Proc. of the Invitational Workshop on Data Integrity, National Institute of Standards and Technology, Special Publication 500-168, September 1989, section A.4, pages 1-14.

TERMINOLOGY, CRITERIA AND SYSTEM ARCHITECTURES FOR DATA INTEGRITY

Ravi Sandhu
Department of Computer and Information Science
The Ohio State University, Columbus, Ohio 43210

Abstract. In response to the strawman document [9] we propose that trust be treated as synonymous with integrity rather than synonymous with confidence. We also propose that mandatory controls be taken to mean controls based on properties of the object and/or the subject. Label-based mandatory controls are then a special case of this more general notion. The TCSEC [11] presents criteria for establishing prescribed levels of confidence in trusted systems with particular objectives. We consider how these criteria might be generalized to a broader context. Finally regarding architectures for trusted systems we suggest enhancements to the current security kernel approach.

1 INTRODUCTION

This paper discusses three interrelated topics pertaining to data integrity. In the spirit of this workshop the concepts are not presented as final, definitive or absolute. They do raise many interesting questions which must be confronted, in one form or another; even if the terminology suggested here needs modification and refinement, as it almost surely will.

Our first topic concerns basic terminology for which we have specific proposals regarding "trust" and "mandatory controls." For the most part we agree with the positions argued in the strawman document [9] for this workshop. The document is a significant contribution to our understanding of integrity and lays the foundation for productive debate in future. However we suggest that trust be treated as synonymous with integrity rather than synonymous with confidence. We believe the same arguments used to support a binary view of integrity also apply to the notion of trust. That is trust is a binary property, relative to some context, in whose evaluation we have varying degrees of confidence. The main advantage of our proposal is its explicit recognition that there are two independent issues involved in evaluating trust:

1. What functions is the system trusted to perform or not perform?

2. What is the degree of confidence in our trust?

Regarding mandatory controls we propose the notion be generalized so it is not tied to labels. In our view label-based mandatory controls are a special case of controls based on properties of the object and/or the subject. In the military non-disclosure context these properties turn out to be best expressed as partially ordered labels, obtained by combining levels and compartments. In other contexts these properties are more naturally obtained in other ways. For instance the type of an object determines what operations can be executed on it. The security community has no handy term for "controls based on properties of the object and/or the subject" although their fundamental importance has often been recognized [3, 5, 18, 24, 26, for instance]. We propose the term mandatory controls be used in this broad sense and that it be qualified when a specific property is intended, such as in label-based mandatory controls.

Our second topic concerns criteria for evaluating trusted systems. The TCSEC [11] recognizes the separation between functionality and confidence noted above when it states, "Included are two distinct sets of requirements: 1) specific security feature requirements; and 2) assurance requirements." However in its criteria this separation is not clearly maintained. In the transition from one class to the next in the C1 through B3 range both functionality and degree of confidence are simultaneously increased. Whereas in the B3 to A1 transition the functionality is unchanged but the assurance requirements are substantially higher. In the classified sector, with its specific objectives for control of documents, there may be a logical joint progression of functionality and degree of confidence. But in a broader context these two issues are best kept separate. Especially in the commercial arena, there is a need for systems with relatively primitive functionality but with high levels of confidence in their evaluation as trusted. For instance the function might be limited to requiring a audit trail which can be trusted with a high degree of confidence. We need finer criteria to enable users to select the combination of functionality and level of confidence in a trusted system that suit their needs within their budgetary constraints.

Our third topic concerns architecture for trusted systems. The conventional approach to security kernels [13] places all trusted code in the kernel and does not trust any code outside the kernel. This approach has been reasonably successful in the non-disclosure context.[1] For a broader context we suggest a multi-layered approach with a table-driven kernel. The kernel is trusted, with a high level of confidence, to enforce the policy specified in its policy tables. These tables are static so they can be placed in read-only memory when the system is "built." Our objective is to separate policy from mechanism to exploit commonality of mechanism across a variety of policies. This kernel can support the "access-control triple" called for by Clark

[1] Even in this context there are significant deviations from this ideal. The notion of a trusted subject, which is outside the kernel but must nevertheless be trusted, has slowly crept in. So the architecture is anyway moving in the direction we are suggesting.

and Wilson [5, 7] as well as separation of duties, perhaps by using transaction control expressions [29]. Above the kernel are trusted layers of application-independent and application-specific code. Code for well-formed transactions, application specific auditing, and policies not directly supported by the kernel tables resides here.

2 TERMINOLOGY

We have specific proposals regarding the terms "trust" and "mandatory controls." Our main objective concerning trust is to emphasize two distinct issues in its evaluation, viz., functionality and degree of confidence. This distinction is important in establishing criteria for evaluating trusted systems. Regarding mandatory controls we attempt to generalize the notion so that label-based controls turn out to be a special case. We specifically propose that mandatory controls be used to mean controls based on properties of the object and/or the subject.

2.1 TRUST

The definition of integrity presented in the strawman paper [9] has two salient features.

I. Integrity is a *binary* property. An object either has integrity or it does not.

II. Integrity is *relative* to an a priori expectation of quality in some context. So the same object can have integrity in one context and be devoid of it in another.

The definition appears intuitively sound and useful and provides much needed clarification of terminology. It does leave open the question of what is quality, which is noted as the next task for the Integrity Working Group.

The strawman paper goes on to assert, "Integrity reflects trust in quality. There can, however, be degrees of trust." In appendix I we find the statement of Willis Ware that, "the confidence level in the decision about determination of presence/absence of integrity is indeed continuous in nature and might well be called the "level of trust" in the decision that has been made." The term "degree of trust" is also used in the TCSEC. For instance one of its objectives is stated to be, "to provide users with a yardstick with which to assess the degree of trust that can be placed in computer systems for the secure processing of classified or other sensitive information."

We agree with the spirit of these statements but propose some modifications. The first, and relatively minor, observation is that confidence levels need not be continuous in a mathematical sense.[2] We propose the following statement to allow for different ways of measuring confidence.

[2] This implication may not be intended but it nevertheless needs clarification.

III. Confidence in our decision as to whether or not an object has integrity, in some given context, is a concept to which degrees or levels (qualitative or quantitative, continuous or discrete, totally ordered or partially ordered) can be assigned. So it is proper to talk about *degrees or levels of confidence*.

The second, and more important, proposal is that trust and confidence should not be treated as synonymous. Instead trust should be treated as a synonym for integrity. Trust is a concept applied for the most part to active agents. It implies an a priori expectation about some aspect of the agent's behavior in a particular context. To be trusted the agent's behavior need only be no worse than we thought it would be. Compare this with the statement [9], "To have integrity it (the quality of an object) need only be no worse than we thought it was." If integrity is treated as a binary attribute, for consistency trust should also be binary. Contrast the following statements.

1. This data has integrity with a high degree of confidence.
2. This data has integrity with a high degree of trust.

The appropriateness of the first statement has been well argued in the strawman paper. We propose the second statement be treated as inadmissible and meaningless, at least in a technical sense. Trust comes in play when we have no choice but to use data in whose integrity we have little confidence. What is being trusted in such cases? We are trusting that all active agents who could have degraded the quality of this data did not do so.

Is this merely hair-splitting? Perhaps, but the proposal is worth investigating if only to spell out its consequences. The proposed viewpoint clearly separates two issues involved in evaluating a system's trust.

1. What functions is the system trusted to perform or not perform?
2. What is the degree of confidence in our trust?

Of course these questions are raised even if trust is viewed as a non-binary attribute. They do become more explicit if trust is defined as a binary property. Moreover there is a recognition that these are really two independent issues.

Our proposal is in direct conflict with the concept of trust in the strawman paper. Consider the following quote from appendix I.

> "... trust is not a binary attribute as the word is commonly used. It is understood that someone (or some thing) is trusted for some purpose (but not others); we are all accustomed to estimating degree of trust in our daily lives. We often say: "Can I trust someone (or some thing) for ..."; thus, we make a value judgment in terms of some end goal, and

> implicitly the goal has a threat attached to it. This is precisely what trusted systems are all about ..."

It appears inconsistent to us that integrity is a binary property but trust is not. The strawman paper argues that integrity is a binary attribute, which is relative to some context, and in which we legitimately can have varying degrees of confidence. The examples quoted above show trust is relative, however, they do not show trust is non-binary.

If we accept trust as a synonym for integrity, applicable mostly to active entities, we can specialize the assertions of the strawman paper as follows.

IV. Trust is a *binary* property usually applied to active agents or subsystems which contain one or more active agents. An active entity is either trusted or not.

V. Trust is *relative* to an a priori expectation of quality, particularly quality of behavior, in some context. So the same agent can be trusted in one context and untrusted in another.

VI. It is proper to talk about *degrees or levels of confidence* regarding the decision as to whether or not an agent is trusted, in some given context.

To be concrete consider the following statements where the term process is used in the technical operating systems sense of an executing program.

1. This process has integrity with a high degree of confidence.
2. This process can be trusted with a high degree of confidence.
3. This process has integrity with a high degree of trust.

We propose the first two statements be treated as synonymous. The only difference being that the second statement draws attention to the active nature of a process and quality of its behavior. The third statement we submit should be inadmissible and meaningless, at least in a technical sense. On the other hand, if we equate trust with confidence, statements 1 and 3 above are equivalent while statement 2 can be rephrased as follows.

4. This process can be trusted with a high degree of trust.

Now this is obviously circular and of questionable value in a technical vocabulary. It can only serve to confuse the issue. Our proposal is to treat statements 3 and 4 as inadmissible. The strawman paper in effect takes the position that 2 and 4 are inadmissible. Given a choice between keeping statement 2 or 3 we believe the choice is clearly in favor of 2.

2.2 MANDATORY CONTROLS

The TCSEC draws a sharp distinction between discretionary controls based on identity and mandatory controls based on labels. There appears to be a consensus that a more general notion is needed which is not tied to labels. Consider the following quote from the first WIPCIS report [22].

> "... two types of mandatory controls are considered here — label-based mandatory controls (enforcing separation based on hierarchical or lattice oriented labels, as in the Orange Book) and general mandatory (which lies between label-based mandatory and discretionary controls)."

We are troubled by this characterization of general mandatory as lying between label-based mandatory and discretionary controls. On the contrary we propose that general mandatory be defined so label-based mandatory controls are a special case of whatever we call general mandatory.[3]

A reasonable working definition is given by Clark and Wilson [6] as follows.

> "... the word "mandatory." In the paper, we want to use it in the more general way, to describe any mechanism which is not put into place at the control of the owner of the data, but which is a necessary part of the operation of the system."

However there are situations where mandatory controls are defined by the owner. For example the owner of checks is responsible for defining the well-formed transactions which can operate on checks as well as for defining the separation of duty requirements for processing checks. Once these decisions have been made, at the owner's discretion, the resulting controls are mandatory for all other users. Clark and Wilson also have another working definition [7] as follows.

> "In this case we define mandatory as those controls which are unavoidably imposed by the operating system between user and data."

This is very broad and can be interpreted to include discretionary controls.

We propose to define mandatory controls as controls based on properties of the object and/or the subject. This is as broad and open ended as the above. However it does suggest that one can categorize mandatory controls in terms of the properties on which the controls are based. In the military non-disclosure context these properties turn out to be best expressed as partially ordered labels. In other contexts these

[3] It is not surprising the meaning of these terms is still controversial. Many common terms, such as virtual memory, process, fairness, etc., remain controversial even after years of productive use. The goal is not so much to discover the Platonic ideal meaning, but rather to assign meanings which are practically useful, technically consistent and widely accepted (eventually).

properties are more naturally obtained in other ways. For instance the type of an object determines what operations can be executed on that object. Subjects are divided into two classes for this purpose: the type manager who can execute arbitrary operations and all others who can only execute operations exported by the type manager.

Traditional lattice-based controls [2, 10, 17] are obviously a special case of our definition. Discretionary controls are also a special case. Consider the TCSEC definition of discretionary controls as "a means of restricting access to objects based on the identity of subjects and/or the groups to which they belong." So in this case the property being used is identity and group membership.

We can exclude discretionary controls by refining the definition of mandatory controls to be "controls based on properties of the object and/or the subject (excluding identity of the subject and/or the groups to which it belongs)." In our opinion it is not unreasonable to actually consider discretionary controls as a special case of mandatory controls. We believe the traditional black and white distinction between discretionary and mandatory controls is inappropriate in many contexts. All authority in a system is ultimately obtained by means of somebody's discretionary decisions [20, 21]. The real difference is to what extent discretionary ability can be granted and acquired during the normal operation of a system, and to what extent it gets fixed at system initialization.

Our proposal allows us to categorize mandatory controls along different dimensions. For instance, consider the following progression.

1. Controls based on *identity*. As discussed above this includes discretionary controls.

2. Controls based on *static properties* of the object and subject. These properties are determined at creation and do not change thereafter. Label-based controls of the Bell and LaPadula model [2] with strong tranquillity (i.e., labels are static) are a well-known example. The type based controls of the schematic protection model [26, 28] are a more general example.

3. Controls based on *dynamic properties* of the object and subject. That is the properties on which the controls are based are themselves changeable, presumably in some controlled manner requiring proper authorization. Controls based on the history of an object and the role of a subject, such as enforced by transaction control expressions [29], are one example. Another example is label-based controls without tranquillity [19] (i.e., labels can be modified).

If we assume identity is immutable, 1 is a special case of 2. Similarly, 2 is a special case of 3 if dynamic is interpreted to include static. So there is a logical progression.

Group membership does not figure in the above categorization. This is deliberate. If group membership is a static attribute we could include it in under 2. However the moment group membership is dynamic a new set of questions is raised [25]. For example consider the following policy.

1. A project group must have a majority of members from within the department.

2. Any department member can unilaterally join any project group.

3. An outsider can be enrolled in a group only by a project supervisor.

This is by no means a complicated policy. Yet there are no systems today which can conveniently support it. A C2 system is not good enough since it cannot enforce the specified mandatory controls. Neither do the labels of B or A systems help.

Another categorization might consider the nature of these properties along a different dimension, for instance as follows.

1. Controls based on properties of an object which *depend on the value* of the data it contains.

2. Controls based on properties of an object which are *independent of the value* of the data it contains.

Such distinctions are important for two reasons. First we can conclude that certain kinds of controls are needed to achieve particular objectives. For instance dynamic separation of duties appears to require controls based on the history of an object whereas static separation can be achieved by controls based on static properties. This gives us guidelines regarding what features are required to achieve our objectives. Secondly we can design operating system mechanisms with the fundamental nature of the controls in mind rather than considering specific applications.

3 CRITERIA

We now turn to consideration of criteria for evaluating trusted systems. There are two major points we wish to make in light of the preceding discussion. Firstly the criteria must clearly separate the issues of functionality and degree of confidence. Secondly we need a finer grain of functionality than provided in the TCSEC.

Separation between functionality and confidence is noted in the TCSEC in its statement, "Included are two distinct sets of requirements: 1) specific security feature requirements; and 2) assurance requirements." However in the TCSEC classes this separation is not clearly maintained. In moving up to higher classes (e.g., from C1 to C2) there is an increase in functionality as well as an increase in the level of confidence required. The sole exception is the B3 to A1 transition which does not introduce additional functionality but considerably increases the required degree of confidence. Clark and Wilson [7] have tentatively proposed a similar progression for integrity evaluation criteria. Since their objective was to arrive at an integrated set of criteria for evaluation for integrity and non-disclosure it is natural that their proposal mirrors the TCSEC. Specifically they propose three divisions with the C to B transition requiring a simultaneous increase in functionality and assurance while the B to A transition is mostly concerned with assurance.

While some joint progression is inevitable we feel it is inappropriate to couple these two issues too tightly. Especially in the commercial arena, there is a need for systems with relatively primitive functionality but with high levels of confidence in their evaluation as trusted. For instance the function might be limited to requiring a audit trail which can be trusted with a high degree of confidence. This makes breach of trusted behavior by individual users detectable so threat of punitive action may be enough to prevent it.

Moreover the progression of functionality defined in the criteria should be finer grained. This is especially so if an integrated set of criteria are proposed. For instance consider a user who requires high confidence in authentication but is willing to accept low confidence for non-disclosure. In the TCSEC the trusted path required for the former is coupled with the requirement of covert channel analysis. We view functionality as inherently multi-dimensional. For instance consider the following progression of functionality of mandatory controls.

Class	Mandatory controls based on
a	Dynamic properties including value
b	Limited dynamic properties (e.g., transaction control expressions)
c	Almost static properties (e.g., weak typing, labels without tranquillity)
d	Static properties (e.g., strong typing, labels with tranquillity)
e	Identity based (i.e., only discretionary controls)

We do not think it proper to require that a system which includes features of class a must also include all features of the classes below a. The progression is multidimensional. For instance we may require sophisticated support for hierarchical groups [27] and some little amount of controls from the other classes. As a user one would like a rating which measures the features provided in each one of these classes so one can shop for the best match. Levels of confidence might be assigned in a strictly increasing sequence, for instance as follows.

Class	Level of confidence
a	Formally verified
b	Informally verified
c	Extensive testing
d	Minimal confidence
e	No confidence

However if we have multi-dimension functionality we expect a different level of confidence to be attached to each dimension to give a multi-dimensional rating.

As a general principle of system design a user should be able pay for the functionality and level of confidence in a trusted system that suits his needs. He should be able to trade one for the other to meet budgetary constraints. The marketplace and technology will determine what combinations of functionality and confidence get supported and at what price. With proper modular designs, system vendors should be able to support large numbers of combinations. It should eventually be possible to upgrade from one package of functionality and confidence to a superior one. We need finer criteria to enable users to select the combination of functionality and level of confidence in a trusted system that suit their needs within their budgetary constraints.

4 SYSTEM ARCHITECTURES

Our third topic concerns architecture for trusted systems. The conventional approach to security kernels [1, 13, 16] places all the trusted code in the kernel and does not trust any code outside the kernel. The ideal picture is given somewhat as follows.

Users
Applications
Operating System
Security Kernel

In practise the ideal is often violated by placing some trusted code outside the security kernel. This gives us the following view.

Everything Else
Trusted Functions
Security Kernel

There are two major reasons why we need trusted functions, or trusted processes, outside the security kernel.

1. Trusted processes need to bypass mandatory controls of the security kernel in order to achieve their objective. For example a downgrader must selectively violate confinement.

2. Trusted processes perform functions which are security related, so we need a high level of confidence in their correctness, but these do not need to violate confinement. For example a labeler or a backup utility.

Trusted functions are clearly part of the overall security component of a system. Trusted functions which are privileged to bypass kernel controls are particularly disturbing. By their very nature, it is difficult to come up with a rigorous definition of what these processes are supposed to do. Without a precise specification it becomes somewhat pointless to try and verify them.

It is possible for enforcement mechanisms in the security kernel to help us increase the level of confidence in these trusted functions. For instance SCOMP [12] uses "integrity labels" for this purpose while SAT [4] provides a type enforcement mechanism. Of course, neither of these can guarantee the correctness of trusted functions. The controls increase our confidence by making it more difficult to plant Trojan Horses in trusted code as well by limiting the damage that might be done. The SAT approach is particularly flexible and attractive.

In view of this experience we propose the following idealized architecture for trusted systems.

Everything Else
Application Dependent Trusted Functions
Application Independent Trusted Functions
Enforcement Kernel

For the moment let us ignore the non-disclosure problem. What then might one expect in these layers? We propose the enforcement kernel implement mandatory controls which are for the most part based on static properties of subjects and objects. To separate policy from mechanism the kernel should be table-driven. This is by no means a new idea [4, 8, 23] and has obvious appeal. The separation of trusted functions outside the kernel, into application independent and application dependent, is intended to encourage reuse of trusted functions across applications.

We propose the mandatory policy of the kernel be defined in terms of types of subjects and objects, and that the kernel enforce strong typing (i.e., the type of a subject or object is determined when it is created and thereafter does not change). Type-based controls are surprisingly powerful. They gives us the basis for enforcing the "access control triple" of Clark and Wilson [5, 7]. They also provide enforcement of, even dynamic, separation of duties by means of transaction-control expressions [29] or some similar mechanism. There is also evidence that policies based on types are easier to analyze for safety [26] as compared with policies specified without a built-in notion of types [14, 15].

How does non-disclosure fit into this picture? There are several approaches one might take. If labels are regarded as a special case of types the enforcement kernel can handle label-based controls. This has been formally demonstrated in [28]. Of course the covert channel problem remains. So, to achieve B2 or higher ratings, the enforcement kernel and policy tables must be scrutinized for covert channels. Another approach could be to run the enforcement kernel above a traditional security kernel. Such a security kernel could be stripped of all features not related to label-based confinement and conceivably could be small enough to meet A1 criteria. The enforcement kernel could then be viewed as an integrity kernel sitting above the security kernel. Since projects such as SAT are proposing similar features it may be better to split them vertically into two layers.

5 CONCLUSION

To summarize we have considered several topics relating to data integrity.

1. In response to the strawman document [9] we propose that trust should be viewed as a synonym for integrity used mostly to describe active agents or systems containing such agents. This serves to emphasize two distinct issues in evaluating trust, viz., functionality and degree of confidence.

2. We propose that mandatory controls be used to mean controls based on properties of the object and/or the subject. Label-based controls are then a special case of this more general notion. If we choose we can exclude discretionary controls by excluding properties based on identity of the subject and/or the groups to which it belongs. Otherwise we can view discretionary controls as a very special case of mandatory controls.

3. With the above perspective we have argued that criteria for evaluating trusted systems must clearly separate the issues of functionality and degree of confidence. While some joint progression may be inevitable we feel it is inappropriate to couple these two issues too tightly. We have also argued that the criteria need to be finer than those presented in the TCSEC [11].

4. Finally we have considered the security kernel approach to trusted systems architecture. Experience indicates trusted code is needed outside the kernel [13]. This is even more so for application specific integrity policies. The architecture should support a distinct layer above the kernel in which this code resides. Policy and mechanism should be separated in the kernel. Policy should be specified by static policy tables which can only be changed, under careful control, when the system is built. A built-in notion of types should be provided for specifying these policies.

In the spirit of this workshop these proposals are somewhat speculative and need further work. Many of the questions we have raised must eventually be confronted, in one form or another, even if the terminology in which they are phrased is modified.

It appears to us that mandatory controls for integrity will be an order of magnitude more complex than label-based mandatory controls for non-disclosure, in all respects except for the formidable covert channel problem. This is the distinguishing characteristic which makes non-disclosure such a difficult problem. As a final note we express our support for the following viewpoint [20].

> "Many people have assumed that security policies for commercial systems are either less friendly versions of academic policies or military policies with fewer teeth. Neither is true. Considerations for commercial security policy differ in quality, because the principles of internal control take a much more sophisticated view of authority."

References

[1] Ames, S.R., Gasser, M. and Schell, R.R. "Security Kernel Design and Implementation: An Introduction." *Computer* 16(7):14-22 (1983).

[2] Bell, D.E. and LaPadula, L.J. "Secure Computer Systems: Unified Exposition and Multics Interpretation." MTR-2997, Mitre, Bedford, Mass. (1975).

[3] Boebert, W.E. and Kain, R.Y. "A Practical Alternative to Hierarchical Integrity Policies." *8th National Computer Security Conference*, 18-27 (1985).

[4] Boebert, W.E., Kain, R.Y., Young, W.D. and Hansohn, S.A. "Secure Ada Target: Issues, System Design, and Verification." *8th National Computer Security Conference*, 18-27 (1985).

[5] Clark, D.D. and Wilson, D.R. "A Comparison of Commercial and Military Computer Security Policies." *IEEE Symposium on Security and Privacy*, 184-194 (1987).

[6] Clark, D.D. and Wilson, D.R. "Comments on the Integrity Model." In [30].

[7] Clark, D.D. and Wilson, D.R. "Evolution of a Model for Computer Integrity." These proceedings.

[8] Cohen, E. and Jefferson, D. "Protection in the Hydra Operating System." *5th ACM Symposium on Operating Systems Principles*, 141-160 (1975).

[9] Courtney, R.H. "Some Informal Comments About Integrity and the Integrity Workshop." These proceedings.

[10] Denning, D.E. "A Lattice Model of Secure Information Flow." *Communications of ACM* 19(5):236-243 (1976).

[11] Department of Defense National Computer Security Center. *Department of Defense Trusted Computer Systems Evaluation Criteria.* DoD 5200.28-STD, (1985).

[12] Fraim, L.J. "Scomp: A Solution to the Multilevel Security Problem." *Computer* 16(7):26-34 (1983).

[13] Gasser, M. *Building a Secure Computer System.* Van Nostrand Reinhold (1988).

[14] Harrison, M.H., Ruzzo, W.L. and Ullman, J.D. "Protection in Operating Systems." *Communications of ACM* 19(8):461-471 (1976).

[15] Harrison, M.H. and Ruzzo, W.L. "Monotonic Protection Systems." In DeMillo, R.A., Dobkin, D.P., Jones, A.K. and Lipton, R.J. (Editors). *Foundations of Secure Computations.* Academic Press (1978).

[16] Landwehr, C.E. "The Best Available Technologies for Computer Security." *Computer* 16(7):86-100 (1983).

[17] Lee, T.M.P. "Using Mandatory Integrity to Enforce "Commercial" Security." *IEEE Symposium on Security and Privacy*, 140-146 (1988).

[18] Linden, T.A. "Operating System Structures to Support Security and Reliable Software." *ACM Computing Surveys* 8(4):409-445 (1976).

[19] McLean, J. "The Algebra of Security." *IEEE Symposium on Security and Privacy*, 2-7 (1988).

[20] Moffett, J.D. and Sloman, M.S. "The Source of Authority for Commercial Access Control." *IEEE Computer* 21(2):59-69 (1988).

[21] Murray, W. H. "On the Use of Mandatory." Position paper in [30].

[22] Parker, D.B. and Neumann, P.G. "A Summary and Interpretation of the Invitational Workshop on Integrity Policy in Computer Information Systems." In [30].

[23] Popek, G.J. and Farber, D.A. "A Model for Verification of Data Security in Operating Systems." *Communications of ACM* 21(9):737-749 (1978).

[24] Saltzer, J.H. and Schroeder, M.D. "The Protection of Information in Computer Systems." *Proceedings of IEEE* 63(9):1278-1308 (1975).

[25] Sandhu, R.S. and Share, M.E. "Some Owner Based Schemes with Dynamic Groups in the Schematic Protection Model." *IEEE Symposium on Security and Privacy*, 61-70 (1986).

[26] Sandhu, R.S. "The Schematic Protection Model: Its Definition and Analysis for Acyclic Attenuating Schemes." *Journal of ACM* 35(2):404-432 (1988).

[27] Sandhu, R.S. "The NTree: A Two Dimension Partial Order for Protection Groups." *ACM Transactions on Computer Systems* 6(2):197-222 (1988).

[28] Sandhu, R.S. "Expressive Power of the Schematic Protection Model." *Computer Security Foundations Workshop*, 188-193 (1988).

[29] Sandhu, R.S. "Transaction Control Expressions for Separation of Duties." *4th Aerospace Computer Security Applications Conference*, 282-286 (1988).

[30] Report of the Invitational Workshop on Integrity Policy in Computer Information Systems (WIPCIS), Bentley College, MA, October 1987, NIST.