Chapter 3

Historical Approaches to Information Security and Information Assurance

Safety, reliability, and security concerns have existed as long as there have been automated systems. The first standards for software safety* and software security** were developed in the late 1970s; the first software reliability*** standards followed a decade later. These standards represented a starting point for defining safety, security, and reliability design, development, assessment, and certification techniques. Implementation, however, was fragmented because safety, security, and reliability were handled by different communities of interest and there was little communication or coordination between them. These techniques were appropriate for the technology and operational environments of their time. A time when computers and telecommunications were separate entities; computer networks consisted of dedicated lines; and textual, image, audio, and video data were isolated. Distributed processing had just begun, but portable computers and media remained unknown. Many of these techniques assumed that the computer was in one room or, at most, a few local buildings.

This chapter reviews the historical approaches to information security and information assurance, specifically the approaches to system security, safety, and

---

* MIL-STD-882A, System Safety Program Requirements, U.S. Department of Defense (DoD), June 28, 1977.

** DoD 5200.28-M, ADP Computer Security Manual — Techniques and Procedures for Implementing, Deactivating, Testing, and Evaluating Secure Resource-Sharing ADP Systems, with 1st Amendment, U.S. Department of Defense (DoD), June 25, 1979.[140]

***IEEE Std. 982.1-1989, IEEE Standard Dictionary of Measures to Produce Reliable Software.[42]

reliability; what these approaches accomplished; and their limitations relative to today's technology. These historical approaches fall into seven main categories:

1. Physical security
2. Communications security (COMSEC)
3. Computer security (COMPUSEC)
4. Information security (INFOSEC)
5. Operations security (OPSEC)
6. System safety
7. System reliability

Many of these approaches originated in the defense and intelligence communities. At the time, only national security information was considered worth protecting. Gradually, these approaches spread to the financial community and others. The limitations of traditional security standards reflect their origin. As Underwood[434] notes:

- They only assess products, not the development processes.
- They evaluate components, not systems.
- They required a specialized skill set by the assessor or the results were invalid.
- They focus on correct solutions, not necessarily cost-effective ones.

3.1 Physical Security

Physical security is defined as:

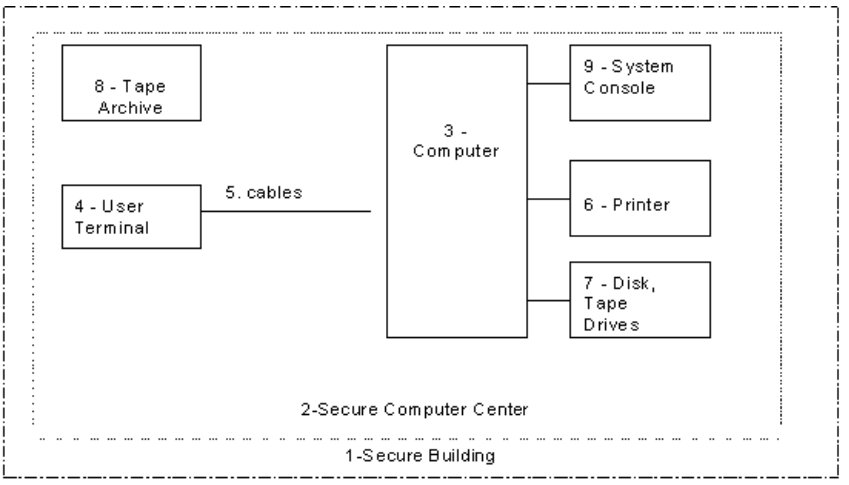
> the protection of hardware, software, and data against physical threats to reduce or prevent disruptions to operations and services and/or loss of assets.

In summary, the purpose of physical security is to protect physical system resources (as opposed to logical system resources) from (1) physical damage that could impair operations and (2) theft.

Historically, physical security plans focused on four major challenges:

1. Protecting computer and communications resources from damage due to fire, water, radiation, earthquake, or other natural disaster
2. Maintaining appropriate temperature, humidity, dust, and vibration levels
3. Providing sustained power levels despite transient spikes, brownouts, and power failures
4. Controlling physical access to computer and communications resources to known authorized personnel

The primary emphasis was on protecting the central computer facility or computer center that housed the mainframe computer, operator console(s), disk packs, tape drives, and high-speed printers. Secondary emphasis was



1. Controlled access to building, badge/visual recognition required.
2. Controlled access to computer center, badge, combination lock, fingerprint scanner, visual recognition required. UPS. Specialized HVAC. Protection from fire, flood, radiation, earthquake, natural disasters.
3. Only local equipment and remote equipment connected by dedicated lines. Only known authorized users allowed in Computer Center. Center staffed 24 hours/day. Memory overwritten before reuse.
4. Controlled access to user terminals. Use of shielded terminals.
5. Use of shielded cables, locked cable ducts.
6. Controlled access to printers and printouts. Sensitive printouts packaged in opaque plastic. Old unneeded printouts shredded, placed in burn bags.
7. Controlled access to disk and tape drives. Magnetic media overwritten before reused, degaussed before disposal.
8. Controlled access to tape archive. Tapes stored in locked cabinet.
9. Controlled access to operator console.

Exhibit 1 Traditional Physical Security Perimeters

placed on protecting remote "dumb" terminals, printers, and modems. Archives and documentation relating to the design and operation of the mainframe were generally protected in the same manner as the computer center (see Exhibit 1).

The computer center was located in a dedicated room, usually the first floor or basement of a building because of the weight involved. The room was constructed with flame-retardant raised floor panels and ceiling tiles, and water sprinklers or chemical fire suppressants. Specialized heavy-duty cooling and air filtering systems were installed to keep the computer center cool, usually 68°F. Specialized flooring was installed to absorb vibration. Robust surge protection and ambient power sources were provided by high-capacity uninterrupted power supplies (UPS) and motor generators (MGs). In some circumstances, computer equipment was designed to be resistant to radiation (Rad Hard). In short, not much was left to chance in terms of the computer facility itself.

Likewise, a variety of measures were employed to control physical access to computer and communications resources. Access to the computer center and rooms containing remote equipment was restricted by badge readers, combination locks, fingerprint scanners, and visual recognition to known authorized operations staff, users, and maintenance staff. People without the

appropriate identification had to be escorted at all times. Some computer centers activated a flashing red light to alert staff when an uncleared person was in the room. Occasionally, the "two man rule" was implemented whereby a minimum of two people had to staff the computer center at any time. Depending on the classification level of printouts, they were sealed in opaque plastic and had to be signed for. Equipment leaving the computer room had to be signed for and accompanied by an authorized property pass. Video surveillance cameras kept track of people and equipment entering and leaving the computer center.

Communications traffic was isolated on different channels according to classification level (red/black separation). Cables, and sometimes peripherals and computer centers, were shielded (Tempest technology) to prevent emanations that could be intercepted and interpreted. To prevent or at least minimize wire tapping, dedicated lines connected remote floors and facilities to the computer center and cable ducts were locked. DoD 5300.28-M[140] required that each remote physical device and location have a unique ID. Prior to establishing a session, the CPU would verify that the device was legitimate and had the proper authorization for the requested classification level; if not, the connection was dropped.

Disk mirroring helped prevent the loss of critical data and the associated downtime. During one phase, removable hard drives were in vogue. At the end of a work day or shift, the hard drives were removed and locked in safes. Locks of various types were used to prevent the theft of computers, disks, tapes, archival media, and printouts. Off-site storage provided continuity of operations in the event of a natural disaster or intentional sabotage.

The safe and reliable disposal of obsolete, unneeded, or inoperable classified resources remains a perennial problem. In the early days, elaborate schemes were developed to erase, degauss, and destroy tapes, disks, hard drives, and memory. For example, DoD 5200.28-M[140] required:

- Using a bulk tape degausser that performed a triple overwrite procedure: first all binary 1's, then all binary 0's, followed by repeating a single random alphanumeric character
- Transitioning core memory from a binary 1 to 0 to 1 again, 1000 times
- Exposing inoperable equipment to a magnetic field of 1500 Oersted three times

Equivalent procedures were specified for destroying nondigital information, such as an analog audio recording, that was recorded on magnetic media. Paper resources were shredded and placed in burn bags. Today, volatile memory should be overwritten before it is reused or powered down, much like the above precautions taken to protect magnetic storage media, because the contents can be reconstructed. As Gollmann[277] notes, "The content loss of volatile memory is neither instantaneous nor complete."

In the past, significant emphasis was placed on physical security — protecting the computer center. It was assumed that (1) an intruder had to be on the premises; and (2) if a security perimeter were compromised, the system would

quickly shut down or switch to only unclassified processing. Some historical physical security paradigms are still valid; however, many are not. The computer is no longer in one room. For example, in the client/server environment, processing power, memory, and disk storage are distributed. Computer and communications equipment are no longer separate; rather, they are integrated. Remote access is the norm rather than the exception, whether through mobile computing, telecommuting, home offices, or remote diagnostics/help. Although some PC CPU chips and operating systems have unique identifiers, not all do; nor is a user tied to a single desktop system. The days of dedicated lines are gone, particularly with the advent of wireless and other high-bandwidth services. Today, physical security must consider the ramifications of LANs, WANs, VPNs, and the Internet. CD-R disks have created a new challenge for disposing of obsolete sensitive material. Finally, intruders are rarely on the premises.

3.2 Communications Security (COMSEC)

Communications security, or COMSEC, is a collection of engineering activities that are undertaken to protect the confidentiality, integrity, and availability of sensitive data while it is being transmitted between systems and networks. Confidentiality ensures that only the intended recipients receive and are able to interpret the transmitted data. As a result, potential losses from theft of information are minimized, including financial loss, loss of competitive advantage, loss of public confidence, loss of privacy, character defamation, national security compromises, and loss of intellectual property rights.[248] Integrity ensures that the data received is an accurate representation of the data sent. In other words, the data has not been accidentally or intentionally altered, corrupted, destroyed, or added to. Availability ensures that the data is received within the specified transmission time(s), plus or minus a reasonable tolerance factor. DoD 5200.28-M[140] required that COMSEC principles be applied to all: communications lines and links, multiplexers, message switches, telecommunications networks and interfaces, and emanations control. In the past, COMSEC focused on protecting end-to-end data transmission across dedicated lines from one secure facility to another, as shown in Exhibit 2. Data leaving a computer center was multiplexed and encrypted, sometimes more than once. A secret key system was used, and the keys were changed simultaneously on both ends of the communications link on a regular basis.

Spectrum management and encryption were the primary means of providing confidentiality. Spectrum management attempted to prevent or at least minimize the ability to intercept, interpret, or block data transmissions through spectrum planning and interference and jamming countermeasures. In some cases, this involved regular changing of call signs, words, suffixes, and frequencies. In addition, switches that permitted users to access systems at different security levels had to do so while ensuring that an electrical connection did not exist between the two systems or networks.

Encryption, assuming that spectrum management and other security measures are not 100 percent foolproof, makes data and messages unintelligible to all

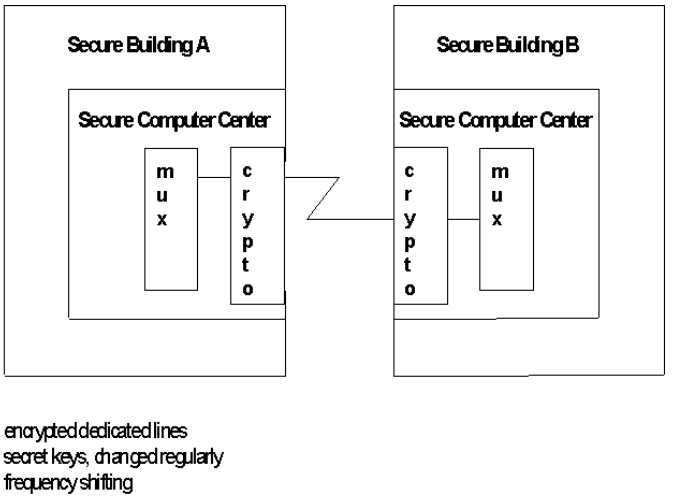


Exhibit 2 Historical COMSEC Architecture

but the intended recipient(s). This is accomplished through a systematic method of scrambling the information. A mathematically based encryption algorithm specifies the steps involved in transforming the data. Encryption algorithms can operate on a single bit or byte (stream ciphers) or a fixed number of bits of data at a time (block ciphers). Encryption algorithms can be implemented in hardware or software.

A key represents a specific instance of the encryption algorithm. Keys can be public, private, or secret and are changed frequently. In contrast, the algorithm remains constant. Both or all parties to a transaction must use the same encryption algorithm and know which type of key scheme is being used. With secret keys, the same key is used for encryption and decryption. More recently, public/private key schemes were developed in which one key is used for encryption and another for decryption. Public/private keys only work in designated pairs. Historically, both the encryption algorithm and the key were safeguarded. Today, a variety of encryption algorithms are publicly available and only private and secret keys are protected. (This arrangement assumes that the private key cannot be determined from the public key.)

Cryptography predates computers and probably has been around as long as humans have had the need to send/receive secret messages.[316] Knott[318] reports on the use of cryptography during the American Revolution:

> *Throughout the American Revolution, General Washington placed great importance on learning British intentions and shielding his own army's activities. Elaborately coded communications were used by the general to communicate with his spy network through a system implemented by staff officers such as Alexander Hamilton. … In 1779, Tallmadge added to this layer of protection by developing a cipher and code that used codebooks available only to himself, Townsend, and Washington.*

Exhibit 3 Simple Illustration of the Steps Involved in Encryption

Step 1:	English language, Roman alphabet
Step 2:	Block cipher, 128-bit, 16-byte block size
Step 3:	a. Move blanks (x+1) spaces to the right.
	b. Move vowels (x–1) spaces to the left.
	c. Replace consonants with the consonant that is x places after it in the alphabet. Loop around the alphabet if necessary.
	d. Enter the message in reverse order.
Step 4:	Secret key, (x) = 2
Step 5:	ASCII

Plaintext message:	Happy Birthday.
Blocked message:	Happy Birthday.^
Intermediate messages:	a. happybir thday.^
	b. ahppyib rthady.^
	c. ajqqzic tvjafz.^
	d. ^.zfajvt cizqqja
Cipher text:	^.zfajvt cizqqja

The following simple example, that of sending a secret message between two parties, illustrates some of the basic principles involved in cryptography. The first step is to determine what language and alphabet the message is to be sent in. This is important because of the different number of letters in each alphabet, the fact that some languages are written right to left while others are written left to right, and the fact that in some alphabets the same character can represent a letter and a number — they have a dual meaning. The second step is to determine whether a block cipher or stream cipher will be used; if a block cipher, define the blocksize of data that the encryption algorithm operates on; that is, how many 8-bit bytes. The third step is to specify the shift/substitution/manipulation algorithm. The fourth step is to define the key type and length. The fifth step is to specify the code in which the data will be represented to the computer, for example, ASCII. Exhibit 3 illustrates these principles.

This simple example starts out with an English language message: Happy Birthday. This is the message the sender wants to send the recipient; it is referred to as the plaintext message. First, the message is blocked to fit the specified encryption block size of 16 bytes. Because the message is only 15 bytes long, it is padded with a blank space (^). (Note that some encryption algorithms pad on the left, and others pad on the right.) Next, the message goes through the four transformations specified by the encryption algorithm. Note that x is the key and in this instance x = 2. The final transformation results in the cipher text, which is transmitted to the recipient. To read or decrypt the message, the recipient goes through the same steps in reverse order using the same key. The sender and the receiver know the encryption algorithm and the key; that information does not have to be transmitted. This is how a secret key encryption system works.

In 1976, the U.S. National Bureau of Standards (NBS), now the National Institute of Standards and Technology (NIST), published the Data Encryption

Standard (DES).[153] DES was developed to protect the transmission of unclassified data by government agencies. The algorithm operated on 64-bit blocks of data with a 56-bit key. In summary, the algorithm consisted of two steps that were repeated 16 times[409]:

1. Exchange the left half of the 64-bit message with the right half.
2. Replace the right half of the message with the bitwise exclusive OR of the right half and a 32-bit word (a complicated function (f) of the left half, the key, and the iteration number).

Since then, many publicly available encryption algorithms have been developed.

The concept of using a pair of keys — one to encrypt and the other to decrypt — began in the late 1970s. This concept became known as public/private keys. The two keys are mathematically related; however, in *theory*, it is infeasible to derive one key from the other. RSA* was the first public key encryption system sufficiently robust to use for both encryption and digital signatures. The following steps summarize the RSA algorithm[409]:

1. Choose two, large random prime numbers (p, q) of equal length.
2. Compute n = p * q.
3. Choose a random prime number e, such that e has no factors in common with ((p–1)(q–1)); e with n comprise the public key.
4. Compute the private key, $d = e^{-1} \bmod ((p-1)(q-1))$.
5. Choose a binary block size that equals the largest power of 2 that is less than n.
6. Break the plaintext message into blocks (m_i); pad if necessary.
7. Generate the cipher text, $c_i = m^e_i \bmod n$.
8. Recover the plaintext, $m_i = c^d_i \bmod n$.

Note that p and q are destroyed, only the public key (e, n) is distributed, and the private key (d) is protected.

The key to the acceptance and widespread use of public key encryption has been the public key infrastructure (PKI) made possible through a set of public key cryptography standards known as PKCS.[179–191] RSA Laboratories has been the driving force behind the development and promulgation of the PKCS suite since the early 1990s. They understood early on that the way to achieve interoperability in a multi-vendor environment is through the use of commercial consensus standards. Most PKCSs exhibit a reasonably high degree of compatibility with the corresponding ISO/IEC standards. The current set of PKCSs at the time of writing are listed below. The latest information on PKCS developments can be found at: www.rsa.com[489]:

- PKCS #1 v2.1 — RSA Cryptography Standard, (draft) September 17, 1999
- PKCS #3 v1.4 — Diffie-Hellman Key Agreement Standard, November 1, 1993

* The algorithm is named for its three inventors: Ron Rivest, Adi Shamir, and Leonard Adleman.

- PKCS #5 v2.0 — Password-based Cryptography Standard, March 25, 1999
- PKCS #6 v1.5 — Extended Certificate Syntax Standard, November 1, 1993
- PKCS #7 v1.5 — Cryptographic Message Syntax Standard, November 1, 1993
- PKCS #8 v1.2 — Private Key Information Syntax Standard, November 1, 1993
- PKCS #9 v2.0 — Selected Object Classes and Attribute Types, February 25, 2000
- PKCS #10 v1.7 — Certification Request Syntax Standard, May 26, 2000
- PKCS #11 v2.11 — Cryptographic Token Interface Standard, (draft) November 2000
- PKCS #12 v1.0 — Personal Information Exchange Syntax, June 24, 1999
- PKCS #13 (proposal) — Elliptic Curve Cryptography Standard, October 7, 1998
- PKCS #15 v1.1 — Cryptographic Token Information Syntax Standard, June 6, 2000

A newer type of public key cryptosystems is referred to as elliptic curve cryptosystems (ECCs). Lee[330] reports that:

> *Protocols based on ECCs are becoming the standard for the information authentication step for wireless devices. … Breaking an ECC requires determining the number of times that a seed value, a known point on an elliptic curve, is multiplied to get to another point on the same elliptic curve.*

Encryption algorithms involve complex mathematical specifications of the transformations performed on the data, such as hashing functions. Often, the entire algorithm is repeated two, three, or more times. Key lengths range from 56 to 128 bits or higher. For a complete discussion of current cryptographic algorithms, see Schneier,[409] Menezes,[353] Kippenhan,[316] and Stallings.[419]

Historically, the primary means of providing integrity was by implementing error detection/correction algorithms on the communications links. This included longitudal and vertical parity checks, cyclical redundancy checks, Hamming codes, convolutional codes, recurrent codes, and checksums. These algorithms verified that the data had not been accidently or intentionally corrupted during transmission, including the deletion of data or insertion of bogus data, and that the packets were reassembled in the correct order. When errors were detected, they were corrected or a request was sent to retransmit the packet.

Historically, the primary means of providing availability was through redundant communications equipment (hot standby) and having alternative communication paths available in case the primary path was not available. For the data to be available when needed, the communications equipment and links had to be engineered to meet reliability requirements.

In the past, COMSEC principles were primarily applied to end-to-end communication links that transmitted textual, voice (STU technology), and image data separately. Today, COMSEC principles are applied to audio, video, image, and textual data that are transmitted together across a variety of network types and topologies, such as ISDN, ATM, SONET, Frame Relay, VPNs, and wireless. The need for data confidentiality, integrity, and availability during transmission remains; what has changed are the implementation strategies. Encryption is applied to data that is stored (files, e-mail, voice mail) as well as data that is transmitted (Internet and cell phone traffic). Data integrity concerns have been expanded to include verifying the true sender of files or e-mail through the use of digital signatures. Likewise, the distribution of public keys is verified. Because dedicated lines are rarely used anymore, firewalls are employed to block unknown and unauthorized people and processes from accessing network resources.

Encryption is not a perfect solution to data confidentiality; instead, it should be considered a temporary solution. All cryptographic algorithms can be broken just like shredder remnants can be pieced together; the variable is the amount of time it takes. Schneier[411] notes the limitations of commercial encryption products:

> *Most cryptography products on the market are insecure. Some don't work as advertised. Some are obviously flawed. Others are more subtly flawed.*

Schneier[410] cites several common weaknesses in implementing encryption algorithms, including:

- Not destroying the plaintext after encryption
- Use of temporary or virtual swapping files
- Buffer overflows
- Weak error detection/correction
- Key escrow accounts
- Use of default parameters
- Ability to reverse-engineer the product

In short, cryptography should be considered only one component of an overall comprehensive security program.[248,277,410]

Three concerns must be addressed when implementing encryption:

1. The time and system resources consumed to perform encryption and decryption
2. When to perform encryption; that is, what layer in the communications protocol suite
3. What encryption algorithm to use or what encryption strength/level of protection is needed

Encryption consumes time and processing power for both the sender and receiver. The more complex the encryption algorithm, the more system resources are consumed. To address the first item, Sandia National Laboratories has developed an ASIC that implements the DES algorithm. It is targeted for use in unclassified networks, digital cell phones, and high-definition television

(HDTV). The ASIC can also support triple-DES. As reported by Hankins,[283] the nominal encryption rate is 6.7 billion bps with a theoretical limit of 9.28 billion bps. The next step is to commercialize the product.

The level of encryption strength needed will depend on what type of information is being protected, for example, national security information, financial transactions, corporate research, or general-purpose e-mail. To address the third item, in 1997, NIST made a formal announcement of its intent to sponsor the development of an advanced encryption standard (AES). NIST chose to engage the resources of the international cryptographic community to develop the new AES algorithm, rather than develop it in-house as had been done with other cryptographic algorithms in the past. The goal of the AES project was to develop a replacement for DES, which is no longer considered sufficiently robust. The basic requirements were that AES support a block size of 128 bits, and key lengths of 128, 192, and 256 bits. Fifteen algorithms from twelve countries were submitted for the initial selection process. In October 2000, Rijndael was selected "as the best overall algorithm for AES".[173] The next step is to issue Rijndael as the approved AES federal (U.S.) information processing standard (FIPS); this is scheduled to occur in the summer/fall of 2001. Commercial products will follow thereafter. For a complete discussion of the new algorithm and the selection process, see Reference 173.

The second item must be addressed on a case-by-case basis within the context of the overall security program, network, system, and application architectures. Chapter 6 provides guidance in this area.

3.3 Computer Security (COMPUSEC)

Computer security is defined as:

> preventing, detecting, and minimizing the consequences of unauthorized actions by users (authorized and unauthorized) of a computer system.

In this case, the term "users" includes authorized users, or insiders, who are attempting to do something for which they lack permission, as well as unauthorized users, or outsiders, who are attempting to break into a system. The term "computer system" applies to any type or configuration of hardware and software, including distributed processing, client/server applications, embedded software, and Internet applications. COMSEC is primarily concerned with protecting data during transmission. COMPUSEC is primarily concerned with protecting data while it is processed and stored. Some of the threats to stored data include[249]:

Active Threats	*Passive Threats*
Overwriting	Browsing
Modifying	Aggregation and inference
Inserting	Replaying
Deleting	Leakage
Blocking access to	Copying and distributing

Primarily defense and intelligence systems employed COMPUSEC in the past. The intent was to prevent deliberate or inadvertent access to classified material by unauthorized personnel, or the unauthorized manipulation of the computer and its associated peripheral devices that could lead to the compromise of classified information.[140] COMPUSEC principles were applied to the design, development, implementation, evaluation, operation, decommissioning, and sanitization of a system.

A secure system operated in one of four modes: controlled, dedicated, multilevel, or system high. Occasionally, a system was designed so that it could be shut down and restarted in a different security mode. These four modes are defined as follows[140]:

1. Controlled security mode. Some users with access to the system have neither a security clearance nor a need-to-know for all classified material contained in the system. The separation and control of users and classified material on the basis of security clearance and security classification are not under operating system control.
2. Dedicated security mode. The computer system and all of its peripherals are exclusively used and controlled by specific users or groups of users who have a security clearance and need-to-know for the processing of a particular category and type of classified material.
3. Multi-level security mode. The system permits various categories and types of classified materials to be concurrently stored and processed and selective access to such material concurrently by uncleared users and users having differing security clearances and need-to-know. Separation of personnel and material on the basis of security clearance and need-to-know is accomplished by the operating system and related system software.
4. System high security mode. All system components are protected in accordance with the requirements for the highest classification category and type of material contained in the system. All personnel having access to the system have a security clearance but not necessarily a needs-to-know for all material contained in the system. The design and operation of the system must provide for the control of concurrently available classified material on the basis of need-to-know.

Each of these four security modes represented a different approach to the control and separation of dissimilar combinations of user clearances, needs-to-know, and level(s) of classified material handled by the system.

DoD 5200.28-M*, one of the first COMPUSEC standards, levied the following requirements on computer systems[140]:

* DoD 5200.28-M, ADP Security Manual — Techniques and Procedures for Implementing, Deactivating, Testing, and Evaluating Secure Resource-Sharing ADP Systems, with first amendment, U.S. Department of Defense (DoD), June 25, 1979.

1. Ensuring that two or more independent controls would have to malfunction simultaneously for a breach of system security to occur (defense in depth)
2. Monitoring protection state variables to control execution of operations and prevent illegal operations
3. Controlling access to memory locations
4. Ensuring predictable translation into object code
5. Protecting registers through error detection and redundancy checks
6. Performing parity checks and address bound checks of all operands/operators
7. Using interrupts to control operator malfunction
8. Verifying read, write, edit, and delete privileges
9. Labeling classified material
10. Clearing memory residue, overwriting memory before reuse
11. Logging attempts to circumvent system security measures
12. Implementing security safeguards during scheduled and unscheduled system shutdown, restart, and start-up
13. Maintaining an audit trail of security-related transactions, such as log on/log off attempts and times, information about resources accessed, created, changed, deleted, outputs generated, etc.
14. Employing user and terminal IDs as part of the access control and authentication system
15. Controlling access to system resources, utilities, and data through the operating system

The purpose of these measures was to prevent accidental or malicious intentional violations of system security and provide historical records of such transactions.

DoD 5200.28-M specified the implementation of COMPUSEC features. The next logical development was a standard that specified how to evaluate the effectiveness of the implementation of these features. The result was CSC-STD-001-83, the Trusted Computer System Evaluation Criteria (TCSEC)*, commonly known as the *Orange Book*, issued by the U.S. DoD in 1983. A second version of this standard was issued in 1985**.

The *Orange Book* proposed a layered approach for rating the strength of COMPUSEC features, similar to the layered approach used by the Software Engineering Institute (SEI) Capability Maturity Model (CMM) to rate the robustness of software engineering processes. As shown in Exhibit 4, four evaluation divisions composed of seven classes were defined. Division A class A1 was the highest rating, while division D class D1 was the lowest. The divisions measured the extent of security protection provided, with each class and division building upon and strengthening the provisions of its predecessors.

* CSC-STD-001-83, Trusted Computer System Evaluation Criteria (TCSEC), National Computer Security Center, U.S. Department of Defense (DoD), August 15, 1983.

** DoD-5200.28-STD, Trusted Computer System Evaluation Criteria (TCSEC), National Computer Security Center, U.S. Department of Defense (DoD), December 1985.

Exhibit 4 Summary of *Orange Book* Trusted Computer System Evaluation Criteria (TCSEC) Divisions

Evaluation Division	*Evaluation Class*	*Degree of Trust*
A - Verified protection	A1 - Verified design	Highest
B - Mandatory protection	B3 - Security domains	
	B2 - Structured protection	
	B1 - Labeled security protection	
C - Discretionary protection	C2 - Controlled access protection	
	C1 - Discretionary security protection	
D - Minimal protection	D1 - Minimal protection	Lowest

Twenty-seven specific criteria were evaluated. These criteria were grouped into four categories: security policy, accountability, assurance, and documentation, as shown in Exhibit 5.

Most often, a basic capability was established at C1, and then new requirements were added at each of the seven layers. Security testing is a good example of this, as illustrated in Exhibit 6. (It is interesting to note that although the *Orange Book* was published in the early 1980s, it required security testing to evaluate the ability of a system to withstand denial-of-service attacks at level B1.) In other cases, a criterion was only required at the higher layers, such as trusted recovery.

The *Orange Book* introduced the concepts of a trusted computing base (TCB) and security kernel or reference monitor. A TCB represents the combination of hardware, firmware, and software that is responsible for enforcing a security policy. A security kernel or reference monitor is the combination of hardware, firmware, and software that mediates all access to system resources while protecting itself from modification. A security kernel enforced the access control portion of a system security policy. The *Orange Book* required that a security model be verified through a formal mathematical proof at class A1. Today, formal mathematical proofs are also used to verify the correctness of requirements and designs for safety-critical systems.

The Honeywell Secure Communications Processor (SCOMP) Trusted Operating Program (STOP) Release 2.1 was the first product to be rated A1/B3. The final evaluation report,[138] issued September 28, 1985, noted that product ratings had to be tied to operational missions and environments. By October 1997, 106 commercial products appeared on the National Computer Security Center TCSEC evaluated products list, including three at A1 and three at B3.[248]

Access control, authentication, and audit trail were the three cornerstones of COMPUSEC in the early days. For example, the majority of the features listed in Exhibit 5 relate to access control and authentication, especially those listed under Security policy and Accountability. The assurance features correspond to software integrity — ensuring that the software not only functions correctly but that it correctly and reliably enforces the security policy. This differs from COMSEC, which promoted data confidentiality.

Exhibit 5 Summary of *Orange Book* Trusted Computer System Evaluation Criteria (TCSEC)

Category	*Feature Evaluated*	*A1*	*B3*	*B2*	*B1*	*C2*	*C1*
Security policy	Discretionary access control	x	+	x	x	+	+
	Sanitize storage before reuse	x	x	x	x	+	
	Security labels	x	x	+	+		
	Label integrity	x	x	x	+		
	Export labeled information	x	x	x	+		
	Export to multi-level secure devices	x	x	x	+		
	Export to single-level secure devices	x	x	x	+		
	Labeling human-readable output	x	x	x	+		
	Mandatory access controls	x	x	+	+		
	Subject sensitivity labels	x	x	+			
	Device labels	x	x	+			
Accountability	Identification and authentication	x	x	x	+	+	+
	Audit trail	x	+	+	+	+	
	Trusted communications path	x	+	+			
Assurance	System architecture	x	+	+	+	+	+
	System integrity	x	x	x	x	x	+
	Security testing	+	+	+	+	+	+
	Design specification verification	+	+	+	+		
	Covert channel analysis	+	+	+			
	Trusted facility management	x	+	+			
	Configuration management	+	x	+			
	Trusted recovery	x	+				
	Trusted distribution	x					
Documentation	Security features users' guide	x	x	x	x	x	+
	Trusted facility manual	x	+	+	+	+	+
	Testing documentation	+	x	+	x	x	+
	Design documentation	+	+	+	+	x	+

Note: x, no additional requirements for this class; +, new or enhanced requirements for this class; (blank), no requirements for this class.

Access control is defined as:

> design features that protect IA-critical and IA-related systems, applications, and data by preventing unauthorized and unwarranted access to these resources.

Access control can be implemented to control access to networks, computer systems, individual software applications, data, utilities, and shared resources such as printers. 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 or to the resources accessed.[248,357] Examples of access privileges include: read, write, edit, delete, execute, copy, print, move, forward,

Exhibit 6 *Orange Book* Testing Requirements

Class	*Reqt. Type*	*Requirement*
C1	New	The security mechanisms shall be tested and found to work as claimed in the system documentation. Testing shall be done to ensure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms.
C2	Add	Testing shall include a search for obvious flaws that would allow violation of resource allocation, or that would permit unauthorized access to the audit trail or authentication data.
B1	Add	A team of individuals who thoroughly understand the specific implementation of the security protection mechanisms shall subject its design, documentation, source code, and object code to thorough analysis and testing. Their objectives shall be: to uncover all design and implementation flaws that would permit a subject external to the security protection mechanism to read, change, or delete data normally denied under the mandatory or discretionary security policy; as well as to ensure that no subject (without authorization to do so) is able to cause the system to enter a state such that it is unable to respond to communications initiated by other users. All discovered flaws shall be removed or neutralized, and the system retested to demonstrate that they have been eliminated and that new flaws have not been introduced.
B2	Change	All discovered flaws shall be *corrected* and the system retested to demonstrate that they have been eliminated and that new flaws have not been introduced.
	Add	The security protection mechanism shall be found relatively resistant to penetration. Testing shall demonstrate that the security protection mechanism implementation is consistent with the descriptive top-level specification.
B3	Change	The security protection mechanism shall be found *resistant* to penetration.
	Add	No design flaws and no more than a few correctable implementation flaws may be found during testing and there shall be reasonable confidence that few remain.
A1	Change	Testing shall demonstrate that the security protection mechanism implementation is consistent with the *formal top-level specification*.
	Add	Manual or other mapping of the formal top-level specification to the source code may form a basis for penetration testing.

Source: From CSC-STD-001-83, Trusted Computer System Evaluation Criteria (TCSEC), National Computer Security Center, U.S. Department of Defense, August 15, 1983; DOD-5200.28-STD, Trusted Computer System Evaluation (TCSEC), National Computer Security Center, U.S. Department of Defense, December 1985.

distribute, etc. Access control rights and privileges can be defined on a need-to-know basis or a security classification scheme. Access control rights and privileges are generally defined in a matrix format by user name, user roles, local or global user groups. Access control is usually implemented through a

combination of commercial operating system utilities and custom code. Access control provides a first layer of defense in protecting IA-critical and IA-related system resources; it enforces authorization policies.[248] Effective implementation of access control depends on:

1. Taking the time to define a comprehensive set of access control rights and privileges, including permissions to create/change these definitions
2. Protecting the table containing these definitions from unauthorized manipulation and corruption
3. A robust authentication capability

As Denning[248] points out, "Access controls are no better than the authentication mechanism on which they are based."

One area that is often overlooked, to the detriment of security, is inferred access control privileges. Inferred access control privileges are implied subsets or extensions to discrete access control privileges. For example, if someone has the discrete privilege to edit a file, that person also has the inferred privilege to read the file. In contrast, does someone who has the discrete privilege to read a file have the inferred privilege to print it? Perhaps not. Inferred access control privileges can occur by accident if sufficient care is not taken in defining discrete access control rights and privileges. Inferred access control privileges apply to processes as well as data. A determination should be made whether or not a user or process should inherit the privileges of an invoked process.[277] This needs to be decided on a case-by-case basis; however, the operating system should always be protected. Users should rarely have direct access to the operating system. In fact, Gollmann[277] recommends: (1) using status flags to distinguish between user, administrative, and operating system function calls; and (2) applying access controls to specific memory locations to prevent illegal modification of the operating system, application programs, and data. He gives a good illustration of the latter — the need to remove the NT registry editor from all user PCs.[277]

Authentication is defined as:

> establishing, verifying, or proving the validity of a claimed identity of a user, process, or system.

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. Authentication is invoked prior to access control rights and privileges. A combination of parameters can be used to establish an identity, such as user name, password, biometric information, and traffic source. There are weaknesses associated with each of these parameters; thus, it is best to use a combination of parameters and not rely on any one of them alone. For example, Gollmann[277] cites common password vulnerabilities, including:

- Password guessing
- Password spoofing
- Use of default passwords

- Compromised password file
- Web browsers that store previous screens which contain user name and password

Chapter 6 discusses the strengths and weaknesses of different authentication methods.

To protect the user and the system, authentication should be bidirectional; that is, the user should be authenticated to a system and a system should be authenticated to a user. The latter is an important step in preventing site switching and other security compromises while connected to the Internet. A strong authentication strategy is essential for implementing effective access control rights and privileges. The effectiveness of an authentication strategy is determined by: (1) the selection of parameters to be verified, and (2) how stringent the verification process is. The goal is to minimize the number of false positives and false negatives.

An audit trail is defined as:

> a set of records that collectively provide documentary evidence of system resources accessed by a user or process to aid in tracing from original transactions forward and backward from records and reports to their component source transactions.

An audit trail is a design feature that provides an ongoing system monitoring and logging function. An audit trail serves four purposes. First, it captures information about what people and processes accessed what system resources and when they did so. Second, it captures information about system state transitions, the availability and loading of system resources, and the general "health" of the system. When abnormal events are logged, they trigger warnings and alarms so that action can be taken to prevent or minimize the effects of hazardous events. For example, an alarm may trigger the shutdown of an unstable nuclear power plant or the blocking of an intrusion attempt. The alarms may trigger a combination of automatic processes and operator alerts. Third, audit trail data is used to develop normal system and user profiles as well as attack profiles for intrusion detection systems. Fourth, audit trails are used to reconstruct events during accident/incident investigations.

An audit trail provides real-time and historical logs of system states, transitions, and usage. It is essential for safe, secure, and reliable system operation and for performing trend analysis and pattern recognition of anomalous events. 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 to record them. A determination must also be made about the interval at which audit trails should be archived and overwritten.

Historically, COMPUSEC focused on protecting defense or national security information that was stored, processed, and generated on mainframe computers.

Today, COMPUSEC principles have been extended to a wide range of applications and system architectures, including client/server applications operating across LANs, WANs, VPNs, and the Internet. The rapid expansion of and interest in computer security is due to the sensitivity and volume of legal, financial, medical, business, and government data that is stored and processed today. In addition, as Denning[248] notes:

> *Almost any illegal activity that can be committed can be accomplished through the use of a computer, or at a minimum, with the computer as a willing accomplice.*

In the past, access control features were concerned with mediating access to a single system and its resources. Today, firewalls are used to mediate access between networks and the multiple systems and processors connected to them, while intrusion detection systems help prevent attempts to bypass security mechanisms. The audit trail now serves a new purpose — the development of normal system and user profiles as well as attack profiles for use by intrusion detection systems.

3.4 Information Security (INFOSEC)

The *Orange Book* was oriented toward custom software, particularly defense and intelligence applications, operating on a mainframe computer that was the predominant technology of the time. Guidance documents were issued*; however, it was difficult to interpret or apply the *Orange Book* to networks or database management systems. When distributed processing became the norm, additional standards were issued to supplement the *Orange Book*, such as the Trusted Network Interpretation** and the Trusted Database Management System Interpretation***. Each standard had a different color cover and collectively they became known as the rainbow series.

---

* (a) CSC-STD-003-85, Guidance for Applying the Trusted Computer System Evaluation Criteria (TCSEC) in Specific Environments, National Computer Security Center, U.S. Department of Defense (DoD), June 1985.[136]
(b) CSC-STD-004-85, Technical Rationale Behind CSC-STD-003-83, National Computer Security Center, U.S. Department of Defense (DoD), 1985.[137]
(c) NCSC-TG-025 version 2, A Guide to Understanding Data Remembrance in Automated Information Systems (AIS), National Computer Security Center, U.S. Department of Defense (DoD), September 1991.[147]

** (a) NCSC-TG-005 version 1, Trusted Network Interpretation of the TCSEC, National Computer Security Center, U.S. Department of Defense (DoD), July 1987.[144]
(b) NCSC-TG-011 version 1, Trusted Network Interpretation of the TCSEC, National Computer Security Center, U.S. Department of Defense (DoD), August 1, 1990.[145]

***NCSC-TG-021 version 1, Trusted DBMS Interpretation of the TCSEC, National Computer Security Center, U.S. Department of Defense (DoD), April 1991.[146]

At the same time, similar developments were proceeding outside the United States Between 1990 and 1993, the Commission of the European Communities*, the European Computer Manufacturers Association (ECMA)**, the Organization for Economic Cooperation and Development (OECD)***, and the Canadian System Security Centre**** all issued computer standards or technical reports. These efforts and the evolution of the rainbow series were driven by three main factors:

1. The rapid change in technology, which led to the need to merge COMSEC and COMPUSEC
2. The more universal use of information technology (IT) outside the defense and intelligence communities
3. The desire to foster a cost-effective commercial approach to IT security that would be applicable to multiple industrial sectors

The new paradigm combines COMSEC and COMPUSEC and is known as INFOSEC. INFOSEC is defined as:

> the protection of information against unauthorized disclosure, transfer, or destruction, whether accidental or intentional.

The emphasis is on protecting information, which is more refined than data, from accidental *and* intentional malicious actions. This is a broader scope than either COMSEC or COMPUSEC. INFOSEC can be applied to any type of software application, system architecture, or security need.

The current internationally recognized approach to INFOSEC is known as the Common Criteria*****. The Common Criteria are the result of a cooperative effort by Canada, France, Germany, the Netherlands, the United Kingdom, and the United States. The first version was published in January 1996, the second in May 1998. The next step was to promulgate the criteria via an international standard, ISO/IEC 15408 (Parts 1–3), which was approved in 1999******. The goal was to develop a standard by which the security of IT products, systems, and networks could be evaluated such that the evaluation would receive mutual

---

* (a) Information Technology Security Evaluation Criteria (ITSEC), Commission of the European Communities, Provisional Harmonised Criteria, version 1.2, June 1991.
(b) Information Technology Security Evaluation Manual (ITSEM), Commission of the European Communities, 1992.

** Secure Information Processing versus the Concept of Product Evaluation, Technical Report ECMA TR/64, European Computer Manufacturers Association (ECMA), December 1993.

***Guidelines for the Security of Information Systems, Organization for Economic Cooperation and Development (OECD), November 1992.[54]

****The Canadian Trusted Computer Product Evaluation Criteria, Canadian System Security Centre, version 3.0e, 1993.

*****Common Criteria for Information Technology (IT) Security Evaluation, version 2.0, Common Criteria Editing Board (CCEB), May 1998.[52]

******(a) ISO/IEC 15408-1(1999-12) Information Technology — Security Techniques — Common Criteria for IT Security Evaluation (CCITSE) — Part 1: General Model.[120]
(b) ISO/IEC 15408-2(1999-12) Information Technology — Security Techniques — Common Criteria for IT Security Evaluation (CCITSE) — Part 2: Security Functional Requirements.[121]
(c) ISO/IEC 15408-3(1999-12) Information Technology — Security Techniques — Common Criteria for IT Security Evaluation (CCITSE) — Part 3: Security Assurance Requirements.[122]

recognition across national borders. At the same time, system developers gain insight into what features and techniques are important through the publication of the evaluation criteria.

The Common Criteria separate functional security requirements from security assurance requirements.[248] As Caplan and Sanders[238] point out:

> *Functional requirements represent a statement of the security functionality or features a product is intended to provide. Satisfying assurance requirements gives you confidence that the functional requirements have been met.*

This is similar to the distinction between functional safety requirements and safety integrity requirements found in many international standards, for example, IEC 61508-3.[65]

ISO/IEC 15408 is written for use by three different communities: customers, developers, and evaluators. Customers define their IT security requirements, in an implementation-independent fashion, in what is referred to as a protection profile (PP). Developers respond to the PP with an implementation-dependent design, referred to as a security target (ST). Evaluators assess the conformance of the as-built system or product, referred to as the target of evaluation (TOE), to requirements stated in the PP*. Many U.S. government agencies are required to follow the Common Criteria methodology; PPs are often included in procurement announcements and offerors are required to submit an ST in response as part of their proposal.

ISO/IEC 15408 defines a standard set of functional security classes, families, and components, as shown in Exhibit 7. A customer selects the appropriate items from this set to define their functional security requirements.

Likewise, a standard set of security assurance classes, families, and components are defined, as shown in Exhibit 8. Security assurance requirements are grouped according to evaluation assurance levels (EALs). A customer specifies the required EAL.

Security assurance provides grounds for confidence that an IT product or system meets its security objectives. The Common Criteria philosophy is to provide assurance based on an evaluation (active investigation) of the IT product or system that is to be trusted. Evaluation techniques can include, but are not limited to[120–122]:

- Analysis and checking of processes and procedures
- Checking that processes and procedures are being applied
- Analysis of the correspondence between TOE design representations
- Analysis of the TOE design representation against the requirements
- Verification of proofs
- Analysis of guidance documents
- Analysis of functional tests developed and the results provided
- Independent functional testing

* Sample PPs are posted on the NIAP and IATF Web sites.[451,471]

Exhibit 7 ISO/IEC 15408-2 Functional Security Classes and Families

Class	*Family*
Security audit (FAU)	Security audit automatic response (FAU_ARP) Security audit data generation (FAU_GEN) Security audit analysis (FAU_SAA) Security audit review (FAU_SAR) Security audit event selection (FAU_SEL) Security audit event storage (FAU_STG)
Communication (FCO)	Nonrepudiation of origin (FCO_NRO) Nonrepudiation of receipt (FCO_NRR)
Cryptographic support (FCS)	Cryptographic key management (FCS_CKM) Cryptographic operation (FCS_COP)
User data protection (FDP)	Access control policy (FDP_ACC) Access control functions (FDP_ACF) Data authentication (FDP_DAU) Export to outside TSF control (FDP_DAU) Information flow control policy (FDP_ITC) Information flow control functions (FDP_IFF) Import from outside TSF control (FDP_ITC) Internal TOE transfer (FDP_ITT) Residual information protection (FDP_RIP) Rollback (FDP_ROL) Stored data integrity (FDP_SDI) Inter-TSF user data confidentiality transfer protection (FDP_UCT) Inter-TSF user data integrity transfer protection (FDP_UIT)
Identification and authentication (FIA)	Authentication failures (FIA_AFL) User attribute definition (FIA_ATD) Specification of secrets (FIA_SOS) User authentication (FIA_UAU) User identification (FIA_UID) User-subject binding(FIA_USB)
Security management (FMT)	Management of functions in TSF (FMT_MOF) Management of security attributes (FMT_MSA) Management of TSF data (FMT_MTD) Revocation (FMT_REV) Security attribute expiration (FMT_SAE) Security management roles (FMT_SMR)
Privacy (FPR)	Anonymity (FPR_ANO) Pseudonymity (FPR_PSE) Unlinkability (FPR_UNL) Unobservability (FPR_UNO)

- Analysis for vulnerabilities (including flaw hypothesis)
- Penetration testing

The validity of documentation and the resulting IT product or system is measured by expert evaluators with increasing emphasis on scope, depth, and rigor. Greater assurance results from the application of greater evaluation

Exhibit 7 ISO/IEC 15408-2 Functional Security Classes and Families (continued)

Class	*Family*
Protection of the TSF (FPT)	Underlying abstract machine test (FPT_AMT) Fail secure (FPT_FLS) Availability of exported TSF data (FPT_ITA) Confidentiality of exported TSF data (FPT_ITC) Integrity of exported TSF data (FPT_ITI) Internal TOE TSF data transfer (FPT_ITT) TSF physical protection (FPT_PHP) Trusted recovery (FPT_RCV) Replay detection (FPT_RPL) Reference mediation (FPT_RVM) Domain separation (FPT_SEP) State synchrony protocol (FPT_SSP) Timestamps (FPT_STM) Inter-TSF TSF data consistency (FPT_TDC) Internal TOE TSF data replication consistency (FPT_TRC) TSF self-test (FPT_TST)
Resource utilization (FRU)	Fault tolerance (FRU_FLT) Priority of service (FRU_PRS) Resource allocation (FRU_RSA)
TOE access (FTA)	Limitation on scope of selectable attributes (FTA_LSA) Limitation on multiple concurrent sessions (FTA_MCS) Session locking (FTA_SSL) TOE access banners (FTA_TAB) TOE access history (FTA_TAH) TOE session establishment (FTA_TSE)
Trusted path/channels (FTP)	Inter-TSF trusted channel (FTP_ITC) Trusted path (FTP_TRP)

effort, and the goal is to apply the minimum effort required to provide the necessary level of assurance. This increasing level of effort is based on[120–122]:

1. **Scope:** the portion of the IT product or system included in the evaluation
2. **Depth:** the level of design and implementation detail evaluated
3. **Rigor:** the application of effort in a structured, formal manner

ISO/IEC 15408 defines seven evaluation assurance levels (EALs), as shown in Exhibit 9. EALs represent the degree of confidence that functional security requirements have been met, with EAL 7 being the highest rating and EAL 1 the lowest. Depending on how a product or system is designed, built, and evaluated, it could be rated anywhere from EAL 1 to EAL 7. However, it is unlikely that a product or system could be rated EAL 4 or higher without prior planning and preparation to receive such a rating. It is possible for variations of a product, targeted for diverse customers, to receive different ratings.

Exhibit 8 ISO/IEC 15408 Security Assurance Classes and Families

Class	*Family*
Protection profile evaluation (APE)	TOE description (APE_DES)
	Security environment (APE_ENV)
	PP introduction (APE_INT)
	Security objectives (APE_OBJ)
	IT security requirements (APE_REQ)
	Explicitly stated IT security requirements (APE_SRE)
Security target evaluation (ASE)	TOE description (ASE_DES)
	Security environment (ASE_ENV)
	ST introduction (ASE_INT)
	Security objectives (ASE_OBJ)
	PP claims (ASE_PPC)
	IT security requirements (ASE_REQ)
	Explicitly stated IT security requirements (ASE_SRE)
	TOE summary specification (ASE_TSS)
Configuration management (ACM)	CM automation (ACM_AUT)
	CM capabilities (ACM_CAP)
	CM scope (ACM_SCP)
Delivery and operation (ADO)	Delivery (ADO_DEL)
	Installation, generation, and start-up (ADO_IGS)
Development (ADV)	Functional specification (ADV_FSP)
	High-level design (ADV_HLD)
	Implementation representation (ADV_IMP)
	TSF internals (ADV_INT)
	Low-level design (ADV_LLD)
	Representation correspondence (ADV_RCR)
	Security policy modeling (ADV_SPM)
Guidance documents (AGD)	Administrator guidance (AGD_ADM)
	User guidance (AGD_USR)
Life-cycle support (ALC)	Development security (ALC_DVS)
	Flaw remediation (ALC_FLR)
	Life-cycle definition (ALC_LCD)
	Tools and techniques (ALC_TAT)
Tests (ATE)	Coverage (ATE_COV)
	Depth (ATE_DPT)
	Functional tests (ATE_FUN)
	Independent testing (ATE_IND)
Vulnerability assessment (AVA)	Covert channel analysis (AVA_CCA)
	Misuse (AVA_MSU)
	Strength of TOE security functions (AVA_SOF)
	Vulnerability analysis (AVA_VLA)
Maintenance of assurance (AMA)	Assurance maintenance plan (AMA_AMP)
	TOE component categorization report (AMA_CAT)
	Evidence of assurance of maintenance (AMA_EVD)
	Security impact analysis (AMA_SIA)

Exhibit 9 Summary of Common Criteria for IT Security Evaluation Assurance Levels (EALs)

Level	*Evaluation Mode*	*Use Scenario*	*Degree of Confidence*
EAL 7	Formal design verification and testing	Suitable for extremely high-risk scenarios; highest security	Highest
EAL 6	Semi-formal design verification and testing	Suitable for high-risk loss scenarios; very high security	
EAL 5	Semi-formal design and testing process	High security	
EAL 4	Methodical design, testing, and review process; independent security evaluation	Medium security	
EAL 3	Methodical testing	Moderate security	
EAL 2	Structural testing	Minimal security	
EAL 1	Functional testing; no security evaluation	No security	Lowest

Sources: Adapted from Caplan, K. and Sanders, J., *IT Professional*, 1(2), 29–34, 1999; Denning, *Information Warfare and Security*, Addison-Wesley, 1999; ISO/IEC 15408-1, 15408-2, 15408-3 (1999-12), Information Technology — Security Techniques — Common Criteria for IT Security Evaluation (CCITSE) — Parts 1, 2, and 3.

ISO/IEC 15408[120–122] permits tailoring of functional security and security assurance requirements through a standard three-step process. This is the decision-making process a customer or developer will follow when developing their proposed approach:

1. Standard classes, components, and activities are identified. This information is derived from ISO/IEC 15408 Parts 2 and 3.[121,122]
2. Each of these classes, components, and activities is examined to determine if they are directly applicable to the specific project. This analysis is strictly a yes/no function. The project team may decide that a component or an activity is not applicable because of the nature of the project. If so, an adequate rationale for deleting a component or activity should be provided. In summary, this step analyzes how *explicit* requirements could be satisfied using standard components and activities.
3. Potential areas for augmenting or extending standard requirements are analyzed. The customer or developer may propose additional components or activities to meet project specific requirements. Augmentation refers to adding standard components to an EAL that are normally associated with a higher EAL. Extension refers to adding new project-specific components to a standard functional security or security assurance class. This step focuses on meeting *implied* requirements, which unfortunately are often overlooked.

A parallel effort, known as the systems security engineering capability maturity model or SSE-CMM*,**, was initiated by the U.S. National Security Agency (NSA), Office of the (U.S.) Secretary of Defense (OSD), and the Communications Security Establishment (Canada) in April 1993. ISO/IEC 15408 is primarily an assessment of a product's (or system's) functional security. In contrast, SSE-CMM is primarily an assessment of the security engineering process used to develop a product or system. The intent is to provide a standardized assessment that assists customers, such as NSA, DoD, and CSA, determine the ability of a vendor to perform well on security engineering projects.

SSE-CMM was derived from the systems engineering capability maturity model (SE-CMM). Additional specialized security engineering needs were added to the model so that it incorporates the best-known security engineering practices.[148] SSE-CMM follows the same philosophy as other CMMs, by identifying key process areas (KPAs) and five increasing capability levels:

0 — not performed
1 — performed informally
2 — planned and tracked
3 — well defined
4 — quantitatively controlled
5 — continuously improving

SSE-CMM identifies eleven security engineering key process areas, as summarized below. Each process area is further subdivided into base practices.

Security Engineering KPAs

PA01 — administer security controls
PA02 — assess impact
PA03 — assess security risk
PA04 — assess threat
PA05 — assess vulnerability
PA06 — build assurance argument
PA07 — coordinate security
PA08 — monitor security posture
PA09 — provide security input
PA010 — specify security needs
PA011 — verify and validate security

Method, version 2.0, April 16, 1999.[149]

In summary, a potential vendor is rated 0 to 5 in each of the eleven security engineering KPAs. An overall rating is given based on the security engineering KPAs and other organizational and project management factors. The customer then determines if the vendor's rating is appropriate for their specific project.

* Systems Security Engineering Capability Maturity Model (SSE-CMM) Model Description Document, version 2.0, April 1, 1999.[148]

** Systems Security Engineering Capability Maturity Model (SSE-CMM) Appraisal Method, version 2.0, April 16, 1999.[149]

Version 2.0 of the SSE-CMM and the associated appraisal method were issued in April 1999. The next step is to promulgate SSE-CMM as an ISO/IEC standard. The latest information about the SSE-CMM can be found at www.sse-cmm.org or www.issea.org.[500,501]

3.5 Operations Security (OPSEC)

Operations security or OPSEC is defined as:

> the implementation of standardized operational procedures that define the nature and frequency of interaction between users, systems, and system resources, the purpose of which is to: (1) maintain a system in a known secure state at all times, and (2) prevent accidental or intentional theft, destruction, alteration, or sabotage of system resources.

As the name implies, OPSEC addresses security concerns related to the operation of a system. OPSEC is more involved with personnel issues, staff responsibilities, and duties than other security measures. OPSEC considers both insider and outsider threats. To illustrate, one historical OPSEC requirement was known as "man-in-the-loop." This operational requirement stated that electronic messages (and sometimes hardcopy printouts) had to be reviewed by a person, to verify that security markings were correct, before they could be released or forwarded. The information was considered too sensitive to rely on automatic processing alone.

Exhibit 10 lists examples of items that should be considered when developing OPSEC procedures. These items fall into three categories: personnel operations, software/data operations, and administrative operations. These procedures should be well-defined, communicated to all stakeholders, and in place before a system is initialized. Note that this list is not exhaustive; there are also many site-specific issues to consider.

In addition to security clearances and background checks, it is important to ensure that staff members have the appropriate education and experience to perform their assigned duties. A person who has little or no understanding of security is unlikely to take it seriously, recognize or respond to potential security problems correctly. This need was recognized in the *Orange Book*, which specified requirements for personnel who conducted security testing. Increasing requirements were specified for each division. For example, the requirements for Division B were[135]:

> The security testing team shall consist of at least two individuals with Bachelor degrees in Computer Science and at least one individual with a Master's degree in Computer Science. Team members shall be able to follow test plans prepared by the system developer and suggest additions, shall be conversant with the flow hypothesis or equivalent security testing methodologies, shall be fluent in the TCB implementation language(s), and shall have assembly-level programming experience. Before testing begins, the team members shall have functional knowledge of, and shall have completed the system developer's internals course for the system being evaluated. At least one team member shall have previously completed a security test on another system.

Exhibit 10 Examples of Items to Address in OPSEC Procedures

1. Personnel Operations (user, system administrator, trainee, maintenance staff, visitors, etc.)
   a. Security clearances, background checks, badges
   b. Proof of staff competence
   c. Searching briefcases, purses, backpacks, etc. when entering/leaving building
   d. Training staff on security features and responsibilities
   e. Defining policy on working alone, after hours, from home, or while traveling (remote access)
   f. Defining policy for taking computers, reports, electronic files out of the office
2. Software/Data Operations (text, image, audio, video)
   a. Schedule for performing system and data integrity checks, backups, generating archives
   b. Schedule and procedures for deleting and disposing of sensitive material, electronic and hardcopy
   c. Procedures for off-site storage
   d. Labeling classified or sensitive data while it is stored, processed, displayed, transmitted, or printed
   e. Defining policy for storing diskettes and other media
   f. Controlling access to archives
   g. Defining audit trail archival and overwrite procedures and schedule
   h. Defining policy for reusing electronic storage media
   i. Procedures and schedule for executing virus scan software on servers and user workstations
   j. Procedures and schedule for updating virus scan software
   k. Site and application specific software and data operations
3. Administrative Operations
   a. Defining hours system can be accessed, and types of transactions that can be done during those hours
   b. Scheduling preventive maintenance
   c. Defining conditions that should trigger an emergency shutdown, automatic or operator assisted, of a communications node, system resource, or the entire system
   d. Defining policy on whether or not PCs should be turned off while someone is away from their desk, at lunch, overnight, and the use of screen savers and privacy screens
   e. Defining how often passwords and other authentication data should be changed and verified
   f. Defining how often access control rights and privileges should be reviewed and updated
   g. Defining procedure for terminating user accounts normally and on an emergency basis
   h. Schedule and procedures for performing security inspections, security assessments, and safe checks
   i. Schedule and procedures for changing combinations
   j. Property pass procedures
   k. Schedule and procedures for managing the distribution, generation, update, storage, replacement, and revocation of cryptographic material and other security tokens

In other words, general-purpose software engineering skills alone were considered inadequate.

OPSEC was relatively straightforward in the days of mainframes and computer centers; it has become much more complex today given mobile computing, telecommuting, client/server applications, and Internet applications.

3.6 System Safety

System safety is defined as[143]:

> *the application of engineering and management principles, criteria, and techniques to achieve acceptable mishap risk, within the constraints of operational effectiveness, time, and cost, throughout all phases of the system life cycle.*

The term "mishap risk" is used to distinguish between type of risk of concern to system safety and that of schedule or cost risk. Mishap risk is defined as[143]:

> *an expression of the possibility and impact of an unplanned event or series of events resulting in death, injury, occupational illness, damage to or loss of equipment or property (physical or cyber), or damage to the environment in terms of potential severity and probability of occurrence.*

As shown in Exhibit 11, system safety is composed of several components. The exact combination of components will vary from system to system. For this book, software safety is the primary component of concern. Software safety is defined as[288]:

> *design features and operational procedures which ensure that a product performs predictably under normal and abnormal conditions, and the likelihood of an unplanned event occurring is minimized and its consequences controlled and contained; thereby preventing accidental injury or death, environmental or property damage, whether intentional or accidental.*

Software is generally categorized as being safety-critical, safety-related, or nonsafety-related. These terms are defined as follows[288]:

- **Safety-critical software:** Software that performs or controls functions which, if executed erroneously or if they failed to execute properly, could directly inflict serious injury to people, property, and/or the environment or cause loss of life.
- **Safety-related software:** Software that performs or controls functions which are activated to prevent or minimize the effect of a failure of a safety-critical system.
- **Nonsafety-related software:** Software that performs or controls functions which are not related to safety.

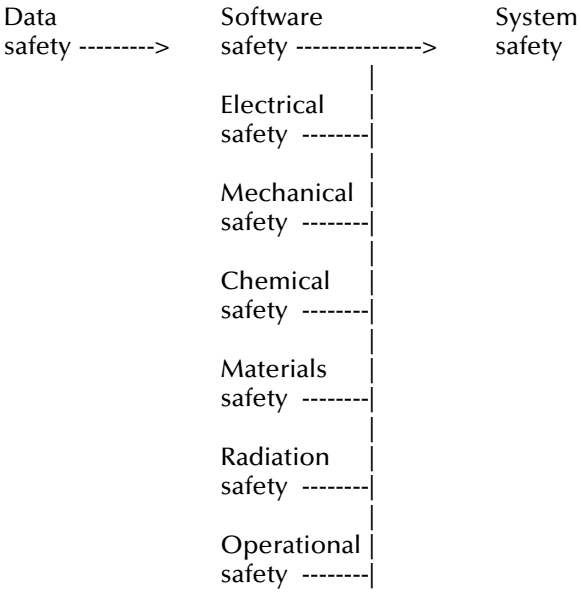


Exhibit 11 Software as a Component of System Safety. (*Source:* Herrmann, D., *Software Safety: The Medical Perspective*, Invited Tutorial, 16th International System Safety Society Conference, September 14–19, 1998, Seattle, WA.)

To illustrate, a software-controlled automobile braking system is classified as safety-critical. A software-controlled air bag deployment system is classified as safety-related. And a software-controlled automobile sound system is classified as nonsafety-related.

The discipline of system safety and software safety originated in the defense and aerospace industries. MIL-STD-882 has been the foundation of system safety for the U.S. military. The original standard was issued in 1969. Revision A was published in 1977, revision B in 1984, and revision C in 1993. MIL-STD-882D,[143] the current version, was adopted in 1999. MIL-STD-882D and its predecessors focus on mishap risks associated with the development, test, acquisition, use, and disposal of DoD weapon systems, subsystems, equipment, and facilities.[143] These standards assigned three types of tasks and activities: safety program management, risk analysis, and risk control, as shown in Exhibit 12.

MIL-STD-1574A (USAF)*, a tailored version of MIL-STD-882A, was developed especially for space and missile systems. Although issued in 1979, MIL-STD-1574A made some observations that are equally applicable today:

> Accident prevention is of major concern throughout the life cycle of a system. Planning and implementation of an effective system safety program, commensurate with the requirements of each phase in the acquisition process, is of prime importance in minimizing risk of accidents and their associated cost impacts during the systems test and

* MIL-STD-1574A, System Safety Program for Space and Missile Systems, U.S. Air Force (USAF), August 15, 1979.

Exhibit 12 System Safety Tasks and Activities Required by MIL-STD-882D

Safety Program Management
- 102 System Safety Program Plan
- 104 Safety Reviews and Audits
- 105 Safety Working Group
- 106 Hazard Tracking

Risk Analysis
- 201 Preliminary Hazard List
- 202 Preliminary Hazard Analysis, Functional FMECA
- 204 Subsystem Hazard Analysis, Design FMECA
- 205 System Hazard Analysis, Interface FMECA
- 206 HAZOP Studies

Risk Control
- 203 Safety Requirements
- 301 Safety Assessments
- 302 Safety Testing
- 303 Safety Review of ECRs and SPRs
- 401 Safety Verification
- 402 Safety Compliance Assessment

Source: From MIL-STD-882D, Mishap Risk Management (System Safety), U.S. Department of Defense (DoD) Standard Practice (draft), October 20, 1998.

> operational phases. System safety responsibilities shall be an inherent part of every program and the implementation of the complete system program requires extensive participation and support by many disciplines and functional areas.

In other words, prior planning is necessary if safety is to be achieved. Second, safety tasks and activities are ongoing throughout the system life cycle. Third, safety engineering should be an integral part of the system engineering process. Finally, effective safety engineering requires an interdisciplinary approach. These four principles are true for reliability engineering and security engineering as well.

The purpose of safety engineering is to manage mishap risks. This is accomplished through a combination of analysis, design, and verification activities, such as those discussed in Annex B, as well as operational procedures. A series of hazard analyses are performed throughout the life cycle to identify risks, their causes, the severity of the consequences should they occur, and the likelihood of them occurring. Risks are then eliminated or controlled through inherent safe (re)design features, risk mitigation or protective functions, system alarms and warnings, and comprehensive instructions for use and training that explain safety features, safety procedures, and the residual risk.

As the quote above stated, the first step is to develop a system safety plan and a corresponding software safety plan. The plan explains the tasks and activities to be performed, the schedule with key milestones and decision points, and the roles and responsibilities of the different stakeholders and the

coordination of their efforts. At the same time, a system safety case and a software safety case are begun. A safety case is a systematic means of gathering and reporting the data needed by contractual, regulatory, and certification authorities to certify that a system has met specified safety requirements and is safe for use in the intended operational environment. Assumptions, claims, evidence, and arguments form the basis of a safety case. A safety plan and a safety case complement each other; the plan states what is intended to be done while the case proves that it was done. A safety case is a living document throughout the system life cycle.

To be achieved, safety requirements must be specified — both the functional safety requirements and the safety integrity requirements. These requirements explain how a system should prevent, detect, respond to, contain, and recover from hazards so that the system remains in a known safe state at all times. This involves specifying must work functions (MWFs) and must not work functions (MNWFs),[126,127] under what conditions a system should fail safe or fail operational, and the time required to safe or shutdown a system before corrective action can be taken.[439]

Since its beginning in the defense and aerospace industries, the need for software safety has expanded to most industrial sectors, including the railway, automotive, power generation, commercial aircraft, air traffic control systems, process control, and biomedical industries. A new application is intelligent transportation systems (ITS). As Jesty[308] points out, software will play a major role in:

- Providing pre-trip information
- Providing route guidance
- Performing demand management and traffic control functions
- Assisting emergency vehicle management
- Monitoring the transportation of HAZMAT

Another reasonably new application is marine navigation systems in which software is responsible for integrating and supplying correct, current, and understandable real-time information from multiple sources. Mills[356] notes the concomitant challenges:

> *In safety-critical situations, the information must be correct and readily available in an instantaneously understandable form. … this is not always the case if cycling through screens is necessary or information such as symbols has to be interpreted. However, there is another problem in that when information is integrated the choice of which information is redundant is often made at the system/chip level so that the user has no idea what has been discarded.*

There are many parallels between safety and security engineering. Security engineering speaks in terms of vulnerabilities and threats, while safety engineering speaks in terms of risks and hazards. In both instances, the intent is to: (1) prevent accidental or malicious intentional actions that could have

negative consequences; (2) minimize or eliminate the probability of unintended or unspecified functionality; and (3) keep the system in a known safe or secure state at all times. Many of the same techniques are used for both safety and security engineering, as shown in Annex B; the difference is the perspective from which the techniques are applied and the results interpreted. For example, access control can be employed to restrict access to sensitive information *and* to prevent unauthorized users from initiating safety-critical functions.[333,422] The concept of defense in depth is employed in both safety and security engineering. The dual usage of Formal Methods was mentioned earlier. There are also many parallels between physical security and physical safety; operational security and operational safety.

3.7 System Reliability

System reliability is the composite of hardware and software reliability predictions or estimations for a specified operational environment. Hardware reliability is defined as:

> the ability of an item to correctly perform a required function under certain conditions in a specified operational environment for a stated period of time.

Software reliability is defined as[288]:

> *a measure of confidence that the software produces accurate and consistent results that are repeatable, under low, normal, and peak loads, in the intended operational environment.*

Hardware is primarily subject to random failures, failures that result from physical degradation over time and variability introduced during the manufacturing process. Hardware reliability is generally measured quantitatively. Software is subject to systematic failures, failures that result from an error of omission, an error of commission, or an operational error during a life-cycle activity.[288] Software reliability is measured both quantitatively and qualitatively. To illustrate, a failure due to a design error in a memory chip is a systematic failure. If the same chip failed because it was old, that would be considered a random failure. A software failure due to a design or specification error is a systematic failure. Hence, system reliability measurements combine quantitative and qualitative product and process assessments.

Reliability engineering emerged as an engineering discipline in earnest following World War II. The defense and aerospace industries led this development; other industries such as the automotive, telecommunications, and consumer electronics became involved shortly thereafter. Initial efforts were focused on components, then subsystems, and systems. A variety of statistical techniques were developed to predict and estimate system reliability. Failure data was collected, analyzed, and shared over the years so that the techniques

could be improved. The notion of software reliability did not begin until the late 1970s.

Early software reliability models tried to adapt hardware reliability models. They applied statistical techniques to the number of errors found during testing and the time it took to find them to predict the number of errors remaining in the software and the time that would be required to find them. Given the difference in hardware and software failures, the usefulness of these models was mixed. The limitations of early software reliability models can be summarized as follows[288]:

1. They do not distinguish between the type of errors found or predicted to be remaining in the software (functional, performance, safety, reliability, etc.).
2. They do not distinguish between the severity of the consequences of errors (insignificant, marginal, critical, catastrophic) found or predicted to be remaining in the software.
3. They do not take into account errors found by techniques other than testing (e.g., static analysis) or before the testing phase.

These limitations led to the development of new software reliability models and the joint use of qualitative and quantitative assessments.

The purpose of reliability engineering is to ensure that a system and all of its components exhibit accurate, consistent, repeatable, and predictable performance under specified conditions. A variety of analysis, design, and verification techniques, like those discussed in Annex B, are employed throughout the life cycle to accomplish this goal. Current and thorough user documentation is an important part of this process because it will explain the correct operation of a system, applications for which the system should and should not be used, and procedures for preventive, adaptive, and corrective maintenance.

As in safety engineering, the first step is to develop a system reliability plan and a corresponding software reliability plan. Similarly, a system reliability case and a software reliability case are begun. A reliability case demonstrates that a system has met specified reliability requirements and is fit for use in the intended operational environment.

Reliability requirements are specified at the system level, then allocated to system components such as software. A series of analyses, feasibility studies, and trade-off studies are performed to determine the optimum system architecture that will meet the reliability requirements. A determination is made about how a system should prevent, detect, respond to, contain, and recover from errors, including provisions for degraded mode operations. Progress toward meeting reliability goals is monitored during each life-cycle phase.

One of the more interesting and promising new developments in this field is the application of Bayesian Belief networks (BBNs) to model system and software dependability. BBNs are graphical networks that represent probabilistic relationships among events or propositions. Bouissou, Martin, and Ourghanlian[221]; Niel, Littlewood, and Fenton[361]; and Neil and Fenton[360] describe several advantages of BBNs:

1. They can be used as a decision aid in the context of uncertainty.
2. They have the ability to combine different types of information: inference, evidence, and expert judgment.
3. They improve communication among different stakeholders.
4. They address known risks, unexpected and unknown results and effects.
5. The probabilities are updated as new knowledge or uncertainty is propagated through the network.

Agena, Ltd., has reported several successful BBN projects. In one project, BBNs were used to predict software defects in consumer digital electronic products[387]:

> The defect prediction BBN models the process of defect insertion and discovery at the software module level. It will be used to predict the number of residual defects, and defect densities, at various life-cycle phases and with various different types of assumptions about the design and testing process.

A second project involved assessing the reliability of military vehicles during all life-cycle phases. A tool composed of five modular BBNs was developed for this project[431]:

1. A Bayesian updating BBN to predict the reliability of subsystems using failure data from historically similar subsystems
2. A recursive BBN used to combine subsystem reliability probability distributions together to achieve a vehicle-level prediction
3. A design-quality BBN used to estimate design unreliability caused by a variety of design process factors
4. A manufacturing-quality BBN used to estimate unreliability caused by poor-quality manufacturing processes
5. A vehicle testing BBN that uses failure data gained from vehicle testing to infer vehicle reliability

Other applications of BBNs reported by Agena, Ltd., include[431]:

1. Assessing risks associated with specific new programmable electronic system components for the transportation industry
2. Modeling risk in air traffic control systems
3. Automated test case generation and software reliability forecasting for the telecommunications industry
4. Operational risk forecasting for the financial and insurance industries
5. Modeling expected jury reasoning for criminal trials

In addition, research is underway to determine if BBNs can be used effectively to predict intrusion-detection profiles prior to an attack.

There are several parallels between reliability engineering and safety or security engineering. Reliability engineering speaks in terms of failure modes and failure rates. In this instance, the term “failure” encompasses all types of

failures, including security compromises and safety violations. The goal of all three disciplines is to prevent, detect, contain, and recover from erroneous system states and conditions. However, reliability engineering does not place as much emphasis on intentional malicious actions as safety or security engineering. Integrity and availability are major concerns of reliability engineering, just as they are for safety and security engineering. Reliability engineering activities are performed throughout the life cycle at the system and software level. As shown in Annex B, many of the same analysis, design, and verification techniques are used by safety, reliability, and security engineering. For example, a combined FTA/FMECA can be used by a reliability engineer to determine failure modes and rates. A safety engineer can use the same analysis to identify potential hazardous failures and the risk mitigation/control measures needed. A security engineer can use the same analysis to identify potential failures that could lead to security compromises. Again, the difference is the perspective from which the techniques are applied and the results interpreted. This concept is developed further in Chapters 4 through 8.

3.8 Summary

This chapter reviewed the seven historical approaches to information security/IA: physical security, communications security (COMSEC), computer security (COMPUSEC), information security (INFOSEC), operations security (OPSEC), system safety, and system reliability. Each of these seven approaches served a different purpose, as summarized in Exhibit 13. A variety of techniques were used by these historical approaches to achieve and maintain information confidentiality, data and system integrity and availability. Some techniques were used by multiple approaches, as shown in Exhibit 14. Although many parallels existed between these approaches, there was a lack of formal coordination and communication among them. For the most part, these activities were performed in isolation; at best, there was limited ad hoc coordination.

All of the approaches have had to evolve and need to continue evolving to correspond to changes in technology and operational environments, profiles, and missions. In the early days, physical security, COMSEC, COMPUSEC, and OPSEC were designed around the concept of a mainframe computer in a secure computer center. The advent of distributed processing, PCs, and LANs led to the initiation of INFOSEC, which merged/superseded COMSEC and COMPUSEC. Originally, system safety and system reliability gave nominal consideration to software. Today, software is a major component of safety engineering and reliability engineering.

At present, almost all systems, particularly infrastructure systems, mission-critical systems, and business-critical systems, have a combination of safety, reliability, and security requirements. A system may have high security, medium reliability, and no safety requirements, or a system may have high safety, high reliability, and medium security requirements. To illustrate:

Exhibit 13 Summary of the Different Roles Played by Historical Approaches to Information Security/IA

Type of IA Activity	*Role or Purpose*
Physical security	Protect system resources from physical damage that could impair operations and services. Protect physical system resources from theft.
Communications security (COMSEC)	Protect the confidentiality, integrity, and availability of sensitive data while it is being transmitted between systems and networks.
Computer security (COMPUSEC)	Prevent, detect, and minimize the consequences of unauthorized actions by users (authorized and unauthorized) of a computer system.
Information security (INFOSEC)	Protect information against unauthorized disclosure, transfer, or destruction, whether accidental or intentional.
Operations security (OPSEC)	Implement standardized operational procedures that define the nature and frequency of interaction between users, systems, and system resources; the purpose of which is to: (1) maintain a system in a known secure state at all times, and (2) prevent accidental or intentional theft, destruction, alteration, or sabotage of system resources.
System safety	Achieve acceptable mishap risk, within the constraints of operational effectiveness, time, and cost throughout all phases of the system lifecycle.[143]
System reliability	Achieve correct functional performance under certain conditions in a specified operational environment for a stated period of time.

- A financial system has high security, medium reliability, and no safety requirements.
- A database of medical records has medium safety, reliability, and security requirements.
- A spy satellite has high security and reliability requirements and low safety requirements. (It should not be able to malfunction and crash in an inhabited area.)
- An air traffic control system has high safety and reliability requirements and medium security requirements.
- An automobile has high safety, high reliability, and low security requirements. (An unauthorized person should not be able to tamper with the onboard PLCs or embedded software.)

Hence, it is essential that safety, reliability, and security engineering efforts be systematically coordinated and integrated. This is the realm of information security/IA.

Next, Chapter 4 explains how to identify what systems and data need to be protected and why.

Exhibit 14 Summary of the Techniques Used by Historical Approaches to Information Security/IA

Type of IA Activity	*Confidentiality Measures*	*Integrity Measures*	*Availability Measures*
Physical security	Isolating data of different classification levels on different channels Shielding equipment and cables Controlling physical access to equipment Remote equipment identified by location		Specialized HVAC UPS Protecting equipment from natural disasters Disk mirroring Off-site storage
Communications security (COMSEC)	Encryption Spectrum management Secure switch isolation	Error detection/ correction algorithms Formal proofs of correctness	Redundant communication equipment Alternate communication paths
Computer security (COMPUSEC)	Access control Authentication Audit trail Process isolation Labeling	Partitioning Information hiding	Trusted recovery
Information security (INFOSEC)	Encryption Spectrum management Secure switch isolation Access control Authentication Audit trail Process isolation Labeling Protection against accidental and intentional actions	Error detection/ correction algorithms Formal proofs of correctness Partitioning Information hiding Protection against accidental and intentional actions EALs	Redundant communication equipment Alternate communication paths Trusted recovery
Operations security (OPSEC)	Personnel operations Data operations Administrative operations	Data operations Administrative operations	Data operations Administrative operations
System safety	Access control	Error detection/ correction algorithms Plausibility checks Defensive programming Hazard analyses Formal proofs of correctness Partitioning Information hiding SILs	Defense in depth Block recovery Fail safe or fail operational

Exhibit 14 Summary of the Techniques Used by Historical Approaches to Information Security/IA (continued)

Type of IA Activity	*Confidentiality Measures*	*Integrity Measures*	*Availability Measures*
System reliability		Error detection/ recovery algorithms Fault tolerance FTA/FMECA Reliability allocation	Reliability block diagrams Reliability estimation and prediction Block recovery Degraded mode operations

3.9 Discussion Problems

1. What role does intrusion detection play in physical security?
2. Describe the role software reliability serves in protecting data confidentiality and availability.
3. Develop a physical security plan for a home-based online business. The business has three mini-tower computers in two different geographic locations, one printer, and a notebook computer that is taken to trade shows. There are five employees, two of which live in the house. The house has two phone lines. The online business is run out of the lower level of the house.
4. What different or additional physical security measures, if any, should be taken for home offices and mobile computing compared to a business office, and vice versa?
5. The XYZ company uses online e-forms to capture, update, and store personnel records, such as address, title, salary, bonuses, and performance appraisals. Develop an access control strategy that accommodates these five user groups: employee, first level supervisor, second level supervisor, personnel officer, and marketing manager.
6. How is the strength of an encryption algorithm measured?
7. What kind of authentication is needed for the following scenarios: remote access, mobile computing, distributed work groups, remote help/diagnostics?
8. What is the difference, if any, between security classification schemes based on security levels and those based on need-to-know? Which of these two approaches accommodates compartmentalization?
9. Describe the role encryption serves in protecting data integrity.
10. Why should or should not access control features be implemented for safety-critical software? For reliability-critical software?
11. What role does an audit trail play in protecting: (a) security-critical software, (b) safety-critical software, and (c) reliability-critical software? What role does an audit trail play in investigating an incident related

to the failure or compromise of: (a) security-critical software, (b) safety-critical software, and (c) reliability-critical software?

12. What are the major components of an OPSEC plan? What does each address?
13. What parallels exist between: (a) safety and reliability engineering, (b) safety and security engineering, and (c) security and reliability engineering?
14. Which of the seven historical approaches to IA considered: (a) accidental actions, and (b) malicious intentional actions?
15. Describe the differences and similarities between the Common Criteria and the SSE-CMM.
16. What benefit does a customer gain by requiring a vendor to be rated at a specific SSE-CMM level? What competitive advantage or disadvantage does a vendor gain by obtaining an SSE-CMM rating?