The architecture of an access control mechanism for the resources of an OS/360 or VS/370 computer system is presented.

The use of this operating system component for data base security and integrity in a research and engineering environment is described.

The techniques described make possible controlled access to the system's processing power, controlled access to the database, decentralized authorization responsibility, measuring dataset usage, and event recording for automatic dataset migration, archiving, and staging.

An access control mechanism for computing resources

by H. M. Gladney, E. L. Worley, and J. J. Myers

Early in 1970, a research project was started at the IBM Thomas J. Watson Research Center to implement a prototype system to improve dataset security and resource allocation in the System/360 Model 91 that was operating there. Late in 1970, the IBM San Jose Research Laboratory began to consider problems of managing a rapidly growing database in their System/360 Model 91. TSO service was being planned and there was already experience with TSS at the Research Center and other installations to indicate that interactive terminal service would create a user database that could require more extensive administrative services and program tools than we then had available. It was clear in both the Research Center and the San Jose Laboratory that the necessary function would eventually have to be delivered without a substantial increase in computer operations manpower. Since a Research Center prototype access control system met many of the San Jose Laboratory requirements, the prototype was transferred to San Jose late in 1970, where the major development activity took place. Certain partly implemented proposals in the TSS/360 and CTSS systems[1] were also attractive to workers at San Jose and were incorporated into the access control system that is described in this paper.

In 1970, the San Jose Research Laboratory was primarily concerned with encouraging users to exercise restraint in creating data and with finding a mechanism for managing the allocation of computer time on a more decentralized basis. By 1972, improved database security and integrity had also become high-priority objectives at both the Research Center and the San Jose Laboratory. We believe we have developed a broadly applicable tool for the decentralized control of computer facility resources. This tool is called the Inventory Control System or the Installation Management Facility.

In this paper, we give our general objectives. Specific design objectives that evolved with the project and which are still evolving are also presented. We sketch the architecture primarily by describing what the user sees, and indicate some unexploited opportunities. Finally, we comment on experience to date. We assume that the reader is generally familiar with the IBM Operating System for Multiprogramming with a Variable Number of Tasks (OS/360 MVT).

General objectives and their implications

research environment

The desirability—even necessity—of improving database control, database security and integrity, and of improving computer access and allocation control over that which has been practiced has been widely discussed[2-7]. Described here are the requirements of the San Jose Laboratory computer installation. We believe that the considerations that led us to implement the inventory control system are broadly applicable. The environment at San Jose is that of a research and engineering laboratory, in which many computer users are also programmers, who often establish and maintain their own databases without the computer installation personnel understanding much of the contents or use of the data. Although the problems addressed typically pertain to the large installations today, as technology and applications evolve, they may become the norm of a much broader set of installations.

database trends

Before 1980, one can anticipate databases at least an order of magnitude larger than those at present. Between 1968 and 1974 the San Jose System/360 Model 195 database grew from under one thousand datasets to over ten thousand datasets. Growth in the direct-access capacity from eight IBM 2314 spindles (250 megabytes) to sixteen IBM 3330 Model II spindles and twenty-four 2314 spindles (4000 megabytes) convinced San Jose management that improved methods of controlling and protecting the database would be necessary. The alternatives considered for controlling the size and integrity of the database were: (1) rigid procedures administered by the operations staff, or (2)

software additions to the operating system to minimize and decentralize administrative procedures. A catalog function designed for a database of 10^5 datasets was desirable, even though its characteristics were not optimally economic for 10^3 datasets. Reclamation of fragmented space on disks by 1970 methods was anticipated to be uneconomical. We wanted to explore the feasibility of utility programs to reorder the database with a minimum of personal intervention, without taking the system offline, and without creating database integrity problems. We wanted to explore a proposed fully automatic mechanism for migrating little-used data out of on-line volumes and staging it back as required at a minimum cost to the installation and to the user. None of these objectives could be met without the maintenance within the system of more detail about the status each dataset than that contained in the OS catalog and the Volume Table of Contents (VTOC).

user trends

Growth and diversity of the San Jose Laboratory user population paralleled that of the database. In 1968, there were fifty users, three operators, and two system programmers. By 1970 their numbers had doubled. And by 1973, there were about eight hundred users, eleven operators, and thirteen system programmers. As time went on less was known about the individual users, not only because they were more numerous, but also because many were remotely located. At one time, there were fifteen Remote Job Entry (RJE) stations (with up to eight active stations online simultaneously), thirty-five ports of TSO activity, three graphic display consoles, and five System/7 satellite processors. The operations staff began to experience increasing difficulty in administering access to the system, and the staff could not be enlarged. Security and management controls and procedures also had to be correctly administered. These and other considerations could be addressed by the maintenance of information about each user in the system together with his dataset information.

authorization integrity, and security

There is an ever more rapid evolution toward an environment in which users would like to treat the computer facility as a trustworthy repository of their data and in which several users may share data. This situation is interesting because it demands that the computer facility take direct responsibility to see that data are used properly. Alternatively, the installation could provide a mechanism to encourage users to establish clear lines of authority and responsibility, and insist that responsibility for using an authorization mechanism rest with the users and the appropriate managers. Some policy intermediate between these extremes will probably be chosen by most installations, although our own preference would tend to emphasize the latter alternative. We have based the San Jose authorization procedures on the individual and managerial alternative. In either case, the comput-

ing center must make it possible for users and managers to ensure dataset integrity and dataset security appropriate to the nature and value of the data stored. Here, *security* means protection against deliberate misuse, and *integrity* is protection against erroneous modification. To meet this objective, information must be stored about the relationships among system users.

Besides authorization, integrity, and security there were some secondary objectives. In the years since the definition of OS and the Job Control Language (JCL), the technology of data management has developed to the point that the user need not—in principle—concern himself with many of the characteristics of the data such as space, location, and device type, unless he has special requirements. However, the syntax and rules of JCL usage have remained relatively invarient to protect the very large user investment in procedures that use JCL. TSO had removed much of the detail of JCL for the terminal user, and we wanted to investigate the opportunity of reducing burden of JCL even further. In addition, we had the opportunity to collect otherwise unavailable statistics about dataset usage.

inventory control system

The central point of this paper is that all these objectives can be met by making one addition to the data maintained by the system about itself, and by adding a set of utility programs that would modify and/or act on the contents of such an *inventory dataset*. The resulting inventory control system may be thought of as a mechanism for controlling and recording access to computing system resources. CPU time is conventionally regarded as a resource. Permission to use a dataset, a terminal, or a program also should be thought of as permission to use a resource.

The inventory control system is also a mechanism that permits the decentralization of authorization by providing a framework with clear lines of authority and responsibility. It should be possible to shift the burden of access administration from the computer center installation management to those parts of the organization that are better qualified to deal with the details and to make specific management decisions. Such decentralization reduces potential security-breeching temptations on the part of computer facility employees in the following four ways:

- Data available to installation personnel may be restricted.
- Only the owner needs to know which of his datasets are sensitive.
- Great effort is required to make an unauthorized request.
- An audit record is created when subversion is attempted.
- Access to data catalogs can be limited.

Architecturally, the inventory control system adds several concepts to OS/360 that users must understand. An *organization* is

a collection of *groups* with specific interconnections. A *group* is an entity with which users and/or other groups may be associated. A *user* is one who requests access to resources. The *owner* of a resource is a user or group who has the right to define the access privileges of other users to that resource. The user who creates a resource, such as a new user identification, is refered to as the *author* of that resource.

Implementation objectives

From the previously discussed general problems, requirements, and definitions, there have evolved specific implementation objectives that include the following features.

- A centralized inventory of the resources of the system and the privileges of each user.
- A dataset security and integrity mechanism for the entire system database, with as strict or loose security for each dataset as is desired by the owner.
- An audit trail for all attempts to use an unauthorized resource, or—at the option of the owner of a dataset—an audit trail of all access attempts.
- Support for four levels of dataset storage—on-line, migrated, archived, and backup—with date (and possibly time) stamps for the last migration, archiving, backup, open-for-read, and open-for-modification.
- Independence of device types, except that removable volumes may be treated differently from on-line volumes.
- Acceptable search speed. It is assumed that the extra function may be worth some performance degradation, but we also assume the possibility—for very large databases—of a design for which the search time increases only as the logarithm of the number of inventory entries.
- Minimal system modification. This has three objectives: to minimize the cost of implementation, to maximize reliability, and to approach release independence.
- Clearly defined system interfaces.
- No new required operator intervention. For a large database of which the larger portion has some form of access control, one can assume that the protected datasets will be opened at the rate of several per minute. Thus changes to access control information will occur frequently, so that manual maintenance of and reference to an access list seems to be precluded. In particular, no system security officer is to be required.
- Reduction of operator or administrative interventions from that which is currently required, wherever possible.
- No change to the user environment, with the exception of system commands that provide new function.

- Coexistence with the system catalog. This is necessary to permit conversion and to supply catalog functions not supported by the inventory.
- The structure of inventory record types is to be an installation option.
- The system commands that provide the access control functions should be simple enough that the user is able to operate with a minimum of help from consultants.
- If a user were for some reason not known to the inventory system (e.g., if an installation were to decide to support some users solely with the catalog), it should not be possible for him to access any inventory dataset—except for reading—provided that a dataset has READ access for all users.

The inventory control system can be only a portion of the software component of a security system. Comprehensive treatment of the subject of protection against subversion of the operating system is a separate topic that is not addressed here apart from the help that is implicit in limiting access to key system datasets.[4-10] Although security (and other) requirements are likely to stimulate fundamental changes in computer system design, we believe that many of the inventory control system concepts will continue to have value. The degree of protection supplied by the inventory control system is limited to that provided by other parts of the operating system. It does not plug any holes in operating system integrity. The inventory control system does provide increased system function with little impact on system performance. Another topic not addressed here is penetration protection against the determined, resourceful user in the sense of military penetration protection. Our primary target is the casual misuse that is currently possible in commercial installations.

Architecture and implementation

Possible architectures have proved to be readily definable using the implementation objectives just given. Appendix 1 illustrates the relationship of groups to users and to datasets. Appendices 2 and 3 use inventory commands to display portions of a tree of groups and default connect entries. Each user has a default group to which he is connected (i.e., from which he may withdraw resources) unless he specifies connection to another group during his LOGON or on his //JOB card. A Connect Entry specifies authorization to a specific user to connect to a specific group and to use specific resources. The first-level qualifier of any dataset name identifies the author, i.e., the original owner. Each group has resource limits that may be allocated to dependent groups or to Connect Entries.

Table 1 GROUP entry

Superior group
Authorization date
Name of authorizer
Subgroup count plus names
Connect entry count plus names
Universal access flags
Name and privilege of special authority users
Statistics and limits for • Jobs • CPU time • EXCPs • Direct access space • Output lines, cards • Charges

Table 2 USER entry

Date of authorization
Authorizer
Rights and description of user
Password
Date password changed
Programmer name
Default group
Pointers to connect entries
Last job name, date, time

Table 3 CONNECT entry

Authorization Date
Name of Authorizer
Statistics and limits for • Jobs • CPU time • EXCPs • Direct access space • Output lines, cards • Charges
Last job name, date, time

The Connect Entry of a user to a group specifies the authority a user has relative to that group. A user with RUN authority may use the system under control of the group, but may not create datasets. USE authority permits only the use of computer power and the creation of datasets whose first qualifier is the user name. CREATE authority not only implies USE, but also permits the user to create datasets whose owner is the group. CONTROL authority not only implies CREATE, but also permits the user to authorize users already known to the inventory to connect to the group and to specify their privileges. JOIN implies CONTROL and also permits a user to define new users to the system as well as new subgroups that depend on the group.[11] Appendix 4 shows an example of the privileges permitted to a user.

The group structure defined by the JOIN function is a tree structure, with authorization control propagating down the tree. Any user may propagate to a subordinate user up to the level of authorization that he himself has. The tree structure permits as much centralization as an installation requires. Judicious assignment of the JOIN and CONTROL authorization permits some measure of "local centralization," which might be attractive in some evironments.

Access to a dataset may be specified for groups of other users, and for all users not explicitly named. The supported codes READ, APPEND, WRITE, R/W, and NONE are self-explanatory, except that WRITE and R/W do not include permission to scratch. The code ALL allows all privileges that the system without inventory control supports, and additionally permits a named user to change the access authorization list for the dataset. It is possible for a user to read the access authorization list if, and only if, he has ALL authority relative to the dataset itself. This limitation hinders an unauthorized search for data. If a user is named in a dataset entry, the specified access is enforced; if he is not named—but his current connect group is named—the group access is enforced. Otherwise, the Universal access is enforced. Appendix 5 illustrates this with privilege lists for a user's dataset, GLADNEY.C. CLIST, and for a key system dataset, SYS1.LINKLIB. The dataset entry is related to the group or user entry by name implication only. The first-level qualifier of the dataset name must be a valid group or user name. The inventory control system provides no access control at the record or field level. In online personnel files, for example, it is often desirable to permit user read-only access to part of a record, write-only access to another part, read-write for a third part, and permit no access to the remainder of a record. Such requirements are common in database applications.

The inventory itself is the dataset, similar to the catalog, except that it does not use index levels. Since the group structure is a

tree, and since each dataset is associated with a group—either directly or through a user—the number of accesses to find a dataset is proportional to the logarithm of the number of inventory entries. The first two levels of the tree are typically very small and could be kept in main storage. The next levels could be paged so that the number of disk accesses for a search is much less than the number of levels.

The current inventory control system at the San Jose Laboratory has the following four types of inventory entries: GROUP, USER, CONNECT, and DATASET. Tables 1 through 4 list the more important contents of each entry type. Another installation may choose different record formats within broad limits, without reprogramming any service program. Some fields are obligatory.

It is possible to remove all access privileges from a user without removing the user from the inventory. This makes possible the prompt removal of the privileges of a terminated employee.

Audit records for attempted unauthorized accesses are written as installation-defined records into the system accounting file, which, at the San Jose Research Laboratory, is the OS System Management Facilities (SMF) output dataset. Clearly, another dataset could be specified, but the SMF output has much to recommend it. At the option of the owner of a dataset, all accesses to that dataset may be recorded in the audit file. This permits improved control over sensitive data. In a computing center, mechanisms must be installed to prevent tampering with an installation accounting file derived from the SMF file.[11] These mechanisms also protect the audit trail data. Procedures for extracting audit data and generating reports may be combined with the accounting processing with only minimal additional effort, machine time, and administrative handling, particularly since audit and accounting reports are often addressed to the same individuals.

Since the inventory is itself a dataset, it may be used to protect itself against unauthorized access.

Tables 5 and 6 summarize the TSO commands available to the user, and are self-explanatory. Appendices 2–6 illustrate the information provided by several of the listing commands. A subset of the TSO commands is available to batch processing users.

Table 4 DATASET entry

- Date
  - Last used
  - Last changed
  - Last dumped to disk
  - Last dumped to tape
  - Last dumped by user
- Open Count
  - For input
  - For output
- Open interval statistics
- Flags for migration
- Security status
  - None
  - Internal use
  - Confidential
  - Registered confidential
- Universal access rights
- Users (groups) with privileges
- Access count for each other user
- Number of directory blocks
- Size
- Migration date
- Volume currently used
  - Count
  - Type(s)
  - Serial(s)
  - File sequence

Potential function

allocation control

We now consider briefly some inventory control system functions that are available in principle, but which have not as yet been included. It is ironic that, because the needs of our own comput-

Table 5 Privileged user commands

ADDGROUP	Adds a subordinate group and specifies its resource limits.
ADDUSER	Adds a user to one or more groups and specifies his privileges.
ALTGROUP	Alters the access list and account number of a group.
ALTUSER	Alters the resources permitted a user.
CONNECT	Authorizes a user to connect to a group and specifies his privileges.
DELGROUP	Removes a group entry; effective only if no users, other groups, or datasets are subordinate.
DELUSER	Removes all references to a user from the inventory; effective only if there are no datasets still owned by that user.
LISTGRP	Lists resources available to a group, superior and subordinate groups, and users authorized to connect.
LISTREE	Lists the hierarchical structure dependent upon the specified group.
LISTUSER	Lists the resources available to the specified user.
RELEASE	Removes the connect entry of the specified user from the specified group.

ing installation have changed, we have not yet implemented either direct-access space control or processing time allocation control functions, which were the original stimuli of the inventory control system project. Allocation controls evoke interesting and complex policy problems, many of which have received a lot of attention.[11] There are also questions as to what action the system should take when a user has (or is about to) exceed his allocation limit. In the case of CPU time, one could – for example – prohibit further use, or, less drastically, reduce the priority of the job. With the advent of automatic dataset migration and staging mechaisms, if direct-access space limitations are about to be exceeded, a clear policy is available – to migrate off-line selected datasets owned by the same user. We intend to explore such topics by using the data-collection features of the inventory and experiencing for ourselves the consequences of one or more policies.

dataset usuage

We have not yet had the time to make the detailed investigation of dataset usage for which we now have the data. We have found that looking at the raw counts of accesses has often been helpful. For example, we determined which Fortran compilers were most used prior to discontinuing the installation support for some of them. However, we have no insight into how carefully users with sensitive data limit access to it, and how many unauthorized accesses are attempted.

It would be possible to use the inventory as the input dataset to an installation accounting program that requires information about affiliation of users if each upward path from any point in the organizational tree has an account number in at least one group node. We suggest that this be done by supplying the account number to the OS-SMF at the time of each transaction.

Table 6 Database user commands

BACKUP	Creates a duplicate copy of a dataset on an off-line volume. For checkpoint purposes, the new dataset has the same access list as the referenced dataset. If a prior backup version exists, it is destroyed.
RESTORE	Creates an on-line dataset from the backup version. The backup version remains intact.
MIGRATE	Moves a dataset from an on-line volume to an off-line volume, to save the cost of on-line space if data will not be used in the near future.
STAGE	Opposite of MIGRATE.
SCRBACK	Scratches the backup version of a dataset.
SCRMIG	Scratches a dataset that has been migrated by the user or archived by the installation.
LISTINV	Lists dataset names that belong to user or group, with an option to include volume identification.
DISPOSE	Presents the name of each off-line dataset owned by a user, and gives him the option of saving, deleting, or staging that dataset.
LISTDSE	Lists location, space, access list, creation date, last use date, last change date, and use counts.
PERMIT	Modifies the access list of a dataset, according to arguments provided; optionally resets the audit trail flag on or off.

This procedure would be easier to use and be a better subject to audit than that described in Reference 11. No new system dataset would be required because the inventory can store information about disconnected users and expired accounts.

We have considered what would be required to make access control support available to another terminal support system, e.g. APL/370 with the shared-variable (SV) and time-shared input-output (TSIO) support.[12] We found that APL-SV-TSIO has a similar definition of several access authorizations, which include controlling the use of system resources, assuring the integrity of the system, and serving the interests of privacy. Each APL user may dynamically offer and request access to resources under the control of an OS system task, of which TSIO is one possibility. TSIO specifies which users it recognizes and whether it permits SPACE, DEVICE, ACCESS, or SYSTEM authority as follows:

- SPACE permits dataset creation and direct access space allocation only.
- DEVICE extends the SPACE privileges by permitting the user to select the location of his datasets.
- ACCESS extends SPACE and DEVICE privileges by permitting the user to read other TSIO users' datasets.
- SYSTEM gives complete access to any dataset, subject only to the installation's other security provisions.

In APL-SV, there is provision to limit the number of shared variables that are simultaneously defined by each user. Clearly, the

repository of these authorizations could be the inventory. We have also studied TSIO closely enough to believe that APL analogues of the TSO commands in Tables 1–4 can be provided with simple changes to our TSO implementation, and that the format of the APL vector that defines each command could be identical to the TSO command format. We are led to speculate that similar provisions could be made in other interactive support programs.

Access control at the record or field level is not provided by the inventory control system because the smallest entity explicitly known by the inventory is a dataset. Field- or record-level access control must be provided by application programs. Some aid, however, could be included in the inventory to support record or field-level control. Such aid might be given as additions to the inventory dataset entry (Table 4) to identify program names and/or program-supplied passwords, which may optionally be associated with users. As an example of such a provision, user A might be permitted to access a dataset with programs B and C, but user D could access it only with program B. A further extension that might be useful would permit a user to have a different level of access to a dataset, depending on which program he is using. Such a capability requires the provision of a mechanism to prevent one program from masquerading as another.

Experience

The inventory control system was installed at the San Jose Research Laboratory on the IBM System/360 Model 195 in November of 1972, and the user conversion to inventory support was then completed by May of 1973. One-hundred seventy TSO users were being supported by the end of 1974, and about seven thousand datasets were being protected. With the inventory on an IBM 3330, each inventory access takes about seventy milliseconds. The access time is added in the DASD OPEN process. Further delays are possible in the scheduler allocation-deallocation process. The inventory control system was installed in March 1974 on an IBM System/370 Model 165 running under OS MVT and on a System/370 Model 158 running under OS/VS2 in June of that year. The protected database on the System/370 Model 165 is growing rapidly and exceeded that on the System/360 Model 195 before the end of 1974.

The reliability of the inventory control system has been satisfactory. We are unaware of any dataset integrity problem that has been caused by the inventory control system itself. We further believe that the overall incidence of lost or destroyed data is much improved, although we cannot measure this independently, since other systems and operations improvements were intro-

duced during the same period. The incidence of errors in new programs has been very low; most of these errors have been caused by discrepancies between documentation and implementation. We are aware of only a very small number of service interruptions caused by the inventory control system.

Although there is a large number of TSO commands available, most users exploit only the following three commands: LISTINV, LISTDSE, and PERMIT. The decentralized addition and removal of users is working well with a noticable reduction of the administrative burden in the computer center. That a new user can be joined to the system without administrative delays is much appreciated by the users.

The installation systems programming staff was the group by far most affected by the existence of the inventory. Rather common, but questionable, techniques—such as duplicate dataset names on several volumes, ability to access user datasets, and volume restore facilities—are usually prevented. Normal software system maintenance is much improved because we have not only been able to assign responsibility and control for system components to particular individuals, but we have also been able to enforce the assignment. Systems programmers find it very useful that the date of latest change is maintained by the system.

Since we built and tested the inventory control system in an operating environment, unusual care was required to prevent adverse impact on system users. Integrity of the users' data was an absolute requirement. To provide integrity while building the inventory control system we have insisted that utility programs for database maintenance be available early in the development and installation periods. For this reason, the research laboratory computer facility staff developed a package of utility programs for database integrity checking, backup and recovery, and space reclamation. These programs were to be available before they were actually needed at the risk of occasionally delaying installation of the inventory control system itself. As far as we know, no user dataset was damaged as part of the conversion process.

Concluding remarks

We have described the objectives and architecture of an experimental system component, the inventory control system, which provides controlled access to system resources and data with decentralized authorization responsibility. In addition, the inventory control system is a prerequisite for an automatic dataset migration and staging facility. The control system provides a mechanism for the collecting of dataset usage statistics not other-

wise provided in OS MVT or OS/VS2 and is a supplement of the catalog function. The system has been implemented and meets the implementation objectives that have been described in this paper. The system has proved to be reliable in every measureable respect.

Security is not an absolute attribute; our objective was to obtain a significant improvement in protection within the costs acceptable to a commercial data processing installation. The described mechanism provides a considerable obstacle against a marginal informed violator. Protection against the informed and determined opponent will require deep-reaching changes in systems design as well.[2-7, 10] We believe, however, that many of the concepts tested with the inventory control system will be necessary in future "secure" systems.

ACKNOWLEDGMENTS

For many early discussions, we thank J. Jaffe, and for early experimentation with the IBM Research Center prototype, we thank W. H. Kwok. B. S. Brawn contributed greatly through discussions and early project management. K. G. Field contributed discussions of system internals and the PASSWORD interface. R. Y. Shimizu implemented many of the database commands. For his support, we thank A. H. Eschenfelder. We also wish to thank C. K. De Long and L. Ferguson.

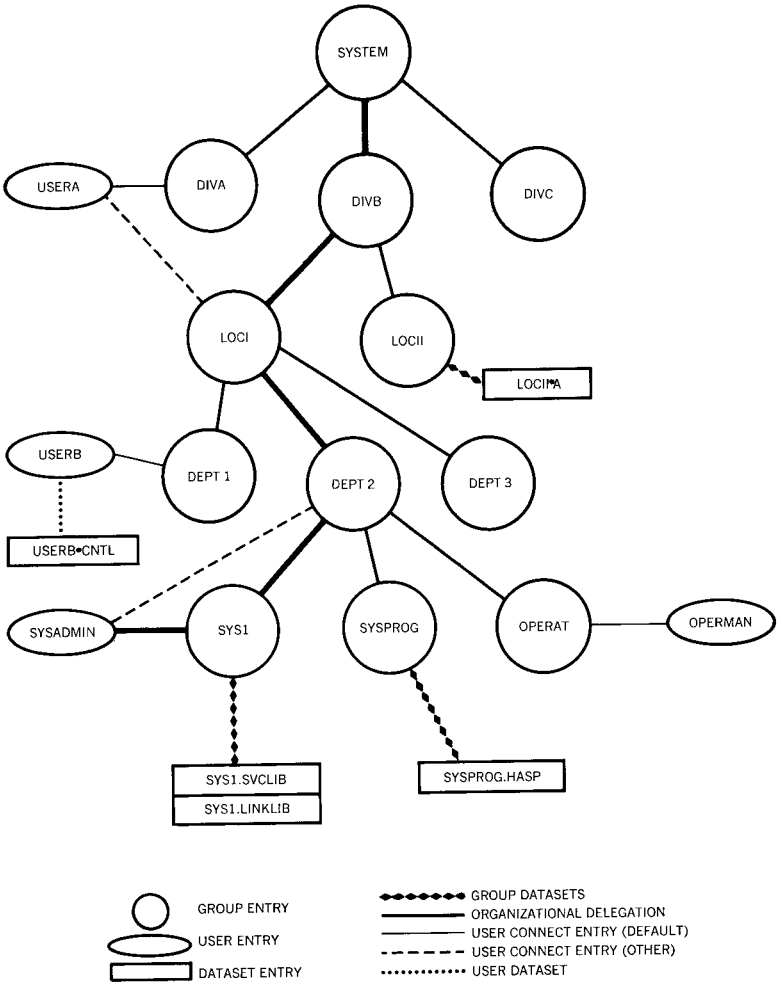
CITED REFERENCES

1. For Project MAC background, see L. L. Selwyn, *BUYTIM: A System for CTSS Resource Control,* Project MAC-M-379R (July 1969); D. H. Vanderbilt, *Controlled Information Sharing in a Computer Utility,* Project MAC TR-67, (October 1969); and R. C. Owens, *Primary Access Control in Large-Scale Time-Shared Decision Systems,* Project MAC TR-89 (July 1971).
2. P. S. Browne and D. D. Steinauer, "A model for access control," *Proceedings of the ACM SIGFIDET WORKSHOP,* 241 (1971).
3. *Data Security and Data Processing,* IBM Systems Reference Library, Form G320-1370 to G320-1376 inclusive (1974); may be obtained from the IBM Corporation, Data Processing Division, White Plains, New York 10604.
4. A. Harrison, *Computer Information Security and Protection. A Bibliography with Abstracts,* National Technical Information Service Reports (October 1973).
5. J. G. Bergart, M. Denicoff, and O. K. Hsiao, *Bibliography on Computer Security and Access Control,* Ohio State University Report CI SRC-TR-72-12 (November 1972).
6. W. L. Lupton, *Study of Computer-Based Data Security Techniques,* Master's Thesis, U.S. Naval Postgraduate School, Monterey, California. (June 1973).
7. Hughes Aircraft Ground Ground Systems Group, *SECURITY OF THE TACC Data Base Study,* Report ESD-TR-71-370 (October 1971).
8. G. J. Popek and C. S. Kline, "Verifiable secure operating system," *AFIPS Conference Proceedings* **43,** 145 (1974).

9. S. B. Lipner, W. A. W. Self, R. B. Shell, G. J. Popek, P. G. Newman, C. Weissman, and T. A. Linden, "A panel session—security kernels," AFIPS *Conference Proceedings* **43,** 973 (1974).
10. R. F. Mathes, S. R. Cricker, M. Denicoff, V. Mitchell, F. W. Weengarten, E. Feustel, L. J. Hoffman, D. Hsiao, R. Turn, and W. A. Wolf, "A panel discussion—research in data security-policies and projects," *AFIPS Conference Proceedings* **43,** 993 (1974).
11. H. M. Gladney, D. l. Johnson, and R. L. Stone, *On Installation Accounting,* IBM Research Report RJ1443, may be obtained from the IBM Research Laboratory, Monterey and Cottle Roads, San Jose, California 95193.
12. IBM Systems Reference Library, Form #SH20-1463, may be obtained from the IBM Corporation, Data Processing Division, White Plains, New York 10604.

Appendix I

Illustrated are components of a hypothetical inventory topology. This example illustrates that user SYSADMIN has group SYS1 as his default group and that he may also join group DEPT2. The



group SYS1 is a leaf on the group tree with superior nodes that represent department DEPT2, location LOC1, and DIVB within the root-node SYSTEM. (These relationships are indicated by the extra heavy organizational delegation lines.) Datasets are associated by name implication either with group—as exemplified by LOC11·A—or with a user—as exemplified by USERB·CNTL.

Appendix 2

A LISTREE command obtains part of the System/360 Model 195 inventory. This illustrates the group-tree structure and user connect entries.

```
listree group(k51) user

GROUP LEVEL          SUPERIOR GROUP=RES
-- 1  ---- 2  ---- 3  ---- 4  ---- 5  ---- 6  ---- 7  ---- 8  ---- 9  ---- 10 ---- 11 ----

K51
USER=LSB5803  ACCESS=USE
USER=RMR5802  ACCESS=USE
USER=ELW5803  ACCESS=JOIN
USER=GLADNEY  ACCESS=JOIN
--------K53
        USER=DRS4917  ACCESS=JOIN
        USER=JPJ4192  ACCESS=USE
        USER=CAG4192  ACCESS=USE
        USER=C221584  ACCESS=USE
----------------K534919
                USER=DRS4917  ACCESS=JOIN
------------------------NO SUBGROUPS
----------------K534917
                USER=SLK4917  ACCESS=USE
                USER=SSL4917  ACCESS=USE
                USER=FMS9909  ACCESS=USE
                USER=GSS4917  ACCESS=USE
                USER=DRS4917  ACCESS=JOIN
                USER=TCK4172  ACCESS=USE
                USER=PDH4917  ACCESS=USE
------------------------NO SUBGROUPS
----------------K534915
                USER=DRS4917  ACCESS=JOIN
------------------------NO SUBGROUPS
----------------K534918
                USER=DCH5803  ACCESS=USE
                USER=DAVEHIL  ACCESS=JOIN
                USER=BOB4915  ACCESS=USE
                USER=FRANK00  ACCESS=USE
                USER=WGT9969  ACCESS=USE
                USER=TCK4919  ACCESS=USE
                USER=DRS4917  ACCESS=JOIN
                USER=PUTZOLU  ACCESS=USE
------------------------K53DAVEH
                          NO USERS
--------------------------------NO SUBGROUPS
--------K52
        USER=SPC4462  ACCESS=USE
        USER=EES4467  ACCESS=USE
```

Appendix 3

A LISTGRP command gives more detail about the K51 group in Appendix 2. The presently unused fields for resource control are included.

```
listgrp group(k51) all

GROUP=K51      SUPERIOR GROUP=RES      AUTHOR=ELW5803  ACCT.NO=R8030000
  RESOURCE(USED,SUB-ALLOCATED,LIMIT)
  DA=(00000000,00000000,00000000) MACHINE UNITS=(00000000,00000000,00000000)
    SUBGROUP=K53
    SUBGROUP=K52
    SUBGROUP=K54
    SUBGROUP=K55
    USER=DSD5803  ACCESS=USE      USE=00000001
    USER=RMR5802  ACCESS=USE      USE=00000078
    USER=ELW5803  ACCESS=JOIN     USE=00000009
    USER=GLADNEY  ACCESS=JOIN     USE=00000002
```

Appendix 4

LISTUSER command informs the current user of his privileges in the K51, K54, and SPECIAL groups. Note that since the user who issued the command has no privilege relative to K54, the system has refused his request for information.

```
listuser group(k51,k54,special) all

USER=GLADNEY  NAME=H.GLADNEY              DEFAULT GROUP=SPECIAL
  BATCHDATE=074211   TSODATE=074226   PASSDATE=074198
  INSTALLATION DATA:K34/282-1361
  AUTHOR=PSB4164
  GROUP DATASET ACCESS = USER  NAME.
  DIVISION=2201     CTLG=NO     TSO=YES    REVOKE=NO
    GROUP=K51       ACCTNO=R8030000 AUTHORITY=JOIN    AUTHOR=ELW5803
    CTLG=YES    TSO=NO     REVOKE=NO
      USER LIMITS(USED,LIMIT)
        DA=(00000000,00000000) MACHINE UNITS=(00000000,00000000)
    DEFAULT UNIVERSAL ACCESS = READ .
 NOT AUTHORIZED TO LIST G=K54      ,U=DEFAULT
USER=GLADNEY  NAME=H.GLADNEY              DEFAULT GROUP=SPECIAL
  BATCHDATE=074211   TSODATE=074226   PASSDATE=074198
  INSTALLATION DATA:K34/282-1361
  AUTHOR=PSB4164
  GROUP DATASET ACCESS = USER  NAME.
  DIVISION=2201     CTLG=NO     TSO=YES    REVOKE=NO
    GROUP=SPECIAL  ACCTNO=40630000 AUTHORITY=JOIN    AUTHOR=JEF4165
    CTLG=NO     TSO=NO     REVOKE=NO
      USER LIMITS(USED,LIMIT)
        DA=(00000000,00000000) MACHINE UNITS=(00000000,00000000)
    DEFAULT UNIVERSAL ACCESS = READ .
```

Appendix 5

LISTDSE describes the use and access relevant to a user dataset and to a system dataset.

```
listdse da(c.clist) all

DSN=GLADNEY.C.CLIST
  INQUIRERS ACCESS=ALL          UNIVERSAL ACCESS=READ
                                                  AUTHOR=SPECIAL
  SPACE=00000004  VOL=SYS01C  UNIT=3330
     CREATE DATE=074043  LAST REF=074228  LAST CHG=074204
     OPEN COUNT:  INPUT=00000188  OUTPUT=00000003
       ID=SPECIAL    ACCESS=READ/WRITE  USE=00000000
       ID=K34        ACCESS=READ        USE=00000000
       ID=HENRY      ACCESS=ALL         USE=00000000
       ID=DES4165    ACCESS=NONE        USE=00000000


listdse da('sys1.linklib') all

DSN=SYS1.LINKLIB
  INQUIRERS ACCESS=NONE         UNIVERSAL ACCESS=READ
                                                  AUTHOR=R275897
  SPACE=00000450  VOL=SYSLN2  UNIT=2305-2
     CREATE DATE=074092  LAST REF=074226  LAST CHG=074220
     OPEN COUNT:  INPUT=00002583  OUTPUT=00000075
       ID=G936090    ACCESS=ALL         USE=00000000
       ID=EIN5802    ACCESS=ALL         USE=00000001
       ID=B306241    ACCESS=ALL         USE=00000293
       ID=M564921    ACCESS=READ/WRITE  USE=00000048
       ID=B706723    ACCESS=READ/WRITE  USE=00000002
```

Appendix 6

LISTINV supplies the identities of the datasets owned by RMR-5802, who is a user who is connected to group K51.

```
listinv g(rmr5802) all vol
@B.IKJRON6B.ASM                    VOL=BACK01   BACKED-UP
@B.J2.LOAD                         VOL=BACK01   BACKED-UP
@B.REPORT.PLI                      VOL=BACK02   BACKED-UP
@B.UPDTE.DATA                      VOL=BACK01   BACKED-UP
@L.VTOCREAD.PLI                    VOL=BACK01   BACKED-UP
ASM.ASM1                           VOL=MSTV13   MIGRATED
ASM.CLIST                          VOL=SYS015
BACKUP.ASM.ASM                     VOL=BCKU01
BACKUP.CNTL                        VOL=SYS024
BITLIST.CNTL                       VOL=MSTV22   MIGRATED
BITLIST.PLI                        VOL=MSTV24   MIGRATED
```

```
BUILDDS.CNTL                    VOL=MSTV07   MIGRATED
BUILDDS.PLI                     VOL=MSTV22   MIGRATED
C.LOAD                          VOL=SYS025
CAT.CNTL                        VOL=MSTV21   MIGRATED
CHECKRUN.PLI                    VOL=MSTV21   MIGRATED
CHKDS.CNTL                      VOL=MSTV21   MIGRATED
CLIST.ASM                       VOL=MSTV09   MIGRATED
CLIST.CLIST                     VOL=SYS025
CLIST.CNTL                      VOL=MSTV18   MIGRATED
CLIST.CNTL2                     VOL=MSTV18   MIGRATED
CLIST.LOAD                      VOL=MSTV29   MIGRATED
COMGLOB.DATA                    VOL=MSTV29   MIGRATED
C2.LOAD                         VOL=MSTV04   MIGRATED
DATASETS                        VOL=SYSIMV
DATASET3                        VOL=SYSIMV
DEL                             VOL=SYSIMV
DELETE.CNTL                     VOL=SYS013
DELETE.PLI                      VOL=MSTV04   MIGRATED
DELINK0.CLIST                   VOL=SYS015
```

Reprint Form No. G321-5011