Visa Merchant Data Secure with Point to Point Encryption (VMDS with P2PE)

Hardware Security Module Guide

Version 2.0

May, 2013

Version History

Version	Date	Revision Notes
1.0	February 7, 2013	Initial Publication
2.0	May, 2013	Minor updates

Table of Contents

Tables

1 Introduction

Merchant data compromises continue to increase. As a result, merchants prefer not to see, store, or have to protect card data and are looking for solutions. Point-to-Point encryption is a key pillar of Visa's security and processing strategies. This document will focus on Visa Merchant Data Secure with Point-to-Point Encryption (VMDS with P2PE), and what Hardware Security Module (HSM) manufacturers will need to do to enable this service in their products. The Visa Merchant Data Secure service will follow the requirements set out in the *PCI Point-to-Point Encryption: Solution Requirements – Encryption, Decryption, and Key Management within Secure Cryptographic Devices (Hardware/Hardware)* document released in April 2012 by the Payment Card Industry Security Standards Council (PCI SSC).

1.1 About Visa and VisaNet

VisaNet is the most comprehensive retail electronic payment network in the world, consisting of a family of integrated systems that support and deliver Visa's payment processing services to its participants. These services include authorization, clearing and settlement, risk management, information, merchant, acquirer and issuer solutions. Please contact your Visa representative for details.

Through continued development over the last 20 plus years, VisaNet now connects over 21,000 financial institutions (FIs) in over 200 countries worldwide. The network utilizes high-speed secure connections with redundant routing to maximize availability.

1.2 Scope

This Guide focuses on the information needed to understand Visa Merchant Data Secure with P2PE processing and requirements. It includes basic technical information and high-level tasks required to implement the service in Point-of-Interface (POI) terminals or devices and terminal and host applications.

It is organized into the following sections:

Section 1: Introduction – introduction to the product and Visa

Section 2: Visa Merchant Data Secure with P2PE Overview – basic overview of the product

Section 3: How it Works – description of the solution

Section 4: Point of Decryption Requirements – detailed requirements for the point where decryption or transaction is taking place.

Section 5: HSM Functions Required for VMDS – describes the encryption and decryption functions that are necessary in the HSM for processors to use VMDS

Section 6: Standard Encryption Option – detailed requirements for implementing data protection via standard encryption with separate encryption blocks.

Section 7: Visa Format Preserving Encryption Option – detailed requirements for implementing the Visa Format Preserving Encryption algorithm.

Section 8: Supporting Information – helpful information to those implementing the service.

Appendix A: Glossary – glossary of terms and acronyms used in this document.

1.3 PCI SSC P2PE requirements

Terminal and HSM device manufacturers that want to participate in the VMDS with P2PE service will implement the service in compliance with the PCI SSC P2PE Solution Requirements, and will, as part of their partnership with Visa, perform all of the appropriate PCI SSC validations.

1.4 Related Documentation

Key materials referenced throughout this Guide are listed below. Please ensure you are using the latest versions of the documents applicable to your implementation, which may vary based on your market.

https://www.pcisecuritystandards.org/documents/P2PE_%20v%201-1.pdf

https://www.pcisecuritystandards.org/documents/P2PE_Program_Guide_June_2012_v1.pdf

https://www.pcisecuritystandards.org/documents/P2PE_glossary_v1-1.pdf

2 Visa Merchant Data Secure with P2PE Overview

This product will provide field-level encryption services to protect sensitive cardholder data from the Point of Interaction (POI) through the merchant's environment and the payment network infrastructure to VisaNet. This protection is expected to lower merchant PCI compliance costs and reduce compromise risk for merchants.

The POI may be a point of sale terminal that supports any method of card number entry, such as magnetic track read, contact and contactless chip read, key entered, etc. It may also be a device that attaches to a terminal to perform the card read and encryption.

VisaNet will be enhanced to provide decryption services for transactions that utilize the new field-level encryption scheme being developed with this project.

3 How it Works

This section provides an overview of how Visa Merchant Data Secure with P2PE works, including the processing and settlement flows.

3.1 Solution Overview

Merchants that implement field-level encryption at the point of service will typically encrypt all payment card transactions that occur at that terminal, regardless of brand.

For card present transactions, the field-level encryption will occur within a Secure Cryptographic Device (SCD) that is tamper resistant. This is referred to as a 'hardware' encryption solution, and will leverage existing PIN key management and encryption standards for these devices.

The service will have two methods:

- Standard Encryption Option
- Format-Preserving Encryption (FPE) Option

Both methods require the following features within the POI device:

- Symmetric Key Management using the Derived Unique Key Per Transaction (DUKPT) method, as defined in ANS X9.24.
- The Data encryption key variant of the DUKPT method shall be used for protecting the data described in this document. This is separate and unique from the PIN encryption variant that is commonly used in devices today.
- The same Initial Key shall be used for generation of the transaction unique and separate PIN Encryption key variant and Data encryption key variant for a given transaction in the device
- PIN, when used, and Data encryption shall use the same Key Serial Number for a given transaction
- The Triple-DES encryption algorithm (TDEA) with double-length keys shall be used for all data encryption operations.

Note: PIN encryption standards are currently being enhanced to support AES cryptographic technology. We envision adding support for AES in the future when it is standardized and used for PIN processing.

Field-level encrypted payment card transactions will be sent to a host for authorization. The host may pass the encrypted data unaltered or translated to VisaNet or their acquiring processor for decryption.

Transaction data capture, which may be performed at the terminal or host for settlement/reconciliation, will contain encrypted data fields from the authorization. When the authorization was sent to VisaNet for decryption, the capture files will also be sent to VisaNet for decryption. Visa will forward the unencrypted capture files to their acquiring processor.

The terminal/device will use the same Base Derivation Key for PIN encryption as for VMDS data encryption. PIN will use the PIN derivative and VMDS will use the data derivative. Please note that this is accomplished via a single terminal/device injection. We do not require reinjection, as long as the entity that owns the BDK can securely convey it to the decryption or translation point.

The terminal/device will use the same KSN for PIN data as for VMDS data within a transaction.

4 Point of Decryption Requirements

The following requirements apply to the usage of a Hardware Security Module to perform decryption or translation of data that was encrypted at the point of service with the VMDS service.

4.1 Processors

The VMDS service allows for protection or visibility within many different environments to address the various processing needs of different organizations. To support the requirements of these parties, a processor utilizing an HSM may use one or both of the following modes of operation:

- Decryption of data – used when the processor is the final decryption point
- Translation of data – used when the processor does not have a need for the decrypted data and the next party also participates in the VMDS service. The protected data is never visible in plain text form to the host.

The following illustrates some of the potential processor host scenarios that may deploy HSMs for one or both of the modes. Other types of organizations and processing models may also use VMDS.

Merchant Host or Gateway – when routing to more than one acquirer or network

- Uses decryption when routing to a party that does not support VMDS
- Uses translation when routing to a party that supports VMDS

Acquirer Processor –

- May use decryption function
- May use translation when routing to card network that supports VMDS

Card Network –

- Uses decryption function

The processor that is performing translation or decryption will be required to:

- Implement the appropriate key management for the source and destination zones
- Identify the specific transactions that have had VMDS with P2PE applied and the options used to perform translation or decryption
- Utilize an HSM to perform decryption or translation securely without exposing keys or translated data
- Update all occurrences of the protected data in the payment transaction with decrypted or translated values prior to sending to the next party

4.2 Hardware Security Modules

Processors using the VMDS service may receive encrypted data with two different key management schemes and two different VMDS encryption options. Processors shall utilize HSMs for decrypting or translating the

data protected with VMDS. When performing translation, the destination zone must always utilize the standard encryption option using data zone encryption keys.

The following matrix summarizes the combinations of key management schemes and VMDS encryption options that may be utilized for encrypted data with the VMDS service.

Table 4-1: Supported Key Management Schemes and VMDS Encryption Options

FUNCTION	SOURCE		TARGET	
	KEY MANAGEMENT	VMDS OPTION	KEY MANAGEMENT	VMDS OPTION
Decryption	DUKPT	Standard	n/a	n/a
		VFPE		
	Zone Encryption Keys	Standard		
Translation	DUKPT	Standard	Zone Encryption Keys	Standard
		VFPE		
	Zone Encryption Keys	Standard		
Encryption[1]	n/a	n/a	DUKPT	Standard
				VFPE
			Zone Encryption Keys	Standard

4.2.1 DUKPT Requirements

VMDS utilizes the transaction unique general purpose Data Encryption keys generated by the DUKPT process at the point of service for data encryption, as described in ANS X9.24, part 1.

Two Key Serial Number (KSN) formats are currently in use in the market – both of which shall be supported for VMDS. Please note that the Key Set Identifier (KSI) is a different length in the two formats, both of which support 4-bit characters in the BASE-16 alphabet (0-9 and A-F). The two formats are summarized below.

Table 4-2: Supported DUKPT Key Serial Number Formats

	KEY SET IDENTIFIER	DEVICE ID	TRANSACTION COUNTER
Old (8 bytes)	6 digits (3 bytes)	19 bits	21 bits
New (10 bytes)	10 digits (5 bytes)		

VMDS currently only utilizes the “Data encryption, request or both ways” variant for key calculation of transaction unique Data Encryption Keys.

4.2.2 Zone Encryption Key Requirements

Double-length Triple-DES symmetric keys are used with the VMDS service. When used, they shall be used for the sole purpose of general data encryption between parties (transient), and must not be the same as PIN, MAC or other specific encryption keys.

Standard key conveyance and protection schemes shall also be utilized for general data encryption keys, as defined in ANS X9.24.

[1] Encryption functions are required to support processor testing requirements.

5 HSM Functions Required for VMDS

To accommodate different processing scenarios, several HSM functions are required. This chapter describes these functions that are required to support the VMDS service.

Visa Merchant Data Secure with Point-to-Point Encryption supports two basic options for encryption of data. The commands listed here utilize one or both of these options, which are explained in further detail in other chapters within this document, as noted below.

- **Standard Encryption Option** – where data is formatted into blocks and then encrypted via normal Triple-DES encryption with double-length keys. Details of block formatting may be found in section 6, "Standard Encryption Option".
- **Visa Format Preserving Option (VFPE)** – where data is encrypted in place, without changing the data type or length of the field. VFPE details may be found in section 7, "Visa Format Preserving Encryption Option".

VMDS uses **DUKPT** key management at the point of service where the sensitive payment details are encrypted. It also provides support for **zone encryption**, where fixed or dynamic keys are used for encryption of sensitive payment data between parties.

This section identifies the functions that are required to support the valid combinations of key management and encryption options. While these functions are discretely listed, it should not be construed as a requirement for the HSM to deploy them as separate commands.

5.1 Data Encryption Functions

5.1.1 Encrypt Standard Option with DUKPT

The following information is required for this encryption function:

- Plain Text Data to be encrypted
- Source data type (e.g., ASCII text, EBCDIC text, 4-bit hexadecimal, etc.)
- Target Encryption Block Type – ASCII text or 4-bit hexadecimal digits
- DUKPT Key Serial Number (KSN)
- Base Derivation Key (BDK)

The function shall encrypt the data in the following manner:

1. Convert the data from the Source Data Type into the format required for the encryption block as specified in Target Encryption Block Type, if different
2. Format one or more structured encryption blocks in the format specified in the Target Encryption Block Type with the data and all necessary control, random, length and fill characters
3. Calculate the transaction specific data encryption key from the supplied BDK, using the "Data encryption, request or both ways" variant.

4. The data blocks shall then be encrypted using the calculated transaction specific data encryption key, with TDES in CBC mode.

The function shall return the following to the calling application host:

- Encrypted data block(s)

5.1.2 Encrypt VFPE Option with DUKPT

The following information is required for this encryption function:

- Plain Text Data to be encrypted
    - If PAN data, requires full PAN (including digits that will not be encrypted) and indication that data is a PAN
- Source Data Type (e.g., ASCII text, EBCDIC text, 4-bit hexadecimal, etc.)
- VFPE Alphabet Identifier
- DUKPT Key Serial Number (KSN)
- Base Derivation Key (BDK)

The function shall encrypt the data in the following manner:

1. The function shall calculate the transaction specific data encryption key from the supplied BDK, using the “Data encryption, request or both ways” variant.
2. Using the appropriate Alphabet table, the Plain Text Data shall be converted from its Source Data Type to the equivalent alphabet number character.
3. Once converted to the alphabet, the data is encrypted with the VFPE algorithm using the calculated transaction specific data encryption key.
4. After encryption, the encrypted data is converted from the alphabet number character into the Source Data Type.

Please note that special processes are used in steps 2 through 4 when the encrypted data is the PAN, as noted in section 7.2, “Primary Account Number (PAN)”.

The function shall return with at least the following to the calling application host:

- Encrypted data in the Source Data Type format
- If data contains PAN: VFPE MOD10 signal

5.1.3 Encrypt Standard Option with Zone Encryption Keys

The following information is required for this encryption function:

- Plain Text Data to be encrypted
- Source data type (e.g., ASCII text, EBCDIC text, 4-bit hexadecimal, etc.)
- Target Encryption block type – ASCII text or 4-bit hexadecimal digits
- Target Zone Encryption Key

The function shall encrypt the data in the following manner:

1. Convert the data from the Source Data Type into the format required for the encryption block as specified in Target Encryption Block Type, if different
2. Format one or more structured encryption blocks in the format specified in the Target Encryption Block Type with the data and all necessary control, random, length and fill characters
3. Perform TDES decryption with the Target Zone Encryption key, using CBC mode

The function shall return with at least the following to the calling application host:

- Encrypted data block(s)

5.2 Data Decryption Functions

5.2.1 Decrypt Standard Option with DUKPT

The following information is required for this decrypt function:

- Encrypted data
- Encryption block type – ASCII text or 4-bit hexadecimal digits
- DUKPT Key Serial Number (KSN)
- Base Derivation Key (BDK)
- Target data type (e.g., ASCII text, EBCDIC text, 4-bit hexadecimal, etc.)

The function shall decrypt the data in the following manner:

1. Calculate the transaction specific data encryption key from the supplied BDK, using the “Data encryption, request or both ways” variant.
2. The data blocks shall then be decrypted using the calculated transaction specific data encryption key, with TDES in CBC mode.
3. Using the encryption block type, extract the actual data from within the structured block (do not include block formatting characters, such as random or fill characters in the extracted data)
4. Convert the extracted data to desired target data type

The function shall return with at least the following to the calling application host:

- The unencrypted data field (not the decrypted encryption block), in the target data type format

5.2.2 Decrypt VFPE Option with DUKPT

The following information is required for this decrypt function:

- Encrypted data
    - If PAN data, requires full PAN (including non-encrypted digits) and indication that encrypted data is a PAN
- If Encrypted data contains PAN: VFPE MOD10 signal (may use expiration date)

- Source encoding format (e.g., ASCII text, EBCDIC text, 4-bit hexadecimal, etc.)
- VFPE Alphabet Identifier
- DUKPT Key Serial Number (KSN)
- Base Derivation Key (BDK)

The function shall decrypt the data in the following manner:

1. The function shall calculate the transaction specific data encryption key from the supplied BDK, using the "Data encryption, request or both ways" variant.
2. The encrypted data shall be converted from its source encoding format to the equivalent alphabet number.
3. Once converted to the alphabet, it is decrypted with the VFPE algorithm using the calculated transaction specific data encryption key.
4. After decryption, the decrypted data is converted into the source encoding format.

Please note that special processes are used in steps 2 through 4 when the encrypted data is the PAN, as noted in section 7.2, "Primary Account Number (PAN)".

The function shall return with at least the following to the calling application host:

- The decrypted data in the source encoding format

5.2.3 Decrypt Standard Option with Zone Encryption Keys

The following information is required for this decrypt function:

- Encrypted data
- Encryption block type – ASCII text or 4-bit hexadecimal digits
- Source Zone Encryption Key (of encrypted data)
- Target data type (e.g., ASCII text, EBCDIC text, 4-bit hexadecimal, etc.)

The function shall decrypt the data in the following manner:

1. Perform TDES decryption with the source zone encryption key, using CBC mode
2. Using the encryption block type, extract the actual data from within the structured block (do not include block formatting characters, such as random or fill characters in the extracted data)
3. Convert the extracted data to desired target data type

The function shall return with at least the following to the calling application host:

- The unencrypted data field (not the decrypted encryption block), in the target data type format

5.3 Data Translation Functions

Processors may use translation functions to avoid having plain text payment data within their environment. The VMDS translation functions utilize zone encryption, similar to PIN zone encryption. The target for

translation always uses Zone Encryption Keys and the Standard Encryption Option – VFPE is not a supported target mode for translation functions.

5.3.1 Translate Standard Option with DUKPT

The following information is required for this translate function:

- Encrypted data
- DUKPT Key Serial Number (KSN)
- Base Derivation Key (BDK)
- Target Zone Encryption Key (to translate data to)

The function shall translate the data in the following manner:

1. Calculate the transaction specific data encryption key from the supplied BDK, using the "Data encryption, request or both ways" variant.
2. The data blocks shall then be decrypted using the calculated transaction specific data encryption key, with TDES in CBC mode.
3. The unencrypted data blocks shall then be re-encrypted using the supplied zone encryption key with TDES in CBC mode.

The function shall return with at least the following to the calling application host:

- The translated (re-encrypted) data encryption block(s)

The translated data shall not be visible outside of the HSM in its decrypted (plain text) form.

5.3.2 Translate VFPE Option with DUKPT

The following information is required for this translate function:

- Encrypted data
    - If PAN data, requires full PAN (including non-encrypted digits) and indication that encrypted data is a PAN
- If encrypted data contains PAN: VFPE MOD10 signal (may use expiration date)
- Source encoding format (e.g., ASCII text, EBCDIC text, 4-bit hexadecimal, etc.)
- VFPE Alphabet Identifier
- DUKPT Key Serial Number (KSN)
- Base Derivation Key (BDK)
- Target Zone Encryption Key (to translate data to)
- Target Encryption Block Type – ASCII text or 4-bit hexadecimal digits

The function shall translate the data in the following manner:

1. The function shall calculate the transaction specific data encryption key from the supplied BDK, using the "Data encryption, request or both ways" variant.

2. The encrypted data shall be converted from its source encoding format to the equivalent alphabet number.
3. Once converted to the alphabet, it is decrypted with the VFPE algorithm using the calculated transaction specific data encryption key.
4. Using the Target Encryption Block Type, convert the decrypted data into either normal ASCII encoding or 4-bit hexadecimal digits
5. Format one or more encryption block(s) with the decrypted data into the format specified in the Target Encryption Block Type
6. Encrypted the encryption block(s) using the supplied zone encryption key with normal TDES in CBC mode.

Please note that special processes are used in steps 2 through 4 when the encrypted data is the PAN, as noted in section 7.2, "Primary Account Number (PAN)".

The function shall return with at least the following to the calling application host:

- The translated (re-encrypted) data encryption block(s)

The translated data shall not be visible outside of the HSM in its decrypted (plain text) form.

5.3.3 Translate Standard Option with Zone Encryption Keys

The following information is required for this translate function:

- Encrypted data
- Source Zone Encryption Key (of encrypted data)
- Target Zone Encryption Key (to translate data to)

The function shall translate the data in the following manner:

1. Decrypt the supplied encrypted data block(s) with the Source Zone Encryption Key, using TDES in CBC mode
2. Encrypt the decrypted encryption data block(s) using the Target Zone Encryption Key, using TDES in CBC mode.

The function shall return with at least the following to the calling application host:

- The translated (re-encrypted) data encryption block(s)

The translated data shall not be visible outside of the HSM in its decrypted (plain text) form.

5.4 PIN Translation Functions

PIN transaction processors that use data translation functions to avoid plain text payment data within their environment will also require new or enhanced PIN translation functions.

Encrypted PIN blocks use a special blocking format that incorporates the PAN within it. PIN translation functions require the PAN to perform the decryption / re-encryption within the HSM.

PIN translation functions shall be made available to these processors to translate PINs with encrypted PAN data. For transactions that contain encrypted PIN and PAN data, VMDS requires that the same key management scheme and type of keys are used for both. For example, if the PIN was encrypted using DUKPT keys, the PAN data was also encrypted using DUKPT. Similarly, if the PIN was zone encrypted, the PAN data shall also be zone encrypted.

For all of these functions, the decrypted PAN shall not be visible outside of the HSM.

The functions listed below each shall accommodate all currently supported PIN block formats.

5.4.1 DUKPT PIN Translation with Standard Option Encrypted PAN

When DUKPT was used to encrypt the PIN, it shall also be used for encrypting the PAN and other data elements.

VMDS currently requires that the same Base Derivation Key (BDK) and Key Serial Number (KSN) are used for both PIN and Data, when both are present and encrypted.

This translation function requires the full encrypted PAN block (all 16 bytes) to be presented as input to the PIN translation function.

The HSM shall decrypt the PAN block and use the appropriate positions within it for translating the PIN. The decrypted PAN shall not be visible outside of the HSM.

5.4.2 DUKPT PIN Translation with VFPE Option Encrypted PAN

When DUKPT was used to encrypt the PIN, it shall also be used for encrypting the PAN and other data elements.

VMDS currently requires that the same Base Derivation Key (BDK) and Key Serial Number (KSN) are used for both PIN and Data, when both are present and encrypted.

This translation function requires the VFPE encrypted PAN (all positions, not just the encrypted positions) to be presented as input to the PIN translation function.

The HSM shall decrypt the proper positions within the PAN and then use the appropriate PAN positions for translating the PIN. The decrypted PAN shall not be visible outside of the HSM.

5.4.3 Zone Encryption Key PIN Translation with Encrypted PAN

This translation function requires the full encrypted PAN block (all 16 bytes) and the zone encryption key used to encrypt the PAN to be presented as input to the PIN translation function. Please note that the zone encryption key used to encrypt the PIN is separate and distinct from the zone encryption key used to encrypt the PAN.

The HSM shall decrypt the PAN block and use the appropriate positions within it for translating the PIN. The decrypted PAN shall not be visible outside of the HSM.

6 Standard Encryption Option

The Triple-DES encryption algorithm (TDEA) shall be used with Cipher Block Chaining (CBC) for encrypting all of the fields protected with this option. Data that is to be encrypted shall be converted into a specific encoding and then formatted into one or more 8-byte (64-bit) blocks. Two encodings are currently supported: 4-bit hexadecimal and normal ASCII. This section identifies the specific encoding method that must be used for each field (a field is always encoded with a specific method).

Each of these encryption blocks is specially formatted prior to encipherment with a consistent structure to facilitate processing and additional security. When more than one encryption block is used for a given data element, the first block contains additional control information that is not present in the remaining blocks.

- A control field *(first block only)*
- A second control field *(first block only)*
- Randomization digits or characters *(first block only).* Randomization method should conform to X9.82.
- A two-digit data length field – contains the number of data digits or characters to be encrypted *(first block only).* Excludes control, reserved, length, fill and randomization digits or characters
- Data element to be encrypted – converted to normal 7-bit ASCII or 4-bit hexadecimal digits, as required for the specific data element
- Remaining unused positions: Fill digits or characters in unused data positions *(these positions are used as needed in the final block, number of positions varies based on data length)*

6.1 Primary Account Number (PAN)

6.1.1 Encryption Block Formatting

Two 64-bit encryption blocks have been constructed with the full PAN in 4-bit hexadecimal digits. Please note that this is not the same encoding format used in Track 1 or 2 – the encrypting device converts from the modified 5-bit or 7-bit ASCII into 4-bit hexadecimal digits prior to encipherment. Please refer to the BASE-10 Alphabet table for conversion details.

Encryption blocks containing the PAN element are always encoded in 4-bit hexadecimal format. This is also true for translated blocks.

First block

Bits	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40	41-44	45-48	49-52	53-56	57-60	61-64
Data	*C1*	*C2*	*R*	*R*	*R*	*R*	*R*	*R*	*R*	*R*	*L1*	*L2*	*D*	*D*	*D*	*D*

Second block

Bits	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40	41-44	45-48	49-52	53-56	57-60	61-64
Data	*D*	*D*	*D*	*D*	*D*	*D*	*D*	*D*	*D*	*D*	*D*	*D/F*	*D/F*	*D/F*	*D/F*	*F*

where:
C1 = Control field 1 - shall be binary 0000
C2 = Control field 2 - shall be binary 0000
R = Randomization digit
- 4-bit field containing a random digit

L1 & L2 = Length of data - two 4-bit decimal digits containing the length of the PAN data that is present in the encryption block
e.g., L1=0001 (1) and L2=0110 (6) indicates 16 digits are present
D = PAN digit - 4-bit field with permissible values of 0000 (zero) to 1001 (9)
D/F = PAN/Fill digit - Designation of these fields is determined by the length field
F = Fill digit - shall be binary 1111 (F)

PAN Example 1: 16-digit PAN=1234567890123456

Bits	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40	41-44	45-48	49-52	53-56	57-60	61-64
1st blk	**0**	**0**	**C**	**1**	**7**	**F**	**D**	**4**	**8**	**2**	**1**	**6**	**1**	**2**	**3**	**4**
2ndblk	**5**	**6**	**7**	**8**	**9**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**F**	**F**	**F**	**F**

PAN Example 2: 19-digit PAN=1234567890123456789

Bits	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40	41-44	45-48	49-52	53-56	57-60	61-64
1st blk	**0**	**0**	**2**	**B**	**0**	**7**	**0**	**E**	**3**	**7**	**1**	**9**	**1**	**2**	**3**	**4**
2ndblk	**5**	**6**	**7**	**8**	**9**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**F**

PAN Example 3: 15-digit PAN=123456789012345

Bits	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40	41-44	45-48	49-52	53-56	57-60	61-64
1st blk	**0**	**0**	**0**	**F**	**A**	**9**	**4**	**8**	**2**	**F**	**1**	**5**	**1**	**2**	**3**	**4**
2ndblk	**5**	**6**	**7**	**8**	**9**	**0**	**1**	**2**	**3**	**4**	**5**	**F**	**F**	**F**	**F**	**F**

6.2 Cardholder Name

6.2.1 Encryption Block Formatting

Two or more 64-bit encryption blocks have been constructed with the full cardholder name field in normal 7-bit ASCII-encoded characters. Please note that this is not the same encoding format used in Track 1 – the encrypting device converts from the modified 7-bit ASCII into normal 7-bit ASCII encoded characters prior to encipherment. Please refer to the Track 1 Alphabet table for conversion details.

Encryption blocks containing the cardholder name element are always encoded in normal 7-bit ASCII format. This is also true for translated blocks.

First block

Bits	1-8	9-16	17-24	25-32	33-40	41-48	49-56	57-64
Data	*C1*	*C2*	*R*	*R*	*R*	*R*	*L1*	*L2*

Blocks Two through Five (as needed)

Bits	1-8	9-16	17-24	25-32	33-40	41-48	49-56	57-64
Data	*D*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*

where
C1 = Control field 1 - shall be binary 00100000 (a blank in ASCII text format)
C2 = Control field 2 - shall be binary 00100000 (a blank in ASCII text format)
R = Randomization character
- Contains a random 7-bit ASCII character.

L1 & L2 = Length of data
- two 7-bit ASCII digits containing the length (number of characters) of the Cardholder Name data field that is present in the encryption block
e.g., L1=00110010 (ASCII '2') and L2=00110110 (ASCII '6') indicates twenty six characters are present

D = Cardholder Name character
- 7-bit ASCII character

D/F = Cardholder Name character/Fill character
- Designation of these fields is determined by the length field

F = Fill character - shall be binary 00100001 ('!' in ASCII text format)

Cardholder Name Example 1: "NAME/ "

Bits	1-8	9-16	17-24	25-32	33-40	41-48	49-56	57-64
1st blk			**Z**	**0**	**S**	**A**	**0**	**6**
2ndblk	**N**	**A**	**M**	**E**	**/**		**!**	**!**

Cardholder Name Example 2: "LASTNAME/FIRSTNAME M.TITLE"

Bits	1-8	9-16	17-24	25-32	33-40	41-48	49-56	57-64
1st blk			**7**	**4**	**H**	**3**	**2**	**6**
2ndblk	**L**	**A**	**S**	**T**	**N**	**A**	**M**	**E**
3rdblk	**/**	**F**	**I**	**R**	**S**	**T**	**N**	**A**
4thblk	**M**	**E**		**M**	**.**	**T**	**I**	**T**
5thblk	**L**	**E**	**!**	**!**	**!**	**!**	**!**	**!**

6.3 Track 1 Discretionary Data

6.3.1 Encryption Block Formatting

Two or more 64-bit encryption blocks have been constructed with the full Track 1 Discretionary Data field in normal 7-bit ASCII-encoded characters. Please note that this is not the same encoding format used in Track 1 – the encrypting device converts from the modified 7-bit ASCII into normal 7-bit ASCII encoded characters prior to encipherment. Please refer to the Track 1 Alphabet table for conversion details.

Encryption blocks containing the Track 1 Discretionary Data element are always encoded in normal 7-bit ASCII format. This is also true for translated blocks.

First block

Bits	1-8	9-16	17-24	25-32	33-40	41-48	49-56	57-64
Data	*C1*	*C2*	*R*	*R*	*R*	*R*	*L1*	*L2*

Blocks Two through Eight (as needed)

Bits	1-8	9-16	17-24	25-32	33-40	41-48	49-56	57-64
Data	*D*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*

where
C1 = Control field 1 - shall be binary 00100000 (blank in ASCII text format)
C2 = Control field 2 - shall be binary 00100000 (blank in ASCII text format)
R = Randomization character
- Contains a random 7-bit ASCII character

L1 & L2 = Length of data
- two 7-bit ASCII numbers containing the length of the Track 1 Discretionary Data field that is present in the encryption blocks
e.g., L1=00110000 (ASCII '0') and L2=00110101 (ASCII '5') indicates five characters are present

D = Discretionary Data character
- 7-bit ASCII characters

D/F = Discretionary Data character/fill character
- Designation of these fields is determined by the length field

F = Fill character - shall be binary 00100001 ('!' in ASCII text format)

Track 1 Discretionary Data – Example 1: "84923"

Bits	1-8	9-16	17-24	25-32	33-40	41-48	49-56	57-64
1st blk			**Q**	**3**	**8**	**Y**	**0**	**5**
2ndblk	**8**	**4**	**9**	**2**	**3**	**!**	**!**	**!**

Track 1 Discretionary Data – Example 2:
"ABCDEFGHIJKLMNOPQRSTUVWXYZ12ABCDEFGHIJKLMNOPQRSTUVWXYZ"

Bits	1-8	9-16	17-24	25-32	33-40	41-48	49-56	57-64
1st blk			**V**	**B**	**S**	**P**	**5**	**4**
2ndblk	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
3rdblk	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**
4thblk	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**
5thblk	**Y**	**Z**	**1**	**2**	**A**	**B**	**C**	**D**
6thblk	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**
7thblk	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**
8thblk	**U**	**V**	**W**	**X**	**Y**	**Z**	**!**	**!**

6.4 Track 2 Discretionary Data

6.4.1 Encryption Block Formatting

One or more 64-bit encryption blocks have been constructed with the full Track 2 Discretionary Data field in 4-bit hexadecimal digits. Please note that this is not the same encoding format used in Track 2 – the encrypting device converts from the modified 5-bit ASCII into 4-bit hexadecimal digits prior to encipherment. Please refer to the BASE-16 Alphabet table for conversion details.

Encryption blocks containing the Track 2 Discretionary data element are always encoded in 4-bit hexadecimal format. This is also true for translated blocks.

First block

Bits	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40	41-44	45-48	49-52	53-56	57-60	61-64
Data	*C1*	*C2*	*R*	*R*	*R*	*R*	*R*	*R*	*R*	*R*	*L1*	*L2*	*D*	*D/F*	*D/F*	*D/F*

Second block (if needed)

Bits	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40	41-44	45-48	49-52	53-56	57-60	61-64
Data	*D*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*D/F*	*F*

where

C1 = Control field 1 - shall be binary 0000
C2 = Control field 2 - shall be binary 0000
R = Randomization digit
- 4-bit field containing a random digit
L1 & L2 = Length of data
- two 4-bit decimal digits containing the length of the Track 2 Discretionary Data that is present in the encryption block
e.g., L1=0001 (1) and L2=1001 (9) indicates 19 digits are present
D = Discretionary Data digit
- 4-bit field with permissible values of 0000 (zero) to 1111 (F)
D/F = Discretionary Data/Fill digit - Designation of these fields is determined by the length field
F = Fill digit - shall be binary 1111 (F)

Track 2 Discretionary Data – Example 1: "12345"

Bits	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40	41-44	45-48	49-52	53-56	57-60	61-64
1st blk	**0**	**0**	**5**	**1**	**A**	**9**	**F**	**2**	**E**	**4**	**0**	**5**	**1**	**2**	**3**	**4**
2ndblk	**5**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**

Track 2 Discretionary Data – Example 2: "12345678901234567890"

Bits	1-4	5-8	9-12	13-16	17-20	21-24	25-28	29-32	33-36	37-40	41-44	45-48	49-52	53-56	57-60	61-64
1st blk	**0**	**0**	**A**	**3**	**0**	**8**	**E**	**0**	**4**	**C**	**1**	**9**	**1**	**2**	**3**	**4**
2ndblk	**5**	**6**	**7**	**8**	**9**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**F**

7 Visa Format Preserving Encryption Option

Note: VFPE is currently only supported with DUKPT. It must not be used for zone encryption.

7.1 Overview

When VFPE is applied to a transaction, it must always be applied to all occurrences of the following fields (when present):

- PAN (Track 1, Track 2, chip MSI, chip account number)
- Cardholder Name (Track 1 / chip data)
- Track 1 Discretionary Data
- Track 2 Discretionary Data (including chip MSI)

Due to the characteristics of format preserving encryption, it is not apparent from the fields themselves that they have been encrypted. Signals shall be included in the Expiration date field to indicate when VFPE has been applied, and to indicate other specific conditions necessary for decryption. These signals will be applied during VFPE processing and must be present in all occurrences of the Expiration date that is presented to the terminal application for authorization and capture processing.

When the Expiration date is not present and VFPE is applied, the field shall be constructed using the current year and month. The signals discussed below are then applied. Please note that the expiration date field shall not be constructed when absent from the card or key entry for transactions that do not have VFPE applied.

The following signals are defined for the Expiration date:

- VFPE performed
  - '40' added to the year – always set when VFPE encryption is applied
- Non-compliant check digit
  - '20' added to the month – used when the last digit of the PAN does not contain a valid check digit per ISO/IEC 7812-1.
- Missing Expiration date
  - '40' added to the month – used when the Expiration date field was missing from the card read or key entry and was created by the VFPE process.

The following parameters must be set for the VFPE algorithm when used with Visa Merchant Data Secure with Point-to-Point Encryption. The values used for A, b, k, P and L vary by the field to be encrypted.

- Alphabet ('A')

  The alphabet defines a sequential number set for all potential characters of the field that is to be encrypted. Data must be translated to the appropriate alphabet for the specific field prior to encryption.
- Number of characters in Alphabet ('n')

The value of 'n' is specific to the alphabet used. Please refer to the alphabet definitions in this document for the value to use for this parameter.

- Block cipher size, in 'b' bits

  The value of 'b' is specific to the length of the encryption key that is used. VMDS currently only utilizes Triple-DES double-length keys, which are 64 bits in length.

- Encryption key 'K'

  The transaction-specific double-length 64-bit Triple-DES DUKPT data encryption key variant shall be used.

- Counter block length, specified in 'k' digits

  The value of 'k' is specific to the alphabet used. Please refer to the alphabet definitions in this document for the value to use for this parameter.

- Plaintext 'P'

  The data characters, in the numeric form for the specified Alphabet, to be encrypted

- Plaintext length 'L'

  Number of characters represented in 'P'

- Counter 'T'

  The rightmost x bits of the transaction-specific DUKPT Key Serial Number shall be used to initialize 'T' for each field that is encrypted.

- Counter 'S'

  Used within the VFPE function. Initialized to zero for each encryption block.

7.2 Primary Account Number (PAN)

The first six and last four digits of the PAN shall remain unencrypted with their original values. The remaining middle digits shall be protected via format preserving encryption (VFPE).

VFPE shall function differently for PANs that contain valid check digits vs. those that do not. Valid check digits are those that conform to the algorithm specified in ISO/IEC 7812-1. Any other scheme used to calculate check digits are deemed as not valid check digits for this specification.

The PAN must be converted into the VFPE BASE-10 Alphabet prior to encryption.

After the VFPE algorithm has been applied, the resulting encrypted characters in Alphabet form must be converted to the original field's code set and format. The converted encryption result shall be used to replace the PAN in the original field(s) that were entered or read for presentation to the terminal application.

The original PAN must not be accessible or visible outside of the protected memory space of the HSM.

7.2.1 VFPE Processing for PANs with Valid Check Digits

- The middle digits, except for the last of these digits, shall be encrypted with the VFPE algorithm.

- The encrypted result shall be used to replace the original values for the corresponding positions in the original PAN.
- The last digit of the 'middle digits' shall then be calculated such that the original last digit of the full PAN is still a valid check digit for the newly constructed PAN.

7.2.2 VFPE Processing for PANs without Valid Check Digit

- The middle digits shall all be encrypted with the VFPE algorithm.
- The encrypted result shall be used to replace the original values for the corresponding positions in the original PAN. Please note that the original last digit of the encrypted full PAN may appear to be a valid check digit after the VFPE function has been performed.
- Turn on the signal in the Expiration date field that indicates the original PAN has a non-compliant check digit position.

7.3 Cardholder Name

The Cardholder Name field, whether in Track 1 or Chip data, shall be protected.

The entire contents of the Cardholder Name field, inclusive of embedded separators ('/', ' ' and '.'), shall be encrypted with VFPE. When protecting the Cardholder Name field in Track 1 data, all positions between the FS separators ('^') shall be presented for encryption. The FS separators are not included in the encryption operation.

The Cardholder Name must be converted into the VFPE Track 1 Alphabet prior to encryption.

After the VFPE algorithm has been applied, the resulting encrypted characters in Alphabet form must be converted to the original field's code set and format. The converted encryption result shall be used to replace the Cardholder Name that was read from Track 1 or Chip data for presentation to the terminal application for constructing payment transaction data.

7.4 Track 1 Discretionary Data

When present, the entire Discretionary Data field between the Service Code and End Sentinel elements within Track 1 shall be encrypted via VFPE.

Prior to encryption, it shall be converted into the Track 1 Alphabet.

After the VFPE algorithm has been applied, the resulting encrypted characters in Alphabet form must be converted to the original field's code set and format. The converted encryption result shall be used to replace the positions of the Discretionary Data field in the original Track 1 read from the magnetic stripe or chip data prior to presentation to the terminal application for constructing payment transaction data.

7.5 Track 2 Discretionary Data

When present, the entire Discretionary Data field between the Service Code and End Sentinel elements shall be encrypted via VFPE.

The Track 2 Discretionary Data field shall be converted to the MOD-16 Alphabet prior to performing the VFPE algorithm.

After the VFPE algorithm has been applied, the resulting encrypted characters in Alphabet form must be converted to the original field's code set and format. The converted encryption result shall be used to replace

the original Track 2 Discretionary Data field that was read from the magnetic stripe or chip data for presentation to the terminal application for constructing payment transaction data.

7.6 Visa FPE Algorithm

The Visa Format Preserving Encryption mode (VFPE) operates as a stream cipher that is format preserving.

The easiest way to think of VFPE is as Counter Mode (CTR) from NIST SP800-38A generalized to modulo-*n* addition instead of modulo-2 addition.

Let *A* be an alphabet with *n* different characters, where *n* is a natural number greater than 1. We denote by *A** the set of strings with elements from *A*, including the empty string. In this description it is assumed that the alphabet *A* is the set {0, …,*n*-1}. If this is not the case, a translation is needed, based simply on the number of different characters in the alphabet *A*. The translation would happen prior to encryption, and again, after decryption, so that encryption and decryption will always work on alphabets of the form {0, …,*n*-1} for some positive integer n, greater than 1.

VFPE uses Counter (CTR) mode as defined in [5] with a block cipher CIPH (AES or TDEA) with block size *b* bits, and encryption key *K* for CIPH, and a sequence of counter blocks (called counters in [5]) T_1, T_2, … , to produce a sequence of output blocks, one for each counter block. Each output block consists of *k* base-*n* digits, where *k* is a configurable parameter which must be chosen from the interval $\{1, \dots, \lfloor \log_n 2^b \rfloor\}$. For reasons explained below, each counter block is *b*-7 bits, rather than *b* bits as in [5]. The mechanism for how to produce the output blocks is also described below.

To encipher a plaintext *P* of length *L*, with $1 \leq L$, as many output blocks as necessary (but no more) are generated, so that the total number of base-*n* digits in the output blocks is at least *L*, that is, we calculate the unique integers *p* and *r* such that $\frac{L}{k} \leq p < \frac{L}{k} + 1$ and $0 \leq r < k$, such that $L = pk - r$, and generate output blocks $G_1, \dots, G_p$. Then each plaintext base-*n* digit *P*[*i*] is added, modulo-*n*, to the *i*'th base-*n* digit from the concatenation of the output blocks, $G_1 \| G_2 \| \dots \| G_p$, to form the *i*'th digit of the ciphertext:

$$C[i] = \left(P[i] + \left(G_1 \| \dots \| G_p\right)[i]\right) mod\ n.$$

Since *k* may not divide *L*, some digits of the last output block, G_p may be ignored. In fact, the last *r* base-*n* digits of G_r are not used.

To decipher a ciphertext *C* of length *L*, with $1 \leq L$, as many output blocks as necessary (but no more) are generated, so that the total number of base-*n* digits in the output blocks exceed *L*, which is done in the same way as for encryption. Then from each ciphertext base-*n* digit *C*[*i*] is subtracted, modulo-*n*, the *i*'th base-*n* digit from the concatenation of the output blocks, $G_1 \| \dots \| G_p$, to form the *i*'th digit of the plaintext:

$$P[i] = \left(C[i] - \left(G_1 \| \dots \| G_p\right)[i]\right) mod\ n.$$

For VFPE, as for Counter mode itself, the sequence of counter blocks must have the property that each block in the sequence is different from every other block. This condition is not restricted to a single encryption: across all of the messages that are encrypted under a given key K, all counters must be distinct. See [5] for methods for generating counters.

Given a block cipher CIPH with block length *b*, a key *K* for CIPH, a *b*-7 bit counter *T*, a natural number *n*>1, which is the base of the plaintext to be enciphered, and an integer *k* with $0 < k \leq \lfloor \log_n(2^b) \rfloor$, an output block consisting of *k* base-*n* digits is produced in the following way:

A 7-bit counter, *S*, is initialized to 0. Then CIPH_K is applied to $S \| T$ to produce a block *B* of *b* bits. B is interpreted as an integer in the interval $\{0, \dots, 2^b - 1\}$, and if $B < n^k \left\lfloor \frac{2^b}{n^k} \right\rfloor$, then it is accepted, otherwise *S* is incremented and CIPH_K is applied again to $S \| T$, *etc.,* until *B* is accepted or *S* equals 127. If S = 127, an error

is raised, otherwise *B* is converted to base-*n* and is the *k*-digit base-*n* output block, possibly with leading zeros . Under the assumption that $CIPH_K$ is a pseudorandom permutation, the probability in each iteration that *B* is accepted is at least 0.5, and the probability that an error is raised, is at most 2^{-128}. The pseudocode below describes this algorithm:

```
i = 0;
Input_Block = Si || T ;
max_B = (n^k)*((2^b) div (n^k));
B = CIPH(K, Input_Block);
while ( (AsInteger(B) ≥ max_B) AND (i < 127)) {
        i = i+1;
        Input_Block = Si || T ;
        B = CIPH(K, Input_Block);
};
if (i=127) return ERROR;
Output_Block = Convert(B, k, n);
return Output_Block;
```

Here it is assumed that S_0, S_1, ..., S_{127} enumerate the 128 different 7-bit combinations, that "AsInteger" takes a string of b bits *B*[1], ..., *B*[*b*] and converts it to the integer $\sum_{i=1}^{b}(B[i]\cdot 2^{b-i})$, and that "Convert" converts *B* to *k* base-*n* digits, with leading zeros if necessary:

```
Convert(B, k, n) {
        M = AsInteger(B);
        for (i=1; i≤k; i++){
                D[i] = M mod n;
                M = M div n;
        };
        return D;
}
```

The maximum value for *L*, that is, the bit length of the longest plaintext that can be enciphered is $2^{b/2}$.

The upper bound $n^k \left\lfloor \frac{2^b}{n^k} \right\rfloor$ for *B* interpreted as an integer is chosen as the largest possible whole multiple of n^k, that makes it possible to extract a *k*-digit base-*n* number uniformly from it, assuming the distribution of *B* is uniform.

Table 7-1: Extracting decimal digits from 8 bits

below shows as an example the values of *B* which are acceptable and those which are not, when *n* = 10 (decimal digits), *b* = 8 (numbers from 0 to 256) and *k* = 1 or 2 (extracting 1 or 2 decimal digits).

Table 7-1: Extracting decimal digits from 8 bits

***k* = 1**			***k* = 2**		
B	**Status**	***B* mod n^k**	***B***	**Status**	***B* mod n^k**
0	OK	0	0	OK	0
…	OK	…	…	OK	…
249	OK	9	199	OK	99
200	Try again	n/a	250	Try again	n/a

k = 1			*k* = 2		
B	**Status**	***B* mod n^k**	***B***	**Status**	***B* mod n^k**
…	Try again	n/a	…	Try again	n/a
255	Try again	n/a	255	Try again	n/a

7.7 Performance Analysis

To ensure that there are enough bits for the *k* base-*n* digits and a uniform distribution can be achieved, there are $n^k \left\lfloor \frac{2^b}{n^k} \right\rfloor$ different acceptable possibilities for the *b*-bit number *B*, so that we successfully can extract the *k*-digit base-*n* number from *b* bits with equal likelihood for each *k*-digit base-*n* number, which are the cases where B < $n^k \left\lfloor \frac{2^b}{n^k} \right\rfloor$. In this case the extracted number is *B* mod n^k . The likelihood of success (of extracting a *k*-digit base-*n* number from *b* bits) is $p(n,k,b) = \frac{n^k}{2^b} \cdot \left\lfloor \frac{2^b}{n^k} \right\rfloor$. If we don't have success, CIPH is called again with an updated counter, using the same key, *K*, and we try again the output from CIPH_K. This gives the average number of calls to the block cipher to extract *k* base-*n* digits as $a(n,k,b) = \frac{1}{p(n,k,b)}$, since in general $\sum_{i=0}^{\infty} p(1-p)^i(i+1) = p^{-1}$ for $0 < p \leq 1$, ignoring the possibility of error. We have probability *p* of iterating once (success the first iteration), and thus probability (1-*p*) of iterating again. In that case, we have conditional probability *p* of terminating for a total of 2 iterations, giving the total probability for this case as (1-*p*)*p*, and in general the probability of iterating *i*+1 times and terminating thereafter is $(1-p)^i p$, which explains the sum above. Thus the average yield of base-*n* digits from one call to CIPH_K is $y(n,k,b) = \frac{k}{p(n,k,b)}$.

Table 5-2 below shows the values of $p(n,k,b)$, $a(n,k,b)$, and $y(n,k,b)$ for various typical and/or illustrative values of n, k, and b. The value 95 for the base *n* is chosen as an example because the number of printable ascii characters is 95 (when space is considered a printable ascii character).

Table 7-2: Yields of base-*n* numbers from *b* bits of key material

Base, *n*	Number of bits, *b*	Number of Base-*n* digits to extract, *k*	Likelihood of success per iteration, *p*(*n*,*k*,*b*)	Average number of iterations, *a*(*n*,*k*,*b*)	Average yield of base-*n* digits per iteration, *y*(*n*,*k*,*b*)
10	4	1	0.6250	1.600	0.6250
10	13	3	0.9766	1.024	2.9297
10	64	19	0.5421	1.845	10.2999
10	64	18	0.9758	1.025	17.5641
10	128	38	0.8816	1.134	33.5016
10	128	37	0.9992	1.001	36.9693
10	128	36	0.9992	1.001	35.9701
95	64	9	0.9908	1.009	8.9173

95	64	8	0.9998	1.000	7.9984
95	128	19	0.9980	1.002	18.9629
95	128	18	0.9992	1.001	17.9859

The two cases that are likely to be most important in practice are highlighted in the table above. The table shows that the highest possible yield of decimal numbers from one round of AES on average (when extracting 37 digits per AES block) is in excess of 36.97 decimal digits, and that the similar number for TDEA is in excess of 17.56 decimal digits achieved when extracting 18 decimal digits from a TDEA block.

It can be seen from the table that the value *k* (number of base-*n* digits extracted from each block of bits output by CIPH) can sometimes beneficially be set to one less than its maximally allowable value, in order to optimize the average number of base-*n* digits generated per call to CIPH.

7.8 Limitations and other security considerations

The maximum allowable value for the base, *n*, is 2^b, where *b* is the block size of the underlying block cipher. If $n = 2^t$, for a *t* such that $1 \leq t \leq b$, and where *t* divides *b*, VFPE degenerates essentially to CTR since every possibility of the *b* bits in the bit stream are acceptable as a $\frac{b}{t}$-digit base-*t* number.

The minimum allowable value for *n* is 2, since it normally does not make practical sense to work in base-1, and in particular it does not make sense to talk about a *random* digit in base-1.

Given *n* and *b*, the maximum number of base-*n* digits that can be enciphered using only one counter value is $k = \lfloor \log_n(2^b) \rfloor$.

We refer to [5] Appendix B for generation of counter blocks, and to *ibid*. Section 6.5 for the definition of CTR. In particular the following quote highlights the requirement for unique counters across the lifetime of a key: "The sequence of counters must have the property that each block in the sequence is different from every other block. This condition is not restricted to a single message: across all of the messages that are encrypted under the given key, all of the counters must be distinct."

In order to mitigate against precomputation attacks as described in [6], it is recommended that each initial counter block incorporates information unique to the encrypting party. That way the amortization that is crucial for Hellman's time-memory tradeoff attack is prevented.

Following the analysis in [6], the plaintext length is limited by the condition that if (for a given encrypting party) the counter blocks T_i have *d* variable bits each, excluding bits used for redundancy or to identify uniquely the encrypting party, then no more than $2^{d/2}$ blocks of length *b* should be generated with one key. Since *k* base-*n* digits are encrypted with each block of *b* bits, the limit for the plaintext is then $\frac{k}{b} \cdot 2^{d/2}$ base-*n* digits. For example, if AES is used as CIPH, *b*=128. The counter blocks are 128-7 = 121 bits each. Assuming 16 bits are used to uniquely identify the encrypting party, *d*=105. When extracting *k*=37 decimal digits from one AES block, this amounts to a limit of $37 \cdot 2^{45}$ decimal digits enciphered with any one key.

For TDEA *b*=64. The counter blocks are 64-7 = 57 bits each. Assuming 16 bits are used to uniquely identify the encrypting party, *d*=41. When extracting 18 decimal digits from one TDEA block, this amounts to a limit of $9 \cdot 2^{15} = 294{,}912$ decimal digits enciphered with any one key.

These limits are more conservative than those defined in [5] for CTR.

7.9 Summary of Properties

Security Function	Encryption
Error Propagation	None, since key stream is operating on plaintext digit-by-digit
Synchronization	None built-in. Has to be done outside of mode if required. For example Derived Unique Key Per Transaction (DUKPT) can be used. (See X9.24-1).
Parallelizability	Fully parallelizable. Each block of the key stream can be generated independently, and encryption of each block can happen independently.
Keying Material Requirements	One AES-128 (or TDEA) key required
Counter/IV/Nonce Requirements	8 or 16 byte counter required for key stream generation to maintain a state, for TDEA and AES respectively
Memory Requirements	Very low: 8 or 16 byte counter (for TDEA or AES), 16 byte key, plaintext block, ciphertext block, plus executable
Pre-processing Capability	Key stream can be pre-computed
Message Length Requirements	If (for a given encrypting party) a counter block *T* has *d* variable bits, excluding bits used for redundancy or to identify uniquely the encrypting party, then no more than $\frac{k}{b} \cdot 2^{d/2}$ base-n digits should be encrypted with any one key.
Other Characteristics	Similar to CTR mode, the decryption of any ciphertext block is vulnerable to the introduction of specific bit errors into that ciphertext block if its integrity is not protected. This malleability may be mitigated by protecting the integrity of the ciphertext in other ways, e.g. with a MAC independent of the encipherment.

7.10 References

[1] M. Bellare, A. Desai, E. JokiPii, and P. Rogaway, *A concrete security treatment of symmetric encryption*, Sept 2000.
[2] M. Bellare, P. Rogaway, T. Spies, *The FFX mode of operation for format-preserving encryption*, Draft 1.1, Feb 20, 2010.
[3] J. Black and P. Rogaway, *Ciphers with arbitrary finite domains*, 2002.
[4] E. Brier, T. Peyrin, J. Stern, *BPS: a format-preserving encryption proposal*, Ingenico, France, 2010.
[5] M. Dworkin, NIST SP 800-38A: *Recommendation for block cipher modes of operation – Methods and techniques*, 2001.

[6] D. McGrew, *Counter mode security: Analysis and recommendations*, Cisco Systems, Nov 15, 2002.
[7] J. Vance, *VAES3 scheme for FFX – An addendum to "The FFX mode of operation for format-preserving encryption*", Draft 1.0, May 2011.
[8] X9.24 Part 1: 2009, *Retail financial services: Symmetric key management Part 1: Using symmetric techniques*.

7.11 Intellectual Property Statements / Agreements / Disclosures

Visa has filed a patent application (USPTO-20090063354, 12/202,978, Account transaction fraud detection) for parts of the technology described in this submission. Should the patent be granted and the technology included in the forthcoming NIST document on format preserving encryption, Visa intends to provide license to the technology on reasonable and non-discriminatory (RAND) terms.

8 Supporting Information

8.1 4-bit Hexadecimal Format

The following table illustrates the how values are encoded with this format.

Decimal Value	Hexadecimal Format	Binary Format
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

8.2 BASE-10 Alphabet

This alphabet is used when the character set only consists of numbers zero (0) through nine (9). The original data type of the source field may be of any type. The table below is representative of several commonly used formats.

VFPE alphabet number	Character	ISO 7811 Modified 5-bit ASCII	ISO 7811 Modified 7-bit ASCII	Normal data type encoding		
				4-bit Binary Coded Decimal	7-bit ASCII	8-bit EBCDIC
0	0	10000	0010000	0000	0110000	11110000
1	1	00001	1010001	0001	0110001	11110001
2	2	00010	1010010	0010	0110010	11110010
3	3	10011	0010011	0011	0110011	11110011
4	4	00100	1010100	0100	0110100	11110100
5	5	10101	0010101	0101	0110101	11110101
6	6	10110	0010110	0110	0110110	11110110
7	7	00111	1010111	0111	0110111	11110111
8	8	01000	1011000	1000	0111000	11111000
9	9	11001	0011001	1001	0111001	11111001

This alphabet requires the following values to be used in the VFPE algorithm:

- Number of characters in Alphabet (‘*n*’): 10

- Counter block length, specified in '*k*' digits: ??

8.3 BASE-16 Alphabet

Cards are encoded with the special ISO 7811 Modified 5-bit ASCII encoding for Track 2. This data type allows parity checking of the digits. Many systems require this encoding to be converted into standard data types for processing.
Other data fields may use base-16 encoding, and would use this same alphabet when performing VFPE. These data types support values of zero (0) through nine (9) and (A) through (F).
VFPE requires translation of the characters to the VFPE alphabet number prior to encryption, thus any of the data types shown in the table below are supported. Decryption may use the same or a different data type than the original encoding.

VFPE alphabet number	ISO 7811 Modified 5-bit ASCII encoding		Normal data type encoding			
	Character	Binary	Character	4-bit	7-bit ASCII	8-bit EBCDIC
0	0	10000	0	0000	0110000	11110000
1	1	00001	1	0001	0110001	11110001
2	2	00010	2	0010	0110010	11110010
3	3	10011	3	0011	0110011	11110011
4	4	00100	4	0100	0110100	11110100
5	5	10101	5	0101	0110101	11110101
6	6	10110	6	0110	0110110	11110110
7	7	00111	7	0111	0110111	11110111
8	8	01000	8	1000	0111000	11111000
9	9	11001	9	1001	0111001	11111001
10	:	11010	A	1010	1000001	11000001
11	;	01011	B	1011	1000010	11000010
12	<	11100	C	1100	1000011	11000011
13	=	01101	D	1101	1000100	11000100
14	>	01110	E	1110	1000101	11000101
15	?	11111	F	1111	1000110	11000110

This alphabet requires the following values to be used in the VFPE algorithm:

- Number of characters in Alphabet ('*n*'): 16
- Counter block length, specified in '*k*' digits: ??

8.4 Track 1 Alphabet

The Track 1 format on cards supports a limited set of characters, encoded with a special modified 7-bit ASCII format as specified in ISO 7811-2 and ISO 7813. This special data type allows parity checking of the digits. Many systems require this encoding to be converted into standard data types for processing.
VFPE requires translation of the characters to the VFPE alphabet number prior to encryption, thus any of the data types shown in the table below are supported.
Decryption may use the same or a different data type than the original encoding.
Please note that some characters are represented differently in ASCII and EBCDIC formats (e.g., '^' in ASCII is '¬' in EBCDIC).

VFPE alphabet number	Character	ISO 7811 Modified 7-bit ASCII	Standard Data Types			VFPE alphabet number	Character	ISO 7811 Modified 7-bit ASCII	Standard Data Types	
			7-bit ASCII	8-bit EBCDIC					7-bit ASCII	8-bit EBCDIC
0	space	1000000	0100000	01000000		30	@	0100000	1000000	01111100

1	!	0000001	0100001	01011010		31	A	1100001	1000001	11000001
2	“	0000010	0100010	01111111		32	B	1100010	1000010	11000010
3	#	1000011	0100011	01111011		33	C	0100011	1000011	11000011
4	$	0000100	0100100	01011011		34	D	1100100	1000100	11000100
5	%	1000101	0100101	01101100		35	E	0100101	1000101	11000101
6	&	1000110	0100110	01010000		36	F	0100110	1000110	11000110
7	‘	0000111	0100111	01111101		37	G	1100111	1000111	11000111
8	(	0001000	0101000	01001101		38	H	1101000	1001000	11001000
9	)	1001001	0101001	01011101		39	I	0101001	1001001	11001001
10	*	1001010	0101010	01011100		40	J	0101010	1001010	11010001
11	+	0001011	0101011	01001110		41	K	1101011	1001011	11010010
12	,	1001100	0101100	01101011		42	L	0101100	1001100	11010011
13	-	0001101	0101101	01100000		43	M	1101101	1001101	11010100
14	.	0001110	0101110	01001011		44	N	1101110	1001110	1101-101
15	/	1001111	0101111	01100001		45	O	0101111	1001111	11010110
16	0	0010000	0110000	11110000		46	P	1110000	1010000	11010111
17	1	1010001	0110001	11110001		47	Q	0110001	1010001	11011000
18	2	1010010	0110010	11110010		48	R	0110010	1010010	11011001
19	3	0010011	0110011	11110011		49	S	1110011	1010011	11100010
20	4	1010100	0110100	11110100		50	T	0110100	1010100	11100011
21	5	0010101	0110101	11110101		51	U	1110101	1010101	11100100
22	6	0010110	0110110	11110110		52	V	1110110	1010110	11100101
23	7	1010111	0110111	11110111		53	W	0110111	1010111	11100110
24	8	1011000	0111000	11111000		54	X	0111000	1011000	11100111
25	9	0011001	0111001	11111001		55	Y	1111001	1011001	11101000
26	:	0011010	0111010	01111010		56	Z	1111010	1011010	11101001
27	;	1011011	0111011	01011110		57	[	0111011	1011011	?
28	<	0011100	0111100	01001100		58	\	1111100	1011100	11100000
29	=	1011101	0111101	01111110		59	]	0111101	1011101	?
30	>	1011110	0111110	01101110		60	^	0111110	1011110	01011111
31	?	0011111	0111111	01101111		61	_	1111111	1011111	01101101

This alphabet requires the following values to be used in the VFPE algorithm:

- Number of characters in Alphabet (‘*n*’): 62
- Counter block length, specified in ‘*k*’ digits: ??

Appendix A Glossary

Term	Definition
Advanced Encryption Standard (AES)	Block cipher used in symmetric key cryptography adopted by NIST in November 2001 as U.S. FIPS PUB 197 (or "FIPS 197"). It is an alternative to and a stronger encryption algorithm than DES.
AES	see Advanced Encryption Standard
Acquirer Working Key (AWK)	Visa specific terminology used to describe a Zone Encryption Key used between an acquiring processor and Visa. AWKs are currently used for processing encrypted PIN data. The Visa Merchant Data Secure with P2PE service will also define another set of AWKs for use with other encrypted data.
Base Derivation Key (BDK)	A derivation key normally associated with Derived Unique Key Per Transaction
BDK	see Base Derivation Key
CA	see Certificate Authority
Capture File	A file that contains details captured at the point of service or a host for payment transactions. These files are typically presented to an acquiring processor for settlement. The acquiring processor may use these files for multiple purposes, including, but not limited to, creating clearing drafts to appropriate network (for dual-message only), reconciliation and settlement with the merchant (including merchant discount).
Card Not Present (CNP)	Environment where the cardholder does not present the card or chip representation of the card at the point of service. The cardholder may or may not be physically present at the point of service. CNP includes Electronic Commerce (e-Commerce), Mail-order, Telephone-order and other environments.
Card Present (CP)	Environment where the cardholder presents the card or chip representation of the card at the point of service. The point of service may be attended or unattended.
Certificate Authority	An entity that issues digital certificates
Composite Field Format	One of three types of field structures defined in ISO 8583:2003-1. This type of field is variable-length, and is structured using TLV formatted elements contained within one or more datasets. See also "Primitive Field Format."
Crypto	see Cryptography

Term	Definition
Cryptography	A method to protect data. Includes both encryption (which is reversible) and hashing (which is not reversible, or “one way”).
Data Encryption Standard (DES)	A block cipher that uses shared secret encryption. It is based on a symmetric-key algorithm that uses a 56-bit key.
Derivation Key	A key that is used to compute cryptographically another key. Normally a single derivation key is used in a transaction-receiving (e.g., acquirer) TRSM to derive or decrypt the Transaction Keys used by a large number of originating (e.g., terminal) TRSMs.
Derived Unique Key Per Transaction	A key management method that uses a unique key for each transaction, and prevents the disclosure of any past key used by the transaction-originating TRSM. The unique Transaction Keys are derived from a base derivation key using only non-secret data transmitted as part of each transaction.
DES	see Data Encryption Standard
Direct Exchange	Direct Exchange, a method for connecting directly to VisaNet
DSS	Acronym for “Data Security Standard” and also referred to as “PCI DSS”
See also PCI SSC.	
DUKPT	see Derived Unique Key Per Transaction
ECR	Acronym for “Electronic Cash Register.”
Federal Information Processing Standards (FIPS)	Standards that are publicly recognized by the U.S. Federal Government; also for use by non-government agencies and contractors.
FIPS	see Federal Information Processing Standards

Term	Definition
Hardware Security Module (HSM)	A special type of TRSM. It is a secure cryptoprocessor targeted at managing digital keys, accelerating cryptoprocesses and for providing strong authentication to access critical keys for server applications. These modules are physical devices that traditionally come in the form of a plug-in card or an external security device that can be attached directly to the server or general purpose computer. The goals of an HSM are (a) onboard secure generation, (b) onboard secure storage, (c) use of cryptographic and sensitive data material, (d) offloading application servers for complete asymmetric and symmetric cryptography. HSMs provide both logical and physical protection of these materials from non-authorized use and potential adversaries.
Host	Main computer hardware on which computer software is resident. Within this document, it typically identifies one or more systems that are responsible for performing merchant processing, routing decision and/or capture. These systems may resident at a merchant, gateway, processor or other entity.
HSM	see Hardware Security Module
ISO	Acronym for “International Organization for Standardization.” A non-governmental standards organization consisting of a network of the national standards institutes of over 150 countries, with one member per country and a central secretariat in Geneva, Switzerland, that coordinates the system.
ISO (Payment Industry)	Acronym for Independent Service Organization.
Issuer Discretionary Data (IDD)	Data that resides in Track 1 and Track 2 of a card’s magnetic stripe or chip magnetic stripe image (MSI). This data is located between the Service Code and the End Sentinel. It is variable in length and may contain customer and/or card verification data (e.g., PIN offset/PVV, CVV) and other data defined by card brands and/or issuers, including merchant/issuer co-brand information (e.g., merchant/loyalty info, fleet data, etc.)
Issuer Working Key (IWK)	Visa specific terminology used to describe a Zone Encryption Key used between an issuer processor and Visa. AWKs are currently used for processing encrypted PIN data.

Term	Definition
Key Serial Number (KSN)	A security element created by the terminal that performed encryption using the DUKPT method for key management. This element contains information that is required to perform decryption or translation of the data that was encrypted via this method.
Key Set Identifier (KSI)	A sub-element of the Key Serial Number that identifies the Base Derivation Key from which the current PIN or Data transaction encryption key was derived. The KSI occupies the left-most positions of the KSN and is variable in length. The length of the KSI (in bytes) is derived by subtracting 5 from the total length (in bytes) of the KSN (note that each byte represents 2 digits).
Local Master Key (LMK)	Used for encrypting locally held keys. The LMK is never distributed outside of its own environment and is never used for encrypting data. In the VisaNet environment, the LMK is housed in an HSM.
Merchant Direct Exchange (MDEX)	A program where the merchant connects directly to VisaNet for authorization processing.
NIST	Acronym for “National Institute of Standards and Technology.” A non-regulatory federal agency within U.S. Commerce Department‹s Technology Administration. Their mission is to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology to enhance economic security and improve quality of life.
PAN	Acronym for “primary account number.” A unique payment card number (typically for credit or debit cards) that identifies the issuer and the particular cardholder account.
PCI	Acronym for “Payment Card Industry.”

Term	Definition
PCI SSC	Acronym for "PCI Security Standards Council." This council offers standards and supporting materials to enhance payment card data security. These materials include a framework of specifications, tools, measurements and support resources to help organizations ensure the safe handling of cardholder information at every step. The keystone is the PCI Data Security Standard (PCI DSS), which provides an actionable framework for developing a robust payment card data security process -- including prevention, detection and appropriate reaction to security incidents. See https://www.pcisecuritystandards.org/security_standards/ for more information.
PED	Acronym for "PIN Entry Device."
PIN Block	An encrypted block of data used to encapsulate a PIN. The PIN block format defines the content of the PIN block and how it is processed to retrieve the PIN. The PIN block is composed of the PIN, the PIN length, and may contain subset of the PAN.
Plain Text	Describes data that is not encrypted. Applies to any type of non-encrypted encoding format, such as ASCII text, EBCDIC text, 4-bit hexadecimal digits, VFPE Alphabet number, etc.
POS	Acronym for "point of sale." Used interchangeably to identify the merchant environment where a purchase is made or the hardware/software at a merchant used to process payment card transactions at merchant locations.
Point of Interface	The initial point where the card data is read or otherwise obtained. The point of interface may be attended or unattended and applies to card present and card not present environments.
Primitive Field Format	One of three types of field structures defined in ISO 8583:2003-1. This type of field only contains a single element and does not have any implicit structure within it. The data type and content of the field is defined in the data dictionary within the ISO specification or proprietary technical specification, as appropriate. See also "Composite Field Format."
PTS	Acronym for "PIN Transaction Security," PTS is a set of modular evaluation requirements managed by PCI Security Standards Council, for PIN acceptance POI terminals. See www.pcisecuritystandards.org for more information.

Term	Definition
QSA	Acronym for "Qualified Security Assessor." A company approved by the PCI SSC to conduct PCI DSS on-site assessments.
RSA	Algorithm for public-key encryption described in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at Massachusetts Institute of Technology (MIT); letters RSA are the initials of their surnames.
Scoping (PCI)	Process of identifying all system components, people, and processes to be included in a PCI DSS assessment. The first step of a PCI DSS assessment is to accurately determine the scope of the review.
Sensitive Authentication Data	Security-related information (including but not limited to card validation codes/values, full magnetic-stripe data, PINs, and PIN blocks) used to authenticate cardholders and/or authorize payment card transactions.
Strong Cryptography	Cryptography based on industry-tested and accepted algorithms, along with strong key lengths and proper key-management practices. Examples of industry-tested and accepted standards and algorithms for encryption include AES (128 bits and higher), TDES (minimum double-length keys), RSA (1024 bits and higher), ECC (160 bits and higher), and ElGamal (1024 bits and higher). See NIST Special Publication 800-57 (www.csrc.nist.gov/publications/) for more information. See also Cryptography.
Symmetric Key	A cryptographic key that is used in a symmetric cryptographic algorithm (e.g., TDEA). The same symmetric key that is used for encryption is also used for decryption.
Tamper-Resistant Secuirity Module (TRSM)	A device that incorporates physical protections to prevent compromise of Cryptographic Security Parameters (CSP) therein contained. There are varying levels of protection afforded by TRSMs: Tamper-resistance: make intrusion difficult, usually by employing hardened casing. Tamper-evident: make intrusion attempts evident to subsequent viewers, often by employing seals that must be broken during intrusion Tamper-responsiveness: detect the intrusion attempt and destroy the contents in the process
TDES	Acronym for "Triple Data Encryption Standard" and also known as "3DES" or "Triple DES." See also Triple Data Encryption Algorithm
TDEA	see Triple Data Encryption Algorithm

Term	Definition
Triple Data Encryption Algorithm (TDEA)	A block cipher that applies the Data Encryption Standard (DES) cipher algorithm three times to each data block. Commonly referred to as TDES or Triple DES.
Triple DES	see Triple Data Encryption Algorithm
TRSM	see Tamper-Resistant Security Module
Zone Control Master Key (ZCMK)	A type of key that is used to encrypt other keys for transport. This type of key is also known as a Key Transport Key. ZCMKs are never used for encrypting data – they are only used for encrypting keys.
Zone Encryption Key (ZEK)	A type of key that is used to encrypt data between two specific points (e.g., between acquirer processor and Visa, between Visa and Issuer processor). Visa currently utilizes ZEKs for encrypted PIN processing. See Acquirer Working Key (AWK) and Issuer Working Key (IWK).