Chapter 2

Information Security

This chapter provides a general explanation of information security (IS) and its major components. Firstly, a definition and brief introduction of IS is given. Following this, the fundamental IS services as defined by the ISO's Information Security Architecture, together with the most common techniques used to implement these services, are discussed. Lastly, an important IS service that does not form part of the ISO's IS architecture is discussed.

2.1 What is Information Security?

Information security can be briefly defined as the act of shielding all computer resources including data, software, and the network from any unauthorised access or use (Hannon, 1998). This also includes ensuring the integrity of a computer system's data, and if needed, ensuring the accountability for a user's actions. The process of ensuring that a computer system is secure involves the management of certain factors that can minimise the risk of unauthorised access by an entity (the word *entity* will be used to refer to a user, software component, or other computer system).

The importance of IS has continually increased in the last decade. This has been mainly due to the transformation of the Internet and associated developments such as e-commerce, WWW advertising, and Internet banking. All of these networked systems are widely designed to allow many entities to perform specific functions such as electronic money transfers, order placements, information gathering, marketing activities, etc. It is therefore clear that a required level of IS is needed, depending on the case at hand, to ensure that these activities are performed within a secure environment.

The main assets of a computer system are hardware, software and data. A computer system has four types of security threats which exploit weaknesses in these assets. The first threat, *interruption*, occurs when one of the assets become unavailable, inoperative, or gets lost. Examples of this is when a hardware device is intentionally damaged or when a software program is accidentally deleted. Another type of security threat is *interception*, which occurs when an unauthorised entity gains access to an asset e.g. password sniffing over a network which allows an unauthorised user to be "legitimately" authenticated on the system. The third type of threat, *modification*, occurs when an unauthorised entity makes changes or deletions to an asset e.g. when a user modifies a financial transaction being sent over a network. The last type of threat is *fabrication*, and involves the unauthorised addition of counterfeit objects on a computer system e.g. the unauthorised insertion of a transaction in a bank's database (Pfleeger, 1997).

IS has evolved into a broad field within the Computer Science domain, and is based on theoretical principles or services (as characterised by the original ISO 7498/2 document that defines an Information Security Architecture). This document forms the basis for internationally accepted IS Architectures, and a particular system/environment is generally considered secure if these services are properly implemented. The following diagram illustrates the components of the ISO's Information Security Architecture:

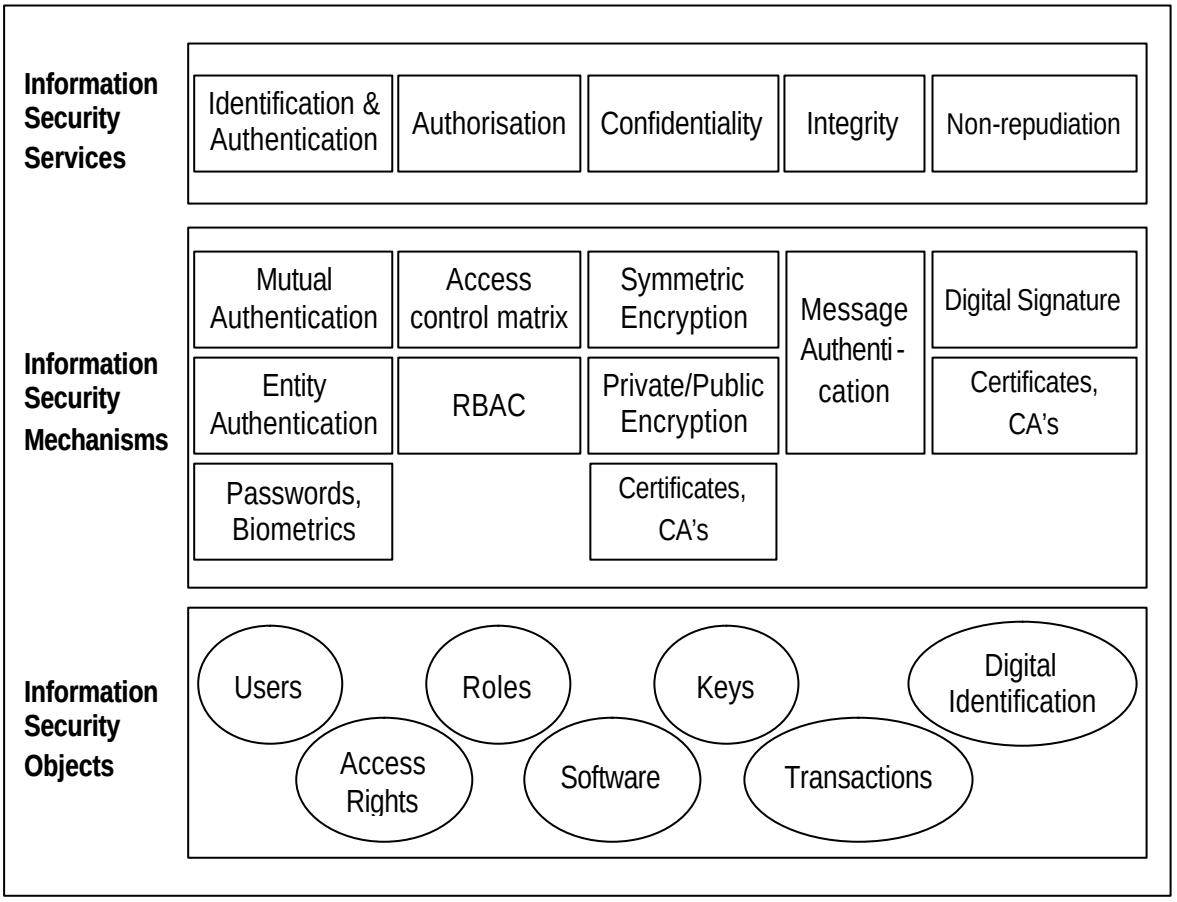

Figure 2.1: ISO 7498/2 Information Security Architecture (Adapted from von Solms & Eloff, 1999)

As can be seen from the above diagram, the architecture defines five services, namely: Identification and Authentication, Authorisation (or Logical Access Control), Confidentiality, Integrity, and Non-repudiation (or Non-denial). IS management therefore involves ensuring that these services are correctly implemented and continuously maintained. Each of these 5 services contain mechanisms that are used to implement the service. For example, passwords and biometrics are mechanisms used to implement the service of identification and authentication (von Solms & Eloff, 1999).

These five ISO defined services constitute the five fundamental building blocks of IS. There is, however, another IS service which is not part of the ISO's Information Security Architecture, but is nevertheless considered important. This "6th service" is availability, and will be briefly discussed after the five fundamental ISO services. The service of availability ensures that all software, hardware (including the network), and data are at the disposal of any authorised user when needed. It is becoming an increasingly important IS service, especially recently due to the large number of denial of service attacks.

2.2 The five ISO Information Security Services

A particular computer system does not necessarily require all five IS services to be implemented in order to ensure its security. The combination of IS services used depends very much on what level of security is required for that particular system. A large company may for example need to communicate (via a network) with its amalgamated companies. The large company may require authenticated identification from the merged companies, as well as encrypted communication to ensure secrecy of the data being sent over the network. The logon username and password may however be the same for all the merged companies, since it may be a case where the larger company does not need to know which merged company (once authenticated) provides feedback on any information given (especially when anonymity is required). The IS service of non-repudiation is therefore not needed in this scenario. More complex systems may however require all the IS services to be implemented (especially if data is to be sent over a public network).

2.2.1 Identification and Authentication

Identification is when an entity provides a characteristic that is unique within the system (such as a name), which can be used to control access to the system. Although this characteristic is unique within the system, it does not necessarily have to belong exclusively to that entity (e.g. it could be the name of a group that the entity belongs to). It is however more common to provide a unique logon name or number (or combination of these) to each entity within a system. This is a very important IS service, since most systems need to be able to identify which entity/group is making use of its services in order to ensure proper security measures (for example to ensure that only authorised entities gain access to the

system, and to ensure proper access control measures). This service is therefore required in most cases as a prerequisite to the other four ISO services discussed in this chapter.

A computer system can be classified according to its environment and how it identifies its entities. A computer system that is in a *closed environment* is one that contains a list of pre-registered entities. When an entity requests access to the system, the system will first check to ensure that the entity exists in the pre-registered list (if not, access is denied). Alternatively, a computer system that is in an *open environment* is one that does not contain a list of pre-registered entities (often because the design characteristics of the system does not make it possible for an entity to be known before an interaction is necessary). This is often the case, for example, with e-commerce transactions. A user can order a product online without being registered on that system, but must however be properly identified and held accountable for the purchase before delivery of the product (often via credit cards). This section will mainly refer to identification and authentication in a closed environment; section 2.2.5 (Non-repudiation) will discuss identification and authentication in an open environment.

Authentication is the process of associating some form of proof that an entity is who he/she/it claims itself to be. This process therefore assures the computer system being logged onto that the entity claiming to have the stated identity is not an impostor. This is done by providing something that only the entity/group should have access to (such as a password). When correctly implemented, the process of identification and authentication ensures that only legitimate users of the computer system acquire access to it (Hannon, 1998).

This IS service is often the first and the last line of defence in a computer system. This is because an entity is usually first identified and authenticated before the other IS services proceed, and it is the last line of defence because the other IS services rely on the correct output of this service for them to be properly implemented (if this service is compromised, the other services may well be compromised too). It is therefore very important to have an authentication service that is highly available and secure.

Entity authentication occurs when only one side needs to assure the other of its claim of identity. In contrast, mutual authentication occurs when both sides mutually assure each other of their claims of identity. In a typical client-server architecture, a number of clients are connected to a single authentication server. To increase availability and security however, a number of servers can be used to

provide a distributed authentication service. In this case, the servers often communicate using shared keys together with checksums (for integrity), which allows the servers to share the responsibility of authentication. Using this approach, a small number of compromised servers should not compromise the entire authentication service (Gong, 1993).

2.2.1.1 Authentication Mechanisms

There are three different categories of mechanisms used to provide the proof needed during the identification and authentication process. Firstly, an entity can be authenticated by *something it knows*, such as a password or Personal Identification Number (PIN). An entity can also be authenticated by *something it possesses*, such as a smart card. Lastly, *some unique attribute of the entity* can be used to authenticate it, such as a fingerprint or facial characteristics. A combination of these mechanisms may also be used to enhance the level of security (Hendry, 1995). These mechanisms will now be further discussed.

(a) Something the entity knows

The most common mechanism used in this category is a password or PIN. A user name or identifier is associated with a secret piece of information that should only be known to the entity or group of entities associated with that identifier. The secret information is a string of characters that should be securely recorded together with its associated identifier in a password repository.

An entity requiring access to the system would need to provide its identifier and password/PIN to successfully be granted access to the system. Once successfully logged onto the system, the entity's access rights can be determined (which will be discussed in more detail in section 2.2.2). A number of safety measures should be taken to improve the security of a password/PIN (thereby decreasing the chance of it being guessed, or obtained via a dictionary or brute force attack). For example, a good password should be at least eight characters in length, should be alphanumeric, should contain special characters, and should avoid dictionary words. This authentication mechanism will correctly function *only* if the password or PIN is known by the relevant entity/entities associated with that identity exclusively, and if it is securely stored and transmitted.
A similar mechanism within this category is zero knowledge tests. A single fixed question that is asked to an entity may not be very secure if an attacker can guess or somehow find out the answer to it. It is however far less probable that an

attacker would answer a series of unknown questions correctly. The *zero knowledge principle* is therefore used, which entails asking an entity enough questions to ensure the system that the claimed identity actually belongs to it (Hendry, 1995). This mechanism is often used in Web-based services such as email, typically in the case when the account's password is forgotten.

(b) Something the entity possesses

In this case, the entity must use a hardware device or physical token as part of the identification and authentication process. Examples of such devices include hand-held password generators, magnetic cards, electronic keys, and smart cards. The entity usually inserts the token/card into a hardware device which will read data embedded within it. The entity's identification is usually read from it, after which the entity is (usually) required to enter in a password/PIN associated with that identity. Using this approach, an attacker requires both the token/card and the password/PIN to gain access to the system (this combination therefore improves the identification and authentication process).

Magnetic cards are commonly used by banks to allow their clients access to their accounts via ATM's (combined with a PIN). A current example of a hand-held password generator is the SecurID token developed by Security Dynamics Technologies, Inc. which generates unique and unpredictable access codes every sixty seconds, to be used together with a user's PIN. Once entered into the system, the access code and PIN is compared with the server's code before allowing the user access to the system (Shim, Qureshi & Siegel, 2000).

A password/PIN entered by an entity is compared with the correct password/PIN that can be embedded on the token/card *or* on the server(s). The former approach will not be secure if a magnetic card is used, because the password/PIN can be read or duplicated with relative ease. A smart card can however be used in this case, because a password/PIN can be sealed within its ROM chip. A smart card has a built-in CPU and memory, allowing encrypted data to be transferred and decrypted by the smart card. The data does not need to be decrypted before transferring it to the card, and it is encrypted before leaving the card, allowing the former approach to be implemented securely. The declining cost of smart cards together with the advances in smart card technology have caused an increasing demand for this authentication mechanism, growing at a rate of about 40 percent per year (Dreifus & Monk, 1998).

(c) Some unique attribute of the entity

Human beings can be identified using techniques that measure a person's unique physical characteristics such as their fingers, eyes, and voice. These techniques include finger and palm print scans, iris scans (the coloured portion of the eye), voice pattern recognition, facial recognition, and DNA scans. The constant growth in available processing power allows this form of identification and authentication to be more feasible and effective. For example, Sukthankar & Stockton (1999) present ARGUS, an identification system which is specifically designed to automatically notify the residents of a building of any guest arrivals by using facial recognition to tell the visitors apart. The entrance to the building is constantly monitored via a security camera. Motion detection algorithms are used together with a neural-network-based face detector, which provides the input for a memory-based face recognition system called ARENA. Once a visitor is identified (after comparison with a local database), the relevant resident can be notified through a messaging system. ARGUS uses a number of machines to perform the processor intensive computations on the images, and has proven successful after several months of testing.

All the above mentioned biometrics are converted into an equivalent digital form and stored either at a remote location or a secure local repository (such as a smart card). When authentication is needed, the derived biometric of the user claiming to have a certain identity is compared with the stored biometric for that identity. If they match, the user is successfully authenticated. This technique is more secure than the passwords/PIN approach because it is generally very difficult to reproduce biometrics.

An entity's behaviour can also be used as part of the authentication process. The manner in which an entity logs on to a system can be monitored, as well as the activities performed after access to the system is granted e.g. a user requesting a file that would not be requested by the authentic user may be immediately brought to the attention of the administrator. The way in which a user types in his/her identity and password/PIN can be analysed. Brown & Rogers (1994) present a method which analyses keystroke patterns that can be used, together with the password/PIN mechanism, to assist in authenticating a user. Three techniques were used to test the method with real data: a statistical technique (distance variation), the ADALINE and the backpropagation neural network. This method ensures a 0% impostor pass rate (IPR) by adjusting the threshold (for each technique) and allowing the false alarm rate (FAR) to increase as needed to

ensure this. This follows the logic that most system administrators would prefer false alarms to impostor passes.

2.2.2 Authorisation

An entity has now been successfully authenticated, what next?
A computer system is rarely designed to allow all its entities equal access rights on each and every system resource. The computer system must therefore retrieve and control the access rights for each authenticated entity. The process of ensuring that all actions performed are legitimate (i.e. conform with the security policy) is known as authorisation. A simple computer system may retrieve all access rights for that entity after identification and authentication occurs. It is however more common, especially for complex computer systems, that each entity's access rights be retrieved only when a specific action is requested (such as the request for a file). An important "law" of security in general is that an entity should only receive the minimum amount of access to a system required to accomplish its specific tasks. This applies to IS too, and should be kept in mind when the security policy is designed and implemented.

Access control of data is often performed at the file, volume, or resource level i.e. it is often the case where an entity or group of entities are designated access rights for the entire file, volume, or resource. Although it is not commonly utilised by system administrators, certain computer programs or systems (such as operating systems) allow access control to be performed at a smaller level, such as a record, field, variable, byte or even bit. This level of detail is however very difficult to manage, and is normally performed by a programming language or operating system exclusively (Madron, 1992).

A number of issues need to be addressed when planning this IS service for a particular computer system. Firstly, all *authorisation information* must be defined which is specific to that computer system and environment, including objects, groups or departments, subjects, and access rights. Secondly, the security policy must be designed to include an *authorisation policy*. This will contain policy decisions regarding this particular IS service, such as whether or not an entity is allowed to decide which other entities or groups have access to its own data. Thirdly, an *authorisation mechanism* that will be used to implement this service must be developed (von Solms & Eloff, 1999). These three issues will now be further discussed.

2.2.2.1 Authorisation Information

One of the first tasks to be performed when implementing this service is to define the objects, subjects, and access rights relating to that particular computer system, since each system contains its own composition and user requirements. The *objects* in the system are those components that provide a particular service to the entities e.g. a file or hardware device. The entities that make a request to the system for a particular service are known as *subjects* e.g. people or application programs. A relationship is therefore formed between the object and the subject. These relationships are known as *access rights*, and they represent the types of actions that an entity can perform on a specific object (depending on what actions are defined and allowed for that system) e.g. read access on a particular directory or execute access on a file. Authorisation therefore ensures that all subjects on that computer system only perform actions on objects that are within their assigned access rights.

Management of these access rights can be difficult if a relationship must be manually defined between every subject and object. Most systems therefore do allow default access rights to be defined e.g. an owner of a file has the default 'read/write' access rights, while everyone else defaults to 'no access' for that file. This may not help the situation much though because a large number of users may need access to that file, and therefore access rights need to be provided for each of these users anyway. A common approach is to define *roles*, *departments*, or *security classifications*, and to assign entities to one or more of these. A role can therefore be defined for a group of entities that have a similar set of responsibilities or functions, such as accounts or database administration. Access rights can now be defined between objects and roles, departments, or security classifications instead of individual subjects e.g. all army generals for a specific country have full access to a top-secret directory.

2.2.2.2 Authorisation Policies

An authorisation policy must be used to decide who has the authority to assign and change access rights for the objects in the computer system. There are two categories that authorisation policies can be divided into. The first is *mandatory access control* (MAC), which means that policy decisions regarding access control is made by a central authority. The second category, *discretionary access control* (DAC), allows the owner or another authorised entity to make access control decisions regarding certain objects.

MAC allows certain standards regarding access control to be enforced, and is usually performed by a central group of individuals (e.g. the administrators) and/or the system itself (using preset defaults). Access rights regarding an entity's objects can therefore conform with other entities that have the same role or classification. An individual entity does not therefore directly have a say as to which of its objects are accessible and by whom. A military security policy typically uses the MAC approach, preventing a low classification soldier from changing a secret document to public (even if that belongs to him/her). DAC allows any entity to decide who or what has access to its objects, as well as what access rights are associated with them; the entity therefore has more authority over its data. Although this approach is very useful, an entity must be careful not to give incorrect access rights to another entity, because this could lead to disastrous consequences! The DAC approach is typically used for large web servers, allowing each user to specify object access rights for their web pages.

It is clear that each category has its advantages and disadvantages. The best approach therefore depends on the security requirements for that particular computer system. A combination of the above two approaches may however be preferred, especially for larger systems with different needs within its sub-systems or departments (Pfleeger, 1997).

2.2.2.3 Authorisation Mechanisms

An authorisation mechanism needs to be developed, which ensures that all access requests within the system are checked for legitimacy. The most popular mechanisms used generally fall into two categories, depending on which authorisation policy approach is used for that system (DAC or MAC).

Authentication mechanisms that form a part of the DAC category are:

- access control matrixes (ACMs)
- directories, and
- access control lists (ACLs).

An ACM is a table in which each column represents an object, each row represents a subject, and each intersection or entry represents the access rights between that object and subject. A simple example of an ACM can be seen below in Table 2.1. Each user of the system has his/her own folder containing files. To keep things simple, only read/write access is defined, and the entire folder is treated as a single object (each file would normally be treated as an object).

	Admin files	*Joe's folder*	*Bob's folder*	*Jill's folder*	*Paul's folder*
Administrator	read/write	read/write	read/write	read/write	read/write
Joe (Jill's friend)		read/write		read	
Bob			read/write		
Jill (Joe's friend)		read		read/write	
Paul					read/write

Table 2.1: ACM example

As can be seen from the above example, an ACM contains many blank entries and may not be very efficient to implement for a specific system. A directory is a listing of all objects that are accessible by a specific subject. Each subject in the system will therefore be associated with a list of all its accessible objects, together with the access rights for each object. An ACL on the other hand is a listing of all subjects that have access to a specific object. Each object in the system will therefore be associated with a list of all subjects that have access to it, together with their access rights.

Authentication mechanisms that form a part of the MAC category are:

- the role-based access control (RBAC) model
- the military model
- the Bell and LaPadula model, and
- the schematic protection model (SPM).

RBAC models are ideal for environments where its entities are associated with defined roles, such as account debiting, office administration, and high-level management. Each role is then assigned access rights to the system's objects, instead of each subject. The military model is used to associate access rights between an object and a security classification. Each subject is given a security classification and can only read/write to objects equal to or below its classification.

The Bell and LaPadula model is similar to the military model. As with the military model, a subject can only read objects whose classification is lower than or equal to its own. The difference with this model is that a subject can only write to objects whose classification is higher than or equal to its own. An SPM has two main properties. Firstly, a user cannot directly or indirectly create itself as a new user in the system. Secondly, the set of access rights assigned to a new user is always a subset of its creator's access rights. This property therefore prevents a user from having higher access rights than its creator (Muftic, Patel, Sanders, Colon, Heijnsdijk & Pulkkinen, 1993).

2.2.3 Confidentiality

An entity has now been successfully authenticated, and the authorisation IS service has been implemented on the system to ensure that the entity only performs legitimate actions. As it stands, a potentially serious problem still exists with this system. A user may for whatever reason want more access to the system than that given to him/her, and may try to intercept network traffic in order to get the administrator (or another user's) password. Even worse, this user may find the user database and retrieve the passwords for all users! If this is possible, the above two IS services are potentially useless. A legitimate user may also not want his/her personal data and messages to be accessed and viewed by other entities (especially sensitive data such as bank transactions).

The above scenario indicates that there is a definite need to ensure that certain data can only be viewed by authorised users, especially when being transmitted over a network or hardware device (e.g. when in memory), and during storage. This is accomplished by changing the data into a form that is deceiving or unintelligible to other entities; this process is called *encryption*. The process used to convert the encrypted data back to its original form is called *decryption*. Encryption is a very powerful mechanism used to implement the IS service discussed in this section, *confidentiality*, which if used correctly allows only authorised entities to view specified data.

Encryption can also be used as a mechanism to implement a part of the remaining two ISO IS services. This section will however concentrate on the actual process of encryption used to change the visibility of data. It should also be noted that data can be encrypted by replacing one word for another using *codes*, or by transforming the data into an unreadable form using *ciphers*. The most applicable to information security is to use ciphers, and will therefore be further discussed in this section (encryption using codes will not be discussed; for further discussion on this topic see Pfleeger, 1997).

The encryption process uses the original data (the *plaintext*) as the input for an encryption algorithm which produces the encrypted data (the *ciphertext*) as the output. The ciphertext may then be used as the input for the decryption algorithm in order to reproduce the plaintext. Two types of encryption are currently used: *symmetric encryption* and *asymmetric encryption*. Symmetric encryption is when the decryption algorithm (including its components) is similar to the encryption algorithm used (the operations performed are usually the inverse of the encryption algorithm's operations). An asymmetric decryption algorithm (together with its

components) is however significantly different to the encryption algorithm used (together with its components). These two types will now be discussed in more detail in the following subsections.

2.2.3.1 Symmetric encryption

At first, encryption was performed using an algorithm only i.e. no components were used together with the encryption algorithm. The sender of a message encrypts the message using the encryption algorithm and transmits it to the receiver. An entity that intercepts the ciphertext will not be able to view the original message without first decrypting it. The receiver must be in possession of the decryption algorithm, which is used to convert the ciphertext back into the original message. Similarly, any data can be encrypted before storage and decrypted when retrieved using these algorithms (which applies generally to all methods of encryption). In the above traditional process, the algorithms used for encryption and decryption are related – this is therefore a form of symmetric encryption. The following diagram illustrates this process:

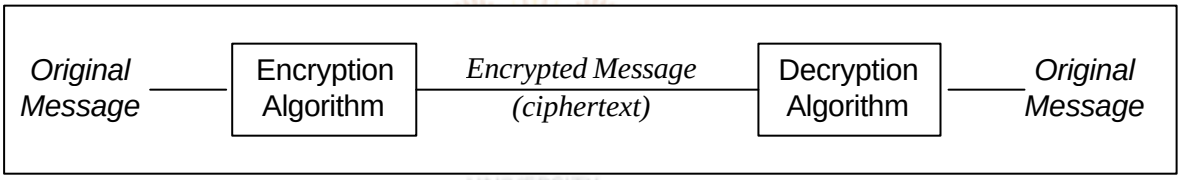

Figure 2.2: The traditional symmetric encryption process

The traditional process worked well for situations involving a small number of people. The problem with this process is that the actual algorithms used had to be kept secret; no entity aside from the sender and receiver(s) can have access to the algorithms in order for the data to be kept confidential. This means that an entity requiring confidential and exclusive communication with many entities must use a different algorithm for each receiver.

An easier approach is to use a standard encryption and decryption algorithm together with one or more extra components that uniquely encrypt the data. These components are unique and easy-to-manage additions to the encryption process which allow the algorithm to be made public (therefore promoting standardisation), requiring that only these unique components be kept secret when a symmetric encryption algorithm is used. Commonly, the extra component is a distinctive string of bits called a *key*, which is used extensively in the encryption process to ensure that only that particular key correctly decrypts the ciphertext. When a symmetric encryption algorithm is used, the key used in the encryption process is

the same as the key used in the decryption process. The public algorithm remains the same for both processes, and it is for this reason that the key(s) used must be kept secret (Stallings, 1999). The following diagram illustrates this process:

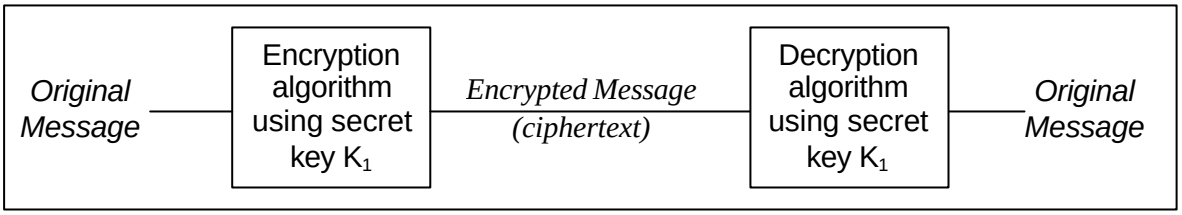

Figure 2.3: The key-based symmetric encryption process

The most common internationally accepted symmetric algorithm used is the Data Encryption Standard (DES). The DES algorithm has a block size of 64 bits and uses a 64-bit key to encrypt and decrypt the data, even though only 56 bits are actually used (because eight bits are dropped initially). The algorithm consists of sixteen cycles of permutations and substitutions, with each cycle producing outputs that is used as inputs for the next cycle. Although it is internationally accepted, the DES has recently been shown to have flaws due to its 56-bit key length and the increasing growth of available processing power (at the same cost). An improvement of the DES, Triple DES, encrypts the data using the DES first with one key, then with a second key (using the ciphertext of the first as data input), and finally with a third key (using the ciphertext of the second as data input). Another encryption algorithm called the Advanced Encryption Algorithm (AES) has recently been standardised which has a block size of 128 bits, a key length of 128, 192, or 256 bits, and a cryptographic life expectancy of more than twenty years (Graff, 2001).

2.2.3.2 Asymmetric encryption

The above encryption standards are used in many applications including the financial sector and in business-to-business communication. One of the difficulties with symmetric encryption is to securely distribute the key(s) which will be used to encrypt and decrypt the data on both sides. If an unauthorised entity possesses the key(s) used, it can view any ciphertext that has been encrypted with that key(s). Common approaches to distribute the key(s) include sending it over couriers, certified mail, and secure networks. A further hassle is that even if the actual algorithm can be made public, each exclusive communication line requires its unique key to be managed and distributed.

Key-based asymmetric encryption/*public key encryption* is the solution to these problems. Public key encryption makes use of two keys, and goes one step further than symmetric encryption by allowing the algorithm *and* one of the keys to be made public. Each entity owns two keys, a *private key* and a *public key*. The private key is kept secret, only the owner must have access to this key. The public key is made available to anyone in need of that particular public key via trusted key storage repositories. The keys are both unique within the entire system – no other entity can have the same private or public key.

The decryption and encryption algorithms used are related, just as the public and private keys belonging to a particular entity are mathematically related (even though they are significantly different). When the private key that is assigned to an entity is used to encrypt data, the corresponding public key must be used to decrypt the data. Similarly, data that has been encrypted using a particular public key must be decrypted using the corresponding private key. This type of encryption is asymmetric because data that has been encrypted with an entity's public key cannot be decrypted with that same key (and similarly, a private key cannot be used to encrypt *and* decrypt). An entity can therefore encrypt data with another entity's freely available public key knowing that only the authorised entity would be able to decrypt the data using its private key (assuming that the key was not compromised) (Ahuja, 1996). The following diagram illustrates this process:

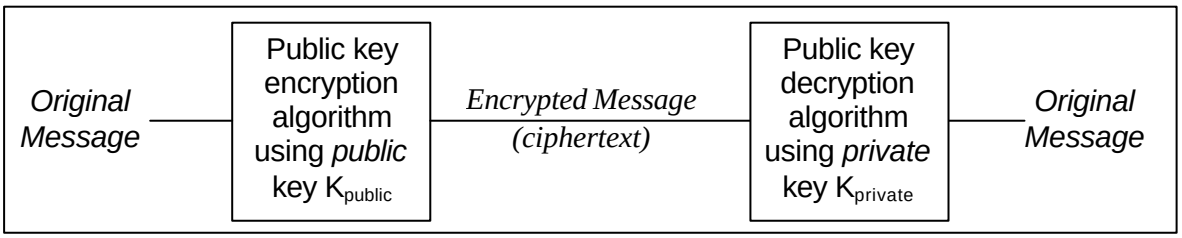

Figure 2.4: The public key encryption process (to ensure confidentiality)

When an entity wants to send an encrypted document or message to another entity using public key encryption, all it requires is the receiver's public key. This has helped eliminate the key distribution problem because the public key is freely available. The only remaining problem is that the entity must be sure that the public key really belongs to the intended recipient. This problem is solved by verifying the public key with a trusted entity called a certification authority (CA), which supplies a certificate containing the public key and other verified information specific to the owner of the public key. The CA therefore encrypts the certificate using its own private key, allowing an entity to decrypt the verified certificate using the CA's public key (which is usually distributed securely with the public key encryption software) (Oaks, 2001).

The most common internationally accepted public key encryption algorithm used is the Rivest-Shamir-Adelman (RSA) algorithm. The RSA algorithm was first developed in 1978, and after extensive cryptanalysis has still remained secure. The RSA algorithm is based on finding the prime factors of large numbers used to determine the public and private keys. This factorisation process is exponential in time, causing the encryption and decryption of large messages or documents to be rather slow. For this reason, it is common not to encrypt an entire message or document using public key encryption. Instead, symmetric encryption is used to encrypt the message or document, and only the symmetric key used is encrypted using public key encryption. This method therefore allows efficient encryption and decryption, and still solves the key distribution problem (Pfleeger, 1997).

2.2.4 Integrity

A computer system that has the above three IS services correctly implemented ensures that its entities are identified and authenticated, that they perform only legitimate actions, and that specified data can only be viewed by the authorised entities. An important aspect that has been left out is that an unauthorised entity can modify the data (whether accidentally or deliberately) without the system even detecting these unauthorised changes. Even if the data is encrypted before transmission, an unauthorised entity may for whatever reason decide to intercept and slightly modify the ciphertext and retransmit it (even though the entity may not know what it is changing), which could result in the acceptance of an (undetected) illegitimate change on the receiver's end.

The IS service of integrity ensures that only legitimate modifications have been made to specified data. This service guarantees the receiving entity that an object is in its original state, aside from any authorised changes that have been made to it (whether in storage or transmission). Stated in another manner, the *authenticity* of the object is tested and if it is legitimate, the receiving entity is ensured that the integrity of the object has been maintained.

An *authenticator* or Message Authentication Code (MAC) is used as a mechanism to provide the IS service of integrity. An authenticator is an addition to an object that is used when verifying the integrity of the object (i.e. the object's authenticity). It can be thought of as a unique value for that object that is calculated based on the exact composition of the object (e.g. the entire text of a message to be sent

over a network). The algorithm used to calculate the authenticator is (ideally) designed to produce a different authenticator if any part of the object is altered.

An authenticator for a particular object is created and attached to the object before storage or transmission. An entity retrieving the object uses the same algorithm to recalculate the authenticator for it. If the attached and recalculated authenticators match, the object is considered authentic (if not, the object is usually discarded and further action may be taken). This technique is often made more secure by calculating the authenticator for a plaintext object and attaching it to the object after encryption, thereby making it more difficult for an interceptor in possession of the algorithm to correctly recalculate the authenticator after illegitimate changes have been made. As a further improvement, it is common to use another key as part of the authenticator algorithm which is shared with the receiving entity (von Solms & Eloff, 1999).

2.2.4.1 Authenticator algorithms

There are two main types of authenticator algorithms, hash functions and checksums. Each of these two types decrease the amount of data to a much smaller amount, called a *digest* or *check value*. This process is however not reversible (the authenticator does change if the original data changes, but cannot be used to reproduce the original data).

A hash function is a complex algorithm that determines the output by dividing the original data into smaller (n-bit) blocks, which are processed by the algorithm in sequence to generate the digest. The algorithm is shared between the sender and receiver, but is usually not kept secret. For this reason, an unauthorised entity could intercept a transmission and make modifications to it in a manner that these changes would not alter the digest, or even recalculate the digest after modifying the data. To prevent this, a cryptographic hash function can be used which includes a cryptographic function as part of the hash function. Only the sender and authorised receiver(s) would have access to the cryptographic function, making it improbable that an unauthorised entity would make modifications without being detected.

The two most popular hash functions in use are the MD5 message digest algorithm and the Secure Hash Algorithm (SHA). MD5 is a fast one-way hash function which produces 128-bit hashes (MD5 replaced the MD4 hash algorithm due to minor weaknesses). SHA is a variant of the MD5 algorithm which produces

160-bit hashes, and was specifically designed to be used in the Digital Signature Standard (discussed in the next section).

A checksum is a less complex algorithm which also determines the check value by processing the data in blocks as input. A checksum often uses a secret key together with its algorithm, which is shared between the sender and receiver. Hash functions are however more commonly used to provide integrity on the application level (between two or more entities during transmission and storage). Checksums are often used to provide error detection and correction for network transmissions on a lower level e.g. parity checking (Stein, 1998).

2.2.5 Non-repudiation

Repudiation occurs when an entity that takes part in a communication denies involvement in that communication, whether in part or in whole. Traditionally, paper-based transactions and agreements formed a critical role within our social and economic environment. A method had to be developed to ensure that a participating entity could not deny ever taking part in the transaction/agreement. The entities involved must therefore manually write their signature on the document, which can be analysed and verified by experts if a dispute occurs. The entity's signature therefore serves to hold that person accountable for all statements specified within that document.

An important ISO IS service that has not yet been discussed is the service of *non-repudiation*. A computer system may also need to ensure that an entity can be held accountable for its actions i.e. the service of non-repudiation is required. In today's digital world, an entity cannot simply sign its signature (by hand) for a digital transaction. These transactions cannot be performed face-to-face, which means that less physical evidence is available. We must therefore find another method to enforce non-repudiation for electronic transactions, agreements, etc. (Zhou and Gollmann, 1996).

A *digital signature* can be used to provide this IS service. A digital signature is a unique attachment for a digital object that can only be made by a specific entity i.e. no other entity can forge this entity's digital signature. Any entity should also be able to determine without doubt which entity digitally signed the object. Symmetric key encryption cannot therefore be used to digitally sign a document or message because two or more entities share a key i.e. even in the case where only two entities are communicating, the receiving entity cannot prove that it did

not send itself the message (even though it may know that the other entity actually sent the message).

Public key encryption can however be used to implement the IS service of non-repudiation. An entity can encrypt a message or other data using its *private* key, allowing any entity in possession of its public key to decrypt the data. The sender would also include its identity together with the object so that the receiver would know which public key to use to decrypt it. Non-repudiation can now be enforced because only the legitimate owner of the private key could have encrypted the message. This is because no other entity has access to its private key (unless it was compromised), and only that particular entity's public key correctly decrypts the ciphertext. Since this process cannot be forged and the legitimate owner can be identified, non-repudiation is assured (Madron, 1992). The following diagram illustrates this process:

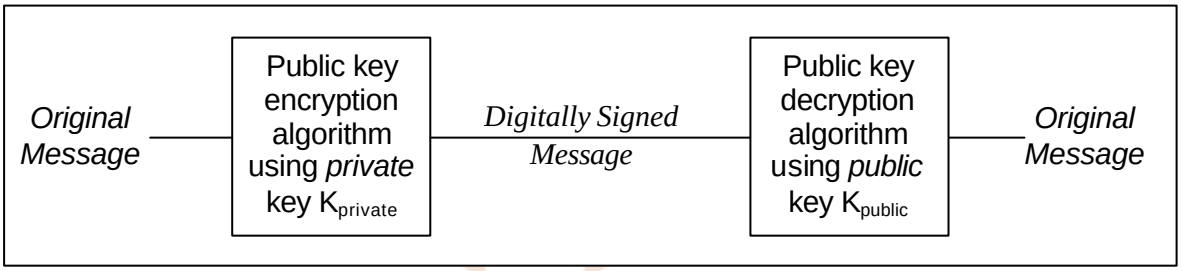

Figure 2.5: The public key encryption process (to ensure non-repudiation)

Any data that is digitally signed using a private key can be read by anyone in possession of that entity's public key. In order to ensure confidentiality, the sender must encrypt the data using the receiver's public key after digitally signing the data with its own private key. The receiving entity will then first decrypt the data with its own private key, and then complete the process by decrypting the (digitally signed) data with the sender's public key. Using this process, the IS services of non-repudiation and confidentiality are ensured.

As mentioned earlier, the encryption of large objects using public key encryption exclusively can be rather slow. Realistically, a digital signature is created by hashing the data and digitally signing the hash only, which is attached with the data. If necessary, the data and digitally signed hash can be encrypted using a faster symmetric encryption process. The symmetric key used can then be encrypted using the receiver's public key. All these components are then put together before storage or transmission. To prevent the entire data from being sent again by an unauthorised entity, a time stamp can also be encrypted and included with the data. This should prevent the receiving entity from accepting the

data if the time stamp is invalid e.g. if the time is not within predefined bounds or if an incrementing integer has previously been accepted.

Any public key encryption algorithm can be used to produce digital signatures. The National Institute of Standards and Technology (NIST) have however specified an algorithm specifically to create digital signatures called the Digital Signature Algorithm (DSA). The DSA was designed to be used in its Digital Signature Standard (DSS), and uses a variable key length of between 512 and 1024 bits (Stein, 1998).

2.3 The IS service of availability

The ISO IS services are mainly concerned with ensuring that only authorised entities have access to specified data, that only legitimate actions are performed, and if necessary that an entity is made accountable for its actions. Although these five IS services help to ensure that an unauthorised entity does not for example delete or otherwise directly prevent an entity from accessing its data, none of these services guarantee that the data and other system resources will always be at the disposal of an authorised entity. They also do not help to recover the system to its normal state when a crisis does arise.

The IS service of availability is used to complement the five fundamental IS services by (ideally) ensuring that these accessibility requirements are satisfied at all times. This IS service therefore aims to ensure that all the assets of a computer system (i.e. software, hardware, data, and the network) can be accessed when necessary by any authorised user. This service therefore helps the system to function as normal even when attacks do occur, whether the causes are unintentional (e.g. natural disasters) or intentional (e.g. an entity deliberately prevents an e-commerce web-site from functioning correctly). Availability is closely related to reliability, especially with regards to unintentional attacks. This is because a more reliable system should increase the probability that the system's resources will still be accessible under abnormal situations. A system can often be made more reliable by making use of redundancy (e.g. using backup systems) and increasing adaptability, which will in turn improve availability since alternative approaches to access the system's resources exist when attacks on the system occur (especially for physical attacks such as fire) (Ettinger, 1993).

Availability with regards to intentional attacks is mainly ensured by preventing *denial of service* (DoS) attacks. DoS attacks threaten any system or network by

disrupting or even denying legitimate entities access to the system. These attacks can cause loss of revenues, system delays and even system failures (not to mention the human resource costs and funds needed to identify and counter these attacks). Although these attacks are sometimes a necessary part of a hack, the main reasons for these attacks include vengeance, frustration (e.g. when a hacking attempt fails), and the thrill of being in control. What makes things worse is that tools needed to perform DoS attacks are easily available for some common operating systems and applications (without installed patches). It is also generally easier to perform a DoS attack on a system than to gain illegitimate access to it.

There are five main types of DoS attacks. *Bandwidth consumption attacks* occur when all the available bandwidth of the victim's network is consumed. Another type, *resource starvation attacks*, consume other system resources as opposed to network resources. A very common DoS attack takes advantage of *programming flaws* in operating systems, applications/services and embedded logic chips, and can cause failures in these. *Routing attacks* occur when an attacker alters a routing table in order to prevent a service (which makes use of that router) from being provided. Finally, *DNS attacks* are used to trick an entity into thinking that the attacker's server is the requested server. This is done by changing DNS cache entries (McClure, Scambray & Kurtz, 1999) .

Risk analysis must be performed for all potential threats to the system, whether intentional or unintentional. All vulnerabilities should first be identified for each necessary system resource. A probability of occurrence and a possible damage estimate should then be calculated for all of the identified vulnerabilities. Although this may be very difficult, especially for complex systems, it is a necessary part of the prevention process. Preventative and backup measures can then be engaged to help ensure the availability of the resources. *System survivability* involves the ability to continue to make system resources available *after* attacks occur. If risk analysis is correctly performed and the countermeasures correctly implemented, the time taken to recover the system back to its normal state after an attack takes place will be minimised (Neumann, 1995).

2.4 Conclusion

Information security is becoming increasingly important for most application domains. It is becoming essential for publicly accessible computer-based services and applications such as e-commerce, web-based communication, and electronic banking, especially due to the increased number of attacks on these. The services

discussed in this chapter (including availability) form the six fundamental services of information security. As already mentioned, it is not always necessary to implement all the IS services in order for a particular computer system to be secure, since this depends on the security requirements of that particular computer system.

My work on secure multimedia databases focuses mainly on the authorisation IS service; the main function of the models presented is to ensure that an entity requesting multimedia data only receives the parts that it is authorised to access. The models therefore provide access control at the multimedia level instead of at the file level, allowing an entity to at least access parts of the requested multimedia (as apposed to the “all or nothing” approach). The IS service of identification and authentication does however form an important role in ensuring that an entity actually possesses the claimed security classification. Similarly, the IS service of confidentiality has the important role of ensuring that the requesting entity is the only entity to receive the multimedia in decrypted form. Although not as important, the IS service of integrity may optionally be implemented to ensure that the requested multimedia has not been tampered with during transmission or storage. The service of non-repudiation may also be required to prove which entities have made multimedia requests. Finally, the amount of work needed to ensure the availability of the data depends on the actual implementation and requirements of the models.