

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
29 June 2006 (29.06.2006)

PCT

(10) International Publication Number  
**WO 2006/069312 A2**

(51) International Patent Classification:  
**G06F 3/06** (2006.01)

San Jose, CA 95138 (US). **BARZILAI, Ron** [IL/IL]; 67 Meron Street, 25147 Kfar-Vradim (IL).

(21) International Application Number:  
PCT/US2005/046795

(74) Agents: **HSUE, James, S.** et al.; Parsons, Hsue & de Runtz LLP, 595 Market Street, Suite 1900, San Francisco, CA 94105 (US).

(22) International Filing Date:  
21 December 2005 (21.12.2005)

(81) Designated States (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:  
60/638,804 21 December 2004 (21.12.2004) US  
11/313,538 20 December 2005 (20.12.2005) US  
11/314,055 20 December 2005 (20.12.2005) US

(71) Applicant (*for all designated States except US*): **SAN-DISK CORPORATION** [US/US]; 140 Caspian Court, Sunnyvale, CA 94089 (US).

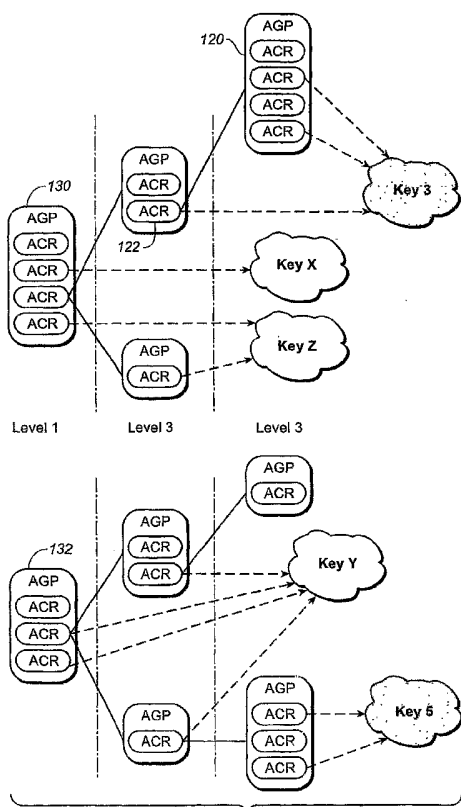
(72) Inventors; and

(75) Inventors/Applicants (*for US only*): **JOHAND-COULOMB, Fabrice** [FR/US]; 855 Buckland Avenue, San Carlos, CA 94070 (US). **HOLTZMAN, Michael** [IL/US]; 7602 Barnhart Place, Cupertino, CA 95014 (US). **QAWAMI, Bahman** [US/US]; 5899 Killarney Circle,

(84) Designated States (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: SYSTEM FOR CREATING CONTROL STRUCTURE FOR VERSATILE CONTENT CONTROL



(57) Abstract: The mobile storage device may be provided with a system agent that is able to create at least one hierarchical tree comprising nodes at different levels for controlling access to data stored in the memory by corresponding entities. Each node of the tree specifies permission or permissions of a corresponding entity or entities for accessing memory data. The permission or permissions at the node of each of the trees has a predetermined relationship to permission or permissions at nodes at a higher or lower or the same level in the same tree. Thus, the mobile storage devices may be issued without any trees already created so that the purchaser of the devices has a free hand in creating hierarchical trees adapted to the applications the purchaser has in mind. Alternatively, the mobile storage devices may also be issued with the trees already created so that a purchaser does not have to go through the trouble of creating the trees. In both situations, preferably certain functionalities of the trees can become fixed after the devices are made so that they cannot be further changed or altered. This provides greater control over access to the content in the device by the content owner. Thus, in one embodiment, the system agent can preferably be disabled so that no additional trees can be created.



**Published:**

— without international search report and to be republished  
upon receipt of that report

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

## **SYSTEM FOR CREATING CONTROL STRUCTURE FOR VERSATILE CONTENT CONTROL**

### **BACKGROUND OF THE INVENTION**

[0001] This invention relates in general to memory systems, and in particular to a memory system with versatile content control features.

[0002] The computing device market is developing in the direction of including content storage on mobile storage devices so as to increase the average revenue by generating more data exchanges. This means that the content in a mobile storage medium has to be protected when used on a computing device. Content includes valuable data, which may be data owned by a party other than the one that manufactures or sells the storage device.

[0003] One type of storage device with encryption capability is described in U. S. Patent No. 6,457,126. The capability provided by this device, is however, quite limited. It is therefore desirable to provide a memory system with more versatile content control features.

### **SUMMARY OF THE INVENTION**

[0004] The protection of content in a mobile storage medium can involve the encryption of data in the medium so that only authorized users or applications have access to keys used for encrypting data stored in the medium. In some prior systems, the key used for encrypting and decrypting data is stored in devices external to the mobile storage medium. In such circumstances, the company or individual who owns proprietary interest in the content may not have much control over the usage of the content in the medium. Since the key used for encrypting data in the medium exists external to the medium, this key may be passed from one device to another in a manner not subject to control by the content proprietor. The owner of proprietor interest will be in the better position to control access to the content in the medium if

the encryption-decryption key is stored in the medium itself and substantially inaccessible to external devices, according to one of the features of the invention.

[0005] By making the key essentially inaccessible from outside the medium, this feature provides portability to secured content. Thus, the storage device containing secured content ciphered with such a key can be used for access by a variety of host devices without the danger of security breach, since the device has exclusive control of access to the key. Only those host devices with the proper credentials are able to access the key.

[0006] To enhance the commercial value of the content stored in the mobile storage medium, it is desirable for the owner of proprietary interest in the content to be able to grant different permissions to different entities for accessing the content. Therefore another feature of the invention is based on the recognition that an access policy may be stored which grants different permissions (e.g. to different authorized entities) for accessing data stored in the medium. A system incorporating a combination of the two above features is particularly advantageous. On the one hand, the content owner or proprietor has the ability to control access to the content by using keys that are substantially inaccessible to external devices and at the same time has the ability to grant different permissions for accessing content in the medium. Thus, even where external devices gain access, their access may still be subject to the different permissions set by the content owner or proprietor recorded in the storage medium.

[0007] Yet another feature is based on the recognition that when the above-described policy, where different permissions are granted to different authorized entities, is implemented in a flash memory, this results in a particularly useful medium for content protection.

[0008] Many storage devices are not aware of file systems while many computer host devices read and write data in the form of files. According to another feature, the host device provides a key reference or ID, while the memory system generates a key value in response which is associated with the key ID, where the key value is used in cryptographic processing data in a file associated with the key ID. The host associates the key ID with the file to be processed cryptographically by the memory

system. Thus the key ID is used by the computing device and memory as the handle through which the memory retains complete and exclusive control over the generation and use of the key value for cryptographic processes, while the host retains control of files.

[0009] In some mobile storage devices such as smart cards, the card controller manages the file system. In many other types of mobile storage devices, such as flash memories, magnetic or optical discs, the device controller is not aware of the file system; instead, the device controller relies on a host device (e.g. a personal computer, digital camera, MP3 player, personal digital assistants, cellular phones) to manage the file system. The various aspects of this invention may be readily incorporated into such types of storage devices where the device controller is not aware of the file system. This means that the various features of this invention may be practiced on a wide variety of existing mobile storage devices without requiring a re-design of such devices to make the device controller in such devices become aware of and able to manage the file system.

[0010] A tree structure stored in the storage medium provides control over what an entity can do even after gaining access. Each of the nodes of the tree specifies permissions by an entity who has gained entry through such node of the tree. Some trees have different levels, where the permission or permissions at a node of the tree has a predetermined relationship to permission or permissions at another node at a higher or lower or the same level in the same tree. By requiring entities to comply with the permissions so specified at each of the nodes, the tree feature of this application allows a content owner to control which entities can take action, and which actions each of the entities can take, irrespective of whether the tree has different levels.

[0011] To enhance the commercial value that can be provided by the mobile storage medium, it is desirable for mobile storage devices to be capable of supporting more than one application simultaneously. When two or more applications are accessing the mobile storage device at the same time, it can be important to be able to separate the operations of the two or more applications so that they do not interfere with one another in a phenomena referred to herein as crosstalk. Therefore another feature of

the invention is based on the recognition that two or more trees which are preferably hierarchical may be provided for controlling access to the memory. Each tree comprises nodes at different levels for controlling access to data by a corresponding set of entities where a node of each tree specifies permission or permissions of the corresponding entity or entities for accessing memory data. The permission or permissions at a node of each of the trees has a predetermined relationship to permission or permissions at another node at a higher or lower level in the same tree. Preferably, there is no crosstalk between at least two of the trees.

[0012] From the above, it will be evident that trees are powerful structures that can be used for content security. One of the important controls provided is the control over the creation of trees. Thus, according to another feature of the invention, the mobile storage device may be provided with a system agent that is able to create at least one hierarchical tree comprising nodes at different levels for controlling access to data stored in the memory by corresponding entities. Each node of the tree specifies permission or permissions of a corresponding entity or entities for accessing memory data. The permission or permissions at the node of each of the trees has a predetermined relationship to permission or permissions at nodes at a higher or lower or the same level in the same tree. Thus, the mobile storage devices may be issued without any trees already created so that the purchaser of the devices has a free hand in creating hierarchical trees adapted to the applications the purchaser has in mind. Alternatively, the mobile storage devices may also be issued with the trees already created so that a purchaser does not have to go through the trouble of creating the trees. In both situations, preferably certain functionalities of the trees can become fixed after the devices are made so that they cannot be further changed or altered. This provides greater control over access to the content in the device by the content owner. Thus, in one embodiment, the system agent can preferably be disabled so that no additional trees can be created.

[0013] In some mobile storage devices, content protection is afforded by dividing the memory into separate areas where access to protected areas requires prior authentication. While such feature does provide some protection, it does not protect against a user who obtained a password by illicit means. Thus, another aspect of the

invention is based on the recognition that a mechanism or structure may be provided to divide a memory into partitions and so that at least some data in the partitions can be encrypted with a key, so that in addition to authentication that is required for accessing some of the partitions, access to one or more keys may be required to decrypt the encrypted data in such partitions.

[0014] In some applications, it may be more convenient to the user to be able to log in the memory system using one application, and then be able to use different applications to access protected content without having to log in again. In such event, all of the content that the user wishes to access in this manner may be associated with a first account, so that all such content can be accessed via different applications (e.g. music player, email, cellular communication etc.) without having to log in multiple times. Then a different set of authentication information may then be used for logging in to access protected content that is in an account different from the first account, even where the different accounts are for the same user or entity.

[0015] The above-described features may be used individually, or may be combined in any combination, in storage systems to provide greater versatility of control and/or protection for the content owner.

### BRIEF DESCRIPTION OF THE DRAWINGS

[0016] Fig. 1 is a block diagram of a memory system in communication with the host device useful for illustrating this invention.

[0017] Fig. 2 is a schematic view of different partitions of a memory and of unencrypted and encrypted files stored in different partitions where access to certain partitions and the encrypted files is controlled by access policies and authentication procedures to illustrate an embodiment of the invention.

[0018] Fig. 3 is a schematic view of a memory illustrating the different partitions in the memory.

[0019] Fig. 4 is a schematic view of file location tables for the different partitions of the memory shown in Fig. 3 where some of the files in the partitions are encrypted to illustrate an embodiment of the invention.

[0020] Fig. 5 is a schematic view of access control records in an access controlled record group and the associated key references to illustrate an embodiment of the invention.

[0021] Fig. 6 is a schematic view of tree structures formed by access controlled records groups and access controlled records to illustrate an embodiment of the invention.

[0022] Fig. 7 is a schematic diagram of a tree illustrating three hierarchical trees of access controlled record groups to illustrate a process of formation of the trees.

[0023] Fig. 8A and 8B are flow charts illustrating the processes carried out by a host device and a memory device such as a memory card for creating and using a system access control record.

[0024] Fig. 9 is a flow chart illustrating a process using a system access control record to create an access controlled record group to illustrate the invention.

[0025] Fig. 10 is a flow chart illustrating a process for creating an access control record.

[0026] Fig. 11 is a schematic view of two access control record groups useful for illustrating a particular application of the hierarchical tree.

[0027] Fig. 12 is a flow chart illustrating a process for delegation of specific rights.

[0028] Fig. 13 is a schematic view of an access controlled record group and an access control record to illustrate the process of delegation of Fig. 12.

[0029] Fig. 14 is a flowchart illustrating the process for creating a key for the purpose of encryption and/or decryption.



[0030] Fig. 15 is a flow chart illustrating a process for removing access rights and/or permission for data access according to an accessed controlled record.

[0031] Fig. 16 is a flow chart illustrating a process for requesting access when access rights and/or permission to access has been deleted or has expired.

[0032] Fig. 17A and Fig. 17B are schematic views illustrating an organization of a rule structure for authentication and policies for granting access to cryptographic keys to illustrate another embodiment of the invention.

[0033] Fig. 18 is a flow diagram illustrating sessions of authentication and access when some sessions are open.

[0034] Fig. 19-22 are flow charts illustrating different authentication processes.

[0035] For simplicity in description, identical components are labeled by the same numerals in this application.

#### DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

[0036] An example memory system in which the various aspects of the present invention may be implemented is illustrated by the block diagram of Fig. 1. As shown in Fig. 1, the memory system 10 includes a central processing unit (CPU) 12, a buffer management unit (BMU) 14, a host interface module (HIM) 16 and a flash interface module (FIM) 18, a flash memory 20 and a peripheral access module (PAM) 22. Memory system 10 communicates with a host device 24 through a host interface bus 26 and port 26a. The flash memory 20 which may be of the NAND type, provides data storage for the host device 24. The software code for CPU 12 may also be stored in flash memory 20. FIM 18 connects to the flash memory 20 through a flash interface bus 28 and port 28a. HIM 16 is suitable for connection to a host system like a digital camera, personal computer, personal digital assistants (PDA), digital media players, MP-3 players, cellular telephones or other digital devices. The peripheral access module 22 selects the appropriate controller module such as FIM, HIM and BMU for communication with the CPU 12. In one embodiment, all of the

components of system 10 within the dotted line box may be enclosed in a single unit such as in memory card or stick 10' and preferably encapsulated.

[0037] While the invention is illustrated herein by reference to flash memories, the invention may also be applicable to other types of memories, such as magnetic disks, optical CDs, as well as all other types of rewrite-able non volatile memory systems.

[0038] The buffer management unit 14 includes a host direct memory access (HDMA) 32, a flash direct memory access (FDMA) 34, an arbiter 36, a buffer random access memory (BRAM) 38 and a crypto-engine 40. The arbiter 36 is a shared bus arbiter so that only one master or initiator (which can be HDMA 32, FDMA 34 or CPU 12) can be active at any time and the slave or target is BRAM 38. The arbiter is responsible for channeling the appropriate initiator request to the BRAM 38. The HDMA 32 and FDMA 34 are responsible for data transported between the HIM 16, FIM 18 and BRAM 38 or the CPU random access memory (CPU RAM) 12a. The operation of the HDMA 32 and of the FDMA 34 are conventional and need not be described in detail herein. The BRAM 38 is used to store data passed between the host device 24 and flash memory 20. The HDMA 32 and FDMA 34 are responsible for transferring the data between HIM 16/FIM 18 and BRAM 38 or the CPU RAM 12a and for indicating sector completion.

[0039] For improved security of the content stored in memory 20, memory system 10 generates the key value(s) that are used for encryption and/or decryption, where this value(s) is substantially not accessible to external devices such as host device 24. However, encryption and decryption is typically done file by file, since the host device reads and writes data to memory system 10 in the form of files. Like many other types of storage devices, memory device 10 is not aware of files or file systems. While memory 20 does store a file allocation table (FAT) where the logical addresses of the files are identified, the FAT is typically accessed and managed by the host device 24 and not by the controller 12. Therefore, in order to encrypt data in a particular file, the controller 12 will have to rely on the host device to send the logical addresses of the data in the file in memory 20, so that the data of the particular file can be found and encrypted and/or decrypted by system 10 using the key value(s) available only to system 10.

[0040] To provide a handle for both the host device 24 and memory system 10 to refer to the same key(s) for cryptographically processing data in files, the host device provides a reference for each of the key values generated by system 10, where such reference may simply be a key ID. Thus, the Host 24 associates each file that is cryptographically processed by system 10 with a key ID, and the system 10 associates each key value that is used to cryptographically process data with a key ID provided by the host. Thus, when the host requests that a file be cryptographically processed, it will send the request along with a key ID along with the logical addresses of data to be fetched from or stored in memory 20 to system 10. System 10 generates a key value and associates the key ID provided by the host 24 with such value, and performs the cryptographic processing. In this manner, no change needs to be made in the manner memory system 10 operates while allowing it to completely control the cryptographic processing using the key(s), including exclusive access to the key value(s). In other words, system 10 continues to allow the host 24 to manage the files by having exclusive control of FAT, while it maintains exclusive control for the generation and management of the key value(s) used for cryptographic processing. The host device 24 has no part in the generation and management of the key value(s) used for cryptographic processing of data.

[0041] The key ID provided by the host 24 and the key value generated by the memory system form two attributes of a quantity referred to below as the “content encryption key” or CEK in one of the embodiments. While the host 24 may associate each key ID with one or more files, host 24 may also associate each key ID with unorganized data or data organized in any manner, and not limited to data organized into complete files.

[0042] In order for a user or application to gain access to protected content or area in system 10, it will need to be authenticated using a credential which is pre-registered with system 10. A credential is tied to the access rights granted to the particular user or application with such credential. In the pre-registration process, system 10 stores a record of the identity and credential of the user or application, and the access rights associated with such identity and credential determined by the user or application and provided through the host 24. After the pre-registration has been completed, when the

user or application requests to write data to memory 20, it will need to provide through the host device its identity and credential, a key ID for encrypting the data, and the logical addresses where the encrypted data is to be stored. System 10 generates a key value and associates this value with the key ID provided by the host device, and stores in its record or table for this user or application the key ID for the key value used to encrypt the data to be written. It then encrypts the data and stores the encrypted data at the addresses designated by the host as well as the key value it generated.

[0043] When a user or application requests to read encrypted data from memory 20, it will need to provide its identity and credential, the key ID for the key previously used to encrypt the requested data, and the logical addresses where the encrypted data is stored. System 10 will then match the user or application identity and credential provided by the host to those stored in its record. If they match, system 10 will then fetch from its memory the key value associated with the key ID provided by the user or application, decrypt the data stored at the addresses designated by the host device using the key value and send the decrypted data to the user or application.

[0044] By separating the authentication credentials from the management of keys used for cryptographic processing, it is then possible to share rights to access data without sharing credentials. Thus, a group of users or applications with different credentials can have access to the same keys for accessing the same data, while users outside this group have no access. While all users or applications within a group may have access to the same data, they may still have different rights. Thus, some may have read only access, while others may have write access only, while still others may have both. Since system 10 maintains a record of the users or application identities and credentials, the key IDs they have access to, and the associated access rights to each of the key IDs, it is possible for system 10 to add or delete key IDs and alter access rights associated with such key IDs for particular users or applications, to delegate access rights from one user or application to another, or even to delete or add records or tables for users or applications, all as controlled by a properly authenticated host device. The record stored may specify that a secure channel is required for

accessing certain keys. Authentication may be done using symmetric or asymmetric algorithms as well as passwords.

[0045] Especially important is the portability of the secured content in the memory system 10. Since the key value is generated by the memory system and substantially not available to external systems, when the memory system or a storage device incorporating the system is transferred from one external system to another, security of the content stored therein is maintained, and external systems are not able to access such content unless they have been authenticated in a manner completely controlled by the memory system. Even after being so authenticated, access is totally controlled by the memory system, and external systems can access only in a manner controlled according to preset records in the memory system. If a request does not comply with such records, the request will be denied.

[0046] To provide greater flexibility in protecting content, it is envisioned that certain areas of the memory referred to below as partitions can be accessed only by properly authenticated users or applications. When combined with the above described features of key-based data encryption, system 10 provides greater data protection capability. As shown in Fig. 2, the flash memory 20 may have its storage capacity divided into a number of partitions: a user area or partition and custom partitions. The user area or partition P0 is accessible to all users and applications without authentication. While all bit values of data stored in the user area can be read or written to by any application or user, if the data read is encrypted, the user or application without authority to decrypt would not be able to access the information represented by the bit values stored in a user area. This is illustrated, for example, by files 102 and 104 stored in user area P0. Also stored in the user area are unencrypted files such as 106 which can be read and understood by all applications and users. Thus, symbolically, the files that are encrypted are shown with locks associated with them such as for files 102 and 104.

[0047] While an encrypted file in a user area P0 cannot be understood by unauthorized applications or users, such applications or users may still be able to delete or corrupt the file, which may be undesirable for some applications. For this purpose, memory 20 also includes protected custom partitions such as partitions P1

and P2 which cannot be accessed without prior authentication. The authentication process permitted in the embodiments in this application is explained below.

[0048] As also illustrated in Fig. 2, a variety of users or applications may access the files in memory 20. Thus users 1 and 2, and applications 1-4 (running on devices) are shown in Fig. 2. Before these entities are allowed to access protected content in memory 20, they are first authenticated by an authentication process in a manner explained below. In this process, the entity that is requesting access needs to be identified at the host side for role based access control. Thus, the entity requesting access first identifies itself by supplying information such as "I am application 2 and I wish to read file 1." Controller 12 then matches the identity, authentication information and request against the record stored in memory 20 or controller 12. If all requirements are met, access is then granted to such entity. As illustrated in Fig. 2, user 1 is allowed to read from and write to file 101 in partition P1, but can only read files 102 and 104 in addition to user 1 having unrestricted rights to read from and write to files 106 in P0. User 2, on the other hand, is not allowed access to file 101 and 104 but has read and write access to file 102. As indicated in Fig. 2, users 1 and 2 have the same login algorithm (AES) while applications 1 and 3 have different login algorithms (e.g. RSA and 001001) which are also different from those of users 1 and 2.

[0049] The Secure Storage Application (SSA) is a security application of the memory system 10, and illustrates an embodiment of the invention, which can be used to implement many of the above-identified features. SSA may be embodied as software or computer code with database stored in the memory 20 or a non-volatile memory (not shown) in CPU 12, and is read into RAM 12a and executed by CPU 12. The acronyms used in reference to the SSA are set forth in the table below:

## Definitions, Acronyms &amp; Abbreviations

ACR	Access Control Records
AGP	ACR Group
CBC	Chain Block Cipher
CEK	Content Encryption Key
ECB	Electronic Codebook
ACAM	ACR Attributes Management
PCR	Permissions Control Record
SSA	Secure Storage Application
Entity	Any thing that has real and individual existence (host side) that logs in the SSA and thus utilizes its functionalities.

**SSA System Description**

[0050] Data security, integrity and access control are the major roles of the SSA. The *data* are files that would otherwise be stored plainly on a mass-storage device of some kind. The SSA system sits atop of the storage system and adds the security layer for the stored host files.

[0051] The main task of the SSA is to manage the different rights associated with the stored (and secured) content in the memory. The memory application needs to manage multiple users and content rights to multiple stored content. Host applications from their side, see drives and partitions that are visible to such applications, and file allocation tables (FATs) that manage and portray the locations of the stored files on the storage device.

[0052] In this case the storage device uses NAND flash chip divided to partitions, although other mobile storage devices may also be used and are within the scope of this invention. These partitions are continuous threads of logical addresses, where a *start* and an *end* address define their boundaries. Restrictions may therefore be imposed on access to hidden partitions, if desired, by means of software (such as software stored in memory 20) that associates such restrictions with the addresses within such boundaries. Partitions are fully recognizable to the SSA by their logical address boundaries that are managed by it. The SSA system uses partitions to physically secure data from unauthorized host applications. To the host, the partitions are a mechanism of defining proprietary spaces in which to store data files. These partitions can either be public, where anyone with access to the storage device can see

and be aware of the partition's presence on the device, or private or hidden, where only the selected host applications have access to and are aware of their presence in the storage device.

[0053] Fig. 3 is a schematic view of a memory illustrating the partitions of the memory: P0, P1, P2 and P3 (obviously fewer or more partitions than four may be employed), where P0 is a public partition which can be accessed by any entity without authentication.

[0054] A private partition (such as P1, P2 or P3) hides the access to the files within it. By preventing the host from accessing the partition, the flash device (e.g. flash card) delivers protection of the data files inside the partition. This kind of protection, however, engulfs all of the files residing in the hidden partition by imposing restrictions on access to data stored at the logical addresses within the partition. In other words, the restrictions are associated with a range of logical addresses. All of the *users/hosts* that have access to that partition will have unlimited access to all of the files inside. To isolate different files from one another – or groups of files – the SSA system provides another level of security and integrity per file - or groups of files – using keys and key references or Key IDs. A key reference or key ID of a particular key value used for encrypting data at different memory addresses can be analogized to a container or domain that contains the encrypted data. For this reason, in Fig. 4, the key references or key IDs (e.g. “key 1” and “key 2”) are shown graphically as areas surrounding the files encrypted using the key values associated with the key IDs.

[0055] In reference to Fig. 4, for example, File A is accessible to all entities without any authentication, since it is shown as not enclosed by any key ID. Even though File B in the public partition can be read or overwritten by all entities, it contains data encrypted with a key with ID “key 1”, so that the information contained in File B is not accessible to an entity unless such entity has access to such key. In this manner using key values and key references or Key IDs provide logical protection only, as opposed to the type of protection provided by the partition described above. Hence, any host that can access a partition (public or private) is capable of reading or writing the data in the entire partition, including the encrypted data. However, since the data



is encrypted, unauthorized users can only corrupt it. They preferably cannot alter the data without detection or use it. By restricting the access to the encryption and/or decryption keys, this feature can allow only the authorized entities to use the data. Files B and C are also encrypted using a key with key ID "key 2" in P0.

[0056] Data confidentiality and integrity can be provided through symmetric encryption methods that use Content Encryption Keys (CEK), one per CEK. In the SSA embodiment, the CEKs are generated by the flash device (e.g. flash card), used internally only, and kept as secrets from the outside world. The data that is encrypted or ciphered may also be either hashed or the cipher is chain blocked to ensure data integrity.

[0057] Not all the data in the partition is encrypted by different keys and associated with different key IDs. Certain logical addresses either in public or user files or in the operating system area (i.e. FAT) may not be associated with any key or key reference, and thus are available to any entity that can access the partition itself.

[0058] An entity that calls for the ability to create keys and partitions as well as writing and reading data from them or using the keys, needs to login to the SSA system through an Access Control Record (ACR). The privileges of an ACR in the SSA system are called *Actions*. Every ACR may have *Permissions* to perform Actions of the following three categories: Creating partitions and keys/key IDs, accessing partitions and keys and creating/updating other ACRs.

[0059] ACRs are organized in groups called ACR Groups or AGPs. Once an ACR has successfully authenticated, the SSA system opens a *Session* through which any of the ACR's actions can be executed..

#### User Partition(s)

[0060] The SSA system manages one or more public partitions, also referred to as the user partition(s). This partition exists on the storage device and is a partition or partitions that can be accessed through the standard read write commands of the

storage device. Getting information regarding the size of the partition(s) as well as its existence on the device preferably cannot be hidden from the host system.

[0061] The SSA system enables accessing this partition(s) either through the standard read write commands or the SSA commands. Therefore, accessing the partition preferably cannot be restricted to specific ACRs. The SSA system, however, can enable the host devices to restrict the access to the user partition. Read and write accesses can be enabled/disabled individually. All four combinations (e.g. write only, read only (write protect), read and write and no access) are allowed.

[0062] The SSA system enables ACRs to associate key IDs with files within the user partition and encrypt individual files using keys associated with such key IDs. Accessing encrypted files within the user partitions as well as setting the access rights to the partitions will be done using the SSA command set (refer to Appendix A for detailed description of the SSA commands—In the Appendix, key ID is referred to as “domain” ).The above features also apply to data not organized into files.

#### SSA partitions

[0063] These are hidden (from the host operating system or OS) partitions that can be accessed only through the SSA commands. The SSA system will preferably not allow the host device to access an SSA partition, other than through a session (described below) established by logging onto an ACR. Similarly, preferably the SSA will not provide information regarding the existence, size and access permission of an SSA partition, unless this request is coming through an established session.

[0064] Access rights to partitions are derived from the ACR permissions. Once an ACR is logged into the SSA system, it can share the partition with other ACRs (described below). When a partition is created, the host provides a reference name or ID (e.g. P0-P3 in Figs. 3 and 4) for the partition. This reference is used in further read and write commands to the partition.

Partitioning of the storage device

[0065] All available storage capacity of the device is preferably allocated to the user partition and the currently configured SSA partitions. Therefore, any repartition operation may involve reconfiguration of the existing partitions. The net change to the device capacity (sum of sizes of all partitions) will be zero. The IDs of the partitions in the device memory space are defined by the host system.

[0066] The host system can either repartition one of the existing partitions into two smaller ones or, merge two existing partitions (which may or may not be adjacent) into one. The data in the divided or merged partitions can be either erased or left untouched, at the host's discretion.

[0067] Since repartitioning of the storage device may cause loss of data (either because it was erased or moved around in the logical address space of the storage device) severe restrictions on repartitioning are administered by the SSA system. Only an ACR residing in a root AGP (explained below) is allowed to issue a repartition command and it can only reference partitions owned by it. Since the SSA system is not aware of how data is organized in the partitions (FAT or other file system structure) it is the host's responsibility to reconstruct these structures any time the device is repartitioned.

[0068] Repartitioning of the user partition will change the size and other attributes of this partition as seen by the host OS.

[0069] After repartitioning, it is the host system's responsibility to make sure any ACR in the SSA system is not referencing the non-existing partitions. If these ACRs are not deleted or updated appropriately, future attempts, on behalf of these ACRs, to access the non-existing partitions will be detected and rejected by the system. Similar care is taken, regarding deleted keys and key IDs.

Keys, Key IDs and Logical Protection

[0070] When a file is written to a certain hidden partition, it is hidden from the general public. But, once an entity (hostile or not) gets knowledge and access to this

partition the file becomes available and plain to see. To further secure the file, the SSA can encrypt it in the hidden partition, where the credentials for accessing the key for decrypting the file are preferably different from those for accessing the partition. Due to the fact that files are not something that the SSA is aware of (totally controlled and managed by the host), associating a CEK with a file is a problem. Linking the file to something the SSA acknowledges – the key ID, rectifies this. Thus, when a key is created by the SSA, the host associates the key ID for this key with the data encrypted using the key created by the SSA.

[0071] The key value and key ID provide logical security. All data associated with a given key ID, regardless of its location, is ciphered with the same content encryption key (CEK) whose reference name or key ID is uniquely provided at creation by the host application. If an entity obtains access to a hidden partition (by authenticating through an ACR) and wishes to either read or write an encrypted file within this partition, it needs to have access to the key ID that is associated with the file. When granting access to the key for this key ID, the SSA loads the key value in CEK associated with this key ID and either decrypts the data before sending it to the host or encrypts the data before writing it to the flash memory 20. A key value in CEK associated with a key ID is randomly created once by the SSA system and maintained by it. No one outside the SSA system has knowledge or access to this key value in CEK. The outside world only provides and uses a reference or key ID, not the key value in CEK. The key value is entirely managed and only accessible by the SSA

[0072] The SSA system protects the data associated with the key ID using any one (user defined) of the following cipher modes (the actual cryptographic algorithms used, as well as the key values in CEKs, are system controlled and not revealed to the outside world):

[0073] Block mode – Data is divided into blocks, each one of them, encrypted individually. This mode is generally considered less secure and susceptible to dictionary attacks, However, it will allow users to randomly access any one of the data blocks.

[0074] Chained mode – Data is divided into blocks, which are chained during the encryption process. Every block is used as one of the inputs to the encryption process of the next one. This mode, although considered as more secure, requires that the data is always written and read sequentially from start to end, creating an overhead not always acceptable to the users.

[0075] Hashed – Chain mode with the additional creation of a data digest that can be used for validating data integrity.

#### ACRs and Access Control

[0076] The SSA is designed to handle multiple applications where each one of them is represented as a tree of nodes in the system database. Mutual exclusion between the applications is achieved by ensuring no cross talk between the tree branches.

[0077] In order to gain access to the SSA system, an entity needs to establish a connection via one of the system's ACRs. Login procedures are administered by the SSA system according to the definitions embedded in the ACR the user chose to connect with.

[0078] The ACR is an individual login point to the SSA system. The ACR holds the login credentials and the authentication method. Also residing in the record are the login permissions within the SSA system, among which are the read and write privileges. This is illustrated in Fig. 5, which illustrates  $n$  ACRs in the same AGP. This means that at least some of the  $n$  ACRs may share access to the same key. Thus, ACR #1 and ACR # $n$  share access to a key with key ID "key 3", where ACR#1 and ACR# $n$  are the ACR IDs, and "key 3" is a key ID for the key that is used to encrypt data associated with "key 3". The same key can also be used to encrypt and/or decrypt multiple files, or multiple sets of data.

[0079] The SSA system supports several types of login onto the system where authentication algorithms and user credentials may vary, as may the user's privileges in the system once he logged in successfully. Fig. 5 again illustrates different login algorithms and credentials. ACR#1 requires a password login algorithm and password as credential whereas ACR#2 requires a PKI (public key infrastructure)

login algorithm and public key as credential. Thus, to login, an entity will need to present a valid ACR ID, as well as the correct login algorithm and credential.

[0080] Once an entity is logged into an ACR of the SSA system, its permissions - its rights to use SSA commands - are defined in the Permissions Control Record (PCR) which is associated with the ACR. In Fig. 5, ACR#1 grants read only permission to data associated with "key 3", and ACR #2 grants permission to read and write data associated with "key 5" according to the PCR shown.

[0081] Different ACRs may share common interests and privileges in the system such as in keys with which to read and write. To accomplish that, ACRs with something in common are grouped in AGPs – ACR Groups. Thus, ACR #1 and ACR #n share access to a key with key ID "key 3".

[0082] AGPs and, the ACRs within, are organized in hierarchical trees and so aside from creating secure keys that keep sensitive data secure; an ACR can preferably also create other ACR entries that correspond to his key ID/partitions. These ACR children will have the same or less permissions as their father – creator and, may be given permissions for keys the father ACR himself created. Needless to add, the children ACRs get access permissions to any key that they create. This is illustrated in Fig. 6. Thus, all of the ACRs in AGP 120 were created by ACR 122 and two of such ACRs inherit from ACR 122 permission(s) to access to data associated with "key 3".

#### AGP

[0083] Logging onto the SSA system is done by specifying an AGP and an ACR within the AGP.

[0084] Every AGP has a unique ID (reference name), which is used as an index to its entry in the SSA database. The AGP name is provided to the SSA system, when the AGP is created. If the provided AGP name already exists in the system, the SSA will reject the creation operation.

[0085] AGPs are used to administer restrictions on delegation of access and management permissions as will be described in the following sections. One of the functions served by the two trees in Fig. 6 is to administer the access by entirely

separate entities, such as two different applications, or two different computer users. For such purposes, it may be important for the two access processes to be substantially independent of one another (i.e. substantially no cross-talk), even though both occur at the same time. This means that the authentication, permissions as well as the creation of additional ACRs and AGPs in each tree are not connected to and do not depend on those of the other tree. Hence, when the SSA system is used in memory 10, this allows the memory system 10 to serve multiple applications simultaneously. It also allows the two applications to access two separate sets of data independently of one another (e.g. a set of photographs and a set of songs). This is illustrated in Fig. 6. Thus, the data associated with “keys 3”, “key X” and “key Z” for the application or user accessing via nodes (ACRs) in the tree in the top portion of Fig. 6 may comprise photographs. The data associated with “key 5” and “key Y” for the application or user accessing via nodes (ACRs) of the tree in the bottom portion of Fig. 6 may comprise songs. The ACR that created the AGP has the permission to delete it only when the AGP is empty of ACR entries

#### The entity's SSA entry point: Access Control Record (ACR)

[0086] An ACR in the SSA system describes the way the entity is permitted to log into the system. When an entity logs into the SSA system it needs to specify the ACR that corresponds to the authentication process it is about to perform. An ACR includes a Permissions Control Record (PCR) that illustrates the granted actions the user can execute once authenticated as defined in the ACR as illustrated in Fig. 5. The host side entity provides all of the ACR data fields.

[0087] When an entity has successfully logged onto an ACR, the entity will be able to query on all of the ACR's partition and key access permissions and ACAM permissions (explained below).

#### ACR ID

[0088] When an SSA system entity initiates the login process it needs to specify the ACR ID (as provided by the host when the ACR was created) that corresponds to the login method so that the SSA will set up the correct algorithms and select the correct

PCR when all login requirements have been met. The ACR ID is provided to the SSA system when the ACR is created.

#### Login/Authentication Algorithm

[0089] The authentication algorithm specifies what sort of login procedure will be used by the entity, and what kind of credentials are needed to provide proof of user's identity. The SSA system supports several standard login algorithms, ranging from no procedure (and no credential) and password-based procedures to a two-way authentication protocols based on either symmetric or asymmetric cryptography.

#### Credentials

[0090] The entity's credentials correspond to the login algorithm and are used by the SSA to verify and authenticate the user. An example for credential can be a password/PIN-number for password authentication, AES-key for AES authentication, etc. The type/format of the credentials (i.e. the PIN, the symmetric key, etc...) is predefined and derived from the authentication mode; they are provided to the SSA system when the ACR is created. The SSA system has no part in defining, distributing and managing these credentials, with the exception of PKI based authentication where the device (e.g. flash card) can be used to generate the RSA key pair and the public key can be exported for certificate generation.

#### The Permissions Control Record (PCR)

[0091] The PCR shows what is granted to the entity after logging into the SSA system and passing the ACR's authentication process successfully. There are three types of permission categories: Creation permissions for partition and keys, Access permissions to partitions and keys and management permissions for Entity-ACR Attributes

#### Accessing Partitions

[0092] This section of the PCR contains the list of partitions (using their IDs as provided to the SSA system) the entity can access upon completing the ACR phase successfully. For each partition the access type may be restricted to write-only or read-only or may specify full write/read access rights. Thus, the ACR#1 in Fig. 5 has



access to partition #2 and not partition #1. The restrictions specified in the PCR apply to the SSA partitions and the public partition.

[0093] The public partition can be accessed either by regular read and write commands to the device (e.g. flash card) hosting the SSA system, or by SSA commands. When a root ACR (explained below) is created with the permission to restrict the public partition, he can pass it on to his children. An ACR can preferably only restrict the regular read and write commands from accessing the public partition. ACRs in the SSA system can be restricted preferably only upon their creation. Once an ACR has the permission to read/write from/to the public partition, preferably it cannot be taken away.

#### Accessing Key IDs

[0094] This section of the PCR contains the data associated with the list of key IDs (as provided to the SSA system by the host) the entity can access when the ACR policies have been met by the entity's login process. The key ID specified is associated with a file/files that reside in the partition appearing in the PCR. Since the key IDs are not associated with logical addresses in the device (e.g. flash card), when more than one partition is associated with a specific ACR, the files can be in either one of the partitions. The key IDs specified in the PCR can have each, a different set of access rights. Accessing data pointed to by key IDs can be restricted to write-only or read-only or may specify full write/read access rights.

#### ACR Attributes Management (ACAM)

[0095] This section describes how in certain cases the ACR's system attributes can be changed.

[0096] The ACAM actions that may be permitted in the SSA system are:

[0097] Create/delete/update AGPs and ACR.

[0098] Create/delete Partitions and Keys.

[0099] Delegate access rights to keys and partitions.

[00100] A father ACR preferably cannot edit ACAM permissions. This would preferably require the deletion and recreation of the ACR. Also the access permission to a key ID created by the ACR can preferably not be taken away.

[00101] Create/delete/update AGPs and ACR

[00102] An ACR may have the capacity to create other ACRs and AGPs. Creating ACRs also may mean delegating them some or all of the ACAM permissions possessed by their creator. Having the permission to create ACRs means having the permission for the following actions:

1. Define and edit the child's credentials – the authentication method preferably cannot be edited once set by the creating ACR. The credentials may be altered within the boundary of the authentication algorithm that is already defined for the child.
2. Delete an ACR.
3. Delegate the creating permission to the child ACR (thus having grandchildren).

[00103] An ACR with the permissions to create other ACRs has the permission to delegate the unblocking permission to ACRs it creates (although it probably does not have the permission to unblock ACRs). The father ACR will place in the child ACR a reference to his unblocker.

[00104] The father ACR is the only ACR that has the permission to delete his child ACR. When an ACR deletes a lower level ACR that he created, then all ACRs spawned by this lower-level ACR are automatically deleted as well. When an ACR is deleted then all the key IDs and partitions that it created are deleted.

[00105] There are two exceptions by which an ACR can update its own record:

[00106] Passwords/PINs, although set by the creator ACR, can be updated only by the ACR that includes them.

[00107] A root ACR may delete itself and the AGP that it resides in.

Delegate access rights to Keys and partitions

[00108] ACRs and their AGPs are assembled in hierarchical trees where the root AGP and the ACRs within are at the top of the tree (e.g. root AGPs 130 and 132 in Fig. 6). There can be several AGP trees in the SSA system though they are totally separated from one another. An ACR within an AGP can delegate access permissions to its keys to all ACRs within the same AGP that it is in, and to all the ACRs created by them. The permission to create keys preferably includes the permission to delegate access permissions to use the keys.

Permissions to keys are divided into three categories:

1. Access – this defines the access permissions for the key i.e. Read, Write.
2. Ownership – an ACR that created a key is by definition its owner. This ownership can be delegated from one ACR to another (provided that they are in the same AGP or in a child AGP). An ownership of a key provides the permission to delete it as well as delegate permissions to it.
3. Access Rights Delegation – this permission enables the ACR to delegate the rights he holds.

[00109] An ACR can delegate access permissions to partitions he created as well as other partitions he has access permissions to.

[00110] The permission delegation is done by adding the names of the partitions and key IDs to the designated ACR's PCR. Delegating key access permissions may either be by the key ID or by stating that access permission is for all of the created keys of the delegating ACR.

Blocking and Unblocking of ACRs

[00111] An ACR may have a blocking counter which increments when the entity's ACR authentication process with the system is unsuccessful. When a certain maximum number (MAX) of unsuccessful authentications is reached, the ACR will be blocked by the SSA system.

[00112] The blocked ACR can be unblocked by another ACR, referenced by the blocked ACR. The reference to the unblocking ACR is set by its creator. The unblocking ACR preferably is in the same AGP as the creator of the blocked ACR and has the “unblocking” permission.

[00113] No other ACR in the system can unblock the blocked ACR. An ACR may be configured with a blocking counter but without an unblocker ACR. In this case, if this ACR get blocked it cannot be unblocked.

#### Root AGP – Creating an application database

[00114] The SSA system is designed to handle multiple applications and isolate the data of each one of them. The tree structure of the AGP system is the main tool used to identify and isolate application specific data. The root AGP is at the tip of an application SSA database tree and adheres to somewhat different behavior rules. Several root AGPs can be configured in the SSA system. Two root AGPs 130 and 132 are shown in Fig. 6. Obviously fewer or more AGPs may be used and are within the scope of this invention.

[00115] Registering the device (e.g. flash card) for a new application and/or issue credentials of a new applications for the device are done through the process of adding new AGP/ACR tree to the device.

[00116] The SSA system supports three different modes of root AGP creation (as well as all of the ACRs of the root AGP and their permissions):

1. Open: Any user or entity without requiring any sort of authentication, or users/entities authenticated through the system ACR (explained below), can create a new root AGP. The open mode enables creation of root AGPs either without any security measures while all data transfer is done on an open channel (i.e. in the secure environment of an issuance agency) or, through a secure channel established through the system ACR authentication (i.e. Over The Air (OTA) and post issuance procedures).

If the system ACR is not configured (this is an optional feature) and the root AGP creation mode is set to Open, only the open channel option is available.

2.       Controlled: Only entities authenticated through the System ACR can create a new root AGP. The SSA system cannot be set to this mode if system ACR is not configured.

3.       Locked: Creation of root AGPs is disabled and no additional root AGPs can be added to the system

Two SSA commands control this feature (these commands are available to any user/entity without authentication):

1.       Method configuration command – Used to configure the SSA system to use any one of the three root AGP creation modes. Only the following mode changes are allowed: Open -> Controlled, Controlled -> Locked (i.e. if the SSA system is currently configured as Controlled, it can only be changed to locked).

2.       Method configuration lock command - Used to disable the method configuration command and permanently lock the currently selected method.

**[00117]**       When a root AGP is created, it is in a special initializing mode that enables the creation and configuration of its ACRs (using the same access restrictions that applied to the creation of the root AGP). At the end of the root AGP configuration process, when the entity explicitly switches it to operating mode, the existing ACRs can no longer be updated and additional ACRs can no longer be created

**[00118]**       Once a root AGP is put in standard mode it can be deleted only by logging into the system through one of its ACRs that is assigned with the permission to delete the root AGP. This is another exception of root AGP, in addition to the special initialization mode; it is preferably the only AGP that may contain an ACR with the permission to delete its own AGP, as opposed to AGPs in the next tree level.

**[00119]**       The third and last difference between a root ACR and a standard ACR is that it is the only ACR in the system that can have the permission to create and delete partitions.

#### SSA System ACR

**[00120]**       The system ACR may be used for the following two SSA operations:

1. Create an ACR/AGP tree under the protection of a secured channel within hostile environments.

2. Identify and authenticate the device hosting the SSA system.

[00121] There may preferably be only one System ACR in the SSA and once defined it preferably cannot be changed. There is no need for system authentication when creating the System ACR; only a SSA command is needed. The create-system-ACR feature can be disabled (similarly to the create-root-AGP feature). After the system ACR is created, the create-system-ACR command has no effect, since preferably only one System ACR is allowed.

[00122] While in the process of creating, the System ACR is not operational. Upon finishing, a special command needs to be issued indicating that the System ACR is created and ready to go. After this point the System ACR preferably cannot be updated or replaced.

[00123] The System ACR creates the root ACR/AGP in the SSA. It has permission to add/change the root level until such time that the host is satisfied with it and blocks it. Blocking the root AGP essentially cuts off its connection to the system ACR and renders it temper proof. At this point no one can change/edit the root AGP and the ACRs within. This is done through an SSA command. Disabling creation of root AGPs has a permanent effect and cannot be reversed. The above features involving the system ACR are illustrated in Fig. 7. The system ACR is used to create three different root AGPs. At a certain time after these are created, the SSA command is sent from the host to block the root AGPs from the system ACR, thereby disabling the create-root-AGP feature, as indicated by the dotted lines connecting the System ACR to the root AGPs in Fig. 7. This renders the three root AGPs temper proof. The three root AGPs may be used to create children AGPs to form three separate trees, before or after the root AGPs are blocked.

[00124] The above described features provides great flexibility to the content owner in configuring secure products with content. Secure products need to be "Issued". Issuance is the process of putting identification keys by which the device can identify the host and vice versa. Identifying the device (e.g. flash card) enables the host to decide whether it can trust its secrets with it. On the other hand,

identifying the host enables the device to enforce security policies (grant and execute a specific host command) only if the host is allowed to.

[00125] Products that are designed to serve multiple applications will have several identification keys. The product can be "pre-issued" - keys stored during manufacturing before shipping, or "post issued" - new keys are added after shipping. For post issuance, the memory device (e.g. memory card) needs to contain some kind of master or device level keys which are being used to identify entities which are allowed to add applications to the device.

[00126] The above described features enables a product to be configured to enable/disable post issuance. In addition, the post issuance configuration can be securely done after shipping. The device may be bought as a retail product with no keys on it in addition to the master or device level keys described above, and then be configured by the new owner to either enable further post issuance applications or disable them.

[00127] Thus, the system ACR feature provides the capability to accomplish the above objectives:

- Memory devices with no system ACR will allow unlimited and uncontrolled addition of applications.

- Memory devices without system ACR can be configured to disable the system ACR creation, which means there is no way to control adding of new applications (unless the feature of creating new root AGP is disabled as well)

- Memory devices with system ACR will allow only controlled addition of applications via a secure channel to establish through an authentication procedure using the system ACR credential.

- Memory devices with system ACR may be configured to disable the application adding feature, before or after applications have been added.

Key ID list

**[00128]** Key IDs are created per specific ACR request; however, in the memory system 10, they are used solely by the SSA system. When a key ID is created the following data is provided by or to the creating ACR:

1. Key ID. The ID is provided by the entity through the host and is used to reference the key and data that is encrypted or decrypted using the key in all further read or write accesses.
2. Key Cipher and data integrity Mode (the Blocked, Chained and Hashed Modes above and as explained below)

**[00129]** In addition to the host provided attributes, the following data is maintained by the SSA system:

1. Key ID Owner. The ID of the ACR that is the owner. When a key ID is created the creator ACR is its owner. Key ID ownership may, however, be transferred to another ACR. Preferably only the key ID owner is allowed to transfer ownership of, and delegate, a key ID. Delegating access permission to the associated key, and revoking these rights can be administered either by the key ID owner or any other ACR assigned with delegation permissions. Whenever an attempt is made to exercise any one of these operations, the SSA system will grant it only if the requesting ACR is authorized.
2. CEK. This is the CEK used to cipher the content associated with or pointed to by the key ID. The CEK may be a 128 bit AES random key generated by the SSA system.
3. MAC and IV values. Dynamic information (message authentication codes and initiation vectors) used in the Chained Block Cipher (CBC) encryption algorithms.

**[00130]** The various features of the SSA are also illustrated in reference to the flow charts in Figs. 8A-16, where 'H' to the left of a step means the operation is performed by the host, and 'C' means the operation is performed by the card. In order to create a System ACR, the host issues to the SSA in the memory device 10 a command to create System ACR (block 202). The device 10 responds by checking



whether a System ACR already exists (block 204, diamond 206). If it already exists, then device 10 returns failure and stops (oblong 208). If it does not, then memory 10 checks to see if System ACR creation is allowed (diamond 210), and returns a failure status if not allowed (block 212). Thus, there may be instances where the device issuer does not allow the creation of a System ACR, such as in the case where the security features needed have been predetermined so that no System ACR is needed. If this is allowed, the device 10 returns OK status and waits for System ACR credentials from the host (block 214). The host checks the SSA status and whether the device 10 has indicated that the creation of a System ACR is allowed (block 216 and diamond 218). If creation is not allowed or if a system ACR already exists, the host stops (oblong 220). If the device 10 has indicated that the creation of a System ACR is allowed, the host issues a SSA command to define its login credential and sends it to the device 10 (block 222). The device 10 updates a System ACR record with the credential received and returns OK status (block 224). In response to this status signal, the host issues SSA command indicating the system ACR is ready (block 226). The device 10 responds by locking the System ACR so that it cannot be updated or replaced (block 228). This locks in the features of the system ACR and its identity for identifying the device 10 to the host.

**[00131]** The procedure for creating new trees (New Root AGPs and ACR) is determined by the way these functions are configured in the device. Fig 9 explains the procedures. Both the host 24 and the memory system 10 follow it. If adding new root AGP is disabled altogether, new root AGPs cannot be added (diamond 246). If it is enabled but requires a system ACR, the host authenticates through the system ACR and establishes a secure channel (diamond 250, block 252) prior to issuing the Create *Root\_AGP* command (block 254). If system ACR is not required (diamond 248) the host 24 can issue the create root AGP command without authentication and proceed to block 254. If system ACR does exist, the host may use it even if it is not required (not shown in the flow chart). The device (e.g. flash card) will reject any attempt to create a new root AGP if the function is disabled and it will reject an attempt to create a new root AGP without authentication, if system ACR is required (diamonds 246 and 250). The newly created AGP and ACR in block 254, are now switched to Operational Mode so that the ACRs in such AGPs cannot be updated or otherwise changed, and

no ACRs can be added to them (block 256). The system is then, optionally locked so that additional root AGPs cannot be created (block 258). The dotted line box 258 is a convention indicating that this step is an optional step. All the boxes in the flow charts of the figures of this application in dotted lines are optional steps. This allows the content owner to block the use of device 10 for other illicit purposes that may imitate a genuine memory device with legitimate content.

**[00132]** To create ACRs (other than the ACRs in the root AGP as described above), one may start with any ACR that has the right to create an ACR (block 270) as shown in Figure 10. An entity may attempt to enter through the host 24 by providing the entry point ACR identity, and the ACR with all the necessary attributes that it wishes to create (block 272). The SSA checks for a match to the ACR identity and whether the ACR with such identity has the permission to create an ACR (diamond 274). If the request is verified to be authorized, the SSA in device 10 creates an ACR (block 276).

**[00133]** Fig. 11 shows two AGPs that illustrate a tree useful in security applications using the method of Fig. 10. Thus, the ACR with identity m1 in the marketing AGP has the permission to create an ACR. The ACR m1 also has the permission to use a key for reading and writing data associated with the key ID "Marketing Information" and data associated with the key ID "Price List". Using the method of Fig. 10, it creates the Sales AGP with two ACRs: s1 and s2 with only read permission to the key for accessing pricing data associated with the key ID "Price List", but not to the key necessary for accessing data associated with the key ID "Marketing Information". In this manner, the entities with the ACRs s1 and s2 can only read but not change the pricing data, and will have no access to marketing data. The ACR m2, on the other hand, has no permission to create ACRs, and has only read permission to the keys for accessing data associated with the key ID "Price List" and with the key ID "Marketing Information".

**[00134]** Thus, access rights may be delegated in the manner explained above where m1 delegates rights to read pricing data to s1 and s2. This is particularly useful where large marketing and sales groups are involved. Where there are but one or a few sales people, there may be no need to use the method of Fig. 10. Instead, the

access rights may be delegated, by an ACR to one at a lower or the same level within the same AGP, as illustrated in Fig. 12. First, the entity enters the tree for such AGP by specifying an ACR in the manner described above in the tree through the host (block 280). Next the host will specify the ACR and the rights to delegate to. The SSA checks the tree(s) for such ACR and whether the ACR has the permission to delegate rights to the specified another ACR (diamond 282). If it does, the rights are delegated (block 284); if not it stops. The result is illustrated in Fig. 13. The ACR m1 in this case has the permission to delegate read permission to the ACR s1, so that s1 will be able to use a key to access pricing data after the delegation. This may be performed if m1 has the same or greater rights to access pricing data and the permission to so delegate. In one embodiment, m1 retains its access rights after the delegation. Preferably access rights may be delegated under restricted conditions (rather than permanently) such as for a limited time, limited number of accesses, etc...

**[00135]** The process for creating a key and key ID is illustrated in Fig. 14. The entity authenticates through an ACR (block 302). The entity requests the creation of a key with an ID specified by the host (block 304). The SSA checks and see if the ACR specified has the permission to do so (diamond 306). For example, if the key is to be used for accessing data in a particular partition, the SSA will check and see if the ACR may access such partition. If the ACR is authorized, then the memory device 10 creates a key value associated with the key ID provided by the host (block 308), and stores the key ID in the ACR, and the key value in its memory (either in the controller-associated memory or memory 20) and assigns rights and permissions according to information supplied by the entity (block 310) and modifies the PCR of such ACR with such assigned rights and permissions (block 312). Thus, the creator of the key has all available rights, such as read and write permissions, right to delegate and share with other ACRs in the same AGP or an ACR at a lower level, and the right to transfer ownership of the key.

**[00136]** An ACR can change the permissions (or the existence altogether) of another ACR in the SSA system as illustrated in Fig. 15. An entity may enter a tree through an ACR as before; in one case the entity is authenticated and then it specifies

an ACR (blocks 330, 332). It requests the deletion of a target ACR or the permission in a target ACR (block 334). If the ACR specified or the one active at such time has the right to do so (diamond 336), the target ACR is deleted, or the PCR of the target ACR is altered to delete such permission (block 338). If this is not authorized the system stops.

**[00137]** After the above described process, the target will no longer be able to access the data it was able to prior to the process. As shown in Fig. 16, an entity may attempt to enter at the target ACR (block 350) and finds that the authentication process fails, since the previously existing ACR ID is no longer present in the SSA, so that access rights are denied (diamond 352). Assuming that the ACR ID has not been deleted, the entity specifies an ACR (block 354) and the key ID and/or data in a particular partition (block 356), and the SSA then checks to see the key ID or partition access request is permitted according to the PCR of such ACR (diamond 358). If the permission has been deleted or has expired, then the request is again denied. Otherwise, the request is granted (block 360).

**[00138]** The above process describes how access to protected data is managed by the device (e.g. flash card), regardless of whether the ACR and its PCR were just changed by another ACR or were so configured to begin with.

### Sessions

**[00139]** The SSA system is designed to handle multiple users, logged in concurrently. This feature requires that every command received by the SSA is associated with a specific entity and executed only if the ACR, used to authenticate this entity, has the permissions for the requested action.

**[00140]** Multiple entities are supported through the session concept. A session is established during the authentication process and assigned a session-id by the SSA system. The session-id is internally associated with the ACR used for logging into the system and is exported to the entity to be used in all further SSA commands.

[00141] The SSA system supports two types of sessions: Open, and Secure sessions. The session type associated with a specific authentication process is defined in the ACR. The SSA system will enforce session establishment in a way similar to the way it enforces the authentication itself. Since the ACR defines the entity permissions, this mechanism enables system designers to associate secure tunneling either with accessing specific key IDs or invoking specific ACR management operations (i.e. creating new ACRs and setting credentials)

#### Open session

[00142] Open session is a session identified with a session-id but without bus encryption, all commands and data are passed in the clear. This mode of operation is preferably used in a multi-user or multi-entity environment where the entities are not part of the threat model, nor is eavesdropping on the bus.

[00143] Although not protecting the transmission of the data nor enabling efficient fire-walling between the applications on the host side, the Open session mode enables the SSA system to allow access only to the information allowed for the currently authenticated ACRs.

[00144] The Open session can also be used for cases where a partition or a key needs to be protected. However, after a valid authentication process, access is granted to all entities on the host. The only thing the various host applications need to share, in order to get the permissions of the authenticated ACR is the session-id. This is illustrated in Fig. 17A. The steps above the line 400 are those taken by the host 24. After an entity is authenticated (block 402) for ACR1, it requests access to a file associated with a key ID X in the memory device 10 (blocks 404, 406 and 408). If the PCR of the ACR 1 allows such access, device 10 grants the request (diamond 410). If not, the system returns to block 402. After authentication is completed, the memory system 10 identifies the entity issuing a command only by the assigned session id (and not the ACR credentials). Once the ACR 1 gains access to the data associated with the key IDs in its PCR, in an open session, any other application or user can access the same data by specifying the correct session ID which is shared between the different applications on the host 24. This feature is advantageous in applications where it is more convenient to the user to be able to log in only once, and be able to

access all the data tied to the account through which the log in is performed for different applications. Thus, a cellular phone user may be able to access stored emails, and listen to stored music in memory 20 without having to log in multiple times. On the other hand, data not encompassed by the ACR1 will not be accessible. Thus, the same cellular phone user may have valuable content such as games and photographs accessible through a separate account ACR2. This is data that he does not wish others who borrow his phone to access, even though he may not mind others accessing data available through his first account ACR1. Separating access to the data into two separate accounts while allowing access to ACR1 in open session provides ease of use as well as affording protection of valuable data.

[00145] To even further ease the process of sharing the session-id amongst the host applications, when an ACR is requesting an Open session it can specifically request that the session will be assigned the "0 (zero)" id. This way, applications can be designed to use a pre-defined session-id. The only restriction is, for obvious reasons, that only one ACR, requesting session 0, can be authenticated at a specific time. An attempt to authenticate another ACR requesting session 0, will be rejected.

#### Secure session

[00146] To add a layer of security, the session id may be used as shown in Fig. 17B. The memory 10 then also stores the session ids of the active sessions. In Fig. 17B, for example, in order to be able to access a file associated with key ID X, the entity will need to also provide a session id, such as session id "A" before it is allowed to access the file (blocks 404, 406, 412 and 414). In this manner, unless the requesting entity is aware of the correct session id, it cannot access the memory 10. Since the session id is deleted after the session is over and will be different for each session, an entity can gain access only when it has been able to provide the session number.

[00147] The SSA system has no way to make sure that a command is really coming from the correct authenticated entity, other than by using the session number.

For applications and use cases where there is a threat that attackers will try to use an open channel to send malicious commands, the host application uses a secure session (a secure channel).

**[00148]** When using a secure channel, the session-id, as well as the entire command, is encrypted with the secure channel encryption (session) key and the security level is as high as the host side implementation.

#### Terminating a session

**[00149]** A session is terminated and, the ACR is logged off, in any one of the following scenarios:

1. The entity issues an explicit end-session command.
2. Time out on communication. A specific entity issued no command for a time period defined as one of the ACR parameters.
3. All open sessions are terminated after device (e.g. flash card) reset and/or power cycle.

#### Data Integrity services

**[00150]** The SSA system verifies the integrity of the SSA database (which contains all the ACRs, PCRs, etc...). In addition data integrity services are offered for entity data through the key ID mechanism.

**[00151]** If a key ID is configured with Hashed as its encryption algorithms the hash values are stored along side with the CEK and IV in the CEK record. Hash values are calculated and stored during write operation. Hash values are again calculated during read operations and compared with the values stored during the previous write operations. Every time the entity is accessing the key ID the additional data is concatenated (cryptographically) to the old data and the appropriate Hash value (for read or for write) updated.

**[00152]** Since only the host knows the data files associated with or pointed to by a key ID, the host explicitly manages several aspects of the data integrity function in the following manner:

1. A data file associated with or pointed to by a key ID is written or read from the beginning to end. Any attempt to access portions of the file will mess it up since the SSA system is using a CBC encryption method and generates a hashed message digest of the entire data
2. There is no need to process the data in a contiguous stream (the data stream can be interleaved with data streams of other key IDs and may be split over multiple sessions) since intermediate Hash values are maintained by the SSA system. However, the entity will need to explicitly instruct the SSA system to reset the Hash values if the data stream is restarted.
3. When a read operation is completed, the host must explicitly request the SSA system to validate the read Hash by comparing it with the Hash value calculated during the write operation.
4. The SSA system provides a “dummy read” operation as well. This feature will stream the data through the encryption engines but will not send it out to the host. This feature can be used to verify data integrity before it is actually read out of the device (e.g. flash card).

#### Random number generation

**[00153]** The SSA system will enable external entities to make use of the internal random number generator and request random numbers to be used outside of the SSA system. This service is available to any host and does not require authentication.

#### RSA key pair generation

**[00154]** The SSA system will enable external users to make use of the internal RSA key pair generation feature and request an RSA key pair to be used outside of the SSA system. This service is available to any host and does not require authentication.

#### Alternative Embodiment

**[00155]** Instead of using a hierarchical approach, similar results can be achieved using a data base approach, as illustrated in Fig. 18.



[00156] As shown in Fig. 18, a list of credentials for entities, authentication methods, the maximum number of failed attempts, and the minimum number of credentials needed to unblock may be entered into a database stored in controller 12 or memory 20, which relates such credential requirements to the policies (read, write access to keys and partitions, secure channel requirement) in the database carried out by the controller 12 of memory 10. Also stored in the database are constraints and limitations to the access to keys and partitions. Thus, some entities (e.g. system administrator) may be on a white list, which means that these entities can always access all keys and partitions. Other entities may be on a black list, and their attempts to access any information will be blocked. The limitation can be global, or key and/or partition specific. This means that only certain entities can always access certain specific keys and partitions, and certain entities always cannot do so. Constraints can also be put on the content itself, irrespective of the partition it is in or the key used to encrypt or decrypt it. Thus, certain data (e.g. songs) may have the attribute that they can only be accessed by the first five host devices that access them, or that other data (e.g. movies) can only be read for a limited number of times, irrespective of which entities had access.

## AUTHENTICATION

### Password Protection

- Password-protect means that a password needs to be presented to access the protected area. Unless it cannot be more than one password then passwords could be associated with different rights such as read access or read/write access.
- Password protect means that the device (e.g. flash card) is able to verify a password provided by the host i.e. the device also has the password stored in device managed secured memory area.

### Issues and limitations

- Passwords are subject to replay attack. Because the password does not change after each presentation it can be identically resent. It means that password as is must not be use if the data to be protected are valuable, and the communication bus is easily accessible.
- Password could protect access to stored data but should NOT be used to protect data (not a key)

To increase the security level associated with passwords, they can be diversified using a master key, with the result that hacking one does not crack entire system. A session key based secure communication channel can be use to send the password.

**[00157]** Fig. 19 is a flow chart illustrating authentication using a password. The entity sends in an account id and password to system 10 (e.g. flash memory card). The system checks to see if the password matches that in its memory. If it matches, authenticated status is returned. Otherwise, the error counter is incremented for that account, and the entity is asked to re-enter an account id and password. If the counter overflows, the system return status that access is denied.

#### Challenge Response

**[00158]** Fig. 20 is a Flow Chart illustrating authentication using a challenge/response type method. The entity sends in an account id and requests a challenge from system 10. System 10 generates a random number and presents it to the host. The host computes a response from the number and sends it to the system 10. System 10 compares the response to the value stored. The remaining steps are similar to those in Fig. 19 for determining whether to grant access.

**[00159]** Fig. 21 is a Flow Chart illustrating authentication using another challenge/response type method. Fig. 21 differs from that in Fig. 20 in that, in addition to requiring the host to be authenticated by the system 10, it also requires the system 10 to be authenticated by a challenge/response where system 10 also requests a challenge from the host and returns a response to be checked by the host.

**[00160]** Fig. 22 is a Flow Chart illustrating authentication using another challenge/response type method. In this case, only the system 10 needs to be authenticated, where the host sends a challenge to system 10, which computes a response that is checked by the host for a match with its record of system 10.

#### Symmetric key

**[00161]** Symmetric key algorithm means that the SAME key is used on both sides to encrypt and decrypt. It means that the key must have been pre-agreed prior to

communicating. Also each side should implement the reverse algorithm of each other i.e. encrypt algorithm on one side and decrypt on the other. Both sides do not need to implement both algorithms to communicate.

#### Authentication

- Symmetric key authentication means that device (e.g. flash card) and host share the same key and have the same cryptographic algorithm (direct and reverse e.g. DES and DES-1).

- Symmetric key authentication means challenge-response (protect against replay attack). The protected device generates a challenge for the other device and both compute the response. The authenticating device sends back the response and the protected device check the response and validate authentication accordingly. Then rights associated with authentication can be granted.

#### Authentication could be:

- External: the device (e.g. flash card) authenticates the outside world i.e. the device validates credentials of a given host or application
- Mutual: a challenge is generated on both sides
- Internal: the host application authenticates the device (e.g. flash card) i.e. host checks if device is genuine for its application.

To increase the security level of the entire system (i.e. breaking one does not break all)

- Symmetric key are usually combined with diversification using a master key
- Mutual authentication uses challenge from both side to ensure challenge is a real challenge

#### Encryption

Symmetric key cryptography is also used for encryption because it is a very efficient algorithm i.e. it does not need a powerful CPU to handle cryptography.

**[00162]** When used to secure a communication channel:

Both devices have to know the session key used to secure the channel (i.e. encrypt all outgoing data and decrypt all incoming data). This session key is usually established using a pre-shared secret symmetric key or using PKI.

Both devices have to know and implement the same cryptographic algorithms

#### Signature

**[00163]** Symmetric key can also be used to sign data. In that case the signature is a partial result of the encryption. Keeping the result partial allows to sign as many time as needed without exposing the key value.

#### Issues and Limitations

**[00164]** Symmetric algorithms are very efficient and secure but they are based on a pre-shared secret. The issue is securely share this secret in a dynamic manner and possibly to have it random (like a session key). The idea is that a shared secret is hard to keep safe in a long term and is almost impossible to share with multiple people.

**[00165]** To facilitate this operation, public key algorithm has been invented as it allows the exchange of secrets without sharing them.

#### Public key cryptography

**[00166]** Asymmetric key algorithm is commonly referred Public Key cryptograph. It is a quite complex and usually CPU intensive mathematical implementation. It has been invented to solve the issues of key distribution associated with symmetric key algorithms. It also provides signing capabilities used to ensure data integrity.

**[00167]** Asymmetric key algorithm uses a key which has private and public elements respectively referred as private key and public key. Both private key and public key are mathematically linked together. The public key can be shared whereas the private has to remain secret. As for the keys, asymmetric algorithm uses two mathematical functions (one for the private key and one for the public key) to provide wrap and unwrap or sign and verify.

### Key exchange and Key distribution

[00168] Key exchange becomes very simple using PK algorithm. The device sends its public key to the other device. The other device wraps its secret key with the public key and returns the encrypted data to the first device. The first device uses its private key to unwrap the data and retrieve the secret key which is now known to both sides and can be used to exchange data. Because the symmetric key can be exchanged that easily it is usually a random key.

### Signature

[00169] Because of its nature public key algorithm is usually used only to sign small amount of data. To ensure data integrity it is then combine with a hash function that provides a one-way foot print of the message.

[00170] The Private key is used to sign the data. The public key (freely available) allows to verify the signature.

### Authentication

[00171] Authentication usually uses signature: a challenge is signed and returned for validation

[00172] The public part of the key is used for verification. Because anyone can generate a key pair, there is a need to certify the owner of the public key in order to prove that this is the right person using the correct key. Certification authority provides certification and will include the public key in a signed certificate. The certificate is signed by the authority itself. Then using a public key to verify a signature means that the authority that issued the certificate containing that key is trusted and that one is able to verify that the certificate has not been hacked i.e. that the certificate hash signed by the authority is correct; meaning that the user has and trusts the authority public key certificate.

[00173] The most common way to provide PK authentication is to trust the authority or root certificate and indirectly trust all key pairs certified by the given authority. Authenticating is then a matter of proving that the private key that one has matches the certificate by signing a challenge and providing the challenge response,

and the certificate. Then certificate is checked to make sure it has not been hacked and it is signed by a trusted authority. Then the challenge response is verified. Authentication is successful if the certificate is trusted and the challenge response is correct.

**[00174]** Authentication in a device (e.g. flash card) means that the device is loaded with trusted root certificates and that the device is able to verify the challenge response as well as the certificate signed hash.

#### File encryption

**[00175]** PK algorithm is not used to encrypt large amounts of data because it is too CPU intensive, but is usually used to protect a randomized encryption / decryption key generated to encrypt the content. For example SMIME (Secure email) generate a key which is then encrypted with all recipients' public key.

#### Issues and Limitations

**[00176]** Because anything can generate a key pair, it has to be certified to ensure its origin. During key exchange one may want to make sure that the secret key is provided to the right device i.e. the origin of the provided public key will need to be checked. Certificate management then becomes part of the security as it can inform about the validity of the key, and whether the key has been revoked.

**[00177]** While the invention has been described above by reference to various embodiments, it will be understood that changes and modifications may be made without departing from the scope of the invention, which is to be defined only by the appended claims and their equivalent. All references referred to herein are incorporated by reference.

## 1 SSA Commands

The SSA system commands are passed to a memory card using standard (for the relevant form factor protocol) write and read commands. Therefore, from the host point of view, sending an SSA command really means writing data to a special file, on the memory device, used as the buffer file. Getting information from the SSA system is done via reading data from the buffer file. The host application must make sure data is always written and read from the first LBA of the buffer file. Managing the buffer files in the host OS is beyond the scope of this specification.

### 1.1 Communicating with the SSA system

The following sections define how SSA related commands and data are communicated with the SSA system by using the form factor standard write/read commands.

#### 1.1.1 Sending commands/data to the SSA system

The first data block of every write commands is scanned for a pass through signature. If found, the data is interpreted as an SSA commands. If not found, the data is written to the designated address.

SSA application specific write commands, may include multiple sector transfer where the first sector holds the required signatures and command's arguments and the rest of the data blocks hold the relevant data if any.

Table ... defines the format of the first block (data blocks are always 512 bytes as used in standard OS file systems) of an SSA command.

Byte Index	Length [bytes]	Description	Comments
0-31	32	Application pass through signature	Must be the ASCII string: "SSTA Pass Through Mode Supported"
32	4	SSA Application ID	Must be: 0x00000000
36	4	SSA session ID	The SSA session ID as provided by the SSA system through an authentication process. If no session is open, this field shall contain the value 0x00000000. When a secure channel is used the remainder of the command arguments (starting at byte offset 64 of the first block) and data blocks are encrypted with the session key
40	24	Reserved for future use	Data is undefined
64	4	SSA session ID	A second copy of the SSA session ID. This field is used to verify usage of session key.
68	4	SSA Application Command op-Code	As defined in the detailed SSA command description in the following sections

72	4	SSA application data blocks	Number of additional data blocks. 0 if no data blocks are used.
76-511	436	SSA Application command arguments	As defined in the detailed SSA command description in the following sections

**Table 1: SSA command Argument LBA Format****1.1.2 Reading data from the SSA system**

Read commands will be executed in two parts:

1. Initiating the *read* command by first sending a *write* command with a single data block defining all the arguments of the *read* command.
2. After the *write* command sets the card application on the correct state of transfer, a *read* command is used to initiate the actual data transfer from the card to the host. The read command must use the same LBA address the previous write command used. This is the only indication to the card that the host is trying to get the SSA data, previously requested.

The write/read command pairs must be carefully synchronized. The next session defines how sequence errors are handled and recovered from. As defined, the SSA system supports multiple host side users, which may be concurrently logged on. Each user is expected to, independently and asynchronously, initiate write/read command pairs hence, not requiring any special behavior of the host OS. From the card point of view these individual pairs are identified by the LBA address used in the write half of the sequence. From the host point of view it means each user must use a different file buffer.

**1.1.3 Write/read sequence errors****1.2 Commands Detailed Description**

Table 2 provides a general overview of the SSA commands.

The command name column provides a basic description of the commands usage and also index to the detail description of the command. The command op-code is the actual value used in the SSA command. Argument length (Arg Len) column defines the size of the argument field of the command (a value of zero means no argument). The arguments are command specific and specified in the detailed command description.

Data length is the size of the command data in the additional data blocks associated with the commands. A value of zero means no data, a value of "Var" means the command has variable data sizes and the actual size is specified in the command itself. For fixed sized data commands this column stores the size of data size.

Data direction can be either blank if the command has no data (meaning that the command arguments as specified in Table 1 all fit in the space between byte 76 and byte 511 – beyond this lays the data payload accompanying the command sector), "write" if the data is moving from the host to the card (appended to the argument block of the write command), or "read" if the data is moving from the card to the host (in read command following a write command that provides the arguments, as described above).

All size related columns use byte units.



Cmd Op-code	Cmd Name	Arg Len	Data Len	Data Dir	Description
ACR/AGP management commands					
1	CREATE_SYSTEM_ACR	1	0		Creates a system ACR entry in the SSA database and start the system ACR configuration sequence
2	SYS_ACR_CREATION_DONE	0	0		Terminates the system ACR configuration sequence and renders the system ACR active
3	PASSWORD_CREDENTIAL			Write	Provides the credential data for ACR that use password authentication.
4	SYMMETRIC_CREDENTIAL			Write	Provides the credential data for ACR that use symmetric authentication.
5	ASYMMETRIC_CREDENTIAL			Write	Provides the credential data for ACR that use asymmetric authentication.
6	GET_ACR_PUBLIC_KEY			Write	Get the public key of an ACR for signing by a CA. ACR RSA key pair where created in the SSA system when the ACR was created.
7	SEND_CERTIFICATE			Read	Provides a certificate signing the ACR public key
8	CONFIGURE_ACAM			Write	Sets the ACAM (ACR management permissions) record of an ACR.
9-15	Reserved for future use				
16	CREATE_ROOT_AGP			Write	Creates a root AGP entry in the SSA system data base
17	ROOT_AGP_CREATION_DONE	0	0		Terminates the configuration process of a root AGP and renders it active
18	DISABLE_SYSTEM_ACR_CREATION	0	0		Disables the feature of creating and configuring system ACR
19	SET_ROOT_AGP_CREATION_MODE	1			Defines the mode (open, controlled or block) of root AGP creation
20	DISABLE_ROOT_AGP_CHANGE_MODE				Disables the feature of changing the creation mode of root AGPs
21-25	Reserved for future use				
26	CREATE_AGP			Write	Creates an AGP entry in the SSA system data base
27	DELETE_AGP			Write	Deletes an AGP entry in the SSA system data base
28	CREATE_ACR			Write	Creates an ACR entry in the SSA

					system data base
29	CREATE_ACR_D ONE	0	0		Terminates the creation and configuration process of an ACR and renders it active
30	DELETE_ACR			Write	Deletes and ACR entry from the SSA system database.
31	UNBLOCK_ACR			Write	Unblocks a blocked (due to authentication failures) ACR
32-49	Reserved for future use				
Partition & Domain management commands					
50	CREATE_PARTIT ION			Write	This command splits a given partition into two. It can be issued only by a root ACR.
51	UPDATE_PARTIT ION			Write	Changes the size of two existing partitions. The net change to total size of these two partitions must be 0.
52	DELETE_PARTIT ION			Write	Merges two existing partitions into a single one.
53	RESTRIC_PUBLI C_PARTITION_A CCESS			Write	Enable/disable ac accessing the public partition of the device using the standard (not SSA) commands.
54-59	Reserved for future use				
60	CREATE_DOMAI N			Write	Creates a security domain in the SSA database
61	DELET_DOMAIN			Write	Deletes a security domain in the SSA database
62-69	Reserved for future use				
70	DELEGATE_DO MAIN_PERMISSI ONS			Write	Delegates access and ownership permission of a Domain to a specific ACR
71	DELEGATE_PAR TITION_PERMISS ION			Write	Delegates access permission of a Partition to a specific ACR
72-99	Reserved for future use				
System login and authentication commands					
100	SYSTEM_LOGIN			Write	
101	SYSTEM_LOGOU T	0	0		
102- 109	Reserved for future use				
110	SEND_PASSWOR D TO SSA			Write	
111- 119	Reserved for future use				
121	GET_SYMETRIC_ CHALLENGE			Read	

122	SEND_SYMMETRIC_CHALLENGE			Write	
123	GET_SYMMETRIC_CHALLENGE_RESPONSE			Read	
124	SEND_SYMMETRIC_CHALLENGE_RESPONSE			Write	
125-129	Reserved for future use				
130	SEND_ASYMMETRIC_CHALLENGE			Write	
131	GET_ASYMMETRIC_CHALLENGE			Read	
132	SEND_USER_CERTIFICATE			Write	
133	GET_SSA_PRE_MASTER_SECRET			Read	
134	GET_ACR_CERTIFICATE			Read	
135	SEND_HOST_PRE_MASTER_SECRET			Write	
136	START_SESSION			Write	
137	AUTHENTICATION_COMPLETE	0	0	Read	
138-199	Reserved for future use				
Read Write and status commands					
200	WRITE		Var	Write	Write data command
201	READ		Var	Read	Read data command
202	COMMAND_STATUS		Var	Read	Get the current SSA command execution status
203	SYSTEM_QUERY		Var	Read	Gets the requesting ACR current configuration data

**Table 2: SSA commands****1.2.1 Create System ACR**

Create system ACR builds a system ACR entry in the SSA database. Once the entry is created the credentials can be configured according to the specified login algorithm. Finally the CREATE\_SYSTEM\_ACR\_DONE command is used to terminate the sequence and render the system ACR active.

Create system ACR command will be rejected if an ACR entry already exists or create system ACR feature is disabled. System ACR may be configured only with a subset of the available login modes (refer to section 1.3.2 for details). If an invalid mode is used the command will be rejected.

Command arguments are given in Table 3. The byte offset is relative to the start of the command argument LBA (see section 1.1.1). Argument length is given in byte units. Argument name defines the purpose of the argument and can be used as index to the detailed argument description.

Byte Offset	Arg Length	Argument Name	Commentes
76	1	Login Algorithm	System ACR can be configured only with the following login algorithms: <ul style="list-style-type: none"> <li>• AES, DES, 3DES, Asymmetric authentications in mutual mode only.</li> </ul>

**Table 3: Create System ACR command argument**

### **1.2.2 System ACR Creation Done**

This command is sent only after the system ACR creation began. In any other time the command will be rejected. Sending this command ends the system ACR creation and will leave the ACR with the current configuration forever.

There are no arguments for this command.

•

### **1.2.3 PASSWORD CREDENTIAL**

After sending SSA command [28] – CREATE\_ACR – it is followed by sending the ACR's credentials. In this case it is password in a certain length – maximum length in bytes is 20.

Byte Offset	Arg Length	Argument Name	Commentes
76	As specified in the Password Length in Bytes argument field.	PASSWORD_CRE DENTIAL	See section 1.3.2 on the password phrase format and length.

**Table 4: Password Credential command arguments**

### **1.2.4 SYMMETRIC CREDENTIAL**

When picking a symmetrical login procedure for an ACR it will be followed by sending the ACR's symmetrical credential in the form of AES, DES or 3DES key. The nature of the algorithm will indicate the credential's (key) length in bytes. This command can be used at regular ACRs and system ACR creation time.

**Error! Reference source not found.** Table 13 describes the different types of asymmetrical credentials.

Byte Offset	Arg Length	Argument Name	Commentes
76	1	Credential Type	See Table 13 <b>Error! Reference source not found.</b> for type values and symbols

78	1	Credential Length in Bytes	
79	As specified in the Credential Length in Bytes field	Symmetric Credential	

Table 5: Symmetric Credential command arguments

**1.2.5 ASYMETRIC CREDENTIAL**

For an ACR with asymmetric login procedure there are several credentials that have to be passed to the SSA. The following Table 14 describes the different types of asymmetrical credentials:

Byte Offset	Arg Length	Argument Name	Commentes
76	1	Session ID	Session ID eliminates the need of the ACR ID. When in the case of the system ACR creation this field stays NULL.
77	1	Credential Type	See <b>Error! Reference source not found.</b> for type codes
78	1	Credential Length in Bytes	
79	As specified in the Credential Length in Bytes field	Symmetric Credential	

Table 6: Asymmetric Credential command arguments

**1.2.6 EXPORT PUBLIC KEY****1.2.7 IMPORT CERTIFICATE****1.2.8 CONFIGURE ACAM**

Sending this command configures the ACR management permissions. The command is sent during the ACR creation only. The command is not valid for system ACR. ACAM types and codes are described in Table 16: ACAM Types

Byte Offset	Arg Length	Argument Name	Commentes
76	1	Session ID	Valid only when used after system ACR login procedure.

			Otherwise left NULL.
77	1	AGP NAME/ID Length in Bytes	Maximum length is 20 bytes.
78	As specified in the AGP NAME/ID Length in Bytes argument field.	AGP NAME/ID	

Table 7: Configure ACAM command arguments

### 1.2.9 Create Root AGP

To create a root AGP under a secure channel, an SSA system login through the system ACR has to be executed. After the login, a session ID will be created and used for the creation sequence. The session ID is available when requesting system-command return status right after the system ACR login sequence is done.

Creating a root AGP without logging in to the system ACR first (create root AGP with a secure channel) does not require a session ID.

Table 8 reviews the commands arguments. When not using the system ACR the *Session ID* field is left with NULL (NA). AGP name/ID is preceded by the bytes number of its length.

Byte Offset	Arg Length	Argument Name	Commentes
76	1	Session ID	Valid only when used after system ACR login procedure. Otherwise left NULL.
77	1	AGP NAME/ID Length in Bytes	Maximum length is 20 bytes.
78	As specified in the AGP NAME/ID Length in Bytes argument field.	AGP NAME/ID	

Table 8: Create Root AGP command arguments

#### Command Structure:

- Command Name/OP Code – 1 Byte: SSA\_CREATE\_ROOT\_AGP\_CMD [3]
- Command Arguments –
  1. Session ID – is it needed???
  2. AGP Name/ID Length in Bytes – 1 Byte
  3. AGP Name/ID –

### 1.2.10 Root AGP Creation Done

This command is delivered when the root AGP is done – meaning all of the ACRs in the AGP are created. This command will *lock* the AGP so no more ACRs can be created.

There are no arguments for this command.

**Command Structure:**

- Command Name/OP Code – 1 Byte:  
SSA\_ROOT\_AGP\_CREATION\_DONE\_CMD [4]
- Command Arguments –
  1. Session ID – is it needed???
  2. AGP Name/ID Length in Bytes – 1 Byte
  3. AGP Name/ID –

**1.2.11 DISBALE SYSTEM ACR CREATION**

Sending this command will terminate the ability to create the system ACR.  
This command has no arguments.

**1.2.12 SET ROOT AGP CREATION MODE**

Controlling the creation of root AGPs is handled with SSA command [19]  
SET\_ROOT\_AGP\_CREATION\_MODE. The codes for the different modes is  
described in Table 9. This command does not require to login to the SSA thus there is  
no Session ID needed.

Mode Name	Code	Description
OPEN	1	Root AGP creation can be either through system ACR or regular open channel.
CONTROLLED	2	Root AGP creation through system ACR only.
LOCKED	3	No root AGP can be created.

**Table 9: Root AGP Creation Modes**

Byte Offset	Arg Length	Argument Name	Commentes
76	1	Root AGP Creation Mode.	

**Table 10: Set Root AGP Creation Mode command arguments****1.2.13 DISBALE ROOT AGP CHANGE MODE**

This command renders SET\_ROOT\_AGP\_CREATION\_MODE command inoperable  
and it will be rejected by the SSA. This command has no arguments.

**1.2.14 Create AGP**

Byte Offset	Arg Length	Argument Name	Commentes
76	1	Session ID	
77	1	AGP Name/ID Length in Bytes	Maximum length is 20 bytes.
78	As specified in the AGP NAME/ID Length in Bytes argument field.	AGP Name/ID	

**Table 11: Create AGP command arguments**

**Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_CREATE\_AGP\_CMD [5]
- Command Arguments –
  1. Session ID – 1 Byte
  2. AGP Name/ID Length in Bytes – 1 Byte
  3. AGP Name/ID –

**1.2.15 Delete AGP**

This command is valid for the ACR that created the AGP and given that it is empty of ACRs.

**Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_DELETE\_AGP\_CMD [6]
- Command Arguments –
  1. Session ID – 1 Byte
  2. AGP Name/ID Length in Bytes – 1 Byte
  3. AGP Name/ID –

**1.2.16 Create ACR****Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_CREATE\_ACR\_CMD [7]
- Command Arguments –
  1. AGP Name/ID –
  2. ACR Name/ID –
  3. Login Algorithm – 1 Byte
  4. Key Length
  5. Unblocking ACR Name/ID
  6. Number of Management Rights (ACAM) – 1 Byte
  7. ACAM #1
  8. ACAM #n

**1.2.17 Update ACR**

This command can be sent only by the ACR creator to update the child ACR. ACRs residing in the root AGP can't be updated, as they haven't a father ACR.

**Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_UPDATE\_ACR\_CMD [8]
- Command Arguments –
  1. Session ID – 1 Byte
  2. AGP Name/ID Length in Bytes – 1 Byte
  3. AGP Name/ID –
  4. ACR Name/ID Length in Bytes – 1Byte
  5. ACR Name/ID –

**1.2.18 Delete ACR**

This command can be sent only by the ACR creator to delete the child ACR. ACRs residing in the root AGP have the ability to delete themselves.

**Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_DELETE\_ACR\_CMD [9]
- Command Arguments –



1. Session ID – 1 Byte
2. AGP Name/ID Length in Bytes – 1 Byte
3. AGP Name/ID –
4. ACR Name/ID Length in Bytes – 1Byte
5. ACR Name/ID –

### **1.2.19 Unblock ACR**

This command can be sent only by an ACR with this explicit permission to unblock a certain ACR.

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_UNBLOCK\_ACR\_CMD [10]
- Command Arguments –
  1. Session ID – 1 Byte
  2. AGP Name/ID Length in Bytes – 1 Byte
  3. AGP Name/ID –
  4. ACR Name/ID Length in Bytes – 1Byte
  5. ACR Name/ID –

### **1.2.20 Delegate Domain Permissions**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_DELEGATE\_DOMAIN\_PERMISSION\_CMD [11]
- Command Arguments –
  1. Session ID – 1 Byte
  2. Number of Permissions to Delegate – 1 Byte
  3. Delegated Permission Code
  4. Domain Name/ID Length in Bytes – 1 Byte
  5. Domain Name/ID

### **1.2.21 Create Partition**

This command can be sent only by an ACR residing in a root AGP.

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_CREATE\_PARTITION\_CMD [12]
- Command Arguments –
  1. Session ID – 1 Byte
  2. Partition Name/ID Length in Bytes – 1 Byte
  3. Partition Name/ID
  4. Partition Size in Sectors [512 bytes] – 4 Bytes
  5. Decreased Partition Name/ID Length in Bytes – 1 Byte
  6. Decreased Partition Name/ID

### **1.2.22 Update Partition**

This command can be sent only by an ACR residing in a root AGP.

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_UPDATE\_PARTITION\_CMD [13]

- Command Arguments –
  1. Session ID – 1 Byte
  2. Partition Name/ID Length in Bytes – 1 Byte
  3. Partition Name/ID
  4. Partition Size in Sectors [512 bytes] – 4 Bytes
  5. Decreased Partition Name/ID Length in Bytes – 1 Byte
  6. Decreased Partition Name/ID

### **1.2.23 Delete Partition**

This command can be sent only by an ACR residing in a root AGP.

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_DELETE\_PARTITION\_CMD [14]
- Command Arguments –
  6. Session ID – 1 Byte
  7. Partition Name/ID Length in Bytes – 1 Byte
  8. Partition Name/ID

### **1.2.24 Restrict Public Domain Access**

This command will restrict regular Read/Write commands (commands sent by the host and are not part of the SSA command protocol) to/from the public partition (a.k.a. *user area*).

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_RESTRICT\_PAUBLIC\_PARTITION\_CMD [15]
- Command Arguments –
  1. Session ID – 1 Byte
  2. Public Partition Restriction Code – 1 Byte

### **1.2.25 Create Domain**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_CREATE\_DOMAIN\_CMD [16]
- Command Arguments –
  1. Session ID – 1 Byte
  2. Partition Name/ID Length in Bytes – 1 Byte
  3. Partition Name/ID
  4. Domain Name/ID Length in Bytes – 1 Byte
  5. Domain Name/ID

### **1.2.26 Delete Domain**

Only the Domain owner may send this command and delete a Domain.

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_DELETE\_DOMAIN\_CMD [17]
- Command Arguments –
  1. Session ID – 1 Byte
  2. Partition Name/ID Length in Bytes – 1 Byte
  3. Partition Name/ID

4. Domain Name/ID Length in Bytes – 1 Byte
5. Domain Name/ID

### **1.2.27 System Login**

This command is issued when a host user wishes to use the SSA system through one of the ACRs. The command will start the login/authentication process.

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_SYSTEM\_LOGIN\_CMD [18]
- Command Arguments –
  1. AGP Name/ID Length in Bytes – 1 Byte
  2. AGP Name/ID –
  3. ACR Name/ID Length in Bytes – 1Byte
  4. ACR Name/ID –

### **1.2.28 System Logout**

This command is issued when the host user wishes to terminate a working session with the SSA system. The command ends all of the user activity for the current login session. After this command the host user will need to start the login process again to be able to execute further actions with the SSA system.

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_SYSTEM\_LOGOUT\_CMD [19]
- Command Arguments –
  1. AGP Name/ID Length in Bytes – 1 Byte
  2. AGP Name/ID –
  3. ACR Name/ID Length in Bytes – 1Byte
  4. ACR Name/ID –

### **1.2.29 Read**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_READ\_CMD [20]
- Command Arguments –
  1. Session ID – 1 Byte
  2. Partition Name Length in Bytes - 1 Byte
  3. Partition Name
  4. Domain Name Length in Bytes – 1 Byte
  5. Domain Name
  6. Partition Address (LBA) – 4 Bytes
  7. Number of LBAs (Sectors – Sector=512bytes) to read – 4 Bytes

### **1.2.30 Write**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_WRITE\_CMD [21]
- Command Arguments –
  1. Session ID – 1 Byte
  2. Partition Name Length in Bytes - 1 Byte
  3. Partition Name
  4. Domain Name Length in Bytes – 1 Byte

5. Domain Name
6. Partition Address (LBA) – 4 Bytes
7. Number of LBAs (Sectors – Sector=512bytes) to read – 4 Bytes

### **1.2.31 Command Status**

This status command can be sent to get the return status of the previous command sent. The status deals with the command process and SSA system state.

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_CMD\_STATUS\_CMD [22]
- Command Arguments –
  1. Session ID – 1 Byte

### **1.2.32 System Query**

The system query command reads SSA information that is in the scope of the ACR that is logged in.

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_SYS\_QUERY\_CMD [23]
- Command Arguments –
  1. Session ID – 1 Byte

### **1.2.33 Password Authentication Commands**

#### **1.2.33.1 Send Password To SSA**

The command sends the actual ACR password to be verified by the SSA. Sending the *Command Status* command (22) will, the host will be able to read the command status and upon command completion the status of the authentication process – PASS/FAIL.

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_PWD\_AUTH\_SEND\_PWD\_CMD [24]
- Command Arguments –
  1. Password Length in Bytes – 1 Byte
  2. Password Data.

### **1.2.34 Symmetrical Authentication Commands**

#### **1.2.34.1 Get Challenge from SSA**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_SYM\_AUTH\_GET\_CHLG\_CMD [25]
- Command Arguments –

#### **1.2.34.2 Send Challenge to SSA**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte: SSA\_SYM\_AUTH\_SEND\_CHLG\_CMD [26]
- Command Arguments –

### **1.2.34.3 Get Challenge Response from SSA**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte:  
SSA\_SYM\_AUTH\_GET\_CHLG\_RES\_CMD [27]
- Command Arguments –

### **1.2.34.4 Send Challenge Response from SSA**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte:  
SSA\_SYM\_AUTH\_SEND\_CHLG\_RES\_CMD [28]
- Command Arguments –

## **1.2.35 Asymmetrical Authentication Process Commands**

### **1.2.35.1 Send Challenge to SSA**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte:  
SSA\_ASYM\_AUTH\_SEND\_CHLG\_CMD [29]
- Command Arguments – Challenge random number – 28 Bytes

### **1.2.35.2 Get Challenge from SSA**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte:  
SSA\_ASYM\_AUTH\_GET\_CHLG\_CMD [30]
- Command Arguments – NA

### **1.2.35.3 Send CA Certificate to SSA**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte:  
SSA\_ASYM\_AUTH\_SEND\_CA\_CERT\_CMD [31]
- Command Arguments –

### **1.2.35.4 Get SSA Pre-Master Secret**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte:  
SSA\_ASYM\_AUTH\_GET\_PRE\_MASTER\_SECRET\_CMD [32]
- Command Arguments –

### **1.2.35.5 Get ACR Certificate from SSA**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte:  
SSA\_ASYM\_AUTH\_GET\_CHLG\_CMD [33]
- Command Arguments –

### **1.2.35.6 Send Host Pre-Master Secret to SSA**

#### **Command Structure:**

- Command Name/OP Code – 1 Byte:  
SSA\_ASYM\_AUTH\_SEND\_PRE\_MASTER\_SECRET\_CMD [34]
- Command Arguments –

### 1.2.35.7 Send *Start Session* Message

#### Command Structure:

- Command Name/OP Code – 1 Byte:  
SSA\_ASYM\_AUTH\_SEND\_START\_SESSION\_MSG\_CMD [35]
- Command Arguments –
  1. PIN option –
  2. PIN Length in Bytes –
  3. PIN string –

### 1.2.35.8 Get *Authentication Complete* Message from SSA

#### Command Structure:

- Command Name/OP Code – 1 Byte: SSA\_SYM\_AUTH\_GET\_CHLG\_CMD [36]
- Command Arguments –

## 1.3 SSA Command Arguments

### 1.3.1 Not Applicable

- –

All fields defined as Not Applicable (NA) in the argument list must be set to 0.

### 1.3.2 Password and PIN Structure

Password and PIN phrases are 20 bytes long and are of binary value to the SSA system. Any phrase shorter than 20 bytes must be padded with '0'.

'0' Padding							Phrase													LSE
MSB																				0
19																				
00	00	00	00	00	00	00	00	49	F3	70	15	CC	52	74	A1	EC	2B	00	01	05

### 1.3.3 Login Algorithm

This argument defines the login algorithm of an ACR. It is a 1 byte long. Available values defined in the following table:

Symbol	Value	Description
NONE	0	No authentication is required. The session is opened as soon as the system login command is issued for this ACR.
PASSWORD	1	Password based authentication
Reserved for future use	2-9	
AES_HOST_AUTH	10	One way symmetric authentication using AES algorithm. Card is authentication user.
AES_HOST_AUTH_SEC	11	One way symmetric authentication using AES algorithm. Card is

		authenticating user. Secure channel is established and used for this ACR.
AES_HOST_AUTH_SEC_PIN	12	One way symmetric authentication using AES algorithm. Card is authenticating user. Secure channel is established and used for this ACR. Authentication is complete after an additional PIN is provided.
AES_MUTUAL_AUTH	13	Two way symmetric authentication using AES algorithm. Card and host authenticate each other
AES_MUTUAL_AUTH_SEC	14	Two way symmetric authentication using AES algorithm. Card and host authenticate each other. Secure channel is established and used for this ACR.
AES_MUTUAL_AUTH_SEC_PIN	15	A two-factor authentication using AES algorithm. Card and host authenticate each other. Secure channel is established and used for this ACR. Authentication is complete after an additional PIN is provided.
Reserved for future use	16-19	
DES_HOST_AUTH	20	Similar to the AES group of login modes with the exception that DES algorithm is used.
DES_HOST_AUTH_SEC	21	
DES_HOST_AUTH_SEC_PIN	22	
DES_MUTUAL_AUTH	23	
DES_MUTUAL_AUTH_SEC	24	
DES_MUTUAL_AUTH_SEC_PIN	25	
Reserved for future use	26-29	

3DES_HOST_AUTH	30	Similar to the AES group of login modes with the exception that 3DES algorithm is used.
3DES_HOST_AUTH_SEC	31	
3DES_HOST_AUTH_SEC_PIN	32	
3DES_MUTUAL_AUTH	33	
3DES_MUTUAL_AUTH_SEC	34	
3DES_MUTUAL_AUTH_SEC_PIN	35	
Reserved for future use	36-39	
RSA_HOST_AUTH	40	
RSA_HOST_AUTH_PIN	41	
RSA_MUTUAL_AUTH	42	
RSA_MUTUAL_AUTH_PIN	43	
Reserved for future use	44-255	

Table 12: Login Algorithm Types

#### 1.3.4 Symmetric Credentials Symbols

Symbol	Value	Description
SYMMETRIC_KEY	1	A symmetric key that corresponds to the chosen symmetric authentication sequence. The chosen authentication sequence will also reflect on the key length.
USER_PIN	2	PIN is binary value of maximum 20 bytes

Table 13: Symmetrical Credential Types

#### 1.3.5 Asymmetrical Credential Types

Symbol	Value	Description
CA_ID		
CA_PUBLIC_RSA_KEY	1	
ACR_CERTIFICATE	2	
USER_PIN	4	

Table 14: Asymmetrical Credential Types

#### 1.3.6 Partition Rights

Partition Rights BYTE Bitmap							
Read	Write	Delegate	Reserved	Reserved	Reserved	Reserved	Reserved



**1.3.7 Domain Rights**

Domain Rights BYTE Bitmap							
Read	Write	Delegate	Reserved	Reserved	Reserved	Reserved	Reserved

**1.3.8 Domain Permission Codes**

Symbol	Value	Description
READ	1	
WRITE	2	
DOMAIN_PERMISSION_DELEGATION	3	
DOMAIN_OWNERSHIP	4	

Table 15: Domain Permission Types

**1.3.9 ACAM**

Symbol	Value	Description
CREATE_AGP	1	
ACAM_CREATE_ACR	2	Create/delete/update AGPs and ACR.
ACAM_CREATE_PARTITION	3	Create/delete Partitions.
ACAM_CREATE_DOMAIN	4	Create/delete Domains.
ACAM_DELEGATE_DOMAIN_RIGHTS	5	Delegate access rights to domain – this one is per domain.
ACAM_DELEGATE_PARTITION_RIGHTS	6	Delegate access rights to partitions – this one is per partition.
UNBLOCK_ACR	7	

Table 16: ACAM Types

**1.3.10 Public Partition restriction Codes**

Symbol	Value	Description
READ_RESTRICTION	1	
WRITE_RESTRICTION	2	
READ_WRITE_RESTRICTION	3	

Table 17: Public Partition Restriction Types

**1.3.11 command Status**

Field Name	Contents	Number of Bytes
Session ID	ID number	1
Last Command OP-Code	A valid SSA command OP-Code	1

Last Command Status	<ul style="list-style-type: none"> <li>• COMPLETE_OK – 0</li> <li>• COMPLETE_ERROR – 1</li> <li>• BUSY – 2</li> </ul>	1
Error Code		1
Authentication state	Applicable only for authentication commands	1
Number of transferred sectors	Applicable for only for data transfer commands	

### 1.3.12 SSA Query

Field Name	Contents	Number of Bytes
Session ID	ID number	1
Last Command OP-Code	A valid SSA command OP-Code	1
Last Command Status	<ul style="list-style-type: none"> <li>• COMPLETE_OK – 0</li> <li>• COMPLETE_ERROR – 1</li> <li>• BUSY – 2</li> </ul>	1
Error Code		1
SSA Version	Version number	
List of accessible partitions	The partition ID, net size and access permissions	
List of accessible domains	The domain ID and access permissions	

### 1.3.13 Command Sequences

#### 1.3.13.1 Command Sequence for SSA login via Mutual Symmetric Authentication

Sequence Index	Command Name & Op-Code	Argument Description	General Description
1.	SSA_SYSTEM_LOGIN_CMD [18]	ACR & AGP names	Starts the login sequence. Acts as a request only.
2.	SSA_CMD_STATUS_CMD [22]	Session ID – NA	Get status on CMD18. If

			CMD18 fails then the login sequence terminates.
3.	SSA_SYM_AUTH_SEND_CHLG_CMD [26]	Challenge #1	Send challenge #1 to the SSA
4.	SSA_CMD_STATUS_CMD [22]	Session ID – NA	Get status on CMD26. If CMD26 fails then the login sequence terminates.
5.	SSA_SYM_AUTH_GET_CHLG_RES_CMD [27]	NA	Read the SSA response to challenge #1. The host verifies that the response is valid.
6.	SSA_CMD_STATUS_CMD [22]	Session ID – NA	Get status on CMD27. If CMD27 fails then the login sequence terminates.
7.	SSA_SYM_AUTH_GET_CHLG_CMD [25]	NA	Read challenge #2 from the SSA.
8.	SSA_CMD_STATUS_CMD [22]	Session ID – NA	Get status on CMD25. If CMD25 fails then the login sequence terminates.
9.	SSA_SYM_AUTH_SEND_CHLG_RES_CMD [28]	Challenge #2 response	Send challenge #2 response to the SSA.
10.	SSA_CMD_STATUS_CMD [22]	Session ID – NA	Get status on CMD28. If CMD28 fails then the login sequence terminates. At this stage the command status should show if the authentication process completed successfully or failed.

When this sequence is done successfully the SSA's ACR is logged in and SSA operations can begin.

### 1.3.13.2 Command Sequence for creating a root AGP

A root AGP can be created either via the system ACR (which requires to execute a login sequence to the system ACR) or forgo the secure channel and skip the system ACR authentication process. Command **SSA\_CREATE\_ROOT\_AGP\_CMD** [3] is sent with the root AGP's identity.

This command can be followed by **SSA\_CMD\_STATUS\_CMD** [22] to make sure that the SSA did not reject the command and that it was done without an error.

When the root AGP is done and all of its ACRs are created then to *seal* the root AGP, **SSA\_ROOT\_AGP\_CREATION\_DONE\_CMD** [4] command will be sent.

### 1.3.13.3 Command Sequence for Creating an AGP

To create an AGP, the user must first login to the SSA by executing the login command sequence shown in 1.3.13.1. The AGP must be created before creating new group of ACRs. The AGP is created by sending command **SSA\_CREATE\_AGP\_CMD** [5] with the AGP Name/ID.

To verify that CMD [5] was received and executed without an error the user sends **SSA\_CMD\_STATUS\_CMD** [22] and reads the status of the previous sent command. When the user is done creating the AGP he can proceed with creating an ACR or logout from the SSA system.

### 1.3.13.4 Command Sequence for Creating an ACR

To create an ACR, the user must first login to the SSA by executing the login command sequence shown in 1.3.13.1. Also, there must be an AGP where the new ACR belong. Then the user sends command **SSA\_CREATE\_ACR\_CMD** [7] with all of the new ACR data (name, AGP, login methods...etc.). To verify that CMD [7] was received and executed without an error the user sends **SSA\_CMD\_STATUS\_CMD** [22] and reads the status of the previous sent command. When the user is done creating the ACR he can proceed with other SSA operations or logout from the SSA system.

## 1.4 Product parameters

- Maximum number of all entities (MAROs, ARCR, sessions in parallel, etc...)
- Add definition for cryptographic parameter, where applicable. i.e. RSA key length.
- Need to define error conditions and messages per protocols.
- Need to define time out and busy handling.
- Specify number of levels on the trees
- limit # of root MAROS
- limit # of Childs (on the root) on all? delegate up to.
- There will be a limit on the number of CBC contexts in parallel, such as 5-10
- Protocol and product versions

**WHAT IS CLAIMED IS:**

1. A secure storage method for use in a storage system, comprising a non-volatile memory; and a controller controlling access to the memory, said method comprising:

creating, by means of a system agent stored in the controller or the memory, at least one hierarchical tree comprising nodes at different levels for controlling access to data stored in the memory by corresponding entities, wherein each node of the at least one tree specifies permission(s) of a corresponding entity or entities for accessing memory data, wherein permission(s) at a node of each of the trees has a predetermined relationship to permission(s) at another node at a higher or lower level in the same tree; and

using the at least one hierarchical tree to control access to memory data.

2. The method of claim 1, wherein the creating creates at least two trees, and there is no cross-talk between the at least two trees created by the agent.

3. The method of claim 1, wherein the first node of any tree created by the agent is the root node, said method further comprising disabling the agent, wherein the root node of any tree(s) created by the agent cannot be altered after the agent is disabled.

4. The method of claim 1, further comprising disabling the agent, wherein no additional tree(s) can be created.

5. The method of claim 1, wherein said permission(s) at a node of each of the trees indicate access rights to data in the memory not less than those indicated by permission(s) at a node at a lower level in the same tree.

6. The method of claim 1, wherein a node in a tree has the capability to create a child node at a lower level for enabling access by an entity to data in the memory, said method further comprising creating the child node.

7. The method of claim 6, wherein the permission(s) of the child node is not greater than that of the node that created the child node.

8. The method of claim 1, wherein said permission(s) at a node of a tree permits a corresponding entity to create and/or delete another node in the same tree, and/or alter the permission at a node in the same tree, and/or delegate permission(s) to another node in the same tree and/or transfer association with a node to another node in the same tree, said method further comprising deleting another node, altering permission at a node, delegating permission(s) to another node and/or transfer association with a node to another node, in the same tree, in accordance with the permission(s).

9. The method of claim 1, wherein said permission(s) at a node of a tree is for access to a key for encrypting and/or decrypting data in the memory.

10. The method of claim 1, wherein said permission(s) at a node of a tree is for access to one or more partitions of the memory, said method further comprising accessing the one or more partitions of the memory, in accordance with the permission(s).

11. The method of claim 1, wherein the permission(s) at a node permits key(s) for data encryption and/or decryption to be created on behalf of a corresponding entity, said method further comprising creating said key(s), in accordance with the permission(s).

12. The method of claim 11, wherein the permission(s) at a node permits the corresponding entity to delegate ownership of key(s) created on its behalf and/or permission(s) of access to key(s), said key(s) used in encrypting and/or decrypting data stored in the memory, said method further comprising delegating ownership of key(s) created on its behalf and/or permission(s) of access to key(s), in accordance with the permission(s).

13. The method of claim 11, wherein the permission(s) at a node permits a corresponding entity for whom key(s) have been created to delete such key(s), said method further comprising deleting such key(s), in accordance with the permission(s).

14. A secure storage system, comprising:

a non-volatile memory; and

a controller controlling access to the memory, said controller or memory storing a system agent that is capable of creating at least one hierarchical tree comprising nodes at different levels for controlling access to data stored in the memory by corresponding entities, wherein each node of the at least one tree specifies permission(s) of a corresponding entity or entities for accessing memory data, wherein permission(s) at a node of each of the trees has a predetermined relationship to permission(s) at another node at a higher or lower level in the same tree.

15. The system of claim 14, wherein there is no cross-talk between at least two of the trees created by the agent.

16. The system of claim 14, wherein the first node of any tree created by the agent is the root node, and wherein when the agent is disabled, the root node of any tree(s) created by the agent cannot be altered.

17. The system of claim 14, wherein when the agent is disabled, no additional tree(s) can be created.

18. The system of claim 14, wherein said permission(s) at a node of each of the trees indicate access rights to data in the memory not less than those indicated by permission(s) at a node at a lower level in the same tree.

19. The system of claim 14, wherein a node in a tree has the capability to create a child node at a lower level for enabling access by an entity to data in the memory.

20. The system of claim 19, wherein the permission(s) of the child node is not greater than that of the node that created the child node.

21. The system of claim 14, wherein said permission(s) at a node of a tree permits a corresponding entity to create and/or delete a node for another node in the same tree, and/or alter the permission at a node in the same tree, and/or delegate permission(s) to a node for another node in the same tree or transfer association with a node to another node in the same tree.

22. The system of claim 14, wherein said permission(s) at a node of a tree is for access to a key for encrypting or decrypting data in the memory.

23. The system of claim 14, wherein said permission(s) at a node of a tree is for access to one or more partitions of the memory.

24. The system of claim 14, wherein the permission(s) at a node permits key(s) for data encryption and/or decryption to be created on behalf of a corresponding entity.

25. The system of claim 24, wherein the permission(s) at a node permits the corresponding entity to delegate ownership of key(s) created on its behalf and/or permission(s) of access to key(s), said key(s) used in encrypting and/or decrypting data stored in the memory.

26. The system of claim 24, wherein the permission(s) at a node permits a corresponding entity for whom key(s) have been created to delete such key(s).



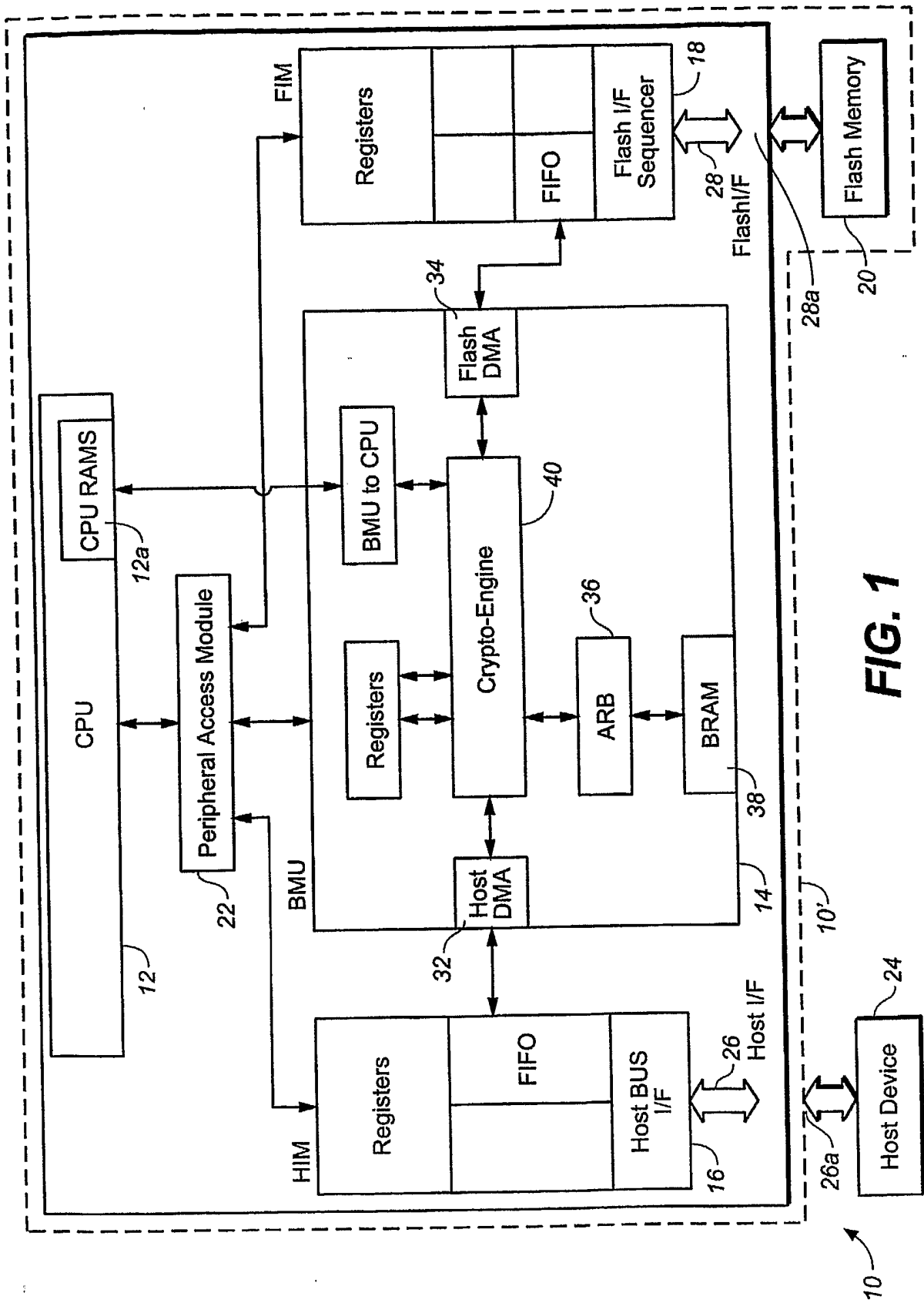
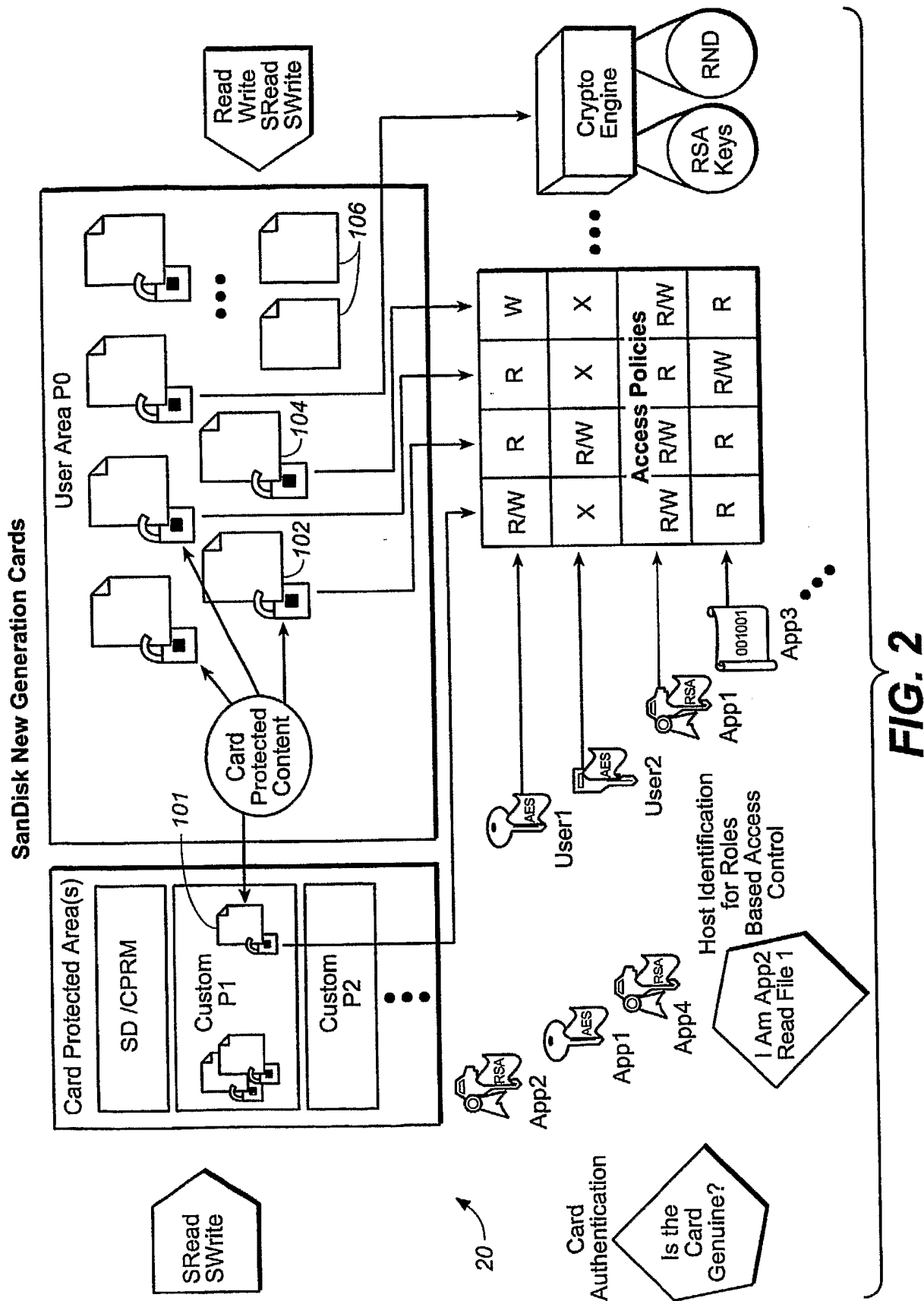


FIG. 1

2 / 14



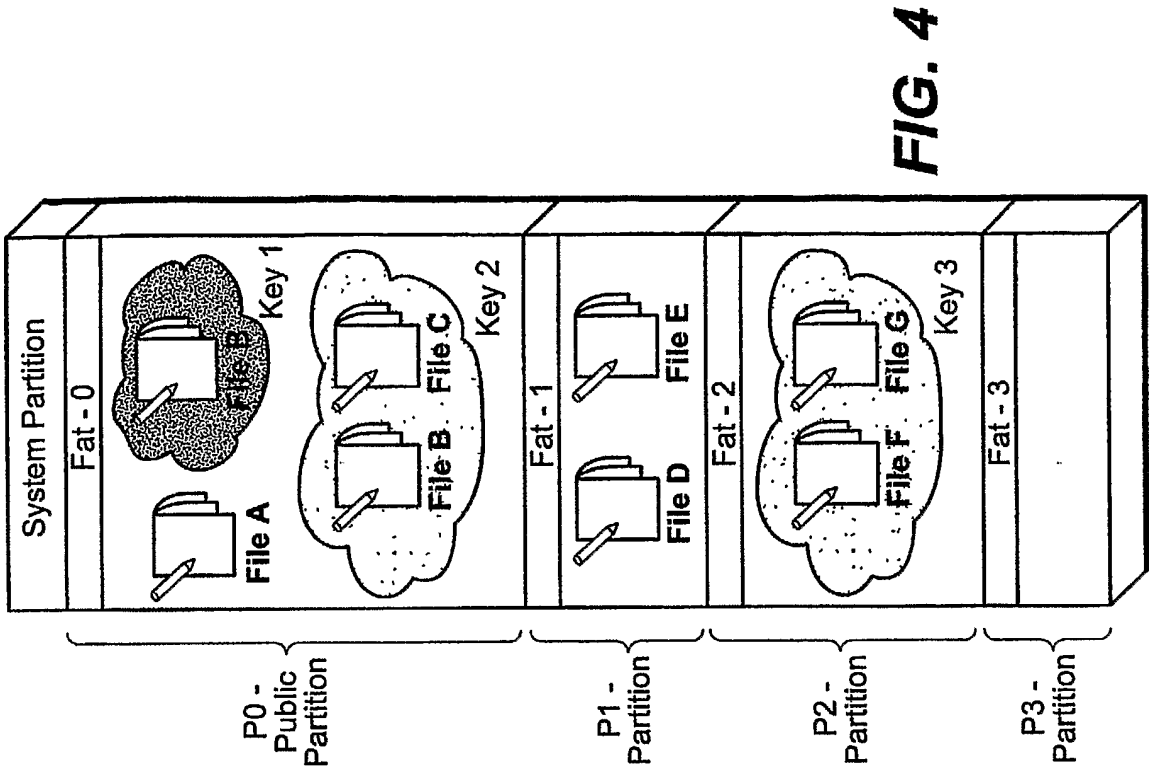


FIG. 4

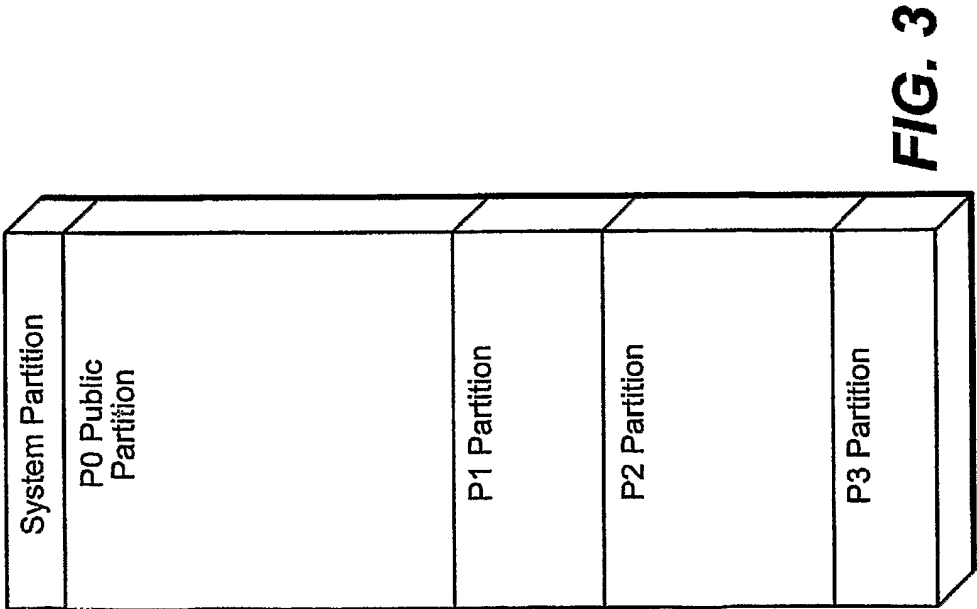
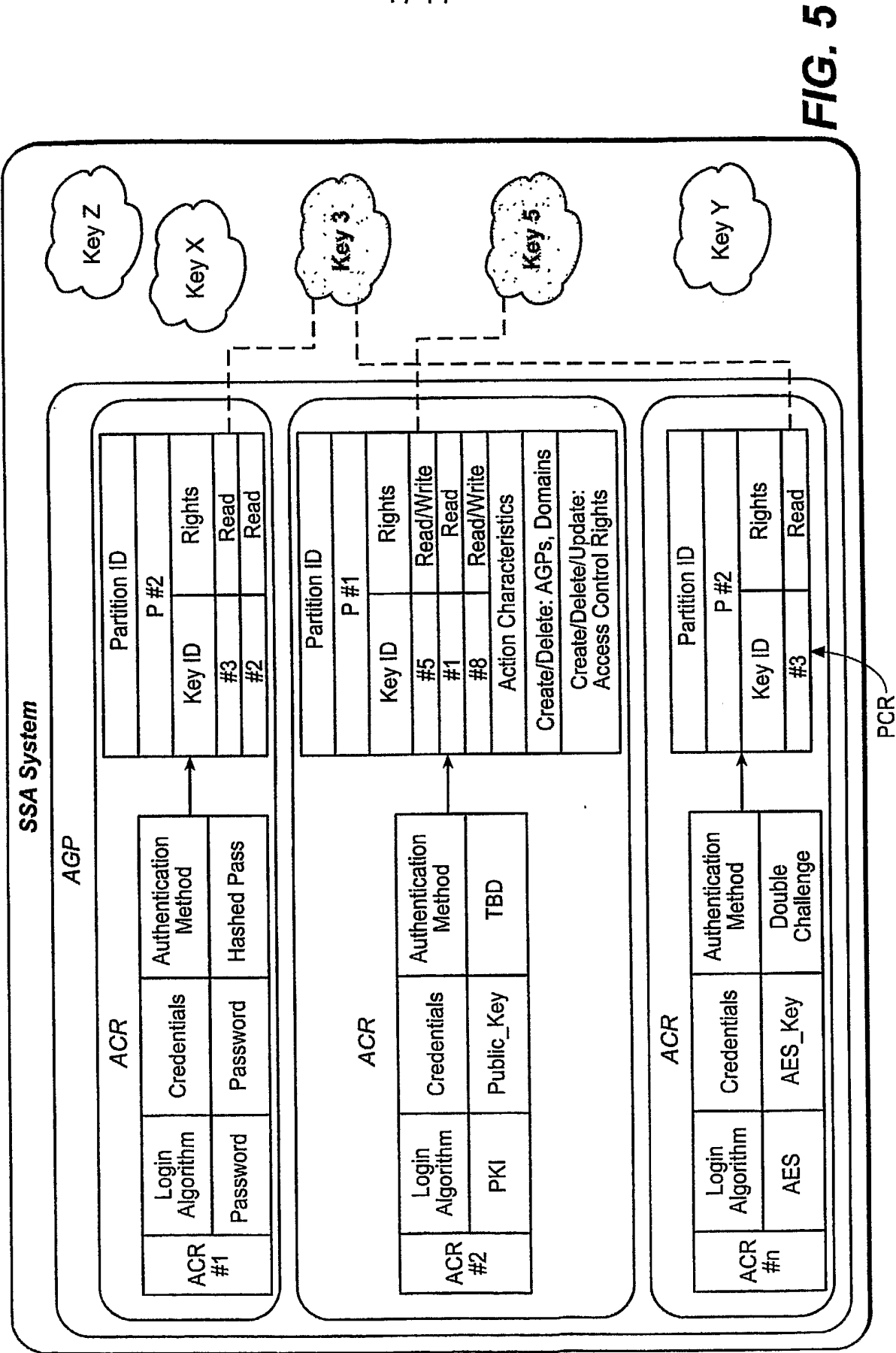
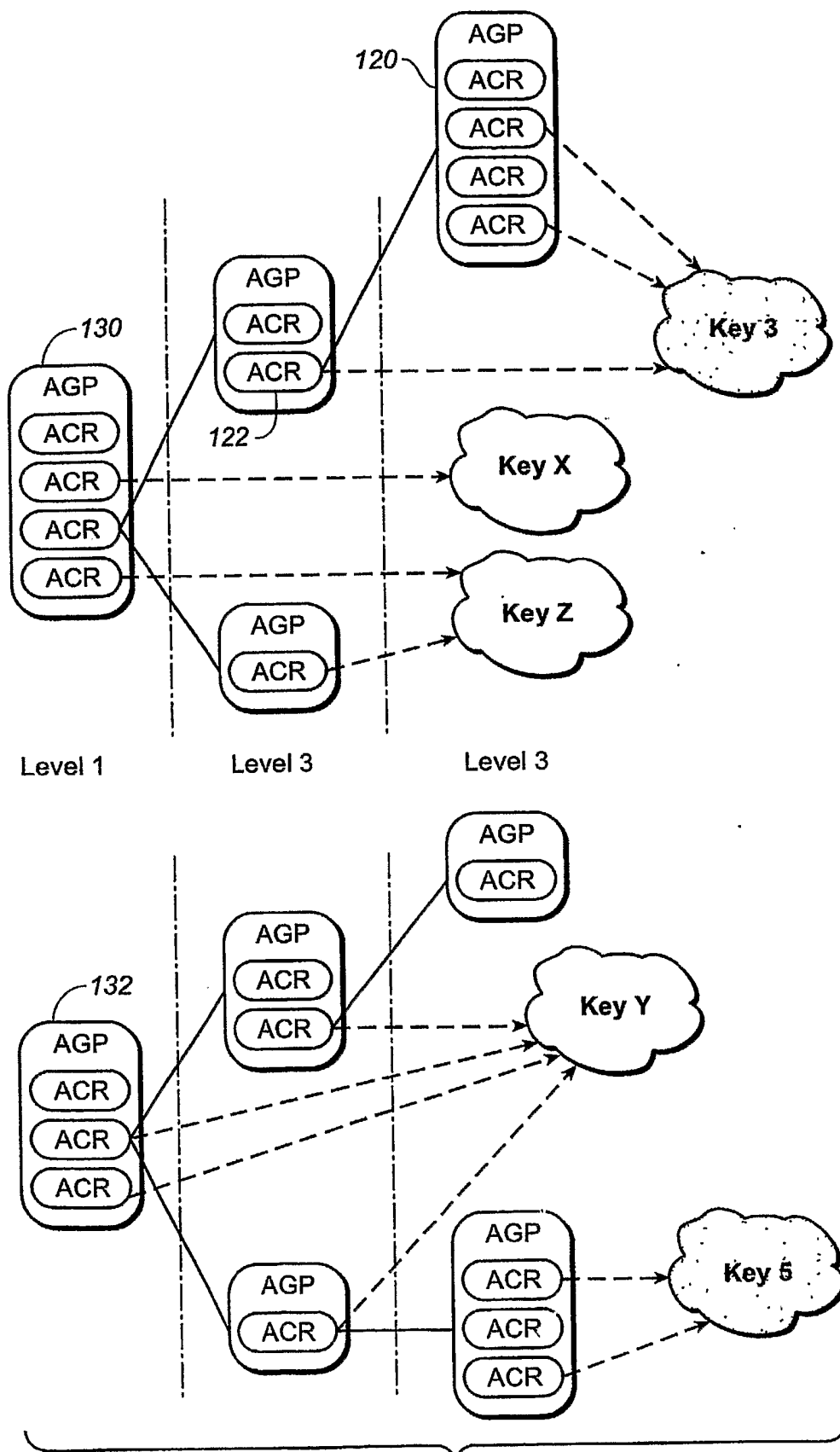


FIG. 3



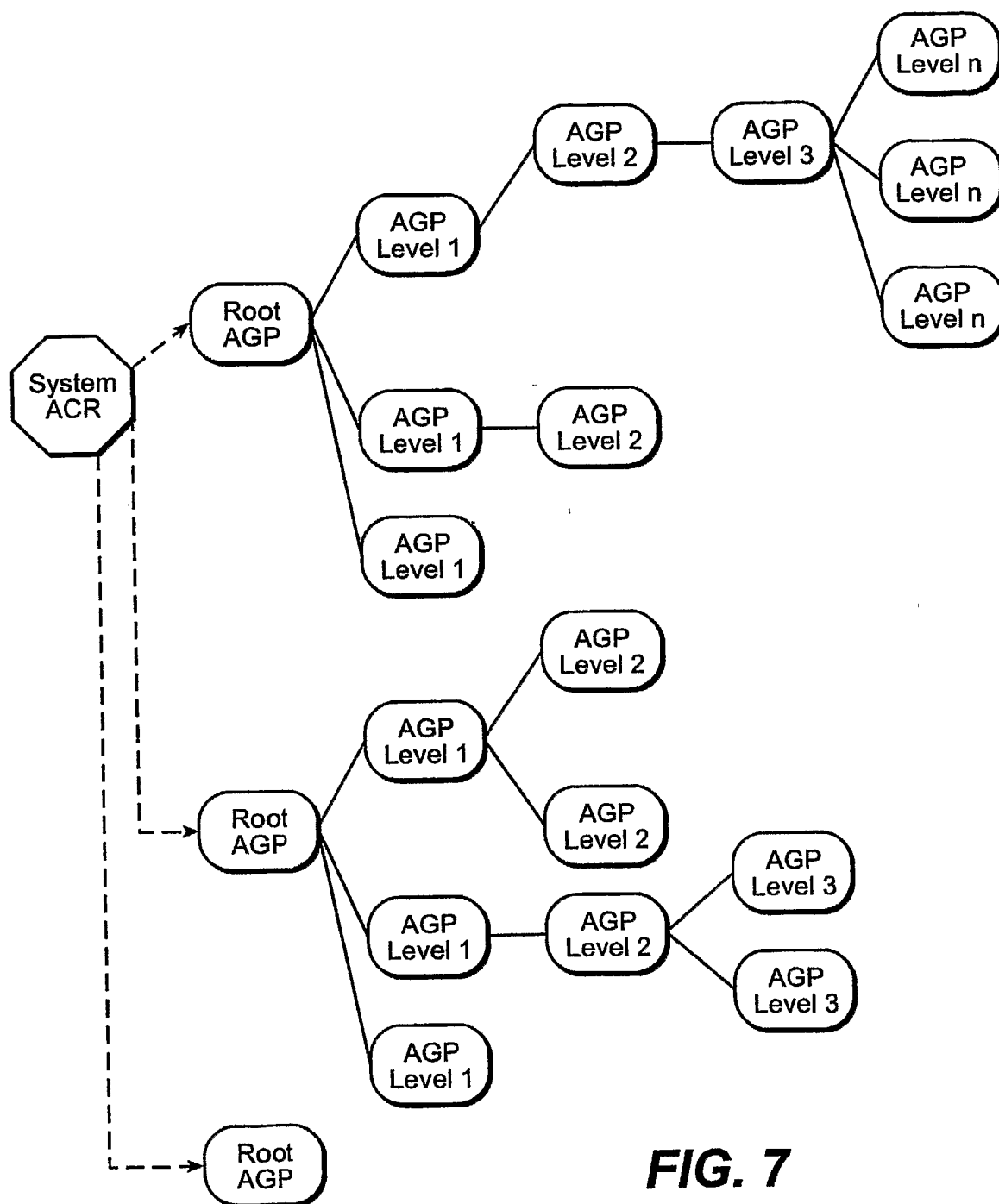
**FIG. 5**

5 / 14

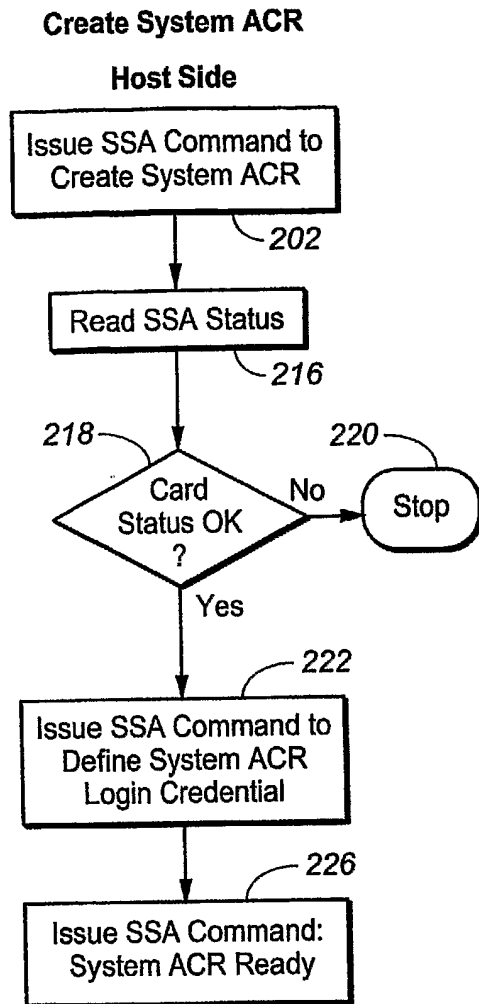
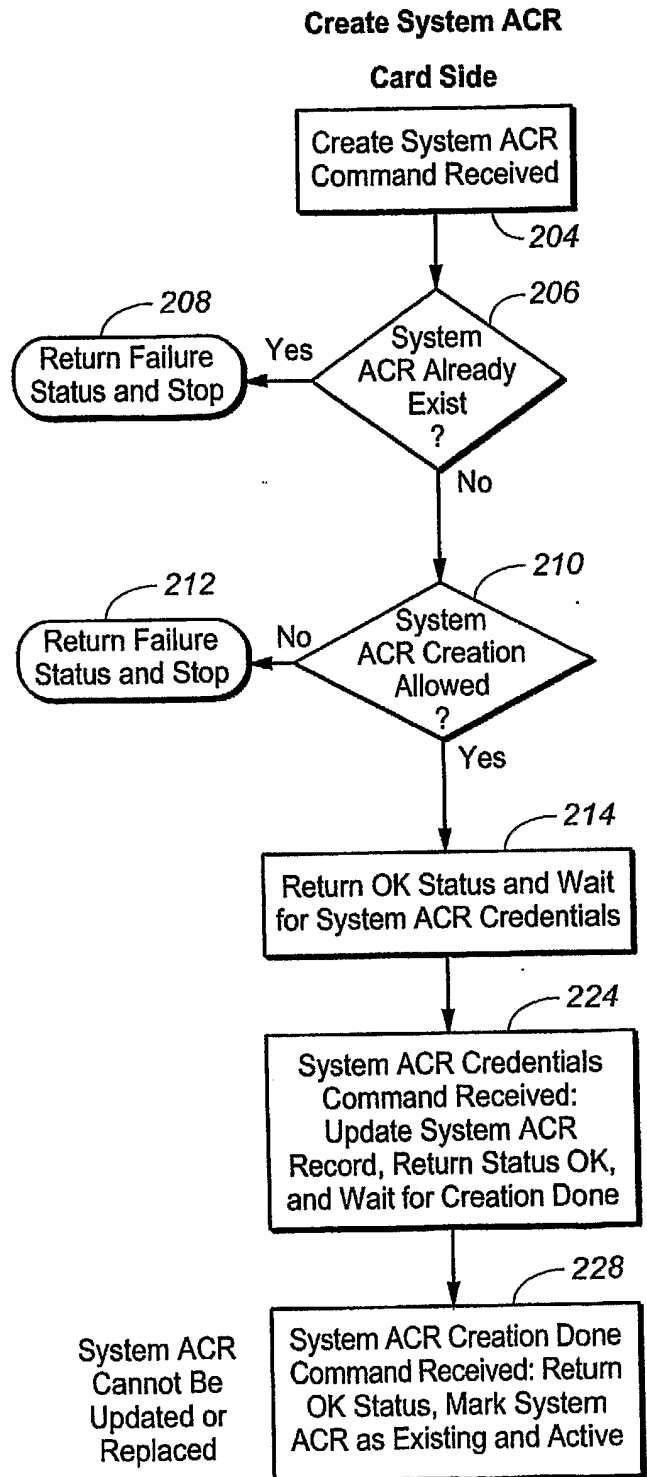


**FIG. 6**  
SUBSTITUTE SHEET (RULE 26)

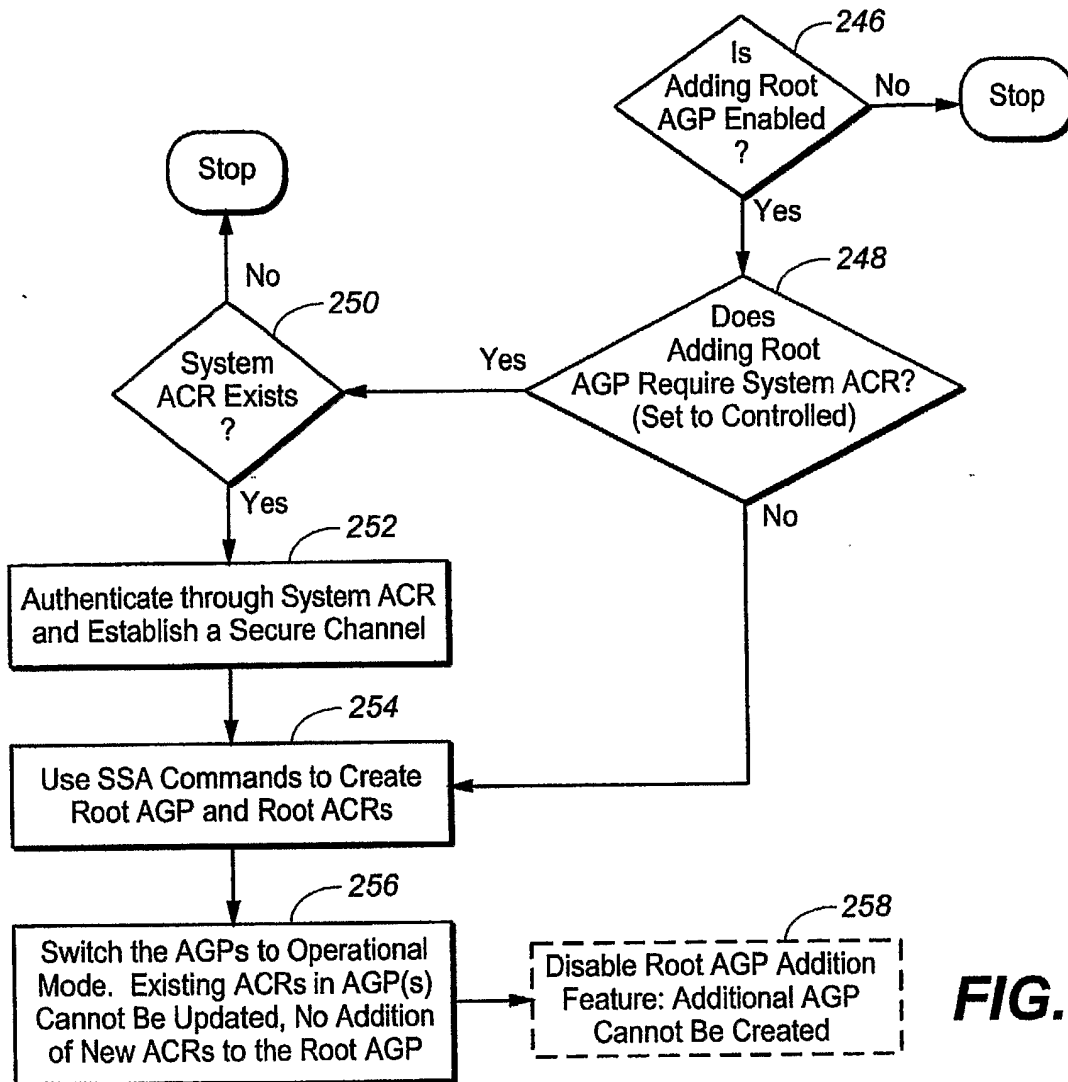
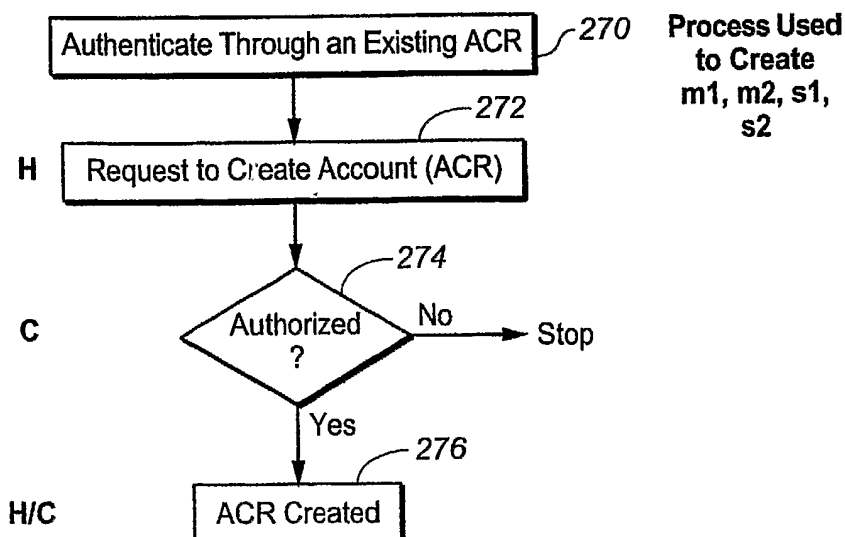
6 / 14

**FIG. 7**

7 / 14

**FIG. 8A****FIG. 8B**

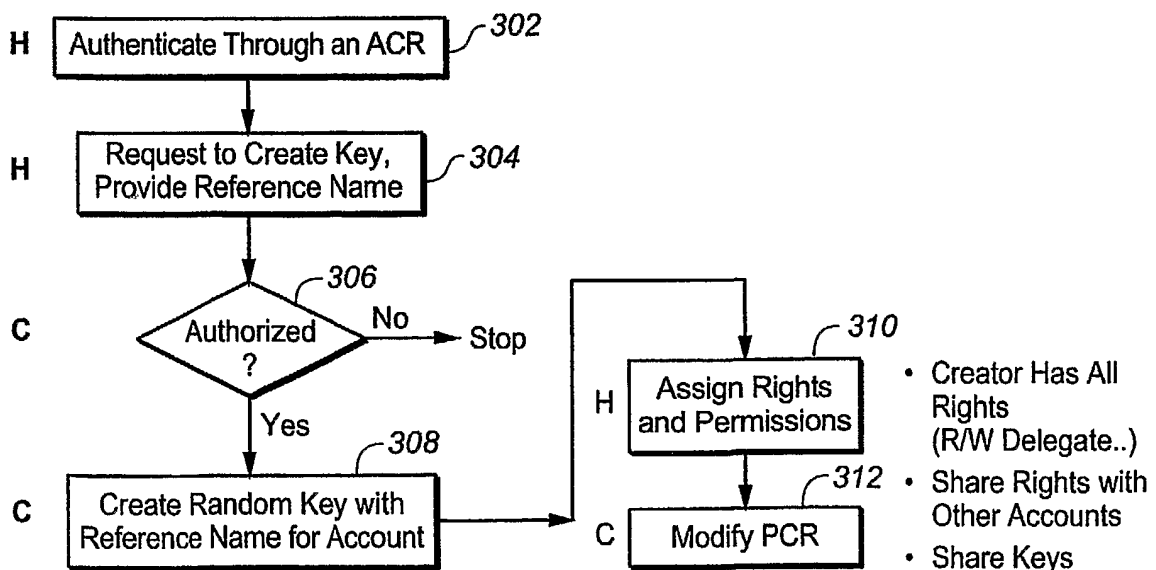
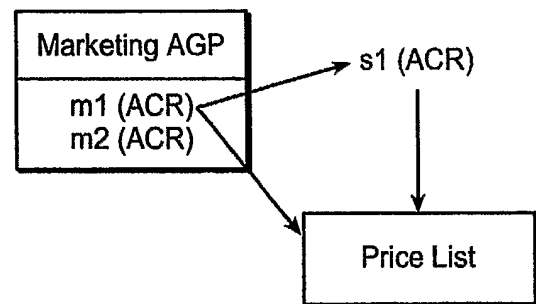
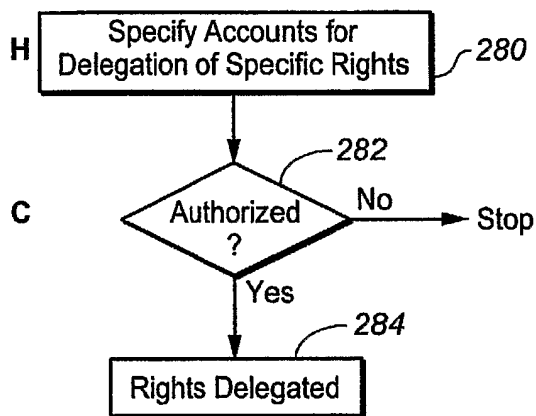
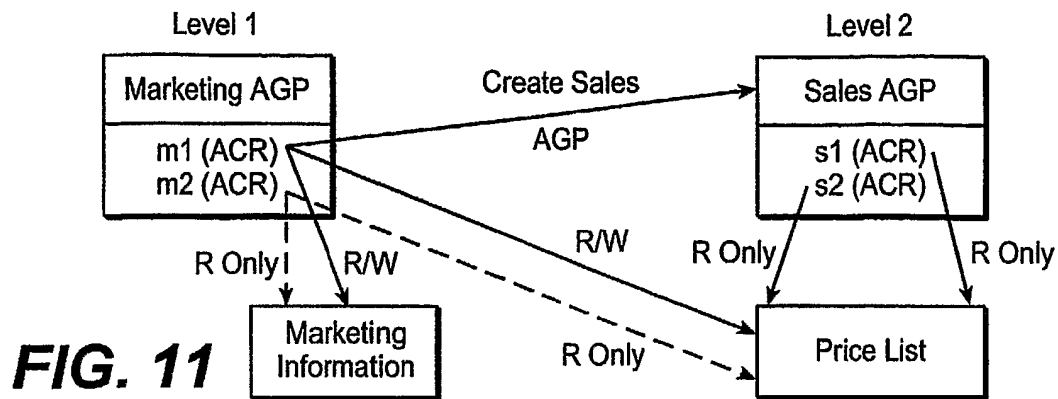
8 / 14

**FIG. 9****FIG. 10**



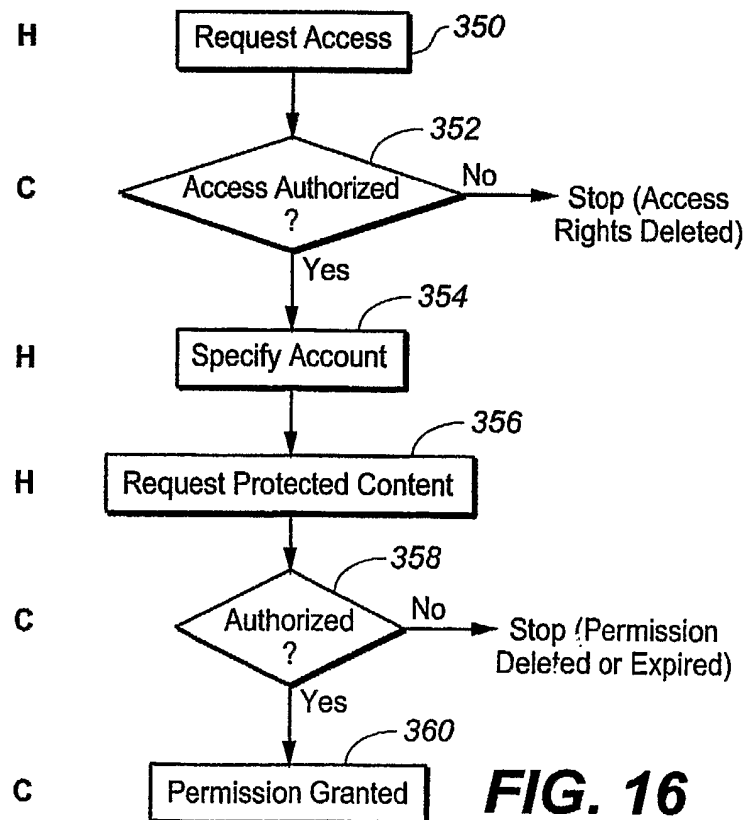
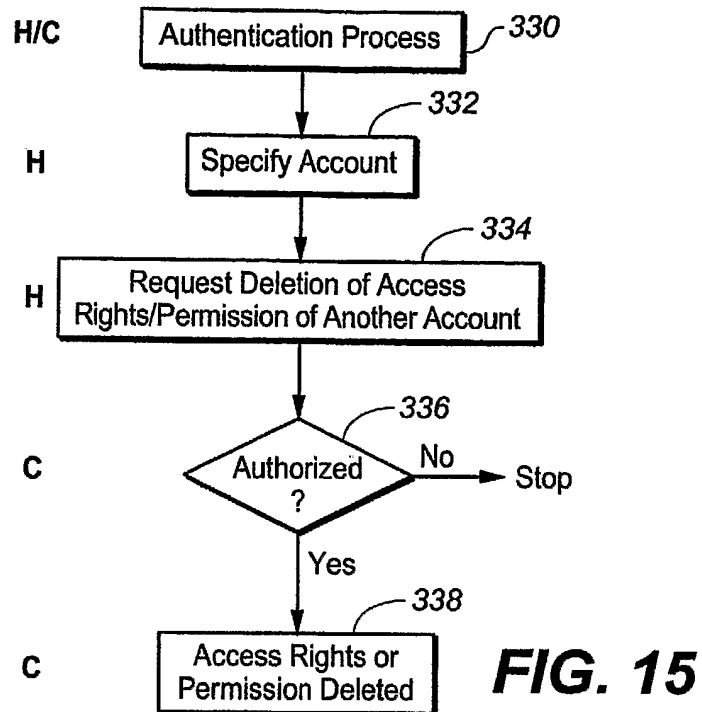
9 / 14

Create 2 ACRs (m1, m2) in Marketing AGP, 2ACRs (s1, s2) in Sales AGP



**FIG. 14**  
SUBSTITUTE SHEET (RULE 26)

10 / 14



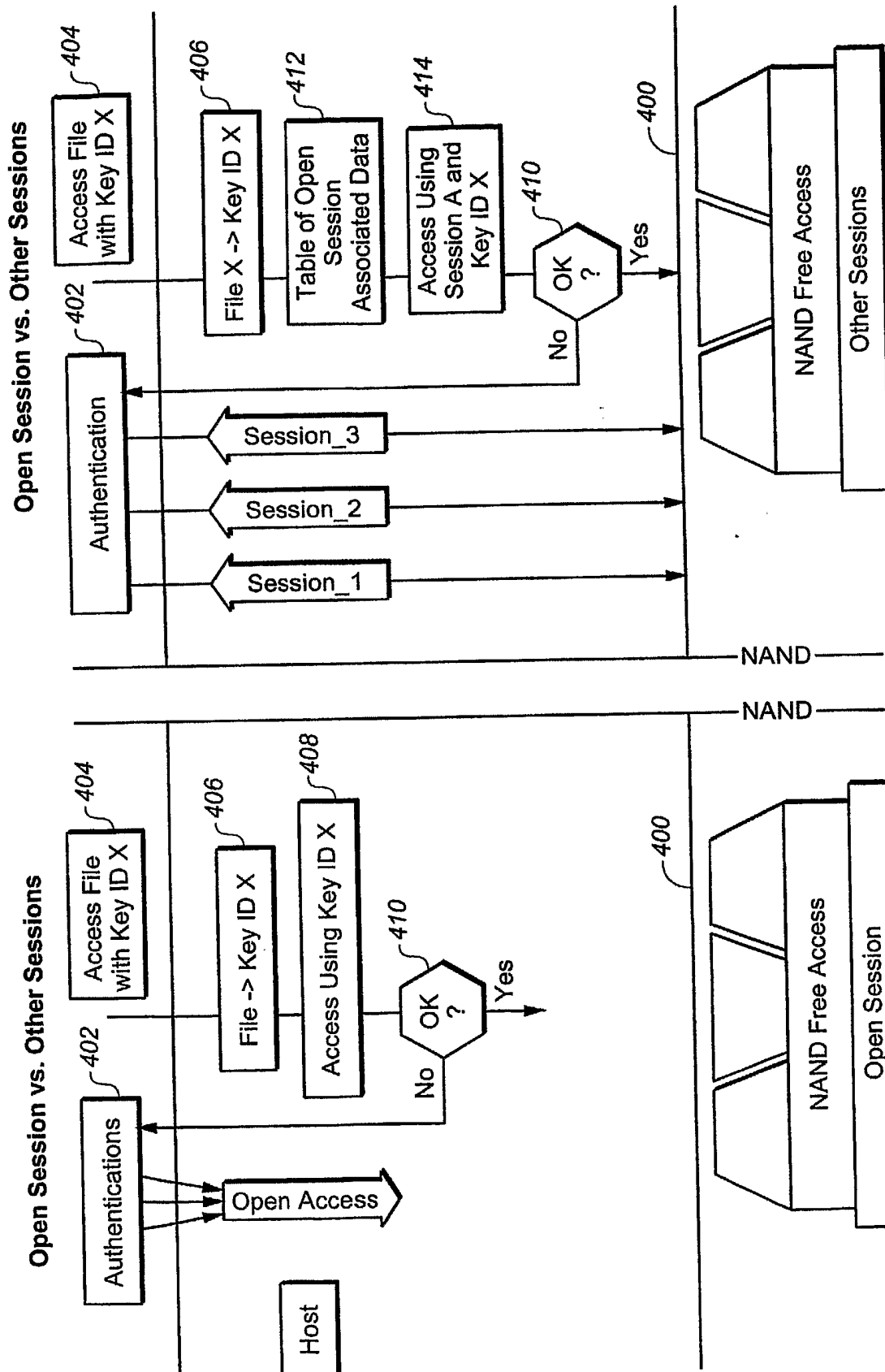


FIG. 17B

FIG. 17A

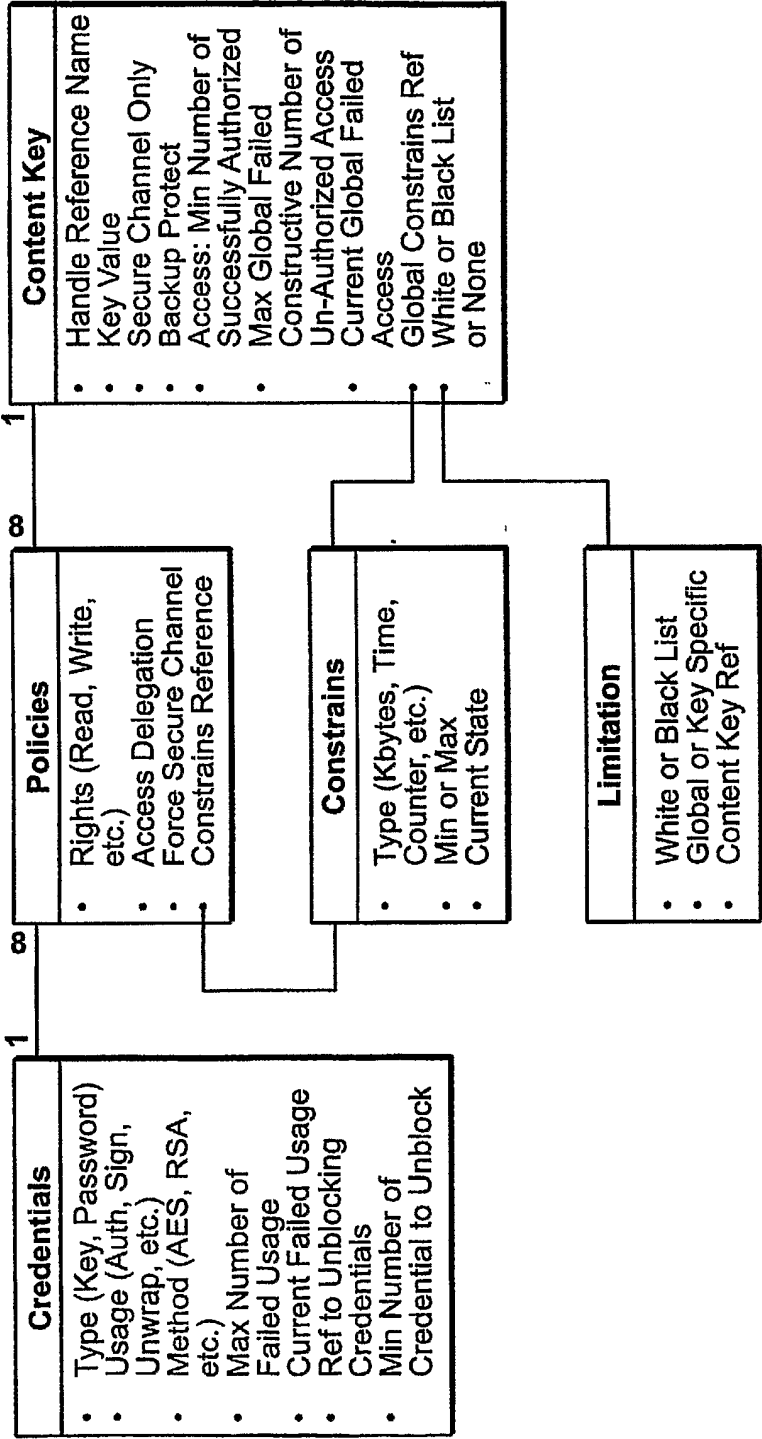
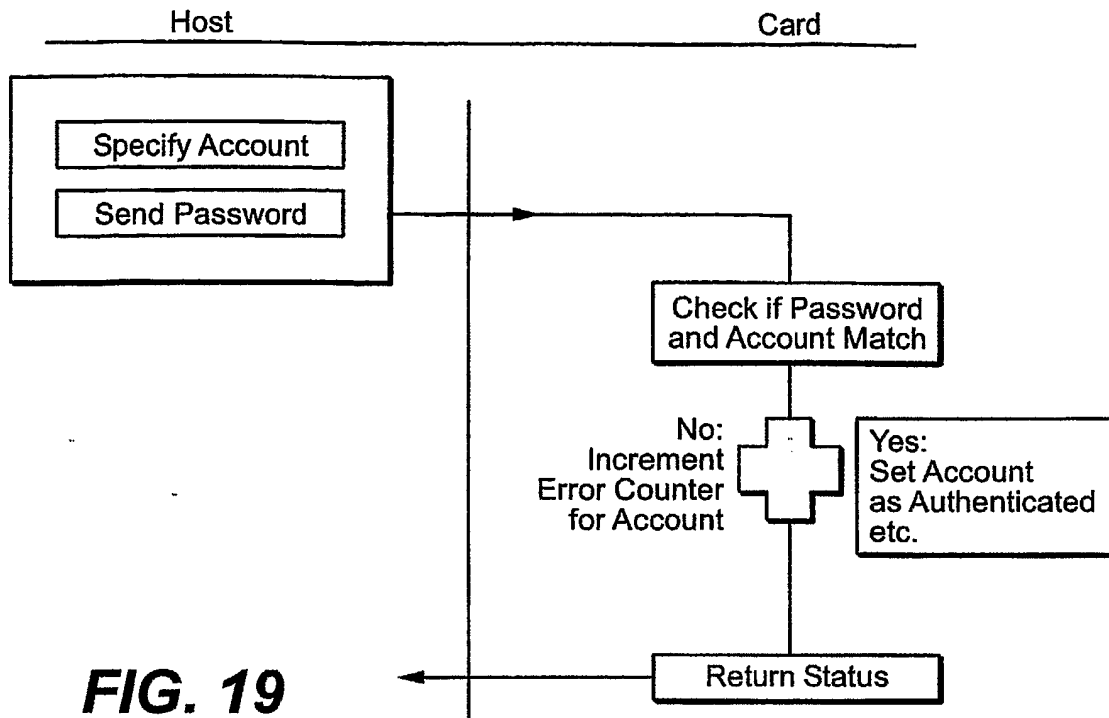
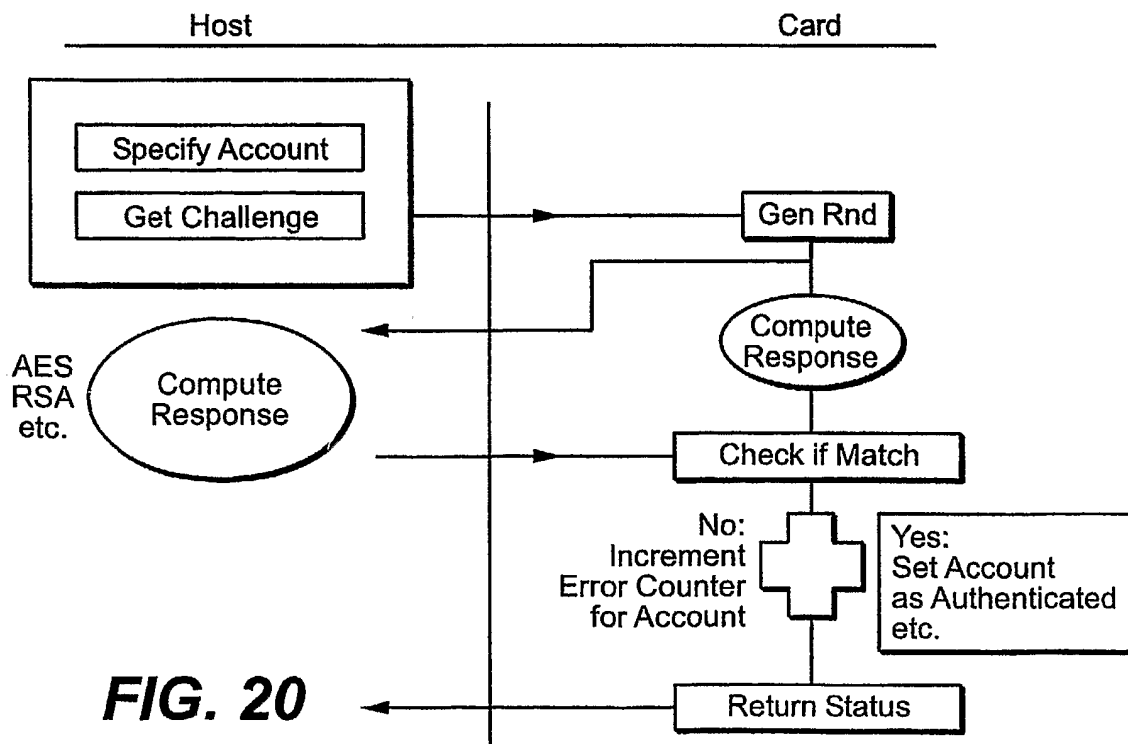
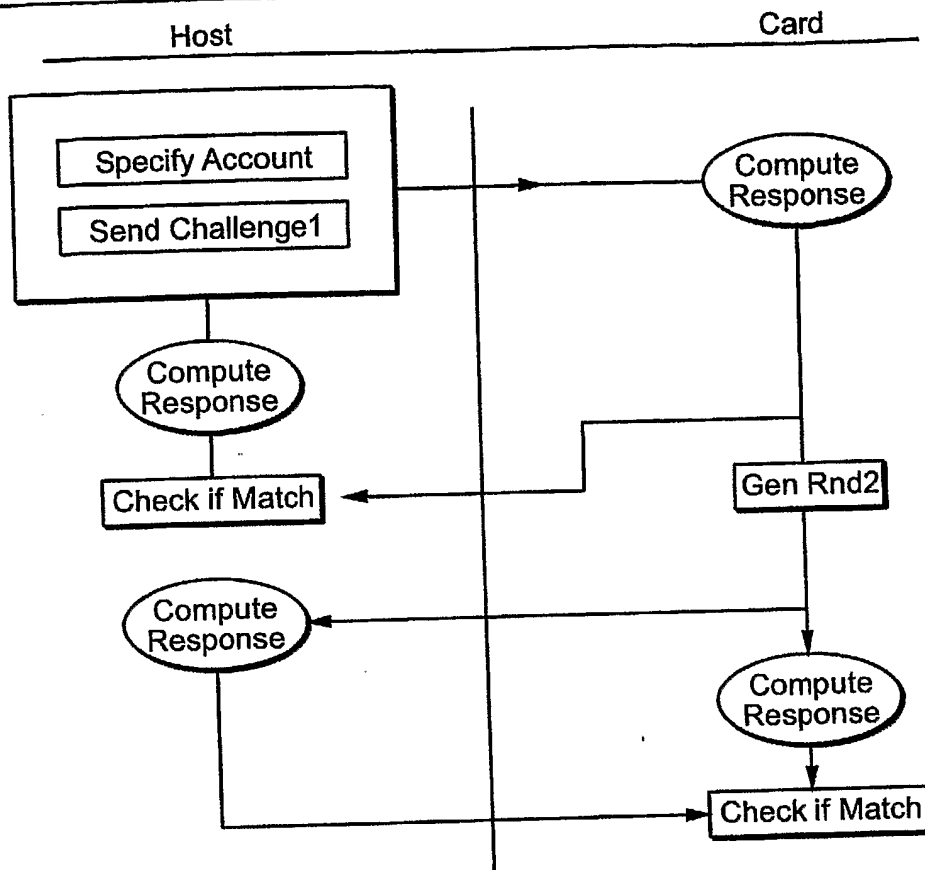
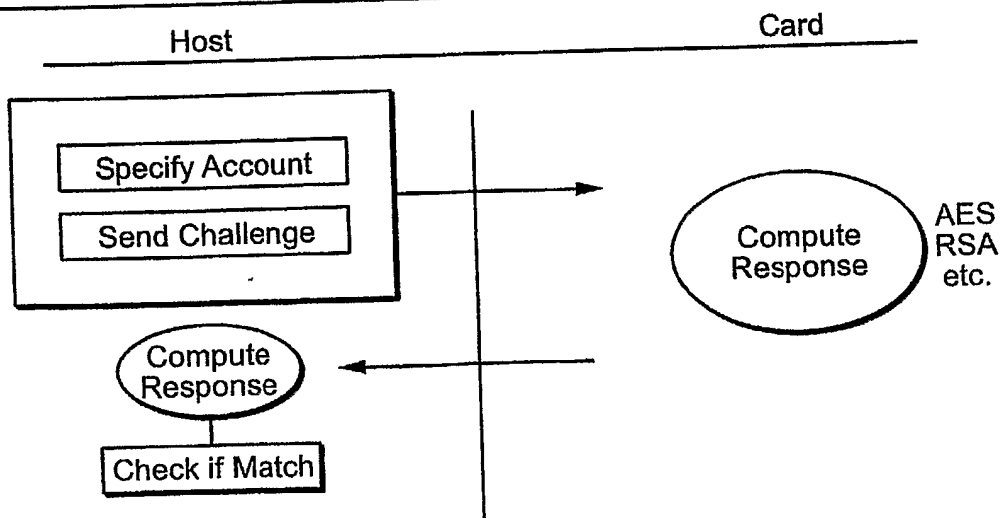


FIG. 18

13 / 14

**Login/Password Type****FIG. 19****Challenge/Response Type  
Host Authentication****FIG. 20**

14 / 14

**Challenge/Response Type  
Mutual Authentication****FIG. 21****Challenge/Response Type  
Card Authentication****FIG. 22**