2013 5th International Conference on Computational Intelligence and Communication Networks

# Efficient Data Structure Based Smart Card Implementation

Shalini Jain , Anupam Shukla
Indian Institute of Information Technology, Gwalior
Morena Link Road, Gwalior, India
Shalinijain58@gmail.com

Bishwajeet Pandey, Mayank Kumar
Embedded System Design R&D Lab
Centre for Development of Advanced Computing
Noida, India

***Abstract*—to make smart card much faster, we need efficient data structure. Access time of on chip data depends on how and where we stored. Some Data Structure take maximum time and some take minimum time depending on the space and time complexity of data structure. In this work, we have taken some data structures and find that BST is the best suitable data structure for performing smart card operations in compare to other possible data structures.**

***Keywords—Data Structure, Smart Card, RAM, Memory, File Management, Master File, Dedicated File, Elementary File, Personal Identification Number(PIN), ROM, Flash***

## I. INTRODUCTION

In real life, we deal with different types of smart card. Subscriber identification module (SIM) cards is the smart card which we use almost maximum time. From the usage point of view, in real life, debit card, atm card and credit card is the card used after the sim card. Third usage is commuter cards (like smart card in Delhi metro), and radio frequency identification (RFID) cards used in office and institute. Smart cards are small in size so it is easy to carry.They have enough processing power and sufficient data storage capabilities to store user profiles, to carry out cryptographic functions, and to support electronic commerce or other type of applications.

Data is organized in smart card in the form of files. File organization for data structure of smart card is of types, one is Dedicated file (DF) and another is Elementary file (EF). Then, these organized file can be referenced either by file identifier or path or short EF identifier (SFID) or DF name. EF again classified into two types, one is transparent elementary file and record elementary file. Data referencing methods are also described.

After detailed analysis of different data structures in term of the time taken by insertion, deletion and search operation, we have seen the BST is best for search operation and also optimal to insertion and deletion operation.

In the next section, we have discussed the related work done in the field of data structure of smart card. In section III, we discussed the available data structures and file organization of smart card and also focussed on the referencing methods. In section IV, we search for the most efficient data structure for insertion, deletion and search operations. Then we conclude our work in section V.

## II. RELATED WORK

The state of the art of unified read-write of the smart card of different data formats is presented in [1]. Based on the framework designing and system organizational structures of "card type determining-> function calling ->Data Conversion ->unified read-write", a data reading, writing and reception data management of the smart card of different data formats through PC is implemented in [1]. The experimental results on smart cards of water meter, such as the RF card, TM card and IC card show that the general read-write system is feasible, and the read-write efficiency is not affected, possessing good operating conditions and scalability in [1].System like Smart card is a portable media which store sensible data. The information protection is possible with personal identification number (PIN), or through finger-print or retina based biometrics. Algorithms and datastructures are developed in [2] to solve security problem. There is possibility of data damage in absence of anti-IFD-switch off function, a dual-data memory structure with check sum is used in [3], which guarantees the correctness and availability of the data in the card when a data disaster possible. Reference[4]suggests an alternative to improve the speed of the execution by using the improved data store mechanism and memory structure. [4]Stored Java Objects stored area in EEPROM into RAM toachieve high performance Java Card and is similar with memory management in Java System of PC environment. According to [5] data or applications in the form of documents which in the smart card are stored storage medium such as FLASH or EEPROM, COS through the smart card file system management and organization to achieve data and application storage and management. The file system of Smart Card in [5] includes three types of files which is master file MF (Master File), dedicated file DF (Dedicated File), and basic file EF (Elementary File). MF and DF known as catalog files, EF as the data file. MF is the entrance of the file cards. Each card has one and only one MF, also known as root, all other subfolders (grandson) of MF files. In addition to MF, all the files that contain subfolders are considered as DF. In the file tree, if the file node itself is a leaf node, which has no child nodes, then this file is known as EF. EF is the basic carrier of

978-0-7695-5069-5/13 $26.00 © 2013 IEEE
DOI 10.1109/CICN.2013.102

data in the card, which useful for applications that require data in the EF. Reference [5] classified according to the data structure, EF also includes a transparent binary file (Transparent EF), the linear Fixed-length record file (Linear fixed EF), circular recording document (Cyclic EF) and linear variable length record files (BER-TLV EF). According to [6], Smart card deployment is increasing pressure for vendor regarding security features and improvements in computing power to support encrypt-decrypt algorithms with bigger footprints in the smart card chips in the last decade. Typical applications described in [6] include subscriber identification module (SIM) cards (in telecommunications), micropayments (in financial transactions), commuter cards (in urban transportation systems), and identification (ID) cards. Although the share of cards used for identification applications (which we'll call smart ID cards) is relatively small within the overall smart card market, it's one of the fastest growing segments. Smart ID cards is used for physical access to secure facilities and logical access to IT systems (Web servers, database servers, and workstations) which is discussed in [6].

## III. Smart Card Data Structure and Referencing Methods

A data structure contains information on the logical structure of data as seen at the interface, when processing entersindustry commands for interchange.

### *1. File organization*

Current organization of smart card supports following two types of files, i.e. dedicated file (DF) and elementary file (EF).

Dedicated file (DF) - In a smart card data can be organized logically in dedicated files in the structural hierarchy. The root dedicated file is always known as the master file (MF). And MF is mandatory. Dedicated files other thanMF are optional.

Elementary file (EF) - The smart card can have following two types of EFs, One is Internal EF and other is Working DF. Those EFs which are designed for storing data processed by the card, i.e. the card management and control purposes analyzes and uses data are known as Internal EF. And, Working EF are those EFs that are intended for storing data not interpreted by the card, i.e. data to be used by the outside world exclusively.

### *2. File referencing methods*

Files, which are not implicitly selected, can be selected by one of the following:

- Referenced by file identifier – Any file could be referenced by a 2 byte length file identifier. Tounambiguously point a file, EF's and DF's, just below the given DF, should have different identifiers. Some of the reserved file identifiers are: 3F00 (Master file identifier), FFFF (for future use), 3FFF (refer referenced by path).
- Referenced by path – Path of a file (concatenation of file identifiers, starting from MF/current DF, till the file's identifier itself) contains consecutive parent DFs. 3FFF points current DF and can be used if it's identifier is unknown. Path of a file uniquely selects it from its MF or the current DF.
- Referenced by short EF identifier – Short EF identifier is a referencing method to uniquely identify a file. It consists of a 5 bit value which can hold values from 1 to 30. The SFI value '0' refers to the currently selected EF. IT cannot be used as a file identifier or in a path.
- Referenced by DF name – Any DF may have a unique DF name of length 1 to 16 bytes, which could be used to uniquely identify a file. DF name shall be unique within a card, to unambiguously select a DF by its name.

### *3. Elementary file structures*

Two types of elementary file structures are defined. The first is: at interface EF is interpreted as a sequence of data units i.e. transparent structure EF. The second is: at interface EF is interpreted as a sequence of individually identifiable records i.e. record structure EF. Attributes for EF's structured records are: Record size (fixed or variable), Organization of records (sequenced-linear or ring-cyclic). One of the following four methods which should be supported by the smart cards for structuring EF's are Transparent EF, Linear EF with fixed sized records, linear file with variable size records and Cyclic EF with fixed sized records.

### 4. *Data referencing methods*

Data from any smart card can be referenced either in the form of records or data objects or data units. Within anrecord structure EF, data is stored in a single continuous sequence of records. While within an transparent structure EF, data is stored in a single continuous sequence of data units. We can not reference a record or a data unit which is not present within the EF or which is out of scope of an EF, will give an error. Referencing of data from EF, numbering of records within an EF and size of data unitsall are dependent on EF. And different EFs may contain different type of these information or structure. Each smart card have ATR (Answer –TO-Reset) which provides information related to card. For referencing a record or an data unit, related EF must be selected (for performing operation on that EF) before selecting particular data unit or record. For which any of the file referencing methods can be used. After that particular record

or data unit can be referenced either by record numberor by setting some parameter bytes.

*4.1Record referencing*

Either with record identifier or with the record number, each and every record of any selected EF can be referenced. It is 1 byte field with unsigned integer values ranging from '01' to 'FE'. So, any file can have maximum 254 number of records. '00' is reserved for special purposes. 'FF' is Reserved ForUsed.Forreferencing by record identifier need to manage a record pointer.Card reset command, CREATE FILE command, SELECT FILE command and any other commands carrying a valid file identifier or short EF identifier or path to a specific file can affect the record pointer. Referencing by record number will not affect the record pointer.

Referencing by record identifier– Application provides record identifier for each record within an EF. If in a data field of a record is a SIMPLE-TLV data object, then the record identifier will be the first byte of the data object. Within anrecord structure EF, records may have the same record identifier, in that case data contained in the records may be used for discriminating between them.

When a reference is given with respect to record identifier, an indication will also be there for specifying the logical position of the record that may be either the first occurrence or the last occurrence or the next occurrence or the previous occurrence with respect to the record pointer:

- In each linear structure EF, all the records will be assigned the logical position in sequential manner. When insert or write an record within EF, the first record will be assigned in first logical position than next record in next logical position and so on. The records will be arranged in the order of creation of records within EF.
- In each cyclic structure EF, all the records will be sequentially assigned the logical positions in the opposite order. When insert or write an record within EF, it will be always assigned in the first logical position. That's why we will find first record at the last logical position and recently inserted record at the first position always.

These are some additional rules defined for linear structure elementary files and cyclic structure elementary files:

- The very first occurrence of an EFwill be the record present in the first logical position and with the specified identifier. Similarly the last occurrence will be the record present in the last position and with the specified identifier.
- If there is no current record within an EF, than first occurrence will be considered as a next occurrence. Also last occurrence will be considered as a previous occurrence.
- If there is a current record within an EF, than nearest record from specified identifier with the greater logical position than the current record, will be the next occurrence. Similarly nearest record from specified identifier with smaller logical position than the current record, will be the previous occurrence.
- The value '00' does not depend on the record identifier. It will refer to the first, last, next and previous record in the numbering sequence.

**Referencing by record number**–All the records within an record structure EF will be unique and in sequentialform:

- In each linear structure EF, all the records will be assigned at the logical position in sequential manner. When insert or write an record within an EF, it will be assigned logical position in the order of creation. Hence first created record is the first record.
- In each cyclic structure EF, all the records will be sequentially assigned the logical positions in the opposite order. When insert or write an record within an EF,the record will be sequentially assigned in the opposite order. Hence the most recently created record is the first record.

4.2 Data unit referencing

In Transparent Structure EF, data units could be referred by an offset as in command READ BINARY. It is limited to 8 or 15 bits, unsigned integer, as per the option in the command. First data unit of the EF offset value is 0. For every consecutive data unit offset is incremented by 1. Default data unit size is one byte, if not defined in the command APDU given by the card. A record structure EF may support data unit referencing and in the case when it supports data unit referencing, data units may contain some structural information along with data, like record numbers may be contained by linear structure EF. Within arecord structure EF, since storage order of the records in the EF is unknown, so data unit referencing may not provide the intended result.

## IV. Efficient Data Structures

Different data structure takes different time for insertion or deletion or search operation. In order to search efficient data structure to store data for handling with smart card operating system, we take array, linked list, doubly linked list, stack, queue, binary search tree, hash and Heap under consideration.

Table 1: Time Complexity of Data Structures for Insert Operation

| Insert | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Data Structures** | **Array** | **Linked List** | **Doubly Linked** | **Stack** | **Queue** | **BST** | **Hash** | **Heap** |

| | | | List | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Time Complexity** | O(1) | O(1), at the head | O(1), at the head | O(1) | O(1) | O(log n) | O(1) | O(log n) |
| | O(n), insert after require element. | O(n), at position by traveseing the linkedlist | O(n),at position by traveseing the linkedlist | | | | | |

In insertion operation, each and every data structure except BST and Heap takes almost same time. Stack, Hash and Queue will take less time to insert record in smart card. While BST and Heap takes O (logn) time. Then, we analyze the time required for deletion of record.

Table 2: Time Complexity of Data Structures for Delete Operation

| **Delete** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Data Structures** | **Array** | **Linked List** | **Doubly Linked List** | **Stack** | **Queue** | **BST** | **Hash** | **Heap** |
| **Time Complexity** | O(1), deletion by position | O(1), at the head, | O(1), at the head, | O(1) | O(1) | O(log n) | O(1) | O(log n) |
| | O(n), delete element | O(n), deletion at position by traveseing the linkedlist | O(n),at the position by traveseing the linkedlist | | | | | |

In deletion operation, each and every data structure except BST and Heap takes almost same time. Stack, Hash and Queue will take less time to insert record in smart card. While BST and Heap takes O (logn) time. Then, we analyze the time required for search of record.

Table 3:Time Complexity of Data Structures for Search Operation

| **Search** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Data Structures** | **Array** | **Linked List** | **Doubly Linked List** | **Stack** | **Queue** | **BST** | **Hash** | **Heap** |
| **Time Complexity** | O(n) | O(n) | O(n) | O(n) | O(n) | O(log n) | O(n) | O(n) |
| | O(log n), if binary search is performed and the array isalready sorted. | | | | | | | |

To access the data stored in smart card, we perform operation called search. In search operation, each and every data structure except BST takes O(n) time. While BST take less time to insert record in smart card i.e. O(logn).

The time taken by Smart card in performing operation is depend on the time taken by searching of records not on the time taken by inserting and deleting record.

Insertion of record and deletion of record is secondary concern for any smart card user. Smart card usage time is directly proportional to the time of searching of stored data in smart card. We have seen that BST is taking less time to search records stored in smart card and also it is good for insertion and deletion.

## V. CONCLUSION

Searching of record is primary concern for the time analysis of any smart card. Whereas, insertion of record and deletion of record is secondary concern for any smart card user. In traditional smart card application, Inserting and deleting record is done on the time of manufacturing by card developer. While, user uses search operation mostly. Here, we have seen the BST is best for search operation and also optimal to insertion and deletion operation.

## REFERENCES


[1]. Rang-ding Wang , Wei Wang ,"Design and implementation of the general read-write system of the smart card", 3rd IEEE International Conference on Ubi-media Computing (U-Media), 2010.

[2]. R Sanchez Reillo, "Securing information and operations in a smart card through biometrics", Proceedings of IEEE 34th Annual International Carnahan Conference on Security Technology, 2000.

[3]. Ming-Sheng Liu; Hui Liu; Yin-Hua Ma; Wen-Xiong Li, "Research on precautions against data disaster of logic security smart card", Proceedings of International Conference on Machine Learning and Cybernetics, 2004.

[4]. Yoon-Sim Yang; Won-Ho Choi; Min-Sik Jin; Cheul-Jun Hwang; Min-Soo Jung, "An Advanced Java Card System Architecture for Smart Card Based on Large RAM Memory", International Conference on Hybrid Information Technology, 2006.

[5]. Chen Yuqiang; Hu Xuanzi ; Guo Jianlan ; Liu Liang, "Design and implementation of Smart Card COS", International Conference on Computer Application and System Modeling (ICCASM), 22-24 October 2010.

[6]. Chandramouli, R.; Lee, P., "Infrastructure Standards for Smart ID Card Deployment", IEEE Security & Privacy, Volume:5, Issue:2, pp. 92-96, 2007