Ruter AS

Electronic Ticketing in Ruter
Travelcard Production Specification

Version 1.0
2010-09-23

Revision History

Date	Version	Description	Author
2010-09-23	1.0	First edition.	Jan Gunnar Brorson

Contents

1. INTRODUCTION

This document describes the requirements for travelcards to be used in electronic ticketing in the RUTER system. It covers the following topics:

- Hardware requirements to the physical card
- Initialization of the card (initial programming of the card)
- Formatting of the card number printed on the card
- Security requirements

1.1 REFERENCES

1. ISO 7810 – Identification cards – Physical characteristics
2. ISO 7813 – Identification cards – Financial transaction cards
3. ISO 14443 – Identification cards – Contactless integrated circuit cards – Proximity Cards
4. NXP Mifare DESFire – Contactless Multi-Application IC with DES and 3DES Security – MF3 IC D40 Functional Specification (and later versions)
5. ENV 1545:1998 – Identification card systems – Surface transport applications
6. ISO 8824 – Information Technology – Abstract Syntax Notation One (ASN.1)
7. Common Requirements Specification for Interoperability – Smart Media Application Note (Thales 4020 14018 O)
8. 4020 15664 OSLO IOS SECIS H – IOS Security Specification
9. User Guide Alternative Key Distribution
10. Non-Disclosure Agreement – Production Keys

2. HARDWARE REQUIREMENTS

The card shall have the physical characteristics as described in ISO 7810/7813 [1,2], i.e. normal "credit card" format and thickness. Electrical characteristics shall be according to ISO 14443 type A [3].

The card shall be of type Mifare DESFire 4k (MF3 IC D40) [4]. DESFire EV1 4k (MF3 IC D41 01) shall be offered as an option.

The graphical layout is described in a separate document.

The vendor must give RUTER information regarding the following:

- Expected failure rate over time
- Electrical and physical reliability (ref test criteria given in ISO 7816, ISO 10536 and ISO 10373)

3. INITIALIZATION OF THE CARD

This chapter covers the initialization of the card (AKA pre-personalization). This includes putting the card in the state it has before it is issued to a customer, including writing the whole file structure to the card.

3.1 OVERVIEW

The electrical initialization, or pre-personalization process should include the following steps:

- Creation of Application ID 0x578000 that will contain up to 4 keys in its key-set
- Select AppID 0x578000
- Create one standard file (Id=0xC type=standard 1 record of 16bytes Comm settings=0)
- Write an IOCardHeader-A record in this file. This data includes the card serial number (or part of it) that is printed on the card surface. Other data elements are fixed for a batch of card (like max validity date).
- Lock the file (modify access rights so that the card access comply with the access rights given in table 3.X).
- Create a second Application (ID 0x578001 8 keys in the key set)
- Select AppId 0x578001
- Create 8 files (IDs = 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0xA, 0xC) as defined in table 2 below, length are fixed. Files 0x5 and 0x6 should be filled with all records full of 0s. All the other files shall be initialized as described in chapter 3.2.
- Change file settings to comply with access rights defined in table 3.2.

The content of each separate file is encoded using Abstract Syntax Notation One (ASN.1), Packed Encoding Rules (PER) [6].

For each production batch two values will be given as input:

- Card Id range, to encoded into CI_Header.cardIdNumber32Bits
- Card expiration date, to be encoded into CI_Header.cardValidityEndDate and T_Environment.envApplicationEndDat.

3.2 FILES AND DIRECTORY DESCRIPTION

What is described here is what is necessary for card initialization. For a complete description of the card layout, see the Oslo CRSI Smart Media Application Note [7].

3.2.1 Master File

The application is the default application of the card and represents the card level itself (PICC).

This 'Application', even if always present on the card, is not used by the transport application. Its main characteristics are:

AID: 0x00 00 00 (refers to the PICC itself),

KeySettings: 0x0B (configuration changeable, no free create/delete file, free directory list, master key changeable),

Number of keys: 1 (Master key M #0).

3.2.2 Cardissuer Directory

This application is created using the "CreateApplication" command whose parameters are set as follows:

- AID: 0x578000 CardIssuer application:given here as hexadecimal value (ie 0x57 is the MSB and 0x00 is the LSB)
- Key settings: 0x12 (configuration not changeable, no free create/delete file, free directory list, master key not changeable, other key changeable using key # 1).
- number of 3DES keys: 4 (Master key (M, #0), card Issuer key (I, #1), Card retailer key (C, #2), and Read key (R, #3).

The Card Issuer directory is designed to host 4 keys (M, I, C, R) and to handle one file. i.e. the CI_Header.

3.2.2.1 CI_Header file

The CI_Header file is created on the DESFire card to hold the Issuer fixed information. It contains a unique IOCardHeader-A structure.

This file is a standard data file (DESFire file type 0x00) with a file ID set to 0xC . It is created with a size of 32 bytes, even if less is needed.

Its access rights and security level are as follows:

- Read: always ; read is free and no key is required
- Write: never
- Read&Write: never
- Change Config: Only the Card Issuer can update this file (using the Card Issuer key which is key#1 in the application key set)
- Security level: plain text. (communication settings = 0x00)

This file is created during the initialization process and is never written to during normal process. It is 32 bytes long (256 bits) and contains one IODESfireCardHeader-IO- A data element

Note that the card number is a 32 bits serial number that makes the cards unique within the IO ticketing system. This number is stored in the chip during the card manufacturing process and is under the card issuer responsibility.

NB: Even if the definition below allows room for other ID (IODESfireCardIdNumberISO7812), the IO cards at first will have only a 32 bits serial number.

CardHeader structure description:

Encoding tutorial (ASN.1 definition prevails when conflicting).

presence bit #		Data element identifier	Data type	Size (bits)	initial value, comment
OIOCardHeader-A			SEQUENCE OF		
preamble			SEQUENCE OF	0	
•	country		0..999	10	578
•	format		0..1048575	20	0: this format version
data			SEQUENCE OF	0	
•	cardIdNumber		SEQUENCE OF	0	
	↙	choice bitmap	0..2	2	2 (= cardIdNumber32bits)
	0	cardIdNumberISO14443		0	unused
	1	cardIdNumberISO7812		0	unused
	2	cardIdNumber32bits	0..4294967295	32	given for each production batch
•	cardValidityEndDate		DateCompact	14	The day the card is no more valid. Date is set up at initialization and is set to 10 years from the date of card production.
•	owner		0..1048575	20	1 (Application owner Company)
•	retailer		0..1048575	20	3 (Retailer organisation)
•	cardKeyVersion		0..15	4	Used to identify which keys are in use. Initial = 1
				122	TOTAL size

Tab 3.1: OIODESfireCardHeader-A data fields

ASN.1 description is as follows:

```
IODESfireIdentifier::= INTEGER(0..1048575)

-- GENERIC HEADER TYPES
IODESfireCardHeaderGeneric::= SEQUENCE {
    country     INTEGER(0..999),       -- ISO 3166-1 country numeric id.
    format      IODESfireIdentifier   -- as defined by country registration authority.
}

IODESfireCardHeaderPreamble(theCountry INTEGER, theFormat IA5String)::=
   IODESfireCardHeaderGeneric WITH COMPONENTS ({country (theCountry), format
(theFormat)})

IODESfireCardIdNumberISO7812::= SEQUENCE(SIZE(10..19))
       OF NumericString (FROM "0".."9") as defined by ISO 7812/CCITT E.118.

IODESfireCardIdNumber32bits ::= INTEGER(0..4294967295)
--  as usually managed by ticketing devices, and as usual S/N.

IODESfireCardIdNumber::= CHOICE {
   cardIdNumberISO14443    NULL,
   cardIdNumberISO7812     IODESfireCardIdNumberISO7812,
   cardIdNumber32bits      IODESfireCardIdNumber32bits
}
```

```
IODESfireCardHeaderData-IO-A::= SEQUENCE {
    -- uses registration defined by (countryNumber = 578, format = 0)
    -- IODESfireCardIdNumber is for the moment only the 32 bits S/N variant
    -- IMPORT DateCompact FROM ENV1545-1: 1998 - day count from Jan. 1st, 1997.
    cardIdNumber          IODESfireCardIdNumber,
    cardValidityEndDate DateCompact,             -- the day the card is no more
valid.
    owner                 IODESfireIdentifier,     -- card owner (brand owner).
    retailer              IODESfireIdentifier      -- card retailer.
    cardKeyVersion        INTEGER(0..15)           -- Key version in use (4 bits)
}

--  IO CARD HEADER DEFINITION for Norway (578)  -:-  IO-A
--  country = Norway = 578, format = 0
--  This is the initial format definition.
IOCardHeader-A::= SEQUENCE {
    preamble   IODESfireCardHeaderPreamble (country 578, format 0),
    data       IODESfireCardHeaderData-IO-A
}
```

3.2.3 Transport Directory

The Transport application is created using the "CreateApplication" using the following parameters:

- AID = 0x578001, given here as hexadecimal value (ie 0x57 is the MSB and 0x00 is the LSB)
- Key settings: 0x12 (configuration not changeable, no free create/delete file, free directory list, master key not changeable, other key changeable using key # 1),
- Number of keys: 8 (Master key (M, #0), application Owner/issuer key (O, #1), Application retailer key (A, #2), Card retailer key (C, #3), Product retailers key (P, #4), add-Value key (V, #5), Service providers key (S, #6) and Read key (R, #7)).
- The Transport directory is designed to handle up to 8 files which are:
  - T_Environment
  - T_CardHolder,
  - T_PoductRetailer,
  - T_ServiceProvider,
  - T_SpecialEvent,
  - T_StoredValue,
  - T_GeneralEventLog,
  - T_SVReloadLog.

3.2.3.1 T_Environment file

The T_Environment file is file #0xA in the Transport Application (AID=0x578001). It is a standard file not backed-up which is configured at creation to contain a maximum of 256 bits.

Access conditions and security level are as follows:

- Read: this right allows only the ReadRecords command after having presented the

Transport Read Key (key #7 in Application 0x578001 key set).

- Write: never
- Read&Write: this right is controlled by the Transport Application Owner key (key #2 in AID 0x578001 key set).
- Change Config: controlled by the Application Owner key (key#1 in AID 0x578000 application key set). This access should normally never be used.
- Security level: plain text (communication settings = 0x00)

The initial contents of this file will be as described in table 3.3.

presence bit #	data element	data type	size (bits)	Initial value and comments
IOEnvironment		SEQUENCE	0	
↙	presence bitmap	bit string	7	1110000
•	envVersionNumber	0..63	6	0 (refers to the current application version)
0	envNetworkId	NetworkId	24	0x578000
1	envApplicationIssuerId	Company	16	3 (Company code)
2	envApplicationEndDate	DateCompact	14	Last day of validity for the transport application. Date is set up at initialization and is set to 10 years from the date of card production.
3	envPayMethod	PayMethod	0	unused
4	envAuthenticator	OCTET STRING	0	unused
5	envSelectList	SEQUENCE	0	unused
6	envData	OCTET STRING	0	unused
			67	TOTAL

Tab 3.2: IOEnvironment data fields

The Environment structure is compatible with the Environment structure defined in the ENV1545 standard [5], which defines 7 optional fields (envNetworkId, envApplicationIssuerId, envApplicationEndDate, envPayMethod, envAuthenticator, envSelectList and envData).

The IOEnvironment structure is derived from the ENV1545-1 Environment structure [5] and is as follows:

```
IOEnvironment ::= SEQUENCE {    -- 7 bits in the bit-field always 1110000 (binary)
    EnvVersionNumber       VersionNumber,    -- 6 bits 000000 (binary) = this version
    envNetworkId           [0]NetworkId     OPTIONAL,  -- 24 bits always 0x578000
```

```
    envApplicationIssuerId [1]Company       OPTIONAL,  -- 16 bits present
    envApplicationEndDate  [2]DateCompact   OPTIONAL,  -- 14 bits present
    envPayMethod           [3]PayMethod     OPTIONAL,  -- unused
    envAuthenticator       [4]Authenticator OPTIONAL,  -- unused
    envSelectList          [5]SEQUENCE SIZE(1..10) OF Pointer  OPTIONALFeil! Bokmerke er ikke
definert., -- unused
    envData                [6]OCTET STRING(SIZE(0..255))       OPTIONALFeil! Bokmerke er ikke
definert.  -- unused
} -- TOTAL : 67 bits used.
```

3.2.3.2 T_CardHolder File

The CardHolder file is file number 0xC in the Transport Application. It is a standard file not backed-up which is configured at creation to contain a maximum of 32 bytes (256 bits).

Access conditions and security level are as follows:

- Read: this right allows sonly the ReadRecords command after having presented the Transport Read Key (key #7).
- Write: never
- Read&Write: this right is controlled by the Transport Card Retailer key (key #3).
- Change Config: controlled by the Application Owner key (key#1). This access should normally never be used.
- Security level: plain text. (communication settings = 0x00)

Note that in normal use, this file is always read.

The holder file contains one Holder structure which is described in a structure that is derived from the 'Holder' structure defined in the ENV1545 standard [5].

presence bit #	data element identifier	data type	size (bits)	Initial value and comments
IOHolder.base		SEQUENCE		
↙	presence bitmap	bit string	8	
0	holderName	SEQUENCE	0	unused
1	HolderBirth	SEQUENCE	0	Only the Date Of Birth is used
	• HolderBirthDate	Datef		
	• • year	4 × BDC 0..9	16	YYYY
	• • month	2 × BDC 0..9	8	MM
	• • day	2 × BDC 0..9	8	DD

presence bit #	data element identifier	data type	size (bits)	Initial value and comments
	• HolderBirthPlace	IA5String	0	unused , set to an empty string
	• • length	0..255	8	zero
	• • component × length	character 0..127	0	unused
2	HolderBirthName	IA5String	0	unused
3	holderIdNumber	IA5String	0	unused
4	holderCountryAlpha	PrintableString	21	3 characters 0..127, set to "NOR" according to ISO 3166-1 in order to identify the country
5	holderCompany	Company	0	unused
6	holderProfiles	SET OF	0	
	• length	0..15	4	can be 0, 1 or 2, depending on subtypes, see below
	• Profiles	SEQUENCE	0	
	↙ presence bitmap	bit string	3	0b111
	0 holderNetworkId	NetworkId	24	0x578000 for Norway
	1 holderProfileNumber	CustomerProfile	6	see profile table below
	2 holderProfileDate	DateCompact	14	End of validity for this profile
7	holderData	OCTET STRING	0	Optional, present only for Agent profiles
	• length	0..255	8	
	• component × length	octet	8xn	n = length
IOHolder.extension				
↙	presence bitmap	bit string	3	
0	storedValueAutoReloadData	SEQUENCE		
	• StoredValueAutoreloadGround	Integer	16	initial = 0
	• StoredValueAutoreloadAmount	Integer	16	

presence bit #	data element identifier	data type	size (bits)	Initial value and comments
	• StoredValueAutoreloadEndDate	Integer	14	Indicates the last day the auto-reload process is still available. Day count from starting date (Jan 1st 1997 = 0).
1	holderPlusRFU-1	NULL	0	
2	holderPlusRFU-2	NULL	0	

Tab 3.3: IODESfireHolder data fields

For the initialization only the field holderCountryAlpha is set.

3.2.3.3 T_ProductRetailer file

The ProductRetailer file is file #1 in the Transport Application (AID=0x578001). It is a back-up file which is configured at creation to contain a maximum of 384 bytes (but the total amount of memory used on the card is twice this amount: 768 bytes).

Access conditions and security level are as follows:

- Read: this right allows only the ReadRecords command after having presented the Transport Read Key (key #7 in Application 0x578001 key set).
- Write: never
- Read&Write: this right is controlled by the Product Retailer key (key #4 in AID 0x578001 key set).
- Change Config: controlled by the Transport Application Owner key (key#1 in AID 0x578001 application key set). This access should normally never be used.
- Security level: plain text.with MAC computation (communication settings = 0x01)

The file is not a record file, but its linear content is logically organized in 8 blocks of 48 bytes (384bits) each. Each of these records contains one Contract data element. This way the reader can access directly the adequate contract by computing easily the offset of the record to read. The first record always starts at byte 0 from the beginning of the file. An empty record slot will be totally filled with 0x00

The unused memory (either between contract structures or at the end of the file) is filled with 0x00.

3.2.3.4 T_ServiceProvider file

This file is file #2 in the Transport Application. It is a backup file which is configured at creation to host a maximum of 128 bytes but the actual used size on the card is twice this amount i.e. 256 bytes.

Access conditions and security level are as follows:

- Read: this right allows only the ReadRecords command after having presented the Transport Read Key (key #7),
- Write: devices that have the service Provider key (key #6) are allowed to write in this file,
- Read&Write: this right is controlled by the Product Retailer key (key #4),
- Change Config: controlled by the Transport Application Owner key (key#1). This access should normally never be used.
- Security level: plain text.with MAC computation. (communication settings = 0x01)

The T_ServiceProvider file consists of three separate parts, as described in table 3.4:

- First 72 bytes is for counter structures. These shall be filled with 0x00 at initialization.
- The next 8 bytes is for context and sequence numbers. This shall always be filled with `0x00 00 10 00 00 00 00 00.`
- The last 48 bytes is the contractList. This shall be filled with 0x00 at initialization.

3.2.3.5 T_SpecialEvent File

This file is file #3 in the Transport Application. It is a backup file which is configured at creation to host a maximum of 288 bytes but the actual used size on the card is twice this amount i.e. 576 bytes.

Access conditions and security level are as for T_ServiceProvider file.

The SpecialEvent File is the placeholder of the following data structures:

- SpecialEventLog [1..8]: eight 256bit records are provided, with a total of 256 bytes. This shall be filled with 0x00 at initialization.
- SpecialEventList (3 + 3x19 = 60 bits at maximum). The structure is 32 bytes long, and shall be filled with the following value at initialization:
  ```
  0x67 00 14 5C 00 52 70 01 4D C0 05 47 00 15 5C 00
  0x56 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  ```

3.2.3.6 T_StoredValue file

The StoredValue file is file #4 in the Transport Application (AID=0x578001). It is a Value file which is used to hold the running value of the card Store Value. The value file is created with a null LowerLimit and a UpperLimit that is set to the system absolute maximum allowed value. The value in the different Stored Value data (including StoredValue event data, IOHolder StoredValue data and StoredValue files) are stored in tenth of NOK (0,1 NOK).

Access conditions and security level are as follows:

- Read: this right allows Debit and GetValue commands after having presented the Transport Read Key (key #7 in Application 0x578001 key set).
- Write: never

- Read&Write: this right allows the GetValue, Debit, LimitedCredit and Credit commands. This right is controlled by the Transport Add-Valuer key (key #5 in AID 0x578001 key set).
- Change Config: controlled by the Transport Application Owner key (key#1 in AID 0x578001 application key set). This access should normally never be used.
- Security level: plain text.with MAC computation. (communication settings = 0x01)

The LimitedCredit functionality is not used and the file is created without this functionality.

- Stored Value File Lower Limit: 0
- Stored Value File Upper limit: set to max possible value (wrt card spec: 0x7FFFFFFF).
- Stored Value File Initial value: 0
- Stored Value File Limited Credit: unused

3.2.3.7 T_GeneralEventLog file

The ProductRetailer file is file #5 in the Transport Application. It is a cyclic file which is configured at creation to contain a maximum of 288 bytes (8 records of 288 bits).

Access conditions and security level are as follows:

- Read: this right allows only the ReadRecords command after having presented the Transport Read Key (key #7).
- Write: Devices having the ServiceProvider Validation key (key #6) can write to this file.
- Read&Write: never.
- Change Config: controlled by the Transport Application Owner key (key#1). This access should normally never be used.
- Security level: plain text.

This file contains exactly 8 validation records that keep track of the different validations that the card was subject to. In order to have at 8 usable records with a back-up mechanism, the file as to be created with 9 records.

Each record shall be filled with 0x00 at initialization.

3.2.3.8 T_SVReloadLog file

The SVReloadLog file is file #6 in the Transport Application (AID=0x578001). It is a cyclic file which is configured at creation to contain a maximum of 512 bits (64 bytes) record.

Access conditions and security level are as follows:

- Read: this right allows only the ReadRecords command after having presented the Transport Read Key (key #7 in Application 0x578001 key set).
- Write: Devices having the Add-Value key (key #5 in the transport Application key set) can write to this file.

- Read&Write: never.
- Change Config: controlled by the Transport Application Owner key (key#1 in AID 0x578001 application key set). This access should normally never be used.
- Security level: plain text. (communication settings = 0x00)

This file contains up to 2 Log structures (IOStoredValueReloadEvent) relevant to add-value reload. Since this file is managed cyclically, only the current and last value Logs are available. In order to have 2 last usable records and a backup facility, on the card, the file should be created with 3 records.

Each record shall be filled with 0x00 at initialization.

3.3 DESFIRE FILE ORGANISATION SUMMARY

Application #	File #	File/Directory name	type	Access Conditions*				Record: contents	Nb of Records	Record length (bytes)	Comm. settings
				RO	W	R&W	Chg				
000000	—	MF (Master File)	DF						—	—	—
	—	Keys	key					Keys (M)①	1		—
578000(hex)	—	CardIssuerDF	DF						—	—	—
	—	CI_Keys	key					Keys (M, I, C, R)②	4		—
	0xC	CI_Header	std	alw	never	never	I	CardHeader	1	16	0
578001(hexadecimal)	—	TransportDF	DF						—	—	—
	—	T_Keys	key					Keys (M, O, A, C, P, V, S, R)	8	—	—
	0xA	T_Environment	std	R	never	A	O	Environment	1	16	0
	0xC	T_CardHolder	std	R	never	C	O	Holder	1	32	0
	0x1	T_ProductRetailer	bkup	R	never	P	O	Contract	8	48	1
	0x2	T_ServiceProvider	bkup	R	S	P	O	Counter	8	9	1
								ContextsAndSequenceNumbers	1	8	
								ContractList	1	48	
	0x3	T_SpecialEvent	bkup	R	S	P	O	SpecialEventLog	8	32	1
								SpecialEventList	1	32	
	0x4	T_StoredValue	value	R	never	V	O	StoredValue	—	—	1
	0x5	T_GeneralEventLog	cyclic	R	S	never	O	GeneralEventLog	8	36	0
	0x6	T_SVReloadLog	cyclic	R	V	never	O	StoredValueReloadLog	2	32	0

Tab 3.4: DESFire data layout

Note: green lines (light-grey) show files that have backup: ,T_ProductRetailer, T_ServiceProvider, T_SpecialEvent, T_StoredValue, T_GeneralEventLog and T_SVReloadLog are backup files. Whereas CI_Keys, CI_Header, TransportDF, T_Keys, T_Environment and T_CardHolder aren't.

(*): File Access Conditions are given by four columns: ‘RO, W, R&W, Chg’, giving which key is required for following access method:

- 'RO' for read-only (💰 read+debit),
- 'W' for write-only (💰 read+canceldebit, ↔🔃 append),
- 'R&W' for read & write (💰 read+credit, ↔🔃 read+clearfile),
- 'Chg' for change-configuration.

Key description:

key	MF (Master File)	CardIssuerDF (application #0x578000)	TransportDF (application #0x578001)
M	Master key (0)	Master key (0)	Master key (0)
A	—	—	Application Retailer key (2)
C	—	Card Retailer key (2)	Card Retailer key (3)
I	—	Card Issuer key (1)	—
O	—	—	Application Owner key (1)
P	—	—	Product retailers key (4)
R	—	Read key (3)	Read key (7)
S	—	—	Service Providers key (6)
V	—	—	Add-Value key (5)

Tab 3.5: DESFire key set

3.4 EXAMPLE

An example of the card contents after initialization is showed below. Note that the bytes in red bold text are values given as input to the initialization, and will vary for each batch. The cardId32Bits (underlined) varies for each card. In this example the cardIdNumber32Bits is 1200002008 and the cardValidityEndDate and envApplicationEndDate is 6938 (Dec 31, 2015).

```
AID        : 57 80 00
```

```
  FILE      : 0C - 00 Standard data file
   off0000 : 90 80 00 02 47 86 93 D8 6C 68 00 00 40 00 0C 40

AID         : 57 80 01

  FILE      : 0A - 00 Standard data file
   off0000 : E0 02 BC 00 00 00 1B 63 40 00 00 00 00 00 00 00
   off0010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

  FILE      : 0C - 00 Standard data file
   off0000 : 08 9D 3E 90 00 00 00 00 00 00 00 00 00 00 00 00
   off0010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

  FILE      : 01 - 01 Backup data file
   off0000 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0030 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0040 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0050 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0060 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0070 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0080 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0090 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off00A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off00B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off00C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off00D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off00E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off00F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0100 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0110 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0120 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0130 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0140 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0150 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0160 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0170 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

  FILE      : 02 - 01 Backup data file
   off0000 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0030 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0040 : 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00
   off0050 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0060 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
   off0070 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

  FILE      : 03 - 01 Backup data file
   off0000 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
    off0010 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off0020 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off0030 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off0040 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off0050 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off0060 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off0070 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off0080 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off0090 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off00A0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off00B0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off00C0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off00D0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off00E0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off00F0 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    off0100 : 67 00 14 5C 00 52 70 01 4D C0 05 47 00 15 5C 00
    off0110 : 56 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

  FILE      : 04 - 02 Backup value file
      value : 00 00 00 00

  FILE      : 05 - 04 Backup cyclic record file
      Rec00 : 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 00 00 00 00 00
      Rec01 : 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 00 00 00 00 00
      Rec02 : 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 00 00 00 00 00
      Rec03 : 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 00 00 00 00 00
      Rec04 : 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 00 00 00 00 00
      Rec05 : 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 00 00 00 00 00
      Rec06 : 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 00 00 00 00 00
      Rec07 : 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 00 00 00 00 00

  FILE      : 06 - 04 Backup cyclic record file
      Rec00 : 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 00
      Rec01 : 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 00
```

4. CARD PRODUCTION REPORT

After manufacturing cards a report file must be created in a given format, containing information about each card that has been produced. The format we have to agree on during negotiation.

5. FORMATTING OF THE PRINTED CARD NUMBER

The card printed number will be 16 digits (grouped 4 by 4 as 1234 5678 9101 1121).

0	2	0	0	3	N	N	N	N	N	N	N	N	N	N	L
County		Card issuer			cardIdNumber32bits										Luhn

The county code is according to the numbering scheme for Norwegian counties, as given in table 5.1. Ruter use 02 for county code, even if we operate in both Oslo and Akershus. Card issuer is the public transport operator's (Ruter's) company code within the Norwegian electronic ticketing scheme. cardIdNumber32bits is the sequential card number encoded in the CI_Issuer file in the card header. This number is globally unique, and will be specified for each production batch of cards. L indicates CRC computed by the Luhn algorithm, described in chapter 5.1.

Code	County
00	National level
01	Østfold
02	Akershus
03	Oslo
04	Hedmark
05	Oppland
06	Buskerud
07	Vestfold
08	Telemark
09	Aust-Agder

Code	County
10	Vest-Agder
11	Rogaland
12	Hordaland
14	Sogn og Fjordane
15	Møre og Romsdal
16	Sør-Trøndelag
17	Nord-Trøndelag
18	Nordland
19	Troms
20	Finnmark

Tab 5.1: County codes

Example:

If the cardIdNumber32Bits is 650001 (hex 00 09 EB 11) the 16 digit card number will be as follows:

0	2	0	0	3	0	0	0	0	6	5	0	0	0	1	3
County code		Card issuer			cardIdNumber32bits										Luhn

It shall then be printed on the card as:

0200 3000 0650 0013

5.1 LUHN CRC

The Luhn Algorithm is a one decimal digit added at the end of the ticket printed number and is computed as follow:

Digits are taken from the last right digit to the left

Each even ranked digits are multiplied by 2, if the result is greater than 10, the 2 digits of the result are summed up

Then all the digits are summed up and the result should be a multiple of 10

An example of how programmatically check the printed number can be given by the following code example (taken from some visual basic script):

CheckLuhn: checks if the string given as input has a correct CRC.

```
Function CheckLuhn(szText As String) As Boolean
  Dim iDigit   As Integer     ' one digit
  Dim iSum     As Integer     ' Sum of all digits
  Dim fOddRank As Boolean     ' Flag set if the digit has to be multiplied by 2
  iSum = 0       ' Start initially with 0
  fOddRank = True ' Rank of last digit is odd (they are counted from the end)
  For i = Len(szText) To 1 Step -1
    iDigit = CInt(Mid(szText, i, 1))
    If fOddRank = True Then ' if the rank is odd (counted from last)
       fOddRank = False   '   Next will be even
    Else                    ' else it's to be multiplied by 2
       fOddRank = True    '   next will be Odd
       iDigit = iDigit * 2
       If iDigit >= 10 Then   ' If greater than 10, add the 2 digits of the result
                              ' (in this case the first digit is always 1)
         iDigit = iDigit - 9 '=1+(iDigit-10)
       End If
    End If
    iSum = iSum + iDigit        ' Add the result
  Next
  If (iSum Mod 10) = 0 Then
    CheckLuhn = True
  Else
    CheckLuhn = False
  End If
End Function
```

GetLuhn: from a string without the CRC computes a digit that is a correct CRC

```
Function GetLuhn(szText As String) As Integer
  Dim iDigit   As Integer     ' one digit
  Dim iSum     As Integer     ' Sum of all digits
  Dim fOddRank As Boolean     ' Flag set if the digit has to be multiplied by 2
  szText = szText + "9"       ' Adds a 'wrong' CRC that will be truncated at the end
  iSum = 0       ' Start initially with 0
  fOddRank = True ' Rank of last digit is odd (they are counted from the end
  For i = Len(szText) To 1 Step -1
    iDigit = CInt(Mid(szText, i, 1))
    If fOddRank = True Then ' if the rank is odd (counted from last)
       fOddRank = False   '   Next will be even
    Else                    ' else it's to be multiplied by 2
       fOddRank = True    '   next will be Odd
       iDigit = iDigit * 2
       If iDigit >= 10 Then   ' If greater than 10, add the 2 digits of the result
                              ' (in this case the first digit is always 1)
           iDigit = iDigit - 9 '=1+(iDigit-10)
       End If
    End If
```

```
    iSum = iSum + iDigit       ' Add the result
  Next
  GetLuhn = 9 - (iSum Mod 10)
End Function
```

6. SECURITY REQUIREMENTS

During card initialization the production card keys must be used. These keys are secret, and shall be handled confidentially. The supplier must have implemented and document internal procedures and quality assurance to ensure a secure and proper handling of the keys. These procedures shall ensure that no single person will be able to access the keys in clear text. Only through a key distribution ceremony shall the keys be accessed, minimum including three different persons. The keys shall at no time be stored in clear text.

The vendor must sign the standardized non-disclosure agreement [10] to become eligible to receive the production keys.

6.1 KEY DISTRIBUTION

Today two alternative ways to distribute the keys to the card manufacturer exist.

1. Use of physical media, encrypted and decrypted by SAMs delivered by the Norwegian national security administrator as described in SECIS [8].
2. Split up in three parts, encrypted by a personal PGP public key for three parties within the card manufacturer's organization and sent per e-mail, as described in [9].

As of now alternative 2 is used.

7. TEST AND PRODUCTION

The following main steps shall be followed before manufacturing the production cards:

1. Produce test batch of 10 cards with test keys, to verify initialization procedure. These cards do not need any graphic details. Also the report file shall be produced. This shall happen with the same software as will be used for production cards. CardIdNumber32Bits will be assigned separately for this test.
2. Produce test batch of 10 cards with full graphic layout. These cards do not need any initialization.
3. Produce test batch of 10 cards with production keys, with full initialization and full graphic layout, including the 16 digit card number. For this process the 10 first cardIdNumber32Bits values within the range shall be used.

Each test will be repeated as many times as necessary, until the result is satisfactory.

For each test run also the card report XML shall be created.

7.1 TEST KEYS

For test of procedures and card initialization the following keys shall be used:

```
MF (Master File):
  M: Master key (0):         x00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

CardIssuerDF (app 0x578000):
  M: Master key (0):         x00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  I: Card Issuer key (1):    x00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  C: Card Retailer key (2):  x00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  R: Read key (3):           x00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

TransportDF (app 0x578001):
  M: Master key (0):         x00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  O: App. Owner key (1):     x01 01 01 01 01 01 01 01 00 00 00 00 00 00 00 00
  A: App. Retailer key (2):  x02 02 02 02 02 02 02 02 00 00 00 00 00 00 00 00
  C: Card Retailer key (3):  x03 03 03 03 03 03 03 03 00 00 00 00 00 00 00 00
  P: Prod. Retailer key (4): x04 04 04 04 04 04 04 04 00 00 00 00 00 00 00 00
  V: Add-Value key (5):      x05 05 05 05 05 05 05 05 00 00 00 00 00 00 00 00
  S: Serv. Provider key (6): x06 06 06 06 06 06 06 06 00 00 00 00 00 00 00 00
  R: Read key (7):           x07 07 07 07 07 07 07 07 00 00 00 00 00 00 00 00
```

8. APPENDIXES

A. User Guide Alternative Key Distribution

B. Non-Disclosure Agreement – Production Keys