Proceedings of the 2006 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY

Covert Timing Channel Analysis of Rate Monotonic Real-Time Scheduling Algorithm in MLS systems

Joon Son, Jim Alves-Foss

Abstract— **The modern digital battlesphere requires the development and deployment of multi-level secure computing systems and networks. A portion of these systems will necessarily be operating under real-time processing constraints. High assurance systems processing national security information must be analyzed for possible information leakages, including covert channels. In this paper we provide a mathematical framework for examining the impact the rate-monotonic real-time scheduling algorithm has on covert timing channels. We prove that in some system configurations, it will not be possible to completely close the covert channel due to the rate-monotonic timing constraints. In addition, we propose a simple method to formulate a security metric to compare covert channels in terms of the relative amount of possible information leakage.**

I. Introduction

A covert timing channel is an illicit communication path in which one entity ($High$) signals information to another entity (Low) in violation of the security policy by modulating its use of system resources in such a way that this manipulation affects the response time observed by Low. In this paper, we classify a covert (timing) channel into four different categories: non-deducible, positive-deducible, negative-deducible, and partially-deducible based upon the degree of deducibility and types of information deducible by Low. This categorization uses the concepts of nondeducibility [16] by Sutherland and nondeducibility on strategies [19] by Wittbold. In their definition of nondeducibility, there is no information flow if and only if Low cannot deduce with certainty anything about the activities of $High$.

A real-time operating system employs a scheduling algorithm to schedule multiple tasks so that each task can meet its real-time constraints such as deadline. Most real-time scheduling algorithms are priority based. Each task is assigned a priority based on its importance. In priority-based scheduling, control of the CPU is always given to the highest priority task ready to run. How a scheduling priority is assigned to a task and when the highest priority task gets the CPU, however, are determined by the type of scheduling algorithms used. Most commercial real-time scheduling algorithms are preemptive: when a higher priority task is ready to run, the current task is preempted, and the higher priority task is immediately given control of the CPU. It is not hard to conjecture that scheduling priorities and preemptiveness of a task may cause a covert timing channel and that noise introduced by a third party task may have an influence on the channel. One of our research goals is to formally analyze how a covert timing channel is created and exploited by a $High$ task (τ_H) and a Low task (τ_L) while a third party task is simultaneously running with them under a well-known real-time scheduling algorithm.

J. Son, J. Alves-Foss: Center for Secure and Dependable Systems, University of Idaho, Moscow, ID 83844
Email: son2320@uidaho.edu, jimaf@cs.uidaho.edu

This material is based on research sponsored by AFRL and DARPA under agreement number F30602-02-1-0178. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expresses or implied, of AFLR and DARPA or the U.S. Government.

The Rate Monotonic (RM) scheduling algorithm [2], [11] is one of the most widely used scheduling strategies due to its rich theoretical background and simplicity of implementation. However, the major research effort has focused on analysis of schedulability of real-time tasks running in various conditions. This motivate us to formally identify and analyze a covert timing channel present in a Multi-Level Secure (MLS) system which uses the RM scheduling algorithm to execute multiple real-time tasks. In RM scheduling which is preemptive, scheduling priorities are inversely proportional to task arrival rates. We show that a timing channel present in an RM based MLS system belongs to one of our four classifications and how this covert timing channel can be exploited due to a generic vulnerability of RM scheduling. In addition, we prove that if an RM based real-time system is configured in a certain way, no matter how noisy a timing channel is, there exists a covert timing channel between $High$ and Low.

One important characteristic of a real-time operating system is that it performs operations at fixed, predetermined times (called preemption points) or within predetermined intervals [15]. Under the RM scheduling algorithm, when a preemption point occurs, a currently running task is preempted if a higher-priority task is waiting. For the purpose of this paper we define a unit of time as the interval between preemption points, giving us a well-defined discrete time domain in which to analyze covert timing channels.

II. Related Work

Analysis of a covert timing channel in multi-level systems has been an active research area for many years. To prevent a covert timing attack, researchers had devised numerous defensive measures [7], [8], [10], [14], [17]. However, some of these defensive measures for timing channels may be unacceptable for applica-

Defensive measure	Mechanism	Real-time requirements
Fuzzy time	A security kernel produces random clock ticks to distort an accuracy of a clock.	A real-time task must have access to an accurate time source.
Adding delays (noise)	Add random service requests to cause unwanted delays in Low's response time.	Adding random service requests may cause a real-time task to fail to meet its real-time constraint such as deadline.
Fixing quanta	Tasks are scheduled in a round-robin fashion.	Tasks are scheduled according to a real-time scheduling algorithm to meet their real-time constraints.
Clock resolution	Increase the clock resolution that is available to untrusted tasks.	A real-time task must have access to an acuurate time source.

TABLE I

tions running on real-time systems (see Table I).

The first priority of a hard real-time system[1] is that a scheduler must make sure that every task running in the system meets its real-time constraints, e.g., deadline, computation time, etc. If every task meets its real-time constraints, a set of such tasks is called *schedulable*. This schedulability requirement cannot be compromised by security requirements if the system is to be usable. Devising a real-time scheduling algorithm which satisfies both real-time constraints and security requirements is not an easy task.

Research in covert channels consists of several areas: covert channel identification and characterization, covert channel capacity estimation, and covert channel prevention (defensive measures) to close a channel or reduce the bandwidth of a channel. In this paper, we identify and characterize a covert timing channel present between two real-time tasks $High$ and Low which are schedulable under RM scheduling. We also propose a simple security metric to compare covert channels in terms of the relative amount of possible information leakage.

III. Types of Covert Channels

Since Goguen and Meseguer [6] formalized information flow based on the concept of noninterference, numerous variants of noninterference have been proposed [4], [5], [6], [13], [16], [19]. A central idea of non-interference in a MLS system is that in order to show that there is no information flow from $High$ to Low, one must demonstrate that Low's view of a system is independent of the behavior of $High$, or that Low should not be able to deduce with certainty anything about the activities of $High$. Even though the variants have many different names, e.g. nondeducibility, nondeducibility on strategy, generalized noninterference, and restrictiveness, they all share the same underlying concept of noninterference. The mathematical treatment of these works is often rigorous and it is sometimes hard to see the similarities and differences of these definitions [12].

[1]In a hard real-time system, an operation performed after the deadline usually has no value.

Since our approach is based upon a simple communication channel model commonly used in information theory [3], [18], a communication path between $High$ and Low can be easily modeled as a transition diagram shown in Figure 1. One of our goals is to transform a model which describes timing behaviors of real-time tasks running under RM scheduling to a simple communication channel model, so that one can easily identify and characterize how the channel can be exploited by malicious subjects for covert communication.

A. Channel Characteristics

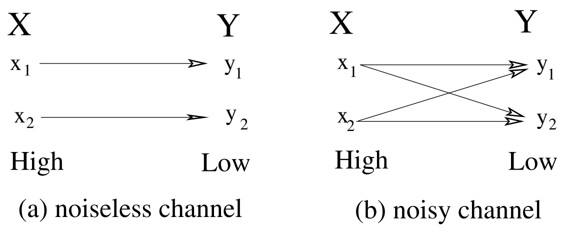

Fig. 1. Covert Channel

A covert communication path can be modeled as a discrete memoryless channel which describes a medium through which symbols flow from a sender ($High$) to a receiver (Low) as shown in Figure 1. The channel is discrete when the input alphabets $X = \{x_1, \ldots, x_J\}$ and the output alphabets $Y = \{y_1, \ldots, y_K\}$ are finite. It is memoryless when the current output depends on only the current input. When noise occurs during the transmission of a symbol, the symbol may be altered and affect the output observed by the receiver. The behavior of a noisy

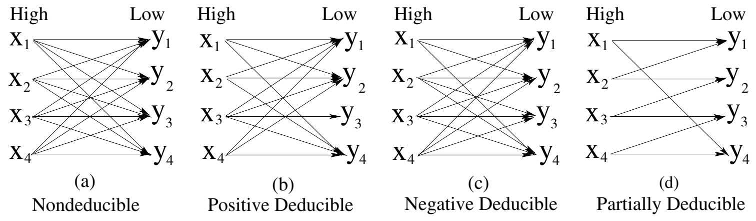

Fig. 2. Types of Covert (Signaling) Channels

channel may be nondeterministic in a sense that the output observed by the receiver is no longer a function of the input symbol transmitted. For example, consider the noisy channel shown in Figure 1(b). Upon receiving y_1, the receiver cannot reliably deduce which value (x_1 or x_2) was the input symbol transmitted by the sender. In this paper, we assume that a covert channel is noisy, which is typical in real world applications.

A covert channel can be characterized as a binary relation between an input symbol and a corresponding output symbol(s). We define $channel$ as a set which has such a binary relation.

Definition 1: Let X be a set of all possible symbols available for a sender to transmit and Y be a set of all possible symbols a receiver receives through a noisy channel.

$$channel = \{(x, y) \mid \ x \in X, \ y \in Y\} \qquad \text{(III.1)}$$

Note that there may exist multiple output symbols which correspond to a single input symbol when a channel is noisy. For instance, the noisy channel shown in Figure 1(b) can be characterized as follows:

$$channel = \{(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)\}$$

B. Classification of Covert Channels

By extending the concepts of positive and negative channels introduced by McCullough [13] and nondeducibility (deductive inference) by Sutherland and Wittbold [16], [19], we classify channels into four different categories: nondeducible, partially deducible, positive-deducible, and negative-deducible. This classification is based upon the degree of deducibility and types of information deducible by Low. Figure 2 shows different types of signaling channels which a malicious subject can attempt to transform into covert channels.

Definition 2: Let $x \in X$ and $y \in Y$. Let $Domain(y)$ be a set of input symbol(s) which may cause a output symbol y.

$$Domain(y) = \{ \ x \ \mid \ (x, y) \in channel\}$$

If $\mid X \mid > 1$ (the number of symbols available for $High$ to transmit is greater than 1), a channel can be characterized as follows:

Definition 3: A noisy channel is *nondeducible* if and only if $\forall \ y \in Y \bullet Domain(y) = X$.

When a channel is characterized as nondeducible (Figure 2(a)), upon receiving any symbol, Low cannot reliably deduce which symbol has been sent by $High$. "Reliably deducible" means that Low does not have to make any possibilistic guess or probabilistic inference about which symbol has been sent or not sent, given a symbol received.

Definition 4: A noisy channel is *positive-deducible* if and only if $\exists y \in Y \bullet \mid Domain(y) \mid = 1$.

If a noisy channel is positive-deducible, Low can tell exactly which symbol has been sent by $High$ by observing some symbols. For instance, as shown in Figure 2(b), upon receiving y_3, Low can reliably deduce that x_3 has been transmitted by $High$.

Definition 5: A noisy channel is *negative-deducible* if and only if $\exists y \in Y \bullet \mid Domain(y) \mid = \mid X \mid -1$

In a negative-deducible channel, by receiving a particular symbol, Low can determine that exactly which symbol has not been transmitted by $High$. For example, in the negative channel shown in Figure 2(c), upon receiving y_3, Low can reliably deduce that $High$ has not transmitted x_1. Please note that a noisy channel can be classified as both positive-and negative-deducible.

Definition 6: A noisy channel is *partially-deducible* if and only if it is neither nondeducible, positive-deducible, nor negative-deducible.

If a noisy channel is partially-deducible (Figure 2(d)), Low can deduce the occurrence of a set of multiple input symbols or the nonoccurrence of a set of multiple input symbols. For instance, upon receiving y_2, Low can reliably deduce that either x_2 or x_3 has occurred. At the same time, Low can also reliably deduce that neither x_1 nor x_4 has been transmitted.

Let us assume for a moment that $High$ is a Trojan horse and Low is an adversary. The intention of the Trojan horse is to send a message to the adversary if it is able to attain classified information and wants to signal the adversary while hiding in a computer system. If a channel between the Trojan horse and the adversary is nondeducible, the adversary cannot reliably deduce the intention of the Trojan horse. If the channel is positive-deducible, the adversary can easily deduce the intention of the Trojan horse. A possible strategy of the Trojan horse and the adversary could be the following. The normal mode of operation for the Trojan horse is to transmit either x_1, x_2 or x_4. When the Trojan horse is able to access some classified data and needs to signal the adversary, it immediately changes its mode of operation and continues sending x_3. Meanwhile, the adversary simply waits until it observes y_3 while ignoring other symbols. Upon observing y_3, it collects the classified information.

The negative-deducible channel could be as insecure as the positive-deducible one. Let us assume that a channel is classified as negative-deducible. A possible strategy between the Trojan horse and the adversary is described as follows. The normal mode of operation for the Trojan horse could be to transmit x_1. During this time, the adversary *never* observes y_3. Immediately after the Trojan horse attains classified information and wants to signal the adversary, it starts to transmit either x_2, x_3 or x_4. Meanwhile, the adversary waits until it detects y_3, ignoring other symbols. Upon receiving y_3, the adversary begins to collect the classified information.

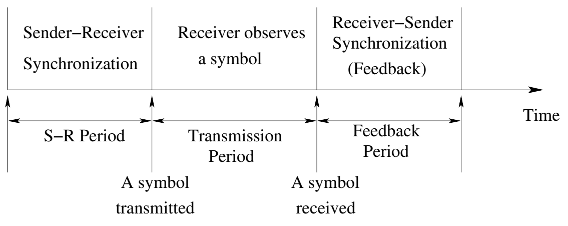
Sender–Receiver Synchronization
Receiver observes a symbol
Receiver–Sender Synchronization (Feedback)
Time
S–R Period
Transmission Period
Feedback Period
A symbol transmitted
A symbol received

Fig. 3. Transmission cycle of a covert channel

However, in both positive and negative channels, it is important to note that the Trojan horse and the adversary cannot use the exact count of y_3 as a means of reliable communication if no feedback path exists from the adversary to the Trojan horse. Without feedback, the Trojan horse (transmitter) is not able to know if the adversary (receiver) has observed y_3. Thus, the Trojan horse is not able to know when to start the transmission of its next symbol. However, if a system allows information flow from the adversary (Low) to the Trojan horse ($High$), the rate at which the Trojan horse can transmit information depends on how fast a noisy channel can deliver y_3 to the adversary and return the acknowledgement back to the Trojan horse (feedback rate). As illustrated in Figure 3, a covert channel has a transmission cycle which consists of sender-receiver synchronization (S-R) period, transmission period and feedback period. During the S-R period, a sender needs to notify a receiver that it is ready to transmit a new symbol. Note that no S-R period may be needed if a sender and a receiver have some previous agreement, e.g. every t units of time, a new symbol is transmitted. The feedback period must exist in order for reliable communication to continuously flow.

For the partially deducible channel case, the strategy of sending information is similar. During the normal mode of operation, the Trojan horse transmits x_3 and x_4. During this time, y_1 is never observed by the adversary. After sensing classified information, the Trojan horse sends x_1. The adversary waits until it observes y_1.

It is not easy to formulate a correct metric to measure the security of each deducible channel since the degree of deducibility depends on various factors[2]: the noise level of the environment on which the Trojan horse and the adversary are running, the longevity of a channel (short- or long-term channel), the existence of a feedback path, the intention of the Trojan horse (transmission of short message or long message), etc.

[2]Note that our analysis is based upon the possibilistic approach, not probabilistic one. If the transition probability of each transition is known, channel capacity can be calculated using Shannon's information theory [3], [18] and used as a security measure.

C_i:	Worst case computation (execution) time that may be required by an invocation of task τ_i. It is also denoted as C_i^{max}.
T_i:	Lower bound between successive arrivals of a task τ_i. This is the period of a periodic task τ_i.
D_i:	The deadline for each invocation of task τ_i, measured from its arrival time. Usually $D_i \leq T_i$.
I_i:	Worst case interference an invocation of task τ_i may experience due to preemptions by higher-priority tasks.
J_i:	Worst case release jitter for an invocation of task τ_i due to scheduling delay.
R_i:	Worst case response time for an invocation of task τ_i, measured from its arrival time to its termination time. A schedulable task must have $R_i \leq D_i$.
r_i:	Worst case response time for an invocation of task τ_i, measured from its release time to its termination time.

Fig. 4. Mathematical notations used to characterize a task

IV. Rate Monotonic Scheduling Algorithm

A. Background, Notations, and Assumptions

Liu and Layland's Rate Monotonic (RM) scheduling algorithm [11] has become one of the most widely used scheduling algorithms in real time systems. It has the following rules:

- Tasks with shorter periods (higher request rates) will have higher priorities.
- A priority assigned to a task is fixed; priorities do not change over time.
- It is intrinsically preemptive: a currently executing task is preempted by a newly arrived task with a higher priority (shorter period).

The mathematical symbols used are defined in Figure 4. In order to simplify our analysis the following assumptions are made on a periodic task:

A1. The periodic tasks running on a single processor are *independent* (no shared resource among tasks other than a processor).
A2. The time between any successive arrivals of a task τ_i is fixed as T_i (A task τ_i is activated at a constant rate T_i).
A3. The deadline for each invocation of task τ_i is equal to the period (D_i=T_i)
A4. There is no release time jitter ($J_i = 0$). A task τ_i is released as soon as it arrives; thus, R_i = r_i.
A5. The initial release time of all the tasks is zero.

From the above assumptions, a periodic task τ_i can be completely characterized as $\tau_i(T_i, C_i)$. Thus, we can denote a set of periodic tasks running under RM scheduling algorithm as:

$$\Gamma_{RM} = \{\tau_i(T_i, C_i),\;\; i = 1 \ldots n\}$$

B. Worst Case Response Time Analysis

Let $\Gamma_{RM} = \{\tau_i(T_i, C_i),\;\; i = 1 \ldots n\}$. Let $\pi(\tau_i)$ represent a scheduling priority assigned to a task τ_i. Assume that $\pi(\tau_1) > \pi(\tau_2) \cdots > \pi(\tau_n)$. Joseph and Pandya [9] showed that a task set Γ will meet all its deadlines if:

$$\begin{aligned} &for\ all\ tasks\ \tau_i \in \Gamma \quad R_i \le D_i, \\ &\qquad where\ R_i = C_i + I_i \qquad \text{(IV.1)} \\ &\qquad and\ \ I_i = \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \end{aligned}$$

The definition of R_i (Eq IV.1) is recursive. If a set of tasks are not schedulable, one cannot find a solution for R_i of the lowest priority task, which satisfies $R_i \le D_i$. Please refer to [1] to see how the recurrent equation (Eq IV.1) with a ceiling function is solved.

V. Covert Timing Channel Analysis

In this section, we introduce a model which describes a covert timing channel between a high-level task τ_H and a low-level task τ_L, while a third party task τ_N (noise) is running concurrently with them under RM scheduling. We then transform this model into a simple communication channel model described as a set with input-output binary relations (Definition 1). We make the following assumptions in our model.

- **Periodic tasks:** Three tasks are running under RM scheduling, i.e. $\Gamma_{RM} = \{\tau_i(T_i, C_i),\;\; i = N, H, L\}$
- **Schedulability:** Γ_{RM} is schedulable.
- **Scheduling priority:** We assume $T_N < T_H < T_L$, i.e. $\pi(\tau_N) > \pi(\tau_H) > \pi(\tau_L)$. The outcome of analysis will be very similar when $T_H < T_N < T_L$, i.e. $\pi(\tau_H) > \pi(\tau_N) > \pi(\tau_L)$. What is important here is that a low level task τ_L is assumed to be a task with the longest period. There will be no covert channel if a task τ_L has the shortest period; however, it may not be a viable solution for some real-time application where a high level task needs to consume more CPU time.
- **Ability of *High***: $High$ is given as much opportunity as possible for creating a covert timing channel. At every release time, the computation of a task τ_H may vary from one unit to C_H^{max} units of time.
- **Ability of *Low***: Low cannot measure the time between any two occurrences of context switch. However, for each period, Low is able to assess the response time of its own task (the time between the submission of its task to a scheduler and the notification that it is completed). We assume there is no overhead associated with the task submission and the notification.
- **Timing behavior of τ_N:** At every release time, the computation time of a task τ_N may vary from one unit to C_N^{max} units of time. However, its timing behavior is nondeterministic in a sense that neither Low nor $High$ can reliably predict the computation time performed at each release.
- **Periods of τ_N, τ_H, and τ_L:** To simplify our analysis, we assume that T_N divides T_L and T_H divides T_L. With this assumption, Low can obtain a single output sample (the response time of its own task) by monitoring at most T_L units of time. In addition, the number of periods of τ_N and τ_H which affect the response time of Low can be well defined. This assumption is not necessary, but without the assumption, Low has to wait for the maximum of $l.c.m.$ (T_N, T_H, T_L) [3] units of time in each period to sample the output.
- **Sampling factor of *Low***: In order to detect (sample) a response time of it own task, Low submits a task with known computation time to a real-time scheduler. This fixed computation time is called the sampling factor of τ_L. We use $\hat{C}_L$ to denote the sampling factor.
- **S-R period:** We assume the S-R period is zero, e.g. $High$ begins to transmit a new symbol every t units of time, $t \ge$ (worst case transmission period) + (worst case feedback period). This provides us with an upper bound on the transmission rate of a channel.

A. Extending Response Time Analysis

We extend Eq (IV.1) to calculate all possible response times (not just the worst response time) of a task τ_L.

Let $\hat{C}_H[k]$ denote the computation time of a task τ_H at the k^{th} release (during the k^{th} period), $1 \le \hat{C}_H[k] \le C_H^{max}$. We call $\hat{C}_H[k]$ as a *timed action* of a task τ_H at the k^{th} release. Let $\hat{C}_H$ be a vector (tuple) which represents a sequence of timed actions of a task τ_H from the first up to $\left\lceil \frac{T_L}{T_H} \right\rceil^{th}$ release:

$$\hat{C}_H = (\hat{C}_H[1], \hat{C}_H[2], \ldots, \hat{C}_H[k], \ldots, \hat{C}_H[\left\lceil \frac{T_L}{T_H} \right\rceil])$$

Similarly, assuming $1 \le \hat{C}_N[k] \le C_N^{max}$, we denote a sequence of timed actions of a task τ_N from the first up to $\left\lceil \frac{T_L}{T_N} \right\rceil^{th}$ release as:

$$\hat{C}_N = (\hat{C}_N[1], \hat{C}_N[2], \ldots, \hat{C}_N[l], \ldots, \hat{C}_N[\left\lceil \frac{T_L}{T_N} \right\rceil])$$

The response time of a task τ_L is computed by adding up the sampling factor $\hat{C}_L$, interference time caused by τ_H and interference time caused by τ_N:

$$\begin{aligned} \hat{R}_L =&\ \hat{C}_L + Interference_{\tau_H} + Interference_{\tau_N} \qquad \text{(V.1)} \\ =&\ \hat{C}_L + \sum_{k=1}^{\left\lceil \frac{\hat{R}_L}{T_H} \right\rceil} \hat{C}_H[k] + \sum_{l=1}^{\left\lceil \frac{\hat{R}_L}{T_N} \right\rceil} \hat{C}_N[l] \qquad \text{(V.2)} \end{aligned}$$

[3] $l.c.m.(T_N, T_H, T_L)$ represents the least common multiple of the periods of tasks specified in the argument. It is often called the hyper-period.

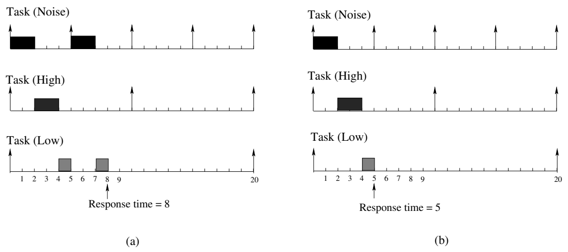

Fig. 5. Timing Diagram

We solve (Eq V.2) for the response time $\hat{R}_L$ using the recurrent relationship:

$$w_L^{n+1} = \hat{C}_L + \sum_{k=1}^{\left\lceil \frac{w_L^n}{T_H} \right\rceil} \hat{C}_H[k] + \sum_{l=1}^{\left\lceil \frac{w_L^n}{T_N} \right\rceil} \hat{C}_N[l] \qquad \text{(V.3)}$$

However, the recurrent equation has two free variables $\hat{C}_H$ and $\hat{C}_N$. The value of $\hat{R}_L$ depends on the specific variable assignments ν_h and ν_n from sets of positive integers to the variables. We denote these variable assignments as follows:

$$\nu_h(\hat{C}_H) = (h_1, h_2, \ldots, h_i, \ldots, h_{\left\lceil \frac{T_L}{T_H} \right\rceil}),\ h_i \in \{1, \ldots, C_H^{max}\}$$

After the variable assignment ν_h, $\hat{C}_H[1] = h_1$, $\hat{C}_H[2] = h_2$, ..., and $\hat{C}_H[\left\lceil \frac{T_L}{T_H} \right\rceil] = h_{\left\lceil \frac{T_L}{T_H} \right\rceil}$. Likewise,

$$\nu_n(\hat{C}_N) = (n_1, n_2, \ldots, n_i, \ldots, n_{\left\lceil \frac{T_L}{T_N} \right\rceil}),\ n_i \in \{1, \ldots, C_N^{max}\}$$

After the variable assignment ν_n, $\hat{C}_N[1] = n_1$, $\hat{C}_N[2] = n_2$, ..., and $\hat{C}_N[\left\lceil \frac{T_L}{T_N} \right\rceil] = n_{\left\lceil \frac{T_L}{T_N} \right\rceil}$.

If we provide $\nu_h(\hat{C}_H)$, $\nu_n(\hat{C}_N)$, and $\hat{C}_L$ to the recurrent equation (V.3), we can calculate the response time $\hat{R}_L$ [1]. Once the response time $\hat{R}_L$ is computed, we can have a more accurate trace or history of the actual execution sequences of tasks τ_H and the τ_N which cause $\hat{R}_L$. Let $\hat{\nu_h}(\nu_h, \hat{R}_L)$ be a sequence of timed actions of a task τ_H which is responsible for the delay of a task τ_L (the first $\left\lceil \frac{\hat{R}_L}{T_H} \right\rceil$ elements of $\nu_h(\hat{C}_H)$):

$$\hat{\nu_h}(\nu_h, \hat{R}_L) = (h_1, h_2, \ldots, h_{\left\lceil \frac{\hat{R}_L}{T_H} \right\rceil})$$

Similarly, let $\hat{\nu_n}(\nu_n, \hat{R}_L)$ be a sequence of timed actions of a task τ_N which is responsible for the delay of a task τ_L (the first $\left\lceil \frac{\hat{R}_L}{T_N} \right\rceil$ elements of $\nu_n(\hat{C}_N)$):

$$\hat{\nu_n}(\nu_n, \hat{R}_L) = (n_1, n_2, \ldots, n_{\left\lceil \frac{\hat{R}_L}{T_N} \right\rceil})$$

The response time $\hat{R}_L$ of a task τ_L can be represented as a function f of $\hat{\nu_h}(\nu_h, \hat{R}_L)$, $\hat{\nu_n}(\nu_n, \hat{R}_L)$, and $\hat{C}_L$. Let us denote this functional relation f by

$$\hat{R}_L = f(\hat{\nu_h}(\nu_h, \hat{R}_L), \hat{\nu_n}(\nu_n, \hat{R}_L), \hat{C}_L) \qquad \text{(V.4)}$$

Given a fixed variable assignment $\hat{\nu_h}(\nu_h, \hat{R}_L)$ and a constant value $\hat{C}_L$, multiple values of $\hat{R}_L$ may be generated by the recurrent equation, depending on various variable assignments $\hat{\nu_n}(\nu_n, \hat{R}_L)$. If we consider the response time $\hat{R}_L$ of a task τ_L as the output symbol observed by a receiver, and a variable assignment $\hat{\nu_h}(\nu_h, \hat{R}_L)$ as the input symbol transmitted by a sender, a simple communication channel model (*channel*-definition 1) can be constructed as follows:

Let V_h and V_n be sets of all possible variable assignments $\nu_h(\hat{C}_H)$ and $\nu_n(\hat{C}_N)$, respectively. Provided that $\hat{C}_L$ is given,

$$\{(\hat{\nu_h}(\nu_h, \hat{R}_L), \hat{R}_L) \mid \hat{R}_L = f(\hat{\nu_h}(\nu_h, \hat{R}_L), \hat{\nu_n}(\nu_n, \hat{R}_L), \hat{C}_L), \nu_h(\hat{C}_H) \in V_h,\ \nu_n(\hat{C}_N) \in V_n\} \qquad \text{(V.5)}$$

By inputting all possible combinations of variable assignments $\nu_h(\hat{C}_H) \in V_h$ and $\nu_n(\hat{C}_N) \in V_n$ and $\hat{C}_L$ into the recurrent equation (V.3), we can construct a set *channel* with the input symbol being a sequence of timed actions $\hat{\nu_h}(\nu_h, \hat{R}_L)$ of a task τ_H and the output symbol being a corresponding response time $\hat{R}_L$. We specifically denote such a channel Eq (V.5) as $channel_{\Gamma_{RM}}$.

Example 1: Let $\Gamma_{RM} = \{\tau_N(5,2), \tau_H(10,2), \tau_L(20,2)\}$ and $\hat{C}_L = 2$. Assume $\nu_h(\hat{C}_H) = (2,2)$ and $\nu_n(\hat{C}_N) = (2,2,2,2)$. Then, using Eq (V.2) , $\hat{R}_L = 8$. The result can be compared with the timing diagram (also known as Gantt diagram) provided in Figure 5(a). Using Eq (V.4), this can be formally represented as $\hat{R}_L = f((2),(2,2),2) = 8$.

Example 2: All the conditions are same as in Example 1 except $\hat{C}_L$. Assume $\hat{C}_L = 1$. Then, using Eq (V.2), $\hat{R}_L = 5$. Please refer to Figure 5(b). Using Eq (V.4), this can be formally represented as $\hat{R}_L = f((2),(2),1) = 5$. Note that the small variation in the sampling factor may result in (significant) change in response time.

Example 3: Let $\Gamma_{RM} = \{\tau_N(5,2), \tau_H(10,2), \tau_L(20,2)\}$ and $\hat{C}_L = 1$. We can construct $channel_{\Gamma_{RM}}$ by providing all possible combinations of variable assignments $\nu_h \in V_h$ and $\nu_n \in V_n$ and $\hat{C}_L$ into the recurrent equation. After all the computations, $channel_{\Gamma_{RM}} = \{((1),3),((1),4),((2),4),((2),5)\}$ as shown in Figure 6(a). When *Low* observes that $\hat{R}_L = 3$ (or $\hat{R}_L = 5$), it

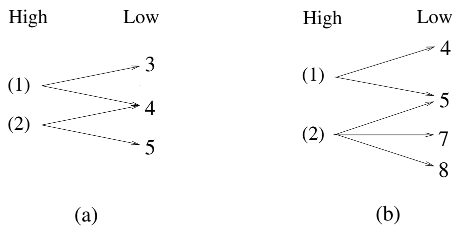

Fig. 6. *channel* $_{\Gamma_{RM}}$ for Γ_{RM}={ $\tau_N(5,2), \tau_H(10,2), \tau_L(20,2)$} (a) $\hat{C}_L$= 1 (b) $\hat{C}_L$ = 2

can reliably deduce that a sequence of timed actions of a task τ_H is (1) (or (2)). When it observes that $\hat{R}_L$ = 4, it cannot reliably deduce that which sequence of timed actions ((1) or (2)) causes $\hat{R}_L$ = 4.

Example 4: Let $\Gamma_{RM} = \{\tau_N(5,2), \tau_H(10,2), \tau_L(20,2)\}$ and $\hat{C}_L = 2$. Please note that we only modify the sampling factor, leaving other parameters unchanged, compared to the previous example. Similarly, after all the computations, $channel_{\Gamma_{RM}}$ = $\{((1),4),((1),5),((2),5),((2),7),((2),8)\}$ as shown in Figure 6(b). When Low observes that $\hat{R}_L$ = 4, 7, or 8, it can narrow down which sequence of timed actions of τ_H affects its response time. However, when Low observes that $\hat{R}_L$ = 5, it cannot narrow down which sequence of timed actions ((1) or (2)) causes $\hat{R}_L$ = 5.

B. Classification of *channel* $_{\Gamma_{RM}}$

Before we figure out which category our model $channel_{\Gamma_{RM}}$ belongs to, we summarize a few important assumptions we made to build it. The important assumptions are:

- We have a schedulable set of tasks $\Gamma_{RM} = \{\tau_i(T_i, C_i),\ i = N, H, L\}$ and $\pi(\tau_N) > \pi(\tau_H) > \pi(\tau_L)$. The schedulability implies that transmission period, i.e. $\hat{R}_L$ is less than or equal to T_L
- Low submits a job with known computation time ($\hat{C}_L$) to observe the response time of its own job. The time between any two context switches cannot be measured.
- $High$ is given as much opportunity as possible: $1 \leq \hat{C}_H[k] \leq C_H^{max}$.
- The timing behavior $\hat{C}_N$ of τ_N at every release is nondeterministic. However, its range is $1 \leq \hat{C}_N[k] \leq C_N^{max}$.

We can derive two interesting properties from our model $channel_{\Gamma_{RM}}$. If Low observes the maximum (or minimum) response time, Low can uniquely determine a sequence of timed actions of a high-level task τ_H which is responsible for the maximum (or minimum) response time. The following two lemmas formalize these two properties. Let $\hat{R}_L^{max}$ be the the maximum response time and $\hat{R}_L^{min}$ be the maximum response time.

Lemma 1: If $(\hat{\nu_h}(\nu_h, \hat{R}_L^{max}),\ \hat{R}_L^{max}) \in channel_{\Gamma_{RM}}$, the one and only possible variable assignment $\hat{\nu_h}(\nu_h, \hat{R}_L^{max})$ which causes the delay of a task τ_L is $\hat{\nu_h}(\nu_h, \hat{R}_L^{max})$ = $(C_H^{max}, C_H^{max}, \ldots, C_H^{max})$.

Proof: The proof is trivial. Since we assume $\hat{R}_L = \hat{R}_L^{max}$, Eq V.1 becomes $\hat{R}_L^{max} = \hat{C}_L + Interference_{\tau_H} + Interference_{\tau_N}$, where $Interference_{\tau_H}$ = $\sum_{k=1}^{\left\lceil \frac{\hat{R}_L^{max}}{T_H} \right\rceil} \hat{C}_H[k]$. $\hat{R}_L^{max}$ implies that both $Interference_{\tau_H}$ and $Interference_{\tau_N}$ must be maximum ($\hat{C}_L$ is constant). The maximum value of $Interference_{\tau_H}$ means that *all* the values assigned to $\hat{C}_H[k]$ must be C_H^{max}. ■

Lemma 2: If $(\hat{\nu_h}(\nu_h, \hat{R}_L^{min}),\ \hat{R}_L^{min}) \in channel_{\Gamma_{RM}}$, the one and only possible variable assignment $\hat{\nu_h}(\nu_h, \hat{R}_L^{min})$ which causes the delay of a task τ_L is $\hat{\nu_h}(\nu_h, \hat{R}_L^{min})$ = $(1, 1, \ldots, 1)$.

Proof: Since the approach of proving this lemma is very similar to that in Lemma 1, we leave the proof to the reader. ■

From the previous two lemmas, we can prove the following theorem:

Theorem 1: The noisy timing channel $channel_{\Gamma_{RM}}$ is always positive-deducible.

Proof: Lemma 1 implies that, upon observing the maximum response time $\hat{R}_L^{max}$, Low can reliably deduce that $(C_H^{max}, C_H^{max}, \ldots, C_H^{max})$ is the only possible sequence of high level actions to have occurred. Therefore, $Domain(\hat{R}_L^{max})$ = $\{(C_H^{max}, C_H^{max}, \ldots, C_H^{max})\}$ and thus, $\mid Domain(\hat{R}_L^{max}) \mid$ = 1. Likewise, Lemma 2 implies that, upon detecting the minimum response time $\hat{R}_L^{min}$, Low can reliably deduce that $(1, 1, \ldots, 1)$ is the only possible sequence of high level actions to have occurred. Therefore, $Domain(\hat{R}_L^{max}) = \{(1, 1, \ldots, 1)\}$ and thus, $\mid Domain(\hat{R}_L^{min}) \mid$ = 1. Since $channel_{\Gamma_{RM}}$ has at least two response times $\hat{R}_L^{max}$ and $\hat{R}_L^{min}$, which satisfy the definition of positive-deducible (Definition 4), $channel_{\Gamma_{RM}}$ is always positive-deducible. ■

VI. Security Metric - Relative Channel Capacity

Low can deduce more information about the intention of $High$ and, thus, more information is transferred from $High$ to Low if a channel becomes less noisy. Security researchers commonly uses Shannon's information theory [3], [18] to quantify the amount of information transferred from $High$ to Low. This *absolute* quantity, called channel capacity, is used to assess and evaluate the severity of a covert channel. Channel capacity is an effective metric to measure the degree of deducibility if the goal of $High$ is to send a long message and a noisy channel lasts enough to complete the transmission. In order to compute the channel capacity of a noisy channel, a transition probability $p(y \mid x), x \in X, y \in Y$ must be known.

It is not easy to formulate a metric to quantify information flow present in a noisy environment if a transition probability is not known (in many practical situations such statistical data are either unavailable or unreliable). However, if one is less ambitious and thus not seeking for an absolute measure such as channel capacity, an alternative way may be devised to measure the relative and approximate quantity of information flow. The relative quantity of information flow may be useful to make comparisons between the amount of information transmitted in one

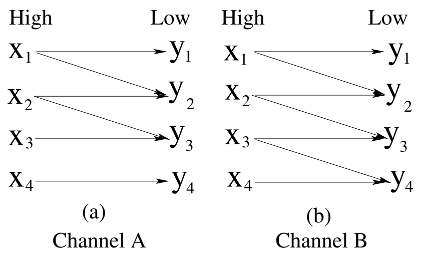

Fig. 7. The degree of deducibility

channel and that of information in another channel. We propose a simple method to formulate a security metric which can indicate the relative quantity of information flow. We call this security metric as *relative channel capacity*. The new method begins with the use of Shannon's general formula for computing the amount of information generated by the reduction of n equally likely possibilities to m. This general formula is $\log_2\left(\frac{n}{m}\right)$. We write $I_S(y)$ to denote the measure of information carried by a particular symbol y about the state of a sender S. For example, in the noisy channel A shown in Figure 7(a), upon observing y_1, a receiver can deduce that x_1 has been transmitted by a sender, e.g. the reduction of four possibilities (x_1, x_2, x_3, or x_4) to one (x_1). Thus, $I_S(y_1) = \log_2\left(\frac{4}{1}\right) = 2$, assuming that the probabilities associated with occurrences of input symbols are all equal. Upon receiving y_2, a receiver is able to deduce that either x_1 or x_2 was sent by a sender, e.g. the reduction of four possibilities to two (x_1 or x_2). Thus, $I_S(y_1) = \log_2\left(\frac{4}{2}\right) = 1$. The relative channel capacity $I_S(Y)$ of information flow in the noisy channel A is computed by adding all $I_S(y_j)$, $j \in \{1, 2, 3, 4\}$. Therefore, $I_S(Y) = I_S(y_1) + I_S(y_2) + I_S(y_3) + I_S(y_4) = 2 + 1 + 1 + 2 = 6$. If the same method is applied to the noisy channel of Figure 7(b), we get $I_S(Y) = I_S(y_1) + I_S(y_2) + I_S(y_3) + I_S(y_4) = 2 + 1 + 1 + 1 = 5$. These relative quantities can be used as effective metrics to compare the degrees of deducibility of two positive noisy channels when they are modeled in a possibilistic way. Since $I_S(Y)$ of the noisy channel A is greater than that of the channel B, we may conclude that Low can deduce more information about the intention of $High$ if both of them communicate covertly in an environment which could be modeled as the channel A.

VII. Conclusion & Discussion

Using a simple communication channel model, we classify a noisy channel into four categories based upon the degree of deducibility and the type of information deducible by Low. From each deducible channel, we show that how $High$ and Low can exploit the noisy channel for covert communication. To prove that noise introduced by a third party application as a part of an RM based system will not completely close a covert timing channel, we show that the covert timing channel $channel\ \Gamma_{RM}$ always falls into the positive-deducible category. Finally, the security metric called relative channel capacity is proposed to compare between the degree of deducibility of one covert channel and that of another channel.

Currently, we are researching how to quantify the information deducible from a specific observation when transition probabilities of a channel are known. When Low observes a specific symbol, it can positively (or negatively) deduce a set of input symbols $High$ tries to signal. If the amount of deducible information can be measured for each specific symbol received, it can be found out that which output symbol(s) carries more information about the state of $High$ than others. Such quantities may be useful in designing an effective defensive measure against a timing attack.

References

[1] N.C. Audsly, A. Burns, M.F. Richardson, K. Tindell, and A.J. Wellings. Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. 8(5):284–292, 1993.

[2] G. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, 1997.

[3] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[4] R. Focardi and R. Gorrieri. A classification of security properties. *Journal of Computer Security*, 3(1):5–33, 1994.

[5] R. Forster. *Non-Interference Properties for Nondeterministic Processes*. PhD thesis, Oxford University, 1999.

[6] J. Goguen and J. Meseguer. Security policies and security models. In *IEEE Symposium on Research in Security and Privacy*, pages 11–20, 1982.

[7] W. Hu. Reducing Timing Channels with Fuzzy Time. In *IEEE Symposium on Research in Security and Privacy*, May 1991.

[8] J. Janeri, D. Darby, and D. Schnackenberg. Building Higher Resolution Synthetic Clocks for Signaling in Covert Timing Channels. In *The Eighth IEEE Computer Security Foundations Workshop*, 1995.

[9] M. Joseph and P. Pandya. Finding Response Time in a Real-Time System. 29(5):390–395, 1986.

[10] M.H. Kang, I. Moskowitz, and S.Chincheck. The pump: A Decade of Covert Fun. In *21st Annual Computer Security Applications Conference*, 2005.

[11] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery*, 20(1), 1973.

[12] H. Mantel. Possibilistic Definitions of Security - An Assembly Kit. In *13 th IEEE Computer Security Foudnations Workshop*, pages 235–243, 2000.

[13] D. McCullough. Covert Channel and Degree of Insecurity. In *Proceedings of the Computer Security Foundations Workshop*, 1988.

[14] I. Moskowitz and M.H. Kang. Covert Channels - Here to Stay. In *Computer Assurance, COMPASS 94*, pages 235–243, 1994.

[15] W. Stallings. *Operating Systems: Internals and Design Principles*. Prentice Hall, fifth edition, July 2004.

[16] D. Sutherland. A Model of Information. In *Proceedings of the 9th National Computer Security Conference*, pages 175–183, 1986.

[17] J. Gray III. On introducing noise into the Bus-Contentiion Channel. In *IEEE Symposium on Research in Security and Privacy*, 1993.

[18] W. Weaver and C.E. Shannon. *The Mathematical Theory of Communication*. University of Illionois Press, 1963.

[19] J. Wittbold and D.M. Johnson. Information Flow in Nondeterministic Systems. In *IEEE Symposium on Security and Privacy*, pages 144–161, 1990.