CIPURSE™ V2

Integrating CIPURSE™ V2 into an existing Automated Fare Collection system

OSPT Alliance
Prinzregenten Str. 159
D-81677 Munich
Germany

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, OSPT Alliance hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Information

For further information, please see OSPT Alliance Web Page: www.osptalliance.org or contact info@osptalliance.org.

We listen to your comments:

We are constantly striving to improve the quality of all our specification and documentation. We have spent an exceptional amount of time to ensure that it is correct. However, we realize that we may have missed a few things. If you find any information that is missing or appears to be in error, please use the contact section to inform us. We appreciate your assistance in making this a better document.

Please be aware of that for the implementation of the CIPURSE specification into products further IPR licenses may be required: CIPURSE cryptography IPR can be licensed from the OSPT IP Pool GmbH. Contact: info@ospt-ip-pool.com.

Transaction mechanism is covered by additional third party IPR, not licensable from the OSPT IP Pool. It is the responsibility of the implementer to contact the IPR owner

Trademarks:
CIPURSE and OSPT are trademarks of OSPT Alliance.

MIFARE, MIFARE Classic, MIFARE Ultralight, MIFARE PLUS, MIFARE DESFire are registered trademarks of NXP B.V.

Table of Contents

List of Figures

List of Tables

1 Scope

This document gives technical guidelines for integrating CIPURSE™V2 in an existing Automated Fare Collection (AFC) system.

It analyses the most common AFC systems and exposes different solutions depending on the hardware, the proximity integrated circuit card (PICC) type and the security of the current system.

Finally, it provides several use cases with detailed technical information to perform all the steps in the integration process. It also recommends the appropriate CIPURSE™V2 profile for each case.

2 Introduction

The OSPT Alliance is helping the transit community move towards the next generation of secure, cost-effective, and flexible fare collection solutions through a global, multi-provider community.

Its charter is to define a new open standard for secure transit fare collection solutions, while providing industry education, creating workgroup opportunities, and to be a catalyst for the development and adoption of innovative fare collection technologies, applications, and services. The OSPT Alliance is also building a global ecosystem of transit operators, transit consultants and integrators, technology solution providers, and government agencies to stimulate development and delivery of next-generation fare collection solutions.

The CIPURSE open security standard addresses the needs of local and regional transit authorities to have future-proof fare collection systems with more advanced security than those currently in use. The standard defines an authentication scheme, a secure messaging protocol, four minimum mandatory file types and a minimum mandatory command set to access these file types. It also specifies encryption keys and access conditions.

CIPURSE builds upon existing proven open standards - the [ISO/IEC 7816-4] smart card standard, as well as the 128-bit advanced encryption standard [AES-128] and the [ISO/IEC 14443-4] protocol layer to provide a platform for securing both new and legacy transit fare collection applications, and has the potential to be used within existing application frameworks around the world. At the same time, because it is an open standard, it promotes vendor neutrality, cross-vendor system interoperability, lower technology adoption risks, and higher quality and improved market responsiveness; all of which result in lower operating costs and greater flexibility for transport system operators and authorities.

This whitepaper is designed to overcome the misconceptions around the costs and the level of complexity of integrating CIPURSE into an existing AFC system. It explains how to implement the standard quickly and easily while minimizing investment. CIPURSE offers significant benefits, such as reduced risk, an improved cost structure and less dependence on single source.

Following the OSPT and CIPURSE philosophy, this paper discusses integration not migration, as a key objective of OSPT is vendor neutrality, so avoiding monopolies and vendor lock-in wherever possible.

3 Current AFC Systems

There are many types of AFC systems around the world which is due to the fact that system requirements and limitations vary from city to city and the systems have been designed around different sets of criteria. CIPURSE can still be integrated despite these variations, and these different methods according to the characteristics of the AFC system, will be defined in this paper.

AFC systems are mainly composed of, as shown in Figure 3-1, different types of fare media, such as cards, mobile phones or other types of tags (this paper will define all fare media as PICC), different types of system terminals, such as validators or top up machines, that use secure modules (often in a small card form factor also called SAM) to protect system keys. These are networked with a backend that provides system configuration to the terminals and processes all the information generated by the terminals.

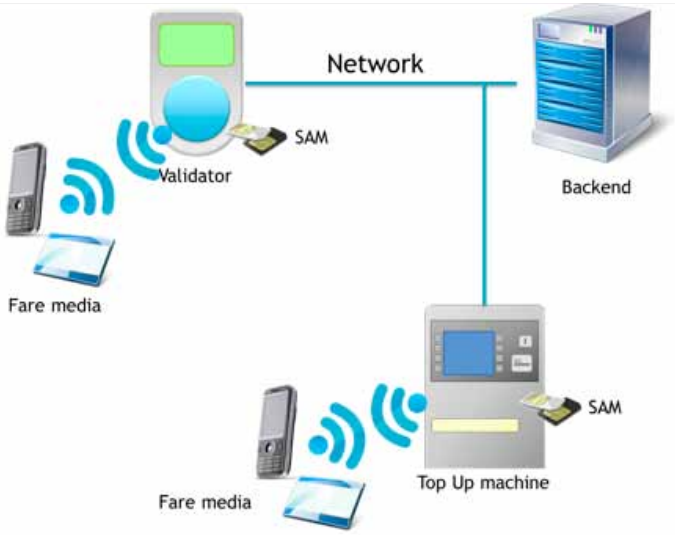

Figure 3-1 Main AFC system components

Despite most AFC systems having this structure in common, the technologies in use may differ considerably, as this paper will illustrate.

3.1 Data Location

Every AFC system has a set of data fields to identify, configure or customize the fare product associated with the customer's PICC. Moreover, a record is generated of each transaction, which includes information such as the time and place of the transaction. AFC systems can be divided in to those that store this data in the PICC, and those that store it in the backend.

The first type is considered to be an offline system, while the second is considered to be an online systems because it cannot operate without being connected to the backend as this provides the information required to perform transactions.

Transaction time is a key factor in public transportation. As online systems usually have longer transaction times than offline systems, they are not as common. Therefore, offline systems will be the focus of this paper. However, CIPURSE can be integrated into online systems as they normally use the PICC to store only a unique identifier (UID) that identifies each individual user's PICC in the system, and a secret key to perform an authentication process. There is not a common procedure for both operations, but they can usually be easily replicated in CIPURSE PICCs.

There are many types of offline systems that can be classified according to their features as discussed in the following sections.

3.2 Memory Organization

PICCs can be classified according to the way its memory is organized including those that divide memory into blocks as well as those with a file system. In both cases, reading and writing of data may be protected by security algorithms (see section 3.3).

Organizing memory with files has distinct advantages over block memory organization:

Data can be structured logically, without needing to know how and where data is stored physically in the PICC. This greatly simplifies data handling in terminals and makes data structure independent from the PICC, which facilitates future technology integrations and migrations. Since file size varies, files usually store data fields with similar characteristics regardless of the amount of bytes. In contrast, the same data in a block oriented memory organization needs to be split up and stored in a number of blocks depending on size instead of being filed in logical groups. In addition to dividing effort, the need for multiple commands to process each block causes performance degradation and is the main issue with this type of memory organization.

Data with the same access conditions can be stored in the same file in order to simplify the security architecture design and make it more robust and less vulnerable to breaches and errors.

Files can provide extra functionality which can make the data they contain more useful depending on the file type. Common file types include:

Binary or transparent files are basic files that simply store data.

Linear and cyclic record files provide record functionality. This is useful to store different types of data groups such as historical data containing the information of each journey.

Value files provide extra functionality to manage quantities and counters. They have special commands to increase and decrease their value which may have restrictions according to their access conditions.

The withdrawal of the PICC from the radio frequency (RF) field during a data update can interrupt the data transmission, resulting in partially written and inconsistent data being stored in the PICC. In some PICCs, the file system has consistent data update mechanisms to avoid this.

3.3 Security

PICCs used in public transportation usually incorporate security mechanisms to avoid copies and modifications by non-authorized actors. In the market, almost every PICC type has its own security mechanism. This makes the coexistence of several different PICCs in the same AFC system difficult.

Amongst all security mechanisms, we can differentiate between open mechanisms and closed mechanisms, depending on whether the specification is public or not, respectively. Open mechanisms usually use standard cryptographic algorithms such as Triple DES or AES and since its specification is open it can be tested and trusted by all. Therefore, the strength of open cryptographic algorithms rely on the algorithm itself. The strength of closed cryptographic algorithms, usually relies on the fact that their specification is closed, and that there is no access to it, rather than being integral to the algorithm itself.

For this reason, if the specification of the cryptographic algorithm or a part of it is revealed, any weaknesses in the algorithm are also exposed. Consequently, open algorithms are preferable to closed algorithms. Another disadvantage of closed algorithms is that they usually make use of an external element to perform cryptographic operations. This element is normally embedded in a custom chip soldered in the terminal printed circuit board (PCB), reducing the number of compatible terminal suppliers.

All security systems use keys to perform encryptions and calculate the Message Authentication Code (MAC). Depending on their role, terminals perform different operations on the PICC and access different data with different access conditions. Depending on the access conditions, terminals use a key or such like to perform cryptographic operations during transactions.

To protect the keys from attack, there are some security rules that are commonly used, such as key diversification. Diversification is performed with an algorithm that uses a PICC unique identifier to derive the original key (usually called the master key), resulting in a new and unique key for each PICC. It means that in the event of a breach where a key is discovered, there will be no information about the original key. Usually, the [ISO/IEC 14443-3] UID is used as a PICC unique identifier. In CIPURSE, PICC identification data (see [CIPURSE_OpInt]) should be used rather than the [ISO/IEC 14443-3] UID.

In order to perform the transaction as quickly as possible, keys are usually stored in the terminal. An effective and widely used measure against attackers is storing keys in security modules instead of in the memory of the terminals. These

security modules are commonly called Secure Access Module or Secure Application Module (SAMs). SAMs are usually smartcard ICs that have many layers of protection against attacks, making the access to the data they store very difficult.

OSPT recommendation is to use SAM implementing CIPURSE cryptography, independently of the CIPURSE profile (L, S or T).

SAM modules also have cryptographic accelerators to enhance security and cryptography performance. They are used to perform encryptions and MAC quickly, so terminals can delegate this responsibility, simplifying the firmware and often reducing transaction times.

3.4 Radio Frequency Standards

In spite of the large number of different PICCs found on the market, very few radiofrequency standards are used to transmit data. In particular, ISO/IEC 14443 and JIS X 6319-4. More recently, ISO/IEC 18092 (NFC), which is derived of these two standards, has also been released. All three utilize the 13.56 MHz frequency.

The most common standard is ISO/IEC 14443 but not all PICCs are fully compliant. ISO/IEC 14443 has four parts:

- Part 1 specifies physical characteristics.
- Part 2, radiofrequency power and signal interface.
- Part 3, initialization and anti-collision.
- And Part 4 specifies the transmission protocol.

PICCs partially or fully compliant with ISO/IEC 14443 are compliant with Part 1 and 2; most of them with Part 3, while only a few with Part 4. For example, MIFARE® Classic and MIFARE Ultralight are only Part 1, 2 and 3 compliant, but MIFARE DESFire and Calypso are also Part 4 compliant.

ISO/IEC 14443 defines two communication modes that are addressed as Type A and Type B. Almost all PICCs can use only one mode so PICCs are commonly addressed as Type A PICCs or Type B PICCs. Since most of the AFC systems are using just one type of PICC their terminals tend to be prepared for the communication mode of this PICC (for example Type A). Despite the requirements of ISO/IEC 14443, they are not compatible with the other communication mode (for example Type B). Thus, the AFC systems can also be described as Type A or Type B systems.

Examples of Type A PICCs are MIFARE family PICCs. The most common Type B PICC is Calypso, despite the fact that legacy Calypso systems tend to be Type B', not to be confused with Type B, and is not fully compliant with ISO/IEC 14443. Newer versions of Calypso PICCs are Type B compliant.

CIPURSE V2 is fully compliant with all parts of ISO/IEC 14443 and it can be both Type A and Type B, depending on the implementation.

The JIS X 6319-4 standard is used by FeliCa PICCs. It was proposed that FeliCa should be categorized as Type C for ISO/IEC 14443, but it was rejected. However, FeliCa RF interface got standardized as part of ISO/IEC 18092.

4 CIPURSE V2 Integration

The ease of CIPURSE V2 integration will vary depending on the existing AFC system technology's implementation and may be divided into backend and terminals integration.

4.1 Backend Integration

CIPURSE integration may affect the backend, especially those parts dealing with life cycles of PICCs, SAMs and keys. If life cycle implementation does not support adding new PICC, SAM or Key types, it will need an upgrade. If it is not possible to modify the backend to include the new CIPURSE life cycles, a simple solution is to implement a backend extension in order to manage them. This extension is not necessarily integrated into the existing backend, but can be implemented as an independent part of it; as life cycles do not usually affect terminal/backend transactions.

Another element that has to be checked to ensure the backend compatibility is the PICC serial number (PICC SN). In most AFC systems, each PICC is identified using a unique identifier in the system (serial number) that is used to provide card dependent functionalities, such as action lists (e.g. black lists). In current AFC systems, the [ISO/IEC 14443-3] UID is commonly used as PICC SN, whereas some systems store it in a field inside the card.

Current and CIPURSE PICC [ISO/IEC 14443-3] UID may differ in size. If this is the case, and UID is being used as a PICC SN field in the terminal/backend transactions, or in any table in the backend, it will probably be necessary to upgrade parts of the backend using this field. Especially if the PICC SN field has a fixed length smaller (e.g. 4 bytes) than the CIPURSE PICC [ISO/IEC 14443-3] UID.

If it is hard to modify the backend, this issue can be resolved without any backend modification by adding a field in CIPURSE V2 PICCs that acts as the PICC SN in the system. Additionally, terminals will have to use this PICC SN field (instead of the [ISO/IEC 14443-3] UID) in the backend transactions. Note that this solution involves generating and managing these SN and ensuring their uniqueness, taking special care to avoid overlapping ranges of both types of SN. An additional benefit of this solution is that the system operator can exclude double assignment problems that are getting more and more critical due to reuse of [ISO/IEC 14443-3] UIDs from the exhausted 4-byte UID range.

4.2 Terminal Integration

The most affected part of the system are the system terminals (validators, vending machines, etc.). Since CIPURSE V2 is ISO/IEC 14443 compliant and the hardware of most of the existing AFC systems is also ISO/IEC 14443 compliant, in the majority of cases no hardware upgrades are needed to integrate CIPURSE V2 into the current AFC System. However, the firmware must be upgraded, and this document outlines

a step-by-step method to upgrade the firmware in most AFC systems. Please note, however, that this guide may not be the most appropriate for all existing AFC systems.

Figure 4-2 shows a simplified view of a common Terminal Application structure dividing it into two parts: Fare Collection Management Application (FCMA), where the business logic of the fare collection is located, and the PICC Management Layer that implements the logic that deals with the PICC specific implementation, such as the command set, security operations (e.g. authentications and cryptographic operations) or the PICC memory organization. keys are usually stored in a Secure Module that it is usually found with an ID-000 (defined in the [ISO/IEC 7810]) format (usually called SAM), but it may have other formats, such as USB or even remote Secure Modules connected to the terminal. Secure Modules often include specific cryptographic functionalities to facilitate the firmware implementation or to provide extra security. In other systems, keys are (insecurely) stored in the memory of the terminal.

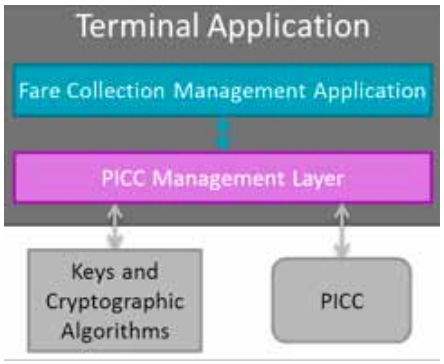

Figure 4-2 Common structure of Terminal Application

Integrating CIPURSE V2 will add new keys, cryptographic algorithms, command sets, and possibly a new memory structure to the system. Consequently, the firmware of the terminals has to be modified to manage them. In order to minimize the introduction of errors with the upgrade, the FCMA should not be modified. This can be achieved by making the PICC Management layer capable of dealing with both types of PICCs and providing the FCMA with a frontend that can manage abstracted fare collection data and operations.

The new Terminal Application structure can be seen in Figure 4-3, where the old PICC Management Layer is replaced with a new Layer that provides abstract data and operations. This abstraction is accomplished by adding two abstraction sublayers: Security Abstraction Layer (SAL) and Data Abstraction Layer (DAL). The former provides an abstraction of the security operations (authentication, cryptographic algorithms, etc.) whereas the latter provides abstraction of the PICC command set and the PICC memory organization.

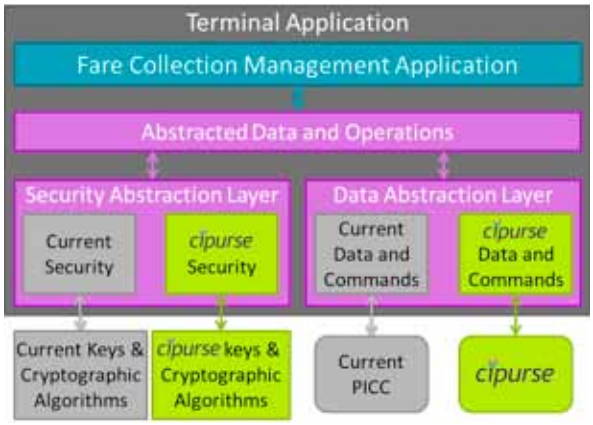

Figure 4-3 Terminal Application structure after CIPURSE V2 integration

After the integration there will be two different types of keys and cryptographic algorithms that can be managed in many different ways. For example, if the current keys and cryptographic algorithms are stored in a SAM, CIPURSE V2 keys and cryptographic algorithms can be integrated by adding a CIPURSE V2 SAM, as illustrated in Figure 4-4.

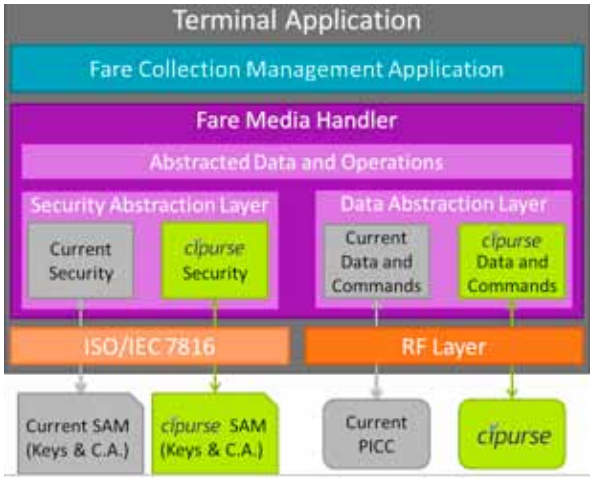

Figure 4-4 CIPURSE integration using two SAMs

Another option is to use a new custom secure module (local or remote) that is able to manage keys and cryptographic algorithms for both the current and the CIPURSE PICC, as illustrated in Figure 4-5.

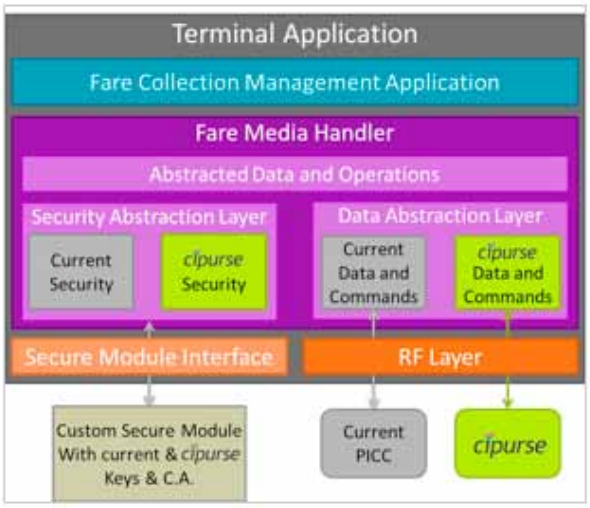

Figure 4-5 CIPURSE integration using a custom secure module

Or, the abstraction layers logic can be moved to the secure module. For example, SAL is moved to the secure module in Figure 4-6.

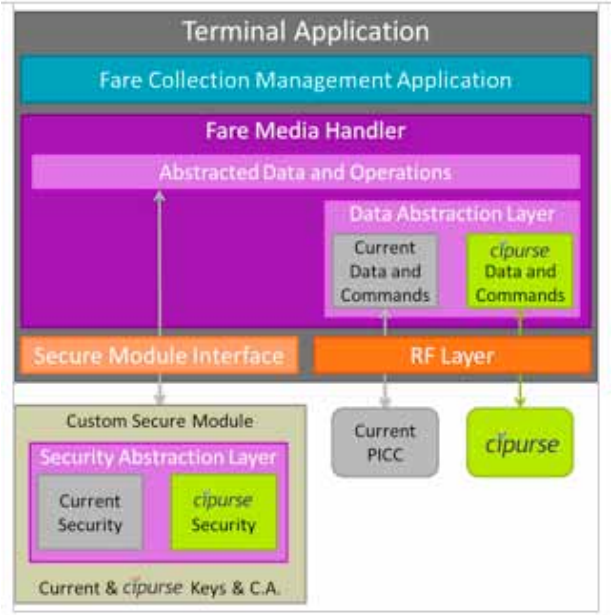

Figure 4-6 CIPURSE integration using a custom secure module that implements the SAL

The illustrations above also show the RF Layer between DAL and PICCs. This layer is where the low level RF communications are implemented, such as ISO/IEC 14443 communications. If the existing terminals are not compliant with ISO/IEC 14443 Parts 1, 2, 3 and 4, the RF Layer must be upgraded to match the RF interface implemented in CIPURSE V2 products.

In the following sections, a solution to upgrade the firmware of the terminals is proposed. The upgrade methodology is divided into three sections: Data Integration, where the upgrade to a DAL is described for AFC systems using memory block PICCs and systems using file system PICCs; Security Integration, where the upgrade to a SAL is described; and, RF Layer Update, where the requirement to update this layer is discussed.

4.2.1 Data Integration

The FCMA operates using data fields. There are many data fields in every AFC System. Some of them control the validity of a ticket, such as the expiration date or the wallet balance; others are used to control the top up operations, such as the date and time of the top up operation or the place where it has been performed. Many additional fields may exist including: user identification fields, personalization fields, version control fields, and such like.

These data fields are stored in PICCs, which can be read and written by the system terminals. Memory organizations amongst different PICC types (for example, MIFARE Classic, DESFire and CIPURSE) are not usually compatible. In order to integrate CIPURSE V2 into one of these systems, a DAL must be implemented to handle these differences and provide to the FCMA an abstract access to data fields. In the proposed solution, DAL is implemented in the firmware of the terminals, but there are alternatives as described in section 4. The DAL is the main trend in new AFC systems and it may be found in some existing systems.

As described in section 3.2, two types of memory organization are found in current PICCs: blocks and file system memories. This chapter describes how to easily and quickly implement the DAL, minimizing the time and the cost of the development.

If time and budget are not high-priority, redesigning the data structure from the beginning, knowing that two or more different PICCs will be used, will result in a more efficient and scalable system.

DAL also includes PICC command abstraction. Commands such as reading and writing are usually different in each PICC type. DAL will make the FCMA layer (see Figure 4-3) independent of these differences. For example, it will convert generic reading commands of PICC independent data coming from the FCMA layer to PICC dependent commands with PICC dependent data addresses.

4.2.1.1 Data Integration in Systems with Memory Block PICCs

Each block of memory has a defined capacity which is determined by an address in the memory. Some commands are provided to access the blocks to read and modify their content. Block access is usually protected by conditions that need to be satisfied by a mutual authentication between the PICC and the terminal using a pre-determined key. Occasionally, these access conditions can apply to a group of blocks at the same time.

Since CIPURSE V2 memory is organized in files as defined in the [ISO/IEC 7816-4], to port current data to CIPURSE V2, current data has to be reorganized from memory blocks to files.

The most straightforward way to integrate CIPURSE V2 into an AFC system that uses block memory PICCs is to create a file for each block. However, in CIPURSE, as well as in other file system PICCs, the number of files that can be created is limited and usually lower than the number of blocks available in block memory PICCs. This method often results in poor performance as many files have to be read and written in each transaction.

The recommended solution is grouping memory blocks into CIPURSE V2 files. These files can be Binary Files, Record Files or Value Record Files.

As illustrated in section 4.2.2, CIPURSE protects data using roles. This is the best approach for securing PICC data as all terminals in the system can be classified by their role in the system. The most common roles are Administration, Validation, Top Up and Read Only, however, not all have to be present in the current AFC system and additionally, others may exist. The same procedure is applicable in each case.

To facilitate the security integration, blocks are grouped into files by role write access. In other words, blocks that are written by terminals with a particular role are grouped. Normally, Administration role has permission to write all blocks, Top Up role can only write a subset of blocks, and only a subset of this subset can be written by the Validation role. Read Only role cannot write any block. Therefore, the following order should be adhered to:

1. Group all blocks where at least one byte is written by Validation terminals.
2. Amongst the remaining blocks, group all blocks where at least one byte is written by Top Up terminals.
3. Finally, the Administration group will be formed by all the remaining blocks.

This results in three groups that correspond to three files in CIPURSE, as illustrated in Figure 4-7.

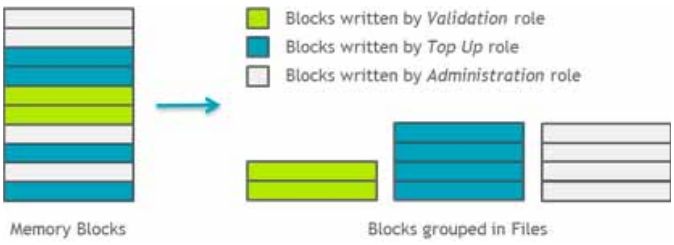

Figure 4-7 Block grouping by role write access

Block oriented PICCs commonly have two types of blocks: data blocks and value blocks. Data blocks store bytes whereas the value blocks store numeric values that can be increased and decreased with special commands. Hence, role block groups have to be split further into data block groups and value block groups to keep the increase and decrease value functionality, which can be achieved storing value block groups into CIPURSE V2 Linear Value-Record Files.

Data block groups may be stored in either Binary Files or Record Files. Binary Files are the preferred option in terms of performance, however, it has to be noted that CIPURSE L profile does not support the creation of Binary Files. If the same solution is to be used with all CIPURSE products, the Record File option should be selected. There are other reasons that tip the balance in favor of Record Files, such as having a block-like structure.

Binary Files

In this scenario, grouped data blocks are stored in CIPURSE V2 Binary Files. A Binary File is created for each data block group with a file size equal to the sum of all bytes of all blocks in the respective group. In the firmware, block number addresses have to be translated to file number and offset addresses.

The transaction performance can be increased by caching write block commands: instead of sending each write block command separately, they will be cached in the DAL until the last write block command is sent. Then, using an additional instruction, the FCMA informs the DAL that no more blocks have to be written. This instruction acts as a trigger for the DAL, which sends the CIPURSE write commands.

Record Files

At least three sub-scenarios are possible within the Record Files scenario:

Record Files with one record

This scenario is similar to the creation of Binary Files but using Record Files. A Linear Record File containing only one record is created for each data block group. The record size is equal to the sum of all bytes of all blocks in the respective group. In the firmware, block number addresses have to be translated to file number and offset addresses.

Note that offset parameter is not transmitted with READ_RECORD and UPDATE_RECORD commands. Therefore, this may cause reading or writing extra data if a record has to be read or written partially from an offset position (see READ_RECORD and UPDATE_RECORD commands in [CIPURSE_OpInt] for more information). For this reason, Binary Files may have better performance than Record Files.

Like in the Binary Files scenario, where possible, caching is recommended to increase the transaction performance.

Record Files with some records

If record files are to be used, a better approach with improved performance is to split block groups into sub-groups to skip the reading or writing of those sub-groups that are not used during a transaction. Thus, transaction time may be reduced.

This approach may be used if, for example, the PICC stores more than one ticket, and ticket data fields are replicated for each ticket. In this case, a sub-group may be created for each ticket and only data fields of the ticket being used needs to be read and written.

Therefore, a Linear Record File is created with a total number of records equal to the total number of sub-groups and a record size equal to the largest sub-group size.

As illustrated in previous scenarios, caching, if possible, is recommended to increase the transaction performance.

Record Files with one record for each block

This scenario simulates in the CIPURSE device, the block structure of the current memory block device. A Linear Record File with a total number of records equal to the number of blocks and a record size equal to the block size, is created for each group. Hence, the Record File acts as a memory block area storing all the blocks of the group.

In the firmware, block number addresses have to be translated to file and record number addresses.

This scenario simplifies the firmware implementation, since caching is not possible: each block read/update command is translated into a read/update record command. However, it has an inferior performance due to latency times of each command.

As said, block oriented PICCs usually have a special type of block called value block. This type of block contains a value that can be scaled up and down incrementally – incremented or decremented with special commands. CIPURSE V2 also has special record files that store values that can be incremented and decremented, with which the same functionality can be kept. Therefore, value blocks will be put into similar groupings albeit separately. Instead of storing them in Linear Record Files, they will be stored in Linear Value-Record files, and instead of their write access, they will be grouped by their scalability permissions.

Three types of counters stored in value blocks are usually found in AFC Systems:

Counters that can be decremented by the Validation role and decremented and incremented by the Top Up role and the Administration role, such as purses or trip counters. These are the most common counters found in AFC Systems (probably the only type to be found).

Counters that can be decremented by the Top Up role and decremented and incremented by the Administration role, such as top up counters to limit the number of top ups that can be performed on a PICC. Validation role cannot decrement or increment these blocks.

Counters that can be decremented by the Validation and Top Up roles, but only incremented by the Administration role, such as access counters (counters are used to track how many times the PICC has been accessed, enabling in the system to find lost transactions once all the data has been processed in the backend).

If the current AFC System has all this three types of counters, it will result in three CIPURSE Linear Value-Record Files. However, it is also possible for the current system to have fewer or more types of counters which results in less or more Value-Record Files (one for each type of value blocks).

Note that counters with inverted behavior (decremented key is used to increment the counter and vice versa) may be grouped into the same CIPURSE files by inverting its behavior in the DAL. In other words, counters that are decremented and counters that are incremented by the same role key may be grouped into the same CIPURSE file. The DAL inverts the behavior to provide the correct behavior to the FCMA.

4.2.1.2 Data Integration in Systems with File System PICCs

Data integration is straightforward in AFC systems where CIPURSE V2 has to be integrated using a File System PICC: replicate PICC Dedicated File (DF) and Elementary Files (EF) structure in CIPURSE V2, taking into account the file types and the access conditions. The most common file types used are Binary Files, Linear and Cyclic Record Files and Value Files. CIPURSE V2 implements all these file types so by creating a file of the same type will maintain functionality.

Finally, the DAL in the firmware will have a hash table to convert application IDs and file IDs from current PICC to CIPURSE V2.

4.2.1.3 Data Integration in New Systems

The previous two sections describe the gold standard of integrating CIPURSE V2 into existing AFC Systems with little effort. However, it may not be the best solution for all systems (for example in newly designed systems or if the current system is to be redesigned), where other aspects may have to be taken into account, such as PICC and terminal performance, backend performance, effective use of the on-PICC resources or such like.

For new systems using more than one PICC type, the data structure has to be designed without PICC structure influence. Data fields are grouped only by their use in the system (PICC Application Data, User Profile Data, Fare Data, History Data, and such). Once the PICC independent data structure is designed, it is mapped to each PICC type memory structure. Therefore, if a new PICC type is to be added in the system, a new map is created and added to the DAL. PICC type mappings have to take into account data access conditions, and PICC specific features in order to make the system more efficient and secure. For example, groups of data fields with different concepts can be mapped to the same file if they have the same access conditions. Therefore, reading and writing files will take less time.

Every AFC system has its own characteristics that can make its DAL design very different from another AFC system. Since there is not a ‘one size fits all’ DAL, each case must assessed and may be implemented in different ways.

4.2.2 Security Integration

CIPURSE V2 has integral state-of-the-art security and has many improvements over existing PICCs on the market. Although integrating CIPURSE V2 into an existing AFC system will improve its level of security, it will be necessary to implement CIPURSE V2 security in system terminals, which includes the authentication algorithm, the secure messaging protocol and the access rules.

CIPURSE V2 security can be implemented as software in the firmware of the terminals. The CIPURSE Evaluation Kit provides example code for such implementation. However, if terminals have the space for a SAM module and some minimum requirements (see [CIPURSE_SAM]) are satisfied, OSPT recommends CIPURSE SAM is used in every terminal. Nevertheless, other secure elements can be used where CIPURSE SAM does not fit the requirements, such as remote secure elements or SAMs with extra functionalities, such as multi-PICC support (see Figure 4-5 and Figure 4-6).

CIPURSE SAMs can securely store keys and are protected against attacks, preventing keys from being stolen and disabling unauthorized PICC modification or PICC replication. In addition, CIPURSE SAMs have already implemented all CIPURSE V2 cryptographic operations saving development time and making CIPURSE security integration easier. This type of implementation is tested and certified and as a result is far less prone to errors.

To integrate CIPURSE security, a SAL has to be implemented. The SAL will make the FCMA independent from the security keys and algorithms of the PICC being used. The SAL will perform all cryptographic operations either using SAMs or by software, making the use of a SAM module transparent for the FCMA.

One advantage of using more than one PICC type is that, if the security of one of the PICCs has been breached, PICCs of this type can be replaced by the other PICC types on the system. However, this is only possible if different PICC types have different keys. It is therefore good practice to generate a different key set for each PICC type. When integrating CIPURSE into an existing system, a new key set has to be generated and used only with CIPURSE PICCs.

In CIPURSE, each key corresponds to a role, e.g. Validation, Top Up, Administration or Read Only. Each key's (each role) rights are defined in the Access Rights Table (ART) attribute for each file (either the Application Dedicated Files or EF). ART table has, for each key, an entry of one byte long where each bit enables or disables the execution of a certain group of commands. For example, the Validation key has a byte in the ART table of each file that specifies what commands can be executed after authentication with it. Or, a Linear Value-Record File, where the five first bits of the access byte enable or disable the execution of the following commands: READ_RECORD, UPDATE_RECORD, READ_VALUE, DECREASE_VALUE and INCREASE_VALUE. See [CIPURSE_OpInt] for further information.

This is the best approach for securing PICC data and it is the main trend in new AFC systems. Access to files is defined accurately and clearly for each role in the system, this enhances and facilitates the design of the security architecture and reduces possible design mistakes.

Since this access rights concept is not common in existing AFC systems, it has to be taken into account both in the SAL and in the DAL, as described in section 4.2.1.1.

4.2.3 RF Layer Update

An update of the RF layer is only needed if the firmware of terminals does not fully support ISO/IEC 14443: Parts 1, 2, 3 and 4. Although not necessary, this may occur in AFC systems using PICCs that support block oriented PICCs only. For example, MIFARE Classic and MIFARE Ultralight PICCs do not support Part 4, and FeliCa PICCs do not support ISO/IEC 14443 at all. In those systems the firmware of all terminals has to be checked to ensure they fully support ISO/IEC 14443 including Part 4.

Over time, more and more PICC types fully support ISO/IEC 14443 and integrators are implementing it into their terminals. If current terminals do not fully support this standard, the system integrator may already have a firmware update that can be installed in the existing terminals adding support for all parts of ISO/IEC 14443.

If the AFC system is already using Calypso, MIFARE Plus Level 3 or MIFARE DESFire PICCs, RF layer modifications will not be necessary since Calypso and DESFire are already compliant with ISO/IEC 14443-4.

4.2.4 Integration Roadmap

4.2.4.1 Business case for integration

The first stage of a CIPURSE Integration Roadmap should be to explore successful business cases. CIPURSE provides a clear business case in the medium term, such as encouraging multiple supply chains, mitigating potential future security weaknesses and so on; however, these strategic goals will not generate new revenue as the integration work will represent a cost.

Therefore, it is often advisable to explore ways to leverage the value of the transport card base as a source of new revenue and tie integration work into these investments, as a straight forward effective protection of such investments. This is discussed in more detail in OSPT's executive white paper titled 'Shifting the Ticketing Paradigm' (http://www.osptalliance.com/resources).

4.2.4.2 Development

This is the study, design and implementation of all the required modifications in the firmware of the terminals as well as the study and design of the CIPURSE V2 PICC application.

If there are several system integrators with different terminal models, all the firmware of all the different terminals should be redesigned and modified. The CIPURSE V2 PICC application must be designed only once and given to all system integrators and this is usually done by the central transport authority.

This phase includes lab tests to ensure that any modifications work as expected. If there are several system integrators, the terminals of all the integrators must pass the tests; and tests must include interoperability between different terminal models i.e. all terminal generated data should be interpreted correctly in all other terminals.

During this phase, the current AFC system is kept intact.

4.2.4.3 Pilot

In order to validate and test the modifications in a controlled real world environment, it is strongly recommended that a pilot is undertaken. The pilot should be conducted in tandem with the original system – perhaps a specific route with real terminals installed in selected buses and stations with customers that are briefed and understand the nature of the pilot.

Another option is to implement a migration product that supports both the legacy and new CIPURSE functionality. This is often easier as PICC can be operated as CIPURSE PICC in the pilot environment and as legacy PICC in the existing system. Consequently, the traveler does not need to be trained on the pilot.

Since both PICC environments can be used in the pilot, the new PICC is commonly subsidized and incentivized to encourage its use.

4.2.4.4 Deployment

Deployment consists of two phases:

1. Installation of the new firmware in all terminals on the system without using any CIPURSE PICC at this stage. Since the new firmware works with old PICCs, this process is transparent to the customer and can be phased in gradually.

2. Once the firmware on all the terminals is upgraded, CIPURSE PICCs can be rolled out into the CIPURSE enabled infrastructure. As the infrastructure integrates both the legacy and new CIPURSE PICCs, the transit authority is free to decide on the mix of PICC types and is able to react to issues as they arise, for example, supply shortage, increase of price and security weaknesses/breaches.

Once the firmware of all terminals is upgraded the system will be ready for CIPURSE use. Only then can CIPURSE PICCs be issued.

It is important to note that deploying PICCs that support both CIPURSE and the legacy system while installing the new firmware can reduce the transition phase significantly.

4.2.4.5 Using CIPURSE V2

Once the system is CIPURSE-ready, the CIPURSE products will respond to different criteria such as economical, security, functional or strategic. Depending on these criteria, will dictate the rate that CIPURSE is phased in. Wherever possible, its implementation is gradual so as to have the least impact on the public transportation ecosystem.

Economic criteria are those that save and reduce costs by using CIPURSE V2. For example, if a CIPURSE PICC is cheaper than the existing PICC, CIPURSE PICCs will be introduced gradually as current PICCs expire or break. Using this method, customers are unaware of any changes as the system can manage both PICCs at the same time.

Security criteria are, for example, if a current PICC's security is breached. When a PICC becomes vulnerable, fraud may increase and this can result in loss of revenue. In this case, immediate replacement of the affected PICCs may be justified.

As CIPURSE V2 offers an upgraded and robust security, replacement of the PICCs should start with the most expensive fares in order to prevent fraud on those fares that may incur the highest revenue losses.

Functional criteria are those that take advantage of CIPURSE V2 functionality. For example, CIPURSE V2 can establish a secure channel between CIPURSE V2 PICC and a remote secure server through an insecure channel such as internet, and an insecure terminal such as a mobile phone. This enables new services for the traveler such as topping up or purchase of tickets via the internet.

Other criteria may be strategic. For example, it is a good practice not to get locked into one chip manufacturer as working with several manufacturers promotes competition and lowers the chip price, while also offering protection against security breaches and vulnerabilities. The proportions of different PICCs varies over time, but a minimum percentage of each PICC is always kept in order to ensure that all PICCs are working correctly on the system.

5 Use Cases

This section highlights use cases for the most common PICCs. The following table is an outline of each use case.

	Data integration	SAM Socket available	RF Layer update	Recommended CIPURSE Level
MIFARE Classic MIFARE Plus L1 MIFARE Plus L2	Blocks Files	Not often	Probably needed	CIPURSE S
MIFARE Plus L3	Blocks Files	Often	Not needed	CIPURSE S
MIFARE DESFire	Files Files	Very often	Not needed	CIPURSE T
MIFARE Ultralight	Blocks Files	Not often	Probably needed	CIPURSE L
Calypso	Files Files	Always	Not needed	CIPURSE T
FeliCa	Blocks Files	Very often	Probably needed	CIPURSE S

Table 5-1 Use cases in brief

5.1 MIFARE Classic and MIFARE Plus Level 1 and 2

MIFARE Classic is a PICC with a memory that is structured in arrays of 16 bytes blocks. The blocks are grouped in sectors of four blocks – in the first two kilobytes of memory and any remaining in sectors of 16 blocks in the rest of the memory. Each sector uses one block called sector trailer to store two keys, key A and key B, which are used in the authentication process and to encrypt and decrypt data. This sector trailer block also stores the access conditions of the sector.

Depending on the access conditions, an authentication with key A or key B is needed to operate with the sector blocks. They can be configured to allow (or not) reading and/or writing of the blocks in the sector, including the sector trailer block.

Once authenticated, MIFARE Classic encrypts and decrypts transmitted data using CRYPTO-1 algorithm, which is a NXP proprietary algorithm. It is of public knowledge that copying or modifying MIFARE Classic PICCs is possible without knowing the keys and using cheap PC/SC readers so the use of these PICCs in an AFC system is a risk and a migration (instead of integrating) of PICCs is recommended1.

MIFARE Classic is [ISO/IEC 14443-3] compliant but not [ISO/IEC 14443-4]. Therefore, to integrate CIPURSE V2 into these systems' terminals, the firmware must have [ISO/IEC 14443-4] protocol implemented, and if it isn't, an update must be carried out.

MIFARE Plus is an upgrade of MIFARE Classic PICCs. In addition to the MIFARE Classic functionality, it comes with some levels of security resulting in a different behavior depending on the level configured.

MIFARE Plus PICCs in Level 1 are also compatible with MIFARE Classic PICCs. The only functional difference is that MIFARE Plus Level 1 PICCs can execute an extra authentication using AES instead of CRYPTO-1, but this authentication is not combined with any session key within the transaction. Therefore, this authentication can only be used to check if a PICC actually belongs to the system. It does not prevent modifying a PICC or copying it to another PICC that also belongs to the system without knowing the keys.

MIFARE Plus PICCs in Level 2 use AES for the authentication and CRYPTO-1 for the encryption of data, using a dynamic key generated in the authentication process.

Since MIFARE Classic and MIFARE Plus Level 1 and 2 have the same data structure and they all are only [ISO/IEC 14443-3] compliant, the same integration process is valid and this is reflected in the same use case.

5.1.1 Data integration

Both MIFARE Classic and MIFARE Plus PICCs (in all levels) have their memory structured in 16 byte blocks, which in turn are grouped into sectors of or 16 blocks depending on their address in the memory.

As described in 4.2.1.1, the recommended solution to port this memory structure to CIPURSE V2 is grouping blocks with the same write access conditions into CIPURSE V2 files.

The procedure to follow is:

1. Make a list of all keys used in the system.
2. Group all keys used to write blocks by validator terminals. These keys will be referred to as validation keys.
3. Amongst the remaining keys, group all keys used to write blocks by top up terminals. These keys will be referred to as top up keys.
4. All remaining keys should be grouped and referred to as administration keys.
5. Ignore all sector trailer blocks.
6. Separate blocks in data blocks and value blocks.
7. Group all data blocks that can be written with any of the validation keys. These blocks will be referred to as validation data blocks.
8. Amongst the remaining blocks, group all data blocks that can be written with any of the top up keys. These blocks will be referred to as top up data blocks.
9. All the remaining data blocks should be grouped and referred to as administration data blocks.
10. Group all value blocks that can be decremented (but not incremented) with any of the validation keys and incremented by all other keys (validation and administration keys). These blocks will be referred to as validation decremented value blocks.
11. Group all value blocks that cannot be decremented or incremented with any of the validation keys, decremented (but not incremented) by any of the top up keys and incremented by administration keys. These blocks will be referred to as top up decremented value blocks.
12. Group all value blocks that can be decremented (but not incremented) with any of the validation or top up keys and incremented by administration keys. These blocks will be referred to as validation and top up decremented value blocks.
13. In CIPURSE PICC, create an ADF with four keys – Administration, Top Up, Validation and Read Only keys.
14. In this ADF, create a file to store validation data blocks. As explained in section 4.2.1.1, different scenarios are possible but only one should be selected:
    - Create a Binary File with its size equal to the total number of bytes of the validation data blocks.
    - Create a Record File with only one record with its size equal to the total number of bytes of the validation data blocks.
    - Create a Record File with a number of records equal to the number of generated sub-groups and with a record size equal to the largest sub-group size.
    - Create a Record File with a number of records equal to the number of validation data blocks with a size of 16 bytes, so a MIFARE-like structure is simulated. This file will act like a big MIFARE sector containing all the validation data blocks.
    - The ART table of this file must be set so that Administration, Top Up and Validation keys can read and update records (0xC0) and Read Only key can read records (0x80).

15. Similarly, create a file to store top up data blocks. The ART table of this file must be set so that Administration and Top Up keys can read and update records (0xC0) and Validation and Read Only keys can only read records (0x80).

16. In a similar process described in point 14, create a file to store administration data blocks. The ART table of this file must be set so that only Administration key can read and update records (0xC0) and Validation, Top Up and Read Only keys can only read records (0x80).

17. If validation decremented value blocks exist, create a linear value-record file to store them. The number of records will equal the number of validation decremented value blocks. The ART table of this file must be set so that Administration and Top Up keys can read and update records, and read, decrease and increase values (0xF8); Validation key can read records, and read and decrease values (0xB0); Read Only key can only read records and values (0xA0).

18. If top up decremented value blocks exist, create a linear value-record file to store them. The number of records will equal the number of top up decremented value blocks. The ART table of this file must be set so that only Administration key can read and update records, and read, decrease and increase values (0xF8); Top Up key can read records, and read and decrease values (0xB0); Validation and Read Only keys can only read records and values (0xA0).

19. If validation and top up decremented value blocks exist, create a linear value-record file to store them. The number of records will equal the number of validation and top up decremented value blocks. The ART table of this file must be set so that only Administration key can read and update records, and read, decrease and increase values (0xF8); Validation and Top Up keys can read records, and read and decrease values (0xB0); Read Only key can only read records and values (0xA0).

20. For each EF, Secure Messaging Rules (SMR) will be set to ENC'ed (10B) for both command and response and for all commands if confidentiality of all data is to be kept. That is SMR=1010 1010 0000 0000B. If the efficiency is to be improved, the recommendation is to set SMR to MAC'ed for those commands where confidentiality is not needed or even SM_PLAIN where permissible.

21. The following figure shows the resulting CIPURSE application.

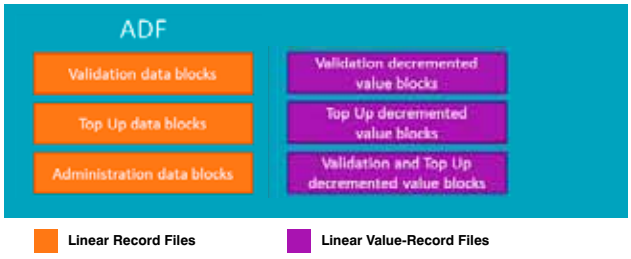

Figure 5-8 Resulting CIPURSE application in MIFARE Classic use case

Finally, the DAL in the firmware will have a hash table to convert MIFARE block numbers to CIPURSE file addresses, which may be different depending on the selected scenario for points 14, 15 and 16 described above. This means that, address of block x: addrx, is converted to address of file y and record z, addry,z. and command caching shall be applied if applicable (see section 4.2.1.1).

The following example illustrates this process in a common MIFARE application. A common MIFARE application could be one with the following features:

128 bytes of data divided into eight blocks used to store some general information, such as fare owner, PICC number and PICC type. All blocks can be written with key k2 and read with key k1.

144 bytes of data divided into nine blocks used to store media information, such as expiry date, maximum allowed users and transport validity area. All blocks can be written with key k3 and read with key k1. These 144 bytes are mirrored into a further nine blocks with the same access conditions.

192 bytes of data divided into 12 blocks used to store historical information, such as last validation time, and last validation point. All blocks can be written with key k4 and read with key k1. These 192 bytes are mirrored into a further 12 blocks with the same access conditions.

1 value block to store purse data. This value block can be increased with key k3 and decreased with key k4. This value block is mirrored into another value block with the same access conditions.

Note: k1– k4 are the names given to all the different keys the current MIFARE system has. In MIFARE PICCs, these keys are stored in the sector trailer bytes of each sector, having each sector capacity to store two keys (Key A and Key B). In this example, key k1 is the key used to read the blocks of each sector, which implies that this key is stored as Key A or B for all sectors, and consequently the access control bits are configured to allows the blocks to be read with this key.

This results in the following CIPURSE S application using the Binary Files scenario:

One Binary File of 128 bytes that can be written with the key ka and read with the key kr. This file stores administration data blocks. Key ka is the Administration key and corresponds to MIFARE k2. Key kr is the Read Only key and corresponds to MIFARE k1.

One Binary File of 144 bytes that that can be written with the key kt and read with the key kr. This file stores top up data blocks. Key kt is the Top Up key and corresponds to MIFARE k3. Key kr is the Read Only key and corresponds to MIFARE k1.

One Binary File of 192 bytes that can be written with the key kv and read with the key kr. This file stores validation data blocks. Key kr is the Top Up key and corresponds to MIFARE k4. Key kr is the Read Only key and corresponds to MIFARE k1.

One Linear Value-Record File with two value records (validation decremented value records) that can be read, updated, increased or decreased with the keys kt and ka, read or decreased with the key kv, and just read with key kr.

Key ka corresponds to the Administration key, kt to the Top Up key, kv to the Validation key and kr to the Read Only key.

Regarding data commands abstraction, the following should be considered.

5.1.1.1 Selection

In MIFARE, PICCs do not have the capability to handle Applications, so after the anti-collision process the PICC is ready to accept authentication, read and write commands. Since CIPURSE has ADF there is an extra step in the PICC selection process. Besides the anti-collision process described in [ISO/IEC 14443-3] the ADF containing the application must be selected. This is performed with the SELECT command.

It is important to note that all CIPURSE V2 PICCs feature an advanced selection system used for efficient application selection. This feature consists of a special ADF called (Proximity Transport System Environment (PTSE) with an AID=A0000005070100H the sole purpose of which is to provide a response to SELECT that indicates the transit applications available on the PICC. Furthermore, it can be configured to implicitly select a default application after the selection of the PTSE application. See [CIPURSE_OpInt] for more details.

5.1.1.2 Authentication

To perform a mutual authentication at least three messages have to be exchanged. Consequently, at least two pairs of command-response have to be sent. However, in MIFARE Classic PICCs, the authentication process must be managed by the NXP RF chip as CRYPTO-1 algorithm is proprietary. From the firmware side, this process is seen as only one command, commonly called authentication command.

In CIPURSE, to perform the authentication process, two commands have to be sent: GET_CHALLENGE and MUTUAL_AUTHENTICATE (see [CIPURSE_CryProt] for more details).

5.1.1.3 Secure Messaging

After the authentication, secure data exchange in CIPURSE is performed through secure messages that can be used in data integrity protection mode or in confidential communication mode. Depending on the chosen mode, extra operations will be performed by the security module on each command. See [CIPURSE_CryProt] for more details of these operations.

5.1.1.4 Read

Depending on the selected scenario, Binary or Record Files, MIFARE Read commands have to be translated to READ_BINARY or READ_RECORD respectively, converting block numbers to file number + offset in the scenarios of “Binary Files” and “1 record Record File” or file number + record number in the scenario of “some records Record File” and “16 bytes records Record File” (MIFARE-like Record File scenario).

5.1.1.5 Write

Similar to the above, depending on the selected scenario, Binary or Record Files, MIFARE Write commands have to be translated to UPDATE_BINARY or UPDATE_RECORD respectively, converting block numbers to file number + offset in the scenarios of “Binary Files” and “1 record Record File” or file number + record number in the scenario of “some records Record File” and “16 bytes records Record File” (MIFARE-like Record File scenario).

As explained in section 4.2.1.1, caching write block commands is recommended whenever possible.

5.1.1.6 Increment/Decrement

In MIFARE, two commands have to be sent to increment or decrement the value of a value block: Increment + Transfer or Decrement + Transfer respectively. The equivalent process in CIPURSE is performed with just one command: INCREASE_VALUE or DECREASE_VALUE respectively, which implies caching the Increment/Decrement commands and actually send the INCREASE_VALUE and DECREASE_VALUE command when Transfer command is requested by FCMA.

Again, block numbers have to be converted to file and record numbers.

5.1.2 Security integration

As explained in section 4.2.2, OSPT recommends the use of SAM together with CIPURSE V2 PICCs. MIFARE Classic PICCs are commonly used without SAM. For this reason the AFC system terminals are unlikely to have physical sockets for SAM modules. In some systems it may be possible to add a simple PCB that provides add-on SAM slots to the existing device.

If the use of a secure module (CIPURSE SAM, custom SAM, custom secure module, etc.) is not possible, CIPURSE security algorithms may be implemented in the firmware of the terminals, though this is not recommended by OSPT Alliance.

If CIPURSE security algorithms are implemented in the firmware, keys will probably be stored in the memory of the terminals, as is usually the case with MIFARE PICCs. It is important to note that without properly protecting the terminal, the system could be compromised and vulnerable to attacks.

To make the FCMA independent of PICC security, a SAL will be implemented in the firmware that will perform the corresponding cryptographic operations depending on the PICC that is being used. This SAL also handles the new role-based key concept, where each role has a CIPURSE application key instead of having multiple MIFARE sector keys.

5.1.3 RF Layer update

MIFARE Classic and MIFARE Plus Level 1 and 2 are not compliant with [ISO/IEC 14443-4] although it does not mean that terminals have not implemented this protocol. It may be that some terminals in the systems have it implemented while others do not.

Firmware of terminals that do not implement [ISO/IEC 14443-4] have to be upgraded to allow interaction with CIPURSE V2 products.

5.1.4 CIPURSE recommended profile

In MIFARE Classic and MIFARE Plus Level 1 and 2 PICCs, commands are always encrypted. It is not possible to configure the PICCs to exchange data in plain mode or with a MAC attached. It is common as well as best practice to use the encrypted mode only for confidential data, as it adds complexity and computation time to operations. MAC mode is enough for sensitive data but not confidential data, and plain mode is used for non-sensitive data, reducing the computation cost.

If confidentiality is to be maintained, CIPURSE's recommended profile for MIFARE Classic and MIFARE Plus Level 1 and 2 is CIPURSE S. The most suitable product amongst all CIPURSE S compliant products should be chosen, taking into account the amount of memory required. CIPURSE S specifies a minimum of one KB of available memory for data.

In some AFC systems using MIFARE Classic, data confidentiality may not be needed, despite this a minimum of security is necessary to restrict PICC access. In those systems CIPURSE L profile can be used, since it uses MAC to restrict file accesses. Other requirements have to be checked because CIPURSE L products may have less memory, less number of files and less keys available than CIPURSE S products. Also note that CIPURSE L does not support Binary Files; consequently, the scenario described in section 4.2.1.1 is not possible.

CIPURSE profiles are upwards compatible. That means that PICCs can be upgraded to CIPURSE T products without making changes in the system other than those related to adding the consistent transaction mechanism.

CIPURSE T products relate to transactions. A transaction is defined as a sequence of commands which perform updates on several EFs within one DF. CIPURSE T provides a consistent transaction mechanism that ensures that all updates in a transaction are performed automatically. CIPURSE T can be used in future improvements of the AFC system.

5.2 MIFARE Plus Level 3

MIFARE Plus Level 3 data structure is the same as MIFARE Classic, but its security is completely different and the communication protocol is [ISO/IEC 14443-4] compliant.

Regarding security, MIFARE Plus Level 3 does not use CRYPTO-1, but instead uses AES for both authentication and data encryption. The sector trailer is kept for the access conditions but because AES keys are 16 bytes long, key A and key B are stored in a different area of the memory. Sector trailer bytes corresponding to CRYPTO-1 keys are not used.

5.2.1 Data integration

Since the memory organization of MIFARE Plus Level 3 is the same as MIFARE Classic, follow the same guidelines described in section 5.1.1.

5.2.2 Security integration

Security in MIFARE Plus Level 3 PICCs is different from MIFARE Classic or even from MIFARE Plus Level 1 and MIFARE Plus Level 2 PICCs. The main difference is that CRYPTO-1 algorithm is not used and AES is used instead for both authentication and encryption.

Contrary to MIFARE Classic, terminals supporting MIFARE Plus Level 3 tend to use SAM both as key store and to perform cryptographic operations. In most cases, terminals have SAM slots available in these AFC systems.

To make the FCMA independent of PICC security, a SAL will be implemented in the firmware that will perform the corresponding cryptographic operations and will access either secure module depending on the PICC type being used, making the PICC type being used transparent to the FCMA layer.

To take the maximum advantage of having several PICC types, CIPURSE PICCs will use a new keyset, different from the MIFARE Plus keysets.

Similar to the MIFARE Classic use case, where the use of secure modules is not possible, although it is not recommended by OSPT, CIPURSE security algorithms may be implemented in the firmware of the terminals. Keys would be stored in the memory of the terminals, as is usually the case with MIFARE PICCs. It is important to note that the system could be compromised and vulnerable to attacks.

5.2.3 RF Layer update

MIFARE Plus Level 3 PICCs are already compliant with [ISO/IEC 14443-4].

5.2.4 CIPURSE recommended profile

As illustrated in the MIFARE Classic use case, the CIPURSE recommended profile for most systems using MIFARE Plus Level 3 is CIPURSE S. For the same reasons as before, as transmitted data in MIFARE PICCs is encrypted and the most basic CIPURSE profile with encryption feature is CIPURSE S.

The most suitable product amongst all CIPURSE S compliant products should be chosen, taking into account the amount of memory required. CIPURSE S specifies a minimum of one KB of available memory for data.

It should be noted again that CIPURSE profiles are compatible with newer versions. This means that PICCs can be upgraded to CIPURSE T products without making changes to the system.

5.3 MIFARE DESFire

5.3.1 Data integration

MIFARE DESFire PICCs, contrary to the rest of MIFARE PICCs, use a file system structure that is a subset of the [ISO/IEC 7816-4]. It has a Master File (MF) that can contain several DF that in turn can contain several EF.

Each DF has a maximum of 14 keys that can be used for both, application management and EF access conditions. Each EF has its own access conditions.

Although DESFire D40 was replaced by DESFire EV1, both have the same memory organization. Therefore, the same guidelines are valid for both PICCs and are those described in section 4.2.1.2, specific to DESFire PICCs.

For each DF in DESFire PICC an ADF will be created in CIPURSE V2 and for each EF in each DF an EF will be created in the corresponding CIPURSE ADF.

DESFire has five types of EF: Standard Data File, Backup Data File, Value File, Linear Record File and Cyclic Record File. For each file type, CIPURSE has its equivalent type that can be found in the following table. File type identifiers are enclosed in parentheses.

DESFire	CIPURSE
Standard Data File (0x00)	Non-Transactional Binary File (0x01)
Backup Data File (0x01)	Transactional Binary File (0x11)
Value File (0x02)	Transactional Linear Value-Record File (0x1F)
Linear Record File (0x03)	Transactional Linear Record File (0x12)
Cyclic Record File (0x04)	Transactional Cyclic Record File (0x16)

Table 5-2 DESFire – CIPURSE file equivalence

In spite of the fact that file types have almost the s ame functionality in both PICCs, there are some differences that should be highlighted.

DESFire Value Files can only contain one value while CIPURSE Linear Value-Record Files are record files that can contain N values. This allows grouping several DESFire Value Files into one CIPURSE Linear Value-Record File.

Another difference lies in the creation of Cyclic Record Files. In DESFire they have to be created with N+1 records, where N is the number of records used by the application. One more record is needed to be specified and it will be used for the backup mechanism. In CIPURSE this is not necessary and the number of records is specified with N in the creation of Record Files.

Access conditions are specified differently in DESFire and CIPURSE. In DESFire each file has its own Access Rights specified by four nibbles (two bytes). Each nibble represents a certain permission: read, write, read and write, change rights respectively. To gain one of these permissions, an authentication using the key specified by the nibble bits needs to be performed. CIPURSE access rights are different. They are controlled with the ART (Access Rules Table). Each file has an ART, and for each key, eight bits are used to specify the command group(s) enabled on successful authentication with this key (see [CIPURSE_OpInt] for detailed information).

Also the way of setting the communication mode is different. In DESFire, each file is set using the eight bit parameter ComSettings, that specifies if the communication is plain, MAC'ed or encrypted. In CIPURSE it is set using the SMR (Secure Messaging Rules) which specify the command and response of each command group if the message is to be transmitted in plain but with secure messaging, MAC'ed or Encrypted.

Finally, the DAL in the firmware will have a hash table to convert DESFire application and file IDs to CIPURSE application and file IDs. Note that application and file ID formats are different between these PICC types.

Regarding applicable commands, on the one hand CIPURSE uses APDU, as defined in [ISO/IEC 7816-4], to send commands and the command set is a subset of [ISO/IEC 7816-4] command set. On the other hand, both DESFire D40 and DESFire EV1 use native commands with proprietary structure. EV1 offers a limited set of ISO commands and supports encapsulation of native commands into ISO command structure using CLA=0x90.

As PICCs, DESFire and CIPURSE, use file systems as described in [ISO/IEC 7816-4], their command set is almost equivalent. Main DESFire native command equivalences are shown in the following table.

DESFire	CIPURSE
CreateApplication	CREATE_FILE
SelectApplication	SELECT
CreateFile	CREATE_FILE
CreateStdDataFile, CreateBackupDataFile, CreateValueFile, CreateLinearRecordFile, CreateCyclicRecordFile	CREATE_FILE
ReadData	READ_BINARY
WriteData	WRITE_BINARY
GetValue	READ_VALUE
Credit	INCREASE_VALUE
Debit	DECREASE_VALUE
WriteRecord	UPDATE_RECORD, APPEND_RECORD
ReadRecords	READ_RECORD
Commit	PERFORM_TRANSACTION
Abort	CANCEL_TRANSACTION

Table 5-3 DESFire – CIPURSE command equivalence

5.3.2 Security integration

DESFire PICCs are almost always used with SAMs. Most DESFire AFC systems terminals will have a SAM socket available for the CIPURSE SAM.

To make the FCMA independent of PICC security, a SAL will be implemented in the firmware that will perform the corresponding cryptographic operations and will access one of the secure modules depending on the PICC type being used, making the PICC type transparent to the FCMA layer.

In order to take maximum advantage of supporting several PICC types, CIPURSE PICCs will use a new keyset, different from MIFARE DESFire's.

5.3.3 RF Layer update

MIFARE DESFire PICCs are compliant with [ISO/IEC 14443-4]. However, if the RF layer has been implemented taking into account the DESFire specification only, it is possible that the frame size is limited to DESFire frame size (64 bytes). In this case, since CIPURSE supports the maximum [ISO/IEC 14443-4] frame size (256 bytes), protocol overhead can be reduced by extending frame size to 256 bytes. This results in improved transaction times when transferring/receiving large amounts of bytes to/from CIPURSE PICCs.

5.3.4 CIPURSE recommended profile

DESFire applications are transactional: changes in backup files are not performed until COMMIT command is sent. For this reason, the recommended profile for MIFARE DESFire systems is CIPURSE T.

References

[CIPURSE_OpInt]	OSPT Alliance: CIPURSE™V2, Operation and Interface Specification, Revision 1.1 / 2012-10-30
[CIPURSE_CryProt]	OSPT Alliance: CIPURSE™ V2, Cryptographic Protocol, Revision 1.0 / 2012-09-28
[CIPURSE_SAM]	OSPT Alliance: CIPURSE™ V2 SAM Specification Revision 1.0 / 2013-10-14
[ISO/IEC 14443-3]	ISO/IEC 14443 International Standard. Identification cards — Contactless integrated circuit cards — Proximity cards — Part 3: Initialization and anticollision
[ISO/IEC 14443-4]	ISO/IEC 14443 International Standard. Identification cards — Contactless integrated circuit cards — Proximity cards — Part 4: Transmission protocol
[ISO/IEC 7810]	ISO/IEC 7810 International Standard: Identification cards — Physical characteristics is an international standard that defines the physical characteristics for identification cards
[ISO/IEC 7816-4]	ISO/IEC 7816 International Standard: Identification cards — Integrated circuit cards; Part 4: Organization, security and commands for interchange. Edition 2005
[AES-128]	Advanced Encryption Standard (FIPS PUB 197) using keys of 128 bits

Terminology

ADF	Application Dedicated File
AFC	Automatic Fare Collection
AES	Advanced Encryption Standard
APDU	Application Protocol Data Unit
DES	Data Encryption Standard
DAL	Data Abstraction Layer
DF	Dedicated File
EF	Elementary File
FCMA	Fare Collection Management Application
MAC	Message Authentication Code
MF	Master File
PICC	Proximity Integrated Circuit Card
RFU	Reserved for Future Use
SAL	Security Abstraction Layer
SAM	Secure Application Module
SMR	Secure Messaging Rules
UID	Unique ID

Find out more

The OSPT Alliance is an international association chartered to provide a new open standard for secure transit fare collection solutions. It provides industry education, creates workgroup opportunities and catalyzes the development and adoption of innovative fare collection technologies, applications and services. The OSPT Alliance was founded by leading technology companies, and membership is open to technology providers, transit operators, consultants, solution vendors, government agencies and other stakeholders in the transit ecosystem. For additional information, please visit www.osptalliance.org.

www.osptalliance.org