CRANE PAYMENT SOLUTIONS

NRI CashCode Telequip Money Controls

DES Encryption for Coin Acceptors and Bill Validators

ccTalk Protocol

Issue 2.2

2nd January 2013

Revision History

Issue	Date	Comments
1.0	06-03-09	First draft.
1.1	16-07-09	New options for [command level] in 'Request encryption support' command
1.2	18-03-10	Addition of new section : Implementation Rules
2.0	05-10-10	General Release
2.1	28-01-11	Option to remove support for unencrypted commands to force security. Added paragraph on the 'responsibility of the gaming machine manufacturer' ccTalk revision level may be 4.6 or greater
2.2	02-01-13	Comment added to Overview section about legacy gaming machine support Changed to Crane logo

1 Contents

2 Permissions

The security for peripheral DES encryption lies in a public domain algorithm with the 'key' being the secret rather than the algorithm. For this reason this document is not as sensitive as those published for the ccTalk protocol layer encryption and not subject to the same restrictions. However, it is good practice to limit the number of copies circulated and to make sure that only the people who need to see it do so.

3 Overview

The most sensitive message packet on coin acceptors and bill validators is the poll of buffered coin or bill events. It is that packet which conveys monetary value from the peripheral to the game machine. The most obvious fraud is to inject spurious credit information onto the bus or to change the contents of the message packet to show fictitious high value items. This fraud is not easy because the machine will be continuously polling the peripheral and unless the timing is perfect there will be communication and checksum errors. Also there is an event counter in the message packet which must be kept synchronised with the host. Bill validators until now have been protected with the ccTalk protocol level encryption which uses a proprietary Money Controls' algorithm to obfuscate message data with a short key which rotates every few seconds. This document proposes

the use of DES encryption on certain ccTalk commands in coin acceptors and bill validators to protect message data.

Each peripheral will be manufactured with a unique DES key. Unless the DES key is known then no credit validation will be possible. This DES key can be used for the lifetime of the peripheral taking into account that if it is cracked on one peripheral it does not compromise another peripheral. The time to crack a single DES key is estimated at just less than a day given vast processing capability on thousands of CPUs. In case this becomes economically attractive in future, a command has been provided that allows the game machine to rotate the DES key every e.g. 24 hours. This means a fraudster returning to a game machine the following day with a cracked key will find that it doesn't work; the key has moved on !

The single DES algorithm uses a 64-bit key (only 56 bits are actually used, the rest are parity) giving over 10,000,000,000,000,000 possibilities. It would take too long to manually enter a DES key into a game machine using simple up / down / left / right keys. Consideration also has to be given to servicing and spares which require a quick method of replacing a faulty peripheral. Therefore we have introduced the concept of a 'trusted key exchange mode'. In this mode it will be possible for the game machine to request the security keys of the peripheral. Once obtained, the keys need not be transferred again until the peripheral is replaced. How trusted key exchange mode is entered into and the security aspects of that are discussed later.

The original proposal was to poll with non-DES commands and then only use the DES commands when the event counter changed. This approach reduced the processing overhead in legacy gaming machines where a multi-block decryption every 200ms could affect performance. However, it would mean that DES peripherals would function correctly in non-DES machines and this could lead to security vulnerabilities which would be hard to track down without a detailed knowledge of the machine architecture. For this reason there will be a peripheral manufacturers' option to remove the corresponding non-DES commands from peripherals that support DES. Whether this has been done should be made clear in the product manual. **Please note, removal of non-DES commands will prevent the peripheral operating in legacy gaming machines.**

On coin acceptors :

Remove header 229, 'Read buffered credit or error codes'
Replace with header 112, 'Read encrypted events'

Remove header 184, 'Request coin id'
Replace with header 108, 'Request encrypted monetary id'

On bill validators :

Remove header 159, 'Read buffered bill events'
Replace with header 112, 'Read encrypted events'

Remove header 157, 'Request bill id'
Replace with header 108, 'Request encrypted monetary id'

For some additional security it is possible to combine the DES encryption algorithm with the ccTalk protocol layer encryption which uses a 6-digit key to encrypt each ccTalk command packet. This 6-digit key can also be rotated by the game machine every few seconds to guard against replay attacks. Protocol encryption has previously been used on bank note validators (BNVs) and is often referred to as ***BNV encryption***.

Peripherals that support DES encryption will return a ccTalk comms revision (header 004) of at least 4.6 and will have implemented the additional commands described in this document.

4 Event Polling using DES

The same command header is used to read the encrypted event buffer on coin acceptors and bill validators as the data is in the same format.

There are 2 event polling options, the 2nd of which reduces the DES overhead in legacy machines. Which option is implemented should be made clear in the product manual.

4.1 Option 1 – Legacy commands not supported

On coin acceptors :

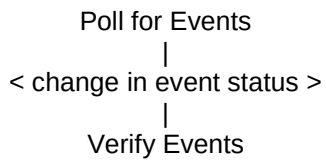
Header 112, Read encrypted events

On bill validators :

Header 112, Read encrypted events

4.2 Option 2 – Legacy commands supported

The ccTalk command sequence is to poll the peripheral until a change in the event counter is seen and then confirm any events with a DES-encrypted command.



The actual ccTalk commands that do this are…

On coin acceptors :

Header 229, Read buffered credit or error codes
|
< change in event status >
|
Header 112, Read encrypted events

On bill validators :

Header 159, Read buffered bill events
|
< change in event status >
|
Header 112, Read encrypted events

5 Encrypted Events

Header 112 : Read encrypted events

```
Transmitted data : [ challenge 1 ]
Received data :    [ CRC checksum LSB ] [ random 1 ] [ event counter ]
                   [ result 1A ] [ result 1B ]
                   [ result 2A ] [ result 2B ]
                   [ challenge 1 ]

                   [ random 2 ]
                   [ result 3A ] [ result 3B ]
                   [ result 4A ] [ result 4B ]
                   [ result 5A ] [ result 5B ]
                   [ CRC checksum MSB ]
```

For security, 1 challenge byte is sent out by the game machine as plaintext. These is included in the reply from the peripheral along with 2 random or undefined bytes. The peripheral adds a CRC-16 checksum to the data using the same algorithm as the ccTalk packet structure when protocol level encryption is enabled. The CRC is calculated for the 14 bytes in between. The reply blocks are then DES encrypted by the peripheral.

In the examples which follow, the entire ccTalk command packet is shown. The peripheral is a bill validator on address 40 (for a coin acceptor the address would typically be 2 and the checksums would be different). The byte values between square brackets are shown in hexadecimal.

The examples shown do not use protocol level encryption so we use source addresses and 8-bit checksums rather than 16-bit CRC checksums.

TX is data from the game machine to the peripheral.
RX is data from the peripheral to the game machine.

```
Read encrypted events
TX :  [ 28 ] [ 01 ] [ 01 ] [ 70 ] [ 45 ] [ 21 ]
RX :  [ 01 ] [ 10 ] [ 28 ] [ 00 ]
      [ 14 ] [ 41 ] [ 4E ] [ 6B ] [ 79 ] [ CD ] [ 7D ] [ 6B ]
      [ 01 ] [ 74 ] [ 3F ] [ 25 ] [ 7F ] [ F5 ] [ 90 ] [ B3 ]
      [ FB ]
```

TX Decode :
Destination address = 40
No. of data bytes = 1
Source address = 1
Command 0x70 = 112 decimal = Read encrypted events
Challenge data = 0x45
8-bit checksum = 0x21

RX Decode :
Destination address = 1
No. of data bytes = 16
Source address = 40
Packet 1 = **0x14414E6B79CD7D6B**
Packet 2 = **0x01743F257FF590B3**
8-bit checksum = 0xFB

Assume the DES key in our example is the 64-bit number 0xFEDCBA9876543210

The DES algorithm takes an array of 8 bytes which represent the key. The first byte in our example is the LSB byte or [10], followed by [32]… up to [FE].

Running this through the industry-standard DES decryption algorithm…

Block 1

Encrypted	[14]	[41]	[4E]	[6B]	[79]	[CD]	[7D]	[6B]
DES key	[10]	[32]	[54]	[76]	[98]	[BA]	[DC]	[FE]
Decrypted	[5C]	[10]	[09]	[00]	[00]	[04]	[00]	[45]

Block 2

Encrypted	[01]	[74]	[3F]	[25]	[7F]	[F5]	[90]	[B3]
DES key	[10]	[32]	[54]	[76]	[98]	[BA]	[DC]	[FE]
Decrypted	[34]	[04]	[01]	[00]	[01]	[00]	[02]	[B8]

Red : CRC bytes
Green : Challenge bytes
Blue : Random / Undefined bytes

Event counter = 9
Event 1 = 0x00 0x00 = Master inhibit active
Event 2 = 0x04 0x00 = Bill type 4 sent to cashbox
Event 3 = 0x04 0x01 = Bill type 4 held in escrow
Event 4 = 0x00 0x01 = Bill returned from escrow
Event 5 = 0x00 0x02 = Invalid bill (due to validation fail)
CRC-16 checksum = 0xB85C

When the game machine receives a reply to this command it should…

- Verify the ccTalk packet address and packet checksum as usual
- Decrypt both blocks with the current DES key
- Verify the CRC-16 checksum
- Verify the challenge byte
- Handle the new events in the event buffer

5.1 DES Technical Information

5.1.1 Parity

Although the DES key is 64 bits long, and passed as an array of 8 bytes, only 56 bits of this are actually used in the algorithm. 8 bits of the array are allocated to parity bits. The peripheral ignores these parity bits and they can be set to anything. The parity bit here is bit 0 of each array byte.

The following keys are therefore identical i.e. produce the same result when run through the DES encryption algorithm.

0xFEDCBA9876543210
0xFFDDBB9977553311

In other words, each byte of the DES key array may be odd or even without consequence.

Note that when changing key with the ‘Switch encryption key’ command, the old key must match exactly i.e. parity bits are not ignored in the comparison.

5.1.2 IV

DES algorithms often refer to the IV or initialisation vector. This is only used when encrypting multiple blocks in one operation and block chaining is used. For coin acceptors and bill validators we are only encrypting one or two blocks and are working in ECB mode (Electronic Code Book). For these peripherals there is no IV so we do not define one. If you are unsure how it is being used then set it to all zeros (an array of 8 null bytes).

5.1.3 Symmetry

DES is a symmetrical encryption algorithm so that the key to encrypt a data block is the same key that decrypts the data block.

5.1.4 Weak Keys

It is important that when the game machine changes the DES key in the peripheral, weak keys are avoided as they are easier to crack.

DES has 4 documented weak keys and 6 semi-weak key pairs.

Weak keys :
0101010101010101
FEFEFEFEFEFEFEFE
E0E0E0E0F1F1F1F1
1F1F1F1F0E0E0E0E

Most DES key generator libraries will automatically avoid all weak and semi-weak keys.

5.2 Microsoft API

Cryptographic services are built into many high-level programming languages such as Microsoft Visual Basic.Net. Here is an example of encrypting data with DES.

```
Dim keyDES As New DESCryptoServiceProvider()
keyDES.Key = key64
keyDES.Mode = CipherMode.ECB
keyDES.Padding = PaddingMode.Zeros
Dim ms As MemoryStream = New MemoryStream
Dim xStream As CryptoStream = New CryptoStream(ms, keyDES.CreateEncryptor(),
CryptoStreamMode.Write)
Dim sw As New BinaryWriter(xStream)
sw.Write(dataBlock64)
sw.Close()
xStream.Close()
Dim encryptedData() As Byte = ms.ToArray()
ms.Close()
keyDES.Clear()
Return encryptedData
```

5.3 Further Information

An excellent primer on DES encryption may be found on Wikipedia.
http://en.wikipedia.org/wiki/Data_Encryption_Standard
http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation
http://en.wikipedia.org/wiki/Weak_key

The DES algorithm itself cannot be obtained from Money Controls but it has been widely published and is available in many programming languages and in libraries optimised for different processors.

Most microcontrollers today should be able to encrypt a DES block in well under 10ms.

6 Encrypted Identifiers

Header 108 : Request encrypted monetary id

```
Transmitted data : [ position ] [ challenge 1 ]
Received data :    [ CRC checksum LSB ] [ position ]
                   [ C1 ] [ C2 ] [ C3 ] [ SF ] [ DP ]
                   [ challenge 1 ]

                   [ random 1 ]
                   [ V1 ] [ V2 ] [ V3 ] [ V4 ]
                   [ IL ] [ IN ]
                   [ CRC checksum MSB ]
```

The ‘Read encrypted events’ command transfers monetary value using a coin or bill position index. The conversion of that position into an actual monetary value requires a look-up table. Rather than hard-coding that table, it is preferable for the game machine to interrogate the peripheral to find out the monetary item for each position 1 to 16 or 1 to 64 etc. as it is possible the ordering or content could change from one release of a coin or bill specification to another. Protecting this information through DES encryption is just as important as protecting the event buffer as a fraudster could possibly intercept and manipulate the monetary values.

The 2 blocks of return data are DES-encrypted by the peripheral. The host machine decrypts them with the current DES key and confirms the validity of the position byte, challenge byte and CRC checksum.

This command can be used on both coin acceptors and bill validators. It replaces header 184, ‘Request coin id’, header 157, ‘Request bill id’ and header 156, ‘Request country scaling factor’.

Note that each monetary item now has a separate scaling factor although for the vast majority of currencies they will be the same. Up to now coins have not used scaling factors as they are always in base currency units. So for coins SF=0 (**`10^0 = 1`**) and DP=0. However, game machine software should be aware of scaling factors and use them to calculate monetary value for each item separately. Coins and bills also use a 4 digit value giving a range 1000:1; previously coins used a 3 digit value field rather than 4.

[challenge 1]
1 byte of random challenge data is generated by the host. This helps prevent replay attacks where the DES key is still secret but responses are stored and injected onto the bus later.

[random 1]
1 byte of random data generated by the peripheral to mix up the plaintext.

[CRC]
A 16-bit CRC checksum is calculated for the received data. This is the same checksum algorithm as the protocol layer and provides a quick and easy hashing algorithm to verify the decryption.

The checksum is calculated on 14 bytes from [position] to [IN].

[position]
This is the position or index of the coin or bill within the currency specification. It always start from 1 and is typically in the range 1 to 16 or 1 to 64.

[C1] [C2] [C3]
This is the country or currency code in ASCII.

2 formats are supported…
ISO 3166-1-A2 country code and ISO 4217 currency code (not ISO 3166-1-A3).

ISO 3166 requires 2 letters, ISO 4217 requires 3 letters.

If ISO 3166 is being used then the first character [C1] is '#'.

ISO 3166 is still the default and recommended coding system for all ccTalk peripherals.

[SF]
Scaling Factor.
This is a number between 0 and 255.

This is the scaling factor for this monetary item in powers of 10, relative to the base or minor currency unit e.g. cents or pence.

The common values are 0, 1, 2, 3 and 4.
0 = value x 1
1 = value x 10
2 = value x 100
3 = value x 1000
4 = value x 10,000

[DP]
Decimal Places.
This is a number between 0 and 255.

The common values are 0 and 2.

A value of 100 with no decimal places would be displayed as 100 and with 2 decimal places as 1.00

The DP field is used primarily for the machine display of bill insertion values.

[V1] [V2] [V3] [V4]
Value in ASCII

e.g. 0000 to 9999

By ccTalk convention, coin acceptors report values in 3 digits and bill validators in 4 digits. However, using this encrypted command all values are reported in 4 digits with a leading zero digit if necessary.

[IL]
Issue Level in ASCII

This is a letter between ‘A’ and ‘Z’.

[IN]
Issue Number in ASCII

This is a number between ‘1’ and ‘9’.

The standard issue number progression is A1 to Z1 and then A2 to Z2 etc.

If there is no programmed bill at the specified position then a blank field designator will be returned.

```
CCC  = ... ( ASCII code 046 decimal )
VVVV = ....
IL   = .
IN   = .
SF   = 0
DP   = 0
```

Here’s a bill validator example…

```
Request encrypted monetary id
TX :  [ 28 ] [ 02 ] [ 01 ] [ 6C ] [ 04 ] [ 35 ] [ 30 ]
RX :  [ 01 ] [ 10 ] [ 28 ] [ 00 ]
      [ E7 ] [ 38 ] [ A1 ] [ AC ] [ 6A ] [ 0A ] [ 49 ] [ EC ]
      [ 1A ] [ 2F ] [ B7 ] [ 91 ] [ CE ] [ 80 ] [ 5D ] [ CA ]
      [ AC ]
```

TX Decode :
Destination address = 40
No. of data bytes = 2
Source address = 1
Command 0x6C = 108 decimal = Request encrypted monetary id
Position = 4
Challenge data = 0x35
8-bit checksum = 0x30

RX Decode :
Destination address = 1
No. of data bytes = 16
Source address = 40
Packet 1 = **0xE738A1AC6A0A49EC**
Packet 2 = **0x1A2FB791CE805DCA**
8-bit checksum = 0xAC

Assume the DES key in our example is the 64-bit number 0xFEDCBA9876543210

Running this through the industry-standard DES decryption algorithm…

Block 1

```
Encrypted  [ E7 ] [ 38 ] [ A1 ] [ AC ] [ 6A ] [ 0A ] [ 49 ] [ EC ]
DES key    [ 10 ] [ 32 ] [ 54 ] [ 76 ] [ 98 ] [ BA ] [ DC ] [ FE ]
Decrypted  [ B2 ] [ 04 ] [ 23 ] [ 47 ] [ 42 ] [ 02 ] [ 02 ] [ 35 ]
```

Block 2

```
Encrypted  [ 1A ] [ 2F ] [ B7 ] [ 91 ] [ CE ] [ 80 ] [ 5D ] [ CA ]
DES key    [ 10 ] [ 32 ] [ 54 ] [ 76 ] [ 98 ] [ BA ] [ DC ] [ FE ]
Decrypted  [ EA ] [ 30 ] [ 30 ] [ 30 ] [ 35 ] [ 45 ] [ 31 ] [ 26 ]
```

```
Red   : CRC bytes
Green : Position / Challenge bytes
Blue  : Random / Undefined bytes

C1 = 0x23 = '#'
C2 = 0x47 = 'G'
C3 = 0x42 = 'B'
SF = 0x02 =  2; 10^2 = 100 decimal
DP = 0x02 =  2 decimal
V1 = 0x30 = '0'
V2 = 0x30 = '0'
V3 = 0x30 = '0'
V4 = 0x35 = '5'
IL = 0x45 = 'E'
IN = 0x31 = '1'
```

CRC-16 checksum = 0x26B2

Therefore Bill 04 = #**GB0005E1** SF = 2 DP = 2

7 Trusted Key Exchange Mode

A new command has been introduced into issue 4.6 of the ccTalk generic specification. This command allows a game machine to identify the level of encryption that exists on a peripheral and in certain conditions actually read the encryption keys. This command works unencrypted regardless of whether BNV protocol encryption is in use; previously only encrypted commands could be sent to an encrypted peripheral preventing any self-configuration of the gaming board interface.

The new command is as follows…

Header 111 : Request encryption support

```
Transmitted data : [ AA ] [ 55 ] [ 00 ] [ 00 ] [ 55 ] [ AA ]
Received data :    [ protocol level ] [ command level ]
                   [ protocol key size ]
                   [ command key size ] [ command block size ]
                   [ trusted mode ]
                   [ BNV2 | BNV1 ] [ BNV4 | BNV3 ] [ BNV6 | BNV5 ]
                   [ DES1 ] [ DES2 ] [ DES3 ] [ DES4 ]
                   [ DES5 ] [ DES6 ] [ DES7 ] [ DES8 ]
```

The 6 data bytes transmitted (0xAA, 0x55, 0x00, 0x00, 0x55, 0xAA) are a validation signature for this command. They never change but must be transmitted as shown.

The receive data consists of the following information.

[protocol level]
0 – no encryption
1 – ccTalk Serial Protocol Encryption Standard 1.2

This refers to the protocol layer encryption and is nothing to do with the DES key on peripherals. The '1.2' refers to the latest implementation of this algorithm at the time of publication of this document. It has been the standard encryption algorithm on ccTalk bill validators for many years.

Documentation on the ccTalk protocol layer encryption can be obtained from Money Controls on request but it is currently strictly controlled and subject to NDA. Manufacturers of coin acceptors and bill validators do not have to support protocol layer encryption and can just use DES instead. The high security level obtained with DES alone should be perfectly adequate.

[command level]
0 – no encryption
11 – Serial Hopper Encryption Standard CMF1-1 (SCH2 L1 encryption)
12 – Serial Hopper Encryption Standard CMF1-2 (SCH3 L2 encryption)
13 – Serial Hopper Encryption Standard CMF1-3 (SCH3E L3 encryption)
21 – Serial Hopper Encryption Standard CMF2-1 (Combi encryption)
101 – DES Encryption
102 – AES Encryption (future support only)
103 – Triple DES Encryption (future support only)

At the command level on coin acceptors and bill validators, one of the options is DES encryption as put forward by this document (level = 101).

[protocol key size]

This is the size in bits of the protocol layer encryption key. The BNV algorithm uses a 6 decimal digit encryption key 000000 to 999999. It is encoded in BCD format in 3 bytes or 24 bits. But although we will return '24' by convention the actual key space is a lot less than this due to the decimal sub-group and a slight key reduction on 3 byte packets due to the way it has been implemented.

[command key size]

This is the size in bits of the command level encryption key. For single DES we will return '64'. The value of 0 will be reserved for 256 bits if required in future encryption algorithms.

[command block size]

Block sizes of 64 bit and 128 bits are common for symmetrical encryption algorithms. On these peripherals the block size is '64'.

[trusted mode]
0 – normal operating mode
255 – trusted key exchange mode

The game machine should treat any value other than 255 as normal operating mode. In trusted key exchange mode the protocol level BNV key and the DES encryption key will follow. ***In normal operating mode the keys will still be returned as data bytes but they will all be zero***.

This is typical packet exchange for a bill validator to show the format of the keys.

```
Request encryption support
TX :  [ 28 ] [ 06 ] [ 01 ] [ 6F ]
      [ AA ] [ 55 ] [ 00 ] [ 00 ] [ 55 ] [ AA ]
      [ 64 ]
RX :  [ 01 ] [ 11 ] [ 28 ] [ 00 ]
      [ 00 ] [ 65 ] [ 18 ] [ 40 ] [ 40 ] [ FF ]
      [ 00 ] [ 00 ] [ 00 ]
      [ 10 ] [ 32 ] [ 54 ] [ 76 ] [ 98 ] [ BA ] [ DC ] [ FE ]
      [ 92 ]
```

Note that this command is always sent unencrypted with 8-bit checksums, even if the peripheral uses protocol level encryption.

TX Decode :
Destination address = 40
No. of data bytes = 6
Source address = 1
Command 0x6F = 111 decimal = Request encryption support
8-bit checksum = 0x64

RX Decode :
Destination address = 1
No. of data bytes = 17
Source address = 40
Protocol level = 0x00 = no encryption
Command level = 0x65 = 101 decimal = DES Encryption
Protocol key size = 0x18 = 24 decimal (bits)
Command key size = 0x40 = 64 decimal (bits)
Command block size = 0x40 = 64 decimal (bits)
Trusted mode = 0xFF = 255 decimal = yes
BNV key = 0x000000
DES key = 0xFEDCBA9876543210
8-bit checksum = 0x92

Note that ccTalk message packets are little-endian – we return LSB bytes first and MSB bytes last. So the first DES key byte is returned first, in this case [10], and the last DES key byte last, in this case [FE]. The order it is written in or displayed is fairly arbitrary. The important point is that when you call the DES library, the first byte of the key array will be 0x10.

When not in trusted mode the following information is returned.

```
TX :  [ 28 ] [ 06 ] [ 01 ] [ 6F ]
      [ AA ] [ 55 ] [ 00 ] [ 00 ] [ 55 ] [ AA ]
      [ 64 ]
RX :  [ 01 ] [ 11 ] [ 28 ] [ 00 ]
      [ 00 ] [ 65 ] [ 18 ] [ 40 ] [ 40 ] [ 00 ]
      [ 00 ] [ 00 ] [ 00 ]
      [ 00 ] [ 00 ] [ 00 ] [ 00 ] [ 00 ] [ 00 ] [ 00 ] [ 00 ]
      [ C9 ]
```

The [00] after the [40] indicates this is normal operating mode, not trusted mode.

The BNV keys and DES keys are still returned (17 data bytes in the packet), but the values are now cleared to zero to maintain secrecy.

If protocol level encryption is being used in trusted mode then the return data may look like this…

```
TX :  [ 28 ] [ 06 ] [ 01 ] [ 6F ]
      [ AA ] [ 55 ] [ 00 ] [ 00 ] [ 55 ] [ AA ]
      [ 64 ]
RX :  [ 01 ] [ 11 ] [ 28 ] [ 00 ]
      [ 01 ] [ 65 ] [ 18 ] [ 40 ] [ 40 ] [ FF ]
      [ 21 ] [ 43 ] [ 65 ]
      [ 10 ] [ 32 ] [ 54 ] [ 76 ] [ 98 ] [ BA ] [ DC ] [ FE ]
      [ C8 ]
```

Protocol level = 0x01 = ccTalk Serial Protocol Encryption Standard 1.2

BNV key = 0x654321, usually written on labels as 123456

7.1 Entering Trusted Key Exchange Mode

For security it is essential that trusted key exchange mode can only be entered with direct physical access to the peripheral. If it is possible to switch to this mode using a serial command then there is the possibility that security could be defeated across the entire range of DES peripherals. Therefore trusted mode assumes the game cabinet has been opened up and physical contact can be made with the peripheral. This would occur anyway during installation of new peripherals and the servicing of old ones. Trusted key security then becomes the physical security of the game cabinet itself with regards to locks, hinges, steel covers and ventilation holes etc.

This document does not specify how trusted mode is entered; this can be left to the manufacturers of the peripheral concerned. However, it does warrant careful design consideration so as not to jeopardise the benefits of DES encryption on coin acceptors and bill validators.

Some examples of how this mode can be engaged are given below.

7.1.1 DIP Switch

A DIP switch on the product could be used to enter trusted mode. Special attention should be paid to prevent a wire being inserted through an access point in the game cabinet and pushed against the switch. Perhaps the DIP switch is underneath the peripheral or behind a protective cover.

The DIP switch should only be read at power-up or after a software reset. If the DIP switch is left in the wrong position then any attempt to accept coins or notes should be refused.

7.1.2 PCB Jumper

A 2-pin PCB jumper could be used to enter trusted mode. A jumper is normally left attached, shorting the 2 terminals.

If the jumper is missing at power-up or after a software reset then trusted mode is entered. Any attempt to accept coins or notes should be refused if the jumper is missing. The jumper should not be in an exposed position on the peripheral to prevent it being lifted off remotely through 'keyhole surgery'.

8 Switching DES Key

Header 110 : Switch encryption key

```
Transmitted data : [ old1 ] [ new1 ] [ old2 ] [ new2 ] [ old3 ] [ new3 ]
                   [ old4 ] [ new4 ]
                   [ old5 ] [ new5 ] [ old6 ] [ new6 ] [ old7 ] [ new7 ]
                   [ old8 ] [ new8 ]
Received data :    ACK
```

The old key is interleaved with the new key and sent to the peripheral. Before it is sent, both 8 byte blocks of transmit data are DES encrypted with the old key. The peripheral decrypts the data with the old key and if the old key matches then an immediate switch is made to the new key and an ACK returned. If the old key is incorrect then there is no reply. An ACK reply takes at least **100ms** to make guessing key values using this command totally impractical.

8.1 Key Verification

It is possible when a game starts to verify that the correct DES key is being used. This is useful after a peripheral is replaced as it may be some time before an event is generated.

The 'Switch encryption key' command is used and the new key is made equal to the old key. The peripheral does not store any new data; it just confirms that the old key is correct.

```
Transmitted data : [ old1 ] [ old1 ] [ old2 ] [ old2 ]
                   [ old3 ] [ old3 ] [ old4 ] [ old4 ]
                   [ old5 ] [ old5 ] [ old6 ] [ old6 ]
                   [ old7 ] [ old7 ] [ old8 ] [ old8 ]
Received data :    ACK
```

For example, this is the verification of our default 0xFEDCBA9876543210 key.

```
TX :  [ 28 ] [ 10 ] [ 01 ] [ 6E ]
      [ B0 ] [ 2D ] [ F1 ] [ 0E ] [ 88 ] [ 6C ] [ FC ] [ FF ]
      [ 9B ] [ EC ] [ 58 ] [ F1 ] [ 89 ] [ 6C ] [ A6 ] [ 9C ]
      [ 87 ]
RX :  [ 01 ] [ 00 ] [ 28 ] [ 00 ] [ D7 ] = ACK
```

9 Game Machine Initialisation

When the game machine starts-up, it will need to check each ccTalk peripheral on the expected address, taking into account that a service engineer could have replaced the peripheral with a new one and that would mean the DES key is no longer valid. It can do this with the 'Request encryption support' command.

If there is no reply to the 'Request encryption support' command then the possibilities are…

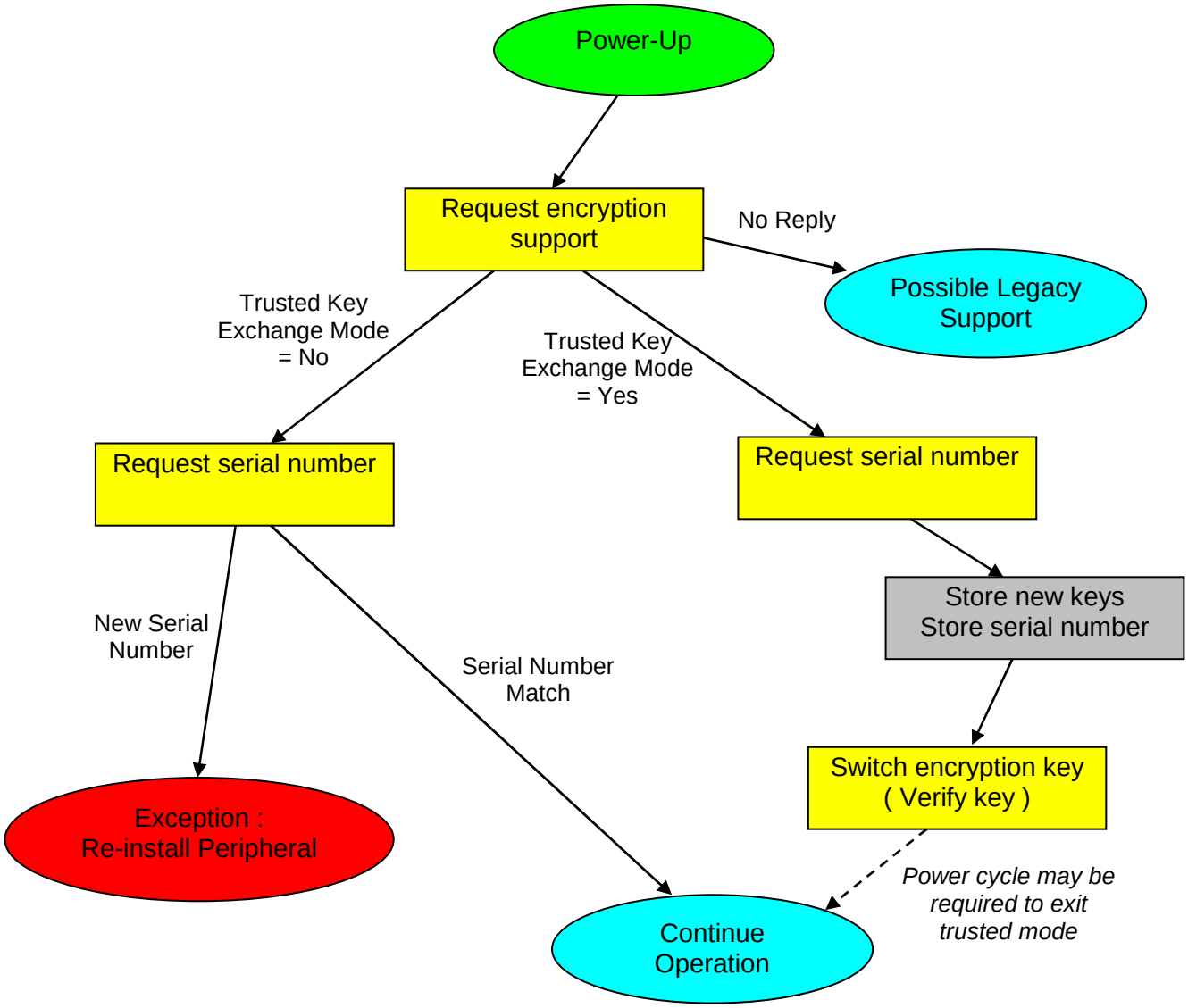
- The peripheral is missing or unpowered
- The peripheral is on a different address to that expected
- This is a 'legacy' peripheral with no support for command header 111

If a legacy peripheral then other initialisation procedures should be used. It is recommended that legacy peripherals are not allowed to operate in DES machines.

If the encryption support reply shows 'trusted key exchange mode' then new values of BNV and DES keys should be stored as appropriate. The key can then be verified with the 'Switch encryption key command' before it is actually required (which could be hours later).

The peripheral may exit trusted key exchange mode automatically on reading the keys or it may have to be power cycled / reset; this will vary according to the peripheral and the exact operating requirements should be explained in the product manual. In trusted mode the peripheral is non-functional for obvious security reasons.

If the encryption support reply does not show ‘trusted key exchange mode’ then it must be assumed the keys are the same as when the peripheral was powered off. In this case the serial number should be checked to confirm the peripheral has not changed. If it has then the peripheral must be re-installed in trusted mode and the game should throw an exception.



10 Combined Protocol Level and Command Level Encryption

It is possible to combine protocol level encryption using the BNV key and command level encryption using the DES key. The BNV key wraps the entire message packet and within that the DES key protects payload data. The combined encryption strength is not much better than DES but for some applications the masking of command headers and checksums may provide some additional complexity to simple bus attacks.

There are some complications to be aware of. For instance with BNV and DES encryption enabled, the peripheral address may have been changed and stored to a non-standard value. How can we discover the address ? If BNV encryption is enabled then no ccTalk commands can be transmitted without knowing the BNV key, including header 253, ‘Address poll’, which can be used to discover a peripheral address. To obtain the BNV key we need to send the ‘Request encryption support’ command in trusted mode. Fortunately we can send this command with the broadcast address and assuming no other peripherals are connected to the bus at this time the BNV key will be obtained. We can then use the ‘Address poll’ command to find the address or quickly poll the ccTalk address space (2 to 255) with header 254, ‘Simple poll’, until the peripheral replies with an ACK. The ‘Address change’ command on header 251 can then be used to put the address back.

11 Miscellaneous Security Issues

For peripherals using DES encryption, some of the security features available on older ccTalk products are redundant.

11.1 Header 219, 'Enter new PIN number'

PIN number protection only provides limited security against peripherals being used in unauthorised game machines. The transfer of the PIN number during machine start-up is also susceptible to bus snooping.

It is recommended that peripherals do not use this command.

11.2 Header 218, 'Enter PIN number'

It is recommended that peripherals do not use this command. See above.

11.3 Header 137, 'Switch encryption code'

If BNV encryption is being used alongside DES encryption then this command can be used to rotate the 6-digit key every couple of seconds.

The use of this command is optional.

11.4 Header 136 , 'Store encryption code'

If BNV encryption is being used alongside DES encryption then this command can be used to store the latest 6-digit key inside the peripheral so that it will become the new starting value at power-up.

If the BNV key is lost at any point then it can be recovered through trusted key exchange mode.

The use of this command is optional.

12 Example Communication Dump

```
Bill validator is on address 40

TX Header = Request encryption support
TX Packet = 28 06 01 6F AA 55 00 00 55 AA 64
RX Packet = 01 11 28 00 00 65 18 40 40 00 00 00 00 00 00 00 00 00 00 C9
Trusted key exchange mode is off
BNV Key not used

In the examples below,
DES Key = 0xB2F3D7FA4544C837

desKey[0] = 0x37
desKey[1] = 0xC8
desKey[2] = 0x44
desKey[3] = 0x45
desKey[4] = 0xFA
desKey[5] = 0xD7
desKey[6] = 0xF3
desKey[7] = 0xB2

TX Header = Request serial number
TX Packet = 28 00 01 F2 E5
RX Packet = 01 03 28 00 83 01 00 50
S/N = 0x000183 = 387

TX Header = Modify master inhibit status
TX Packet = 28 01 01 E4 01 F1
RX Packet = 01 00 28 00 D7
Clear master inhibit

TX Header = Modify inhibit status
TX Packet = 28 08 01 E7 FF FF FF FF FF FF FF FF F0
RX Packet = 01 00 28 00 D7
Enable all 64 bills

TX Header = Read buffered bill events
TX Packet = 28 00 01 9F 38
RX Packet = 01 0B 28 00 00 00 00 00 00 00 00 00 00 00 CC
No new events

TX Header = Read buffered bill events
TX Packet = 28 00 01 9F 38
RX Packet = 01 0B 28 00 01 04 01 00 00 00 00 00 00 00 C6
Event 1 = Bill type 4 held in escrow

TX Header = Read encrypted events
TX Packet = 28 01 01 70 1E 48
RX Packet = 01 10 28 00 A4 C0 C5 49 44 88 AA 88 0B 82 17 92 D4 D3 06 69 0B

Block 1 Decrypted = 2F DE 01 04 01 00 00 1E
Block 2 Decrypted = E8 00 00 00 00 00 00 E2
Verify event

TX Header = Route bill
TX Packet = 28 01 01 9A 01 3B
RX Packet = 01 00 28 00 D7
Route bill to cashbox
```

```
TX Header = Read buffered bill events
TX Packet = 28 00 01 9F 38
RX Packet = 01 0B 28 00 01 04 01 00 00 00 00 00 00 00 00 C6
No new events

TX Header = Read buffered bill events
TX Packet = 28 00 01 9F 38
RX Packet = 01 0B 28 00 02 04 00 04 01 00 00 00 00 00 00 C1
Event 2 = Bill type 4 sent to cashbox

TX Header = Read encrypted events
TX Packet = 28 01 01 70 2C 3A
RX Packet = 01 10 28 00 97 A8 97 57 2D 2C DC 20 85 72 13 65 01 C2 8A 10 79

Block 1 Decrypted = BD 8C 02 04 00 04 01 2C
Block 2 Decrypted = FA 00 00 00 00 00 00 36
Verify event

TX Header = Request encrypted monetary id
TX Packet = 28 02 01 6C 04 99 CC
RX Packet = 01 10 28 00 AB E4 9E BF FB BF 8D 08 4D 45 DB 81 1C 7A 03 FE 07

Block 1 Decrypted = BD 04 23 47 42 02 02 99
Block 2 Decrypted = BE 30 30 30 35 45 31 9A
Convert position to monetary value

CCC = 234742 = #GB
SF = 02
DP = 02
VVVV = 30303035 = 0005
IL = 45 = E
IN = 31 = 1
Bill 04 = #GB0005E1 SF = 2 DP = 2
```

13 Implementation Rules

The following rules must be adhered to meet the ccTalk DES standard.

13.1 Gaming Machine Manufacturers

For maximum security it is recommended that the gaming machine changes the DES key to a new, random value every 24 hours. However, key changes MUST not be done more often than once every 8 hours to simplify the storage requirements on the peripheral. A peripheral product lifetime of 10 years would then require about 11K write cycles.

It is the responsibility of the gaming machine manufacturer to ensure that trusted mode on the peripheral is secured by a mechanical lock or other equivalent means, that once the DES keys have been read they are not stored in a way that is easy to access, that subsequent key changes are to random values which do not follow any kind of pattern or repeat cycle, and that weak or fixed test keys are avoided. Any deviation from this will reduce the security of DES encryption and could lead to a successful attack strategy.

The gaming machine must not use a fixed mapping table for credit assignment. It must use header 108, ‘Request encrypted monetary id’, during initialisation to dynamically assign monetary position to monetary value.

13.2 Peripheral Manufacturers

If trusted mode is entered on a switch then the peripheral should be inhibited from accepting money until the switch is restored to its normal operating state.

‘Exiting trusted mode’ means restoring the physical switch position to its normal operating state but there is also a software lockout to prevent the key being read multiple times. See next.

On entering trusted mode, only 1 read of the DES key using header 111, ‘Request encryption support’, is allowed. Another read will report normal operating mode. If the machine needs another go at reading the keys then trusted mode has to be physically exited and re-entered.

It is not a requirement to have header 111 sent by the gaming machine before the peripheral can accept money.

Monetary values in header 108, ‘Request encrypted monetary id’, MUST use the ISO 3166 format for maximum compatibility. The first letter of the 3 letter country code is set to ‘#’, e.g. ‘#EU’ for Euro.

Peripherals implementing DES must report a ccTalk revision level of at least 4.6 with header 4, ‘Request comms revision’. The reply data = [xxx] [004] [006].