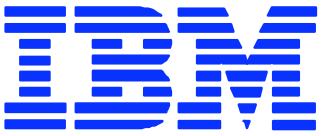
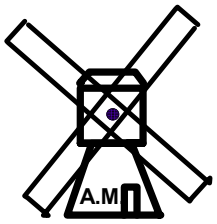


CCA / ICSF Implementation Workshop

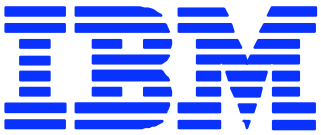
PIN Processing

PIN Verification

- A bank client withdraws Cash out of ATM
- Easy to understand transaction
- Secrecy of PIN is obvious



A. Mulder
driesmulder@kpnmail.nl



CCA / ICSF Implementation Workshop

An Example of PIN Processing

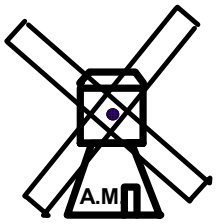
- A bank's customer complains about a PIN withdrawal

"I didn't do that cash withdrawal"

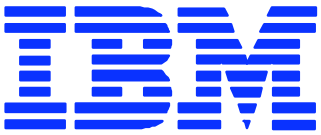
How can the bank prove that its not their fault ?

- By proving that no single employee can get a customer's PIN value
- By using certified cryptographic hardware
- By having strict procedures in place
  - To protect keys (that protect PIN's)
  - To protect PIN's

In a dispute in court, they should be able to win the case



A. Mulder
driesmulder@kpnmail.nl



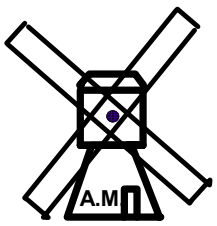
CCA / ICSF Implementation Workshop

How is the PIN usually derived ?

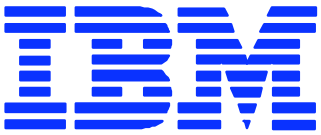
PINs are calculated as follows.

- Take the last five significant digits of the account number, and prefix them by eleven digits of validation data.
- These are often the first eleven digits of the account number. They could also be a function of the card issue date.
- In any case, the resulting sixteen digit value is input to an encryption algorithm (TDES)
- The result is then Decimalised (mapping on 0,1,...9 digits)

  Result of decimalising is the "Natural PIN"



A. Mulder
driesmulder@kpnmail.nl



CCA / ICSF Implementation Workshop

How is the PIN usually derived ? Example

Data input to TDES algorithm: 0123456789012345
PIN key input to TDES algorithm: 23232323 89ABCDEF FEDCBA98 76543210
Output of DES algorithm: EC122671 C6B1AC05
Take first 4 digits (nibbles) and decimalise

First four digits decimalised --> 4212 E -> 4 C -> 2 1 -> 1 2 -> 2
Natural PIN = 4212 (question: Is this decimalisation OK ?)

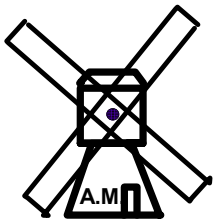
Selected Customer PIN: = 5446
Calculated Natural PIN = 4212
Offset: = 1234

Offset = Customer_PIN - Natural_PIN

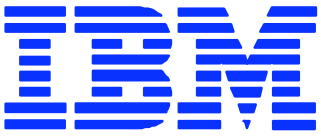
Note: Offset is only used if the customer has a selectable PIN

Decimalisation table (example)

0 -> 0
1 -> 1
2 -> 2
3 -> 3
4 -> 4
5 -> 5
6 -> 6
7 -> 7
8 -> 8
9 -> 9
A -> 0
B -> 1
C -> 2
D -> 3
E -> 4
F -> 5



A. Mulder
driesmulder@kpnmail.nl



CCA / ICSF Implementation Workshop

Decimalization of the PIN

Example: The Natural PIN data is DC 88

Now we assume here in this example the following decimalisation table:

decimalisation table = 6028 0786 0808 3644

On offset 0 we have a 6	On offset 1 we have a 0
On offset 2 we have a 2	On offset 3 we have a 8
On offset 4 we have a 0	On offset 5 we have a 7
On offset 6 we have a 8	On offset 7 we have a 6
On offset 8 we have a 0	On offset 9 we have a 8
On offset 10 (A) we have a 0	On offset 11 (B) we have a 8
On offset 12 (C) we have a 3	On offset 13 (D) we have a 6
On offset 14 (E) we have a 4	On offset 15 (F) we have a 4

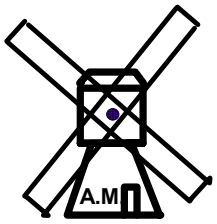
DC 88 now translates as follows to a natural PIN

D ---> 6
C ---> 3
8 ---> 0
8 ---> 0

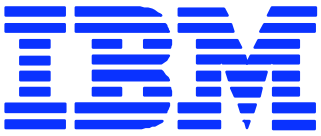
So the PIN becomes 6300

Decimalisation table used here

0 -> 6
1 -> 0
2 -> 2
3 -> 8
4 -> 0
5 -> 7
6 -> 8
7 -> 6
8 -> 0
9 -> 8
A -> 0
B -> 8
C -> 3
D -> 6
E -> 4
F -> 4



A. Mulder
driesmulder@kpnmail.nl



CCA / ICSF Implementation Workshop

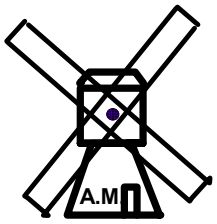
PIN Blocks

example of an **ISO0** PIN block:

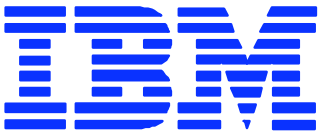
```
PAN = 12 34 56 78 90 12  (6 bytes)
PIN = 42 12  ---> PIN length = 4

Interm. PIN Bl. = 04 42 12 FF FF FF FF FF
PAN Block       = 00 00 12 34 56 78 90 12
Clear PIN block = 04 42 00 CB A9 87 6F ED
```

The PIN block is always encrypted

$$E_K(PB)$$



A. Mulder
driesmulder@kpnmail.nl



CCA / ICSF Implementation Workshop

Verification of the PIN

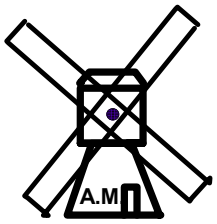
In concept there are 2 methods:

- PIN calculation method

  After receiving the encrypted trial PIN block the system calculates the correct PIN, based on PAN and other info (dec. table etc.) , decrypts the trial PIN block and compares the two clear PIN values. (inside hardware) The **CSNBPVR** call is used.

- PIN Database method (use reference PIN)

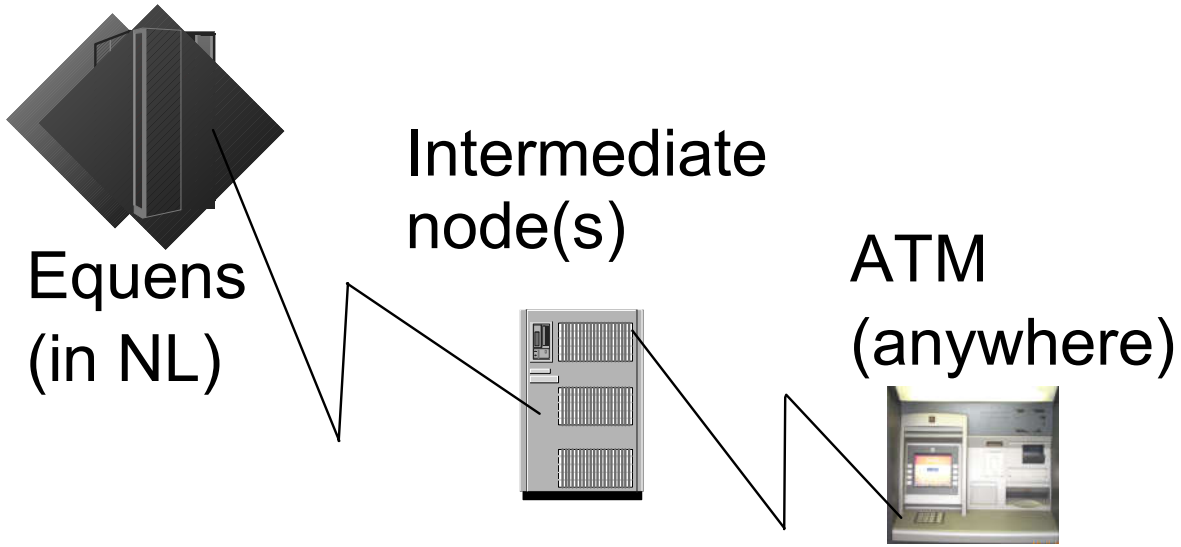
  After receiving the encrypted trial PIN block the system retrieves the encrypted reference PIN block from a database and compares the two values. This requires to use the **CSNBPTR** verb with keyword TRANSLAT.



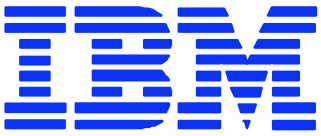
A. Mulder
driesmulder@kpnmail.nl

CCA / ICSF Implementation Workshop

Verification of the PIN (simplified)



A. Mulder
driesmulder@kpnmail.nl



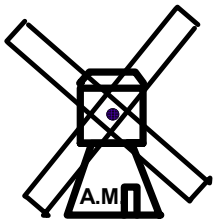
CCA / ICSF Implementation Workshop

Verification of the PIN (simplified)

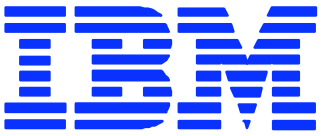
INSIDE HSM at bank-side:

- Decrypt trial PIN block $E_{K_{atm}}(PB)$
- Derive / Calculate reference PIN
- Compare decrypted trial PIN and reference PIN
- Return YES or NO to caller

No clear PIN ever shows up



A. Mulder
driesmulder@kpnmail.nl



CCA / ICSF Implementation Workshop

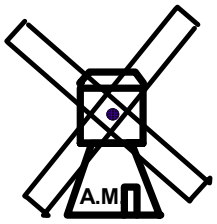
Example of HSM PINVER function (simplified)

Encrypted_PIN_Verify (
PIN_encr_key,
PIN_ver_key,
PIN_info,
PAN_info,
Encrypted_trial_PIN_block,
Result)

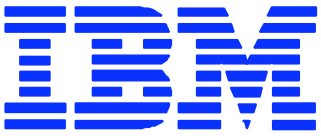
To decrypt trial PIN → PIN_encr_key

To derive reference PIN → PIN_ver_key

The function checks the correct key type



A. Mulder
driesmulder@kpnmail.nl



CCA / ICSF Implementation Workshop

Key Separation (1)

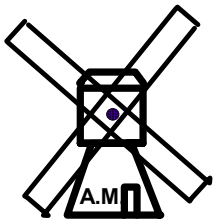
What would happen if PIN_DECIPHER_key could be used in Decipher function ?

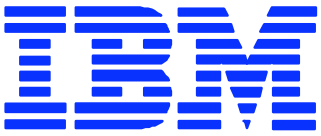
BIG TROUBLE CAN OCCUR

BECAUSE $E_{K_{atm}}(PB) \rightarrow PB$

Therefore we MUST use key separation, so that a PIN-key never can exist as a decipher-key

A. Mulder
driesmulder@kpnmail.nl



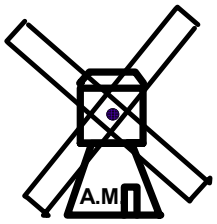


CCA / ICSF Implementation Workshop

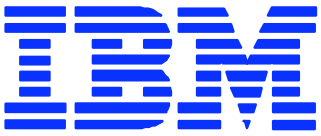
Key Separation (2)

- We limit a key to just the function it is designed to run.
- The key-type is determined before the key exists (or is generated)
- The key-type can't be changed afterwards

In a well designed key management system
no tricks are applicable on the key



A. Mulder
driesmulder@kpnmail.nl

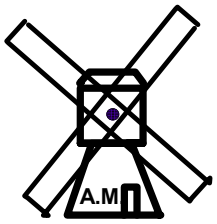


CCA / ICSF Implementation Workshop

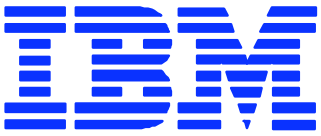
Key Separation (3)

Many hardware vendors use key-tokens that contain extended key-information

- value of key (encrypted)
- type of key
- how should the key be treated
- activation / expiration date



A. Mulder
driesmulder@kpnmail.nl

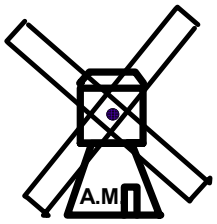


CCA / ICSF Implementation Workshop

Key Separation (4)

Many properties of the keys are coded in "Variants" or "Control Vectors" (IBM)

- A property of the key is a bit in the CV
- The CV is part of the encryption of the key

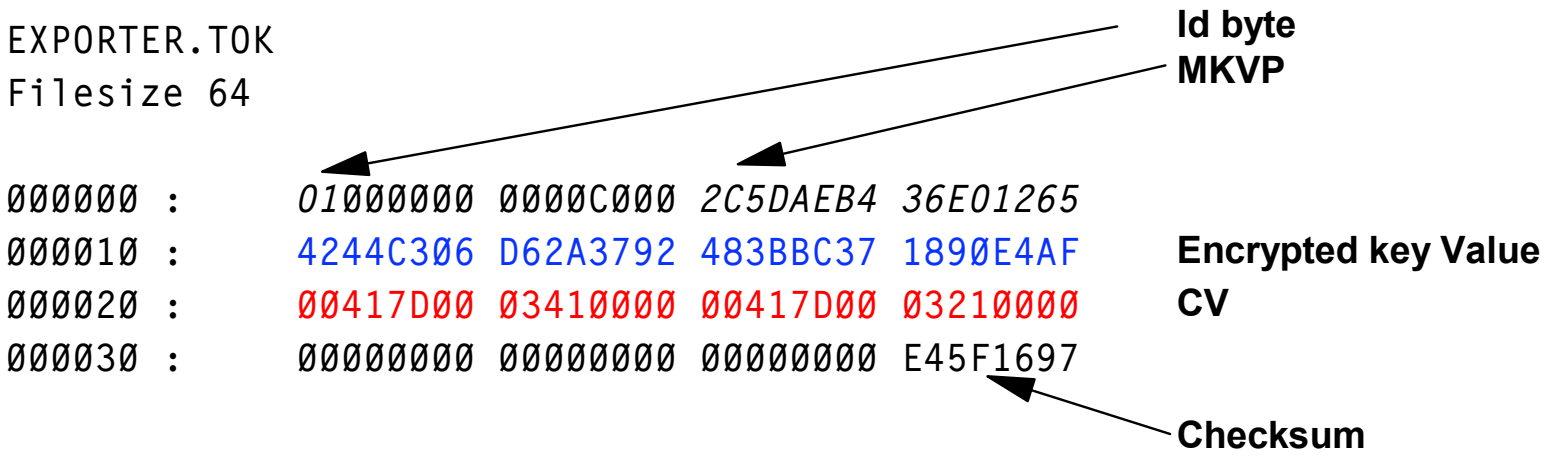


A. Mulder
driesmulder@kpnmail.nl

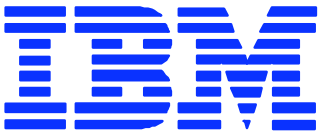
CCA / ICSF Implementation Workshop

Key Separation (5)

An example of an IBM key-token



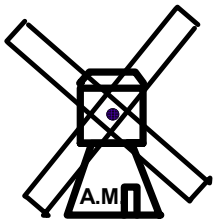
A. Mulder
driesmulder@kpnmail.nl



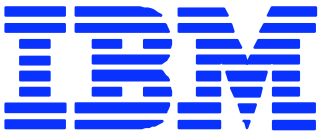
CCA / ICSF Implementation Workshop

Most commonly used PIN Verbs

CSNBCPE	Clear PIN Encrypt
CSNBPGN	Clear PIN Generate
CSNBEPG	Encrypted PIN Generate
CSNBPVR	Encrypted PIN Verify
CSNBPTR	PIN Block Translate



A. Mulder
driesmulder@kpnmail.nl

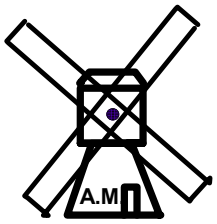


CCA / ICSF Implementation Workshop

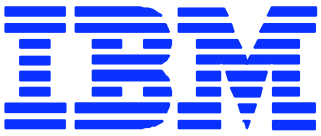
CSNBCPE Clear PIN Encrypt

Builds a Clear PIN Block and Encrypts it

```
CSNBCPE(      &rc,
              &rs,
              &exit_data_length,
              exit_data,
              opinenc_key_token,   ──►  to encrypt PIN block
              &rule_array_count,
              rule_array,
              clear_PIN,
              input_PINprofile,    ──►  to build the PAN block
              PAN_data,
              &sequence_nr,
              enc_PIN_block_ISO0);
```



A. Mulder
driesmulder@kpnmail.nl



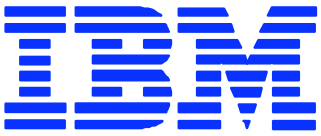
CCA / ICSF Implementation Workshop

CSNBPGN Clear PIN Generate

Generates a Clear PIN from Validation data

```
strncpy((char * ) decimalization_table, "0123456789012345"      , 16);
memcpy((char * ) data_array     , decimalization_table                  , 16);
memcpy((char * ) data_array + 16, validation_data                       , 16);
memcpy((char * ) data_array + 32, "                "                    , 16);

CSNBPGN(      &rc,
              &rs,
              &exit_data_length,
              exit_data,
              pingen_key_token,  ------>  to generate the PIN
              &rule_array_count,
              rule_array,
              &PIN_length,
              &PIN_check_length,
              data_array,        ------>  Dec table and
              clear_PIN_from_CCA);        Validation data
```

A. Mulder
driesmulder@kpnmail.nl

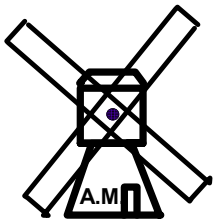


CCA / ICSF Implementation Workshop

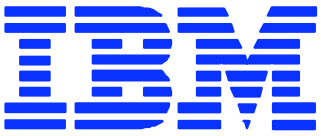
CSNBEPG Encrypted PIN Generate

Generates a PIN
Makes a PIN Block based on PIN and PAN
Encrypts the PIN block

```
CSNBEPG(      &rc,
              &rs,
              &exit_data_length,
              exit_data,
              pingen_key_token,    // to generate the clear PIN
              opinenc_key_token,   // to encipher the clear PIN
              &rule_array_count,
              rule_array,
              &PIN_length,
              data_array,              // dec. table and val. data
              input_PINprofile,
              PAN_data,          // to build PAN block
              &sequence_nr,
              enc_PIN_block_bin);
```



A. Mulder
driesmulder@kpnmail.nl

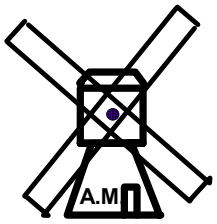


CCA / ICSF Implementation Workshop

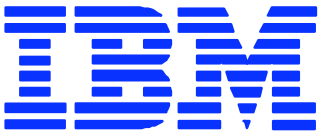
CSNBPVR Encrypted PIN Verify

Verifies a PIN from an Encrypted PIN block

```
CSNBPVR(      &rc,
              &rs,
              &exit_data_length,
              exit_data,
              ipinenc_key_token,    // IPINENC key to decipher the trial PIN block
              pinver_key_token,     // PINVER key, to generate the A-PIN
              PIN_profile,
              PAN_data,                          // To build the PAN Block
              Trial_Enc_ISO0_PIN_block,    // Coming from ATM or POS
              &rule_array_count,
              rule_array,
              &PIN_check_length,
              data_array);              // Dec. table, Validata data
                                           and offset ( if IBM-PINO)
```



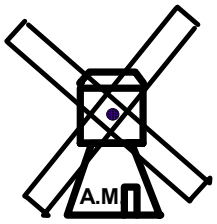
A. Mulder
driesmulder@kpnmail.nl



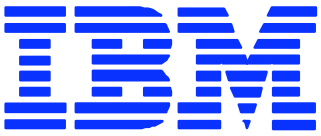
CCA / ICSF Implementation Workshop

Tricks with Decimalization tables

- There exist tricks with Dec.Tables
- An example is a DEC.table of 1111111111111111
  This will always deliver a PIN of 1111
- Another example is 0100000000000000
  Here when you enter 0000 as a trial PIN it will tell
  you whether a '2' is in the Natural PIN
  You can repeat this from 0 to 9
- See also "**Decimalisation table attacks for PIN cracking**"
  Written in 2003 by Mike Bond and Piotr Zielinski
- So you might want to protect your DEC. tables



A. Mulder
driesmulder@kpnmail.nl

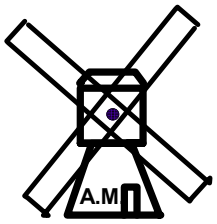


CCA / ICSF Implementation Workshop

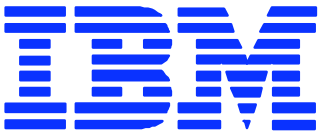
HSM Internal Decimalization tables

- To protect the INTEGRITY of DEC. tables you can store them inside the HSM
- You can load DEC. table inside the crypto hardware
  - For ICSF you store DEC. tables inside HW. using TKE
  - On a Workstation you store DEC. tables inside HW. using CSUACFC
- There is Dual control on the load of DEC tables
- "Load" and "Activate" use different ACP's

- At CSNBPVR call, you must supply the DEC.table that is to be used. The call will check then, that the same table is Loaded and set ACTIVE inside the card.



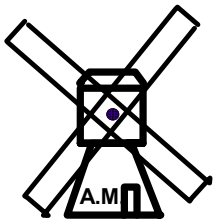
A. Mulder
driesmulder@kpnmail.nl



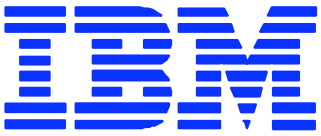
CCA / ICSF Implementation Workshop

ACP's involved in PIN Processing

- 0X353 Load Decimalization tables
- 0X354 Delete Decimalization tables
- 0X355 Activate Decimalization tables
  Note: If this ACP is ON,
  a Load will also Activate
- 0X356 Use Only Valid Decimalization tables
  To enforce a DEC Table that was
  ACTIVE already in the card



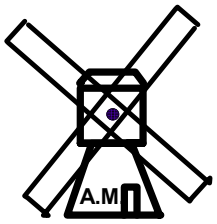
A. Mulder
driesmulder@kpnmail.nl



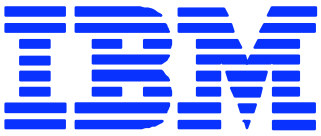
CCA / ICSF Implementation Workshop

Customer Selectable PIN's (1)

- Some banks want to enable their customers to select their own PIN
- The Natural (calculated PIN) stays the same
- There is a value OFFSET involved
- Offset = Customer PIN - Natural PIN



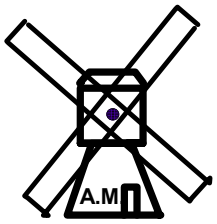
A. Mulder
driesmulder@kpnmail.nl



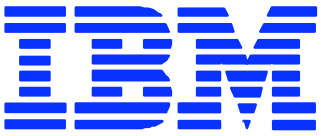
CCA / ICSF Implementation Workshop

Customer Selectable PIN's Terminology

- C-PIN The Customer selected PIN
- A-PIN The Natural (calculated PIN)
- O-PIN The OFFSET value
- T-PIN The Trial PIN



A. Mulder
driesmulder@kpnmail.nl

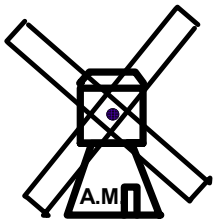


CCA / ICSF Implementation Workshop

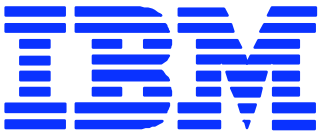
Customer Selectable PIN's (2)

The subtraction is done without carry

```
C-PIN        = 1453
Natural PIN  = 2506
             ----------
Offset       = 9957
```



A. Mulder
driesmulder@kpnmail.nl

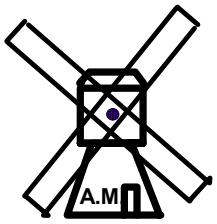


CCA / ICSF Implementation Workshop

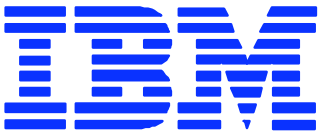
Customer Selectable PIN's (3)

- The Trial PIN block AND the Offset are Input to CSNBPVR call
- The Offset is in 3rd 16 bytes of Input parameter data_array
- Rule_array keyword is IBM-PINO and not IBM-PIN

  Then CSNBPVR will include the Offset



A. Mulder
driesmulder@kpnmail.nl

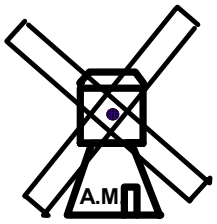


CCA / ICSF Implementation Workshop

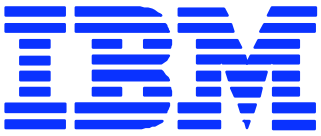
Customer Selectable PIN's (4)

These are the Customer Selectable PIN's that I use in my examples

C-PIN	= 1234	02489
Natural PIN	= 4212	91862
	---	---
Offset	= 7022	11627



A. Mulder
driesmulder@kpnmail.nl



CCA / ICSF Implementation Workshop

AM. PIN Example programs

- pin_pgn (Generate Clear PIN)
- pin_pgn_bad_dec_table

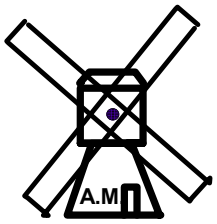
- pin_load_dec_table
- pin_activate_dec_table
- pin_list_dec_table

- pin_cpe (Build PIN Blk and Encrypt)

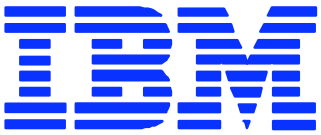
- pin_create_trial_block (emulate an ATM)
- pin_create_trial_block_offset

- pin_pvr
- pin_pvr_offset

- pin_epg (Encrypted PIN Generate)



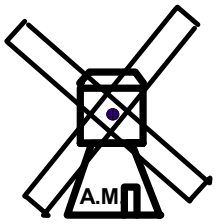
A. Mulder
driesmulder@kpnmail.nl



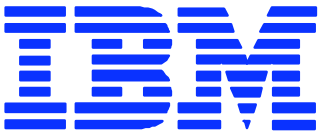
CCA / ICSF Implementation Workshop

AM. PIN Example programs Input Data

Clear PINGEN key 23232323 89ABCDEF FEDCBA98 76543210
Clear OPINENC key 01234567 89ABCDEF FEDCBA98 76543210
Decimalization table 0123456789012345



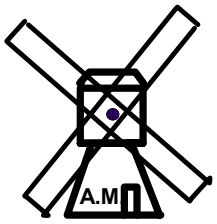
A. Mulder
driesmulder@kpnmail.nl



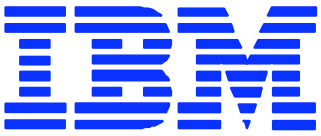
CCA / ICSF Implementation Workshop

AM. PIN Example programs Data TC1

PIN_length	4
Validation data	"123456789012"
Validation data padded	"1234567890120000" right padded w. '0'
Validation data length	12
Validation data length f. enc	16
PAN_data	"123456789012"
Enciphered val. data bin (hex)	EC122671 C6B1AC05
A-PIN , C-PIN, O-PIN	4212 1234 7022
Intermediate PIN block (hex)	044212FF FFFFFFFF
PAN_block (hex)	00001234 56789012
Clear ISO-0 PIN block	044200CB A9876FED
Enciphered ISO-0 PIN block	AAE7EAA6 26FA17D4



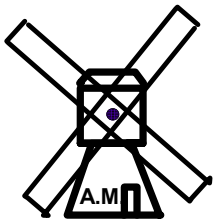
A. Mulder
driesmulder@kpnmail.nl



CCA / ICSF Implementation Workshop

AM. PIN Example programs Data TC2

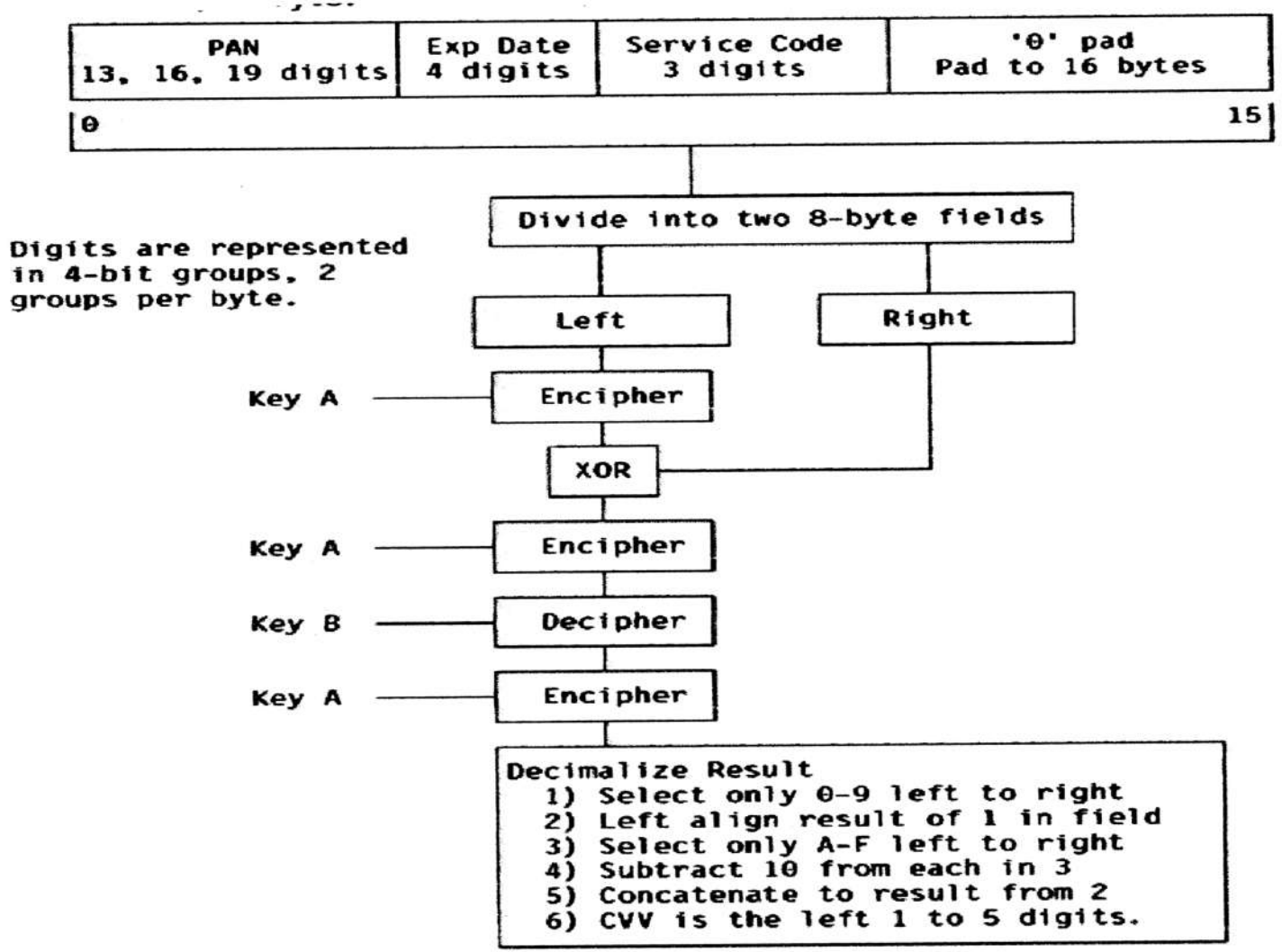
PIN_length	5
Validation data	"123456789"
Validation data padded	"123456789000000"
Validation data length	9
Validation data length f. enc	16
PAN_data	"123456789012"
Enciphered val. data bin (hex)	918621FB 8F5A853D
A-PIN , C-PIN, O-PIN	91862 02489 11627
Intermediate PIN block (hex)	0591862F FFFFFFFF
PAN_block (hex)	00001234 56789012
Clear ISO-0 PIN block	0591941B A9876FED
Enciphered ISO-0 PIN block	7F200768 16951CC8



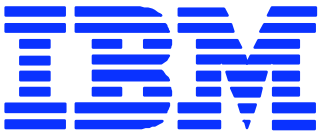
A. Mulder
driesmulder@kpnmail.nl

CCA / ICSF Implementation Workshop

CVV = Card Verification Value



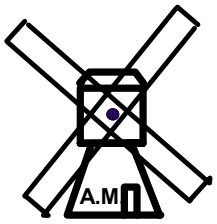
A. Mulder
driesmulder@kpnmail.nl



CCA / ICSF Implementation Workshop

CVV Generate and Verify

- The CVV will be generated at Credit Card Issue and will be printed on the card.
- Therefore the issuer needs to run CVV Generate
- CVV generate is an ICSF API call CSNDCSG
- At transaction verification a CVV Verify might run
- CVV Verify is an ICSF API call CSNDCSV
- See demo program des_cvv_generate_verify



A. Mulder
driesmulder@kpnmail.nl