# Chapter 6

# Implement Threat Control Measures

This chapter describes the third component of an effective information security/IA program — implementing threat control measures. Outputs from the previous component, performing vulnerability and threat analyses, are used to prioritize the implementation of threat control measures. The following activities are performed during the implementation of threat control measures:

- The extent of protection needed is determined.
- Controllability, operational procedures, and in-service considerations are evaluated.
- Plans are made for contingencies and disaster recovery.
- The use of perception management is considered.
- IA design features and techniques are selected and implemented.

This book purposely uses the term "threat control measures" rather than "countermeasures." Countermeasures denote reactive responses to attacks. In contrast, threat control measures designate a proactive strategy designed to reduce the incidence of successful attacks and the severity of their consequences.

In 1991, Zebroski[445] identified 11 common precursors to an accident/incident by analyzing several major engineering catastrophes. Recently, Long and Briant[340] developed a corresponding proactive mitigating response to each accident precursor. While Zebroski, Long, and Briant focused on safety, their results are equally applicable to security and reliability. It is interesting to review these recommendations in the context of threat control measures because it demonstrates why the psychology of individuals and organizations — both insiders and outsiders — must be considered in addition to technical issues (see Exhibit 1).

## Exhibit 1 Proactive Responses to Common Accident/Incident Precursors

| *Accident/Incident Precursor*[a] | *Proactive Response*[b] |
|---|---|
| 1. Risk management techniques not used | 1. Use proven techniques as an aid to technical excellence (see Annex B):<br>■ IA analysis techniques (Exhibit B.2)<br>■ IA design techniques and features (Exhibit B.3)<br>■ IA verification techniques (Exhibit B.4)<br>■ IA accident/incident investigation techniques (Exhibit B.5) |
| 2. Little preparation for severe events | 2. Prepare and practice for emergencies:<br>■ Involve all stakeholders in contingency planning<br>■ Keep contingency plans up to date<br>■ Conduct practice drills regularly |
| 3. Invincible mindset | 3. Respect for technology:<br>■ Awareness of associated hazards<br>■ Awareness of residual risk exposure<br>■ Anticipating and preparing for the unexpected |
| 4. Unnecessary acceptance of hazards in system design or operation | 4. Maximize safe, secure, and reliable system design and operation:<br>■ Eliminate avoidable hazards, reduce remaining hazards to ALARP<br>■ Conduct regular interdisciplinary safety and security reviews<br>■ Conduct regular independent safety and security reviews |
| 5. Safety, security, reliability matters not recognized or integrated into work of organization | 5. Clearly defined responsibilities and authority for safety, security, and reliability matters:<br>■ Real authority<br>■ Prominent organizational role<br>■ Appropriate staffing levels and competency |
| 6. Low priority given to safety, security, and reliability | 6. Safety, security, and reliability are paramount:<br>■ Safety, security, and reliability are an integral part of system design and operation<br>■ Employees accept responsibility for their actions |
| 7. No systematic processing of experience from elsewhere | 7. Learning from others' experiences:<br>■ Systematic gathering and analysis of lessons learned from other projects: within and outside organization, industrial sector, and country |
| 8. Lessons learned disregarded | 8. Learning from ourselves:<br>■ Encouraging open discussion of what went wrong and why<br>■ Implementing corrective and preventive action |
| 9. Compliance means safe, secure, reliable enough | 9. Striving for excellence:<br>■ Compliance necessary, but insufficient in itself<br>■ Continuous process/product improvement |
| 10. Groupthink instead of teamwork | 10. Teamwork with robust decision-making:<br>■ Open sharing of ideas, opinions, concerns<br>■ Valuing dissenting views |

**Exhibit 1 Proactive Responses to Common Accident/Incident Precursors (continued)**

| *Accident/Incident Precursor*[a] | *Proactive Response*[b] |
|---|---|
| 11. Diffuse responsibilities | 11. Accountability and openness:<br>■ Clearly defined duties and responsibilities<br>■ Procedures that allow room for professional judgment<br>■ Open communication channels |

[a] Column one summarized/adapted from Zebroski, E., Lessons Learned from Catastrophes, *Risk Management: Expanding Horizons in Nuclear Power and Other Industries*, (Knief, R. et al., Eds.), Hempshire Publishing Corporation, 1991.

[b] Column two summarized/adapted from Long, R. and Briant, V., Vigilance Required: Lessons for Creating a Strong Nuclear Culture, *Journal of System Safety*, Q4, 1999, pp. 31–34.

Exhibit 2 illustrates the chronology of threat control measures. These measures may be proactive or reactive, depending on whether or not they are preceded by proper planning and analyses. It goes without saying that actions which are preplanned will be more successful than those which are not. Initially, threat control efforts focus on preventing faults, failures, vulnerabilities, and attacks that could compromise a system or render it inoperable. Given that this will not be possible in all situations, the ability to detect faults, failures, and attempted attacks is provided.[344] Anomalous behavior is characterized to contain the consequences and formulate an appropriate response. Once the appropriate response is taken, effort is focused on instigating recovery[344] or an emergency shutdown, as the situation warrants, and a return to normal operations. The following chapter sections discuss how these threat control measures are implemented.

## 6.1 Determine How Much Protection Is Needed

Chapter 4 explained how to determine what needs to be protected and why. Chapter 5 explained how to determine the initial risk exposure. Now it is time to determine the type, level, and extent of protection needed. The type, level, and extent of protection needed is unique to each system. A variety of factors are analyzed and synthesized to determine how much protection a particular system needs, such as (see Exhibit 3):

1. A comparison of the initial risk exposure to the target risk exposure
2. An analysis of system entities and functions that are IA critical or IA related
3. The specification of must work functions (MWFs) and must not work functions (MNWFs)
4. A reassessment of entity control analysis
5. An evaluation of the time element relative to proposed threat control measures
6. A reexamination of privacy issues

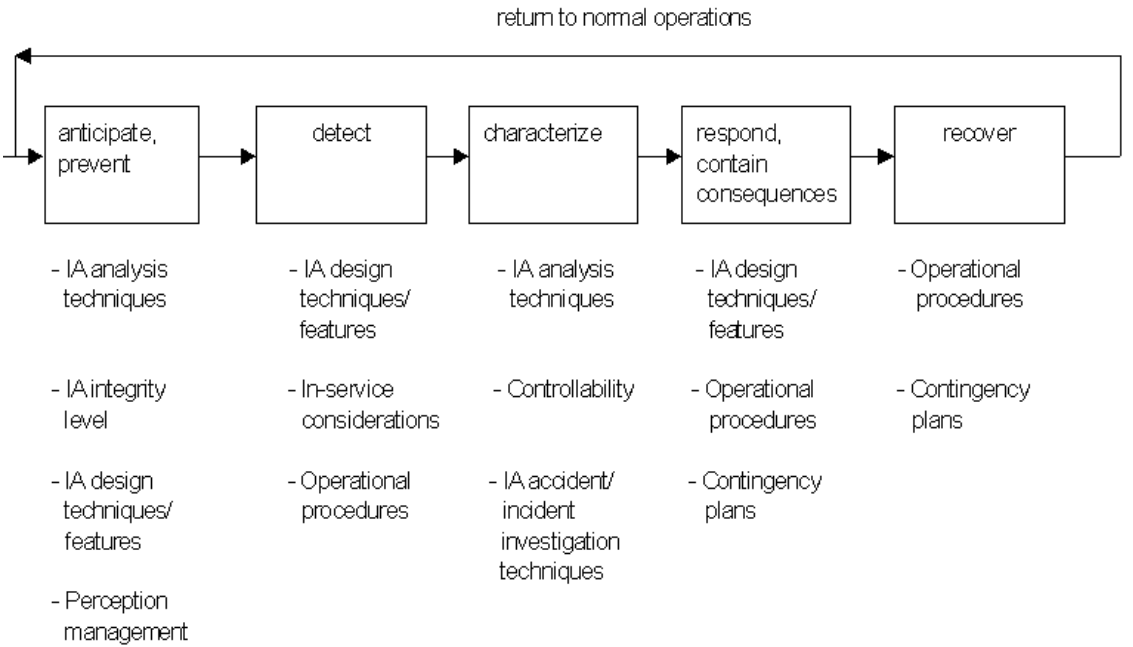


**Exhibit 2 Chronology of Threat Control Measures**

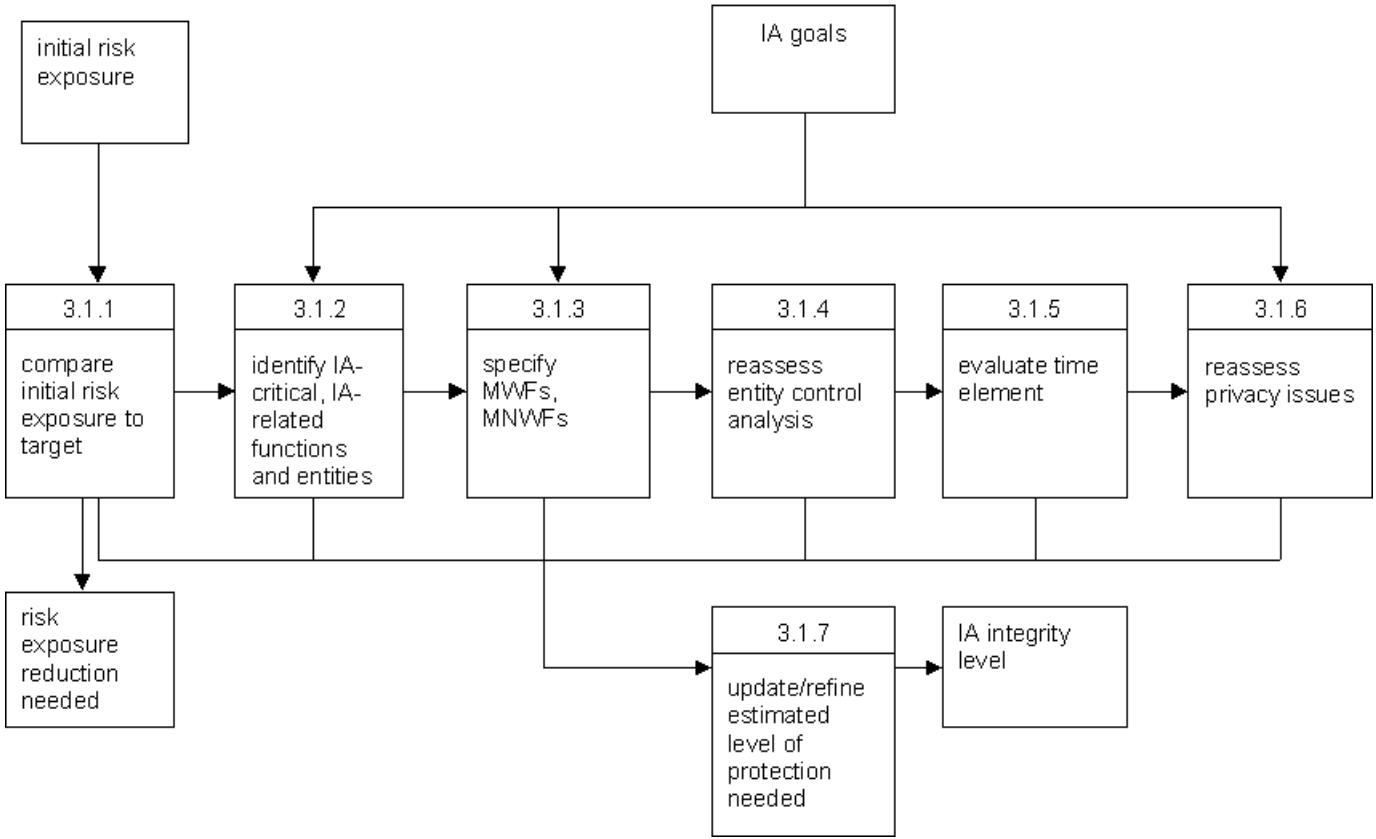


**Exhibit 3 Summary of the Activities Involved in Determining the Level of Protection Needed**

**Exhibit 4 High-Level Identification of Entity Criticality**

| *Example* | *IA-Critical Entity/Function* | *IA-Related Entity/Function* | *Not IA-Critical or IA-Related* |
|---|---|---|---|
| Radiation therapy system | Functions that control the release of radiation<br>Functions that verify that the type of radiation, dosage, etc. are within known safe parameters and consistent with the treatment plan | Functions that abort or inhibit the release of energy when an anomaly is detected | Remote billing system |
| Online banking system | Access control, authentication functions | Intrusion detection and response functions | Ancillary functions not related to account transactions |
| ATC system | Aircraft transmitter that sends location signal<br>Radar signal transmitter/ receiver<br>ATC signal receiver<br>ATC terminal displays | Voice communication system | Ancillary functions on ATC terminals not related to air traffic control mission |

The delta between the initial and target risk exposures is a prime factor in determining the extent of protection needed and the threat control measures to be implemented. The vulnerability and threat analyses (Chapter 5) produced the initial risk exposure, which melded the severity and likelihood of each vulnerability/threat pair with critical threat zones. This initial risk exposure is compared to specified IA goals, applicable laws and regulations, etc. to determine whether or not it is acceptable. In other words, is the initial risk exposure consistent with the target risk exposure? If so (a rare occurrence), no further action is needed. If not (the more likely case), threat control measures need to be implemented to reduce the initial risk exposure to the desired target. Residual risk is that which remains after threat control measures have been implemented. Chapter 7 explains how to assess the effectiveness of threat control measures.

A second factor used in determining the extent of protection needed is the identification of IA-critical and IA-related system entities and functions. IA-critical designates any condition, event, operation, process, or item whose proper recognition, control, performance, or tolerance is essential to the safe, reliable, and secure operation and support of a system. IA-related denotes a system or entity that performs or controls functions which are activated to prevent or minimize the effect of a failure of an IA-critical system or entity. A third category consists of entities and functions that are neither IA-critical nor IA-related. This distinction is illustrated in Exhibit 4 using the three hypothetical systems developed thus far.

The determination of the level of protection needed is tempered by the criticality of the system entity/function. After system entities and functions

have been identified as being IA-critical, IA-related, or neither, this information is correlated to the initial risk exposure. In other words, what is the risk exposure for IA-critical and IA-related entities and functions? Is it higher or lower than that for entities and functions that are not IA-critical or IA-related?

Many international standards recommend implementing automatic inhibits to prevent the inadvertent operation of IA-critical functions. The standard formula is that two independent inhibits are required to prevent the inadvertent operation of an IA-critical function that could result in a critical accident/incident, while three independent inhibits are required to prevent the inadvertent operation of an IA-critical function that could result in a catastrophic accident/incident.[18,31,60,64,125,127,130]

A third factor used in determining the extent of protection needed is the specification of must work functions (MWFs) and must not work functions (MNWFs). The concept of and need to specify MWFs and MNWFs originated with NASA. An MWF is software that if not performed or performed incorrectly, inadvertently, or out of sequence could result in a hazard or allow a hazardous condition to exist; for example: (1) software that directly exercises command and control over potentially hazardous functions or hardware, (2) software that monitors critical hardware components, and (3) software that monitors the system for possible critical conditions or states.[126,127] An MNWF is a sequence of events or commands that is prohibited because it would result in a system hazard;[126,127] that is, an illegal state that would have severe consequences. Remember that in the IA domain the consequences of a hazard may or may not be physical.

In most cases, MWFs will be IA-critical. The logic that prevents an MNWF from executing is also IA-critical. The risk exposure of MWFs and MNWFs is evaluated as described above to determine the level of protection needed.

Often, there is a tight coupling between MWFs and MNWFs (see Exhibit 5). For example, in simplest terms, an MWF may specify:

$$\text{if } x \rightarrow \text{then } y$$

A corresponding MNWF may be:

$$\text{if not } x \rightarrow \text{then not } y$$

MNWFs are uncovered through a review of MWFs, particularly an analysis of all possible logic states or truth tables associated with MWFs. A HAZOP study is also a good way to uncover MNWFs because of the inclusion of domain experts in the process. Due to the rigor with which they are developed, formal specifications are also a good source from which to identify MNWFs. Adequate time is usually not taken to specify MNWFs; the emphasis during design and development is on implementing, not inhibiting, functionality. Neglecting to specify MNWFs creates an opportunity for serious vulnerabilities.

While estimating the type, level, and extent of protection a system needs, it is useful to reassess the results of the entity control analysis. A careful review of the entity control analysis should ensure that there is no single point of failure. Entities over which the system owner has partial or no control should

**Exhibit 5 High-Level Identification of MWFs and MNWFs**

| *Example* | *MWFs* | *MNWFs* |
|---|---|---|
| Radiation therapy system | Functions responsible for controlling the accurate release of energy<br>Functions that verify which parameters for a treatment session are within known safe limits and are consistent with the treatment plan | Functions that would allow the erroneous release of radiation<br>Functions that would allow ongoing treatment plans or historical treatment records to be altered or deleted without authorization |
| Online banking system | Functions responsible for maintaining the confidentiality and integrity of transactions and data | Functions that would allow access to account information without authorization<br>Functions that would allow access control/authentication features to be bypassed |
| ATC system | Functions responsible for maintaining the integrity and availability of r/t signal location information | Functions that allow r/t location signal information to be altered, deleted, or delayed<br>Functions that would allow two controllers to direct multiple aircraft to the same runway or flight path at the same time |

be of particular concern, especially if one of these entities is essential to the correct operation of an IA-critical function or an MWF or the non-operation of an MNWF. In this case, the level of protection needed increases, sometimes dramatically. Contingency planning (see chapter section "Contingency Planning") should be undertaken to plan and prepare for the nonavailability or suboptimal performance of these resources. At the same time, alternative designs that employ redundancy, diversity, and fault tolerance should be evaluated. In the spring of 1998, a major communications satellite and its automatic backup failed for several hours. As reported by Garber,[269] 90 percent of U.S. pagers, or 35 to 40 million customers, some wireless ISPs, television and radio feeds, and credit card verification companies lost service and were caught without a backup strategy. This is a good example of what happens when the results of entity control analysis and contingency planning are ignored.

The time element should not be ignored either when determining the level of protection needed. There are two aspects of the time element to evaluate: (1) the time window during which the protection is needed, and (2) the time interval during which the proposed threat control measures will be effective. In general, different operational modes/states and profiles occur at different times during the day, during the week, on holidays, etc. The level of protection needed may or may not be the same for each. As a result, the type, level, and extent of protection needed for each of these scenarios should be determined. Failing to do so could cause a system to be over- or under-protected in a given situation.

A final factor to consider is privacy. As Morris[357] observes:

> *Secrecy and security are terms usually connoting national and specifically military interests. They have applications, however, far below the national level, ranging from commercial espionage to individual privacy.*

Privacy issues should be reexamined in light of the system design, operation, and operational environment to ensure that corporate or organizational assets, intellectual property rights of the organization and others, and information maintained about employees, customers, vendors, and others are adequately protected. IA goals, business ethics, laws and regulations, and societal norms will each contribute to the determination of what constitutes adequate privacy and the level of protection needed to guarantee it. In one situation, data privacy may be of utmost concern; in another, data integrity may be the driving factor.

In today's technological environment, no industrial sector or application domain is immune from privacy considerations; two of many possible examples follow. Wang, Lee, and Wang[436] have developed a taxonomy of privacy concerns related to e-Commerce, including:

- Improper access to consumer's private computer
- Improper collection of consumer's private information
- Improper monitoring of consumer's Internet activities without notice or authorization
- Improper analysis of consumer's private information
- Improper transfer of consumer's private information to a third party
- Sending unwanted solicitations
- Improper storage of consumer's private information

Rindfleisch[395] has identified privacy concerns related to medical records, to include:

- Insider threats
  - Accidental disclosure
  - Insider curiosity
  - Insider subornation
- Secondary user (quasi-insider) threats
  - Uncontrolled access/usage
- Outsider threats
  - Unauthorized access/usage

Through an analysis and synthesis of these six factors, the risk exposure reduction needed is identified and the estimated level of protection needed is refined. As a result, an IA integrity level is defined for the system or major system entities, as appropriate. IA integrity is a property reflecting the likelihood of a system, entity, or function achieving its required security, safety, and reliability features under all stated conditions within a stated measure of use.[130] An IA integrity level represents the level of IA integrity that must be achieved

or demonstrated to maintain the IA risk exposure at or below its acceptable level. Many national and international standards promote the concept of safety integrity levels (SILs) as part of the design, certification, and approval process.[31,38,57,63–65,124,129–130] This book expands that concept to the broader realm of IA. There are five levels of IA integrity, comparable to the five widely used SILs:

4 — Very high
3 — High
2 — Medium
1 — Low
0 — None

IA integrity levels are used to: (1) prioritize the distribution of IA resources so that resources are applied effectively and to the most critical need(s); and (2) to select appropriate threat control measures based on the type, level, and extent of protection needed. Depending on the system architecture, operation, and mission, one system entity may have an IA integrity level of 3 while another entity has an IA integrity of 1. Also, the IA integrity level for security, safety, and reliability functions may vary. For practicality of implementation and assessment, IA integrity levels should not be assigned to too low a level of a component.

While related, IA integrity levels should not be confused with EALs (discussed in Chapter 3). EALs, which do not measure safety or reliability, reflect confidence in security functionality. In contrast, IA integrity levels reflect confidence that a system will achieve and maintain required safety, security, and reliability features under all stated conditions so that the risk exposure is maintained at or below the target; that is, a measure of the robustness and resiliency of a system's IA features and the process(es) used to develop and verify them. This distinction is similar to the distinction between functional safety and safety integrity in IEC 61508[63–69] and other standards.

## 6.2 Evaluate Controllability, Operational Procedures, and In-Service Considerations

Threat control measures go beyond design features and techniques. Threat control measures encompass any aspect that could enhance the safe, secure, and reliable operation of a system. All entities, including people, are examined for opportunities to reduce risk exposure and improve system integrity. People are often cited as the weakest link in a system because often they: (1) are not aware of safety or security procedures; (2) do not understand the importance of following safety or security procedures, particularly the ramifications of not doing so; and (3) do not execute safety or security features, choosing instead to ignore, bypass, or disable them. At the same time, people have the potential to influence system integrity in a positive manner. As a result, the implementation of threat control measures necessitates an evaluation of additional parameters beyond system design and risk exposure. Specifically, controllability, operational procedures, and in-service considerations are evaluated.

Controllability is a measure of the ability of human action to control the situation following a failure. The concept of controllability originated with the automotive industry, where controllability was defined as "the ability of vehicle occupants to control the situation following a failure."[53] Five controllability categories have been defined by the automotive industry[53]:

**Uncontrollable**: human action has no effect
**Difficult to control**: potential for human action
**Debilitating**: sensible human response
**Distracting**: operational limitations, normal human response
**Nuisance**: safety (or security) not an issue

The five categories are mapped directly to the IA integrity levels (4–0). Jesty and Buckley[309] describe the relationship between controllability and integrity levels:

> *The controllability category for each hazard defines an integrity level required for the design of the new (sub)system, which in turn defines the requirements for the process of development.*

Exhibit 6 illustrates this relationship.

Controllability reflects the potential mitigating effect of human action subsequent to vulnerability exploitation and threat instantiation. In other words, can human action be taken to mitigate, contain, or preempt the unfolding consequences of a hazard, physical or cyber? Controllability is derived from both (1) the technical feasibility of taking a mitigating action, and (2) the time interval during which the mitigating action can be taken. Most hazardous situations do not instantly transition from a nonhazardous state to a catastrophic state; rather, there is a chain of events or ripple effect before reaching a catastrophic state. This interval, however small, represents the time when action can be taken to control a hazard or mitigate its consequences. It is important to note that the definition used by the automotive industry refers to the vehicle occupants and not just to the driver. Most likely, the ability to take a controlling action will vary, depending on the threat perspective. Therefore, it is useful to review transaction paths from different threat perspectives to determine controllability.

Most mission-critical systems are designed to be fault tolerant and to either fail safe/secure or fail operational. In both instances, the intent is to keep the system in a known safe/secure state at all times. These proactive design techniques enable a system to respond to one or more failures and hence protect itself. Controllability can be thought of as a human-assisted form of fault tolerance or failing safe/secure. As such, design provisions such as manual override, emergency shutdown, critical bypass, etc. should be implemented to facilitate controllability.

Operational procedures are a major component of threat control measures, although they are often overlooked as such. Operational procedures encompass the totality of IA concerns relative to the safe, secure, and reliable operation of a system in its operational environment. This includes many items

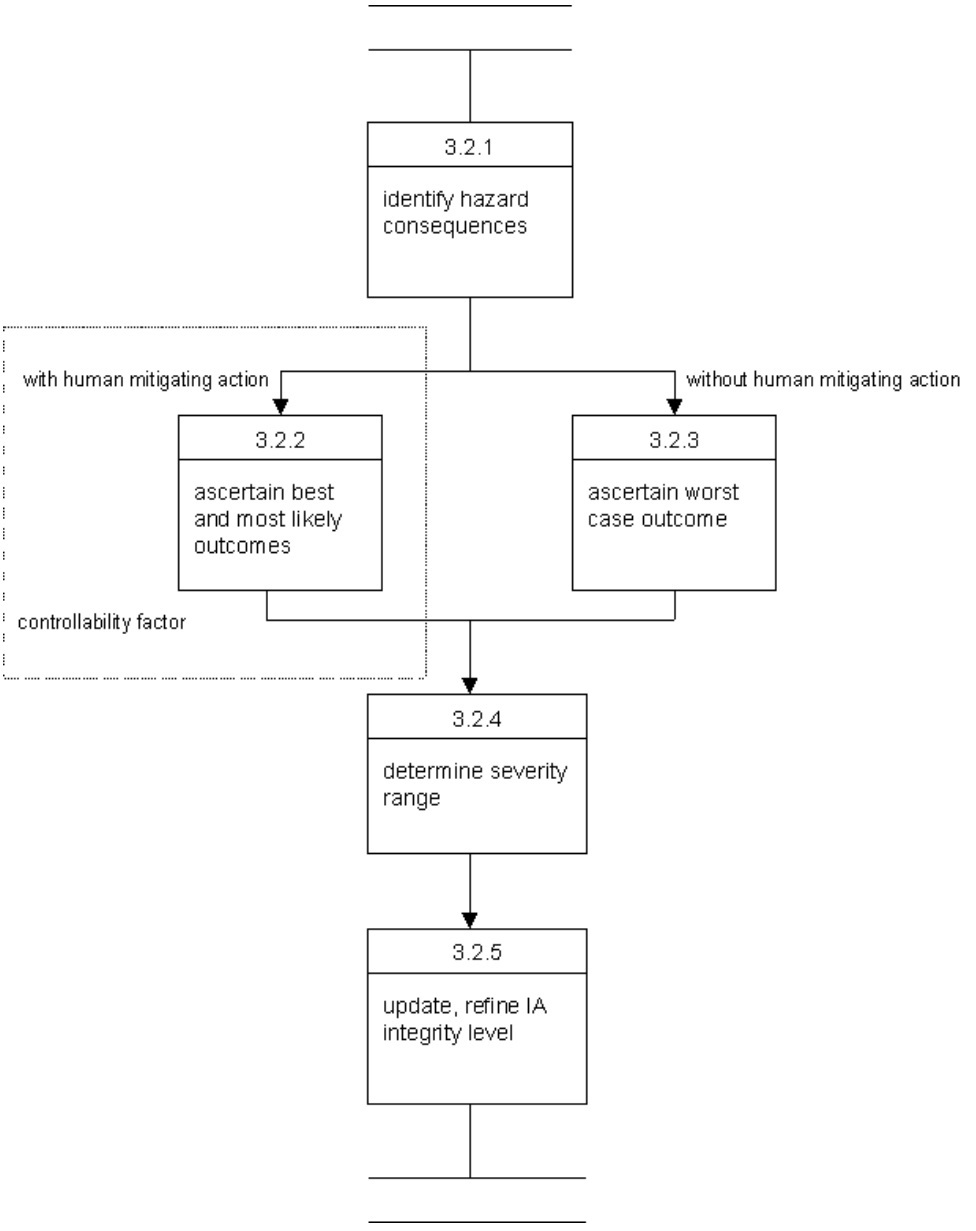


**Exhibit 6 Relationship Between Controllability and IA Integrity Levels**

traditionally associated with physical security and OPSEC, such as managing cryptographic keys and other security tokens. Safety and security features and procedures that lead to correct operation are described. Procedures are developed for each operational mode/state, including normal operations, abnormal operations, and recovery, and all operational profiles. If developed correctly and followed, operational procedures provide an opportunity to contribute to system integrity; the opposite is equally true.

Operational procedures are reviewed as part of the implementation of threat control measures to determine whether or not they are consistent with and support the IA goals, IA integrity level, and contingency plans; that is, do they treat the broader issues of safety, security, and reliability and not just functionality? Operational procedures are reviewed to ensure that they adequately address all issues related to personnel operations, software/data operations, and administrative operations identified in Chapter 3, Exhibit 10. For each of these issues, the following questions should be pursued:

1. Are the procedures consistent with the IA goals, IA integrity level, and contingency plans? Are all safety and security features and procedures explained?
2. Do the procedures conform to the current as-built system, or is an update needed? Do the procedures define the correct operational environment and any limitations or constraints imposed by it?
3. Are the procedures complete? Do they address all operational modes/states, missions, and profiles? Do they address the decommissioning of sensitive systems and disposal of sensitive information, including expired passwords and keys? Is enough detail provided? Is the information clear, concise, unambiguous, and accessible in a reasonable amount of time?
4. Have staff members been trained in how to follow the procedures?
5. Are the procedures being followed?

The IA challenge hardly disappears once a system is fielded. Consequently, in-service considerations should be evaluated when implementing threat control measures. In-service considerations, as they relate to threat control measures, take two forms: (1) system maintainability, and (2) the system usage profile.

Whetton[439] points out that:

> *… maintainability has an indirect effect on system safety [and security] in that any maintenance action must be done in such a way that it does not introduce new [vulnerabilities or] hazards into the system.*

Maintenance actions, hardware upgrades, software enhancements, new versions of COTS products, etc. can all potentially impact the IA integrity level of a system. As a result, systems — including their threat control measures — should be designed to be maintainable. There are two aspects to this: (1) designing a system so that its functionality, especially IA-critical and IA-related functions, can be maintained; and (2) designing a system so that it is maintainable without disrupting threat control measures. Highlighting requirements that are likely to change and reflecting this in the architecture through information hiding, partitioning, etc. promote the design of systems that are maintainable.[18] A change in operational environment, a change or addition to a system's mission, or extensive maintenance actions should trigger a reassessment/revalidation of the vulnerability and threat analyses.

A second in-service consideration is the system usage profile. A profile should be developed that defines anticipated low, normal, peak, and overload or saturation conditions. System loading may vary by operational mode/state, number of users, time of day, time of week, time of year (holidays), etc. Characteristics for each system load category should be defined and compared against known system constraints/capacity. The integrity of IA-critical functions, IA-related functions, and threat control measures should be verified under low, normal, and peak loading scenarios.

System overload often leads to unpredictable behavior. As a result, systems should be designed to prevent themselves from reaching this state; overload should be defined as an illegal state. Once peak loading has been reached, a monitor should be activated to ensure that the system does not transition from the peak loading threshold to overload. Before a critical situation is reached, protective measures should be invoked to block further user logons, e-mail, database queries, and other transactions, as appropriate. This is a simple technique for blocking denial-of-service attacks. Occasionally, low system loads can cause anomalous system behavior. This situation should also be investigated and remedied, if necessary.

## 6.3 Contingency Planning and Disaster Recovery

Planning for contingencies is an integral part of risk management in general and implementing threat control measures in particular. Webster's Dictionary defines a contingency as:

> (a) an event, such as an emergency, that is of possible but uncertain occurrence, (b) something liable to happen as an adjunct to something else, (c) something that happens by chance or is caused by circumstances not completely foreseen.

Contingency implies the notion of uncertainty, unforeseen events, and the unknown. Thus, contingency plans identify alternative strategies to be followed or action to be taken to ensure ongoing mission success should unknown, uncertain, or unforeseen events occur. Contingency plans provide planned measured responses to these events, in contrast to an unplanned workaround, the wrong response, or no response at all, in order to return a system to a known safe/secure state.

The activities described thus far in this book have been directed toward preventing, containing, and minimizing the likelihood and severity of system failures/compromises. Given that accidents are not 100 percent preventable, contingency planning provides an opportunity to prepare a measured response beforehand. Contingency planning allows a system owner to be prepared for the loss, unavailability, or anomalous performance of one or more system entities, whether or not the entity is internal (under their control) or external (not under their control). In addition, this planning takes place in an environment in which cool heads and logical thinking prevail, in contrast to panic responses during a crisis situation. The goal is to be prepared for any

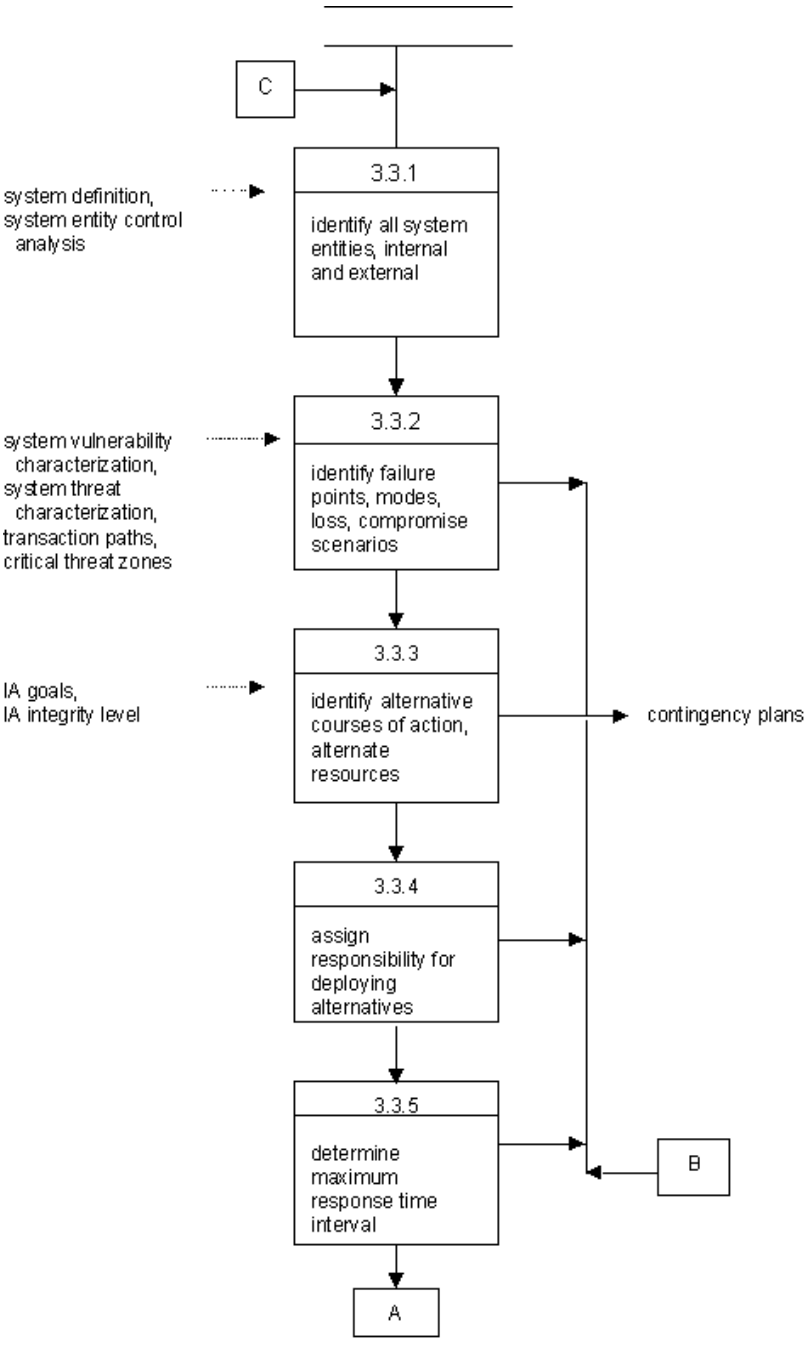


**Exhibit 7 Contingency Planning Process**

eventuality, and thereby ensure that there is no single point of failure that could lead to an inoperable or compromised system.

Exhibits 7 and 8 illustrate the contingency planning process. The first step is to identify all internal and external system entities and the degree of control the system owner has over each. This information is derived from the system

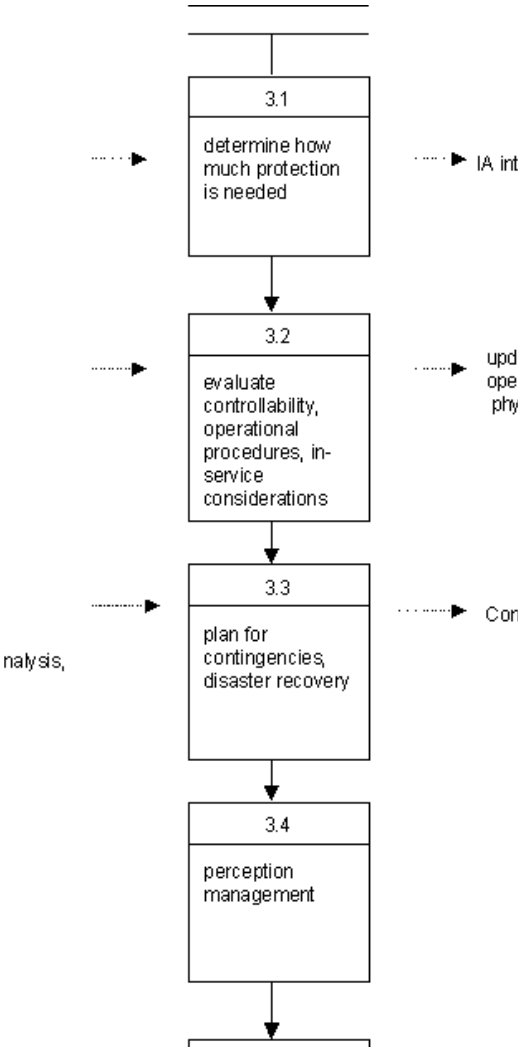


**Exhibit 8 Contingency Planning Process (continued)**

definition and entity control analysis (Chapter 4). It is important to be thorough when identifying entities and the dependencies between them. Exhibit 9 provides a checklist to review for this step.

The second step is to identify what could go wrong with a system and its entities: the failure points/modes and loss/compromise scenarios. This question is tackled from two angles[47]:

- **Cause and effect**: what could happen and what will ensue (cause consequence analysis)
- **Effects and causes**: what outcomes are to be avoided or encouraged and how each might occur*

* The movie *Frequency*, which was released in May 2000, is a good (fictional) example of this approach; historical events were changed to produce desired outcomes.

Vulnerability/threat characterizations, transaction paths, and critical threat zones (Chapter 5) are analyzed during this process. Particular attention is paid to IA-critical and IA-related entities/functions. Contingency planning assumes worst-case scenarios. For example, consider the ATC system in Chapter 5, Exhibit 16. At a high level, contingency plans should be made for the following scenarios:

- Loss of the radar system (no transmission or reception)
- Loss of voice communication between pilot and air traffic controller
- Loss of ATC DBMS
- Loss of ATC terminals
- Loss of location signal from aircraft (no transmission or reception)
- No air traffic controllers in the control tower

Consider another example closer to home: your neighborhood branch bank. In the last five years, the United States has experienced a wave of bank merger mania. My (what once was a) local bank has merged three times in three years, each time with a larger, more geographically dispersed bank. The merger requires, among other things, that the financial systems of the "old" bank be incorporated into those of the "new" bank and steps be taken to eliminate potential duplicate account numbers. Each time a merger has taken place, the new financial systems have been down two or more days. Given that financial systems are considered critical infrastructure systems, this is unacceptable. Obviously, (1) more robust contingency planning is needed so that transactions do not come to a halt following a merger; (2) an extended period of parallel operations is needed before switching to the new system; and (3) a capability to fall back to the old system is needed, should the new system prove unstable.

Once the various contingencies have been identified, an appropriate response for each is defined, consistent with the IA goals and IA integrity level. This involves formulating alternative courses of action and identifying alternative system resources. Priorities are established for restoring and maintaining critical functionality (degraded mode operations). The availability of alternative sources, services, and resources are specified. This may include redundant or diverse cold spare or hot standby systems/components, switching to a secondary operational site, relying on voice communications instead of data, etc. Exhibit 9 provides a generic list of alternatives to consider.

The fourth step is to assign responsibility for deploying the alternative course of action and resources. Next, the maximum time interval during which the responsive action can be invoked is defined. In almost all situations, there is a fixed time period during which a response is ameliorative; after that interval, the response has no effect or may even make the situation worse. In a crisis situation, it may not always be possible to respond in a timely manner. Hence, secondary courses of action/resources to invoke, if the maximum time interval for the primary response is exceeded, need to be identified.

With any plan, if a contingency plan is to be successful, it must be communicated and staff must be trained. Practice drills should be conducted regularly to both familiarize staff with the plan's provisions and to uncover

**Exhibit 9 Contingency Planning Checklist (partial)**

**Step 1: Identify all system entities, both internal and external.**

- Hardware components
- Communications equipment
- System software
- Applications software
- Services
  - Power
  - Environmental
  - Voice communications
  - Data communications
  - Facility concerns
- Archives
  - Electronic
  - Hardcopy
- People
  - Employees
  - Customers
  - System administrators
  - Maintenance technicians
  - Visitors
  - Trainers

**Step 3: Formulate alternative courses of action; identify alternate resources.**

- Activate cold spare
- Activate hot standby
- Reconfigure system
- Switch to degraded-mode operations, fail operational
- Emergency shutdown, logoff, fail safe/secure
- Restart system
- Restore system/data from local archives
- Restore system/data from offsite archives
- Switch operations to remote location
- Switch to alternate service provider
- Deploy emergency personnel
- Site/application-specific responses, actions, commands

any defects in the plan. Finally, contingency plans should be reviewed, updated, and revalidated at fixed intervals.

## 6.4 Perception Management

Perception management is a useful tool in many endeavors, including IA. Vendors have a vested interest in managing customers' expectations. Speakers have a vested interest in managing audience expectations. Likewise, system owners have a vested interest in managing the reality users perceive relative to the safe, secure, and reliable operation of a system.

Perception management serves multiple purposes. End users, whether customers of an online business or employees of an organization, gain confidence in a system and the results it produces if the system appears to be robust and provides accurate information quickly while protecting privacy. At the same time, this perception may serve as a deterrent to would-be attackers, both inside and outside the organization; the system is perceived as being extremely difficult to attack. However, one should not go overboard and attempt to give the impression that system safety/security is invincible — that may have the opposite effect by posing a challenge some attackers cannot resist. By the same token, a system should not appear too easy to attack.

Decoys are another perception management device. As Gollmann[277] points out:

> *Sometimes it is not sufficient to hide only the content of objects. Also, their existence may have to be hidden.*

In this case, it may be advisable to deploy decoy servers, decoy screens, decoy files/data, decoy passwords, etc.[248,277,305,375] Decoys can function as a benign security filter, an aggressive security trap that is meant to lure would-be attackers away from critical systems/data and catch them, or some combination thereof. Decoys, which must appear authentic or no one will be fooled, are also an effective method of blocking denial-of-service attacks.

## 6.5 Select/Implement IA Design Techniques and Features

Threat control measures are primarily implemented through design techniques and features, with operational procedures, contingency plans, and physical security practices being the other main contributing factors. Consequently, design techniques and features should be carefully chosen because of the pivotal role they play in achieving and maintaining IA integrity.

Threat control measures are selected based on the target risk exposure and the level of protection and IA integrity level needed. Controllability, in-service considerations, and perception management are also major determinants. As far back as 1979, DoD 5200.28-M[140] directed that security design techniques and features be chosen based on trade-off studies that evaluated risk analysis, the level of risk that could be tolerated, and cost. Particular threat control measures are chosen in response to specific vulnerabilities, hazards, and threats. Threat control measures represent a solution to a specific defined problem, the intent being to reduce the initial risk exposure to at or below the target. In summary, as Morris[357] notes, threat control measures should be implemented that are efficient, do not degrade system performance, and are appropriate for the level of risk exposure.

Exhibit 10 lists 25 current, proven IA design techniques and features. A description of each technique or feature is provided in Annex B, which describes the purpose, benefits, and limitations of each technique or feature and provides pointers to references for further information.

## Exhibit 10 IA Design Techniques and Features

| *IA Design Techniques and Features* | *C/R* | *Type* | *Life-Cycle Phase in which Technique is Used* | | |
|---|---|---|---|---|---|
| | | | *Concept* | *Development* | *Operations* |
| Access control:<br>Rights<br>Privileges | C2 | SA, SE | x | x | x |
| Account for all possible logic states | C2 | SA, SE | | x | x |
| Audit trail, security alarm | C2 | SE | x | x | x |
| Authentication:<br>Biometrics<br>Data origin<br>Digital certificates<br>Kerberos<br>Mutual<br>Peer entity<br>Smartcards<br>Unilateral | C2 | SA, SE | x | x | x |
| Block recovery | C2 | All | | x | x |
| Confinement:<br>DTE<br>Least privilege<br>Wrappers | C2 | SA, SE | | x | x |
| Defense in depth | C2 | All | x | x | x |
| Defensive programming | C2 | All | | x | x |
| Degraded-mode operations, graceful degradation | R2/C2 | All | | x | x |
| Digital signatures:<br>Nonrepudiation of origin<br>Nonrepudiation of receipt | C2 | SE | | x | x |
| Diversity<br>Hardware<br>Software | C2 | SA, SE | x | x | x |
| Encryption:<br>Asymmetric<br>Symmetric<br>Block<br>Stream<br>Hardware<br>Software | C2 | SE | x | x | x |
| Error detection, correction | C2 | ALL | | x | x |
| Fail safe/secure, fail operational | R2/C2 | SA, SE | | x | x |
| Fault tolerance | C2 | All | | x | x |
| Firewalls, filters | C2 | SA, SE | | x | x |
| Formal specifications, animated specifications | C2 | SA, SE | x | x | x |

**Exhibit 10 IA Design Techniques and Features (continued)**

| *IA Design Techniques and Features* | *C/R* | *Type* | *Life-Cycle Phase in which Technique is Used* | | |
|---|---|---|---|---|---|
| | | | *Concept* | *Development* | *Operations* |
| Information hiding | C2 | SA, SE | | x | x |
| Intrusion detection, response | C2 | SA, SE | | x | x |
| Partitioning:<br>Hardware<br>Software<br>Logical<br>Physical | C2 | SA, SE | x | x | x |
| Plausibility checks | C2 | All | | x | x |
| Redundancy | C2 | RE | x | x | x |
| Reliability allocation | C2 | RE | x | x | |
| Secure protocols:<br>IPSec, NLS<br>PEM, PGP, S/MIME<br>SET<br>SSL3, TLS1 | C2 | All | | x | x |
| Virus scanners | C2 | All | | | x |

*Source:* Adapted from Herrmann, D., *Software Safety and Reliability: Techniques, Approaches and Standards of Key Industrial Sectors*, IEEE Computer Society Press, 1999.

**Legend for the codes used in Exhibit 10:**

| *Column* | *Code* | *Meaning* |
|---|---|---|
| Type | SA | Technique primarily supports safety engineering |
| | SE | Technique primarily supports security engineering |
| | RE | Technique primarily supports reliability engineering |
| | All | Technique supports a combination of safety, security, and reliability engineering |
| C/R | Cx | Groups of complementary techniques |
| | Rx | Groups of redundant techniques; only one of the redundant techniques should be used |

Design techniques and features are a collection of methods by which a system (or component) is designed and capabilities are added to a system to enhance IA integrity. For custom software/systems, they represent techniques and features to employ when designing and developing a system. For COTS software/systems, they represent techniques and features to specify and evaluate during the product selection/procurement process. In the case of COTS software or systems, the EAL should be specified and verified. The delineation between the two categories (techniques and features) is not exact; hence, they will be considered together.

**Exhibit 11 Comparison of ISO OSI Information/Communications and TCP/IP Internet Reference Models**

| *TCP/IP Four-layer Internet Reference Model* | *ISO OSI Seven-layer Information/ Communications Reference Model* | *Sample Protocols* | *Functions Performed* | *Sample Primitives* |
|---|---|---|---|---|
| 4: Application layer | 7: Application layer | FTP, HTTP, SMTP, SNMP, Telnet, APIs | Execution of distributed applications | End-user data, files, queries, and responses |
| | 6: Presentation layer | Context management, syntax management (ASN) | Conversion of data between systems | Character sets, special characters, file formats |
| | 5: Session layer | | Session management, synchronization | |
| 3: Transport layer | 4: Transport layer | TCP, TP0-TP4, UDP | Reliable packet assembly, disassembly, sequencing; end-to-end integrity checks | Packets |
| 2: Internet layer | 3: Network layer | IP, X.25, ATM | Routing | Packets |
| 1: Network interface layer | 2: Data Link layer | IEEE 802.3, LAP-B,D Frame Relay | Transmission, framing, error control | Frames |
| — | 1: Physical layer | V.90, OC-3 SONET, RS-422 | Establish physical circuit, electrical or optical | Bits |

Exhibit 11 compares the International Organization for Standardization (ISO) Open Systems Interconnection (OSI) seven-layer Information/Communications reference model to the TCP/IP four-layer Internet reference model. Both models present a layered approach to specifying and achieving the reliable exchange and processing of information in distributed environments. The Internet model is more simplistic; it does not address physical connectivity, but it does merge session management, context management, syntax management, and application management into one layer. The table also identifies sample protocols, functions, and primitives associated with each layer.

These models are important in the IA domain because they highlight the need to deploy threat control measures at each layer. This need is often overlooked (by system owners, not necessarily attackers!) and all effort is (mistakenly) focused on protecting the application layer. To help organizations overcome this deficiency, the ISO and International Electrotechnical Commission

(IEC) have jointly published more than 50 security standards[70–123] based on the ISO OSI model. These standards address a variety of topics, applicable to different layers in the model, such as: access control, audit trails, authentication, digital signatures, block ciphers, hashing functions, key management, and security alarms. Readers are encouraged to consult these standards.

The physical layer is the bottom or first layer of the ISO OSI model. The function of this layer is to establish physical connectivity between two or more systems/components that want to communicate. This connection can be established by an electrical connection (V.90), optical connection (OC-3 SONET), a microwave link, a satellite feed, etc. The primitive associated with this layer is a stream of bits. Physical safety and security concerns are dealt with at this layer, along with concerns such as wiretapping, eavesdropping, EMI/RFI, and jamming.

The second layer is the data link layer, which is responsible for transmission, framing, and error control.[403] LANs employ data link layer protocols such as IEEE 802.3. The primitive associated with this layer is data frames. The data link layer shares many of the same safety, security, and reliability concerns as the physical layer.

The network layer is the third layer. This layer is responsible for routing data packets between networks. Protocols commonly used at this layer include IP, X.25, and ATM. In general, the system owner is responsible for the first two layers, while the third layer is part of the critical telecommunications infrastructure system or NII.

The transport layer is the fourth layer. This layer is responsible for ensuring reliable packet assembly, disassembly, and sequencing and performing end-to-end integrity checks. TCP and UDP are common protocols that are employed at this layer. In the TCP/IP model, the transport layer defines port information for layer 4 applications; for example, 21 - FTP, 23 - Telnet, 25 - SMTP, 80 - HTTP. The fifth layer performs session management between two communicating nodes, controlling when each can transmit and receive.

The sixth layer, the presentation layer, performs context management and syntax management, allowing communication between open systems. In the past, for remote systems to communicate, all parties had to use the same operating system, file format, character sets, etc. Today, the ability to send and receive e-mail and execute online applications is near-universal. All of this is made possible by presentation layer protocols.

The seventh or top layer is the application layer. This is the layer that controls and facilitates the distributed execution of applications. End users interact with applications at this layer. End-user data, files, and queries/responses are processed. FTP, SMTP, SNMP, and HTTP are common application layer protocols.

In summary, each layer: (1) serves a different purpose, (2) operates on a different unit of data; and (3) presents different IA challenges. All layers should be reflected in the entity control analysis. Different groups of people (end users, system administrators, software engineers, hardware engineers, communications engineers, …) and organizations (system owner, ISP, telecommunications company, hardware vendor, LAN vendor, …) interact with and are

**Exhibit 12 Assignment of Common Vulnerabilities and Threats to ISO OSI and TCP/IP Reference Model Layers**

| *Common Vulnerability/Threat*[a] | *ISO OSI Layer(s)* | *TCP/IP Layer(s)* |
|---|---|---|
| 1. Accidental action, command, response | 1–4, 7 | 1–4 |
| 2. Blocking access to system resources | 2–4, 7 | 1–4 |
| 3. Browsing | 7 | 4 |
| 4. Corruption of resource management information (accidental or intentional) | 2–7 | 1–4 |
| 5. Deletion of information or message (accidental or intentional) | 3, 4, 6, 7 | 2–4 |
| 6. Denial of service, network flooding, system saturation, lack of capacity planning | 2–4 | 1–3 |
| 7. EMI/RFI | 2, 3 | 1, 2 |
| 8. Environmental, facility, or power faults or tampering | 1 | — |
| 9. Illegal operations, transactions, modes/states | 2–4, 7 | 1–4 |
| 10. Inference, aggregation | 7 | 4 |
| 11. Insertion of bogus data, "man-in-the-middle" | 2–4, 7 | 1–4 |
| 12. Jamming | 2–4 | 1–3 |
| 13. Lack of contingency planning, backups | 1–7 | 1–4 |
| 14. Masquerade, IP spoofing | 3, 4, 7 | 2–4 |
| 15. Modification of information (accidental or intentional) | 2–4, 7 | 1–4 |
| 16. No fault tolerance, error detection or correction | 2–7 | 1–4 |
| 17. Overwriting information (accidental or intentional) | 6, 7 | 4 |
| 18. Password guessing, spoofing, compromise | 2–4, 7 | 1–4 |
| 19. Replay, reroute, misroute messages | 2–4 | 1–3 |
| 20. Repudiation of receipt, origin | 2–4, 7 | 1–4 |
| 21. Site/system/application-specific vulnerabilities and threats | 1–7 | 1–4 |
| 22. Theft of information, copying, distributing | 2–4, 7 | 1–4 |
| 23. Theft of service | 2–4, 7 | 1–4 |
| 24. Trojan horse | 4, 7 | 3, 4 |
| 25. Unauthorized access to system resources | 2–4, 7 | 1–4 |
| 26. Unauthorized use of system resources | 2, 3, 7 | 1, 2, 4 |
| 27. Uncontrolled, unprotected portable systems and media, archives, hardcopy | 2–4, 7 | 1–4 |
| 28. Unpredictable COTS behavior | 2–7 | 1–4 |
| 29. Virus attack | 7 | 4 |
| 30. Wiretapping, eavesdropping, leakage | 1–4 | 1–3 |

[a] *Sources:* Adapted from Denning, D., *Information Warfare amd Security*, Addison-Wesley, 1999; Denning D., *Cryptology and Data Security*, Addison-Wesley, 1982; Gollmann, D., *Computer Security*, John Wiley & Sons, 1999; Morris, D., *Introduction to Communication Command and Control Systems*, Pergamon Press, 1977; Rozenblit, M., *Security for Telecommunications Network Management*, IEEE, 1999.

responsible for the services provided by different layers. Often, these people and organizations are oblivious to layers other than their own. A clear understanding should be established about what needs to be/is/is not protected at each layer. A threat control measure deployed at layer x may provide some protection to the layers above it, but none to the layers below it. Any serious or organized attack will simply attack the weakest layer. Consequently, appropriate threat control measures need to be implemented at each layer.

Exhibit 12 lists 30 common vulnerabilities and threats. Each is assigned to the layer or layers in the ISO OSI and TCP/IP reference models in which it might appear. This knowledge is essential in selecting appropriate IA design techniques and features; it also underscores the need to provide threat control measures at all layers.

Exhibit 13 identifies the IA integrity function provided by each of the 25 IA design techniques and features. In addition, each technique/feature is assigned to the layer or layers in the ISO OSI and TCP/IP reference models in which it can be implemented. Using this knowledge, IA design techniques and features can be selected to eliminate or mitigate specific vulnerabilities/threats at particular layers in the model.

Next, the IA design techniques/features are examined in detail. There is a high degree of interaction and interdependence between the techniques/features; the output of one technique is the input to another technique and the techniques complement or reinforce each other.

## *Access Control*

Access control is a design feature that prevents unauthorized and unwarranted access to systems, applications, data, and other resources. Access control consists of two main components: (1) access control rights that define which people and processes can access which system resources; and (2) access control privileges that define what these people and processes can do with and to the resources accessed.[248] Examples of access control privileges include: read, write, edit, delete, execute, copy, print, move, forward, distribute, etc.

Access controls should be operative at all layers. For example, at the network layer, access control restrains access to one or more networks and the establishment of network sessions. This is similar to blocking outgoing or incoming telephone calls. At the application layer, access control restricts access to, and the execution of, systems, applications, data, and other shared resources. Access may be permanently denied, permanently granted, or granted conditionally based on some variable parameters.

Access control mechanisms are activated immediately after authentication. In simplest terms, an **initiator** (person or process) requests to perform an **operation** on a target **resource**. Access control mechanisms mediate these requests based on predefined access control rules. The initiator/resource combination reflects access control rights, while the initiator/operation combination reflects access control privileges. As noted by Rozenblit,[403] access control rules can be defined three ways:

**Exhibit 13 Assignment of IA Techniques and Features to ISO OSI and TCP/IP Reference Model Layers**

| *Technique/Feature* | *IA Integrity Function* | *ISO OSI Layer* | *TCP/IP Layer* |
|---|---|---|---|
| Access control:<br>Rights<br>Privileges | Protect IA-critical and IA-related systems, applications, data, and other resources by preventing unauthorized and unwarranted access. | 1: Physical<br>2: Data Link<br>3: Network<br>7: Application | 1: Network<br>2: Internet<br>4: Application |
| Account for all possible logic states | Prevent system from entering unknown or undefined states that could compromise IA integrity. | 7: Application | 4: Application |
| Audit trail, security alarm | Capture information about which people/processes accessed what system resources and when.<br>Capture information about system states and transitions; trigger alarms if necessary.<br>Develop normal system and user profiles for intrusion detection systems.<br>Reconstruct events during accident/incident investigation. | 2: Data link<br>3: Network<br>4: Transport<br>7: Application | 1: Network<br>2: Internet<br>3: Transport<br>4: Application |
| Authentication:<br>Biometrics<br>Data origin<br>Digital certificates<br>Kerberos<br>Mutual<br>Peer entity<br>Smartcards<br>Unilateral | Establish or prove the claimed identity of a user, process, or system. | 1: Physical<br>2: Data Link<br>3: Network<br>4: Transport<br>7: Application | 1: Network<br>2: Internet<br>3: Transport<br>4: Application |
| Block recovery | Enhance IA integrity by recovering from an error and transitioning the system to a known safe and secure state. | 7: Application | 4: Application |
| Confinement:<br>DTE<br>Least privilege<br>Wrappers | Restrict an untrusted program from accessing system resources and executing system processes. | 7: Application | 4: Application |

| | | | |
|---|---|---|---|
| Defense in depth | Provide several overlapping subsequent barriers with respect to one safety or security threshold, so that the threshold can only be surpassed if all barriers have failed.[a] | 1: Physical<br>2: Data Link<br>3: Network<br>4: Transport<br>7: Application | 1: Network<br>2: Internet<br>3: Transport<br>4: Application |
| Defensive programming | Prevent system failures and compromises by detecting errors in control flow, data flow, and data during execution and reacting in a predetermined and acceptable manner.[b] | 7: Application | 4: Application |
| Degraded-mode operations, graceful degradation | Ensure that critical system functionality is maintained in the presence of one or more failures.[b] | 3: Network<br>4: Transport<br>5: Session<br>6: Presentation | 2: Internet<br>3: Transport<br>4: Application |
| Digital signatures: | | 3: Network | 2: Internet |
| Nonrepudiation of origin | Provide reasonable evidence of the true sender of an electronic message or document. | 7: Application | 4: Application |
| Nonrepudiation of receipt | Provide reasonable evidence that an electronic message or document was received. | | |
| Diversity:<br>Hardware<br>Software | Enhance IA integrity by detecting and preventing systematic failures. | 1: Physical<br>2: Data link<br>3: Network<br>7: Application | 1: Network<br>2: Internet<br>4: Application |
| Encryption:<br>Asymmetric<br>Symmetric<br>Block<br>Stream<br>Hardware<br>Software | Provide confidentiality for information while it is stored or transmitted. | 2: Data Link<br>3: Network<br>4: Transport<br>7: Application | 1: Network<br>2: Internet<br>3: Transport<br>4: Application |
| Error detection, correction | Increase data integrity. | 2: Data Link<br>4: Transport<br>7: Application | 1: Network<br>3: Transport<br>4: Application |

**Exhibit 13 Assignment of IA Techniques and Features to ISO OSI and TCP/IP Reference Model Layers (continued)**

| *Technique/Feature* | *IA Integrity Function* | *ISO OSI Layer* | *TCP/IP Layer* |
|---|---|---|---|
| Fail safe/secure, fail operational | Ensure that a system remains in a known safe and secure state following an irrecoverable failure. | 3: Network<br>4: Transport<br>7: Application | 1: Network<br>2: Internet<br>3: Transport<br>4: Application |
| Fault tolerance | Provide continued correct execution in the presence of a limited number of hardware and/or software faults.[a] | 2: Data Link<br>3: Network<br>4: Transport<br>7: Application | 1: Network<br>2: Internet<br>3: Transport<br>4: Application |
| Firewalls, filters | Block unwanted users, processes, and data from entering a network while protecting legitimate users, sensitive data, and processes | 3: Network | 2: Internet |
| Formal specifications, animated specifications | Ensure correctness, consistency, completeness, and unambiguousness of the requirements and design for IA-critical and IA-related functions | 7: Application | 4: Application |
| Information hiding | Enhance IA integrity by: (1) preventing accidental access to or corruption of critical software and data, (2) minimizing introduction of errors during maintenance and enhancements, (3) reducing the likelihood of CCFs, and (4) minimizing fault propagation. | 7: Application | 4: Application |
| Intrusion detection, response | Recognize and respond to a security breach either as it is happening or immediately afterward; initiate appropriate response. | 3: Network<br>4: Transport<br>7: Application | 2: Internet<br>3: Transport<br>4: Application |
| Partitioning:<br>Hardware<br>Software<br>Logical<br>Physical | Enhance IA integrity by preventing IA-critical and IA-related functions/entities from being accidentally or intentionally corrupted by non-IA-related functions/entities. | 1: Physical<br>2: Data Link<br>3: Network<br>4: Transport<br>7: Application | 1: Network<br>2: Internet<br>3: Transport<br>4: Application |

| | | | |
|---|---|---|---|
| Plausibility checks | Enhance IA integrity by verifying the validity and legitimacy of critical parameters before acting upon them, detect faults early in the execution cycle and prevent them from progressing into system failures or compromises. | 7: Application | 4: Application |
| Redundancy | Enhance hardware reliability and system availability. | 1: Physical<br>2: Data Link<br>3: Network<br>4: Transport | 1: Network<br>2: Internet<br>3: Transport |
| Reliability allocation | Distribute reliability and maintainability requirements, derived from IA goals, among system entities. | 1: Physical<br>2: Data Link<br>3: Network<br>4: Transport<br>7: Application | 1: Network<br>2: Internet<br>3: Transport<br>4: Application |
| Secure protocols:<br>IPSec, NLS<br>PEM, PGP, S/MIME<br>SET<br>SSL3, TLS1 | Enhance the confidentiality of distributed data communications. | 3: Network<br>7: Application<br>7: Application<br>4: Transport | <br>2: Internet<br>4: Application<br>4: Application<br>3: Transport |
| Virus scanners | Automatically detect and remove computer viruses before they are activated. | 7: Application | 4: Application |

[a] See IEC 60880 (1986-09), Software for Computers in Safety Systems of Nuclear Power Stations.

[b] See IEC 61508-7 (2000-3) Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems — Part 7: Overview of Techniques and Measures.

1. Through the use of access control lists that specify the approved initiator(s) for each (group of) target(s)
2. Through the use of access capability lists that specify the target(s) accessible to a (group of) initiator(s)
3. Through the use of security labels, such that each initiator and target is assigned to one or more security label (confidential, secret, top secret, etc.), which in turn defines access control rights and privileges

(See also ISO/IEC 10164-9 and ISO/IEC 10181-3*.)

Exhibit 14 illustrates the three methods for specifying access control rules, using the set of initiators, operations, and resources listed below. Note that initiators can be individuals or user groups.

Often, it is easier to think in terms of the access control list. It is useful to develop the access control list first, and then rotate the matrix to develop the corresponding access capability list. This serves as a crosscheck to ensure that no unintended inferred access control privileges or information flow have been specified. For example, to execute application A, send/receive foreign e-mail, perform Internet searches, or import foreign files access to the organization's LAN/WAN is inferred. Because Henani has access to everything, that is alright. Malachi and Omri only have limited access; hence, design features must be employed to restrict them from other resources connected to the LAN/WAN to which they do not have explicit access control rights. The third option, security labels, is a variation of access control lists. Initiators are groups of people with a certain security clearance who have rights/privileges to access resources having the same or lower classification.

| *1. Initiators* | *2. Operations* | *3. Resources* |
|---|---|---|
| a. Malachi | a. Execute desktop office automation functions | a. Desktop PC |
| b. Omri | b. Send/receive local e-mail | b. Application server |
| c. Henani | c. Send/receive foreign e-mail | c. E-mail server, LAN/WAN |
| | d. Remote access | d. Web server, Internet |
| | e. Perform Internet searches | |
| | f. Import foreign files | |
| | g. Execute application A (limited)<br>■ View some data<br>■ Print some reports | |
| | h. Execute application A (full)<br>■ View all data<br>■ Enter new data<br>■ Delete data<br>■ Edit data<br>■ Copy files to desktop<br>■ Print reports | |

* ISO/IEC 10164-9(1995-12) Information Technology — Open Systems Interconnection — Systems Management: Objects and Attributes for Access Control.[92]
ISO/IEC 10181-3(1996-09) Information Technology — Open Systems Interconnection — Security Framework for Open Systems: Access Control Framework.[95]

**Exhibit 14 Comparison of Methods for Specifying Access Control Rules**

| *Initiators* | *Resources* | | | |
|---|---|---|---|---|
| | *Desktop PC, Printer* | *Application Server* | *E-mail Server, LAN/WAN* | *Web Server, Internet* |
| **A. Access Control List** | | | | |
| Malachi | 2.a: Execute desktop office automation application | None | 2.b: Send/ receive local e-mail | None |
| Omri | 2.a: Execute desktop office automation application | 2.g: Execute application A (limited) | 2.b: Send/ receive local e-mail | None |
| Henani | 2.a: Execute desktop office automation application | 2.h: Execute application A (full) | 2.b: Send/ receive local e-mail<br>2.d: Remote access | 2.c: Send/ receive foreign e-mail<br>2.e: Perform Internet searches<br>2.f: Import foreign files |
| **B. Access Capability List** | | | | |
| 2.a: Execute desktop office automation application | Malachi, Omri, Henani | — | — | — |
| 2.b: Send/receive local e-mail | — | — | Malachi, Omri, Henani | — |
| 2.c: Send/ receive foreign e-mail | — | — | [a] | Henani |
| 2.d: Remote access | — | — | Henani | — |
| 2.e: Perform Internet searches | — | — | [a] | Henani |
| 2.f: Import foreign files | — | — | [a] | Henani |
| 2.g: Execute application A (limited) | — | Omri | [a] | — |
| 2.h: Execute application A (full) | — | Henani | [a] | — |

Regardless of which method is used, adequate time must be taken to define access control rights and privileges to the appropriate level of detail. A default of "access denied" should be invoked if the system encounters an unknown or undefined state. Access control rules should be regularly

**Exhibit 14 Comparison of Methods for Specifying Access Control Rules (continued)**

| *Initiators* | *Resources* | | | |
|---|---|---|---|---|
| | *Desktop PC, Printer* | *Application Server* | *E-mail Server, LAN/WAN* | *Web Server, Internet* |
| **C. Security Label** | | | | |
| Confidential | 2.a: Execute desktop office automation application | None | 2.b: Send/ receive local e-mail | None |
| Secret | 2.a: Execute desktop office automation application | 2.g: Execute application A (limited) | 2.b: Send/ receive local e-mail | None |
| Top secret | 2.a: Execute desktop office automation application | 2.h: Execute application A (full) | 2.b: Send/ receive local e-mail,<br>2.d: Remote access | 2.c: Send/ receive foreign e-mail,<br>2.e: Perform Internet searches,<br>2.f: Import foreign files |

[a] Inferred right/privilege.

reviewed, updated, and revalidated. A capability should be implemented to provide emergency revocation of access control rights and privileges. Files defining access control rules must themselves be protected from unauthorized access and modification. An extension to defining access control rules is defining who has the right to update/modify the access control rules, in both normal and abnormal situations.

Specifying access control rights for data files can be complicated. Depending on the application and sensitivity of the information, access control rights can be specified at the field, record, or file level. If access control rights are not specified carefully, a vulnerability is created for aggregation and inference. Also, keep in mind that it is usually easier and less error-prone to (re)design data structures to accommodate a security architecture than to develop complex access control software.

A novel way of expressing access control rights is by time of access. To illustrate:

1. A user/process may be allowed to access certain system resources only at certain times during the day.
2. A user/process may only be allowed to access system resources during a specified time interval after their identity has been authenticated.

3. Time-sensitive information may only be accessed "not before" or "not after" specific dates and times.
4. E-mail, public keys, and other security tokens may have built-in (hidden) self-destruct dates and macros.[277]

This approach is particularly useful for information that is distributed outside the owner's system.

Due to the variety of resources being protected, access control mechanisms are implemented throughout a system. For example, server and network configuration files, username/password, groupname/password, file access privileges, default permissions, server log, server root access, etc. need to be protected as well as data files.[277,405] For some situations, the capabilities of commercial products are employed — using NT™ to define shared directories. In other situations, custom code is written to operate stand-alone or as an enhancement to a commercial capability.

Finally, physical access control issues, such as control of and accountability for portable systems and media, physical access to desktop PCs, servers, cable plant, shared printers, archives, and hardcopy output, should not be overlooked.

## *Account for All Possible Logic States*

One way to prevent a system from entering unknown or undefined states, and thus potentially unstable states, that could compromise IA integrity is to account for all possible logic states. This technique is based on the same concepts and rationale behind specifying MWFs and MNWFs. That is, for each critical decision point or command, all possible logic states that a system could encounter are defined. Truth tables are a straightforward method to uncover logic states. Once the logic states have been identified, an appropriate response (continue normal operations, trigger alarm, request further input/clarification, emergency shutdown, etc.) for each is defined. An extra level of safety and security is provided by implementing an OTHERWISE or default clause to trap exceptions and transient faults. This technique should be applied to all types of software: system software, applications software, firmware, etc. This technique is useful for uncovering missing and incomplete requirements, simple to implement, and of significant benefit in maintaining IA integrity. Exhibit 15 provides an illustration of how to account for and specify responses to all possible logic states. In this example, there are two parameters — temperature and pressure — that can be in any of three states: normal, too high, or too low.

## *Audit Trail, Security Alarm*

An audit trail/security alarm provides several IA integrity functions, including:

1. Capturing information about which people/processes accessed what system resources and when
2. Capturing information about system states and transitions and triggering alarms if necessary

## Exhibit 15 How to Account for All Possible Logic States

**Step 1: Determine all possible logic states:**

| *Parameter* | *Possible Logic States* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Temperature | 0 | – | + | 0 | – | + | 0 | – | + |
| Pressure | 0 | – | + | – | 0 | 0 | + | + | – |

**Step 2: Specify appropriate responses:**

```
Do case:
   case temperature = normal .and. pressure = normal
      do continue_normal_operations
   case (temperature = too low .and. pressure = too low) .or.
      (temperature = normal .and. pressure = too low) .or.
      (temperature = too low .and. pressure = normal)
         do trigger_warning
   case (temperature = normal .and. pressure = too high) .or.
      (temperature = too high .and. pressure = too low) .or.
      (temperature = too high .and. pressure = normal)
         do trigger_alert
   case (temperature = too high .and. pressure = too high) .or.
      (temperature = too low .and. pressure = too high)
         do activate_shutdown
```

**Step 3: Trap exceptions and transient faults:**

```
   Otherwise
      do trigger_alert
Endcase;
```

*Note:* 0: normal; –: too low; +: too high

3. Developing normal system and user profiles for intrusion detection systems
4. Providing information with which to reconstruct events during accident/incident investigation

Exhibit 16 illustrates the ways in which an audit trail contributes to IA integrity. (See also ISO/IEC 10164-7, ISO/IEC 10164-8, and ISO/IEC 10181-7*.)

An audit trail provides real-time and historical logs of system states, transitions, and resource usage. When a system compromise is expected, a security alarm is triggered. Alarm contents and primary and secondary recipients are defined during implementation. Potential components of a security alarm include[90,99,403]:

* ISO/IEC 10164-7(1992-05) Information Technology — Systems Management: Security Alarm Function.[90]
ISO/IEC 10164-8(1993-06) Information Technology — Systems Management: Audit Trail Function.[91]
ISO/IEC 10181-7(1996-08) Information Technology — Security framework for Open Systems: Security Audit and Alarm Framework.[99]

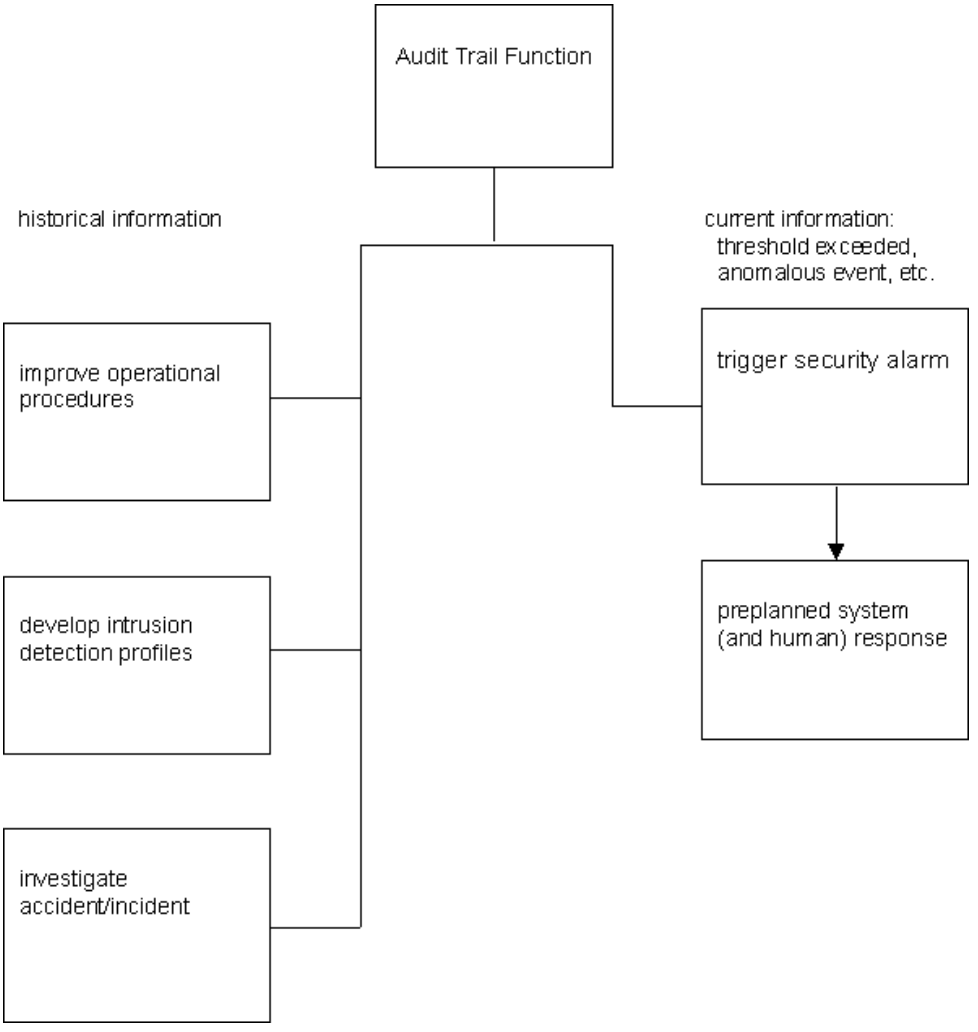


**Exhibit 16 Use of Audit Trail Data to Maintain and Improve IA Integrity**

- Identity of the resource experiencing the security event
- Date/timestamp of the security event
- Security event type (integrity violation, operational violation, physical violation, security feature violation, etc.)
- Parameters that triggered the alarm
- Security alarm severity (indeterminate, critical, major, minor, warning)
- Source that detected the event
- Service user who requested the service that led to the generation of the alarm
- Service provider that provided the service that led to the generation of the alarm

Audit trails are implemented at several layers in the ISO OSI and TCP/IP reference models. The completeness of the events/states recorded and the

timeliness in responding to the anomalous events determines the effectiveness of the audit trail. An audit trail consumes system resources; thus, care should be exercised when determining what events to record and how frequently they should be recorded. A determination also has to be made about the interval at which audit trails should be archived and overwritten. Another implementation issue is whether or not safety and security events should be recorded in a separate audit trail. Audit trails should be protected from unauthorized access to prevent: (1) masking current events that should trigger an alarm, and (2) analysis of historical information to facilitate a masquerade attack. The analysis of historical audit trail information, particularly information collated from several different audit trails, can help identify persistent, previously undetected, low-level attacks and improvements needed in operational procedures.[403]

## *Authentication*

Authentication is a design feature that permits the claimed identity of a user, process, or system to be proven to and confirmed by a second party. Accurate authentication is an essential first layer of protection, upon which access control, audit trail, and intrusion detection functions depend. Authentication may take place at several levels: logging onto an NT™ desktop, e-mail, a bank ATM, a specific application system, etc. In each instance, a user is required to identify him- or herself and prove it through some supporting evidence. A username and supposedly secret password are provided in most cases. There is movement toward the use of more sophisticated parameters because of the vulnerabilities associated with using just usernames and passwords. For example, browsers store previous pages, including usernames and passwords.[277] As several sources conclude, common sense dictates that a combination of factors should be used to authenticate a user or process, such as[260,277,344]:

- individual username/passwords
- User group or category
- Security level, token, PIN
- Time of day
- Terminal ID or location
- Network node, traffic source
- Transaction type
- Biometric information

Fegghi, Fegghi, and Williams[260] point out that, when choosing authentication parameters, consideration should be given to what information is supplied, what information is derived, and what information can be faked.

There are several authentication methods: unilateral, mutual, digital certificates, Kerberos, data origin, peer entity, smartcards, and biometrics. These methods are used for different purposes and at different layers in the ISO OSI and TCP/IP reference models.

Authentication can be unilateral or mutual. When a user logs onto a system, the user is authenticated to the system but the system is not authenticated to the user. In many situations, particularly e-Commerce, it is highly desirable to have mutual authentication in which both parties (users, processes, or systems) are authenticated to each other before any transactions take place. A challenge-response protocol is commonly used to perform mutual authentication. The basic exchange is as follows[403]:

1. x sends an association establishment request, plus a unique string to y.
2. y encrypts the string and sends it back to x along with a new unique string.
3. x decrypts the string, verifies that it is the string sent, then encrypts the second string and sends it to y.
4. y decrypts the message and verifies that it is the string sent.

This protocol makes use of public key encryption and requires a minimum of three message exchanges.[403] The association request can be aborted at any time if a discrepancy is detected.

Digital certificates are used to authenticate the distribution of public keys, software, and other material by binding a public key to a unique identifier. Trusted certificate authorities (CAs) issue digital certificates. The format of digital certificates has been standardized since June 1996 through CCITT X.509 version 3[50,260,403]:

a. X.509 version (1, 2, or 3)
b. Certificate serial number assigned by CA
c. Algorithm used to generate the certificate signature (k)
d. CA name
e. Certificate validity period (start/end dates)
f. Subject name (unique individual or entity)
g. Subject public key information (public key, parameters, algorithm)
h. Optional issuer unique identifiers
i. Optional subject unique identifiers
j. Optional extensions
k. CA digital signature of preceding fields

Certificates can revoked before they expire; hence, it is prudent to check current certificate revocation lists (CRLs) maintained by trusted CAs. Also, digital certificates should be bound to a specific request to prevent replay.[403] It is important to remember that digital certificates guarantee the source; they do not guarantee that software is virus-free or will operate safely, securely, and reliably.[277]

Kerberos, which provides trusted third-party authentication for TCP/IP networks,[409] is an authentication product. It supports unilateral and mutual authentication, primarily user to host, and provides a reasonable degree of confidentiality. Kerberos utilizes tickets as its basic security token. Kerberos

is primarily available as a shareware product, although some commercially supported products are beginning to emerge.[433,453,466,478] Kerberos is under consideration to become an Internet standard.[182] At present, NT™ 5.0 uses Kerberos for authentication.[277]

Data origin authentication ensures that messages received are indeed from the claimed sender and not an intruder who hijacked the session.[260,403] Data origin authentication is initiated after an association setup is established and may be applied to all or selective messages.[403]

Peer entity authentication provides mutual application to application authentication. As Rozenblit[403] reports:

> *It provides a (usually successful) second line of defense against intruders who have successfully bypassed connection access control.*

Smartcards are a physical security token that a user presents during the authentication process. They represent an evolution of ATM or credit cards with magnetic strips, in that they contain a limited amount of processing power. Smartcards are currently used to access mobile phone networks, store electronic funds, and perform debit/credit card transactions. In the near future, they may replace employee badges for authentication purposes: entry into secure office spaces, desktop logon, etc.

Chadwick[239] cites the advantages and disadvantages of smartcards:

- **Advantages:**
  - Increased security: private key is unlikely to be copied unless the smartcard is stolen and the thief knows the password and PIN
  - Potential mobility of users; however, mobility is dependent on the availability of smartcard readers
  - Sequential access to one desktop PC or other machine by multiple users

- **Disadvantages:**
  - Cost: which may improve over time
  - Slower performance: 5 to 100 percent slower during message signing and encryption
  - Interoperability problems associated with new technology

As reported by Garber,[271] American Express, Visa, Banksys, and ERG Systems have formed a joint venture called ProtonWorld International to develop an open standard for smartcards, and hence solve the interoperability issues worldwide. Their immediate goal is to define the common electronic purse specifications (CEPS) that will standardize interfaces, electronic cash formats, and security mechanisms, such as public keys. Standardization will also help to minimize smartcard fraud. See www.protonworld.com[486] for the latest information about standardization efforts.

Biometric systems are one of the newest modes of authentication. In simplest terms, a biometric system is a pattern recognition system that establishes the

authenticity of a specific physiological or behavioral characteristic possessed by a user.[374] A biometric system has two major components: (1) a high-resolution scanner that acquires and digitizes information, and (2) computer hardware and software that perform the pattern recognition. The two major functions of a biometric system are enrollment and identification. Enrollment, which involves registering biometric information with a known identity, consists of three steps[427]:

1. Capturing a raw biometric data sample from a scanner
2. Processing the raw biometric data to extract the unique details
3. Storing a composite of raw and unique data with an identifier

Identification involves comparing a current biometric sample to known stored samples to verify a claimed identity or to identify a person.[266,427] Identification repeats the capture and processing steps. Then, pattern recognition algorithms are invoked to perform the comparison. Current and planned future applications of biometric identification include access to secure facilities, access to desktop PCs, verification for receipt of welfare payments, verification for home banking privileges, and verification for bank ATM access.

Nine types of biometric systems are currently in use or under development. Each measures a different physical characteristic[266,332]: fingerprints, iris, retina, face, hand, ear, body odor, voice, and signature.

Fingerprint scanning is the oldest technology. Automated fingerprint identification standards began appearing in 1988*. Fingerprint scanning offers 40 degrees of freedom. Lerner[332] reports that the false positive rate, which can be improved if two fingers are scanned, is ~1 percent. Fingerprint scanning may replace passwords in the near future for access to desktop computers.

The algorithm for iris scanning was developed in 1980 by John Daugman of the Cambridge University Computer Science Department. It is only recently, given improvements in computer processing power and cost, that iris scanning technology has become commercially viable. IrisScan, Inc., of New Jersey currently holds the patent for this technology. In simplest terms, iris scanning involves wavelet analysis on a 512-byte pattern, similar to Fourier analysis. As reported by Lerner,[332] iris scanners support 266 degrees of freedom and can perform 100,000 scans per second.

Biometric authentication can also be performed through voice verification. In this case, the enrollment process consists of extracting unique feature vectors from a passphrase that is recorded and stored in a voice print database. Biometric authentication through voice verification is geared toward reducing fraud in three environments: e-Commerce over the Internet, t-Commerce over

* ANSI/IAI 1-1988, Forensic Identification — Automated Fingerprint Identification Systems — Benchmark Tests of Relative Performance.[16]
ANSI/IAI 2-1988, Forensic Identification — Automated Fingerprint Identification Systems — Glossary of Terms and Acronyms.[17]
ANSI/NIST-CSL 1-1993, Information Systems — Data Format for the Interchange of Fingerprint Information.[15]

fixed-line telephones, and m-Commerce over mobile/wireless devices. An advantage of voice verification is that it requires "little or no additional infrastructure investment due to the wide availability and low cost of computer microphones and telephones, whether fixed-line or cellular."[282]

Configate's[463] Verimote is a leading voice verification product. As reported by Gordon[282]:

> *… they have developed patent-pending software and adaptive algorithms that use 40 objective voice features and 40 subjective features to confirm the identity of an individual by verifying the unique aspects of his voice.… Subsequent comparisons of the feature vectors have yielded 99.6 percent accuracy.*

Verimote is language independent; unlike some products, it is not tied to English. In addition, the level of security is selectable. For example, a user may be required to repeat randomly selected numbers to further reduce the likelihood of a false positive.

A drop in the cost of biometric systems has expanded their usage. This is turn has given impetus to biometric standardization efforts. As reported by Tilton,[427] the BioAPI consortium, consisting of some 45 companies, is developing a multi-level, platform-independent architecture to support the enrollment and identification functions. The architecture specification is being written in an object-oriented notation. This architecture envisions a biometric service provider (BSP) that will operate between the input device and specific API. A draft standard was released for comment December 1999. At the same time, the biometric data format, common biometric exchange file format (CBEFF), is being defined by the U.S. Biometric consortium and ANSI X.9F4 subcommittee. For the latest information on biometric standards, see www.ibia.org[474] and www.biometrics.org.[459]

The use of biometric identification systems raises performance and privacy issues. While biometric systems are considered more accurate than nonbiometric systems,[266] they still raise concerns about false positives and false negatives given the variability in biometric characteristics.[374] For example, changes in makeup, hair style or color, tinted contact lenses, plastic surgery, a suntan, and presence or absence of a mustache or beard would all change facial characteristics, as would an illness or the normal aging process. Also, what is to prevent a person from placing a photograph or hologram in front of the scanner, or in the case of a voice recognition system, playing a tape recording? The accuracy of biometric systems is, not surprisingly, tied to cost. Some experts think that multi-mode biometric identification may be more accurate than single mode[266]; however this has not yet been proven. The integrity of stored data samples and current data samples is another concern. Biometric data is not immune to misuse and attacks anymore than other types of data.[332] Enrollment fraud is a major concern for the system owners and the individual whose identity has been hijacked.[393] Likewise, the privacy and confidentiality of biometric data must be protected. Until standardization efforts take hold, system integration and interoperability issues will remain. In summary, biometric

identification systems are expected to reduce fraud, forgery, and theft[332]; but like other authentication methods, they are not a panacea.

## Block Recovery

Block recovery is a design technique that provides correct functional operation in the presence of one or more errors.[69] Block recovery is implemented to increase the integrity of modules that perform critical functions. Exhibit 17 illustrates the basic logic of block recovery.

For each critical module, a primary and a secondary module (employing diversity) are developed. After the primary module executes, but before it performs any critical transactions, an acceptance test is run. This test checks for possible error conditions, such as runtime errors, excessive execution time, or mathematical errors, and performs plausibility checks.[422] If no error is detected, normal execution continues. If an error is detected, control is switched to the corresponding secondary module and another acceptance test is run. If no error is detected, normal execution resumes. However, if an error is detected, the system is reset either to a previous (backward block recovery) or future (forward block recovery) known safe/secure state.

If the system is reset backward, internal states have to be saved at well-defined checkpoints and some compensatory action must be taken to account for events that took place after the state to which the system is being reset.[69,333,422] Forward block recovery is appropriate for real-time systems with fast changing internal states and a small amount of data.[69] After the system has been reset, normal operation continues. Jajodia[305] recommends implementing forward block recovery for anticipated errors and backward block recovery for unanticipated errors.

## Confinement

Confinement is a design feature that restricts an untrusted program from accessing system resources and executing system processes. The intent is to prevent an untrusted program from exhibiting unknown and unauthorized behavior, such as:

- Accidentally or intentionally corrupting data
- Accidentally or intentionally triggering the execution of critical sequences
- Initiating a trapdoor or Trojan horse through which executables are misused or corrupted
- Opening a covert channel through which sensitive data is misappropriated

Noninterference is the goal of confinement — preventing interference between independent functions that utilize shared resources and unintended intercomponent communication.[255] Interference can lead to[255]:

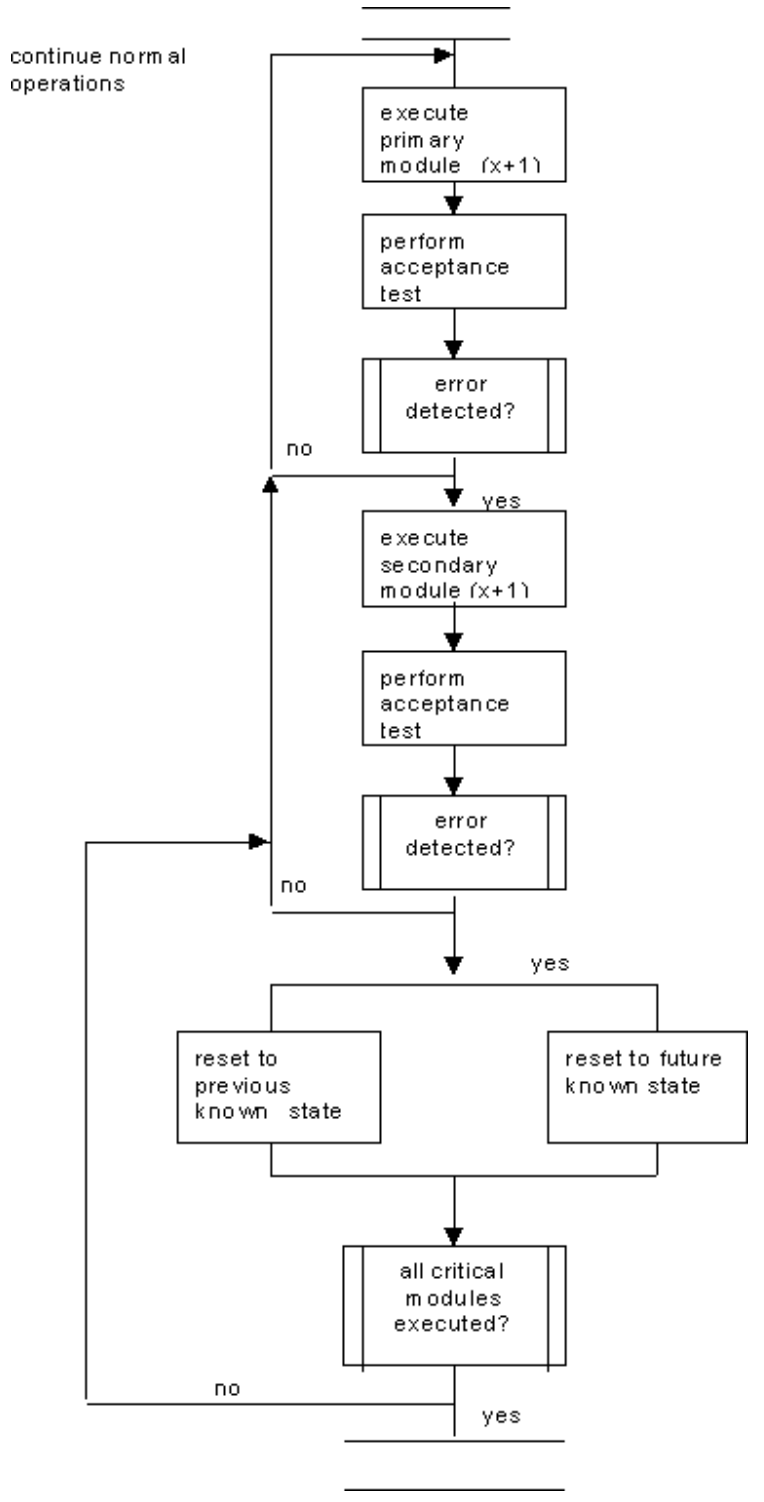


**Exhibit 17 Illustration of Block Recovery Logic**

- **Data corruption.** Untrusted components overwrite vital information stored in common memory and used by trusted components.
- **Denial of access to critical resources.** Untrusted components can prevent or delay the execution of critical functions by restricting or preventing access to a shared resource. In particular, untrusted components can use too much CPU time or can fail to terminate or crash and hence prevent the trusted components from executing.

Noninterference and separability are used in the information security domain much the same as partitioning and isolation are used in the computer safety domain.

COTS software, mobile code,[405,406] reused software, shareware, and the active content of Web pages are all good candidates for confinement. Confinement can be implemented by:

1. Restricting a process to reading data it has written[277]
2. Limiting executable privileges to the minimum needed to perform its function, referred to as least privilege or sandboxing; for example, child processes do not inherit the privileges of the parent processes[335,405]
3. Mandatory access controls (MAC)
4. Domain and type enforcement (DTE)
5. Language-based confinement of mobile code, per the Java™ security model[335]
6. Wrappers

DTE is a confinement technique in which an attribute called a domain is associated with each subject (user or process), and another attribute called a type is associated with each object (system resource). A matrix is defined that specifies whether or not a particular mode of access to objects of type x is granted to subjects in domain y.[335]

Gollmann[277] gives examples of language-based confinement:

- Applets do not get to access the user's file system
- Applets cannot obtain information about the user's name, e-mail address, machine configuration, etc.
- Applets may make outward connections only back to the server they came from
- Applets can only pop-up windows that are marked "untrusted"
- Applets cannot reconfigure the system, for example by creating a new class loader or a new security manager

Wrappers are a confinement technique that encapsulates datagrams to control invocation and add access control and monitoring functions.[277] They were originally developed for use with secure protocols, such as the encapsulated payload in IPSec or NLS. In the case of IPSec and NLS, the wrapper

is used to protect what is encapsulated from the outside world. In the case of confinement, wrappers are used to protect the outside word from what is encapsulated (and untrusted). Since then, their usage has expanded. Fraser, Badger, and Feldman[264] recommend using wrappers to confine COTS software:

> *Using appropriate activation criteria, wrappers can be applied selectively to points of vulnerability, or can be globally applied to all processes on a system. In either case, with respect to a wrapped program, the mediation and additional functionality provided by a wrapper is nonbypassable and protected from tampering. These characteristics, in addition to the fine-grained control that wrappers may provide by potentially processing every system call, give wrappers a great deal of power to add to and enforce security policies.*

Mobile code is also a good candidate for confinement. Sander and Tschudin[406] cite several concerns about mobile code that confinement can help to mitigate:

- Protecting host computers from accidental or intentional errant mobile code
- Protecting mobile agent's code and data from accidental or intentional tampering, corruption, and privacy violations
- Secure routing of mobile code
- Protecting mobile code from I/O analysis

To be effective, all six confinement techniques require thorough upfront analysis to determine how to restrict the untrusted program and to what to restrict it.

## *Defense in Depth*

Defense in depth provides several overlapping subsequent limiting barriers with respect to one safety or security threshold, so that the threshold can only be surpassed if all barriers have failed.[60] The concept of defense in depth, as applied to systems, originated in the nuclear power industry. Defense in depth should be applied to all layers in the ISO OSI and TCP/IP reference models.

Defense in depth is a design technique that reflects common sense. In short, everything feasible is done to prepare for known potential hazards and vulnerabilities. Then, acknowledging that it is impossible to anticipate all hazards and vulnerabilities, especially unusual combinations or sequences of events, extra layers of safety and security features are implemented through multiple complementary design techniques and features. For example, partitioning, information hiding, plausibility checks, and block recovery could be implemented; four layers of protection are better than one. Exhibit 18 illustrates the concept of defense in depth. In this example, each of the six successive defensive layers would have to be surpassed for a compromise to occur. Note that robust access control and authentication are the foundation of this defense strategy.

Defensive Layers

| |
|---|
| Layer 6<br>t: intrusion detection<br>r: attempted intrusions are detected, preempted, and contained |
| Layer 5<br>t: audit trail, security alarm<br>r: anomalies are detected as they occur |
| Layer 4.<br>t: secure protocols<br>r: if obtained, data is unintelligible, if altered, data is rejected |
| Layer 3.<br>t: encryption<br>r: if obtained, data is unintelligible |
| Layer 2.<br>t: access control<br>r: unknown or unauthorized users, processes cannot access data or systems |
| Layer 1.<br>t: user, peer entity, and data origin authentication<br>r: only known authorized users and processes are allowed on the system |
| Layer 0.<br>t: physical security measures<br>r: only known authorized personnel are allowed entry to facilities, equipment rooms, archives, cable plant; portable equipment and media are controlled. |

t - IA design technique/feature
r - result

**Exhibit 18 Illustration of Defense in Depth**

## Defensive Programming

Defensive programming prevents system failures or compromises that could affect IA integrity by detecting errors in control flow, data flow, and data during execution and reacting in a predetermined and acceptable manner.[69] Defensive programming is applied to all IA-critical and IA-related functions. Defensive programming is approached from two directions. First, potential software design errors are compensated for; this is accomplished by: (1) performing range, plausibility, and dimension checks at procedure entry and before executing critical commands, and (2) separating read-only and read-write parameters to prevent overwriting. Second, failures in the operational environment are anticipated; this is accomplished by: (1) performing control flow sequence checks to detect anomalous behavior, especially in state transitions; (2) regularly verifying the hardware and software configuration; and (3) conducting plausibility checks on critical input, intermediate, and output variables before acting upon them. In summary, all actions and transitions are verified beforehand as a preventive strategy.

## Degraded-Mode Operations, Graceful Degradation

The purpose of degraded-mode operations — or graceful degradation as it is sometimes called — is to ensure that critical system functionality is maintained in the presence of one or more failures.[69] IA-critical and IA-related functions can rarely just cease operation in response to an anomalous event, suspected attack, or compromise. Rather, some minimum level of service must be maintained. Degraded-mode operations allows priorities to be established for maintaining critical functions, while dropping less critical ones, should insufficient resources be available to support them all. The total system (hardware, software, and communications equipment) is considered when planning for degraded-mode operations; often, a system reconfiguration is necessary.[422] Degraded-mode operations is tied directly to and consistent with operational procedures and contingency plans.

The prioritized set of IA-critical and IA-related functions should be specified during the requirements and design phases. Criteria for transitioning to degraded-mode operations is specified. The maximum time interval during which a system is allowed to remain in degraded-mode operations is also defined. Degraded-mode operations should include provisions for the following items, at a minimum[126,127]:

- Notifying operational staff and users that the system has transitioned to degraded-mode operations
- Error handling
- Logging and generation of warning messages
- Reduction of processing load (execute only core functionality)
- Masking nonessential interrupts
- Signals to external resources to slow down inputs
- Trace of system states to facilitate post-event analysis
- Specification of the conditions required to return to normal operations

Degraded-mode operations provides an intermediate state between full operation and system shutdown; hence the name graceful degradation. This allows the minimum priority system functionality to be maintained until corrective action can be taken. Degraded mode operations is another preventive strategy where decisions are made beforehand about how to respond to a potential crisis situation. Without this prior planning and preparation, the ensuing system degradation and compromise will be most ungraceful indeed.

## Digital Signatures

Digital signatures provide reasonable evidence of the true sender of an electronic message or document, which is often referred to as nonrepudiation of origin. Digital signatures are created using public key encryption, like RSA. A signature generation algorithm and a signature verification algorithm are involved. The initial digital signature standard (DSS) was issued in May 1994.[165]

A digital signature consists of a fixed-length string of bits that is derived from the original message using public key encryption. This string of bits is attached to the original message before it is sent. In general, a digital signature is generated on the clear text message, the message is encrypted, and then the signature is attached and the message is transmitted.[403] The recipient decrypts the string to verify that the signature reflects the original message. In this way, the recipient knows (1) the message has not been altered, and (2) the real origin of the message. Nonrepudiation of receipt requires the recipient to sign the message and return it to the sender. This provides roundtrip message delivery confirmation.

Digital signatures help to establish the identity of a sender of a document. However, they do not necessarily prove that the sender created the contents of the message.[248] Digital signatures consume additional system resources and require that a reliable key management process be followed.

The use of digital signatures will grow as e-Commerce grows. Legislation passed in the United States during the summer of 2000 will also accelerate this process. As reported by Shuman[413]:

> *At the end of June, U.S. President Bill Clinton signed the Electronic Signature Act. This was no ordinary signature. The President used a smartcard encrypted with his digital signature to "e-sign" the legislation. The new law officially grants electronic digital signatures legal status in court.*

## Diversity

Diversity is a design technique employed to enhance IA integrity by detecting and preventing systematic failures. While diversity does not prevent specification errors, it is useful for uncovering specification ambiguities.[422] Diversity can be implemented in hardware and software.

In software diversity, also referred to as n-version programming, more than one algorithm is developed to solve the same problem. The same input is

supplied to the n versions, and then the outputs are compared. If they agree, the appropriate action is taken. Depending on the criticality of the application, 100 percent agreement or majority agreement can be implemented[288]; if the results do not agree, error detection and recovery algorithms take control. Diverse software may execute in parallel on different processors or sequentially on the same processor. The first approach increases hardware size, cost, and weight, while the second approach increases processing time.[422] Diversity can be implemented at several stages in the life cycle[129]:

- Development of diverse designs by independent teams
- Development of diverse source code in two or more different languages
- Generation of diverse object code by two or more different compilers
- Implementation of diverse object code using two or more different linking and loading utilities

Hardware diversity employs multiple, different components and modules to perform the same function. This contrasts with hardware redundancy in which multiple units of the same hardware are deployed. To the extent possible, components and modules are chosen that have different rates and types of failures.

The goal of diversity is to decrease the probability of common cause and systematic failures, while increasing the probability of detecting errors.[69] Diversity may complicate supportability issues and synchronization between diverse components operating in parallel.[129] Accordingly, diversity should only be implemented for IA-critical and IA-related functions.

## *Encryption*

Encryption is an IA design feature that provides confidentiality for data while it is stored or transmitted. This is accomplished by manipulating a string of data (clear text) according to a specific algorithm to produce cipher text, which is unintelligible to all but the intended recipients. As shown in Exhibit 19, a series of decisions must be made when implementing encryption.

The first question to answer is: What data needs to be encrypted? This information is derived from the IA goals and may include items such as e-mail, application-specific data, authentication data, private keys, and video teleconference sessions. At the same time, data that does not need to be encrypted is identified; for example, certain fields in a database record may need to be encrypted but not the entire record or file. Because of the time and resources utilized, it is important not to over identify data needing encryption. Conversely, it is equally important not to under identify data needing encryption and in so doing create opportunities for aggregation and inference.

The second question to answer is: Where is the data created, stored, and transmitted? The intent is to uncover all instances of this data so that the most efficient and effective encryption strategy can be employed. The stored and transmitted categories should include printouts, local and organizational hard-copy archives, and electronic archives; there is no point in encrypting active electronic data if printouts and archives are unprotected.

**Exhibit 19 Key Decisions to Make when Implementing Encryption**

1. What data needs to be encrypted?
   a. E-mail
   b. Text, graphic, video, audio files
   c. Database files
   d. Application data
   e. Telephone conversations and voice mail
   f. Fax transmissions
   g. Video teleconferences
   h. Authentication data files
   i. Private keys
   j. Access control rights and privileges
2. Where is the data:
   a. Created? b. Stored? c. Transmitted?
3. What strength of encryption is needed?
   a. Low b. Medium c. High d. Very high
4. At what level(s) in the ISO OSI or TCP/IP reference models should encryption take place?
   a. Physical b. Data link c. Network d. Transport e. Application
5. Should hardware or software encryption be used?
6. Should block or stream ciphers be used?
7. What cipher mode of operation should be used?
   a. ECB b. CBC c. OFB d. CFB e. Counter
8. Should symmetric or asymmetric keys be used?
9. What key management procedures should be used for:
   a. Key generation
   b. Key distribution
   c. Key verification
   d. Controlling key use
   e. Responding to key compromise
   f. Key change
   g. Key storage
   h. Key recovery, backup
   i. Destroying old keys
10. What encryption algorithm should be used?

*Note:* Questions 3 through 10 are repeated for each collection of data identified in Questions 1 and 2.

Third, the encryption strength needed is determined; this can vary from low to medium for random office e-mail traffic, to very high for defense or intelligence applications. This information is derived from the IA goals.

Fourth, the level or levels of the ISO OSI and TCP/IP reference models in which to implement encryption are identified. Exhibit 20 depicts potential encryption points in a typical information architecture. As shown, encryption can be implemented at the physical, data link, network, transport, and application layers.

Data link level encryption encrypts all traffic on a single point-to-point link. It is easy to implement because of well-defined hardware interfaces.[409] Data link level encryption is transparent to higher level applications and provides protection against traffic analysis.[241,409] However, data is exposed at network nodes because it must be decrypted to obtain routing information.[241,409]

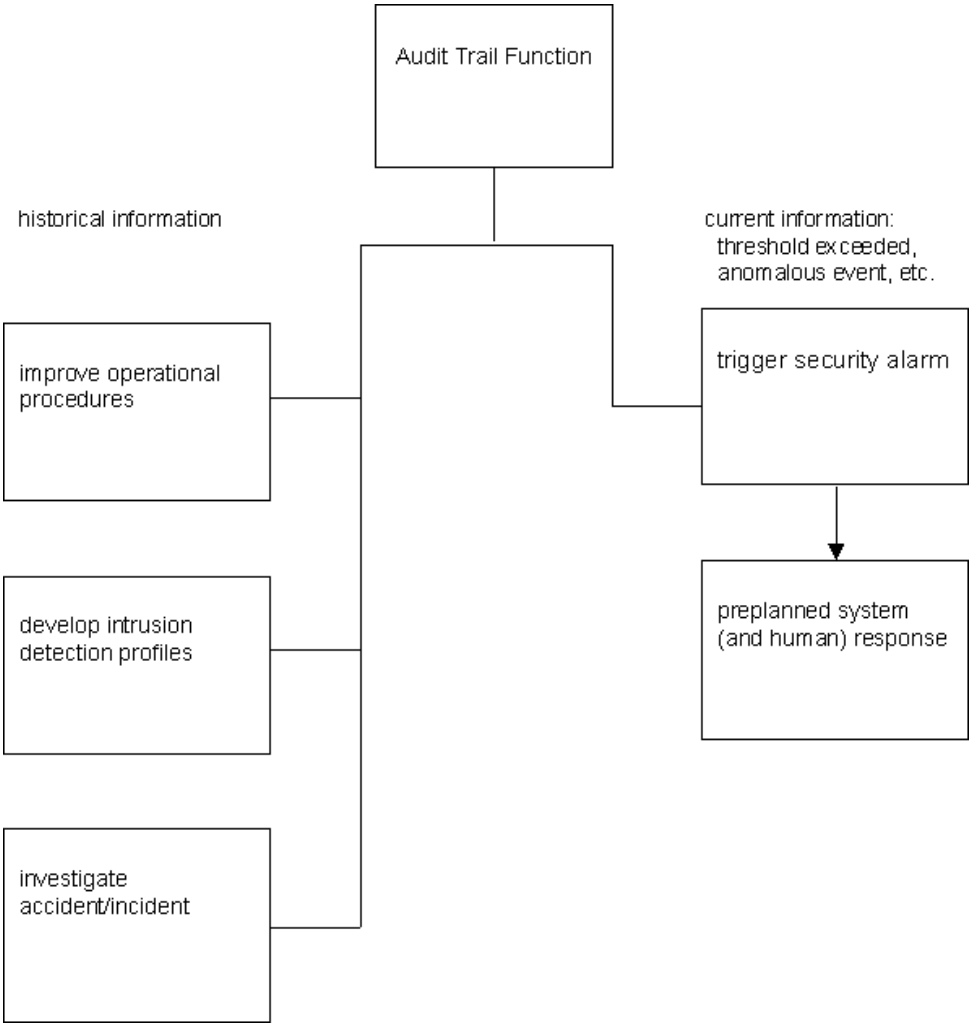


**Exhibit 20 Potential Encryption Points in a Typical Information Architecture**

Network and transport level encryption utilize a key ID that explains the encryption algorithm, blocksize, integrity check, and validity period. ATM encryption standards are under development.[426] Transport level encryption is implemented using the transport level security (TLS1) protocol, which makes use of TCP virtual circuits. This permits different circuits between the same pair of hosts to be protected with different keys.[241] TLS1 encrypts the TCP header and segment, but not the IP header. Network level encryption is implemented using either IPSec or the network level security (NLS) protocol, which make use of encapsulation. IPSec and NLS encrypt entire packets, including the original IP header, and generate a new IP header. IPSec, NLS, and TLS1 are transparent to higher level applications. IPSec and NLS protect subnets from traffic analysis,[241] while TLS1 does not. Key management is

**Legend for Exhibit 20**

| *Encryption Point* | *Level of Encryption* | *Type of Encryption* |
|---|---|---|
| a. Keyboard to CPU | Data link | Hardware |
| b. CPU to portable media | Data link | Hardware |
| c. Local workstation hard drive | Application | Hardware or software |
| d. Local workstation to LAN | Network | Hardware |
| e. Local workstation to shared printer/smart copier | Network | Hardware |
| f. Electronic archives | Application | Software |
| g. Application server | Application | Software |
| h. Database management system | Application | Software |
| i. E-mail server | Transport | Software |
| j. Internet server | Transport | Hardware |
| k. Telephone conversations | Data link | Hardware |
| l. Voice mail storage | Data link | Hardware |
| m. Fax transmissions | Data link | Hardware |
| n. Remote access to LAN | Data link | Hardware |

more complex than data link level encryption.[409] (See also the discussion of secure protocols.)

Application layer encryption can be implemented in a variety of ways. For example, a custom application can encrypt data stored on a server or in a database. Financial data in spreadsheets can be encrypted on local workstations. Corporate personnel files can be encrypted. Perhaps the best-known instance of application level encryption is e-mail. Several e-mail encryption protocols are available, including PEM, PGP, and S/MIME. In addition to encryption, some of these protocols support digital signatures and digital certificates. A common vulnerability of all application layer security is that the data can be attacked through the operating system before it is encrypted. Another concern is when encryption takes place relative to the browser function, because browsers store previous Web pages.[277] Accordingly, it is beneficial to employ encryption at multiple levels.

To supplement application level encryption, data can also be encrypted while it is stored on a local workstation, on an application server, in backup files, archives, and other portable media. As mentioned, the clear text stores of this information must be controlled as well or the encryption will be to no avail.

The fifth decision is whether hardware or software encryption should be used, inasmuch as encryption algorithms can be implemented in either. Hardware encryption is the primary choice for critical applications and is used almost exclusively by the defense and intelligence communities.[409] This is true for several reasons. Hardware encryption provides algorithm and to some extent key security because the units are designed to: (1) be tamperproof, (2) erase keys if tampering is attempted, and (3) eliminate emanations through shielding.[409] Hardware encryption is considerably faster than software encryption and it offloads intensive calculations from the CPU, thus improving overall system

performance. Hardware encryption is implemented in modules, boards, and boxes that are easy to install. Software encryption, while easy to use and upgrade, presents several vulnerabilities not found in hardware encryption. For example, the software encryption task runs the risk of being preempted by a higher priority task or interrupt and being written to disk. This leaves both the key and the data exposed.[409] Likewise, software encryption algorithms are vulnerable to unauthorized and potentially undetected alterations. Key management is also more complex with software encryption.

The sixth decision is whether block or stream ciphers should be used. Block ciphers operate on a fixed number of bits or bytes; if necessary, the last block is padded. Both the cipher text and clear text have the same block size. Block ciphers can be implemented in hardware or software. Stream ciphers operate on asynchronous bit streams, transforming a single bit or byte of data at a time. Stream ciphers are implemented in hardware at the data link level.

The next decision concerns the mode the block or stream cipher will operate in — its mode of operation. Some modes are only applicable to block ciphers, while others work for both block and stream ciphers. The differences between the modes are, for the most part, subtle. A notable difference is the extent to which errors are propagated. The five most common modes are: electronic code book (ECB), cipher block chaining (CBC), output feedback (OFB), cipher feedback (CFB), and counter. ECB mode produces the same results from the same block of data each time. This feature is convenient and simplifies verification, but facilitates cryptoanalysis.[241] In CBC mode, each block of clear text is exclusive OR'd with the previous block of cipher text before encryption. Initialization vectors are supplied for the first block of data. Block ciphers or stream ciphers can operate in OFB mode. In this mode, n-bits of the previous cipher text are exclusive OR'd with the clear text, starting at the right-most bit. This mode has the advantage that errors are not propagated. In CFB mode, the left-most n-bits of the last cipher text block are exclusive OR'd with the first/next n-bits of the clear text. Unfortunately, this mode propagates errors. In counter mode, sequence numbers (rather than previous cipher text) are used as input to the encryption algorithm. The counter is increased by a constant value after each block is encrypted. Block ciphers or stream ciphers can operate in counter mode. Errors are not propagated.

The choice of encryption key type is next. Symmetric or asymmetric keys can be used. When symmetric or secret keys are used, the same key is used for encryption and decryption. The use of symmetric keys is the traditional approach to encryption and was used in the past primarily for defense and intelligence applications. Symmetric keys are appropriate in the following situations:

- The sender and receiver are known to each other and are in the same organization or cooperating organizations.
- The sender and receiver remain constant for a fixed period of time.
- The sending and receiving nodes remain constant for a fixed period of time.

- A long-term relationship between the sender and receiver is anticipated, with a regular need to exchange sensitive information.
- The sender and receiver have the ability to cooperate on key management and other encryption issues.

In contrast, with asymmetric keys, a pair of public and private keys are used. The public key (used for encryption) is shared, while the private key (used for decryption) is not. The two keys are mathematically related but it is not feasible to uncover the private key from the public key. In practice, when A wants to send B a sensitive message, A encrypts the message with B's public key. Then, B decrypts the message with their private key. The first asymmetric key cryptosystems were announced in the late 1970s. Since then, several other systems have been developed. Asymmetric key cryptosystems are considerably slower and use much longer key lengths than symmetric key systems. As Schneier[409] points out, symmetric and asymmetric key systems are designed for different operational profiles:

> *Symmetric cryptography is best for encrypting data. It is orders of magnitude faster and is not susceptible to chosen cipher text attacks. Public key cryptography can do things symmetric cryptography can't; it is best for key management and a myriad of other protocols [digital signatures, key exchange and authentication, digital cash, and so forth].*

Most organizations employ a combination of symmetric and asymmetric key systems. As Lee[330] observes:

> *In practice, the symmetric key and public key systems are not in competition. Most cryptographic schemes on which e-Commerce operations rely use a hybrid of the two systems to exploit the key management flexibility of a public key system and the fast scrambling speeds of symmetric key systems. This hybrid approach is often called key wrapping.*

Key management issues are the next logical decision. Given that most encryption algorithms are publicly available, it is the keys that must be protected. The extent to which a key is protected should be proportional to the sensitivity of the information being encrypted. Several issues must be decided when developing key management plans and procedures; these include:

- Algorithm to use to generate the key
- Process and schedule for changing keys
- How to distribute keys securely
- How to store keys securely, both local and backup copies
- How to verify the authenticity of keys
- Process for recovering "lost" keys

- Process for controlling and revoking keys
- Process for destroying all instances of old keys
- Process for responding to the compromise of a secret or private key

These policies and procedures need to be established by an organization, with the involvement of all stakeholders, prior to implementing encryption. All staff should be thoroughly trained to use the procedures. Periodic audits should be conducted to verify that the procedures are being followed and to look for opportunities to improve them.

The final decision to be made concerns which encryption algorithm to use. Of course, several of the decisions made above will narrow this choice. The strength of an encryption algorithm depends in part on the sophistication of the algorithm and the length of the key.[241,403,409] However, as both Schneier[410] and Ritter[396] point out, longer key lengths by themselves do not necessarily guarantee more security. Rozenblit[403] notes that "the amount of computation it takes, on average, to uncover the secret key increases exponentially with the size of the key." Most sources recommend changing symmetric keys at least once a day[241,403,409]; in fact, in very critical applications, separate keys may be used for each session.[241] Likewise, it is recommended that asymmetric systems employ timestamps to prevent replay.[403]

In addition, there are several implementation details to consider. The processing efficiency of the algorithm, in terms of time and resources used, is a major factor. Efficiency is improved if the encryption blocksize is consistent with the data bus size. Files should be compressed before they are encrypted, while error detection/correction codes should be added after encryption.[409] In situations where confidentiality is particularly important, it may be beneficial to implement multiple or cascade encryption to further inhibit opportunities for cryptoanalysis. In multiple encryption, an algorithm is repeated several times on the same block of data using multiple keys; triple DES is a well-known example of this. In cascade encryption, multiple different algorithms are performed on the same block of data. Finally, while encrypting e-mail increases privacy for the sender and receiver, it potentially decreases system and data integrity for the receiver because, as Garber[272] notes, many commercial anti-virus products have difficulty scanning encrypted e-mail effectively.

## *Error Detection/Correction*

Error detection/correction algorithms are used to increase data integrity during the transmission of data within and among networks and system integrity during the execution of application software. At the network level, error detection/correction algorithms examine data to determine if any data was "accidentally" corrupted or lost, and to discover if any unauthorized changes were "intentionally" made to the data.[357] These errors are compensated for by self-correcting codes at the receiving end or requests for retransmission. At the application software level, error detection/correction algorithms detect anomalous or illegal modes/states, parameters, etc., and initiate the appropriate error handling routines.

Examples of common error detection/correction algorithms used at the network level include longitudal and vertical parity checks, Hamming codes, convolutional codes, recurrent codes, checksums, and cyclical redundancy checks (CRCs).[357] The receiving end performs the error detection and notifies the transmitting end. In response, the transmitting end, in most cases, provides a reason for the error along with an indication of whether the condition is temporary or permanent.[334] If the error condition is expected to be prolonged, the receiving end may request that a new session be established. Morris[357] has identified several factors to consider when selecting which error detection/correction algorithms to implement:

- Type of data to be transmitted
- Degree of accuracy required in received data
- Inherent level of reliability required
- Number (or percent) of incorrect digits or messages that are allowed through
- Delays allowed in the system
- Accepted redundancy of the data allowed (the volume of the data transmitted versus the accuracy required)
- Required throughput in the system (throughput is reduced by redundancy, coding delays, and requests for retransmission)
- Type of links available and the interferences that may be anticipated in the links
- Efficiency of the data transmission or communication links
- Implementation cost and the cost efficiency of the various error control techniques

At the application software level, this technique involves: (1) identifying where possible errors could occur in accessing, manipulating, and relaying information; and (2) defining the appropriate corrective action to be taken in each instance. It is unlikely that error detection/correction will be implemented for all potential error conditions due to program size, response time, schedule, and budget constraints; hence, the focus should be on IA-critical and IA-related functions/entities. Error conditions that might result from accidentally and intentionally induced anomalies, as well as potential transient faults, should be included in the analysis.

## *Fail Safe/Secure, Fail Operational*

Fail safe/secure and fail operational are IA design techniques which ensure that a system remains in a known safe/secure state following an irrecoverable failure. To fail safe/secure means that a component automatically places itself in a known safe/secure mode/state in the event of a failure. In many instances, known safe default values are assumed. Then the system is brought to a known safe/secure mode/state by shutting it down; for example, the shutdown of a nuclear reactor by a monitoring and protection system.[345] To fail operational

means that a system or component continues to provide limited critical functionality in the event of a failure. In some instances, a system cannot simply shut down; it must continue to provide some level of service if it is not to be hazardous, such as an aircraft flight control system.[345]

Fail safe/secure and fail operational ensure that a system responds predictably to failures by making proactive design decisions. The first step is to identify all possible failure modes. This is done by developing transaction paths and using IA analysis techniques such as FTA, FMECA, and HAZOP studies. Next, the appropriate response to each failure is specified so that the system will remain in a known safe/secure state. Examples of different types of fail safe/secure or fail operational modes include[288,333,422]:

- Fail operational by transitioning to degraded mode operations
- Fail safe/secure by assuming known safe/secure default values and then activating an emergency shutdown
- Fail operational by transferring to manual or external control
- Fail operational by activating a restart
- Fail safe/secure by activating a hold state, whereby no functionality is provided while action is taken to maintain the system in a safe/secure state and minimize the extent of damage

The correct response to one failure may be to fail safe/secure, while the correct response to another failure in the same system may be to fail operational. Also, the operational mode/state will influence the choice of a failure response.

Fail safe/secure and fail operational designs should be implemented for all IA-critical and IA-related functions/entities at the hardware, software, and system levels. This is essential for maintaining IA integrity. Fault tolerance prevents a limited number of faults from progressing to failures. Those that cannot or are not expected to be handled sufficiently by fault tolerance must be dealt with by fail safe/secure and fail operational designs. Combining fault tolerance and fail safe/secure and fail operational designs is another example of defense in depth.

## *Fault Tolerance*

Fault tolerance increases IA integrity by providing continued correct execution in the presence of a limited number of hardware or software faults.[60,225] As Jajodia[305] notes:

> *Fault tolerance is a natural approach for dealing with information attacks because it is designed to address system loss, compromise, and damage during operation.*

Fault tolerance is a category of IA design techniques that focuses on containing and mitigating the consequences of faults, rather than preventing them. It is important to clarify the terminology used in this regard:

- An **error** is the difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.[44]
- A **fault** is a defect that results in an incorrect step, process, data value, or mode/state.
- A **failure** is failing to or to inability of a system, entity, or component to perform its required function(s), according to specified performance criteria, due to one or more fault conditions.
- A **mistake** is an erroneous human action (accidental or intentional) that produces a fault condition.

Either an error or a mistake can cause a fault, which can lead to a failure:

$$\text{Error or mistake} \rightarrow \text{Fault} \rightarrow \text{Failure}$$

Fault tolerance attempts to prevent a fault from progressing to a failure, which could compromise a system or render it inoperable. Faults provide an opening for possible attacks, especially if the fault condition can be induced.

There are three types of fault tolerance: hardware, software, and system. As Levi and Agrawala[334] point out, hardware faults are generated by design errors (overload, improper states, etc.) and environmental stresses, which cause physical degradation of materials, while software faults are caused by design errors and runtime errors. However, they note that[334]:

> *One cannot always distinguish hardware from software failures. …hardware failures can produce identical faulty behavior to that generated by software. Memory failures are equivalent to software failures if they occur during instruction fetch cycles of the processors, generating an erroneous execution of an instruction. A processor whose program counter is inappropriately altered produces an out of order execution of instructions as does a software design error.*

As a result, a combination of hardware, software, and system fault tolerance is needed.

Hardware fault tolerance is usually implemented through redundancy, diversity, power-on tests, BITE, and other monitoring functions. The concept is that if a primary component fails, the secondary component will take over and continue normal operations.

Software fault tolerance is usually implemented through block recovery, diversity, error detection/correction, and other IA design techniques. The basic premise of software fault tolerance is that it is nearly impossible to develop software that is 100 percent defect-free; therefore, IA design techniques should be employed to detect and recover from errors while minimizing their consequences. Software fault tolerance should be implemented in both application and system software. In September 1999, Enea OSE Systems of Sweden announced the first fault tolerant, real-time operating system to be IEC 61508 certified. This product, OSE, offers a "full featured

real-time operating system for fault tolerant high-end systems" and supports the "direct message-passing method of interprocess communication, memory protection, and error detection/correction."[415]

System fault tolerance combines hardware and software fault tolerance, with software monitoring the health of both the hardware and the software. System fault tolerance should be employed for IA-critical and IA-related functions. Fault tolerance is an effective method to increase system reliability and availability. It may increase the physical size and weight of a system, which can conflict with specified constraints.

## *Firewalls, Filters*

A firewall is a security gateway between one network and another that uses a variety of techniques to block unwanted users and processes from entering a network while protecting legitimate users, sensitive data, and processes, in accordance with IA goals. Firewalls control access to resources between one network and another. They determine if a particular message or process should be allowed to enter or exit a system[248] by monitoring both incoming and outgoing traffic. Firewalls perform several critical functions, similar to caller-ID and call-blocking on a telephone, which help to maintain IA integrity, including[248,277]:

- Access control based on sender or receiver addresses
- Access control based on the service requested
- Hiding internal network topology, addresses, and traffic from the outside world
- Virus checking incoming files
- Authentication based on traffic source
- Logging Internet activities for future analysis
- Blocking incoming junk e-mail
- Blocking outgoing connections to objectionable Web sites

There are three main types of firewalls: packet filters, application level gateways, and circuit level gateways. Packet filters determine if a packet should be let into a network based on the communication endpoint identifiers.[253] Depending on the source or destination address or port, packets may be purposely dropped. Filtering rules (block or allow) are specified based on source, port, destination, port, and flags. Packets not explicitly allowed by a filter rule are rejected. This, of course, assumes that filtering rules were created with tight specifications. Packet filters can be configured to examine both incoming and outgoing packets. Packet filters are essentially ineffective when UDP is used.[241]

Application level gateways are, in essence, firewalls that are developed to protect specific applications. The most common usage is with e-mail servers. A separate application level gateway is required for each service or application. Usually, all traffic is logged by an application level gateway.

Circuit level gateways, sometimes called proxy firewalls, mediate between two devices attempting to communicate across a firewall[253]; they relay TCP connections. In practice, a caller connects to a port on the outside gateway. If the session is approved, the gateway forwards the information to the destination that is connected to an internal gateway port. Circuit level gateways log the number of bytes sent by TCP address. They are not effective against insider attacks.[241] Because of the different services provided, most organizations use a combination of firewall types.

Several issues arise during the implementation of firewalls. One of the first is how to configure the firewall; the answer, representing a trade-off between security and connectivity, is derived from the IA goals and IA integrity level. Firewalls primarily provide protection at the lower levels of the ISO OSI and TCP/IP reference models. They should be evaluated against known vulnerabilities, specific protection profiles, and content-based attacks.[251] Many commercial firewall products are available. To facilitate the selection process, the Computer Security Institute (CSI)[470] publishes an annual report that compares these products. A persistent concern has been the ability to provide firewall-like protection for home PCs, especially those operating home-based businesses. A remedy for this situation was recently announced. Groner[281] reports that a scaled-down version of CheckPoint's firewall software is being sold on a chip to modem manufacturers for use with home computers.

## *Formal Specifications, Animated Specifications*

Formal methods use mathematical techniques in the specification, design, and verification of computer hardware and software.[422] More precisely, formal methods are[129]:

> *a software specification and production method, based on mathematics that comprises: a collection of mathematical notations addressing the specification, design, and development processes of software production; [the result being] a well-founded logical inference system in which formal verification proofs and proofs of other properties can be formulated [and] a methodological framework within which software may be developed from the specification in a formally verifiable manner.*

In short, formal methods use a formal notation, based on discrete mathematics, to specify and verify system requirements and design. There are a variety of formal notations, or languages, with supporting toolsets available today: B, calculus of communicating systems (CCS), communicating sequential processes (CSP), higher order logic (HOL), language for temporal ordering specification (LOTOS), OBJ, temporal logic, Vienna development method (VDM), and Z. These languages support a combination of computer hardware and software specification and design in concurrent, parallel, sequential, and real-time environments.

A formal specification is a system description written in a formal language. As such, it can be subjected to rigorous mathematical analysis to detect various classes of inconsistencies, incorrectness, and incompleteness.[69,422] The precision required by formal specification languages makes this analysis possible. With some toolsets, the analysis can be performed automatically, similar to a compiler performing syntax checks.[69] Another advantage of formal specifications is that they can be animated to illustrate specified system behavior and, in so doing, either validate requirements or highlight the need for clarifications and corrections. As IEC 61508-7 notes[69]:

> *Animation can give extra confidence that the system meets the* real *requirement as well as the formally specified requirement, because it improves human recognition of the specified behavior.*

Formal specifications were developed to compensate for weaknesses inherent in natural language specifications, in particular the ambiguity and susceptibility to misunderstanding and multiple interpretations.[422] They attempt to bridge the gap between a set of natural language requirements and a design implemented in a computer language. The development of formal languages began in the 1970s. The primary use was in verifying that security kernels correctly implemented a specified security policy and model. The mathematical precision of these languages was well suited to the task. In the 1980s, the safety community adapted and expanded the languages and methods to real-time, safety-critical control systems.[422] The languages, methods, and supporting toolsets continued to evolve and, by the mid-1990s, several national and international standards either mandated or highly recommended their use.[18,24,31,60,69,129]

Formal specifications have many advantages in the IA domain. Incorrect specifications are considered the source of most errors in computer systems[422]; in fact, some consider them to be the source of the most serious errors.[333] A technique that reduces the likelihood of errors being introduced during the specification phase is very beneficial in terms of cost, schedule, and ultimate system performance. Errors, inconsistencies, and ambiguities are resolved before coding begins. Formal specifications help to clarify access control requirements and operational modes/states of IA-critical and IA-related functions. The rigor imposed by the notation forces both positive and negative instances to be specified, similar to specifying MWFs and MNWFs. The process by which formal specifications are verified helps to ensure that stated IA integrity levels will be achieved and can be maintained. Animation permits the feasibility of a system, against specified constraints (memory, I/O, bandwidth, response times, etc.), to be evaluated before full-scale development.[81]

Exhibit 21 illustrates the precision of formal specifications by rewriting Exhibits 14c and 15 in a formal notation.

## Information Hiding

Information hiding is an IA design technique that enhances IA integrity by: (1) preventing accidental access to and corruption of critical software and

## Exhibit 21 Sample Formal Specifications

### A. Exhibit 14C Written as a Formal Specification

SECURITY LABELS

Confidential, Secret, TopSecret, AllModes: F SECURITY LABELS
---
Confidential ∪ Secret ∪ TopSecret = AllModes
Confidential _ Secret _ TopSecret = 0

DesktopOA ∪ LocalEmail = Confidential
DesktopOA _ LocalEmail = 0

Secret ⊃ Confidential
DesktopOA ∪ LocalEmail ∪ LimitedApplA = Secret
DesktopOA _ LocalEmail _ LimitedApplA = 0

TopSecret ⊃ Secret
DesktopOA ∪ LocalEmail ∪ FullApplA ∪ ForeignEmail ∪ InetSearch ∪
  ImportFiles = TopSecret
DesktopOA _ LocalEmail _ FullApplA _ ForeignEmail _ InetSearch _
  ImportFiles = 0

### B. Exhibit 15 Written as a Formal Specification

OPERATIONAL MODES

NormalOps, TriggerWarning, TriggerAlert, ActivateShutdown, AllModes:
 F OPERATIONALMODES
---
NormalOps ∪ TriggerWarning ∪ TriggerAlert ∪ ActivateShutdown = AllModes
NormalOps _ TriggerWarning _ TriggerAlert _ ActivateShutdown = 0

NormTemp ∪ NormPress = NormalOps
NormTemp _ NormPress = 0

(LowTemp ∪ LowPress) .OR. (NormTemp ∪ LowPress) .OR.
  (LowTemp ∪ NormPress) = TriggerWarning
(LowTemp _ LowPress) ∪ (NormTemp _ LowPress) ∪
  (LowTemp _ NormPress) = 0

(NormTemp ∪ HighPress) .OR. (HighTemp ∪ LowPress) .OR.
  (HighTemp ∪ NormPress) = TriggerAlert
(NormTemp _ HighPress) ∪ (HighTemp _ LowPress) ∪
  (HighTemp _ NormPress) = 0

(HighTemp ∪ HighPress) .OR. (LowTemp ∪ HighPress) = ActivateShutdown
(HighTemp _ HighPress) ∪ (LowTemp _ HighPress) = 0

data, (2) minimizing the introduction of errors during maintenance and enhancements, (3) reducing the likelihood of CCFs, and (4) minimizing fault propagation. IEEE Std. 610.12-1990 defines information hiding as:

> 1. A software development technique in which each module's interfaces reveal as little as possible about the module's innerworkings and other modules are prevented from using information about the module that is not in the module's interface specification;
>
> 2. A software development technique that consists of isolating a system function, or set of data and operations on those data, within a module and providing precise specifications for the module.

Information hiding can be applied to data and program logic. Information hiding increases reliability and maintainability by minimizing coupling between modules, while maximizing their cohesion.[81] Data structures and logic are localized and as self-contained as possible. This allows the internal data structures and logic of a module to be changed at a later date without affecting the behavior of other modules or necessitating that they also be changed; hence the four benefits listed above. Accordingly, information hiding is highly recommended by several national and international standards.[18,27,38,69] Object-oriented designs are quite amenable to information hiding.[277]

## *Intrusion Detection, Response*

Intrusion detection and response systems recognize and respond to a security breach, either as it is happening or immediately afterward. Intrusion detection and response systems operate behind a firewall; following the concept of defense in depth, they catch outsider attacks that penetrated a firewall. Insider attacks can also be foiled by intrusion detection and response systems. Lehtinen and Lear[331] quote from a CSI report that "financial losses from network security breaches at 163 businesses surveyed amounted to $123.7 million in 1998." They note that this figure is probably low because many companies under-report information security related losses in order to maintain customer confidence. Regardless, the need for robust intrusion detection and response systems will expand in proportion to the growth of e-Commerce and corporate and government dependence on information infrastructures. It is important to note that the consequences of security breaches are not limited to financial concerns; safety, legal, national security, and other issues can be raised as well. For example, when someone breaks into your house, that is a security breach. What is important, however, is what they do once they are inside.

There are three main types of intrusion detection and response systems[287]: statistical anomaly detection, rules-based detection, and hybrid. Statistical anomaly detection analyzes audit trail data for abnormal user or system behavior that may indicate an impending attack. Current audit trail data is compared against historical audit trail data that is presumed to reflect normal activity. While this approach is straightforward, it has some shortcomings.

First, if the historical data contains an undetected attack, that activity will be built into the normal profile. Second, it is possible for an attacker to learn what normal behavior is and as a result fool the system.[248] Third, if the normal profiles are specified too loosely or too tightly, a wave of false positives or false negatives will be triggered.[248,253,287] Fourth, real-time analysis of audit trail data is resource intensive.[287]

Rules-based detection monitors audit trail data for patterns of activity that match known attack profiles. This approach also has weaknesses. First, only known attack profiles will be detected; new or unforeseen attacks will not be recognized.[248,253,287] Second, the library of known attack profiles must be updated frequently or the system will quickly become obsolete.[248] Third, like statistical anomaly detection, it is resource intensive. Hence, the ideal approach is to deploy a hybrid intrusion detection and response system that combines statistical anomaly and rules-based detection.

There are several details to consider when implementing intrusion detection and response systems. Intrusion detection should be implemented for networks and host computers[331]; the latter is often overlooked. Denning[248] recommends several criteria to evaluate when developing potential attack profiles:

- Unexplained system crashes or restarts
- A series of unsuccessful login attempts
- Creation of accounts with no passwords
- Creation of accounts that grant root access
- Modifications to system programs
- Modifications to Internet configuration parameters
- A reduction in audit trail size (deleting footprints of an attack)
- Hidden files
- Sudden activity on a previously dormant account
- Logging in at strange hours
- Logging in more than once simultaneously
- Unusual activity in guest/visitor accounts

In addition, Cohen[244] makes the astute observation that the first phase of an attack may be to crash the intrusion detection system itself.

Intrusion detection software can be implemented on the network and application servers being monitored, although this is not recommended. As noted above, intrusion detection systems are resource intensive and will affect system performance. Instead, the use of self-contained intrusion detection units is preferred. This practice also helps to isolate the intrusion detection system from an attack.[277] There are several commercial products available that provide this capability. Industry and government are working together to develop standards that will promote vendor interoperability and common product evaluation criteria.[331] A key effort in this direction is the development of the common intrusion specification language (CISL) by the common intrusion detection framework (CIDF) working group.[315] The language will allow multiple vendors to use common parameters and syntax to detect and profile attack modes. As Kenyon reports[315]:

*Such communication allows applications to identify and locate assaults more accurately while enabling administrators to deploy advanced response and recovery procedures.*

A final implementation detail concerns how to respond to an attack. One option is to have the system automatically respond, with no human intervention. A second option is to trigger an alarm, equivalent to the audit trail security alarm, which requires human action. Primary and secondary recipients of the alarm must be identified along with the time interval during which a response must be taken. A third option is to trigger an alarm. Then, if no human action occurs within a specified time interval, the system automatically responds. It is likely that it will be preferable to use different options in response to the criticality of diverse attack profiles. To preempt a crisis situation, an appropriate response to each attack profile must be (1) specified in the operational procedures for the second and third options, or (2) preprogrammed into the system for the first and third options. The appropriate response is determined by reviewing/revalidating the fail safe/secure and fail operational design provisions.

## Partitioning

Partitioning enhances IA integrity by preventing the corruption and compromise of IA-critical and IA-related functions/entities. Partitioning performs two functions: it prevents (1) the non-IA-critical functions/entities of a system from *accidentally* corrupting or interfering with the IA-critical and IA-related functions/entities, and (2) the non-IA-critical functions/entities from being used as a vehicle for *intentionally* corrupting or compromising IA-critical and IA-related functions/entities.

The implementation of partitioning is straightforward. Similar to information hiding, partitioning requires complete interface specifications. Hardware and software partitioning can be implemented. Software partitioning can be logical or physical (such as logically partitioning a hard drive). Ideally, a combination of hardware, software, logical, and physical partitioning should be deployed. IA-critical and IA-related functions/entities are isolated from non-IA-related functions/entities. Both design and functionality are partitioned to prevent accidental and intentional corruption and interference. For example, one approach to protecting desktop workstations and home computers is hardware partitioning. Blackburn[215] reports that Voltaire has developed a security card that physically divides a PC into two different workstations — one public/unsecured and one private/secured. Only one segment can be accessed at a time; however, both segments utilize the same NIC. This product received NSA certification in July 1999.

This technique offers several advantages. First, it reduces the effort required to verify IA-critical and IA-related functions/entities. Second, through the use of partitioning, resources can be focused on the most critical elements in a system. Third, partitioning facilitates fault isolation and minimizes the potential for fault

propagation. Partitioning is often referred to as separability in the security community. Several national and international standards either mandate or highly recommend the use of partitioning.[18,24,31,53,126,127,129]

## *Plausibility Checks*

A plausibility check is an IA design technique that enhances IA integrity by verifying the validity and legitimacy of critical parameters before acting upon them. Plausibility checks detect faults early in the execution cycle and prevent them from progressing into failures or system compromises.

The implementation of plausibility checks is straightforward. Checks are performed on parameters that affect IA-critical and IA-related functions/entities before critical operations are performed to verify that the value of the parameters are both plausible and legal. The specific parameters checked will vary by application. Plausibility checks can be used to enhance safety, reliability, and security. Examples of items that can be checked to enhance safety and reliability include[60,129,130]:

- Parameter size (number of bits, bytes, digits, etc., to prevent overflow)
- Array bounds
- Counter values
- Parameters type verification, especially illegal combinations of parameters
- Legitimate called from routine
- Timer values
- Assertions about parameter values, operational mode/state, and pre- and post-conditions
- Range checks of intermediate results

Examples of items that can be checked to enhance security include:

- Is this a normal or feasible time for this user to be logging in (shift/travel/training/leave status)?
- Is this a normal or feasible location for this user to be logging in from (logical and physical network address, terminal ID, etc.)?
- Does this session reflect a normal activity level for this user account in terms of resources accessed and used (connection duration, application usage, files copied, printer usage, etc.)?
- What is the time interval between when this user was authenticated and the present (if too long, should reauthentication be initiated to preempt an imposter)?
- Is this type of request normally initiated by a user or a process?
- Are hard and soft copies being directed to a location normally used by this user?
- Does this account normally receive e-mail from this destination?
- Is it normal for 100 people to receive e-mail from different senders that has the same header? (Prevent distribution of e-mail virus.)

There is a fair degree of synergy between the specification of plausibility checks to be performed by a software application and the definition of normal profiles used by an intrusion detection system.

## *Redundancy*

Redundancy is a fault tolerance design technique employed to increase hardware reliability and system availability.[69,131,368,422] IEEE Std. 610.12-1990 defines redundancy as:

> the presence of auxiliary components in a system to perform the same functions as other elements for the purpose of preventing or recovering from failure.

In this context, identical components are used to perform identical functions. This contrasts with diversity, in which diverse components are used to perform identical functions.

The terms "reliability" and "availability" are often confused. Hence, it is important to clarify them in order to understand the role of redundancy in relation to reliability and availability. Hardware reliability refers to the ability of a system or component to correctly perform its function under certain conditions in a specified operational environment for a stated period of time. Availability is a measurement indicating the rate at which systems, data, and other resources are operational and accessible when needed, despite accidental and intentional subsystem outages and other disruptions. Availability is an outcome of reliability and a reflection of IA integrity. Availability is usually defined as:

$$A = MTBF/(MTBF + MTTR)$$

where: MTBF = Mean time between failures
MTTR = Mean time to repair

This definition is inadequate in the IA domain because it does not take into account failures induced by malicious intentional acts. If, however, the definitions of MTBF and MTTR are expanded such that:

$MTBF_{IA}$ = Mean time between accidental and intentional failures

$MTTR_{IA}$ = Mean time to repair and recover

then the calculation

$$A_{IA} = MTBF_{IA}/(MTBF_{IA} + MTTR_{IA})$$

would be appropriate in the IA domain. One problem remains. While both accidental and intentionally induced failures are random, predicting a failure rate for the latter will be difficult.

Redundancy, and the increased hardware reliability and system availability it provides, are important to IA for several reasons. The historical COMPUSEC model focused on confidentiality, integrity, and availability; without reliability, none of these can be achieved. To be effective, security features must function reliably. To do so, they are dependent on the reliable operation of hardware components, communications equipment, etc. If a hardware or communications component is not reliable, security features can be bypassed and defeated. For example, if firewall hardware is subject to intermittent faults, unauthorized users/processes may slip through. Unreliable hardware and communications equipment can yield incorrect results. Transient memory or CPU faults could lead to data corruption and compromise. Unreliable hardware and communications equipment may cause a critical function, service, or mission not to be performed on time, or at all. This could have security and safety consequences; for example, the loss of a telecommunications backbone or an ATC system. In summary, reliability is essential in achieving security goals and should not be overlooked. As Arbaugh et al.[202] succinctly state:

> *All secure systems assume the integrity of the underlying [hardware and] firmware. They usually cannot tell when that assumption is incorrect. This is a serious security problem.*

In other words, these assumptions need to be backed up by engineering facts.

Hardware redundancy is implemented three ways: active, standby, and monitored. Active redundancy utilizes multiple identical components operating simultaneously to prevent or detect and recover from failures. The redundant components operate in parallel. If a fault is detected in the primary unit, control is switched to the redundant or "hot standby" unit. The transition can be automatic or manual. Standby redundancy also utilizes multiple identical components. However, the redundant or "cold standby" units are not switched on until a fault is detected in the primary unit. Monitored redundancy, often referred to as m-out-of-n redundancy, is a variation of active redundancy. This method monitors the outputs of the parallel components. If discrepancies are found, voting logic is activated (hence the name m-out-of-n) to determine which output is correct and what action should be taken (e.g., switching to a new primary component).[69,422] Monitored redundancy is frequently used in PLCs as triple modular redundancy (TMR).

There are several issues to consider when implementing redundancy. Active redundancy permits a faster transition — the redundant unit is already powered-up and initialized; however, this unit consumes additional power, space, and weight and has been subjected to the same environmental stresses as the primary unit.[362] Standby redundancy provides a slower transition because of the need to power-up and initialize the system.[422] However, the remaining life of this unit is longer than a hot standby because it has not been stressed. Regardless of which method is used, two redundant units are required to trap a single fault, three redundant units are required to trap two faults, etc.[368,422]

Trade-off studies conducted early in the design phase evaluate which option is best for a given application.[131] These studies should ensure that single points

of failure are eliminated and that there are no common cause failure modes between redundant units. Redundancy can be implemented anywhere from low-level electronic components to major subsystems. As O'Connor[368] notes:

> *Decisions on when and how to design in redundancy depend upon the criticality of the system or function and must always be balanced against the need to minimize complexity and [development and operational] costs.*

Redundancy is not implemented in software; redundant software simply replicates the same systematic errors.[288,333,422] Instead, diverse software is implemented. Data redundancy is employed, through the use of parity bits and such, with data communication error detection/correction codes. The use of redundancy should be addressed in operational procedures and contingency plans.

## *Reliability Allocation*

Reliability allocation distributes reliability and maintainability requirements among system entities. Reliability is an essential part of IA integrity, as explained above. System reliability requirements are derived from IA goals. Reliability does not just "happen"; rather, like any other requirement, it has to be engineered into a system. The first step in this process is to allocate or apportion reliability and maintainability requirements to the hardware, software, and communications equipment that comprise the system. This step is analogous to the decomposition of functional requirements. Because of the impact of maintainability on operational reliability, these requirements are allocated at the same time.

Reliability allocation serves three main purposes[131]:

1. **Reliability and maintainability targets.** It provides designers, developers, and manufacturers of each part of a system with their target reliability and maintainability requirements. Most large systems today involve multiple organizations and vendors; in this way, each is provided with their reliability and maintainability target.
2. **Monitoring and assessment.** Reliability and maintainability values are available for comparison with assessments made later during the design and development phases. As a result, progress toward achieving reliability and maintainability goals can be monitored. If the evidence indicates that these goals will not be met, corrective action can be taken early in the life cycle.
3. **Trade-off studies.** The process of allocating reliability promotes architectural trade-off studies early in the design phase. Trade-off studies evaluate the technical feasibility and development and operational costs of alternate architectures that meet reliability and maintainability requirements. The extent of fault prevention and fault tolerance needed is also identified.

From its name, reliability allocation sounds simple. In fact, reliability allocation is a sophisticated iterative process that involves four steps.[131] The first step is to assign numerical reliability requirements to system entities and components. It is important to assign reliability requirements at a level (subsystem, component, subcomponent) that is meaningful and verifiable — requirements should not be assigned at a level that is too high or too low. Several factors should be considered when assigning reliability requirements[131,368]: operational mode (continuous or demand), duty cycle, criticality in relation to stated IA integrity level, whether or not the unit is mission repairable, the use of BITE, development risk (especially for new technology), complexity, uncertainty, and historical experience with similar units. The intent is to be realistic when allocating reliability requirements. Acceptable failure rates and the probability of surviving a failure or attack are used to express reliability requirements.

The second step is to determine the maintainability requirements needed to support the reliability requirements. The same conditions listed above are taken into account. Maintainability requirements are generally expressed in terms of an MTTR.

Third, system reliability and maintainability is calculated from the individual components. This is the reverse of the first step. Then system reliability is broken down into values for individual components. Now the values of the individual components are being combined to see if, in fact, if the system reliability and maintainability requirements will be met. In practice, the aggregate system reliability value should be somewhat higher than the stated requirement to compensate for uncertainty.[131,368]

Fourth, component reliability and maintainability requirements are refined through an iterative process of (re)allocation and (re)calculation until the goals are met. Reliability allocations and calculations are generally expressed and analyzed through the use of reliability prediction models, reliability block diagrams,[69,131,132,368] and more recently BBNs.[221,307,361]

Reliability and maintainability requirements are assigned to all system entities (internal, external, hardware, software, and communications equipment) except humans*. Techniques such as FTA, FMECA, and HAZOP studies support the allocation of reliability requirements and are encouraged because they are applicable to both hardware and software. The distinction between random hardware failures and systematic software failures must be maintained when allocating reliability requirements. It is recommended that separate reliability requirements be stated for different types and consequences of failure, taking into account factors such as[132]:

- The severity of the consequences of the failure
- Whether or not recovery from the failure is possible without operator intervention

* Human reliability analysis (HRA) is still subject to academic debate. Reliable, correct, and timely administrator and end-user actions are an important part of mission success. However, at this time, there is no consensus on how this should be estimated or measured.

- Whether or not the failure causes corruption of software or data
- The time required to recover from the failure

Reliability requirements for COTS and reused software should consider historical experience with these products if the previous application is quite similar to the new proposed use.

## *Secure Protocols*

Secure protocols enhance the confidentiality of distributed data communications by providing security services not available through basic communications protocols such as TCP, IP, or X.25. Secure protocols address interoperability issues related to implementing security features, such as placement of a digital signature, algorithms to use to sign and encrypt messages, and key sharing and authentication schemes.[403] Current secure protocols include IPSec, NLS, TLS1, SSL3, SET, PEM, PGP, and S/MIME. While these protocols are used for different purposes, they share a common two-step process. The first step is a digital handshake between the parties that want to communicate. During this step, the parties are authenticated to each other and they negotiate the context of their security association. Encryption algorithms, keys, and other security mechanisms to be used are agreed upon. The second step is the secure exchange of information; sensitive information is exchanged at the level of protection negotiated. If need be, the context of the security association can be updated.

IPSec provides strong one-way authentication for IPv4 and IPv6 packets.[403] Bidirectional communication requires the establishment of two security associations (client → server and server → client) and two sets of encryption keys. IPSec is transparent to users and applications. It attempts to address risks associated with password sniffing, IP spoofing, session hijacking, and denial-of-serve attacks.[372] The IPSec protocol can operate on a router or a firewall. The handshaking process is performed offline. The IPSec authentication header provides data integrity,[372] but not nonrepudiation of origin.[403] A keyed one-way hash function is used to calculate and verify the authentication header data. The second component of IPSec, the encapsulating security payload, can be configured in two modes: transport mode and tunnel mode. Transport mode is used for host-to-host connectivity and provides end-to-end security. In this mode, transport layer frames are encapsulated. In tunnel mode, the entire IP packet is encapsulated and a second IP header is generated for the security gateway. Sequence numbers are assigned to security payloads to prevent replay attacks. Optionally, security payloads can be encrypted before encapsulation, for example, with DES CBC.[277,403] IPv6 mandates the use of IPSec. IPSec is expected to be widely used with virtual private networks (VPNs) and remote connections to corporate intranets.[372] NLS and IPSec perform essentially the same functions.

TLS1, the Transport Layer Security protocol, represents an Internet Engineering Task Force (IETF) standardized version of SSL3. Once it is commercially available, TLS1 is expected to supersede SSL3 in many applications, especially in the ISO OSI environment.[403] TLS1 utilizes TCP virtual circuits. The TCP header and segment are encrypted, but the IP header is not.

SSL3, the Secure Socket Layer protocol, has been promoted by Netscape. It is designed for use in client/server and mobile code environments. The handshake step provides strong peer entity authentication based on X.509v3 digital certificates.[403,405] A separate pair of keys are used for client → server and server → client transactions. The secure transfer step compresses packets and transmits end-user data in encrypted, sequenced blocks.[277,405] Data integrity is ensured through the use of checksums that are produced by a one-way hash function. Currently, SSL3 does not support X.25[405] or provide nonrepudiation.[204]

SET, the Secure Electronic Transactions protocol, supports credit card-based online payment systems. The SET protocol specification, begun in 1996, represents a joint effort by the software industry, VISA, Mastercard, JCB, and American Express. SET is one of the strictest protocols in use today because it must meet the multi-party transactional audit integrity requirements of the financial industry.[260] It provides a secure messaging standard for electronic payments over open and untrusted networks. Internet users are, in effect, linked to credit card payment networks. SET combines X.509 digital certificates and key management framework with extensions to PKCS #7 digital signatures and digital envelopes. Selective multi-party field confidentiality is a key feature of SET; for example, an online merchant cannot see a customer's credit card number, but the financial institution can process the transaction.[260] As e-Commerce expands, so will SET usage. Potential future uses include home banking and online bill payment. The SET protocol is evolving; current information is posted at www.setco.org.[493]

PEM, the Privacy Enhanced Mail protocol, was developed in the late 1980s by the IETF. It was one of the first attempts to provide Internet e-mail privacy. The specification included encryption, digital signatures, and support for symmetric and asymmetric keys. However, because it is incompatible with the Internet mail system, due to seven-bit text messages, it lacks commercial support.

PGP, the Pretty Good Privacy protocol, was the next attempt at providing e-mail privacy. PGP supports digital signatures and encryption but lacks authentication services. It is useful for a small group of casual e-mail users who know each other, but falls short when it comes to scalability and accountability, particularly for large organizations.[260]

The development of S/MIME, the Secure MIME protocol, was a joint effort begun in 1995 by multiple companies under the leadership of RSA Data Security, Inc. The MIME protocol supports the exchange of text, created in diverse languages and character sets, among different computer systems. S/MIME adds PKCS #7 and PKCS #10 security features. Specifically, message origin authentication, message integrity, and nonrepudiation of origin are provided by digital signatures, while message confidentiality is provided by encryption. These features can be optionally selected through human action or set to automatically execute. Originally, S/MIME was used with e-mail; now its use is expanding to secure electronic document interchange (EDI), e-Commerce, and other applications. At present, S/MIME is supported by Netscape Messenger™ and Microsoft Outlook Express™. The IETF is developing S/MIMEv3 as an Internet standard, specifying both message syntax and digital certificate syntax. Current information can be found at www.rsa.com[489] and www.ietf.org.[475]

## *Virus Scanners*

Virus scanners are an IA design feature that automatically detects and removes computer viruses before they are activated. It is important to note that virus scanners detect the presence of viruses — they do not prevent infection. The purpose of virus scanners is to detect and remove viruses before they cause damage, such as deleting, overwriting, prepending, or appending files. The detect/remove process also helps to slow, but not eliminate, the spread of viruses. Virus alerts, real and hoaxes, regularly make the evening news because of the speed with which viruses spread and the extent of damage they are capable of inflicting. The Melissa virus made the rounds in 1999, infecting 1.2 million computers and 53,000 e-mail servers with a damage assessment of $560 million.[272] The Iloveyou virus, in May 2000, was even more destructive. As a result, the need for robust virus scanners is not expected to diminish any time soon.

Viruses are equivalent to cyber termites in that they use the resources of host computers to reproduce and spread, without informed operator action.[245,416] Viruses are spread via floppy disks, CD-ROMs, e-mail attachments, mobile code, and Web browsing.[248] Viruses are classified a variety of different ways[248,277,375]:

- **Boot virus:** Viruses that attack the boot sector of a hard drive or floppy disk and are activated at power-on
- **Macro virus:** viruses that are embedded in word processing documents or spreadsheets as macros and are activated when the macros are executed
- **Program virus:** viruses that attach to and attack *.exe, *.com, *.sys, and *.dll files when executed
- **Transient viruses:** viruses that are active only when the infected program is executing
- **Resident viruses:** viruses that remain in memory and link themselves to the execution of other programs

There are subcategories of viruses: parasitic, stealth, polymorphic, etc. For a complete description of virus types and categories, see Slade*.

Virus scanners are first cousins of intrusion detection systems. They examine code that is on or about to enter a system, looking for known virus profiles.[248] There are four main types of virus scanners[416]: (1) activity monitors, (2) change detection or integrity monitors, (3) pure scanners, and (4) hybrids. Activity monitors are the oldest type of antiviral software. They function similar to intrusion detection systems by looking for suspicious activity patterns that could indicate the presence of virus activity. Change detection or integrity monitors flag system and user files that have been changed and perform integrity checksums. They are apt to generate a lot of false positives unless a method is provided for flagging legitimate changes.[416] Pure scanners examine system and user files, boot sectors, and memory for evidence of infection through comparison with known virus signatures. Some new scanners incorporate heuristic features to detect viruses that have not yet been identified.[416] Most virus scan products on the market today are a hybrid of all three types.

* Slade, R., *Guide to Computer Viruses*, Springer-Verlag, 1994.[416]

There are several details to consider when implementing virus scanners. First, the use of virus scanners needs to be complemented with robust anti-virus operational procedures. For example[277]:

- All files should be checked before being used.
- Files should only be accepted from known trusted sources.
- Backups should be generated regularly and the integrity of data should be verified before it is backed up.[329]
- The FAT and interrupt tables should be protected.
- The use of default search paths should be minimized.
- Strong file access controls should be implemented.
- E-mail from unknown sources should be deleted.
- Unsolicited e-mail attachments should be considered suspicious and quarantined until they can be verified.
- E-mail chain letters should be blocked at the server.
- A large number of e-mail messages with the same header but different senders should be discarded.

Operational procedures should also specify how frequently virus scan software is executed and updated. It is recommended that both the execution and update process be run automatically and not rely on the actions of end users. Users should be aware that the virus removal process may corrupt original data; this fact reinforces the need for backups. Infected sectors on a hard disk or floppy disk and infected memory locations should be overwritten, more than once, to minimize the likelihood of a virus reappearing. Collaboration between the audit trail function, intrusion detection system, and virus scanner produces the best results. Finally, it should be remembered that the virus scanner itself can be subjected to attack. The latest information on virus alerts and antiviral products is available from http://service.symantec.com,[452] www.cert.org,[460] www.mcaffee.com,[479] and www.symantec.com/avcenter.[495]

In addition to IA design techniques/features, IA integrity depends on common sense, such as having comprehensive and current operational procedures, contingency plans, and physical security practices or protecting Web pages from tampering by storing them on CD-ROMs. As the biblical book of Amos (5:19) says, you do not want to run from a lion only to be caught by a bear.

## 6.6 Summary

The third component of an effective information security/IA program is the implementation of threat control measures. Five activities are performed during the implementation of threat control measures, as shown in Exhibit 22:

- The type, level, and extent of protection needed are determined.
- Controllability, operational procedures, and in-service considerations are evaluated.

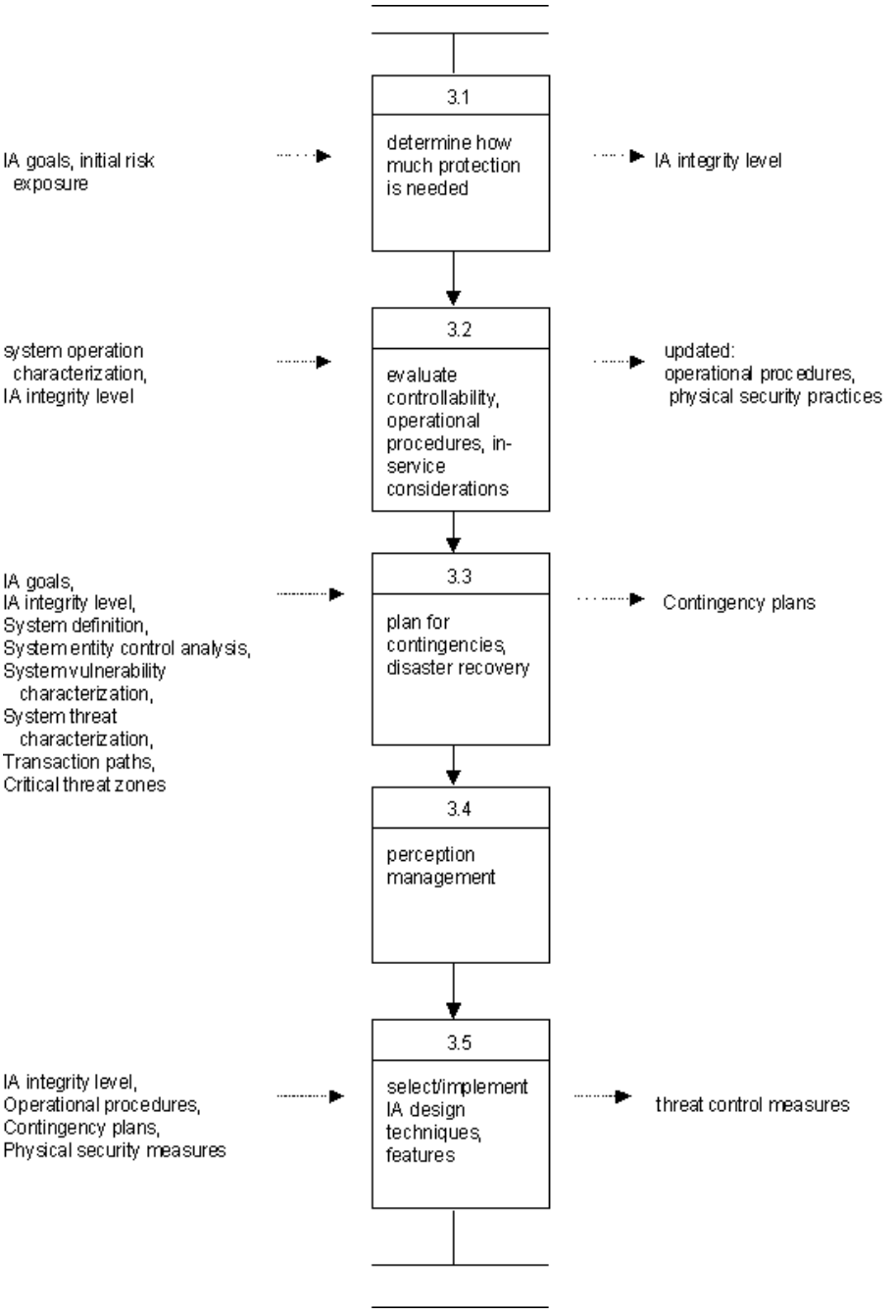


**Exhibit 22 Summary of Activities Involved in Implementing Threat Control Measures**

- Plans are made for contingencies and disaster recovery.
- The use of perception management is considered.
- IA design techniques and features are selected and implemented.

**Exhibit 23 Correlation of IA Design Techniques/Features to the Chronology of Threat Control Measures**

| *Anticipate/Prevent* | *Detect/Characterize* | *Respond/Recover* |
|---|---|---|
| Access control | Audit trail, security alarm | Audit trail, security alarm |
| Account for all possible logic states | Block recovery | Block recovery |
| Authentication | Digital signatures | Degraded-mode operations |
| Confinement | Diversity | Diversity |
| Defense in depth | Encryption | Error detection/correction |
| Defensive programming | Error detection/correction | Fail safe/secure, fail operational |
| Digital signatures | Intrusion detection | Intrusion detection, response |
| Encryption | Redundancy | Redundancy |
| Firewalls, filters | Virus scanner | |
| Formal specifications, animated specification | | |
| Information hiding | | |
| Partitioning | | |
| Plausibility checks | | |
| Reliability allocation | | |
| Secure protocols | | |

Threat control measures follow a fixed chronology: anticipate/prevent, detect/characterize, and respond/recover. The first step in implementing threat control measures is to determine the level of protection needed. This is accomplished by comparing the initial risk exposure to the target risk specified in the IA goals to determine the level of risk reduction needed. IA-critical and IA-related functions/entities and MWFs and MNWFs are identified. The entity control analysis and privacy issues are reassessed. Time intervals during which the level of protection is needed and the threat control measures will be effective are examined. From this, the required level of protection needed is updated and refined, leading to the specification of an IA integrity level.

Next, controllability, operational procedures, and in-service considerations are evaluated as opportunities to enhance — not detract from — IA integrity. Contingency plans are made to ensure that IA integrity is maintained in the event that one or more entities or services is inoperable or unavailable. Perception management is employed both to instill confidence in end users and to deter would-be attackers.

Finally, IA design techniques and features are selected and implemented based on (1) where they are effective in the threat control chronology, and (2) the specific vulnerabilities/threats they eliminate or mitigate. Protection is provided at all layers of the ISO OSI and TCP/IP reference models. Exhibit 23 correlates IA design techniques/features to the chronology of threat control measures, and Exhibit 24 correlates IA design techniques/features to common vulnerabilities and threats.

Next, Chapter 7 explains how to determine the effectiveness of threat control measures.

## 6.7 Discussion Problems

1. How is the need for reducing risk exposure determined?
2. What is the relationship, if any, between threat control measures and time?
3. Is it possible for an entity or function to be both IA-critical and IA-related? Explain your reasoning.
4. Identify MWFs and MNWFs for: (a) a nuclear power generation system, (b) a national security information system, (c) an intelligent transportation system, and (d) an online business.
5. Under what conditions would the IA integrity level be higher for: (a) a system entity than a system, (b) a system than a system entity, (c) security functions/entities than safety functions/entities, and (d) safety functions/entities than security functions/entities?
6. How is the IA integrity level affected by maintainability and operational procedures?
7. Explain the relationship between controllability and: (a) threats, (b) vulnerabilities, and (c) operational procedures.
8. Identify the controllability of the events in transaction path D ← 2.4.3 ← H (Chapter 5, Exhibits 21 and 25) from the six threat perspectives listed in Chapter 5, Exhibit 14. Assume the initial compromise was successful.
9. Who is involved in contingency planning? When are contingency plans developed?
10. How might a tactical defense intelligence system employ decoy entities? How would that differ from the use of decoys by a financial institution?
11. Identify a set of contingencies and responses for the online banking system discussed in Chapter 5.
12. What is the relationship between a system usage profile and threat control measures?
13. What is the relevance, if any, of the ISO OSI and TCP/IP reference models to IA?
14. Which layer in the ISO OSI and TCP/IP reference models is the easiest to attack? Which layer in the ISO OSI and TCP/IP reference models is the most difficult to attack?
15. Explain the differences, similarities, and relationship between: (a) digital certificates and digital signatures, (b) defense in depth and defensive programming, and (c) diversity and redundancy.
16. Which of the IA design techniques and features listed in Exhibit 13 are or are not applicable for: (a) COTS software, (b) custom software, and (c) reused software? Why?
17. Which method of specifying access control rules is best?
18. How many parameters should be used during authentication? How many authentication methods should be used?
19. How can partitioning be used to achieve: (a) security objectives, (b) safety objectives, and (c) reliability objectives?
20. Give examples of IA design techniques/features that complement, interact with, or are dependent on each other.

**Exhibit 24 Assignment of IA Design Techniques/Features to Common Vulnerabilities and Threats**

| *Vulnerability/ Threat*[a] | *Protective IA Design Techniques/Features* | |
|---|---|---|
| Accidental action, command, response | Account for all possible logic states<br>Defense in depth<br>Error detection/correction<br>Fault tolerance<br>Partitioning<br>Reliability allocation | Block recovery<br>Defensive programming<br>Fail safe/secure, fail operational<br>Formal specifications<br>Plausibility checks |
| Blocking access to system resources | Audit trail, security alarm<br>Firewalls, filters<br>Physical security | Diversity<br>Intrusion detection, response<br>Redundancy |
| Browsing | Access control<br>Authentication<br>Firewalls, filters | Audit trail, security alarm<br>Encryption<br>Intrusion detection, response |
| Corruption of resource management information (accidental and intentional) | Access control<br>Authentication<br>Defensive programming<br>Diversity<br>Fault tolerance<br>Information hiding<br>Plausibility checks | Account for all possible logic states<br>Defense in depth<br>Degraded-mode operations<br>Encryption<br>Formal specifications<br>Partitioning<br>Reliability allocation |
| Deletion of information or message (accidental and intentional) | Access control<br>Authentication | Account for all possible logic states<br>Information hiding |
| Denial of service, network flooding, system saturation, lack of capacity planning | Access control<br>Audit trail, security alarm<br>Diversity<br>Fault tolerance<br>Formal specifications<br>Redundancy | Account for all possible logic states<br>Degraded mode operations<br>Fail safe/secure, fail operational<br>Firewalls, filters<br>Intrusion detection, response<br>Reliability allocation |
| EMI/RFI | Error detection/correction | Physical security |
| Environmental, facility, or power faults or tampering | Degraded mode operations<br>Error detection/correction<br>Fault tolerance<br>Redundancy | Diversity<br>Fail safe/secure, fail operational<br>Physical security<br>Reliability allocation |
| Illegal operations, transactions, modes/states | Account for all possible logic states<br>Block recovery<br>Defensive programming<br>Fail safe/secure, fail operational<br>Formal specifications<br>Redundancy | Audit trail, security alarm<br>Defense in depth<br>Diversity<br>Fault tolerance<br>Plausibility checks<br>Reliability allocation |
| Inference, aggregation | Access control<br>Encryption | Authentication<br>Information hiding |
| Insertion of bogus data, "man-in-the-middle" | Access control<br>Digital signatures<br>Secure protocols | Authentication<br>Encryption |

**Exhibit 24 Assignment of IA Design Techniques/Features to Common Vulnerabilities and Threats (continued)**

| *Vulnerability/ Threat*[a] | *Protective IA Design Techniques/Features* | |
|---|---|---|
| Jamming | Degraded-mode operations<br>Physical security | Diversity |
| Lack of contingency planning, backups | Degraded-mode operations<br>Fail safe/secure, fail operational<br>Operational procedures | Diversity<br>Fault tolerance |
| Masquerade, IP spoofing | Authentication<br>Error detection/correction<br>Firewalls, filters<br>Secure protocols | Digital signatures<br>Encryption<br>Intrusion detection, response |
| Modification of information (accidental and intentional) | Access control<br>Authentication<br>Encryption<br>Formal specifications | Account for all possible logic states<br>Digital signatures<br>Error detection/correction<br>Information hiding |
| No fault tolerance, error detection or correction | Account for all possible logic states<br>Defense in depth<br>Diversity<br>Information hiding<br>Redundancy | Block recovery<br>Defensive programming<br>Fail safe/secure, fail operational<br>Plausibility checks<br>Reliability allocation |
| Overwriting information (accidental and intentional) | Access control<br>Encryption | Authentication<br>Information hiding |
| Password guessing, spoofing, compromise | Authentication<br>Secure protocols | Intrusion detection, response |
| Replay, reroute, misroute messages | Encryption<br>Firewalls, filters<br>Secure protocols | Error detection/correction<br>Intrusion detection, response |
| Repudiation of receipt, origin | Digital signatures | |
| Site/system/ application-specific vulnerabilities and threats | Defense in depth<br>Degraded-mode operations<br>Fail safe/secure, fail operational<br>Plausibility checks<br>Redundancy | Defensive programming<br>Diversity<br>Fault tolerance<br>Operational procedures<br>Reliability allocation |
| Theft of information, copying, distributing | Access control<br>Encryption<br>Operational procedures<br>Secure protocols | Authentication<br>Intrusion detection, response<br>Physical security |
| Theft of service | Access control<br>Digital signatures<br>Secure protocols | Authentication<br>Intrusion detection, response |

**Exhibit 24 Assignment of IA Design Techniques/Features to Common Vulnerabilities and Threats (continued)**

| *Vulnerability/ Threat*[a] | *Protective IA Design Techniques/Features* | |
|---|---|---|
| Trojan horse | Defense in depth<br>Degraded-mode operations<br>Fail safe/secure, fail operational<br>Partitioning | Defensive programming<br>Diversity<br>Fault tolerance |
| Unauthorized access to system resources | Access control<br>Authentication<br>Digital signatures<br>Firewalls, filters<br>Physical security | Audit trail, security alarm<br>Confinement<br>Encryption<br>Intrusion detection, response<br>Secure protocols |
| Unauthorized use of system resources | Access control<br>Authentication<br>Physical security | Audit trail, security alarm<br>Intrusion detection, response |
| Uncontrolled, unprotected portable systems and media, archives, hardcopy | Encryption<br>Physical security | Operational procedures |
| Unpredictable COTS behavior | Block recovery<br>Defense in depth<br>Diversity<br>Fail safe/secure<br>Information hiding<br>Plausibility checks | Confinement<br>Defensive programming<br>Error detection/correction<br>Fault tolerance<br>Partitioning<br>Reliability allocation |
| Virus attack | Audit trail, security alarm<br>Diversity<br>Fault tolerance<br>Partitioning<br>Virus scanner | Degraded-mode operations<br>Fail safe/secure, fail operational<br>Intrusion detection, response<br>Physical security |
| Wiretapping, eavesdropping, leakage | Access control<br>Encryption<br>Physical security | Authentication<br>Intrusion detection, response<br>Secure protocols |

*Note:* There is no one-to-one correspondence between vulnerabilities/threats and IA design techniques/features. Instead, it is the cumulative effect of multiple techniques/features that eliminates or mitigates vulnerabilities/threats. Also, the design techniques/features are effective at different points in the threat control chronology (anticipate/prevent, detect/characterize, respond/recover), as illustrated in Exhibit 23.

*Sources:* Adapted from Denning, D., *Information Warfare amd Security*, Addison-Wesley, 1999; Denning D., *Cryptology and Data Security*, Addison-Wesley, 1982; Gollmann, D., *Computer Security*, John Wiley & Sons, 1999; Morris, D., *Introduction to Communication Command and Control Systems*, Pergamon Press, 1977; Rozenblit, M., *Security for Telecommunications Network Management*, IEEE, 1999.