A Consideration of the Modes of Operation for Secure Systems

C L Robinson and S R Wiseman

Secure Information Systems Group
Defence Research Agency
Malvern
Worcestershire, WR14 3PS, UK

email: harrold@dra.hmg.gb or wiseman@dra.hmg.gb

Abstract. Secure systems are often characterised by a 'mode of operation'. This acts as a shorthand for the degree of risk to the information on the system and the minimum security functionality required as a countermeasure. This paper examines the UK definitions of these modes and proposes a model of a system which can be used to capture the distinctions between them. The variations of possible secure system functionality within each mode are then discussed. Some new definitions, which are orthogonal to the modes of operation, are proposed which can be used to resolve ambiguities.

Keywords. Security Mode of Operation, Dedicated, System High, Compartmented, Multi-Level, System Model, Z Notation

1 Introduction

Within the UK, secure systems are often characterised by a 'mode of operation'. This acts as a shorthand for the degree of risk to information on the system and the minimum security functionality required as a countermeasure. There are currently four UK modes of operation: Dedicated; System High; Compartmented and Multi-Level. The CESG Glossary [1] provides definitions of these modes as particular combinations of user need-to-know, formal clearances and security functionality. Similar terms and definitions are used in the US [2].

Any definition relating to a secure system requires a precise understanding of the boundary of a system and its users. Such an understanding was relatively straightforward when systems were large standalone mainframes located in computer rooms. However, in the age of desktop computers and networks the boundary of the system and the identification of its users becomes more problematical. This difficulty does appear to be recognised, since the definitions of the UK modes have been clarified with subsequent issues of the glossary. Similar problems with the US definitions are discussed in [3].

The main problem with the definitions of the modes of operation is that they are not sufficiently detailed enough to give a complete picture of a system. Systems

which have the same mode of operation can have different security functionality requirements. Thus, the use of the mode of operation alone as a shorthand to describe a system can potentially lead to misunderstandings. This paper proposes additional sets of terms to characterise the 'boundary mode', 'output class mode' and 'output need-to-know mode' of a system. The intention is that the definitions can provide a framework to assist in the identification of appropriate security requirements for individual projects.

The definitions in this paper are firmly based upon an abstract mathematical model of a system. This model has been used to explore the distinctions between the different modes of operation and how they relate to the system boundary and to justify the proposed new definitions. Therefore, in this paper the purpose of the model is to explore the different ways in which secure systems may be used in military environments. The model is at a high level of abstraction, without many of the details of system functionality or security mechanisms. This use of a model is therefore different to the commonly quoted models of operating system security mechanisms, due to Bell and LaPadula [4].

In order to avoid ambiguity, the Z notation [5] is used to precisely define the system model and definitions. The notation is based on standard set theory, and a brief explanation of the symbols used is given as Annex A. However, the overview of the model and English commentary to the formal specifications should be sufficient to gain an appreciation of the model, the distinctions between the different modes of operation and the proposed new definitions.

The structure of this paper is as follows: Section 2 considers the standard UK definitions of the modes of operation and identifies the concepts which are required to model them; Section 3 describes the model of a system, which is then specified formally in Section 4; Section 5 defines the four modes of operation for secure systems, and for completeness an additional mode that corresponds to an insecure system; Section 6 discusses these formal definitions and proposes some orthogonal criteria which can be used to distinguish particular types of system within the mode categories; Finally, Section 7 draws some conclusions.

2 Identification of Model Elements

The following definitions are directly taken from the official UK glossary of computer security terminology [1].

DEDICATED. "A mode of operation in which all the users of a system are cleared for, need to know about and have access to all the data handled by it. Hence, the system does not enforce national security rules or need-to-know and little or no technical security functionality is required."

SYSTEM HIGH. "A mode of operation in which all the users are cleared for, and have formal access approval for, all the information handled by it, but not all of whom actually need to know about all of the data. In this mode of operation DAC will normally be applied".

COMPARTMENTED. "A mode of operation in which all the users of a system are cleared for all the data handled by the system; and who only need to have, and are only given, access to some of the data by means of MAC."

MULTILEVEL. "A mode of operation in which a computer system (or network of computer systems) handles data at various classifications etc., but for which there are users not cleared for all that data whose access is restricted appropriately. Hence, the system (or network) is relied upon to enforce the national security rules."

Thus, it can be seen that the basic concepts in the model will need to be user clearances, data classifications and need-to-know requirements.

The definitions for System High and Compartmented modes are similar. Indeed, the Compartmented mode has only relatively recently been introduced into UK terminology, although it appears to have been used in the US for some time. The distinction between the two modes is in the type of security functionality used to control the user's need-to-know. In a System High mode, need-to-know is controlled by Discretionary Access Control (DAC) mechanisms, whereas in a Compartmented mode Mandatory Access Control (MAC) mechanisms are used.

Thus, the model will need to distinguish between systems with no technical security functionality (i.e. for Dedicated Mode) and between those with DAC or MAC functionality. The model also needs to distinguish between MAC functionality which is used to control need-to-know (i.e. for Compartmented Mode) and MAC based on clearances (i.e. for Multi-Level Mode).

The standard definitions for MAC and DAC, for example in [1], are not sufficiently precise to clearly explain the intended differences between the System High and Compartmented modes of operation. However, the mode definitions suggest that the difference relates to the strength of the mechanism. With DAC functionality, there would appear to be the potential for users to be given access to information for which they do not have a need-to-know. Obviously, to operate in this mode, the system risk assessment must have determined that this did not represent a significant threat.

It is assumed that in a System High Mode, the discretion of the users of a system can be trusted. Therefore, a reasonable interpretation of the difference between MAC and DAC for need-to-know relates to the trustworthiness of software. In other words, software acting on behalf of a user might misuse discretionary controls but cannot step outside of the bounds imposed by mandatory controls. Precise definitions for the different types of access control are proposed in [6].

Therefore, the model of a system includes the notion of software proxies. These are the active agents working for the human users within the computer system. The model distinguishes between software which can be trusted only to alter a discretionary control with the authority of the user, and software which cannot be so trusted.

The technical security controls within a system are not modelled at the same level of abstraction as in the Bell and LaPadula models of [4]. Instead, they are modelled by identifying the possible handling restrictions applied to outputs. This is based on the modelling approach of Goguen and Meseguer [7]. In a system with no technical security controls all the output has to be treated at the level of the highest possible security class on the system. Alternatively, in a system with technical security functionality, i.e. MAC or DAC, to control the flows of information within the system, output can be labelled more accurately.

3 An Overview of the Model

This section gives an overview of the proposed model of a system. This model is subsequently used to explore the modes of operation and propose further definitions. The purpose of the model is to define a generalised secure system, and therefore many of the details of realistic implementations are omitted. Further, the model permits cases which may not be required in particular environments. However, it is not considered to be sensible to complicate the model by explicitly excluding some situations.

Figure 1 below depicts the elements of the model and their relationship to each other and the system boundary. The elements of the model are described in the following text, and then precise definitions are given in the formal model of section 4.

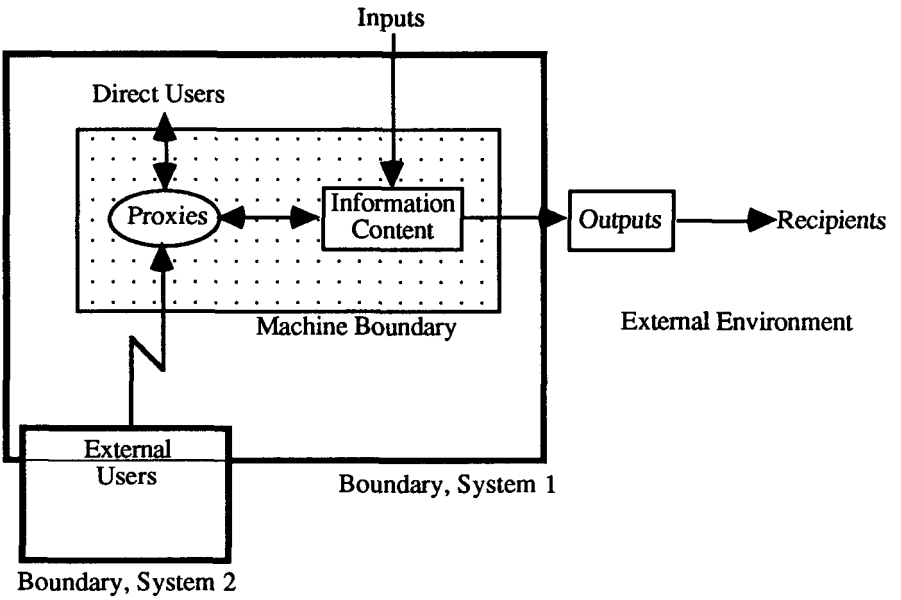


Fig. 1. The Model of a System

3.1 Information and its Security Class and Need-to-Know

A system contains information. This comprises input from other systems and input from the users of the system, together with information generated as a result of calculations and associations made within the machine. All the information in a system has an inherent security class and need-to-know requirement associated with it. The need-to-know for information is abstractly represented by the set of people who need-to-know the information in order to effectively carry out their job function.

A system may not contain information at all possible security classes. Neither do the need-to-know sets for a particular system necessarily contain all combinations of people. This is because a system generally exists in an environment which can restrict what is entered into the system. The functionality of a system can also limit the security classes and need-to-knows for the information it can create.

The inherent need-to-know of information in a system can include people who are not users of the machine. However, someone has to have a need-to-know for each item of information, either in the system or output from it (see below). In other words, the model does not consider something to be information if no one needs to know it.

However, the model does not restrict the inherent need-to-know for information to the users of the machine. Instead, the need-to-know in the model represents all the people who need to have access to the information in some form. Thus, the model permits there to be information in a system for which no user has a need-to-know.

3.2 Users

A system has at least one authorised user. The users are the people who influence the contents of the system. They are the people who enter and manipulate information on the system via the functionality of the machine. The users are outside of the machine boundary and interact with the contents via software proxies within the machine boundary.

A distinction is made between the direct and external users. Direct users are under the direct control of the system management, and will generally access the system via its physical components, such as screens and keyboards. The external users access the system via an interconnection, and will be the direct users of a different system. A particular person could be both an external and a direct user of a system, for example if they were able to logon both at a connected workstation and remotely from another system.

The reason for the distinction between direct and external users relates to the positioning of a system boundary when many computers from different management domains are interconnected. However, the exact positioning of the boundary is not important to the model. What is important is that all the people who can influence the information in the system are identified by the two sets.

3.3 Outputs and Recipients

A system produces outputs, although there may be some information on a system which is never output. An output is something, either paper-based or electronic, which enters the environment outside of the direct control of the system. Within the external environment there are people who are the potential recipients of the outputs. These people could also be users of the system. The difference between a recipient and a user is that the former cannot influence the information, but only receive it.

A system has to 'trust' its external environment to ensure that its outputs are handled appropriately. To assist the external environment, a system will label its outputs with respect to the minimum security clearance and need-to-know requirements.

3.4 Clearances and User Need-to-Knows

The authorised users may have different formal clearances, and there may also be users of a system who hold no formal clearance. However, every user of the system has to have a need-to-know for some item of information on the system. This represents the obvious requirement that people should only be given access to a computerised

system if they have a need to do so. Therefore, this definition rules out people hacking into a system across a network, and thus the model requires that a system has appropriate identification and authentication measures in place.

It is less obvious that it is reasonable for there to be information on the system for which no user is formally cleared. For example, it is possible to postulate a messaging system where highly classified information is transported by users not cleared to access the information. They obviously need-to-know some aspect of the information, such that it exists and the address it is intended for, but do not need-to-know or be cleared for the contents.

In addition, UK Security Policy contains an "Occasional Access" rule. This permits a person with insufficient formal clearance to be given access to a limited amount of information above their clearance because they have a genuine need-to-know that information in order to effectively carry out their duties. Therefore, the model of an electronic system should not rule out this case.

However, although the system model permits otherwise, in practice it would be expected that there will always be a user who is actually formally cleared for all information on a system, such as system security administrators. In this case the interesting point is that they need to be formally cleared in order to manage the system, but do not in fact have a genuine need-to-know for the information they are managing. Thus, the managers represent the dual of the Occasional Access rule. In effect the need for a system manager who is able to access all the information on the system arises from the implementation of the system and is not a system requirement. Therefore, from an abstract viewpoint the system managers could be considered as an implementation detail.

3.5 Software Proxies and Confidence

Although software acts as a user's proxy within the machine, the user does not necessarily have complete confidence that their wishes are being faithfully carried out. This is because software is generally not well specified and is frequently accepted and used when it contains errors. It can thus exhibit surprising or unexpected behaviour. Where this is the case the user will have a fairly low level of confidence that their wishes are being carried out.

On the other hand, using a Trusted Path to invoke a security critical function gives the user a great deal of confidence that their wishes are being carried out. This is because additional effort has been expended in the specification, design, testing and evaluation of the software and its means of invocation. Thus, in a system there will be differing degrees of confidence in the software proxies which act on behalf of the human users.

Control over need-to-know is achieved through human discretion, actioned via software proxies within the machine boundary. Therefore, the model associates with each software proxy which is capable of altering need-to-know controls, a degree of confidence that it only does so with the authority of the responsible person.

3.6 Security Policy

A secure system is always 'no flows down' with respect to security classes. Thus CONFIDENTIAL information may be output labelled CONFIDENTIAL, SECRET, etc., but cannot be labelled UNCLASSIFIED.

A system may contain technical security functionality which is able to maintain separation between the different classes of information on the system and monitor the flows of classified information around the system. Where this is the case, information can be output labelled at its inherent security class. Such functionality is generally referred to as MAC.

Other systems may not contain security functionality capable of maintaining accurate labels within the system. When this is the case, all the outputs must be labelled at the level of the highest security class on the system. Otherwise the system might not be 'no flows down'.

No information flow property is given for need-to-know. Thus, no constraints are placed on the possible output labels with respect to need-to-know for the information on the system. Appropriate control is achieved for need-to-know by applying human judgement and discretion.

3.7 Summary

To summarise, a system contains information at various security classes and with various need-to-know requirements. The information is accessed by its users via software proxies, and there are varying degrees of confidence in the trustworthiness of this software. A system loses direct control over its outputs. It therefore applies security markings in order that the recipients of output can be limited to those with adequate clearance or need-to-know. A formal specification is given in section 4.

4 The Formal Model

This section specifies the model of a system using the Z notation [5] along with a brief English commentary that explains the intended interpretation of the mathematics. The meaning of the Z symbols used is included as Annex A.

Computerised systems are used by people. Thus a set is introduced to represent all the possible people of interest. This set includes the users, i.e. the people who are able to influence the state of the computer system, and the people who receive the outputs from the system.

```
[ PERSON ]
```

However, the users cannot directly manipulate the information within a computer, but must use software to act on their behalf. Thus, the model identifies a set of software agents, or proxies.

```
[ PROXY ]
```

Although software acts as a user's proxy within the computer, the user does not necessarily have complete confidence that their wishes are being faithfully carried out. For the purposes of this model a set is introduced to represent the various degrees of confidence that people could have in software. The confidence levels are partially ordered by a relation called $\succeq$ (dominates), which defines which degrees of confidence are 'better' than others.

```
[ CONFIDENCE ]
| _ ≥ _ : partial_order CONFIDENCE
```

One particular confidence is important to the modes of operation, although the system model permits there to be other confidences. This confidence is called NTKfaithful and is the point at which the people start to believe that the software carries out their wishes with respect to need-to-know (NTK) controls. Thus software with a confidence which dominates NTKfaithful is trusted not to misuse the discretion of the human user with respect to the need-to-know controls. Software whose confidence level does not dominate NTKfaithful is not so trusted.

```
| NTKfaithful : CONFIDENCE
```

Secure systems manipulate classified information, and thus a set of security classes is defined. Note that a security class encompasses categories and caveats in addition to a hierarchical component, as defined in [1]. Thus, security classes can encompass certain need-to-know controls such as codewords. The security classes are partially ordered by a relation called ≥ (dominates). A least upper bound function is given for security classes, but the definition is omitted for clarity.

```
[ CLASS ]

| _ ≥ _ : partial_order CLASS
| LUB : ℙ CLASS → CLASS
|-------------------------------
| ...
```

A system is characterised by the non-empty set of identities for its users and the non-empty set of clearances which they hold. At the level of abstraction used in this model, it is not necessary to directly relate users to clearances. A distinction is made between the direct users, who are under the control of the system management, and external users, who access the system via an interconnection. Not all the authorised users of a system need to hold a formal clearance.

```
┌─ USERS ──────────────────────────────────────────┐
| DirectNames, ExternalNames : ℙ PERSON
| DirectClearances, ExternalClearances : ℙ CLASS
|--------------------------------------------------
| DirectNames ∪ ExternalNames ≠ {}
| DirectClearances ∪ ExternalClearances ≠ {}
└──────────────────────────────────────────────────┘
```

Additional sets of people and clearances are identified in the model. These represent the identities of all the people who may potentially receive outputs from the system, either paper-based or electronic, and their clearances.

```
RECIPIENTS
RecipientNames : ℙ PERSON
RecipientClearances : ℙ CLASS
---
RecipientClearances ≠ {} ⇒ RecipientNames ≠ {}
```

The security classes and need-to-know for information on the system (section 3.1) and the labelling of output from a system (section 3.6) are both captured in the Z specification using functions called OutputClasses and OutputNTKs.

The domain of the OutputClasses function models the inherent security classes for the information contained within the system. For each possible security class within the system, the OutputClasses function gives the set of possible output security labels. OutputClasses is restricted to ensure that the system is 'no flows down'. In other words, for each security class on the system, all the possible output labels must be at least as high.

The distinction between no technical security functionality within the system and functionality to separate different classes is also captured. The CLASS_CONTROLS schema states that either information can be output at its inherent security class, i.e. separation is maintained, or alternatively all the outputs must be labelled at the level of the highest security class within the system.

```
CLASS_CONTROLS
OutputClasses : CLASS ⇸ ℙ CLASS
---
OutputClasses ≠ {}
∀ c : dom OutputClasses •
   ∀ l : OutputClasses( c ) • l ≳ c
( ∀ c : dom OutputClasses • c ∈ OutputClasses( c )
   ∨ rng OutputClasses = {{ LUB (dom OutputClasses) }} )
```

Similarly, the domain of the OutputNTKs function models the sets of people who need to know the various pieces of information within the system. For each possible need-to-know set within the system, the OutputNTKs function gives the set of possible output security labels. However, no information flow restrictions are defined for need-to-know labelling.

```
NTK_CONTROLS
OutputNTKs : ℙ1 PERSON ⇸ ℙ ℙ1 PERSON
-----------------------------------------
OutputNTKs ≠ {}
```

Note that a possible interpretation of the abstract type $\mathbb{P}_1$ PERSON could be an Access Control List, although other interpretations, such as all the people who are cleared into a particular codeword, are not ruled out by the model. Alternatively, certain codewords could be modelled by security classes.

The final element of the formal model is a function to capture the level of confidence in the trustworthiness of the particular software proxies with respect to the need-to-know controls. The domain of this function represents all the software proxies within the system which have the ability to alter the need-to-know controls. For each, the function gives the level of confidence that they carry out the human user's wishes. An empty set for this function is intended to capture the case where the system contains no technical need-to-know controls.

```
SOFTWARE
NTKProxyConfidence : PROXY ⇸ CONFIDENCE
```

Thus, in this model a system is defined by: its users; the recipients of output; the security classes of information it contains and how it labels its outputs; the need-to-know for information and how this is labelled on output, together with the confidence that software does not misuse any need-to-know controls.

```
SYSTEM
USERS
RECIPIENTS
CLASS_CONTROLS
NTK_CONTROLS
SOFTWARE
-----------------------------------------
∀ u : DirectNames ∪ ExternalNames •
   ∃ ntk : dom OutputNTKs • u ∈ ntk
```

5 Definition of Security Modes of Operation

Returning to the glossary definitions of the modes of operation, given in Section 2, it can be seen that they require four criteria with which to partition the model of a system.

i) Whether all the users (both direct and external) are cleared for all of the information on the system.
ii) Whether all the users have a need-to-know for all of the information on the system.
iii) The kind of technical security mechanisms which are used for need-to-know.
iv) Whether the system contains security functionality which can ensure that information is accurately labelled on output with a security class.

A Dedicated system is distinguished from all the other modes of operation by the fact that all the users are both formally cleared and have a need-to-know for all of the information on the system. Consequently, a Dedicated system is secure whether or not it contains technical security functionality for need-to-know or security classes.

```
Dedicated
SYSTEM
─────────────────────────────────────────────
(DirectClearances ∪ ExternalClearances) ×
   (dom OutputClasses) ⊆ _ ≳ _
∀ ntk : dom OutputNTKs •
   DirectNames ∪ ExternalNames ⊆ ntk
```

For a System High mode of operation all the users are formally cleared, but do not have a need to know for all of the information. In other words, there must exist at least one need-to-know grouping of information on the system which does not include at least one of the users. However, the main distinguishing feature of the System High mode is that the need-to-know controls are vulnerable to misuse by the software acting on behalf of the users. This is expressed by stating that at least one software proxy capable of altering the need-to-know controls, is not trusted to always faithfully carry out the user's wishes.

```
SystemHigh
SYSTEM
─────────────────────────────────────────────
(DirectClearances ∪ ExternalClearances) ×
   (dom OutputClasses) ⊆ _ ≳ _
∃ ntk :dom OutputNTKs •
   ¬( DirectNames ∪ ExternalNames ⊆ ntk )
∃ c : rng NTKProxyConfidence • ¬( c ≳ NTKfaithful )
```

The distinction between the System High and Compartmented modes is captured by the fact that in the Compartmented mode all of the software proxies capable of altering need-to-know controls are trusted not to misuse this ability. Note that in

practice this is usually achieved by arranging that the 'untrusted' software cannot alter the controls.

```
┌─ Compartmented ──────────────────────────────────────────┐
│ SYSTEM                                                   │
├──────────────────────────────────────────────────────────┤
│ (DirectClearances ∪ ExternalClearances) ×                │
│    (dom OutputClasses) ⊆ _ ≳ _                           │
│ ∃ ntk :dom OutputNTKs ·                                  │
│    ¬( DirectNames ∪ ExternalNames ⊆ ntk )                │
│ NTKProxyConfidence ≠ {}                                  │
│ rng NTKProxyConfidence × {NTKfaithful} ⊆ _ ≳ _           │
└──────────────────────────────────────────────────────────┘
```

The feature which distinguishes the Multi-Level mode from the others is that not all of the users have formal security clearance for all of the information and technical security functionality is present which is capable of maintaining accurate security labels. Thus for every inherent security class on the system the information can be output labelled at that class. Such functionality is capable of preventing the users from accessing information for which they are not cleared.

Note that this proposed definition of the Multi-Level Mode of operation also requires that if there are users with no need-to-know for information there must be some technical security controls to prevent them from gaining access.

```
┌─ Multi-Level ────────────────────────────────────────────┐
│ SYSTEM                                                   │
├──────────────────────────────────────────────────────────┤
│ ¬((DirectClearances ∪ ExternalClearances) ×              │
│    (dom OutputClasses) ⊆ _≳_)                            │
│ ∀ c : dom OutputClasses · c ∈ OutputClasses( c )         │
│ ∃ ntk :dom OutputNTKs ·                                  │
│    ¬( DirectNames ∪ ExternalNames ⊆ ntk )                │
│ ⇒  NTKProxyConfidence ≠ {}                               │
└──────────────────────────────────────────────────────────┘
```

A system can be insecure for two reasons. Firstly, it is insecure for there to be no functionality to maintain accurate security labels and yet have users with insufficient formal security clearance. Secondly, it is insecure to have users with no need-to-know for information, and yet have no technical security controls to prevent them from gaining access.

```
┌─ InsecureClass ──────────────────────────────────────────────┐
│ SYSTEM                                                       │
├──────────────────────────────────────────────────────────────┤
│ ¬((DirectClearances ∪ ExternalClearances) ×                  │
│    (dom OutputClasses) ⊆ ≳)                                  │
│ rng OutputClasses = { ( LUB ( dom OutputClasses ) ) }        │
└──────────────────────────────────────────────────────────────┘

┌─ InsecureNTK ────────────────────────────────────────────────┐
│ SYSTEM                                                       │
├──────────────────────────────────────────────────────────────┤
│ ∃ ntk :dom OutputNTKs ·                                      │
│    ¬( DirectNames ∪ ExternalNames ⊆ ntk )                    │
│ NTKProxyConfidence = {}                                      │
└──────────────────────────────────────────────────────────────┘
```

$$Insecure \triangleq InsecureClass \vee InsecureNTK$$

As discussed earlier, this model of a system recognises the existence of the UK Occasional Access rule. This permits people limited access to information for which they have a genuine need-to-know, but are not formally cleared. A system which requires this rule to be invoked in a controlled manner will need to contain security functionality which is capable of maintaining accurate security labels on information. Thus, the definitions above place such a system as operating in the Multi-Level mode. However, in a computer with software acting on behalf of the human users, the precise difference between a system which requires the Occasional Access rule to be invoked and an insecure system is unclear.

These five types of system partition the model of a system given in this paper. Thus, all systems which conform to the model described in Section 4 will meet exactly one of the above definitions. An informal justification for this theorem is given as Annex B. This contains a table listing the combinations of criteria and identifies the corresponding mode.

$$\vdash \langle\, Dedicated, SystemHigh, Compartmented, Multi\text{-}Level, Insecure \,\rangle \ \underline{partition}\ SYSTEM$$

6 Discussion and Further Definition

Systems which meet the definition of a particular mode of operation can have different technical security requirements. In particular, the definitions of both System High and Compartmented allow systems within the mode to have either accurate labelling of output or to label at the level of the highest security class on the system. This could be a potential source of confusion and ambiguity whenever the mode of operation is used as a shorthand to describe a system.

Furthermore, systems may meet different modes of operation definitions depending upon where the system boundary is drawn with respect to the direct and external users. For example a system could be Compartmented if only the direct users were counted, and yet be Multi-Level if the clearances of all the people who could access the system via an interconnection, i.e. external users, were considered.

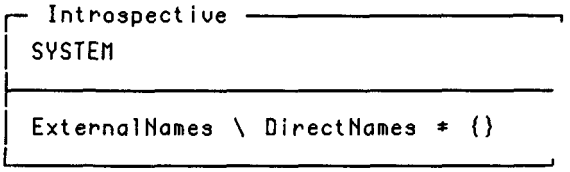
However, even when all the users have been taken into consideration, and all have formal security clearances, a system may still require accurate labelling of output. In other words, accurate labelling of output may be required in systems other than those which operate in the Multi-Level mode (as defined in this paper). Such a situation arises quite naturally where there is a requirement for a system to pass its outputs into an environment where there will be people who are not fully cleared.

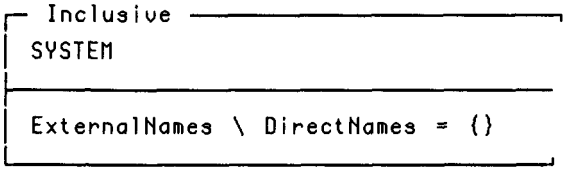
This section proposes three further terms which can be used to more fully characterise a system. These are derived from the model given in this paper, and describe the 'boundary mode', the 'output class mode' and the 'output need-to-know mode' of a system.

6.1 Boundary Mode

As discussed above, a system may meet different modes of operation depending upon where the boundary has been drawn with respect to the direct and external users. The problem partly results from the fact that the official terminology does not precisely define the users of a system. In terms of the model of a system used in this paper, choosing just the direct users who are under the direct control of the system management may give one mode, whilst including the external users gives another. Such a situation could occur if an interconnection requirement for a system arises at a late stage in its procurement.

This paper proposes that systems which have not included external users in their mode of operation be referred to as *Introspective*, since the system boundary has been drawn between its direct and external users. It is proposed that a system which does consider both kinds of user be called *Inclusive*.





⊢ ⟨ Introspective, Inclusive ⟩ partition SYSTEM

6.2 Output Class Mode

An output class mode for a system is proposed. This characterises the potential recipients of the outputs from the system with respect to the security clearances which they hold. Four modes are defined, *Isolated*, *Connected* and *StrongBounded*, and a mode corresponding to a system with an insecure boundary.

In determining the output class mode for a system, the clearances of the potential recipients are considered with respect to the highest security class contained within the system. It is important to note that they are not just compared with the security classes of the outputs they should receive.

An Isolated system is one where all of the possible recipients of output have sufficient clearance for all of the security classes contained on the system. Thus, no technical security functionality within the system is necessary, since labels are not required to control access to outputs. Without technical security functionality to maintain separation between security classes, all the outputs will be labelled at the level of the highest. Such a system can be considered to be isolated with respect to security classes, since it does interact with people who are not cleared for all of its information.

```
┌─ Isolated ──────────────────────────────────────────────────┐
│ SYSTEM                                                      │
├─────────────────────────────────────────────────────────────┤
│ RecipientClearances × (dom OutputClasses) ⊆ _ ≥ _            │
│ rng OutputClasses = { { LUB dom OutputClasses } }           │
└─────────────────────────────────────────────────────────────┘
```

A Connected system is one where the outputs may enter an environment where there are potential recipients who are not formally cleared for all of the information on the system. Thus, the system needs to provide accurate labels on its outputs in order to ensure that the environment to which it passes responsibility for information is able to make appropriate access control decisions.

```
┌─ Connected ─────────────────────────────────────────────────┐
│ SYSTEM                                                      │
├─────────────────────────────────────────────────────────────┤
│ ¬( RecipientClearances × (dom OutputClasses)  ⊆ _ ≥ _ )      │
│ ∀ c : dom OutputClasses  ·  c ∈ OutputClasses( c )          │
└─────────────────────────────────────────────────────────────┘
```

A system would have an insecure boundary if there were potential recipients of output who were not cleared for all of the classes of information on the system, and yet the system did not contain any security functionality to ensure that security labels were accurate.

```
┌─ InsecureBoundary ──────────────────────────────────────────┐
│ SYSTEM                                                      │
├─────────────────────────────────────────────────────────────┤
│ ¬( RecipientClearances × (dom OutputClasses)  ⊆ _ ≳ _ )     │
│ rng OutputClasses = { { LUB ( dom OutputClasses ) } }       │
│ # dom OutputClasses # 1                                     │
└─────────────────────────────────────────────────────────────┘
```

There is one final case to consider. This is where all the potential recipients of outputs have sufficient clearance for all of the information on the system, and yet the system maintains accurate security labels on the various classes of information in the system. Such a situation could be needed if there were authorised users (as opposed to the recipients of output) with insufficient clearance.

```
┌─ StrongBounded ─────────────────────────────────────────────┐
│ SYSTEM                                                      │
├─────────────────────────────────────────────────────────────┤
│ RecipientClearances × { dom OutputClasses }  ⊆ _ ≳ _        │
│ ∀ c : dom OutputClasses · c ∈ OutputClasses( c )            │
│ # dom OutputClasses # 1                                     │
└─────────────────────────────────────────────────────────────┘
```

```
⊢ < Isolated, Connected, StrongBounded, InsecureBoundary >
        partition SYSTEM
```

6.3 Output Need-to-Know Mode

Finally, an output need-to-know mode is proposed. This is similar to the output class mode, except that the potential recipients of the outputs are considered with respect to their need-to-know for information. Five modes are defined, *Continuous*, *Linked*, *Discrete* and *StrongNTKBounded*, together with a mode corresponding to a system with an insecure need-to-know boundary.

A Continuous system is defined to be one where all the potential recipients of outputs have a need-to-know for all of the information within the system. Thus, no technical security functionality is required to ensure that the external environment, either electronic or human, can make appropriate access control decisions. Such a system could be considered as continuous, because the boundary between the electronic systems, or between the electronic and human world, is not significant.

```
Continuous
SYSTEM
─────────────────────────────────────────────
∀ ntk : dom OutputNTKs • RecipientNames ⊆ ntk
NTKProxyConfidence = {}
```

A Linked system is defined to be one where there are potential recipients of output who do not have a need-to-know for all of the information, and where the threat is such that 'weak' need-to-know controls are sufficient. The system contains security controls to provide need-to-know labelling which may be misused by software acting on behalf of the users, i.e. without their authority. Thus, the need-to-know markings applied to output could be inappropriate. Such a type of system can be referred to as linked with its external environments. Thus, there is a weak boundary between the system and the environment of its outputs.

```
Linked
SYSTEM
─────────────────────────────────────────────
∃ ntk : dom OutputNTKs • ∃ u : RecipientNames • u ∉ ntk
∃ c : rng NTKProxyConfidence • ¬( c ≳ NTKfaithful )
```

Thirdly, a system can be considered to be a discrete entity with external interfaces requiring 'strong' need-to-know controls. Such controls ensure that the environments which take over responsibility for the protection of information can make appropriate decisions. In this case there are potential recipients of outputs who do not have a need-to-know for all of the information on the system. The system contains security controls for need-to-know which cannot be altered inappropriately by the software acting on behalf of its users.

```
Discrete
SYSTEM
─────────────────────────────────────────────
∃ ntk : dom OutputNTKs • ∃ u : RecipientNames • u ∉ ntk
NTKProxyConfidence ≠ {}
rng NTKProxyConfidence x { NTKfaithful } ⊆ _ ≳ _
```

A system is insecure across its boundary to external environments if there are potential recipients who do not have a need-to-know for all of the information on the system, and yet the system has no technical security functionality to provide need-to-know controls.

```
InsecureNTKBoundary
SYSTEM
─────────────────────────────────────────────────────
∃ ntk : dom OutputNTKs • ∃ u : RecipientNames • u ∉ ntk
NTKProxyConfidence = {}
```

The final case to consider is where all the potential recipients of output have a need-to-know for all of the information on the system, and yet the system contains some need-to-know controls. This situation could be needed if the authorised users (as opposed to the recipients of output) do not all have a need-to-know for the information on the system.

```
StrongNTKBounded
SYSTEM
─────────────────────────────────────────────────────
∀ ntk : dom OutputNTKs • RecipientNames ⊆ ntk
NTKProxyConfidence ≠ {}
```

$\vdash$ < Continuous, Linked, Discrete, StrongNTKBounded, InsecureNTKBoundary> partition SYSTEM

7 Conclusions

This paper has considered the standard UK glossary definitions of the modes of operation for secure systems. A mathematical model of a system has been proposed and precise definitions for Dedicated, System High, Compartmented and Multi-Level have been given in terms of this model. New, orthogonal, definitions have been proposed which can be used to describe the differences between various kinds of system which have the same mode of operation.

Therefore, this paper concludes that it is possible to devise finer-grained descriptions of a system than the standard modes of operation currently provide. What remains for future work is to determine whether the finer-grained descriptions proposed in this paper can be usefully applied to the problems of determining the appropriate security functionality and assurance requirements which are encountered in real system procurements.

A second conclusion from the work presented in this paper concerns the value of the use of formal methods. The development of the mathematical model of a system required the issues of the boundary of a system and its users to be explored in depth. Furthermore, the mathematical specification has underpinned the development of the ideas contained in this paper. Whether or not the proposed definitions of the existing and new modes are appropriate, the mathematical specification provides a solid foundation for further discussions in this area. It is concluded that the use of mathematical modelling can be an effective tool in the development of the conceptual foundations and terminology for the science of Computer Security.

8 References

1. CESG Computer Security Memorandum 1, Glossary of Computer Security Terms, Issue 2.2, November 1993

2. Guidance for Applying the Department of Defense Trusted Computer System, Evaluation Criteria in Specific Environments, CSC-STD-003-85, June 1985

3. H O Lubbes: COMPUSEC, A Personal View, Proceedings of the 9th Annual Computer Security Applications Conference, Orlando, Florida, December 6 - 10, 1993

4. D E Bell, L J LaPadula: Secure Computer Systems: Mathematical Foundations, MTR-2547, Volume 1, November 1973; Secure Computer Systems: A Mathematical Model, MTR-2547 Volume II, November 1973; Secure Computer Systems: A Refinement of the Mathematical Model, MTR-2547 Volume III, April 1974; and Secure Computer System: Unified Exposition and Multics Interpretation, MTR-2997, January 1976

5. J M Spivey: The Z Notation: a Reference Manual, 2nd Edition, Prentice Hall International, 1992

6. S R Wiseman, C L Robinson and M M Adams: A Mathematical Definition of Access Control, DRA report DRA/CIS/CSE2/94007, April 1994

7. J A Goguen, J Meseguer: Security Policies and Security Models, Proceedings of the 1982 Symposium on Security and Privacy, Oakland, California, April 1982

Annex A: An Overview of the Z Notation

Z is a mathematical notation that has been developed by the Programming Research Group at Oxford University. The underlying basis of Z is standard set theory, and it makes use of the associated notation. Properties about sets are described using predicate calculus. A Z specification is structured into self contained parts using schemas.

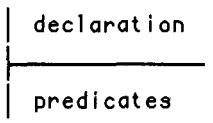
$[A]$	introduction of a new set, called A
$\{\}$	set, with no members
$a \in A$	a is a member of the set A
$a \notin A$	a is not a member of the set A
$A \subseteq B$	A is a subset of the set B (possibly equal)
$A \cup B$	union of members of the sets A and B

$A \setminus B$	members of set A which are not in set B
$A \times B$	set consisting of all the possible pairings of members of A and B
$x : T$	declaration, x is an element drawn from the set T
$\mathbb{P}\ A$	powerset, i.e. the set of all possible subsets of A (including empty set)
$\mathbb{P}_1\ A$	powerset, excluding empty set
AS partition A	the set of sets AS are disjoint sets and cover A

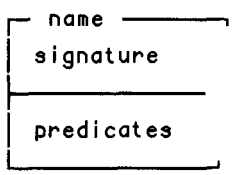
A relation may be viewed as a set of ordered pairs. Functions are a special type of relation where there is a single element in the range for each element of the domain. Thus, the operators defined for sets are applicable to both functions and relations.

$dom\ r$	domain of relation, r, i.e. set of all the first elements of the ordered pairs
$rng\ r$	range of a relation, r, i.e. set of all the second elements of the ordered pairs
$f : A \rightarrow B$	f is a total function, i.e. domain is all possible members of the set A
$f : A \nrightarrow B$	f is a partial function, i.e. domain is not necessarily all of the set A
$r : partial\_order\ A$	r is an ordered relation where all pairs of elements are not necessarily comparable

$P \vee Q$	either predicate P holds or Q does
$\neg P$	predicate P does not hold
$P \Rightarrow Q$	if P holds then Q does
$\forall\ x:T \cdot P$	for all x of type T predicate P holds
$\exists\ x:T \cdot P$	there exists an x of type T for which predicate P holds
$\vdash P$	P is a theorem



An axiomatic definition. The declarations are global and the predicates define properties about them. The predicates are optional.



A schema. The signature declares some variables and their types. The predicates define some properties about them.
The declarations from one schema are made available to another by including the name of the schema in the signature. They are in scope until the end of the schema. The predicates are conjoined with those of the new schema.

$S \triangleq T$	schemas S and T are equivalent

Annex B: Modes of Operation

The table below lists the possible combinations of the mode of operation partitioning criteria and gives the mode of operation for each. Certain combinations represent insecure systems, and where this is the case a note gives the reason. The table also indicates where the combination of clearance and need-to-know suggests that the UK Occasional Access rule is being applied.

Users Cleared:	all	all users (direct and external) cleared for all security classes on the system
	some	at least one not cleared
Users NTK:	all	all users (direct and external) have a need-to-know for all information on the system
	some	at least one does not have a need-to-know for something
NTK Labels:	none	no technical security functionality to control access based on need-to-know
	weak	need-to-know mechanisms vulnerable to inappropriate use by untrusted software
	strong	need-to-know mechanisms not vulnerable
Class Labels:	high	all output must be labelled at highest security class on system
	accurate	security functionality present to maintain accurate security class labels

Users Cleared	Users NTK	NTK Labels	Class Labels	Mode
all	all	none	high	DEDICATED
all	all	none	accurate	DEDICATED
all	all	weak	high	DEDICATED
all	all	weak	accurate	DEDICATED
all	all	strong	high	DEDICATED
all	all	strong	accurate	DEDICATED
all	some	none	high	insecure^{NTK}
all	some	none	accurate	insecure^{NTK}
all	some	weak	high	SYSTEM HIGH
all	some	weak	accurate	SYSTEM HIGH
all	some	strong	high	COMPARTMENTED
all	some	strong	accurate	COMPARTMENTED

some	all	none	high	insecure[CLASS]
some	all	none	accurate	MULTI-LEVEL[OAR]
some	all	weak	high	insecure[CLASS]
some	all	weak	accurate	MULTI-LEVEL[OAR]
some	all	strong	high	insecure[CLASS]
some	all	strong	accurate	MULTI-LEVEL[OAR]
some	some	none	high	insecure[NTK and CLASS]
some	some	none	accurate	insecure[NTK]
some	some	weak	high	insecure[CLASS]
some	some	weak	accurate	MULTI-LEVEL
some	some	strong	high	insecure[CLASS]
some	some	strong	accurate	MULTI-LEVEL

[NTK]This combination is insecure because although not all users need-to-know all the information, no technical security controls are applied to prevent them getting access.

[CLASS]This combination is insecure because although not all users are cleared, no controls are applied.

[OAR]This combination suggests that the Occasional Access rule applies since although not all users are cleared for all the information, all have a need-to-know.