

|  |
| --- |
| DUKPT - PIN |

|  |  |  |  |
| --- | --- | --- | --- |
| Processo: | FIDELITY - Consultoria DUKPT | Versão: | 1.0 |
| Autor: | Eduardo Mendes Campos | Data: | 07/02/2021 |
| Revisor: | Thiago Cox Alonso | Data: |  |
| Aprovador: |  | Data: |  |

Índice

[1. Introdução 4](#_Toc63670371)

[2. Variantes Single-DES, Triple-DES e AES do DUKPT 5](#_Toc63670372)

[3. Chaves DUKPT 5](#_Toc63670373)

[3.1. Base Derivation Key (BDK – Chave Base de Derivação) 5](#_Toc63670374)

[3.2. Initial Key (IKEY ou IK – Chave Inicial) 6](#_Toc63670375)

[3.3. Key Serial Number – KSN 6](#_Toc63670376)

[3.4. Transaction Keys – Chaves de Transação 7](#_Toc63670377)

[4. Processamento pelo Adquirente 8](#_Toc63670378)

[5. Resumo das Operações de DUKPT 8](#_Toc63670379)

[6. Domínios Criptográficos 9](#_Toc63670380)

[6.1. Variant 9](#_Toc63670381)

[6.1.1. Criação da BDK 9](#_Toc63670382)

[6.1.2. Simulação da criação do PIN Block no Terminal 10](#_Toc63670383)

[6.1.3. Tradução do PIN Block DUKPT para ZPK 12](#_Toc63670384)

[6.2. Key Block 13](#_Toc63670385)

[6.2.1. Criação da BDK 13](#_Toc63670386)

[6.2.2. Simulação da criação do PIN Block no Terminal 14](#_Toc63670387)

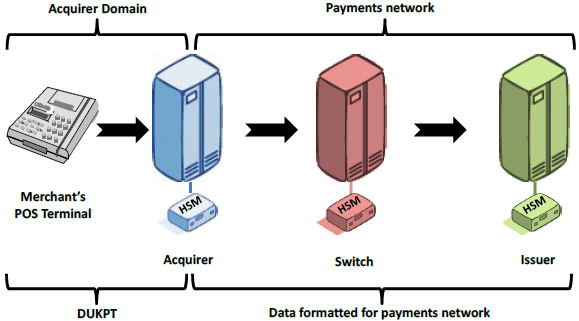
[6.2.3. Tradução do PIN Block DUKPT para ZPK 14](#_Toc63670388)

[6.3. Verificação do PIN Block traduzido 16](#_Toc63670389)

# Introdução

DUKPT (*Derived Unique Key Per Transaction* – Chave única derivada por transação) é um esquema para gerenciar chaves de criptografia em um ambiente de pagamento por cartões. Tem sido usado tradicionalmente com terminais POS (*Point of Sale* – Pontos de Venda), mas a adoção tem crescido em para outras aplicações – e assim o DUKPT tem se tornado cada vez mais importante na área de pagamentos.

DUKPT é usado para encriptar PIN Blocks, encriptar outros dados e mensagens de autenticação (MACing). DUKPT é tipicamente usado entre comerciantes e seus adquirentes. O adquirente irá remontar os dados de transação para a transição para a rede de pagamentos (a qual usa chaves de gerenciamento Master/Session) antes de enviá-la para um switch de pagamento.



A força do DUKPT está no fato de que uma chave nova e única é gerada para cada transação, de tal forma que se uma chave de transação é comprometida ela não pode ser usada para atacar transações anteriores naquele terminal ou em qualquer outro terminal. Além disso, o gerenciamento de chaves no ambiente DUKPT é simplificado por possuir uma única master key que pode gerenciar um lote inteiro de terminais.

Este método envolve o uso de um valor em claro KSN (*Key Serial Number*) e uma chave secreta, a BDK. Em cada transação o POS usa uma chave única baseada na chave anterior e o KSN, o qual contém um contador de transações. O POS encripta a chave com esta nova chave e então retorna tanto o PIN Block encriptado quanto o KSN ao adquirente. No HSM, a chave gerada pelo terminal é derivada dinamicamente e independente do POS usando a BDK original em conjunto com o KSN enviado pelo POS.

A mesma BDK pode ser usada por milhares de terminais pois cada POS tem um serial number único. Portanto, cada terminal produz uma chave exclusiva para cada transação e um ataque criptográfico bem sucedido em um PIN Block não surtirá efeito em outros. O adquirente deve gerir uma quantidade relativamente pequena de BDKs e o algoritmo para derivação de uma chave de transação é desenhado te tal forma que causa pouca sobrecarga ao HSM.

O Host é responsável pela gestão das BDKs. Para cada transação, deve verificar se o KSN fornecido pelo POS é válido e buscar nos ambientes de armazenamento de chaves a BDK apropriada para a transação, identificando-a pela porção mais à esquerda do KSN.

# Variantes Single-DES, Triple-DES e AES do DUKPT

O HSM payShield suporta DUKPT para os algoritmos *single DES*, *Triple-DES* e *AES*.

# Chaves DUKPT

Três níveis de chaves são utilizados no DUKPT:

1. **Base Derivation Key** (BDK – Chave Base de Derivação) – uma chave mestra de propriedade do adquirente. A BDK é utilizada para um grande de número de terminais – talvez todos os terminais de um fornecedor, para um modelo de terminais ou uma faixa de números de série.
2. **Initial Key** (IKEY ou IK – Chave Inicial) – uma chave que é única para um terminal. A IKEY é usada para iniciar a sequência de chaves de transação e então é descartada pelo terminal.
3. **Transaction Key** (Chaves de Transação) – gerada dentro do terminal. Chaves para encriptar PIN, encriptação de dados e MACing são derivadas da chave de transação. Cada transação é processada com uma chave única para proteger seus dados. Quando o dado encriptado é recebido pelo adquirente, ele irar derivar a mesma chave de transação usando o mesmo processo que o terminal usou para derivar a chave de encriptação.

Pode ser visto por este processo que não há exigências para que o adquirente e o terminal troquem chaves – exceto no improvável evento de um terminal gerar um milhão de transações e, portanto, necessitar de uma nova IKEY.

O método DUKPT é definido no padrão ANSI X9.24-3-2017 (*Retail Financial Services Symmetric Key Management - Part 3: Derived Unique Key Per Transaction*).

## Base Derivation Key (BDK – Chave Base de Derivação)

Uma BDK é uma chave que geralmente é gerada e de propriedade do adquirente e é usada para muitos terminais. (Se uma BDK é de propriedade de uma organização que não seja o adquirente, será necessário que seja distribuída para qualquer outra organização que esteja envolvida na geração de IKEYs.) BDKs múltiplas serão geralmente suportadas para permitir diferentes famílias ou grupos de terminais. A distribuição de BDKs pode ser feita:

* Eletronicamente, com a BDK protegida por uma ZMK, ou
* Na forma de componentes impressos, com custódios diferentes reunindo-se para habilitar a BDK ser formada através de seus componentes.

Uma BDK armazenada precisa ser protegida por encriptação usando uma chave que encripta chave (KEK) apropriada sempre que estiver fora de domínio criptográfico seguro. O payShield protege suas chaves usando a LMK.

Os comandos de host abaixo são adequados para gestão de BDKs:

|  |  |
| --- | --- |
| **Código de Comando** | **Descrição** |
| A0 | Gerar uma BDK randômica e retorná-la à aplicação encriptada pela LMK |
| A6 | Importar uma BDK |
| A8 | Exportar uma BDK |

## Initial Key (IKEY ou IK – Chave Inicial)

A IKEY (originalmente citada como IPEK – Initial PIN Encryption Key) é única por terminal. A IKEY é calculada através:

* Da BDK
* Do Key Serial Number (KSN) que é único por terminal.

Uma vez criada, a IKEY é instalada no terminal. (The IKEY é também recriado transientemente pelo adquirente quando processa transações do terminal com a finalidade de derivar a mesma chave de transação que o terminal usou para encriptar seus dados.).

## Key Serial Number – KSN

O KSN tem um tamanho máximo de 80 bits e tem a seguinte estrutura:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Tamanho** | **Descrição** |
| **KSI** - Conjunto de identificação da Chave | 5 a 9 caracteres hexadecimais (20 a 36 bits) | Identifica a BDK a ser usada por este terminal. |
| Identificador da sub-chave | 1 caractere Hexadecimal (4 bits) | Definido como 0 |
| **DID** – Identificador do dispositivo | 2 a 5 caracteres hexadecimais (8 a 20 bits) | Identificador único (i.e., número de série) para este terminal (sempre par) |
| **TC** – Contador de transações | 1 bit + 5 caracteres hexadecimais (21 bits) | Conta o número de encriptações de PIN desde a inicialização do terminal. |

Os primeiros 3 elementos na tabela acima formam a chave inicial de número de série e não mudam durante a vida do terminal (a não ser que uma nova IKEY seja carregada por qualquer razão).

Muitas vezes apenas 64 bits do KSN são utilizados, com o KSN preenchida por caracteres “F” hexadecimais à esquerda. Neste esquema o KSN teria a seguinte estrutura:

1. Preenchimento – 4 caracteres “F” hex, 16 bits
2. KSI – 6 caracteres Hex, 24 bits. Isto permite em torno de 16 milhões de BDKs diferentes.
3. DID – 5 caracteres Hex, 20 bits. Isto inclui um bit do contador de transações, restando 19 bits para o real identificador de dispositivos. Isto significa que em torno de meio milhão de dispositivos diferentes podem ser gerenciados pelo identificador de dispositivos.
4. TC – 5 caracteres Hex, 20 bits (mais o bit incluído no identificador de dispositivos). O contador de transações é fornecido pelo terminal para identificar a chave de uma transação em particular. É usado pelo HSM para computar a chave em questão. Isto permite em torno de 1 milhão de transações antes que uma nova IKEY seja necessária – um limite que é improvável que seja alcançado.

O terminal não pode aceitar um KSN maior que 20 caracteres, portando o Host deve garantir que o tamanho total dos três primeiros campos não excedam 15 caracteres.

O Host também deve fornecer ao HSM o *KSN Descriptor*, um campo que define o tamanho (em caracteres) de cada um dos 3 campos. Deve ser armazenado junto ao KSN e é usado para identificar a BDK. O *KSN Descriptor* consiste em:

* Caractere esquerdo: tamanho do KSI.
* Caractere central: tamanho do identificador da sub-chave (sempre 0).
* Caractere direito: tamanho do DID.

## Transaction Keys – Chaves de Transação

Quando a IKEY é instalada no terminal, ela calcula até 21 “futuras chaves”. Estas chaves de transação são as chaves que serão usadas na encriptação de futuras transações. O cálculo destas chaves envolve o valor do contador de transações, o qual incrementa a cada transação.

Quando uma quantidade inicial de chaves futuras é derivada, a IKEY não é mais necessária e é deletada do terminal.

Quando uma transação é processada, a próxima chave de transação é usada. As chaves usadas para encriptação de PIN Block, MACing e encriptação de dados são derivadas dessa chave de transação.

O KSN é também modificado pelo incremento do contador de transações.

O terminal DUKPT envia seus dados encriptados e o KSN, junto com outros dados de transação para o adquirente.

A cada chave de transação usada, esta é deletada do terminal e substituída pela chave de transação futura. Isto significa que mesmo que a segurança do terminal seja comprometida de alguma forma e suas chaves extraídas, elas não podem ser usadas para atacar uma transação anterior deste ou de outro terminal pois a chave para esta transação já foi deletada e cada terminal gera uma chave diferente.

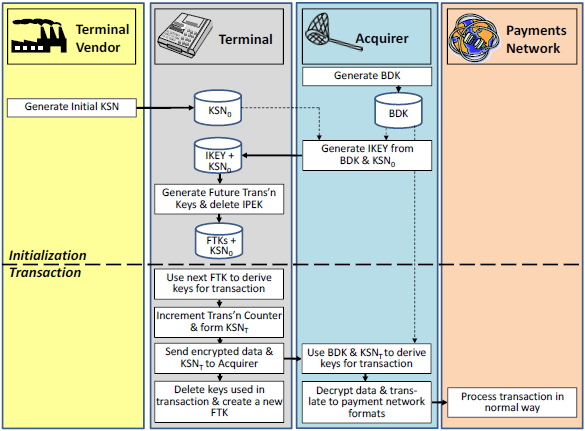
# Processamento pelo Adquirente

Quando o dado encriptado é mandado ao terminal pelo adquirente, o KSN (incluindo o contador de transações) é também mandado. O adquirente pode reconstruir a Chave de Transação usada pelo terminal através do KSN e da BDK apropriada (como identificada no KSI do KSN)

O adquirente precisa reembalar o dado recebido do terminal nos formatos padrão usados pelas redes de pagamento. Isto incluirá ações tais como:

* Traduzir um PIN Block encriptado DUKPT com a chave de transação DUKPT em um dos formatos de PIN Block padrão que são usados pelas redes de pagamento e encriptado com uma ZPK (Zone PIN Key – Chave PIN de Zona).
* Verificar e traduzir MACs.

# Resumo das Operações de DUKPT



Notas:

1. KSN0 = KSN inicial (com a contadora de transações = 0). Pode ser modificada pelo adquirente antes de gerar a IKEY.
2. KSNT = O KSN da transação, com o contador de transações incrementado.
3. A BDK mantida pelo adquirente será protegida usando um HSM.
4. As operações do adquirente mostradas aqui envolverão o uso de um HSM para várias funções de criptografia.

# Domínios Criptográficos

O *PCI PIN Security Requirements v2.0* publicado em dezembro de 2014, introduziu um novo requerimento para aumentar a segurança de chaves criptográficas. As chaves simétricas encriptadas devem ser gerenciadas em estruturas chamadas Key Block. O uso da chave deve ser criptograficamente vinculado à chave usando métodos aceitáveis.

Este requerimento torna necessária a criação de um segundo ambiente criptográfico para realizar a transição entre o método Variant e o método Key Block. Desta maneira, abordaremos a utilização do HSM payShield para criação e utilização do padrão DUKPT sob os dois métodos.

## Variant

### Criação da BDK

A BDK pode ser criada com ou sem a criação prévia de componentes. Para melhor endereçar o projeto da Fidelity, adotaremos a criação de uma BDK com componentes criados para esta simulação.

Componente 1:

|  |
| --- |
| Online-AUTH>gc    Enter LMK id [0-1]: 0  Enter key length [1,2,3]: 2  Enter key type: 009  Enter key scheme: U    Clear component: **0175 89A4 AE15 FE3E D98A 4904 893D 4051**  Encrypted component: U4DB3 C375 98DB FACE 29FB 0641 3FCA B4D5  Key check value: BB99B7 |

Componente 2:

|  |
| --- |
| Online-AUTH>gc    Enter LMK id [0-1]: 0  Enter key length [1,2,3]: 2  Enter key type: 009  Enter key scheme: U    Clear component: **7CAE 314C A889 C4F1 614C 86B0 4A37 8946**  Encrypted component: U8C59 7819 9043 211C 2808 F5B7 1DC6 16ED  Key check value: 3B27E9 |

Componente 3:

|  |
| --- |
| Online-AUTH>gc    Enter LMK id [0-1]: 0  Enter key length [1,2,3]: 2  Enter key type: 009  Enter key scheme: U    Clear component: **C851 83F1 20C2 25EF F8C7 F46D B3B3 B064**  Encrypted component: UA6CD FB4E 6448 1E2E CEA3 58A1 5E55 AC63  Key check value: 4B7EE4 |

Com os componentes formados, procede-se com a formação da chave BDK:

|  |
| --- |
| Online-AUTH>FK    Enter LMK id [0-1]: 0  Enter key length [1,2,3]: 2  Enter key type: 009  Enter key scheme: U  Enter component type [X,H,T,E,S]: X  Enter number of components [1-9]: 3    Enter component 1: 0175 89A4 AE15 FE3E D98A 4904 893D 4051  Component 1 check value: BB99B7  Continue? [Y/N]: y    Enter component 2: 7CAE 314C A889 C4F1 614C 86B0 4A37 8946  Component 2 check value: 3B27E9  Continue? [Y/N]: y    Enter component 3: C851 83F1 20C2 25EF F8C7 F46D B3B3 B064  Component 3 check value: 4B7EE4  Continue? [Y/N]: y      Encrypted key: **U4EC4 3434 714F B65D 1DCE EE00 2BA9 0032**  Key check value: **741FD1** |

### Simulação da criação do PIN Block no Terminal

Com a chave BDK formada já é possível realizar a tradução de um PIN Block do método DUKPT para qualquer outra chave, inclusive outra BDK. Neste item simularemos um POS realizando o encapsulamento do PIN Block com o método DUKPT para criarmos os parâmetros necessários na execução do comando de host para translate.

Como simulador, usaremos o programa *BP-Tools - Cryptographic Calculator* versão 21.01.

BDK em claro do item anterior:

|  |
| --- |
| [2021-02-07 13:22:38]  Thales Keys: Lookup finished - Exact match found  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Input Key: U4EC43434714FB65D1DCEEE002BA90032  Input KCV: 741FD1  Input Parity: Odd  ----------------------------------------  LMK pair [Variant]: Plain key KCV Parity LMK  28-29 [0]: **B58A3B19265E1F2040013BD970B97973** 741FD1 Odd DOUBLE  ----------------------------------------  Key type: 009 - BDK-1 - DUKPT Base Derivation Key (type 1) |

Formação do PIN Block em claro:

|  |
| --- |
| [2021-02-07 13:29:26]  PIN blocks: PIN block encrypt operation finished  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  PAN: 1111222233334444  PIN: 1234  PAD: N/A  Format: Format 0 (ISO-0)  ----------------------------------------  Clear PIN block: **041226DDDCCCCBBB** |

KSN:

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Tamanho** | **Valor** |
| **KSI** | 5 a 9 caracteres hexadecimais (20 a 36 bits) | **AAAAAAAAA** |
| Identificador da sub-chave | 1 caractere Hexadecimal (4 bits) | **0** |
| **DID** | 2 a 5 caracteres hexadecimais (8 a 20 bits) | **CCCCC** |
| **TC** | 1 bit + 5 caracteres hexadecimais (21 bits) | **0000D** |
| **AAAAAAAAA0CCCCC0000D** | | |

Derivação da BDK:

|  |
| --- |
| [2021-02-07 13:43:32]  DUKPT: PEK derivation finished  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  BDK: B58A3B19265E1F2040013BD970B97973  KSN: AAAAAAAAA0CCCCC0000D  ----------------------------------------  Derived IPEK: 53E9413273A129BF10B70A65BECEBDCD  KCV: 6C3AA7  Derived PEK: 444A9E756214F4EAF32DFBCA4CDB858B  KCV: 30A2BE |

Na derivação o BP-Tools exibe a IPEK (IKEY) e a PEK, chave derivada de acordo com o **TC** definido no KSN.

Com a PEK encapsulamos o PIN Block em claro e obtemos o PIN Block encriptado que será traduzido posteriormente:

|  |
| --- |
| [2021-02-07 13:46:28]  DUKPT: PIN operation finished  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  PEK: 444A9E756214F4EAF32DFBCA4CDB858B  PIN block: 041226DDDCCCCBBB  ----------------------------------------  Encrypted PIN: **9D254DD76AE5649D** |

### Tradução do PIN Block DUKPT para ZPK

Esta tradução utiliza o comando de host **G0** do HSM payShield. Traduziremos para uma ZPK formada abaixo:

|  |
| --- |
| Online-AUTH>KG    Enter LMK id [0-1]: 0  Enter key length [1,2,3]: 2  Enter key type: 001  Enter key scheme (LMK): U  Enter key scheme (ZMK):  Enter ZMK:    Key under LMK: **UC7BB 0907 1DF3 B383 CD84 5776 F8E2 C4D2**  Key check value: **6615A0** |

Comando **G0** detalhado:

|  |
| --- |
| Message Header 0000  Command Code G0  Source Key U4EC43434714FB65D1DCEEE002BA90032  Destination Key UC7BB09071DF3B383CD845776F8E2C4D2  Source KSN Descriptor A05  Source Key Serial Number AAAAAAAAA0CCCCC0000D  Source PIN Block 9D254DD76AE5649D  Source PIN Block Format Code 01  Destination PIN Block Format Code 01  PAN 122223333444  Delimiter %  LMK Identifier 00 |

Comando de host **G0** concatenado:

|  |
| --- |
| 0000G0U4EC43434714FB65D1DCEEE002BA90032UC7BB09071DF3B383CD845776F8E2C4D2A05AAAAAAAAA0CCCCC0000D9D254DD76AE5649D0101122223333444%00 |

Resposta do comando **G0**, **G1**:

|  |
| --- |
| 0000G1000436633E836EE4470101 |

|  |
| --- |
| Message Header 0000  Response Code G1  Error Code 00  PIN Length 04  PIN Block **36633E836EE44701**  PIN Block Format Code 01 |

Em destaque acima, o PIN Block criptografado pela chave destino, **ZPK**.

## Key Block

Usaremos os mesmos componentes do item Variant para criar a mesma BDK, mas agora sob a LMK Key Block AES de testes da Thales.

### Criação da BDK

|  |
| --- |
| Online-AUTH>fk    Enter LMK id [0-1]: 1  Enter algorithm type [D=DES, A=AES]: D  Enter key length [2,3]: 2  Enter key scheme: S  Enter component type [X,E,S]: X  Enter number of components [1-9]: 3  Enter Key Usage: B0  Enter Mode of Use: X  Enter Key Version Number: 00  Enter Exportability: S  Enter Optional Blocks? [Y/N]: N    Enter component 1: 0175 89A4 AE15 FE3E D98A 4904 893D 4051  Component 1 check value: BB99B7  Continue? [Y/N]: y    Enter component 2: 7CAE 314C A889 C4F1 614C 86B0 4A37 8946  Component 2 check value: 3B27E9  Continue? [Y/N]: y    Enter component 3: C851 83F1 20C2 25EF F8C7 F46D B3B3 B064  Component 3 check value: 4B7EE4  Continue? [Y/N]: y      Encrypted key: **S10096B0TX00S00014248EC21C04266A3996D5AFE71645D59D652E751373D665B5B389CA0E0FEE235A35742BD563ED702**  Key check value: **741FD1** |

Pode-se notar que o KCV da BDK Key Block é o mesmo KCV da BDK Variant, isto nos garante que se trata da mesma chave, porém protegida por LMKs e métodos distintos.

### Simulação da criação do PIN Block no Terminal

A criação do PIN Block será idêntica ao item 6.1.2 pois se trata da mesma chave e mesmos parâmetros de entrada como por exemplo o KSN.

### Tradução do PIN Block DUKPT para ZPK

Utiliza-se o mesmo comando de host **G0** e mesma ZPK, entretanto precisaremos da ZPK sob a LMK Key Block AES de testes da Thales:

|  |
| --- |
| Online-AUTH>fk    Enter LMK id [0-1]: 1  Enter algorithm type [D=DES, A=AES]: D  Enter key length [2,3]: 2  Enter key scheme: S  Enter component type [X,E,S]: X  Enter number of components [1-9]: 1  Enter Key Usage: P0  Enter Mode of Use: B  Enter Key Version Number: 00  Enter Exportability: S  Enter Optional Blocks? [Y/N]: N    Enter component 1: C22C 1346 2091 BA6D CB7F 542A 85A4 BFDC  Component 1 check value: 6615A0  Continue? [Y/N]: y      Encrypted key: **S10096P0TB00S000170C8DCA496E52148D2AB26C808FEB73DF5BE1EC785DDAD0FD12DF06F4B9E7A2C478A6A79BFAC740C**  Key check value: **6615A0** |

O KCV novamente nos indica que se trata da mesma ZPK do item 6.1.3.

Comando **G0** detalhado:

|  |
| --- |
| Message Header 0000  Command Code G0  Source Key S10096B0TX00S00014248EC21C04266A3996D5AFE71645D59D652E751373D665B5B389CA0E0FEE235A35742BD563ED702  Destination Key S10096P0TB00S000170C8DCA496E52148D2AB26C808FEB73DF5BE1EC785DDAD0FD12DF06F4B9E7A2C478A6A79BFAC740C  Source KSN Descriptor A05  Source Key Serial Number AAAAAAAAA0CCCCC0000D  Source PIN Block 9D254DD76AE5649D  Source PIN Block Format Code 01  Destination PIN Block Format Code 01  PAN 122223333444  Delimiter %  LMK Identifier 01 |

Pode-se notar que as diferenças entre ambos os detalhamentos dos comandos **G0**, Variant e Key Block, são as chaves. Todos os outros parâmetros permanecem os mesmos.

Comando de host **G0** concatenado:

|  |
| --- |
| 0000G0S10096B0TX00S00014248EC21C04266A3996D5AFE71645D59D652E751373D665B5B389CA0E0FEE235A35742BD563ED702S10096P0TB00S000170C8DCA496E52148D2AB26C808FEB73DF5BE1EC785DDAD0FD12DF06F4B9E7A2C478A6A79BFAC740CA05AAAAAAAAA0CCCCC0000D9D254DD76AE5649D0101122223333444%01 |

Resposta do comando de host **G0**, **G1**:

|  |
| --- |
| 0000G1000436633E836EE4470101 |

|  |
| --- |
| Message Header 0000  Response Code G1  Error Code 00  PIN Length 04  PIN Block **36633E836EE44701**  PIN Block Format Code 01 |

A resposta acima deixa ainda mais claro que se trata das mesmas chaves, os dois detalhamentos da resposta **G1** são idênticos.

## Verificação do PIN Block traduzido

Ambas as respostas do comando de host **G0** nos trouxeram o mesmo PIN Block encriptado pela ZPK, **36633E836EE44701**. Verificaremos se o conteúdo do PIN Block corresponde aos dados iniciais definidos no item 6.1.2.

Decriptando o PIN Block com a chave ZPK:

|  |
| --- |
| [2021-02-07 14:49:30]  DES/3DES operation finished  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Key: C22C13462091BA6DCB7F542A85A4BFDC  Algorithm: 3DES ECB  Crypto operation: Decoding  Data: 36633E836EE44701  Padding Method: None  ----------------------------------------  Decoded data: **041226DDDCCCCBBB**  DES operations count: 3 |

Decodificando o PIN Block em claro obtido no item acima:

|  |
| --- |
| [2021-02-07 14:51:03]  PIN blocks: PIN block decode operation finished  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  PIN block: 041226DDDCCCCBBB  PAN: 1111222233334444  PAD: N/A  Format: Format 0 (ISO-0)  ----------------------------------------  Decoded PIN: **1234** |

O PIN obtido acima é o mesmo PIN definido, portanto os processos de tradução tanto do Variant quanto do Key Block, foram executados com sucesso.