Information-Theoretic Modeling and Analysis of Interrupt-Related Covert Channels

Heiko Mantel and Henning Sudbrock*

Department of Computer Science, TU Darmstadt,
Hochschulstraße 10, 64289 Darmstadt, Germany
{mantel,sudbrock}@cs.tu-darmstadt.de

Abstract. We present a formal model for analyzing the bandwidth of covert channels. The focus is on channels that exploit interrupt-driven communication, which have been shown to pose a serious threat in practical experiments. Our work builds on our earlier model [1], which we used to compare the effectiveness of different countermeasures against such channels. The main novel contribution of this article is an approach to exploiting detailed knowledge about a given channel in order to make the bandwidth analysis more precise.

1 Introduction

Confidentiality and integrity on the application level heavily depend on mechanisms to restrict communication in underlying system layers. Even if one closes all communication channels between two applications, the danger of covert communication remains, i.e., that there are channels that are not intended as communication channels [2]. The problem of identifying covert channels and analyzing their bandwidth has received much attention by the research community (see, e.g., [3–7]), but covert channel analysis and mitigation is far from being solved.

Covert channels can be established based on various system-level resources that are virtualized or otherwise shared between multiple processes. For instance, the physical memory can be exploited to establish a covert channel as follows: a sender sends a signal by heavily allocating memory, and a receiver decides whether a signal was sent or not, depending on the paging rate that he observes when allocating memory. In this article, we focus on *interrupt-related covert channels*, i.e., covert channels that are established based on the CPU time used for handling interrupts. Unlike many other covert channels, interrupt-related channels cannot be mitigated by assigning a constant quota of resource usage to each process. Therefore, it is of particular interest to obtain reliable upper bounds on the bandwidth of such channels. More generally, if one cannot mitigate some covert channels then one should at least be able to assess their dangers.

In this article, we investigate the information-theoretic modeling of interrupt-related covert channels. The main novel contribution is an approach to exploiting detailed knowledge about a given channel in order to make the bandwidth analysis more precise. This contribution is twofold: On the one hand, we demonstrate how to refine a model

* The authors gratefully acknowledge support by the German Research Foundation (DFG). The authors furthermore thank the anonymous reviewers for their helpful comments.

Appeared in P. Degano et al. (Ed.): Preproceedings of FAST 2008; © Springer-Verlag (to be transferred)

based on detailed knowledge about a given channel, and we show at two concrete examples that such refinements can result in significantly more precise upper bounds on the bandwidth. On the other hand, we show that even in the case when the knowledge needed for a refinement is incomplete, the available knowledge can still be exploited to improve the bandwidth analysis in a significant way.

In earlier work, we investigated several mechanisms to reduce the bandwidth of interrupt-related covert channels [1]. The model employed in this article constitutes an improvement over the earlier model as it faithfully reflects the capabilities of senders and receivers and is therefore suitable for obtaining reliable upper bounds on the bandwidth of interrupt-related covert channels.

To our knowledge, there is no prior work on refining models of covert channels by exploiting knowledge about probability distributions to improve the bandwidth analysis. Our work is the first to exploit incomplete knowledge about probability distributions in the bandwidth analysis of covert channels, in general. We have also implemented an exploit of interrupt-related channels that works in practice. Using a simple ad hoc encoding the exploit allows to transmit a four-digit PIN (i.e., approximately 13 bits of information) in about 30 seconds. However, the focus of the current article is not on such practical exploitations but rather on sound techniques for analyzing the bandwidth.

2 Modeling Interrupt-Related Covert Channels

In this section, we present the information-theoretic model for analyzing the bandwidth of covert channels. Before presenting the model, we recall how interrupt-related channels operate and select a class of such channels as a running example.

2.1 Interrupt-Related Covert Channels

The transmission of information over an *interrupt-related covert channel* is based on operations that result in asynchronous interrupt requests. The *receiving process* continuously probes a clock during its execution in order to notice when it has been preempted by an interrupt request. For instance, the observation that it was preempted at least once during a given time-slot could be interpreted as the value 1 and that it was not preempted as the value 0. To send a 0 over this channel, the *sending process* only needs to refrain from executing operations that result in interrupt requests during the receiving process' time-slot and, to send a 1, it performs such operations. Such an interrupt-related channel cannot be mitigated by assigning a constant quota of resource usage to each process [1], a technique that can be used to mitigate many other types of covert channels.

Interrupt-related covert channels can be established based on many different operations and various hardware devices. To make things concrete, we focus throughout this article on channels that are based on the transmission of packets via a network interface card (NIC). We call such an interrupt-related channel an *NIC-channel*.

More concretely, we consider an NIC that requests interrupts on two occasions: after a packet has been transmitted to the network and after a packet has been received from the network. An interrupt request causes the CPU to suspend its current activity in order to execute an interrupt handler. This behavior can be exploited to establish a

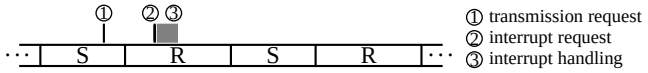


Fig. 1. Covert transmission of a single bit via an NIC-channel

covert channel as follows: The sending process requests the transmission of a packet via the NIC. After the NIC has transmitted the packet, it acknowledges the transmission by an interrupt request. The handling of this interrupt will occur during the receiving process' time-slot if the transmission request is issued at the proper time by the sending process. We illustrate the transmission of a single bit in a simple example scenario where the sending process and the receiving process are scheduled alternately, and no other processes are active. In Figure 1, each box in the time-line represents a time-slot of the process indicated by the label inside the box, i.e., by S and R for the sending process and the receiving process, respectively. Above the time-line, the occurrence of a transmission request and the occurrence of an interrupt request are indicated by the two vertical bars labeled with 1 and 2, respectively. The time interval used for handling the interrupt is indicated by the gray rectangle labeled with 3.

2.2 The Information-Theoretic Model

We model each NIC-channel as a discrete, memoryless channel C, i.e., as a triple (I_C, O_C, P_C). Further information about discrete, memoryless channels can be found in, e.g., [8]. The *input alphabet* I_C models the set of values that can be sent on C, and the *output alphabet* O_C models the set of values that can be received from C. The *channel matrix* $P_C = (p(o|i))_{i \in I_C, o \in O_C}$ defines the probability $p(o|i)$ that a given output o is received given that the input equals i. We first model the transmission from a given time-slot of the sender to a given time-slot of the receiver before we generalize the setting to multiple time-slots. In the model, we measure time abstractly in discrete *time units*, which could, e.g., be microseconds or clock cycles, and assume that all time-slots of the sending process and the receiving process have a fixed length of l time units.

The input alphabet. An input to the channel corresponds to the points in time at which the sending process requests the transmission of a network packet. Formally, an input is an ordered list $[t_1, \ldots, t_k]$, where each element t_i represents a transmission request at time t_i in the sending process' time-slot. We measure time relative to the starting point of the given time-slot and require $t_i \leq l$ for all elements of the list, where l denotes the number of time units in a process' time-slot. The input alphabet is

$$I_C = \{[t_1, \ldots, t_k] \mid \forall i \in \{1, \ldots, k-1\}.\, 1 \leq t_i < t_{i+1} \leq l\}.$$

As an example, the input symbol $i = [2, 7, 8]$ is illustrated by the diagram on the left hand side of Figure 2. The three vertical bars on top of the sending process' time-slot represent the transmission requests after 2, 7, and 8 time units, respectively. An alternative, but equivalent representation of inputs are bit strings of length l.

The output alphabet. An output symbol contains information about interrupt handling in the receiving process' time-slot. For each interrupt request that is handled during that

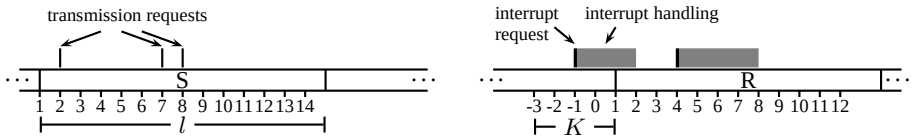


Fig. 2. Input symbol $[2, 7, 8]$ (left diagram) and output symbol $[(-1, 3), (4, 4)]$ (right diagram)

time-slot, the output symbol contains a pair (s, d), where s represents the time at which the interrupt request occurs, and d represents the duration of handling this request. This modeling is conservative as output symbols contain the information when interrupts occur and how long their handling takes, while the receiver only sees during which time intervals it was not running on the CPU. We use this approach as it allows us to model the correspondence between input and output symbols independently of how the system deals with interrupts that are requested during the handling of another interrupt request.

We formalize an output symbol as an ordered list $[(s_1, d_1), \ldots, (s_k, d_k)]$:

$$O_C = \{\, [(s_1, d_1), \ldots, (s_k, d_k)] \mid \\ \forall i \in \{1, \ldots, k-1\} \,.\, (-K < s_i \leq s_{i+1} \leq l \,\wedge\, (s_i = s_{i+1} \Rightarrow d_i \leq d_{i+1}))\},$$

where K is the maximal amount of time that interrupt handling may take. Note that the lists in the output alphabet are ordered in the sense that either $s_i < s_{i+1}$ or $s_i = s_{i+1} \wedge d_i \leq d_{i+1}$ holds for $i \in \{1, \ldots, k-1\}$. If $s_i = s_j$ holds for $i \neq j$ then this represents that the two interrupt requests occur at the same time. Note that we demand $-K < s_i$ instead of $0 < s_i$ and $s_{i+1} \leq l$ instead of $s_{i+1} + d_{i+1} \leq l$. This ensures an adequate treatment of interrupt requests that occur at the boundaries of the receiving process' time-slot. That is, our output symbols include interrupt requests that occur before the receiving process' time-slot but that are handled at least partially during this time-slot as well as interrupts whose handling exceeds the receiving process' time-slot.

As an example, the output symbol $[(-1, 3), (4, 4)]$ is represented by the diagram on the right-hand side of Figure 2. In the diagram, the vertical bars represent the occurrence of interrupt requests, and the gray boxes represent the time used for interrupt handling.

The channel matrix. While the input alphabet and the output alphabet can be defined for NIC-channels in general, the channel matrix must be defined dependent on the particular instance of an NIC-channel. We illustrate this for a simple example channel.

Example 1. Assume a scenario where the sending process and the receiving process are scheduled alternately in a Round-Robin fashion and no other processes are active. The length of each time-slot shall be 100 milliseconds, the latency of the NIC shall be 10 milliseconds (i.e. an interrupt request occurs exactly 10 milliseconds after a given transmission request), and handling a single interrupt shall take exactly 1 millisecond.

We choose milliseconds as granularity of time in the model, i.e. one time unit in the model corresponds to one millisecond in reality. At each time unit, the sending process either requests a transmission or not. Given an input symbol $[t_1, \ldots, t_i, t_{i+1}, \ldots, t_{i+k}]$ with $t_i \leq l - 10$ and $t_{i+1} > l - 10$, the transmission requests at $t_1, \ldots, t_i$ do not result in interrupt requests in the receiver's time-slot and, hence, can be ignored in an input symbol. Given an input $[t_1, \ldots, t_k]$ with $t_1 > l - 10$ and $k \leq 10$, the receiver observes

the output $[(t_1 - l + 10, 1), \ldots, (t_k - l + 10, 1)]$, which means that 2^{10} different output symbols can be generated. The channel matrix can then be defined as follows:

$$P_C[i,o] = p(o|i) = \begin{cases} 1 & \text{, if } i = [t_1, \ldots, t_i, t_{i+1}, \ldots, t_{i+k}],\ k \leq 10, \\ & t_i \leq l - 10,\ t_{i+1} > l - 10, \\ & \text{and } o = [(t_{i+1} - l + 10, 1), \ldots, (t_{i+k} - l + 10, 1)] \\ 0 & \text{, otherwise.} \end{cases}$$

The channel in the prior example features a functional dependency between input and output. That is, for a given input $i \in I_C$, the corresponding output $o(i) \in O_C$ and a given list element (s_j, d_j) in the output $o(i)$, there is exactly one corresponding list element t_k in the input i. This is reflected in the model by the fact that no other values than 0 and 1 occur as entries in the channel matrix.[1]

Bandwidth analysis. The capacity *CAP*(C) of a discrete, memoryless channel C is defined as an upper bound on the amount of information (in number of bits) that can be transmitted over C on average with an arbitrarily small error probability. For the formal definition of capacity and of other basic concepts of information theory see, e.g., [8].

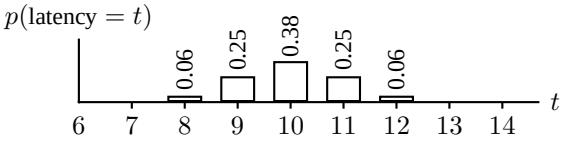
Example 2. In the scenario from Example 1, 2^{10} different output symbols can be generated. Each of these symbols can be generated by sending an input symbol $[t_1, \ldots, t_k]$ with $t_1 > l - 10$ and $k \leq 10$. Hence, we obtain a capacity of 10 bits per scheduler round for our simple example channel.

For a contemporary NIC and scheduler, it is plausible that a given time-slot of the receiving process is only influenced by transmissions in the immediately preceding time-slot of the sending process. This observation allows us to generalize the model to multiple scheduler rounds.

Example 3. In Example 1, a scheduling round consists of a time-slot of each of the two processes. Hence, 5 scheduler rounds occur every second. Since the capacity is 10 bits per scheduler round (see Example 2), we obtain a capacity of 50 bits per second.

In Examples 1–3, the latency of the NIC as well as the handling time of an interrupt request are constant. Random effects influencing these time values can be taken into account by an appropriate definition of the channel matrix.

Example 4. Reconsider the scenario from Examples 1–3, but with a non-constant latency of the NIC. We assume that the latency of the NIC is between 8 and 12 milliseconds, where the probability that the latency equals t milliseconds is denoted as $p(\text{latency} = t)$. For the probability distribution depicted in the following graph, we obtain a capacity of 2.3 bits per scheduler round, or equivalently 11.5 bits per second.[2]



[1] Examples of channels where the dependency between input and output is not functional are given in Example 4 as well as in Section 3.2.

[2] Due to space restrictions, the formalization of the probabilities $P_C[i,o]$ used in the analysis is provided in an appendix to this article which is available on the second author's website (`http://www.mais.informatik.tu-darmstadt.de/FAST08-app.html`).

If the number of processes and the behavior of the interrupt mechanism remain constant over time, like in Examples 3 and 4, then the capacity per second can be calculated by the formula $\mathit{CAP}(C) * \frac{1}{L*(n+2)}$ where n is the number of active processes besides sender and receiver and L is the duration of a single time-slot in seconds. In a more dynamic setting where, e.g., the number of processes or the dependency between input and output changes over time, one would need to perform a more complicated calculation, possibly having to adapt the channel matrix between individual scheduler rounds.

Remark 1. In [1], we defined an information-theoretic model in order to analyze the effectiveness of various countermeasures against interrupt-related channels. In this model, we would obtain a capacity of approximately 3.5 bits per scheduler round for the scenario from Examples 1–3 (see Example 5 in [1]). This significant difference in the capacity is due to a somewhat ad hoc simplification in the model that results in an inaccurate treatment of the receiving process' capabilities. In our earlier model, the receiver could observe less than he can observe in reality and, therefore, the bandwidth analysis resulted in a capacity that is too low. More concretely, we assumed that the receiving process could only perceive the accumulated delay caused by all interrupts that occur in a given time-slot. Unlike our earlier model, the model proposed in this article (including all refinements presented in the subsequent sections) provides a suitable basis for determining reliable upper bounds on the bandwidth as it reflects the capabilities of senders and receivers in an adequate way.

3 Exploiting Additional Knowledge in a Bandwidth Analysis

Additional knowledge about a particular NIC-channel can be exploited in the bandwidth analysis. In this section, we demonstrate how our information-theoretic model can be refined based on such knowledge. We illustrate how to refine the model with two examples: the first exploits knowledge about the peculiarities of the NIC and the second exploits information about the run-time environment of the sender and receiver. In each case, the objective of refining the model is to increase the precision of the bandwidth analysis. The fact that the model from Section 2 is conservative already guarantees that the calculated bandwidths constitute reliable upper bounds. As we demonstrate by concrete examples, refinements of the model can lead to significant increases in precision.

3.1 Exploiting Peculiarities of the Network Interface Card

In Section 2, we made the rather conservative assumption that the sender can request a transmission at each time unit during its time-slot. Let us now consider an NIC that requires that two subsequent transmission requests are at least T_{tr} time units apart. Consequently, the sender can only generate input symbols from the following subset:

$$I'_C = \{[t_1, \ldots, t_k] \mid \forall i \in \{1, \ldots, k-1\}.\, 1 \leq t_i < t_{i+1} \leq l \wedge\ t_{i+1} - t_i \geq T_{tr}\} \subseteq I_C.$$

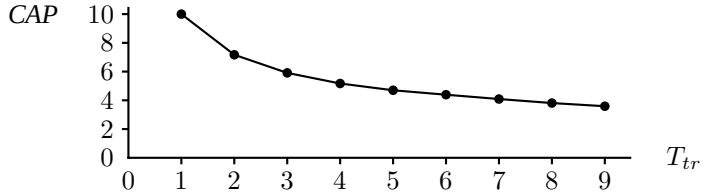
The number of ordered lists over a set $\{1, \ldots, j\}$ where two adjacent elements in a list have at least a distance of T_{tr} time units can be computed recursively as follows:

$$\begin{aligned} N(j) &= j + 1 && \text{, if } j \leq T_{tr} \\ N(j) &= N(j - T_{tr}) + N(j-1) && \text{, if } j > T_{tr} \end{aligned}$$

The first equation above reflects the fact that there cannot be two elements in $\{1, \ldots, j\}$ with a distance of at least T_{tr} if $j \leq T_{tr}$ holds. In this case, the empty list $[\,]$ and the singleton lists $[1], \ldots, [j]$ are the only ordered lists that satisfy the given constraints. The second equation reflects the fact that the set of ordered lists can be partitioned into the following two subsets: the lists in which j occurs as the last element (hence, there are $N(j - T_{tr})$ possibilities for the prefix of a list without this last element) and the lists in which j does not occur (hence, there are $N(j-1)$ possibilities for such lists).

Example 5. We consider the same scenario as in Examples 1–3 (i.e. $l = 100$ and $K = 1$), but with the additional knowledge that two subsequent transmission requests must be at least two time units apart (i.e. $T_{tr} = 2$). Like before, an input symbol $[t_1, \ldots, t_i, t_{i+1} \ldots, t_{i+k}]$ with $t_j \leq l - 10$, $t_{j+1} > l - 10$, and $k \leq 10$ results in the output symbol $[(t_{i+1} - l + 10, 1), \ldots, (t_{i+k} - l + 10, 1)]$. Note that the set of output symbols that can actually occur is only a proper subset of the one in Example 1. The cardinality of this subset equals $N(10)$ (for $T_{tr} = 2$) because only transmission requests in the last 10 milliseconds of the sender's time-slot are relevant and because two transmission requests must be at least 2 milliseconds apart. That is, we obtain a bandwidth of $\log_2(N(10)) = \log_2(144) \approx 7.2$ bits per scheduler round.

The above example demonstrates that our refinement of the model leads to an upper bound on the bandwidth that is significantly lower (reduction by more than 25%) than in the model from Section 2. This justifies the slightly more complex model that results from taking restrictions on the sender side into account. Note that the difference in the bandwidth becomes even greater if larger values of T_{tr} are used. The capacities for $T_{tr} \in \{1, \ldots, 9\}$ are depicted in in the following graph. For instance, for $T_{tr} = 6$, we obtain a capacity of approximately 4.4 bits per scheduler round.



Remark 2. Refinements of the model, like the one just presented, must be constructed with care. In particular, one must be careful to not endanger the conservativity of the model. Otherwise, one could obtain bandwidths that are no reliable upper bounds.

We illustrate this by an example. To this end, we consider the same scenario as in Example 5, but define the set of input symbols that can be generated less carefully than before. To simplify the bandwidth analysis, we assume that transmission requests occur only at odd time units. This ensures that two subsequent transmission requests are at least two time units apart. Under this assumption, only elements from the following subset of I_C can be generated:

$$I''_C = \{[t_1, \ldots, t_k] \mid \forall i \in \{1, \ldots, k-1\}.\, 1 \leq t_i < t_{i+1} \leq l \wedge\ t_i, t_{i+1} \text{ are odd}\}.$$

Under this assumption, the set of output symbols that can be generated is

$$O''_C = \{[(t_1, 1), \ldots, (t_k, 1)] \mid \forall i \in \{1, \ldots, k-1\}.\, 1 \leq t_i < t_{i+1} \leq 10 \wedge\ t_i, t_{i+1} \text{odd}\}.$$

This set contains 2^5 different elements. Hence, we obtain a capacity of 5 bits per scheduler round. Note how much easier it is to determine the bandwidth here in comparison to Example 5, where we had to employ recursive equations. However, the calculated capacity does not constitute a reliable upper bound on the bandwidth of the channel because there are output symbols (e.g., $[(2,1),(4,1)]$) that can be generated under the given assumptions in reality, but that do not occur in O''_C. The deviation from the correct result is significant (5 bits instead of 7.2 bits in Example 5) and, hence, not acceptable.

This illustrates how careful one must be when exploiting additional knowledge about a channel in the bandwidth analysis. In particular, one should avoid ad hoc refinements that simplify the analysis technically because, otherwise, one might obtain significantly inaccurate results that do not constitute upper bounds on the bandwidth.

3.2 Exploiting Knowledge about Noise

So far, we only considered those interrupt requests in a time-slot of the receiver that are caused by transmission requests of the sender. In reality, however, other sources of interrupts could introduce noise into the communication and, thereby, lower the bandwidth of the covert channel. For instance, any hardware device that uses interrupt-driven communication could add to the noise. This includes the NIC, which also generates interrupt requests to signal other events than the successful transmission of a packet to the network. To simplify the presentation in the following, we focus on noise due to interrupts that the NIC requests to signal that a packet has arrived from the network.

Since the occurrences of additional interrupt requests depend on the behavior of the environment of the system, for instance on other clients attached to the network, the receiving process, in general, does not know in advance when interrupt requests caused by arriving network packets are handled during its time-slot. To model the occurrences of these interrupt requests, we use a random variable Env, which takes values in the set

$$\{[(r_1,d_1),\ldots,(r_k,d_k)] \mid \forall i \in \{1,\ldots,k-1\}.\, -K < r_i < r_{i+1} \leq l\},$$

where the list $[(r_1,d_1),\ldots,(r_k,d_k)]$ represents occurrences of interrupt requests caused by arriving network packets at the times $r_1,\ldots,r_k$ with respect to the receiving process' time-slot, and the interrupt request at time r_i is handled in d_i time units.

The probabilistic dependency of a channel's output on the input is captured by the probability matrix P_C in our model of the channel. If no noise disturbs the transmission over the channel, the dependency between input symbols and output symbols is functional, like in Examples 1–3. In the presence of noise, we usually loose this functional dependency. For instance, the input symbol $[91]$ could result in the output $[(1,1)]$, which is the only possible output in the scenario without noise from Examples 1–3, but it could also result, e.g., in the output $[(1,1),(3,1)]$ where the second list element represents an interrupt request that signals the arrival of a network packet at time 3.

We introduce some notation for the definition of probability matrices. We write $o_1 \subseteq o_2$ to express that the list o_2 contains all occurrences of elements in the list o_1. We use $o_2 \ominus o_1$ to denote the list that results by removing one occurrence of an element from the list o_2 for each occurrence of this element in o_1, given that the element occurs in o_2 (e.g., $[(1,1),(1,1),(3,1)] \ominus [(1,1)] = [(1,1),(3,1)]$). For a given input $i \in I_C$

and output $o \in O_C$, one can determine the list of all elements in o that are caused by transmission requests in i. We use $o(i) \in O_C$ to denote this sublist of o.

If an output symbol $o \in O_C$ occurs for a given input $i \in I_C$, then $o(i)$ denotes the sublist of o that contains all interrupts requests that are generated by transmission requests in i. Hence, all interrupt requests in $o \ominus o(i)$ must be caused by noise, i.e. by the arrival of packets from the network. Therefore, we define

$$P_C[i,o] = p(o|i) = \begin{cases} 0 & \text{, if } o(i) \not\subseteq o, \\ p(Env = o \ominus o(i)) & \text{, if } o(i) \subseteq o. \end{cases}$$

Note that $o(i) \not\subseteq o$ implies that o was not generated by i. Hence, $P_C[i,o] = 0$ must hold.

Example 6. We consider the scenario from Examples 1–3, but admit noise due to interrupt requests caused by the arrival of packets from the network. We assume that 3 network packets arrive, on average, during a 10 millisecond interval, and that each of the corresponding interrupt requests is handled within 1 millisecond. Additionally, we assume that the probability that the environment causes an interrupt request at a given time unit is independent of the same probability for another time unit, i.e.,

$$p(Env = [(r_1,1),\ldots,(r_k,1)]) = 0.3^k * (1-0.3)^{l-k}.$$

For this scenario, we obtain a capacity of approximately 5.6 bits per scheduler round.[3]

Example 6 illustrates that using a more complex model, in which interrupt requests caused by the arrival of network packets are taken into account, can result in a significantly more precise upper bound on the bandwidth. Compared to Example 1, where we obtained an upper bound of 10 bits per scheduler round, the upper bound on the bandwidth is decreased by 4.4 bits per scheduler round, i.e., by 44%.

Remark 3. One must be careful when choosing a probability distribution for the random variable Env because an inappropriate choice may result in significant errors in the calculation. To illustrate this, we reconsider Example 6, but assume that it is likely that directly after the arrival of a packet from the network, further packets arrive. In such a scenario, the occurrence of an interrupt that signals the arrival of a packet increases the likelihood that further interrupts occur immediately afterwards. Therefore, we cannot assume anymore that the probability that noise occurs at one time unit is independent from the occurrence of noise at other points in time (which we assumed in Example 6).

Assume, for instance, that packets were always arriving in batches of size 2. Then one obtains a capacity of approximately 8.7 bits per scheduler round instead of 5.6 bits. That is, the effect of noise on the bandwidth is significantly reduced. However, note also that even if the effect of noise is limited in this way, we still obtain some increase in the precision of the calculated bandwidth (8.7 bits instead of 10 bits).

[3] An analytical computation of the capacity in this scenario is too difficult. For an approximation, we compute capacities here, and in the remainder of the article, using an algorithm due to Arimoto and Blahut [9, 10] that allows the computation of arbitrarily precise approximations of the capacity of discrete, memoryless channels. The computations are performed using the authors' straightforward Java implementation of the algorithm.

4 Effects on the Analysis of a Countermeasure

In Section 3, we demonstrated that improving the information-theoretic model can have quite significant effects on the results of a bandwidth analysis. Interestingly, modifications to the model not only have an effect on the capacity of the channel, but can also influence the evaluation of the effectiveness of countermeasures against interrupt-related channels. In the current section, we investigate this second effect at the example of *interrupt-rate limiting*. We base our investigation on an earlier evaluation and comparison which covered interrupt-rate limiting and five other countermeasures [1].

Interrupt-rate limiting mitigates NIC-channels by interrupting the CPU less frequently. The technique is used in network interface cards, where interrupt requests are delayed until a certain number v of packets has arrived, rather than requesting an interrupt for each packet (see, e.g., [11]). The evaluation of interrupt-rate limiting as a countermeasure against covert channels in [1] led to two conclusions:

- The countermeasure is capable to reduce the bandwidth arbitrarily close towards 0.
- For high values of v, the capacity is decreasing only slowly.

Both of these observations remain valid for the improved information-theoretic model that we proposed in Section 2. Nevertheless, the modifications to the model can still have an observable effect on the effectiveness of the countermeasure, at least in some cases. We illustrate this more concretely in the remainder of this section.

Effects of modeling the receiver adequately. To elaborate the effects of changing the output alphabet on the effectiveness of interrupt-rate limiting, let us revisit the scenario from Examples 1–3. The analysis shows that, at least for this scenario, the change of the output alphabet has a significant effect on the effectiveness of interrupt-rate limiting.

For the analysis, we integrate interrupt-rate limiting into the model by defining an appropriate channel matrix. The definition of the probabilities $P_C[i, o]$ is fairly straightforward. The only subtlety results from the need to take into account that an unknown number of network packets is pending at the NIC at the beginning of the receiving process' time-slot. Like in [1], we assume that the number of pending interrupts is uniformly distributed on the set $\{0, \ldots, v - 1\}$.

The analysis results for the model using the refined output alphabet from Section 2 as well as for the earlier model from [1] are displayed in Figure 3. The solid dots ($\bullet$) represent the resulting capacities for the refined output alphabet from Section 2, and the circles ($\circ$) those for the earlier model. When increasing the parameter v for small values, the reduction of the capacity is significantly stronger for the model from Section 2. For instance, increasing the value of v from 1 to 2 reduces the capacity by 39% in the refined model, but only by 27% in the earlier model. For larger values of v (i.e. for $v \geq 10$), interrupt-rate limiting reduces the capacity in both models nearly to the same level.

Effects of exploiting noise and peculiarities of the NIC in the model. Unlike the refinement of the output alphabet from Section 2, the two refinements from Section 3 do not have an observable effect on the effectiveness of interrupt-rate limiting. In both scenarios, we could not identify any significant differences in the reduction of the capacity when applying interrupt-rate limiting.

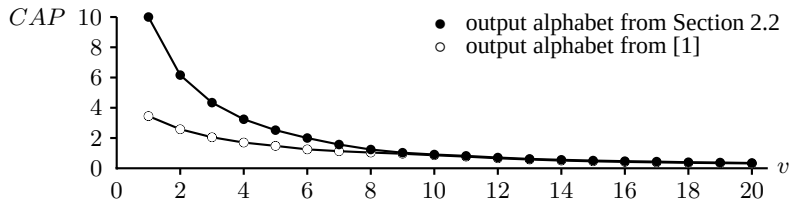


Fig. 3. Effect of the refined output alphabet on the analysis of interrupt-rate limiting

5 Incomplete Knowledge of Probabilistic Behavior

In Section 3, we demonstrated how additional knowledge about a particular covert channel can be exploited to increase the precision of the bandwidth analysis. When exploiting the bandwidth reduction due to noise in Section 3.2, we assumed the probability distribution to be known for the additional interrupts that disturb the covert communication. However, one can easily imagine cases where only incomplete information about the probability distributions is available. For instance, one might only know the number of packets that arrive from the network on average, but not what the exact probability distribution is. In such a scenario, one should refrain from simply choosing one arbitrary probability distribution from the set of possible distributions. Such an ad-hoc choice could lead to significant errors in the resulting capacity, as the difference between the capacities obtained in Example 6 and in Remark 2 prove (5.6 bits versus 8.7 bits per scheduler round). This raises the question whether there is any possibility to obtain reliable upper bounds on the bandwidth when exploiting incomplete knowledge about probability distributions for noise in order to increase the precision of the capacity analysis. In this section, we give an affirmative answer to this question and show how incomplete knowledge about probability distributions can be exploited.

The following example illustrates how reliable upper bounds can be achieved by performing the analysis in multiple instances of the refined model from Section 3.2. For the used approach it is essential that the refinement from Section 3.2 is parametric in the probability distribution of the noise represented by the random variable Env.

Example 7. We consider a scenario, where an interrupt request occurs exactly t time units after a given transmission request. Handling an interrupt request shall take exactly one time unit. Furthermore, on average, x interrupt requests are caused by arriving network packets during the first t time units of the receiving process' time-slot.

Without considering the additional information about arriving network packets, we obtain an upper bound on the capacity of t bits per scheduler round.

To obtain a more precise upper bound by using the additional knowledge, we denote with $\#_{Env}$ the number of interrupt requests caused by arriving network packets during the first t time units of the receiving process' time-slot, and with $\mu(\#_{Env})$ the mean value of $\#_{Env}$. In the scenario under consideration $\mu(\#_{Env})$ equals x. For different probability distributions of the random variable Env (see Section 3.2), $\mu(\#_{Env})$ takes different values. We denote with D the set of all possible probability distributions of the random variable Env for which $\mu(\#_{Env}) = x$. With the given information that, on average, x interrupt requests are caused by arriving network packets during the first

t time units of the receiving process' time-slot, the distribution of the random variable Env could be any probability distribution in the set D.

Using the refined model from Section 3.2, we can compute an upper bound on the bandwidth for each $d \in D$. Note that when D is too large it becomes infeasible to calculate the upper bounds for all $d \in D$ in this fashion. We denote the upper bound obtained under the assumption that Env is distributed according to d with $\mathrm{UB}(d)$. Then the maximal value of $\mathrm{UB}(d)$, for $d \in D$, is an upper bound on the bandwidth of the channel, as this is the highest upper bound for those probability distributions of Env for which $\mu(\#_{Env}) = x$.

Assume that $t = 2$ and $x = 0.8$. In this case, the set D is small enough such that the individual computation of $\mathrm{UB}(d)$ for all $d \in D$ is feasible. To compute these values, we implemented the analysis for the refined model from Section 3.2 for arbitrary distributions of Env. In the implementation the probabilities $p(Env = [(r_1, 1), \ldots, (r_k, 1)])$ are represented with a precision of four digits. To compute the upper bound, we needed to compute the capacity for approximately 160.000 instances of the model. The resulting upper bound (which equals the maximum of all computed capacities) equals approximately 1.7 bits per scheduler round.

The preceding example demonstrates that the consideration of additional knowledge that is not sufficient to instantiate the refined model from Section 3.2 still allows one to obtain more precise upper bounds. Compared to an analysis where the additional knowledge is not taken into account, the obtained upper bound is reduced by 15%.

The following example illustrates that exploiting further information can result in further improvements of the upper bound.

Example 8. We consider the scenario from Example 7, but assume in addition that the variance of the random variable $\#_{Env}$ equals 0.4. Using this additional information, we denote with D' the set containing all probability distributions of Env where the mean value of $\#_{Env}$ equals 0.8 and the variance of $\#_{Env}$ equals 0.4. By determining the maximal value of $\mathrm{UB}(d)$ for $d \in D'$, we obtain an upper bound on the bandwidth equal to approximately 1.3 bits per scheduler round.

Remark 4. When we consider the scenario from Example 7, the size of the set D for which $\mathrm{UB}(d)$ needs to be computed during the analysis grows exponentially when t is increased. This is because interrupt requests during the first t time units in the receiving process' time-slot can be caused by the sending process, therefore interrupt requests caused by arriving network packets that occur up to t time units after the start of the receiving process' time-slot influence the channel's capacity, and, hence, the analysis. In consequence, if the value of t is too large the brute-force approach used in Examples 7 and 8 is no longer feasible. Luckily, there are more efficient algorithms from the domain of global optimization [12] that can be used to compute the maximal value of $\mathrm{UB}(d)$ for $d \in D$. More precisely, we showed that the problem to find the maximal value of $\mathrm{UB}(d)$ for $d \in D$ in the above scenarios is a convex maximization problem over a linearly constrained set [13]. For this class of problems algorithms that are more efficient than the brute-force approach employed in Examples 7 and 8 have been developed [14].

6 Related Work

Covert channels have been firstly identified in [2]. They have been researched extensively throughout the last 35 years, where the main research areas were covert channel identification, covert channel analysis, and covert channel mitigation.

The focus of the current article is on covert channel modeling and analysis, not on their identification and their mitigation. A guide to covert channel identification is provided in [3], covering various approaches including syntactic information flow analysis (e.g., [15, 16]) and the shared resource matrix method [17, 18]. More recent approaches use security type systems for the identification of information leaks (e.g., [19, 20]). Various methods for the mitigation of covert timing channels have been proposed [21–24], including mechanisms targeted specifically at interrupt-related channels [1].

Concerning the analysis of covert channels, the focus of most previous research has been on deriving the bandwidth based on information theory. In [25, 26] the connection between notions of noninterference and the capacity of a channel is investigated. Various other articles (e.g., [27–31]) investigate how the capacity of certain abstract classes of channels can be computed. In [28, 32] effects of noise on the transmission time of a symbol over a covert channel are studied. This differs from our treatment of noise in Section 3.2, where noise affects the output symbol itself. Gray focuses on a particular class of covert channels in [33, 22], where the probabilistic behavior considered in the analysis originates from two mechanisms for mitigating the channel. This is also the case in the analysis of the NRL pump, another mitigation mechanism that targets covert channels exploiting acknowledgments in conventional communication [23, 34, 24].

In [35], games between an attacker exploiting a covert channel and a jammer disturbing the communication are investigated. Different jamming strategies result in a family of channel models. In this setting, the jammer completely determines the probability distributions. This differs from our treatment in Section 5, where we also consider families of probability distributions but assume that each distribution is possible given the available knowledge. In [36] and [37], models of the NRL pump that are parametric in the probability distributions are used to assess how parameter variations influence the capacity of covert channels. However, these models are not used to find upper bounds on the channel capacity when the concrete parameter values are unknown. We are not aware of any prior work that derives upper bounds on the bandwidth in a probabilistic model, where the information about the probability distributions is incomplete.

Most approaches, as also the current article, use information theory for the analysis of covert channels. However, there are also a few other approaches. For instance, [38] uses Markov models to compute the bandwidth of a covert channel. An approach that does not consider a probabilistic setting is based on counting the number of different sender behaviors that can be distinguished by the receiver [5].

7 Conclusion

In this article, we showed how additional knowledge about particular instances of covert channels can be exploited to improve the precision of the bandwidth analysis. We demonstrated with two concrete examples that such improvements can have a rather significant effect on the calculated capacity. In addition, we showed that and how even

incomplete knowledge about probability distributions can be exploited in the bandwidth analysis in a sound way.

The employed information-theoretic model is similar to the model used in [1]. Our new model constitutes a technical improvement over our earlier model, and allows to compute reliable upper bounds on the bandwidth of interrupt-related channels.

In parallel to our theoretical investigations, we experimented with practical exploitations of interrupt-related covert channels and NIC-channels in particular. This effort is on-going, but the results obtained so far already strongly confirm that interrupt-related covert channels pose a threat that should be taken serious. Using the exploit, the transmission of a 13-bit PIN (i.e. the order of magnitude used in authentication mechanisms for banking machines) takes approximately 30 seconds. In contrast to our theoretical analysis which results in upper bounds on the bandwidth, the practical evaluation allows us to obtain lower bounds on the bandwidth of interrupt-related covert channels.

References

1. Mantel, H., Sudbrock, H.: Comparing Countermeasures against Interrupt-Related Covert Channels in an Information-Theoretic Framework. In: Proc. of the IEEE Computer Security Foundations Symposium. (2007) 326–340
2. Lampson, B.W.: A Note on the Confinement Problem. Communications of the ACM **16**(10) (1973) 613–615
3. Gligor, V.: A Guide to Understanding Covert Channel Analysis of Trusted Systems. CSC-TG-030, Rainbow Series (Light Pink Book) (1993)
4. Shieh, S.P.: Estimating and Measuring Covert Channel Bandwidth in Multilevel Secure Operating Systems. Journal of Inform. Science & Engineering **15** (1999) 91–106
5. Lowe, G.: Quantifying Information Flow. In: Proc. of the IEEE Computer Security Foundations Workshop. (2002) 18–31
6. Beauquier, D., Lanotte, R.: Hiding Information in Multi Level Security Systems. In: Proc. of the Workshop on Formal Aspects in Security and Trust, LNCS 4691. (2006) 250–269
7. Son, J., Alves-Foss, J.: Covert Timing Channel Analysis of Rate Monotonic Real-Time Scheduling Algorithm in MLS Systems. In: Proc. of the IEEE Information Assurance Workshop. (2006) 361–368
8. Cover, T.M., Thomas, J.A.: Elements of Information Theory. second edn. John Wiley & Sons, Inc. (2006)
9. Arimoto, S.: An Algorithm for Computing the Capacity of Arbitrary Discrete Memoryless Channels. IEEE Trans. on Information Theory **18**(1) (1972) 14–20
10. Blahut, R.: Computation of Channel Capacity and Rate-Distortion Functions. IEEE Trans. on Information Theory **18**(4) (1972) 460–473
11. Intel Corporation: Interrupt Moderation Using Intel Gigabit Ethernet Controllers, Application Note (AP-450), Revision 1.1 (2003)
12. Horst, R., Tuy, H.: Global Optimization. Deterministic Approaches. Springer (1996)
13. Horst, R.: On the Global Minimization of Concave Functions. OR Spectrum **6**(4) (1984) 195–205
14. Benson, H.P.: Deterministic Algorithms for Constrained Concave Minimization: A Unified Critical Survey. Naval Research Logistics **43**(6) (1996) 765–795
15. Denning, D.E.: A Lattice Model of Secure Information Flow. Communications of the ACM **19**(5) (1976) 236–243

16. Denning, D.E., Denning, P.J.: Certification of Programs for Secure Information Flow. Communications of the ACM **20**(7) (1977) 504–513
17. Kemmerer, R.A.: Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels. ACM Trans. on Comp. Sys. **1**(3) (1983) 256–277
18. Kemmerer, R.A.: A Practical Approach to Identifying Storage and Timing Channels: Twenty Years Later. In: Proc. of the Annual Computer Security Applications Conference. (2002) 109–118
19. Volpano, D., Smith, G., Irvine, C.: A Sound Type System for Secure Flow Analysis. Journal of Computer Security **4(3)** (1996) 1–21
20. Sabelfeld, A., Myers, A.C.: Language-based Information-Flow Security. IEEE Journal on Selected Areas in Communication **21**(1) (2003) 5–19
21. Hu, W.M.: Reducing Timing Channels with Fuzzy Time. In: Proc. of the IEEE Symposium on Research in Security and Privacy. (1991) 8–20
22. Gray III, J.W.: On Introducing Noise into the Bus-Contention Channel. In: Proc. of the IEEE Symposium on Research in Security and Privacy. (1993) 90–98
23. Kang, M.H., Moskowitz, I.S.: A Pump for Rapid, Reliable, Secure Communication. In: Proc. of the ACM Conference on Computer and Communications Security. (1993) 119–129
24. Kang, M.H., Moskowitz, I.S., Chincheck, S.: The Pump: A Decade of Covert Fun. In: Proc. of the Annual Computer Security Applications Conference. (2005) 352–360
25. Millen, J.K.: Covert Channel Capacity. In: Proc. of the IEEE Symposium on Security and Privacy. (1987) 60–66
26. Moskowitz, I.S.: Quotient States and Probabilistic Channels. In: Proc. of the IEEE Computer Security Foundations Workshop. (1990) 74–83
27. Millen, J.K.: Finite-State Noiseless Covert Channels. In: Proc. of the IEEE Computer Security Foundations Workshop. (1989) 81–86
28. Moskowitz, I.S.: Variable Noise Effects Upon a Simple Timing Channel. In: Proc. of the IEEE Symposium on Security and Privacy. (1991) 362–372
29. Moskowitz, I.S., Miller, A.R.: Simple Timing Channels. In: Proc. of the IEEE Symposium on Research in Security and Privacy. (1994) 56–64
30. Moskowitz, I.S., Greenwald, S.J., Kang, M.H.: An Analysis of the Timed Z-channel. In: Proc. of the IEEE Symposium on Security and Privacy. (1996) 2–11
31. Martin, K., Moskowitz, I.S.: Noisy Timing Channels with Binary Inputs and Outputs. In: Proc. of the Workshop on Information Hiding, LNCS 4437. (2006) 124–144
32. Moskowitz, I.S., Miller, A.R.: The Channel Capacity of a Certain Noisy Timing Channel. IEEE Trans. on Information Theory **38**(4) (1992) 1339–1344
33. Gray III, J.W.: On Analyzing the Bus-Contention Channel under Fuzzy Time. In: Proc. of the IEEE Computer Security Foundations Workshop. (1993) 3–9
34. Kang, M.H., Moskowitz, I.S., Lee, D.C.: A Network Pump. IEEE Trans. on Software Engineering **22**(5) (1996) 329–338
35. Giles, J., Hajek, B.: An Information-theoretic and Game-theoretic Study of Timing Channels. IEEE Trans. on Information Theory **48**(9) (2002) 2455–2477
36. Lanotte, R., Maggiolo-Schettini, A., Tini, S., Troina, A., Tronci, E.: Automatic Analysis of the NRL Pump. Electr. Notes Theor. Comput. Sci. **99** (2004) 245–266
37. Aldini, A., Bernardo, M.: An Integrated View of Security Analysis and Performance Evaluation: Trading QoS with Covert Channel Bandwidth. In: Proc. of the Conference on Computer Safety, Reliability, and Security, LNCS 3219. (2004) 283–296
38. Tsai, C.R., Gligor, V.D.: A Bandwidth Computation Model for Covert Storage Channels and its Applications. In: Proc. of the IEEE Symposium on Security and Privacy. (1988) 108–121