

Tracing Secure Information Flow Through Mode Changes

Colin Fidge¹

Tim McComb²

¹School of Software Engineering and Data Communications, Queensland University of Technology

²School of Information Technology and Electrical Engineering, The University of Queensland

Abstract

Communications devices intended for use in security-critical applications must be rigorously evaluated to ensure they preserve data confidentiality. This process includes tracing the flow of classified information through the device's circuitry. Previous work has shown how this can be done using graph analysis techniques for each of the device's distinct operating modes. However, such analyses overlook potential information flow between modes, via components that store information in one mode and release it in another. Here we show how graph-based analyses can be extended to allow for information flow through sequences of consecutive modes.

1 Introduction

Electronic communications devices safeguard classified information in government and military computer networks. In particular, 'domain separation' devices allow the flow of information between high and low-security networks to be controlled. Examples of such devices include: data diodes, which allow one-way information flow only; multi-computer switches, which allow peripheral devices to be shared between different domains; context filters, which constrict information flow; and cryptographic devices, which allow transmission of classified information over insecure networks.

Before such a device can be deployed, however, it must be carefully evaluated to ensure that it maintains data confidentiality. Such evaluations involve a detailed analysis of the device's design and construction (Bishop 2003, Ch. 21), as prescribed by international security standards (Herrmann 2003).

Previous work has shown how information flow can be traced through the circuitry of domain-separation devices to see if there are unintended data pathways from a high-security domain to a low-security one (Rae & Fidge 2005a, Rae & Fidge 2005b, McComb & Wildman 2005). Because a device will typically send data to different destinations in different modes, such analyses must take each of the device's different operating modes into account. These include 'failure'

modes in which a faulty component alters the flow of information.

However, previous analyses fail to identify unintended information-flow pathways created by *changes* in operating mode. A potential security risk may be caused by components within a device that can store information in one mode and release it in another. It is then possible for classified data en route to a high-security destination to be accidentally diverted to a low-security domain when the device's mode is changed.

Here we use a series of worked examples to show how a simple extension to existing graph-theoretic analysis techniques allows us to identify potential security leaks caused by mode changes. This is done by allowing information-flow pathways to be defined not only as those in which data is transmitted in a particular operating mode, but also as those in which data is communicated *between* different operating modes, providing the information flows intersect at components that may store and forward data.

2 Previous Work

International standards such as the *Common Criteria for Information Technology Security Evaluation* define the need to undertake detailed analyses of communications devices intended for use in high-security computer networks (Common Criteria 1999). Inevitably, however, general standards like these offer broad guidelines only, so much work is required to put them into practice (Herrmann 2003).

Our own research has been dedicated to automatable evaluation techniques for domain-separation devices. In previous work we have shown how to analyse electronic circuitry to identify information flow paths (Rae & Fidge 2005a, Rae & Fidge 2005b), and have implemented this theory in a practical tool (McComb & Wildman 2005). However, this work was limited to exploring information flow occurring within a single operating mode of the device. Behaviours that span modes were not considered.

In the context of safety-critical, rather than security-critical, systems the 'mode logic' of embedded controllers has been well explored elsewhere (Miller 1998, Paynter 1996). Although similar in motivation to our research, this work is not concerned with information flow, but rather with the need to model the way the system may change operating modes, in order to prove that no undesirable sequences of mode changes are possible.

3 Motivation

In previous research we explained how the schematic circuit diagram of a communications device could be treated as an information flow graph for the purposes

We wish to thank the anonymous reviewers for their helpful comments, especially a suggestion which inspired the example in Section 8. This research was funded by the Defence Signals Directorate and the Australian Research Council via Linkage-Projects Grant LP0347620, *Formally-Based Security Evaluation Procedures*.

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at Twenty-Ninth Australasian Computer Science Conference (ACSC2006), Hobart, Tasmania, Australia, January 2006. Conferences in Research and Practice in Information Technology, Vol. 48. Vladimir Estivill-Castro and Gill Dobbie, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

of security evaluation (Rae & Fidge 2005b). Discrete components such as logic gates and microprocessor chips are treated as nodes, and the wiring connecting them as arcs. The way each type of component connects its inputs to its outputs is defined for each of the device’s operating modes. The device’s circuit diagram is then modelled as an adjacency matrix of the following form.

	outputs
inputs	modes

Each row represents a connection acting as an input and each column a connection acting as an output. Based on the behaviour of the various components, the cells in the matrix are populated with sets of operating modes, possibly empty, defining the particular modes in which the corresponding input and output are connected.

Connectivity across the whole circuit is then defined as the transitive closure of the adjacency matrix. The security evaluator can then easily see whether there is any information flow from those inputs to the device which come from a high-security domain and those outputs which lead to a low-security domain.

Importantly, the transitive closure is calculated under the assumption that two adjacent arcs are connected only if they are both ‘active’ in the same mode. Since a circuit diagram is usually fully-connected, this assumption limits the information flow analysis to realistic cases.

Although adequate for components such as logic gates, that can be modelled as if they ‘instantaneously’ transfer information from their inputs to their outputs, this assumption is not always satisfactory. If it possible for a particular component to store information when the device is one mode, and subsequently release the information when the device is in another mode, then classified data may flow through the device in a way that will not be identified by a mode-specific analysis. Therefore, our goal below is to extend the approach to allow for the possibility that some components, such as memory chips and flip flops, can transfer information from one operating mode to another.

4 Tracing Information Flow Through Mode Changes

To do this we must extend the notion that a component connects its inputs to its outputs in a particular mode to allow for components that receive information in one mode and release it in a subsequent mode. Therefore, rather than reasoning about particular modes of the device, we must consider *sequences* of consecutive modes.

Let M, N, \dots represent distinct operating modes of a device. In practice these may be ‘normal’ behaviours, fault modes or both. We firstly introduce mode sequences within angled brackets, e.g., $\langle M, N, O, P \rangle$, to model the notion that the device has switched between the particular sequence of modes, in the order shown.

To model the mode-specific way a component connects its inputs to its outputs we also need a special operator for joining mode sequences end-to-end. This is denoted here by ‘ $X \oplus Y$ ’, which joins (non-empty) mode sequences X and Y provided that the last mode in sequence X is the same as the first mode in sequence Y .

$$\begin{aligned} & \langle M, \dots, O, P \rangle \oplus \langle P, Q, \dots, S \rangle \\ = & \langle M, \dots, O, P, Q, \dots, S \rangle \end{aligned} \quad (1)$$

Sequences whose last/first modes don’t match cannot be joined. For example, $\langle M, \dots, P \rangle \oplus \langle Q, \dots, S \rangle$ does not produce a valid mode sequence.

To model components that may exchange information in *any* mode, we also introduce mode ‘wildcards’, denoted ‘ \star ’ that match any mode.

$$\begin{aligned} & \langle M, \dots, O, P \rangle \oplus \langle \star, Q, \dots, S \rangle \\ = & \langle M, \dots, O, P, Q, \dots, S \rangle \end{aligned} \quad (2)$$

Similarly for the symmetric case with the wildcard at the end of the first sequence. Notice that, given this rule, we can use the sequence $\langle \star, \star \rangle$ as a ‘connector’ to join any two mode sequences.

Since it is unhelpful to say that a device switches from a given mode P to the same mode P , we also assume that duplicated, non-wildcard modes are always compressed.

$$\begin{aligned} & \langle M, \dots, O, P, P, Q, \dots, S \rangle \\ = & \langle M, \dots, O, P, Q, \dots, S \rangle \end{aligned} \quad (3)$$

We then follow our previous approach (Rae & Fidge 2005b) of populating the adjacency matrix with the mode-specific connections implied by the individual components in the device and performing matrix multiplications to calculate the transitive closure and achieve an end-to-end connectivity analysis. Now, however, the cells of the adjacency matrix contain sets of mode sequences, rather than just sets of modes.

Let Z be an $N \times N$ adjacency matrix. Let $Z_{(i,j)}$ be the value in the cell in Z ’s i th row and j th column. Then we define the square Z^2 as a matrix in which the cell in the i th row and j th column is calculated as follows.

$$Z^2_{(i,j)} = \bigcup_{1 \leq k \leq N} \{s \oplus t \mid s \in Z_{(i,k)} \wedge t \in Z_{(k,j)}\} \quad (4)$$

In other words, if information can flow from input i to output k via mode sequence s , and information can flow from input k to output j via mode sequence t , then information can flow from input i to output j via mode sequence $s \oplus t$ (provided that sequences s and t can be joined using Rules 1 to 3 above). The transitive closure of a graph can be found by squaring its adjacency matrix until it stops changing.

To illustrate the technique and its properties, the following sections present four distinct examples in which ‘inter-mode’ information flow may occur.

5 Example: A Device With a Modal, Buffered Component

Here we consider an example where a component with memory transmits information in different directions in different operating modes. An abstract view of the device of interest is shown in Figure 1. It comprises four components, A , B , C and D , which act as serial interface converters, and a switching component X , which directs the flow of information. Such a device may, for instance, form part of a multi-computer switch, used to allow peripheral devices to be shared between different security domains, while still maintaining separation of data.

We assume that component X has two different operating modes, M and N , selected by a physical switch on the device (not shown in the figure). In mode M information flows along connections e , f , g and h , via components B , X and A . In mode N information flows along connections i , j , k and l , via components D , X and C . Connections e and h interface to a high-security domain, while connections i

Table 1: Adjacency matrix for the switching device using global modes

	e	f	g	h	i	j	k	l
e		$\{M\}$						
f			$\{M\}$					
g				$\{M\}$				
h								
i						$\{N\}$		
j							$\{N\}$	
k								$\{N\}$
l								

Table 2: Connectivity matrix for the switching device using global modes

	e	f	g	h	i	j	k	l
e		$\{M\}$	$\{M\}$	$\{M\}$				
f			$\{M\}$	$\{M\}$				
g				$\{M\}$				
h								
i						$\{N\}$	$\{N\}$	$\{N\}$
j							$\{N\}$	$\{N\}$
k								$\{N\}$
l								

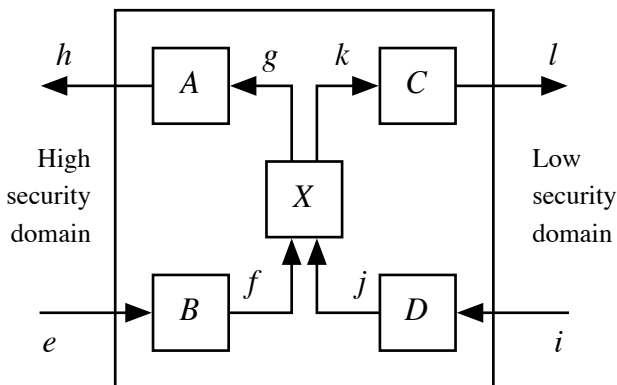


Figure 1: A buffered switching device

and l interface to a low-security domain. We also assume that component X contains a data buffer, to smooth the flow of data packets through the device.

To evaluate such a device’s security properties we must determine whether there are any information-flow pathways from the high-security domain to the low-security one in any operating mode. We first perform the analysis using the previous mode-specific approach (Rae & Fidge 2005b) to illustrate its weakness.

Table 1 shows the initial adjacency matrix as populated by the security evaluator, showing connectivity through the individual components in different modes. For instance, it says that information flows from e to f , from f to g and from g to h in mode M , as per the above description of the device’s behaviour. Similarly for mode N . All unoccupied cells are assumed to contain the empty set of modes. We omit the cells on the diagonal since it is unhelpful to note that an arc is connected to itself.

To analyse the device’s overall connectivity, we then perform matrix multiplications until a fixed point is reached, populating each cell (i, j) with the intersection of the mode sets from cells (i, k) and (k, j)

(Rae & Fidge 2005b). This is consistent with the notion that components must be in the same mode to interact. The resulting connectivity matrix is shown in Table 2.

This analysis correctly shows that information may flow from e to h in mode M and from i to l in mode N . Moreover, the two clusters of occupied cells in Table 2 clearly suggests that the device successfully achieves domain-separation in its two modes.

Unfortunately, this analysis completely overlooks the possibility that component X retains information between modes. To solve this we redo the analysis using mode sequences as defined in Section 4 above.

Table 3 shows our initial population of the adjacency matrix using mode sequences. As before it includes the explicit information flow in operating modes M and N . However, this time the security evaluator has chosen to use ‘mode wildcard’ sequences to indicate that buffered component X may forward information received on input j in one mode to output g in another mode. Similarly for input f and output k . (We have added only those additional connections created by component X ’s ability to store-and-forward information in different modes. Adding ‘wildcards’ to the (f, g) and (j, k) cells would also be reasonable but does not change the outcome in this case.)

Table 4 then shows the matrix after multiplying twice (at which point it stops changing) using Rule 4 above. This time the (i, h) cell tells us that information may flow between these two arcs when component X ’s mode is switched from N to M . This is an unexpected but harmless behaviour of the device.

Of much more concern is that the (e, l) cell shows potential information flow when the mode changes from M to N . This is especially disturbing since this is a previously unsuspected flow from the high-security domain to the low-security one.

In this way the security evaluator is alerted to a potentially dangerous flow of information and the need to study this pathway in more depth. For instance, if it can be proven that component X ’s buffer is cleared

Table 3: Adjacency matrix for the switching device using mode sequences

	e	f	g	h	i	j	k	l
e		$\{\langle M \rangle\}$						
f			$\{\langle M \rangle\}$				$\{\langle \star, \star \rangle\}$	
g				$\{\langle M \rangle\}$				
h								
i						$\{\langle N \rangle\}$		
j			$\{\langle \star, \star \rangle\}$				$\{\langle N \rangle\}$	
k								$\{\langle N \rangle\}$
l								

Table 4: Connectivity matrix for the switching device using mode sequences

	e	f	g	h	i	j	k	l
e		$\{\langle M \rangle\}$	$\{\langle M \rangle\}$	$\{\langle M \rangle\}$			$\{\langle M, \star \rangle\}$	$\{\langle M, N \rangle\}$
f			$\{\langle M \rangle\}$	$\{\langle M \rangle\}$			$\{\langle \star, \star \rangle\}$	$\{\langle \star, N \rangle\}$
g				$\{\langle M \rangle\}$				
h								
i			$\{\langle N, \star \rangle\}$	$\{\langle N, M \rangle\}$		$\{\langle N \rangle\}$	$\{\langle N \rangle\}$	$\{\langle N \rangle\}$
j			$\{\langle \star, \star \rangle\}$	$\{\langle \star, M \rangle\}$			$\{\langle N \rangle\}$	$\{\langle N \rangle\}$
k								$\{\langle N \rangle\}$
l								

every time the device changes from mode M to N then the device may still be considered secure. If not, however, it may need to be rejected.

The transitive closure also produces ‘wildcard’ mode sequences in the (i, g) , (j, h) , (e, k) and (f, l) cells, denoting potential information flow along these paths in different modes, but since none of these paths both begin and end in the device’s environment they are of little security interest. Generally, we are interested only in pathways leading from a high-security domain, or some other source of classified information, to a low-security domain.

6 Example: A Device with a Non-Moded Buffered Component

In this section we consider an example where the component that retains information between modes is not itself affected by mode changes, but another component that uses it is.

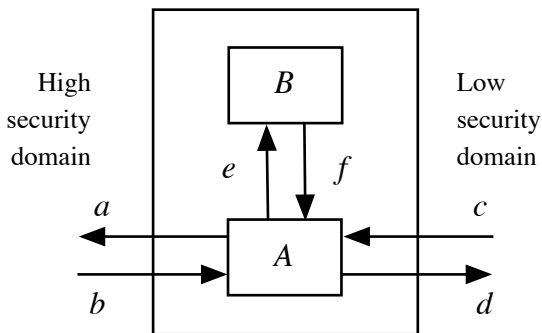


Figure 2: A cryptographic device

Figure 2 shows a (highly abstract) view of a cryptographic device. It connects a high-security computer to a low-security network. The device has two modes, encryption E and decryption D . In encryption

mode E plaintext messages are read from the computer via input b , encrypted by processor A , using memory chip B to temporarily store each block of the resulting ciphertext, and these are then sent to the network via output d . In decryption mode D ciphertext messages are read from the network via input c , decrypted by processor A , using memory chip B to temporarily store each block of the resulting plaintext, and these are then sent to the computer via output a . In practice, such devices are used in pairs, allowing computers in two different high-security domains to communicate over a low-security network. Each device must thus perform both encryption and decryption functions to support bidirectional communication. The device’s overall mode, encryption or decryption, is controlled by signals (not shown) sent to processor A from either the local or remote high-security domain.

Once again, we first attempt to evaluate the device using the old approach. Table 5 shows the security evaluator’s population of the adjacency matrix, based on our understanding of the way components A and B work. To model the fact that information can be written to and subsequently read from memory chip B in either mode, we put both modes E and D into the (e, f) cell. (This seems unnatural, because a memory chip doesn’t normally have ‘modes’, but leaving the cell empty would imply there is no information flow through this component. Indeed, this dilemma highlights how poorly suited the previous approach is to handling devices that store-and-forward information.)

The transitive closure after performing matrix multiplications is shown in Table 6. It correctly identifies information flow from c to a in decryption mode D and from b to d in encryption mode E , both of which are expected for this device.

Again, however, this approach fails to recognise the possibility that information stored in the memory chip in one mode may accidentally be released in another mode. Therefore, we redo the analysis using mode sequences. To model the memory component’s ability to connect information flow in any

Table 5: Adjacency matrix for the cryptographic device using global modes

	a	b	c	d	e	f
a						
b					$\{E\}$	
c					$\{D\}$	
d						
e						$\{E, D\}$
f	$\{D\}$			$\{E\}$		

Table 6: Connectivity matrix for the cryptographic device using global modes

	a	b	c	d	e	f
a						
b				$\{E\}$	$\{E\}$	$\{E\}$
c	$\{D\}$				$\{D\}$	$\{D\}$
d						
e	$\{D\}$			$\{E\}$		$\{E, D\}$
f	$\{D\}$			$\{E\}$		

pair of modes—because the memory chip itself is not influenced by changes to the device’s encryption/decryption mode—we put a wildcard entry into the (e, f) cell in Table 7.

The resulting transitive closure after matrix multiplication is shown in Table 8. This time the (b, a) cell tells us that information put into memory in encryption mode E could potentially be sent back to the high-security domain in decryption mode D . Although this is not an intended behaviour of the device, it would be harmless from a security perspective.

More seriously, though, the (c, d) cell alerts us to the danger that

1. processor A , in decryption mode, reads a ciphertext message from input c ,
2. it decrypts the ciphertext and stores the resulting blocks of plaintext data in memory chip B via arc e ,
3. processor A ’s mode is switched to encryption,
4. processor A reads the contents of memory component B , via arc f , and sends the blocks of decrypted plaintext to the low-security domain via output d .

This series of events would allow the device to decrypt a ciphertext message received from the low-security domain and send the resulting plaintext back to the low-security domain!

Having been alerted to this risk, the security evaluator would be obliged to look for mitigations against it. This would most likely require a careful analysis of the software on processor A to ensure that it accesses arrays in memory in a safe way or, better still, that it completely clears the contents of memory chip B when the mode changes from encryption to decryption.

7 Example: Combining Buffered and Modal Modules

The next example demonstrates that the approach has good compositional behaviour. In this case we show that combining a buffered component with an otherwise ‘secure’ modal circuit makes the overall device insecure in the presence of mode changes.

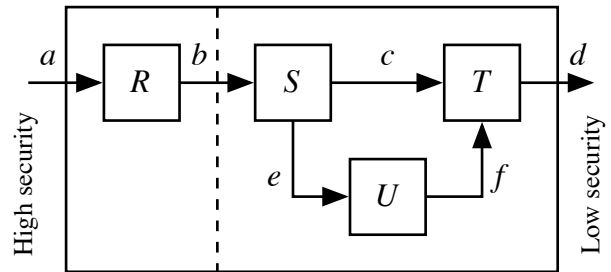


Figure 3: A context filtering device

Figure 3 shows a (highly abstract) design for a context filter. Such devices are used to restrict the flow of information from a high-security domain to a low-security one. In this case we assume the device has two operating modes, filter F and bypass B . When the device is in filter mode it is intended to accept information from the high-security domain and forward only data packets that satisfy some predefined security criterion to the low-security domain. When the device is in bypass mode information may flow from the high-security to the low-security domain without restriction. (Bypass modes are typically needed in such devices to allow binary control data through while setting up communication links.)

The device’s design consists of two main modules. On the right is a modal (and essentially memoryless) module in which component S directs the flow of information depending on whether the device is in bypass or filter mode. Component T is a simple merge component (probably just an ‘or’ gate in practice). Component U performs the filtering function, which normally involves excising messages being sent from the high-security domain to the low-security one if they are not in some dictionary of allowed messages. On the left of Figure 3 is a separate module containing an unmoded input buffer R which is used to smooth the flow of traffic into the filter.

The interesting aspect of this device is that, in isolation, both the left and right-hand modules have well-understood, ‘secure’ behaviours, but their composition in the presence of mode changes does not.

Table 7: Adjacency matrix for the cryptographic device using mode sequences

	a	b	c	d	e	f
a						
b					$\{\langle E \rangle\}$	
c					$\{\langle D \rangle\}$	
d						
e						$\{\langle \star, \star \rangle\}$
f	$\{\langle D \rangle\}$			$\{\langle E \rangle\}$		

Table 8: Connectivity matrix for the cryptographic device using mode sequences

	a	b	c	d	e	f
a						
b	$\{\langle E, D \rangle\}$			$\{\langle E \rangle\}$	$\{\langle E \rangle\}$	$\{\langle E, \star \rangle\}$
c	$\{\langle D \rangle\}$			$\{\langle D, E \rangle\}$	$\{\langle D \rangle\}$	$\{\langle D, \star \rangle\}$
d						
e	$\{\langle \star, D \rangle\}$			$\{\langle \star, E \rangle\}$		$\{\langle \star, \star \rangle\}$
f	$\{\langle D \rangle\}$			$\{\langle E \rangle\}$		

Table 9 shows our initial population of the adjacency matrix for this device using mode sequences. In particular, the (a, b) cell shows that component R can receive information in either mode B or F and release it in any mode.

(Putting $\langle \star, \star \rangle$ in this cell would have exactly the same meaning, since the only two modes are B and F , but explicitly enumerating the modes for cells that are on the outermost boundary of a diagram produces a more readable result. Another modelling decision in this table is that the evaluator has chosen to separately specify the two distinct modes in which information is *intended* to flow through component T , either from c to d in mode B or from f to d in mode F . Although reasonable, we could equally well take the view that this simple ‘or’ gate-like component is essentially unmoded and put both modes B and F in the (c, d) and (f, d) cells. This has no significant impact on the overall example, however.)

Table 10 then shows the calculated connectivity matrix. If we consider just the b to f rows and columns, we can see the overall connectivity for the right-hand module in Figure 3. In particular, the (b, d) cell tells us that this unbuffered module is ‘secure’ in the sense that information flows from end-to-end only in modes B or F , as we expect from such a device.

However, with the addition of buffered component R (i.e., the ‘ a ’ row and column in Table 10) we see that the device as a whole is *not* secure. Specifically, the $\langle F, B \rangle$ sequence in the (a, d) cell alerts us to the fact that information accumulated in component R while the device is in filter mode F can be subsequently released to the low-security domain when the device is switched into bypass mode B , and thus won’t get filtered as the operator intended.

Once again, therefore, the security evaluator will be obliged to carefully consider the device’s behaviour when it switches from filter to bypass mode. Although component S is the primary one concerned with the device’s overall mode, we must ensure that all data in component R is cleared when the mode is changed. If this cannot be proven by more detailed analysis of the device’s design it must be rejected as insecure.

8 Example: Clearing a Buffered Component

In the preceding examples we have assumed a worst-case scenario in which a buffered component may release any information acquired in previous modes. If, however, we know that the buffer is cleared when certain mode changes occur we can incorporate this information in the adjacency matrix and thus eliminate false positives from the connectivity analysis. Using sets of mode sequences in the cells makes this possible because we can selectively omit sequences from the sets when we know that this particular sequence of modes never allows information flow.

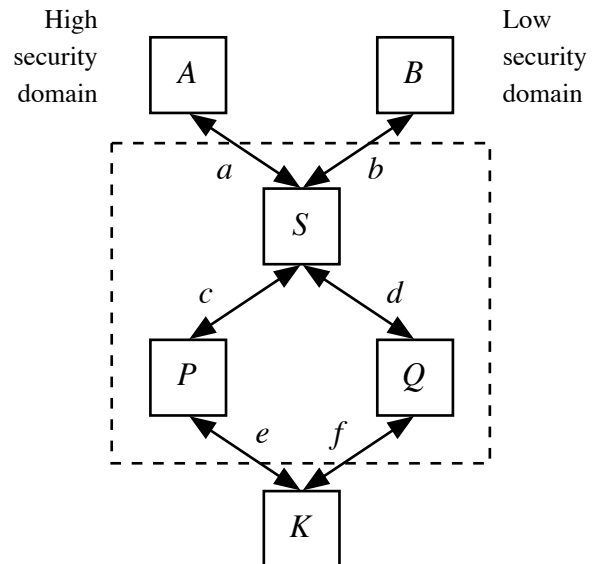


Figure 4: A keyboard switch

For example, consider the (highly abstract) design for a keyboard switch in Figure 4. Its purpose is to allow a single keyboard K to be shared between high-security computer A and low-security computer B . The switching device contains three components, a ‘mode switch’ S , and two microprocessors, P and Q , for interfacing between the keyboard and the high and low-security computers, respectively.

Table 9: Adjacency matrix for the filter device

	a	b	c	d	e	f
a		$\{\langle F, \star \rangle, \langle B, \star \rangle\}$				
b			$\{\langle B \rangle\}$		$\{\langle F \rangle\}$	
c				$\{\langle B \rangle\}$		
d						
e						$\{\langle F \rangle\}$
f				$\{\langle F \rangle\}$		

Table 10: Connectivity matrix for the filter device

	a	b	c	d	e	f
a		$\{\langle F, \star \rangle, \langle B, \star \rangle\}$	$\{\langle B \rangle, \langle F, B \rangle\}$	$\{\langle F \rangle, \langle B \rangle, \langle F, B \rangle, \langle B, F \rangle\}$	$\{\langle F \rangle, \langle B, F \rangle\}$	$\{\langle F \rangle, \langle B, F \rangle\}$
b			$\{\langle B \rangle\}$	$\{\langle B \rangle, \langle F \rangle\}$	$\{\langle F \rangle\}$	$\{\langle F \rangle\}$
c				$\{\langle B \rangle\}$		
d						
e				$\{\langle F \rangle\}$		$\{\langle F \rangle\}$
f				$\{\langle F \rangle\}$		

The device has two modes, H and L , where the keyboard is connected to the high-security computer and the low-security computer, respectively. The mode of choice is controlled by the operator, using a physical switch connected to component S , which in turn sends appropriate control signals (not shown) to the two microprocessors.

In such a device, component S does not physically disconnect the keyboard from the ‘disconnected’ computer, to avoid problems caused by the computer detecting that its keyboard has been unplugged. Instead, the microprocessor linked to the disconnected computer simulates the behaviour of the keyboard so that both computers believe they have a dedicated keyboard attached at all times. For instance, when the device is in high-security mode H , microprocessor P acts as an interface between the keyboard and computer A , while microprocessor Q simultaneously simulates an idle keyboard for computer B .

Primary information flow through the keyboard switch consists of keystrokes forwarded from the keyboard to whichever computer is ‘connected’, depending on the operating mode. However, the computers also send signals to the keyboard, to drive LED and LCD displays, maintain the ‘Caps-Lock’ status, and to store information about programmable function keys. This is why all arcs in Figure 4 are shown as bidirectional. For instance, when the device is in mode H , not only do keystrokes travel from component K to component A , via components P and S , but some data also travels from the high-security computer to the keyboard in the reverse direction.

The presence of even a small memory buffer in the keyboard thus introduces the danger of a covert channel via which classified information can be sent from the high-security computer to the keyboard while the device is in mode H and subsequently forwarded from the keyboard to the low-security computer when the device is in mode L . Indeed, undertaking a connectivity analysis of this device, using ‘wildcard connectors’ to model information flow through the keyboard, would reveal that all the components in this device are potentially connected.

In this example, however, we wish to introduce the additional knowledge that a safeguard has been built into the device. Assume that the device has been

engineered so that when it is switched from mode H to mode L , microprocessor P instructs the keyboard to clear all of its memory buffers.

We can incorporate this knowledge into our model as shown by the adjacency matrix in Table 11. Instead of putting ‘ $\langle \star, \star \rangle$ ’ connectors into the (e, f) and (f, e) cells, we have instead included only the possibility that information flows through the keyboard when the mode changes from L to H . Omitting the $\langle H, L \rangle$ sequence from these cells models our assumption that information flow is prevented when switching from mode H to mode L .

(Also note that component S is assumed to successfully keep the high and low-security data streams separate. There is no information flow between arcs a and d or between arcs b and c .)

The resulting transitive closure in Table 12 then shows that this device can be considered secure thanks to the assumption we have made about clearing the buffer. The (b, a) cell shows that information may flow from the low-security computer to the high-security one when the mode is switched from L to H , but this is harmless. Most importantly, the (a, b) cell is empty. Even though the (a, e) cell shows that information may flow from the high-security computer to the keyboard in mode H , and the (f, b) cell shows that information may flow from the keyboard to the low-security computer in mode L , our deliberate omission of a $\langle H, L \rangle$ link in the (e, f) cell successfully captured the notion that the keyboard does not leak classified data in this design.

9 Conclusion

Evaluating information flow through communications devices intended for secure applications is both tedious and intellectually taxing. Our research is devising ways of automating various aspects of the problem. In this short paper we have described an improvement to previous analysis techniques which will help highlight potential security problems in devices which possess both different operating modes and the ability to temporarily store classified information between modes.

Although calculation of connectivity matrices is readily automatable, population of the adjacency ma-

Table 11: Adjacency matrix for the keyboard switch

	a	b	c	d	e	f
a			$\{\langle H \rangle\}$			
b				$\{\langle L \rangle\}$		
c	$\{\langle H \rangle\}$				$\{\langle H \rangle\}$	
d		$\{\langle L \rangle\}$				$\{\langle L \rangle\}$
e			$\{\langle H \rangle\}$			$\{\langle L, H \rangle\}$
f				$\{\langle L \rangle\}$	$\{\langle L, H \rangle\}$	

Table 12: Connectivity matrix for the keyboard switch

	a	b	c	d	e	f
a			$\{\langle H \rangle\}$		$\{\langle H \rangle\}$	
b	$\{\langle L, H \rangle\}$		$\{\langle L, H \rangle\}$	$\{\langle L \rangle\}$	$\{\langle L, H \rangle\}$	$\{\langle L \rangle\}$
c	$\{\langle H \rangle\}$				$\{\langle H \rangle\}$	
d	$\{\langle L, H \rangle\}$	$\{\langle L \rangle\}$	$\{\langle L, H \rangle\}$		$\{\langle L, H \rangle\}$	$\{\langle L \rangle\}$
e	$\{\langle H \rangle\}$		$\{\langle H \rangle\}$			$\{\langle L, H \rangle\}$
f	$\{\langle L, H \rangle\}$	$\{\langle L \rangle\}$	$\{\langle L, H \rangle\}$	$\{\langle L \rangle\}$	$\{\langle L, H \rangle\}$	

trix used as a starting point inevitably relies on a degree of skill from the security evaluator to model assumed device characteristics properly. Including too many modes in the cells will produce false positives, while omitting modes in which information is actually transmitted through a component runs the more serious risk of overlooking security problems. If in doubt about how to model a particular component, therefore, the evaluator would be best advised to put in too many modes, rather than too few.

All of the examples presented above were trivial block-diagram abstractions of the types of devices we normally consider. In practice, information security evaluations are based on circuitry schematics with dozens of distinct components. Such evaluations are practical only with tool support. We have already built a software tool capable of analysing circuit-level system descriptions (McComb & Wildman 2005) and now plan to extend it with the analysis capability described above.

In future work, this general area of research can be extended in two directions. Firstly, since we treat circuitry merely as a connectivity graph, the approach can also be applied to whole communications networks, not just single communications devices. Also, we plan to extend the type of analysis described above to embedded software running on microprocessors within communications devices, since embedded code has a major influence on the transfer of information between modes.

References

- Bishop, M. (2003), *Computer Security: Art and Science*, Addison-Wesley.
- The Common Criteria Project Sponsoring Organizations (1999), *Common Criteria for Information Technology Security Evaluation*, 2.1 edn. ISO/IEC Standard 15408.
- Herrmann, D. S. (2003), *Using the Common Criteria for IT Security Evaluation*, Auerbach Publications.
- McComb, T. & Wildman, L. P. (2005), SIFA: A tool for evaluation of high-grade security devices, in

C. Boyd & J. Nieto, eds, ‘Information Security and Privacy: Tenth Australasian Conference (ACISP 2005)’, Vol. 3574 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 230–241.

- Miller, S. P. (1998), Specifying the mode logic of a flight guidance system in CoRE and SCR, in M. Ardis, ed., ‘Proceedings of FMSP’98: The Second Workshop on Formal Methods in Software Practice’, ACM Press, pp. 44–53.
- Paynter, S. (1996), Real-time mode-machines, in B. Jonsson & J. Parrow, eds, ‘Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT’96)’, Vol. 1135 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 90–109.
- Rae, A. J. & Fidge, C. J. (2005a), ‘Identifying critical components during information security evaluations’, *Journal of Research and Practice in Information Technology* **37**(4), 391–402.
- Rae, A. J. & Fidge, C. J. (2005b), ‘Information flow analysis for fail-secure devices’, *The Computer Journal* **48**(1), 17–26.