC45 8B4A 4564
```

The example above can be demonstrated on an HSM by using the FK console command, with inputs as follows:

```
Offline-AUTH> FK <Return>
Enter key length [1,2,3]: 2 <Return>
Enter key type: 209 <Return>
Enter key scheme: U <Return>
Enter component type [X,H,T,E,S]: X <Return>
Enter number of components [1-9]: 2 <Return>
Enter component 1: 50505050505050505050505050505050 <Return>
Enter component 2: A1A1A1A1A1A1A1A19090909090909090 <Return>
Encrypted key: U 5178 C9D3 D105 2B15 BF6A EC45 8B4A 4564
Key check value: 8357D9
```

When the Variants are applied to the standard test LMK set (see next section), the first bytes of the second key are as follows:

7.2.3.1 Double-length Variant LMK

LMK Pair	First byte of second key of the LMK				
	Double-length Key Scheme Tag 'U'		Triple-length Key Scheme Tag 'T'		
	1 of 2	2 of 2	1 of 3	2 of 3	3 of 3
04-05	F7	0B	3B	8F	7A
06–07	D6	2A	1A	AE	5B
14–15	57	AB	9B	2F	DA
16–17	A7	5B	6B	DF	2A
18-19	A7	5B	6B	DF	2A
20–21	A2	5E	6E	DA	2F
22-23	B6	4A	7A	CE	3B
24–25	B3	4F	7F	CB	3E
26–27	BF	43	73	C7	32
28–29	BA	46	76	C2	37
30-31	83	7F	4F	FB	0E
32–33	8F	73	43	F7	02
34-35	8A	76	46	F2	07
36–37	97	6B	5B	EF	1A
38-39	A7	5B	6B	DF	2A

7.2.3.2 Triple-length Variant LMK

LMK Pair	First byte of second key of the LMK				
	Double-length Key Scheme Tag "U"		Triple-length Key Scheme Tag "T"		
	1 of 2	2 of 2	1 of 3	2 of 3	3 of 3
04-05	CE	32	02	B6	43
06–07	68	94	A4	10	E5
14–15	BC	40	70	C4	31
16–17	94	68	58	EC	19
18-19	DA	26	16	A2	57
20–21	B6	4A	7A	CE	3B
22-23	07	FB	CB	7F	8A
24–25	0E	F2	C2	76	83
26–27	F8	04	34	80	75
28–29	A8	54	64	D0	25
30-31	79	85	B5	01	F4
32–33	6B	97	A7	13	E6
34-35	29	D5	E5	51	A4
36–37	7A	86	B6	02	F7
38-39	43	BF	8F	3B	CE

Note: The term "LMK Pair" and the associated numbering scheme is retained for historical reasons, even though for triple-length LMKs each key is actually a triplet.

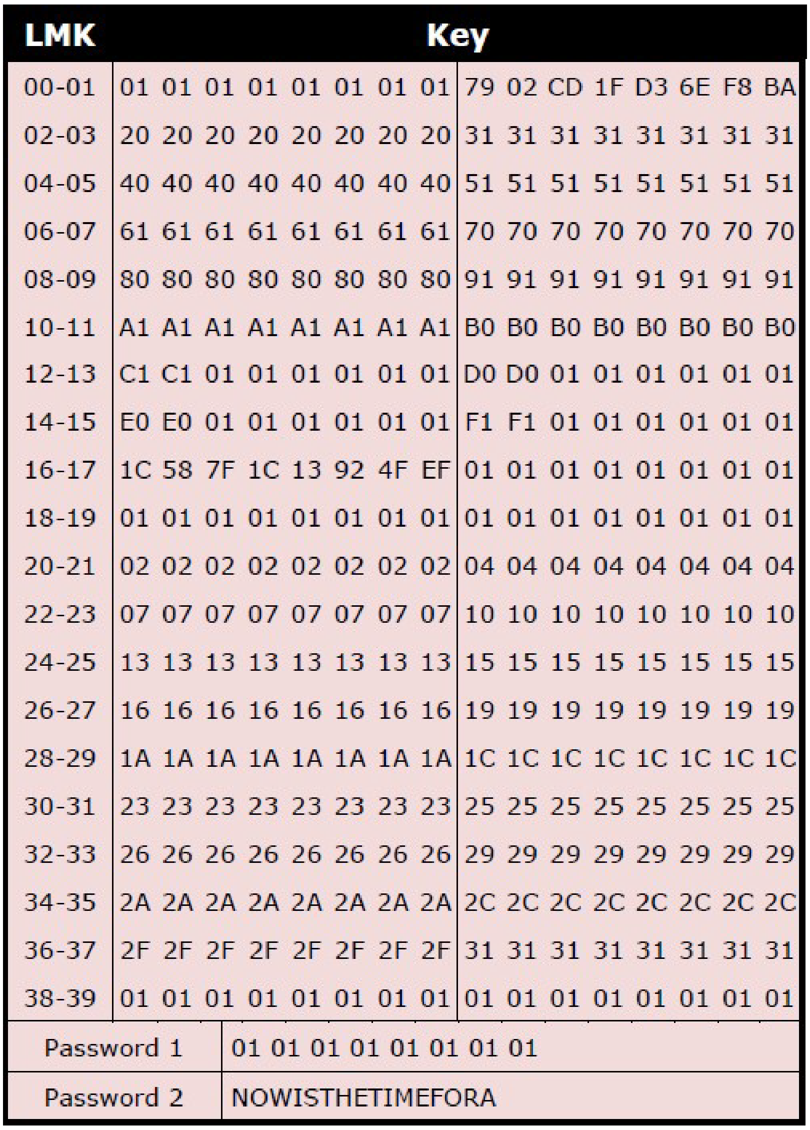
When the HSM encrypts a key component under a variant of the LMK, an additional variant (0xFF) is applied to the first byte of the LMK. This is to prevent a single encrypted component from masquerading as an encrypted key.

7.3 Test Variant LMK

The values of the LMK pairs contained in the "Test LMK" Smartcard are shown in the table below. The two Passwords are also held in this device, and their values are also shown below.

The PIN for each Test LMK Smartcard is: **1 2 3 4**

7.3.1 Double-length Variant LMK



LMK	Key	
00-01	01 01 01 01 01 01 01 01	79 02 CD 1F D3 6E F8 BA
02-03	20 20 20 20 20 20 20 20	31 31 31 31 31 31 31 31
04-05	40 40 40 40 40 40 40 40	51 51 51 51 51 51 51 51
06-07	61 61 61 61 61 61 61 61	70 70 70 70 70 70 70 70
08-09	80 80 80 80 80 80 80 80	91 91 91 91 91 91 91 91
10-11	A1 A1 A1 A1 A1 A1 A1 A1	B0 B0 B0 B0 B0 B0 B0 B0
12-13	C1 C1 01 01 01 01 01 01	D0 D0 01 01 01 01 01 01
14-15	E0 E0 01 01 01 01 01 01	F1 F1 01 01 01 01 01 01
16-17	1C 58 7F 1C 13 92 4F EF	01 01 01 01 01 01 01 01
18-19	01 01 01 01 01 01 01 01	01 01 01 01 01 01 01 01
20-21	02 02 02 02 02 02 02 02	04 04 04 04 04 04 04 04
22-23	07 07 07 07 07 07 07 07	10 10 10 10 10 10 10 10
24-25	13 13 13 13 13 13 13 13	15 15 15 15 15 15 15 15
26-27	16 16 16 16 16 16 16 16	19 19 19 19 19 19 19 19
28-29	1A 1A 1A 1A 1A 1A 1A 1A	1C 1C 1C 1C 1C 1C 1C 1C
30-31	23 23 23 23 23 23 23 23	25 25 25 25 25 25 25 25
32-33	26 26 26 26 26 26 26 26	29 29 29 29 29 29 29 29
34-35	2A 2A 2A 2A 2A 2A 2A 2A	2C 2C 2C 2C 2C 2C 2C 2C
36-37	2F 2F 2F 2F 2F 2F 2F 2F	31 31 31 31 31 31 31 31
38-39	01 01 01 01 01 01 01 01	01 01 01 01 01 01 01 01
Password 1	01 01 01 01 01 01 01 01	
Password 2	NOWISTHETIMEFORA	

The check value for the Double-length Variant Test LMK is: 268604

7.3.2 Triple-length Variant Test LMK

LMK	Key		
00-01	D3 CB 07 68 76 A2 07 04	01 01 01 01 01 01 01 01	32 B9 7F 73 34 AE B6 5E
02-03	8A CD 34 CE F4 91 79 9D	F1 19 94 8F E5 E6 B6 9B	61 97 8A 40 D0 83 04 32
04-05	3D 80 AD C8 6D 83 97 2F	68 EC 6B 7A 23 25 DA 98	A2 23 6D 1A 89 9B 07 32
06-07	01 34 76 B6 F4 08 BA 6B	CE 45 4C 2C 6D A8 B3 5E	BA C2 4A E6 1F 43 70 49
08-09	B5 7A E3 58 A2 1A DA 89	19 C2 5E 9E F4 8A B3 01	61 C1 23 1A 8F C4 2A 38
10-11	6B F7 10 C1 DF 13 7C EC	7F B3 7F E9 38 F2 A7 3D	BA F2 C4 B5 9B FD 1C 54
12-13	51 DC F1 58 D6 CD 0E CE	A2 E9 BC 0B 1F 85 EF 8C	EA C8 10 A8 1A C8 A7 EA
14-15	89 B5 CB BA 43 80 C8 91	1A 6E 1A D6 61 1C 1C DA	94 68 E3 CB 1A 26 9E FB
16-17	B3 FB D3 4A 5E 51 EC 52	32 AD FE BA 32 0D 68 7C	7A 68 31 EF 25 58 C4 A7
18-19	C8 B9 49 D6 2C 57 9E A7	7C CD CE A1 D0 3D 9E 6B	F4 A1 E6 3B E5 85 8A 83
20-21	8F 32 B9 10 E9 D5 6E DF	10 02 FB B5 57 AB 8F 73	0D 34 4A B3 89 38 F2 FD
22-23	CD 34 89 FB 38 54 97 61	A1 31 1F CB 92 AE 54 B3	16 76 13 16 76 A8 76 DF
24-25	1A CB 8C C1 26 1F FE EA	A8 E9 EA 58 80 1A A7 85	46 20 91 85 AB 3D 89 37
26-27	67 AB EC 46 16 23 D3 70	5E 85 F8 2F 15 26 62 68	02 15 D3 2F 8A CE D5 91
28-29	CE 23 B0 98 B0 34 B6 CD	0E 08 CD FE 3D 08 B5 0D	4F 80 D3 83 9D 73 85 BF
30-31	FE 3E 64 3E 92 C1 23 D3	DF 89 B6 83 43 A2 61 6D	AB 61 10 C4 A7 9E EA AB
32-33	94 DF 13 58 3B B5 E3 1F	CD B6 B5 32 AE 6D A8 DC	0D A8 6B EF 34 C7 51 8A
34-35	80 76 7F FD 76 F1 CE 57	8F 1F EC 15 AE 3E 7F 10	16 38 80 D6 29 58 08 CD
36-37	BC FB D6 89 FE 86 15 E0	DC AD 8F 8F 49 F8 0D 61	3D EA 73 F2 EC C2 F2 7C
38-39	68 B6 3D 1A F8 73 D5 92	E5 1C 1F D5 80 C1 D3 A1	2C 85 32 2A 1F 07 D9 08
Password 1	D3 CB 07 68 76 A2 07 04		
Password 2	01 01 01 01 01 01 01 01		

Note: the term "LMK Pair" and the associated numbering scheme is retained for historical reasons, even though for triple-length LMKs each key is actually a triplet.

The check value for the Triple-length Variant Test LMK is: 665641

7.4 Variant Key Type Codes

Each key has a Key Type as defined in the following table. The relationship between Key Type and LMK Variant is shown in the Key Type Table in the next section.

Key Type	Key Type Code [1]	Key Type Code [2]	Description
ZMK	000	000	Zone Master Key (also known as ZCMK)
ZMK (Comp)	100	100	Zone Master Key Component (legacy commands only)
KML	200	200	Master Load Key (Visa Cash)
KEKr	300	300	(AS 2805) Key Encryption Key of Recipient
KEKs	400	400	(AS 2805) Key Encryption Key of Sender
ZPK	001	001	Zone PIN Key
PVK	002	002	PIN Verification Key
PVVK	002	002	(OBKM) PVV Key
TPK	002	70D	Terminal PIN Key
PEK	002	70D	(AS 2805) PIN Encipherment Key
PEK	002	70D	(LIC031) PIN Encryption Key
TMK	002	80D	Terminal Master Key
KT	002	80D	(AS 2805) Transaction Key
TK	002	80D	(AS 2805) Terminal Key
KI	002	80D	(AS 2805) Initial Transport Key
KCA	002	80D	(AS 2805) Sponsor Cross Acquirer Key
KMA	002	80D	(AS 2805) Acquirer Master Key Encrypting Key
TKR	002	90D	Terminal Key Register
TMK1	102	102	(AS 2805) Terminal Master Key
TMK2	202	202	(AS 2805) Terminal Master Key
IKEY[ϕ]	302	302	Initial Key (DUKPT)
CK-ENK	30D	30D	(Issuing) Card Key for Cryptograms
CVK	402	402	Card Verification Key
CSCK	402	402	Card Security Code Key
CK-MAC	40D	40D	(Issuing) Card Key for Authentication
CK-DEK	50D	50D	(Issuing) Card Key for Authentication
KIA	602	602	(AS 2805) Acquirer Initialization Key
TAK	003	003	Terminal Authentication Key
TAKr	203	203	(AS 2805) Terminal Authentication Key of Recipient
TAKs	103	103	(AS 2805) Terminal Authentication Key of Sender
KML	105	105	(OBKM) Master Load Key
KML$_{ISS}$	105	105	(OBKM) Master Load Key for Issuer
KMX	205	205	(OBKM) Master Currency Exchange Key
KMX$_{ISS}$	205	205	(OBKM) Master Currency Exchange Key for Issuer
KMP	305	305	(OBKM) Master Purchase Key
KMP$_{ISS}$	305	305	(OBKM) Master Purchase Key for Issuer

Key Type	Key Type Code [1]	Key Type Code [2]	Description
KI$_{S.5}$	405	405	(OBKM) S5 Issuer Key
KM3L	505	505	(OBKM) Master Key for Load & Unload Verification
KM3L$_{ISS}$	505	505	(OBKM) Master Key for Load & Unload Verification for Issuer
KM3X	605	605	(OBKM) Master Key for Currency Exchange Verification
KM3X$_{ISS}$	605	605	(OBKM) Master Key for Currency Exchange Verification for Issuer
KMAC$_{S4}$	705	705	(OBKM)
KMAC$_{S5}$	805	805	(OBKM)
KMAC$_{ACQ}$	905	905	(OBKM)
KMAC$_{ACX}$	905	905	(OBKM)
WWK	006	006	Watchword Key
KMAC$_{UPD}$	106	106	(OBKM)
KMAC$_{MA}$	206	206	(OBKM)
KMAC$_{CI}$	306	306	(OBKM)
KMAC$_{ISS}$	306	306	(OBKM)
KMSC$_{ISS}$	406	406	(OBKM) Secure Messaging Master Key
BKEM	506	506	(OBKM) Transport key for key encryption
BKAM	606	606	(OBKM) Transport key for message authentication
KEK	107	107	(Issuing) Key Encryption Key
KMC	207	207	(Issuing) Master Personalization Key
SK-ENC	307	307	(Issuing) Session Key for cryptograms and encrypting card messages
SK-MAC	407	407	(Issuing) Session Key for authenticating card messages
SK-DEK	507	507	(Issuing) Session Key for encrypting secret card data
KD-PERSO	507	507	(Issuing) KD Personalization Key
ZKA MK	607	607	Master key for GBIC/ZKA key derivation
MK-KE	807	807	(Issuing) Master KTU Encipherment key
MK-AS	907	907	(Issuing) Master Application Signature (MAC) key
ZAK	008	008	Zone Authentication Key
ZAKs	108	108	(AS 2805) Zone Authentication Key of Sender
ZAKr	208	208	(AS 2805) Zone Authentication Key of Recipient
BDK-1	009	009	Base Derivation Key (type 1)
MK-AC	109	109	Master Key for Application Cryptograms
MK-SMI	209	209	Master Key for Secure Messaging (for Integrity)
MK-SMC	309	309	Master Key for Secure Messaging (for Confidentiality)
MK-DAC	409	409	Master Key for Data Authentication Codes
MK-DN	509	509	Master Key for Dynamic Numbers
BDK-2	609	609	Base Derivation Key (type 2)
MK-CVC3	709	709	Master Key for CVC3 (Contactless)
BDK-3	809	809	Base Derivation Key (type 3)
BDK-4	909	909	Base Derivation Key (type 4)
ZEK	00A	00A	Zone Encryption Key

Key Type	Key Type Code [1]	Key Type Code [2]	Description
ZEKs	10A	10A	(AS 2805) Zone Encryption Key of Sender
ZEKr	20A	20A	(AS 2805) Zone Encryption Key of Recipient
DEK	00B	00B	Data Encryption Key
TEK	00B	00B	(AS 2805) Terminal Encryption Key
TEKs	10B	10B	(AS 2805) Terminal Encryption Key of Sender
TEKr	10B	10B	(AS 2805) Terminal Encryption Key of recipient
TEK	30B	30B	Terminal Encryption Key
RSA-SK	00C	00C	RSA Private Key
HMAC	10C	10C	HMAC key
RSA-PK	00D	00D	RSA Public Key
TPK	002	70D	Terminal PIN Key
PEK	002	70D	(AS 2805) PIN Encipherment Key
TMK	002	80D	Terminal Master Key
KT	002	80D	(AS 2805) Transaction Key
TK	002	80D	(AS 2805) Terminal Key
KI	002	80D	(AS 2805) Initial Transport Key
KCA	002	80D	(AS 2805) Sponsor Cross Acquirer Key
KMA	002	80D	(AS 2805) Acquirer Master Key Encrypting Key
TKR	002	90D	Terminal Key Register

Notes:

[1]This Key Type Code column applies when the security setting "*Enforce key type 002 separation for PCI HSM compliance*" has a value of 'N' – i.e. key separation is not PCI HSM complaint.

[2]This Key Type Code column applies when the security setting "*Enforce key type 002 separation for PCI HSM compliance*" has a value of 'Y' – i.e. key separation is PCI HSM complaint.

7.5 Key Type Table

The payShield 10K HSM provides a set of commands for key generation, key export and key import. An export command is one that translates a key from LMK encryption to encryption under a transport key, for sending to another party. Import is the reverse, for receiving keys and translating to local storage. The Key Type Table (see below) controls 'permitted actions' for the console and host commands used to generate, import and export keys when the transport key is a DES/3DES key. When using other types of transport keys (e.g. RSA), consult the appropriate command reference manual to determine 'permitted actions' and authorization requirements.

Errors are reported when an action breaks the rules imposed by the table. For example:

29 : Key function not permitted

The Key Type table is automatically modified dependent on the value of the Security Setting

"*Enforce key type 002 separation for PCI HSM compliance*": a setting of 'N' indicates non-compliance with the requirements of PCI HSM, and a setting of 'Y' indicates compliance with the key separation requirements of PCI HSM.

7.5.1 Non PCI-HSM Compliant (for backwards compatibility)

Variant ⇨		0			1			2			3			4			5			6			7			8			9		
LMK ⇩		G	E	I	G	E	I	G	E	I	G	E	I	G	E	I	G	E	I	G	E	I	G	E	I	G	E	I	G	E	I
Pair	Code																														
04 – 05	00	**ZMK**			**ZMK (Comp)**			**KML**			*KEKr*[1]			*KEKs*[1]																	
		A	A[6]	A[7]	U	A	U	U	A	U	U	A	U	U	A	U															
06 – 07	01	**ZPK**			*Auth Para*																										
		U	A	U																											
14 – 15	02	**PVK** **TPK KEYVAL** **TMK** **TKR** *KT*[1] **DbTAB** *KCA*[1] *KMA*[1] *KI*[1] *PEK*[1] *TK*[1] *PVVK*[2]			*TMK1*[1]			*TMK2*[1]			**IKEY**			**CVK** **CSCK**						*KIA*						*PPASN*					
		U	A	U	U	A	U	U	A	U	U	A	U	U	A	U				U	A	U									
16 – 17	03	**TAK**			*TAKs*			*TAKr*																							
		U	A	U	U	A	U	U	A	U																					
18 – 19	04				**DTAB**[1]			**IPB**																							
20 – 21	05				*KML* *KML*$_{ISS}$			*KMX* *KMX*$_{ISS}$			*KMP* *KMP*$_{ISS}$			*KI*$_{S,6}$			*KM3L* *KM3L*$_{ISS}$			*KM3X* *KM3X*$_{ISS}$			*KMAC*$_{S4}$			*KMAC*$_{S5}$			*KMAC*$_{ACQ}$ *KMAC*$_{ACK}$		
					U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U
22 – 23	06	**WWK**			*KMAC*$_{UPD}$			*KMAC*$_{MA}$			*KMAC*$_{CI}$ *KMAC*$_{ISS}$			*KMSC*$_{ISS}$			*BKEM*			*BKAM*											
		U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U									
24 – 25	07				*KEK*			*KMC*			*SK-ENC*			*SK-MAC*			*SK-DEK* *KD-PERSO*			**ZKA MK**						*MK-KE*			*MK-AS*		
					U	A	A	U	A	A										U	A	U				U	A	U	U	A	U
26 – 27	08	**ZAK**			*ZAKs*			*ZAKr*																							
		U	A	U	U	A	U	U	A	U																					
28 – 29	09	**BDK-1**			**MK-AC**			**MK-SMI**			**MK-SMC**			**MK-DAC**			**MK-DN**			**BDK-2**			**MK-CVC3** *MK-DCVV*			**BDK-3**			**BDK-4**		
		U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U
30 – 31	0A	**ZEK**			*ZEKs1*			*ZEKr1*																							
		U	A	A/U[2]	U			U																							
32 – 33	0B	**DEK** *TEK*[1]			*TEKs*			*TEKr*			**TEK**																				
		U			U			U			U	A																			
34 – 35	0C	**RSA-SK**			**HMAC**																										
		A	A[5]	A[5]	U	A	U																								
36 – 37	0D	**RSA-PK**									*CK-ENC*			*CK-MAC*			*CK-DEK*						*Note 3*			*Note 3*			*Note 3*		
		A		A							U	A	U	U	A	U	U	A	U				U	A	U	U	A	U	U	A	U
38 – 39	0E	Reserved			Reserved			Reserved			Reserved			Reserved			Reserved			Reserved			Reserved			Reserved			Reserved		

Key Type Table 1 - "Enforce key type 002 separation for PCI HSM compliance" is set to "N" (i.e., key separation is not PCI HSM compliant)

7.5.2 PCI Compliant

Variant ⇨		0			1			2			3			4			5			6			7			8			9		
LMK ⇩		G	E	I	G	E	I	G	E	I	G	E	I	G	E	I	G	E	I	G	E	I	G	E	I	G	E	I	G	E	I
Pair	Code																														
04 – 05	00	**ZMK**			**ZMK (Comp)**			**KML**			*KEKr*			*KEKs*																	
		A	A^6	A^7	U	A	U	U	A	U	U	A	U	U	A	U															
06 – 07	01	**ZPK** *PEK*			*Auth Para*																										
		U	A	U																											
14 – 15	02	**PVK** *PVVK*			*TMK1*			*TMK2*			**IKEY**			**CVK** **CSCK**						*KIA*						*PPASN*					
		U	A	U	U	A	U	U	A	U	U	A	U	U	A	U				U	A	U									
16 – 17	03	**TAK**			*TAKs*			*TAKr*																							
		U	A	U	U	A	U	U	A	U																					
18 – 19	04				**DTAB**			**IPB**																							
20 – 21	05				*KML* KML_{ISS}			*KMX* KMX_{ISS}			*KMP* KMP_{ISS}			$KI_{S,5}$			*KM3L* $KM3L_{ISS}$			*KM3X* $KM3X_{ISS}$			$KMAC_{S4}$			$KMAC_{S5}$			$KMAC_{ACQ}$ $KMAC_{ACK}$		
					U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U
22 – 23	06	**WWK**			$KMAC_{UPD}$			$KMAC_{MA}$			$KMAC_{CI}$ $KMAC_{ISS}$			$KMSC_{ISS}$			*BKEM*			*BKAM*											
		U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U									
24 – 25	07				*KEK*			*KMC*			*SK-ENC*			*SK-MAC*			*SK-DEK* *KD-PERSO*			**ZKA MK**						*MK-KE*			*MK-AS*		
					U	A	A	U	A	A										U	A	U				U	A	U	U	A	U
26 – 27	08	**ZAK**			*ZAKs*			*ZAKr*																							
		U	A	U	U	A	U	U	A	U																					
28 – 29	09	**BDK-1**			**MK-AC**			**MK-SMI**			**MK-SMC**			**MK-DAC**			**MK-DN**			**BDK-2**			**MK-CVC3** *MK-DCVV*			**BDK-3**			**BDK-4**		
		U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U	U	A	U
30 – 31	0A	**ZEK**			*ZEKs*			*ZEKr*																							
		U	A	A/U^2	U			U																							
32 – 33	0B	**DEK** *TEK*			*TEKs*			*TEKr*			**TEK**																				
		U			U			U			U	A																			
34 – 35	0C	**RSA-SK**			**HMAC**																										
		A	A^5	A^5	U	A	U																								
36 – 37	0D	**RSA-PK**									*CK-ENC*			*CK-MAC*			*CK-DEK*			**DbTAB**^{1}			**TPK** **KEYVAL** *PEK PEK*			**TMK** *KT* *KCA KMA* *KI T*			**TKR**		
		A		A							U	A	U	U	A	U	U	A	U				U	A	U	U	A	U	U	A	U
38 – 39	0E	Reserved			Reserved			Reserved			Reserved			Reserved			Reserved			Reserved			Reserved			Reserved			Reserved		

Key Type Table 2 - "Enforce key type 002 separation for PCI HSM compliance" set to 'Y' (i.e., key separation is PCI HSM compliant)

The tables above show the actions that can be applied to each specific LMK pair/variant. For each key type, the 3 boxes below the key type refer, from left to right, to:

G = Generate

E = Export

I = Import

Each of these 3 boxes contains one of the following entries to define permissions:

blank = Not allowed
A = allowed only in Authorized State
U = allowed Unconditionally, i.e. without Authorized State

Notes:

1. DTAB = Decimalization Table; DbTAB = Diebold Table
2. Authorization requirements for Import are dependent on the security setting "*Enable ZEK encryption of ASCII data or Binary data or None*".
3. Keys of this key type can be generated/imported/exported, but may only be used for other operations when the security setting "*Enforce key type 002 separation for PCI HSM compliance*" is set to "YES". This allows users to pre-generate their keys before migrating to a PCI-compliant environment.
4. The term "LMK pair" and the associated numbering scheme is retained for historical reasons, even though for triple-length LMKs each key is actually a triplet.
5. If the security setting "*Enable import and export of RSA Private keys*" is set to "YES", otherwise, this operation is not allowed.
6. If the security setting "*Enable export of a ZMK*" is set to "YES", otherwise, this operation is not allowed.
7. If the security setting "*Enable import of a ZMK*" is set to "YES", otherwise, this operation is not allowed.

Not all key type codes are available in all commands for security reasons.

The Key Type code used within commands is formed by using the Variant code as the first character then the LMK pair code as the second character. For example, the code for a ZPK is 001.

The payShield 10K HSM provides a set of commands for key generation, key export and key import. An export command is one that translates a key from LMK encryption to encryption under a ZMK or an RSA public key, for sending to another party. Import is the reverse, for receiving keys and translating to local storage. The Key Type Table controls "permitted actions" for the console and host commands used to generate, import and export keys.

Errors are reported when an action breaks the rules imposed by the table.

For example:

29 : Key function not permitted

8 Key Block LMK Key Scheme

8.1 Introduction

A Key Block LMK is a more recent type of Local Master Key, and is used to encrypt keys in a key block format. It is not compatible with a Variant LMK, and it can only be used to encrypt keys in key block form.

When using a Key Block LMK, working keys are encrypted as a Thales Key Block.

Two types of Key Block LMKs are supported in the payShield 10K:

DES Key Block LMK - based on a triple-length 3-DES key. This key provides a security strength of 112-bits, and can be used to protect subordinate DES, TDES, RSA (up to 2048 bits), and HMAC keys.

AES Key Block LMK - based on a 256-bit AES key. This key provides a security strength of 256-bits, and can be used to protect subordinate AES, DES, TDES, RSA, ECC & HMAC keys.

Notes:

The term 'Key Block LMK' refers to the 'key block' method of encrypting keys; a Key Block LMK is not itself stored in key block form.

AES Key Block LMKs must be used to protect AES keys and RSA keys longer than 2048 bits and ECC keys.

The tables in this chapter use blue shading to indicate that the information relates to keys encrypted using Key Blocks LMKs.

8.2 Thales Key Blocks

Thales Key Blocks use the same structure as X9 ANSI X.9.143 Key Blocks, which are offered as a way of meeting the requirements of ANSI X9.24 when exporting keys between products from different vendors. The main difference is that Thales Key Blocks use some additional key usage and optional block values, as allowed for by the ANSI X.9.143 specification.

8.3 Thales Key Block & PCI Compliance

A Thales Key Block cryptographically binds the encrypted key with its intended key usage, along with other meta data, using a MAC, and thus complies with the requirements of PCI PIN Security Requirement 18.

8.4 Support for Thales Key Blocks

Not all console commands and host commands can be used with Key Block LMKs. Most commands that do not support the use of a Key Block LMK have a newer, alternative command that should be used in their place, or relate to functionality in optional licenses. You should check for Key Block support in the descriptions of all commands you intend to use in the appropriate documentation, e.g.,:

- payShield 10K Console Guide
- payShield 10K Core Host Commands reference manual

8.5 Key Block Format

The Thales Key Block is denoted by key scheme 'S' and has the following format:

Key Scheme Tag ("S") (1 byte)	**Key Block Header** (16 ASCII characters)	**Optional Header** (ASCII characters, variable length)	**Encrypted Key Data** (variable length, ASCII encoded)	**Authenticator** (8 or 16 ASCII characters)

🟦	a 16-byte (clear) Key Block Header, which defines the key usage (e.g. Visa PVV) and mode of use (e.g. verification only), the algorithm with which the key is used (e.g. AES 256-bit) and the limitations on the exportability of the key (e.g. no export permitted); the Header also identifies the LMK used to encrypt the key in the Key Block;
🟨	a (clear) Optional Header block, which can be used (for example) to define the period of validity of the key contained in the Key Block;
🟥	the Encrypted Key Data, which includes the actual key itself, encrypted under the "encryption variant" of the identified LMK (or ZMK/TMK);
🟩	a Key Block Authenticator, calculated using the "authentication variant" of the identified LMK (or ZMK/TMK); the use of the Authenticator prevents unauthorized modification to the Key Block.

The entire key block will be ASCII encoded.

8.5.1 Key Block Header

The Thales Key Block Header is 16 (ASCII) bytes in length and has the following format:

Byte(s)	Field	Comments
0	Version ID	value = "0" (X'30) when protected by a 3-DES key value = "1" (X'31) when protected by an AES key
1-4	Key Block Length	total length of key block
5-6	Key Usage	e.g. key encryption, data encryption
7	Algorithm	e.g. DES, 3-DES, AES
8	Mode of Use	e.g. encrypt only
9-10	Key Version Number	e.g. version of key in the key block or used to indicate that the key is a key component
11	Exportability	e.g. exportable under a trusted key
12-13	Number of optional blocks	number of Optional Header Blocks
14-15	LMK ID	LMK identifier (to support multiple LMKs); numeric values "00" - "19" (X'3030 - X'3139)

8.5.1.1 Key Block Length (Bytes 1-4)

Bytes 1-4 of the Header contain the length of the entire key block, namely Header, Optional Header Blocks, encrypted Key Data and the Authenticator. The length of the key block is calculated after encoding and is represented as 4 numeric (ASCII) digits.

For example, if the total key block length is 112 characters (bytes) then the value in byte 1 will be "0", the value in bytes 2 and 3 will be "1" and the value in byte 4 will be "2" (i.e. X'30313132).

8.5.1.2 Key Usage (Bytes 5-6)

Bytes 5-6 of the Header define the primary usage of the key contained in the key block. The following table defines the usage code. The table also indicates whether the usages applies to ANSI X.9.143 Key Blocks:

- A blank entry means the usage is not appropriate to ANSI X.9.143.
- An entry of 'Y' means that the usage is appropriate to ANSI X.9.143.
- An entry of the form ("XX") means that the usage is not appropriate to ANSI X.9.143 Key Blocks, but that when exporting from Thales Key Block format to ANSI X.9.143 format the usage code is converted to the code in brackets.

Thales KB Key Usage	Equivalent ANSI X.9.143 KB Key Usage	Supported Algorithm(s)	Supported Mode of Use(s)	Description
'01'	-	'D', 'T'	'C', 'G', 'N', 'V'	WatchWord Key (WWK)
'02'	-	'R'	'E', 'N', 'V'	RSA Public Key
		'E'	'N', 'V', 'X'	RSA Public Key
'03'	-	'R'	'D', 'N', 'S'	RSA Private Key (for signing/key mgt)
		'E'	'N', 'S', 'X'	ECC Private Key
'04'	-	'R'	'D', 'N', 'S'	RSA Private Key (for ICCs)
'05'	-	'R'	'D', 'N', 'S'	RSA Private Key (for PIN translation)
'06'	-	'R'	'D', 'N', 'S'	RSA Private Key (for TLS pre-master secret decryption)
'B0'	'B0'	'A', 'T'	'N', 'X'	Base Derivation Key (BDK-1)
'41'	'B0'	'A', 'T'	'N', 'X'	Base Derivation Key (BDK-2)
'42'	'B0'	'T'	'N', 'X'	Base Derivation Key (BDK-3)
'43'	'B0'	'A', 'T'	'N', 'X'	Base Derivation Key (BDK-4)
'44'	'B0'	'T'	'N', 'X'	Base Derivation Key (BDK-5)
'B1'	'B1'	'A', 'T'	'N', 'X'	DUKPT Initial Key (IKEY$^{\phi}$)
'C0'	'C0'	'A', 'T'	'C', 'G', 'N', 'V'	Card Verification Key

Thales KB Key Usage	Equivalent ANSI X.9.143 KB Key Usage	Supported Algorithm(s)	Supported Mode of Use(s)	Description
'11'	'C0'	'T'	'C', 'G', 'N', 'V'	Card Verification Key (American Express CSC)
'12'	'C0'	'T'	'C', 'G', 'N', 'V'	Card Verification Key (Mastercard CVC)
'13'	'C0'	'A', 'T'	'C', 'G', 'N', 'V'	Card Verification Key (Visa CVV)
'D0'	'D0'	'A', 'D', 'T'	'B', 'D', 'E', 'N'	Data Encryption Key (Generic)
'21'	'D0'	'A', 'D', 'T'	'B', 'D', 'E', 'N'	Data Encryption Key (DEK)
'22'	'D0'	'A', 'D', 'T'	'B', 'D', 'E', 'N'	Data Encryption Key (ZEK)
'23'	'D0'	'A', 'D', 'T'	'B', 'D', 'E', 'N'	Data Encryption Key (TEK)
'24'	-	'A'	'B', 'D', 'E', 'N'	Key Encryption Key (Transport Key)
'25'	'D0'	'A'	'B', 'D', 'E', 'N'	CTR Data Encryption Key (CTRDEK)
'E0'	'E0'	'A', 'T'	'N', 'X'	EMV/Chip card Master Key: Application Cryptogram (MK-AC)
'E1'	'E1'	'A', 'T'	'N', 'X'	EMV/Chip card Master Key: Secure Messaging for Confidentiality (MK-SMC)
'E2'	'E2'	'A', 'T'	'N', 'X'	EMV/Chip card Master Key: Secure Messaging for Integrity (MK-SMI)
'E3'	'E3'	'A', 'T'	'N', 'X'	EMV/Chip card Master Key: Data Authentication Code (MK-DAC)
'E4'	'E4'	'T'	'N', 'X'	EMV/Chip card Master Key: Dynamic Numbers (MK-DN)
'E5'	'E5'	'T'	'N', 'X'	EMV/Chip card Master Key: Card Personalization
'E6'	'E6'	'A', 'T'	'N', 'X'	EMV/chip card Master Key: Other
'E7'	-	'A', 'T'	'N', 'X'	EMV/Master Personalization Key
'32'	'E6'	'A', 'T'	'N', 'X'	Dynamic CVV Master Key (MK-CVC3)
'33'	-	'A'	'N', 'X'	Mobile Remote Management Master key for message confidentiality (M_KEY_CONF)
'34'	-	'A'	'N', 'X'	Mobile Remote Management Master key for message integrity (M_KEY_MAC)
'35'	-	'A'	'B', 'D', 'E', 'N'	Mobile Remote Management Session key for message confidentiality (MS_KEY_CONF)

Thales KB Key Usage	Equivalent ANSI X.9.143 KB Key Usage	Supported Algorithm(s)	Supported Mode of Use(s)	Description
'36'	'M3'	'A'	'C', 'N'	Mobile Remote Management Session key for message integrity (MS_KEY_MAC)
'37'	-	'A', 'T'	'N', 'X'	EMV Card Key for cryptograms
'38'	-	'A', 'T'	'N', 'X'	EMV Card Key for integrity
'39'	-	'A', 'T'	'N', 'X'	EMV Card Key for encryption
'40'	-	'T'	'N'	EMV Personalization System Key
'47'	-	'A', 'T'	'B', 'N'	EMV Session Key for cryptograms
'48'	-	'A', 'T'	'G', 'V', 'N'	EMV Session Key for integrity
'49'	-	'T'	'B', 'E'	EMV Session Key for encryption
'K0'	'K0'	'A', 'D', 'T'	'B', 'D', 'E', 'N'	Key Encryption / Wrapping Key (Generic)
'K1'	'K1'	'A', 'T'	'B', 'D', 'E', 'N'	Key Block Protection Key
'51'	'K0', 'K1'	'A', 'D', 'T'	'B', 'D', 'E', 'N'	Terminal Key Encryption (TMK)
'52'	'K0', 'K1'	'A', 'D', 'T'	'B', 'D', 'E', 'N'	Zone Key Encryption (ZMK)
'53'	‘11’	'A', 'T'	'X'	ZKA Master Key (for German transactions)
'54'	-	'A', 'T'	'B', 'D', 'E', 'N'	Key Encryption Key (KEK)
'55'	-	'A'	'E'	Key Encryption Key (Transport Key)
'56'	-	'D', 'T'	'N', 'X'	MPKC (for Italian transactions)
‘57’	-	'D', 'T'	'N', 'X'	MKPOS/MKSER (for Italian transactions)
'M0'	'M0'	'T'	'C', 'G', 'N', 'V'	ISO 16609 MAC algorithm 1
'M1'	'M1'	'D', 'T'	'C', 'G', 'N', 'V'	ISO 9797-1 MAC algorithm 1
'M2'	'M2'	'D', 'T'	'C', 'G', 'N', 'V'	ISO 9797-1 MAC algorithm 2
'M3'	'M3'	'T'	'C', 'G', 'N', 'V'	ISO 9797-1 MAC algorithm 3
'M4'	'M4'	'T'	'C', 'G', 'N', 'V'	ISO 9797-1 MAC algorithm 4
'M5'	'M5'	'A', 'T'	'C', 'G', 'N', 'V'	ISO 9797-1:1999 MAC algorithm 5
'M6'	'M6'	'A'	'C', 'G', 'N', 'V'	ISO 9797-1:2011 MAC algorithm 5/CMAC
'61'	-	'H'	'C', 'G', 'N', 'V'	HMAC key (using SHA-1)
'62'	-	'H'	'C', 'G', 'N', 'V'	HMAC key (using SHA-224)

Thales KB Key Usage	Equivalent ANSI X.9.143 KB Key Usage	Supported Algorithm(s)	Supported Mode of Use(s)	Description
'63'	-	'H'	'C', 'G', 'N', 'V'	HMAC key (using SHA-256)
'64'	-	'H'	'C', 'G', 'N', 'V'	HMAC key (using SHA-384)
'65'	-	'H'	'C', 'G', 'N', 'V'	HMAC key (using SHA-512)
'P0'	'P0'	'A', 'D', 'T'	'B', 'D', 'E', 'N'	PIN Encryption Key (Generic)
'71'	'P0'	'A', 'D', 'T'	'B', 'D', 'E', 'N'	Terminal PIN Encryption Key (TPK)
'72'	'P0'	'A', 'D', 'T'	'B', 'D', 'E', 'N'	Zone PIN Encryption Key (ZPK)
'73'	P0'	'D', 'T'	'N'	Transaction Key Scheme Terminal Key Register (TKR)
'V0'	'V0'	'D', 'T'	'C', 'G', 'N', 'V'	PIN Verification Key (Generic)
'V1'	'V1'	'D', 'T'	'C', 'G', 'N', 'V'	PIN Verification Key (IBM 3624 algorithm)
'V2'	'V2'	'A', 'D', 'T'	'C', 'G', 'N', 'V'	PIN Verification Key (Visa PVV algorithm)

□IKEY is also known as IPEK.

Note 1: Generation, import and export of the Master MKPOS/MKSER (key usage 57) is only supported in the following host commands: A0 (Generate Key), GK (Export Key under an RSA Public Key), A6 (Import Key), A8 (Export Key and BW (Translate Key from Old LMK to New LMK). Note that for BW, only translation from Key Block LMK to Key Block LMK is supported for this key – as this key is not supported in base when using a Variant LMK, translation using a Variant LMK is not supported.

Note 2: The MKPOS/MKSER (key usage 57) is a regional scheme specific key that is not included in the ANSI X.9.143 standard, and so, not supported.

Note 3: Generation, import and export of the AES ZKA (key usage 53) is only supported in the following host commands: A0 (Generate Key – generate and export is not supported), A6 (Import Key), A8 (Export Key), BW (Translate Keys from Old LMK to New LMK) and ‘BU’ (Generate Key Check Value).

Note 4: The AES ZKA (key usage 53) is supported in ANSI X.9.143 as key usage ‘11’ as defined in the GBIC standards. When importing, the key usage will be converted from key usage ‘11’ in ANSI X.9.143 to key usage ‘53’ for Thales Key Block. When exporting, the key usage will be converted from ‘53’ in Thales Key Block to ‘11’ in ANSI X.9.143. The 3DES ZKA is not supported in ANSI X.9.143.

8.5.1.3 Algorithm (Byte 7)

Byte 7 of the Header defines the cryptographic algorithm with which the key contained in the key block will be used. The following values are defined, although only six of these values are currently supported:

Value	Hex	Algorithm
'A'	X'41	AES
'D'	X'44	DES
'E'	X'45	Elliptic curve
'H'	X'48	HMAC
'R'	X'52	RSA
'S'	X'53	DSA (included for future reference)
'T'	X'54	3-DES

8.5.1.4 Mode of Use (Byte 8)

Byte 8 of the Header defines the operation that the key contained in the key block can perform.

Value	Hex	Description
'B'	X'42	The key may be used to perform both encrypt and decrypt operations.
'C'	X'43	The key may be used to perform MAC calculation (both generate & verify) operations.
'D'	X'44	The key may only be used to perform decrypt operations.
'E'	X'45	The key may only be used to perform encrypt operations.
'G'	X'47	The key may only be used to perform MAC generate operations.
'N'	X'4E	No special restrictions apply.
'S'	X'53	The key may only be used to perform digital signature generation operations.
'V'	X'56	The key may only be used to perform digital signature verification operations.
'X'	X'58	The key may only be used to derive other keys.

The 'C', 'G' and 'V' field values will apply to HMAC keys, as well as "ordinary" MAC keys.

The mode of use corresponding to the above values will be extended to include other key usage, as follows:

Value	Hex	Mode of Use
'B'	X'42	Both encryption and decryption
'C'	X'43	Both generate and verify
'D'	X'44	Decrypt only
'E'	X'45	Encrypt only
'G'	X'47	Generate only
'N'	X'4E	No special restrictions or not applicable
'V'	X'56	Verify only

Value	Hex	Mode of Use
'X'	X'58	Derivation only

For example, if, in a particular key block, bytes 5-8 have the values '11TV', this indicates that the key in the key block is a 3-DES key that can only be used for the verification of American Express CSCs. Similarly, if bytes 5-8 have the values '52TE' then the key in the key block is a 3-DES ZMK that can only be used for key export and if bytes 5-8 have the values '01DN' then the key is a single-length WatchWord key that can be used for both response generation and verification.

8.5.1.5 Key Version Number (Bytes 9-10)

Bytes 9-10 of the Header define the version number of the key contained in the key block or that the key is actually a key component. The following values will be supported:

Value		Hex	Key Version Number
(byte 9)	(byte 10)		
0	0	X'3030	Key versioning is not used for this key
c	Any	X'6300	A 'c' in byte 9 indicates that the key carried in the key block is a key component.
Any other combination of printable characters			The version of the key or key component carried in the key block

Note: If byte 9 is set to 'c' then byte 10 will be used to indicate the component number. For example, if bytes 9-10 = 'c2' (X'6332) then this indicates that the key in the key block is the second component of a key. Note, however, that this gives no information regarding the total number of key components.

8.5.1.6 Exportability (Byte 11)

Byte 11 of the Header defines the conditions under which the key contained in the key block can be exported outside the cryptographic domain in which the key is found. A key is defined to be trusted if it is in either Thales Key Block or ANSI X.9.143 Key Block format.

Any other format key is said to be untrusted. The following values will be supported.

Value	Hex	Exportability
'E'	X'45	May only be exported in a trusted key block, provided the wrapping key itself is in a trusted format.
'N'	X'4E	No export permitted.
'S'	X'53	Sensitive; all other export possibilities are permitted, provided such export has been enabled (existing Authorized State requirements remain).

"Sensitive" export includes, for example, when a Key Block LMK is used and the key is exported in ANSI X9.17 format. Export in X9.17 format is disabled by default in the payShield 10K security settings and must be specifically enabled (e.g. via the CS console command) if required.

Note: When a key block is exported in a Thales or ANSI X.9.143 Key Block format, byte 11 in the exported key block dictates how further export must be handled by the recipient of the key block. Hence, if the received byte 11 has value 'E' then further "trusted" export is permitted, but if byte 11 has value 'N' then no further export will be permitted. Such considerations must be taken into account when the key is initially generated. It will be possible to change the value of byte 11 from 'E' or 'S' to 'N' (so that no further export will be permitted), via an optional field at the end of the command message.

8.5.1.7 Number of Optional Blocks (Bytes 12-13)

The Thales Key Block format allows a key block to contain up to 99 Optional Header Blocks which can be used to include additional (optional) data within a Thales Key Block. Optional Header Blocks are described below.

Bytes 12-13 of the Header specify the number of Optional Header Blocks in the key block. A value of '00' (X'3030) indicates there are no optional blocks. A value of '12' (X'3132) indicates that there are 12 optional blocks.

8.5.1.8 LMK Identifier (Bytes 14-15)

Bytes 14-15 of the Header identify the LMK used to encrypt and authenticate the key block. Such identification is required to support multiple LMKs. The LMK Identifier field may take numeric values '00' to '19' except when exporting, when the value is 'FF'. The value 'FF' is used when exporting since the LMK ID on the recipient payShield may not be the same as that on the payShield used to export the key. The number of LMKs that can be loaded into the HSM is determined by the HSM's license file.

Note: A security setting exists to instruct the HSM to ignore the LMK Identifier stored in the header of Thales Key Blocks. This allows users to share Thales Key Blocks across multiple HSMs without requiring the LMK to be loaded into the same "slot" in all their HSMs.

Example

As an example, a Key Block Header of

00072**V2**T**G**22**N**0033

indicates:

- a Key Block of 72 bytes,
- that the key contained in the Key Block may only be used with the Visa PVV algorithm (the 'V2' field),
- that it is a 3-DES key (the 'T' field),
- that it may only be used for PVV generation ('G'),
- The key version number is '22',
- the key may not be exported (the 'N' field),
- there are no Optional Header blocks in the Key Block, and
- the Key Block has been encrypted/authenticated using the LMK with identifier '33'.

With this example, the use of the Key Block is highly restricted. It cannot even be used to verify PVVs. Users must take considerable care when generating Key Blocks to ensure that the correct Header fields are created. The CS host command allows some modifications to Header fields - but only modifications that further restrict the use of the Key Block.

Carrying on with the above example, the Key Block may have been originally generated with the Mode of Use field (byte 8) set to 'N', meaning that the key could be used for PVV generation or verification, and the

Exportability field (byte 11) set to 'E', meaning that "trusted" export of the key was permitted. Following export of the key, it would exist in two places, namely the PVV generation system and the PVV verification system. Once there, byte 8 could be changed to 'G' or 'V', respectively, and byte 11 set to 'N'. Thereafter the Header values cannot be changed any further.

8.5.2 Optional Header

An Optional Header Block has the following structure:

Byte(s)	Field	Comments
0-1	Identifier	Optional Header Block identifier.
2-3	Length	Optional Header Block length; this field contains the length in bytes of the complete Optional Header Block, represented as a hexadecimal value and ASCII encoded; for example, if the overall length is 24 (decimal), then this is represented as X'18 and encoded as X'3138; if an Optional Header Block contains no data then the length field contains the value X'04 (encoded as X'3034).
4-n (n ≤ 251)	Data	Optional Header Block data

Thus the maximum length of an Optional Header block is 255 bytes (and the minimum length is 4 bytes). The overall length of all the Optional Header Blocks must be a multiple of the encryption block length (8 bytes in the case of 3-DES), which is achieved (if necessary) by the inclusion of a "Padding" block that must be the last Optional Header Block.

Note: In theory, a Key Block may have a maximum of 99 optional blocks. In practice, however, a Key Block may not possess more than one optional block with the same identifier. If a key block (Thales or ANSI X.9.143) contains more than one Optional Header Block with the same identifier then the HSM will return an error:

```
BC – Repeated optional block
```

This error will be returned if any attempt is made to use a host command to generate, import or export such a key block.

8.5.2.1 Optional Header Block Types

The following Optional Header Blocks will be supported in the Thales Key Block. The first three Optional Header Blocks are also supported by ANSI X.9.143:

Optional Header Block ID	Hex	Field Name	Comments
KS	X'4B53	Key Set Identifier	See examples in ANSI X9.24 part 3.
KV	X'4B56	Key Block version	Used to define the version of the set of key block field values and that the key block contains provisional values not yet approved by ANSI. This field must contain 4 printable ASCII characters.

Optional Header Block ID	Hex	Field Name	Comments
PB	X'5042	Padding block	Used to ensure that the overall length of the Optional Header Blocks is a multiple of the encryption block length; the data field is filled with readable (random) ASCII characters; if used, the Padding block must be the last Optional Header Block.
00	X'3030	Key Status	Key status; permitted values: 'E' (Expired) 'L' (Live) 'P' (Pending) 'R' (Revoked) 'T' (Test) See notes below on permitted Key Status transitions.
01	X'3031	Key Block Encryption	Algorithm and mode used to encrypt the Key Data in the key block; only permitted value: '00' (current mechanism).
02	X'3032	Key Block Authentication	Algorithm and mode used to authenticate the key block; only permitted value: '00' (current mechanism).
03	X'3033	Start Date/Time	Date and time from which the key block is valid; format: YYYY:MM:DD:HH
04	X'3034	End Date/Time	Date and time after which the key block is invalid; format: YYYY:MM:DD:HH
05	X'3035	Text	Any combination of printable characters; a zero length data field will not be permitted

The 'KS', 'KV' and 'PB' optional blocks are inherited from the ANSI X.9.143 standard, but the remaining optional block types are proprietary to the Thales key block. The '01' and '02' optional blocks are for future-proofing, in the event that the LMK is changed to another algorithm (as happened with AES).

For example, a key block that contains a "live" key could have an Optional Header Block structure as follows (the space is to aid readability and is not part of the Optional Header Block):

```
0005L PB0Brrrrrrr
```

In this case, the "Number of Optional Blocks" field in the key block header (bytes 12-13) will contain the value '02' (X'3032).

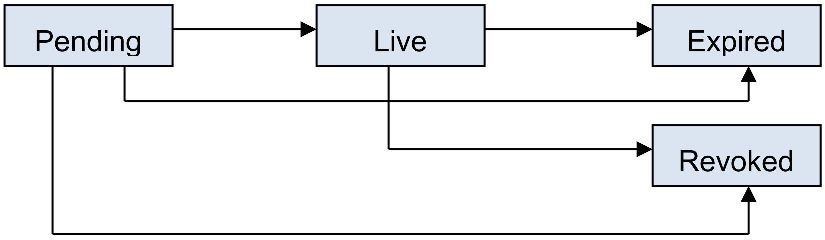
The Padding block (containing 7 random padding characters in this case) is required to ensure that the total length of the Optional Header Block is a multiple of 8 bytes. Note that any optional block must be at least 4 bytes in length.

Notes:

- The order in which optional blocks appear in a key block is immaterial, except that 'PB' Padding block must be the last optional block.
- Optional Header Blocks with identifiers '01' and '02' will not normally be used, but are included in the event that multiple encryption and authentication modes may be required in the future. If they are used then they must have data value '00'.

8.5.3 Permitted Key Status transitions

The following transitions (only) will be permitted for Key Status:



Notes:

- The host command CS may be used to modify the status of a key.
- The HSM will not permit the use of a key that is marked as "Pending", "Expired" or "Revoked".
- Keys with status "Test" or "Live" may be used as normal by an HSM.

8.6 Encrypted Key Data

The only part of the key block that is encrypted is the Key Data, which contains the actual key stored in the key block. The key types that may be protected in a Thales Key Block are DES and TDES keys, HMAC keys, AES keys, and RSA public and private keys. Note that an RSA public key is not encrypted, but the key block is still authenticated.

The encryption algorithm used to protect the Key Data depends on the specific Key Block scheme being used:

- When using a DES Key Block LMK, the Key Data portion of the key block is encrypted using 3-DES Cipher Block Chaining (CBC), using bytes 0-7 of the Header as the Initialization Vector (IV). The encryption key will be a variant of the LMK.
- When using an AES Key Block LMK, the Key Data portion of the key block is encrypted using AES Cipher Block Chaining (CBC), using bytes 0-15 of the Header as the Initialization Vector (IV). The encryption key will be cryptographically derived from the LMK.

Note: An Optional Header Block will be reserved so that other encryption algorithms and modes of encryption can be supported in the future.

The Key Data block has the following format:

Field	Length	Notes
Key Length	2 bytes	Contains the length in bits of the key that is to be encrypted (see next field); the length is written as a 16-bit binary number; for example, if the key is a 192-bit

Field	Length	Notes
		(triple-length) 3-DES key then this field contains the value X'00C0.
Key	variable, depending on key that is being encrypted	Contains the key data, in binary format; for example, a 192-bit (triple-length) 3-DES key would be represented as 24 bytes.
Padding	Variable	Contains random padding, used to ensure that the length of the entire Key Data block is a multiple of the block length of the encrypting key; for example, if a 3-DES key is used as the encryption key then the Key Data block must be a multiple of 8 bytes, so with the examples above the padding field could contain 6, 14, 22,.. bytes; the padding field can be used to disguise the true length of the key in the key block, if required.

A security setting (e.g. in the CS console command) allows users to ensure that all double-length keys in Thales Key Blocks are padded with an additional 8 bytes of random values to disguise the length of the key. If this option is not selected then the minimum padding necessary will be applied in all cases.

Note: The minimum padding necessary for AES, RSA and HMAC keys will always be applied, regardless of this security setting.

8.7 Authenticator

The key block Authenticator ensures the integrity of the key block, and is calculated over the Header, Optional Header Blocks and the Key Data. The authentication algorithm used depends on the specific Key Block LMK being used:

- When using a DES Key Block LMK, the key block Authenticator is calculated using a 3-DES CBC-MAC, with a zero IV. (No padding is required, as the data to be authenticated is always a multiple of 8 bytes in length.) The leftmost 4 bytes of the result will be used as the Authenticator. The authentication key will be a variant of the LMK.
- When using an AES Key Block LMK, the key block Authenticator is calculated using an AES CMAC over the clear key block. The leftmost 8 bytes of the result will be used as the Authenticator. The authentication key will be cryptographically derived from the LMK.

Note: The Key Scheme Tag 'S' is not included in the authenticated data.

8.8 Key Block Local Master Keys (LMKs)

A Key Block LMK is a either a triple-length DES key, or a 256-bit AES key, and is used to encrypt keys in a key block format. It is not compatible with a Variant LMK, and it can only be used to encrypt keys in key block form.

Note: The term "Key Block LMK" refers to the "key block" method of encrypting keys; a Key Block LMK is not itself stored in key block form.

8.8.1 3DES Key Block Test LMK

The value of the LMK contained in the 3DES Key Block Test LMK smartcard is shown in the table below. The two Passwords are also held in this device, and their values are also shown below.

The PIN for the 3DES Key Block Test LMK smartcard is: 1 2 3 4

LMK	01 23 45 67 89 AB CD EF 80 80 80 80 80 80 80 80 FE DC BA 98 76 54 32 10
Password 1	89 89 89 89 89 89 89 89
Password 2	THEQUICKBROWNFOX

The check value is 165126.

8.8.2 AES Key Block Test LMK

The value of the LMK contained in the AES Key Block Test LMK smartcards is shown in the table below. The AES Key Block Test LMK can only be used in smartcard authentication mode, so password authentication is not supported.

The PIN for each AES Key Block Test LMK smartcard is: 1 2 3 4

LMK	9B 71 33 3A13 F9 FA E7 2F 9D 0E 2D AB 4A D6 78 47 18 01 2F 92 44 03 3F 3F 26 A2 DE 0C 8A A1 1A

The check value is 9D04A0.

8.9 Console Command examples

8.9.1 CK Console Command (Generate Check Value)

In this first example, the CK console command is used to generate a check value for a key contained in a Key Block. No additional fields are required.

```
Online> CK <Return>

Enter LMK id: 03 <Return>

Enter Key Block: S 00072V2TG22N0003xx………xxxxx <Return>


Key check value: cccccc
Online>
```

8.9.2 KG Console Command (Generate Key)

In this example, the console KG command is used to generate a non-exportable, double length Base Derivation Key (BDK) used in the DUKPT scheme. The Key Block will contain start and end date/time optional blocks.

```
Online> KG <Return>

Enter LMK id: 07 <Return>

Enter key length [1,2,3]: 2 <Return>

Enter key scheme (LMK): S <Return>

Enter key scheme (ZMK): <Return>

Enter ZMK: <Return>

Enter key usage: B0 <Return>

Enter mode of use: N <Return>

Enter key version number: 00 <Return>

Enter exportability: N <Return>

Enter optional blocks? [Y/N]: Y <Return>

Enter optional block identifier: 03 <Return>

Enter optional block data: 2005:12:21:00 <Return>

Enter more optional blocks? [Y/N]: Y <Return>

Enter optional block identifier: 04 <Return>

Enter optional block data: 2007:12:21:00 <Return>

Enter more optional blocks? [Y/N]: N <Return>


Key under LMK:
S 00112B0TN00N030703112005:12:21:0004112007:12:21:00PB06rrxx……xxxxx

Key check value: cccccc
Online>
```

In the above example, the Key Usage value 'B0' indicates that the key is a BDK and the Mode of Use value 'N' means that there are no specific restrictions on the use of the key.

8.10 Host Commands

Note: All Host commands are disabled by default.

8.10.1 Host Command Examples

Refer to the *payShield 10K Core Host Commands reference manual* for examples of Host Commands and Responses when Key Block LMKs are being used.

8.10.2 Error Codes

Implementation of Key Blocks is a complex matter, so a large number of error codes are provided to aid development and testing. For example:

- In an A6 command where a Key Usage of 72 has been specified, if a Mode of Use value other than 'N' was input an error code would be generated (as 'N' is the only permitted value when the Key Usage is '72').
- Similarly, if an Exportability value 'E' was supplied then an error would be generated (since the imported key is untrusted).

Some of the more obvious (generic) key block error conditions include key block authentication failure, the use of incompatible key schemes or LMK identifiers, the attempted use of an expired or revoked key and the attempted use of an exportability mode that is not permitted. In addition, when used with specific commands only certain Header values are allowed; an invalid Header will cause an error code or error message to be returned.

9 Multiple LMKs

9.1 Introduction

The LMK (Local Master Key) on the payShield 10K is used to protect (by encryption) all of the operational keys plus some additional sensitive data that are processed by the HSM.

The payShield has the capability of Multiple LMKs, such that up to 20 LMKs of different types can be in use at any one time. Each LMK can be managed by a separate security team, allowing a single payShield 10K to be used for multiple purposes – such as different applications or different clients.

9.2 Need for Multiple LMKs

9.2.1 About LMKs

The LMK is stored in the tamper-resistant memory of the HSM. All other cryptographic keys used in the system are encrypted under the LMK and are usually stored externally to the HSM (See Section 13, User Storage), usually in a key database on the host system that is accessible by host applications. The LMK may be common to a number of HSMs, so that applications do not need to concern themselves with the particular HSM used for cryptographic processing. Since there is only a single key stored in the HSM, in the event of a problem with the device, recovery can be effected quickly and with minimal disruption to operations.

There are two types of LMK:

- Variant LMK
- Key Block LMK Multiple LMKs

9.2.2 Multiple LMKs

The LMK mechanism is simple, easy to understand and secure, but it does have some limitations. In particular, the use of a single LMK within an HSM (or cluster of HSMs) means that it is not possible to achieve cryptographic isolation of applications that are calling the HSM. This may breach the requirements of some PCI security standards and may be a cause for other concerns.

For example:

- In a bureau situation, there is generally a need to support keys from many different clients, yet maintain cryptographic isolation of such keys; the use of multiple LMKs is one way to achieve this.
- Some users may wish to use the HSM in both a live operational environment and a test or development environment (not recommended but may be a necessity in some cases).

The availability of Multiple LMKs also makes it easier to migrate operational keys from an old LMK to a new one. Such LMK migration should be performed every few years for security purposes but may also be necessary for operational reasons – for example when upgrading from double- to triple-length Variant LMKs or from Variant LMKs to Key Block LMKs.

Although the payShield 10K allows for changing the LMK, it means that all operational keys need to be translated from encryption under the old LMK to encryption under the new LMK before they can be used. A "big bang" approach typically requires very careful planning and coordination, with possible downtime or need for additional HSM capacity. Therefore, some HSM users are reluctant to change the LMK.

The use of multiple LMKs allows users to adopt a phased approach to LMK change.

9.3 Multiple LMK Licensing

All payShield 10K HSMs are delivered with the capability to support 2 LMKs – one Variant LMK plus one Key Block LMK.

Additional optional licenses can be added to the payShield 10K to enable use of up to 20 LMKs, of any mix of Variant and Key Block LMKs, on a single payShield 10K.

9.4 Managing Multiple LMKs

9.4.1 LMK Component Generation

LMK components are generated using the GK console command. This command also stores the component(s) to a smart card.

Note: This command may be used to generate components for the following types of LMKs:

- Double-length (2DES) Variant LMK
- Triple-length (3DES) Variant LMK
- Triple-length (3DES) Key Block LMK
- 256-bit AES Key Block LMK.

When creating a Variant LMK or a 3DES Key Block LMK, this command generates the data for a single LMK component card.

When creating an AES Key Block LMK, this command generates the data for all the required number of LMK component cards.

```
Secure> GK <Return>

Variant scheme or key block scheme? [V/K]: K <Return>

Key status? [L/T]: L <Return>

LMK component set [1-9]: 1 <Return>

Enter secret value A: <Return>

Enter secret value B: <Return>

Enter secret value C: <Return>
```

```
Insert blank card and enter PIN: ******** <Return>

Writing key

Checking key

Device write complete, check: 012345

Make another copy? [Y/N]: Y <Return>

…

Secure>
```

Equivalent functionality when using payShield Manager is found under Operational / Local Master Keys.

9.4.2 Migration from Old to New LMK

Once the LMK has been loaded, an "old" LMK can be loaded into key change storage (e.g. using the console LO command) and operational keys migrated from encryption under the old LMK to encryption under the new. Keys can be migrated from Variant > Variant LMK, Variant > Key Block LMK or Key Block > Key Block LMK, but not from Key Block > Variant LMK.

```
Secure-AUTH> LO <Return>

Enter LMK id: 02 <Return>

Enter comments: Old LMK for PQR Bank <Return>

Load old LMK from components

Insert card and enter PIN: ******** <Return>

Check: 678901

Load more components? [Y/N]: Y <Return>

…

Check: 085392

Load more components? [Y/N]: N <Return>

LMK id: 02

LMK key scheme: Key block

LMK algorithm: 3DES (3key)

LMK status: Live

Comments: Old LMK for PQR Bank

Confirm details? [Y/N]: Y <Return>
```

```
…
Secure-AUTH>
```

Equivalent functionality when using payShield Manager is found under Operational / Local Master Keys.

Once the old and new LMKs are loaded in the correct locations, migration of keys from old to new LMK (and from variant to key block, if required) can take place. The same mechanism can be used to provide a phased migration of keys from one LMK to another. In this case, the "old" key may continue to be used as an operational LMK at the same time as keys are migrated to encryption under a new LMK. This mechanism allows, for example, existing keys for different applications to be cryptographically isolated from each other.

9.5 Authorization

Authorization is required to enable certain console and host commands to be executed. Either the whole HSM can be put into Authorized State, which enables all commands requiring authorization to be executed; or the HSM can be configured to support Multiple Authorized Activities: in this mode, authorization can be applied to individual activities or groups of activities.

To set authorization, two smartcard holders must authenticate themselves to the HSM by presenting the smartcards and entering the smartcards' PINs. These PINs are set up when the smartcards are formatted (e.g. by using the FC console command).

With multiple LMKs, each LMK will have its own authorizing smartcards and PINs. This means that commands can be authorized for a specific LMK. So, for example, in the previous section when using the LO command for loading an "old" LMK in key change storage, LMK with identifier=02 has been specified: authorization requires the use of the authorization smartcards and PINs used when setting up LMK 02 and so only permits the old LMK to be loaded in key change storage slot 02.

This technique provides for highly granular control over sensitive LMK operations. The A console command can be used to see which activities have been authorized:

```
Online> A <Return>
Enter LMK id: 03 <Return>
No activities are authorized for LMK id 03
List of authorizable activities:
…
…
The following activities are pending authorization for LMK id 03:
pin.mailer
First Officer:
Insert card for Security Officer and enter the PIN: ******** <Return>
```

```
Second Officer:
Insert card for Security Officer and enter the PIN: ******** <Return>
The following activities are authorized for LMK id 03:
pin.mailer


Online-AUTH>
```

In this example, only PIN mailer printing is authorized and then only for the LMK with identifier 03. Equivalent functionality when using payShield Manager is found under Operational / Local Master Keys.

Note: Different formats are used for LMK storage and saving HSM settings. payShield Manager cards do not need to be formatted.

9.6 LMK Table

From a management perspective, it is important to know which LMKs have been loaded into the HSM (and in which locations). This information can be displayed using the VT console command:

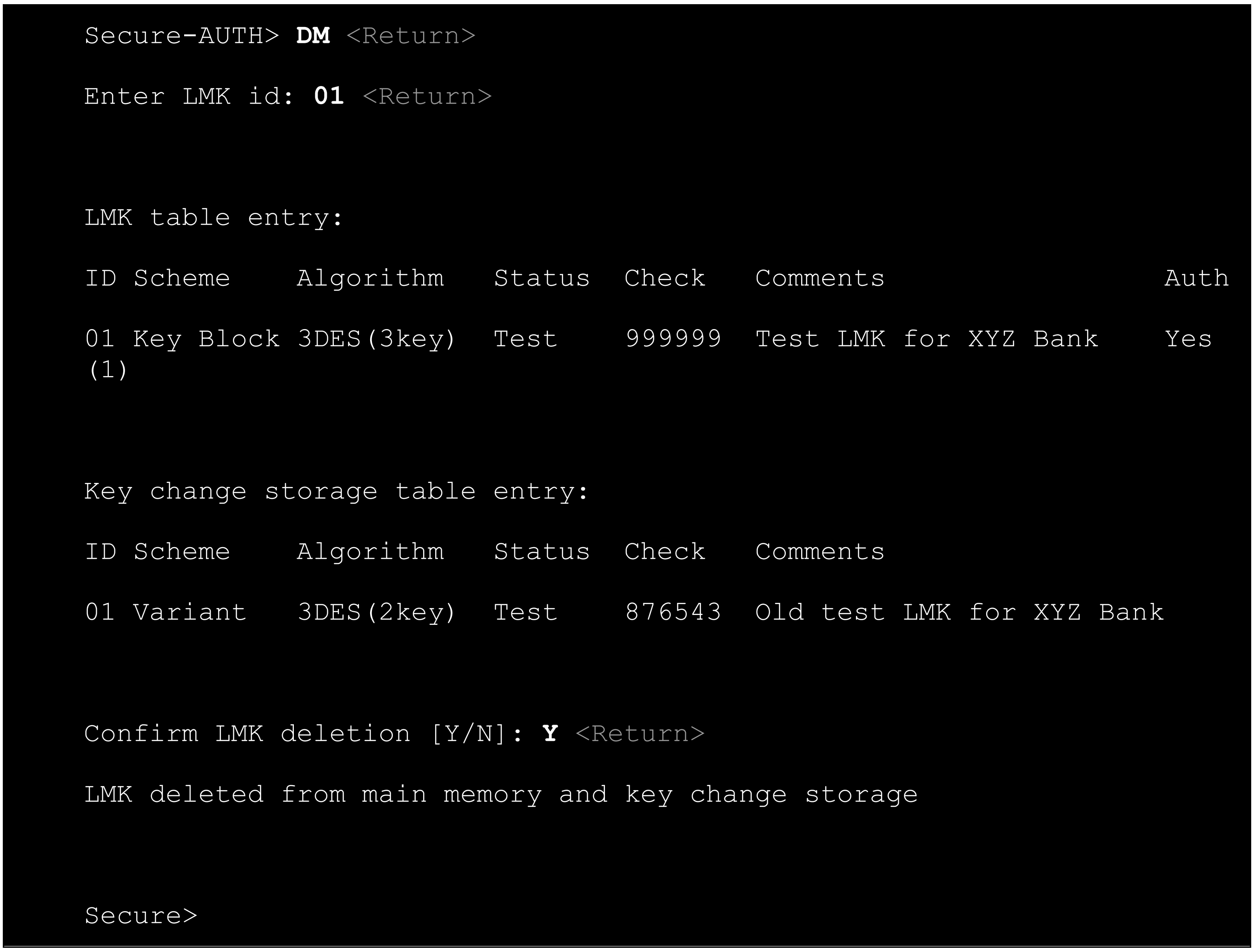
```
Online> VT <Return>


LMK table:
ID Authorized  Scheme    Algorithm  Status Check  Comments
00 Yes(1H,1C)  Variant   3DES(2key) Test   268604 For RST Bank
01 No          Key block 3DES(3key) Test   999999 For XYZ Bank
02 Yes(4H,0C)  Key block AES-256    Live   324365 For PQR Bank
03 Yes(0H,1C)  Key block AES-256    Live   963272 Mngmnt LMK


Key change storage table:
ID             Scheme    Algorithm  Status Check  Comments
01             Variant   3DES(2key) Test   876543 For XYZ Bank
02             Key block 3DES(3key) Live   085392 For PQR Bank


Online>
```

As can be seen, details of operational LMKs and any LMKs that have been loaded into key change storage are displayed. The second column of the LMK table indicates whether any activities are authorized for that LMK and, if so, how many ('H' = Host commands, 'C' = Console commands).

Equivalent functionality when using payShield Manager is found under Operational / Local Master Keys.

9.7 Deleting LMKs

With multiple LMKs in the HSM, it is necessary to be able to delete individual LMKs from the HSM's memory without deleting any other LMK. This is achieved using the DM console command:



```
Secure-AUTH> DM <Return>

Enter LMK id: 01 <Return>


LMK table entry:

ID Scheme    Algorithm   Status  Check   Comments                 Auth

01 Key Block 3DES(3key)  Test    999999  Test LMK for XYZ Bank    Yes
(1)


Key change storage table entry:

ID Scheme    Algorithm   Status  Check   Comments

01 Variant   3DES(2key)  Test    876543  Old test LMK for XYZ Bank


Confirm LMK deletion [Y/N]: Y <Return>

LMK deleted from main memory and key change storage


Secure>
```

As can be seen from the above example, if an LMK is deleted then any LMK in the corresponding "slot" in key change storage is also deleted.

Similarly, individual LMKs that have been loaded into key change storage may be deleted without deleting the live LMK by using the DO console command.

Equivalent functionality when using payShield Manager is found under Operational / Local Master Keys.

9.8 Identifying the Required LMK

Clearly, operational commands need to know which LMK should be used for processing.

9.8.1 Console Commands

Console commands are identified using the LMK identifier.

The Generate a Check Value (CK) console command is used to generate a key check value (KCV) for a key encrypted under a specified LMK.

```
Online> CK <Return>

Enter LMK id: 03 <Return>

Enter key block: S 0xxxxxxxxxxxxx03xx………xxxxx <Return>


Key check value: CCCCCC

Online>
```

In the above example, the LMK with identifier 03 is a Key Block LMK and the command returns the check value for the key contained within the key block. Note that the key block itself also includes the identifier of the LMK used to encrypt and authenticate the key block, in bytes 14-15. If this value does not agree with the value entered by the user, then an error message will be displayed.

9.8.2 Host Commands

For host commands, the situation is somewhat more complicated, and a number of mechanisms can be used to indicate which LMK to use in the processing of the command.

9.8.2.1 Key Block LMKs

One of the fields in a key block identifies the LMK used to encrypt and authenticate the key block (see bytes 14-15 of the key block in the earlier example).

The HSM can be set to ignore this field, using the security setting accessed via the CS console command ("Configure Security") or Configuration / Security Settings / Initial in payShield Manager.

Normal behavior is to not ignore this field, and so if Thales Key Blocks are presented in a host command, the LMK identified in the key block header(s) will be used, and if key blocks with different LMK identifiers are presented in the same command then the HSM will return an error.

However, if the HSM is set to ignore this field, then the method of identifying the LMK to use in with a command is identical to when a Variant LMK is used.

9.8.2.2 Default LMK

One LMK is identified as the "default" LMK and, in the absence of any other indication in the command message, it is assumed that the default LMK is being used. The principal benefit of this mechanism is that it provides a backwards-compatibly for host applications written without consideration of the multiple LMK capability.

The identifier for the default LMK (which may be the same as the management LMK – see below) is defined using the CS console command ("Configure Security") or Configuration / Security Settings / General in payShield Manager.

9.8.2.3 Management LMK

One LMK is designated as the "management" LMK and is required for a small number of host commands, mainly associated with the HSM's audit functions.

The identifier for the management LMK (which may be the same as the default LMK discussed above) is defined using the CS console command ("Configure Security") or Configuration / Security Settings / General in payShield Manager.

9.8.2.4 Explicit LMK Identifier in Host Commands

An explicit mechanism that can be used is simply to include optional fields at the end of the command message to identify the correct LMK to use when processing the command. This technique can be used if there are no other fields in the command message that identify the LMK – for example, when using the A0 command to generate a key or when using a Variant LMK, which is neither the default LMK nor the management LMK. The optional fields are a delimiter ('%') and the LMK identifier:

Field	Length & Type	Details
Delimiter	1 A	Optional; if present the following field must be present; value '%'
LMK Identifier	2 N	LMK identifier; permitted values '00' to '99'; must be present if the above Delimiter is present

If there is a "mis-match" between the LMK identifiers in a command then the HSM will return an error. This could happen, for example, if:

- no LMK is loaded in the identified location, or
- key blocks contain different identifiers, or
- the optional fields are present at the end of the command message and indicate an LMK that is different from the LMK identified elsewhere in the command message.

The explicit identification of the required LMK in the host command normally must match the LMK identification using the TCP Port number (see below). However, when the payShield 10K security setting ("Ensure LMK Identifier in command corresponds with host port") is set to "No", the HSM will ignore any mismatches, and just use the explicit identification of the LMK in the host command.

9.8.2.5 Ethernet TCP Port

An additional mechanism is available for Ethernet-attached host computers. The HSM can infer the LMK Identifier to use for a particular command from the TCP port on which the command is received.

When configuring the payShield 10K's host ports, a "Well-Known Port" is specified for host command traffic arriving from the host: by default, this is 1500 (or 2500 if Secure Host Communications is being used). In the absence of any other indicator of the LMK to be used:

- host commands directed to the Well-Known Port will use the Default LMK;
- host commands directed to [Well-Known Port +1] will automatically use LMK Id 00;
- host commands directed to [Well-Known Port +2] will automatically use LMK Id 01;
- more generally, host commands directed to [Well-Known Port + n, where n□1] will automatically use LMK Id [n-1].

The situation for an HSM using the default W ell-Known Port value of 1500 is summarized in the table below:

Command received on TCP Port	LMK Used
1500	Default LMK Id
1501	LMK Id 00
1502	LMK Id 01
1503	LMK Id 02
…	…

The explicit identification of the required LMK in the host command (as described in the previous section) normally must match the LMK identification using the TCP Port number. However, when the payShield 10K security setting ("Ensure LMK Identifier in command corresponds with host port") is set to "No", the HSM will ignore any mismatches, and just use the explicit identification of the LMK in the host command.

9.9 Settings per LMK

9.9.1 Summary

A new feature included in payShield 10K software release 2.0a and above allows certain settings to be configured for each LMK. In previous versions, settings applied to all LMKs installed on payShield 10K.

By providing the ability to configure unique settings on a per LMK basis on a single payShield, payShield 10K customers can consolidate their payment’s workloads and use their payShield estate(s) more efficiently.

For example, by using the new Settings per LMK feature, a PIN length of 4 can be configured for "LMK ID 1" and a PIN length of 5 can be configured for "LMK ID 2". Previously separate payShield 10Ks would have had to be used for these separate settings.

The settings that can be selected per LMK are:

- Selected security settings to be set per LMK

- Host Commands to be enabled and disabled per LMK
    - Note that Console Commands are enabled or disabled on a per HSM basis as before.
- PIN Block Formats to be enabled and disabled per LMK

Some important restrictions to be aware of when using Settings per LMK:

- The LMK ID to be used by the Host Commands is defined by the host port
    - The alternative methods, specifying the LMK at the end of the host command using a '%' delimiter, or specifying the LMK ID in the Thales Key Block are not supported when using settings per LMK.
- A TCP or TLS connection is required
    - UDP and FICON are not supported when using Settings per LMK
- PCI HSM approval is assessed on an HSM basis
    - This means that for PCI HSM compliance, the Security Settings for all LMK IDs must comply to PCI requirements.
- The second and third smart cards for the LMK components must be managed by different security officers

9.9.2 Configuration of Settings per LMK

A new Security Setting is provided to allow the new "Settings per LMK" to be enabled. payShield Manager or the Console can be used for this.

For payShield Manager, the setting is included in the Security Settings option in the Configuration Tab. Please refer to the *payShield 10K Installation and User Guide*, Chapter 9.

For the Console, please refer to the 'CS' Configure Security command in the *payShield 10K Console Guide*.

9.9.3 Security Settings

Once "Settings per LMK" is enabled, the Security Settings can be changed for each LMK ID. There are a small number of security settings that are applicable to payShield 10K as a whole and these remain applicable to the HSM. These are:

- Echo
- User storage key length
- Display general information on payShield Manager Landing page?
- Solicitation batch size
- Use default card issuer password
- Default LMK identifier
- Management LMK identifier
- Enable settings per LMK?

The settings can be changed using either payShield Manager or the console.

For payShield Manager, the Security Settings option in the Configuration Tab is used. The LMK ID required to be used for the update to the security settings is selected first using a drop-down menu and then the relevant security settings can be viewed or updated as required.

For the Console, 'QS' (Query Settings) and the 'CS' (Configure Security) commands are used to view and update the Security Settings on a per LMK basis as required. In both cases, the LMK ID required is selected as the first prompt and then the relevant security settings can be viewed or updated as required.

For security reasons, changes to some of the Security Settings require the LMK to be deleted. When operating with "Settings per LMK" enabled, if one of these settings is required to be changed, it is only the relevant LMK ID that is deleted and is required to be reinstalled – the other LMK IDs remain in place.

Some additional points to note when enabling Settings per LMK:

- UDP and FICON protocols must be disabled and either TCP and/or TLS enabled
- When enabling "Settings per LMK", the settings for each LMK ID are the same as those specified before "Settings per LMK" was selected.
- Enabling settings per LMK automatically deletes all the LMKs and so these are then required to be reinstalled.
- The settings used if "Settings per LMK" is disabled, are taken from those used from the default LMK. The LMKs will need to be reinstalled again in this case

9.9.4 Security Settings and PCI HSM Compliance

For the HSM to be in PCI Compliant Mode, the security settings for all LMK ID must be set to comply with PCI requirements. The relevant security settings are described in the *payShield 10K Security Operations manual*.

When "Settings per LMK" is first enabled, each LMK will have the same settings that were in use previously. This implies that if the security settings were set to comply with PCI HSM requirements, then the settings for all LMK ID will comply with PCI HSM requirements and the HSM will continue to be PCI HSM compliant.

During operation with "Settings per LMK" enabled with PCI HSM compliant settings, to configure one of more LMK ID to be non-PCI HSM compliant, all LMKs will be deleted and must be reinstalled.

While operating with "Settings per LMK" enabled, with settings not PCI Compliant, the settings for each LMK ID must be set to comply before the HSM will enter PCI HSM compliant mode.

9.9.5 PIN Block Formats

Once "Settings per LMK" is enabled, the PIN Block Format settings can be changed for each LMK ID.

For payShield Manager, the setting is included in the PIN Blocks tab in the General Settings option in the Configuration Tab. The LMK ID is selected first using a drop-down menu and then the relevant PIN Block Formats can be viewed or updated as required.

For the Console, the 'CONFIGPB' (Configure PIN Blocks) command is used to view and update the PIN Block Settings on a per LMK basis. The LMK ID required is selected as the first prompt and then the relevant PIN Block settings can be viewed or updated as required.

9.9.6 Enabling and Disabling Host Commands

Once "Settings per LMK" is enabled, Host Commands can be enabled and disabled for each LMK ID.

For payShield Manager, the setting is included in the Configure Commands option in the Configuration Tab. The LMK ID is selected first using a drop-down menu and then the settings for the relevant Host Commands can be viewed or updated as required.

For the Console, the ‘CONFIGCMNDS’ (Configure Commands) command is used to view the settings and enable / disable host commands as required. The LMK ID required is selected as the first prompt.

9.9.7 Saving Configuration to Smart Card

With Settings per LMK enabled, settings are saved separately for each LMK ID to smart card. The LMK ID is selected when storing or retrieving the settings using payShield Manager or the Console.

The settings that are saved to each smart card for the selected LMK ID are:

- Settings that are separately specified per LMK ID
- Settings that are applicable to payShield 10K as a whole i.e.:
  - Echo
  - User storage key length
  - Display general information on payShield Manager Landing page?
  - Solicitation batch size
  - Use default card issuer password
  - Default LMK identifier
  - Management LMK identifier
- Note that the “Settings per LMK” setting itself is not saved to smart card.

When the settings are loaded, only the LMK for the LMK ID that is being restored is deleted – the other LMK IDs remain in place.

The settings can be saved to smart card when “Settings per LMK” is enabled and restored when “Settings per LMK” is disabled. In this case the settings restored will apply to all LMK ID.

The settings can also be saved to smart card when “Settings per LMK” is disabled and restored when “Settings per LMK” is enabled. In this case the settings restored will apply only to the LMK ID selected.

9.9.8 Adding Additional Multiple LMK Licenses

It is possible to add licences to increase the number of LMKs supported on payShield 10K. If the licence is added when “Settings per LMK” is enabled, then the settings used for the new LMK ID added are those selected for the “Default LMK”. This applies to the Security Settings, the PIN Block settings and the Host Commands settings.

9.9.9 Use of Default and Management LMK ID

There are a number of Console Commands and Host Commands that do not use the LMK. In these cases, the settings used for the Default LMK are used. If Authorization is required for these commands, then the Management LMK is used.

9.9.10 LMK Identifier Security Settings

As noted previously, when "Settings per LMK" is enabled, the LMK ID used for Host Commands is determined from the host port used. There are two Security Settings that determine the functionality if a mismatch is found between the methods used to determine the LMK ID and these are summarized below. The relevant settings are:

- Ensure LMK Identifier in command corresponds with host port: Yes or No

  When set to 'Yes', an LMK Id field within a host command must match the LMK ID implied by the TCP port used otherwise an error is returned.

- Ignore LMK ID in Key Block Header: Yes or No

  When set to 'No', the LMK ID inside the header (bytes 14-15) of Thales Key Blocks must match the LMK ID implied by the TCP port used otherwise an error is returned.

The four possible scenarios are:

1. To ignore the LMK ID both in the Host Command and in the Key Block Header the settings are as follows:
   - Ensure LMK Identifier in command corresponds with host port: No
   - Ignore LMK ID in Key Block Header: Yes
2. To check the LMK ID in Host Command corresponds to host port but ignore LMK ID in Key Block Header the settings are as follows:
   - Ensure LMK Identifier in command corresponds with host port: Yes
   - Ignore LMK ID in Key Block Header: Yes
3. To ignore LMK ID in Host Command but to check the LMK ID in Key Block Header corresponds to host port the settings are as follows:
   - Ensure LMK Identifier in command corresponds with host port: No
   - Ignore LMK ID in Key Block Header: No
4. To check both the LMK ID in Host Command and LMK ID in Key Block Header both correspond to host port the settings are as follows:
   - Ensure LMK Identifier in command corresponds with host port: Yes
   - Ignore LMK ID in Key Block Header: No

9.9.11 Disabling Settings per LMK

The feature can be disabled if required. This is achieved by disabling the option in payShield Manager or the Console Command – please see Section 9.9.2 Configuration of Settings per LMK for more details.

When disabled, there will only be one set of settings available for the whole HSM. The settings used for this will be those defined for the "Default LMK" before the feature was disabled.

If the settings for the "Default LMK" were set to comply with PCI requirements, then after disabling the feature, the HSM will be placed in PCI Compliant Mode. This is the case whether the HSM was in PCI Compliant Mode or not before the feature was disabled.

When disabling this feature, the LMK will be required to be re-installed.

Please note that the feature is also disabled if the software is downgraded to a version that does not support Settings per LMK. In this case, the handling of the settings and the requirement to re-install the LMK are the same as when disabling the feature, as described above.

9.9.12 Additional Information

Additional points to note when “Settings per LMK” is set:

- There is no change to the way overall performance is managed – this continues to be managed on an HSM basis.
- TLS certificates continue to be managed on an HSM basis.
- Host commands will continue to be audited on an HSM basis.
- If “Settings per LMK” is not configured, then the software is backward compatible with one exception. The exception is that there is a minor update to the order of security settings displayed in payShield Manager and the Console to allow the settings applicable to the HSM to be catered for.
- With SNMP, the OID settings for the Security Settings have been updated to support “Settings per LMK”– see the SNMP Security Settings Appendix in the payShield 10K Installatin and User Guide.
- The Global List of Weak PINs loaded using Host Command ‘BM’ are set per HSM, as before.
- User storage remains on a per HSM basis. The user continues to determine what is stored at (and retrieved from) different user storage locations.

10 Migrating LMKs

10.1 Introduction

Thales payment HSMs have always provided a facility to migrate between LMKs - i.e. to re-encrypt operational keys and other data from encryption under one (old) LMK to encryption under another (new) LMK. The need to do this is more important than in the past because:

- Card schemes are requesting that customers change their master keys every 2 years.
- Adoption of Key Block LMKs, with their added security, requires a migration from Variant LMKs.

This chapter outlines the migration process.

10.2 Multiple LMKs

By default, the payShield 10K is delivered with the ability to install 1 or 2 LMKs. If 2 LMKs are installed, one must be a Variant type and one must be a Key Block type (see below).

Each LMK can be managed by its own team of security officers.

The multiple LMK facility can be used to provide separation between multiple clients, applications, or purposes serviced on the same HSM - and they also make the process of migrating LMKs easier.

10.3 LMK Migration Process Overview

The LMK Migration process takes keys which are encrypted under an old LMK and re-encrypts them under a new LMK. Both the old and the new LMKs must be installed in the payShield 10K. There are two types of LMK storage:

- LMK Live storage. Transaction processing and other LMK functions can make use only of LMKs in Live storage.
- Key Change storage. LMKs in Key Change storage cannot be used for any purpose other than as part of the LMK migration process. Where multiple LMKs are deployed, there is one Key Change storage "slot" for each LMK in the Main storage.

There are 2 ways of allocating old/new keys to Main/Key Change storage:

- The new LMK (which has not yet been deployed for live operation) is loaded into Live storage, and the old LMK (which is still being used for live processing) is loaded into Key Change storage using the LO console command (as described later). This means that the payShield 10K being used for migration cannot be used to process transactions until the LMK migration process is completed and the new LMK comes into operational use, but it is then immediately ready to process transactions because the new LMK is already loaded in Live storage.
- The old LMK (still being used for live operation but about to be obsoleted) is left in Main Live, and the new LMK (which has not yet been deployed for live operation) is loaded into Key Change storage using the LN console command (as described later). This is a more recent option and means that the payShield 10K can continue processing transactions using the current LMK at the same time as it is

used for migrating keys to the new LMK. On the other hand, when the new LMK is ready to go live, the new LMK must be loaded into Live storage before any transactions can be processed.

The steps needed to migrate from an old LMK to a new LMK are:

1. Create smartcards with components for the new LMK.
2. Load the new LMK (from components cards) into either LMK Live storage or LMK Key Change storage.
3. Either:
    - leave the old LMK in LMK Live storage and load the new LMK (from component cards) into LMK Key Change storage, or
    - load the new LMK (from component cards) into LMK Live storage and load the old LMK (from components cards) into LMK Key Change storage in the same HSM.
4. Re-encrypt the operational keys from the old LMK to the new LMK and hold these in a pending new key database.
5. Re-encrypt PINs from the old LMK to the new LMK and hold these in a pending new PIN database.
6. Re-encrypt decimalization tables from the old LMK to the new LMK and hold these in a pending new decimalization table database.
7. If the new LMKs have been loaded into Key Change storage, re-load them into Live storage.
8. Make the pending key/PIN/decimalization table databases the live databases.

The following sections describe each of these options.

Note: Subsequent sections look at the following associated considerations:

- Migrating from Variant to Key Block LMKs
- Key type changes for PCI HSM compliance.
- The Multiple LMK capability

10.3.1 Generating New LMK Component Smartcards

LMKs are set up in the payShield 10K by loading a number (typically 3) of components which are then combined within the HSM to form the LMK. (The formed LMK is never available outside of the HSM.) The LMK components are loaded from LMK smartcards.

The first stage, therefore, is to create smartcards which have the components for the new LMK. These components have completely random values, and are created on any payShield 10K.

Each component must be held by a different security officer, and access to the component cards must be securely controlled (e.g. by storing the card securely and requiring security officers to check the cards out and in).

All component cards are required to load (or form) an LMK, and so loss of any card or absence of a card holder prevents the LMK being loaded (or re-loaded at a later date if necessary). Therefore at least one backup should be made of each component card.

Note that the terms "LMK card" and "LMK component card" are interchangeable. Only LMK components are ever written to cards - the whole LMK is never written to a card.

10.3.2 Types of LMK Component Cards

There are two types of LMK component cards:

- HSM LMK cards - using the card reader built into the HSM. This type of card is created and used by operators using a console and the HSM card reader.
- payShield Manager RLMK cards - created by operators using payShield Manager and the card reader attached to the remote management PC.

The principles are the same for both types of card, although the detail of the processes is different. The two types of card are incompatible, although either type of card can be created from the other.

10.4 Formatting LMK Smart Cards

10.4.1 HSM LMK Cards

Before they can be written to, smart cards must be formatted.

Cards which have been used previously and are no longer required can be re-formatted to enable the new components to be written to them.

Do not re-format the component cards for the old LMK that you are about to migrate from - you will be needing them !

Each component holder should format their own card plus at least one backup per component.

HSM LMK smartcards are formatted using the FC console command (described in the *payShield 10K Console Guide*).

10.4.2 payShield Manager LMK Cards

With payShield Manager, the LMK components are written to RLMK cards which are provided by Thales. RLMK cards do not require formatting.

10.5 Generating LMK component cards

10.5.1 HSM LMK Cards

Each component holder should now generate a component and write it to their smart card and backup card(s). This is done using the GK console command (described in the *payShield 10K Console Guide*).

Various warnings and errors may be reported during this process. These are easy to understand, and appropriate responses should be made.

10.5.1.1 payShield Manager RLMK Cards

LMK components for use with payShield Manager are written to RLMK cards using the Generate button in payShield Manager's Operational / LMK Operations / Local Master Keys tab. These cards use a quorum (i.e. "m of n") approach to define how many of the cards must be used when loading an LMK. The operator provides the following information when generating the LMK:

- Number of LMK shares, i.e. 'n' (Default: 2)
- Number of shares to rebuild, i.e. 'm' (Default: 2)
- Key scheme (Variant or Key Block)
- Algorithm
- Status (Live or Test)

For more detailed information regarding payShield Manager, see the *payShield 10K Console Guide*.

10.6 Creating Copies of LMK Component Cards

Because all component cards are needed when the LMK is loaded, copies of each LMK card should be made to allow for misplacement or for issuing to deputies. There are several ways of doing this.

10.6.1 HSM LMK cards

10.6.1.1 During LMK Card Generation

As mentioned above in the section on Generating HSM LMK cards, multiple copies may be made at the time of generating the LMK card.

10.6.1.2 Copying an Existing HSM LMK Card

It is possible at any time to copy an existing HSM LMK card using the DC console command.

10.6.2 payShield Manager RLMK Card

10.6.2.1 Duplicating a payShield Manager RLMK Card

A copy of an existing RLMK component card can be made using the Duplicate button in payShield Manager's Operational / LMK Operations / Local Master Keys tab.

10.7 Loading the New LMK

In the previous section we explained how to create a set of cards containing the components for the new LMK. Each component is "owned" by a different security officer, with no one security officer having access to more than one component. One holder of each of the required number of components must be present to allow the LMK to be loaded onto the payShield 10K using the component smart cards.

The new LMK now needs to be installed into either LMK Live storage or LMK Key Change storage depending on the approach being taken.

The new LMK can be loaded using a Console or payShield Manager.

10.7.1 Using the Console

10.7.1.1 Loading (or forming) the LMK

The LMK is loaded using either:

- the LK console command if the new LMK is to be loaded into LMK Live storage, or
- the LN console command if the new LMK is to be loaded into LMK Key Change storage.

The payShield 10K must be in Secure state. In addition, if the LN console command is being used then the HSM must be in Authorized state. If multiple authorized states is enabled, the activity category is admin (with no sub-category), and the console interface should be selected.

The smart cards used must be HSM cards - not cards created for payShield Manager.

10.7.1.2 Checking the LMK

It is recommended that a check is made that the new LMK has been properly loaded.

This can be done using the A console command, to put the HSM into authorized state (followed by the C command to cancel the authorized state). The A command can be run in any HSM state. The operation of this command depends on whether multiple authorized activities has been enabled in the security settings (e.g. by using the CS console command) – see the *payShield 10K Console Guide* for full details.

10.7.2 Load Using payShield Manager

10.7.2.1 Installing the LMK

The new LMK is loaded using the Install button in the appropriate payShield Manager tab:

- Operational / LMK Operations / Local Master Keys where the new LMK is to be loaded into LMK Live storage, or
- Operational / LMK Operations /Key Change Storage where the new LMK is to be loaded into LMK Key Change storage.

The LMK ID will need to be specified. See the *payShield Manager 10K Installation and User Guide* for more detailed information.

10.7.3 Checking the LMK

The installed LMK can be checked by viewing the LMK list displayed at Operational / LMK Operations / Local Master Keys or Operational / LMK Operations / Key Change Storage.

10.8 Loading the Old LMK

So far we have created a set of cards containing the components for the new LMK, and used them to load into the HSM the "new" LMK that keys and data to be re-encrypted to.

To migrate keys from encryption under an old (current) LMK to encryption under the new LMK, we also need to have the old LMK into the HSM. The old LMK can be left in LMK Lives storage or loaded into LMK Key Change Storage, depending on the approach being taken.

If the old LMK is to be loaded into Key Change Storage, this can be done using a Console or payShield Manager.

10.8.1 Using the Console

The old LMK is loaded into Key Change Storage using the LO (where 'O' is the alphabetic 'O for Oscar') console command.

The payShield 10K must be in Secure state. In addition, the HSM must be in Authorized state. If multiple authorized states is enabled, the activity category is admin (with no sub-category), and the console interface should be selected.

The use of the LO console command is the same as for the LK console command mentioned previously, except that no existing LMK needs to be erased and so you will not be prompted to confirm an erasure.

After loading the old LMK, the HSM should be returned to Online state by turning the physical keys.

10.8.2 Load Using payShield Manager

The old LMK is loaded using the Install button in payShield Manager's Operational / LMK Operations / Key Change Storage tab. This can only be done if there is an LMK with the same ID in the LMK table. See the *payShield 10K Installation and User Guide* for more detailed information.

10.9 Migrating Keys Between Variant LMKs

We now have installed in the HSM both the old LMK that the operational keys are currently encrypted under and the new LMK that they need to be encrypted under for the future. We now need to take each existing operational key in the old key database (encrypted under the old LMK), re-encrypt it using the new LMK, and put it in a new key database.

In order to do this, an application needs to be set up at the host that:

- Takes each operational key (encrypted under the old LMK) from the old key database
- Sends the encrypted key to the HSM using the BW host command.
- Receives the BX response from the HSM containing the operational key encrypted under the new LMK.
- Puts the operational key encrypted under the new LMK into the new key database.

10.9.1 The BW Host Command

Note: In order to migrate Single DES keys from 3DES LMKs to AES LMKs, ensure that the Single-DES security setting is enabled. Security settings are described in the *payShield 10K Security Manual*.

Here we will look at the BW host command as it is used to convert an operational key encrypted under an old LMK of the Variant type to encryption under a new LMK of the Variant type. Other ways of using the BW command are discussed later in this document.

The BW host command automatically adapts its processing depending on where the old and new LMKs are stored:

- If the old LMK was loaded into Key Change storage (e.g. the LO console command was used), then keys and data are re-encrypted from encryption under the (old) LMK in Key Change storage to encryption under the (new) LMK in Live storage.
- If the new LMK was loaded into Key Change storage (e.g. the LN console command was used), then keys and data are re-encrypted from encryption under the (old) LMK in Live storage to encryption under the (new) LMK in Key Change storage.

The table below indicates the structure of the BW host command when it is used in this way.

Field	Length & Type	Notes
Message Header	m A	This field contains whatever the user wants. The length of the field is defined using the *CH* console command or *Configuration / Host Settings* in payShield Manager. It is subsequently returned unchanged in the response to the host.
Command Code	2 A	Must have the value 'BW'.
Key Type code	2 H	Indicates the LMK under which the key is encrypted: '00' : LMK pair 04-05 (Key Type 000) '01' : LMK pair 06-07 (Key Type 001) '02' : LMK pair 14-15 (Key Type 002) '03' : LMK pair 16-17 (Key Type 003) '04' : LMK pair 18-19 (Key Type 004) '05' : LMK pair 20-21 (Key Type 005) '06' : LMK pair 22-23 (Key Type 006) '07' : LMK pair 24-25 (Key Type 007) '08' : LMK pair 26-27 (Key Type 008) '09' : LMK pair 28-29 (Key Type 009) '0A' : LMK pair 30-31 (Key Type 00A) '0B' : LMK pair 32-33 (Key Type 00B) '10' : Variant 1 of LMK pair 04-05 (Key Type 100) '42' : Variant 4 of LMK pair 14-15 (Key Type 402) 'FF' : Use this value where the key type is specified after the first ';' delimiter below. This allows key types other than those listed above to be specified.

Field	Length & Type	Notes
Key length flag	1 N	'0' : for single-length key '1' : for double-length key '2' : for triple-length key.
Key	16/32 H or 1 A + 32/48 H	The operational key to be translated, encrypted under the old LMK.
Delimiter	1 A	Optional. Only present if 'FF' was supplied above for the Key Type code and the following field is present. Value ';'.
Key Type	3 H	Where 'FF' was entered for Key Type Code, this is the 3-digit key type code of the key being translated. For a list of key type codes, see the appropriate table of Key Type Codes in Section *7.4.*
Delimiter	1 A	Optional. If present the following three fields must be present. Value ';'.
Reserved	1 A	Optional. If present must be '0' (zero).
Key Scheme (LMK)	1 A	*Optional. Key scheme for encrypting key under LMK (or '0' (zero)). For a list of key schemes, see the Key Scheme Table in Appendix A - Key Scheme Table*
Reserved	1 A	Optional. If present must be '0' (zero).
Delimiter	1 A	Value '%'. Optional; if present, the following field must be present.
LMK Identifier	2 N	Where the user is using multiple LMKs on the same HSM, this allows the ID of the LMK being migrated to be selected. Minimum value = '00'; maximum value is defined by license. This field must be present if the above Delimiter is present. If the field is not present, then the default LMK will be used.
End Message Delimiter	1 C	Must be present if a message trailer is present. Value X'19'.
Message Trailer	n A	Optional. The contents of the trailer is as required by the user, and is returned unchanged in the response. Maximum length 32 characters.

10.9.2 The BX Response to the Host

In response to the BW host command, the payShield 10K will return the following BX response to the host:

Field	Length & Type	Notes
Message Header	m A	This is a play-back of the header provided in the *BW* command.
Response Code	2 A	Has the value 'BX'.
Error code	2 N	Indicating the general outcome of the *BW* command: '00' : No error '04' : Invalid key type code '05' : Invalid key length flag '10' : Key parity error '44' : migration not allowed: key migration requested when the security setting "Enforce key type 002 separation for PCI HSM compliance" has the value "Y". '45' : Invalid key migration destination key type. '68' : Command disabled or any standard error code (see the payShield 10K Core Host Commands reference manual).
Key	16/32 or 1 A + 32/48 H	The resulting key, re-encrypted under the new LMK.
End Message Delimiter	1 C	Present only if a message trailer is present. Value X'19'.
Message Trailer	n A	This is simply a play-back of any trailer included in the *BW* command.

10.10 Migrating Keys from Variant to Key Block LMKs

Key Block LMKs provide additional security compared to Variant LMKs.

The BW host command already described for Variant LMK > Variant LMK migration can also be used for Variant LMK > Key Block LMK migration. When used for this purpose, the BW command and BX response are modified as indicated below.

Note that it is **not** possible to migrate from:

- Key Block LMKs to Variant LMKs.
- AES Key Block LMKs to TDES Key Block LMKs.

10.10.1 The BW Host Command

The table below indicates the structure of the BW host command when it is used to migrate from Variant-type LMKs to Key Block-type LMKs. Only the differences compared to Variant LMK > Variant LMK migration are described.

Full details on the use of the BW host command can be found in the *payShield 10K Core Host Commands reference manual*.

Field	Length & Type	Notes
Message Header	m A	(As for Variant LMK ⇨ Variant LMK)
Command Code	2 A	Must have the value 'BW'.
Key Type code	2 H	(As for Variant LMK ⇨ Variant LMK)
Key length flag	1 N	(As for Variant LMK ⇨ Variant LMK)
Key	16/32 H or 1 A + 32/48 H	(As for Variant LMK ⇨ Variant LMK)
Delimiter	1 A	(As for Variant LMK ⇨ Variant LMK)
Key Type	3 H	(As for Variant LMK ⇨ Variant LMK)
Delimiter	1 A	(As for Variant LMK ⇨ Variant LMK)
Reserved	1 A	(As for Variant LMK ⇨ Variant LMK)
Key Scheme (LMK)	1 A	(As for Variant LMK ⇨ Variant LMK)
Reserved	1 A	(As for Variant LMK ⇨ Variant LMK)
Delimiter	1 A	(As for Variant LMK ⇨ Variant LMK)
LMK Identifier	2 N	(As for Variant LMK ⇨ Variant LMK)
Delimiter	1 A	Must have value '#'
Key Usage	2 A	The required key usage for the key encrypted under the Key Bock LMK. This information is included in the Key Block header and should be determined using the Key Usage Table. This field determines type of the operational key (e.g. RSA private key, BDK, ZEK), and enforces key separation.
Mode of Use	1 A	The required mode of use for the key encrypted under the Key Bock LMK. This information is included in the Key Block header, and should be determined using the Mode of Use Table This field determines how the operational key can be used (e.g. encryption, decryption, MACing).

Field	Length & Type	Notes
Key Version Number	2 N	A value from '00' to '99', for inclusion in the Key Block header. Determined by the user. '00' denotes that key versioning is not in use for this key.
Exportability	1A	The required exportability for the key encrypted under the Key Bock LMK. This information is included in the Key Block header, and should be determined using the Exportability Table. This field determines how the operational key can be exported (e.g. no export allowed, may only be exported as a Key Block).
Number of Optional Blocks	2 N	A value from '00' to '08', identifying how many optional data blocks the user wants to add into the Key Block Header. Optional data blocks are used to identify parameters (such as key validity dates, key status, algorithm). For a value greater than 0, the following three fields must be repeated for each optional block.
Optional Block Identifier	2 A	The available Optional Block Identifiers (or Types). Note that the value 'PB' may not be used.
Optional Block Length	2H	The length in bytes of the optional block (including the Identifier and Length). A value of X'04' indicates that the block contains only the identifier and length, and so the next field would not be present.
Optional Data Block	N A	The payload of the optional data block.
End Message Delimiter	1 C	(As for Variant LMK ⇨ Variant LMK)
Message Trailer	n A	(As for Variant LMK ⇨ Variant LMK)

10.10.2 The BX Response to the Host

In response to the BW host command, the payShield 10K will return the following BX response to the host:

Field	Length & Type	Notes
Message Header	m A	(As for Variant LMK ⇨ Variant LMK)
Response Code	2 A	Has the value 'BX'. (As for Variant LMK ⇨ Variant LMK)
Error code	2 N	(As for Variant LMK ⇨ Variant LMK)
Key	1 A + n A	The operational key, encrypted under the new Key Block LMK.
End Message Delimiter	1 C	(As for Variant LMK ⇨ Variant LMK)
Message Trailer	n A	(As for Variant LMK ⇨ Variant LMK)

10.11 Migrating Keys Between Key Block LMKs

Migration of operational keys between Key Block LMKs is supported in addition to the Variant LMK > Variant LMK and Variant LMK > Key Block LMK migrations already described. This section describes the BW host command when used for this purpose.

Note that it is not possible to migrate from:

- Key Block LMKs to Variant LMKs.
- AES Key Block LMKs to TDES Key Block LMKs.

10.11.1 The BW Host Command

The table below indicates the structure of the BW host command when it is used to migrate between Key Block-type LMKs. Only the differences compared to Variant LMK > Key Block LMK migration are described.

Field	Length & Type	Notes
Message Header	m A	(As for Variant LMK ⇨ Key Block LMK)
Command Code	2 A	Must have the value 'BW'.
Key Type code	2 H	Must be set to 'FF'.
Key length flag	1 H	Must be set to 'F'.
Key	1 A + n A	The operational key to be translated, encrypted under the old Key Block LMK.
Delimiter	1 A	Must have value ';'.
Key Type	3 H	Must be set to 'FFF'.
Delimiter	1 A	(As for Variant LMK ⇨ Key Block LMK)
Reserved	1 A	(As for Variant LMK ⇨ Key Block LMK)
Key Scheme (LMK)	1 A	(As for Variant LMK ⇨ Key Block LMK)
Reserved	1 A	(As for Variant LMK ⇨ Key Block LMK)
Delimiter	1 A	(As for Variant LMK ⇨ Key Block LMK)
LMK Identifier	2 N	(As for Variant LMK ⇨ Key Block LMK)
Delimiter	1 A	(As for Variant LMK ⇨ Key Block LMK)
Key Usage	2 A	(As for Variant LMK ⇨ Key Block LMK)
Mode of Use	1 A	(As for Variant LMK ⇨ Key Block LMK)

Field	Length & Type	Notes
Key Version Number	2 N	(As for Variant LMK ⇨ Key Block LMK)
Exportability	1A	(As for Variant LMK ⇨ Key Block LMK)
Number of Optional Blocks	2 N	(As for Variant LMK ⇨ Key Block LMK)
Optional Block Identifier	2 A	(As for Variant LMK ⇨ Key Block LMK)
Optional Block Length	2H	(As for Variant LMK ⇨ Key Block LMK)
Optional Data Block	N A	(As for Variant LMK ⇨ Key Block LMK)
End Message Delimiter	1 C	(As for Variant LMK ⇨ Key Block LMK)
Message Trailer	n A	(As for Variant LMK ⇨ Key Block LMK)

10.11.2 The BX Response to the Host

In response to the BW host command, the payShield 10K will return the following BX response to the host:

Field	Length & Type	Notes
Message Header	m A	(As for Variant LMK ⇨ Key Block LMK)
Response Code	2 A	Has the value 'BX'. (As for Variant LMK ⇨ Key Block LMK)
Error code	2 N	(As for Variant LMK ⇨ Key Block LMK)
Key	1 A + n A	(As for Variant LMK ⇨ Key Block LMK)
End Message Delimiter	1 C	(As for Variant LMK ⇨ Key Block LMK)
Message Trailer	n A	(As for Variant LMK ⇨ Key Block LMK)

10.12 Migrating Keys from Key Block to Variant LMKs

This migration is not permitted because Variant LMKs are not as strong as Key Block LMKs.

10.13 Migrating Keys for PCI HSM Compliance

When it is required to make a payShield 10K compliant with the requirements of the PCI PTS HSM security standard, it may be necessary to move some keys from Variant key type 002 (LMK pair 14-15, Variant 0) to other key types.

Although this can be done as a separate operation, it can be achieved at the same time as migrating between LMKs using the BW host command by entering 'F2' as the Key Type Code, and the desired destination key type in the Key Type field: see the *payShield 10K Core Host Commands reference manual* for further details.

10.14 Re-encrypting PINs

Where PINs have been stored encrypted under the old LMK (in LMK Live storage or LMK Key Change storage) these will need to be re-encrypted using the new LMK (in LMK Key Change storage or LMK Live storage). This can be done by using the BG host command.

A host application will be required that takes each PIN from the old PIN database, re-encrypts it using the BG host command, and stores the re-encrypted PIN into the new PIN database.

10.14.1 The BG Host Command

The structure of the BG host command is given in the following table:

Field	Length & Type	Notes
Message Header	m A	This field contains whatever the user wants. The length of the field is defined using the *CH* console command or *Configuration / Host Settings* in payShield Manager. It is subsequently returned unchanged in the response to the host.
Command Code	2 A	Has the value 'BG'.
Account Number	12 N	The 12 right-most digits of the account number, excluding the check digit.
PIN	L_1 N or L_1 H	The PIN encrypted under the old LMK, where L_1 is the old encrypted PIN length. L_1 N applies where PIN encryption algorithm A (Visa method) is specified in the security settings, and L_1 H applies where PIN encryption algorithm B (Racal method) is specified.
Delimiter	1 A	Value '%'. Optional; if present, the following field must be present.

Field	Length & Type	Notes
LMK Identifier	2 N	Where the user is using multiple LMKs on the same HSM, this allows the required LMK to be selected. Minimum value = '00'; maximum value is defined by license. This field must be present if the above Delimiter (%) is present. If the field is not present, then the default LMK will be used.
End Message Delimiter	1 C	Must be present if a message trailer is present. Value X'19'.
Message Trailer	n A	Optional. The contents of the trailer is as required by the user, and is returned unchanged in the response. Maximum length 32 characters.

10.14.2 The BH Response

The HSM will return the following BH response to the host's BG command:

Field	Length & Type	Notes
Message Header	m A	This is a play-back of the header provided in the *BG* command.
Response Code	2 A	Has the value 'BH'.
Error code	2 N	Indicating the general outcome of the *BG* command: '00' : No error '68' : Command disabled or any standard error code (see the payShield 10K Core Host Commands reference manual).
PIN	L_2 N Or L_2 H	The PIN encrypted under the new LMK, where L_2 is the new encrypted PIN length. L_2 N applies where PIN encryption algorithm A (Visa method) is specified in the security settings, and L_2 H applies where PIN encryption algorithm B (Racal method) is specified.
End Message Delimiter	1 C	Present only if a message trailer is present. Value X'19'.
Message Trailer	n A	This is simply a play-back of any trailer included in the BW command.

10.15 Re-encrypting Decimalization Tables

For security, it is recommended that decimalization tables are encrypted. They are encrypted under the LMK, and so will need to be re-encrypted when migrating to a new LMK.

This is achieved by having a host application which takes each decimalization table from the old decimalization table database and re-encrypting it under the new LMK using the LO (where "O" is the alphabetic "O for Oscar") host command (not to be confused with the LO console command discussed earlier!) and then storing it in a new decimalization table database. The new LMK can be in either Key Change storage or Live storage.

The structure of the LO host command is as follows:

Field	Length & Type	Notes
Message Header	m A	This field contains whatever the user wants. The length of the field is defined using the *CH* console command or *Configuration / Host Settings* in payShield Manager. It is subsequently returned unchanged in the response to the host.
Command Code	2 A	Most have the value 'LO'.
Decimalization Table (old LMK)	16 H	A decimalization table encrypted under the old LMK.
Delimiter	1 A	Value '%'. Optional; if present, the following field must be present.
LMK Identifier	2 N	Where the user is using multiple LMKs on the same HSM, this allows the required LMK to be selected. Minimum value = '00'; maximum value is defined by license. This field must be present if the above Delimiter (%) is present. If the field is not present, then the default LMK will be used.
End Message Delimiter	1 C	Must be present if a message trailer is present. Value X'19'.
Message Trailer	n A	Optional. The contents of the trailer is as required by the user, and is returned unchanged in the response. Maximum length 32 characters.

The payShield 10K will return the following LP response to the host:

Field	Length & Type	Notes
Message Header	m A	This is a play-back of the header provided in the *LO* command.
Response Code	2 A	Has the value 'LP'.
Error code	2 N	Indicating the general outcome of the *LO* command: '00' : No error '68' : Command disabled or any standard error code (see the payShield 10K Core Host Commands reference manual).
Decimalisation Table (new LMK)	16 H	The decimalisation table encrypted under the new LMK.
End Message Delimiter	1 C	Present only if a message trailer is present. Value X'19'.
Message Trailer	n A	This is simply a play-back of any trailer included in the BW command.

10.16 Switching to the New LMK

Following the activities described above, the system is now in the following state:

- Old databases of operational keys, PINs, and decimalization tables, encrypted under the old LMK, are still being used for production.
- New databases of operational keys, PINs, and decimalization tables, encrypted under the new LMK are pending but not yet being used for production.
- One or more HSMs may have been taken out of service in order to re-encrypt the operational keys, PINs, and decimalization tables.
    - These would be HSMs that have the old (current) LMK (which is still being used on other HSMs for production) loaded in Key Change Storage (e.g. by using the LO console command), and the new LMK (not yet in use for production work) in their Live storage.
    - In this case there are other HSMs with the old LMK in their Live storage, which are doing production work using keys, PINs, and decimalization tables in the old versions of the databases.
- Production host applications are still using the old databases of operational keys, PINs, and decimalization tables.

In order to start using the new LMK, the following changes must be synchronized:

- Host production applications start using the new databases of operational keys, PINs, and decimalization tables.
- If the re-encryption of keys was done on an HSM with the new LMK in Live storage, then this HSM is immediately ready to start processing transactions using the new LMK. However, the new LMK needs to be loaded into LMK Live storage on those HSMs that were processing transactions using the old LMK.
- On the other hand, if the re-encryption of keys was done on an HSM with the new LMK in Key Change storage, then the new LMK needs to be loaded into LMK Live storage on all the HSMs in the system.

A total interruption of service can be avoided by a gradual switchover from the old LMK to the new - but in this case the host applications must know whether an HSM is using the old or new LMK and must retrieve the key or data from the appropriate database.

The use of the Multiple LMK feature of the payShield 10K offers additional options, and is described in the following section.

10.17 Taking Advantage of Multiple LMKs

The payShield 10K supports multiple concurrent LMKs. The base product allows the user to implement one Variant-type LMK and one Key Block-type LMK, and optional licenses are available to provide up to 20 LMKs in any combination of types.

The multiple LMK feature offers a number of valuable benefits, and provides additional flexibility to simplify the process.

Here is an example of how the multiple LMK feature can be used where the old (still Live) LMK is in LMK Key Change storage and the new (future) LMK is in LMK Live storage:

Let us take as a starting point a production system which has the live LMK at LMK 00.

- LMK 00 is set up as the default LMK. This means that it is the LMK that is used by default in host commands where no LMK is identified: this provides backwards compatibility to applications developed before the multiple LMK facility was introduced.
- The future, new LMK is loaded as LMK 01 in LMK Live storage.
- The existing, "old" LMK, which is LMK 00 and is being used for production, is also loaded into LMK Key Change Storage for LMK 01.
- The BW, BG, and LO host commands can now be used to re-encrypt operational keys, PINs, and decimalization tables from the old LMK (which is in Key Change Storage, and also still in LMK 00 and therefore available for production) to the new LMK, which is loaded as LMK 01. This is achieved by setting the LMK Identifier in the host commands to a value of "01" : this must be preceded by a delimiter of "%".
- When all of the operational keys, PINs, and decimalization tables have been re-encrypted under the new LMK, the host application can start using the new key database when one of the following actions have been taken:
    - The new LMK is re-loaded on the payShield 10K as LMK 00. Or
    - Host commands sent to the payShield 10K are amended to use LMK 01 by either:

Specifying the value "01" for the LMK identifier in host commands (see the structure of commands in the *payShield 10K Core Host Commands reference manual*). Or

Directing commands to the relevant TCP port (see the section on Multiple LMKs in Chapter 1 of the *payShield 10K Core Host Commands reference manual*).

The benefit of this approach is that there is no need to take one or more HSMs out of productive use while the LMK migration is being performed, and therefore the key migration using the BW host command can be spread over as many HSMs as desired.

Multiple LMKs could also be used to avoid a "big bang" switchover from old to new LMKs: with the old LMK in one Live storage slot and the new LMK in a second Live storage slot, individual elements of the system can be moved to the new LMK one at a time.

10.18 Tidying Up After Migration to a New LMK

10.18.1 Deleting the Old LMK from Key Change Storage

The LMK in Key Change Storage should be deleted once it is no longer needed. There are multiple ways of doing this.

10.18.1.1 Using the Console

The LMK can be deleted from Key Change Storage using the DO (where O is the alphabetic "O for Oscar") console command. The payShield 10K must be in Secure state.

10.18.1.2 Using payShield Manager

The LMK is deleted using the ⚙ button displayed against the LMK in payShield Manager's Operational / LMK Operations / Key Change Storage tab. This can only be done in Secure state. See the *payShield 10K Installation and User Guide* for more detailed information.

10.18.1.3 Using a Host Command

The BS host command allows the host to erase the LMK in Key Change Storage. The structure of the command is given in the following table:

Field	Length & Type	Notes
Message Header	m A	This field contains whatever the user wants. The length of the field is defined using the *CH* console command or *Configuration / Host Settings* in payShield Manager. It is subsequently returned unchanged in the response to the host.
Command Code	2 A	Must have the value 'BS'.
Delimiter	1 A	Value '%'. Optional; if present, the following field must be present.
LMK Identifier	2 N	Where the user is using multiple LMKs on the same HSM, this allows the host to select which Old LMK is to be deleted. Minimum value = '00'; maximum value is defined by license. This field must be present if the above Delimiter (%) is present. If the field is not present, then the default LMK will be used.
End Message Delimiter	1 C	Must be present if a message trailer is present. Value X'19'.
Message Trailer	n A	Optional. The contents of the trailer is as required by the user, and is returned unchanged in the response. Maximum length 32 characters.

The BT response has the following structure:

Field	Length & Type	Notes
Message Header	m A	This is a play-back of the header provided in the *BS* command.
Response Code	2 A	Has the value 'BT'.
Error code	2 N	Indicating the general outcome of the *BS* command: '00' : No error '68' : Command disabled or any standard error code (see Chapter 4 of the *payShield 10K Core Host Commands reference manual*).

Field	Length & Type	Notes
End Message Delimiter	1 C	Present only if a message trailer is present. Value X'19'.
Message Trailer	n A	This is simply a play-back of any trailer included in the BW command.

10.18.2 Deleting the New LMK

In the section *Taking advantage of Multiple LMKs*, one suggested approach requires the new LMK to be unloaded from a temporary LMK Identifier or location (where it was loaded to enable the migration of keys and data to take place) and loaded to the location where it is required for production work.

The section *Loading the new LMK* explains how to load the LMK to its desired location, but in addition, the LMK should be deleted from its temporary location. This can be done by using one of the following methods:

10.18.2.1 Using the Console

LMK deletion is achieved using the DM console command. This command requires Secure state and authorization - in a multiple authorize state environment, the activity to be authorized is admin.console.

Note that the DM console command also deletes the relevant old key in Key Change Storage, avoiding the need to do this separately.

10.18.2.2 Using payShield Manager

The LMK is deleted using the ⚙ button displayed against the LMK in payShield Manager's Operational / LMK Operations / Local Master Keys tab. This can only be done in Secure state. See the *payShield 10K Installation and User Guide* for more detailed information.

11 X9.143/TR-31 Key Blocks

In the payments environment, keys often have to be shared between two or more parties, and these organizations will be using different Local Master Keys and/or different types of HSM. There is therefore a need to securely export the operational keys from one system and import them into another using a common standard.

The secure common standard is based on the use of “Key Blocks” which are cryptograms that not only contain the key being exported encrypted using a shared key, but also the properties of the key include the operations the key can be used for.

The specification that has been used for a number of years for the format of key blocks is ASC X9 TR31 from ANSI entitled “Interoperable Secure Key Exchange Key Block Specification”. The latest version of this specification is 2018.

This has now been replaced by both an ANSI and an ISO standard – ANSI X9.143 (Retail Financial Services Interoperable Secure Key Block Specification) and ISO 20038 (Banking and related financial services — Key wrap using AES). The ANSI standard is backward compatible with TR-31. The ISO standard only supports AES but is otherwise identical to the ANSI standard.

payShield 10K has supported key blocks as given in the TR-31 specification for a number of years and so also supports the ANSI X9.143 standard. payShield 10K also supports the ISO 20038 standard when AES keys are used. Please note that as with most standards, a number of optional items are included – the optional items supported by payShield 10K are described in the sections below.

Please note that payShield 10K also supports other formats for exchanging keys (e.g. ANSI X9.17) and these are described in Appendix A.

This chapter describes X9.143/TR-31 Key Blocks and their implementation in payShield 10K.

11.1 X9.143/TR-31 Key Block Structure

Key Scheme Tag ("R") (1 byte)	Key Block Header (16 ASCII characters)	Optional Header (ASCII characters, variable length)	Encrypted Key Data (variable length, ASCII encoded)	Authenticator (8 or 16 ASCII characters)

🟦	a 16-byte (clear) Key Block Header, which defines the key usage (e.g. Visa PVV) and mode of use (e.g. verification only), the algorithm with which the key is used (e.g. TDES) and the limitations on the exportability of the key (e.g. no export permitted);
🟨	a (clear) Optional Header block, which can be used (for example) to define the key set identifier;
🟥	the Encrypted Key Data, which includes the actual key itself, encrypted under the "encryption variant" of the wrapping key;

	a Key Block Authenticator, calculated using the "authentication variant" of the wrapping key; the use of the Authenticator prevents unauthorized modification of the Key Block.

X9.143/TR-31 supports the transfer of DES, TDES, AES and HMAC keys. The secure transfer of RSA and ECC keys and certificates are also included in the standard but are not currently supported by payShield 10K. The wrapping key must be a TDES or an AES key. Note that in order to comply with PCI-HSM requirements, the HSM will not permit a lower strength key to encrypt a higher strength key, using the NIST SP800-57 recommended definitions of relative key strength.

11.2 Key Block Header

The Key Block Header governs how the key contained in the Key Block may be used. Only specific field values for the Header are permitted for HSM commands. If a Key Block with the "wrong" Header is submitted in a command message, then the HSM will return an error code. The structure of the Header is as follows:

Byte(s)	Field	Comments
0	Version ID	version of X9.143/TR-31 key block format
1-4	Key Block Length	total length of Key Block (decimal)
5-6	Key Usage	e.g. key encryption, data encryption
7	Algorithm	e.g. DES, TDES, AES
8	Mode of Use	e.g. encrypt only
9-10	Key Version Number	e.g. version of key in the Key Block or used to indicate that the key is a key component
11	Exportability	e.g. exportable under a trusted key
12-13	Number of optional blocks	number of Optional Header blocks (Decimal)
14-15	Reserved for future use	value = "00"

The sections that follow define valid values for the fields described in the Key Block Header.

Note: there are constant additions to the range of available values: the latest sets of values can be found in the X9.143 standard.

11.2.1 Version ID (Byte 0)

Byte 0 of the Header contains a single character that indicates the format of the X9.143/TR-31 key block:

Value	Definition
'A'	Originally defined in TR-31:2005 Key block protected using the Key Variant Binding Method Note: This version is now deprecated and should not be used for new applications. Refer to X9.143:2022 for details.

Value	Definition
'B'	Originally defined in TR-31:2010 Key block protected using the Key Derivation Binding Method Note: This version is the preferred version for all new TDES applications. Refer to X9.143:2022 for details.
'C'	Originally defined in TR-31:2010 Key block protected using the Key Variant Binding Method The MACing and encryption keys are derived from the Key Block Protection Key using the same XOR process as used with Key Block Version ID = 'A'. Refer to X9.143:2022 for details.
'D'	Originally defined in TR-31:2018 Key block protected using the AES Key Derivation Binding Method.

11.2.2 Key Block Length (Bytes 1-4)

Bytes 1-4 of the Header contain the length of the entire key block, namely Header, Optional Header Blocks, encrypted Key Data and the Authenticator. The length of the key block is calculated after encoding and is represented as 4 numeric (ASCII) digits. For example, if the total key block length is 112 characters (bytes) then the value in byte 1 will be "0", the value in bytes 2 and 3 will be "1" and the value in byte 4 will be "2" (i.e. X'30313132).

11.2.3 Key Usage (Bytes 5-6)

Bytes 5-6 of the Header define the primary usage of the key contained in the key block. The following table defines the usage code, and how Thales key block usage codes are converted to X9.143/TR-31 codes.

Value	Definition	Corresponding Thales KB Key Usage
B0	Base Derivation Key (BDK)	B0, 41, 42, 43
B1	DUKPT Initial Key (IKEY$^{\phi}$)	B1
C0	Card Verification Key	C0, 11, 12, 13
D0	Data Encryption Key (Generic)	D0, 21, 22, 23
E0	EMV/Chip card Master Key: Application Cryptogram (MKAC)	E0
E1	EMV/Chip card Master Key: Secure Messaging for Confidentiality (MKSMC)	E1
E2	EMV/Chip card Master Key: Secure Messaging for Integrity (MKSMI)	E2
E3	EMV/Chip card Master Key: Data Authentication Code (MKDAC)	E3
E4	EMV/Chip card Master Key: Dynamic Numbers (MKDN)	E4
E5	EMV/Chip card Master Key: Card Personalisation	E5
E6	EMV/chip card Master Key: Other	E6, 31, 32
I0	Initialization Value	I0
K0	Key Encryption / Wrapping Key (Generic)	K0, 51, 52

Value	Definition	Corresponding Thales KB Key Usage
K1	Key Block Protection Key	K1
M0	ISO 16609 MAC algorithm 1 (using 3-DES)	M0
M1	ISO 9797-1 MAC algorithm 1	M1
M2	ISO 9797-1 MAC algorithm 2	M2
M3	ISO 9797-1 MAC algorithm 3	M3
M5	ISO 9797-1:1999 MAC algorithm 5	M5
M6	ISO 9797-1:2011 MAC algorithm 5/CMAC	M6
M7	HMAC Key (Hash Algorithm specified in 'HM' optional block)	61, 62, 63, 64, 65
P0	PIN Encryption Key (Generic)	P0, 71, 72, 73
V0	PIN Verification Key (Generic)	V0
V1	PIN Verification Key (IBM 3624 algorithm)	V1
V2	PIN Verification Key (Visa PVV algorithm)	V2
11	ZKA Master Key (AES only)	53
All numeric values except 11	Reserved for Proprietary Use	

Note 1: For additional code conversion information, refer to Section 11.4.2, Optional Blocks.

Note 2: The AES ZKA (key usage 53) is supported in TR-31 as key usage '11' as defined in the GBIC standards. When importing, the key usage will be converted from key usage '11' in ANSI X9.143/TR-31 to key usage '53' for Thales Key Block. When exporting, the key usage will be converted from '53' in Thales Key Block to '11' in TR-31. The 3DES ZKA is not supported in ANSI X9.143/TR-31.

11.2.4 Algorithm (Byte 7)

Byte 7 of the Header defines the cryptographic algorithm with which the key contained in the key block will be used. The following values are defined, although only two of these values are currently supported:

Value	Hex	Algorithm
A	X'41	AES
D	X'44	DES (included for backwards compatibility)
E	X'45	Elliptic Curve (included for future reference)
H	X'48	HMAC
R	X'52	RSA (included for future reference)
S	X'53	DSA (included for future reference)
T	X'54	Triple DES (TDES)
Numeric Values		Reserved for Proprietary Use

11.2.5 Mode of Use (Byte 8)

Byte 8 of the Header defines the operation that the key contained in the key block can perform.

Value	Hex	Definition
B	X'42	Both Encrypt and Decrypt
C	X'43	MAC Calculate (Generate or Verify)
D	X'44	Decrypt Only
E	X'45	Encrypt Only
G	X'47	MAC Generate Only
N	X'4E	No special restrictions or not applicable
S	X'53	Signature Generation Only
T	X’55	Both Sign and Decrypt
V	X'56	MAC Verify Only
X	X'58	Key Derivation
Y	X’59	Key used to create key variants
Numeric values		Reserved for Proprietary Use

11.2.6 Key Version Number (Bytes 9-10)

Bytes 9-10 of the Header define the version number of the key contained in the key block or that the key is actually a key component. The following values will be supported:

First Character	Second Character	Key value field meaning
0	0	Key versioning is not used for this key.
c	Any character	The value carried in this Key Block is a component of a key. Local rules dictate the proper use of a component.
Any other combination of characters		The key version field indicates the version of the key carried in the Key Block.

11.2.7 Exportability (Byte 11)

Byte 11 of the X9.143/TR-31 Key Block Header specifies the "exportability" of the key contained in the Key Block. This defines the conditions under which the key contained in the Key Block can be exported outside the cryptographic domain in which the key is found. A key is defined to be trusted if it is in either Thales Key Block or X9.143/TR-31 Key Block format. Any other format key is said to be untrusted. The following values are supported:

Value	Exportability
E	May only be exported in a trusted Key Block, provided the wrapping key itself is trusted
N	No export permitted
S	Sensitive; all other export possibilities are permitted, provided such export has been enabled via the payShield 10K security settings

Value	Exportability
Numeric Values	Reserved for Proprietary Use

"Sensitive" export includes, for example, the case when a key is exported in ANSI X9.17 format. This type of export is disabled by default in the payShield 10K security settings and must be specifically enabled (e.g. using the CS console command) if required.

When a key is exported in a X9.143/TR-31 Key Block format, byte 11 in the exported Key Block dictates how further export must be handled by the recipient of the Key Block. Hence, if the received byte 11 has value "E" then further "trusted" export is permitted, but if byte 11 has value "N" then no further export will be permitted.

Such considerations must be taken into account when the key is initially generated. It is possible to change the value of byte 11 from "E" or "S" to "N" (so that no further export will be permitted), via an optional field at the end of an HSM key export command message.

11.2.8 Number of Optional Blocks (Bytes 12-13)

The X9.143/TR-31 Key Block format allows a key block to contain up to 99 Optional Header Blocks which can be used to include additional (optional) data within the Key Block. Optional Header Blocks are described below.

Bytes 12-13 of the Header specify the number of Optional Header Blocks in the key block. A value of "00" (X'3030) indicates there are no optional blocks. A value of "12" (X'3132) indicates that there are 12 optional blocks.

11.2.9 Example X9.143/TR-31 Key Block Header

As an example, a X9.143/TR-31 Key Block Header of

A0072**V2TG**22**N**0000

indicates:

- a Key Block of 72 bytes,
- that the key contained in the Key Block may only be used with the Visa PVV algorithm (the "V2" field),
- that it is a TDES key (the "T" field),
- that it may only be used for PVV generation ("G"),
- the key version number is "22",
- the key may not be exported (the "N" field), and
- there are no Optional Header blocks in the Key Block.

11.3 Optional Header

11.3.1 Standard Optional Header

The Optional Header comprises a number of Optional Header blocks, each with the structure:

Byte(s)	Field	Comments
0-1	Identifier	Optional Header block identifier
2-3	Length	Optional Header block length (hexadecimal)
4-n	Data	Optional Header block data

The following types of optional block are defined in X9.143/TR-31 and are supported in payShield 10K. Note that only KS, KV and PB are supported in payShield 10K base release v1.6a and earlier:

Optional Header Block ID	Usage
HM	The Hash Algorithm used when exporting an HMAC Key. This Optional Block ID is only supported in Host Commands 'LU' (Import an HMAC under a ZMK) and 'LW' - Export an HMAC under a ZMK
IK	Initial Key Identifier for the Initial DUKPT Key. Similar to the 'KS' block, the Initial Key ID is the concatenation of the BDK ID and the Derivation ID encoded in hex-ASCII. For AES DUKPT it is 16 hex-ASCII characters in length. This value is used to instantiate the use of the Initial DUKPT key on the receiving device and it identifies the Initial Key derived from a BDK. (See ANS X9.24 part 3 Annex B for examples.)
KC	Key Check Value of wrapped key; computed according to X9.24-1-2017 Annex A not used as an integrity mechanism.
KP	Key Check Value of Key Block Protection Key (KBPK); computed according to X9.24-1-2017Annex A. not used as an integrity mechanism.
KS	Key Set Identifier, encoded in hex-ASCII; optionally used to identify the key within a system. (See ANS X9.24 part 3 for examples.)
KV	Key Block Values: Informational field indicating the version of the key block field values.
PB	Padding Block; A variable-length padding field used as the last Optional Block. The padding block is used to bring the total length of all Optional Blocks in the key block to a multiple of the encryption block length. The data bytes in this block are filled with readable ASCII characters.
TS	Time Stamp; the time and date (in UTC Time format) that indicates when the key block was formed.
Numeric	Numeric Identifiers; These are reserved for proprietary use. If proprietary identifiers are used, it is the responsibility of the application owner to ensure that all necessary devices support the proprietary Optional Block identifiers that they will use.

Note that the following Option Block Header IDs also defined in X9.143/TR-31 2018 are not currently supported:

- CT; Asymmetric public key certificate

- HM; Hash algorithm for HMAC

The format of the data required for each Optional Block ID is shown below.

Optional Header Block ID	Description	Format of Optional Block Data	Additional Restrictions on Optional Block Data
HM	HMAC Hash Identifier	6 H	See note below
IK	Initial Key Identifier	15 H for TDEA or 16 H for AES	None
KC	Key Check Value of wrapped key	10 H	See note below
KP	Key Check Value of Key Block Protection Key (KBPK)	10 H	See note below
KS	Key Set Identifier	8 H	None
KV	Key Block Values	4 H	None
PB	Padding Block;	NA	
TS	Time Stamp	20 A	Date must be in the format: YYYY-MM-DDTHH:MM:SSZ
Numeric	Numeric Identifiers	n A	None

Notes:

When using Optional Block ID HM, the hash algorithm used with the HMAC key is specified as follows:

'10' – SHA-1
'20' – SHA-224
'21' – SHA-256
'22' – SHA-384
'23' – SHA-512
'24' – SHA-512/224
'25' – SHA-512/256
'30' – SHA3-224
'31' – SHA3-256
'32' – SHA3-384
'33' – SHA3-512

The following restrictions apply to the Optional Block Data entered for Optional Block IDs KC and KP when using Host Commands:

- For KC, bytes 4 and 5 of the Optional Block must 00 if the wrapped key is DES or TDES or 01 if the wrapped key is AES
- For KP, bytes 4 and 5 of the Optional Block must 00 if the wrapping key is DES or TDES or 01 if the wrapping key is AES
- Bytes 6 to 11 must contain dummy data which will be overwritten by the KCV when the final X9.143/TR-31 Key Block is formed

When using KC and KV with the Console Commands, there is no Optional Block Data to enter – this is generated automatically when the key block is formed.

11.3.2 Extended Numeric Optional Blocks

Host Command 'A8' also supports Extended Numeric Optional Blocks as defined in the X9.143/TR-31 2018 standard. Extended Numeric Optional Blocks allow a larger amount of data to be included in the X9.143/TR-31 Block and is indicated by inserting '00' in the standard Length field as shown below. Host Command 'A6' only allows import of a X9.143/TR-31 Key Block that includes Extended Numeric Optional Blocks, but the Extended Numeric Optional Block Data is for use by the application and is ignored by the host command. The fields required when using Extended Numeric Optional Blocks are shown below:

Byte(s)	Field	Comments
0-1	Identifier	Numeric Optional Header block identifier
2-3	Length	'00' to indicate the block includes following 2 fields to support 'Extended Numeric Optional Blocks.'
4-5	Number of Length bytes (Extended)	Only present if the Length field = '00'. Specifies the length (in bytes) of the following field. Value: '02'.
6-7	Optional Block Length (Extended)	Only present if the Optional Block Length field = '00'. Number of characters in the optional block (including the identifier and length); min permitted value X'0101; max permitted value X'2670.
8-n	Data	Optional Header block data

11.4 Using X9.143/TR-31 Key Blocks

X9.143/TR-31 Key Blocks are relevant to HSM commands that are used for key import or key export. Key Blocks themselves generally replace the existing X9.17 format encrypted keys in the HSM command or response messages.

In some cases, additional fields are required at the end of the command message to allow construction of the exported Key Block. Such fields are preceded by a special delimiter, the "&" character.

The key scheme identifier "R" is used to indicate to the HSM that the Key field is a X9.143/TR-31 Key Block.

Most of the HSM legacy commands will not be upgraded to support X9.143/TR-31 Key Blocks since there are more flexible versions of these commands available which will support X9.143/TR-31.

11.4.1 Key Scheme Tags

The table below summarizes the key scheme tags (or identifiers) relevant to key blocks supported in payShield 10K software:

Tag ID	Key Scheme
R	X9.143/TR-31 Key Block format
S	Thales Key Block format
V	VeriFone/GISKE Key Block format

11.4.2 Optional Blocks

This section provides information on how Optional Blocks in X9.143/TR-31 are handled.

When exporting to X9.143/TR-31 key block format using a Variant LMK, the Optional Blocks are included in the X9.143/TR-31 Key Block as follows:

- IK, KC, KP, KS, KV, TS and Numeric Optional Blocks are specified by the application in the host command or entered in the console command if they are required.
- PB Optional Header Block is used for padding and is added automatically as part of command processing.

When exporting to X9.143/TR-31 format using a Key Block LMK, the Optional Blocks are included in the X9.143/TR-31 Key Block as follows:

- KS and KV Optional Header Blocks are included in the X9.143/TR-31 key block automatically when exporting if they are included in the Thales Key Block LMK
- PB Optional Header Block is used for padding and is added automatically into the X9.143/TR-31 Key Block as part of command processing.
- If Numeric Optional Blocks are included in the Thales Key Block LMK, these are NOT included in X9.143/TR-31 Key Block as they are proprietary to Thales and for internal use only.
- KS and KV Optional Header Blocks can also be included in the X9.143/TR-31 key block if they are added using relevant export host or console commands. This is the case only if they are NOT included in Thales Key Block, otherwise a “Duplicate Header Block” error is given.
- IK, KC, KP, TS and Numeric Optional Blocks are included in the X9.143/TR-31 Key Block if they are added using the relevant export host or console commands.

When importing, all Optional Blocks included in the X9.143/TR-31 Key Block are ignored except KS and KV as described below.

It is important to note the following regarding the support for Option Blocks in Thales Key Blocks described in Chapter 8, Key Block LMK Key Scheme, and the support for Optional Blocks in X9.143/TR-31:

- Thales Key Blocks support Optional Header Block ID’s KS and KV. If these are included in Thales Key Block LMK, they are automatically included in the X9.143/TR-31 Key Block when exported. They cannot be added during export to X9.143/TR-31 if they are already included in Thales Key Block LMK.
- The Key Block LMK also support Numeric Optional Block IDs 00 to 05. These are used by Thales for proprietary purposes as described in Section 8.5.2.1. Therefore, these are not included in X9.143/TR-31 key blocks when the key is exported. Numeric Optional Block IDs 00 – 99 can be included in the X9.143/TR-31 Key Block when exporting using the relevant host and console commands and are ignored when importing a X9.143/TR-31 Key Block.
- Optional Header Block ID’s IK, KC, KP and TS are not supported by Thales Key Block LMK and so can only be included in the X9.143/TR-31 Key Block when exporting. These are ignored when importing.

11.5 GISKE Key Block

The GISKE Key Block is an early version of the X9.143/TR-31 Key Block and is supported by some VeriFone terminals. The HSM's host A8 command has been modified to support export of terminal keys in the GISKE format. The command would be the same as for a command structured for standard X9.143/TR-31 but with the "R" key scheme tag replaced with "V" to indicate that the exported key should be in GISKE Key Block format.

12 DES Key Support

12.1 DES Keys

The payShield 10K HSM host commands support single-, double and triple-length DES keys. The current command set is backward compatible with earlier versions. The commands support extensions to enable the specification of key length and key encryption scheme to use.

12.2 Key Usage

If the first character of the key is a hexadecimal character (0 – 9 or A - F) or "K" the commands will operate as previously specified. In most circumstances the key is single-length except for ZMKs when the ZMK length is configured for double-length or for specific keys that are double-length by definition. This is the 16H or 32H length and types referenced in the command structures in the *payShield 10K Core Host Commands reference manual*.

To support double and triple-length keys throughout the command set key scheme tags have been defined these enable an HSM to determine the key length and encryption mechanism used for a key. The key scheme tag prefixes the key. This is the 1A+32H or 1A+48H length and types in the command structures.

12.3 Key Encryption Schemes

The HSM supports numerous encryption schemes:

12.3.1 ANSI X9.17 Method

Each key of a double- or triple-length DES key is encrypted separately using the ECB mode of encryption. This scheme is available for import and export of keys and for keys encrypted under a Variant LMK. The use of this scheme must be enabled in the security settings (e.g. by using the CS (Configure Security) console command).

The tags for this scheme are:

- Z (optional) – Single-length DES keys
- X – Double-length DES keys
- Y – Triple-length DES keys

12.3.1.1 Variant Method

The payShield 10K HSM supports the Thales Key Block method, for the encryption and authentication of keys.

12.3.1.2 Thales Key Block Method

The payShield 10K HSM supports the Thales Key Block method, for the encryption and authentication of keys.

12.3.1.3 X9 ANSI X.9.143 Method

Information on the ANSI X.9.143 key encryption scheme is provided in Chapter ANSI X.9.143 Key Blocks.

12.3.1.4 GISKE Method

For full details of the Verifone/GISKE key encryption scheme, refer to the GISKE specification [Reference 7].

12.4 Key Generate, Import and Export

All the key management commands have extensions to enable the specification of the key scheme to use when encrypting a key. This also defines the key length to generate within key generation commands. For import and export of keys the key schemes must be consistent as far as length is concerned i.e. if a double-length key is input the key scheme flag defining the output must also be for a double-length key.

The extension consists of a delimiter ";" and three single character option fields. If the extension is used all fields must be provided. If the command does not use an option, "0" or any valid value can be entered in that field. The option will be ignored during processing.

The option fields are:

- Key scheme for encrypting the output key under ZMK.
- Key scheme for encrypting the output key under LMK.
- Key check value type.

The valid values for these options are:

Key under ZMK	Z/blank	Thales Variant Method – for single-length DES keys
	U	Thales Variant Method – for double-length DES keys
	T	Thales Variant Method – for triple-length DES keys
	X	ANSI X9.17 Method – for double-length DES keys
	Y	ANSI X9.17 Method – for triple-length DES keys
	R	X9 ANSI X.9.143 Key Block Method – for single/double/triple-length DES keys
	S	Thales Key Block Method – for all symmetric & asymmetric keys

Key under TMK	V	Verifone/GISKE Key Block Method – for double/triple-length DES keys
Key under LMK	Z/blank	Thales Variant Method – for single-length DES keys
	U	Thales Variant Method – for double-length DES keys
	T	Thales Variant Method – for triple-length DES keys
	X	ANSI X9.17 Method – for double-length DES keys
	Y	ANSI X9.17 Method – for triple-length DES keys
	S	Thales Key Block Method – for all symmetric & asymmetric keys
Key check value	0	KCV is 16-hex digits (backwards compatible mode) (except for DW & DY where an 8-hex KCV is returned)
	1	KCV is 6-hex digits
	2	KCV length is defined in the specific command

12.5 Rejection of Weak, Semi-Weak, & Possibly Weak Keys

All HSM commands that generate keys ensure that the standard DES weak, semi-weak, or possibly weak keys cannot be used. If the new key matches one of the listed weak, semi-weak, or possibly weak keys it is rejected and the key generation process is repeated.

12.5.1 DES Weak Keys

```
0101  0101  0101  0101
FEFE  FEFE  FEFE  FEFE

1F1F  1F1F  0E0E  0E0E
E0E0  E0E0  F1F1  F1F1
```

12.5.2 DES Semi-Weak Keys

```
01FE  01FE  01FE  01FE
FE01  FE01  FE01  FE01

1FE0  1FE0  0EF1  0EF1
E01F  E01F  F10E  F10E

01E0  01E0  01F1  01F1
E001  E001  F101  F101

1FFE  1FFE  0EFE  0EFE
FE1F  FE1F  FE0E  FE0E
```

011F	011F	010E	010E
1F01	1F01	0E01	0E01
E0FE	E0FE	F1FE	F1FE
FEE0	FEE0	FEF1	FEF1

12.5.3 DES Possibly Weak Keys

0101	1F1F	0101	0E0E
0101	E0E0	0101	F1F1
0101	FEFE	0101	FEFE
011F	1F01	010E	0E01
011F	E0FE	010E	F1FE
011F	FEE0	010E	FEF1
01E0	1FFE	01F1	0EFE
01E0	1FFE	01F1	F10E
01E0	E001	01F1	F101
01E0	FE1F	01F1	FE0E
01FE	1FE0	01FE	0EF1
01FE	E01F	01FE	F10E
01FE	FE01	01FE	FE01
1F01	011F	0E01	010E
1F01	E0FE	0E01	F1FE
1F01	FEE0	0E01	FEF1
1F1F	0101	0E0E	0101
1F1F	E0E0	0E0E	F1F1
1F1F	FEFE	0E0E	FEFE
1FE0	01FE	0EF1	01FE
1FE0	E01F	0EF1	F10E
1FE0	FE01	0EF1	FE01
1FFE	01E0	0EFE	01F1
1FFE	E001	0EFE	F001
1FFE	FE1F	0EFE	FE0E
E001	01E0	F101	01F1
E001	1FFE	F101	0EFE
E001	FE1F	F101	FE0E
E01F	01FE	F10E	01FE
E01F	1FE0	F10E	0EF1
E01F	FE01	F10E	FE01
E0E0	0101	F1F1	0101

E0E0	1F1F	F1F1	0E0E
E0E0	FEFE	F1F1	FEFE
E0FE	011F	F1FE	010E
E0FE	1F01	F1FE	0E01
E0FE	FEE0	F1FE	FEF1
FE01	01FE	FE01	01FE
FE01	1FE0	FE01	0EF1
FE1F	01E0	FE0E	01F1
FE1F	E001	FE0E	F101
FE1F	1FFE	FE0E	0EFE
FEE0	011F	FEF1	010E
FEE0	1F01	FEF1	0E01
FEE0	E0FE	FEF1	F1FE
FEFE	0101	FEFE	0101
FEFE	1F1F	FEFE	0E0E
FEFE	E0E0	FEFE	F1F1

13 User Storage

13.1 Introduction

The most common way of storing and managing keys and other sensitive data is on the host system, outside of the payShield 10K. In this scenario, the data is protected by being encrypted using the HSM's Local Master Key (LMK), the volume of stored data and its retrieval mechanism is limited only by the host system's capacity and functional capabilities (such as database systems), and the keys do not need to be replicated to multiple locations. The data can be backed up and replicated as desired - e.g., to create Disaster Recovery data centers.

In addition, keys and other data can also be stored in the User Storage area inside the payShield 10K itself. This method does not provide the capacity or flexibility that is experienced when storing at the host, but ensures that the data is protected by the HSM's security.

13.2 User Storage Area

The User Storage can be used to store any data, not just LMK-encrypted keys. This can be any data that the user wants to protect, and the following data types that are used by the payShield 10K:

- PIN Solicitation data
- PIN Blocks
- The Diebold table
- Decimalization tables

The user storage area has a capacity of 98,304 Bytes. It can hold up to 4,096 blocks of data:

- Each block is identified by an index, with values ranging from “000” to “FFF” in hexadecimal characters (equivalent to 0 to 4,095 in decimal).
- Blocks have either:

  A fixed size (of 16, 32, or 48 hexadecimal characters - corresponding to the size of single-, double-, and triple-length TDES keys),

  or

  A variable size depending on the block size setting.
- Blocks can contain any information that the user desires

The user storage area is in tamper-protected volatile memory, and so the stored data will be lost on events such as a tamper, use of the RESET console command.

There are two different modes of User Storage:

- Fixed Block Size User Storage
- Variable Block Size User Storage

For each mode, the following facilities for storing data are supported:

- General facility to store any data
- Specific method for storing and referencing RSA and ECC keys

Using the general facility to store and retrieve any data:

- Data is stored in User Storage using Host Command 'LA' (Load Data to User Storage).
- Data can be retrieved using Host Command 'LE' (Read from User Storage).
- Alternatively, keys in the appropriate format in User Storage can be referenced using a "Key Reference" (e.g. SK123). This is passed into host commands instead of the key. Restrictions on this facility are provided later below.

Using the specific Key Index facility for RSA and ECC keys:

- Host Command 'EK' (Load Private Key) is used to load a private RSA or ECC key into the Key Index storage
- For use with RSA keys, a Key Index can be used with Host Commands EW, GI, GK
- For use with ECC keys, currently a Key Index can be used with Host Command EW only.
- Key Index storage for up to 21 RSA or ECC private keys is supported with both Fixed and Variable Block Size User Storage

Please note, the following restrictions apply when using Fixed Block Size User Storage:

- Key References are not supported for RSA and ECC keys for any LMK type.
- Key References are not supported for any key type when using a Key Block LMK.

13.3 Defining Block Size

The block size is set using the CS console command or using payShield Manager under Configuration / Security Settings / General.

Using the CS console command for an example, the relevant prompt is:

```
User storage key length [S/D/T/V](SINGLE):
```

Valid entries are S, D, T, or V (or, in fact, any string beginning with these characters) representing the following block sizes:

S = Fixed Block Size for Single-length keys, i.e., block size of 16 hexadecimal characters (8 Bytes)

D = Fixed Block Size for Double-length keys, i.e., block size of 32 hexadecimal characters (16 Bytes)

T = Fixed Block Size for Triple-length keys, i.e., block size of 48 hexadecimal characters (24 Bytes)

V = Variable Block Size

The Variable block size capability provides more flexibility than the S/D/T options, and operates differently. In the sections that follow, the operation of User Storage when the block size is selected to be Single/Double/Triple-length is described first. The operation of User Storage when Variable Block Size is selected is described later.

13.4 User Storage when using Fixed Block Sizes

13.4.1 Loading Data into User Storage with Fixed Block Sizes

Data is loaded into User Storage using the LA host command. This command includes the following fields:

- Index Address (3H - Valid range “000” to “FFF”) - the index pointing to the location where the first data block is to be written.
- Block Count (2H - Valid range “00” to “20”) - the number of data blocks which are included in the command. Up to 32 data blocks can be included with the command. The sum of [(Index Address) + (Block Count-1)] must not exceed FFF (4,095 decimal).
- Data Blocks - a number of blocks of data to be stored. Data must be entered as ASCII-encoded hexadecimal characters (e.g. the byte with a binary value 00111101, hexadecimal 3D, would be entered as the two characters “3” and “D” and stored as 2 bytes.) The number of blocks must correspond to the value of the Block Count field. The size of the data block must correspond to the length set using CS or payShield Manager Configuration / Security Settings / General; if the data has a shorter length than this, the block should be padded with hexadecimal F characters.

Keys shorter than the specified block size can be stored by padding the key to the specified block size with hexadecimal F characters.

13.4.2 Reading Data from User Storage with Fixed Block Sizes

Data can be read from user storage using the LE host command. The command specifies:

- Index Address (3H) - the index pointing to the location where the first data block is to be read from. (Valid range is “000” to “FFF”.)
- Block Count (2H) - the number of data blocks which are to be retrieved. Up to 32 data blocks can be requested. (Valid range is “00” to “20”.) The sum of [(Index Address) + (Block Count-1)] must not exceed FFF (4,095 decimal).

13.4.2.1 Storing and Accessing Symmetric Keys

The host system will receive Variant LMK-encrypted TDES keys from the payShield 10K when, for example, the HSM is used to generate keys (A0 host command) or import a key (A6 host command). Typically the LMK-encrypted keys are stored in the host system's key database, but optionally the host system, can store the LMK-encrypted key inside the payShield 10K user storage, by using the method outlined above. The stored key in this case does not include the alphabetic key scheme indicator (e.g., “U” for double-length Triple DES, “T” for Triple-length Triple DES).

There are two ways that LMK-encrypted keys can be retrieved from user storage by the host system. First, the data can be read by the host system using the LE host command, as outlined above, and then inserting the returned key into a host command which is then sent to the payShield 10K.

Alternatively, host commands can point to an LMK-encrypted key held in user storage, avoiding the need to first retrieve the encrypted key from the payShield 10K. In order to do this, the field in the host command which would normally provide the LMK-encrypted key would instead include a key reference consisting of:

- The key scheme - e.g., [null] for single-length DES, “U” for double-length Triple DES, “T” for Triple-length Triple DES
- “K” to indicate that what follows is an index to user storage
- The 3-hex character index to the LMK-encrypted key in user storage

For example:

LMK-encrypted key	Type	Index in User Storage	Key in host command if key provided by host	Key reference in host command if key stored in user storage
9A73BC83A90012BD	SingleDES	3A8	9A73BC83A90012BD	K3A8
23A7BB616A198EFF1298FAE25D3A4BF5	Double TDES	B17	U23A7BB616A198EFF1298FAE25D3A4BF5	UKB17
18A7DE43129C2CD6BBA965823F5A78A99987AA4F5890B0C3	Triple TDES	179	T18A7DE43129C2CD6BBA965823F5A78A99987AA4F5890B0C3	TK179

Note that it is possible to store and recover keys of a length inconsistent with the Block Size. This is useful, for example, if it is necessary to store keys of different lengths.

- If the Block Size is set to Single (i.e. 16 hex characters), user storage can still be used to store double and triple-length keys. In this case, 2 blocks must be allocated for a double-length key, and three blocks for a triple-length key. When inserting a key reference in a command (e.g. "UKB17"), the appropriate number of blocks will be retrieved. Working in this way reduces the number of keys that can be stored because more than one index address is required for a key - for example, if only double-length keys were being used, then only 2,048 keys could be stored.
    - As an example, if the double-length encrypted key 92385025801691228682054947200919 was to be stored at index 0FE, then the example of the LA host command entitled "LA Command - Load Data to User Storage (double-length TDES key encrypted under a Variant LMK)" could be used.

If the Block Size is set to Triple (i.e. 48 hex characters), for example, single- or double-length keys can still be stored. However, the shorter keys must be padded with hexadecimal F characters to bring them up to the Triple Block size.

13.4.3 Key Block LMKs

When using Key Block LMKs, the LMK-encrypted key can be written to user storage using the LA host command. The host system can later read the key back from user storage using the LE host command and insert the it (with the preceding key scheme indicator of "S") into another host command sent to the payShield 10K.

Note that the key block must be expanded to hex encoding. For example, the following key block:

```
0007271TN00S0002B7CC796C2A4909FF12174898C4286E5AE2E8230E89284ED715D1D
7F0
```

should be written to user storage as:

```
3030303732373154 4E3030533030303242374343373936433241343930394646313232
```

plus any required "F" padding.

If the User Storage Block Size has been defined as Double (32 hexadecimal characters) then this key would be written as the following 5 blocks:

```
30303037323731544E30305330303032
42374343373936433241343930394646
31323137343839384334323836453541
45324538323330453839323834454437
3135443144374630FFFFFFFFFFFFFFFF
```

The facility to use key references in commands (e.g., “SK123”) is not available when using Key Block LMKs.

13.4.4 Storing General Data

As has been mentioned, any hexadecimal character string can be written to User Storage and subsequently retrieved.

For example, let's say we wanted to store the following information into User Storage:

Mary had a little lamb,

Its fleece was white as snow.

This would have to be converted to a hexadecimal string:

```
4D617279206861642061206C6974746C65206C616D622C0D0A49747320666C656563652077617320776869746520617320736E6F772E
```

If the User Storage Block Size has been set to Double (i.e., 32 Hexadecimal characters) this input would have to be written to User Storage as four 32-character blocks, padded with Hexadecimal F:

```
4D617279206861642061206C6974746C
65206C616D622C0D0A49747320666C65
65636520776173207768697465206173
20736E6F772EFFFFFFFFFFFFFFFFFFFF
```

13.4.5 PIN Solicitation Data

When a card issuer has solicited a desired PIN from the cardholder, the cleartext PIN provided by the cardholder must be encrypted and passed, together with the PAN, to the card issuer system. In order to ensure that the cleartext PAN and cleartext PIN do not exist together on the host system, a reference number is substituted for the PAN. A process is required to convert the reference number + cleartext PIN into a PAN + encrypted PIN. To do this, the reference number and PIN are loaded into user storage using the QA and QC commands.

Up to 1,260 records can be loaded using the QC command, and multiple preceding QA commands (each containing up to 25 records) can be used to load a total of 2,520 records. The user cannot define where this data is to be written in the user storage area.

The PIN solicitation data overwrites existing data in user storage. Therefore any user storage data must be reloaded after PIN Solicitation has been performed.

The PIN solicitation process is described in Section 3, PIN Printing and Solicitation.

13.4.6 PIN Blocks

If the user storage area has been used to store PIN Blocks, host commands requiring input of a PIN Block can directly access the PIN Block held in user storage. This is achieved by replacing the input PIN Block in the command with the ["K" + index] structure described earlier for referencing keys held in user storage.

13.4.7 Diebold Table

If a Diebold Table is loaded in the payShield 10K, it is loaded into user storage using the R console command. The table consists of 32 contiguous blocks of 16 hexadecimal characters each. Each LMK has its own Diebold table.

When using the R console command, the user must enter the Index Address for the start of the table. This can have any value from "000" to "FE0" (to ensure that the 32nd block is within the index limit of FFF).

The Diebold Table shares the same storage blocks as other data in user storage, and so the user must ensure that the 32 blocks allocated to the Diebold table are not being used to store any other data.

The Diebold Table stored in the HSM is encrypted using the LMK. For Variant LMKs, this will use LMK pair 14-15 variant 0 or LMK pair 36-37 variant 6, depending on the security settings relating to PCI HSM compliance.

The Diebold Table in user storage can be verified using the LC host command.

13.4.8 Decimalization Tables

Plaintext and Variant LMK-encrypted decimalization tables can be loaded into storage as blocks of 16 hexadecimal characters in the same way as TDES keys or other data.

Where the decimalization table is required in host commands (such as BK, DE, EE, GO), it can be referenced in the command using the same ["K" + index] structure described earlier for symmetric keys.

13.4.9 Storing a Mix of Data Types

As an example, let's say we have set User Storage Block Size to Double (i.e., 32 Hexadecimal characters) and we want to store:

- The key block example in the section on User Storage and Key Block LMKs, at index decimal 123 (hexadecimal 07B)
- The single-length variant-LMK encoded key 9A73BC83A90012BD, at the next available index - which would be 128 (hexadecimal 080)
- The example data in the section on Storing general data, at the next available index - which would be 129 (hexadecimal 081)

This could be done using a single LA host command, where:

- Index address = "07B"
- Block Count = "0A" (i.e., decimal 10)
- The following 10 data blocks are provided:

```
    30303037323731544E30305330303032  ┐
    42374343373936433241343930394646  │
    31323137343839384334323836453541  ├  Key Block (5 blocks)
```

```
45324538323330453839323834454437
3135443144374630FFFFFFFFFFFFFFFF
9A73BC83A90012BDFFFFFFFFFFFFFFFF  -  Single-length key (1 block)
4D617279206861642061206C6974746C
65206C616D622C0D0A49747320666C65
65636520776173207768697465206173     General data (4 blocks)
20736E6F772EFFFFFFFFFFFFFFFFFFFF
```

Alternatively, this could be done using 3 LA host commands, where:

- Command 1 – Key Block
- Index address = "07B"
- Block Count = "05"
- The following 5 data blocks are provided:

```
30303037323731544E30305330303032
42374343373936433241343930394646
31323137343839384334323836453541
45324538323330453839323834454437
3135443144374630FFFFFFFFFFFFFFFF
```

Command 2 – Single-length Key

- Index address = "080"
- Block Count = "01"
- The following 1 data block is provided:

```
9A73BC83A90012BDFFFFFFFFFFFFFFFF
```

- Command 3 – General Data
  - Index address = "081"
  - Block Count = "04"
  - The following 4 data blocks are provided:

```
4D617279206861642061206C6974746C
65206C616D622C0D0A49747320666C65
65636520776173207768697465206173
20736E6F772EFFFFFFFFFFFFFFFFFFFF
```

13.5 User Storage when using Variable Block Sizes

When V is selected for the block size, RSA and ECC keys and Key Block LMK-encrypted keys can be held in User Storage, and to be referenced by host commands.

Note: Stored keys and data do not have to be of a fixed length. This eliminates any need to specify a block size or to pad data blocks to the appropriate size.

The sections that follows assume that the reader is familiar with the information in the earlier section on the operation of User Storage when using fixed block sizes, and therefore identifies only the differences from that information.

13.5.1 Enabling the Enhanced Functionality

The option V(ARIABLE) must be selected for the User storage key length parameter in the payShield 10K Security Settings:

User storage key length [S/D/T/V](VARIABLE):

If the S(ingle), D(ouble), or T(riple) options are selected, user storage operates as described earlier in this chapter, and the enhanced functionality for variable-length user storage is not available.

If the V(ariable) option is selected, the enhanced functionality described here is activated. In this case, the user storage data structure is incompatible with that for the other options, and any existing data in user storage must be re-loaded.

The facility described earlier to store up to a total of 21 RSA or ECC private keys in the HSM continues to be available even if the V(ariable) option is selected.

13.5.2 New Data Structure

The enhanced user storage allows for 4,096 entries, as previously, but the total storage space has been increased by more than a factor of 5. These entries can now be of variable length, subject to the following limits:

- For Index Addresses 000-07F (0-127 decimal), the maximum length is 1,000 bytes. This enables the storage of longer data items such as RSA keys or the Diebold Table, but can be used to store any data such as AES or TDES keys.
- For Index Addresses 080-FFF (128-4095 decimal, i.e., 3,968 entries), the maximum length is 100 bytes. This is designed for use with shorter data items such as AES or TDES keys.

Data to be stored can now be defined as being ASCII, ASCII-encoded hexadecimal, or binary. It is no longer necessary to pad data to the block size.

The stored data can, as previously, be keys, PIN solicitation data, a Diebold table, decimalization tables, or any other data that the user desires.

13.5.3 Loading Data into User Storage

As previously, data is entered into user storage using the LA host command. However, the command is modified in the following way:

- A single data item can be loaded at a time, and as a result there is no need to specify the number of blocks being loaded
- The data being loaded can be ASCII, ASCII-encoded hexadecimal, or binary. The command defines which type-of data is to be loaded
- To allow for variable-length data, the command now includes a field to specify the data length

Where the loaded data represents keys, the keys can be encrypted using any type of LMK (i.e., including AES and TDES Key Block types).

Note that ECC keys can only be encrypted using an AES Key Block LMK.

13.5.4 Storing a Diebold Table

The Diebold Table must be stored as a single block of 512 hexadecimal characters in one of the longer user storage blocks with an index in the range 000-07F (000-127 in decimal).

13.5.5 Reading Data From User Storage

Data can still be read from user storage using the LE host command. As only a single record is returned, a block count is no longer required by the command.

13.5.6 Referencing Keys Stored in User Storage

Once a key is held in user storage it can be referenced by host commands which normally provide the key as part of the command. This now applies to any key type encrypted under any LMK type. The reference is constructed from the Index Flag “K” plus the Index Address of the key in user storage: the value of “K” for the Index Flag is used irrespective of the Index Flag used in the LA host command when the key was loaded into User Storage.

Here are some examples. <…> indicates binary data represented here as hexadecimal characters.

LMK-encrypted key	LMK Type	Key Type	Index in User Storage	Key in host command if key provided by host	Key reference in host command if key stored in user storage
9A73BC83A90012BD	Var TDES	Sngle DES	3A8	9A73BC83A90012BD	K3A8
23A7BB616A198EFF 1298FAE25D3A4BF5	Var TDES	Dble TDES	B17	U23A7BB616A198EFF 1298FAE25D3A4BF5	UKB17
18A7DE43129C2CD6 BBA965823F5A78A9 9987AA4F5890B0C3	Var TDES	Triple TDES	179	T18A7DE43129C2CD6 BBA965823F5A78A9 9987AA4F5890B0C3	TK179
00072B1TN00S0001 3FC2C2BD71EFA6A 1F65A01CC3EA930C ED51E07D8A818AC 06742D1651	Kbl TDES	Dble TDES	A12	S00072B1TN00S00013FC2 C2BD71EFA6A1F65A01CC 3EA930CED51E07D8A818A C06742D1651	SKA12
<AC5DC5A50CA3D9 293629994FF8452E7 67 *… (344 bytes of binary data) …* 402E0399AC47527F 7E881BA68F1> (344 bytes of binary data)	Var TDES	RSA Prv 1024	01C	<AC5DC5A50CA3D929 3629994FF8452E767 *… (344 bytes of binary data) …* 402E0399AC47527F7 E881BA68F1> (344 bytes of binary data)	K01C

LMK-encrypted key	LMK Type	Key Type	Index in User Storage	Key in host command if key provided by host	Key reference in host command if key stored in user storage
0037603RN00N0202 0005TPB0B4vzoOWb <F343F5C367CB557 BC217195EBFD4821 *… (336 bytes of binary data) …* 09E3AD2CFCE4B04 5CEC4D2C23061455 >99653B1A	Kbl TDES	RSA Prv 1024	06E	S0037603RN00N02020005T PB0B4vzoOWb <F343F5C367CB557 BC217195EBFD4821 *… (336 bytes of binary data) …* 09E3AD2CFCE4B045 CEC4D2C23061455> 99653B1A	K06E

13.6 Storing RSA and ECC Keys Using Key Indexes

The User Storage mechanism, when using fixed block size as described above, is not ideal for storing RSA keys in the HSM because of the length and variability on length of such keys. Therefore if fixed block sizes for user storage has been specified, the payShield 10K provides a separate mechanism for storing up to 21 LMK- encrypted RSA and ECC private keys inside the HSM and referencing them using an alternative method in the selected host commands.

Note: ECC keys can only be used with an AES Key Block LMK and in this case the facility to use key references in commands (e.g., "SK123") is not available when using Key Block LMKs.

13.6.1 Storing the RSA and ECC Private Key

The EK host command is used to store LMK-encrypted RSA and ECC private keys inside the payShield 10K. The RSA keys can be encrypted using Variant or Key Block LMKs. The ECC keys can only be encrypted using an AES Key Block LMK.

The command requires a Key index in the range 00-20 (decimal) to identify which location the key is to be loaded to.

13.6.2 Using RSA and ECC Private Keys Held in the payShield 10K

Once an RSA or an ECC private key has been stored inside the payShield 10K using the EK command, it can be directly referenced by relevant host commands which need to use it - EW, GI, GK for RSA and in the current release just EW for ECC.

These commands include a Private Key Flag field to identify the location (i.e., 00-20) where the required private key is stored. (Entering 99 into this field indicates that the private key is included in the command rather than being stored in the HSM.)

13.7 Host Command Summary

The following Host commands are typically used with User Storage:

- CA Command - Translate a PIN from TPK to ZPK Encryption (using keys and PIN Block held in user storage)
- DE Command - Generate an IBM PIN Offset
- EW Command - Generate a signature (with Variant LMK-encrypted RSA key held in user storage)
- EW Command - Generate a signature (with Key Block LMK-encrypted RSA key held in user storage)
- JE Command - Translate a PIN from ZPK to LMK encryption
- JG Command - Translate a PIN from LMK to ZPK encryption
- LA Command - Load Data to User Storage (double-length TDES key encrypted under a Variant LMK)
- LA Command - Load Data to User Storage (RSA private key encrypted under a TDES Variant LMK)
- LA Command - Load Data to User Storage (RSA private key encrypted under a TDES Key Block LMK)
- LE Command - Read Data from User Storage (single/double/triple block size setting)
- LE Command - Read Data from User Storage (variable block size setting)
- M0 Command - encrypt a block of data (using a Key Block LMK-encrypted key held in user storage)

14 PIN Block Formats

14.1 General

For PIN verification and PIN translation, the HSM requires that the PIN to be input as an encrypted 16-character PIN block. The plaintext data within the PIN block comprises the PIN and some form of padding to ensure that this data is 16 characters. The padding mechanism is known as the PIN block format. The HSM supports a number of PIN block formats, each identified by a 2-digit PIN block format code. Formats 34, 35, 41 and 42 are used for EMV PIN change operations and are only available to the KU and KY commands.

14.1.1 PCI HSM Compliance

In order to comply with the requirements of the PCI HSM Certification, there are restrictions on the PIN Block format translations and usage which are allowed. These are enforced by the payShield 10K security setting “Restrict PIN block usage for PCI compliance”.

14.2 PIN Block Translations and PCI

This section describes how certain security settings impact the HSM's PIN block translation capabilities.

The following table lists the standard PIN block translation commands:

Command	Description
CA	Translate a PIN from TPK to ZPK/BDK Encryption (3DES DUKPT)
CC	Translate a PIN from One ZPK to Another
G0	Translate a PIN from BDK to BDK or ZPK Encryption (3DES & AES DUKPT)

For information about other PIN block translation commands, please refer to the relevant manuals describing these commands.

14.2.1 Permitted PIN Block Translations – Not PCI Compliant

The following PIN Block translations are supported when the security setting *Restrict PIN block usage for PCI compliance* is set to **No**.

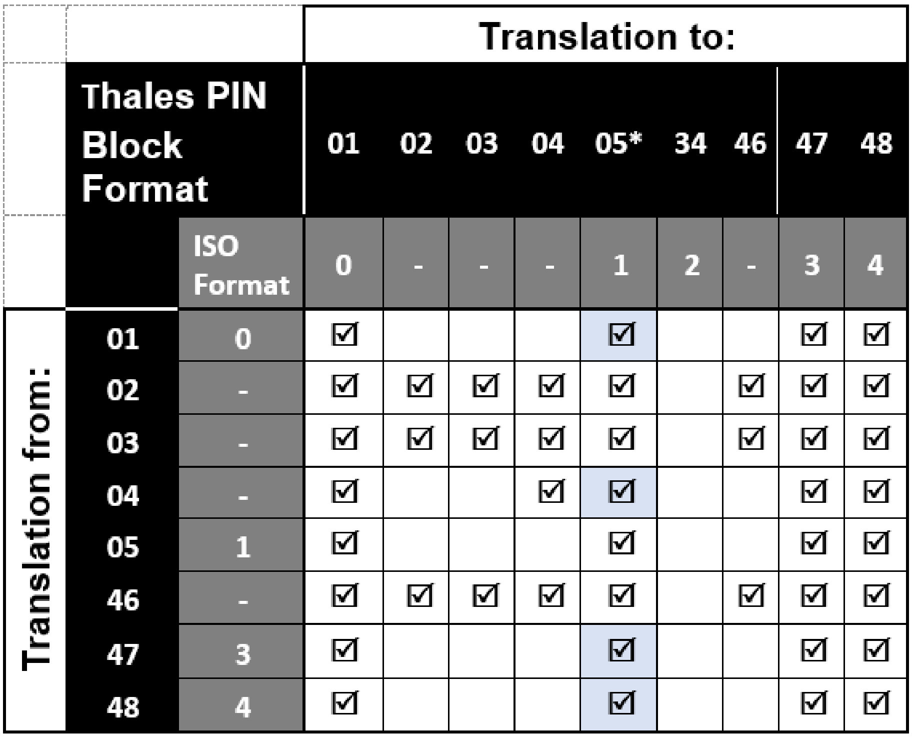
			Translation to:								
	Thales PIN Block Format		01	02	03	04	05	34*	46	47	48
		ISO Format	0	-	-	-	1	2	-	3	4
Translation from:	01	0	☑	☑	☑	☑	☑	☑	☑	☑	☑
	02	-	☑	☑	☑	☑	☑	☑	☑	☑	☑
	03	-	☑	☑	☑	☑	☑	☑	☑	☑	☑
	04	-	☑	☑	☑	☑	☑	☑	☑	☑	☑
	05	1	☑	☑	☑	☑	☑	☑	☑	☑	☑
	46	-	☑	☑	☑	☑	☑	☑	☑	☑	☑
	47	3	☑	☑	☑	☑	☑	☑	☑	☑	☑
	48	4	☑	☑	☑	☑	☑	☑	☑	☑	☑

*Translations to PIN Block Format 34 highlighted in light blue are only permitted if *Enable PIN Block Format 34 as output format for PIN translations to ZPK* is set to **YES**.

Note that Thales Format 48 is used to encrypt a PIN Block under an AES Key Block LMK.

14.2.2 Permitted PIN Block Format Translations – PCI Compliant

The following PIN Block translations are supported when the security setting *Restrict PIN block usage for PCI compliance* is set to **Yes**.



			Translation to:								
	Thales PIN Block Format		01	02	03	04	05*	34	46	47	48
		ISO Format	0	-	-	-	1	2	-	3	4
Translation from:	01	0	☑				☑			☑	☑
	02	-	☑	☑	☑	☑	☑		☑	☑	☑
	03	-	☑	☑	☑	☑	☑		☑	☑	☑
	04	-	☑			☑	☑			☑	☑
	05	1	☑				☑			☑	☑
	46	-	☑	☑	☑	☑	☑		☑	☑	☑
	47	3	☑				☑			☑	☑
	48	4	☑				☑			☑	☑

*Translations to PIN Block Format 05 highlighted in light blue are only permitted when using the GBIC derivation scheme with Host Command G0. This host command enforces the PCI requirement that the PIN Block is encrypted using a unique-key-per-transaction.

Note that Thales Format 48 is used to encrypt a PIN Block under an AES Key Block LMK.

14.3 Thales PIN Block Formats

The following table lists the PIN Block formats supported by the standard base software, together with their default state: each format is described below. Note that PIN Blocks implemented as part of a custom software development will always be enabled.

Thales Format	ISO Format	Description	Algorithm supported	Default
01	0	ISO 9564-1 & ANSI X9.8 format 0	DES/3DES	Enabled
02	-	Docutel ATM format	DES/3DES	Disabled
03	-	Diebold & IBM ATM format	DES/3DES	Disabled
04	-	PLUS Network format	DES/3DES	Disabled
05	1	ISO 9564-1 format 1	DES/3DES	Enabled
34	2	Standard EMV 1996 format	DES/3DES	Disabled
35	-	Mastercard Pay Now & Pay Later format	DES/3DES	Enabled
41	-	Visa / Amex new PIN only format	DES/3DES	Enabled
42	-	Visa / Amex new & old PIN format	DES/3DES	Enabled
46	-	AS2805.3 Format 8 PIN block	DES/3DES	Disabled
47	3	ISO 9564-1 & ANSI X9.8 format 3	DES/3DES	Enabled
48	4	ISO 9564-1 format 4	AES	Enabled

Note that Thales Format 48 is used to encrypt a PIN Block under an AES Key Block LMK.

14.3.1 Format 01

Format 01 is the ISO 9564-1 Format 0, equivalent to the ANSI X9.8 Format 0 PIN Block, and can only be encrypted using a DES/3DES key.

The format combines the customer PIN and account number as follows:

- A 16-digit block is made from the digit 0, the length of the PIN, the PIN, and a pad character (hexadecimal F). For example, for the 5-digit PIN 92389, the block is:

  ```
  0592 389F FFFF FFFF
  ```

- Another 16-digit block is made from four zeros and the 12 right-most digits of the account number, excluding the check digit. For example, for the 13-digit account number 4000 0012 3456 2, where the check digit is 2, the block is:

  ```
  0000 4000 0012 3456
  ```

- The two blocks are exclusive-OR added:

	05	92	38	9F	FF	FF	FF	FF
	00	00	40	00	00	12	34	56
PIN block:	05	92	78	9F	FF	ED	CB	A9

14.3.2 Format 02

Format 02 supports Docutel ATMs, and can only be encrypted using a DES/3DES key. A PIN block is created from the PIN length, a 6-digit PIN, and a user-defined numeric padding string.

If the PIN has fewer than 6 digits, it is left-justified and zero filled.

For example, for the 5-digit PIN 92389, the PIN digits are 923890.

With pad characters added, the PIN block could be, for example:

```
5923 8909 8765 4321
```

Where 98765 4321 is the padding string.

14.3.3 Format 03

Format 03 supports Diebold and IBM ATMs, and can only be encrypted using a DES/3DES key. It also applies to the Docutel format that does not include a PIN length. The PIN block is created from the customer PIN and the hexadecimal F padding character. For example, for the 5-digit PIN 92389, the PIN block is:

```
9238 9FFF FFFF FFFF
```

14.3.4 Format 04

Format 04 is the PIN block format adopted by the PLUS network, and can only be encrypted using a DES/3DES key. The format combines the customer PIN and the related account number as follows:

- A 16-digit block is made from the digit 0, the length of the PIN, the PIN, and a pad character (hexadecimal F). For example, for the 5-digit PIN 92389, the block is:

```
0592 389F FFFF FFFF
```

- Another 16-digit block is made from four zeros and the left-most 12 digits of the account number. For example, for the 16-digit account number 2283 4000 0012 3456, where the check digit is 6, the block is:

```
0000 2283 4000 0012
```

- The two blocks are exclusive-OR added:

	0592	389F	FFFF	FFFF
	0000	2283	4000	0012
PIN block:	0592	1A1C	BFFF	FFED

Notes: Any transaction that requires a PIN block as a parameter accepts Format 04. The major impact of this format is on the account number field length: when a PIN block is formatted according to Format 04, the account number field becomes 18 digits in length.

For the PIN translation CA and CC commands, there are two format fields; if either is 04, the account number field must be 18 digits. If the account number is less than 18 digits, it must be right-justified and padded with X'F on the left.

The following commands can use this format:

BC, BE, CA, CC, CG, DA, DC, EA, EC, EG, G0, JC, JE, BK, FW, CU, DU, JG, KU, KY, GO, GQ, GS, GU and CI.

When reviewing the details for these commands, consider the change to the account field that this format requires.

14.3.5 Format 05

Format 05 is the ISO 9564-1 Format 1 PIN Block, and can only be encrypted using a DES/3DES key.

The PIN block is represented by the following 16 hexadecimal values:

1NPP..P R . . R

Where:

N is the PIN length (4 - C),

PP..P is the N-digit PIN,

R . . R is random padding.

The following validity checks are carried out on incoming Format 05 PIN blocks:

- The first character of the PIN block has value 1

  Error Code 20 is returned if this check fails
- The PIN digits (in positions 3 - (N+2)) are in the range 0 to 9

  Error Code 20 is returned if this check fails.
- The second character (N) is in the hexadecimal range 4 - C

  Error Code 24 is returned if this check fails.

14.3.6 Format 34

Format 34 is the ISO 9564-1 Format 2 PIN Block (the standard EMV PIN block format), and can only be encrypted using a DES/3DES key. The availability of this PIN block format is controlled via the security setting *Enable PIN Block Format 34 as output format for PIN translations to ZPK*:

when set to No (the default), this PIN block format is only available as an output from the EMV PIN Change commands (KU and KY).

- when set to Yes, this PIN block format is available as an output from the EMV PIN Change commands (KU and KY) and also as an output from the PIN Translation commands (CA, CC and G0).
- The PIN block is created from a fixed Control Field, the length of the PIN, the customer PIN itself and the hexadecimal F padding character. PINs from 4 to 12 digits in length are accommodated.

The 16-digit (8 byte) block is constructed as follows:

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

Where:

C is a fixed control field of binary value 0010 (X'2).

N is the length of the PIN and can be any binary value from 0100 to 1100 (X'4 to X'C).

P is a digit of the PIN and can be any binary value from 0000 to 1001 (X'0 to X'9).

P/F is either a PIN digit or the binary value 1111 (X'F) filler depending on the length of the PIN.

F is a filler of binary value 1111 (X'F).

Thus for a 5 digit PIN of 34567, the PIN block would hold the 16 hexadecimal values as shown below:

2	5	3	4	5	6	7	F	F	F	F	F	F	F	F	F

14.3.7 Format 35

PIN Block format 35 is the PIN Block required by Europay/Mastercard for their Pay Now & Pay Later products, and can only be encrypted using a DES/3DES key. This PIN block format is only available as an output from the EMV PIN Change commands (KU and KY).

The PIN block is created from a fixed Control Field, the length of the PIN, the customer PIN itself and the customer account number. PINs from 4 to 12 digits in length are accommodated.

A 16-digit (8 byte) block is constructed as follows:

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

Where:

C is a fixed control field of binary value 0010 (X'2).

N is the length of the PIN and can be any binary value from 0100 to 1100 (X'4 to X'C).

P is a digit of the PIN and can be any binary value from 0000 to 1001 (X'0 to X'9).

P/F is either a PIN digit or the binary value 1111 (X'F) filler depending on the length of the PIN.

F is a filler of binary value 1111 (X'F).

Thus for a 5 digit PIN of 34567, the block would hold the 16 hexadecimal values as shown below:

2	5	3	4	5	6	7	F	F	F	F	F	F	F	F	F

Another 16-digit block is made from four zeros and the 12 right-most digits of the account number, excluding the check digit.

For account number 1234 0000 0123 4562 where 2 is the check digit, the block is:

0	0	0	0	4	0	0	0	0	0	1	2	3	4	5	6

The two blocks are exclusive-ORed on a bit by bit basis:

2	5	3	4	5	6	7	F	F	F	F	F	F	F	F	F
0	0	0	0	4	0	0	0	0	0	1	2	3	4	5	6
2	5	3	4	1	6	7	F	F	F	E	D	C	B	A	9

14.3.8 Format 41

PIN Block Format 41 is the Visa format for PIN change without using the current PIN. The method for constructing the PIN block is defined in section C.11.2 of reference 4, and can only be encrypted using a DES/3DES key. This PIN block format is only available as an output from the EMV PIN Change commands (KU and KY).

The PIN Block is created using the new PIN and part of the card's unique DEA Key as follows:

- Construct a 16 hexadecimal digit block of data, by extracting the eight rightmost digits of the card application's Unique DEA Key A (UDK-A) and zero filling it on the left with eight hexadecimal zeros:

0	0	0	0	0	0	0	0								

8 Rightmost digits of card app's unique DEA key A

- Create a second 16 hexadecimal digit block of data as follows:

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

Where:

C is a fixed control field of binary value 0000 (X'0).

N is the length of the new PIN and can be any binary value from 0100 to 1100 (X'4 to X'C).

P is a digit of the new PIN and can be any binary value from 0000 to 1001 (X'0 to X'9).

P/F is either a PIN digit or the binary value 1111 (X'F) filler depending on the length of the PIN.

The Thales HSM terminology for the UDK is DK-AC. This is the card-unique key that is derived from MK-AC, the Master Key for Application Cryptograms.

F is a filler of binary value 1111 (X'F).

- Perform an exclusive-OR operation on the blocks of data created in steps 1 and 2 to create the final PIN block.

14.3.9 Format 42

PIN Block Format 42 is the Visa format for PIN change using the current (old) PIN. The method for constructing the PIN block is defined in section C.11.1 of reference 4, and can only be encrypted using a DES/3DES key. This PIN block format is only available as an output from the EMV PIN Change commands (KU and KY).

The PIN Block is created using the old PIN, the new PIN and part of the card's unique DEA Key as follows:

- Construct a 16 hexadecimal digit block of data, by extracting the eight rightmost digits of the card application's Unique DEA Key A (UDK-A) and zero filling it on the left with eight hexadecimal zeros:

0	0	0	0	0	0	0	0								

8 Rightmost digits of card app's unique DEA key A

- Create a second 16 hexadecimal digit block of data as follows:

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

Where:

C is a fixed control field of binary value 0000 (X'0).

N is the length of the new PIN and can be any binary value from 0100 to 1100 (X'4 to X'C).

P is a digit of the new PIN and can be any binary value from 0000 to 1001 (X'0 to X'9).

P/F is either a PIN digit or the binary value 1111 (X'F) filler depending on the length of the PIN.

F is a filler of binary value 1111 (X'F).

- Create a third 16 decimal digit block of data using the old PIN as follows:

P	P	P	P	P	P/0	P/0	P/0	P/0	P/0	P/0	P/0	0	0	0	0

Where:

P is a digit of the old PIN and can be any binary value from 0000 to 1001 (X'0 to X'9).

P/0 is either a PIN digit or the binary value 0000 (X'0) filler depending on the length of the PIN.

0 is a filler of binary value 0000 (X'0).

[2] The Thales HSM terminology for the UDK is *DK-AC. This is the card-unique key that is derived from *MK-AC, the Master Key for Application Cryptograms.

14.3.10 Format 46

PIN Block Format 46 is the AS2805.3 Format 8 PIN block. Section **Error! Reference source not found.** provides additional information regarding the Australian commands.

14.3.11 Format 47

PIN Block Format 47 is the ISO 9564-1 Format 3 PIN Block, and can only be encrypted using a DES/3DES key.

This PIN block is constructed by modulo 2 addition of two 64 bit fields: the plain text PIN field and the account number field.

The plain text PIN field is formatted as follows:

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

Where:

C = Control field - Binary 0011 (X'3).

N = PIN length - 4 bit binary number with permissible values of 0100 (X'4) to 1100 (X'C).

P = PIN digit   4 bit field with permissible values of 0000 (X'0) to 1001 (X'9).

P/F = PIN/Fill digit - Designation of these fields is determined by the PIN length field.

F = Fill digit - 4-bit field, with values from 1010 (X'A) to 1111 (X'F), where the fill-digit values are randomly or sequentially selected from this set of six possible values, such that it is highly unlikely that the identical configuration of fill digits will be used more than once with the same account number field by the same PIN encipherment device.

The account number field is formatted as follows:

0	0	0	0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12

Where:

0 = Pad digit - A 4 bit field; the only permissible value is 0000 (0).

A1 ... A12 = Account Number - Content is the 12 right most digits of the primary account number (PAN) excluding the check digit. A12 is the digit immediately preceding the PAN's check digit. If the PAN excluding the check digit is less than 12 digits, the digits are right justified and padded to the left with zeros. Permissible values are 0000 (0) to 1001 (9).

14.3.12 Format 48

PIN Block Format 48 is the ISO 9564-1 Format 4 PIN Block, and can only be used with an AES key.

The PIN block is constructed using two 128-bit blocks: the first containing the plaintext PIN, and the second containing the plaintext account number.

The plaintext PIN field is formatted as follows:

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Where:

C = Control field – 4-bit field value 0100 (decimal 4).

N = PIN length – 4 bit binary number with permissible values of 0100 (4) to 1100 (12).

P = PIN digit – 4 bit field with permissible values of 0000 (0) to 1001 (9).

P/F = PIN/Fill digit – designation of these fields is determined by the PIN length field.

F = Fill digit - 4-bit field, with values from 1010 (10)

R = Random digit - 4-bit field, with values from 0000 (0) to 1111 (15)

The plaintext PAN field is formatted as follows:

M	A	A	A	A	A	A	A	A	A	A	A	A	A/0	A/0	A/0	A/0	A/0	A/0	A/0	0	0	0	0	0	0	0	0	0	0	0	0

Where:

M = PAN length – 4-bit field with permissible values 0000 (0) to 0111 (7), which when added to 12, indicates the PAN length. If the PAN is less than 12 digits, the digits right justified and padded to the left with zeros, and M is set to 0.

A = PAN digit – 4-bit field with permissible values 0000 (0) to 1001 (9)

0 = Pad digit – 4-bit field with only permissible value 0000 (0)

A/0 = PAN/Pad digit – designation of these fields is determined by the PAN length field.

To produce an encrypted PIN block in this format, using an AES PIN encryption key, K:

1. Format the PIN into the plaintext PIN field as shown above.
2. Format the PAN into the plaintext PAN field as shown above.
3. Encrypt the plaintext PIN field (from step 1) using the key, K.
4. XOR the plaintext PAN field (from step 2) with the result of step 3.
5. Encrypt the result of step 4 using the key, K.

To decrypt an encrypted PIN block in this format, using an AES PIN encryption key, K:

1. Format the PAN into the plaintext PAN field as shown above.
2. Decrypt the encrypted PIN block using the key, K.
3. XOR the plaintext PAN field (from step 1) with the result of step 2.
4. Decrypt the result of step 3 using the key, K.
5. Extract the PIN from the result of step 4.

14.4 Support for Deprecated Proprietary PIN Block Format Tag J

14.4.1 Overview

Support for the encryption of PIN Blocks under an AES Key Block LMK for Issuers is provided using an AES Key Block LMK in v1.6a and above. From this release onwards, the tag 'M' is used to indicate the PIN Block is encrypted under an AES Key Block LMK using ISO PIN Block Format 4 (Thales PIN Block Format 48).

In payShield 9000 and payShield 10K, releases up to and including v1.5a, support was provided for encryption of a PIN Block under an AES Key Block LMK using tag 'J' which indicates the PIN Block is encrypted under a proprietary format. Support for this format was withdrawn in v1.6a. However to allow customers more time to migrate to the format indicated by tag 'M', tag 'J' is supported as well tag 'M' in v1.8a and a limited number of releases following v1.8a.

The re-introduction of support for tag 'J' is designed to allow applications still using this format to continue without changing the application. This also allows the relevant host commands to support both Tag 'J' and tag 'M' in order to simplify the migration process to tag 'M'. The only exception are host commands JC (Translate a PIN from TPK to LMK Encryption) and JE (Translate a PIN from ZPK to LMK Encryption) where the format used is determined from a new Security Setting.

There are two differences in the support provided using Tag 'J' when compared to the earlier releases and these are as follows. These both impact host commands JC (Translate a PIN from TPK to LMK Encryption) and JE (Translate a PIN from ZPK to LMK Encryption) only:

- A new Security Setting is included to determine whether Tag J'' is used with host commands JC and JE (the other host commands are not affected by this setting):
- "Use deprecated proprietary format (Tag J) when using PIN Blocks under AES Key Block LMK"
- The default setting is NO and this indicates Tag 'M' is used.

- Translation of a PIN Blocks from ISO Formats 0, 1, 3 and 4 (Thales PIN block formats 01, 05, 47 and 48 respectively) to encryption under an AES Key Block LMK using Tag 'J' are no longer allowed if the security setting "Restrict PIN Block Usage for PCI Compliance" is set to YES.

14.4.2 Miscellaneous Host Commands

14.4.2.1 Overview

This section applies to host commands where the only PIN Block supplied is the PIN Block encrypted under an LMK and a PIN Block encrypted under a TPK or ZPK is not returned.

Here, the format of the PIN Block encrypted under the LMK returned is determined from the tag used for the PIN input parameter.

14.4.2.2 Host Commands Included

The host commands in this set are:

PIN and Offset Generation Host Commands

- DE - Generate an IBM PIN Offset (of an LMK encrypted PIN)
- CE - Generate a Diebold PIN Offset
- DG - Generate an ABA PVV (of an LMK encrypted PIN)

PIN Mailer Printing Host Commands

- PE - Print PIN/PIN and Solicitation Data
- PG - Verify PIN/PIN and Solicitation Mailer Cryptography

Clear PIN Commands

- NG - Decrypt an Encrypted PIN

PIN Translation Host Commands

- QK - Translate Account Number for LMK-encrypted PIN

LMK Translation Host Commands

- BG - Translate a PIN and PIN Length

14.4.2.3 Host Command Field Details

To use Tag J, the following changes apply to the descriptions of the Fields below to the information provided in the Host Command Manual:

Field	**Length & Type**	**Details**
	COMMAND MESSAGE – *PIN, PAN and Delimiter Fields*	
PIN	'J' + 32 H	The PIN encrypted under the LMK. When using an AES Key Block LMK with Tag ‘J’, this field must consist of a 'J' followed by 32 hex digits.
Primary Account Number (PAN)	12 N	The 12 right-most digits of the PAN, excluding the check digit.
Delimiter	1 A	Value ';'. The delimiter is NOT present when using Tag ‘J’.

14.4.2.4 Host Response Message Field Details (BG & QK Only)

To use Tag J, the following changes apply to the descriptions of the Fields provided in the Host Command Manual:

Field	**Length & Type**	**Details**
	RESPONSE MESSAGE – *PIN Field – Host Commands **QK** and **BG** only*	
PIN	'J' + 32 H	The PIN encrypted under the LMK.

14.4.3 PIN Verification Host Commands

14.4.3.1 Overview

This section applies to host commands where the PIN Block is provided both encrypted under an LMK and a ZPK/TPK/BDK.

In this case, the PIN Block Format used is determined from the PIN Block Format provided when using an AES Key Block LMK.

Note that there is a known issue with the original implementation of these commands when the PIN Block is supplied in PIN Block Format 48. This has not been corrected and to address this we recommend migrating to use Tag ‘M’.

Also note that the ‘GU’ Host Command does not support PIN Block 48 when used with Tag ‘J’.

14.4.3.2 Host Commands Included

The host commands in this set are:

PIN Verification Host Commands

- BC - Verify a Terminal PIN Using the Comparison Method
- BE - Verify an Interchange PIN Using the Comparison Method
- GU - Verify a PIN Using the Encrypted PIN Method (3DES & AES DUKPT)

14.4.3.3 Host Command Field Details

To use Tag J, the following changes apply to the descriptions of the Fields provided in the Host Command Manual:

Field	Length & Type	Details
COMMAND MESSAGE – *PIN, PAN and Delimiter Fields*		
PIN	'J' + 32 H	The PIN encrypted under the LMK. When using an AES Key Block LMK with Tag ‘J’, this field must consist of a 'J' followed by 32 hex digits.
For all values of 'PIN Block Format Code' except ‘48’ the following field must be present:		
Primary Account Number (PAN)	18 H or 12 N	The PAN, used to form the PIN Block. If 'PIN Block Format Code' = '04': The 18 digit PAN excluding the check digit. If the PAN is less than 18 digits, it must be right-justified and padded with 'F's on the left. For all other values of 'PIN Block Format Code': The 12 right-most digits of the PAN, excluding the check digit.
For 'PIN Block Format Code' = ‘48’ the following two fields must be present:		
Primary Account Number (PAN)	n N	The PAN, used to form the PIN Block. The 12-19 digit PAN, including the check digit. If present, the delimiter below must also be present.
Delimiter	1 A	Value ';'. Only present if 'PIN Block Format Code' = ‘48’

14.4.4 Clear PIN and Remaining PIN Generation Host Commands

14.4.4.1 Overview

Host commands in this category are those where the PIN Block is not supplied as an input parameter either encrypted under an LMK or encrypted under a ZPK/TPK.

For these host commands, the Tag used is determined form the format of the PAN supplied when using an AES Key Block LMK:

- If the PAN is supplied as 12 N, the Tag 'J' is used as the output.
- If the PAN is supplied as nN with the delimiter, Tag 'M' is used as the output.

14.4.4.2 Host Commands Included

The host commands in this set are:

PIN and Offset Generation Host Commands

- EE - Derive a PIN Using the IBM Offset Method
- GA - Derive a PIN Using the Diebold Method
- JA - Generate a Random PIN

Clear PIN Commands

- BA - Encrypt a Clear PIN

14.4.4.3 Host Command Message Field Details

To use Tag J, the following changes apply to the descriptions of the Fields provided in the Host Command Manual:

Field	Length & Type	Details
COMMAND MESSAGE – *PAN and Delimiter Fields*		
Primary Account Number (PAN)	12 N	The 12 right-most digits of the PAN, excluding the check digit.
Delimiter	1 A	Value ';'. The delimiter is NOT present when using Tag 'J'.

14.4.4.4 Host Response Message Field Details

To use Tag J, the following changes apply to the descriptions of the Fields provided in the Host Command Manual:

Field	Length & Type	Details
RESPONSE MESSAGE – *PIN Field*		
PIN	'J' + 32 H	The PIN encrypted under the LMK.

14.4.5 PIN Translation from Encryption under an LMK Host Commands

14.4.5.1 Overview

Host commands in this category are those where the only PIN Block provided is a PIN Block encrypted under a ZPK/TPK/BDK.

To use Tag 'J' with these host commands, the Security Parameter "Use deprecated proprietary format (Tag J) when using PIN Blocks under AES Key Block LMK" must be set to YES.

Note that there is a known issue with the original implementation of these commands when the PIN Block is supplied in PIN Block Format 48. This has not been corrected and to address this we recommend migrating to use Tag 'M'.

14.4.5.2 Host Commands Included

The host commands in this set are:

PIN Translation Host Commands

- JE - Translate a PIN from ZPK to LMK Encryption
- JC - Translate a PIN from TPK to LMK Encryption

14.4.5.3 Host Command Message Field Details

To use Tag J, the following changes apply to the descriptions of the Fields provided in the Host Command Manual:

Field	Length & Type	Details
COMMAND MESSAGE – *PAN and Delimiter Fields*		
For all values of 'PIN Block Format Code' except '48' the following field must be present:		
Primary Account Number (PAN)	18 H or 12 N	The PAN, used to form the PIN Block. If 'PIN Block Format Code' = '04': The 18 digit PAN excluding the check digit. If the PAN is less than 18 digits, it must be right-justified and padded with 'F's on the left. For all other values of 'PIN Block Format Code': The 12 right-most digits of the PAN, excluding the check digit.
For 'PIN Block Format Code' = '48' the following two fields must be present:		
Primary Account Number (PAN)	n N	The PAN, used to form the PIN Block. The 12-19 digit PAN, including the check digit. If present, the delimiter below must also be present.
Delimiter	1 A	Value ';'. Only present if 'PIN Block Format Code' = '48'

14.4.5.4 Host Response Message Field Details

To use Tag J, the following changes apply to the descriptions of the Fields provided in the Host Command Manual:

Field	Length & Type	Details
RESPONSE MESSAGE – *PIN Field*		
PIN	'J' + 32 H	The PIN encrypted under the LMK.

14.4.6 PIN Translation to Encryption under an LMK Host Commands

14.4.6.1 Overview

The Host command in this category is where the only PIN Block supplied is the PIN Block encrypted under an LMK and a PIN Block under a ZPK is returned.

Here, the format of the PIN Block encrypted under the LMK is be determined from the PIN input parameter.

Note that there is a known issue with the original implementation of these commands when the PIN Block is supplied in PIN Block Format 48. This has not been corrected and to address this we recommend migrating to use Tag 'M'.

14.4.6.2 Host Commands Included

The host commands in this set are:

PIN Translation Host Commands

- JG - Translate a PIN from LMK to ZPK Encryption

14.4.6.3 Host Command Message Field Details

To use Tag J, the following changes apply to the descriptions of the Fields provided in the Host Command Manual:

Field	Length & Type	Details
COMMAND MESSAGE – *PIN, PAN and Delimiter Fields*		
PIN	'J' + 32 H	The PIN encrypted under the LMK. When using an AES Key Block LMK with Tag 'J', this field must consist of a 'J' followed by 32 hex digits.
For all values of 'PIN Block Format Code' except '48' the following field must be present:		
Primary Account Number (PAN)	18 H or 12 N	The PAN, used to form the PIN Block. If 'PIN Block Format Code' = '04': The 18 digit PAN excluding the check digit. If the PAN is less than 18 digits, it must be right-justified and padded with 'F's on the left. For all other values of 'PIN Block Format Code': The 12 right-most digits of the PAN, excluding the check digit.
For 'PIN Block Format Code' = '48' the following two fields must be present:		
Primary Account Number (PAN)	n N	The PAN, used to form the PIN Block. The 12-19 digit PAN, including the check digit. If present, the delimiter below must also be present.
Delimiter	1 A	Value ';'. Only present if 'PIN Block Format Code' = '48'

15 Key Component Printing

15.1 Introduction

Payment processing is a multi-organizational activity, with the various organizations involved needing to exchange encryption keys. The working keys that the parties need to exchange are protected by encrypting them using a KEK; this term covers any Key Encrypting Key, including specialized KEKs such as Zone Master Keys (ZMKs) or Terminal Master Keys (TMKs). The problem is how the KEKs themselves are to be exchanged.

Further information on the TMD can be found in the payShield 10K Installation and User Guide and in the TMD User Guide.

Because all of these parties may be using different, incompatible IT infrastructures, KEKs are typically exchanged by splitting them into multiple components which are printed out, with each component being given to a different officer in the recipient organization; the KEK is re-formed at the recipient organization by the component holders coming together and entering the components into a key-forming application. No one individual acting alone has complete knowledge of the KEK or the ability to form it in the recipient's system.

This chapter describes how the payShield 10K could be used to generate and print KEK components at the issuing organization, and to re-form the KEK from components at the recipient organization.

Note: You will need to review the suggested methods described in this chapter with your organization's security team to ensure that these methods meet the your organizations' security and operational requirements.

15.2 Process Description

15.2.1 Overview

Key Encrypting Keys (KEKs), including specialized keys such as TMKs, ZMKs, and ZCMKs, are shared between organizations to allow them to securely exchange working keys by encrypting them using the KEKs.

A KEK that is to be shared between two organizations is generated by one of the parties and conveyed to the other in a secure manner.

Each user organization designs its own component distribution process to meet its operational and security requirements. One possible sequence of events, using a payShield 10K to provide a high level of security, is as follows:

- At the KEK issuing organization:
    - Set up Key Component Print Form
    - Generate and print the desired number of KEK components, and retain a copy of the components encrypted using the payShield 10K's Local Master Key (LMK)
- Deliver each printed component to its designated officer in the recipient organization

    - Use the encrypted components to form the KEK using a payShield 10K at the issuing organization
    - Store the KEK in the issuing organization's key database, encrypted using the payShield 10K's LMK

- At the recipient organization:
    - The component officers come together at a payShield 10K and individually enter their components
    - The recipient organization's payShield 10K forms the key from the entered components
    - The payShield 10K displays the KEK formed from the components, encrypted with the payShield 10K's LMK
    - The LMK-encrypted KEK is entered into the recipient organization's key database. It can then be included in commands used to exchange working keys with the issuing organization.

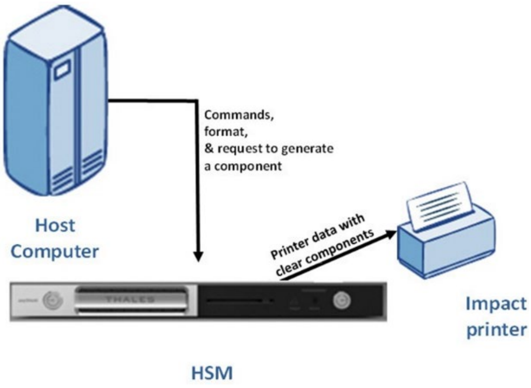
Note: An alternative approach is to use the Thales Trusted Management Device (TMD). The TMD replaces the Thales Key Management Device (KMD).

The possible process model outlined here can be modified to the needs of individual organizations. As an example of this, an approach taken by some users of Thales payment HSMs for setting up TMKs is to pre-print batches of components in secure envelopes, with an associated reference number; the component check value has been used for this by at least one organization, but in this case a process must be put in place to allow for the (unlikely) occurrence of a non-unique value. A supply of components is provided to each component holder. All the components are also held at the host system. The component holders visit the ATMs, and each takes any one of their stock of components and enters it into the ATM to form the ATM's copy of the TMK. Each component holder then calls the operations center and provides the reference number of the component they used. The Host computer can then retrieve the same components and form its copy of the TMK.

As mentioned above, each organization needs to design its own procedures to provide the level of security that the organization feels is appropriate. It is suggested that these procedures include the disposal of printed components after they have been used.

15.2.2 Directly Attached Printers

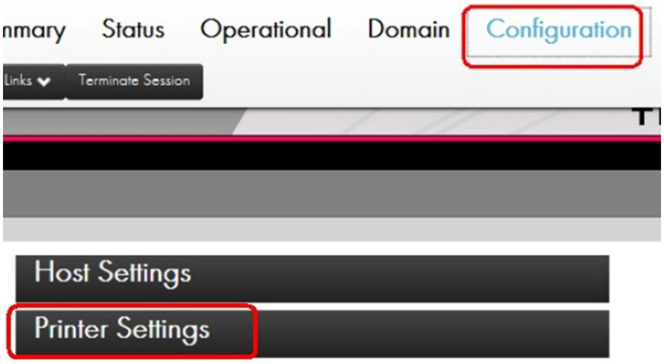
Multiple components need to be printed, but they must only be disclosed to the single user that they are to be addressed to. Using the payShield 10K, the components are created one at a time. It is suggested that these components are printed at an impact printer directly attached to the HSM, and onto specialized multi-part, tamper- evident mailers that prevent the component being accessed until the mailer is opened. Non-sensitive data, such as date and addressee, can be printed on the outside of the mailer.



The payShield 10K can print to:

- A parallel printer attached to a USB port on the payShield 10K via a USB-to-parallel cable available from Thales
- A serial printer attached to a USB port on the payShield 10K via a USB-to-serial cable available from Thales
- A USB printer attached to a USB port on the payShield 10K via a standard USB cable.

To choose which type of printer to use and to configure the printer, use the CP (Configure Printer) console command or payShield Manager Configuration / Printer Settings.



15.2.2.1 Non-impact Printers

It is unlikely that laser or ink-jet printers can be used in this way, because use of this type of printer to print secret information (such as PINs) requires specialized stationery with plasticized windows; it is unlikely that stationery with large enough windows for component printing will be readily available.

Any type of printer could be used to print components onto plain stationery. However, if plain stationery is to be used, security measures should be in place to ensure that the component can be seen by no more than one person and that any one person cannot see more than one component.

15.2.3 Operation Without a Directly Attached Printer

If no directly attached printer is available, components can be displayed to the users. In this case, security processes must be in place to ensure that each component can be seen by no more than one person, that any one person cannot see more than one component, and that procedures are in place to protect the components (e.g., if they are transcribed onto paper).

15.2.4 Setting Up Key Component Print Forms

A form definition needs to be loaded into the payShield 10K so that it can format the output sent to the printer.

The sections that follow describes how this can be used for key component printing.

The payShield 10K can hold a single form definition at a time. Formatting data is loaded onto the HSM as text strings. This data consists of:

- Formatting symbols to format the data
- Constants (text strings)
- Variable print field markers, which will be populated with data when the PIN mailers are to be printed A key component print formatting string might look as follows:

```
>L>003^0>033^1>L>003KEY COMPONENT PART 1: ^P>L>003 KEY COMPONENT PART 2:
^Q>L>003 KEY COMPONENT PART 3: ^R>L>L>003 KEY CHECK VALUE: ^T>L>L>003 DO
NOT DISCLOSE THIS COMPONENT TO ANYONE ELSE>L>F
```

In this example, the following print formatting symbols are used:

- >L carriage return + line feed (CR/LF)
- >nnn skip to output column nnn
- ^n insert variable print field n: the actual value will be provided at print time
- >F Form Feed (FF)
- ^T the key component check value.
- ^P, ^Q, and ^R have the following meanings:

Key length	^P	^Q	^R
Single	1st 8 hex digits of component	2nd 8 hex digits of component	N/A
Double	1st 16 hex digits of component	2nd 16 hex digits of component	N/A
Triple	1st 16 hex digits of component	2nd 16 hex digits of component	3rd 16 hex digits of component

The full set of print formatting symbols is defined in *Appendix C - Print Formatting Symbols.*

The above example would print the following key component form:

```
..(Field 0)....................(Field 1)
..KEY.COMPONENT.PART.1:.xxxxxxxxxxxxxxxx
..KEY.COMPONENT.PART.2:.yyyyyyyyyyyyyyyy
..KEY.COMPONENT.PART.3:.zzzzzzzzzzzzzzzz

..KEY.CHECK.VALUE:.vvvvvv
```

Notes:

- "." is used in this example to indicate where a space would be printed

- "(Field 0)" etc. represents variable data (such as date and addressee) that the host computer application will provide at print time
- xxx…x, yyy…y, zzz…z represent parts of the key component, each representing 16 hexadecimal digits
- vvvvvv represents the component check value

  The maximum length of a form definition is 299 symbols and characters of constant data

  This form definition is loaded onto the HSM using Host commands. The required host commands are as follows - see the *payShield 10K Core Host Commands reference manual* for full details:

payShield 10K Host Command	
ID	**Command Description**
PA	Load Formatting Data to HSM - to load the first part of the set of symbols and constant characters.
PC	Load Additional Formatting Data to HSM - to load any continuation of the symbols and constant characters. (PC is a continuation of PA, and must be preceded by a PA command.)

Generating and printing components

If using a directly-attached printer

The following single payShield 10K Host command performs these actions:

- Generates a random key component.
- Prints the key component at the printer directly attached to the HSM, using the print form currently installed on the HSM (using the PA and PC host commands).
- Encrypts the component using the HSM's LMK and returns it to the originating organization's host computer system.

payShield 10K Host Command	
ID	**Command Description**
A2	Generate and print a component

Typically 3 components will be generated and printed in this way, by running the A2 command once for each component.

Each component will be printed in its own mailer, which should be designed such that the component cannot be read from the outside of the mailer, and any attempt to access the component inside the mailer will be evident.

The HSM will provide 2 responses to the A2 command sent by the host:

payShield 10K Response to Host	
ID	**Command Description**
A3	Initial response, before printing. This returns the component encrypted under the HSM's LMK, and the component check value.
AZ	Second response, providing the outcome of the printing operation.

The issuer's host will receive and hold the encrypted component.

If no HSM-attached printer is available

The A2 Host command depends on the payShield 10K having a directly attached printer. Where such a printer is not available, key components can be generated using the following Console command:

payShield 10K Console Command	
ID	**Command Description**
GC	Generate key component

This command generates both the clear component and the component encrypted under the payShield 10K's LMK and displays them to the user.

The output from this command must be manually written or printed onto the mailer using a separate system.

15.2.5 Security Considerations

Whenever components are output in any way other than inside a secure, tamper-evident mailer, the originating organization should satisfy themselves that procedures are in place to ensure that not more than one person can see any component, and that any one person cannot see more than one component.

15.2.5.1 Avoiding Use of Legacy Commands

The payShield 10K also provides the NE Host command. This might appear to be a more convenient method of creating components because a single command generates the key and prints multiple components.

However, the NE command is provided purely for backwards compatibility with older systems that were designed to make use of it. It is recommended that use of this command is avoided, and that it is disabled using the CONFIGCMDS Console command or in the payShield Manager Configuration / Configure Commands / Host dialogue box, because it generates components as split keys rather than full-length components which are XORed together.

15.2.5.2 Delivery of Printed Components

These mailers are delivered to the appropriate key management officers at the recipient organization, using secure delivery methods.

15.2.6 Forming the KEK at the Issuer System

15.2.6.1 If Components Were Printed at a Directly Attached Printer

If a payShield 10K-attached printer has been used to print components, typically 3 components will have been generated using the HSM's A2 Host command and encrypted under the HSM's LMK. These encrypted components are held on the issuer's host system.

The issuer's host now forms the KEK from the encrypted components it is holding by sending the components to a payShield 10K in the following host command:

payShield 10K Host Command	
ID	**Command Description**
A4	Form a key from encrypted components

The payShield 10K will form the KEK, encrypt it using its LMK, and return the LMK-encrypted KEK to the host system. The host can now add this encrypted KEK to its key database. Whenever a cryptographic operation requires the KEK, the encrypted KEK will be included in the host command sent to the HSM and the HSM will decrypt the KEK and use it within its secure boundary.

15.2.6.2 If a Directly Attached Printer Was Not Used

Where no HSM-attached printer is available and components have been generated using the GC Console command, then the KEK should be formed in the same way.

15.2.7 Forming the KEK at the Recipient System

The key management officers at the recipient organization will each have received their own KEK component. Each one of them has access to only one component, and all of the component holders need to come together to allow the KEK to be formed from the components.

There are 3 ways that this can be done using the payShield 10K. In each case:

- the component holders come together
- each enters the component they hold
- the KEK is displayed, encrypted under the HSM's LMK
- the encrypted KEK is noted down and entered into the recipient host system's key database, using the appropriate key entry tool at the host.

The three payShield 10K mechanisms available for doing this are as follows.

15.2.7.1 Console

The following Console command allows the component holders to enter their components sequentially and then displays the encrypted key and its check value:

payShield 10K Console Command	
ID	**Command Description**
FK	Form key from components

This Console command can also form a key from encrypted components (e.g. if the encrypted components generated by the GC Console command option had been provided to the recipient).

15.2.7.2 payShield Manager

The ability to Install an LMK from RLMK Card Set can be found in payShield Manager at Operational / LMK Operations / Local master Keys.

15.2.7.3 payShield Trusted Management Device (TMD)

The standalone Thales Trusted Management Device (TMD) can be used to form an LMK-encrypted key from components.

Note: The TMD replaces the Thales Key Management Device (KMD).

16 Moving to PCI HSM Compliance

16.1 Introduction

The PCI SSC (Payment Card Industry Security Standards Council) publishes security standards relating to the issuing of credit/debit cards and the processing of their transactions. When these standards become mandated by the card schemes, organizations are audited against those standards that relate to their operations and equipment manufacturers must ensure that their products comply with the standards which are relevant to them.

The PCI HSM standard relates specifically to HSMs, such as the payShield 10K. Compliance with the PCI HSM standard is sought by many users and is required for compliance with certain other PCI standards (e.g., Point-to- Point Encryption and mPOS; Card Production).

The payShield 10K hardware has been certified as being compliant with this standard. In order to achieve this certification, a number of changes were made to the payShield 10K software, some of which apply to user operation of the device and some of which are relevant to application developers. This chapter focusses on the changes which may impact on developers:

- PIN Blocks
- Updating Key Type 002 in host commands
- Migrating keys from Key Type 002
- Diebold Table re-encryption

Note that where changes have been made to the payShield 10K to enable it to comply with the requirements of the PCI HSM standard, the user can decide whether to use these changes through the payShield 10K's security settings. If the changes to the software are not used then the payShield 10K can run with full compatibility with legacy units, but it is then not compliant with the requirements of PCI HSM.

16.2 PIN Blocks

16.2.1 Using PIN Block Formats in a Compliant Manner

PCI HSM compliance requires adherence to the recommendations of ISO 9564 / ANSI X9.8 in terms of allowed PIN Block Format translations and usage.

The allowed translations and usage are a sub-set of what was previously available on the payShield 10K. Therefore users could already be compliant with ISO 9564 / ANSI X9.8 with older HSMs, simply by using only the allowed functionality. However, for PCI HSM compliance the payShield 10K must enforce the limitations.

For users not already conforming to ISO 9564 / ANSI X9.8, the payShield 10K allows the user to continue operating in this fashion until such time as the user wishes to become PCI HSM compliant. This is done by manipulating the security setting Restrict PIN block usage for PCI compliance:

- if this is not set the user can continue to using PIN Block Format translations or usage outside of those allowed by ISO 9564 / ANSI X9.8 and PCI HSM

- if it is set then only the permitted functionality is available.

16.2.2 Permitted PIN Block Format Translations

When the security setting *Restrict PIN block usage for PCI compliance* is set to Yes, the permissible PIN Block format translations are:

			Translation to:								
	Thales PIN Block Format		01	02	03	04	05	34	46	47	48
		ISO Format	0	-	-	-	1	2	-	3	4
Translation from:	01	0	☑							☑	☑
	02	-	☑	☑	☑	☑	☑		☑	☑	☑
	03	-	☑	☑	☑	☑	☑		☑	☑	☑
	04	-	☑			☑				☑	☑
	05	1	☑				☑			☑	☑
	46	-	☑	☑	☑	☑	☑		☑	☑	☑
	47	3	☑							☑	☑
	48	4	☑							☑	☑

With this security setting, users must ensure that their applications are not requesting non-permissible PIN Block Format translations when sending any of the following Host commands to the payShield 10K:

Command	
CA	Translate PIN from TPK to ZPK
CC	Translate PIN from one ZPK to another
G0	Translate a PIN from BDK to BDK or ZPK Encryption (3DES DUKPT)
G6	Translate PIN from one ZPK to another with SEED
G8	Translate PIN from TPK to ZPK with SEED

16.2.3 PIN Block Format for Values Derived from PIN & PAN

When the security setting Restrict PIN block usage for PCI compliance is set, payShield 10K enforces the ISO 9564 / ANSI X9.8 and PCI HSM requirement that only Thales PIN Block formats 01 (ISO format 0), 47 (ISO format 3), and 48 (ISO format 4) may be used when calculating values (e.g. offsets, PVV) from the PIN and PAN.

The Host commands affected by this setting are:

Command	
BK	Generate IBM PIN Offset (customer-selected PIN)
CE	Generate Diebold PIN Offset
CU	Verify & Generate a Visa PVV
DE	Generate IBM PIN Offset
DG	Generate Visa PVV
DU	Verify & Generate IBM PIN Offset
FW	Generate Visa PVV (customer-selected PIN)

Where the security setting is not set, this restriction is not enforced and users can continue to use other PIN Block formats to calculate offsets, PVVs, etc., until such time as the user wants to implement an ISO 9564 / ANSI X9.8 and PCI HSM compliant environment.

16.2.4 PIN Translations to/from LMK Encryption

When the security setting Restrict PIN block usage for PCI compliance is set, payShield 10K will not permit PIN translations to/from any 3DES LMK (since all types of 3DES LMK use proprietary PIN block formats when encrypting PINs). However, an AES Key Block LMK uses ISO 9564-1 format 4 when encrypting PINs, and so PIN block translations to/from encryption under an AES Key Block LMK are always permitted.

16.2.5 Summary of Required Actions

- If not already the case, change applications to use only permitted PIN Block format translations.
- If not already the case, change applications to use only ISO format 0, 3, and 4 PIN Blocks to calculate values from PIN and PAN.

16.3 Updating Key Type 002 in Host Commands

Note: This change does not affect users who have implemented Key Block LMKs. Moving to Key Block LMKs is the better approach from a strategic viewpoint: users who wish to continue working with Variant LMKs should implement the guidelines provided below.

PCI HSM certification requires additional key separation to be applied to the HSM. This means that, for users of Variant LMKs, various data and keys in key type 002 (i.e. encrypted with LMK Pair 14-15, variant 0) must be moved to a different key type.

Users can plan when they want to move the keys from the legacy key type (002) to the new PCI HSM compliant key types by making use of the security setting Enforce key type 002 separation for PCI HSM compliance:

If this is not set, then the affected keys remain in key type 002: this provides interoperability with legacy systems but means that the payShield 10K is not PCI HSM compliant.

If the setting is set, the affected keys use the new key types as required for PCI HSM compliance.

In order to operate with the new key types, Host applications issuing commands to the HSM which specify the key type must be modified to specify the new key type.

The Host commands affected by this are:

Host Command	
A0	Generate a Key
A2	Generate and Print a Component
A4	Form a Key from Encrypted Components
A6	Import a Key
A8	Export a Key
B0	Translate Key Scheme
BK	Gen IBM PIN Offset (customer selected PIN)
BU	Generate a Key Check Value
BW	Translate Keys from Old to New LMK
CU	Verify & Generate a Visa PVV (of a customer selected PIN)
DU	Verify & Generate an IBM PIN Offset (of customer selected new PIN)
FW	Generate a Visa PIN Verification Value (of a customer selected PIN)
GI	Import Key under an RSA Public Key
GK	Export Key under an RSA Public Key
NE	Gen and Print Key as Split Components

16.3.1 Example

If it is required to use the A0 Host command to generate a TPK where:

- the key is to be encrypted under the default Variant LMK
- the key does not need to be encrypted under a ZMK or TMK
- the key does not need to be exported in ANSI X.9.143 format
- no command message trailer is required.

A legacy application might send the following Host command:

Field	Value
Message Header	1234
Command Code	A0
Mode	0
Key Type	002
Key Scheme	U

If the security setting Enforce key type 002 separation for PCI HSM compliance is set, the Host command would be modified to the following:

Field	Value
Message Header	1234
Command Code	A0
Mode	0

Key Type	70D
Key Scheme	U

16.3.2 Interoperability with Older HSMs

Where users have a mixed estate of PCI HSM compliant and non-compliant payShield 10K and HSM 8000 HSMs, all of their HSMs must be able to use the same key types.

HSM 8000 users should upgrade to v3.3a or later. These versions of HSM 8000 software include the same capability of changing key types as described above. Some Host commands will also be disabled. (Note that the HSM 8000 is not PCI HSM compliant, even with version 3.3a of software installed.)

Users of payShield 10K with software versions earlier than v1.2a should upgrade their HSMs to v1.2a or later. This will provide interoperability but will not necessarily make the payShield 10K PCI HSM compliant.

It is not possible for a PCI HSM compliant payShield 10K HSM to inter-operate with older Thales RG6000 and RG7000 payment HSMs.

16.3.3 Summary of Required Actions

- Amend key type values specified in Host commands to take account of the new key types.

16.4 Migrating keys from Key Type 002

Note: This change does not affect users who have implemented Key Block LMKs. Moving to Key Block LMKs is the better approach from a strategic viewpoint: users who wish to continue working with Variant LMKs can migrate their keys following the guidelines provided below.

As described in the section Updating Key Types 002 in Host commands above, a number of keys will have different key types if the security setting Enforce key type 002 separation for PCI HSM compliance has the PCI HSM compliant value. This means that the users' relevant operational keys will need to be migrated from key type 002 to their new key type.

It is suggested that this can be done as part of the regular LMK refreshes that users should be performing as part of their security procedures. LMK refresh is performed by a Host application making use of the payShield 10K's BW Host command. The BW Host command includes an option for the migration of keys from key type 002 at the same time as refreshing the LMK. (The BW command can continue to be used just for LMK refresh, and can also be used to perform the key migration without LMK refresh.)

The processes below indicate approaches that might be taken to achieve migration, but users will need to design processes that suit their own environment. The starting point in these processes is that transaction processing is using the existing key database, and the security setting Enforce key type 002 separation for PCI HSM compliance has not been set on any HSM.

16.4.1 Migrating Keys WITHOUT a Change of LMK

16.4.1.1 Creating a New key Database for the New Key Types

- Create a Host process which will be able to take each affected key from a key database, Use the BW Host command (example below) to migrate the key from key type 002 to the new key type, and replace the original key in the key database with the re-encrypted key returned by the BX response.
- Clone each live key database to a new key database. At this stage the new key database contains the un- migrated keys and is not being used for processing - which continues using the live key database.
- For each set of LMKs in use:
    - Set up one (or more) payShield 10K which has the active LMKs loaded with the security setting Enforce key type 002 separation for PCI HSM compliance not set.
    - Use this HSM with the Host process to migrate all the affected keys in the new key database. The HSM can continue to be used for transaction processing using the old unmigrated keys at the same time, if required.

16.4.1.2 Switching the HSMs to the New Key Types

- The HSMs need to be switched over to use the new key types at a convenient point. During this switch-over, the HSMs will be unavailable for processing for a period of up to a few minutes. Depending on the nature of the applications and the systems design, it may be appropriate to do this switchover for:
    - All HSMs using a particular key database. (This might be all HSMs, or all HSMs for an application, or all HSMs for an LMK - dependent on the organization of the key databases). This makes it easier for the Host application to switch between old and new key databases, but could involve an interruption of service if a Disaster Recovery site is not available to pick up processing using the old unmigrated keys during this switchover.
    - A single HSM or a sub-set of the HSMs using a key database, leaving other HSMs available to continue providing a service using the old unmigrated keys. This avoids a complete interruption of service, but requires the Host application to determine which key database to use for each HSM - e.g. by using the NO command.
- For the HSMs to be switched over:
    - Suspend processing on the HSMs, or use a failover mechanism (such as an SRM) to re-direct commands to an alternative HSM.
    - Put the HSMs into secure state.
    - Set the security setting Enforce key type 002 separation for PCI HSM compliance.
    - Resume processing on the HSMs, but using the new key database and the Host application modified to use the new key types.
- When the process is complete
    - archive the old key database

16.4.2 Migrating Keys WITH a Change of LMK

- Create a Host process which will take each key from a key database, use the BW Host command (example below) to change the LMK for each key and migrate relevant keys from key type 002 to the new key type, and write the re-encrypted key returned by the BX response back to the key database.
- Clone each live key database to a new key database. At this stage the new key database holds the old un- migrated keys and is not being used for processing - processing continues using the live key database.
- For each set of LMKs in use:
    - Set up one (or more) payShield 10K which has the old LMK (i.e., the one currently being used to process transactions) loaded as its live LMK (e.g. by using the LK Console command) and the new LMK stored in its Key Change Storage (e.g. by using the LN Console command). The security setting Enforce key type 002 separation for PCI HSM compliance must not be set. It can still be used for transaction processing if required.
    - Use this HSM with the Host process to migrate all the keys in the new key database by overwriting the old keys with keys newly encrypted using the new LMK and new key types.

16.4.2.1 Switching the HSMs to the New Key Types

- The HSMs need to be switched over to use the new key types at a convenient point. During this switch-over, the HSMs will be unavailable for processing for a period of up to a few minutes. Depending on the nature of the applications and the systems design, it may be appropriate to do this switchover for:
    - All HSMs using a key database. (This might be all HSMs, or all HSMs for an application, or all HSMs for an LMK - dependent on the organization of the key databases). This makes it easier for the Host application to switch between old and new key databases, but could involve an interruption of service if a Disaster Recovery is not available.
    - A single HSM or a sub-set of the HSMs using a key database, leaving other HSMs available to continue providing a service. This avoids a complete interruption of service, but requires the Host application to determine which key database to use for each HSM - e.g., by using the NO command.
- For the HSMs to be switched over:
    - Suspend processing on the HSMs, or use a failover mechanism (such as an SRM) to re-direct commands to an alternative HSM
    - Put the HSMs into secure state
    - On each HSM set the security setting Enforce key type 002 separation for PCI HSM compliance
    - On each HSM delete the old LMK (e.g., using the DM Console command) and load the new LMK (e.g. using the LK Console command)
    - Resume processing on the group of HSMs, but using the new key database and the modified Host application using the new key types

Note: The BW command does not allow keys to be migrated back to key type 002. Therefore if there is any possibility that the security setting Enforce key type 002 separation for PCI HSM compliance is to be unset then the old key database must be retained.

16.4.3 Using Disaster Recovery (DR) Sites

Where a DR site is available that replicates the capabilities of the live site, it may be appropriate to implement the changes on the DR site where they can be trialed and tested. This will increase the level of confidence prior to going live.

16.4.4 Examples of Using the BW Host Command

If a key is being migrated without change of LMK, where the key:

- Is a TPK
- Is a Triple-length DES key
- Is using the default LMK
- Has no message trailer

Field	Value
Message Header	1234
Command Code	BW
Key Type code	E2
Key length	2
Key	U12345678…ABCDEF
Delimiter	;
Key Type	70D

Note that when the Key Type Code is E2, no LMK must be installed in the Key Change Storage.

If the same key is being migrated but this time with a change of LMK:

Field	Value
Message Header	1234
Command Code	BW
Key Type code	F2
Key length	2
Key	U12345678…ABCDEF
Delimiter	;
Key Type	70D

Note that when Key Type Code is F2, both the old LMK (that is being moved away from) and the new LMK (that is being moved to) must be installed on the payShield 10K.

16.4.4.1 Thales Professional Services

Thales professional services can work with users in planning and implementing their key migration.

16.4.4.2 Summary of Actions Required

- Create a Host process to migrate keys from key type 002 (and, if appropriate, change LMK) using the BW Host command, and generate a new key database.
- At the appropriate time, set the HSM security setting Enforce key type 002 separation for PCI HSM compliance and start using the new key database.

16.5 Diebold Table Re-encryption

This change will only affect users who are employing the Diebold Table.

The Diebold Table in legacy systems is encrypted using LMK pair 14-15 variant 0 (key type 002); because of the PCI HSM key separation requirements this will be encrypted under LMK pair 36-37 variant 6 (key type 60D) when the HSM is required to be PCI HSM compliant. The encryption key type used is controlled by the security setting Enforce key type 002 separation for PCI HSM compliance.

Immediately after setting the security setting Enforce key type 002 separation for PCI HSM compliance on a payShield 10K, the following actions must be taken:

- Erase the existing encrypted version of the Diebold table using the LA Host command to overwrite it with hexadecimal F digits. The Index Address must be the same as that used when the encrypted Diebold table was set up; 32 blocks of 16 hexadecimal F digits should be written. (This step is unnecessary if the re-encrypted Diebold table is to be written to the same location in user storage as the existing encrypted table.)
- Use the R Console command to load the new Diebold Table. This requires access to the plaintext version of the Diebold Table
- Use the LC Host command to verify the Diebold Table
- Repeat the above steps for each LMK which has an encrypted Diebold Table

Note that if for any reason the security setting Enforce key type 002 separation for PCI HSM compliance is unset the Diebold table will have to be re-entered again.

16.5.1 Summary of Actions Required

- Re-enter the Diebold table whenever the value of the security setting Enforce key type 002 separation for PCI HSM compliance is changed.

Appendix A - Key Scheme Table

Whenever a key is entered into the HSM, it must be prefixed by a Key Scheme Tag which allows the HSM to interpret the key correctly. The following values are supported:

Key Scheme Tag	Notes
None/Z	Encryption of a single-length DES key using the ANSI X9.17 method. Used for encrypting keys for local use (under a Variant LMK) or for importing/exporting keys (e.g. under a ZMK). The use of this scheme requires some security settings to be changed.
U	Encryption of a double-length DES key using the Thales Variant method. Used for encrypting keys for local use (under a Variant LMK) or for importing/exporting keys (e.g. under a ZMK). The use of this scheme for import/export requires some security settings to be changed.
T	Encryption of a triple-length DES key using the Thales Variant method. Used for encrypting keys for local use (under a Variant LMK) or for importing/exporting keys (e.g. under a ZMK). The use of this scheme for import/export requires some security settings to be changed.
X	Encryption of a double-length DES key using the ANSI X9.17 method. Used for encrypting keys for local use (under a Variant LMK) or for importing/exporting keys (e.g. under a ZMK). The use of this scheme requires some security settings to be changed.
XI	Encryption of a double-length DES key in EBC mode using ISO 9797-1 Padding Mode 2. Used for encrypting keys for local use (under a Variant LMK) or for importing/exporting keys (e.g. under a ZMK). The use of this scheme requires some security settings to be changed. This format only supported in Host Command 'KI' (Derive Card Keys).
Y	Encryption of a triple-length DES key using the ANSI X9.17 method. Used for encrypting keys for local use (under a Variant LMK) or for importing/exporting keys (e.g. under a ZMK). The use of this scheme requires some security settings to be changed.
M	Encryption of a double-length DES key by a 3DES wrapping key using CBC Format. Used for importing keys (e.g. under a ZMK). The use of this scheme requires some security settings to be changed. This format only supported in Host Command 'A6' (Import Key).
O	Encryption of a triple-length DES key by a 3DES wrapping key using CBC Format. Used for importing keys (e.g. under a ZMK). The use of this scheme requires some security settings to be changed. This format only supported in Host Command 'A6' (Import Key).

<table>
<tr><th>Key
Scheme Tag</th><th>Notes</th></tr>
<tr><td>V</td><td>Encryption of double/triple-length DES keys using Verifone/GISKE methods.
Only used for exporting keys (e.g. under a ZMK).</td></tr>
<tr><td>R</td><td>Encryption of single/double/triple-length DES and AES keys using the X9 ANSI X.9.143 Key Block methods.
Only used for importing/exporting keys (e.g. under a ZMK).</td></tr>
<tr><td>S</td><td>Encryption of all DES, AES, RSA, ECC & HMAC keys using Thales Key Block methods.
Used for encrypting keys for local use (under a Key Block LMK) or for importing/ exporting keys (e.g. under a ZMK).</td></tr>
<tr><td>N</td><td>Encryption of an AES Key with an AES Key Encryption Key using AES Key Wrap (KW). The format of the data returned is as follows, with the Encrypted Key in KW format as defined in NIST SP800-38F.
<table>
<tr><td>Scheme (1 A)</td><td>Version (1 b)</td><td>Key Length (1 b)</td><td>Encrypted Key</td></tr>
<tr><td>‘N’</td><td>0x00000000</td><td>2 N</td><td>n B</td></tr>
</table>
This format only supported in Host Command ‘KI’ (Derive Card Keys).</td></tr>
<tr><td>PG</td><td>Encryption of a 128-bit AES key in EBC mode.
Used for importing/exporting keys (e.g. under a AES ZMK).
The use of this scheme requires some security settings to be changed.
This format only supported in Host Commands ‘A6’ (Import Key) and ‘KI’ (Derive Card Keys).</td></tr>
<tr><td>PI</td><td>Encryption of a 128-bit AES key in EBC mode using ISO 9797-1 Padding Mode 2.
Used for importing/exporting keys (e.g. under a AES ZMK).
The use of this scheme requires some security settings to be changed.
This format only supported in Host Command ‘KI’ (Derive Card Keys).</td></tr>
<tr><td>P</td><td>Encryption of a 128-bit AES key in CBC mode.
Used for importing/exporting keys (e.g. under a AES ZMK).
The use of this scheme requires some security settings to be changed.
This format only supported in Host Commands ‘A6’ (Import Key) and ‘A8’ (Export Key).</td></tr>
<tr><td>Q</td><td>Encryption of a 192-bit AES key in CBC mode.
Used for importing/exporting keys (e.g. under a AES ZMK).
The use of this scheme requires some security settings to be changed.
This format only supported in Host Command ‘A8’ (Export Key).</td></tr>
<tr><td>WG</td><td>Encryption of a 256-bit AES key in EBC mode.
Used for importing/exporting keys (e.g. under a AES ZMK).
The use of this scheme requires some security settings to be changed.
This format only supported in Host Command ‘A6’ (Import Key) and ‘KI’ (Derive Card Keys).</td></tr>
</table>

Key Scheme Tag	Notes
WI	Encryption of a 256-bit AES key in EBC mode using ISO 9797-1 Padding Mode 2. Used for importing/exporting keys (e.g. under a AES ZMK). The use of this scheme requires some security settings to be changed. This format only supported in Host Command 'KI' (Derive Card Keys).
W	Encryption of a 256-bit AES key in CBC mode. Used for importing/exporting keys (e.g. under a AES ZMK). The use of this scheme requires some security settings to be changed. This format only supported in Host Commands 'A6' (Import Key) and 'A8' (Export Key).

Appendix B - Reduced Character Sets

The HSM optionally (through the security setting "Enable ZEK/TEK encryption") restricts the character set that can be processed by the data encryption/ decryption commands. The three options are described in the table below:

"ZEK/TEK Encryption" Setting	Byte Value (hex)
ASCII data	0x20 – 0x7E
Binary data	0x00 – 0xFF
None	-

Appendix C - Print Formatting Symbols

Printer Formatting

The table shows the EBCDIC and ASCII codes for the printer formatting when printing from the payShield 10K:

Symbol	EBCDIC	ASCII	Meaning
>L	6E D3	3E 4C	Line feed, carriage return.
>V	6E E5	3E 56	Vertical tab.
>H	6E C8	3E 48	Horizontal tab.
>F	6E C6	3E 46	Form feed.
>nnn	6E Fn Fn Fn	3E 3n 3n 3n	Skip column nnn in relation to left margin, where nnn is a 3-digit decimal number.
^M	5F D6	5E 49	For a key document, print third clear component.
^P	5F D7	5E 50	For a PIN mailer, print clear PIN for mailer 1. For a key document, print clear component.
^Q	5F D8	5E 51	For a PIN mailer, print clear PIN for mailer 2. For a key document, print clear component or encrypted TMK (only one-up printing allowed for key documents).
^R	5F D9	5E 52	Print reference number for PIN mailer 1.
^S	5F E2	5E 53	Print reference number for PIN mailer 2.
^T	5F E3	5E 54	Print last 6 account number digits on PIN mailer 1.
^U	5F E4	5E 55	Print last 6 account number digits on PIN mailer 2.
|<L><hh hh hh ..>	6A <L> <hh hh hh ..>	7C <L> <hh hh hh ..>	Send binary data to printer for example printer control string. | character followed by the length of the string in bytes <L> 0 - F then the expanded hex string <hh hh hh ..>. NOTE: in the columns to the left, the "<" and ">" characters represent field boundaries, and are NOT part of the data to be sent to the printer. For example (using ASCII), a string of 4 bytes which would be represented in the 3rd column as 7C<4><1B283358> would actually be sent as |41B283358 .
^0	5F F0	5E 30	

Symbol	EBCDIC	ASCII	Meaning
^1	5F F1	5E 31	Insert Print Field 0. Insert Print Field 1. Insert Print Field 2. . . . Insert Print Field 15. Insert Print Field 16. Insert Print Field 17. Insert Print Field 18. . . Insert Print Field 31.
^2	5F F2	5E 32	
.	.	.	
.	.	.	
.	.	.	
^F	5F C6	5E 46	
^^10	5F 5F F1 F0	5E 5E 31 30	
^^11	5F 5F F1 F1	5E 5E 31 31	
^^12	5F 5F F1 F2	5E 5E 31 32	
.	.	.	
.	.	.	
^^1F	5F 5F F1 C6	5E 5E 31 46	

Printing PINs in Word Format

Two print formatting symbols are provided for printing PINs in word format. For example:

ONE TWO THREE FOUR.

English is used as the default setting. The symbols can be used in addition to the symbols for printing PINs in numeric format (e.g., 1234).

Symbol	EBCDIC	ASCII	Meaning
^V	5F E3	5E 56	Print the clear PIN in word format for mailer 1. Can be used for either a one-up or a two-up PIN mailer. e.g., ONE TWO THREE FOUR
^W	5F E6	5E 57	Print the clear PIN in word format for mailer 2. Can be used only for a two-up PIN mailer. e.g., ONE TWO THREE FOUR

Printing PINs in Columns

Four print formatting symbols are provided for printing PINs (both words and numeric) in columns. For example:

1	ONE
2	TWO
3	THREE
4	FOUR

For the following definition of print symbols an n is used to indicate which digit of a PIN is to be printed. The relationship between PIN digits and n is as follows:

PIN Digit	1	2	3	4	5	6	7	8	9	10	11	12
'n'	1	2	3	4	5	6	7	8	9	A	B	C

Symbol	EBCDIC	ASCII	Meaning
^Pn	5F D7 F1-F9 or C1-C3	5E 50 31-39 or 41-43	Print the clear PIN digit n in number format for mailer 1. Can be used for either a one-up or a two-up PIN mailer. e.g. 1
^Qn	5F D8 F1-F9 or C1-C3	5E 51 31-39 or 41-43	Print the clear PIN digit n in number format for mailer 2. Can be used only for two-up PIN mailer. e.g. 1
^Vn	5F E3 F1-F9 or C1-C3	5E 56 31-39 or 41-43	Print the clear PIN digit n in word format for mailer 1. Can be used for either a one-up or a two-up PIN mailer. e.g. ONE
^Wn	5F E6 F1-F9 or C1-C3	5E 57 31-39 or 41-43	Print the clear PIN digit n in word format for mailer 2. Can be used only for two-up PIN mailer. e.g. ONE

Appendix D - Example Laser Printer Formatting Control Codes

The following is an example of how an application can pass format control commands for a laser printer to a payShield 10K as print formatting symbols in PA and PC Host commands, as discussed in Chapter 3, PIN Printing and Solicitation.

This information, based on certain types of HP printer, should be viewed simply as an example to help developers generate an appropriate symbol sequence for their own printer.

The example symbol sequence is as follows, and is explained in the subsequent table.

```
>L
>L|91B2661393030763048|91B2666313030793358|51B28313055|B1B28733170333676323554|
D1B266136373930763133343048
>005^P
>L|A1B287330703130763354|91B2661393030763048
>005^^00
>L
>005^^01
>L
>005^^02
>L
>005^^03
>L
>005^^04
>L
>005^^05
>L
>005^^06
>L
>F
```

Symbol	Interpretation	Notes
>L	Line feed, carriage return	
>L	Line feed, carriage return	
|91B2661393030763048	Send the following 9 bytes of binary data to the printer: ".&a900v0H"	Position the print head 900 decipoints vertically, 0 decipoints horizontally.
|91B2666313030793358	Send the following 9 bytes of binary data to the printer: ".&f100y3X"	Select overlay

Symbol	Interpretation	Notes
|51B28313055	Send the following 5 bytes of binary data to the printer: ".(10U"	Select font ID=10U (PC-8)
|B1B28733170333676323554	Send the following 11 bytes of binary data to the printer: ".(s1p36v25T"	Select font Type Face
|D1B266136373930763133343048	Send the following 13 bytes of binary data to the printer: ".&a6790v1340H"	Position the print head 6790 decipoints vertically, 1340 decipoints horizontally.
>005^P	Skip to col. 5 and print the PIN	
>L	Line feed, carriage return	
|A1B287330703130763354	Send the following 10 bytes of binary data to the printer: ".(s0p10v3T"	Select font Type Face
|91B2661393030763048	Send the following 9 bytes of binary data to the printer: ".&a900v0H"	Position the print head 900 decipoints vertically, 0 decipoints horizontally.
>005^^00	Skip to col. 5 and print field 0	
>L	Line feed, carriage return	
>005^^01	Skip to col. 5 and print field 1	
>L	Line feed, carriage return	
>005^^02	Skip to col. 5 and print field 2	
>L	Line feed, carriage return	
>005^^03	Skip to col. 5 and print field 3	
>L	Line feed, carriage return	
>005^^04	Skip to col. 5 and print field 4	
>L	Line feed, carriage return	
>005^^05	Skip to col. 5 and print field 5	
>L	Line feed, carriage return	
>005^^06	Skip to col. 5 and print field 6	
>L	Line feed, carriage return	
>F	Form feed	

Appendix E - List of Authorized Activities

The following table indicates, for each sensitive host or console command, which activity needs to be authorized.Note that the authorization requirements may differ depending on whether a Variant or Key Block LMK is used.

Examples:

- Authorizing the activity export.001.host allows export of ZPKs using a host command (e.g. A8).
- Authorizing the activity export allows export of any (valid) key using a host or console command.
- Authorizing the activity export..console allows export of any (valid) key using a console command.

Note: When using a Variant LMK, the key type table determines whether the HSM needs to be authorized in order to generate, import or export a certain key. Where the key type table entry indicates 'U' (unconditional), it is not necessary to authorize the HSM for that activity. Moreover, authorized activities `genprint.*` and `component.*` do not examine the KTT and are always required.

The following table defines a number of 'groups' which, in turn, contain a set of either key types or key usages. These groups are subsequently referenced in the tables further below.

	Group	Contents
Variant LMK	*generate key types*	`000 200 001 002 402 003 006 008 009 109 209 309 409 509 709 00a 00b 70d 80d 90d`
	genprint key types	`000 200 001 002 402 003 006 008 009 109 209 309 409 509 709 00a 00b 70d 80d 90d`
	component key types	`000 200 001 002 402 003 006 008 009 109 209 309 409 509 709 00a 00b 70d 80d 90d`
	export key types	`200 001 002 402 003 006 008 009 109 209 309 409 509 709 00a 70d 80d 90d`
Key Block LMK	*genprint key usages*	`01 B0 C0 11 12 13 D0 21 22 E0 E1 E2 E3 E4 E5 31 32 K0 51 52 M0 M1 M3 P0 71 72 73 V0 V1 V2`
	component key usages	`01 B0 C0 11 12 13 D0 21 22 E0 E1 E2 E3 E4 E5 31 32 K0 51 52 M0 M1 M3 P0 71 72 73 V0 V1 V2`
	export key usages	`01 B0 C0 11 12 13 D0 21 22 E0 E1 E2 E3 E4 E5 31 32 K0 51 52 M0 M1 M3 61 62 63 64 65 P0 71 72 73 V0 V1 V2`
	import key usages	`01 B0 C0 11 12 13 D0 21 22 E0 E1 E2 E3 E4 E5 31 32 K0 51 52 M0 M1 M3 61 62 63 64 65 P0 71 72 73 V0 V1 V2`

	Command (H=Host, C=Console)	Description	Category	Sub-Category	Inter-face
SYMMETRIC KEY GENERATION					
Variant LMK	H - A0	**Generate** Key	generate	000	host
	C - KG	**Generate** Key			console
	C - GZ	**Generate** Zone Master Key and Write to Smartcards			
	H - BU	**Generate** Key Check Value (for Key Check Values > 6 digits)		*generate key types*	host
	C - CK	**Generate** Key Check Value (for Key Check Values > 6 digits)			console
	H - FG	**Generate** a Pair of PVKs		002	host
SYMMETRIC KEY GENERATION / PRINTING					
Key Block LMK	H - A2	**Generate** and **Print** Component	genprint	*genprint key usages*	host
	H - NE	**Generate** and **Print** Key as Split Components			
Variant LMK	H - A2	**Generate** and **Print** Component	genprint	*genprint key types*	host
	H - NE	**Generate** and **Print** Key as Split Components			
	H - OC	**Generate** and **Print** ZMK component		000	
	H - OE	**Generate** and **Print** TMK, TPK or PVK		002	
SYMMETRIC KEY COMPONENT MANIPULATION					
Key Block LMK	H - A4	Form a Key from Encrypted **Components**	component	*component key usages*	host
	C - EC	Encrypt Clear **Component**			console
	C - FK	Form Key from **Components**			
	C - GS	Generate Key **Components** and Write to a Smartcard			
	C - GC	Generate Key **Component**			
Variant LMK	H - GG	Form ZMK from Three ZMK **Components**	component	000	host
	H - GY	Form ZMK from 2 to 9 ZMK **Components**			
	H - A4	Form Key from Encrypted **Components**			
	C - BK	Form Key from **Components**		*component key types*	console
	C - EC	Encrypt Clear **Component**			

	C – FK	Form Key from **Components**			
	C – GS	Generate Key **Components** and Write to a Smartcard			
	C – GC	Generate Key **Component**			
	C – DE	Form ZMK from Clear **Components**		000	
	C – D	Form ZMK from Encrypted **Components**			
	C – Z	Encrypt clear ZMK **Component**			
SYMMETRIC KEY IMPORT					
Key Block LMK	H – BY	Translate (**Import)** ZMK from ZMK to LMK Encryption (from non-key block format)	import	K0 52	host
	H – A6	**Import** Key (from non-key block format)		*import key usages*	host
	H – LU	**Import** HMAC Key (from non-key block format)		10C	host
Variant LMK	H – FC	Translate (**Import**) TMK, TPK or PVK from ZMK to LMK Encryption	import	002	Host
	H – BY	Translate (**Import)** ZMK from ZMK to LMK Encryption		000	
SYMMETRIC KEY EXPORT					
Key Block LMK	H – A0	Generate Key (when requested to **export** generated key to non-key block format)	export	*export key usages*	host
	H – A8	Export Key (to non-key block format)			
	H – LW	**Export** HMAC Key (to non-key block format)		61 62 63 64 65	
	C – KG	Generate Key (when requested to **export** generated key to non-key block format)		*export key usages*	console
	C – KE	**Export** Key (to non-key block format)			
	H – IG	**Derive** Key (Initiator)	ekai	Derived key uages	host
	H – IG	**Derive** Key (Recipient)	ekar		
Variant LMK	H – A0	Generate Key (when requested to **export** generated key)	export	*export key types*	host
	H – A8	Export Key			
	H – FE	Translate (**Export**) TMK, TPK or PVK from LMK to ZMK Encryption		002	
	H – LW	**Export** HMAC Key		10c	
	C – KG	Generate Key (when requested to **export** generated key)		*export key types*	console
	C – KE	**Export** Key			

	C – WK	Translate (**Export**) Zone PIN Key		001	
ASYMMETRIC KEY MANAGEMENT					
Key Block LMK	H – EI	**Generate** a Public/Private Key Pair	generate	rsa	host
	H – FY	**Generate** ECC Key Pair		ecc	
	H – EO	Import a **Public Key**	import	02	
Variant LMK	H – EI	**Generate** a Public/Private Key Pair	generate	rsa	host
	H – EO	**Import** a Public Key	import	rsa	
CLEAR PIN					
	H – BA	Encrypt a Clear **PIN**	pin	clear	host
	H – NG	Decrypt an Encrypted **PIN**			
PIN MAILER					
	H – PE	Print **PIN**/**PIN** Solicitation Data	pin	mailer	host
	H – OA	Print a **PIN** Solicitation Mailer			
AUDIT					
	H – Q6	Delete Audit Record	audit		host
	C – CLEARAUDIT	Clear the Audit Log			console
	C – AUDITOPTIONS	Audit Options			
	C – A5	Configure Fraud Detection			
	C – A7	Re-enable PIN Verification			
ADMINISTRATION					
	C – DM	Delete LMK	admin		console
	C – SS	Save HSM Settings to a Smartcard			
	C – RS	Retrieve HSM Settings from a Smartcard			
	C – LO	Load 'Old' LMK into Key Change Storage			
	C – LN	Load 'New' LMK into Key Change Storage			
	C – SETTIME	Set the Time and Date			
DIAGNOSTICS					
	H – KQ	ARQC (or TC/AAC) Verification and/or ARPC Generation	diag		host
	H – K2	Verify Truncated Application Cryptogram (CAP)			

H – KW	ARQC (or TC/AAC) Verification and/or ARPC Generation (EMV4.1 including CCD)			
H – KS	Data Authentication Code and Dynamic Number Verification			
H – PM	Verify a Dynamic CVV			
C – HEALTHSTATS	Display Health Check counters. (*Authorization required only to Reset the counters.*)			console
MISCELLANEOUS				
H – B0	Translate Key Scheme	misc		host
H – CS	Modify Key Block Header			
C – R	Load the Diebold Table			console
C – CV	Generate a VISA Card Verification Value			
C – PV	Generate a VISA PIN Verification Value			
C – TD	Translate Decimalisation Table			
C – ED	Encrypt Decimalisation Table			
C – MI	Generate a MAC on an IPB			
COMMAND				
H – GI	Import DES Key (Auth required if backward compatibity mode is enabled by console CS)	command	gi	host
H – TA	Print TMK Mailer		ta	
C – IK	Import Key		ik	

Appendix F - Printer Settings

In serial, parallel or USB mode, some settings must be the same on both the HSM and the printer, in each respective mode.

For Serial Mode:

The printer's interface mode should be in *serial* or *auto* mode and the printer port settings should be the same on both the HSM and the printer.

For additional information on configuring the printers, follow this link:

- Section 1.2, Printer Requirements

or refer to the following manuals:

- payShield 10K Installation and User Guide, Section 9.10.3.5 or the
- payShield 10K Console Manual (CP command).

For Parallel Mode:

The printer's Interface mode should be in *parallel* or *auto* mode and following setting should be *off* on the printer:

- Parallel interface bidirectional mode

The printer's interface mode should be in *USB* or *auto* mode.

Appendix G – SNMP Security Setting

A full description of the Security Settings retrieved using SNMP is provided in the *payShield 10K Security Manual*, Section 3.3.1, Security Parameter Descriptions.

Below are the reference numbers used for the Security Settings when these are retrieved using SNMP.

Serial Number	Security Setting	OID
1	PIN length: 04	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.1
2	Encrypted PIN length: 05	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.2
3	Echo: OFF	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.3
4	Atalla ZMK variant support: ON	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.4
5	Transaction key support: AUSTRALIAN	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.5
6	User storage key length: SINGLE	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.6
7	Display general information on payShield Manager Landing Page: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.7
8	Default LMK identifier: 00	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.8
9	Management LMK identifier: 00	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.9
10	Enforce Atalla variant match to Thales key type: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.10
11	Select clear PINs: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.11
12	Enable ZMK translate command: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.12
13	Enable X9.17 for import: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.13
14	Enable X9.17 for export: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.14
15	Solicitation batch size: 1024	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.15
16	ZMK length: DOUBLE	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.16
17	Decimalization tables: ENCRYPTED	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.17
18	Decimalization table checks: ENABLED	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.18
19	PIN encryption algorithm: A	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.19
20	Use deprecated proprietary format (Tag J) when using PIN Blocks under AES Key Block LMK: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.20
21	Authorized State required when importing a key under an RSA key: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.21
22	Minimum HMAC length in bytes: 10	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.22

Serial Number	Security Setting	OID
23	Enable PKCS#11 import and export for HMAC keys: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.23
24	Enable ANSI X9.17 import and export for HMAC keys: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.24
25	Enable ZEK/TEK encryption of ASCII data or Binary data or None: BINARY	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.25
26	Restrict key check values to 6 hex chars: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.26
27	Return PIN Length in PIN Translation Response: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.27
28	Enable multiple authorized activities: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.28
29	Allow persistent authorized activities: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.29
30	Enable variable length PIN offset: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.30
31	Enable weak PIN checking: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.31
32	Check new PINs using global list of weak PINs: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.32
33	Check new PINs using local list of weak PINs: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.33
34	Check new PINs using rules: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.34
35	Enable PIN block Format 34 as output format for PIN translations to ZPK: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.35
36	Enable translation of account number for LMK encrypted PINs: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.36
37	Use HSM clock for date/time validation: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.37
38	Additional padding to disguise key length: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.38
39	Key export and import in trusted format only: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.39
40	Protect MULTOS cipher data checksums: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.40
41	Enable Key Scheme Tag 'X' (X9.17) for storing keys under LMK: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.41
42	Enable use of Tokens in PIN Translation: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.42
43	Enable use of Tokens in PIN Verification: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.43

Serial Number	Security Setting	OID
44	Enable PIN Translation to BDK Encryption: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.44
45	Ensure LMK Identifier in command corresponds with host port: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.45
46	Ignore LMK ID in Key Block Header: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.46
47	Enable import and export of RSA Private keys: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.47
48	Enable import of a ZMK: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.48
49	Enable export of a ZMK: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.49
50	Prevent single-DES keys masquerading as double or triple-length keys: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.50
51	Single-DES: ENABLED	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.51
52	Card/password authorization (local): C	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.52
53	Restrict PIN block usage for PCI HSM Compliance: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.53
54	Enforce key type 002 separation for PCI HSM compliance: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.54
55	Enforce Authorization Time Limit: YES	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.55
56	Enforce Multiple Key Components: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.56
57	Enforce PCI HSMv3 Key Equivalence: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.57
58	Enforce minimum key strength of 1024-bits for RSA signature verification: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.58
59	Enforce minimum key strength of 2048-bits for RSA: NO	.1.3.6.1.4.1.4096.2.2.10000.15.7.1.2.59

Technical Support Contacts

Our team of knowledgeable and friendly support staff are available to help. If your product is under warranty or you hold a support contract with Thales, do not hesitate to contact us using the link below. For more information, consult our standard Terms and Conditions for Warranty and Support.

https://supportportal.thalesgroup.com/csm

THALES

Contact us

For all office locations and contact
information, please visit
cpl.thalesgroup.com/contact-us

> cpl.thalesgroup.com <