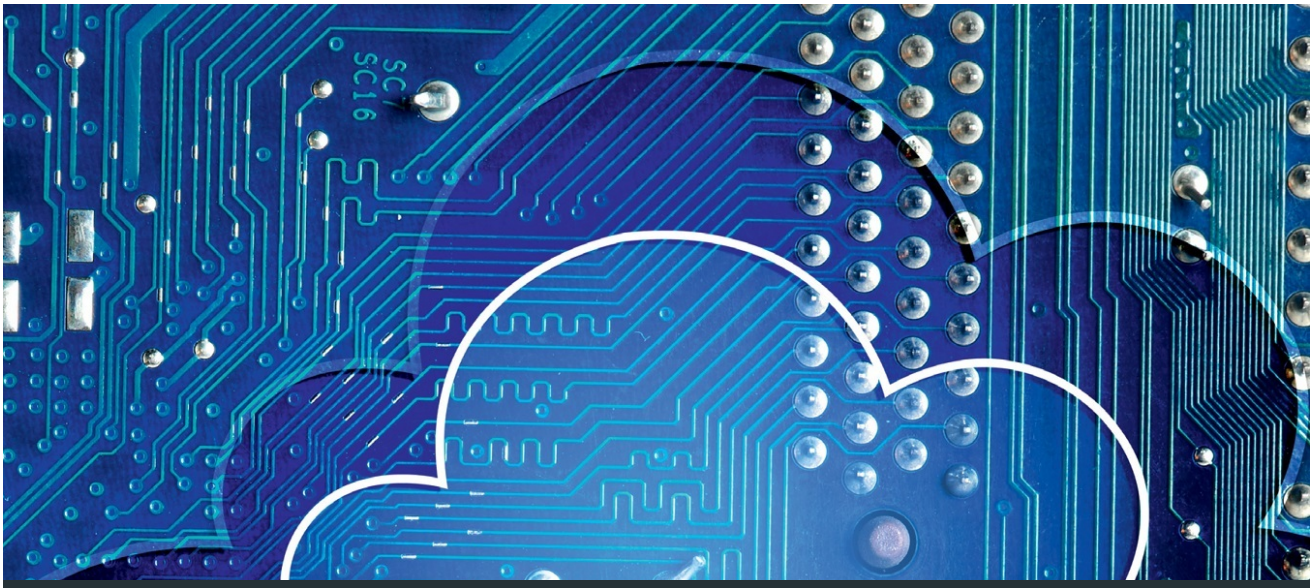
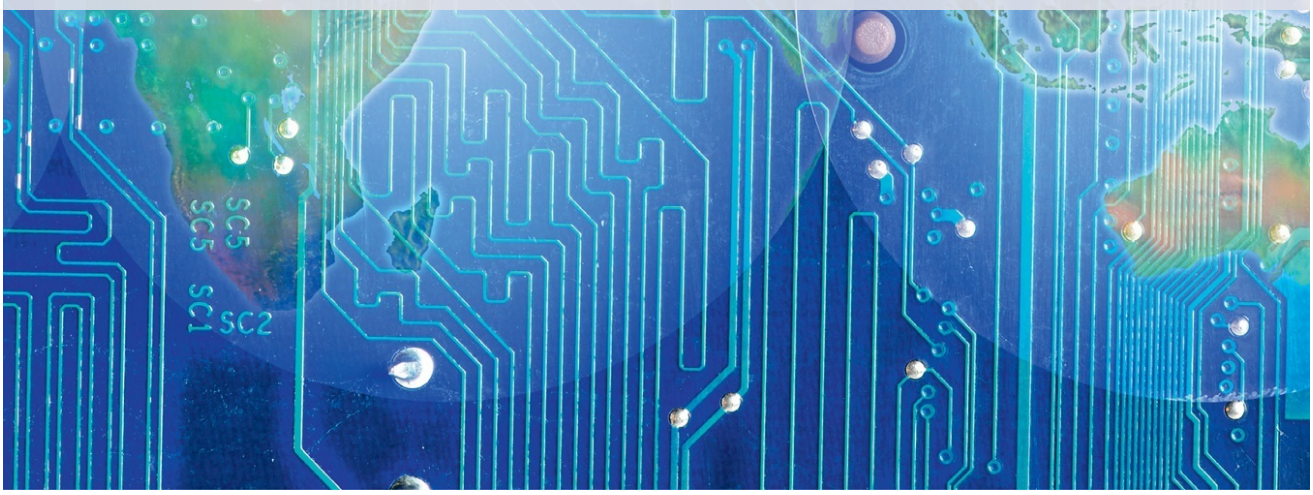


Chapter 9:
Security, privacy and data integrity

Learning objectives

By the end of this chapter you should be able to:

- explain the difference between the terms security, privacy and integrity of data
- show appreciation of the need for both the security of data and the security of the computer system
- describe security measures designed to protect computer systems, ranging from the stand-alone PC to a network of computers
- show understanding of the threats to computer and data security posed by networks and the Internet
- describe methods that can be used to restrict the risks posed by threats
- describe security measures designed to protect the security of data
- describe how data validation and data verification help protect the integrity of data
- describe and use methods of data validation
- describe and use methods of data verification during data entry and data transfer.



9.01 Definitions of data integrity, privacy and security

Data integrity

It is easy to define integrity of data but far less easy to ensure it. Only accurate and up-to-date data has **data integrity**. Any person or organisation that stores data needs it to have integrity. We will discuss ways to achieve data integrity in this chapter, and also in Chapter 11 (Sections 11.01 & 11.02).

Data privacy

Data privacy is about keeping data private rather than allowing it to be available in the public domain. The term may be applied to a person or an organisation. Every individual has an almost limitless amount of data associated with their existence. Assuming that an individual is not engaged in criminal or subversive activities, he or she should be in control of which data about himself or herself is made public and which data remains private. An organisation can have data that is private to the organisation, such as the minutes of management meetings, but we will not discuss this here.

For an individual there is little chance of data privacy if there is not a legal framework in place to penalise offenders who breach this privacy. This framework is provided by a **data protection law**. The following aspects should be noted about such laws.

- The major focus relates to personal, therefore private, data that an individual supplies to an organisation.
- The data is supplied to allow the organisation to use it but only for purposes understood and agreed by the individual.
- Data protection laws oblige organisations to ensure the privacy and the integrity of this data.
- Unfortunately, having laws does not guarantee adherence to them but they do act as a deterrent if wrong-doers can be subject to legal proceedings.

Discussion Point:

What data protection laws are in place in your country? Are you familiar with any details of these laws?

Data protection normally applies to data stored in computer systems with the consent of the individual. Should these laws be extended to cover storage of data obtained from telephone calls or search engine usage?

Data security

Data are secure if they are available for use when needed and the data made available are the data that were stored originally. The security of data has been breached if any data have been lost or corrupted.

Data security must be achieved before either data integrity or data privacy can be achieved, but data security does not by itself guarantee either data integrity or data privacy. One of the requirements for protection of data is the security of the system used to store the data. System security does not just protect data. There are two primary aims of system security measures:

- to ensure that the system continues to carry out the tasks users need
- to ensure that only authorised users have access to the system.

9.02 Threats to the security of a computer system and of the data stored in it

The threats to the security of a system include the following types:

- individual user not taking appropriate care
- internal mismanagement
- natural disasters
- unauthorised intrusion into the system by an individual
- malicious software entering the system.

TIP

Try to keep a perspective with relation to hacking; it is only one of many security issues.

Threats to computer and data security posed by networks and the Internet

We are all continuously at risk from security threats to systems we use ourselves and to all of the systems used by organisations upon which we rely. The dominant factor is that none of these systems are stand-alone; all are connected to networks and through these networks to the Internet. One cause of concern is the hacker who is someone intent on gaining unauthorised access to a computer system. A hacker who achieves this aim might gain access to private data. Alternatively, a hacker might cause problems by deleting files or causing problems with the running of the system. The other major cause of concern is malicious software entering the system.

Types of malware

Malware is the everyday name for malicious software. It is software that is introduced into a system for a harmful purpose. One category of malware is where program code is introduced to a system. The various types of malware-containing program code are:

- virus: tries to replicate itself inside other executable code
- worm: runs independently and transfers itself to other network hosts
- logic bomb: stays inactive until some condition is met
- Trojan horse: replaces all or part of a previously useful program
- spyware: collects information and transmits it to another system
- bot: takes control of another computer and uses it to launch attacks.

The differences between the different types are not large and some examples come into more than one of these categories. The virus category is often subdivided according to the software that the virus attaches itself to. Examples are boot sector viruses and macro viruses.

Malware can also be classified in terms of the activity involved:

- phishing: sending an email or electronic message from an apparently legitimate source requesting confidential information
- pharming: setting up a bogus website which appears to be a legitimate site
- keylogger: recording keyboard usage by the legitimate user of the system.

Question 9.01

Carry out some research to find some examples of how phishing and pharming might be attempted.

System vulnerability arising from user activity

Many system vulnerabilities are associated directly with the activities of legitimate users of a system. Two examples which do not involve malware are as follows.

- The use of weak passwords and particularly those which have a direct connection to the user. A poor choice of password gives the would-be hacker a strong chance of guessing the password and thus being able to gain unauthorised access.
- A legitimate user not recognising a phishing or pharming attack and, as a result, giving away sensitive information.

A legitimate user with a grievance might introduce malware deliberately. More often, malware is introduced accidentally by the user. Typical examples of actions that might introduce malware are:

- attaching a portable storage device
- opening an email attachment
- accessing a website
- downloading a file from the Internet.

Vulnerability arising from within the system itself

Systems themselves often have security weaknesses. The following are examples of this.

1. Operating systems often lack good security. Over time, there is a tendency for operating systems to increase in complexity, which leads to more opportunities for weak security. Operating systems have regular updates, often because of a newly discovered security vulnerability.
2. In the past, commonly used application packages allowed macro viruses to spread, but this particular problem is now largely under control.
3. A very specific vulnerability is buffer overflow. Programs written in the C programming language, of which there are very many, do not automatically carry out array bound checks. A program can be written to deliberately write code to the part of memory that is outside the address range defined for the array, set up as a buffer. The program overwrites what is stored there so when a later program reads this overwritten section it will not execute as it should. Sometimes this only causes minor disruption, but a cleverly designed program can permit an attacker to gain unauthorised access to the system and cause serious problems.

9.03 Security measures for protecting computer systems

Disaster recovery

Continuity of operation is vital for large computer installations that are an integral part of the day-to-day operations of an organisation. Measures are needed to ensure that the system continues working whatever event occurs or, if there has to be a system shut-down, at the very least to guarantee that the service will start again within a very short time.

Such measures come under the general heading of disaster recovery contingency planning. The contingency plan should be based on a risk assessment. The plan will have provision for an alternative system to be brought into action. If an organisation has a full system always ready to replace the normally operational one, it is referred to as a 'hot site'. By definition, such a system has to be remote from the original system to allow recovery from natural disasters such as earthquake or flood.

Safe system update

A special case of system vulnerability arises when there is a major update of hardware and/or software. Traditionally, organisations had the luxury of installing and testing a new system over a weekend when no service was being provided. In the modern era, it is more usual to have systems that can be accessed at any time and (often) from any location. A company is never closed for business. As a result, organisations may need to have the original system and its replacement running in parallel for a period to ensure continuity of service.

Discussion Point:

Major failings of large computer systems are well documented. You could carry out research to find some examples. Find an example of where the crisis was caused by technology failure and a different example where some natural disaster was the cause.

User authentication

Even if a PC is used by only one person there should be a user account set up. User accounts are, of course, essential for a multi-user (timesharing) system. The main security feature of a user account is the **authentication** of the user. The normal method is to associate a password with each account. In order for this to be effective the password needs a large number of characters including a variety of those provided in the ASCII scheme.

TASK 9.01

1. Create an example of a secure password using eight characters (but not one you are going to use).
2. Assuming that each character is taken from the ASCII set of graphic characters, how many different possible passwords could be defined by eight characters?
3. Do you think this is a sufficient number of characters to assume that the password would not be encountered by someone trying all possible passwords in turn to access the system?

Alternative methods of authentication include biometric methods and security tokens. A biometric method might require examination of a fingerprint or the face or the eye. A security token can be a small item of hardware provided for each individual user that confirms their identity. Similar protection can be provided by software with the user required to provide further input after the password has been entered. Normal practice is to combine one of these alternative methods with the password system.

Good practice

General good practice that helps to keep a personal computer secure includes not leaving the computer

switched on when unattended, not allowing someone else to observe you accessing the computer and not writing down details of how you access it.

Users may attach portable storage devices to a system, but this increases the risk of transferring malware into the system. This risk is reduced by an organisation having a policy banning the use of such devices or at least limiting their use. Unfortunately, this is difficult if normal business processes require portability of data.

Firewall

The primary defence to malware entering a system through a network connection is to install a **firewall**. Ideally a firewall will be a hardware device that acts like a security gate at an international airport. Nothing is allowed through without it being inspected. Alternatively, a firewall can run as software. Data must enter the system, but it can be inspected immediately. A firewall can inspect the system addresses identified in the transmission of data, but can sometimes also inspect the data itself to check for anything unusual or inappropriate.

Digital signature

If an incoming transmission is an email, you might want to check the identity of the sender. The solution is to insist that the sender attaches a digital signature to the email. Some details of this are discussed in Chapter 21 (Section 21.02).

Anti-virus software and intrusion detection

Security measures restricting access to a system do not guarantee success in removing all threats. It is therefore necessary to have, in addition, programs running on a system to check for problems. One option is to install what is normally referred to as anti-virus software but which is usually aimed at combating any type of malware. This carries out regular scans to detect any malware and to remove or deactivate it. Possibly special-purpose anti-spyware software might be installed. Another option is to install an intrusion detection system that will take as input an audit record of system use and look for examples that do not match expected system activity.

Unfortunately, people intent on causing damage to systems are using methods that are becoming ever more sophisticated. The defence methods have to be improved continually to counter these threats.

9.04 Security measures for protecting data

Recovering from data loss

In addition to problems arising from malicious activity there are a variety of reasons for accidental loss of data:

- a disk or tape gets corrupted
- a disk or tape is destroyed
- the system crashes
- the file is erased or overwritten by mistake
- the location of the file is forgotten.

For these reasons, every system should have a backup procedure to recover lost data. The system administrator decides on the details of the procedure. The principles for the procedure traditionally followed are straightforward:

- a full backup is made at regular intervals, perhaps weekly
- at least two generations of full backup are kept in storage
- incremental backups are made on a daily basis.

For maximum security the backup disks or tapes are stored away from the system in a fire-proof and flood-proof location.

This works well when an incremental backup occurs done overnight with the full backup handled at the weekend. With systems running 24/7, data in the system might be changing at any time, and a simple approach to backup would leave data in an inconsistent state.

One solution is to have a backup program that effectively freezes the file store while data is being copied. At the same time changes that are happening due to system use are recorded elsewhere within the system. The changes can then be implemented once the backup copy has been stored.

An alternative approach is to use a disk-mirroring strategy. In this case, data is simultaneously stored on two disk systems during the normal operation of the system. The individual disk systems might be at remote locations as part of a disaster recovery plan.

Restricting access to data

If a user has logged in, they have been authorised to use the computer system but not necessarily all of it. In particular, the system administrator may identify different categories of user with different needs with respect to the data they are allowed to see and use. A typical example is that one employee should be able to use the system to look up another employee's internal phone number. This should not allow the employee at the same time to check the salary paid to the other employee.

The solution is to have an **authorisation** policy which gives different access rights to different files for different individuals. For a particular file, a particular individual might have no access at all or possibly read access but not write access. In another case, an individual might have read access but not unrestricted write access.

Protecting data content

Even with appropriate security measures in place, a system and its data might still be accessed by someone who overcomes security to break into the system. This type of access can still be made a waste of time and effort if the stored data cannot be read. Data can be encrypted to ensure this. Some details of encryption methods are discussed in Chapter 21 (Sections 21.01, 21.03, 21.04, 21.05 & 21.06).

9.05 Data validation and verification

Data integrity can never be guaranteed, but the chances are improved if appropriate measures are taken when data originally enters a system or when it is transmitted from one system to another.

Validation of data entry

The term **validation** is a somewhat misleading one. It seems to imply that data is accurate if it has been validated. This is far from the truth. For example, if entry of a name is expected but the wrong name is entered, it will still be recognised as a name and therefore accepted as valid. Validation can only prevent incorrect data if there is an attempt to input data that is of the wrong type, in the wrong format or out of range.

Data validation is implemented by software associated with a data entry interface. There are a number of different types of check that can be made. Typical examples are:

- a presence check to ensure that an entry field is not left blank
- a format check, for example a date has to be dd/mm/yyyy
- a length check, for example with a telephone number
- a range check, for example the month in a date must not exceed 12
- a limit check, for example a maximum number of years for a person's age
- a type check, for example only a numeric value for the month in a date
- an existence check, for example that a file exists with the filename referred to in the data entry.

Verification of data entry

When data is entered into a system, **verification** means getting the user to confirm that the data entered was what was intended to be entered. Unfortunately, this still does not mean that the data entered is correct. Double entry is one method of verification. The most common example is when a user is asked to supply a new password. There will usually be a request for the password to be re-entered. A second method is to use a visual check of what has been entered. If a form has been filled in, it always makes sense to read through the data entered before sending the form off to its destination.

Check digit

When a series of numbers are used to identify something, it is possible to use a check digit method of verification. There are many different options here but they all require a calculation to be made with the numbers that have been entered. The final part of the calculation involves an integer division from which the remainder is added as an extra one or two digits at the end of the series of numbers. In one scheme where only a single check digit is allowed, the letter X is used when the remainder is calculated as 10. When the data is subsequently read the same calculation is carried out and the result is compared to the check digit that had been stored. This technique is used for a bar code or for the ISBN for a book.

Verification during data transfer

It is possible for data to be corrupted during transmission. Often, this happens when an individual bit is flipped from 1 to 0 or from 0 to 1. Verification techniques need to check on some property associated with the bit pattern.

The simplest approach is to use a simple one-bit parity check. This is particularly easy to do if data is transferred in bytes using a seven-bit code. Either even or odd parity can be implemented in the eighth bit of the byte. Assuming even parity, this is the procedure.

1. At the transmitting end, the number of 1s in the seven-bit code is counted.
2. If the count gives an even number, the parity bit is set to 0.

3 If the count gives an odd number, the parity bit is set to 1.

4 This is repeated for every byte in the transmission.

5 At the receiving end, the number of 1s in the eight-bit code is counted.

6 If the count gives an even number, the byte is accepted.

7 This is repeated for every byte in the transmission.

If no errors are found, the transmission is accepted. However, we cannot guarantee that the transmission is error free. It is possible for two bits to be flipped in an individual byte, which would mean that the transmission is incorrect but the parity check is passed. Fortunately, this is rather unlikely so it is sensible to assume no error. The limitation of the method is that it can only detect the presence of an error. It cannot identify the actual bit that is in error. If an error is detected, re-transmission has to be requested.

An alternative approach is to use the checksum method. At the transmitting end a block is defined as a number of bytes. Then, no matter what the bytes represent, the bits in each byte are interpreted as a binary number. The sum of these binary numbers in a block is calculated and supplied as a checksum value in the transmission. This is repeated for each block. The receiver does the same calculation and checks the sum of the numbers with the checksum value transmitted for each block in turn. Once again, an error can be detected but its position in the transmission cannot be determined.

TIP

Don't confuse a check digit with a checksum. A check digit is used for stored data; a checksum is only used for transmitted data.

Detecting the exact position of an error so as to correct it is considerably more complex. One approach is to use the parity block check method. Like the checksum method this is a longitudinal parity meaning that it is used to check a sequence of binary digits contained in a number of bytes.

WORKED EXAMPLE 9.01

Using a parity block check

At the transmitting end, a program reads a group of seven bytes as illustrated in Figure 9.01. The data is represented by seven bits for each byte. The most significant bit in each byte, bit 7, is undefined so we have left it blank.

	Seven-bit codes						
	1	0	1	0	0	1	1
	0	1	1	0	0	0	1
	1	0	1	1	0	0	0
	0	0	1	1	1	0	0
	0	1	1	0	0	1	0
	0	1	1	0	0	0	1
	0	1	1	0	0	0	1

Figure 9.01 Seven bytes to be transmitted

The parity bit is set for each of the bytes, as in Figure 9.02. The most significant bit is set to achieve even parity.

Parity bits	**Seven-bit codes**						
0	1	0	1	0	0	1	1

1	0	1	1	0	0	0	1
1	1	0	1	1	0	0	0
1	0	0	1	1	1	0	0
1	0	1	1	0	0	1	0
1	0	1	1	0	0	0	1
1	0	1	1	0	0	0	1

Figure 9.02 Bytes with the parity bit set

An additional byte is then created and each bit is set as a parity bit for the bits at that bit position. This includes counting the parity bits in the seven bytes containing data. This is illustrated in Figure 9.03.

Parity bits	Seven-bit codes						
0	1	0	1	0	0	1	1
1	0	1	1	0	0	0	1
1	1	0	1	1	0	0	0
1	0	0	1	1	1	0	0
1	0	1	1	0	0	1	0
1	0	1	1	0	0	0	1
1	0	1	1	0	0	0	1
0	0	0	1	0	1	0	0

← **Parity byte**

Figure 9.03 Parity byte added

The program then transmits the eight bytes in sequence.

At the receiving end, a program takes the eight bytes as input and checks the parity sums for the individual bytes and for the bit positions.

Note that the method is handling a serial transmission so it includes longitudinal checking, but the checking algorithm is working on a matrix of bit values. If there is just one error in the seven bytes this method will allow the program at the receiving end to identify the position of the error. It can therefore correct the error so the transmission can be accepted.

Question 9.02

Assume that the seven bytes shown in Figure 9.04 contain data.

01001000 01000101 01110010 01100011

00101100 01010101 00110010

Figure 9.04 Seven bytes to be transmitted

The most significant bit is set to 0 but it is undefined at this stage because a seven-bit ASCII code represents character data. Choose a parity and change the value stored in the most significant bit to match this parity for each byte. Then create the eighth byte that would be used for transmission in a parity block check method.

Question 9.03

The eight bytes shown in Figure 9.05 have been received in a transmission using the parity block method. The first seven bytes contain the data and the last byte contains the parity check bits.

01001000 11000101 11110001 01100011

01001010 01010101 01110010 01110010

Figure 9.05 Eight bytes received in a transmission

a Identify what has gone wrong during the transmission.

b What would happen after the transmission is checked?

Reflection Point:

This chapter contains a lot of terminology. It is very easy to get confused about the definitions of the different terms used. Have you considered how you are going to attempt to remember all of the definitions and not get confused?

Summary

- Important considerations for the storage of data are: data integrity, data privacy and data security.
- Data protection laws relate to data privacy.
- Security measures for computer systems include authentication of users, prevention of unauthorised access, protection from malware and methods for recovery following system failure.
- Security methods for data include backup procedures, user authorisation and access control.
- Data entry to a system should be subject to data validation and data verification.
- Verification for data transmission may be carried out using: a parity check, a checksum or a parity block check method.

Exam-style Questions

1 **a** It is important that data has integrity.

i Identify the missing word in the sentence 'Concerns about the integrity of data are concerns about its [1]

ii Validation and verification are techniques that help to ensure data integrity when data is entered into a system.

Explain the difference between validation and verification. [3]

iii Define a type of validation and give an example. [2]

iv Even after validation has been correctly applied data may lack integrity when it comes to be used. Explain why that might happen. [2]

b Data should be protected from being read by unauthorised individuals.

Explain **two** policies that can be used to provide the protection. [4]

2 **a** Security of data is an important concern for a system administrator.

i Identify **three** reasons why data might not be available when a user needs it. [3]

ii Describe what could be features of a policy for ensuring data security. [3]

b It is important for mission-critical systems that there is a disaster recovery contingency plan in place.

i Define what type of disaster is under consideration here. [2]

ii Define what will be a major feature of the contingency plan. [2]

c Measures to ensure security of a computer system need to be in place on a daily basis if the system is connected to the Internet.

Describe **two** measures that could be taken to ensure security of the system. [4]

3 **a** When data is transmitted measures need to be applied to check whether the data has been transmitted correctly.

i If data consists of seven-bit codes transmitted in bytes, describe how a simple parity check system would be used. Your account should include a description of what happens at the transmitting end and what happens at the receiving end. [5]

ii An alternative approach is to use a checksum method. Describe how this works. [3]

b For either of these two methods there are limitations as to what can be achieved by them. Identify **two** of these limitations. [2]

c A different method which does not have all of these limitations is the parity block check method.

The following diagram represents eight bytes received where the parity block method has been applied at the transmitting end. The first seven bytes contain the data and the last byte contains parity bits.

Byte 1	0	1	0	1	0	0	1	1
Byte 2	1	0	1	1	0	0	0	1
Byte 3	1	1	0	1	0	0	0	0
Byte 4	1	0	0	1	1	1	0	0
Byte 5	1	0	1	1	0	0	1	0

Byte 6	1	0	1	1	0	0	0	1
Byte 7	1	0	1	1	0	0	0	1
Byte 8	0	0	0	1	0	1	0	0

Identify the problem with this received data and what would be done with it by the program used by the receiver. [4]

4 **a** Give the definition of the terms firewall and authentication. Explain how they can help with the security of data. [3]

b Describe **two** differences between data integrity and data security. [2]

c Data integrity is required at the input stage and also during transfer of the data.

i State **two** ways of maintaining data integrity at the input stage. Use examples to help explain your answer. [3]

ii State **two** ways of maintaining data integrity during data transmission. Use examples to help explain your answer. [3]

Cambridge International AS & A level Computer Science 9608 paper 13 Q3 June 2015

5 Verification and validation can be applied during data entry.

Describe what is meant by these terms. For each method, explain why it is needed. [4]

Cambridge International AS & A level Computer Science 9608 paper 12 Q8 November 2015