ООО «КРИПТО-ПРО»

ЖТЯИ.00096-03 92 04

ПАКМ «КриптоПро HSM»

Версия 2.0 R3

Модуль обеспечения защиты платежных систем

Руководство программиста

2025 г.

ПАКМ «КриптоПро HSM» версия 2.0 R3
Модуль обеспечения защиты платежных систем
Версия ПО: 288111

Оглавление

Введение

Модуль обеспечения защиты платёжных систем (также Платёжный HSM) предназначен для обеспечения защиты чувствительной информации, обрабатываемой в платёжной системе, и безопасного хранения криптографических ключей в соответствии с российскими и зарубежными стандартами. Модуль обеспечения защиты платёжных систем выполняет криптографические операции в физически защищенной аппаратной среде ПАКМ «КриптоПро HSM» версия 3.0 (далее – HSM).

Платёжный HSM реализует 2 интерфейса: команд консоли и команд хоста.

Интерфейс команд консоли предназначен для взаимодействия HSM с консолью управления HSM. Консольные команды используются для конфигурации HSM и базовых операций с закрытыми ключами (в том числе хранящимися в виде разделенных компонент на смарт-картах).

Интерфейс команд хоста используется для взаимодействия HSM с хост-системой. Для реализации взаимодействия хост-компьютер (прикладная программа/приложение) отправляет хостовые команды на HSM и получает ответы от HSM по каналу передачи данных. Хостовые команды реализуют операции по управлению ключами, PIN, MAC и др.

Настоящее руководство содержит описание реализации и форматы криптографических объектов и правила взаимодействия с ними для разработчиков хостовых приложений.

Каждая команда и ответ состоят из переменного количества полей, подробнее о формате команд см. в гл. 2.

Перечень и описание поддерживаемых команд хоста приведены в документе «КриптоПро HSM. Команды хоста». Консольные команды описаны в документе «КриптоПро HSM. Команды консоли».

HSM поддерживает печать данных (например, PIN-конвертов и компонент ключей) на подключенном к HSM принтере. Для отправки данных на печать HSM необходима информация о формате этих данных, поэтому перед началом печати требуется отправить HSM команду установки формата данных для печати (команда 'PA'), затем отправить команду печати (например, 'A2'). Подробнее см. в гл. 3.

Глава 1

Подключение к HSM

Взаимодействие удаленного хоста и HSM осуществляется по протоколу TCP. IP-адрес сетевого интерфейса HSM настраивается вручную в соответствии с «КриптоПро HSM. Инструкция по использованию».

Для подключения к HSM и возможности передачи команд хоста необходимо запустить сервис хоста HSM (пункт **Enable Host** меню Платежной системы LCD панели HSM, подробнее см. п. 5.4 «КриптоПро HSM. Инструкция по использованию»).

HSM поддерживает передачу команд хоста по защищенному с использованием протокола TLS каналу. Для этого необходимо выполнить настройку и запустить приложение для создания TLS-туннеля stunnel, входящее в комплект поставки HSM. Настройка stunnel выполняется аналогично п. 7.1 «КриптоПро HSM. Инструкция по использованию»с указанием соответствующих портов (см. ниже).

По умолчанию для приема команд хоста используется порт 1500.

С целью обеспечения возможности одновременной работы с несколькими установленными LMK в HSM используются порты 1511-1520, которые автоматически открываются одновременно с портом 1500 при запуске сервиса хоста.

Номер идентификатора LMK, который необходимо использовать в команде, определяется по номеру порта, на который поступила команда, в соответствии со следующим правилом: $LMK_{id} = N_{port} - 1511$.

Подробнее об использовании нескольких LMK см. в разделе гл. 7.

Глава 2

Формат команд и представление данных

2.1 Представление данных

При отправке данных на HSM необходимо кодировать каждую шестнадцатеричную цифру (0-9, A-F) как символ (за исключением данных, которые уже представлены в символьном формате). Например, для отправки шестнадцатеричного значения 0x1234ABCD требуется 8 символов. В некоторых случаях HSM принимает определённые поля представленными в двоичном формате. Полная информация о формате полей команд хоста указана в документе «КриптоПро HSM. Команды хоста».

Таблица символов ASCII

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

2.2 Формат команды

Прикладная программа хоста отправляет команды, содержащие все необходимые данные, на HSM в виде последовательности символов.

Каждая команда состоит из следующих полей:

- **Заголовок команды**

Длина поля — 4 символа. Заголовок может содержать любые печатные символы (SSSS, HEAD, HDR1 и т.п.), которые HSM возвращает без изменений в ответном сообщении хосту. Заголовок используется для выделения команд и их ответов в системах, реализующих пакетные очереди или многопоточные команды.

- **Код команды**

Уникальный двухсимвольный код. Полный список команд с соответствующими кодами приведён в документе «КриптоПро HSM. Команды хоста».

- **Данные и параметры**

Большинство команд содержат обязательные и опциональные поля с различными данными и параметрами, необходимыми для обработки команды. Для каждой команды формат и набор параметров уникален. Полный перечень допустимых полей для соответствующей команды приведён в документе «КриптоПро HSM. Команды хоста».

- **Трейлер**

Опциональное поле переменной длины (максимальная длина — 32 символа) для передачи дополнительных данных, требуемых хосту. Если в команде передается трейлер, перед началом поля должен присутствовать управляющий символ EM (в кодировке ASCII значение 0x19) — признак конца полей данных команды.

Трейлер команды может содержать любые печатные символы, которые возвращаются в ответе без изменений в случаях выполнения команды без ошибок (код ошибки '00') или возврата кода ошибки, относящегося к типу "предупреждение" (как указано в описании поля *Код ошибки* ответа). Для других значений кода ошибки трейлер в ответе не возвращается.

Параметр	Формат	Описание
КОМАНДА		
Заголовок команды	m A	Должен быть возвращен хосту без изменений.
Код команды	2 A	Значение 'A0'.
Параметр 1	...	...
...	...	...
Параметр n	...	...
Символ конца команды	1 C	Значение 0x19. Опционально. Должен присутствовать, если в команде передается трейлер.
Трейлер	n A	Опционально. Максимальная длина — 32 символа.

Команды хоста передаются на HSM в бинарном формате, при этом перед командой необходимо указывать её длину:

ДЛИНА КОМАНДЫ	КОМАНДА ХОСТА
2 байта	n байтов

В рамках одного TCP-пакета на HSM могут быть отправлены несколько команд. Каждая команда должна быть представлена в указанном выше формате.

Пример для команды 'NC' — *Выполнение диагностики HSM* (без опциональных параметров):

0x00	0x06	0x31	0x32	0x33	0x34	0x4E	0x43
длина команды (в байтах) = 6		заголовок команды = 1234 (в HSM установлена длина заголовка = 4)				'N'	'C'

2.3 Формат ответа

По результатам обработки команды HSM отправляет хосту ответ, содержащий всю необходимую информацию и результаты выполнения команды.

Каждый ответ состоит из следующих полей:

- **Заголовок ответа**

Поле является копией заголовка, полученного в команде от хоста. Данные заголовка возвращаются в ответе хосту без изменений. Заголовок используется для выделения команд и их ответов в системах, реализующих пакетные очереди или многопоточные команды.

- **Код ответа**

Уникальный двухсимвольный код. Как правило, первый символ кода ответа равен первому символу кода команды, второй символ кода ответа на единицу больше второго символа кода команды (в случае использования букв латинского алфавита — следующая по алфавиту буква). Например, если код команды 'AA' — код ответа 'AB'. Полный список команд и ответов на них с соответствующими кодами приведён в документе «КриптоПро HSM. Команды хоста». При разработке хостовых приложений необходимо убедиться, что каждый код ошибки команды обрабатывается приложением корректно.

- **Код ошибки**

Двухсимвольный буквенно-цифровой код, используется для сообщения об ошибках, обнаруженных HSM во время обработки команды. Значение '00' означает, что команда выполнена без ошибок. Если возвращается код ошибки, отличный от '00', HSM не возвращает последующие поля ответа, кроме символа конца текста (ETX). Список специфических кодов ошибок, соответствующих определенным командам, а также список стандартных кодов ошибок приведены в документе «КриптоПро HSM. Команды хоста».

- **Данные**

Данные, полученные в результате обработки команды. Полный перечень возвращаемых данных для соответствующего ответа приведён в документе «КриптоПро HSM. Команды хоста». Как правило, в случае ошибки обработки команды (*Код ошибки* $\neq$ '00'), результат обработки команды не возвращается в ответе. Исключение составляют некоторые коды ошибок, отмеченные как "предупреждение" — например, команда импорта ключа ('A6') возвращает код ошибки '01' в случае, если нарушена четность импортируемого ключа, при этом в ответе содержатся все необходимые поля с данными.

- **Трейлер**

Трейлер возвращается в ответе без изменений, только если он присутствовал в команде, и в случаях выполнения команды без ошибок (код ошибки '00') или возврата кода ошибки, относящегося к типу "предупреждение" (как указано в описании поля *Код ошибки* ответа). Для других значений кода ошибки трейлер в ответе не возвращается.

Поле является копией трейлера, полученного в команде от хоста. Если в команде передается трейлер, в конце ответа должен присутствовать символ EM (в кодировке ASCII значение 0x19) — признак конца полей данных ответа.

Параметр	Формат	Описание
ОТВЕТ		
Заголовок ответа	m A	Заголовок команды, возвращаемый хосту без изменений.
Код ответа	2 A	Значение 'A1'.
Код ошибки	2 H	'00': Без ошибок '68': Команда недоступна ... или другой стандартный код ошибки.
Поле данных 1	...	...
...	...	...
Поле данных n	...	...
Символ конца ответа	1 C	Значение 0x19. Присутствует, только если присутствует в команде.
Трейлер	n A	Присутствует, только если присутствует в команде. Максимальная длина — 32 символа.

HSM возвращает хосту ответ в бинарном формате, при этом перед ответом указывается его длина:

ДЛИНА ОТВЕТА	ОТВЕТ HSM
2 байта	n байтов

Ответ на каждую команду, отправленную HSM, возвращается хосту отдельно, в том числе и в случае, когда несколько команд были направлены HSM в одном TCP-пакете.

Пример ответа HSM на команду 'NC' (пример команды см. выше):

0x00	0x21	0x31	0x32	0x33	0x34	0x4E	0x44	0x30	0x30	0x32	0x36
длина ответа (в байтах) = 33		заголовок команды = 1234 (длина заголовка = 4)				'N'	'D'	Код ошибки		KCV_{LMK} = 26..	

0x38	0x36	0x30	0x34	0x37	0x34	0x34	0x34	0x39	0x31	0x32	0x34
..26860474449124..											

0x32	0x32	0x58	0x58	0x58	0x58	0x58	0x58	0x58	0x58	0x58	
..22		Версия прошивки = XXXXXXXXX									

Глава 3

Печать PIN-конвертов и запросов PIN

HSM поддерживает функцию печати на подключенном к HSM принтере следующих данных:

• PIN-конвертов (PIN mailer) при выпуске новых карт;

• запросов о присвоении PIN (PIN Solicitation mailer) с предложением клиенту выбрать желаемый PIN-код.

3.1 PIN-конверты

При выпуске новой карты или запросе смены PIN информационные системы эмитента генерируют новый PIN-код, который печатается на специальном бумажном конверте, защищённом от несанкционированного доступа, через который PIN-код не может быть просмотрен. PIN-конверт передается держателю карты отдельно от самой карты.

HSM используется таким образом, чтобы PIN-код не был доступен в открытом виде. PIN-код находится в открытом виде только на этапе передачи по кабелю между HSM и подключенном у нему принтером.

Для печати PIN-конвертов необходимо:

1. настроить формат печати PIN-конверта в HSM;
2. сгенерировать зашифрованный PIN-код;
3. отправить данные, которые необходимо напечатать в PIN-конверте, на HSM;
4. осуществить проверку корректности выполнения команды печати.

3.1.1 Настройка формата печати PIN-конверта

PIN-конверты распечатываются на принтере, подключенном к HSM, в ответ на команду от хост-системы.

Перед началом печати в HSM должен быть загружен формат печати PIN-конверта для дальнейшего форматирования вывода. При вызове разных команд печати требуются различные форматы печати, однако одновременно HSM может хранить только один формат печати. Поэтому формат печати PIN-конверта должен загружаться на HSM перед каждым вызовом команды печати PIN-конверта.

Формат печати загружается в HSM в виде текстовой строки, которая состоит из:

• констант;

• служебных символов для форматирования вывода;

• маркеров полей печати, которые заполняются данными при печати PIN-конвертов.

Максимальная длина определения формата печати — 299 символов и констант.

Формат печати хранится в HSM до тех пор, пока не будет отключено питание или выполнен сброс.

Формат печати позволяет печатать цифры PIN словами (используя символы ^V и ^W).

Для загрузки форматов печати на HSM используются следующие команды хоста:

- PA — Загрузка данных форматирования в HSM
- LI — Переопределение текстовых значений для цифр PIN (при необходимости печати цифр PIN-кода словами на языке, отличном от английского)

Существует 2 типа форматов PIN-конвертов: 1 документ на листе (PIN-конверты печатаются друг за другом), 2 документа на листе (PIN-конверты печатаются парами рядом друг с другом).

Пример команды для установки формата печати PIN-конвертов **"2 документа на листе"**:

Команда:

```
PA>L>003^0>033^3>L>003^1>023^P>033^4>053^Q>L>003^2>033^5>L>F
```

Ответ:

```
PB00
```

Результат форматирования:

1	IVANOV IVAN	SIDOROV SERGEY
2	MOSCOW MAROSEYKA STREET 37 1782	MOSCOW LOMONOSOV STREET 25 3690
3	****230056	****103351

Пример команды для установки формата печати PIN-конвертов **"1 документ на листе"**:

Команда:

```
PA>L>013^0>L>013^1>041^P>L>013^2>L>F>
```

Ответ:

```
PB00
```

Результат форматирования:

1	IVANOV IVAN
2	MOSCOW MAROSEYKA STREET 37 1782
3	****230056
1	SIDOROV SERGEY
2	MOSCOW LOMONOSOV STREET 25 3690
3	****103351

Полный перечень символов, используемых для установки формата печати в HSM, и их значения приведены в Приложении А.

3.1.2 Генерация зашифрованного PIN

PIN-код должен генерироваться во время подготовки данных карты при выпуске карты. PIN-код должен быть представлен в зашифрованном виде при любом обращении человека или приложения.

В HSM реализованы следующие команды хоста для работы с PIN-кодом:

- JA — Генерация случайного PIN
- EE — Выработка PIN с использованием метода IBM 3624

PIN-коды хранятся в хост-системе в зашифрованном под LMK виде и передаются на HSM для последующей печати также в зашифрованном виде. PIN никогда не доступен хосту в незашифрованном виде.

3.1.3 Отправка данных для печати PIN-конверта

Для печати PIN-конвертов используется команда хоста PE, её параметры включают:

- тип документа (1 документ на листе или 2 документа на листе);
- PAN;
- PIN, зашифрованный под LMK;
- поля печати, значения которых определяются форматом печати.

HSM расшифровывает PIN и передает его на печать в заданном формате. Команда PA должна вызываться перед каждым обращением к принтеру, за исключением печати второго документа на одном листе.

3.1.4 PIN, созданные вне HSM

PIN-коды в зашифрованном виде могут быть созданы и храниться вне хост-системы и HSM. В этом случае они, как правило, представлены в виде PIN-блока, зашифрованного под ZPK.

Для получения PIN в формате PIN-блока, зашифрованного под LMK (как этого требует команда печати PE) используется следующая команда:

- JE — Трансляция PIN (из-под ZPK под LMK)

Команда расшифровывает PIN, зашифрованный под ZPK, и зашифровывает под LMK.

3.1.5 Проверка корректности выполнения команды печати PIN

Хост не «видит» данные, которые HSM отправляет на принтер, однако он должен убедиться в корректности печати данных PIN-конверта.

Для контроля корректности криптографических вычислений, выполненных в процессе печати по команде PE, HSM отправляет хост-системе ответ, содержащий проверочное значение (check value) PIN, вычисленное с использованием LMK.

Приложение хоста должно проверить полученное проверочное значение и подтвердить корректность выполнения команды печати PE, отправив на HSM команду PG.

Команда содержит те же значения PAN и зашифрованного PIN, что и команда PE, а также проверочное значение PIN, отправленное HSM в ответе на команду PE.

Команду PG рекомендуется отправлять на другой HSM, отличный от выполнявшего команду печати, но использующий тот же LMK.

Ответ PH от HSM на команду PG содержит информацию о соответствии или несоответствии значения зашифрованного PIN проверочному значению PIN.

3.2 Запрос PIN

Запрос PIN может быть выполнен в следующих ситуациях:

• при выпуске новой карты владельцу карты предлагается выбрать желаемый PIN-код и сообщить его эмитенту карты;

• владелец карты изменяет PIN-код на желаемый для ранее выпущенной карты.

Выбранный пользователем PIN-код может быть возвращен эмитенту различными способами в зависимости от архитектуры приложений эмитента карты:

• ответ на запрос PIN;

• через банкомат;

• по сети Интернет;

• по телефону.

После возврата PIN-кода он ассоциируется с данными карты в приложении эмитента карты.

Для обеспечения безопасности запрос PIN и возвращаемые данные не должны содержать номер карты (PAN). Вместо PAN используется ссылочный номер (reference number), который является результатом криптографического преобразования 10 последних цифр номера счета (за исключением контрольной цифры). Ссылочный номер представляет собой 10-значное число, за которым следуют 2 контрольные цифры.

Знание PIN-кода и ссылочного номера не представляет ценности для нарушителя. Соответствие PAN и ссылочного номера устанавливается только в HSM, при этом PAN и PIN никогда не передаются вместе.

Ссылочный номер является единственной ссылкой на PIN-код владельца карты, который вводится вручную. Поэтому необходимо обеспечивать корректность ввода PIN-кода, например, с помощью двойного ввода или визуального сравнения значений.

Корректность ссылочного номера, в отличие от PIN, может быть проверена хост-программой (подробнее см. в разд. 3.2.1). Контрольные цифры могут быть верифицированы во время или после ввода данных.

Данные запросов PIN обрабатываются пакетно с помощью команд хоста. Количество записей в пакете должно быть больше или равно минимальному размеру пакета (устанавливается консольной командой CS в настройке Solicitation batch size). Каждая запись состоит из 12-значного ссылочного номера и выбранного пользователем PIN-кода.

После загрузки пакета запросов PIN HSM зашифровывает PIN-коды под LMK 02-03 и расшифровывает ссылочные номера, получая значение 10 крайних правых цифр номера карты (за исключением контрольной цифры). Зашифрованный PIN-код и 10 цифр номера счета возвращаются хосту. Хост может сохранить зашифрованный PIN (в соответствии с полученной частью номера счета) для последующей обработки (в целях проверки или создания PIN Offset и т.д.).

3.2.1 Алгоритм проверки ссылочного номера

Ссылочный номер (reference number) дополняется двумя контрольными цифрами.

Расчёт первой контрольной цифры ссылочного номера $a_1a_2a_3...a_9a_{10}$:

• выбираются восемь цифр ссылочного номера с 3-ей по 10-ую: $a_3a_4...a_9a_{10}$;

• для каждой цифры a_i вычисляется вес p_i по таблице:

Номер цифры (i)	**Вес p_i**
3	9
4	7
5	8
6	6
7	7

8	9
9	6
10	8

- первая контрольная цифра равна сумме (mod 10): $(-1) * (a_3 * p_3 + a_4 * p_4 + ... + a_9 * p_9 + a_{10} * p_{10})$.

Расчёт второй контрольной цифры ссылочного номера $a_1a_2a_3...a_9a_{10}$:

- выбираются 10 цифр ссылочного номера: $a_1a_2...a_9a_{10}$;
- вычисляется первая контрольная цифра;
- используется следующая перестановка цифр f:

Цифра	**f(цифра)**
0	0
1	2
2	4
3	6
4	8
5	1
6	3
7	5
8	7
9	9

- вторая контрольная цифра равна сумме (mod 10):

$(-1) * (f(a_1) + a_2 + f(a_3) + a_4 + ... + f(a_9) + a_{10} + f(\textit{первая контрольная цифра}))$.

Пример 1:

ссылочный номер										контрольные цифры	
3	9	4	2	3	5	4	9	9	8	9	1

9 = (-1)*(4*9+2*7+3*8+5*6+4*7+9*9+9*6+8*8) (mod 10) = (-1)*(36+14+24+30+28+81+54+64) (mod 10) = (-1)*331 (mod 10)

1 = (-1)*(6+9+8+2+6+5+8+9+9+8+9) (mod 10)

Пример 2:

ссылочный номер										контрольные цифры	
5	8	0	1	5	1	4	7	1	4	8	0

8 = (-1)*(0+7+40+6+28+63+6+32) (mod 10)

0 = (-1)*(1+8+0+1+1+1+8+7+2+4+7) (mod 10)

3.2.2 Порядок запроса PIN

Для обработки запроса PIN необходимо:

1. настроить формат печати запроса PIN;
2. сгенерировать зашифрованный PIN (опционально);
3. отправить данные, которые необходимо напечатать в запросе PIN, на HSM;
4. осуществить проверку корректности выполнения команды печати;
5. обработать полученный PIN, выбранный держателем карты.

3.2.2.1 Настройка формата печати запроса

Настройка формата печати запроса PIN осуществляется аналогично настройке формата печати PIN-конверта (подробнее см. разд. 3.1.1) с помощью команды хоста PA. Полный перечень используемых символов для установки формата печати в HSM приведён в Приложении А.

В зависимости от настройки формата печати запроса он может содержать как только ссылочный номер без PIN (в случае, когда держатель карты должен самостоятельно выбрать PIN), так и ссылочный номер и PIN одновременно (в случае, когда держателю карты предлагается изменить PIN по умолчанию, установленный эмитентом, который будет использоваться, если клиент не укажет другой желаемый PIN).

PAN в открытом виде никогда не печатается в запросе, однако для удобства держателя карты могут печататься последние 6 цифр.

3.2.2.2 Генерация зашифрованного PIN

Генерация PIN требуется только в случае, когда владельцу карты предоставляется PIN по умолчанию, который предлагается изменить. Если эмитент требует, чтобы владелец карты всегда устанавливал собственный PIN, генерация PIN не требуется. Для генерации PIN используются команды хоста JA, EE.

3.2.2.3 Отправка данных для печати запроса

Для печати запроса PIN используются следующие команды хоста:

• PE — используется, когда необходимо напечатать и ссылочный номер (reference number), и PIN, установленный по умолчанию;

• OA — используется, когда необходимо напечатать только ссылочный номер (reference number).

Ссылочный номер (reference number) не передается в командах PE и QA в явном виде, однако он вычисляется HSM из 10 последних цифр PAN (за исключением контрольной цифры) перед печатью.

3.2.2.4 Проверка корректности выполнения команды печати запроса PIN

Хост не «видит» данные, которые HSM отправляет на принтер, однако он должен убедиться в корректности печати данных запроса PIN.

Для контроля корректности криптографических вычислений, выполненных в процессе печати по команде PE/OA, HSM отправляет хост-системе ответ, содержащий проверочное значение (check value) ссылочного номера (и PIN, в случае использования команды PE), вычисленное с использованием LMK.

Приложение хоста должно проверить полученное проверочное значение и подтвердить корректность выполнения команды печати PE (OA), отправив на HSM команду PG (RC).

Команда PG (RC) содержит те же значения PAN и зашифрованного PIN, что и команда PE (OA), а также проверочное значение ссылочного номера (и PIN), отправленное HSM в ответе на команду PE (OA).

Команду PG (RC) рекомендуется отправлять на другой HSM, отличный от выполнявшего команду печати, но использующий тот же LMK.

Ответ PH (RD) от HSM на команду PG (RC) содержит информацию о соответствии или несоответствии номера карты проверочному значению ссылочного номера и значения зашифрованного PIN проверочному значению PIN (в случае установки PIN).

3.2.2.5 Обработка PIN, выбранного держателем карты

После выбора держателем карты PIN он возвращается эмитенту вместе с ссылочным номером. Эти данные необходимо обработать до выдачи карты.

PIN и ссылочный номер необходимо ввести в приложение хоста. Хост должен проверить корректность ввода ссылочного номера с помощью контрольных цифр (подробнее см. разд. 3.2.1).

Ссылочный номер и незашифрованный PIN должны быть преобразованы в PAN и зашифрованный PIN, которые в таком виде могут быть переданы в систему выпуска карт. Это преобразование осуществляет HSM.

Загрузка данных запросов PIN в виде записей (ссылочный номер + выбранный PIN) в HSM осуществляется с помощью команды хоста QC.

Команда QC осуществляет загрузку записей пакетом, который может содержать до 1260 записей.

В ответ на команду QC HSM возвращает ответ QD, который содержит 10 крайних правых цифр PAN и PIN, зашифрованный под LMK, для каждой загруженной записи. Эти данные впоследствии будут использоваться приложением хоста для генерации данных при выпуске карты.

Чтобы не допустить возможность соотнести значение незашифрованного PIN, передаваемого в команде QC, с соответствующим номером счета в ответе QD, записи в ответе QD приводятся в случайном порядке. Для обеспечения дополнительной безопасности минимальный размер пакета (определяемый консольной командой CS в настройке Solicitation batch size) необходимо установить максимально большим.

Глава 4

Схема управления ключами транзакций

4.1 Общее описание

Схема управления ключами транзакций — метод защиты информации, при котором для каждой транзакции используется уникальный ключ шифрования данных, выработанный из определенного ключа. Таким образом, если диверсифицированный ключ скомпрометирован, данные прошлых транзакций будут по-прежнему защищены.

Обычно такие схемы используются в системах электронных платежей EFTPOS (Electronic Fund Transfer at Point of Sale), где запросы переводы денежных средств и ответы на них передаются между продавцом (терминалом EFTPOS) и эквайером, либо между эквайером и эмитентом карт.

HSM поддерживает схему **DUKPT** в соответствии с ANSI X9.24-3:2017.

DUKPT (Derived Unique Key Per Transaction) — схема управления ключами шифрования 3DES и AES в платежных системах.

DUKPT используется для шифрования PIN-блоков и других данных и аутентификации сообщений (MAC). Схема DUKPT обычно используется при передаче данных между продавцом и эквайером. Эквайер переупаковывает данные транзакции и отправляет их в процессинговый центр, затем данные отправляются эмитенту.

Для каждой транзакции PIN pad использует уникальный ключ, связанный с предыдущим ключом и несекретным серийным номером ключа, включающий счетчик транзакций. PIN pad зашифровывает PIN с использованием этого ключа и возвращает защифрованный PIN и серийный номер ключа эквайеру. HSM независимо от PIN pad вырабатывает то же ключ, используя исходный BDK и предоставленный PIN pad-ом серийный номер ключа.

За обслуживание и управление генерацией BDK отвечает хост. Для каждой транзакции хост проверяет серийный номер, предоставленный PIN pad-ом, и находит в хранилище соответствующий зашифрованный BDK, который идентифицируется крайними левыми цифрами серийного номера.

4.2 Ключи DUKPT

В DUKPT используются ключи 3 уровней:

1. BDK (Base Derivation Key, базовый ключ диверсификации) — мастер-ключ, принадлежащий эквайеру. Один BDK используется для большого количества терминалов, поставляемых провайдером (это могут все терминалы провайдера, или определенные модели терминалов, или терминалы с определенным диапазоном серийных номеров).

2. IKEY (Initial Key, начальный ключ диверсификации) (также IK или IPEK) — уникален для терминала, выработан из ключа BDK, используется для последующей выработки текущего ключа транзакции.

3. Ключ транзакции (Transaction key) — генерируются терминалом. Ключи шифрования PIN, шифрования данных и аутентификации сообщений (MAC) вырабатываются из ключа транзакции. Для каждой транзакции вырабатывается уникальный ключ для защиты данных. Когда эквайер получает зашифрованные данные, он вырабатывает тот же ключ транзакции и расшифровывает данные транзакции.

Эквайер и терминал используют одинаковый ключ IKEY и меняют его после миллиона транзакций.

4.2.1 BDK

BDK обычно генерируется и принадлежит эквайеру. Если BDK принадлежит организации, отличной от эквайера, его необходимо будет передать эквайеру для обработки транзакций. Его также необходимо будет передать любой другой организации, участвующей в генерации IKEY. При использовании нескольких BDK они, как правило, относятся к определенным логическим группам терминалов.

Ключи BDK могут передаваться в зашифрованном под ZMK виде или формироваться из разделенных ключевых компонент.

В ANSI X9.24-3:2017 определены 2 метода диверсификации ключей шифрования и аутентификации данных:

1. Двунаправленный

Для защиты данных, передаваемых от терминала к хосту и от хоста к терминалу, используется один ключ. Данный метод поддерживают BDK-1 (ключи аутентификации и шифрования) и BDK-3 (только ключи шифрования).

2. Однонаправленный

Используются 2 разных ключа: один для защиты данных, передаваемых от терминала к хосту, второй — данных, передаваемых от хоста к терминалу. Данный метод поддерживают BDK-2 и BDK-4 (ключи аутентификации и шифрования).

HSM поддерживает операции со следующими типами BDK:

Тип BDK	Тип ключа (Variant LMK)	Использование ключа (Key Block LMK)	Описание
BDK-1	009	B0	Используется для защиты данных транзакций между терминалом и эквайером. Следующие типы ключей могут быть выработаны из BDK-1: • ключи шифрования PIN • ключи аутентификации данных (MAC) • ключи шифрования данных Двунаправленный (один и тот же ключ используется для защиты сообщений терминал-эквайер и эквайер-терминал)
BDK-2	609	41	Используется для защиты данных транзакций между терминалом и эквайером. Следующие типы ключей могут быть выработаны из BDK-2: • ключи шифрования PIN (терминал-эквайер) • ключи аутентификации данных (MAC) (терминал-эквайер) • ключи аутентификации данных (MAC) (эквайер-терминал) • ключи шифрования данных (терминал-эквайер) • ключи шифрования данных (эквайер-терминал) Однонаправленный (для защиты сообщений терминал-эквайер и эквайер-терминал используются разные ключи)

BDK-3	809	42	Используется для защиты части данных транзакций между терминалом и эквайером, не включая выработку ключей шифрования PIN и аутентификации данных. Из BDK-3 может быть выработан только ключ шифрования данных (несмотря на то, что при этом используется вариант «проверка PIN»). Двунаправленный (один и тот же ключ используется для защиты сообщений терминал-эквайер и эквайер-терминал).
BDK-4	909	43	Используется для защиты данных транзакций между поставщиком платежных услуг (payment service provider, PSP), который эмулирует функцию терминала, и эквайером. Следующие типы ключей могут быть выработаны из BDK-4: • ключи шифрования PIN (PSP-эквайер) • ключи аутентификации данных (MAC) (PSP-эквайер) • ключи аутентификации данных (MAC) (эквайер-PSP) • ключи шифрования данных (PSP-эквайер) • ключи шифрования данных (эквайер-PSP) Одноправленный (для защиты сообщений PSP-эквайер и эквайер-PSP используются разные ключи)

Параметры ключей BDK (3DES и AES), используемых в командах шифрования PIN и аутентификации данных, а также значения вариантов (для 3DES BDK) и индикатора использования ключа (для AES BDK), используемые при выработке рабочих ключей шифрования PIN/аутентификации данных из BDK, приведены в разделе «Команды обработки транзакций DUKPT (X9.24)» документа «КриптоПро HSM. Команды хоста».

Параметры ключей BDK (3DES и AES), используемых в командах шифрования сообщений, а также значения вариантов (для 3DES BDK) и индикатора использования ключа (для AES BDK), используемые при выработке рабочих ключей шифрования данных из BDK, приведены в разделе «Команды шифрования сообщений» документа «КриптоПро HSM. Команды хоста».

4.2.2 IKEY

IKEY (IK/IPEK) является уникальным для каждого терминала. IKEY вычисляется из BDK и серийного номера ключа (Key Serial Number, KSN), также уникального для терминала.

IKEY может быть сгенерирован HSM с помощью команды хоста 'A0'. После генерации IKEY должен быть установлен в терминал.

4.2.3 Серийный номер ключа (KSN)

В случае 3DES BDK размер KSN 12-20 H (48-80 бит) и состоит из следующих элементов:

Название	**Длина**	**Описание**
Идентификатор ключевого набора (KSI)	5-9 H (20-36 бит)	Определяет BDK, который используется для терминала
Зарезервировано	1 H (4 бита)	Значение '0'

Идентификатор устройства	2-5 Н (8-20 бит)	Уникальный идентификатор (серийный номер) терминала (всегда четный)
Счетчик транзакций	1 бит + 5 Н (21 бит)	Счетчик операций зашифрования PIN с момента инициализации терминала

Часто встречается использование только 64-битных KSN, в этом случае KSN имеет следующую структуру:

Дополнение	4 Н (16 бит)	Значение 'FFFF'.
Идентификатор ключевого набора (KSI)	6 Н (24 бита)	
Идентификатор устройства	5 Н (20 бит)	Включает 1 бит счетчика транзакций (крайний правый бит) и 19 бит реального идентификатора устройства.
Счетчик транзакций	5 Н (20 бит)	Крайний левый бит счетчика транзакций является крайним правым битом идентификатора устройства.

Терминал не может обработать KSN длиннее 20 шестнадцатеричных символов, поэтому хост должен удостовериться, что длина первых трех полей KSN не превышает 15 символов.

Кроме того хост предоставляет HSM 3-символьный дескриптор KSN, который определяет длину каждой из первых 3 частей KSN. Дескриптор KSN совместно с KSN идентифицируют BDK.

3-символьный дескриптор KSN включает: длину идентификатора BDK (KSI) - левый символ, значение '0' — средний символ, длину идентификатора устройства — правый символ.

В случае AES BDK размер KSN 24 Н (96 бит).

4.2.4 Ключи транзакции

После установки IKEY в терминал он вычисляет «будущие» ключи транзакции, которые будут использоваться при шифровании транзакций. При вычислении ключей используется значение счетчика транзакций, значение которого увеличивается для каждой транзакции.

После вычисления первой части ключей IKEY удаляется из терминала.

При обработке транзакции используется следующий доступный ключ транзакции. Ключи, используемые для шифрования PIN-блока, аутентификации и шифрования данных, вырабатываются из этого ключа транзакции.

После использования ключ транзакции удаляется из терминала и заменяется следующим из ранее сгенерированных «будущих» ключей.

4.3 Обработка транзакции эквайером

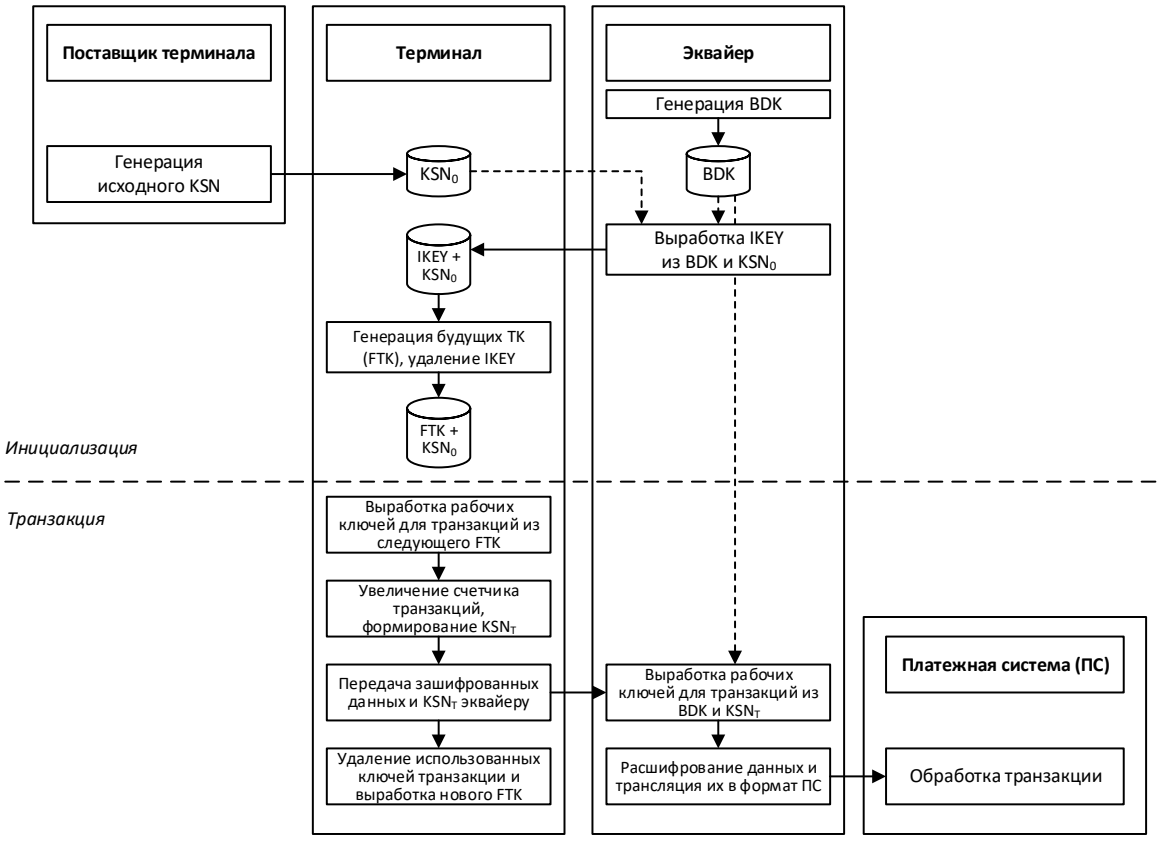
При отправке терминалом зашифрованных данных эквайеру также направляется KSN.

Совместно с зашифрованными данными транзакции терминал направляет эквайеру KSN (включая

счетчик транзакций). Эквайер может самостоятельно выработать ключ транзакции терминала, используя BDK (в соответствии со значение KSI) и полученный KSN.

После этого эквайер переупаковывает полученные от терминала данные в соответствии со стандартами платежной сети, что включает такие действия, как трансляцию PIN и проверку и трансляцию значений MAC.

Операции управления ключами транзакций представлены на схеме ниже.



Глава 5

Алгоритмы электронной подписи

RSA и ECC — асимметричные криптографические алгоритмы с открытым и закрытым ключами, используемые для электронной подписи и протоколов выработки общего симметричного ключа.

5.1 RSA

HSM поддерживает RSA ключи длиной от 320 до 4096 бит. В случае генерации RSA ключей длиннее 2048 бит необходимо использовать AES Key Block LMK.

Следующие команды хоста используются для поддержки операций с ключами RSA:

- 'EI' — Генерация ключевой пары RSA;
- 'EK' — Загрузка закрытого ключа;
- 'EM' — Трансляция закрытого ключа;
- 'EO' — Импорт открытого ключа;
- 'EQ' — Проверка открытого ключа;
- 'ES' — Проверка сертификата и импорт открытого ключа;
- 'EU' — Трансляция открытого ключа;
- 'EW' — Генерация подписи;
- 'EY' — Проверка подписи;
- 'GI' — Импорт ключа или данных, зашифрованных под открытом ключом RSA;
- 'GK' — Экспорт ключа, зашифрованного под открытым ключом RSA;
- 'GM' — Вычисление значения хэш-функции для блока данных.

Примечание:

Указанные выше команды хоста корректно обрабатывают только ключи RSA с нечетной открытой экспонентой.

Если в команде генерации ключевой пары RSA ('EI') указывается открытая экспонента, ее значение должно быть нечетным (т.е. младший бит должен быть равен 1), в противном случае HSM вернет ошибку.

5.2 ECC

Следующие команды хоста используются для поддержки операций с ключами ECC:

- 'FY' — Генерация ключевой пары ECC;
- 'QE' — Генерация запроса на сертификат;

- 'EO' — Импорт открытого ключа;
- 'EK' — Загрузка закрытого ключа;
- 'EM' — Трансляция закрытого ключа;
- 'EU' — Трансляция открытого ключа;
- 'EW' — Генерация подписи;
- 'EY' — Проверка подписи.

Операции с открытыми и закрытыми ключами ECC доступны только при использовании с AES Key Block LMK.

Эллиптические кривые, используемые в платёжной индустрии, определены в FIPS 186-3:

- NIST P-256;
- NIST P-384;
- NIST P-521.

5.3 Параметры команд

В таблице представлены допустимые значения параметров перечисленных выше команд хоста, используемых для поддержки операций с ключами RSA/ECC:

Параметр	**Допустимые значения**
Тип ключа	Параметры LMK, используемого для шифрования ключа в формате 'PPVV', где PP обозначает ключевую пару LMK, VV — номер варианта LMK.
Алгоритм подписи	'01': RSA '02': ECC
Идентификатор кривой (ECC)	'00': P-256 '01': P-384 '02': P-521
Алгоритм шифрования	'01': RSA

Алгоритм хэширования	'01': SHA-1
	
	`sha-1 OBJECT IDENTIFIER ::=`
	`{ iso(1) identified-organization(3) oiw(14) secsig(3) 2 26}`
	
	2B 0E 03 02 1A

	'02': MD5
	
	`md5 OBJECT IDENTIFIER ::=`
	`{iso(1) member-body(2) US(840) rsadsi(113549) digest Algorithm(2) 5}`
	
	2A 86 48 86 F7 0D 02 05

	'03': ISO 10118-2
	
	`iso_10118-2 OBJECT IDENTIFIER ::=`
	`{2 10 67 4}`
	
	5A 43 04

	'05': SHA-224
	
	`sha-224 OBJECT IDENTIFIER ::=`
	`{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)`
	`csor(3) nistalgorithm(4) hashalgs(2) sha224(4)}`
	
	60 86 48 01 65 03 04 02 04

	'06': SHA-256
	
	`sha-256 OBJECT IDENTIFIER ::=`
	`{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)`
	`csor(3) nistalgorithm(4) hashalgs(2) sha256(1)}`
	
	60 86 48 01 65 03 04 02 01

	'07': SHA-384
	
	`sha-384 OBJECT IDENTIFIER ::=`
	`{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)`
	`csor(3) nistalgorithm(4) hashalgs(2) sha384(2)}`
	
	60 86 48 01 65 03 04 02 02

	'08': SHA-512
	
	`sha-512 OBJECT IDENTIFIER ::=`
	`{joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)`
	`csor(3) nistalgorithm(4) hashalgs(2) sha512(3)}`
	
	60 86 48 01 65 03 04 02 03

Режим дополнения	'01': PKCS#1 v2.2 (EMSA-PKCS1-v1_5)
	'02': PKCS#1 v2.2 (EME-OAEP) Требуются дополнительные параметры: • Функция генерации маски (MGF) '01': MGF1 • Хэш-функция в MGF '01': SHA-1 '05': SHA-224 '06': SHA-256 '07': SHA-384 '08': SHA-512 • Длина OAEP Label • OAEP Label Если используется схема дополнения OAEP без значения Label, поле *Длина OAEP Label* должно иметь значение '00', а поле *OAEP Label* должно отсутствовать
	'04': PKCS#1 v2.2 (EMSA-PSS)

Тип Key Block	Используется в командах импорта/экспорта ключей, зашифрованных под открытом ключом RSA. Key Block типа '01' или '03' может использоваться для импорта DES/AES ключей. Key Block типа '03' или '04' может использоваться для импорта HMAC ключей.
	'01': Стандартный Key Block
	
	`Key block :: = SEQUENCE {`
	`key OCTET STRING,`
	`iv OCTET STRING`
	`}`
	
	Для ключей DES размер 'key' — 8, 16 или 24 байта, размер 'iv' — 8 байтов. Для ключей AES размер 'key' — 16, 24 или 32 байта, размер 'iv' — 16 байтов.

	'03': Неформатированный Key Block 8, 16 или 24 байта ключевых данных 2DES/3DES без кодировки или дополнительной информации.

	'04': Key Block в формате ASN.1
	
	`Key block :: = SEQUENCE {`
	`key OCTET STRING,`
	`iv OCTET STRING`
	`}`

Метод кодирования открытого ключа	'01': открытый ключ RSA в формате DER, ASN.1 (беззнаковое целое) '02': открытый ключ RSA в формате DER, ASN.1 (целое в формате дополнительного кода) '03': несжатый открытый ключ ECC в формате X9.62 DER, ASN.1

Глава 6

Локальные мастер-ключи (LMK)

Локальный мастер-ключ (Local Master Key, LMK) — главный локальный криптографический ключ HSM, применяемый для шифрования других криптографических ключей, используемых для шифрования, вычисления MAC, подписи и т.д., и данных, участвующих в платежных операциях. LMK является закрытым ключом, который формируется в оперативной памяти HSM с помощью разделенных ключевых компонент на смарт-картах. Каждый HSM должен содержать уникальный LMK (за исключением ситуаций, когда один LMK установлен в нескольких HSM в одной системе для целей резервирования, балансировки нагрузки и т.д.).

LMK обеспечивает классификацию операционных ключей по целям использования. HSM поддерживает два типа LMK:

1. Variant LMK — упорядоченный набор независимых ключей 2DES или 3DES, обеспечивающий классификацию ключей за счет шифрования разных типов операционных ключей с использованием разных вариантов LMK. Рекомендуется использовать 2DES Variant LMK только в целях совместимости с более старыми решениями.

2. Key Block LMK — ключ 3DES или AES-256, обеспечивающий классификацию ключей с помощью различных значений параметров в Key Block, которые определяют характеристики защищаемого ключа (зашифровываемого под LMK): назначение, режим использования, экспортируемость и др.

В один HSM могут быть загружены несколько LMK разных типов, подробнее см. Использование нескольких LMK.

6.1 Сравнительная таблица Variant LMK и Key Block LMK

Название ключа	**Variant**		**Key Block**		
	Тип ключа (Key Type)	**LMK**	**Использование ключа (Key Usage)**	**Алгоритм (Algorithm)**	**Режим использования (Mode of Use)**
BDK-1	009	28-29/0	'B0'	'T','A'	'X','N'
BDK-2	609	28-29/6	'41'	'T','A'	'X','N'
BDK-3	809	28-29/8	'42'	'T'	'X','N'
BDK-4	909	28-29/9	'43'	'T','A'	'X','N'
BDK-5	-	-	'44'	'T'	'X','N'
CK-DEK	50D	36-37/5	'39'	'T'	'X'
CK-DEK	50D	36-37/5	'39'	'A'	'B'
CK-ENC	30D	36-37/3	'37'	'T','A'	'X'

CK-MAC	40D	36-37/4	'38'	'T','A'	'X'
CSCK	402	14-15/4	'C0','11'	'T'	'C','G','V','N'
CTRDEK	-	-	'25'	'A'	'B','D','E','N'
CVK	402	14-15/4	'C0','12','13'	'T'	'C','G','V','N'
DEK	00B	32-33/0	'D0','21'	'T','A'	'B','D','E','N'
HMAC	10C	34-35/1	'61','62','63','64', '65'	'H'	'C','G','V','N'
IKEY (IPEK)	302	14-15/3	'B1'	'T','A'	'X','N'
KEK	107	24-25/1	'54'	'T','A'	'B','D','E','N'
KEK (транспортный ключ)	-	-	'55'	'A'	'E'
KEK (транспортный ключ)	-	-	'24'	'A'	'B','D','E','N'
KMC	207	24-25/2	'E7'	'T'	'X','N'
KML	200	04-05/2	'E6','31'	'T','A'	'X','N'
MK-AC	109	28-29/1	'E0'	'T','A'	'X','N'
MK-CVC3 (MK-DCVV)	709	28-29/7	'E6','32'	'T','A'	'X','N'
MK-DAC	409	28-29/4	'E3'	'T'	'X','N'
MK-DN	509	28-29/5	'E4'	'T'	'X','N'
MK-SMC	309	28-29/3	'E1'	'T'	'X','N'
MK-SMI	209	28-29/2	'E2'	'T','A'	'X','N'
M_KEY_CONF	-	-	'33'	'A'	'X','N'
M_KEY_MAC	-	-	'34'	'A'	'X','N'
MS_KEY_CONF	-	-	'35'	'A'	'B','D','E','N'
MS_KEY_MAC	-	-	'36'	'A'	'C','G','V','N'
MKCP	-	-	'E5'	'T'	'X','N'
PSK	507	24-25/5	'40'	'T'	'N'
PVK	002	14-15/0	'V0','V1','V2'	'T'	'C','G','V','N'
RSA (закрытый ключ)	00C	34-35/0	'03','04','05','06'	'R'	'D','N','S'
RSA (открытый ключ)	00D	36-37/0	'02'	'R'	'E','N','V'
ECC (закрытый ключ)	-	-	'03'	'E'	'X','N','S'
ECC (открытый ключ)	-	-	'02'	'E'	'X','N','V'
SK-DEK	507	24-25/5	'49'	'T'	'B','D','E','N'
SK-ENC	307	24-25/3	'47'	'T','A'	'B','D','E','N'
SK-MAC	407	24-25/4	'48'	'T','A'	'C','G','V','N'
SK-RMAC	008	26-27/0	'48'	'T','A'	'C','G','V','N'

TAK	003	16-17/0	'M0','M1','M3', 'M5','M6'	'T','A'	'C','G','V','N'
TEK	30B	32-33/3	'D0','23'	'T','A'	'B','D','E','N'
TKR	002 или 90D	14-15/0 или 36-37/9	'P0','73'	'T'	'N'
TMK	002 или 80D	14-15/0 или 36-37/8	'K0','K1','51'	'T','A'	'B','D','E','N'
TPK	002 или 70D	14-15/0 или 36-37/7	'P0','71'	'T','A'	'B','D','E','N'
WWK	006	22-23/0	'01'	'T'	'C','G','V','N'
ZAK	008	26-27/0	'M0','M1','M3', 'M5','M6'	'T','A'	'C','G','V','N'
ZEK	00A	30-31/0	'D0','22'	'T','A'	'B','D','E','N'
ZKA MK	607	24-25/6	'53'	'T'	'X','N'
ZMK	000	04-05/0	'K0','52'	'T','A'	'B','D','E','N'
ZPK	001	06-07/0	'P0','72'	'T','A'	'B','D','E','N'

6.2 Таблица соответствия наименований ключей

Разные платежные системы имеют собственные соглашения об именах используемых в HSM ключей. Ниже представлена сводная таблица соответствия наименований ключей, используемых в спецификациях платежных систем Mastercard и Visa, проприетарным наименованиям ключей в HSM.

Платежная система (ПС)	**Наименование ключа, используемое ПС**	**Проприетарное наименование ключа**	**Описание ключа**
Mastercard	Issuer MK	MK-AC	Мастер-ключ эмитента для генерации и проверки Application Cryptogram (AC)
		MK-SMI	Мастер-ключ эмитента для обеспечения целостности передаваемых данных (secure messaging)
		MK-SMC	Мастер-ключ эмитента для обеспечения конфиденциальности передаваемых данных (secure messaging)
		MK-DAC	Мастер-ключ эмитента для вычисления и проверки Data Authentication Code (DAC)
		MK-DN	Мастер-ключ эмитента для генерации Dynamic Number (DN)
	ICC MK	DK-AC	Диверсифицированный ключ для генерации и проверки Application Cryptogram (AC)
		DK-SMI	Диверсифицированный ключ для обеспечения целостности передаваемых данных (secure messaging)
		DK-SMC	Диверсифицированный ключ для обеспечения конфиденциальности передаваемых данных (secure messaging)
		DK-DN	Диверсифицированный ключ для генерации Dynamic Number (DN)

Visa	AWK (рабочий ключ эквайера)	ZPK	Зональный ключ шифрования PIN
	C2KA (CVK для генерации CVV2)	CVK	CVKA (первая часть 2DES CVK)
	C2KB (CVK для генерации CVV2)	CVK	CVKA (вторая часть 2DES CVK)
	CAKA (CVK для генерации CAVV)	CVK	CVKA (первая часть 2DES CVK)
	CAKB (CVK для генерации CAVV)	CVK	CVKA (вторая часть 2DES CVK)
	DMK-AC	MK-AC	Мастер-ключ эмитента для генерации и проверки Application Cryptogram (AC)
	DMK-MAC	MK-SMI	Мастер-ключ эмитента для обеспечения целостности передаваемых данных (secure messaging)
	DMK-ENC	MK-SMC	Мастер-ключ эмитента для обеспечения конфиденциальности передаваемых данных (secure messaging)
	IWK (рабочий ключ эмитента)	ZPK	Зональный ключ шифрования PIN

6.3 Variant LMK

2DES Variant LMK представляет собой упорядоченный набор из 20 независимых ключей 2DES, которые также обозначаются как «пары LMK». Различные пары LMK (и соответствующие варианты) из этого набора используются для шифрования различных типов платежных ключей.

Каждый ключ 2DES (пара LMK) состоит из левой и правой частей. Каждая часть представляет собой 16 шестнадцатеричных цифр.

Пример Тестового 2DES Variant LMK приведен на стр. 34.

3DES Variant LMK представляет собой упорядоченный набор из 20 независимых ключей 3DES (пар LMK).

Каждый ключ 3DES (пара LMK*) состоит из левой, средней и правой частей. Каждая часть представляет собой 16 шестнадцатеричных цифр.

Пример Тестового 3DES Variant LMK приведен на стр. 34.

*Термин «пара LMK» и связанная с ним схема нумерации независимых ключей в составе 3DES Variant LMK сохранены по историческим причинам, несмотря на то, что для 3DES Variant LMK каждый такой ключ состоит из 3 частей.

Variant LMK нельзя применять для защиты ключей ECC, AES и ключей RSA длины более 2048 бит.

Другие платежные ключи, используемые для шифрования (например, ZMK) и отличные от LMK, могут быть 2DES или 3DES ключами.

6.3.1 Тестовые Variant LMK

Для удобства в примерах зашифрования ключей, приведенных ниже, используется следующие тестовые Variant LMK.

6.3.1.1 2DES Variant LMK

Проверочное значение (KCV) для Тестового 2DES Variant LMK: **3D3639**

Пара (XX-YY)	**Пара LMK** (краткое обозначение: LMK XX-YY)	
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

6.3.1.2 3DES Variant LMK

Проверочное значение (KCV) для Тестового 3DES Variant LMK: **E75262**

Пара (XX-YY)	**Пара LMK** (краткое обозначение: LMK XX-YY)		
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

6.3.2 Применение вариантов к Variant LMK

Вариантом называется фиксированное двузначное шестнадцатеричное число.

Платежные ключи, в зависимости от их назначения, зашифровываются с использованием разных пар LMK с применением разных вариантов LMK. Таблица типов ключей (см. стр. 40) определяет пары LMK и их варианты. Например, CK-ENC зашифровывается с использованием пары LMK 36-37 с применением варианта 3.

В процессе формирования ключа шифрования вариант применяется к определенной паре LMK. Применение варианта происходит следующим образом: значение варианта побитово складывается (XOR) с первым байтом указанной пары LMK.

В HSM используются следующие варианты:

- Вариант 1 — A6
- Вариант 2 — 5A
- Вариант 3 — 6A
- Вариант 4 — DE
- Вариант 5 — 2B
- Вариант 6 — 50
- Вариант 7 — 74
- Вариант 8 — 9C
- Вариант 9 — FA

Пример 1. Для ключа CK-ENC необходимо вычислить вариант ключа LMK, для этого:

1. по таблице типов ключей (см. стр. 40) определяем, что для зашифрования CK-ENC необходимо использовать пару LMK 36-37 и вариант 3 (краткий формат записи: LMK 36-37/3):

Пара 36-37 для Тестового 2DES Variant LMK (см. стр. 34): 2F 2F 2F 2F 2F 2F 2F 2F 31 31 31 31 31 31 31 31

Вариант 3: 6A

2. побитово суммируем (XOR) значение варианта 3 и первый байт пары 36-37:

2F XOR 6A = 45

3. заменяем первый байт на полученную сумму и получаем итоговый ключ LMK 36-37/3 для зашифрования CK-ENC:

45 2F 2F 2F 2F 2F 2F 2F 31 31 31 31 31 31 31 31

Пример 2. Для ключа MK-SMI необходимо вычислить вариант ключа LMK, для этого:

1. по таблице таблице типов ключей (см. стр. 40) определяем, что для зашифрования MK-SMI необходимо использовать пару LMK 28-29 и вариант 2 (краткий формат записи: LMK 28-29/2):

Пара 28-29 для Тестового 2DES Variant LMK: 1A 1A 1A 1A 1A 1A 1A 1A 1C 1C 1C 1C 1C 1C 1C 1C

Вариант 2: 5A

2. побитово суммируем (XOR) значение варианта 2 и первый байт пары 28-29:

1A XOR 5A = 40

3. заменяем первый байт пары 28-29 LMK на полученную сумму и получаем итоговый ключ LMK 28-29/2 для зашифрования MK-SMI:

40 1A 1A 1A 1A 1A 1A 1A 1C 1C 1C 1C 1C 1C 1C 1C

6.3.2.1 Пример применения вариантов к Тестовому 2DES Variant LMK

Следующая таблица показывает, как изменяется первый байт указанной пары Тестового 2DES Variant LMK (см. стр. 34) при применении различных вариантов.

Пара LMK	**Вариант** (значение)								
	1 (A6)	**2** (5A)	**3** (6A)	**4** (DE)	**5** (2B)	**6** (50)	**7** (74)	**8** (9C)	**9** (FA)
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

6.3.2.2 Пример применения вариантов к Тестовому 3DES Variant LMK

Следующая таблица показывает, как изменяется первый байт указанной пары Тестового 3DES Variant LMK (см. стр. 34) при применении различных вариантов.

Пара LMK	**Вариант** (значение)								
	1 (A6)	**2** (5A)	**3** (6A)	**4** (DE)	**5** (2B)	**6** (50)	**7** (74)	**8** (9C)	**9** (FA)
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
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

6.3.3 Схема шифрования 2DES/3DES ключей под Variant LMK

Описанная выше схема применения вариантов к LMK и последующее шифрование платежного ключа под полученным ключом (выведенным из LMK) обеспечивает корректное использования платежного ключа в соответствии с его назначением.

Аналогичная схема используется для итогового формирования ключа шифрования: константы, фиксированные для каждой части шифруемого ключа, побитового складываются (XOR) с правой частью пары LMK в случае 2DES Variant LMK и со средней частью в случае 3DES Variant LMK.

Это обеспечивает однозначное применение частей 2DES/3DES ключей в соответствии с их «расположением».

В зависимости от шифруемой части платежного ключа используются следующие константы:

- для 2DES ключей:

левая часть — A6

правая часть — 5A

- для 3DES ключей:

левая часть — 6A

средняя часть — DE

правая часть — 2B

Для определения схемы шифрования платежного ключа используются следующие признаки схемы (tag):

- U — платежные ключи 2DES, зашифрованные под Variant LMK
- T — платежные ключи 3DES, зашифрованные под Variant LMK

Порядок зашифрования 2DES/3DES платежных ключей под Variant LMK:

1. по таблице типов ключей (см. стр. 40) определяется соответствующая пара LMK и вариант;

2. к паре LMK применяется вариант: побитово суммируются (XOR) значение варианта и первый байт **левой** части пары LMK, первый байт левой части пары LMK заменяется на полученную сумму;

3. для каждой из частей шифруемого платежного ключа: побитово суммируются (XOR) константа, определенная для этой части платежного ключа, и первый байт **правой** части (в случае 2DES Variant LMK) или **средней** части (в случае 3DES Variant LMK) пары LMK, затем первый байт правой части (в случае 2DES Variant LMK) или средней части (в случае 3DES Variant LMK) пары LMK заменяется на полученную сумму;

4. каждая часть шифруемого платежного ключа отдельно зашифровывается (с использованием режима ECB) под соответствующим ключом, состоящим из частей, полученных в пп. 2 и 3;

5. отдельно зашифрованные части конкатенируются для получения итогового зашифрованного ключа.

Пример. Платёжный ключ MK-SMI = F1 F1 F1 F1 F1 F1 F1 F1 C1 C1 C1 C1 C1 C1 C1 C1, который является 2DES ключом, зашифруем под Тестовым 2DES Variant LMK (см. стр. 34):

1. по таблице типов ключей (см. стр. 40) определяем, что для зашифрования MK-SMI необходимо использовать пару LMK 28-29 и вариант 2:

Пара 28-29 для Тестового LMK: **1A** 1A 1A 1A 1A 1A 1A 1A 1C 1C 1C 1C 1C 1C 1C 1C

Вариант 2: 5A

2. применяем вариант 2 к паре 28-29 (побитово суммируем (XOR) первый байт и значение варианта):

1A XOR 5A = 40

LMK 28-29/2: **40** 1A 1A 1A 1A 1A 1A 1A 1C 1C 1C 1C 1C 1C 1C 1C;

3. для левой части MK-SMI:

(a) формируем правую часть ключа зашифрования: побитово суммируем (XOR) первый байт правой части пары LMK 28-29 и константу A6, соответствующую левой части 2DES MK-SMI:

1C XOR A6 = BA

Правая часть ключа зашифрования: **BA** 1C 1C 1C 1C 1C 1C 1C

(b) объединяем левую (шаг 2) и правую части ключа зашифрования:

Итоговый ключ зашифрования левой части MK-SMI: **40** 1A 1A 1A 1A 1A 1A 1A **BA** 1C 1C 1C 1C 1C 1C 1C

(c) зашифровываем левую часть MK-SMI с использованием полученного ключа:

Левая часть MK-SMI: F1 F1 F1 F1 F1 F1 F1 F1

Результат зашифрования левой части MK-SMI: 51 78 C9 D3 D1 05 2B 15

4. для правой части MK-SMI:

(a) формируем правую часть ключа зашифрования: побитово суммируем (XOR) первый байт правой части пары LMK 28-29 и константу 5A, соответствующую правой части 2DES MK-SMI:

1C XOR 5A = 46

Правая часть ключа зашифрования: **46** 1C 1C 1C 1C 1C 1C 1C

(b) объединяем левую (шаг 2) и правую части ключа зашифрования:

Итоговый ключ зашифрования правой части MK-SMI:

40 1A 1A 1A 1A 1A 1A 1A **46** 1C 1C 1C 1C 1C 1C 1C

(c) зашифровываем правую часть MK-SMI с использованием полученного ключа:

Правая часть MK-SMI: C1 C1 C1 C1 C1 C1 C1 C1

Результат зашифрования правой части MK-SMI: BF 6A EC 45 8B 4A 45 64

5. формируем итоговый зашифрованный MK-SMI из отдельно зашифрованный частей

Зашифрованный MK-SMI: 51 78 C9 D3 D1 05 2B 15 BF 6A EC 45 8B 4A 45 64

6.3.4 Типы Variant-ключей

Все платёжные ключи, обрабатываемые HSM, имеют свой код типа ключа (Key Type Code) в соответствии с таблицей ниже. Соответствие между типом ключа и используемом для его шифрования вариантом LMK приведено в следующем разделе.

Некоторые ключи имеют два разных кода типа ключа, применяемых в зависимости от выставленного значения настройки безопасности `Enforce key type 002 separation for PCI HSM compliance`. Код типа ключа, указанный в колонке **Код типа (non-PCI)**, применяется, если настройка имеет значение "No". Код типа ключа, указанный в колонке **Код типа (PCI)**, применяется, если настройка имеет значение "Yes".

Ключ	**Код типа (non-PCI)**	**Код типа (PCI)**	**Описание ключа**
ZMK	000		Зональный мастер-ключ (также ZCMK)
KML	200		Мастер-ключ загрузки (Visa Cash)
ZPK	001		Зональный ключ шифрования PIN
PVK	002		Ключ проверки PIN
TPK	002	70D	Терминальный ключ шифрования PIN
TMK	002	80D	Терминальный мастер-ключ
TKR	002	90D	Регистр терминального ключа
IKEY (IPEK)	302		Начальный ключ DUKPT
CK-ENK	30D		Ключ карты для шифрования криптограмм (эмиссия)
CVK	402		Ключ проверки карты
CSCK	402		Ключ вычисления/проверки CSC
CK-MAC	40D		Ключ карты для аутентификации (эмиссия)
CK-DEK	50D		Ключ карты для шифрования данных (эмиссия)
TAK	003		Терминальный ключ аутентификации
WWK	006		Watchword Key
KEK	107		Ключ шифрования ключей (эмиссия)
KMC	207		Мастер-ключ для персонализации (эмиссия)
SK-ENC	307		Сессионный ключ для криптограмм и шифрования сообщений карты (эмиссия)
SK-MAC	407		Сессионный ключ для аутентификации сообщений карты (эмиссия)
SK-DEK	507		Сессионный ключ для шифрования конфиденциальных данных карты (эмиссия)
PSK	507		Ключ системы персонализации (эмиссия)
KD-PERSO	507		Ключ персонализации KD (эмиссия)
ZKA MK	607		Мастер-ключ для GBIC/ZKA диверсификации ключа
ZAK	008		Зональный ключ аутентификации
SK-RMAC	008		Сессионный ключ для аутентификации ответов карты (R-MAC)
BDK-1	009		Базовый ключ диверсификации DUKPT BDK-1
MK-AC	109		Мастер-ключ для генерации и проверки криптограмм

MK-SMI	209	Мастер-ключ для обеспечения целостности соообщения
MK-SMC	309	Мастер-ключ для обеспечения конфиденциальности сообщения
MK-DAC	409	Мастер-ключ для генерации и проверки кодов аутентификации данных
MK-DN	509	Мастер ключ для генерации Dynamic Number
BDK-2	609	Базовый ключ диверсификации DUKPT BDK-2
MK-CVC3 (MK-DCVV)	709	Мастер ключ DCVV (бесконтактный)
BDK-3	809	Базовый ключ диверсификации DUKPT BDK-3
BDK-4	909	Базовый ключ диверсификации DUKPT BDK-4
ZEK	00A	Зональный ключ шифрования
DEK	00B	Ключ шифрования данных
TEK	30B	Терминальный ключ шифрования
RSA-SK	00C	Закрытый ключ RSA
HMAC	10C	Ключ HMAC
RSA-PK	00D	Открытый ключ RSA

6.3.5 Таблица типов ключей

В таблицах ниже указаны операции, которые можно применить для каждой пары LMK и соответствующего варианта:

- Г — генерация ключа;
- Э — экспорт ключа (расшифрование ключа, зашифрованного под LMK, и последующее зашифрование под транспортным ключом для передачи другой стороне);
- И — импорт ключа (расшифрование полученного ключа, зашифрованного под транспортным ключом, и последующее зашифрование под LMK).

Каждое из этих 3 полей (Г/Э/И) содержит один из флагов, определяющих допустимость и условия операции:

- пустое поле — операция не допускается;
- А — для операции требуется авторизация;
- Б — операция разрешена безусловно (вне зависимости от авторизации).

Код типа ключа, используемый в командах, формируется путем конкатенации номера варианта и кода пары LMK. Например, ключ ZEK имеет код типа 00A (0 — номер варианта, 0A — код пары LMK).

6.3.5.1 Без совместимости с PCI HSM

Следующая таблица используется для определения возможных операций (генерации, экспорта и импорта ключей) с использованием пары LMK с применением различных вариантов, если выставлена настройка `Enforce key type 002 separation for PCI HSM compliance: No`.

Вариант →		0			1			2			3			4			5			6			7			8			9		
LMK↓		Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И
Пара	Код																														
04-05	00	ZMK						KML																							
		А	А[6]	А[7]				Б	А	Б																					
06-07	01	ZPK																													
		Б	А	Б																											
14-15	02	PVK TPK TMK TKR									IKEY			CVK CSCK																	
		Б	А	Б							Б	А	Б	Б	А	Б															
16-17	03	TAK																													
		Б	А	Б																											
18-19	04				DTAB[1]			IPB[3]																							
20-21	05																														
22-23	06	WWK																													
		Б	А	Б																											
24-25	07				KEK			KMC			SK-ENC			SK-MAC			SK-DEK PSK KD-PERSO			ZKA MK											
					Б	А	А	Б	А	А	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б									
26-27	08	ZAK SK-RMAC																													
		Б	А	Б																											
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		
		Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б
30-31	0A	ZEK																													
		Б	А	А/Б[2]																											
32-33	0B	DEK									TEK																				
		Б	А	Б							Б	А	А/Б[2]																		
34-35	0C	RSA-SK			HMAC																										
		А	А[5]	А[5]	Б	А	Б																								
36-37	0D	RSA-PK									CK-ENC			CK-MAC			CK-DEK						TPK[4]			TMK[4]			TKR[4]		
		А		А							Б	А	Б	Б	А	Б	Б	А	Б				Б	А	Б	Б	А	Б	Б	А	Б
38-39	0E	Резерв			Резерв			Резерв			Резерв			Резерв			Резерв			Резерв			Резерв			Резерв			Резерв		

Примечания:

1. DTAB — таблица децимализации

2. Необходимость авторизации при импорте (И) зависит от значения настройки безопасности `Enable ZEK/TEK encryption of ASCII data or Binary data or None`;

3. IPB — Issuer Proprietary Bitmap

4. Ключи TPK, TMK, TKR могут быть сгенерированы, экспортированы, импортированы, но их использование возможно, только если настройка безопасности `Enforce key type 002 separation for PCI HSM compliance` установлена в значение "Yes";

5. Допускается, если настройка безопасности `Enable import and export of RSA Private keys` установлена в значение "Yes";

6. Допускается, если настройка безопасности `Enable export of a ZMK` установлена в значение "Yes";
7. Допускается, если настройка безопасности `Enable import of a ZMK` установлена в значение "Yes".

6.3.5.2 В режиме соответствия PCI HSM

Следующая таблица используется для определения возможных операций (генерации, экспорта и импорта ключей) с использованием пары LMK с применением различных вариантов, если выставлена настройка `Enforce key type 002 separation for PCI HSM compliance: Yes`.

Вариант →		0			1			2			3			4			5			6			7			8			9		
LMK↓		Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И	Г	Э	И
Пара	Код																														
04-05	00	ZMK						KML																							
		А	А[5]	А[6]				Б	А	Б																					
06-07	01	ZPK																													
		Б	А	Б																											
14-15	02	PVK									IKEY			CVK CSCK																	
		Б	А	Б							Б	А	Б	Б	А	Б															
16-17	03	TAK																													
		Б	А	Б																											
18-19	04				DTAB[1]			IPB[3]																							
20-21	05																														
22-23	06	WWK																													
		Б	А	Б																											
24-25	07				KEK			KMC			SK-ENC			SK-MAC			SK-DEK PSK KD-PERSO			ZKA MK											
					Б	А	А	Б	А	А	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б										
26-27	08	ZAK SK-RMAC																													
		Б	А	Б																											
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		
		Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б	Б	А	Б
30-31	0A	ZEK																													
		Б	А	А/Б[2]																											
32-33	0B	DEK									TEK																				
		Б	А	Б							Б	А	А/Б[2]																		
34-35	0C	RSA-SK			HMAC																										
		А	А[4]	А[4]	Б	А	Б																								
36-37	0D	RSA-PK									CK-ENC			CK-MAC			CK-DEK						TPK			TMK			TKR		
		А		А							Б	А	Б	Б	А	Б	Б	А	Б				Б	А	Б	Б	А	Б	Б	А	Б
38-39	0E	Резерв			Резерв			Резерв			Резерв			Резерв			Резерв			Резерв			Резерв			Резерв			Резерв		

Примечания:

1. DTAB — таблица децимализации

2. Необходимость авторизации при импорте (И) зависит от значения настройки безопасности `Enable ZEK/TEK encryption of ASCII data or Binary data or None`;

3. IPB — Issuer Proprietary Bitmap

4. Допускается, если настройка безопасности `Enable import and export of RSA Private keys` установлена в значение "Yes";

5. Допускается, если настройка безопасности `Enable export of a ZMK` установлена в значение "Yes";

6. Допускается, если настройка безопасности `Enable import of a ZMK` установлена в значение "Yes".

6.4 Key Block LMK

Key Block LMK является более новым типом LMK, не совместимым с Variant LMK. При использовании Key Block LMK зашифрованные под ним платежные ключи представляются в формате Проприетарного Key Block (подробнее описан ниже).

Использование термина «Key Block LMK» означает, что защищаемые им ключи представлены в формате «key block», при этом сам LMK не хранится в формате «key block».

HSM поддерживает 2 типа Key Block LMK в зависимости от используемого алгоритма:

- 3DES Key Block LMK

Основан на 168-битном ключе 3DES. Такие LMK могут использоваться для защиты ключей 3DES, RSA (не более 2048 бит) и HMAC.

- AES Key Block LMK

Основан на 256-битном ключе AES. Такие LMK могут использоваться для защиты ключей AES, 3DES, RSA, ECDSA и HMAC.

6.4.1 Тестовые Key Block LMK

6.4.1.1 3DES Key Block LMK

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
Проверочное значение (KCV)	8E0EC0

6.4.1.2 AES Key Block LMK

LMK	9B 71 33 3A 13 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
Проверочное значение (KCV)	9D04A0

6.4.2 Проприетарный Key Block

В HSM используется Проприетарный Key Block, аналогичный по структуре X9 TR-31 Key Block (описан в гл. 9), который соответствует стандарту ANSI X9.24 и используется для обеспечения совместимости форматов ключей при экспорте между оборудованием различных производителей.

Основное отличие Проприетарного Key Block от TR-31 Key Block заключается в поддержке дополнительных значений Использования ключа (key usage) и опциональных блоков в заголовке (Key Block Header).

Проприетарный Key Block обеспечивает связь зашифрованного ключа с его предполагаемым использованием и другими метаданными с помощью MAC, таким образом обеспечивая соответствие требованиям PCI PIN Security Requirements.

Некоторые команды консоли и хоста не поддерживают работу с Key Block LMK. Информация о поддержке Key Block LMK указана в описании команд в документах «КриптоПро HSM. Команды хоста» и «КриптоПро HSM. Команды консоли».

6.4.3 Структура Проприетарного Key Block

Проприетарный Key Block обозначается буквой 'S' и имеет следующий формат (все блоки должны быть представлены в ASCII-кодировке):

Ключевая схема	Заголовок Key Block	Опциональный заголовок	Зашифрованный ключ	Аутентификатор
1 байт	16 байт	переменная длина	переменная длина	8 или 16 байт
значение 'S'	определяет область использования, режим использования, алгоритм и ограничения на экспорт ключа; также определяет LMK, используемый для зашифрования ключа в формате Key Block	дополнительные характеристики ключа (например, срок действия ключа)	ключ, зашифрованный с использованием соответствующего ключа, диверсифицированного из LMK (или ZMK/TMK)	индикатор несанкционированного изменения Key Block, вычисленный с использованием соответствующего ключа, диверсифицированного из LMK (или ZMK/TMK)

6.4.3.1 Заголовок Key Block

Заголовок Проприетарного Key Block состоит из 16 байтов (ASCII символов) и имеет следующий формат:

Байт(ы)	**Поле**	**Определение**
0	Идентификатор версии (Version ID)	Значение '0' (0x30) для защиты под 3DES ключом Значение '1' (0x31) для для защиты под AES ключом
1-4	Длина Key Block	Общая длина Key Block
5-6	Использование ключа (Key Usage)	Например, шифрование ключей или шифрование данных
7	Алгоритм	Например, 3DES
8	Режим использования (Mode of Use)	Например, только для шифрования
9-10	Номер версии ключа (Key Version Number)	Например, версия ключа в Key Block или указание на то, что это компонента ключа
11	Экспортируемость	Например, экспорт не разрешен
12-13	Количество опциональных блоков	Количество опциональных блоков заголовка
14-15	LMK ID	В случае ключа шифрования LMK — идентификатор LMK (поддержка использования нескольких LMK); значение '00' .. '09' (0x3030 .. 0x3039) В случае ключа шифрования ZMK/TMK — зарезервированное значение 'FF'

6.4.3.1.1 Длина Key Block (Байты 1-4) Байты 1-4 заголовка содержат длину всего Key Block (заголовка, опциональных полей, зашифрованного ключа и аутентификатора). Значение длины Key Block вычисляется после кодирования и представляется 4-значным набором ASCII символов.

Например, если общая длина Key Block составляет 112 символов (байтов), значение 1-го байта равно '0', значение 2-го байта равно '1', значение 3-го байта равно '1', значение 4-го байта равно '2' (т.е. 0x30313132).

6.4.3.1.2 Использование ключа (Байты 5-6) Байты 5-6 заголовка определяют назначение ключа в Key Block. В следующей таблице представлены допустимые значения Использования ключа для

Проприетарного Key Block и соответствующие значения этого поля для TR-31 Key Block.

Проприетарный Key Block	**TR-31 Key Block**	**Алгоритм**	**Описание**
01	-	3DES	WatchWordKey (WWK)
02	-	RSA/ECC	RSA/ECC (открытый ключ)
03	-	RSA	Закрытый ключ RSA подписи/управления ключами
03	S0/K3/03	ECC	Закрытый ключ ECC подписи/обмена/общего назначения
04	-	RSA	RSA (закрытый ключ карты)
05	-	RSA	RSA (закрытый ключ для трансляции PIN)
06	-	RSA	RSA (закрытый ключ для расшифрования TLS premaster secret)
B0	B0	3DES/AES	Базовый ключ диверсификации DUKPT BDK-1
41	B0	3DES/AES	Базовый ключ диверсификации DUKPT BDK-2
42	B0	3DES	Базовый ключ диверсификации DUKPT BDK-3
43	B0	3DES/AES	Базовый ключ диверсификации DUKPT BDK-4
B1	B1	3DES/AES	Начальный ключ диверсификации DUKPT IKEY (IPEK)
C0	C0	3DES	Ключ проверки карты CVK
11	C0	3DES	Ключ проверки карты CVK (American Express CSC)
12	C0	3DES	Ключ проверки карты CVK (Mastercard CVC)
13	C0	3DES	Ключ проверки карты CVK (Visa CVV)
D0	D0	AES/3DES	Ключ шифрования данных (универсальный)
21	D0	AES/3DES	Ключ шифрования данных (DEK)
22	D0	AES/3DES	Ключ шифрования данных (ZEK)
23	D0	AES/3DES	Ключ шифрования данных (TEK)
24	-	AES	Ключ шифрования ключей (транспортный ключ)
25	D0	AES	Ключ шифрования данных (CTRDEK)
E0	E0	AES/3DES	Мастер-ключ EMV для генерации и проверки Application Cryptogram (MK-AC)
E1	E1	3DES	Мастер-ключ EMV для обеспечения конфиденциальности (secure messaging) (MK-SMC)
E2	E2	3DES/AES	Мастер-ключ EMV для обеспечения целостности (secure messaging) (MK-SMI)

E3	E3	3DES	Мастер-ключ EMV для вычисления и проверки Data Authentication Code (MK-DAC)
E4	E4	3DES	Мастер-ключ EMV для генерации Dynamic Number (MK-DN)
E5	E5	3DES	Мастер-ключ EMV персонализации карт
E6	E6	3DES/AES	Мастер-ключ EMV (другое)
E7	E7	3DES/AES	Ключ персонализации EMV
31	E6	3DES	Мастер-ключ загрузки Visa Cash (KML)
32	E6	3DES/AES	Мастер-ключ DCVV (MK-CVC3)
33	-	AES	Мастер-ключ Mobile Remote Management для обеспечения конфиденциальности (M_KEY_CONF)
34	-	AES	Мастер-ключ Mobile Remote Management для обеспечения целостности (M_KEY_MAC)
35	-	AES	Сессионный ключ Mobile Remote Management для обеспечения конфиденциальности (MS_KEY_CONF)
36	-	AES	Сессионный ключ Mobile Remote Management для обеспечения целостности (MS_KEY_MAC)
37	-	3DES/AES	Ключ карты EMV для криптограмм
38	-	3DES/AES	Ключ карты EMV для обеспечения целостности
39	-	3DES/AES	Ключ карты EMV для шифрования
40	-	3DES	Ключ системы персонализации EMV
47	-	3DES/AES	Сессионный ключ EMV для криптограмм
48	-	3DES/AES	Сессионный ключ EMV для обеспечения целостности
49	-	3DES	Сессионный ключ EMV для шифрования
K0	K0	AES/3DES	Шифрование ключа или key wrapping (универсальный)
K1	K1	AES/3DES	Шифрование ключа или key wrapping (универсальный)
51	K0/K1	AES/3DES	Терминальный ключ шифрования ключей (TMK)
52	K0/K1	AES/3DES	Зональный ключ шифрования ключей (ZMK)
53	-	3DES	Мастер-ключ ZKA (ZKA-MK)
54	K0	AES/3DES	Ключ шифрования ключей (KEK)
55	-	AES	Ключ шифрования ключей (транспортный ключ)
M0	M0	3DES	ISO 16609 MAC algorithm 1 (с использованием 3DES)
M1	M1	3DES	ISO 9797-1 MAC algorithm 1

M2	M2	3DES	ISO 9797-1 MAC algorithm 2
M3	M3	3DES	ISO 9797-1 MAC algorithm 3
M4	M4	3DES	ISO 9797-1 MAC algorithm 4
M5	M5	AES	CBC-MAC (с использованием AES)
M6	M6	AES	CMAC (с использованием AES)
61	-	HMAC	Ключ HMAC (с использованием SHA-1)
62	-	HMAC	Ключ HMAC (с использованием SHA-224)
63	-	HMAC	Ключ HMAC (с использованием SHA-256)
64	-	HMAC	Ключ HMAC (с использованием SHA-384)
65	-	HMAC	Ключ HMAC (с использованием SHA-512)
P0	P0	AES/3DES	Ключ шифрования PIN (универсальный)
71	P0	AES/3DES	Терминальный ключ шифрования PIN (TPK)
72	P0	AES/3DES	Зональный ключ шифрования PIN (ZPK)
73	P0	3DES	Terminal Key Register (TKR)
V0	V0	3DES	Ключ проверки PIN (универсальный)
V1	V1	3DES	Ключ проверки PIN (IBM 3624)
V2	V2	3DES	Ключ проверки PIN (ABA PVV)

6.4.3.1.3 Алгоритм (Байт 7) Байт 7 заголовка определяет криптографический алгоритм ключа в Key Block в соответствии со следующей таблицей.

Значение	**Hex**	**Алгоритм**
'A'	0x41	AES
'E'	0x45	ECC
'H'	0x48	HMAC
'R'	0x52	RSA
'T'	0x54	3DES

6.4.3.1.4 Режим использования (Байт 8) Байт 8 заголовка определяет доступные операции для ключа в Key Block в соответствии со следующей таблицей.

Значение	**Hex**	**Описание**
'B'	0x42	Зашифрование и расшифрование
'C'	0x43	Вычисление MAC (генерация или проверка)
'D'	0x44	Расшифрование
'E'	0x45	Зашифрование
'G'	0x47	Генерация MAC
'N'	0x4E	Ограничений нет
'S'	0x53	Генерация ЭП

'V'	0x56	Проверка ЭП или MAC
'X'	0x58	Диверсификация других ключей

Для ключей HMAC применяются значения 'C', 'G' и 'V' (как для «обычных» ключей MAC).

6.4.3.1.5 Номер версии ключа (Байты 9-10) Байты 9-10 заголовка определяют номер версии ключа в Key Block или указывает на то, что в Key Block содержится ключевая компонента.

Байт 9	**Байт 10**	**Описание**
0 (0x30)	0 (0x30)	Для данного ключа версия не применяется
c (0x63)	любой	В Key Block содержится компонента ключа
Другая комбинация символов		Версия ключа/ключевой компоненты в Key Block

Если байт 9 содержит значение 'c', то байт 10 используется для обозначения номера компоненты. Например, если байты 9-10 = 'c3' (0x6333), ключ в Key Block является третьей компонентой ключа. При этом определить общее количество ключевых компонент невозможно.

6.4.3.1.6 Экспортируемость (Байт 11) Байт 11 заголовка определяет условия, при которых ключ, содержащийся в Key Block, может быть экспортирован. Ключ считается доверенным, если он представлен в формате Key Block. Другие форматы ключей считаются недоверенными.

Значение	**Hex**	**Описание**
'E'	0x45	Допускается экспорт только в доверенный формат, при этом wrapping key также должен быть представлен в одном из этих форматов
'N'	0x4E	Экспорт ключа не допускается
'S'	0x53	Разрешен экспорт в любые форматы (при условии, что они включены в настройках безопасности)

Значение экспортируемости 'S' применяется, например, если используется Key Block LMK и ключ экспортируется в формат ANSI X9.17. Экспорт в ANSI X9.17 формат по умолчанию отключен в настройках безопасности HSM и должен быть специально включен, например, с помощью консольной команды CS.

Когда ключ экспортируется в форматы Проприетарный Key Block или TR-31 Key Block, байт 11 нового экспортированного Key Block определяет, возможен ли дальнейший экспорт получателем ключа в формате Key Block. Например, если байт 11 в полученном Key Block имеет значение 'N', то дальнейший экспорт ключа не допускается. Поэтому необходимо учитывать данный параметр при первоначальной генерации ключа. Значение байта 11 можно изменить с 'E' или 'S' на 'N' с помощью специального поля (Новое значение экспортируемости) команд экспорта ключа.

6.4.3.1.7 Количество опциональных блоков (Байты 12-13) Проприетарный Key Block позволяет включать в заголовок до 99 опциональных блоков (Optional Header Blocks), которые содержат дополнительные (опциональные) данные о Key Block.

Байты 12-13 заголовка определяют количество опциональных блоков заголовка в Key Block. Например, значение '12' (0x3132) означает, что Key Block содержит 12 опциональных блоков; значение '00' (0x3030) указывает на отсутствие опциональных блоков.

6.4.3.1.8 Идентификатор LMK (Байты 14-15) Байты 14-15 заголовка определяют номер LMK, который используется для защиты ключа в формате Key Block, обеспечивая поддержку использования в HSM нескольких LMK одновременно.

Идентификатор LMK может принимать значения от '00' до '09'.

HSM поддерживает настройку безопасности `Ignore LMK ID in Key Block Header`. Если настройка включена, идентификатор LMK в заголовке Проприетарного Key Block игнорируется. Вместо этого HSM использует тот же механизм получения идентификатора LMK, что и в случае использования Variant LMK — путем указания идентификатора LMK в команде. Это позволяет работать с одним и тем же Key Block на разных HSM без необходимости загрузки LMK в HSM под одинаковыми номерами (идентификаторами).

Если для защиты ключа используется ZMK или TMK, то это поле зарезервировано и имеет фиксированное значение 'FF'.

6.4.3.1.9 Пример заголовка Проприетарного Key Block

Байт	0	1-4	5-6	7	8	9-10	11	12-13	14-15
Значение	0	0072	**V2**	T	G	22	N	00	03

- для защиты используется 3DES ключ (идентификатор версии 0)
- длина Key Block 72 байта (0072)
- ключ может использоваться только с методом проверки PIN ABA PVV (**V2**)
- алгоритм 3DES (T)
- ключ может использоваться только для генерации PVV (G)
- версия ключа 22
- ключ нельзя экспортировать (N)
- опциональные блоки заголовка отсутствуют (00)
- ключ в Key Block защищен с использованием LMK с идентификатором 03

В примере выше варианты использования Key Block сильно ограничены. При генерации Key Block необходимо заранее удостовериться в правильности указываемых параметров заголовка. Для изменения некоторых полей заголовка Key Block может использоваться команда хоста 'CS', однако она позволяет лишь усилить ограничения использования Key Block.

6.4.3.2 Опциональный заголовок

Опциональный заголовок состоит из нескольких **опциональных блоков**, которые имеют следующую структуру:

Байты	**Поле**	**Комментарий**
0-1	Идентификатор блока	Идентификатор опционального блока
2-3	Длина блока	Длина опционального блока в байтах (шестнадцатеричное число, ASCII). Например, длина блока 24 (десятичное число) представляется как 0x18, в ASCII-кодировке — 0x3138 Если опциональный блок не содержит данных, поле длины содержит значение 0x04 (0x3034 в ASCII-кодировке)
4-n (n $\leq$ 251)	Данные блока	Данные опционального блока

Максимальная длина опционального блока — 255 байтов, минимальная — 4 байта. Общая длина всех опциональных блоков заголовка должна быть кратна длине блока шифрования (8 байтов в случае 3DES), что достигается включением блока дополнения (padding), который должен быть последним опциональным блоком.

Если Key Block в команде содержит несколько опциональных блоков с одинаковыми значениями идентификатора, HSM вернет ошибку 'BC' (Повторяющийся опциональный блок).

6.4.3.2.1 Типы опциональных блоков заголовка Типы опциональных блоков заголовка, поддерживаемые в Проприетарном Key Block, приведены в таблице ниже. Первые 3 типа также поддерживаются в TR-31 Key Block.

Идентификатор опционального блока	**Hex**	**Название**	**Описание**
KS	0x4B53	Идентификатор ключевого набора	Примеры см. в ANSI X9.24 part 3.
KV	0x4B56	Версия Key Block	Используется для обозначения версии набора значений полей Key Block и идентификации того, что Key Block содержит не утвержденные ANSI значения. Поле должно состоять из 4-х печатных символов ASCII.
PB	0x5042	Блок дополнения	Используется для обеспечения кратности общей длины всех опциональных блоков длине блока шифрования. Состоит из случайных печатных ASCII символов. Если присутствует, должен быть последним опциональным блоком.
00	0x3030	Статус ключа	Допустимые значения: • 'E' — срок действия истёк (expired) • 'L' — действующий (live) • 'P' — принят на обработку (pending) • 'R' — отозванный (revoked) • 'T' — тестовый (test) Описание допустимых изменений статуса ключа см. ниже.
01	0x3031	Алгоритм шифрования	Алгоритм и режим шифрования, используемые для зашифрования ключа в Key Block. Допустимое значение: '00'
02	0x3032	Алгоритм аутентификации	Алгоритм и режим аутентификации ключа в Key Block. Допустимое значение: '00'
03	0x3033	Дата и время начала действия ключа	В формате ГГГГ:ММ:ДД:ЧЧ.
04	0x3034	Дата и время окончания действия ключа	В формате ГГГГ:ММ:ДД:ЧЧ.
05	0x3035	Текстовая строка	Любая комбинация печатных символов, не может быть нулевой длины.

Порядок включения опциональных блоков в заголовок Key Block произволен, за исключением блока 'PB' (блок дополнения), который должен быть последним блоком в заголовке.

6.4.3.2.2 Пример опционального заголовка

	Блок 1			Блок 2		
Байт	0-1	2-3	4	0-1	2-3	4-10
Значение	00	05	L	PB	0B	rrrrrrr

Первый опциональный блок:

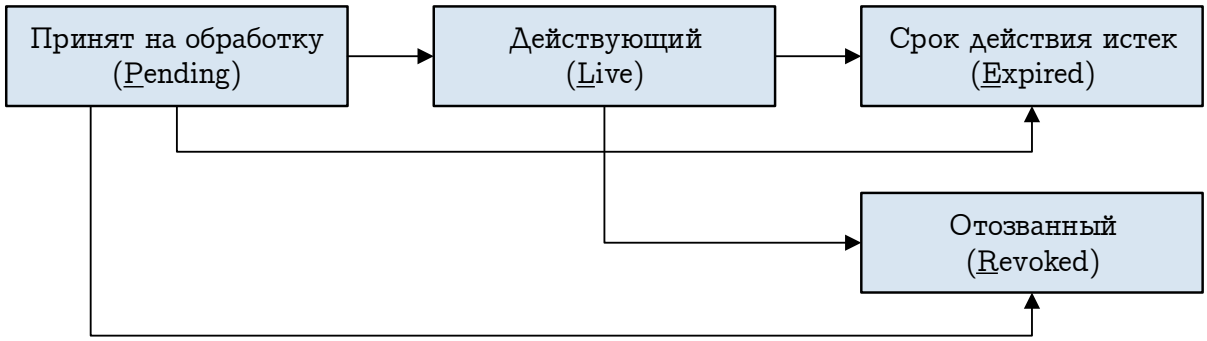
- 00 — идентификатор блока (статус ключа)
- 05 — общая длина блока 5 байтов
- L — данные блока (статус ключа — «действующий»)

Второй опциональный блок:

- PB — идентификатор блока (блок дополнения)
- 0B — общая длина блока 11 байтов
- rrrrrrr — данные блока (случайные ASCII символы заполнения)

В рассмотренном выше случае поле «Количество опциональных блоков» (байты 12-13) заголовка Key Block имеет значение '02' (0x3032). Блок дополнения необходим для обеспечения кратности размеру блока ключа шифрования.

6.4.3.2.3 Изменение статуса ключа Статус ключа может изменяться только в порядке, указанном на следующей схеме. Статус ключа возможно изменить с помощью команды хоста 'CS'. HSM выполняет операции только с ключами со статусом «Действующий» ('L') или «Тестовый» ('T').



6.4.3.3 Зашифрованный ключ (ключевые данные)

В поле «Зашифрованный ключ» Key Block содержится сам защищаемый ключ в зашифрованном виде. Проприетарный Key Block может использоваться для защиты ключей 3DES, AES, HMAC, ECDSA и RSA. Открытые ключи RSA и ECC представлены в незашифрованном виде, однако Key Block аутентифицируется.

HSM поддерживает следующие алгоритмы шифрования ключа:

- при использовании 3DES ключа защищаемые ключевые данные Key Block зашифровываются с помощью алгоритма 3DES в режиме CBC. В качестве вектора инициализации (IV) используются байты 0-7 заголовка Key Block. Ключ шифрования защищаемого ключа является вариантом LMK (или ZMK/TMK).

- при использовании AES ключа защищаемые ключевые данные Key Block зашифровываются с помощью алгоритма AES в режиме CBC. В качестве вектора инициализации (IV) используются байты 0-15 заголовка Key Block. Ключ шифрования защищаемого ключа вырабатывается из LMK (или ZMK/TMK).

Ключевые данные (незашифрованные) имеют следующий формат:

Поле	**Длина**	**Описание**
Длина ключа	2 байта	Длина ключа в следующем поле в битах 16-битное двоичное число Например, если в следующем поле содержится 192-битный 3DES ключ, длина ключа имеет значение 0x00C0
Ключ	переменная, зависит от алгоритма ключа	Ключ в двоичном формате Например, если в поле содержится ключ 3DES (192 бита), он будет представлен 24-байтовым полем
Дополнение	переменная	Случайное дополнение, используемое для обеспечения кратности общей длины ключевых данных Key Block длине блока ключа шифрования ключа Например, если в качестве ключа шифрования используется ключ 3DES, общая длина ключевых данных должна быть кратна 8 байтам. При необходимости поле дополнения может использоваться для маскировки настоящей длины зашифровываемого ключа

6.4.3.4 Аутентификатор

Последним блоком Key Block является блок контроля целостности (Аутентификатор). Он используется для проверки целостности Key Block и вычисляется для заголовка, опциональных блоков и ключевых данных Key Block. При этом признак ключевой схемы (значение 'S') не включается в аутентифицируемые данные.

Алгоритм аутентификации зависит от алгоритма используемого ключа защиты:

• при использовании 3DES ключа значение аутентификатора рассчитывается с помощью алгоритма 3DES CBC-MAC с нулевым IV (дополнение не требуется, так как длина аутентифицируемых данных всегда кратна 8 байтам). В качестве аутентификатора используются крайние левые 4 байта результата вычисления. Ключ аутентификации является вариантом LMK (или ZMK/TMK).

• при использовании AES ключа значение аутентификатора рассчитывается с помощью алгоритма AES CMAC. В качестве аутентификатора используются крайние левые 8 байтов результата вычисления. Ключ аутентификации защищаемого ключа вырабатывается из LMK (или ZMK/TMK) с помощью алгоритма диверсификации.

Глава 7

Использование нескольких LMK

Локальный мастер-ключ (LMK) используется в HSM для криптографической защиты всех рабочих платёжных ключей и дополнительных данных, которые обрабатываются HSM.

В одном HSM можно использовать до 10 LMK одновременно.

LMK хранится в защищённой от несанкционированного доступа памяти HSM и удаляется при обнаружении неавторизованного использования или вскрытия устройства, а также при перезагрузке.

HSM поддерживает 2 типа LMK: Variant LMK и Key Block LMK. Подробное описание типов LMK приведено в гл. 6.

Использование нескольких LMK на одном HSM позволяет упростить процедуру смены LMK. Данную процедуру необходимо тщательно подготовить, чтобы процесс расшифрования всех рабочих ключей, зашифрованных под старым LMK, и повторного зашифрования их под новым LMK не нарушил работу платежных приложений.

Команды управления LMK подробно описаны в «КриптоПро HSM. Команды консоли».

7.1 Генерация компонент LMK

Команда консоли GK используется для генерации компонент LMK и записи их на смарт-карты.

Команда поддерживает генерацию следующих типов ключей:

- Variant LMK — набор ключей 2DES
- Key Block LMK — ключ 3DES или AES-256

```
Offline-AUTH> GK
Variant scheme or key block scheme? [V/K]: K
Enter algorithm type [0 = 3DES, 1 = AES256]: 1
Enter the number of LMK components [1-5]: 3
Enter the number of components required to reconstitute the LMK [1-3]: 3

Unique: 36A41A98
Key check value: 6CB7F0
Choose LMK component for write: [1-3]: 1
Insert blank card and press Enter:
Enter card PIN on the LCD screen.
Writing keys...
Device write complete.
Make another copy? [Y/N]: N
1 copies made.
Choose LMK component for write: [1-3]: 2
```

```
Insert blank card and press Enter:
Enter card PIN on the LCD screen.
Writing keys...
Device write complete.
Make another copy? [Y/N]: N
1 copies made.
Choose LMK component for write: [1-3]: 3
Insert blank card and press Enter:
Enter card PIN on the LCD screen.
Writing keys...
Device write complete.
Make another copy? [Y/N]: N
1 copies made.
Components write complete.
```

7.2 Смена LMK

После загрузки «нового» LMK «старый» ключ может быть загружен в хранилище смены ключей и рабочие платежные ключи перешифрованы.

Для смены LMK используются следующие команды консоли:

- LK — Загрузка компонент LMK;
- V — Проверка загруженных LMK;
- VT — Получение загруженных LMK;
- LN — Загрузка «нового» LMK;
- LO — Загрузка «старого» LMK.

Доступны следующие варианты смены LMK:

Новый LMK / Старый LMK	2DES Variant	3DES Key Block	AES Key Block
2DES Variant	+	+	+
3DES Key Block	–	+	+
AES Key Block	–	–	+

Переход со старого LMK на новый LMK можно осуществлять последовательно: часть данных может работать уже с новым LMK, а часть данных — со старым. Данный механизм также позволяет криптографически изолировать существующие ключи для разных приложений.

7.3 Получение перечня загруженных LMK

С помощью команды VT можно вывести таблицу со списком LMK, установленных в HSM:

```
Online-AUTH> VT
LMK table:
ID Scheme    Algorithm Status  Check
00 KeyBlock  3DES  Live    8E0EC0

Key change storage table:
ID Old/New Scheme    Algorithm   Status  Check

Online-AUTH> VT
```

```
 LMK table:
 ID Scheme    Algorithm   Status  Check

 Key change storage table:
 ID Old/New Scheme    Algorithm   Status  Check
 01 Old     KeyBlock  AES_256     Live    6CB7F0
```

7.4 Удаление LMK

Команда DM удаляет установленный в HSM LMK с указанным идентификатором:

```
Secure-AUTH> DM
Enter LMK id: 9
LMK table:
ID Scheme    Algorithm   Status  Check
09 KeyBlock  AES_256     Live    6CB7F0
Confirm LMK deletion? [Y/N]: Y
LMK deleted from main memory
```

Команда DO удаляет отдельные LMK, которые были загружены в хранилище смены ключей. Можно удалять «новые» и «старые» LMK без удаления текущего LMK.

```
Secure-AUTH> DO
Enter LMK id: 1
Key change storage table:
ID Old/New Scheme    Algorithm   Status  Check
01 Old     KeyBlock  AES_256     Live    6CB7F0
Confirm LMK deletion? [Y/N]: Y
LMK deleted from key change storage
```

7.5 Идентификация LMK

При использовании в HSM нескольких LMK одновременно для идентификации LMK, который необходимо использовать в команде консоли или хоста, используются несколько механизмов.

7.5.1 Команды консоли

В консольных командах необходимый LMK выбирается с помощью идентификатора, который вводится в консоль в процессе выполнения команды.

Например, в команде генерации проверочного значения для ключа, зашифрованного под указанным LMK:

```
Online-AUTH> CK
Enter LMK id: 1
Enter key block: S10096...F420E12116BC8E8A
Key check value: 591B2C
```

7.5.2 Команды хоста

В командах хоста используются несколько механизмов идентификации LMK.

7.5.2.1 Заголовок Key Block

При использовании ключа в формате Key Block байты 14-15 заголовка Key Block определяют номер LMK, который используется для защиты ключа (подробнее см. п. 6.4.3.1.8).

Настройка безопасности `Ignore LMK ID in Key Block Header` позволяет игнорировать идентификатор LMK, указанный в в заголовке Проприетарного Key Block.

По умолчанию настройка настройка выключена и HSM использует для идентификации LMK значение из заголовка Key Block. Если в команде будут присутствовать несколько Key Block с разными идентификаторами LMK, то HSM вернет ошибку.

Если настройка безопасности включена, HSM использует те же механизмы получения идентификатора LMK, что и в случае использования Variant LMK (описаны ниже).

7.5.2.2 Идентификатор LMK по умолчанию

При использовании нескольких LMK одновременно один из них назначается LMK «по умолчанию». Если при получении HSM команды не используются иные механизмы идентификации LMK, HSM считает, что необходимо использовать LMK «по умолчанию».

Идентификатор LMK, который используется по умолчанию, определяется с помощью настройки `Default LMK identifier` (значение настройки по умолчанию — 0).

7.5.2.3 Идентификатор LMK в команде

Идентификатор LMK можно указать явно в команде с использованием 2 дополнительных опциональных полей. Этот метод можно использовать, если в команде нет других полей, идентифицирующих LMK.

Параметр	Формат	Описание
Разделитель	1 A	Значение '%'. Опционально; если присутствует, следующее поле обязательно.
Идентификатор LMK	2 N	Допустимые значения: '00' .. '09'. Присутствует, только если присутствует предыдущее поле.

HSM может вернуть ошибку в следующих случаях несоответствия идентификатора LMK:

- в HSM не загружен LMK с указанным в команде идентификатором
- в заголовке Key Block указан идентификатор, отличный от указанного в команде
- идентификатор в опциональном поле команды отличается от идентификатора LMK, указанного в других полях команды

7.5.2.4 Идентификация по номеру порта

HSM поддерживает возможность идентификации LMK по номеру TCP-порта, на который поступила команда. Для этого во внешнем интерфейсе открывается 11 портов: 1500, 1511-1520.

Порт 1500 является портом по умолчанию. Номера остальных задают «идентификатор LMK по порту» по формуле $LMK_{id} = N_{port} - 1511$. Например, при получении HSM команды на порт 1514 будет использоваться LMK с идентификатором '3'.

Номер TCP-порта	**Идентификатор используемого LMK**
1500	Идентификатор по умолчанию
1511	LMK ID=00
1512	LMK ID=01
...	...
1520	LMK ID=09

Изменить формулу нельзя, однако, если на хост-компьютере для идентификации LMK необходимо использовать другие допустимые номера портов (например, 1501-1510), можно воспользоваться технологией перенаправления сетевых портов (Port Forwarding) или утилитой stunnel, входящей в комплектацию ПАКМ, сконфигурировав ее параметры для переадресации портов.

Если в команде в опциональном поле явно указан идентификатор LMK, то выбирается он, в противном случае, если команда передана на порт по умолчанию (1500), выбирается идентификатор LMK по умолчанию, иначе — идентификатор по номеру порту.

Если в результате указанной выше проверки для команды выбран идентификатор LMK по умолчанию, в нее включены Key Block, и настройка `Ignore LMK ID in Key Block Header` имеет значение 'No', то идентификатор ключа LMK будет взят из заголовка Key Block.

Глава 8

Трансляция данных при смене LMK

При смене LMK для продолжения корректной работы платежных приложений необходима трансляция защищаемой с помощью LMK информации — расшифрование платежных ключей и других данных, зашифрованных под «старым» LMK, и повторное зашифрование под «новым» LMK.

Смена LMK и сопутствующая трансляция данных могут выполняться, например, планово по требованиям регуляторов и других участников платежных систем, а также в случае перехода с Variant LMK на Key Block LMK.

HSM поддерживает одновременное хранение нескольких LMK, что позволяет разделить приложения, клиентов и т.п., обслуживаемых одним HSM, а также облегчает процесс смены LMK (подробнее см. гл. 7).

8.1 Описание процесса

Для перещифрования рабочих платежных ключей и данных при смене LMK необходимо, чтобы «старый» LMK и «новый» LMK были загружены в HSM.

В памяти HSM выделяется 2 логических хранилища LMK:

1. основное

Обработка транзакций и различные операции с LMK и защищаемыми данными доступны только для LMK, хранящихся в основном хранилище

2. хранилище смены ключей

Ключи в этом хранилище могут использоваться только в процессе смены LMK

Существует два способа загрузки «старого» и «нового» LMK в хранилища:

1. «Новый» LMK загружается в основное хранилище с помощью консольной команды LK, а «старый» LMK (который все еще используется для работы) загружается в хранилище смены ключей в качестве «старого» с помощью консольной команды LO. Таким образом, HSM не может выполнять операции со «старым» LMK, но готов обрабатывать транзакции сразу после ввода в эксплуатацию «нового» LMK.

2. «Старый» LMK (который все еще используется для работы) остается в основном хранилище, а «новый» LMK загружается в хранилище смены ключей в качестве «нового» с помощью консольной команды LN. Таким образом, HSM может выполнять операции со «старым» LMK во время смены ключей. По окончании трансляции данных «новый» LMK должен быть перемещен в основное хранилище.

Смена LMK включает следующие шаги:

1. Сгенерировать «новый» LMK в виде ключевых компонент и записать их на смарт-карты;

2. Загрузить «новый» LMK (со смарт-карт) в основное хранилище, загрузить «старый» LMK (со смарт-карт) в хранилище смены ключей;

ИЛИ

Оставить «старый» LMK в основном хранилище и загрузить «новый» LMK (со смарт-карт) в хранилище смены ключей.

3. Выполнить трансляцию (перешифрование) операционных ключей со «старого» на «новый» LMK и разместить их в новой базе ключей;

4. Выполнить трансляцию (перешифрование) PIN со «старого» на «новый» LMK и разместить в новой базе PIN;

5. Выполнить трансляцию (перешифрование) таблиц децимилизации со «старого» на «новый» LMK и разместить в новой базе таблиц децимализации;

6. Если «новый» LMK был загружен в хранилище смены ключей, то перенести его в основное хранилище;

7. Назначить новые базы ключей/PIN/таблиц децимализации текущими.

8.2 Генерация нового LMK

LMK генерируется в HSM в виде ключевых компонент, которые записываются на смарт-карты. Смарт-карты с компонентами LMK распределяются между привилегированными пользователями ПАКМ.

8.2.1 Форматирование смарт-карт

Перед записью компоненты LMK смарт-карта должна быть отформатирована. Смарт-карты, которые использовались для хранения неактуальных более ключевых компонент, могут использоваться повторно после форматирования.

Форматирование смарт-карт осуществляется с помощью консольной команды FC:

```
Secure-AUTH> FC
Insert card and press Enter:
Enter card PIN on the LCD screen.
Clearing card...
Card cleared.
```

Примечание. Не форматируйте смарт-карты с компонентами «старого» LMK до полного завершения процесса смены LMK.

Порядок хранения и использования смарт-карт (ключевых носителей) с компонентой LMK описан в «КриптоПро HSM. Правила пользования».

8.2.2 Генерация компонент LMK

Генерация компонент LMK и запись их на смарт-карты осуществляется с помощью консольной команды GK. При выполнении данной команды указываются:

- число компонент, на которые будет разделен ключ (n);
- число компонент, достаточных для сбора ключа (k).

```
Online-AUTH> GK
Variant scheme or key block scheme? [V/K]: K
Enter algorithm type [0 = 3DES, 1 = AES256]: 1
Enter the number of LMK components [1-5]: 3
Enter the number of components required to reconstitute the LMK [1-3]: 3
Unique: 36A41A98
Key check value: 6CB7F0
Choose LMK component for write: [1-3]: 1
```

```
Insert blank card and press Enter:
Enter card PIN on the LCD screen.
Writing keys...
Device write complete.
Make another copy? [Y/N]: N
1 copy made.
...
Choose LMK component for write: [1-3]: 3
Insert blank card and press Enter:
Enter card PIN on the LCD screen.
Writing keys...
Device write complete.
Make another copy? [Y/N]: N
1 copy made.
Components write complete.
```

8.2.3 Создание копии компоненты (смарт-карты)

Копии компонент можно сделать сразу при их генерации во время выполнения команды GK (см. выше).

Копии сгенерированных ранее компонент можно сделать с помощью консольной команды DC:

```
Secure-AUTH> DC
Insert card to be duplicated and press Enter:
Enter card PIN on the LCD screen.
Reading key...
Container with LMK has been opened. Part Check = E65D4C
Insert blank card and press Enter:
Enter card PIN on the LCD screen.
Writing key...
New copy of LMK component has been written. Part Check = E65D4C
Make another copy? [Y/N]: n
```

8.3 Загрузка «нового» LMK

Для загрузки «нового» LMK необходимо присутствие держателей смарт-карт с компонентами LMK.

LMK формируется в оперативной памяти HSM при предъявлении любых k из n смарт-карт с компонентами ключа.

«Новый» LMK должен быть загружен либо в основное хранилище с помощью консольной команды LK (способ 1 разд. 8.1), либо в хранилище смены ключей в качестве «нового» с помощью консольной команды LN (способ 2 разд. 8.1).

Для загрузки LMK (с помощью команды LK или LN) консоль HSM должна быть запущена в режиме secure.

```
Secure-AUTH> LK
Enter LMK id: 9
Load LMK from components
Insert card and press Enter:
Enter card PIN on the LCD screen.
LMK table:
ID Scheme Algorithm Status Check
09 KeyBlock AES_256 Live 000000
Component has been loaded!
Insert card and press Enter:
```

```
Enter card PIN on the LCD screen.
Component has been loaded!
...
LMK table:
ID Scheme Algorithm Status Check
09 KeyBlock AES_256 Live 6CB7F0
Warning: All Test LMKs will be erased! Confirm details? [Y/N]: Y
LMK loaded
```

8.4 Загрузка «старого» LMK

Для смены LMK наряду с «новым» LMK в HSM должен быть загружен и «старый».

«Старый» LMK, если он используется в текущий момент, может быть уже загружен в основное хранилище, либо он загружается в хранилище смены ключей в качестве «старого» с помощью консольной команды LO.

Для загрузки LMK (с помощью команды LO) консоль HSM должна быть запущена в режиме secure.

```
Secure-AUTH> LO
Enter LMK id: 1
Load LMK from components
Insert card and press Enter:
Enter card PIN on the LCD screen.
LMK table:
ID Scheme    Algorithm   Status  Check
01 KeyBlock  AES_256     Live    000000
Component has been loaded!
Insert card and press Enter:
Enter card PIN on the LCD screen.
Component has been loaded!
Insert card and press Enter:
Enter card PIN on the LCD screen.
Component has been loaded!
LMK table:
ID Scheme    Algorithm   Status  Check
01 KeyBlock  AES_256     Live    6CB7F0
Confirm details? [Y/N]: Y
LMK loaded
```

8.5 Трансляция ключей

После того, как в HSM загружены «старый» и «новый» LMK, можно выполнять трансляцию (перешифрование) данных, в первую очередь рабочих платежных ключей.

Трансляция ключей при смене LMK выполняется с помощью команды хоста 'BW'.

Для этого со стороны хост-системы для каждого рабочего платежного ключа необходимо:

- Извлечь ключ, зашифрованный под «старым» LMK, из старой базы ключей;
- Отправить на HSM команду трансляции ключей 'BW', включающую ключ, зашифрованный под «старым» LMK;
- Получить от HSM ответ на команду 'BX', содержащий ключ, зашифрованный под «новым» LMK;
- Поместить ключ, зашифрованный под «новым» LMK, в новую базу ключей.

8.6 Трансляция PIN

PIN, которые хранятся зашифрованными под «старым» LMK, при смене LMK также необходимо перешифровать.

Для этого можно использовать команду 'BG', которая позволяет расшифровать PIN, зашифрованный под «старым» LMK, и зашифровать его под «новым» LMK.

При смене LMK приложению хоста необходимо проделать процедуру, аналогичную трансляции ключей (см. выше): извлечь каждый PIN, зашифрованный под «старым» LMK, из старой базы PIN, перешифровать его с помощью команды 'BG' и записать PIN, зашифрованный под «новым» LMK, в новую базу PIN.

8.7 Трансляция таблиц децимилизации

Для обеспечения высокого уровня безопасности рекомендуется использование зашифрованных таблиц децимализации (значение настройки `Decimalization tables` по умолчанию), поэтому при смене LMK также может потребоваться их трансляция.

Трансляция таблиц децимализации выполняется с помощью команды хоста 'LO'. Команда предназначена для упрощения процесса смены LMK для эмитентов или обработчиков транзакций, которые используют большое количество таблиц децимализации.

При смене LMK приложению хоста необходимо проделать процедуру, аналогичную трансляции PIN (см. выше): извлечь каждую таблицу, зашифрованную под «старым» LMK, из старой базы таблиц децимализации, перешифровать ее с помощью команды 'LO' и записать таблицу, зашифрованную под «новым» LMK, в новую базу таблиц децимализации.

8.8 Переход на «новый» LMK

К моменту перехода на новый LMK система находится в следующем состоянии:

• старые базы с рабочими платежными ключами, PIN и таблицами децимализации, зашифрованными под «старым» LMK, еще используются

• новые базы с рабочими платежными ключами, PIN и таблицами децимализации, зашифрованными под «новым» LMK, уже подготовлены, но еще не используются

• один или несколько HSM выделены для переноса ключей

Это могут быть HSM со «старым» LMK (который все еще используется в других HSM в качестве текущего), загруженным в хранилище смены ключей (с помощью консольной команды LO), и «новым» LMK в основном хранилище.

В этом случае также эксплуатируются другие HSM со «старым» LMK в основном хранилище, которые выполняют операции с использованием ключей, PIN и таблиц децимализации из старых баз.

Для начала использования нового LMK необходимо синхронизировать и выполнить следующие действия:

• приложения хоста начинают использовать новые базы ключей/PIN/таблиц децимализации;

• если «новый» LMK загружен в основное хранилище, то HSM готов обрабатывать транзакции с использованием «нового» LMK.

В другие HSM, которые обрабатывали транзакции с использованием «старого» LMK, необходимо загрузить «новый» LMK в текущее хранилище.

• если трансляция данных выполнялась на HSM, на котором «новый» LMK был загружен в хранилище смены ключей, «новый» LMK должен быть загружен в основное хранилище всех используемых HSM.

Чтобы избежать прерывания процесса обработки транзакций, возможно применять постепенный переход со «старого» на «новый» LMK, при этом приложения хоста должны контролировать, какой LMK

использует в текущий момент HSM, к которому они обращаются, и извлекать зашифрованные ключи или данные из соответствующей базы (старой или новой).

8.8.1 Использование нескольких LMK

HSM поддерживает возможность использования до 10 LMK одновременно, установленных в основном хранилище оперативной памяти. Использование нескольких LMK может упростить процесс смены LMK.

Ниже приведен пример использования нескольких LMK в случае процесса смены LMK, при котором «старый» (все еще используемый) LMK установлен в хранилище смены ключей, а «новый» LMK (на который планируется переход) — в основное хранилище.

1. Текущий LMK, который используется в настоящий момент, располагается в основном хранилище с идентификатором ID = 0.

2. Будущий LMK («новый») также загружается в основное хранилище с идентификатором ID = 1.

3. Текущий LMK загружается в хранилище смены ключей как «старый».

4. Выполняется трансляция данных (ключей, PIN и таблиц децимализации), зашифрованных под «старым» LMK (который все еще используется), с зашифрованием их под «новым» LMK. Для этого в командах трансляции 'BW', 'BG', 'LO' явно указывается идентификатор «нового» LMK (01).

5. После перешифрования всех ключей, PIN и таблиц децимализации приложение хоста может использовать новые базы с перешифрованными данными, если:

• «Новый» LMK повторно загружается в HSM, но теперь с идентификатором ID = 0

ИЛИ

• Команды, посылаемые на HSM, предписывают использовать «новый» LMK с идентификатором ID = 1 путем указания явного значения идентификатора в команде (подробнее см. Идентификатор LMK в команде) или направлении команды на соответствующий TCP-порт (подробнее см. Идентификация по номеру порта)

8.9 Удаление неиспользуемых LMK

После завершения процесса смены LMK рекомендуется удалить из HSM неиспользуемый «старый» или «новый» LMK, загруженный в хранилище смены ключей.

Это можно сделать, например, с помощью консольной команды DO:

```
Secure-AUTH> DO
Enter LMK id: 1
Key change storage table:
ID Old/New Scheme    Algorithm   Status  Check
01 Old     KeyBlock  AES_256     Live    6CB7F0
Confirm LMK deletion? [Y/N]: Y
LMK deleted from key change storage
```

Также удалить LMK из хранилища смены ключей можно с помощью команды хоста 'BS', указав его идентификатор в поле *Идентификатор LMK*.

Кроме того, в случае смены LMK путем загрузки нескольких LMK в основное хранилище (см. разд. 8.8.1) «новый» LMK должен быть удален из временного расположения (в основном хранилище с идентификатором ID = 1) с помощью консольной команды DM.

Глава 9

Поддержка TR-31 Key Block

В системах управления платежами участвует несколько сторон, которые могут использовать разные типы HSM и, соответственно, LMK. В процессе их взаимодействия регулярно необходимо экспортировать операционные ключи из одной системы и импортировать в другую.

HSM поддерживает использование ключей в формате Key Block в соответствии с ANSI X9.24 TR-31, а также в устаревшем формате, описанном в ANSI X9.17.

9.1 Структура TR-31 Key Block

Ключевая схема	Заголовок Key Block	Опциональный заголовок	Зашифрованный ключ	Аутентификатор
1 байт	16 байт	переменная длина	переменная длина	8 или 16 байт
значение 'R'	определяет область использования ключа, режим использования, алгоритм и ограничения на экспорт ключа	дополнительные характеристики ключа	ключ, зашифрованный под ZMK/TMK	индикатор несанкционированного изменения Key Block, вычисленный с помощью ZMK/TMK

9.2 Заголовок Key Block

Заголовок Key Block определяет, как может использоваться ключ в Key Block. Допустимые значения полей заголовка Key Block, которые используется в командах хоста, строго ограничены. При отправке в команде недопустимого значения заголовка HSM вернет ошибку и команда не будет обработана.

Формат заголовка Key Block:

Байт(ы)	**Поле**	**Определение**
0	Идентификатор версии (Version ID)	Определяет формат хранения Key Block
1-4	Длина Key Block	Общая длина Key Block
5-6	Использование ключа (Key Usage)	Например, шифрование ключей или шифрование данных
7	Алгоритм	Например, 3DES

8	Режим использования (Mode of Use)	Например, только для шифрования
9-10	Номер версии ключа (Key Version Number)	Например, версия ключа в Key Block
11	Экспортируемость	Например, экспорт не разрешен
12-13	Количество опциональных блоков	Количество опциональных блоков заголовка
14-15	Зарезервировано	Значение '00'

Примечание: актуальный перечень допустимых значений полей заголовка Key Block содержится в последней версии стандарта TR-31.

9.2.1 Идентификатор версии (Байт 0)

Идентификатор версии (Version ID) Key Block определяет формат Key Block и метод, используемый для его защиты:

Значение	**Определение**
'A'	Определён в TR-31:2005 Key Block, защищённый методом Key Variant Binding
'B'	Определён в TR-31:2010 Key Block, защищённый методом Key Derivation Binding
'C'	Определён в TR-31:2010 Key Block, защищённый методом Key Variant Binding
'D'	Определён в TR-31:2018 Key Block, защищённый методом AES Key Derivation Binding

9.2.2 Длина Key Block (Байты 1-4)

Поле содержит длину всего Key Block (заголовка, опциональных полей, зашифрованного ключа и аутентификатора). Значение длины Key Block вычисляется после кодирования и представляется 4-значным набором ASCII символов.

Например, если общая длина Key Block составляет 112 символов (байтов), значение 1-го байта равно '0', значение 2-го байта равно '1', значение 3-го байта равно '1', значение 4-го байта равно '2' (т.е. 0x30313132).

9.2.3 Использование ключа (Байты 5-6)

Значение поля определяет назначение ключа в Key Block. В следующей таблице представлены допустимые значения Использования ключа для TR-31 Key Block и соответствующие значения этого поля для Проприетарного Key Block:

Значение (TR-31 KB)	**Описание**	**Значение (Проприетарный KB)**
B0	Базовый ключ диверсификации DUKPT (DUKPT Base Derivation Key, BDK-1)	B0, 41, 42, 43
B1	Начальный ключ DUKPT (DUKPT Initial Key, IK/IKEY/IPEK)	B1
C0	Ключ проверки карты (Card Verifikation Key, CVK)	C0, 11, 12, 13
D0	Ключ шифрования данных (универсальный)	D0, 21, 22, 23, 25
E0	Мастер-ключ EMV/Chip card для генерации и проверки Application Cryptogram (MK_{AC})	E0
E1	Мастер-ключ EMV/Chip card для обеспечения конфиденциальности (secure messaging) (MK_{SMC})	E1
E2	Мастер-ключ EMV/Chip card для обеспечения целостности (secure messaging) (MK_{SMI})	E2
E3	Мастер-ключ EMV/Chip card для вычисления и проверки Data Authentication Code (MK_{DAC})	E3
E4	Мастер-ключ EMV/Chip card для генерации Dynamic Numbers (MK_{DN})	E4
E5	Мастер-ключ EMV/Chip card для персонализации карт	E5
E6	Мастер-ключ EMV/Chip card (другое)	E6, 31, 32
E7	Ключ персонализации EMV	E7
K0	Ключ шифрования ключа/key wrapping (универсальный)	K0, 51, 52, 54
K1	Ключ шифрования ключа/key wrapping (универсальный)	K1
K3	Асимметричный ключ (ECC) обмена	03
M0	ISO 16609 MAC algorithm 1 (с использованием 3DES)	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
M4	ISO 9797-1 MAC algorithm 4	M4
M5	ISO 9797-1:1999 MAC algorithm 5/CBC-MAC	M5
M6	ISO 9797-1:2011 MAC algorithm 5/CMAC	M6
P0	Ключ шифрования PIN (универсальный)	P0, 71, 72, 73
S0	Асимметричный ключ (ECC) подписи	03
V0	Ключ проверки PIN (универсальный)	V0
V1	Ключ проверки PIN (IBM 3624)	V1
V2	Ключ проверки PIN (ABA PVV)	V2
03	Асимметричный ключ (ECC) общего назначения	03

Соответствие значений поля Использование ключа при экспорте ключа из формата Проприетарного Key Block в формат TR-31 Key Block приведено в Приложении Б.

9.2.4 Алгоритм (Байт 7)

Поле определяет криптографический алгоритм ключа в Key Block.

Значение	Hex	Алгоритм
A	0x41	AES
E	0x45	ECC
H	0x48	HMAC
R	0x52	RSA
T	0x54	3DES

9.2.5 Режим использования (Байт 8)

Поле определяет доступные операции для ключа в Key Block.

Значение	Hex	Описание
B	0x42	Зашифрование и расшифрование
C	0x43	Вычисление MAC (генерация или проверка)
D	0x44	Расшифрование
E	0x45	Зашифрование
G	0x47	Генерация MAC
N	0x4E	Ограничений нет
S	0x53	Генерация ЭП
V	0x56	Проверка ЭП или MAC
X	0x58	Диверсификация ключей

9.2.6 Номер версии ключа (Байты 9-10)

Поле определяет номер версии ключа в Key Block.

9.2.7 Экспортируемость (Байт 11)

Данный параметр определяет условия, при которых ключ, содержащийся в Key Block, может быть экспортирован. Ключ считается доверенным, если он представлен в формате Key Block. Другие форматы ключей считаются недоверенными.

Значение	Описание
E	Допускается экспорт только в доверенный формат, при этом wrapping key также должен быть представлен в одном из этих форматов
N	Экспорт ключа не допускается
S	Разрешен экспорт в любые форматы (при условии, что они включены в настройках безопасности)

9.2.8 Количество опциональных блоков (Байты 12-13)

Определенный в TR-31 формат позволяет включать в заголовок Key Block до 99 опциональных блоков (Optional Header Blocks), которые содержат дополнительные опциональные данные о Key Block.

Байты 12-13 определяют количество опциональных блоков заголовка в Key Block. Например, значение '12' (0x3132) означает, что Key Block содержит 12 опциональных блоков; значение '00' (0x3030) указывает на отсутствие опциональных блоков.

9.2.9 Пример заголовка TR-31 Key Block

Байт	0	1-4	5-6	7	8	9-10	11	12-13	14-15
Значение	A	0072	**V2**	T	G	22	N	00	00

- идентификатор версии A
- длина Key Block 72 байта (0072)
- ключ может использоваться только с методом проверки PIN ABA PVV (**V2**)
- алгоритм 3DES (T)
- ключ может использоваться только для генерации PVV (G)
- версия ключа 22
- ключ нельзя экспортировать (N)
- опциональные блоки заголовка отсутствуют (00)

9.3 Опциональный заголовок

Опциональный заголовок состоит из нескольких **опциональных блоков**, которые имеют следующую структуру:

Байты	**Поле**	**Комментарий**
0-1	Идентификатор блока	Идентификатор опционального блока
2-3	Длина блока	Длина опционального блока в байтах (шестнадцатеричное)
4-n	Данные блока	Данные опционального блока

В настоящее время в TR-31 определены 3 типа опциональных блоков:

Идентификатор блока	**Описание**
KS	Идентификатор ключевого набора (подробнее см. ANSI X9.24-2004 Part 1, Annex E)
KV	Версия набора значений полей Key Block
PB	Блок дополнения (padding), для обеспечения кратности общей длины всех опциональных блоков длине блока шифрования

Цифровое значение	Проприетарные идентификаторы. В случае использования владелец приложения хоста должен гарантировать поддержку используемых идентификаторов всеми устройствами.

9.4 Использование TR-31 Key Block

Ключи в формате TR-31 Key Block используются в командах импорта и экспорта ключей. Как правило, они заменяют ключи в формате X9.17.

Ключи в формате TR-31 Key Block обозначаются с помощью идентификатора ключевой схемы 'R'.

Глава 10

Алгоритмы шифрования

10.1 DES

HSM поддерживает работу с ключами алгоритмов 2DES и 3DES.

Ключи, защищенные с использованием Variant LMK, имеют следующий формат в командах хоста:

- 2DES — 1 A + 32 H
- 3DES — 1 A + 48 H

Здесь 1 A — префикс с символом ключевой схемы (признак схемы), который используется для определения длины и способа представления ключа в зашифрованном виде. Допустимые значения см. в Приложении B.

HSM не поддерживает ключи 2DES и 3DES, защищенные с использованием Variant LMK, без признака схемы.

Ключи 2DES и 3DES могут быть представлены в формате Key Block (значения признака схемы см. в Приложении B).

При работе с DES ключами HSM осуществляет проверку, является ли ключ слабым. Если ключ генерируется самим HSM, формирование слабых ключей исключается. Если полученный в команде хоста ключ после расшифровывания и проверки оказывается слабым, HSM вернет код ошибки 50 (нулевой или слабый ключ).

10.2 AES

HSM поддерживает работу с ключа AES длины 128, 192 и 256 бит.

Для защиты AES ключей могут использоваться только AES Key Block LMK с представлением ключей в формате Проприетарного Key Block.

Перечень поддерживаемых HSM платежных ключей, в качестве которых могут использоваться AES ключи, см. в таблице на стр. 45 (значение **AES** в столбце *Алгоритм*).

При необходимости использования AES ключей в системах, не используемых такие ключи ранее, требуется установить в HSM AES Key Block LMK. В этом случае можно использовать дополнительный LMK в основном хранилище, используемый только для защиты новых платежных ключей, включая AES ключи (без перешифрования старых платежных ключей, зашифрованных под текущим(и) LMK), или выполнить смену LMK (подробнее см. гл. 8) и соответствующую трансляцию всех данных. При этом необходимо помнить, что:

- использование AES Key Block LMK не ограничивается лишь защитой AES ключей, любые платежные ключи и другие защищаемые HSM данные могут быть зашифрованы с его помощью;

• для защиты AES ключей может использоваться только AES Key Block LMK (и не могут Variant LMK и 3DES Key Block LMK);

• трансляция ключей и других данных поддерживается при смене любого типа LMK на AES Key Block LMK;

• ключи, зашифрованные под AES Key Block LMK, в будущем могут быть перешифрованы только под другой AES Key Block LMK.

Глава 11

Форматы PIN-блоков

Для корректной обработки (в том числе проверки и трансляции) PIN необходимо, чтобы на HSM они передавались в виде зашифрованного 16-значного PIN-блока. В открытом виде PIN-блок состоит из PIN и данных дополнения, обеспечивающих необходимую длину в 16 символов. В зависимости от используемого механизма дополнения (padding) различают несколько форматов PIN-блоков. Каждый формат PIN-блока идентифицируется двузначным кодом. Форматы 34, 35, 41 и 42 используются для EMV операций смены PIN и доступны только для команд хоста KU и KY.

Для соответствия требованиям PCI HSM использование некоторых форматов PIN-блока ограничено (с помощью настройки безопасности `Restrict PIN block usage for PCI compliance`).

11.1 Трансляция PIN-блоков

Трансляция PIN-блоков включает расшифрование PIN-блока, переданного в зашифрованном виде на HSM, перевод PIN-блока из одного формата в другой и зашифрование PIN-блока под целевым ключом для дальнейшего использования хостом.

11.1.1 Команды трансляции PIN

Для трансляции PIN используются следующие команды хоста:

- BG — Трансляция PIN и длины PIN
- CA — Трансляция PIN (из-под TPK под ZPK/BDK(3DES DUKPT))
- CC — Трансляция PIN (из-под ZPK под ZPK)
- G0 — Трансляция PIN (из-под BDK под BDK/ZPK (3DES и AES DUKPT))
- JC — Трансляция PIN (из-под TPK под LMK)
- JE — Трансляция PIN (из-под ZPK под LMK)
- JG — Трансляция PIN (из-под LMK под ZPK)

11.1.2 Таблица трансляции PIN-блоков

11.1.2.1 Без соответствия PCI HSM

Если выставлена настройка `Restrict PIN block usage for PCI compliance: No`, трансляция PIN-блоков возможна в соответствии со следующей таблицей:

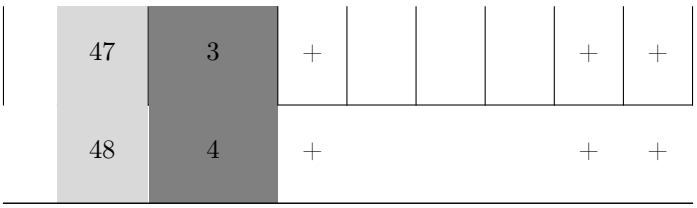
			Трансляция в формат					
	Код формата PIN-блока		01	03	05	34*	47	48
		ISO формат	0	–	1	2	3	4
Трансляция из формата	01	0	+	+	+	+	+	+
	03	–	+	+	+	+	+	+
	05	1	+	+	+	+	+	+
	47	3	+	+	+	+	+	+
	48	4	+	+	+	+	+	+

* Трансляция в формат PIN-блока 34 разрешена, только если выставлена настройка безопасности `Enable PIN Block Format 34 as output format for PIN translations to ZPK: Yes`.

11.1.2.2 Соответствие PCI HSM

Если выставлена настройка `Restrict PIN block usage for PCI compliance: Yes`, трансляция PIN-блоков возможна в соответствии со следующей таблицей:

			Трансляция в формат					
	Код формата PIN-блока		01	03	05	34	47	48
		ISO формат	0	–	1	2	3	4
Трансляция из формата	01	0	+				+	+
	03	–	+	+	+		+	+
	05	1	+		+	+	+	+



11.2 Форматы PIN-блоков

В следующей таблице описаны поддерживаемые HSM форматы PIN-блоков:

Код формата PIN-блока	**ISO формат**	**Описание**	**Алгоритм**	**Включен по умолчанию***
01	0	ISO 9564-1 и ANSI X9.8 формат 0	3DES	+
03	–	Diebold & IBM ATM	3DES	–
05	1	ISO 9564-1 формат 1	3DES	+
34	2	Standard EMV 1996	3DES	–
35	–	Mastercard Pay Now и Pay Later	3DES	+
41	–	Visa/Amex new PIN only	3DES	+
42	–	Visa/Amex new & old PIN	3DES	+
47	3	ISO 9564-1 и ANSI X9.8 формат 3	3DES	+
48	4	ISO 9564-1 формат 4	AES	+

* Перечень включенных в данный момент на HSM форматов PIN-блоков, доступных для использования в командах обработки PIN, можно получить с помощью консольной команды CONFIGPB (см. «КриптоПро HSM. Команды консоли»).

11.2.1 Формат 01

Код формата PIN-блока 01 используется для идентификации PIN-блоков формата ISO 9564-1 формат 0, эквивалентного ANSI X9.8 формат 0. Для зашифрования PIN-блока может использоваться только ключ 3DES.

PIN-блок формата 01 формируется следующим образом:

1. формируется первый 16-значный блок P1:

P1:	0	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

N — длина PIN (от 0x4 до 0xC)

P — цифра PIN, допустимые значения от 0x0 до 0x9 (десятичные цифры)

F — дополнение, значение 0xF

Например, значение P1 для 5-значного PIN 92389: 0592 389F FFFF FFFF

2. формируется второй 16-значный блок P2:

P2:	0	0	0	0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}

0 — дополнение, значение 0x0

$A_1..A_{12}$ — 12 крайних правых цифр номера счета (PAN), за исключением контрольной цифры, допустимые значения от 0x0 до 0x9 (десятичные цифры). Если PAN без контрольной цифры меньше 12 цифр, то цифры PAN выравниваются по правому краю и дополняются нулями слева.

Например, значение P2 для 13-значного номера счета 4000 0012 3456 2 с контрольной цифрой 2: 0000 4000 0012 3456

3. значение PIN-блока PB = P1 $\oplus$ P2

Например:

P1:	05	92	38	9F	FF	FF	FF	FF
P2:	00	00	40	00	00	12	34	56
PB:	05	92	78	9F	FF	ED	CB	A9

11.2.2 Формат 03

Код формата PIN-блока 03 используется для поддержки работы с банкоматами Diebold и IBM. Для зашифрования PIN-блока может использоваться только ключ 3DES.

16-значный PIN-блок формируется из значения PIN, дополненного справа шестнадцатеричными символами 0xF.

Например, значение PIN-блока для PIN 92389: 9238 9FFF FFFF FFFF

11.2.3 Формат 05

Код формата PIN-блока 05 используется для идентификации PIN-блоков формата ISO 9564-1 формат 1. Для зашифрования PIN-блока может использоваться только ключ 3DES.

16-значный PIN-блок формата 05 формируется следующим образом:

1	N	P	P	P	P	P/R	P/R	P/R	P/R	P/R	P/R	P/R	P/R	R	R

N — длина PIN (от 0x4 до 0xC)

P — цифра PIN, допустимые значения от 0x0 до 0x9 (десятичные цифры)

R — случайная цифра дополнения

Для входящих PIN-блоков формата 05 выполняются следующие проверки:

• Первый символ PIN-блока должен быть равен 1, в противном случае возвращается код ошибки 20.

• Цифры PIN (символы 3-(N+2) в PIN-блоке) должны являться десятичными цифрами (0..9), в противном случае возвращается код ошибки 20.

• Второй символ PIN-блока (N) должен являться шестнадцатеричным символом в интервале от 0x4 до 0xC, в противном случае возвращается код ошибки 24.

11.2.4 Формат 34

Код формата PIN-блока 34 используется для идентификации PIN-блоков формата ISO 9564-1 формат 2. Для зашифрования PIN-блока может использоваться только ключ 3DES.

Использование PIN-блока формата 34 ограничено настройкой безопасности `Enable PIN Block Format 34 as output format for PIN translations to ZPK`:

• если настройка имеет значение `No` (по умолчанию), данный формат PIN-блоков доступен только в ответах команд хоста KU и KY;

• если настройка имеет значение `Yes`, данный формат PIN-блоков доступен в ответах команд хоста KU и KY и команд трансляции PIN CA, CC, G0.

16-значный PIN-блок формата 34 формируется следующим образом:

C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

C — контрольное поле со значением 0x2

N — длина PIN, допустимые значения от 0x4 до 0xC

P — цифра PIN, допустимые значения от 0x0 до 0x9 (десятичные цифры)

F — дополнение, значение 0xF

Например, значение PIN-блока для PIN 34567: **2**534567FFFFFFFFF

11.2.5 Формат 35

Код формата PIN-блока 35 используется для идентификации PIN-блоков, требуемых Europay/Mastercard для Pay Now & Pay Later. Для зашифрования PIN-блока может использоваться только ключ 3DES. Данный формат PIN-блоков доступен только в ответах команд хоста KU и KY.

PIN-блок формата 35 формируется следующим образом:

1. формируется первый 16-значный блок P1:

P1:	C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

C — контрольное поле со значением 0x2

N — длина PIN, допустимые значения от 0x4 до 0xC

P — цифра PIN, допустимые значения от 0x0 до 0x9 (десятичные цифры)

F — дополнение, значение 0xF

Например, значение P1 для PIN 34567: **2**534 567F FFFF FFFF

2. формируется второй 16-значный блок P2:

P2:	0	0	0	0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}

0 — дополнение, значение 0x0

A_1..A_{12} — 12 крайних правых цифр номера счета (PAN), за исключением контрольной цифры, допустимые значения от 0x0 до 0x9 (десятичные цифры). Если PAN без контрольной цифры меньше 12 цифр, то цифры PAN выравниваются по правому краю и дополняются нулями слева.

Например, значение P2 для номера счета 1234 0000 0123 456 2, где 2 является контрольной цифрой:

0000 4000 0012 3456

3. значение PIN-блока PB = P1 $\oplus$ P2:

P1:	25	34	56	7F	FF	FF	FF	FF
P2:	00	00	40	00	00	12	34	56
PB:	25	34	16	7F	FF	ED	CB	A9

11.2.6 Формат 41

Код формата PIN-блока 41 используется для идентификации формата Visa для изменения PIN без использования текущего PIN. Для зашифрования PIN-блока может использоваться только ключ 3DES. Данный формат PIN-блоков доступен только в ответах команд хоста KU и KY.

PIN-блок формата 41 формируется с использованием нового PIN и части ключа Unique DEA Key A (в HSM ключ обозначается DK-AC и вырабатывается из MK-AC) следующим образом:

1. формируется первый 16-значный блок P1, состоящий из 8 крайних правых цифр ключа DK-AC, дополненных слева 8 нулями (0x0):

P1:	0	0	0	0	0	0	0	0								
	8 нулей								8 крайних правых цифр DK-AC							

2. формируется второй 16-значный блок P2:

P2:	C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

C — контрольное поле со значением 0x2

N — длина PIN, допустимые значения от 0x4 до 0xC

P — цифра PIN, допустимые значения от 0x0 до 0x9 (десятичные цифры)

F — дополнение, значение 0xF

3. значение PIN-блока PB = P1 $\oplus$ P2.

11.2.7 Формат 42

Код формата PIN-блока 42 используется для идентификации формата Visa для изменения PIN с использованием текущего (старого) PIN. Для зашифрования PIN-блока может использоваться только ключ 3DES. Данный формат PIN-блоков доступен только в ответах команд хоста KU и KY.

PIN-блок формата 42 формируется с использованием старого PIN, нового PIN и части ключа Unique DEA Key A (в HSM ключ обозначается DK-AC и вырабатывается из MK-AC) следующим образом:

1. формируется первый 16-значный блок P1, состоящий из 8 крайних правых цифр ключа DK-AC, дополненных слева 8 нулями (0x0):

P1:	0	0	0	0	0	0	0	0								
	8 нулей								8 крайних правых цифр DK-AC							

2. формируется второй 16-значный блок P2:

P2:	C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

C — контрольное поле со значением 0x2

N — длина нового PIN, допустимые значения от 0x4 до 0xC

P — цифра нового PIN, допустимые значения от 0x0 до 0x9 (десятичные цифры)

F — дополнение, значение 0xF

3. формируется третий 16-значный блок P3 с использованием старого PIN:

P3:	P	P	P	P	P/0	P/0	P/0	P/0	P/0	P/0	P/0	P/0	0	0	0	0

P — цифра старого PIN, допустимые значения от 0x0 до 0x9 (десятичные цифры),

0 — дополнение, значение 0x0.

4. значение PIN-блока PB = P1 ⊕ P2 ⊕ P3. Результат называется Delta PIN.

11.2.8 Формат 47

Код формата PIN-блока 47 используется для идентификации PIN-блоков формата ISO 9564-1 формат 3, эквивалентного ANSI X9.8 формат 3. Для зашифрования PIN-блока может использоваться только ключ 3DES.

PIN-блок формата 47 формируется следующим образом:

1. формируется первый 16-значный блок P1 с использованием значения PIN:

P1:	C	N	P	P	P	P	P/F	P/F	P/F	P/F	P/F	P/F	P/F	P/F	F	F

C — контрольное поле со значением 0x3

N — длина PIN, допустимые значения от 0x4 до 0xC

P — цифра PIN, допустимые значения от 0x0 до 0x9 (десятичные цифры)

F — дополнение, допустимые значения от 0xA до 0xF. Значение F выбирается случайно или последовательно из шести возможных значений таким образом, чтобы максимально снизить вероятность повторного использования того же значения дополнения F..F для текущих номера счета и устройства шифрования PIN.

2. формируется второй 16-значный блок P2 с использованием номера счета:

P2:	0	0	0	0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}

0 — дополнение, значение 0x0

A_1..A_{12} — 12 крайних правых цифр номера счета (PAN), за исключением контрольной цифры, допустимые значения от 0x0 до 0x9 (десятичные цифры). A_{12} — цифра, непосредственно предшествующая контрольной цифре. Если PAN без контрольной цифры меньше 12 цифр, то цифры PAN выравниваются по правому краю и дополняются нулями слева.

3. значение PIN-блока PB = P1 ⊕ P2.

11.2.9 Формат 48

Код формата PIN-блока 48 используется для идентификации PIN-блоков формата ISO 9564-1 формат 4. Для зашифрования PIN-блока может использоваться только ключ AES.

PIN-блок формата 48 формируется следующим образом:

1. формируется первый 32-значный блок P1:

P1:	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

C — контрольное поле со значением 0x4

N — длина PIN, допустимые значения от 0x4 до 0xC

P — цифра PIN, допустимые значения от 0x0 до 0x9 (десятичные цифры)

F — дополнение, допустимые значения от 0xA до 0xF. Значение F выбирается случайно или последовательно из шести возможных значений таким образом, чтобы максимально снизить вероятность повторного использования того же значения дополнения F..F для текущих номера счета и устройства

шифрования PIN.

R — случайная цифра, допустимые значения от 0x0 до 0xF

2. формируется второй 32-значный блок P2:

P2:	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

M — допустимые значения от 0x0 до 0x7; M + 12 = длина PAN. Если PAN меньше 12 цифр, M устанавливается в значение 0.

A..A — PAN, допустимые значения от 0x0 до 0x9 (десятичные цифры). Если PAN меньше 12 цифр, то цифры выравниваются по правому краю и дополняются нулями слева.

0 — дополнение, значение 0x0.

3. значение зашифрованного (с использованием ключа шифрования PIN AES, далее — K) PIN-блока EPB вычисляется следующим образом:

(a) P1 зашифровывается с использованием ключа K: $e_K(P1)$

(b) зашифрованный P1 (полученный на предыдущем шаге) и P2 складываются: $e_K(P1) \oplus P2$

(c) полученная на предыдущем шаге сумма зашифровывается с использованием ключа K: $EPB = e_K(e_K(P1) \oplus P2)$

4. чтобы получить значение незашифрованного PIN из зашифрованного PIN-блока EPB:

(a) зашифрованный PIN-блок EPB расшифровывается с использованием ключа K: $d_K(e_K(e_K(P1) \oplus P2)) = e_K(P1) \oplus P2$

(b) полученный на предыдущем шаге блок и P2 складываются: $e_K(P1) \oplus P2 \oplus P2 = e_K(P1)$

(c) результат сложения расшифровывается с использованием ключа K: $d_K(e_K(P1)) = P1$

(d) значение PIN извлекается из полученного на предыдущем шаге P1.

Глава 12

SNMP

Платежный HSM поддерживает протокол сетевого управления устройствами SNMP (Simple Network Management Protocol) версий 2 и 3. Протокол SNMP используется для предоставления информации о текущем состоянии, доступности и некоторых статистических характеристиках HSM внешним устройствам.

Информация о HSM, предоставляемая по протоколу SNMP, может быть получены устройствами следующими способами:

1. внешнее устройство запрашивает информацию и HSM передает ее в ответ;

2. внешнее устройство получает автоматические SNMP-ловушки (SNMP-trap) — уведомления, отправляемые HSM при наступлении отслеживаемых событий.

Формат управляющей информация, предоставляемой HSM по протоколу SNMP, определяется в базе управляющей информации (Management Information Base, MIB).

Подключение к HSM по протоколу SNMP осуществляется через 161 UDP-порт.

12.1 Перечень управляющей информации HSM

Посредством протокола SNMP можно запросить следующую информацию о HSM:

1. Информация, доступная через стандартные MIB:

- Информация об узле (.1.3.6.1.2.1.1 — RFC1213-MIB, группа System)
- Информация о сетевых интерфейсах (.1.3.6.1.2.1.2 — RFC1213-MIB, группа Interfaces)
- Соответствие между сетевыми и физическими адресами (.1.3.6.1.2.1.3 — RFC1213-MIB, группа Address Translation)
- IP-адреса, статистика IP, маршрутизация (.1.3.6.1.2.1.4 — RFC1213-MIB, группа IP)
- TCP-соединения (.1.3.6.1.2.1.6 — RFC1213-MIB, группа TCP)
- Системная информация (.1.3.6.1.2.1.25.1 — HOST-RESOURCES-MIB, группа Host Resources System)
- Загрузка CPU за последнюю минуту (.1.3.6.1.2.1.25.3.3 — HOST-RESOURCES-MIB, таблица hrProcessorTable)
- Дополнительная информация о сетевых интерфейсах (.1.3.6.1.2.1.31.1.1 — IF-MIB, таблица fXTable)
- Датчики температуры, вентиляторов, напряжений (.1.3.6.1.4.1.2021.13.16 — LM-SENSORS-MIB)
- Значение Engine ID (.1.3.6.1.6.3.10.2.1.1 — SNMP-FRAMEWORK-MIB, параметр snmpEngineID)

2. Информация, доступная через проприетарные MIB (см. Приложение Г):

- Текущие дата и время;
- Состояние сервиса обработки команд хоста;
- Состояние сервиса консоли;
- Состояние платежного модуля;
- Статус web-интерфейса администрирования;

- Статус защиты канала хоста (TLS);
- Информация о загруженных LMK (идентификатор, тип схемы, алгоритм, статус, проверочное значение (KCV), состояние авторизации, количество авторизованных активностей, комментарии);
- Статистика загруженности CPU (в процентах);
- Текущая загрузка CPU (средняя за предыдущий период времени);
- Статистика использования команд хоста: количество выполненных команд и среднее время выполнения;
- Статистика использования команд хоста за предыдущий период времени;
- Статус сбора статистики;
- Количество сообщений в журнале событий;
- Количество сообщений в журнале аудита;
- Количество сообщений в журнале ошибок;
- Количество перезагрузок;
- Версия прошивки;
- Серийный номер;
- Информация о модели HSM;
- Состояние работоспособности HSM;
- Статистика работоспособности HSM:
  - статус сбора статистики (вкл/выкл);
  - дата и время последнего сброса статистики;
  - дата и время окончания сбора статистики;
  - количество перезагрузок HSM с момента последнего сброса;
  - количество обнаруженных случаев вскрытия корпуса HSM с момента последнего сброса;
  - количество превышений лимита ошибок проверки PIN в минуту с момента последнего сброса;
  - количество превышений лимита ошибок проверки PIN в час с момента последнего сброса;
  - количество превышений лимита PIN-атак с момента последнего сброса.
- Информация о результатах самотестирования: время последнего тестирования, общий результат, список тестов с результатами;
- Информация о хостовых сетевых интерфейсах (номера портов, количество сетевых интерфейсов, количество подключений, IP-адрес, маска подсети, шлюз по умолчанию, MAC-адрес, ...) ;
- Список включенных команд хоста;
- Информация о соответствии настроек безопасности PCI HSM.

HSM обеспечивает автоматическую отправку **SNMP-ловушек** (SNMP-trap) при наступлении следующих событий:

- Изменение настроек (безопасности, аудита команд, форматов PIN-блоков) и статуса соответствия PCI HSM;
- Изменение состояния платежного модуля;
- Включение HSM;
- Установка лицензии;
- Вскрытие корпуса HSM;
- Превышение лимита ошибок проверки PIN в минуту/час, лимита PIN-атак;
- Ошибка при самотестировании.

12.2 Консольные команды конфигурации SNMP и настройки SNMP-ловушек

Для конфигурации SNMP используются следующие команды консоли:

- **SNMP** — Конфигурация SNMP

Просмотр текущих настроек SNMP (список пользователей и строк доступа, статус сервиса, значения системных переменных), включение/отключение сервиса SNMP, установка значений системных переменных

- **SNMPADD** — Добавление пользователя или строки доступа SNMP

Добавление строки доступа или пользователя (с указанными именем, алгоритмом (MD5/SHA) и паролем аутентификации, алгоритмом (DES/AES) и паролем защиты данных)

- **SNMPDEL** — Удаление пользователя или строки доступа SNMP

Удаление строки доступа или пользователя с указанным индексом

Для управления SNMP-ловушками используются следующие команды консоли:

- **TRAP** — Конфигурация SNMP-ловушек

Просмотр текущих настроек SNMP-ловушек (список получателей ловушек, статус ловушек), включение/отключение SNMP-ловушек

- **TRAPADD** — Добавление получателя SNMP-ловушек

Добавление получателя SNMP-ловушек (с указанными сетевым адресом, портом, версией SNMP, индексом строки доступам (SNMPv2) или имени пользователя (SNMPv3))

- **TRAPDEL** — Удаление получателя SNMP-ловушек

Удаление получателя SNMP-ловушек с указанным индексом

12.3 База управляющей информации (MIB)

MIB описывает формат управляющих данных HSM и имеет иерархическую структуру, к записям объектов можно обратиться по идентификатору объекта (OID).

Содержание проприетарных MIB для Платежного HSM приведено в Приложении Г.

Глава 13

Печать ключевых компонент

В процессе обработки платежей участвуют несколько различных организаций и подразделений, которым необходимо обмениваться ключами шифрования. Для шифрования рабочих ключей, которыми обмениваются участвующие в процессе стороны, используются ключи шифрования ключей KEK (Key Encrypting Key). В качестве KEK также используются зональный мастер-ключ ZMK и терминальный мастер-ключ TMK.

Обмен KEK происходит посредством разделения ключа на несколько компонент. Ключи KEK генерируются одной стороной, разделяются на компоненты, распечатываются и передаются по отдельности нескольким доверенным лицам второй стороны. Для восстановления KEK все компоненты ключа совместно вводятся в специальное приложение. Знание одной компоненты не позволяет восстановить ключ KEK.

13.1 Описание процесса

Ключ KEK, который должен быть известен двум сторонам, участвующим в обработке транзакций, генерируется одной из сторон и передается другой стороне безопасным способом.

Каждая организация должна разработать свой собственный процесс распространения и применения ключевых компонент (включая уничтожение ключевых компонент после их использования), соответствующий требованиям безопасности.

Пример безопасного процесса распространения ключевых компонент и формирования из них KEK, организованный с использованием HSM:

• Организация, генерирующая KEK:

1. настраивает формат печати ключевой компоненты;

2. генерирует и печатает необходимое число ключевых компонент KEK, сохраняет их копии в зашифрованном под LMK виде;

3. передает каждый ключевой компонент определенному доверенному лицу второй организации-получателя;

4. формирует KEK из зашифрованных ключевых компонент с использованием HSM;

5. сохраняет KEK в зашифрованном под LMK виде в своей базе данных.

• Организация, получившая ключевые компоненты KEK:

1. ответственные за ключевые компоненты доверенные лица собираются вместе и поочередно вводят свои компоненты в HSM;

2. HSM формирует ключ из введенных ключевых компонент;

3. HSM отображает полученный KEK в зашифрованном под LMK виде;

4. KEK в зашифрованном виде сохраняется в базе данных (как правило, расположенной в хост-системе). Далее KEK можно использовать в командах, необходимых для операций с рабочими ключами в HSM.

При использовании HSM ключевые компоненты создаются поочередно и должны быть распечатаны с помощью подключенного к HSM принтера на специальном конверте, защищенном от НСД.

Возможна печать ключевых компонент с помощью обычного принтера без использования специальных конвертов, однако в этом случае с помощью дополнительных организационно-технических мер необходимо обеспечить, чтобы каждая компонента была доступна не более чем одному доверенному лицу и каждое доверенное лицо имело доступ не более чем к одной компоненте ключа.

13.2 Настройка формата печати ключевых компонент

Перед печатью ключевых компонент на принтере в HSM должен быть загружен формат печати, определяющий форматирование вывода.

Одновременно HSM может хранить только один формат печати, поэтому он должен загружаться на HSM перед каждым вызовом команды печати.

Формат печати загружается в HSM в виде текстовой строки, которая состоит из:

- констант;
- служебных символов для форматирования вывода;
- маркеров полей печати, которые заполняются данными при печати ключевых компонент.

Максимальная длина определения формата печати — 299 символов и констант.

Для загрузки форматов печати на HSM используется команда хоста PA.

Пример команды для установки формата печати ключевых компонент:

```
>L>003^0>033^1>L>003 KEY 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
```

Результат форматирования:

```
..(Field 0)...................(Field 1)
..KEY.COMPONENT.PART.1:.xxxxxxxxxxxxxxxx
..KEY.COMPONENT.PART.2:.yyyyyyyyyyyyyyyy
..KEY.COMPONENT.PART.3:.zzzzzzzzzzzzzzzz
..KEY.CHECK.VALUE:.vvvvvv
```

Примечание:

- вместо "." будет напечатан пробел;
- вместо "(Field X)" будут напечатаны данные (например, получатель или дата), предоставляемые хостом в процессе печати.

Полный перечень символов, используемых для установки формата печати в HSM, и их значения приведены в Приложении А.

При генерации и печати ключа в виде нескольких (двух или трех) разделённых компонент (команда хоста NE) служебные символы ^P, ^Q и ^R определяют место печати первой, второй и третьей (при наличии) частей:

Тип ключа	**^P**	**^Q**	**^R**
2DES	первые 16 шестнадцатеричных цифр компоненты	вторые 16 шестнадцатеричных цифр компоненты	н/д
3DES	первые 16 шестнадцатеричных цифр компоненты	вторые 16 шестнадцатеричных цифр компоненты	третьи 16 шестнадцатеричных цифр компоненты

13.3 Генерация и печать ключевых компонент

Для генерации и печати ключевой компоненты на принтере, подключенном к HSM, выполняется команда хоста:

- A2 — Генерация и печать компоненты.

Как правило, генерируются три компоненты, каждая из которых распечатывается отдельно в своем конверте. Для печати каждой компоненты вызывается команда A2.

Ответ A3 от HSM на команду A2 содержит компоненту, зашифрованную под LMK, и проверочное значение для компоненты (опционально).

13.4 Формирование ключа из компонент

13.4.1 Организация, генерирующая KEK

После генерации компонент они хранятся на хосте в зашифрованном под LMK виде. Для формирования KEK из сохраненных зашифрованных компонент используется команда хоста:

- A4 — Формирование ключа из зашифрованных компонент.

HSM расшифровывает ключевые компоненты, формирует из компонент KEK, зашифровывает KEK под LMK и возвращает хосту. Зашифрованный KEK сохраняется в базе данных хоста и при необходимости передается в командах хоста на HSM, где он расшифровывается для дальнейших действий.

13.4.2 Организация, получившая ключевые компоненты KEK

Ключевые компоненты хранятся отдельно у доверенных лиц организации. Каждое доверенное лицо имеет доступ только к одной ключевой компоненте, и для формирования KEK из компонент требуется присутствие всех доверенных лиц.

Для формирования KEK из ключевых компонент используется консольная команда:

- FK — Формирование ключа из компонент.

Владельцы ключевых компонент поочередно вводят компоненты в HSM, HSM формирует KEK и отображает его в зашифрованном под LMK виде.

Затем зашифрованный KEK сохраняется в базе данных хоста.

Глава 14

Дополнительные меры обеспечения безопасности

14.1 Защита от атак на PIN

HSM поддерживает функцию обнаружения и предотвращения атак на PIN полным перебором (брут-форс) — передачи большого массива PIN до тех пор, пока не будет обнаружен верный PIN. Обнаружение таких атак реализовано путем оценки количества неудачных проверок PIN за минуту и за час. Каждый раз, когда количество ошибок проверки PIN (в минуту/час) превышает заданный лимит, счетчик атак на PIN увеличивается.

Установка и просмотр текущих значений лимитов ошибок проверок PIN (в минуту/час) и лимита PIN-атак выполняется с помощью консольной команды A5.

Также посредством команды A5 определяется, как будет реагировать HSM при обнаружении атаки. Поддерживаются 2 режима: автоматическое реагирование в случае обнаружения атаки на PIN (`On`) или только ведение журнала с данными о превышении указанных лимитов (`Logging only`).

Независимо от выбранного режима при превышении любого из описанных выше лимитов данные об этом заносятся в журнал аудита. Данные о превышении лимитов ошибок проверки PIN в минуту/час и количестве атак на PIN относятся к данным о работоспособности HSM. Включить/отключить сбор статистики о работоспособности HSM можно с помощью консольной команды HEALTHENABLE. Получить данные статистики (включая данные о количестве превышений лимитов ошибок проверок PIN в минуту/час и обнаруженных PIN-атак), а также сбросить накопленные данные, можно с помощью консольной команды HEALTHSTATS.

Кроме того, если выбран режим реагирования (`On`), при превышении любого из установленных лимитов ошибочных проверок PIN будут заблокированы команды проверки PIN (команды хоста 'DA', 'EA', 'DC', 'EC', 'BC', 'BE') — HSM будет возвращать код ошибки 39 в ответ на эти команды до тех пор, пока функция защиты от атак на PIN не будет перезапущена с помощью консольной команды A7.

Если количество обнаруженных PIN-атак превысило установленный (командой A5) лимит зафиксированных PIN-атак, все LMK будут удалены.

Приложение A

Служебные символы формата печати

Символ	**ASCII**	**Описание**
>L	3E 4C	Перевод строки, возврат каретки
>V	3E 56	Вертикальная табуляция
>H	3E 48	Горизонтальная табуляция
>F	3E 46	Смена страницы
>nnn	3E 3n 3n 3n	Отступ nnn от левого поля страницы, где nnn — трехзначное десятичное число
^P	5E 50	Для PIN-конверта: печать открытого PIN для конверта 1 Для ключевого документа: печать открытой компоненты
^Q	5E 51	Для PIN-конверта: печать открытого PIN для конверта 2 Для ключевого документа: печать второй открытой компоненты (для ключевых документов разрешена только печать 1 документа на листе)
^R	5E 52	Печать ссылочного номера для PIN-конверта 1 Для ключевого документа: печать третьей открытой компоненты
^S	5E 53	Печать ссылочного номера для PIN-конверта 2
^T	5E 54	Печать последних 6 цифр номера счета для PIN-конверта 1
^U	5E 55	Печать последних 6 цифр номера счета для PIN-конверта 2
|<L><hh hh hh ..>	7C<L><hh hh hh ..>	Отправить двоичные данные на принтер, например, контрольную строку принтера. После символа | следует длина строки в байтах <L> (0-F), затем указывается шестнадцатеричная строка <hh hh hh ..>. *Примечание:* Символы '<' и '>' обозначают границы поля и не являются частью данных, передаваемых на печать. Например, 4-байтовая строка 7C<4><1B283358> передается на печать в виде |41B283358.
^0	5E 30	Вставить поле печати 0
^1	5E 31	Вставить поле печати 1
^2	5E 32	Вставить поле печати 2
^3	5E 33	Вставить поле печати 3
..	..	..
^A	5E 41	Вставить поле печати 10

^B	5E 42	Вставить поле печати 11
..	..	..
^F	5E 46	Вставить поле печати 15
^^10	5E 5E 31 30	Вставить поле печати 16
^^11	5E 5E 31 31	Вставить поле печати 17
^^12	5E 5E 31 32	Вставить поле печати 18
..	..	..
^^1F	5E 5E 31 46	Вставить поле печати 31

Печать PIN словами

HSM поддерживает печать PIN словами, например: ONE TWO THREE FOUR. По умолчанию используется английский язык. Для переопределения текстовых значений для цифр PIN на другом языке используется команда хоста 'LI'.

Печать PIN словами настраивается с помощью следующих служебных символов (могут применяться в дополнение к символам, определяющим печать PIN в цифровом формате).

Символ	**ASCII**	**Описание**
^V	5E 56	Распечатать открытый PIN словами для PIN-конверта 1. Может использоваться для печати 1 документа на листе или 2 документов на листе.
^W	5E 57	Распечатать открытый PIN словами для PIN-конверта 2. Может использоваться только для печати 2 документов на листе.

Печать PIN вертикально («в столбик»)

Возможно настроить печать PIN (цифрами или словами) по одной цифре в строку («в столбик»), например:

1 ONE

2 TWO

3 THREE

4 FOUR

В описании служебных символов n обозначает, какая цифра PIN должна быть напечатана:

Цифра PIN	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

Печать PIN вертикально по одной цифре в строку осуществляется с помощью следующих служебных символов.

Символ	**ASCII**	**Описание**
^Pn	5E 50 31-39 или 41-43	Распечатать цифру n открытого PIN в формате цифры (например, 1) для PIN-конверта 1. Может использоваться для печати 1 документа на листе или 2 документов на листе.
^Qn	5E 51 31-39 или 41-43	Распечатать цифру n открытого PIN в формате цифры (например, 1) для PIN-конверта 2. Может использоваться только для печати 2 документов на листе.
^Vn	5E 56 31-39 или 41-43	Распечатать цифру n открытого PIN в формате слова (например, ONE) для PIN конверта 1. Может использоваться для печати 1 документа на листе или 2 документов на листе.
^Wn	5E 57 31-39 или 41-43	Распечатать цифру открытого PIN n в формате слова (например, ONE) для PIN-конверта 2. Может использоваться только для печати 2 документов на листе.

Приложение Б

Конвертация значения поля «Использование ключа» при экспорте ключа в формате Key Block

При экспорте ключа из формата Проприетарный Key Block в формат TR-31 Key Block конвертация значения поля «Использование ключа» в Key Block выполняется в соответствии со следующей таблицей:

Проприетарный Key Block	**Использование ключа**	**TR-31 Key Block**
'03'	Асимметричный ключ (ECC) обмена	'K3'
'03'	Асимметричный ключ (ECC) подписи	'S0'
'41'	BDK-2	'B0'
'42'	BDK-3	'B0'
'43'	BDK-4	'B0'
'11'	Проверка карты (American Express CSC)	'C0'
'12'	Проверка карты (Mastercard CVC)	'C0'
'13'	Проверка карты (Visa CVV)	'C0'
'21'	Шифрование данных (DEK)	'D0'
'22'	Шифрование данных (ZEK)	'D0'
'23'	Шифрование данных (TEK)	'D0'
'31'	Visa Cash Master Load (KML)	'E6'
'32'	Мастер-ключ Dynamic CVV (MK-CVC3)	'E6'
'51'	Шифрование терминальных ключей (TMK)	'K0'
'52'	Шифрование зональных ключей (ZMK)	'K0'
'54'	Ключ шифрования ключей (KEK)	'K0'
'71'	Терминальный ключ шифрования PIN (TPK)	'P0'
'72'	Зональный ключ шифрования PIN (ZPK)	'P0'
'73'	Terminal Key Register (TKR)	'P0'

Приложение B

Таблица ключевых схем

Символ (признак схемы)	**Описание**
'U'	2DES, Variant Ключ, зашифрованный под Variant LMK или ключом экспорта/импорта (ZMK и пр.)*
'T'	3DES, Variant Ключ, зашифрованный под Variant LMK или ключом экспорта/импорта (ZMK и пр.)*
'X'	2DES, ANSI X9.17 Ключ, зашифрованный под ключом экспорта/импорта (ZMK и пр.)*
'Y'	3DES, ANSI X9.17 Ключ, зашифрованный под ключом экспорта/импорта (ZMK и пр.)*
'V'	2DES/3DES, Verifone/GISKE Key Block Ключ, зашифрованный под ключом экспорта (ZMK и пр.)
'R'	2DES/3DES/AES, TR-31 Key Block Ключ, зашифрованный под ключом экспорта/импорта (ZMK и пр.)
'S'	2DES/3DES/AES/RSA/ECC/HMAC, Проприетарный Key Block Ключ, зашифрованный под Key Block LMK или ключом экспорта/импорта (ZMK и пр.)

* Возможность использования ключевых схем зависит от некоторых настроек безопасности, подробнее см. раздел «Настройки безопасности (security settings)» документа «КриптоПро HSM. Команды хоста»

Приложение Г

Проприетарные SNMP MIB

Корень CryptoPro MIB

```
CRYPTOPRO-MIB DEFINITIONS ::= BEGIN

IMPORTS
        MODULE-IDENTITY, enterprises
                FROM SNMPv2-SMI;

cryptopro MODULE-IDENTITY
        LAST-UPDATED "202106290000Z"
        ORGANIZATION
                "CryptoPro"
        CONTACT-INFO
                "email:      info@cryptopro.ru"
        DESCRIPTION
                "root CryptoPro MIB."

        REVISION "202106291430Z"
        DESCRIPTION
                ""
::= { enterprises 47559 }

END
```

CryptoPro HSM MIB

```
CRYPTOPRO-HSM-MIB DEFINITIONS ::= BEGIN

IMPORTS
        OBJECT-TYPE, MODULE-IDENTITY, Integer32
                FROM SNMPv2-SMI
        TEXTUAL-CONVENTION
                FROM SNMPv2-TC
        cryptopro
                FROM CRYPTOPRO-MIB;

cryptoprohsm MODULE-IDENTITY
        LAST-UPDATED "202111060000Z"
        ORGANIZATION
                "cryptopro mib"
        CONTACT-INFO
                "email:      info@cryptopro.ru"
        DESCRIPTION
                "CryptoPro HSM MIB."

        REVISION "202111060000Z"
        DESCRIPTION
                "initial Cryptopro HSM mib"
::= { cryptopro 1 }


-- TEXTUAL CONVENTION DisplayString

TPS ::= TEXTUAL-CONVENTION
        DISPLAY-HINT         "d-2"
        STATUS     current
        DESCRIPTION
                "Fixed point, two decimal"
        SYNTAX  Integer32

state                OBJECT IDENTIFIER ::= { cryptoprohsm 1 }

cphsmTcpHostState  OBJECT-TYPE
        SYNTAX     INTEGER {
                cphsmHostTcpServerUp (1),
                cphsmHostTcpServerDown (2)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates whether the process to service host commands is currently running over
                the Ethernet host port using the TCP protocol. cphsmHostTcpServerUp (1) indicates
                that it is; all the other states indicate that TCP is currently inactive."
        ::= { state 1 }

cphsmLmkStatusTable  OBJECT-TYPE
        SYNTAX SEQUENCE OF CphsmLmkStatusEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "A table that returns the status of all possible LMK sets."
        ::= { state 2}
```

```
cphsmLmkStatusEntry  OBJECT-TYPE
        SYNTAX         CphsmLmkStatusEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "Row in the LMKStatusTable"
        INDEX { cphsmLmkIndex }
        ::= { cphsmLmkStatusTable 1 }

CphsmLmkStatusEntry ::= SEQUENCE {
        cphsmLmkIndex
                Integer32,
        cphsmLmkId
                Integer32,
        cphsmLmkStatusScheme
                INTEGER,
        cphsmLmkStatusAlgorithm
                INTEGER,
        cphsmLmkStatusLiveTest
                INTEGER,
        cphsmLmkStatusCheckDigits
                OCTET STRING
}

cphsmLmkIndex  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "The index into the LMK Status table"
        ::= { cphsmLmkStatusEntry 1 }

cphsmLmkId  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "ID of loaded LMK"
        ::= { cphsmLmkStatusEntry 2 }

cphsmLmkStatusScheme  OBJECT-TYPE
        SYNTAX     INTEGER {
                lmkSchemeVariant (1),
                lmkSchemeKeyblock (2)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates the scheme of the loaded LMK set (variant, key block)"
        ::= { cphsmLmkStatusEntry 3 }

cphsmLmkStatusAlgorithm  OBJECT-TYPE
        SYNTAX     INTEGER {
                lmkAlgorithm3Des112 (1),
                lmkAlgorithm3Des (2),
                lmkAlgorithmAes128 (3),
                lmkAlgorithmAes256 (4)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
```

```
                "Indicates the algorithm use by the LMK set for encryption."
        ::= { cphsmLmkStatusEntry 4 }

cphsmLmkStatusLiveTest  OBJECT-TYPE
        SYNTAX     INTEGER {
                lmkStatusTest (1),
                lmkStatusLive (2)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates the status of the currently loaded LMK set. This will be either 'Live'
                or 'Test'."
        ::= { cphsmLmkStatusEntry 5 }

cphsmLmkStatusCheckDigits  OBJECT-TYPE
        SYNTAX     OCTET STRING
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates the check digits for an LMK."
        ::= { cphsmLmkStatusEntry 6 }

cphsmCpuUsage OBJECT IDENTIFIER ::= { state 3 }

cphsmCpuUsageTable  OBJECT-TYPE
        SYNTAX SEQUENCE OF CphsmCpuUsageEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "A table with info about CPU usage"
        ::= { cphsmCpuUsage 0 }

cphsmCpuUsageDate  OBJECT-TYPE
        SYNTAX     OCTET STRING
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Date and time of measurement"
        ::= { cphsmCpuUsage 1 }

cphsmCpuUsageEntry  OBJECT-TYPE
        SYNTAX         CphsmCpuUsageEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "Row in CpuUsageTable"
        INDEX { cphsmCpuUsageIndex }
        ::= { cphsmCpuUsageTable 1 }

CphsmCpuUsageEntry ::= SEQUENCE {
        cphsmCpuUsageIndex
                Integer32,
        cphsmCpuUsageStartPercent
                Integer32,
        cphsmCpuUsageEndPercent
                Integer32,
        cphsmCpuUsageSamples
                Integer32
}
```

```
cphsmCpuUsageIndex  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "The index into the CpuUsageTable"
        ::= { cphsmCpuUsageEntry 1 }

cphsmCpuUsageStartPercent  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Start of CPU Usage percent range"
        ::= { cphsmCpuUsageEntry 2 }

cphsmCpuUsageEndPercent  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "End of CPU Usage percent range"
        ::= { cphsmCpuUsageEntry 3 }

cphsmCpuUsageSamples  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Amount of samples where CPU Usage was in specified range"
        ::= { cphsmCpuUsageEntry 4 }

cphsmCommandVolume OBJECT IDENTIFIER ::= { state 4 }

cphsmCommandVolumeTable  OBJECT-TYPE
        SYNTAX SEQUENCE OF CphsmCommandVolumeEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "A table that returns individual commands executions count since last
                initialization"
        ::= { cphsmCommandVolume 0 }

cphsmCommandVolumeDate  OBJECT-TYPE
        SYNTAX     OCTET STRING
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Date and time of measurement"
        ::= { cphsmCommandVolume 1 }

cphsmCommandVolumeEntry  OBJECT-TYPE
        SYNTAX         CphsmCommandVolumeEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "Row in CommandVolumeTable"
        INDEX { cphsmCommandVolumeIndex }
        ::= { cphsmCommandVolumeTable 1 }

CphsmCommandVolumeEntry ::= SEQUENCE {
```

```
        cphsmCommandVolumeIndex
                Integer32,
        cphsmCommandVolumeCommandName
                OCTET STRING,
        cphsmCommandVolumeCommandCount
                Integer32
}

cphsmCommandVolumeIndex  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "The index into the CommandVolumeTable"
        ::= { cphsmCommandVolumeEntry 1 }

cphsmCommandVolumeCommandName  OBJECT-TYPE
        SYNTAX     OCTET STRING
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Name of command"
        ::= { cphsmCommandVolumeEntry 2 }

cphsmCommandVolumeCommandCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Count of command executions from last reset"
        ::= { cphsmCommandVolumeEntry 3 }

cphsmCommandPerf OBJECT IDENTIFIER ::= { state 5 }

cphsmCommandPerfTable  OBJECT-TYPE
        SYNTAX SEQUENCE OF CphsmCommandPerfEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "A table that returns individual commands TPS (Transactions per second)"
        ::= { cphsmCommandPerf 0 }

cphsmCommandPerfDate  OBJECT-TYPE
        SYNTAX     OCTET STRING
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Date and time of measurement"
        ::= { cphsmCommandPerf 1 }

cphsmCommandPerfEntry  OBJECT-TYPE
        SYNTAX         CphsmCommandPerfEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "Row in command stat table"
        INDEX { cphsmCommandPerfIndex }
        ::= { cphsmCommandPerfTable 1 }

CphsmCommandPerfEntry ::= SEQUENCE {
        cphsmCommandPerfIndex
```

```
                Integer32,
        cphsmCommandPerfCommandName
                OCTET STRING,
        cphsmCommandPerfTPS
                TPS
}

cphsmCommandPerfIndex  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "The index into the Command Perf table"
        ::= { cphsmCommandPerfEntry 1 }

cphsmCommandPerfCommandName  OBJECT-TYPE
        SYNTAX     OCTET STRING
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Name of command"
        ::= { cphsmCommandPerfEntry 2 }

cphsmCommandPerfTPS  OBJECT-TYPE
        SYNTAX     TPS
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "TPS for particular command"
        ::= { cphsmCommandPerfEntry 3 }

cphsmRebootCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Reboots from last initialization count"
        ::= { state 6 }

cphsmEventLogCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Event log messages count"
        ::= { state 7 }

cphsmAuditLogCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Audit log messages count"
        ::= { state 8 }

cphsmFirmwareNumber  OBJECT-TYPE
        SYNTAX     OCTET STRING
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Firmware number"
```

```
        ::= { state 9 }

cphsmSerialNumber  OBJECT-TYPE
        SYNTAX     OCTET STRING
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Serial number"
        ::= { state 10 }

cphsmState  OBJECT-TYPE
        SYNTAX     OCTET STRING
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "General description of HSM state"
        ::= { state 11 }
END
```

PS MIB

```
PS-MIB DEFINITIONS ::= BEGIN

IMPORTS
        OBJECT-TYPE, MODULE-IDENTITY,
        enterprises, IpAddress, Integer32, Gauge32
                FROM SNMPv2-SMI
        MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF
        MacAddress, DisplayString, DateAndTime, TruthValue
                FROM SNMPv2-TC;

psMIB MODULE-IDENTITY

        LAST-UPDATED "202407031500Z"
        ORGANIZATION
                ""
        CONTACT-INFO
                ""
        DESCRIPTION
            ""
    ::= { rpMibs 10000 }

t       OBJECT IDENTIFIER ::= { enterprises 4096 }
tMibs   OBJECT IDENTIFIER ::= { t 2 }
rpMibs          OBJECT IDENTIFIER ::= { tMibs 2 }

-- Utilization statistics provide information on the device's current host command
-- processing load.

Util                OBJECT IDENTIFIER ::= { psMIB 1 }

-- State information for the device.

State               OBJECT IDENTIFIER ::= { psMIB 2 }
StateTamper            OBJECT IDENTIFIER ::= { State 2 }

-- LMK information for the device.

Lmk                 OBJECT IDENTIFIER ::= { psMIB 3 }

-- Communication ports information for the device.

Comms               OBJECT IDENTIFIER ::= { psMIB 4 }
CommsMgmt              OBJECT IDENTIFIER ::= { Comms 1 }
CommsHost              OBJECT IDENTIFIER ::= { Comms 2 }
CommsHostPort          OBJECT IDENTIFIER ::= { CommsHost 4 }

-- System information for the device.

System              OBJECT IDENTIFIER ::= { psMIB 5 }

-- Contains information to identify the software running on the device.

Version             OBJECT IDENTIFIER ::= { psMIB 7 }

-- Provides information pertaining to the licensing of the device.

Licensing           OBJECT IDENTIFIER ::= { psMIB 8 }
```

```
-- Provides the number of and list of host commands that are enabled on this device.

EnabledHostCommands OBJECT IDENTIFIER ::= { psMIB 9 }

-- Info on the system logs.

Logs                OBJECT IDENTIFIER ::= { psMIB 10 }

LogsErrorlog                OBJECT IDENTIFIER ::= { Logs 1 }

-- Info about the system audit log.

LogsAuditlog           OBJECT IDENTIFIER ::= { Logs 2 }

-- Data pertaining to the diagnostic self tests and the HealthCheckCounts.

Health              OBJECT IDENTIFIER ::= { psMIB 11 }

-- Provides the number of diagnostic tests last performed and the results of each test.

HealthDiagSelfTest     OBJECT IDENTIFIER ::= { Health 1 }

-- Health check counts for the device.

HealthCheckCounts      OBJECT IDENTIFIER ::= { Health 2 }

-- Contains information about the configuration of Host interfaces. For enabled interfaces
-- details about the  physical and protocol configuration are provided.

HostConnection      OBJECT IDENTIFIER ::= { psMIB 12 }

-- Settings and states of the Host ethernet interfaces.
-- Only available when HostConnectionEthernetEnabled is TRUE.

HostConnectionEthernet OBJECT IDENTIFIER ::= { HostConnection 3 }

-- device settings.

Settings            OBJECT IDENTIFIER ::= { psMIB 15 }

-- Notifications

tNotifications OBJECT IDENTIFIER ::= { t 999 }

psNotifications OBJECT IDENTIFIER ::= { tNotifications 2 }

AlarmObjects OBJECT IDENTIFIER ::= { psNotifications 1 }

Traps OBJECT IDENTIFIER ::= { psNotifications 2 }

-- Information on the new device state

UtilLoad  OBJECT-TYPE
        SYNTAX     Gauge32 (0..100)
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Displays the instantaneous load for all host commands processed
                by the device in the previous time period."
        ::= { Util 1 }
```

```
-- The octet string is formatted as follows (where CC is command code, e.g. A0):
--
-- <measurement period in seconds>,<num commands in report><LINE FEED>
-- CC,<total number commands>;CC,<total number commands>; ... etc...

UtilHostCmdVolume  OBJECT-TYPE
        SYNTAX     OCTET STRING (SIZE(0..484))
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Displays the measurement period in seconds and the number of commands
                in the report followed by a list of all host commands and the number
                of times each of those host commands that have been processed within
                the previous measurement period. Note that if a large variety of commands
                are processed, then this list may be incomplete and will contain the
                commands that consumed the highest proportion of the device's capacity."
        ::= { Util 2 }

UtilEnabled  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "True if Utilization turned on"
        ::= { Util 3 }

StateDevice  OBJECT-TYPE
        SYNTAX     INTEGER {
                stateUnavailable (1),
                stateOnline (2),
                stateOffline (3),
                stateSecure (4)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates the current state of the device."
        ::= { State 1 }

StateTamperState  OBJECT-TYPE
        SYNTAX     INTEGER {
                stateUnknown (1),
                stateOK (2),
                stateTampered (3)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates whether the device is currently in a tamper state."
        ::= { StateTamper 1 }

StateTamperDate  OBJECT-TYPE
        SYNTAX     DateAndTime
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The Date and Time that the last tamper event occurred. A NULL value
                indicates that the unit has not registered any tamper events."
        ::= { StateTamper 2 }
```

```
StateTamperCause  OBJECT-TYPE
        SYNTAX     INTEGER {
                causeUnavailable (1),
                causeTemperatureOutOfRange (2),
                causeBatteryLow (3),
                causeEraseButtonPressed (4),
                causeSensorProcessorWatchdog (5),
                causeSensorProcessorRestart (6),
                causeVoltageOutOfRange (7),
                causeMotionDetected (8),
                causeCaseTampered (9),
                causePowerLoss (10)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The cause of the last tamper event"
        ::= { StateTamper 3 }

LmkNumLoaded  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The number of LMKs currently loaded into the device."
        ::= { Lmk 1 }

LmkNumTestLmksLoaded  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates the number of 'test' LMKs loaded (as opposed to production).
                When the device is operating in a live environment, no test
                LMKs should be loaded and this field should be returned as 0."
        ::= { Lmk 2 }

LmkNumOldLmksLoaded  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates the number of old LMKs currently loaded in the device.
                Old LMKs should be stored in the device only as long as they are needed for
                translation purposes. Once all LMK-protected data has been placed under the
                control of the new LMK set, then the old LMK set should be deleted."
        ::= { Lmk 3 }

LmkStatusTable  OBJECT-TYPE
        SYNTAX SEQUENCE OF LmkStatusEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "A table that returns the status of all possible LMK sets."
        ::= { Lmk 4 }

LmkStatusEntry  OBJECT-TYPE
        SYNTAX         LmkStatusEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
```

```
                "Row in the LMK Status table"
        INDEX { LmkStatusIndex }
        ::= { LmkStatusTable 1 }

LmkStatusEntry ::= SEQUENCE {
        LmkStatusIndex
                Integer32,
        LmkStatusLoaded
                TruthValue,
        LmkStatusAuth
                TruthValue,
        LmkStatusNumAuthActivities
                Integer32,
        LmkStatusScheme
                INTEGER,
        LmkStatusAlgorithm
                INTEGER,
        LmkStatusLiveTest
                INTEGER,
        LmkStatusComments
                OCTET STRING,
        LmkStatusCheckDigits
                DisplayString
}

LmkStatusIndex  OBJECT-TYPE
        SYNTAX     Integer32 (1..20)
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "The index into the LMK Status table"
        ::= { LmkStatusEntry 1 }

LmkStatusLoaded  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates whether or not an LMK set is currently loaded at this index
                in the table. Returns TRUE if loaded, else FALSE."
        ::= { LmkStatusEntry 2 }

LmkStatusAuth  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Authorized state of this LMK set. A value of TRUE indicates that the
                LMK is in an authorized state; FALSE indicates that it is not authorized."
        ::= { LmkStatusEntry 3 }

LmkStatusNumAuthActivities  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates the number of authorized activities if the device
                is set to Multi-Auth mode. If not in multi-auth mode, returns 0."
        ::= { LmkStatusEntry 4 }

LmkStatusScheme  OBJECT-TYPE
```

```
        SYNTAX     INTEGER {
                lmkSchemeUnknown (1),
                lmkSchemeVariant (2),
                lmkSchemeKeyblock (3),
                lmkSchemeAES (4)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates the scheme of the loaded LMK set (variant, keyblock, or AES)."
        ::= { LmkStatusEntry 5 }

LmkStatusAlgorithm  OBJECT-TYPE
        SYNTAX     INTEGER {
                lmkAlgorithmUnknown (1),
                lmkAlgorithm3DES2Key (2),
                lmkAlgorithm3DES3Key (3),
                lmkAlgorithmAES256 (4)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates the algorithm used by the LMK set for encryption."
        ::= { LmkStatusEntry 6 }

LmkStatusLiveTest  OBJECT-TYPE
        SYNTAX     INTEGER {
                lmkStatusUnknown (1),
                lmkStatusLive (2),
                lmkStatusTest (3)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates the status of the currently loaded LMK set.
                This will be either 'Live' or 'Test'. Note that test LMKs should not
                be loaded into a device operating in a live customer environment."
        ::= { LmkStatusEntry 7 }

LmkStatusComments  OBJECT-TYPE
        SYNTAX     OCTET STRING (SIZE(0..41))
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Textual description of the LMK set loaded. A string of length 0
                indicates that no description is stored."
        ::= { LmkStatusEntry 8 }

LmkStatusCheckDigits  OBJECT-TYPE
        SYNTAX     DisplayString
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates the check digits for an LMK."
        ::= { LmkStatusEntry 9 }

CommsMgmtConsoleState  OBJECT-TYPE
        SYNTAX     INTEGER {
                consoleUp (1),
                consoleDown (2),
                consoleDisabledByGui (3),
```

```
                consoleUnavailable (4)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates whether the process to service Console Management requests
                is currently running. CommsMgmtConsoleStateUp (1) indicates that
                it is; all the other states indicate that the console is currently inactive."
        ::= { CommsMgmt 1 }

CommsMgmtGuiState  OBJECT-TYPE
        SYNTAX     INTEGER {
                guiUp (1),
                guiDown (2),
                guiUnavailable (3)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates whether the process to service GUI requests is currently running.
                CommsMgmtGuiStateUp (1) indicates that it is; all the other states
                indicate that the GUI is currently inactive."
        ::= { CommsMgmt 2 }

CommsHostTCPServer  OBJECT-TYPE
        SYNTAX     INTEGER {
                serverUp (1),
                serverDown (2),
                serverNotEnabled (3)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates whether the process to service host commands is currently
                running over the Ethernet host port using the TCP protocol.
                CommsHostTCPServer (1) indicates that it is;
                all the other states indicate that TCP is currently inactive."
        ::= { CommsHost 1 }

CommsHostUDerver  OBJECT-TYPE
        SYNTAX     INTEGER {
                serverUp (1),
                serverDown (2),
                serverNotEnabled (3)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates whether the process to service host commands is currently
                running over the Ethernet host port using the UDP protocol.
                CommsHostUDerverUp (1) indicates that it is; all the other
                states indicate that UDP is currently inactive."
        ::= { CommsHost 2 }

CommsHostFICONServer  OBJECT-TYPE
        SYNTAX     INTEGER {
                serverUp (1),
                serverDown (2),
                serverNotEnabled (3)
        }
        MAX-ACCESS read-only
```

```
        STATUS     current
        DESCRIPTION
                "Indicates whether the process to service host commands is currently
                running over the FICON host port. CommsHostFICONServerUp (1)
                indicates that it is; all the other states indicate that FICON is
                currently inactive. "
        ::= { CommsHost 3 }

CommsHostPortEthernet1  OBJECT-TYPE
        SYNTAX     INTEGER {
                portUp (1),
                portDown (2),
                portUnavailable (3),
                portNotConfigured (4)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates if the first host ethernet port is up and running."
        ::= { CommsHostPort 1 }

CommsHostPortEthernet2  OBJECT-TYPE
        SYNTAX     INTEGER {
                portUp (1),
                portDown (2),
                portUnavailable (3),
                portNotConfigured (4)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates if the second host ethernet port is up and running."
        ::= { CommsHostPort 2 }

CommsHostPortFICON  OBJECT-TYPE
        SYNTAX     INTEGER {
                portUp (1),
                portDown (2),
                portUnavailable (3),
                portNotConfigured (4)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Indicates if the FICON port (if present) is up and running."
        ::= { CommsHostPort 3 }

SystemDateAndTime  OBJECT-TYPE
        SYNTAX     DateAndTime
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The date and time as reported by the system's Real Time Clock."
        ::= { System 1 }

SystemSerialNum  OBJECT-TYPE
        SYNTAX     OCTET STRING (SIZE(0..32))
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Displays the serial number of the device system."
```

```
        ::= { System 2 }

SystemModel  OBJECT-TYPE
        SYNTAX     DisplayString
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Displays the model/series information of the device system."
        ::= { System 3 }

VersionSoftwareFirmwareVersion  OBJECT-TYPE
        SYNTAX     DisplayString
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Semantic version of firmware on the device."
        ::= { Version 9 }

LicensingPerformanceModel  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The maximum calls per second this device unit is licensed for."
        ::= { Licensing 1 }

LicensingPackage  OBJECT-TYPE
        SYNTAX     OCTET STRING (SIZE(0..2048))
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The license package loaded on this device unit."
        ::= { Licensing 2 }

LicensingOptionalLicenseCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The number of optional licenses this device unit has installed."
        ::= { Licensing 3 }

LicensingOptionalLicensesList  OBJECT-TYPE
        SYNTAX     OCTET STRING (SIZE(0..2048))
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The optional licenses this device unit is licensed for separated by ';'."
        ::= { Licensing 4 }

LicensingCryptoAlgorithmCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The number of Cryptographic Algorithms enabled by the device's licensing."
        ::= { Licensing 5 }

LicensingCryptoAlgorithmList  OBJECT-TYPE
        SYNTAX     OCTET STRING
        MAX-ACCESS read-only
```

```
        STATUS     current
        DESCRIPTION
                "List of licensed Cryptographic Algorithms. These will be descriptive strings
                for each algorithm terminated by a semi-colon ';'.
                The number of licensed algorithms in the list will be
                equal to LicensingCryptoAlgorithmCount."
        ::= { Licensing 6 }

EnabledHostCommandsCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The number of host commands enabled on this device."
        ::= { EnabledHostCommands 1 }

EnabledHostCommandsList  OBJECT-TYPE
        SYNTAX     OCTET STRING (SIZE(0..2048))
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Entire list of enabled host commands separated by a semi-colon ';'."
        ::= { EnabledHostCommands 2 }

LogsErrorlogTotalCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Total number of entries in the error log."
        ::= { LogsErrorlog 1 }

LogsAuditlogTotalCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Total number of entries in the audit log."
        ::= { LogsAuditlog 1 }

HealthDiagSelfTestTimeOfDay  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The number of minutes after midnight the diagnostic tests begins at."
        ::= { HealthDiagSelfTest 1 }

HealthDiagSelfTestOK  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "True unless one or more of the tests in the last self test failed."
        ::= { HealthDiagSelfTest 2 }

HealthDiagSelfTestCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
```

```
                "The number of self tests run last test cycle on this device."
        ::= { HealthDiagSelfTest 3 }

HealthDiagSelfTestList  OBJECT-TYPE
        SYNTAX     OCTET STRING (SIZE(0..2048))
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The complete results of the last run self test.
                Syntax of string Testname:testresult;
                Test/results pairs are separated by a colon  ':'
                They are delimited by a semi-colon ; .
                Results are the string 'passed' or 'failed'.
                The number of self tests in the list will be
                equal to HealthDiagSelfTestCount."
        ::= { HealthDiagSelfTest 4 }

HealthHealthCheckEnabled  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Whether the device is presently collecting health check data."
        ::= { HealthCheckCounts 1 }

HealthCheckCountsStartTime  OBJECT-TYPE
        SYNTAX     DateAndTime
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Date/Time of last health stats reset."
        ::= { HealthCheckCounts 2 }

HealthCheckCountsEndTime  OBJECT-TYPE
        SYNTAX     DateAndTime
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Date/time of when stats collecting was last disabled.
                If health stats are not disabled, the Date/Time when this field was read."
        ::= { HealthCheckCounts 3 }

HealthCheckCountsRebootCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Number of times the device rebooted since the last reset of health counters."
        ::= { HealthCheckCounts 4 }

HealthCheckCountsTamperCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The number of tampers detected since the last reset of health counters."
        ::= { HealthCheckCounts 5 }

HealthCheckCountsPinFailuresMinuteLimit  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
```

```
        STATUS     current
        DESCRIPTION
                "The number of times the maximum pin verify failures per minute was exceeded
                since the last reset of health counters."
        ::= { HealthCheckCounts 6 }

HealthCheckCountsPinFailuresHourLimit  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The number of times the maximum pin verify failures per hour was exceeded
                since the last reset of health counters."
        ::= { HealthCheckCounts 7 }

HealthCheckCountsPinAttackLimitExceeded  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The number of times the pin attack limit was exceeded since the last reset of
                health counters."
        ::= { HealthCheckCounts 8 }

HostConnectionEthernetEnabled  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Whether the device is configured to provide ethernet host interfaces."
        ::= { HostConnection 1 }

HostConnectionFICONEnabled  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Whether or not the device is configured to provide a FICON host interface."
        ::= { HostConnection 2 }

HostConnectionEthernetIfCount  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The number of physical ethernet interfaces enabled to process host commands."
        ::= { HostConnectionEthernet 1 }

HostConnectionEthernetSSLEnabled  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "If true TLS has been enabled."
        ::= { HostConnectionEthernet 2 }

HostConnectionEthernetACLsEnabled  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
```

```
                "Whether Access Control Lists are enabled."
        ::= { HostConnectionEthernet 3 }

HostConnectionEthernetUDPEnabled  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Whether UDP is enabled."
        ::= { HostConnectionEthernet 4 }

HostConnectionEthernetTCPEnabled  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Whether TCP is enabled."
        ::= { HostConnectionEthernet 5 }

HostConnectionEthernetMaxTCPConnections  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The maximum number of TCP sessions allowed per interface."
        ::= { HostConnectionEthernet 6 }

HostConnectionEthernetTCPKeepalive  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "TCP keep alive timeout in minutes.
                (0-120)"
        ::= { HostConnectionEthernet 7 }

HostConnectionEthernetWellKnownPortTCP  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The well known TCP port.
                (0-65535)"
        ::= { HostConnectionEthernet 8 }

HostConnectionEthernetWellKnownPortTLS  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The well known TLS port.
                (0-65535)"
        ::= { HostConnectionEthernet 9 }

HostConnectionEthernetTable  OBJECT-TYPE
        SYNTAX SEQUENCE OF psHostConnectionEthernetEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "The interface specific ethernet configuration/status information.
                For interfaces that are disabled or fields that are not relevant,
```

```
                all values will be set to a NULL value."
        ::= { HostConnectionEthernet 10 }

HostConnectionEthernetEntry  OBJECT-TYPE
        SYNTAX         psHostConnectionEthernetEntry
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "An entry containing information applicable to the host ethernet connection"
        INDEX { HostConnectionEthernetIndex }
        ::= { HostConnectionEthernetTable 1 }

psHostConnectionEthernetEntry ::= SEQUENCE {
        HostConnectionEthernetIndex
                Integer32,
        HostConnectionEthernetConfigured
                TruthValue,
        HostConnectionEthernetInterfaceNumber
                Integer32,
        HostConnectionEthernetConfigMethod
                INTEGER,
        HostConnectionEthernetHostName
                DisplayString,
        HostConnectionEthernetIpAddress
                IpAddress,
        HostConnectionEthernetSubnetMask
                IpAddress,
        HostConnectionEthernetGateway
                IpAddress,
        HostConnectionEthernetMacAddress
                MacAddress,
        HostConnectionEthernetPortSpeed
                DisplayString,
        HostConnectionNumberOfConnectionsUsed
                Integer32,
        HostConnectionEthernetLoadCount
                DisplayString
}

HostConnectionEthernetIndex  OBJECT-TYPE
        SYNTAX     Integer32 (1..2)
        MAX-ACCESS not-accessible
        STATUS     current
        DESCRIPTION
                "A unique value (> 0) for each host ethernet connection interface"
        ::= { HostConnectionEthernetEntry 1 }

HostConnectionEthernetConfigured  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Tells whether a row of the table is configured. For a 1 interface system it is
                possible that the configured  interface could be either the first or second one."
        ::= { HostConnectionEthernetEntry 2 }

HostConnectionEthernetInterfaceNumber  OBJECT-TYPE
        SYNTAX     Integer32 (1..2)
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
```

```
                "This binds a row to a physical host ethernet interface.
                1 corresponds to Ethernet interface HostPortEthernet1
                2 corresponds to Ethernet interface HostPortEthernet2"
        ::= { HostConnectionEthernetEntry 3 }

HostConnectionEthernetConfigMethod  OBJECT-TYPE
        SYNTAX     INTEGER {
                ipMethodUnknown (0),
                ipMethodStatic (1),
                ipMethodDHCP (2)
        }
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "DHCP and static IP configuration are the possibilities. When statically
                configured, the addresses reported are from the device's configuration
                for this interface. When DHCP, the addresses obtained via DHCP are reported."
        ::= { HostConnectionEthernetEntry 4 }

HostConnectionEthernetHostName  OBJECT-TYPE
        SYNTAX     DisplayString
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Network Name is present for a DHCP configured port only.
                If the ethernet port is configured statically set to 'NotApplicable'.
                If the ethernet port is not configured set to 'Interface not enabled'."
        ::= { HostConnectionEthernetEntry 5 }

HostConnectionEthernetIpAddress  OBJECT-TYPE
        SYNTAX     IpAddress
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "IP address for this interface.
                If this interface is not enabled returns 0.0.0.0"
        ::= { HostConnectionEthernetEntry 6 }

HostConnectionEthernetSubnetMask  OBJECT-TYPE
        SYNTAX     IpAddress
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Subnet mask for this interface.
                If this interface is not enabled returns 0.0.0.0"
        ::= { HostConnectionEthernetEntry 7 }

HostConnectionEthernetGateway  OBJECT-TYPE
        SYNTAX     IpAddress
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Gateway IP address used by this interface.
                If this interface is not enabled returns 0.0.0.0"
        ::= { HostConnectionEthernetEntry 8 }

HostConnectionEthernetMacAddress  OBJECT-TYPE
        SYNTAX     MacAddress
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
```

```
                "MAC address for this ethernet interface.
                If this interface is not enabled returns 00:00:00:00:00:00"
        ::= { HostConnectionEthernetEntry 9 }

HostConnectionEthernetPortSpeed  OBJECT-TYPE
        SYNTAX     DisplayString
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "Describes the speed/duplex this port is configured at.
                If the port is configured for 'ethernet autoselect' returns:
                    'ethernet autoselect(speed/duplex )'
                If the ethernet interface is not configured returns:
                    'Interface not enabled'"
        ::= { HostConnectionEthernetEntry 10 }

HostConnectionNumberOfConnectionsUsed  OBJECT-TYPE
        SYNTAX     Integer32
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The Number of TCP connections in use for this physical interface.
                range (0-HostConnectionMaxTCPConnections)"
        ::= { HostConnectionEthernetEntry 11 }

HostConnectionEthernetLoadCount  OBJECT-TYPE
        SYNTAX     DisplayString
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "The Load Count for the host machines"
        ::= { HostConnectionEthernetEntry 12 }

SettingsPCICompliant  OBJECT-TYPE
        SYNTAX     TruthValue
        MAX-ACCESS read-only
        STATUS     current
        DESCRIPTION
                "True if all security settings are PCI compliant."
        ::= { Settings 1 }

PowerOnAlarm NOTIFICATION-TYPE
        STATUS     current
        DESCRIPTION
                "SNMP Trap indicating the unit has been powered up."
        ::= { Traps 1 }

NewLicenseAlarm NOTIFICATION-TYPE
        STATUS     current
        DESCRIPTION
                "SNMP trap indicating the installation of a new license."
        ::= { Traps 4 }

-- Information on tamper
AlarmTamper OBJECT IDENTIFIER ::= { AlarmObjects 11 }

TamperCause OBJECT-TYPE
        SYNTAX     INTEGER {
                causeCaseTampered (9)
        }
        MAX-ACCESS accessible-for-notify
```

```
        STATUS     current
        DESCRIPTION
                "The cause of the tamper event."
        ::= { AlarmTamper 1 }

TamperDate OBJECT-TYPE
        SYNTAX     DateAndTime
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
                "The Date and Time that the tamper event occurred."
        ::= { AlarmTamper 2 }

TamperAlarm NOTIFICATION-TYPE
        OBJECTS    {
                TamperCause,
                TamperDate
        }
        STATUS     current
        DESCRIPTION
                "SNMP trap indicating a tamper has been detected. Detected tamper is reported."
        ::= { Traps 9 }

-- Information on fraud attempts
AlarmFraud OBJECT IDENTIFIER ::= { AlarmObjects 7 }

FraudType OBJECT-TYPE
        SYNTAX     INTEGER {
                fraudExceededFailuresPerMinute (1),
                fraudExceededFailuresPerHour (2),
                fraudExceededAttackLimit (3)
        }
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
                "Indicates what sort of fraud event took place."
        ::= { AlarmFraud 1 }

FraudAlarm NOTIFICATION-TYPE
        OBJECTS    {
                FraudType
        }
        STATUS     current
        DESCRIPTION
                "SNMP trap that indicates a fraud limit was exceeded."
        ::= { Traps 11 }

-- Information on diagnostic failures
AlarmDiagnostic OBJECT IDENTIFIER ::= { AlarmObjects 6 }

DiagnosticID OBJECT-TYPE
        SYNTAX     INTEGER {
                diagDES (1),
                diagAES (2),
                diagECDSA (3),
                diagHMAC (4),
                diagMD5 (5),
                diagSHA (6),
                diagRSA (7),
                diagRNG (8),
                diagBattery (12),
```

```
                diagFans (13),
                diagTemperature (14),
                diagVoltage (15)
        }
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
                "Diagnostic identifier as a code."
        ::= { AlarmDiagnostic 1 }

DiagnosticString OBJECT-TYPE
        SYNTAX     DisplayString
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
                "Diagnostic identifier as a string."
        ::= { AlarmDiagnostic 2 }

DiagnosticTestFailureAlarm NOTIFICATION-TYPE
        OBJECTS    {
                DiagnosticID,
                DiagnosticString
        }
        STATUS     current
        DESCRIPTION
                "SNMP trap indicating that a diagnostic test has failed. Indicates which test
                failed."
        ::= { Traps 12 }

-- Information on modified settings
AlarmSettingsModified OBJECT IDENTIFIER ::= { AlarmObjects 9 }

ModifiedSetting OBJECT-TYPE
        SYNTAX     INTEGER {
                settingSecurity (1),
                settingPCI (2),
                settingAuditedItems (3),
                settingEnabledPinblocks (4)
        }
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
                "Collection of settings whose modifications will trigger a notification."
        ::= { AlarmSettingsModified 1 }

SettingsModifiedAlarm NOTIFICATION-TYPE
        OBJECTS    {
                ModifiedSetting
        }
        STATUS     current
        DESCRIPTION
                "SNMP trap that indicates a setting of the device has been modified. It includes
                a ModifiedSetting object which identifies the setting that was changed."
        ::= { Traps 14 }

-- Information on the new device state
AlarmStateChange OBJECT IDENTIFIER ::= { AlarmObjects 10 }

DeviceState OBJECT-TYPE
        SYNTAX     INTEGER {
                stateUnavailable (1),
```

```
                stateOnline (2),
                stateOffline (3),
                stateSecure (4)
        }
        MAX-ACCESS accessible-for-notify
        STATUS     current
        DESCRIPTION
                "Provides the new state of the device."
        ::= { AlarmStateChange 1 }

DeviceStateAlarm NOTIFICATION-TYPE
        OBJECTS    {
                DeviceState
        }
        STATUS     current
        DESCRIPTION
                "SNMP trap that indicates a change in device state. It includes a DeviceState
                object which provides the new state of the device."
        ::= { Traps 15 }

END
```

Приложение Д

Авторизованные активности

Авторизуемые активности имеют вид <**категория**>.[<**подкатегория**>].[<**интерфейс**>] (при отсутствии параметров точки в конце не ставятся).

Таблица ниже содержит список поддерживаемых категорий и подкатегорий авторизуемых активностей. Требования к авторизации активностей зависят от конкретной команды хоста или консоли и могут различаться в зависимости от типа LMK — Variant или Key Block.

В качестве интерфейса могут быть указаны **host** или **console**.

Примеры наименований активностей:

• авторизация активности **export.001.host** позволяет экспортировать ZPK (тип ключа 001) с помощью команды хоста (например, 'A8')

• авторизация активности **export** позволяет экспортировать любой допустимый ключ с помощью команды хоста или консоли;

• авторизация активности **export..console** позволяет экспортировать любой допустимый ключ с помощью команды консоли.

Примечание: при использовании Variant LMK необходимость авторизации для возможности выполнения действия (генерации (Г), импорта (И) или экспорта (Э)) с определенным ключом определяется с помощью таблицы типов ключей (ТТК, см. разд. 6.3.5). Если в поле, соответствующем определенному ключу и требуемому действию, указан флаг 'Б', для выполнения данного действия с этим ключом авторизация не требуется. Активности **genprint.*** и **component.*** не проверяют ТТК и требуются всегда.

	Категория	**Подкатегория**
Variant LMK	generate	000 001 002 003 006 008 009 00a 00b 107 109 10c 200 207 209 302 307 309 30b 30d 402 407 409 40d 507 509 50d 607 609 709 70d 809 80d 909 90d rsa
	export	000 001 002 003 006 008 009 00a 00b 107 109 200 207 209 302 307 309 30b 30d 402 407 409 40d 507 509 50d 607 609 709 70d 809 80d 909 90d hmac rsa
	import	000 00a 107 207 30b rsa
	genprint	000 001 002 003 006 008 009 00a 00b 107 109 200 207 209 302 307 309 30b 30d 402 407 409 40d 507 509 50d 607 609 709 70d 809 80d 909 90d
	component	000 001 002 003 006 008 009 00a 00b 107 109 200 207 209 302 307 309 30b 30d 402 407 409 40d 507 509 50d 607 609 709 70d 809 80d 909 90d

Key Block LMK	generate	03 05 06
	export	01 03 11 12 13 21 22 23 24 25 31 32 33 34 35 36 37 38 39 40 41 42 43 44 47 48 49 51 52 53 54 55 61 62 63 64 65 71 72 73 b0 b1 c0 d0 e0 e1 e2 e3 e4 e5 e6 e7 k0 k1 m0 m1 m2 m3 m4 m5 m6 p0 v0 v1 v2
	import	01 02 03 11 12 13 21 22 23 24 25 31 32 33 34 35 36 37 38 39 40 41 42 43 44 47 48 49 51 52 53 54 55 61 62 63 64 65 71 72 73 b0 b1 c0 d0 e0 e1 e2 e3 e4 e5 e6 e7 k0 k1 m0 m1 m2 m3 m4 m5 m6 p0 v0 v1 v2
	genprint	01 11 12 13 21 22 23 24 25 31 32 33 34 35 36 37 38 39 40 41 42 43 44 47 48 49 51 52 53 54 55 71 72 73 b0 b1 c0 d0 e0 e1 e2 e3 e4 e5 e6 e7 k0 k1 m0 m1 m2 m3 m4 m5 m6 p0 v0 v1 v2
	component	01 11 12 13 21 22 23 24 25 31 32 33 34 35 36 37 38 39 40 41 42 43 44 47 48 49 51 52 53 54 55 71 72 73 b0 b1 c0 d0 e0 e1 e2 e3 e4 e5 e6 e7 k0 k1 m0 m1 m2 m3 m4 m5 m6 p0 v0 v1 v2