ID TECH
Encrypted Data Output

80000502-001 Rev J
6 July 2020

ID TECH
10721 Walker St.
Cypress, CA 90630

Table of Contents

1. SCOPE

The intent of this document is to explain encoding rules as they apply to transaction data produced by ID TECH payment peripherals that perform encryption.

Data encodings, in ID TECH products, take two major forms, depending on whether the transaction stems from a magstripe read (MSR) or a chip-card (ICC/EMV) interaction. The two major formats are described in detail here. Data from magstripe transactions will be in the Enhanced Encrypted MSR format. Data from ICC (chip card) transactions will be in a TLV-based format as described in the section on Encrypted EMV Data. Magstripe data (MSD) constructed from contactless interactions are treated as EMV data.

Once a device has been key-injected and encryption-enabled, no sensitive transaction data will ever be sent in the clear. Non-sensitive data (such as the KSN) continues to be sent in the clear. The purpose of this document is to allow you to know which segments of data are encrypted, and which segments are not encrypted.

ID TECH offers a Universal SDK that greatly facilitates data parsing. If you can do so, we strongly recommend you use the Universal SDK to obtain and manipulate data objects programmatically (in Java or C#).

1.1. Encryption Standards

The two industry-standard encryption methods supported by ID TECH products are Triple DES (TDES) and AES. (Depending on customer choice, a given product will support one or the other of these two algorithms, but not both at once.) Triple DES assumes a block size of 8 bytes; therefore, any data that will be TDES-encrypted will be zero-padded to a length that is a multiple of 8 before encryption. AES assumes a block size of 16 bytes. Data will be padded to a multiple of 16 bytes before AES encryption. For both encryption algorithms, cipher block chaining (CBC) is the default mode used in ID TECH products. The initial vector (where applicable) is all nulls.

No attempt is made here to document TDES or AES encryption methods, since they are industry standards not maintained by ID TECH.

For information on TDES, see NIST Special Publication 800-67, *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*, available at the following URL:
http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf

For information on AES, see FIPS-197, available at:
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

Some ID TECH products support the First Data TransArmor encryption methodology, which uses RSA-based public/private key technology. For information on the TransArmor methodology, see https://www.firstdata.com/downloads/marketing-merchant/TransArmor- FAQs.pdf and/or contact

First Data Corp. (https://www.firstdata.com).

1.2. Key Management

The key management methodology used in ID TECH products that support encryption is predominantly DUKPT (Derived Unique Key Per Transaction). DUKPT results in a unique 16-byte key for every transaction. The same 16-byte key may be used to encrypt or decrypt data using either TDES or AES. (In other words, the choice of key management technology has nothing to do with the choice of encryption technology.) The 10-byte Key Serial Number (KSN), unique for every transaction, is essential for deriving DUKPT keys.

A full discussion of DUKPT key management methodology is beyond the scope of this document. For details, refer to ANSI X9.24 Part 1, *Retail Financial Services Symmetric Key Management Part 1: Using Symmetric Techniques*.

1.3. Decryption

ID TECH card readers do not provide decryption capability in firmware. Decryption of transaction data is usually done on the back end (by the party that will approve and/or clear a transaction). It can also be done in a test environment. But is not typically done in an application, at transaction time, in a live production environment, because the storage or transmission of sensitive customer data in cleartext form runs counter to PCI DSS requirements (and constitutes a "worst practice," in security terms).

Still, you'll probably want to decrypt data for test/validation purposes. Decryption involves deriving the "working key" (or session key) associated with the data, and then submitting the key and data to the appropriate decryption algorithm. Deriving a one-time key using DUKPT is a somewhat intricate process. ID TECH makes available a decryption tool (at http://www.idtechproducts.com/tooling/file) that can derive keys using DUKPT, and decrypt data via TDES or AES. The tool is written in HTML and JavaScript, and uses open-source TDES and AES implementations. You may wish to look at the source code in that tool, if you want to see how DUKPT key derivation and decryption can be done.

1.4. Terminology: MSR vs. EMV

Throughout this document, the terms "magnetic stripe data," "magstripe data," and "MSR data" are considered synonymous.

ICC transactions are generally either "contact" (“insert") transactions, or "contactless” (“tap") transactions. Throughout this document, we will refer to both contact and contactless as "EMV transactions." Likewise, we will sometimes refer to "EMV data" in the context of transaction data stemming from ICC or NFC interactions.

Magstripe data (MSD) constructed from contactless interactions are treated as EMV data.
Manually entered transactions (where the card number, expiration date, etc., are typed into a keypad) are treated as MSR data.

2. GLOSSARY

AES	Advanced Encryption Standard, FIPS-197
CheckLRC	See LRC below
CheckSum	Arithmetic sum of data bytes, ignoring overflow
CVV	Card Verification Value
DES	Data Encryption Standard
DUKPT	Derived Unique Key Per Transaction
EMV	Europay, MasterCard, Visa standards
ETX	End of Text, 0x03
EXP	Expiration Date
ICC	Integrated Circuit Card (chip card)
KSN	Key Serial Number
LRC	Longitudinal redundancy check (XOR of data bytes)
MSD	Magnetic Stripe Data (may come from contactless interaction)
MSR	Magnetic Stripe Read (comes from physical read of magnetic stripe)
NFC	Near Field Communication
PAN	Primary Account Number
RFU	Reserved for Future Use
SHA	Secure Hash Algorithm
STX	Start of Text, 0x02
TDES	Triple DES (Triple Data Encryption Standard)
TLV	Tag/length/value
XOR	Exclusive-OR

3. OUTPUT FORMAT OVERVIEW

3.1. High Level Overview

The transaction data produced by ID TECH products will differ in format depending on the device, the mode of the device, and the type of transaction (e.g., magstripe versus ICC contact, versus ICC contactless). This document assumes that the device in question is operating in an encryption-enabled mode. The determining factor in how encrypted data payloads are constructed is whether data originated from a magnetic stripe read, or not.

Magstripe transactions produce data encoded according to ID TECH's Enhanced Encrypted MSR Format (see below), which is a 29-field format (with some fixed-length fields and some variable-length fields) that can't be fully described without also describing field semantics.
By contrast, chip-card transactions on EMV-capable ID TECH devices produce "EMV data" payloads that are predominantly constructed as unordered sets of TLV triplets, or so-called "tag/length/value" data. Because of the way TLV payloads are constructed, it's possible to talk about data *structure* without having to know about data *semantics*. In this document, we will focus on data *structure* whenever possible. ASN.1-BER encoding rules for tags are briefly discussed, but actual tag semantics are not. For information about EMV tag semantics, consult the EMV documentation at https://www.emvco.com.

This document purposely avoids discussion of *protocols* in order to concentrate on *formats*. Nevertheless, it should be mentioned that depending on the protocol used, data may come back "wrapped" in various kinds of wrappers. For example, ViVOPay devices will prepend encrypted data payloads with a 14-byte header or preamble consisting of the 10-byte string "ViVOtech2\0" followed by a command byte, status code, and two length bytes (MSB, LSB). Most other devices adhere to a simple protocol that encloses data in a wrapper that begins with STX (0x02) and length bytes, and ends with LRC (longitudinal redundancy check bytes: XOR of the payload), 8-bit checksum, and ETX (0x03). Please consult the appropriate ID TECH *Interface Developer's Guide* (IDG) or the appropriate product User's Manual for more information on protocol-specific data packagings.

3.2. Notational Conventions

When bytes are described in terms of bits, we use zero-based numbering of bits: B0 is the least significant bit and B7 is the most significant bit.

MB LB

B7	B6	B5	B4	B3	B2	B1	B0

Hex values are denoted in various ways: 02h, 'H'02, 0x02 (all equivalent).

4. ID TECH ENHANCED ENCRYPTED MSR DATA OUTPUT FORMAT

For ID TECH products that can read magnetic stripe data, "encrypted output" conforms to a 29-field data format as described below, known as the ID TECH Enhanced Encrypted MSR Data Output Format.

Payloads of the "Enhanced Encrypted MSR" type are constructed as shown in the following two tables. The first table is for conventional card-swipe data; the second table is for *manual- entry* data that occurs when a card number is typed into a keypad (during a Card Not Present transaction).

Take care to note that some data fields are variable-length, and some may not occur at all. For example, Fields 10 and 11 are optional. To determine whether they exist, you will need to examine bit 6 of Field 4 (as discussed further below).

4.1. MSR DATA OUTPUT FORMAT

Field #	Length in Bytes	Optional	Field Name
1	1		STX
2	2		Data Length
3	1		Card Encode Type
4	1		Track Status
5	1		Track1 data length
6	1		Track2 data length
7	1		Track3 data length
8	1		Clear/mask data sent status
9	1		Encrypted/Hash data sent status
10	1	Y	Optional bytes length
11	Variable	Y	Optional bytes
12	Variable	Y	Track1 clear/mask data
13	Variable	Y	Track2 clear/mask data
14	Variable	Y	Track3 clear/mask data
15	Variable	Y	Track1 encrypted data
16	Variable	Y	Track2 encrypted data
17	Variable	Y	Track3 encrypted data
18	8	Y	TransactionID (Session ID for Security level 4, Terminal/Merchant ID for TransArmor)
19	20	Y	Track1 hashed
20	20	Y	Track2 hashed
21	20	Y	Track3 hashed
22	10	Y	Reader Serial Number
23	Variable	Y	KSN or Key ID (10 bytes KSN for DUKPT, 10 bytes Key ID for fixed key, 11 bytes Key ID for TransArmor)
24	2	Y	MAC Value length
25	Variable	Y	MAC Value
26	10	Y	KSN for MAC DUKPT

Field #	Length in Bytes	Optional	Field Name
27	1		LRC
28	1		CheckSum
29	1		ETX (0x03)

4.2. MANUAL ENTRY DATA OUTPUT FORMAT

Field #	Length in Bytes	Optional	Field Name
1	1		STX (0x02)
2	2		Data Length
3	1		Card Encode Type (0xC0)
4	1		Track Status (0x17 or 0x37)
5	1		Track1 data length (0x00)
6	1		Length of unencrypted ;PAN= EXP [:CVV]?LRC
7	1		Length unencrypted additional data ZIP and/or ADR
8	1		Clear/mask data sent status
9	1		Encrypted/Hash data sent status
10	1	Y	Optional bytes length
11	Variable	Y	Optional bytes
12	0		Empty
13	Variable	Y	Keyed-in data presented as track 2 ;PAN=EXP[:CVV]?LRC
14	Variable	Y	Additional keyed-in data in ASCII presented as track 3 [1ADR=][0ZIP=]
15	0		Empty
16	Variable		Encrypted data
17	0		Empty
18	8	Y	TransactionID (Session ID for Security level 4, Terminal/Merchant ID for TransArmor)
19	0		Empty
20	20	Y	Hashed (present by default)
21	0		Empty
22	10	Y	Device Serial Number (not present by default)
23	Variable	Y	Key ID (10 bytes KSN for DUKPT, 10 bytes Key ID for fixed key, 11 bytes Key ID for TransArmor)
24	2	Y	MAC Value Length
25	Variable	Y	MAC Value
26	10	Y	KSN for MAC DUKPT
27	1		LRC
28	1		CheckSum
29	1		ETX (0x03)

5. Field Descriptions

5.1. Field 1: STX

Start of Text. 0x02 for most products (0x60 for Spectrum Air and SecureMOIR).

5.2. Field 2: Data Length

Two bytes, little-endian, representing the length of the data payload (which does **not** include the LRC, checksum, nor ETX, nor the leading STX, nor the length bytes themselves). In other words, the layout is:

```
STX LenL LenH Payload LRC SUM ETX
```

The length bytes specify the length of the **Payload** portion only.

5.3. Field 3: Card Encode Type

Value	Encode Type
80	ISO 7813/ISO 4909/ABA format
81	AAMVA format
83	Other
84	Raw; un-decoded format. All tracks are encrypted and no masked data are sent. No track indicator '01', '02' or '03' in front of each track.
85	JIS II Only supported in some products.
86	JIS I Only supported in some products.
87	JIS II SecureKey and Secure MIR.
91	Contactless Visa (Kernel 1)
92	Contactless MasterCard
93	Contactless Visa (Kernel 3)
94	Contactless American Express
95	Contactless JCB
96	Contactless Discover
97	Contactless UnionPay
90	Contactless Others
C0	Manual entry enhanced mode (similar to ABA track 2). Values without the high bit set are reserved.

5.4. Field 4: Track Status

MSR sampling and decode status flags:

MB							LB
0	B6	B5	B4	B3	B2	B1	B0

B0 1: Track 1 decode success (0: Track 1 decode fail)
B1 1: Track 2 decode success (0: Track 2 decode fail)

B2 1: Track 3 decode success (0: Track 3 decode fail)
B3 1: Track 1 sampling data exists (0: Track 1 sampling data does not exist)
B4 1: Track 2 sampling data exists (0: Track 2 sampling data does not exist)
B5 1: Track 3 sampling data exists (0: Track 3 sampling data does not exist)
B6 1: Field 10 “optional bytes length” exists (0: No Field 10)
B7 0 Reserved for future use

5.5. Field 5: Track1 data length Field 6: Track2 data length Field 7: Track3 data length

These one-byte values are the lengths of the actual (raw, unencrypted) Track data. It indicates the number of bytes in the Track masked data fields (Fields 12, 13, 14). It should be used to separate Track 1, Track 2 and Track 3 data after decrypting Track encrypted data field.

For ISO 7813 and ISO 4909 compliant Financial Transaction Cards: Track 1 maximum length is 79 alphanumeric characters.
Track 2 maximum length is 40 numeric digits.
Track 3 maximum length is 107 numeric digits.

5.6. Field 8: Clear/mask data sent status byte

Bit 0: 1— if Track1 clear/mask data present
Bit 1: 1— if Track2 clear/mask data present
Bit 2: 1— if Track3 clear/mask data present
Bit 3: 1— if fixed key; 0 DUKPT Key Management
Bit 4: 0 — TDES; 1 — AES
Bit 5: 1— Chip present on card. (First byte of service code was '2' or '6'.) Use EMV transaction if possible.
Bit 6: 1— PIN Encryption Key; 0—Data Encryption Key Refer to ANSI X9.24 2009 Page 56 for details.
Bit 7: 1 — Serial Number present; 0—not present

5.7. Field 9: Encrypted data sent status

Bit 0: if 1—track1 encrypted data present
Bit 1: if 1—track2 encrypted data present
Bit 2: if 1—track3 encrypted data present
Bit 3: if 1—track1 hash data (SHA digest) present
Bit 4: if 1—track2 hash data (SHA digest) present
Bit 5: if 1—track3 hash data (SHA digest) present
Bit 6: if 1—session ID present
Bit 7: if 1—KSN present

5.8. Field 10: Optional-bytes length

Number of optional bytes in Field 11. This field exists if and only if bit 6 of Field 4 is turned on.

Rationale: This field (Field 10) is present if, and only if, Bit 6 of Field 4 is turned on. The need for this scheme arises because originally, ID TECH products used a 160-bit SHA-1 digest in "hashed track data" of fields 19, 20, and 21. Later products were required to support a 32-byte (256-bit) SHA-2 digest. The purpose of the bit-6 flag in Field 4 is to signal whether the hashed track data fields use the original SHA-1 encryption (flag is zero) or the longer SHA-2 digest (flag is set). If the flag is set, Field 10 contains the length of Field 11, and Field 11 contains data specifying the type of hash. (Fields 10 and 11 provide an extensibility mechanism in case other SHA digest sizes need to be supported in the future.)

5.9. Field 11: Optional status byte 1

Bit 0: If 1—SHA-256. If 0—SHA-1 (Note: SHA-1 is the default if no Field 11.)
Bit 1: If 1—Encryption type follows Field 11 bit 2 &3 &4. If 0—Encryption type follows Field 8 bit 4.
Bit 4, 3, 2: 000—TransArmor. 001—Voltage. 010—Visa FPE. 011—Verifone FPE.
100—TransArmor TDES.
Bit 5: If 1— MAC Value Length, MAC Value, and MAC Key KSN will exist in Fields 24, 25 and 26.
If 0— No MAC Value Length, MAC Value and MAC Key KSN in Field 24, 25
and 26.
Bit 6: RFU
Bit 7: RFU

5.10. Field 12: Track1 clear/masked data Field 13: Track2 clear/masked data Field 14: Track3 clear/masked data

For MSR: Track data masked with the MaskCharID (default is '*'). The first PrePANID (up to 6 for BIN, default is 4) and last PostPANID (up to 4, default is 4) characters can be in the clear (unencrypted).

5.10.1. For Manual Input:

Field 12 is always empty.
Field 13 includes PAN, EXP (in YYMM format) and (CVV) always masked.
The format should be:

1) ;PAN=YYMM[:CVV]
?LRC ';'—start
sentinel

'='—field separator between PAN and EXP
':'—field separator between EXP and CVV if there is a CVV '?'—end sentinel
By default, the least significant digit of PAN is checked against the PAN with the MOD 10 algorithm.
LRC—calculated track 2 longitudinal redundancy check from ';' to '?'

This LRC is calculated on the raw data before conversion to ASCII as it would be encoded on a card, so that the keyed-in data can be checked identically to the card data.
The PAN is 12 to 19 digits; the EXP is 4 digits; and the CVV is 3 or 4 digits.

- For Field 14: The format of the fields ADR and ZIP is:

1 byte field identifier '1'—ADR; '0'—ZIP	ASCII Data	field terminator '='

The maximum number of ADR digits is 20. The maximum number of ZIP digits is 10.
Example: if address is 5555 and ZIP is 99999   15555=099999=

5.11. Field 15: Track1 encrypted data Field 16: Track2 encrypted data Field 17: Track3 encrypted data

These fields are the encrypted Track data, using either TDES-CBC or AES-CBC with initial vector of 0. If the original data length is not a multiple of 8 bytes for TDES or a multiple of 16 bytes for AES, the reader right pads the data with 0 before encryption.

The key management scheme is DUKPT. For DUKPT, the key used for encrypting data is called the Data Key. The Data Key is generated by taking the DUKPT Derived Key exclusive OR'd (XOR'd) with 0000000000FF00000000000000FF0000 to get the resulting intermediate variant key. The left side of the intermediate variant key is then TDES encrypted with the entire 16-byte variant as the key. After the same steps are performed for the right side of the key, combine the two 8-byte key parts to create the 16- byte Data Key.

Tracks 1, 2 and 3 data are encrypted separately. In order to get the number of bytes for each track's encrypted data field, the field length is always a multiple of 8 bytes for TDES or multiple of 16 bytes for AES, rounding up as necessary. This length value will be zero if there was no data on a track. Once the encrypted data are decrypted, all padding bytes need to be removed. The number of bytes of decoded (native) track data is indicated by the track's unencrypted length field as given in Fields 5, 6, and 7.

NOTE: For TransArmor encryption, the field length of each encrypted track is 344 bytes.

5.12. Field 18: Session ID (Security level 4 only)

At the time of this writing, no ID TECH product implements Security Level 4. Hence, Session ID is not used, but this field will contain Terminal/Merchant ID if TransArmor crypto is enabled.

5.13. Field 19: Track1 hash (if encrypted and hash track1 allowed)
5.14. Field 20: Track2 hash (if encrypted and hash track2 allowed)
5.15. Field 21: Track3 hash (if encrypted and hash track3 allowed)

The hash is used for non-SRED products; for SRED products, either all zeroes are used (20 bytes of

00), or the hash is 32 bytes of SHA-256. Refer to product manual for details. The hash may be 20 bytes (SHA-1) or 32 bytes (SHA-256) in length. To determine which kind of hash is present, see the discussion of bit 6, Field 4, and also the discussion under Field 10 & 11, above.

SHA-1 (160-bit digest) is used by default to create a 20-byte hash of the data for track 1 to track 3 raw data. The hash is exactly 20 bytes long for each track. This is provided with two purposes in mind: One is for the host to ensure data integrity by comparing this field with a SHA-1 hash of the decrypted Track data, allowing the detection of corruption in data transmission. The other purpose is to enable the host to store a tokenized version of card data for future use without keeping the sensitive cardholder data in plaintext form. The token may be used for comparison with the stored hash data to determine if they are from the same card.

SHA-256 is another option for the hash; this type of hash is 32 bytes long for each track.

5.16. Field 22: Reader Serial Number (optional)

Always 10 bytes (pad with leading 0x30 if <10 digits).

5.17. Field 23: KSN (DUKPT only) or Key ID (TransArmor).

Key ID (10 bytes KSN for DUKPT, 10 bytes Key ID for fixed key, 11 bytes Key ID for TransArmor).

5.18. Field 24: MAC Value Length

Data Length (two bytes: low byte comes first, aka "little endian"). This field will not exist unless Field 11 exists and Bit 5 is set in that field.

This value is commonly 10 00.

5.19. Field 25: MAC Value

If it exists, this field is used to verify the integrity and authority of the MSR data message; authenticated message is from Field 3 to 24. (The length of MAC Value is defined in Field 24.) *This field will not exist unless Field 11 exists and Bit 5 is set in that field.*

This field contains the HMAC result (the 16-byte digest) used to authenticate messages sent from Device to Host. The hash algorithm used here is SHA-256, but only the first 16 bytes of the result are kept.

MAC-Device = HMAC (MAC_KEY, msgX) Following this field is the MAC_DUKPT_KEY_KSN.
The MAC-Device will be the last field in a MAC-authenticated message, and msgX (the payload that is hashed) will contain everything from the first[t] byte of message being built (Response Data + MAC Value Length) up to, but not including, the MAC-Device first byte. NOTE: Advancing the KSN is controlled by the device.

The hash algorithm is known as HMAC (RFC 2104) and is given by:

$$HMAC(K', msgX) = H((K' \oplus opad) \mid H((K' \oplus ipad) \mid msgX))$$

Use HMAC-SHA256 (Refer to RFC 2104); but retain only the first 16 bytes of the calculation for MAC Authentication.

In the above formula:
H is a cryptographic hash function,
K' is the current MAC Key padded to the right with extra zeros to the input block size of the hash function, or the hash of the original key if it's longer than that block size,
m is the message to be authenticated,
| denotes concatenation,
$\oplus$ denotes XOR,
opad is the outer padding (0x5c5c5c…5c5c, one-block-long hexadecimal constant),
ipad is the inner padding (0x363636…3636, one-block-long hexadecimal constant).

5.20. Field 26: 10 bytes KSN for MAC DUKPT Key.

This field will not exist unless Field 11 exists and Bit 5 is set in that field.

5.21. Field 27: CheckLRC

XOR of all data from Card Encode Type (Field 3) to end of KSN for most ID TECH products; XOR of all data before CheckLRC for SecureMOIR and Spectrum Air.

5.22. Field 28: CheckSum

Sum of all data from Card Encode Type (Field 3) to end of KSN. Use the bottom 8 bits only. Disregard overflow.

5.23. Field 29: ETX

End of Text: 0x03.

5.23.1. Notes

Force Encryption

Force Encryption is a device setting. When Force Encrypt is set, the track will always be encrypted, regardless of card type. No clear/mask data (Field 10, 11 and 12) will be sent. When Force Encrypt is not set, only ABA bank cards (ISO 7813 and 4909 card) or Raw card data will be encrypted.

5.23.2. Handling of Purposely Reading Cards Incorrectly

In order to prevent bank card data from being transmitted if the card is not swiped firmly bottomed in the slot, a card that meets the above requirements, but has the track data shifted up one or two tracks, can also be rejected. That is, if Track 1 data appears as Track 3 data or Track 1 data appears as Track 2 data or Track 1 data appears as Track 2 data and Track 2 data appears as Track 3 data, the card may be rejected rather than being sent unencrypted. This support is only necessary and available on swipe readers.

5.23.3. Ignoring tracks

The reader can be set to ignore one or more tracks. That is, the track is not analyzed (nor sent) so that for purposes of encryption determination it can be ignored.

5.24. Samsung Pay/MST Support

Samsung Pay/MST (LoopPay) is designed to broadcast a magnetic signal to magnetic head. But because this happens contactlessly (devices separated by a centimeter or two), there is no *physical* mechanism by which to detect the origin of track data with respect to physical Track 1, physical Track 2, etc. So microcontrollers will receive magnetic signals on all tracks.

If a device receives identical MSR data on multiple tracks, it will ignore Track 2 and Track 3 data if card data is ISO 7-bit-encoded (treating such data as Track 1 data only) and ignore Track 1 and Track 3 data if card data is ISO 5-bit-encoded encoding (treating it as Track 2 data only).

ID TECH stripe readers will follow the ISO/ABA financial card checking algorithm below to decide card type, encryption, and data masking.

5.25. Card Type

Card Type 80 Cards meeting the conditions below are always encrypted following an ISO/ABA (American Banking Association) Card Encoding method.

Card Type 81 (Not encrypted unless a track is forced to be encrypted.) AAMVA (American Association of Motor Vehicle Administration) Card Encoding method.
Track1 is 7-bit encoded. Track2 is 5-bit encoded. Track3 is 7-bit.

Card Type 83 (Not encrypted unless a track is forced to be encrypted.) Card has a nonstandard format, e.g. 7-bit character data on track 2.

Card Type 84 card where the reader is in raw mode: always encrypted
Any card in raw format (that is, where the reader does not decode the track data but rather sends the track data to the host without interpretation) is never sent masked and is always encrypted, because the reader never did any track data interpretation.

Card Type 85 (Not encrypted unless a track is forced to be encrypted.) JIS II 8 8 0 (wide track, send

Track 2 only, 080).

Card Type 86 (Not encrypted unless a track is forced to be encrypted.) JIS I bits per track on Track 1 or Track 3: 858 855 850 758

Card Type 87 (Not encrypted unless a track is forced to be encrypted.) JIS II 8 8 0 (wide track, send track 2 only, 080).
It has been used for SecureKey and Secure MIR. A compatible setting is available to use Card Type 85 for JIS II.
New Products will use Card Type 85 for JIS II.

Card Type C0
Manual Key-in card data.

5.26. ISO/ABA Card

Only cards encrypted by default are Card Type 0 (bank card format cards). If the reader is so configured, the unusual card type 4 raw format may exist (where the reader is set to not decode and interpret the cards but leave them in the same format as written to the card). Only bank cards send out masked data.

Below is the algorithm used to check bank cards.

5.26.1. ISO/ABA (American Banking Association) Card Type 0 (bank cards):

The first character, the start sentinel, is a ';' on 5-bit/character tracks, and a '%' on 7-bit/character tracks. To be a valid track, the track must have a valid start sentinel, end sentinel, and longitudinal redundancy check character; and the parity on each character must be valid. Any track with 16 or fewer bits of data is invalid, the data are treated as noise.

Encoding method:

Track1 is 7-bit encoding and it was the only track decoded.
Track1 is 7-bit encoding. Track2 is 5-bit encoding and Track 3 was not decoded.
Track1 is 7-bit encoding, Track2 is 5-bit encoding, and Track3 is 5-bit encoding.
Track1 is 7-bit encoding, Track3 is 5-bit encoding, and Track 2 was not decoded.
Track2 is 5-bit encoding and neither track1 nor track3 was decoded. Track2 is 5-bit encoding, track3 is 5-bit encoding, and track 1 was not decoded.
Track3 is 5-bit encoding and neither track 1 nor 3 was decoded.

The reasons a track could be not decoded are it was not a valid 5-bit per character track, 7-bit per character track; 8-bit per character track; or the reader was told to ignore that track, or the track had insufficient bits to be a valid track.

Additional ABA Card Checks
On a track, the first field separator is used to indicate the end of the PAN (Primary Account Number).

The field separator on a 5-bit/character track is '=' and on a 7-bit/character track is caret: '^'.
Track1 second byte is 'B'.
There is a '=' in track 2 so the account number length is 12-19 digits.
There is a '^' on track 1 so the account number length is 12-19 digits (excluding spaces).
Total length of track 1 is above 21 characters.

Expiration date can be missing if there is a separator '^' or '=' replacing the first digit of the expiration date.

Track3 ISO-4909 (with PAN) checking

1. Track1 and Track2 should be in bank card format (Card Type 0, as checked above) or absent.
2. Track3 second and third characters are "01","02" or "90" – "99"
3. Track3 PAN is 12 to 19 digits. The field separator is '=' 4.Track3 total length is from 67 to 107 characters inclusive.

Note: Expiration date starts 36 characters (or optionally 34 characters) downstream of the first '='.

5.27. JIS Card Output

Below is ID TECH's standard output for JIS clear and encrypted output format.

5.27.1. USB KB or PS/2 Interface

SS, ES and LRC default for JIS track data L1, L3 mask and L3: encryption is none (0x00), i.e. not sending out SS, ES and LRC.
JIS is not recognized as ISO financial card; it will not be encrypted unless Force Encryption is on (no masked data).

5.27.2. Other Interfaces

For other interface (RS232, CDC, HID, SPI), SS, ES and LRC will be sent as is. LRC default is off on L1. LRC in L3 masked data is on. LRC in L3 encrypted data is on.

5.28. MSR DATA EXAMPLES

Data formats vary by device model. Most USB-HID and RS-232/UART card readers follow the Enhanced Encrypted MSR format as described above. Those devices output binary data (represented as hex). Some USB-KB and PS2 insert readers output a format that mixes ASCII data (for Tracks 1, 2, and 3) with binary data. See examples to follow.

All the data will be in hex format for RS-232, USB CDC, and USB-HID interface: e.g. ETX will be output as H'03'.

All the data except Track1/2/3 clear/mask data will be in hexadecimal format for keyboard interface; e.g STX will be in two hexadecimal byte '0' (H'30') and '2' (H'32'). TrackX clear/mask data is in ASCII format. e.g. '%' will be output as H'25'.

For PS2 and USB-KB interface readers, up to 15 bytes prefix and postfix can be added to the output. This is a settable feature. By default, prefix and postfix are set to none.

For PS2 and USBKB interface readers, the Data Length, CheckLrc and CheckSum calculations are based on final output bytes, excluding prefix and postfix.

5.29. Example: MSR Output from a USB-HID/RS-232/UART Interface

The data in this example are encrypted using the Enhanced Encryption MSR Format. This can be recognized because the high bit of the fourth byte underlined (80) is 1.

USB-HID / RS-232 / UART output format:

029801**80**3F48236B03BF252A343236362A2A2A2A2A2A2A393939395E425553
48204A522F47454F52474520572E4D525E2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A
2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A3F2A3B343236362A2A2A2A2A2A2A39
3939393D2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A3F2ADA7F2A52BD3F6DD8B96C50
FC39C7E6AF22F06ED1F033BE0FB23D6BD33DC5A1F808512F7AE18D47A60CC3F4
559B1B093563BE7E07459072ABF8FAAB5338C6CC8815FF87797AE3A7BEAB3B10
A3FBC230FBFB941FAC9E82649981AE79F2632156E775A06AEDAFAF6F0A184318
C5209E55AD44A9CCF6A78AC240F791B63284E15B4019102BA6C505814B585816
CA3C2D2F42A99B1B9773EF1B116E005B7CD8681860D174E6AD316A0ECDBC6871
15FC89360AEE7E430140A7B791589CCAADB6D6872B78433C3A25DA9DDAE83F12
FEFAB530CE405B701131D2FBAAD970248A4560009334 18AC88F65E1DB7ED4D10
973F99DFC8463FF6DF113B6226C4898A9D355057ECAF11A5598F02CA31688861
C157C1CE2E0F72CE0F3BB598A614EAABB1629949011900000000206E203

STX, Length(LSB, MSB), captured data type, track status, length track 1, length track 2, length track 3, Clear/mask data sent status, Encrypted/Hash data sent status
02 9801 80 3F 48-23-6B 03BF

The above broken down and interpreted:
02—STX character 98—low byte of total length
01—high byte of total length
80—captured data type byte (interpretation: new format ABA card) 3F—3 tracks of data all good
48—length of track 1 23—length of track 2 6B—length of track 3
03—tracks 1 and 2 have masked/clear data BF—bit 7=1—KSN included
Bit 6=0—no Session ID included so not level 4 encryption Bit 5=1—track 3 hash data present
Bit 4=1—track 2 hash data present Bit 3-1—track 1 hash data present
Bit 2=1—track 3 encrypted data present Bit 1=1—track 2 encrypted data present Bit 0=1—track 1 encrypted data present

Track 1 data masked (length 0x48)
252A343236362A2A2A2A2A2A2A2A393939395E42555348204A522F47454F52474520572E
4D525E2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A
2A2A3F2A

Track 1 masked data in ASCII
%*4266********9999^BUSH JR/GEORGE W.MR^*******************************?*

Track 2 data in hex masked (length 0x23)
3B343236362A2A2A2A2A2A2A2A393939393D2A2A2A2A2A2A2A2A2A2A2A2A2A2A3 F2A

Track2 masked data in ASCII
;4266********9999=**************?*

In this example there is no Track 3 data, whether clear or masked (encrypted and hashed data are shown below).

Track 1 encrypted length 0x48 rounded up to 8 bytes = 0x48 (72 decimal)
DA7F2A52BD3F6DD8B96C50FC39C7E6AF22F06ED1F033BE0FB23D6BD33DC5A1F80851
2F7AE18D47A60CC3F4559B1B093563BE7E07459072ABF8FAAB5338C6CC8815FF87797A
E3A7BE

Track 2 encrypted length 0x23 rounded up to 8 bytes =0x28 (40 decimal)
AB3B10A3FBC230FBFB941FAC9E82649981AE79F2632156E775A06AEDAFAF6F0A18431
8C5209E55AD

Track 3 encrypted length 0x6B rounded up to 8 bytes =0x70 (112 decimal)
44A9CCF6A78AC240F791B63284E15B4019102BA6C505814B585816CA3C2D2F42
A99B1B9773EF1B116E005B7CD8681860D174E6AD316A0ECDBC687115FC89360A
EE7E430140A7B791589CCAADB6D6872B78433C3A25DA9DDAE83F12FEFAB530CE405
B701131D2FBAAD970248A45600093

Track 1 data hashed length 20 bytes 3418AC88F65E1DB7ED4D10973F99DFC8463FF6DF

Track 2 data hashed length 20 bytes 113B6226C4898A9D355057ECAF11A5598F02CA31

Track 3 data hashed length 20 bytes 688861C157C1CE2E0F72CE0F3BB598A614EAABB1

KSN length 10 bytes 62994901190000000002

LRC, CheckSum and ETX 06E203

Clear/Masked Data in ASCII:

Track 1: %*4266********9999^BUSH JR/GEORGE W.MR^*******************************?*
Track 2: ;4266********9999=**************?*
Key Value: 1A 99 4C 3E 09 D9 AC EF 3E A9 BD 43 81 EF A3 34 KSN: 62 99 49 01 19 00 00 00 00 02

Decrypted Data:
Track 1 decrypted
%B4266841088889999^BUSH JR/GEORGE W.MR^0809101100001100000000046000000?!
Track 2 decrypted
;4266841088889999=080910110000046?0

Track 3 decrypted
;33333333337676760707077676763333333333767676070707767676333333333376767607070
70
7767676333333333376767607070?2

Track 1 decrypted data in hex including padding zeros (but there are no pad bytes here)
2542343236363834313038383838393939395E42555348204A522F47454F52474520572E4D5
2
5E303830393130313130303030313130303030303030303030343630303030303F21

Track 2 decrypted data in hex including padding zeros
3B34323636383431303838383839393939393D303830393130313130303030303034363F3000000
00000

Track 3 decrypted data in hex including padding zeros
3B3333333333333333333337363736373630373037303737363736373633333333333333333333
3333736373637363037303730373736373637363333333333333333333333373637363736303 7
303730373736373637363333333333333333333333373637363736303730373F320000000000

5.30. Example: MSR Output from USB KB and PS/2 Interface, Format 1

02E10280 3F4F286F03BF%*4266********9999^BUSH JR/GEORGE
W.MR^*************************************?*;4266********9999=*
******************?*38E2F7E63C3CB4114881A50CAE7A0FBCD391AEE2551
7A8D98FB6A12B58B4F494C7849E9635DC9C22204884735B2624F4CCF2B7334EA
8C746E4E32EE462836445DA36611816B73C141F1F754B2D839A04B83FD38F070
EEC9BB401ED5A4079DB7A2928B92A4D16D8C3007B60D88F9C0C5E352719FD285
69447F20FC37CC789AED41A2FEAEBE48D5A48A1B456C15FC3271A0A8D5BE324A
4878BB3E7A61B1AF45E1E3A509329ED59D8C9A647676B725264864946E226B9C
970AED70C492313BCE0A4893014EDE3A7F4D0ECA8AFF50350CD9EE257F96B1D0
00AAB259D75D807B76A04AF0897E0A292B7C44D56DBB2AA6E57EFEDD08FF7123
426037AA6B19D4955D22FB7BA325CFA81ABAFB8F7ED9387C29B2D7BD32BDC792

7845B1E819C3DCB86238706193818629949015100000C00004439F03

STX, Length(LSB, MSB), captured data type, track status, length track 1, length track 2, length track 3, Clear/mask data sent status, Encrypted/Hash data sent status
02 E102 80 3F 4F-28-6F 03BF

The above broken down and interpreted 02—STX character
E1—low byte of total length 02—high byte of total length
80—captured data type (interpretation: new format ABA card) 3F—3 tracks of data all good
4F—length of track 1 28—length of track 2 6F—length of track 3
03—tracks 1 and 2 have masked/clear data BF— Bit 7=1—KSN included
Bit 6=0—no Session ID included so not level 4 encryption Bit 5=1—track 3 hash data present
Bit 4=1—track 2 hash data present Bit 3-1—track 1 hash data present
Bit 2=1—track 3 encrypted data present Bit 1=1—track 2 encrypted data present Bit 0=1—track 1 encrypted data present

Track 1 data masked (length 0x4F)
%*4266********9999^BUSH JR/GEORGE W.MR^*************************************?* Track 2 data in hex masked (length 0x28)
;4266********9999=********************?*

In this example there is no Track 3 data whether clear or masked. (Encrypted and hashed data are shown below.)
Track 1 encrypted length 0x4F rounded up to 8 bytes = 0x50 (80 decimal)
38E2F7E63C3CB4114881A50CAE7A0FBCD391AEE25517A8D98FB6A12B58B4F494C7849
E9635DC9C22204884735B2624F4CCF2B7334EA8C746E4E32EE462836445DA36611816B7
3C141F1F754B2D839A04
Track 2 encrypted length 0x28 rounded up to 8 bytes =0x28 (40 decimal)
B83FD38F070EEC9BB401ED5A4079DB7A2928B92A4D16D8C3007B60D88F9C0C5E35271
9FD28569447
Track 3 encrypted length 0x6F rounded up to 8 bytes =0x70 (112 decimal)
F20FC37CC789AED41A2FEAEBE48D5A48A1B456C15FC3271A0A8D5BE324A4878BB3E
7A61B1AF45E1E3A509329ED59D8C9A647676B725264864946E226B9C970AED70C492313
BCE0A4893014EDE3A7F4D0ECA8AFF50350CD9EE257F96B1D000AAB259D75D807B76A
04AF0897E0A292B7C4

Track 1 data hashed length 20 bytes 4D56DBB2AA6E57EFEDD08FF7123426037AA6B19D
Track 2 data hashed length 20 bytes 4955D22FB7BA325CFA81ABAFB8F7ED9387C29B2D
Track 3 data hashed length 20 bytes 7BD32BDC7927845B1E819C3DCB86238706193818
KSN length 10 bytes 62994901510000C00004
LCR, CheckSum and ETX 439F03

Decrypted Data:

Track 1 decrypted
%B4266841088889999^BUSH JR/GEORGE
W.MR^0809101100001100000000046000000000000000?1
Track 2 decrypted
;4266841088889999=0809101100004600000?0
Track 3 decrypted
;333333333376767607070776767633333333337676760707077676763333333333767676070
70
776767633333333337676760707070707?2
Track 1 decrypted data in hex including padding zeros (but there are no pad bytes here)
25423432363638343130383838393939395E42555348204A522F47454F52474520572E4D52
5E303830393130313130303030313130303030303030303034363030303030303030303030303
03
03F3100
Track 2 decrypted data in hex including padding zeros
3B343236363834313038383838393939393D3038303931303131303030303034363030303030303
03 F30
Track 3 decrypted data in hex including padding zeros
3B333333333333333333333337363736373630373037303737363736373633333333333333333333
33
33736373637363037303730373736373637363333333333333333333333373637363736303730
373
0373736373637363333333333333333333333373637363736303730373037303730373F3200

5.31. Example: MSR Output USB HID/RS232/UART Interface, Format 2

600198803F48236B03BF252A343236362A2A2A2A2A2A2A393939395E42555348204A52
2F47454F52474520572E4D525E2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2
A2A2A2A2A2A2A2A2A2A2A3F2A3B343236362A2A2A2A2A2A2A393939393D2A2A2
A2A2A2A2A2A2A2A2A2A2A2A2A3F2A26B03F2BD327CA087C159DEA3E77974A36B6E
89CB5BC85EF92D08FB01152089099FE2A348DF2BA8D7AFEF16A1F5F2CEA46946A92CD
C2AB3B750D1AEF8127995EE6A944E12F9DF40E46607F06C68E057DA05CC3BBB2BD68E
CE1D7D89A4671423C4F649082106A785A62D9382968BCF4CFD0ECE3CF33449F265542C
B4AE6240F99CDACD08E92744FFC04C683834EB4D04C9CB9D2A4B4A4FFE15F7C70169
C89288097C4B8BB42C67D33073CFEE68B95D0F88C6CF82F86BF8E7FE5909D1537103999
40C9DAD8BD26E929EE98BEBFA9D3C19AAC047B61E8ED56BE52D4A7F8B5FFFA01341
8AC88F65E1DB7ED4D10973F99DFC8463FF6DF113B6226C4898A9D355057ECAF11A5598
F02CA31688861C157C1CE2E0F72CE0F3BB598A614EAABB1629949011A000BE00003D70 3

60, length(MSB, LSB), card type, track status, length track 1-length track 2- length track 3, mask clear

status, crypt hash status
60 0198 80 3F 48-23-6B 03BF
0198 Total message length in hexadecimal 3F Tracks 1-3 found and properly decoded
48 Length of track 1 data is 48h (72 decimal) bytes 23 Length of track 2 data is 23h (35 decimal) bytes 6B Length of track 3 data is 6Bh (107 decimal) bytes 03 indicates tracks 1 and 2 as masked BF Tracks 1-3 are encrypted, Tracks 1-3 are hashed, the KSN is included

Track one masked track data displayed in hexadecimal
252A343236362A2A2A2A2A2A2A2A393939395E42555348204A522F47454F52474520572E
4D525E2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A
2A2A3F2A
Track two masked track data displayed in hexadecimal
3B343236362A2A2A2A2A2A2A2A393939393D2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A3 F2A
Track one encrypted track data displayed in hexadecimal
26B03F2BD327CA087C159DEA3E77974A36B6E89CB5BC85EF92D08FB01152089099FE2
A348DF2BA8D7AFEF16A1F5F2CEA46946A92CDC2AB3B750D1AEF8127995EE6A944E12
F9DF40E
Track two encrypted track data displayed in hexadecimal
46607F06C68E057DA05CC3BBB2BD68ECE1D7D89A4671423C4F649082106A785A62D938
2968BCF4CF
Track three encrypted track data displayed in hexadecimal
D0ECE3CF33449F265542CB4AE6240F99CDACD08E92744FFC04C683834EB4D04C9CB9D
2A4B4A4FFE15F7C70169C89288097C4B8BB42C67D33073CFEE68B95D0F88C6CF82F86B
F8E7FE5909D153710399940C9DAD8BD26E929EE98BEBFA9D3C19AAC047B61E8ED56B
E52D4A7F8B5FFFA01
First 20-bytes of track one data hashed 3418AC88F65E1DB7ED4D10973F99DFC8463FF6DF
First 20-bytes of track two data hashed 113B6226C4898A9D355057ECAF11A5598F02CA31
First 20-bytes of track three data hashed
688861C157C1CE2E0F72CE0F3BB598A614EAABB1 KSN
629949011A000BE00003
LRC and ETX D7 03

Key Value: 14 81 3F 2E DA E0 EF C0 46 0B 08 AB FA D7 95 87
KSN: 62 99 49 01 1A 00 0B E0 00 01
Decrypted Data:
%B4266841088889999^BUSH JR/GEORGE W.MR^0809101100001100000000046000000?!
;4266841088889999=080910110000046?0
;33333333337676760707077676763333333333767676070707767676333333333376767607070
70
776767633333333337676760707?2
Clear/Masked Data displayed in ASCII:
Track 1: %*4266********9999^BUSH JR/GEORGE W.MR^******************************?* Track 2:
;4266********9999=***************?*

Key Value: 1A 99 4C 3E 09 D9 AC EF 3E A9 BD 43 81 EF A3 34 KSN: 62 99 49 01 19 00 00 00 00 02
Decrypted Data displayed in ASCII:
%B4266841088889999^BUSH JR/GEORGE W.MR^0809101100001100000000046000000?!
;4266841088889999=080910110000046?0
;333333333376767607070776767633333333337676760707077676763333333333767676070
70
7767676333333333376767607 07?2
Track 1 decrypted data in hex including padding zeros (but there are no pad bytes here)
2542343236363834313038383838393939395E42555348204A522F47454F52474520572E4D5
2
5E303830393130313130303030313130303030303030303034363030303030303F21
Track 2 decrypted data in hex including padding zeros
3B343236363834313038383838393939393D303830393130313130303030303034363F3000000
00
000
Track 3 decrypted data in hex including padding zeros
3B333333333333333333333736373637363037303730373736373637363333333333333333333
33
3373637363736303730373037373637363736333333333333333333333737363736373630373 0
373
0373736373637363333333333333333333337363

5.32. Example: Enhanced Manual Entry Output Format

Keyed in PAN   5150710200107903
Keyed in Expiration   0909
Reader Output: (SecureKey Enhanced Key-In Format, USB-KB or PS2)

029200C0170018000292;515071******7903=0909?*FBCE9EFFF7500011FA44
7DC93C11F3816BC7A37EED3CBD0464AB280F610A7035448E0888CDF683D6C5C3
2DBE6299490037000006000161DB103

Masked manually entered data: ;515071******7903=0909?*

Key Value: D1 3F 0B D8 47 AA 1D 27 C1 1C F8 4C D8 66 6A 2E KSN: 62 99 49 00 37 00 00 60 00 16

Decrypted Data:
Data in ASCII Format
;5150710200107903=0909?0

Data in HEX Format 3B35313530373130323030313037393033 3D303930393F30

5.33. Example: Enhanced Manual Entry with ADR and ZIP Output

Keyed in Admin #	5
Keyed in PAN	456789012345678901 23
Keyed in Expiration	1234
Keyed in CVV	9999
Keyed in Address	888888888888888888 88
Keyed in ZIP	7777777777

Reader Output: (SecureKey Enhanced Key-In Format, USB-HID)

028700C03700202206923B343536372A2A2A2A2A2A2A2A2A2A393031323D33
3431323A2A2A2A2A3F2A3138383838383838383838383838383838383838383D
30373737373737373737373DEABBF052D8FBB29F2814B4AFEAE0DF6882ED8CF5
F8AEB2A92B7A956FC51802E2CB35058DBE3FB4C0DD85F200A4929722E815E743
62994901010000 20000FA92703

028700C0370020220692
0087 Total message length in hexadecimal C0 Enhanced manual entry
37 Tracks 2-3 found and properly decoded
00 Length of track 1 data is 00h (0 decimal) bytes 20 Length of track 2 data is 20h (32 decimal) bytes
22 Length of track 3 data is 22h (34 decimal) bytes 06 indicates tracks 2 and 3 as masked
92 Tracks 2 is encrypted, Track 2 is hashed, the KSN is included

Masked Keyed In Data: ;4567***********9012=3412:****?* Additional Data Fields:
18888888888888888888=07777777777= Key Value: EB 8B 4A 63 7E 9E A4 BB 5C 75 E7
99 8F FC 7A 8F KSN: 62 99 49 01 01 00 00 20 00 0F
Decrypted Data in ASCII Format ;45678901234567890 12=3412:9999?4 Data in HEX Format
3B343536373839303132333435363738393031323D333431323A393939393F34

5.34. Example: MSR Output Format with TransArmor TDES-DUKPT

- If track 1 and 2 are present, provide tracks 1+2 concatenated, with all sentinels, and encrypt that whole payload together, needs to be hex.
- If only track 1 is present, provide track 1 with sentinels. Encrypted output must be in HEX.
- If only track 2 is present, provide track 2 with sentinels. Encrypted output must be in HEX.

020B01805F6A0000A1890112252A343736312A2A2A2A2A2A2A303031305E5649534120414
3515549524552205445535420434152442031305E2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2
A2A2A2A2A2A2A3F3B343736312A2A2A2A2A2A2A303031303D2A2A2A2A2A2A2A2A2A2A2A2
A2A2A2A2A2A3F7C87CAA86CD1964EEACE98D034D43EFEF837A638CB3C6B2571FC9C9A81EEE7
2693343B0DC2FC62B87BE144ACBFAE35E036BEB5C9D2963A67226B81518DF47E5BB5EBBD6E

97F7C5F4F8226B8403517B4772A017C938E8AC76A31857130B987A31819376F6C8ACDA6A47F
17DB8AB2E3A9573CC7D71C5DB14A7E29AFF01DDCA093C8B3F34B63030303030303030306
2994901AC00026000065B4F03

02 [STX]
0B01 [Total Len 267]
80 [ISO Bank Card,Enhanced Ecryption]
5F [01011111 Optional field 10,11 present]
6A [T1 length 106,for TransArmor, T1+T2 length]
00
00
A1 [10100001 clear/mask data status- T1 present, chip card, SN present]
89 [10001001 Enc/hash data status - T1 encrypted data present, T1 hash present, KSN present]
0112 [TransArmore TDES]
T1 (T1+T2) mask data: 106 bytes
252A343736312A2A2A2A2A2A2A2A303031305E56495341204143515549524552205445535420
4341524420313035E2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A3F3B3437363
12A2A2A2A2A2A2A2A303031303D2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A3F
T1 (T1+T2) encrypted: 112 bytes (14*8)
7C87CAA86CD1964EEACE98D034D43EFEF837A638CB3C6B2571FC9C9A81EEE72693343B0DC2F
C62B87BE144ACBFAE35E036BEB5C9D2963A67226B81518DF47E5BB5EBBD6E97F7C5F4F8226
B8403517B4772A017C938E8AC76A31857130B987A31819376F6C8ACDA6A47F17DB8AB2E3A9
5
T1 (T1+T2) hash: 20 bytes
73CC7D71C5DB14A7E29AFF01DDCA093C8B3F34B6
SN: 10 bytes
30303030303030303030
KSN: 10 bytes
62994901AC0002600006
5B4F LRC/checksum
03 [ETX]

Software parsing:
Track 1:
%*4761********0010^VISA ACQUIRER TEST CARD
10^**********************?;4761********0010=****************?
TransArmor TDES DUKPT Encryption
KSN: 62 99 49 01 AC 00 02 60 00 06

Decrypted Data:
%B4761739001010010^VISA ACQUIRER TEST CARD
10^1012201114380078000000?;4761739001010010=10122011143878089?
Decrypted Data in HEX:

2542343736313733393030313031303031305E5649534120414351554952455220544553542
0434152442031305E3130313232303131313433383030373830303030303030303F3B34373631
37333930303130313030313030D3130313232303131313433383737383038393F000000000000

6. ENCRYPTED EMV DATA

Transaction data from chip-card interactions (here loosely described as "EMV data") will consist primarily of TLV (tag-length-value) triplets.

Tags may be one or more bytes in length and are constructed according to standard ASN.1 Basic Encoding Rules, otherwise known as BER-TLV. (See discussion below.) Length is specified in one or more bytes using the rules described further below.

"Tag data" can consist of embedded TLV blocks that embed more TLV blocks, etc. The *ordering* of TLV blocks, at a given level, is not significant. The actual number of TLV blocks returned can vary, based on card brand, transaction type, and potentially many other considerations.

Not all TLV data will be encrypted. When TLVs are encrypted, packaging of contents will occur according to one of two schemes (Method One or Method Two), depending on whether or not you've specified the use of custom tag DFEF4B in your terminal settings. In Method One, track data and/or PAN data are encrypted in accordance with preferences specified in tag DFEF4B and the result placed in tag DFEF4D. (The data will *not* contain embedded tags; see further discussion under Method One below.) In Method Two, which is the default method if tag DFEF4B is *not* present in terminal settings, data for sensitive tags (such as tag 5A, 56, or 57, etc.) is padded, then the entire TLV is encrypted and embedded in a new TLV with the same tag name, as described under Method Two below.

6.1. Encrypted TLV Packaging: Method One

If the party that will be decrypting your data wants track data *only*, without any enclosing tags, select this method. The track data provided will be similar to the data provided in a traditional MSR transaction.

To utilize this method, you must set your preferences in tag DFEF4B and supply that tag, as a terminal configuration setting, to the ID TECH reader. (See Appendix A for more information about tags DFEF4A, DFEF4C, and DFEF4D.) Once supplied, this tag does not need to be supplied again, unless your preferences change. (This is a one-time-only setting, in other words, unless you want or need to adjust it more dynamically.)

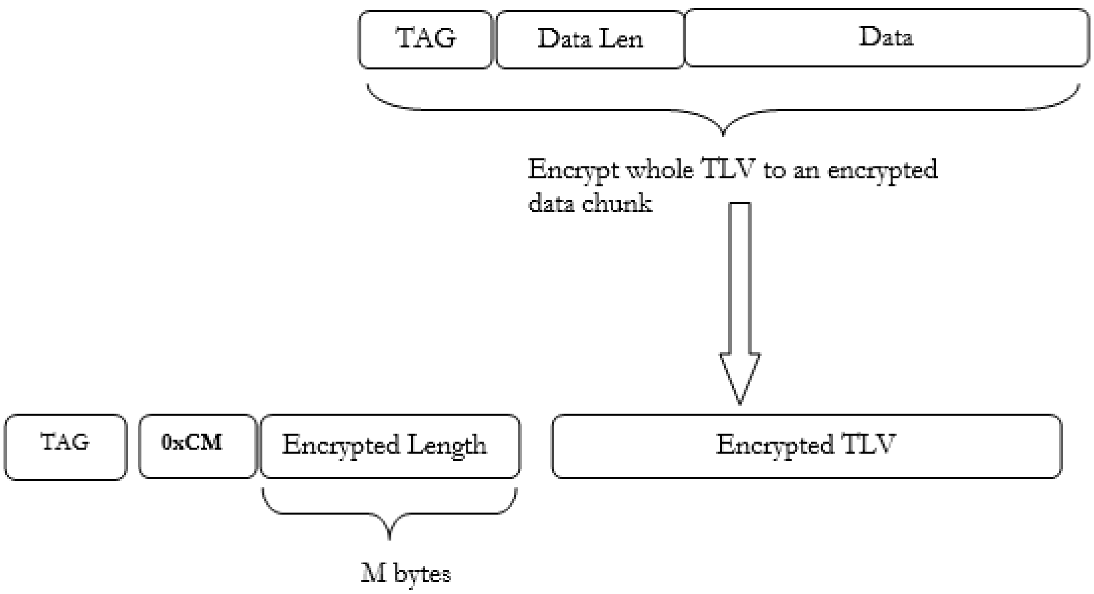
Use tag DFEF4B to specify which track or tracks (1, 2, or 3) you want to receive data for; whether or not to enable sentinels for those tracks; whether or not you wish to (also) receive PAN data; and to control whether the default behavior is to return *all* requested tracks, or just the first track found. (Again, see Appendix A for more information on these configuration options.) Once DFEF4B has been configured, the reader will place the requested data (padded and encrypted) in tag DFEF4D. If you have chosen to retrieve multiple tracks of data, the tracks will be concatenated. To know their lengths, you will need to retrieve tag DFEF4C, which will contain the explicit lengths of any returned data blocks.

Before encryption, the data payload in DFEF4D is zero-padded padded, as necessary, to a final length that's a multiple of 8 bytes (for TDES encryption) or a multiple of 16 bytes (for AES encryption).

6.2. Encrypted TLV Packaging: Method Two

Method Two is the default packaging for encrypted TLVs if tag DFEF4B has not been specified in terminal settings.

Under this packaging methodology, when a TLV has been identified as requiring encryption, the *entire* TLV, including the tag and length, is first padded (as necessary), then encrypted, before being wrapped in a new instance of the (same) tag:



After it has been padded and encrypted, the old TLV becomes the 'V' of a new instance of the tag, with a new length. The length is encoded according to special rules (as discussed below under [Length Byte Semantics]).

6.3. Tag Encoding

ID TECH transaction data will generally contain a mix of industry-standard EMV tags and proprietary ID TECH tags. ASN.1 Basic Encoding Rules apply in either case. (For information about EMV tags and their meanings, consult the EMV documentation at https://www.emvco.com.)

Tags are constructed as follows:

Byte 1: This is the first (and possibly only) value for the Tag.
If the bottom 5 bits are ON, then next byte is also part of the tag. In other words:

```
(firstByte & 0x1F == 0x1F) // TRUE means more tag bytes follow
```

Byte 2 . . *n* (if necessary):
If the most significant bit (B7) is ON, then the next byte is also part of the tag:

```
(Byte & 0x80 == 0x80) // TRUE means more tag bytes follow
```

6.3.1. Examples:
8F 02 03 04: Tag = 8F, Length = 2, Data = 03 04
9F 02 03 04: Tag = 9F02 (and Length = 3, Data = 04) BF A2 92 82: Tag = BFA292

6.4. Length Byte Semantics
The top bits of the first length byte have special significance.

If the most significant bit (B7) of the first length byte is OFF, then that entire byte *is* the data length of the data to follow. (In this case, there is one and only one "length byte" to consider.)

If the most significant bit (B7) is ON, then the lower nibble specifies the number of following bytes that encode the length of the data to follow. In other words, the lower nibble is the "length of the length." (E.g.: the lower nibble of 84 is 4, therefore the number of length bytes is 4.)

6.4.1. Examples:
6F 12 13 14 15 [...] Tag is 6F, Length is 12, Data starts at 13.
9F 20 81 82 83 84 [...]: Tag is 9F20, the lower nibble of 81 is 1 (therefore there is one length byte, with value 82), so data starts at 83.
DF 81 01 82 01 02 03 [...]: Tag is DF8101, Length is the 2 bytes after 82, which is 0x0102, so 258 bytes of data can be found starting at 03.

6.4.2. Using Length Byte to Denote Mask and/or Encryption:
Bits 5, 6, and/or 7 of the first length byte are used in a special way when data are masked or encrypted:

- Bit 7 will be set to 1.
- Bit 6 will be set to 1 if there is encryption.
- Bit 5 will be set to 1 if there is a mask (e.g., for track data).
- Bits 0-4 signify the number of "length bytes" that follow. The actual length must be retrieved from the length bytes.

6.4.3. Examples:
6F 12 13 14 15 . . .: Tag is 6F, Length is 12, Data starts at 13, no mask/encryption.
9F 20 C1 82 83 84 . . .: Tag is 9F20, Length is the 1 byte after C1, which is 0x82, data is encrypted, data starts at 83.
DF 81 01 A2 01 02 03 . . .: Tag is DF8101, Length is the 2 bytes after A2, which is 0x0102, data is masked, data starts at 03.

The following are tags that will contain encrypted and/or masked data:

6.4.4. Tags subject to encryption using Method Two

Tag	Data Object	Note	Plaintext	Mask and format	Encryption and format
5A	ApplicationPAN		None	Mask 5A A1 Len <value> This Value will be masked according to PreCtlNum and PostCtlNum, then output.	Encryption 5A C1 Len <value>
56	Track 1 Data	1. MasterCard-Paypass (MagStripe) defines it. 2. DiscoverZip defines it. 3. Visa MSD – not defined. 4. Amexnot defined. 5. PBOC– not defined.	None	Mask[1] 56 A1 Len <value> (Optional for Contact EMV)	Encryption 56 Cx Len <value>
57	Track 2 Equivalent Data		None	Mask[1] 57 A1 Len <value> (Optional for Contact EMV)	Encryption 57 Cx Len <value>
5F20	Cardholder Name		Full Plaintext	None	None
5F24	Expire Date		Full Plaintext	None	None
5F30	Service Code		Full Plaintext	None	None
9F1F	Track 1 Discretionary Data		None	None	Encryption 9F 1F Cx Len <value>
9F20	Track 2 Discretionary Data		None	None	Encryption 9F 20 Cx Len <value>
9F6B	Track 2 Data	1. MasterCard-Paypass (MagStripe) defines it 2. DiscoverZip –Do	None	Mask[1] 9F6B A1 Len <value> (Contactless MSD/EMV Only)	Encryption 9F 6B Cx Len <value>

		notDefine. 3. Visa MSD – Define it for 'Card CVM Limit'.NowDo NotEncryptitinVisa MSD. 4. Amex–Do not Define. . PBOC–Define it for 'Card CVM Limit'. Now Do Not Encrypt it in PBOC. Ifitisusedfor Track2 Data. The valueneed be encrypted, then Output.			
FFEE13	Track 1 Data	1. DiscoverZip NeedUseit. 2. Visa MSD NeedUseit. 3. Amex Need Use it. 4. PBOC Need Use it.	None	Mask[1] FF EE 13 A1 Len <value> (Contactless MSD Only)	Encryption FFEE 13Cx Len <value>
FFEE14	Track 2 Data	1. DiscoverZip NeedUseit. 2. Visa MSD NeedUseit. 3. Amex Need Use it. 4. PBOC Need Use it.	None	Mask[1] FF EE 14 A1 Len <value> (Contactless MSD Only)	Encryption FFEE 14Cx Len <value>

[1]Mask Data Note:
Data for 9F6B, FFEE13, and FFEE14 are masked for Contactless MSD only.
Values will be masked according to PreCtlNum and PostCtlNum settings in EMV, then output.

6.4.5. Discretionary Data

Tag	Data Object	Note	Plaintext	Mask and format	Encryption and format
DF812A	DD Card Track 1		None	None	Encryption DF 81 2A Cx Len <value>
DF812B	DD Card Track 2		None	None	Encryption DF 81 2B Cx Len <value>
DF31	DD Card Track 1		None	None	Encryption DF 31 Cx Len <value>

DF32	DD Card Track 2		None	None	Encryption DF 32 Cx Len <value>

Additional Encryption Information Tags (for applicable ViVOpay products)

Tag	Data Object	Note	Format
DFEE26	Encryption Status Information ("Attribution bytes")		**Byte 1:** Bit 4/3/0: Captured Data Type 0 0 0 = Contact Card 0 01=ContactlessCard/EMV 1 01=ContactlessCard/MSD 0 1 x = MSR Card Bit 2/1: Encryption Mode0 0 = TDES 1 = AES x = Refer to "Extended Encryption Mode" Bit 5: Reserved for Attribution Byte Extension. Bit 6/7: Encryption Status (For ViVOpay IDG) 0 0 = MSR/MSD off, EMV off 1 = MSR/MSD off, EMV on 0 = MSR/MSD on, EMV off 1 1 = MSR/MSD on, EMV on Byte 2: (Optional) Bit 3/2/1/0: Extended Encryption Mode 0 0 0 0 = TDES 0 0 0 1 = AES 0 0 1 0 = TransArmor Algorithm 0 0 1 1 = Voltage Algorithm 0 1 0 0 = Visa FPE 0 1 0 1 = Verifone FPE Bit 6~4: Reserved Bit 7: 0 = No MAC Verification Data 1 = Has MAC Verification Data

Note:1. DiscoverZip has 56 Tag (Track 1 Data) and Formal Track1 & 2 Data (No Tags). So DiscoverZip will have 56, FF EE 13, FF EE 14 (3 Tags) later.
2. Visa MSD, Amex, PBOC can have FF EE 13, FF EE 14 (2 Tags for Formal Track 1 & 2 Data).

6.5. TLV Encrypted Response Format Examples

6.5.1. Note on Masking

Masked tags include:
57: Optionally masked for Contact EMV 56: Optionally masked for Contact EMV 9F6B: Contactless MSD/EMV Only FFEE13: Contactless MSD Only FFEE14: Contactless MSD Only
5A

6.6. Configuration Note

1. Set PrePANClrData (N)

1 byte parameter, range is 0~6, default value 4

2. Set PostPANClrData (M)
1 byte parameter, range is 0~4, default value 4

3. Set ExpireDateOutputOpt
1 byte parameter, value is 0x30 (Mask) / 0x31 (Not Mask), default value 0x31

4. Set MaskCharID (Mask Character) for Ascii Code Track
   Data 1 byte parameter, range is 0x20~0x7E, default
   value 0x2A (*)

5. Set MaskCharID (Mask Character) for Hex Code Track Data
1 nibble parameter sent as byte value, range is 0x0A~0x0F, default value 0x0C

6. In 57 Tag Value, the data before 0xDx is PAN data, to be Masked as Tag 5A Value.

7. In 57 Tag Value, in the data 0xDy ym ms xz, yy mm is expiry date, and sxz is service code; they need not be Masked.

8. In 57 Tag Value, the data after 0xDy ym ms xz are Other data, they need be Masked.

6.6.1. Example:

ASCII Pan clear data: "012345678912" Pre-PAN clear data characters: 5
Post-PAN clear data characters: 3 Mask Character = "*"
Masked Value = "01234****912"

Hex value clear data: 0x012345678912 Pre-PAN clear data characters: 5 Post-PAN clear data characters: 3 Mask Character = 0x0C
Masked Value = 0x01234CCCC912

6.7. Tag5A Value Mask Configuration Note

1. Set PrePANClrData (N)
   1 byte parameter, range is 0~6, default value 4
2. Set PostPANClrData (M)
   1 byte parameter, range is 0~4, default value 4
3. Set MaskCharID (Mask Character) for Ascii Code Value
   1 byte parameter, range is 0x20~0x7E, default value 0x2A (*)
4. Set MaskCharID (Mask Character) for Hex Code
   Value
   1 byte parameter, range is 0x0A~0x0F, default value 0x0C

6.7.1. Example 1 – TDES / AES mode for Tag5A

1. Plaintext 5A TLV data (**5A 08 47 61 73 90 01 01 00 10**)
2. Encrypt this TLV data (**5A 08 47 61 73 90 01 01 00 10**) to be 16 bytes Encrypted Data (**XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX**):
   - For TDES mode: The Length should be multiple of 8. If it was not multiple of 8, unit should zero padded y bytes data follow automatically (the length +y should be multiple of 8).
   - For AES mode: The Length should be multiple of 16. If it was not multiple of 16, unit should zero padded y bytes data follow automatically (the length +y should be multiple of 16).
3. Re-Create New TLV data for Mask:
   - TAG is 5A
   - Length is A1 08:
     - A1 – Bit 7 is 1 note followed data length bytes. Bit 5 is 1 note data is Masked. Bit 0~4 (1) data note followed n bytes (1 byte) data length.
     - 08 – followed 8 bytes data
   - Data is **47 61 CC CC CC CC 00 10** (0x0C is Mask Data)
4. Re-Create New TLV data for Encryption:
   - TAG is 5A
   - Length is C1 10 (TDES/AES mode):
     - C1 – Bit 7 is 1 note followed data length bytes. Bit 6 is 1 note data is Encrypted. Bit 0~4 data note followed n bytes (1 byte) data length.
     - 10 – followed 16 bytes data
   - Data is XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX

6.7.2. Example 2 – TransArmor mode for Tag5A

1. Plaintext 5A TLV data (**5A 08 47 61 73 90 01 01 00 10**)
2. Change Hex Value (**47 61 73 90 01 01 00 10**) to be Ascii Value (**34 37 36 31 37 33 39 30 30 31 30 31 30 30 31 30**). We encrypt this ASCII Value data (**34 37 36 31 37 33 39 30 30 31 30 31 30 30 31 30**) to be 344 bytes Encrypted Data (**XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX**)
3. Re-Create New TLV data for Mask:
   - TAG is 5A
   - Length is A1 08:
     - A1 – Bit 7 is 1 note followed data length bytes. Bit 5 is 1 note data is Masked. Bit 0~4 (1) data note followed n bytes (1 byte) data length.
     - 08 – followed 8 bytes data
   - Data is **47 61 CC CC CC CC 00 10** (0x0C is Mask Data)
4. Re-Create New TLV data for Encryption:
   - TAG is 5A
   - Length is C2 01 58 (344- TransArmor mode):
     - C2 - Bit 7 is 1 note followed data length bytes. Bit 6 is 1 note data is Encrypted. Bit 0~4 data note followed n bytes (2 byte) data length.

- 01 58 – followed 344 bytes data

- Data is XX XX XX XX XX XX XX XX   XX XX XX XX XX XX XX XX

6.7.3. Example 3 – TDES / AES for Tag57

1. Plaintext 57 TLV data (**57 11 47 61 73 90 01 01 00 10 D1 51 22 01 17 58 98 93 89**)
2. Encrypt this TLV data (**57 11 47 61 73 90 01 01 00 10 D1 51 22 01 17 58 98 93 89**) to be 24 (TDES mode) or 32 bytes (AES mode) Encrypted Data:
   - For TDES mode: The Length should be multiple of 8. If it was not multiple of 8, unit should zero padded y bytes data follow automatically (the length +y should be multiple of 8).
   - For AES mode: The Length should be multiple of 16. If it was not multiple of 16, unit should zero padded y bytes data follow automatically (the length +y should be multiple of 16).
3. Re-Create New TLV data for Mask:
   - TAG is 57
   - Length is A1 11:
     - A1 – Bit 7 is 1 note followed data length bytes. Bit 5 is 1 note data is Masked. Bit 0~4 (1) data note followed n bytes (1 byte) data length.
     - 11 – followed 17 bytes data
   - If ExpireDataOutputOpt was set "Output Plaintext", expiry date and service code all Need Not be Masked. Data is **47 61 CC CC CC CC 00 10 D1 51 22 01 CC CC CC CC CC** (0x0C is Mask Data):
     - **47 61 73 90 01 01 00 10** is PAN, it Need be Masked same as 5A Tag Value
     - In **D1 51 22 01, 1 51 2** is expiry date (2015year, December), **2 01** is service code, they all
     - Need Not be Masked.
     - Followed them all Need be Masked.
   - If ExpireDataOutputOpt was set "Output Mask", expiry date Need be masked, service code Need Not be Masked. Data is **47 61 CC CC CC CC 00 10 DC CC C2 01 CC CC CC CC CC** (0x0C is Mask Data):
     - **47 61 73 90 01 01 00 10** is PAN, it Need be Masked same as 5A Tag Value
     - In **D1 51 22 01, 1 51 2** is expiry date (2015year, December) and Need be Masked, **2 01** is service code and it Need Not be Masked.
     - Following them, all bytes Need be Masked.
4. Re-Create New TLV data for Encryption (TDES mode):
   - TAG is 57
   - Length is C1 18:
     - C1 – Bit 7 is 1 note followed data length bytes. Bit 6 is 1 note data is Encrypted. Bit 0~4 data note followed n bytes (1 byte) data length.
     - 18 – followed 24 bytes data
   - Data is XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX

6.7.4. Example 4 - TransArmor mode for Tag57

1. Plaintext 57 TLV data (**57 11 47 61 73 90 01 01 00 10 D1 51 22 01 17 58 98 93 89**)
2. Change Hex Value (**47 61 73 90 01 01 00 10 D1 51 22 01 17 58 98 93 89**) to be Ascii Value and

Red item 'D' to be '=' (0x3D) (**34 37 36 31 37 33 39 30 30 31 30 31 30 30 31 30 3D 31 35 31 32 32 30 31 31 37 35 38 39 38 39 33 38 39**). Encrypt this Ascii Value data (**34 37 36 31 37 33 39 30 30 31 30 31 30 30 31 30 3D 31 35 31 32 32 30 31 31 37 35 38 39 38 39 33 38 39**) to be 344 bytes Encrypted Data (**XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX**).

3. Re-Create New TLV data for Mask:
   - TAG is 57
   - Length is A1 11:
     - A1 – Bit 7 is 1 note followed data length bytes. Bit 5 is 1 note data is Masked. Bit 0~4 (1) data note followed n bytes (1 byte) data length.
     - 11 – followed 17 bytes data
   - If ExpireDataOutputOpt was set "Output Plaintext", expiry date and service code all Need Not be Masked. Data is **47 61 CC CC CC CC 00 10 D1 51 22 01 CC CC CC CC CC** (0x0C is Mask Data):
     - **47 61 73 90 01 01 00 10** is PAN, it Need be Masked same as 5A Tag Value
     - In **D1 51 22 01, 1 51 2** is expiry date (2015year, December), **2 01** is service code, they all Need Not be Masked.
     - Followed them all Need be Masked.
   - If ExpireDataOutputOpt was set "Output Mask", expiry date Need be masked, service code Need Not be Masked. Data is **47 61 CC CC CC CC 00 10 DC CC C2 01 CC CC CC CC CC** (0x0C is Mask Data):
     - **47 61 73 90 01 01 00 10** is PAN, it Need be Masked same as 5A Tag Value
     - In **D1 51 22 01, 1 51 2** is expiry date (2015year, December) and Need be Masked, **2 01** is service code and it Need Not be Masked.
     - Followed them all Need be Masked.

4. Re-Create New TLV data for Encryption (TDES mode):
   - TAG is 57
   - Length is C2 01 58 (344- TransArmor mode):
     - C2 - Bit 7 is 1 note followed data length bytes. Bit 6 is 1 note data is Encrypted. Bit 0~4 data note followed n bytes (2 byte) data length.
     - 01 58 – followed 344 bytes data
   - Data is XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX, or XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
   - **XX XX XX XX XX XX XX XX XX**

Example 5 – TDES / AES mode

- If all TLVs are same level.

Raw data: **57 11 47 61 73 90 01 01 00 10 D1 51 22 01 17 58 98 93 89 5A 08 47 61 73 90 01 01 00 10 84 07 A0 00 00 00 03 10 10 9F 20 05 01 94 60 02 7F**
New data: **57 A1 11 47 61 CC CC CC CC 00 10 D1 51 22 01 CC CC CC CC CC 57 C1 18 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX 5A A1 08 47 61 CC CC CC CC 00 10 5A C1 10 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX 84 07 A0 00 00 00 03 10 10 9F 20 C1 08 XX XX XX XX XX XX XX XX**

- If all TLVs are not same level (e.g., Paypass application list Record). Raw Data:

<FF 81 06 (Tag00)> <82 01 70 (Len00)> <TLV10> <TLV11>
<FF 81 01 (Tag12)> <7F (Len12)> <TLV20> <TLV21> **57 11 47 61 73 90 01 01 00 10 D1 51 22 01 17 58 98 93 89 5A 08 47 61 73 90 01 01 00 10 84 07 A0 00 00 00 03 10 10 9F 20 05 01 94 60 02 7F** < TLV23 > ... <TLV2n>
<FF 81 01 (Tag13)> <7F (Len13)> <TLV20> <TLV21> **57 11 47 61 73 90 01 01 00 10 D1 51 22 01 17 58 98 93 89 5A 08 47 61 73 90 01 01 00 10 84 07 A0 00 00 00 03 10 10 9F 20 05 01 94 60 02 7F** < TLV23 > ... <TLV2n>
<TLV14> ... <TLV1n>
<FF 81 05 (Tag01)> <60 (Len01)> <TLV10> <TLV11> **57 11 47 61 73 90 01 01 00 10 D1 51 22 01 17 58 98 93 89 5A 08 47 61 73 90 01 01 00 10 84 07 A0 00 00 00 03 10 10 9F 20 05 01 94 60 02 7F** <TLV13> <TLV14> ...

New data:
<FF 81 06 (Tag00)> <82 01 **D5** (Len00)> <TLV10> <TLV11>
<FF 81 01 (Tag12)> <**81 B0** (Len12)> <TLV20> <TLV21> **57 A1 11 47 61 CC CC CC CC 00 10 D1 51 22 01 CC CC CC CC CC 57 C1 18 XX XX XX XX XX XX XX XX XX XX XX**
XX XX XX XX XX XX XX XX XX XX XX XX XX 5A A1 08 47 61 CC CC CC CC 00 10 5A C1 10 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX 84 07 A0 00 00 00
03 10 10 9F 20 C1 08 XX XX XX XX XX XX XX XX < TLV23 > ... <TLV2n>
<FF 81 01 (Tag13)> <81 B0 (Len13)> <TLV20> <TLV21> 57 A1 11 47 61 CC CC CC CC 00 10 D1 51 22 01 CC CC CC CC CC 57 C1 18 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX 5A A1 08 47 61 CC CC CC CC 00 10 5A C1 10 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX 84 07 A0 00 00 00
03 10 10 9F 20 C1 08 XX XX XX XX XX XX XX XX <TLV2**4**> ... <TLV2n>
<TLV14> ... <TLV1n>
<FF 81 05 (Tag01)> <91 (Len01)> <TLV10> <TLV11> 57 A1 11 47 61 CC CC CC CC 00 10 D1 51 22 01 CC CC CC CC CC 57 C1 18 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX 5A A1 08 47 61 CC CC CC CC 00 10 5A C1 10 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX 84 07 A0 00 00 00 03 10
10 9F 20 C1 08 XX XX XX XX XX XX XX XX <TLV1**4**> <TLV1**5**> ...

6.7.5. KSN in TLV format

1. **It only exists in TDES or AES mode.**
2. 3 bytes KSN Tag – DF EE 12.
3. 1 byte Len – 0A
4. 10 bytes KSN

6.7.6. KID in TLV format

KeyID (KID) exists in TransArmor mode (TransArmor - KID). Otherwise, for AES or TDES encryption, KSN will be supplied. (See above.)

3 bytes KID Tag – DF EE 12.

1 byte Len – 0B (TransArmor-KID-Length) Value - 11 bytes (TransArmor-KID-Value)

6.7.7. Contact L2 Response Format

06 + <Transaction Result > <Attribution> [<TLV>] >] [<DFEF48> <IndicatorLen> <IndicatorValue>] [<MAC Verification Data TLV > <MAC Verification KSN TLV>] Or

6.7.8. In response to Retrieve Transaction Result Command:

06 + [<TLV>] [<DFEF48> <IndicatorLen> <IndicatorValue>] [<MAC Verification Data TLV > <MAC Verification KSN TLV>]

Where:

1. Transaction Result: 2 bytes (Approve, Decline, Other)
2. Attribution: 1 Byte

Bit 4/3/0: Captured Data Type 0 0 0 = Contact Card
0 0 1 = Contactless Card / EMV
1 0 1 = Contactless Card / MSD
0 1 x = MSR Card (For ViVOpay IDG)

Bit 2/1: Encryption Mode 0 0 = TDES
0 1 = AES
1 x = Refer to Tag DFEE26 Byte 2 field "Extended Encryption Mode."

Bit 5: Attribution Byte Extension in Encryption Information Tag DFEE26. 0 = Tag DFEE26 with 1 byte, same as the "Attribution Byte."
1 = Tag DFEE26 with 2 or more bytes, extension of the "Attribution Byte."

Bit 6/7: Encryption Status (For ViVOpay IDG) 0 0 = MSR/MSD off, EMV off
0 1 = MSR/MSD off, EMV on
1 0 = MSR/MSD on, EMV off
1 1 = MSR/MSD on, EMV on

3. <TLV> is optional only if transaction was Approved or Declined

<TLV> will include KSN as first tag (DFEE12) while Encryption mode is TDES/AES. Encryption (bit 6) and Masking (bit5) flags will be utilized as appropriate in the Length component of the TLV element

4. [<DFEF48> <IndicatorLen> <IndicatorValue>]:

<DFEF48> is Indicator Tag
<IndicatorLen> is Indicator Length, variable.
<IndicatorValue> are Tags which were not output due to insufficient RAM. Note: Please refer to

Section “Buffer not enough examples for EMV L2.”

5. [<MAC Verification Data TLV > <MAC Verification KSN TLV>] is only valid for SRED & Output MAC Verification Data Option is On; please refer to Section “ MAC Verification Data Format” below.

6.7.9. Contactless L2 Response Format

06 + <Status Code > <Error Code >< Attribution > [<TLV>] [<MAC Verification Data TLV>
<MAC Verification KSN TLV>]

Where:

1. Status Code: 1 Byte. The usage is the same as in KioskII/KioskIII project and are used to specify if transaction was approved or declined.
2. Error Code: 1 Byte. The usage is the same as in KioskII/KioskIII project and are used to specify if transaction was approved or declined.
3. Attribution: 1 Byte
   - Bit 4/3/0: Captured
     Data Type 0 0 0 =
     Contact Card
     0 0 1 = Contactless Card / EMV
     1 0 1 = Contactless Card / MSD

0 1 x = MSR Card (For ViVOpay IDG)

   - Bit 2/1:
     Encryption Mode
     0 0 = TDES
     0 1 = AES
     1 x = Refer Tag DFEE26 Byte 2 field “Extended Encryption Mode”.
   - Bit 5: Attribution Byte Extension in Encryption Information Tag
     DFEE26 0 = Tag DFEE26 with 1 byte, same as the “Attribution Byte”.

1 = Tag DFEE26 with 2 or more bytes, extension of the “Attribution Byte”.

   - Bit 6/7: Encryption Status (For ViVOpay
     IDG) 0 0 = MSR/MSD off, EMV off
     0 1 = MSR/MSD off, EMV on
     1 0 = MSR/MSD on, EMV off

1 1 = MSR/MSD on, EMV on

4. <TLV> is optional only if transaction was Approved or Declined

<TLV> will include KSN as first tag (DFEE12) if encryption mode is TDES/AES. Encryption (bit 6) and Masking (bit5) flags will be utilized as appropriate in the Length component of the TLV element
6. <MAC Verification Data TLV > <MAC Verification KSN TLV> please refer to Section “MAC Verification Data” below.

6.8. MAC Verification Data / KSN TLV Format

<DFEFF41> <10> <MAC Value> <DFEFF42> <0A> <MAC Key KSN>

Where:

- <DFEFF41> is the Tag for MAC Verification Data
- <10> is length of <MAC Value>
- <MAC Value> is 16 bytes – MAC value is MAC-Device (Please refer next Section). The msgX is:
  - For Contact L2: "06 + <Transaction Result > <Attribution> [<TLV>] <DFEFF41> <10> "
  - For Contactless L2: "06 + <Status Code > <Error Code > < Attribution > [<TLV>]

<DFEFF41> <10>"

- <DFEFF42> is the Tag for MAC Verification KSN
- <0A> is length of <MAC Key KSN>
- <MAC Key KSN> is 10 bytes – MAC DUKPT Key KSN

6.8.1. MAC-Device

The HMAC result (16 bytes) is used to authenticate messages sent from Device to Host. MAC-Device = HMAC (MAC_KEY, msgX)
Following is MAC_DUKPT_KEY_KSN

The MAC-Device will be the last field in a message and msgX will include data starting from the first byte of message being built (Response Data + <DFEFF41> <10>) up to (but not including) the MAC-Device first byte.

Advancing the KSN is controlled by Device.

$$HMAC(MAC\_KEY, msgX) = H((MAC\_KEY \oplus opad) \mid H((MAC\_KEY \oplus ipad) \mid msgX))$$

Use HMAC-SHA256 (Refer to RFC2104); retain the left 16 bytes of the calculation to for MAC Authentication.

Where
H is a cryptographic hash function (in this case, SHA-256),
K is a Current MAC Key padded to the right with extra zeros to the input block size of the hash function (64 bytes, in this case), or the hash of the original key if it's longer than that block size, **msgX** is the message to be authenticated,
| denotes concatenation,
$\oplus$ denotes XOR,
opad is the outer padding (0x5c5c5c…5c5c, one-block-long hexadecimal constant),
ipad is the inner padding (0x363636…3636, one-block-long hexadecimal constant).

6.8.2. Example of HMAC

Suppose the data to be hashed with HMAC is **0123456789012345**, and the key is **0123456789ABCDEFFEDCBA9876543210**. The calculation of HMAC proceeds as follows.

$$H((K' \oplus opad) \| H((K' \oplus ipad) \| m))$$

K' is
0123456789ABCDEFFEDCBA98765432100000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000

ipad is
3636363636363636363636363636363636363636363636363636363636363636363636363
6363636363636363636363636363636363636363636363636

K' ⊕ ipad is
37157351BF9DFBD9C8EA8CAE40620426363636363636363636363636363636363636363636
3636363636363636363636363636363636363636363636363636

m is 0123456789012345

K' ⊕ ipad || m is
37157351BF9DFBD9C8EA8CAE40620426363636363636363636363636363636363636363636
36363636363636363636363636363636363636363636363636360123456789012345

H(K' ⊕ ipad || m) is c6f3b8a5fcbf7c77b44b73a87f81a02cfd6b8be138efcdf184427b8880abb691

opad is 5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C
5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C

K' ⊕ opad is
5D7F193BD5F791B3A280E6C42A086E4C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C
5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C

(K' ⊕ opad) || H((K' ⊕ ipad) || m) is
5D7F193BD5F791B3A280E6C42A086E4C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C
5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5C5Cc6f3b8a5
fcbf7c77b44b73a87f81a02cfd6b8be138efcdf184427b8880abb691

H((K' ⊕ opad) || H((K' ⊕ ipad) || m)) is
b8682749f16accceedf9c859869f6d7c305d8e8df3820ab063b61b229d4cefa8

For ID TECH products, retain only the first 16 bytes (32 hex nibbles) of the final result.

6.9. DFEF48 (Insufficient RAM) Examples

6.9.1. Example 1 – EMV L2 Transaction Result

The response body of EMV L2 Transaction Result:
06 + <Transaction Result > <Attribution> <5A C2 01 58 xx xx xx
…… xx xx xx> <9F 1F C2 01 58 xx xx xx …… xx xx xx> <TLV1> <TLV2>
… <TLVn> <DF EF 48 06 9F 20 57 56 9F 6B>

Means – There are not enough RAM resources to output 4 Tags (9F20, 57, 56 and 9F6B). Please send "Retrieve Transaction Result" command (72 46 07 01 <2 Byte Length> <Tags>) with 4 tags to retrieve them.

6.9.2. Example 2 – Retrieve Transaction Result for EMV L2

Terminal sends "Retrieve Transaction Result" command (72 46 07 01 <2 Byte Length> <Tags>) with 4 tags (9F20, 57, 56 and 9F6B).

The response body is:
06 + <9F 20 C2 01 58 xx xx xx …… xx xx xx> <57 C2 01 58 xx xx xx
…… xx xx xx> <DF EF 48 03 56 9F 6B>
Means – There are not enough RAM resources to output 2 Tags Value (56 and 9F6B). Please send "Retrieve Transaction Result" command (72 46 07 01 <2 Byte Length> <Tags>) with 2 tags to retrieve them.

6.9.3. Example 3 – Retrieve Transaction Result again for EMV L2

Terminal sends "Retrieve Transaction Result" command (72 46 07 01 <2 Byte Length> <Tags>) with with 2 tags (56 and 9F6B).

The response body is:
06 + <56 C2 01 58 xx xx xx …… xx xx xx> <9F 6B C2 01 58 xx xx xx
…… xx xx xx>
Means – There are enough RAM resources to output all values.

7. TransArmor TDES-DUKPT (Symmetric Key)

The requirement of FirstData/TransArmor 3DES-DUKPT (TDES-DUKPT) is *for Both Tracks Format.*

1. **For Track1 transactions**, the string to be encrypted consists of the raw Track1 data with Start and End Sentinels.

Example:

TK_1 = `%B4445222999990007^LAST/VISA^14125025432198712345Q?`

2. **For Track2 transactions**, the string to be encrypted consists of the raw Track2 data with Start and End Sentinels.

Example:

TK_2 = `;4445222299990007=14125025432198712345?`

Encrypted TK_2 = $(\%TK_2?)_{ENC\_TDES}$

3. **For Both Tracks transactions**, the string to be encrypted consists of the concatenated raw Track1 and Track2 with Start and End Sentinels.

Example:

```
%B4445229999990007^LAST/VISA^14125025432198712345Q?;4445222299990007=141
25025432198712345?
```

4. **For Manually Entered transactions**, the string to be encrypted consists of the concatenated dummy Track1 and Track2 data with Start and End Sentinels. The dummy Tracks are constructed from the manually-entered PAN, expiration data and CCV Data which are all required when using TDES encryption in manual entry mode.

Example:

```
%M5444009999222205^MANUALLY/ENTERED^12120000001234000000?;544400999922
2205=12120000001234000?
```

In this example, `5444009999222205` is the PAN, `1212` is the expiration data (YYMM), and `1234` is the CCV Data. There will always be six zeroes between the expiration date and the CCV Data. There will always be six zeroes after the CCV Data in Track1, and three zeroes after the CCV Data in Track2.

Note: Even though Track1 is defined in the specification, ID Tech readers never make Track1 for manually Entered transactions. It only makes Track2.

7.1. MSR

- If only Track1 data is present, provide Track1 with sentinels. Encrypted output must be in HEX.
  - Encrypted TK_1 = $(\%TK_1?)_{ENC\_TDES}$

- If only Track2 data is present, provide Track2 with sentinels. Encrypted output must be in HEX.
  - Encrypted TK_2 = (;TK_2?)$_{ENC\_TDES}$
- If Track1 and Track2 data is present, provide Tracks 1+2 concatenated in Track1, with all sentinels. Encrypted output must be in HEX.
  - Encrypted TK_1 = (%TK_1?; TK_2?)$_{ENC\_TDES}$
- If Track1, Track2 and Track3 data is present, provide Tracks 1+2 concatenated in Track1, with all sentinels. Encrypted output must be in HEX.
  - Encrypted TK_1 = (%TK_1?; TK_2?)$_{ENC\_TDES}$
  - Encrypted TK_3 = (%TK_3?)$_{ENC\_TDES}$

7.2. Contact/Contactless

- Track1 Tag includes 56 or DFEF17. Track2 Tag includes 57 or 9F6B or DFEF18.
  - **Tag DFED29 contains the data objects of the encrypted output using TransArmor(TA) TDES encryption type of-DUKPT data encryption key(DEK).The tag DFED29, Both Tracks, will only appear if both Track1 and Track2 exist.**
- If PAN Tag and/or Track2 Tag are present, convert the nibble hex code to ASCII code and change the EMV delimiter from a "D" to a "=" when using TDES encryption.
- If PAN Tag 5A is present, convert the nibble hex code to ASCII code when using TDES encryption.
  - **Tag5A = (PAN data)$_{ENC\_TDES}$**
- If Track1 Tag and Track2 Tag are present, provide Tracks 1+2 concatenated in tag DFED29, with all sentinels, and encrypt that whole payload together, needs to be HEX.
  - **Tag56 or TagDFEF17 = (%TK_1?)$_{ENC\_TDES}$**
  - **Tag57 or Tag9F6B or TagDFEF18 = (;TK_2?)$_{ENC\_TDES}$**
  - **TagDFED29 = (%TK_1?;TK_2?)$_{ENC\_TDES}$**
- If only Track1 Tag is present, provide Track1 with sentinels in Track1 Tag. Encrypted output must be in HEX.
  - **Tag56 or TagDFEF17 = (%TK_1?)$_{ENC\_TDES}$**
- If only Track2 Tag is present, provide Track2 with sentinels in Track2 Tag. Encrypted output must be in HEX.
  - **Tag57 or Tag9F6B or TagDFEF18 = (;TK_2?)$_{ENC\_TDES}$**
- This document covers the default mask character. For TA-TDES, we will follow the original data/value of the EMV rule. The values to not change/transform into the byte hex(HEX).
  - The original data/value format of the tag 5A, 57 and 9F6B are the nibble hex instead of the byte hex.

7.3. A1) MSR Output Format with TransArmor TDES-DUKPT

Ex1: Swiping ISO 4909 3 Tracks and click output format to see breaking down data. List key fields below.

```
6/9/2020 18:34:03.778 [TX] - 56 69 56 4F 74 65 63 68 32 00 02 40 00 0F
0A 9F 02 06 00 00 00 00 10 00 DF EF 37 01 01 FC 24
6/9/2020 18:34:06.760 [RX] - 56 69 56 4F 74 65 63 68 32 00 02 00 02 32
EC DF EE 25 02 00 11 DF EE 23 82 02 05 02 FF 01 80 6F 72 00 68 85 AD
01 12 25 2A 34 35 34 37 2A 2A 2A 2A 2A 2A 2A 2A 30 30 30 30 5E 4C 4C
49 42 52 45 20 52 4F 42 45 52 54 2D 47 55 49 4C 4C 45 52 4D 4F 20 5E
```

31 31 30 32 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
2A 2A 2A 2A 2A 2A 2A 2A 3F 3B 34 35 34 37 2A 2A 2A 2A 2A 2A 2A 2A 30
30 30 30 3D 31 31 30 32 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
3F 3B 2A 2A 34 35 34 37 2A 2A 2A 2A 2A 2A 2A 2A 30 30 30 30 3D 2A 2A
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 3F 55 A0 4D F8 60 00 74 50 6F F7
06 CE F1 F9 F6 2C DF 38 A8 7B FC 84 75 16 8C B0 F2 76 F5 66 A1 32 E9
AA C3 81 99 91 0A ED 80 B7 19 65 40 9A FB CF B8 66 2C 3E B6 B9 87 05
7B 1D EF C6 24 78 9F D6 AE DF 45 97 7A 25 CD 0C 4F 77 D4 BA 5D AA FE
2F B9 8B 2C A3 90 C3 B3 1E 07 37 D4 55 D6 A4 A3 E5 D6 4B 74 01 AA 0D
2A A8 3D C9 D8 D7 FF BC E9 6D B6 78 6B AC 37 75 AB 89 88 3D C1 F3 9A
40 4D 1F 2D 0E 53 6D 21 0F 67 3F A9 79 64 A8 8A 5D 38 4C DA 3D 16 B3
88 94 CE C3 D6 77 95 BC A8 49 2C 70 F4 07 97 E6 AF 16 CF 48 14 E2 47
22 65 F0 42 A8 BC F1 A2 D4 D2 0A FD 33 40 2B B7 9C D8 BB 62 64 8F A4
8B ED EB 00 D3 DE E1 76 B0 F4 56 D4 13 F4 D3 7B B9 9E 44 E0 52 93 5E
EF 5E 1B 61 2D D3 50 0C 71 DB 75 35 FC 9E 88 D0 27 58 99 3B DE 5D 6F
F2 F7 5E B1 20 63 C2 91 F5 D1 54 40 11 6F 6D 62 3B 08 C8 53 B0 EB B9
68 39 31 35 54 36 36 33 32 30 35 AA FF 98 76 54 32 10 00 00 10 0A 2E
03 9F 39 01 90 FF EE 01 05 DF EE 30 01 0C DF EE 26 02 EC 06 9F 02 06
00 00 00 00 10 00 DF 5B 01 05 ED 3A

Track1 Len: 72
Track2 Len: 00
Track3 Len: 68
Field 11 Len: 01
Field 11: 12 (bit 4/3/2: 100, TransArmor TDES)
Track1 Encrypted Data: 55 A0 4D F8 60 00 74 50 6F F7 06 CE F1 F9 F6 2C DF 38 A8 7B FC 84 75 16 8C B0 F2 76 F5 66 A1 32 E9 AA C3 81 99 91 0A ED 80 B7 19 65 40 9A FB CF B8 66 2C 3E B6 B9 87 05 7B 1D EF C6 24 78 9F D6 AE DF 45 97 7A 25 CD 0C 4F 77 D4 BA 5D AA FE 2F B9 8B 2C A3 90 C3 B3 1E 07 37 D4 55 D6 A4 A3 E5 D6 4B 74 01 AA 0D 2A A8 3D C9 D8 D7 FF BC E9 6D B6 78 6B AC 37 75 AB 89
Track3 Encrypted Data: 88 3D C1 F3 9A 40 4D 1F 2D 0E 53 6D 21 0F 67 3F A9 79 64 A8 8A 5D 38 4C DA 3D 16 B3 88 94 CE C3 D6 77 95 BC A8 49 2C 70 F4 07 97 E6 AF 16 CF 48 14 E2 47 22 65 F0 42 A8 BC F1 A2 D4 D2 0A FD 33 40 2B B7 9C D8 BB 62 64 8F A4 8B ED EB 00 D3 DE E1 76 B0 F4 56 D4 13 F4 D3 7B B9 9E 44 E0 52 93 5E EF 5E 1B 61 2D D3 50
KSN: AA FF 98 76 54 32 10 00 00 10

After decrypting Track1 and Track3 Data in Parsomatic:

Track1: %B4547570001070000^LLIBRE ROBERT-GUILLERMO ^1102101000000040000000306000000?;4547570001070000=1102101000003060000?

Track3:
;014547570001070000=797800000000000000003019018040200011024=30250001141401058598==1=000000260000000000000?

Ex2: Swiping ISO 4909 Track2+Track3 and click output format to see breaking down data. List key fields below

```
6/9/2020 18:44:11.352 [TX] - 56 69 56 4F 74 65 63 68 32 00 02 40 00 0F
0A 9F 02 06 00 00 00 00 10 00 DF EF 37 01 01 FC 24
6/9/2020 18:44:15.327 [RX] - 56 69 56 4F 74 65 63 68 32 00 02 00 01 96
EC DF EE 25 02 00 11 DF EE 23 82 01 69 02 63 01 80 77 00 26 68 86 B6
01 12 3B 34 35 34 37 2A 2A 2A 2A 2A 2A 2A 2A 30 30 30 30 3D 31 31 30
32 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 3F 3B 2A 2A 34 35 34
37 2A 2A 2A 2A 2A 2A 2A 2A 30 30 30 30 3D 2A 2A 2A 2A 2A 2A 2A 2A 2A
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
2A 2A 2A 2A 2A 3F 90 14 DA AA B6 93 7E 74 21 CB F6 7E 3A 4D D5 E6 A2
3A 31 65 8F B6 56 B5 92 E1 7F 80 20 75 61 55 8C 4F 32 9F D5 E9 87 31
11 D2 32 9E F8 6D FA 5A D5 4A 42 AA C7 E7 79 6C 12 19 57 88 A0 1B C4
F5 9F 1E 1D 2D 3C D8 8D DA 17 4A 48 04 BC CD 80 DD BC C4 FC D2 6D 1A
4E DC 2D B6 46 E7 7F 1C 99 65 4E D7 A6 EC E3 71 9E 89 F0 38 8C 5C 44
B9 32 1D 0B 34 87 91 97 FB 2B E0 5C 3E 8E 65 54 CD 58 E9 0D F3 0B 71
06 6B B2 2F 5A D1 B2 9C 13 E2 FB 5F A4 AD 54 8C 7D 26 6C 91 89 43 80
C3 B4 96 23 7F 59 F0 A9 90 20 63 C2 91 F5 D1 54 40 11 6F 6D 62 3B 08
C8 53 B0 EB B9 68 39 31 35 54 36 36 33 32 30 35 AA FF 98 76 54 32 10
00 00 11 F0 96 03 9F 39 01 90 FF EE 01 05 DF EE 30 01 0C DF EE 26 02
EC 06 9F 02 06 00 00 00 00 10 00 DF 5B 01 05 87 6B
```

Track1 Len: 00

Track2 Len: 26

Track3 Len: 68

Field 11 Len: 01

Field 11: 12 (bit 4/3/2: 100, TransArmor TDES)

Track2 Encrypted Data: 90 14 DA AA B6 93 7E 74 21 CB F6 7E 3A 4D D5 E6 A2 3A 31 65 8F B6 56 B5 92 E1 7F 80 20 75 61 55 8C 4F 32 9F D5 E9 87 31

Track3 Encrypted Data: 11 D2 32 9E F8 6D FA 5A D5 4A 42 AA C7 E7 79 6C 12 19 57 88 A0 1B C4 F5 9F 1E 1D 2D 3C D8 8D DA 17 4A 48 04 BC CD 80 DD BC C4 FC D2 6D 1A 4E DC 2D B6 46 E7 7F 1C 99 65 4E D7 A6 EC E3 71 9E 89 F0 38 8C 5C 44 B9 32 1D 0B 34 87 91 97 FB 2B E0 5C 3E 8E 65 54 CD 58 E9 0D F3 0B 71 06 6B B2 2F 5A D1 B2 9C 13 E2 FB 5F

KSN: AA FF 98 76 54 32 10 00 00 11

After decrypting Track2 and Track3 Data in Parsomatic,

Track2: ;4547570001070000=1102101000003060000?

Track3:
;014547570001070000=797800000000000000301901804020001102 4=30250001141401058598==1=000000260000000000000?

Ex3: Swiping ISO 4909 Track1+Track3 and click

Track1 Len: 4C
Track2 Len: 00
Track3 Len: 68
Field 11 Len: 01
Field 11: 12 (bit 4/3/2: 100, TransArmor TDES)
Track1 Encrypted Data: 75 10 22 4C 75 4E A7 47 9A FB 39 3E AF 54 7A BE B1 E4 55 F0 6F B5 B3 1E 9F C0 51 3E 48 F8 48 09 5C B2 41 C3 E4 C4 A6 EA 9B E6 15 41 A5 49 37 C4 2D 47 02 64 26 95 EF 51 8D 35 4B 3E 71 B7 6F 3C AE DF 06 22 D1 D2 58 DD E7 22 0B F7 98 67 A1 6C
Track3 Encrypted Data: 9D CB 71 C4 19 3F F4 A4 3D 93 C0 E7 8E 81 59 88 8D 12 BB 5F 97 7F 08 8A B7 7F 6A 82 8C 47 72 36 9C F1 3A 2C C2 D8 82 CB 03 AF 20 05 84 3A 2D 3C B4 36 CC 66 7D 1F 60 D4 F2 C7 92 F2 9C 83 A5 19 6C E9 46 26 FA 95 8C D4 D2 53 43 4B F8 AD F6 16 9A 13 21 91 06 CC F7 CE BD D7 3D 0F 62 BB C2 7B 5A 81 77 A1 C3 C6 78 33
KSN: AA FF 98 76 54 32 10 00 00 12

After decrypting Track2 and Track3 Data in Parsomatic:
Track1: %B4547570001070000^LLIBRE ROBERT-GUILLERMO ^1102101000000040000000306000000?
Track3:
;014547570001070000=797800000000000000301901804020001102 4=30250001141401058598==1=000000260000000000000?

7.4. A2) Contact Output Format with TransArmor TDES-DUKPT

Ex1: Present "EMV Test Card V2 T=0" contact test card using 60-10/11, 60-12 with DFEE1A and view breaking down data below.

```
6/9/2020 17:25:06.850 [TX] - 56 69 56 4F 74 65 63 68 32 00 60 10 00 1A
01 00 C8 00 C8 9C 01 00 9F 02 06 00 00 00 00 01 00 9F 03 06 00 00 00
00 00 00 27 E3
6/9/2020 17:25:06.933 [RX] - 56 69 56 4F 74 65 63 68 32 00 60 63 00 00
FF 0E 56 69 56 4F 74 65 63 68 32 00 60 00 00 E5 E4 DF EE 12 0A AA FF
98 76 54 32 10 00 00 0A DF EE 25 02 00 10 57 A1 11 47 61 CC CC CC CC
00 10 D2 01 2C CC CC CC CC CC CC 57 C1 28 73 BE A4 D5 0C D9 92 B5 2C
1E 36 0C 52 AD F4 06 5E C1 FF 68 5D 36 FF 15 75 83 B8 0B 39 8A 19 7B
4A EB 3B 0F 5F 63 EB F0 5A A1 08 47 61 CC CC CC CC 00 10 5A C1 10 D1
13 BD E1 67 E4 30 D6 2A C3 22 55 BD D0 33 0F 5F 34 01 01 5F 20 0F 46
55 4C 4C 20 46 55 4E 43 54 49 4F 4E 41 4C 5F 24 03 20 12 31 9F 1E 08
54 65 72 6D 69 6E 61 6C 9F 20 00 5F 25 03 95 07 01 5F 2D 08 65 73 65
6E 66 72 64 65 50 0A 56 49 53 41 43 52 45 44 49 54 4F 07 A0 00 00 00
03 10 10 84 07 A0 00 00 00 03 10 10 DF EE 23 00 9F 39 01 05 9F 53 00
DF EE 26 02 E4 06 FF EE 01 05 DF EE 30 01 01 4D 04
6/9/2020 17:25:08.981 [TX] - 56 69 56 4F 74 65 63 68 32 00 60 11 00 03
00 00 05 C9 E9
6/9/2020 17:25:09.042 [RX] - 56 69 56 4F 74 65 63 68 32 00 60 63 00 00
FF 0E 56 69 56 4F 74 65 63 68 32 00 60 00 01 D8 E4 DF EE 12 0A AA FF
98 76 54 32 10 00 00 0A DF EE 25 02 00 04 4F 07 A0 00 00 00 03 10 10
5A A1 08 47 61 CC CC CC CC 00 10 5A C1 10 D1 13 BD E1 67 E4 30 D6 2A
C3 22 55 BD D0 33 0F 50 0A 56 49 53 41 43 52 45 44 49 54 57 A1 11 47
```

```
61 CC CC CC CC 00 10 D2 01 2C CC CC CC CC CC CC 57 C1 28 73 BE A4 D5
0C D9 92 B5 2C 1E 36 0C 52 AD F4 06 5E C1 FF 68 5D 36 FF 15 75 83 B8
0B 39 8A 19 7B 4A EB 3B 0F 5F 63 EB F0 82 02 5C 00 84 07 A0 00 00 00
03 10 10 8E 0C 00 00 00 00 00 00 00 00 5F 03 00 00 95 05 42 80 00 00
00 9A 03 20 06 09 9B 02 C8 00 9C 01 00 99 00 5F 20 0F 46 55 4C 4C 20
46 55 4E 43 54 49 4F 4E 41 4C 5F 24 03 20 12 31 5F 25 03 95 07 01 5F
28 02 08 40 5F 2A 02 08 40 5F 2D 08 65 73 65 6E 66 72 64 65 5F 34 01
01 9F 02 06 00 00 00 00 01 00 9F 03 06 00 00 00 00 00 00 9F 07 02 FF
C0 9F 08 02 00 8C 9F 09 02 00 96 9F 0B 00 9F 0D 05 00 00 00 00 00 9F
0E 05 00 00 00 00 00 9F 0F 05 00 00 00 00 00 9F 10 07 06 01 1A 03 90
00 00 9F 11 01 01 9F 12 0D 43 52 45 44 49 54 4F 44 45 56 49 53 41 9F
13 00 9F 15 02 12 34 9F 16 0F 30 30 30 30 30 30 30 30 30 30 30 30 30
30 30 9F 1A 02 08 40 9F 1C 08 38 37 36 35 34 33 32 31 9F 1E 08 54 65
72 6D 69 6E 61 6C 9F 20 00 9F 21 03 09 24 45 9F 24 00 9F 26 08 74 CF
82 72 55 F8 94 FC 9F 27 01 80 9F 33 03 60 08 C8 9F 34 03 5F 03 02 9F
35 01 25 9F 36 02 00 01 9F 37 04 3A EA FA 85 9F 53 00 DF 21 00 DF EE
23 00 9F 39 01 05 DF EE 51 00 9F 5B 00 DF EE 26 02 E4 06 FF EE 01 05
DF EE 30 01 01 70 E9
6/9/2020 17:25:10.706 [TX] - 56 69 56 4F 74 65 63 68 32 00 60 12 00 17
01 8A 02 30 30 91 0A 11 11 33 44 55 66 77 88 30 30 DF EE 1A 02 56 57
26 44
6/9/2020 17:25:10.738 [RX] - 56 69 56 4F 74 65 63 68 32 00 60 63 00 00
FF 0E 56 69 56 4F 74 65 63 68 32 00 60 00 00 65 E4 DF EE 12 0A AA FF
98 76 54 32 10 00 00 0A DF EE 25 02 00 02 56 00 57 A1 11 47 61 CC CC
CC CC 00 10 D2 01 2C CC CC CC CC CC CC 57 C1 28 73 BE A4 D5 0C D9 92
B5 2C 1E 36 0C 52 AD F4 06 5E C1 FF 68 5D 36 FF 15 75 83 B8 0B 39 8A
19 7B 4A EB 3B 0F 5F 63 EB F0 DF EE 26 02 E4 06 FF EE 01 05 DF EE 30
01 01 F3 12
```

Note 1) Decrypt `57` to HEX `3B3437363137333930303130313030313030D32303132323031303132333435363738393F00000000` and view the decryption detail data, then transfer Hex output to ASCII Text `;4761739001010010=20122010123456789?`

Note 2) Decrypt `5A` to HEX `34373631373339303031303130303130` and view the decryption detail data, then transfer Hex output to ASCII Text `4761739001010010`

Ex2: Present "EMV Test Card V2 T=0" contact test card using 02-40 and view breaking down data below.

```
6/9/2020 17:41:23.420 [TX] - 56 69 56 4F 74 65 63 68 32 00 02 40 00 0F
0A 9F 02 06 00 00 00 00 10 00 DF EF 37 01 04 59 74
6/9/2020 17:41:23.501 [RX] - 56 69 56 4F 74 65 63 68 32 00 02 63 00 00
4B B4 56 69 56 4F 74 65 63 68 32 00 02 00 00 E5 E4 DF EE 12 0A AA FF
98 76 54 32 10 00 00 0B DF EE 25 02 00 10 57 A1 11 47 61 CC CC CC CC
00 10 D2 01 2C CC CC CC CC CC CC 57 C1 28 F7 4E 1C 40 D6 82 AA 06 72
3B 1D 6A 40 1D E3 55 E5 77 07 E5 40 13 BE 11 76 11 E5 70 40 3A BD 73
F4 57 9B DC 3C D1 75 68 5A A1 08 47 61 CC CC CC CC 00 10 5A C1 10 CD
B7 53 8B 3B 14 92 12 F8 0B 07 DA 58 AD CD 13 5F 34 01 01 5F 20 0F 46
55 4C 4C 20 46 55 4E 43 54 49 4F 4E 41 4C 5F 24 03 20 12 31 9F 1E 08
54 65 72 6D 69 6E 61 6C 9F 20 00 5F 25 03 95 07 01 5F 2D 08 65 73 65
6E 66 72 64 65 50 0A 56 49 53 41 43 52 45 44 49 54 4F 07 A0 00 00 00
```

03 10 10 84 07 A0 00 00 00 03 10 10 DF EE 23 00 9F 39 01 05 9F 53 00
DF EE 26 02 E4 06 FF EE 01 05 DF EE 30 01 01 A3 1E

Note 1: Decrypt 57 to HEX
3B3437363137333930303130313030313030313030D3230313232303130313233343536373839
3F00000000 and view the decryption detail data, then transfer HEX output to ASCII Text
;4761739001010010=20122010123456789?

Note 2: Decrypt 5A to HEX
34373631373339303031303130303130 and view the decryption detail data, then transfer Hex output to ASCII Text 4761739001010010

7.5. A3) Contactless Output Format with TransArmor TDES-DUKPT

Ex1: Present "MasterCard pay pass, Demo Card. This is not a real MasterCard card" MSD contactless demo card using 02-40 and view breaking down data below.

6/9/2020 17:51:11.758 [TX] - 56 69 56 4F 74 65 63 68 32 00 02 40 00 0A
0A 9F 02 06 00 00 00 00 10 00 18 73
6/9/2020 17:51:14.242 [RX] - 56 69 56 4F 74 65 63 68 32 00 02 23 02 F8
F5 DF EE 12 0A AA FF 98 76 54 32 10 00 00 0C DF EF 17 A1 3E 2A 35 32
35 36 2A 2A 2A 2A 2A 2A 2A 2A 30 30 30 30 5E 53 75 70 70 6C 69 65 64
2F 4E 6F 74 5E 31 32 31 32 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A DF EF 17 C1 40 32 95 AD CB 07
34 95 4B 4D 8C C9 EB 9F C2 66 D5 69 67 9B 1E 88 42 C9 4C D2 1E 2F E3
2F A1 A8 47 BC 1F 71 6D 35 DB 32 E8 02 35 03 90 60 0F 46 8B AA 15 99
6F 86 F5 5C 6C 1A 7A 68 61 D2 2D 8A 15 DF EF 18 A1 25 35 32 35 36 2A
2A 2A 2A 2A 2A 2A 2A 30 30 30 30 3D 31 32 31 32 2A 2A 2A 2A 2A 2A 2A
2A 2A 2A 2A 2A 2A 2A 2A 2A DF EF 18 C1 28 36 68 D7 DC 5A 90 13 BE 6C
FF 16 0F 20 1C 42 E1 E3 89 F5 78 6D B5 6A 9D 1E B4 30 89 38 19 99 6D
11 03 C6 90 19 16 FD 82 9A 03 20 06 09 9F 21 03 09 50 20 9F 37 04 95
22 15 D7 9F 6A 04 00 00 00 11 9C 01 00 9F 41 04 00 00 00 12 9F 39 01
91 9F 66 02 01 F6 9B 02 00 00 DF 81 16 16 1E 04 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 DF 81 29 08 30 F0 F0 00 30 F0
FF 00 FF 81 06 44 DF 81 15 06 00 00 00 00 00 FF DF 81 2A C1 20 A1 E7
32 86 E2 03 38 5B BC 35 DB 50 E3 EE A0 8F 82 2A E8 3E 07 11 1D 47 2E
8C 47 0A 3D 59 11 D7 DF 81 2B C1 10 B3 83 67 E5 46 81 21 1E A1 42 68
35 88 27 35 55 FF 81 05 81 E1 9F 6D 02 00 01 50 0A 4D 61 73 74 65 72
43 61 72 64 84 07 A0 00 00 00 04 10 10 56 A1 3E 2A 35 32 35 36 2A 2A
2A 2A 2A 2A 2A 2A 30 30 30 30 5E 53 75 70 70 6C 69 65 64 2F 4E 6F 74
5E 31 32 31 32 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
2A 2A 2A 2A 2A 2A 2A 2A 2A 56 C1 40 32 95 AD CB 07 34 95 4B 4D 8C C9
EB 9F C2 66 D5 69 67 9B 1E 88 42 C9 4C D2 1E 2F E3 2F A1 A8 47 BC 1F
71 6D 35 DB 32 E8 02 35 03 90 60 0F 46 8B AA 15 99 6F 86 F5 5C 6C 1A
7A 68 61 D2 2D 8A 15 9F 6B A1 13 52 56 CC CC CC CC 00 00 D1 21 2C CC
CC CC CC CC CC CC CC 9F 6B C1 28 36 68 D7 DC 5A 90 13 BE 6C FF 16 0F
20 1C 42 E1 E3 89 F5 78 6D B5 6A 9D 1E B4 30 89 38 19 99 6D 11 03 C6
90 19 16 FD 82 FF EE 01 05 DF EE 30 01 00 9F 1E 08 35 54 36 36 33 32
30 35 DF ED 29 68 32 95 AD CB 07 34 95 4B 4D 8C C9 EB 9F C2 66 D5 69
67 9B 1E 88 42 C9 4C D2 1E 2F E3 2F A1 A8 47 BC 1F 71 6D 35 DB 32 E8
02 35 03 90 60 0F 46 8B AA 15 99 6F 86 F5 5C 6C 1A 7A 68 61 D2 2D 8A

15 BD 8A 90 A6 1D FC C0 82 4C 53 8D AB F5 8C 02 17 F1 36 CB 64 E7 19
2E 7B EE BD DD 32 55 4B 33 2F EF 67 9B 59 E7 48 C6 6A DF EE 26 02 F5
06 E6 DB

Note 1: Decrypt DFEF17 to HEX
2542353235363833323033303030303030305E537570706C6965642F4E6F745E313231
3235303233343831303333383531313131313131313131313131323F and view the decryption detail data, then transform HEX output to ASCII Text
%B5256832030000000^Supplied/Not^1212502348103385111111111111112?

Note 2: Decrypt DFEF18 to HEX
3B3532353638333230333030303030303D3132313235303231333131303333383531
3131323F00 and view the decryption detail data, then transform HEX output to ASCII Text
;5256832030000000=12125021311033851112?

Note 3: Decrypt 56 to HEX
2542353235363833323033303030303030305E537570706C6965642F4E6F745E313231
3235303233343831303333383531313131313131313131313131323F and view the decryption detail data, then transform HEX output to ASCII Text
%B5256832030000000^Supplied/Not^1212502348103385111111111111112?

Note 4: Decrypt 9F6B to HEX
3B3532353638333230333030303030303D3132313235303231333131303333383531
3131323F00 and see the decryption detail data, then transform HEX output to ASCII Text
;5256832030000000=12125021311033851112?

Note 5: Decrypt DFED29 to HEX
2542353235363833323033303030303030305E537570706C6965642F4E6F745E313231
3235303233343831303333383531313131313131313131313131323F3B3532353638
3332303330303030303030303D3132313235303231333131303333383531313131323F00
and see the decryption detail data, then transform HEX output to ASCII Text
%B5256832030000000^Supplied/Not^1212502348103385111111111111112?;52568
32030000000=12125021311033851112?

Ex2: Present "VISA CLQD011" EMV contactless test card using 02-40 and see breaking down data below.
6/9/2020 18:04:29.999 [TX] - 56 69 56 4F 74 65 63 68 32 00 02 40 00 0A
0A 9F 02 06 00 00 00 00 10 00 18 73
6/9/2020 18:04:32.398 [RX] - 56 69 56 4F 74 65 63 68 32 00 02 23 01 B4
E5 DF EE 12 0A AA FF 98 76 54 32 10 00 00 0D 50 0C 56 49 53 41 20 54
45 53 54 20 30 31 57 A1 13 47 61 CC CC CC CC 00 10 D2 01 2C CC CC CC
CC CC CC CC CC 57 C1 28 17 1B D8 E7 2D 6F 34 AA 87 FF CC C9 8D E6 61
2D D4 9E 3F E9 57 13 AF C5 3E B6 C2 83 D5 2E AB A8 0D 3A 34 E6 7B 3F
66 3C 5A A1 08 47 61 CC CC CC CC 00 10 5A C1 10 71 0E 2D 1A 62 6A 38
15 99 E1 B7 46 2C C0 81 62 82 02 20 00 84 08 A0 00 00 00 03 10 10 05
95 05 00 00 00 00 00 9A 03 20 06 09 9B 02 00 00 9C 01 00 5F 24 03 20
12 31 5F 2A 02 08 40 5F 2D 04 65 73 65 6E 5F 34 01 01 9F 02 06 00 00
00 00 10 00 9F 03 06 00 00 00 00 00 00 9F 06 08 A0 00 00 00 03 10 10

```
05 9F 10 07 06 01 11 03 90 00 00 9F 11 01 01 9F 12 0C 50 72 65 66 20
4E 61 6D 65 20 30 31 9F 1A 02 08 40 9F 1B 04 00 00 17 70 9F 21 03 10
03 39 9F 26 08 AA BB CC DD EE FF 11 22 9F 27 01 40 9F 33 03 00 08 E8
9F 36 02 00 03 9F 37 04 42 CF F4 2C 9F 39 01 07 9F 45 02 00 00 9F 4C
08 00 00 00 00 00 00 00 00 9F 4E 1E 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9F 5D 06 00
00 00 01 00 00 9F 66 04 30 00 40 00 9F 69 07 01 11 22 33 44 20 00 9F
6C 02 20 00 DF 81 26 06 00 00 00 00 80 00 FF EE 01 14 DF EE 30 01 00
DF EE 52 01 00 DF EE 5B 01 05 DF EF 7F 01 00 9F 41 04 00 00 00 13 9F
34 03 3F 00 00 9F 1E 08 35 54 36 36 33 32 30 35 DF EE 26 02 E5 06 50
1F
```

Note 1: Decrypt `57` to HEX `3B3437363137333930303130313030313030D32303132313230303031323333393930303033313F00` and see the decryption detail data, then transfer HEX output to ASCII Text `;4761739001010010=20121200012339900031?`

Note 2: Decrypt `5A` to HEX `34373631373339303031303130303130` and see the decryption detail data, then transfer Hex output to ASCII Text `4761739001010010`

Ex3: Present "VISA MSD-N3" EMV contactless test card using 02-40 and see breaking down data below.

```
6/9/2020 18:11:07.180 [TX] - 56 69 56 4F 74 65 63 68 32 00 02 40 00 0A
0A 9F 02 06 00 00 00 00 10 00 18 73
6/9/2020 18:11:10.010 [RX] - 56 69 56 4F 74 65 63 68 32 00 02 0A 02 32
E5 DF EE 12 0A AA FF 98 76 54 32 10 00 00 0E DF EE 02 04 20 90 00 21
50 04 56 69 73 61 56 A1 35 2A 34 30 31 32 2A 2A 2A 2A 2A 2A 2A 2A 31
38 38 31 5E 54 45 53 54 2F 56 49 53 41 20 4D 53 44 20 5E 31 32 31 32
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 56 C1 38 36 68 61 F6
04 2C B3 E5 4A A2 67 F2 11 3D F7 76 51 03 9A 6F 0D FD DD B6 9A D1 CC
8D 77 DD 59 F5 87 83 CA 33 95 C1 FC 61 05 F8 D1 DA 34 87 CE D2 8A B1
8F CC 89 7A F4 D7 57 A1 13 40 12 CC CC CC CC 18 81 D1 21 2C CC CC CC
CC CC CC CC CC 57 C1 28 FC 4F 60 A0 BE 0B D8 55 A3 21 6E 82 85 1A 8D
09 4B C3 E1 58 0A FA 3D 80 CA 71 07 00 9D 60 1F 4C 87 95 05 80 F1 9D
8B A3 5A A1 08 40 12 CC CC CC CC 18 81 5A C1 10 2F 50 6D 4C 55 AE 2F
84 7E 48 FA 0E EF 4B DE 6C 82 02 00 80 95 05 00 00 00 00 00 9A 03 20
06 09 9B 02 00 00 9C 01 00 5F 20 0E 54 45 53 54 2F 56 49 53 41 20 4D
53 44 20 5F 24 03 12 12 31 5F 2A 02 08 40 9F 02 06 00 00 00 00 10 00
9F 03 06 00 00 00 00 00 00 9F 06 07 A0 00 00 00 03 10 10 9F 09 02 00
01 9F 1A 02 08 40 9F 1B 04 00 00 17 70 9F 21 03 10 10 16 9F 33 03 00
08 E8 9F 37 04 0E 64 6E BA 9F 39 01 07 9F 45 02 00 00 9F 4C 08 00 00
00 00 00 00 00 00 9F 4E 1E 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 9F 66 04 30 00 40 00
FF EE 01 0F DF EE 30 01 00 DF EE 5B 01 05 DF EF 7F 01 00 5F 30 02 05
01 9F 41 04 00 00 00 14 9F 1E 08 35 54 36 36 33 32 30 35 DF ED 29 60
36 68 61 F6 04 2C B3 E5 4A A2 67 F2 11 3D F7 76 51 03 9A 6F 0D FD DD
B6 9A D1 CC 8D 77 DD 59 F5 87 83 CA 33 95 C1 FC 61 05 F8 D1 DA 34 87
CE D2 26 D1 B9 5F 2B 2E D3 8F 1B BE 4C 8A 33 BB 3D F9 70 1F BA 1D 65
99 94 05 7F 6E 7C BD D7 C2 67 BA 48 F1 B2 17 83 23 7A 06 75 C0 D7 CC
C7 CE 0C 40 DF EE 26 02 E5 06 7C 01
```

Note 1: Decrypt 56 to HEX 2542343031323838383838383838383831383831 5E544553542F56495341204D5344205E31323132353031313233343536373839313131313F00 and see the decryption detail data, then transform HEX output to ASCII Text %B4012888888881881^TEST/VISA MSD ^1212501123456789 1111?

Note 2: Decrypt 57 to HEX 3B34303132383838383838383838313838313D31323132313031303030303030303031 3131313F00 and see the decryption detail data, then transform HEX output to ASCII Text ;4012888888881881=1212101000000001111?

Note 3: Decrypt 5A to HEX 34303132383838383838383831383831 and view the decryption detail data, then transform HEX output to ASCII Text 4012888888881881

Note 4: Decrypt DFED29 to HEX 25423430313238383838383838383838313838315E544553542F56495341204D5344205E31 323132353031313233343536373839313131313F3B34303132383838383838383838383138 38313D31323132313031303030303030303031313131 3F0000 and view the decryption detail data, then transfer Hex output to ASCII Text %B4012888888881881^TEST/VISA MSD ^1212501123456789 1111?;4012888888881881=1212101000000001111?

Ex4: Present "DISCOVER Dual Interface D-PAS" EMV contactless test card using 02-40 and view breaking down data below.

```
6/9/2020 18:21:24.186 [TX] - 56 69 56 4F 74 65 63 68 32 00 02 40 00 0A
0A 9F 02 06 00 00 00 00 10 00 18 73
6/9/2020 18:21:25.991 [RX] - 56 69 56 4F 74 65 63 68 32 00 02 23 02 E3
F5 DF EE 12 0A AA FF 98 76 54 32 10 00 00 0F DF EF 17 A1 41 2A 36 35
31 30 2A 2A 2A 2A 2A 2A 2A 2A 30 34 32 32 5E 43 41 52 44 2F 49 4D 41
47 45 20 32 33 20 20 20 20 20 20 20 20 20 20 20 20 20 5E 31 37 31 32
2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A DF EF 17 C1 48 C6 6F
62 F5 80 EE 26 B2 33 4C EF 1E 9A 88 CA 7F 38 B7 21 28 81 1D A2 B7 58
65 95 81 5F EA 1D 75 1A 8E FD 19 5D C3 7A 6B 16 9F 5B 31 86 8F C2 7B
1D 87 76 8A C9 7A BF F7 29 85 F2 D3 57 64 A3 7C 5A 66 F2 E7 1D D6 E8
0C DF EF 18 A1 25 36 35 31 30 2A 2A 2A 2A 2A 2A 2A 2A 30 34 32 32 3D
31 37 31 32 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A DF EF 18
C1 28 17 74 84 6A 7F B8 B1 98 91 2C 21 67 25 CA 2D DB B0 23 F9 F2 56
B6 F2 9B A3 94 60 EA 36 A0 8A 12 66 64 44 EF 4A 6A B5 72 DF 81 29 08
30 F0 F0 F0 A8 F0 FF 00 FF 81 05 82 01 00 9F 02 06 00 00 00 00 10 00
9F 03 06 00 00 00 00 00 00 9F 33 03 00 08 E8 9F 1A 02 08 40 9F 35 01
25 56 A1 41 2A 36 35 31 30 2A 2A 2A 2A 2A 2A 2A 2A 30 34 32 32 5E 43
41 52 44 2F 49 4D 41 47 45 20 32 33 20 20 20 20 20 20 20 20 20 20 20
20 20 5E 31 37 31 32 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A
56 C1 48 C6 6F 62 F5 80 EE 26 B2 33 4C EF 1E 9A 88 CA 7F 38 B7 21 28
```

81 1D A2 B7 58 65 95 81 5F EA 1D 75 1A 8E FD 19 5D C3 7A 6B 16 9F 5B
31 86 8F C2 7B 1D 87 76 8A C9 7A BF F7 29 85 F2 D3 57 64 A3 7C 5A 66
F2 E7 1D D6 E8 0C 57 A1 13 65 10 CC CC CC CC 04 22 D1 71 2C CC CC CC
CC CC CC CC CC 57 C1 28 17 74 84 6A 7F B8 B1 98 91 2C 21 67 25 CA 2D
DB B0 23 F9 F2 56 B6 F2 9B A3 94 60 EA 36 A0 8A 12 66 64 44 EF 4A 6A
B5 72 9A 03 20 06 09 9C 01 00 95 05 00 00 00 00 00 DF 81 16 16 1B 04
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF EE 01
05 DF EE 30 01 00 9F 37 01 76 9F 06 08 A0 00 00 03 24 10 10 01 9F 39
01 91 9F 41 04 00 00 00 15 9F 34 03 00 00 00 84 08 A0 00 00 03 24 10
10 01 9F 21 03 10 20 33 9F 1E 08 35 54 36 36 33 32 30 35 DF ED 29 70
C6 6F 62 F5 80 EE 26 B2 33 4C EF 1E 9A 88 CA 7F 38 B7 21 28 81 1D A2
B7 58 65 95 81 5F EA 1D 75 1A 8E FD 19 5D C3 7A 6B 16 9F 5B 31 86 8F
C2 7B 1D 87 76 8A C9 7A BF F7 29 85 F2 D3 57 64 A3 7C 25 9B B8 0A 54
72 AB 21 57 25 48 C4 AC EE 68 A2 3D 6A 27 30 A1 D9 ED 84 BF 12 D3 BC
9A B5 4B F2 19 54 AD 00 65 F9 8E A2 13 DC 54 60 B1 74 83 2A DF EE 26
02 F5 06 13 70

Note 1: Decrypt DFEF17 to HEX
254236353130303030303030303030303432325E434152442F494D414745203233202020
202020202020202020205E31373132323031313030303038373630303030303F000000
0000 and view the decryption detail data, then transform HEX output to ASCII Text
%B6510000000000422^CARD/IMAGE 23 ^17122011000087600000?

Note 2: Decrypt DFEF18 to HEX
3B3635313030303030303030303034323233D31373132323031313030303038373630 30
3030303F00 and view the decryption detail data, then transform HEX output to ASCII Text
;6510000000000422=17122011000087600000?

Note 3: Decrypt 56 to HEX
254236353130303030303030303030303432325E434152442F494D414745203233202020
202020202020202020205E31373132323031313030303038373630303030303F000000
0000 and view the decryption detail data, then transform HEX output to ASCII Text
%B6510000000000422^CARD/IMAGE 23 ^17122011000087600000?

Note 4: Decrypt 57 to HEX
3B3635313030303030303030303034323233D31373132323031313030303038373630 30
3030303F00 and view the decryption detail data, then transform HEX output to ASCII Text
;6510000000000422=17122011000087600000?

Note 5: Decrypt DFED29 to HEX
254236353130303030303030303030303432325E434152442F494D414745203233202020
202020202020202020205E31373132323031313030303038373630303030303F3B3635
31303030303030303030303432323D3137313232303131303030303837363030303030
3F000000000000 and view the decryption detail data, then transform HEX output to ASCII Text
%B6510000000000422^CARD/IMAGE 23
^17122011000087600000?;6510000000000422
=17122011000087600000?

7.6. TransArmor TDES FAQ

Q: How can I enable TransArmor(TA) TDES encryption for a DUKPT data encryption key (DEK)?

A: Follow these steps:

1. NONSRD load cert. by **PKI tool**
2. Load LCLKEY by **Key Injection.exe**
3. Load DEK with TDES by **Key Injection.exe**
4. Choose DUKPT DEK Encryption Type (TransArmor TDES) with command C7-32
   `[TX] - 56 69 56 4F 74 65 63 68 32 00 C7 32 00 01 02 BC 4D`
5. Enable msr/msd/ct/cl Data Encryption with command C7-36
   `[TX] - 56 69 56 4F 74 65 63 68 32 00 C7 36 00 01 03 6C 97`

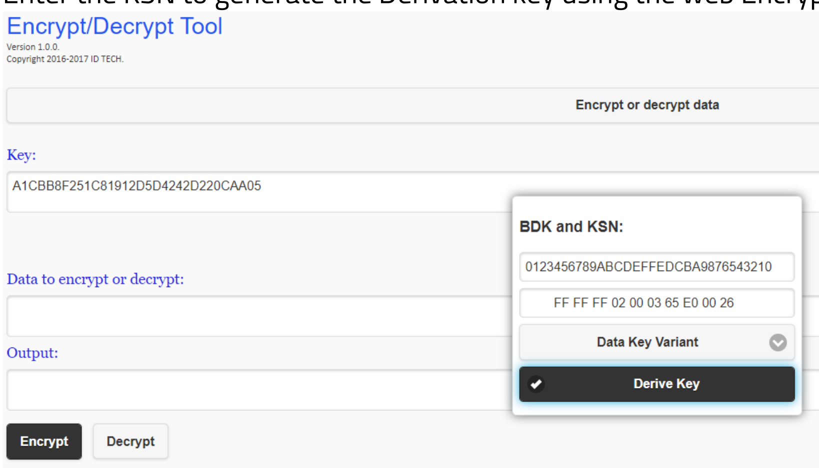
Q: How to clean up the Encryption Type of the TDES-DUKPT DEK?

Use the PKI tool to perform an **SMFG Master Reset**.

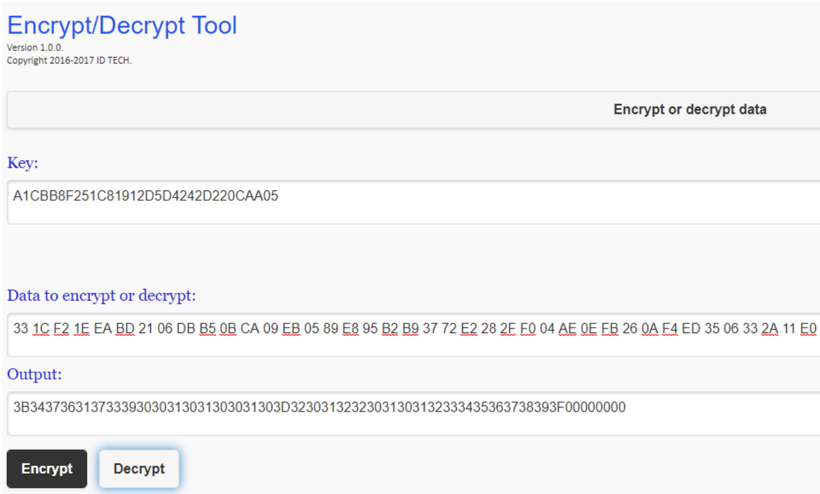
Q: How do I decrypt Tag DFED29?

A: Follow these steps:

1. Go to [the web Encrypt/Decrypt Tool]() or [parsomatic.]()
2. Enter the KSN to generate the Derivation key using the web Encrypt/Decrypt Tool



3. Enter the value of DFED29 and click **Decrypt**.



4. Translate the HEX output to ASCII Text using the [Hex to ASCII Text Converter]
   HEX output:
   `3B343736313733393030313031303031303D323031323230313031323334353637 38393F00000000`
   ASCII Text: `;4761739001010010=20122010123456789?`

8. Appendix A: Tags DFEF4B, DFEF4C, & DFEF4D

ID TECH proprietary tags DFEF4B, DFEF4C, and DFEF4D provide a way for track data (and optionally, PAN data) to be supplied in conjunction with an EMV transaction, with or without sentinels, in a form similar to the form track data would take in a conventional MSR transaction.

8.1. Tag DFEF4B

Tag DFEF4B is a configuration tag. Use it to tell your ID TECH reader which tracks you want to receive in tag DFEF4D, whether or not to use sentinels, and whether or not to include the PAN as a separate string.

Byte 1:

8	7	6	5	4	3	2	1	NOTES
-	-	-	-	-	-	-	X	0 – Disable Track 3 Sentinels 1 – Enable Track 3 Sentinels
-	-	-	-	-	-	X	-	0 – Disable Track 2 Sentinels 1 – Enable Track 2 Sentinels
-	-	-	-	-	X	-	-	0 – Disable Track 1 Sentinels 1 – Enable Track 1 Sentinels
-	-	-	-	X	-	-	-	0 – Disable Track 3 1 – Enable Track 3
-	-	-	X	-	-	-	-	0 – Disable Track 2 1 – Enable Track 2
-	-	X	-	-	-	-	-	0 – Disable Track 1 1 – Enable Track 1
-	X	-	-	-	-	-	-	0 – Disable PAN 1 – Enable PAN
X	-	-	-	-	-	-	-	0 – All Data Elements Found 1 – Only First Element Found

Byte 2: RFU
Byte 3: RFU

You can use the top bit of the first byte of DFEF4B to control search behavior: If the bit is OFF, all data elements requested will be provided (if they exist). If the bit is ON, only the first element found will be retrieved and placed in DFEF4D.

If you request multiple data items, they will be concatenated. To know the original lengths of the items, you must retrieve and inspect Tag DFEF4C (see below).

To use tag DFEF4B, add it (as a TLV) to your terminal configuration settings. Send the settings to your device as you normally would. (For example, in ID TECH's Augusta, use command 72 46 02 03 to Set Terminal Settings.)

NOTE: If this tag does not exist in Terminal Settings, tags DFEF4C and DFEF4D will not be generated.

The default value of this tag is 0x12 (Track 2 enabled, with Sentinels).

8.2. Data Search Order

When **"Only First Element Found" (bit 8 = 1)** is set in DFEF4B, Tag DFEF4D will be populated with a *single* data element according to the following search order

Track 2, Tag 57 (converted to alpha numeric format) Track 2, Tag 9F6B
Track 2, Tag 5F22 Track 1, Tag 56 Track 1, Tag 5F21
PAN, Tag 5A (converted to alpha numeric format) Track 3, Tag 58
Track 3, Tag 5F23

Regardless of the original format, the data will be placed in the DFEF4D tag in alpha numeric format, such that after decryption (and with padding removed) the data will look similar to:

3b343736313733393030313031303031303d31353132323031313134333837383038393f

Which means that after rendering it as ASCII, it would look like:

;4761739001010010=15122011143878089?

When **"All Data Elements Found" (BIT 8),** is specified in DFEF4B, Tag DFEF4D will be populated with a single instance of each requested data element, according to the following order:

Track 1 requested (bit 6 = 1). Includes first instance of:

Tag 56 = Track 1 Equivalent
Tag 5F21 = Track 1, identical to the data coded

Track 2 requested (bit 5 = 1). Includes first instance of:

Tag 57 = Track 2 Equivalent (converted to alpha numeric format) Tag 9F6B = Track 2 Data
Tag 5F22 = Track 2, identical to the data coded

Track 3 requested (bit 4 = 1). Includes first instance of:

Tag 58 = Track 3 Equivalent
Tag 5F23 = Track 3, identical to the data coded

PAN requested (bit 7 = 1). Includes:

Tag 5A = PAN (converted to alpha numeric format)

8.2.1. Sentinels

For any found data element of Track1, Track2 or Track3, sentinels will be included or not included according to the preferences set in bits 1, 2 and 3.

8.2.2. Compressed Numeric Elements

For any data element captured as compressed numeric, the following rules shall apply:

Padding (0xf) shall not be included
Center separators: 0xd shall be converted to 0x3d ("=")
Data shall be encoded as ASCII representation of binary data example 0x123f = 0x313233 = "123" (ignore padding) example 0x1234 = 0x31323334 = "1234"
example 0x123d456f = 0x3132333d343536 = "123=456"

8.3. Tag DFEF4C

This tag's 6-byte value provides the native lengths of tracks 1, 2, and 3, and the PAN (if applicable). Two bytes are reserved for future use.

<Track 1 Length><Track 2 Length><Track 3 length><PAN length><RFU><RFU>

A length of 0 indicates *track disabled* in DFEF4B or *data not available*. This tag also serves as an indicator of which data element was found first, when "Only First Element Found" is enabled in DFEF4B.

8.4. Tag DFEF4D

This variable-length tag contains track and/or PAN data, encrypted. The exact contents will vary depending on values supplied previously in DFEF4B (see above).

When TDES or AES encryption have been used in conjunction with traditional DUKPT, decrypt the data normally, using the 10-byte KSN found in tag DFEE12.

When TransArmor PKI (RSA) data are present, decrypt with the KeyID value in DFEE12 and Terminal ID found in 9F1C.

(Note: Each track encoded with TransArmor RSA will be encrypted to 344 bytes.)

9. Revision History

Rev	Description and Reason for Change	Date	By
Rev 50/A	Initial Draft	1-26-2016	KT
B	Edit to Section 5 to remove reference to IDG. Also, fix "Field 11 is always empty" to say Field 12 is always empty. Increment the two following references to match earlier table.	2-19-2016	KT
C	Revised tables to show that tag 56 and 57 data are masked, and also FFEE13, FFEE14, 9F6B are masked. Substitute "captured data type" for "card type" where appropriate. Updated references to Attribution Byte (and/or DFEE26) to reflect the latest bit-4 semantics. "Magstripe data (MSD) constructed from contactless interactions are treated as MSR data" changed to say "Magstripe data (MSD) constructed from contactless interactions are treated as EMV data." 5/27/2016 Added changes for tags 56, 57, 9F 6/21/2016 Clarified treatment of bit 5, field 8, of MSR data.		KT
D	Added updates related to TransArmor crypto support. Added updates for MAC support. P/N of this document updated to 80000502-001	8/5/2016 8/16/2016 9/6/2016	KT
E	Added DFEF48 tag (insufficient RAM) with examples. Added detailed example of HMAC calculation. Document now aligns with 66A of ICC and 67 of EEMSR.	9/23/2016	KT
F	Add tags DFEF4B, DFEF4C, DFEF4D (new encryption tags).	10/03/2016	KT
G	Add support for TransArmor TDES (symmetric key) encryption.	11/20/2017	KT
H	Replaced **Example: MSR Output Format with TransArmor TDES-DUKPT** Added missing byte 6 in Field 4: Track Status under ID TECH ENHANCED ENCRYPTED MSR DATA OUTPUT FORMAT Field Descriptions.	12/19/2019	CB
J	Added section for TransArmor TDES-DUKPT (Symmetric Key) information. Updated document font/style.	07/6/2020	CB