Static Security Optimization for Real-Time Systems

Man Lin, *Member, IEEE*, Li Xu, *Member, IEEE*, Laurence T. Yang, *Member, IEEE*, Xiao Qin, *Member, IEEE*, Nenggan Zheng, Zhaohui Wu, *Senior Member, IEEE*, and Meikang Qiu, *Senior Member, IEEE*

***Abstract*—An increasing number of real-time applications like railway signaling control systems and medical electronics systems require high quality of security to assure confidentiality and integrity of information. Therefore, it is desirable and essential to fulfill security requirements in security-critical real-time systems. This paper addresses the issue of optimizing quality of security in real-time systems. To meet the needs of a wide variety of security requirements imposed by real-time systems, a group-based security service model is used in which the security services are partitioned into several groups depending on security types. While services within the same security group provide the identical type of security service, the services in the group can achieve different quality of security. Security services from a number of groups can be combined to deliver better quality of security. In this study, we seamlessly integrate the group-based security model with a traditional real-time scheduling algorithm, namely earliest deadline first (EDF). Moreover, we design and develop a security-aware EDF schedulability test. Given a set of real-time tasks with chosen security services, our scheduling scheme aims at optimizing the combined security value of the selected services while guaranteeing the schedulability of the real-time tasks. We study two approaches to solve the security-aware optimization problem. Experimental results show that the combined security values are substantially higher than those achieved by alternatives for real-time tasks without violating real-time constraints.**

***Index Terms*—Embedded real-time systems, optimization, security.**

I. Introduction

RECENTLY there have been some efforts devoted to the development of real-time applications with security requirements. Security requirements in many real-time applications (see, for example, [25] and [27]) must be met in addition to satisfying timing constraints of the real-time applications. Examples of security sensitive real-time applications include online transaction processing systems [18], medical electronics [30], radar for tracking missiles [17], and aircraft control [9]. Sensitive data and processing in a variety of real-time systems must be protected against unauthorized accesses. For example, a radar tracking and processing system needs to read the images from the tracking subsystem periodically and send the tracking commands and directions to the tracking subsystem. In a railway signaling system, a train needs to communicate with the control center about its position, speed and the control center sends commands to the train of which track to follow and sets the train's speed if necessary. Such real-time applications require high quality of security to guarantee the messages between the subsystems not being read or altered by malicious users, and to guarantee the user really be whom he or she claims to be.

A. Security Requirements and Services in Real-Time Systems

To address these *security requirements, security services* like message encryption and decryption, secure connection establishment, etc., can be used.

According to Kaufman *et al.* [13], security services commonly serve confidentiality, authentication and integrity.

- Confidentiality protection service is against unauthorized disclosure of information. Loss of confidentiality means disclosure of asset to unauthorized recipients.
- Authentication service is the process of reliably determining the identity of a communicating party. Loss of authentication means that the one claiming to be the sender of the message is not the true sender.
- Integrity protection service is against the unauthorized modification of data. Loss of integrity means damage to the asset through unauthorized modification.

The security risks that we need to consider for the railroad signaling system include the following.

- The information of the train status (speed and position) sent from the trains to the control center must not be altered by malicious users. Otherwise, the control center might make a wrong decision based on the wrong messages received.
- The information of the commands from the control center to the trains should not be altered either. Disaster can happen if a wrong command is received and followed by the trains.
- The trains and the control center must authenticate the messages that they receive to make sure the messages were really sent by whom claims to send them.
- The messages should not be able to be read by a third party to protect the message from being altered or to avoid terrorists changing the track to cause disaster.

Manuscript received April 30, 2008; revised September 02, 2008, December 06, 2008, and January 14, 2009. Current version published March 06, 2009. The work of M. Lin and L. T. Yang was supported by National Sciences and Engineering Research Council, Canada (NSERC) and the work of Z. Wu and N. Zheng was supported by NSFC for Distinguished Young Scholars (No. 60525202) PCSIRT Program (IRT0652). The work of X. Qin was supported in part by the U.S. National Science Foundation under Grants CCF-0742187, CNS-0757778, CNS-0831502, OCI-0753305, DUE-0621307, and DUE-0830831 and in part by Auburn University under a startup grant. A shorter version [14] of this paper, "Schedulability Driven Security Optimization in Real-Time Systems" was published in the proceedings of ARES 2006. Paper no. TII-08-04-0054.R2.

M. Lin, L. Xu, and L. T. Yang are with Department of Mathematics, Statistics, and Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada.

X. Qin is with Department of Computer Science and Software Engineering, Samuel Ginn College of Engineering, Auburn University, Auburn, AL 36849-5347 USA.

N. Zheng and Z. Wu are with the College of Computer Science, Zhejiang University, Hangzhou 310027, China.

M. Qiu is with Department of Electrical and Computer Engineering, University of New Orleans, New Orleans, LA 70148 USA.

Digital Object Identifier 10.1109/TII.2009.2014055

1551-3203/$25.00 © 2009 IEEE

Thus, the three types security services: confidentiality, authentication and integrity are therefore needed in the railway signalling system to protect the trains from following the wrong track and resulting in accidents. An increasing number of real-time applications like the railway signaling control system require high quality of security to assure confidentiality and integrity of information. Therefore, it is desirable and essential to fulfill security requirements in security-critical real-time systems.

A security service for a particular security requirement can be achieved by various security mechanisms. For example, different cryptographic algorithms can be used to guarantee the confidentiality. The notion of *security variant* was first brought forward by Irvine *et al.* [11]. The idea is that the degree of security is different for different security variants. Examples of [11] are as follows.

- Different cryptographic algorithms, e.g., RSA, RC4, RC5 or DES lead to security variants. The strength of each variant can be measured in terms of the work factor associated with a brute force attack or other metrics [35].
- Different length of cryptographic keys lead to security variants. The longer the key, the more robust the variant.
- Different authentication mechanisms lead to different variants. Example variants are weak password, strong password, biometric, and smart cards with onboard display and input interfaces.

A system may provide many security services which can be partitioned into groups. For example, a group of services can be targeting at authentication specifically and another group of services can be targeting at data integrity, etc.

To meet the needs of a wide variety of security requirements imposed by real-time systems, the *group-based security model* is used in this paper. The services in the same group provide the same type of service but of different quality due to the different mechanism used. The group-based security model is also used by Xie *et al.* [31] for developing an allocation scheme for parallel applications with deadline and security constraints on clusters. The services in the same group can be considered as security variants [11]. Security services from different groups can be combined to achieve better quality.

Applying security services to a system incurs *overhead* to the system including more CPU and memory usage. Therefore, each security service in a group can be modeled with a quality value and parameters to compute its overhead.

B. Schedulability

There are many challenges for real-time system security, such as challenges related to efficient implementations, software engineering and programming models, and schedulability. In this paper, we focus on the schedulability.

Real-time systems require not only logical correctness, but also timing correctness. A vehicle with the ability to brake is not enough, it must also be able to brake in time. The timing constraints of a real-time system are often specified as deadlines. It is desirable (required in many cases) for the system to respond within the specified deadline. If such a condition is violated, then it might cause economic losses or even physical damage. Therefore, *predictability* is a major concern in a real-time system [10], [28]. Static schedulability analysis can be used to predict whether the timing constraints of a real-time system can be met [4], [15]. One of the most commonly used schedulability tests for real-time systems is schedulability test for earliest deadline first (EDF) scheduling [15]. The test is to check whether the utilization or density of the task set in the underlying real-time system is less than 1.

To enhance security over real-time systems, the security service overhead needs to be taken into account to guarantee the schedulability of the system since applying security services incurs extra CPU computation. In this paper, the EDF schedulability test is adapted to security aware real-time systems.

C. Security Optimization With Schedulability Constraints

In the group-based security model, services from different groups can be combined to achieve better security. The problem of finding the best combination of services is a *combinatorial optimization problem* [19]. A naive method would be to exhaustively check all the possible service combinations. This method is obviously too expensive. Two approaches are proposed to select the best combination of security services for real-time systems while guaranteeing their schedulability. One is integer linear programming technique and the other is a search technique. The search technique is efficient since heuristics are developed to prune the branches and back jumping is used to reduce the search space.

The rest of the paper is organized as follows. Section II presents related works. Section III describes system model including task model and security model. Section IV presents the security aware schedulability test. Section V describes the security optimization problem and the two approaches to solve the problem. Section VI shows experiments. Section VII describes the conclusions and future works.

II. Related Works

Schedulability is the most important requirement in real-time systems. Recently, different factors are brought into real-time scheduling to guarantee the quality and efficiency of systems. Examples are energy aware scheduling [3], [5], [22] and fault-tolerance aware scheduling [36]. Techniques have been developed to optimize energy saving or reliability of real-time systems while at the same time guaranteeing the schedulability of the systems.

Quality-of-service (QoS) has received widespread attention during the last decade, especially in networking and multimedia systems. Intuitively, a system can achieve better quality if more resources are available. For example, a real-time multimedia application can obtain higher video quality when a higher frame frequency is used. Imprecise computation [16] is a technique for real-time systems where precise outputs are traded off for timely responses to system events and thus a graceful degradation is achieved. The QoS-based Resource Allocation Model (Q-RAM) developed at Carnegie Mellon University allows multiple QoS requirements such as timeliness, cryptography and reliable data delivery to be addressed and traded off against each other [23]. QoS resource tradeoff technique has been incorporated into a resource management system for an aircraft flight control application for QoS negotiation [2]. The QoS resource

tradeoff technique has also been used in energy efficient computing systems to maximize the system value while satisfying time and energy constraints [24]. Although cryptographic length is listed as a QoS dimension example in [23], security optimization in real-time systems has not been well studied.

Security of a system ensures confidentiality of information, authenticity of entity and integrity of message and data. Earlier works that considered security requirement in real-time systems addressed the quality of security in such systems. Irvine and Levin [12] introduced the notion of security variant. Similar to the security variant model, the group-based security model that we adopt also assumes that a system can have different services of different quality. The difference is that in the group-based security model, the services are partitioned into groups and the effect of combining different types of services to enhance the security of a real-time system is addressed. This group-based security model was also used in [31], [32] as level-based security model when addressing a real-time task allocation scheme on clusters.

One challenge of addressing security requirement in variant-based or group (or level)-based security aware system is how to define the quality value for security services and how to compute the overhead of applying the security services. The solution relies on the security specialists to define the quality value and cost for some certain security service. Irvine and Levin [12] proposed a security taxonomy and overhead framework. Xie *et al.* [31] have also studied and performed experiments to derive security quality and security overhead for certain security mechanism of three types security services: confidentiality, authentication techniques and data integrity.

In the group-based security model, yet another challenge is how the weight of different groups should be assigned. The weight assignment to the groups of services can be done through security risk analysis methodologies [7] which include asset identification, vulnerability analysis, likelihood analysis and countermeasure evaluation.

There have also been other works addressing the security issue [25], [26] in real-time systems. Xie *et al.* [34] have presented challenges and open issues in security-aware real-time scheduling for distributed systems. The focus of [34] is to design a dynamic scheduling algorithm to schedule real-time tasks to achieve high security of the system while maintaining low miss rate of the tasks.

However, the problem of predictability in a real-time system with security requirement was not addressed in these works. Our work addresses static schedulability driven security optimization. To the best of our knowledge, this is the first work on static security aware schedulability analysis and static security optimization in real-time systems.

III. System Model

A. Task Model

We consider a task set T with N independent, preemptable periodic tasks to be scheduled on a single processor. A periodic task t_k is identified by the four tuple $\langle e_k, p_k, d_k, M_k\rangle$ where

- e_k is the (worst case) execution time of the task;
- p_k is the period of the task;
- d_k is the relative deadline of the task;
- M_k is the number of data items in the message that need security service.

Note that $\langle e_k, p_k, d_k\rangle$ is a common model for representing a real-time task [15]. The worst execution time can be derived using high level program analysis [20], [21] and low level program analysis that considers platform features [8]. What we add here is the communication factor of a task that need security services.

B. Security Model

Real-time systems are developed to execute a wide range of unverified user-implemented applications. Therefore, real-time applications as well as systems are very likely to be targets of security threats. Even worse, legitimate users might by accident tamper with shared data or excessively consume real-time computing resources to disrupt real-time services available to other applications. Many existing real-time computing environments do not incorporate security mechanisms to counter various security threats. Consequently, it becomes critical and important to employ security services to protect security-critical real-time systems. The basic idea of our security model is that real-time system users can flexibly select security services to form an integrated security protection against a wide spectrum of threats in real-time systems.

Next, we present a general model to specify the security services of the systems.

1) Security Services: A system may provide many security services which can be partitioned into groups where the services in the same group provide the same type of service but of different quality due to the different mechanism used. For example, a group of services can be targeting authentication specifically and another group of services can be targeting for protecting data integrity etc. We can use RC4, RC5, or DES for message encryption. Their quality is different. Security services from different groups can be combined to achieve better quality.

Assume there are N_g groups of security services. Each group contains different security services that provide the same type of security, but has different quality. NS_i is used to indicate the number of services in group G_i and S_{ij} indicates the jth service in the ith group. Each security service has a quality value. It is assumed that the highest quality of the services in each group is normalized to 1 and the other services have quality value between 0 and 1. The quality value of each service can be thought of as a value compared to the group's best security service. The services are ordered in a decreasing order of their quality. Each service incurs computation overhead to the systems. In this paper, two main factors that introduce overhead to the system are considered. The first one is the overhead to establish (initialize) the service and the second one is the *unit* overhead to apply the security service to the messages. A three tuple is used to represent a security service S_{ij}: $\langle Q_{ij}, I_{ij}, U_{ij}\rangle$ where

- Q_{ij} indicates the quality value for service S_{ij};
- I_{ij} indicates the overhead for initialization of the service S_{ij};
- U_{ij} indicates the unit overhead for applying the service S_{ij} to one data item.

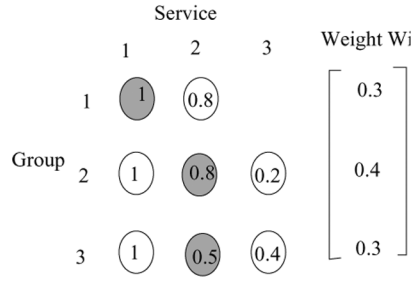


Fig. 1. Combining security services.

2) Security Overhead: To reduce management overhead, all tasks in a system use the same set of security services. The CPU overhead of applying service S_{ij} to a real-time system with a task set T can be computed as follows:

$$O_{ij} = \sum_{k=1}^{N} (I_{ij} + U_{ij} * M_k)$$

where M_k is the message (the number of data items) in task t_k that the security service performs on.

3) Combining Security Services: Multiple security mechanisms may be used to form an integrated security solution, which in turn can fulfill high security requirements. For instance, message encryption can be used in combination with message integrity (see, for example, [17]). A set of security services can perform as building blocks for real-time application developers to implement integrated security mechanisms for real-time systems.

The challenging issue is to compute the quality of the multiple security services that fall into different types of security service groups. The security value of multiple security services depends on the weight of each group. Weights to each group (type of service) are assigned with the sum of the weights for all the groups as 1. By simply computing the weighted sum of quality values of the selected services, we can derive the quality value for the combined services. Let us use W_i to denote the weight of the ith group.

Fig. 1 shows three groups of services. The quality of each service is represented inside the corresponding node. The weight of each group is shown in the vector on the right hand side. If services S_{11}, S_{22}, and S_{32} are selected, then the combined security value can be computed as follows:

$$W_1 * S_{11} + W_2 * S_{22} + W_3 * S_{32} = 0.3 * 1 + 0.4 * 0.8 + 0.3 * 0.5.$$

The intention of security service combination is to provide separate basic security services which offer users an array of fundamental security services. The user can then select multiple security services to form an integrated security protection to meet their needs. It is the users' responsibility to make a meaningful combination of basic security services. This is done through a thorough study on the relationship and interaction between a security mechanism and the system and between different security mechanisms themselves. One should note that not all the security mechanisms can be simply combined. For example, if a shared key encryption (one security mechanism implementation) is chosen for achieving both confidentiality and authentication, then we need to put confidentiality and authentication in the same group. In general, if the implementation of the two types of service mechanisms is not independent, then we cannot separate the services into two groups.

There exist some other cases where a service combination does not make sense. For example, a system that is vulnerable to unauthorized access does not benefit from any integrity or confidentiality services. Our model can still accommodate these cases by properly assigning the weights of the groups. In the above example, the weight of authorization service should be set to 1, whereas the weights of integrity and confidentiality services are 0. In essence, the system is reduced to a single-group and the problem becomes picking the best security service in this group which also guarantees schedulability. Yet one is still encouraged to further analyze the system to identify some fundamental smaller security service building blocks that can be combined to achieve the security requirement of this single group.

C. Security Requirements

There are two types of security requirements.

- First of all, we can specify the minimal security value for each type (or group) of service. Let MinGQ_i denote the minimal security required for the ith service group. It means that the quality value of the selected service in group i must not be less than MinGQ_i. The minimal group securities can be represented as a vector

$$\langle \mathrm{MinGQ}_1 \ldots \mathrm{MinGQ}_i \ldots \mathrm{MinGQ}_{N_g} \rangle.$$

- Second, we can specify the minimal combined security value. Remember that the combined security value is the weighted sum of the quality value of the individual services. We use MinCQ to denote this required minimal combined security value.

IV. Security Aware EDF Schedulability Test

An EDF [29] scheduler schedules tasks based on their deadline. EDF scheduling policy is the optimal scheduling policy for independent preemptable tasks scheduling on single processor meaning that if a set of independent preemptable tasks are schedulable onto a single processor by any scheduler, then they are schedulable by an EDF scheduler. A good property of EDF scheduling is that we can decide the schedulability of a set of tasks statically.

Next, the EDF schedulability analysis is described. Let us define the density of a task t_k as $(e_k)/(\min(d_k, p_k))$ where e_k, d_k, and p_k are the execution time, deadline and the period for task t_k. To verify whether a set of tasks are schedulable under EDF scheduling policy, we can use a very simple and robust schedulability test by checking

$$\sum_{k=1}^{N} \frac{e_k}{\min(d_k, p_k)} \leq 1.$$

This test follows directly from the following theorem [15].

Theorem: A system of independent, preemptable tasks can be feasibly scheduled on one processor if the density of the system is equal to or less than 1.

We use $\delta E = \sum_{k=1}^{N}(e_k)/(\min(d_k, p_k))$ to represent the original density of the system. Now we show how to compute the new density of the system when considering security service overhead.

Recall that the CPU overhead of applying service S_{ij} to a real-time system with a task set T can be computed as follows (see Section III-B2):

$$O_{ij} = \sum_{k=1}^{N}(I_{ij} + U_{ij} * M_k).$$

We define δ_{ij} as the extra density of applying service S_{ij} to the system. We have

$$\delta_{ij} = I_{ij} * \sum_{k=1}^{N} \frac{1}{\min(d_k, p_k)} + U_{ij} * \sum_{k=1}^{N} \frac{M_k}{\min(d_k, p_k)}.$$

Let

$$C_1 = \sum_{k=1}^{N} \frac{1}{\min(d_k, p_k)}$$

and

$$C_2 = \sum_{k=1}^{N} \frac{M_k}{\min(d_k, p_k)}.$$

δ_{ij} can be computed as

$$\delta_{ij} = C_1 * I_{ij} + C_2 * U_{ij}.$$

We define δS as the total extra density of applying all the services in this combination.

The security aware schedulability test then becomes

$$\delta E + \delta S \leq 1.$$

V. Security Optimization for EDF-Based Real-Time Systems

The process to perform security optimization in a real-time system includes the following steps.

1) Identify the groups of security requirements and determine the weight for each group. The process starts with security requirement analysis to identify the type of requirement needed. To do this, we need to analyze the tasks and the data involved in the application, perform risk analysis and decide what kind of security services are needed and determine the weight of each type (group) of requirement services.
2) Identify the number of data items in each task that need security services from each group.
3) Identify security variants for each security service group. For each security service in the groups, we need to identify the security variants with its security value, the two overhead parameters: initialization time, and time for servicing a unit data.
4) Formulate the optimization problem. The formulation needs to consider the constraints of minimal security value of each group and minimal security value of the combined security problem.
5) Solve the problem.

A. Problem Formulation

To achieve high quality of a real-time system, we can combine services from two or more groups. The services in the same group provide the same type of service. Therefore, only one service can be selected from one group. It is possible that we do not need all types of services for a system. To allow zero service be selected from a certain group, one just needs to add an idle service to this group. The idle service's Q, I, and U values are set to 0. Not selecting any service from a group means selecting an idle service from the group.

A combination of services is represented as a vector $\langle s_1, \cdots s_i, \cdots s_{N_g} \rangle$ where i is the group number and s_i is the service selected from group i. The quality value and overhead parameters for s_i are $\langle Q_i, I_i, U_i \rangle$. Our goal is to maximize the weighted sum of quality values $\mathrm{CQ} = \sum_{i=1}^{N_g} W_i * Q_i$, subject to the following constraints.

1) Only one service is selected from each group.
2) Minimal group-security quality constraint: the security value of the selected service in each group has to be bigger than or equal to the minimal required security value for that group. That is,

$$Q_i \geq \mathrm{MinGQ}_i, \text{ for any } i(1 \leq i \leq N_g).$$

3) Minimal combined-security quality constraint: the combined security has to be not less than the minimal required combined security. That is,

$$\mathrm{CQ} \geq \mathrm{MinCQ}, \text{ where } \mathrm{CQ} = \sum_{i=1}^{N_g} W_i * Q_i.$$

4) Schedulability constraint: the overhead will not make the system utilization larger than 1. Recall that $\delta E = \sum_{k=1}^{N}(e_k)/(\min(d_k, p_k))$ is the original density of the system. Also δ_{ij}, the extra density of applying service S_{ij} to the system is

$$\delta_{ij} = I_{ij} * \sum_{k=1}^{N} \frac{1}{\min(d_k, p_k)} + U_{ij} * \sum_{k=1}^{N} \frac{M_k}{\min(d_k, p_k)}.$$

δS, the total extra density of applying all the services in this combination can be computed as

$$\delta S = C_1 * \sum_{i=1}^{N_g} I_i + C_2 * \sum_{i=1}^{N_g} U_i$$

where

$$C_1 = \sum_{k=1}^{N} \frac{1}{\min(d_k, p_k)}$$
$$C_2 = \sum_{k=1}^{N} \frac{M_k}{\min(d_k, p_k)}.$$

The schedulability test requires that: $\delta E + \delta S \leq 1$.

B. Using the Integer Linear Programming Technique

The first method to solve the problem is the integer linear programming (ILP) technique. What needs to be done is to formulate the problem as an ILP problem.

First, the variables of the system under consideration need to be defined. Then the goal and the constraints over the defined variables are described.

For each service of a group, a variable x_{ij} is defined where i represents the group number $(1 \leq i \leq N_g)$ and j represents the service number $((1 \leq j \leq \mathrm{NS}_i)$. x_{ij} being either 1 or 0 indicates that the jth service of the ith group is selected or not selected to apply to the system.

Our aim is to find an assignment for each x_{ij} to maximize the combined security of the system subject to the minimal security requirement constraint and the schedulability constraint.

The goal function can be expressed as

$$\max \mathrm{CQ} = \sum_{i=1}^{N_g} W_i * Q_i = \sum_{i=1}^{N_g} W_i * \left(\sum_{j=1}^{\mathrm{NS}_i} x_{ij} * Q_{ij} \right).$$

There are four types of constraints.

1) First, only one service is selected from each group. Therefore, for each group $i(1 \leq i \leq N_g)$, we have one constraint

$$\sum_{j=1}^{\mathrm{NS}_i} x_{ij} = 1.$$

2) Minimal group-security quality constraint: the quality of the selected service from any group has to be not less than the minimal required quality value for that group. Therefore, for each group i where $(1 \leq i \leq N_g)$, we have one constraint

$$Q_i = \sum_{j=1}^{\mathrm{NS}_i} x_{ij} * Q_{ij} \geq \mathrm{MinGQ}_i.$$

3) Minimal combined-security quality constraint: the combined quality of the selected service cannot be less than the minimal required value. That is,

$$\mathrm{CQ} = \sum_{i=1}^{N_g} W_i * \left(\sum_{j=1}^{\mathrm{NS}_i} x_{ij} * Q_{ij} \right) \geq \mathrm{MinCQ}.$$

TABLE I
SECURITY SERVICE GROUPS

Group	$\mathbf{S}_{ij}$	$\mathbf{Q}_{ij}$	$\mathbf{I}_{ij}$	$\mathbf{U}_{ij}$	$\mathbf{MinGQ}_i$	$\mathbf{W}_i$	**MinCQ**
G1	S_{11}	1	0.4	0.05	0.5	0.5	0.6
	S_{12}	0.8	0.5	0.04			
	S_{13}	0.4	0.4	0.04			
G2	S_{21}	1	0.9	0.28	0.3	0.5	
	S_{22}	0.9	0.2	0.015			
	S_{23}	0.2	0.15	0.007			

TABLE II
TASK SET

Task	$\mathbf{t}_k$	$\mathbf{e}_k$	$\mathbf{p}_k$	$\mathbf{d}_k$	$\mathbf{M}_k$
Task1	t_1	4	40	50	40
Task2	t_2	1	20	25	30
Task3	t_3	1	10	15	10

4) Schedulability constraint: the system should still be able to pass the EDF schedulability test with the overhead incurred with the selected services

$$\delta E + \sum_{i=1}^{N_g} \sum_{j=1}^{\mathrm{NS}_i} x_{ij} * \delta_{ij} \leq 1$$

where

$$\delta_{ij} = C_1 * I_{ij} + C_2 * U_{ij}$$
$$C_1 = \sum_{k=1}^{N} \frac{1}{\min(d_k, p_k)}$$
$$C_2 = \sum_{k=1}^{N} \frac{M_k}{\min(d_k, p_k)}.$$

1) Simple Example: In this simple example, there are two groups of security services and three real-time tasks, tabulated in Tables I and II, respectively.

Now we can formulate our problem.

1) The goal function we try to maximize is

$$\max : w_1 x_{11} Q_{11} + w_1 x_{12} Q_{12} + w_1 x_{13} Q_{13} + w_2 x_{21} Q_{21} + w_2 x_{22} Q_{22} + w_2 x_{23} Q_{23}. \quad (1)$$

In this example, it is

$$\max : 0.5 x_{11} + 0.4 x_{12} + 0.2 x_{13} + 0.5 x_{21} + 0.45 x_{22} + 0.1 x_{23}. \quad (2)$$

2) The problem constraints are as follows.

- Only one service is selected from each group:

$$x_{11} + x_{12} + x_{13} = 1$$
$$x_{21} + x_{22} + x_{23} = 1. \quad (3)$$

- Minimal group-security quality constraint:

$$x_{11} Q_{11} + x_{12} Q_{12} + x_{13} Q_{13} \geq 0.5$$
$$x_{21} Q_{21} + x_{22} Q_{22} + x_{23} Q_{23} \geq 0.3. \quad (4)$$

In this example, it will be

$$x_{11} + 0.8x_{12} + 0.4x_{13} \geq 0.5 \quad (5)$$

and

$$x_{21} + 0.9x_{22} + 0.2x_{23} \geq 0.3. \quad (6)$$

- Minimal combined-security quality constraint:

$$w_1x_{11}Q_{11} + w_1x_{12}Q_{12} + w_1x_{13}Q_{13} + w_2x_{21}Q_{21} + w_2x_{22}Q_{22} + w_2x_{23}Q_{23} \geq 0.6. \quad (7)$$

In this example, it is

$$0.5x_{11} + 0.4x_{12} + 0.2x_{13} + 0.5x_{21} + 0.45x_{22} + 0.1x_{23} \geq 0.6. \quad (8)$$

- Schedulability constraint:

$$\begin{aligned} &x_{11}I_{11}C_1 + x_{12}I_{12}C_1 + x_{13}I_{13}C_1 \\ &+ x_{21}I_{21}C_1 + x_{22}I_{22}C_1 + x_{23}I_{23}C_1 \\ &+ x_{11}U_{11}C_2 + x_{12}U_{12}C_2 + x_{13}U_{13}C_2 \\ &+ x_{21}U_{21}C_2 + x_{22}U_{22}C_2 + x_{23}U_{23}C_2 + E \leq 1 \end{aligned} \quad (9)$$

where

$$\begin{aligned} E &= \frac{e_1}{p_1} + \frac{e_2}{p_2} + \frac{e_3}{p_3} \\ C_1 &= \frac{1}{p_1} + \frac{1}{p_2} + \frac{1}{p_3} \\ C_2 &= \frac{M_1}{p_1} + \frac{M_2}{p_2} + \frac{M_3}{p_3}. \end{aligned} \quad (10)$$

In this example, it is

$$0.0245x_{11} + 0.2275x_{12} + 0.21x_{13} + 1.1375x_{21} + 0.0875x_{22} + 0.0508x_{23} + E \leq 1 \quad (11)$$

where

$$\begin{aligned} E &= \frac{e_1}{p_1} + \frac{e_2}{p_2} + \frac{e_3}{p_3} \\ &= \frac{4}{40} + \frac{1}{20} + \frac{1}{10} \\ &= 0.1 + 0.05 + 0.1 \\ &= 0.25, \end{aligned} \quad (12)$$

$$\begin{aligned} C_1 &= \frac{1}{p_1} + \frac{1}{p_2} + \frac{1}{p_3} \\ &= \frac{1}{40} + \frac{1}{20} + \frac{1}{10} \\ &= 0.025 + 0.05 + 0.1 \\ &= 0.175, \end{aligned} \quad (13)$$

$$\begin{aligned} C_2 &= \frac{M_1}{p_1} + \frac{M_2}{p_2} + \frac{M_3}{p_3} \\ &= \frac{40}{40} + \frac{30}{20} + \frac{10}{10} \\ &= 1 + 1.5 + 1 \\ &= 3.5. \end{aligned} \quad (14)$$

3) x_{ij} are integers.

After we formulate the whole problem, we can write all of them into a x.lp file and solve it using lp_solve function [1].

The running results are

- Value of objective function: 0.95.
- Actual values of the variables:
  1) $x_{11} = 1$
  2) $x_{12} = 0$
  3) $x_{13} = 0$
  4) $x_{21} = 0$
  5) $x_{22} = 1$
  6) $x_{23} = 0$.

C. Using a Search Technique

The second method is a search-based method. First the problem is formulated as a search problem in a graph and then a search technique is used to solve the problem. The algorithm is efficient since some heuristics are applied to reduce the search space.

First a graph which contains N_g layers is constructed. Each layer contains several nodes. A *node* is a pair $\langle i, j\rangle$ where i indicates the group number and j indicates the service number. Node $\langle i, j\rangle$ resides at the jth position of the ith layer.

A *path* $P = s_1, s_2 \ldots s_l (l \leq N_g)$ in the graph is a sequence of nodes chosen from each layer in order, where l is the length of the path. A path can be *complete* or *incomplete*. A complete path contains one node from each layer while an incomplete path contains fewer nodes than N_g.

A path $P = s_1, s_2 \ldots s_l$ *passes* a node $\langle i, j\rangle$ iff $s_i = j \wedge i \leq l$. The quality value of a path $P = s_1, \ldots, s_l$, denoted by $\mathrm{CQ}(P)$, is defined as

$$\mathrm{CQ}(P) = \sum_{i=1}^{l} (W_i * Q_i)$$

where Q_i is the quality value of the s_ith service in the ith group. We define $\delta S(P)$ as the extra density incurred by the services selected in the path. Both $\mathrm{CQ}(P)$ and $\delta S(P)$ can be computed incrementally meaning that if we add a new node into a path P, we can add the new quality value and the δS of the node to derive the new $\mathrm{CQ}(P)$ and $\delta S(P)$. A path P is *infeasible* if and only if P violates one of the four types of constraints. Otherwise, P is said to be *feasible*.

A complete path $P = s_1, s_2, \ldots, s_{N_g}$ corresponds to a service combination $S = \langle s_1, s_2, \ldots, s_{N_g}\rangle$ in the original problem. The aim for the search problem is to find a feasible complete path with maximal combined quality value.

To solve the problem, depth-first search [6] with backtracking is used. Three kinds of heuristics are developed to guide the search procedure and cut branches. The three types of pruning factors are the following.

1) The first type of pruning eliminates the nodes in each layer whose quality value is smaller than the minimal required group-security. This is called *group quality pruning*. This type of pruning greatly reduces the search space. Fig. 2 shows the quality value of each service of the four groups and the required minimal group security value in the

MinGQ vector. Obviously, service S_{13}, S_{14}, S_{23}, and S_{44} can be eliminated before the search starts.

2) Some branches can be pruned because the incomplete path tested already fails the schedulability test. The search is realized by expanding nodes and backtracking. It wastes time in continuing expansion when a path is found to be infeasible. This is because any path with an infeasible path as prefix is still an infeasible path. Therefore, the paths expanded from an infeasible path can be pruned. We call this *schedulability pruning*.
   Let us consider an example with four groups (Fig. 3) where group 1 has two services and the other two groups have three services. Suppose the current path is $P = \langle 1,1\rangle, \langle 2,2\rangle, \langle 3,1\rangle$ [see Fig. 3(b)] and $\delta S(P)$ violates the schedulability test. Therefore, path P is not a feasible path. We should not extend this path any further. That is, the paths shown in Fig. 3(d)–(f) will be pruned. The immediately subsequent path considered during search is shown in Fig. 3(c).
3) The impossibility to exceed the current combined quality bound: some branches can be pruned since the path's combined quality cannot exceed that of the feasible path found before. We called such pruning *quality bound pruning*. When a complete feasible path is found, the search procedure cannot stop since the path found does not necessarily have the largest combined quality value. However not all its subsequent paths are of interest since some of them obviously have a smaller combined quality than that of the current path. It is easy to prove (remember that the services of each group are ordered by decreasing quality) that path $P' = s'_1, \ldots, s'_{N_g}$ has a smaller combined quality value than $P = s_1, \ldots, s_{N_g}$ if

$$\forall i((1 \le i \le N_g) \rightarrow (s_i \le s'_i)).$$

P' has the following property w.r.t. P: at any level, the node selected for P' is on the right-hand side of that selected for P. Look-ahead and back-jumping mechanisms can be used to prune subsequent paths with such a property. With back jumping, the search may jump a few levels up. The level to jump to is derived based on the following two observations.

- Rule 1) an increment at the final level of a path will not lead to a larger combined quality than the original path, for the reason that the security quality values of services in a group are in the decreasing order. Therefore, the level is at most $N_g - 1$. For example, if the search reaches the state in Fig. 4(a), then the next search step will jump back to Fig. 4(a′) without entering into calculating the path passing the third node at level 4.
- Rule 2) if the node of a path P at every level from L to N_g is in the leftmost position, then increasing P at any level between $L-1$ and N will not produce a larger combined quality value than the original path P. Therefore, the backtracking level can be set to $L-2$. For the path illustrated in Fig. 4(b), the nodes from level 3 to level 4 are in the leftmost positions. Here $L = 3$. Intuitively, the search can jump back to the third node at level 2 and starts from there. This path can also be pruned because its security value is definitely less than that of Fig. 4(b′). Therefore, the next search actually reaches the path shown in Fig. 4(b) with the $\text{current level} = L-2 = 1$. Fig. 4(c) is another example applicable using rule 2. Here $L = 2$. Using rule 2, the search can start from a node in level $L - 2 = 0$. Note that there is no subsequent path for the path shown in Fig. 4(c′) meaning that the search can terminate.

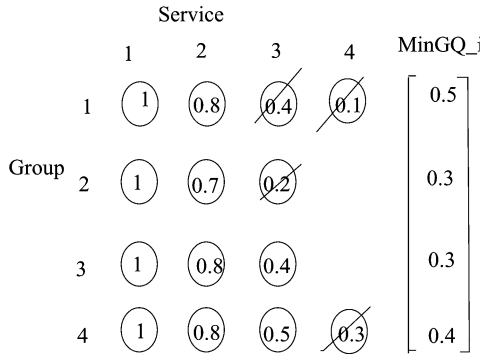


Fig. 2. Group quality pruning.

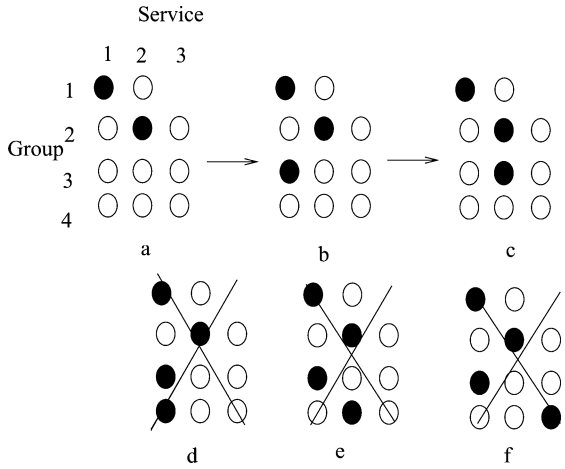


Fig. 3. Schedulability pruning.

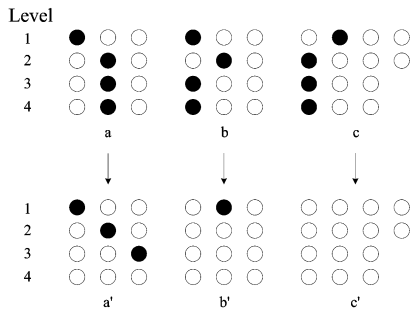


Fig. 4. Quality bound pruning.

Let P be the complete path found that has the maximal combined quality CQ. If $\text{CQ} < \text{MinCQ}$, then it is not possible to find a feasible combination of services satisfying both the security requirement and schedulability requirement.

Given N: the number of tasks; N_g: the number of service groups; NS_i: the number of services in group i, $i \in [0, N_g)$; a set of tasks: $t_k = \langle e_k, p_k, d_k, M_k \rangle$, $k \in [0, N)$; the services: $S_{ij} = \langle Q_{ij}, I_{ij}, U_{ij} \rangle$, $i \in [0, N_g), j \in [0, \mathrm{NS}_i)$; and the security constraints: MinGQ_i for each group i, and MinCQ, the pruning search Algorithm 1 returns the optimal feasible service combination with the maximal security combined value.

Pruning Search for Security Optimization

1) **if** the path found by Minimum policy is not schedulable **then**
2) **return** $\emptyset; \{\mathrm{OptimalPath} = \emptyset\}$
3) **end if**
4) **Group Quality Pruning**
5) delete services not satisfying MinGQ_i from each group;
6) *Initialize the flags and variables*
7) $\delta E = \sum_{k=1}^{N}(e_k)/(\min(d_k, p_k))$;
8) $\mathrm{hasMorePath} = 1; \mathrm{foundMaxCQ} = 0;$
9) $\mathrm{maxCQ} = 0; \{\mathrm{maxCQ} : \text{the CQ for OptimalPath}\}$
10) $\delta S = 0; \{\delta S : \text{density overhead for the current path}\}$
11) **for** $i = 1$ to N_g, $j = 1$ to NS_i **do**
12) compute $\delta_{[i][j]}; \{\delta_{[i][j]} : \text{density overhead of } S_{ij}\}$
13) **end for**
14) $L = 0; \{L : \text{the current group level}\}$
15) $\mathrm{selected}[0] = 0; \{\mathrm{select}[i] :$ the selected service in level $i\}$
16) **while** *hasMorePath && !foundMaxCQ* **do**
17) **while** $0 \leq L < N_g$ **do**
18) $s =$ the first service $y : y \in [\mathrm{selected}[L], \mathrm{NS}_L)$ and $\delta S + \delta_{[L][y]} \leq 1 - \delta E$;
19) **if** $\exists s$ **then**
20) $\mathrm{selected}[L] = s; \delta S+ = \delta_{[L][s]};$
21) $L++; \mathrm{selected}[L] = 0;$
22) **else**
23) **Schedulability Pruning**
24) **if** $L \leq 0$ **then**
25) $\mathrm{hasMorePath} = 0;$ **break**;
26) **else**
27) $L--; \delta S- = \delta_{[L][\mathrm{selected}[L]]};$
28) $\mathrm{selected}[L]++;$
29) **end if**
30) **end if**
31) **end while**
32) **if** $L == N_g$ **then**
33) calculate CQ of the current complete path;
34) **if** $\mathrm{CQ} > \mathrm{maxCQ}$ **then**
35) replace *OptimalPath* with the current path; $\mathrm{maxCQ} = \mathrm{CQ};$
36) **end if**
37) *Search a new path with Quality Bound Pruning*
38) $L = N_g - 1;$
39) **while** $L > 0$ $\mathrm{selected}[L] == 0$ **do**
40) $\delta S- = \delta_{[L][\mathrm{selected}[L]]}; L--;$
41) **end while**
42) **if** $L > 0$ **then**
43) $\delta S- = \delta_{[L][\mathrm{selected}[L]]}; L--;$
44) $\delta S- = \delta_{[L][\mathrm{selected}[L]]}; \mathrm{selected}[L]++;$
45) **else**
46) $\mathrm{foundMaxCQ} = 1;$
47) **end if**
48) **end if**
49) **end while**
50) **if** $\mathrm{maxCQ} \geq \mathrm{MinCQ}$ **then**
51) **return** *OptimalPath*;
52) **else**
53) **return** $\emptyset$;
54) **end if**

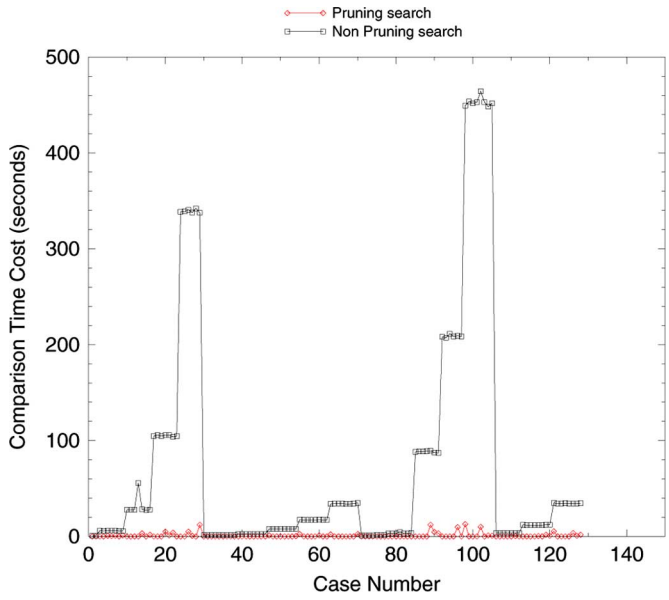


Fig. 5. Pruning search versus nonpruning search.

To show the effectiveness of the proposed heuristics, we compared the time cost of the pruning search algorithm with that of the simple search algorithm without any pruning heuristic (nonpruning search). The hardware platform is a dual Intel E4600 2.40 GHz processors with 3 GB of memory. The operating system is Linux Redhat with kernel 2.6.18–92.1.10.el5xen. We randomly generated 684 test cases where the number of group (N_g) is within the range [4, 10], the number of security services (NS) in each group is within the range [5,12] and the number of tasks is within [3,15] which we believe are the reasonable ranges for the parameters. In the 684 test cases, 483 cases are feasible. The time costs for both algorithms of 355 test cases out of the 483 cases are less than 1 second and the comparison of time cost of the rest of 128 test case is shown in Fig. 5. Note that when the time cost is less than 1 s, it is shown as 0 s in the figure. The results show that the pruning search has great advantages over simple search as the time cost of the pruning search is much less than that of simple search, and the time cost of pruning search increases slower than the simple search does when the parameters (i.e., N_g, NS, N) increase.

We also compare the lp_solve and the pruning search algorithm using the same set of test cases. Among the 483 feasible cases, there are 459 cases where the time costs of lp_solve and pruning search are both less than 1 s. This suggests that both algorithms are efficient. The performance of lp_solve and the

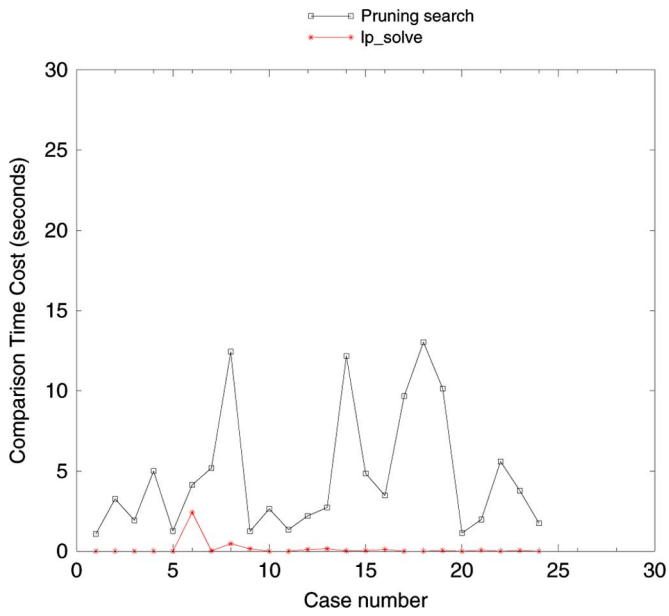


Fig. 6. Comparison of lp_solve and pruning search.

pruning search for the rest 24 test cases (all with $N_g \geq 8$) is compared in Fig. 6. lp_solve is faster for these cases. But the time cost of the pruning search is also reasonably short (from a few seconds to 14 s).

D. Special Case: Service Selection From a Single Group

When there is only one group of security services in the system, the index for the group number is always 1. Remember that our goal is to select the best security service and guarantee the schedulability of the system. If the ith service is the optimal service, it must satisfy the security aware EDF schedulability test which includes the service overhead.

First, the maximal allowed density overhead is computed as $1 - \delta E$. Then the density overhead for all the services in the group is calculated and stored in a vector. A binary search on the vector is used to find the service that has the maximal density overhead which does not exceed $1 - \delta E$. We then check if the quality value constraints are met for the service found. If yes, then the optimal security service is found. Otherwise, the system is not feasible.

VI. Experiments

Using the ILP or the heuristic search technique, a security-aware EDF schedule problem can be solved efficiently to obtain an optimal service combination. This section shows that our method can improve the security value of a system while guaranteeing schedulability comparing with other methods.

Extensive test suites were performed to compare two metrics: the combined security value and the schedulability value (also called the overall density value which is equal to the original task density plus the overhead density), using the following four policies (Minimum, Maximum, Random, and Optimal). If there is no feasible schedule, the policy outputs 0 as the combined security value.

- Minimum policy: the service with the lowest security level in each group is picked for its corresponding lowest computation overhead.
- Maximum policy: the service with the highest security quality in each group is selected.
- Random policy: for each group, the security service is picked randomly.
- Optimal policy: the services achieves the optimal security value while meeting all the constraints are picked.

A. Random Test Suites for the Comparison

There are five random test suites in total. For each test suite, we evaluate the combined security value and schedulability value for different policies. In each test suite, a variable is selected and scaled to observe the properties of the four policies. To limit the exploration states, we predetermine one or two variables, such as the number of tasks or the maximum number of security services in each group. Without loss of generality, all the other variables in our model are generated randomly. These five suites with randomly generated test data for representing the general situations are as follows.

- Test suite 1: The maximum number of services is fixed, so as the number of tasks. We set the number of services to 5 and the number of tasks to 10. Let the number of groups grow from 4 to 14 and the increment is 1. All the other data: security quality, initialization overhead, unit data processing overhead, weighted factors, minimal combined security value, task execution time, task period, task deadline, and the number of data items, are randomly generated. The combined security values and schedulability values of the four policies are illustrated in Fig. 7. When the number of security groups is 8, 10, or 13, the Maximum policy fails to fulfill the schedulability of the task set. It reveals that the security quality greedy approach (i.e., the Maximum policy) has its weakness in scalability. The Minimum policy can produce a feasible schedule, but the combined security value is much lower than that achieved by our optimal method. The average difference of the normalized combined security is 0.7003.
- Test suite 2: The number of security service groups and the number of real-time tasks are predetermined as 10. Let the maximal number of services in each group increase from 4 to 16 and the increment is 2. Fig. 8 shows the combined security and schedulability values. Our Optimal policy achieves the maximum combined security value while satisfying the schedulability constraints. More improvement in combined security value of Optimal policy vs. the minimum policy is shown in Fig. 8 than the curves in Fig. 7. The fact that more security services exist in a group allows more combination choices of the security services. The optimal policy utilizes this to provide more flexible security service combinations for the tasks and thus achieves higher improvement in the combined security values. It is shown that the combined security values of the Optimal policy are close to those of Maximum policy. Note that the schedules of Maximum policy are not

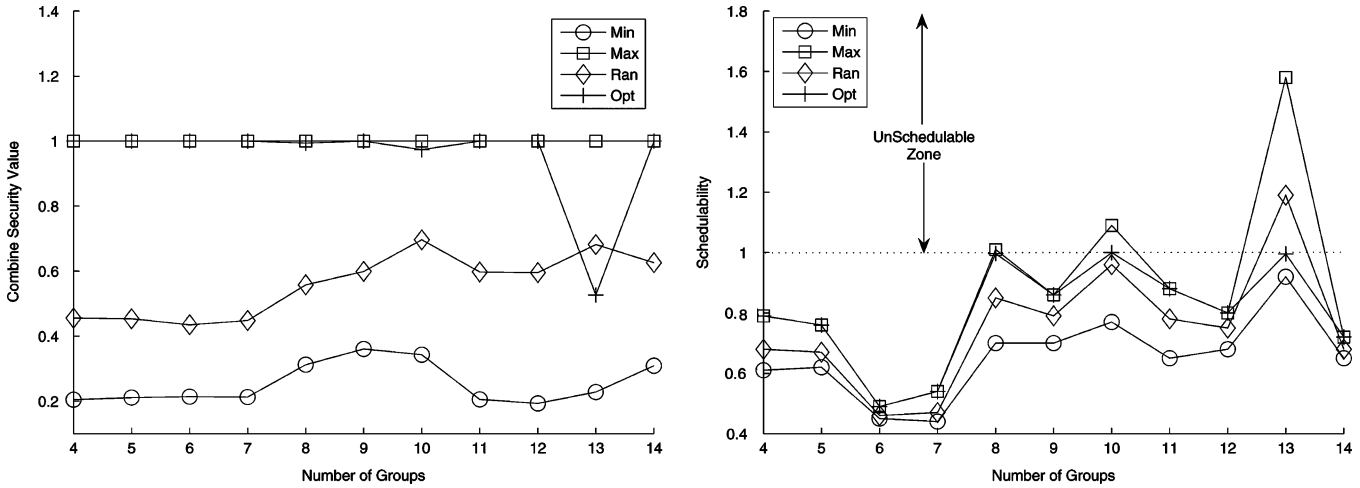


Fig. 7. Combined security and schedulability value for test suite 1: the number of groups as the variable.

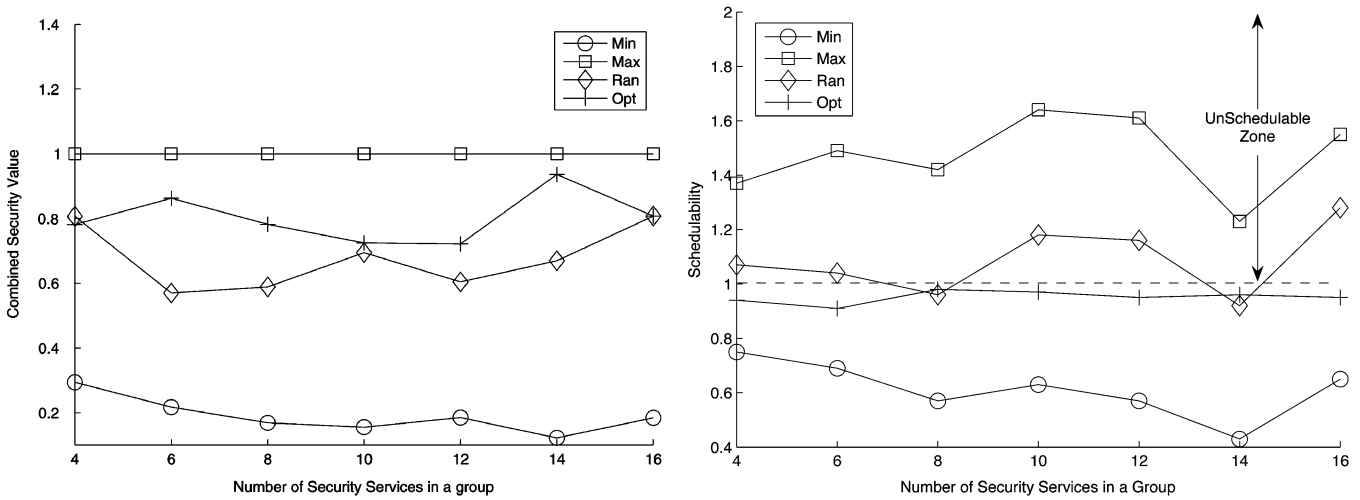


Fig. 8. Combined security and schedulability value for test suite 2: changing the number of security services in a group.

feasible for all seven cases, making its highest combined security values useless.

Furthermore, another three test suites with random test data are conducted to observe the advantage of our Optimal policy over the others in different scheduling overhead situations. Similar results can be concluded as test suite 1 and 2.

- Test suite 3: For ten groups and at most five services in each group, the number of tasks increases from 4 to 12. Fig. 9 illustrates the combined security and schedulability values for this test suite.
- Test suite 4: The number of groups, the number of services and the number of tasks are all fixed. The number of groups is 10. And the number of services in each group is 5 and the number of tasks is 10. The unit overhead U increases from 0.1 to 0.8, and the increment is 0.1. Fig. 10 shows the combined security and schedulability values.
- Test suite 5: In this test case, The number of groups, the number of services and the number of tasks are all fixed. We set the number of groups to 10, the number of services in each group to 5 and the number of tasks to 10. Let the initialization overhead I increase from 0.1 to 0.6, and the increment is 0.1. The results are shown in Fig. 11.

With the increase in the number of tasks (test suite 3), unit overhead (test suite 4) and initialization overhead (test suite 5), the feasible security service combination space is smaller. The Optimal policy ensures the schedulability of the real-time tasks and achieves the optimal combined security values, closest to those achieved by the Maximum policy.

Consequently, from the curves of all these five test suites, we can conclude that the Optimal policy is able to optimize the security services for a system without the loss of schedulability. Although at some points some policies like Maximum or Random policy can get a higher combined security values, the schedulability of the tasks at these points are all unschedulable which makes these higher security values useless.

B. *Effect of Overhead Parameter I and U*

A series of experiments were conducted to evaluate the impact of the parameters I_{ij} and U_{ij} for each security service S_{ij}, where i is the group index and j is the service index in the group.

The parameters predetermined are generated by the programs used in Subsection VI-A, listed in Tables III and IV. We perform fifteen security optimization problems where the scale factor of U steps up from 1 to 15. The results are shown in Fig. 12.

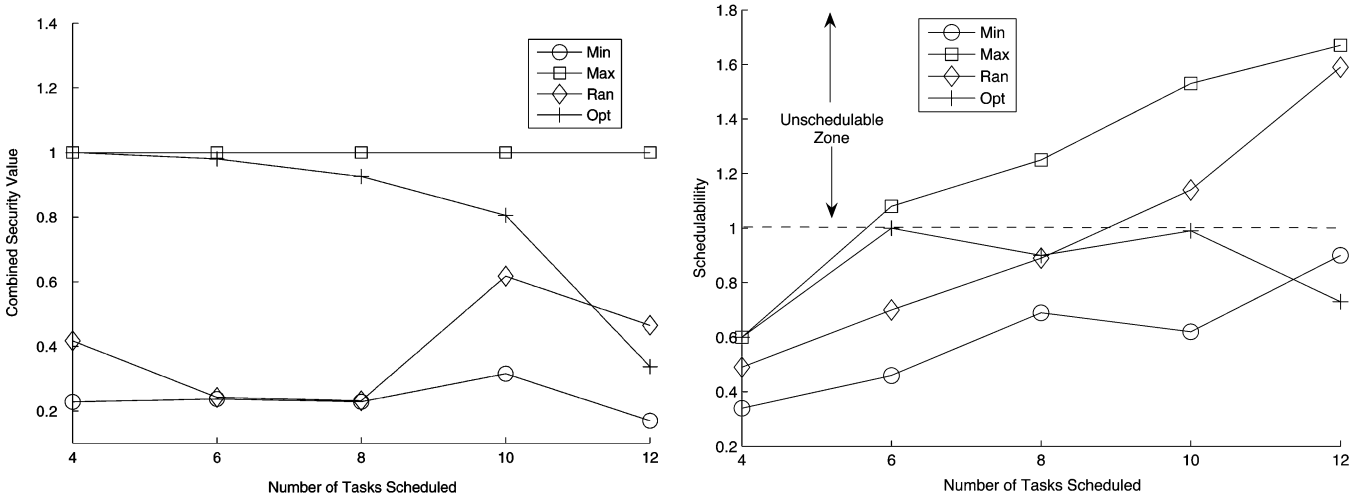


Fig. 9. Combined security and schedulability value for test suite 3.

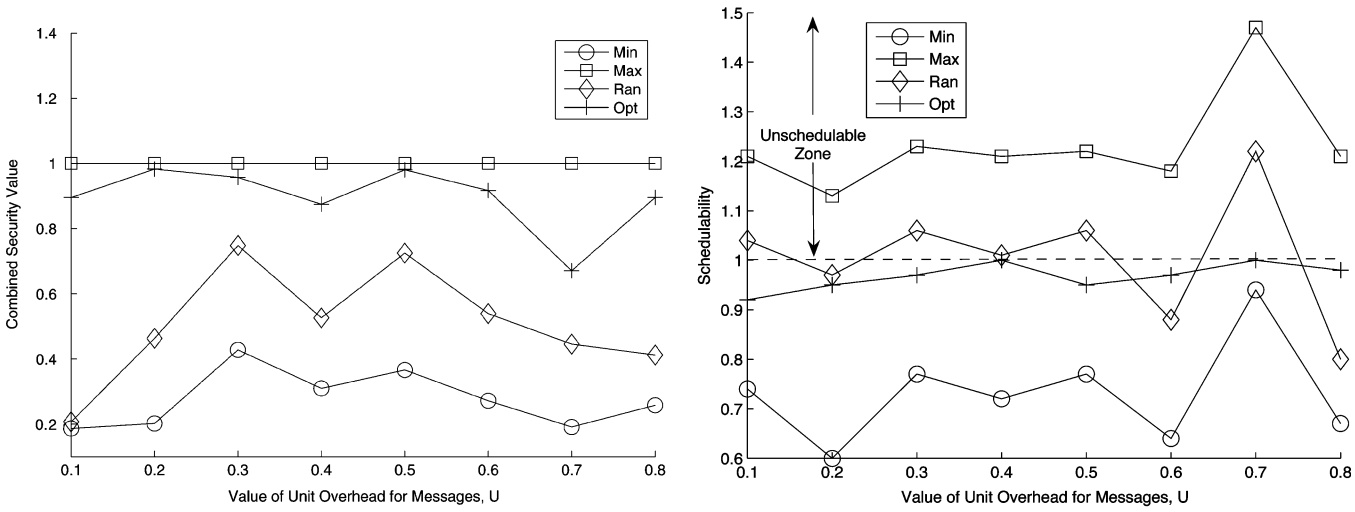


Fig. 10. Combined security and schedulability value for test suite 4.

The Maximum and Minimum policies have constant combined security value because they have fixed combination of security services. And the combined value of the Random policy is less than or equal to that of the Maximum policy and larger than or equal to that of the Minimum policy. When the scale factor of U increases, the overhead of the security services also increases. The schedulability constraint is more stringent. The Optimal policy achieves the highest combined security value the same as achieved by the Maximal policy when the scale factor of U is less than 5. When the scale factor of U is 5, the overall density of the Maximum policy is 1.06, meaning that the system is not schedulable. When the scale factor of U is greater than 5, the combined value achieved by the Optimal policy is lower than that achieved by the Maximum policy. This is because the Optimal policy chooses the services with lower overhead to tolerate the increase of U in order to produce a feasible schedule. When the factor is larger than 10, even the Minimum policy cannot find a feasible combination of services (the overall density of any combinations of services is larger than 1). Obviously, the Optimal policy cannot produce a feasible solution either as there exists no feasible solution. This suggests that the feasible upper bound of the scale factor of U is 10 for this specific security optimization problem when we only consider schedulability constraint. The feasible upper bound of the scale factor of U is reduced to 7 when the constraint of minimum combined security value (here, it is 0.74) is also taken into account. Note that in the figure the combined security value and the schedulablity value of Optimal policy are set to 0 when the case is not feasible (not schedulable or does not fulfill the minimum security requirement).

Similar situations are shown in Fig. 13 for initialization overhead, I. The scale factor of I increases from 1 to 4 with a step of 0.1. Constant combined security values are achieved by the Maximum policy and Minimum policy for their fixed selection of security services. The Optimal policy schedules the tasks with best feasible combined security values, larger than those achieved by the Random policy.

Based on the results shown in Figs. 12 and 13, we can conclude that the increase of overhead (by increasing either I or U) compresses the feasible security service combination space and thus lowers down the combined security level. The Optimal policy considers all the constraints and schedules the tasks with

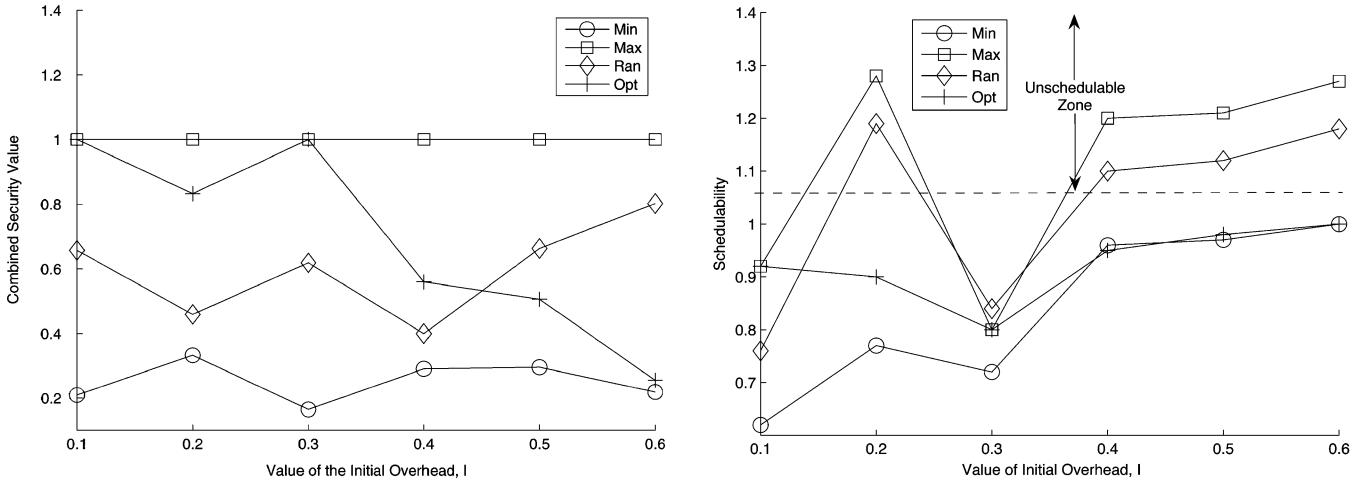


Fig. 11. Combined security and schedulability value for test suite 5.

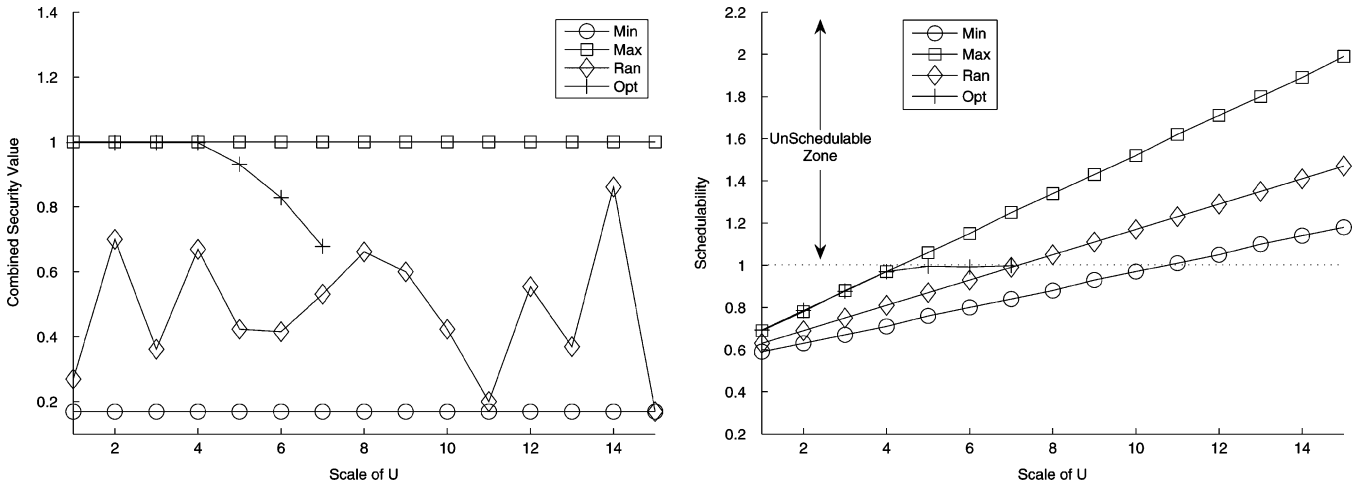


Fig. 12. Combined security and schedulability value for U scaled and other parameters predetermined.

TABLE III
PARAMETERS PREDETERMINED FOR EVALUATING THE IMPACT OF INITIALIZATION OVERHEAD I_{ij} AND THE OVERHEAD OF PROCESSING UNIT DATA, U_{ij}

name	$\mathbf{S}_{ij}$	$\mathbf{Q}_{ij}$	$\mathbf{I}_{ij}$	$\mathbf{U}_{ij}$	$\mathbf{MinGQ}_i$	$\mathbf{W}_i$	**MinCQ**
Group1	S_{11}	1	0.35	0.15	0.2	0.2	0.74
	S_{12}	0.69	0.18	0.12			
	S_{13}	0.21	0.12	0.08			
	S_{14}	0.07	0.10	0.05			
Group2	S_{21}	1	0.9	0.04	0.3	0.35	
	S_{22}	0.62	0.86	0.038			
	S_{23}	0.31	0.80	0.031			
	S_{24}	0.16	0.74	0.027			
Group3	S_{31}	1	0.2	0.072	0.4	0.4	
	S_{32}	0.75	0.15	0.063			
	S_{33}	0.25	0.13	0.047			
Group4	S_{41}	1	0.4	0.015	0.1	0.05	
	S_{42}	0.38	0.33	0.007			
	S_{43}	0.12	0.28	0.004			

TABLE IV
TASK SET

name	$\mathbf{t}_k$	$\mathbf{e}_k$	$\mathbf{p}_k$	$\mathbf{d}_k$	$\mathbf{M}_k$
Task1	t_1	7	57	63	49
Task2	t_2	5	32	39	21
Task3	t_3	3	21	52	7

highest combined security value, while the other three policies ignore the constraints of schedulability, minimal group security qualities and minimal combined security value.

C. Real Application

Next, we show the security optimization of a flight control system which was utilized to fly a simulated model of an F-16 fighter aircraft [2], [32]. The tasks in the system control the aircraft during flight. The set of task include *Guidance* which sets the reference trajectory of the aircraft in term of altitude and heading; *Controller* which executes the close loop control function that deals with actuator commands; the two *Navigation*

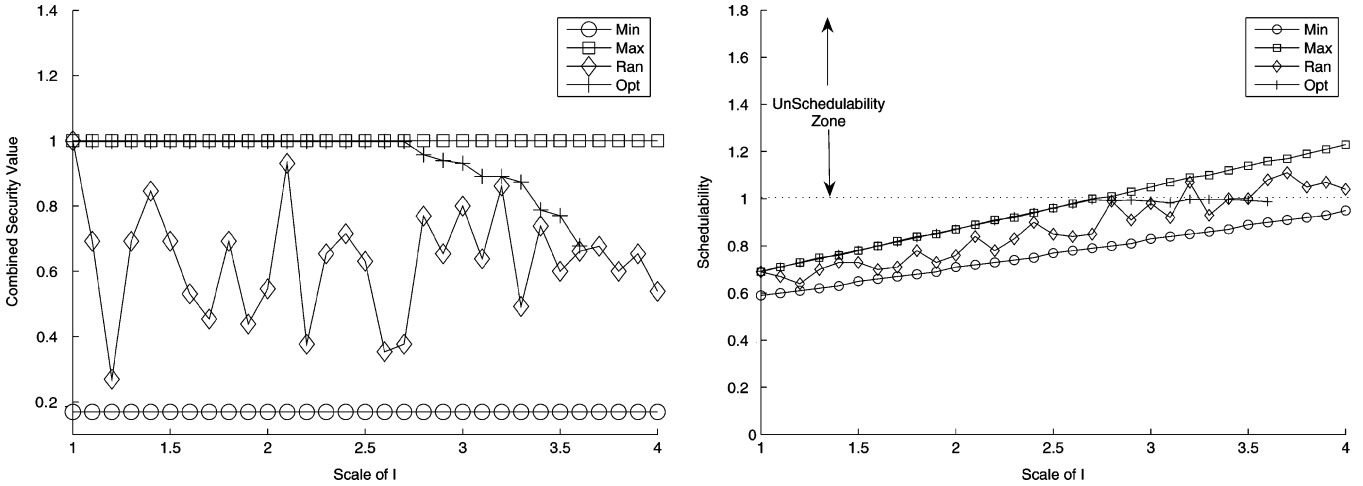


Fig. 13. Combined security and schedulability value for I scaled and other parameters predetermined.

TABLE V
AIRCRAFT CONTROLLER: THE TASK SET

Task	WCET (ms)	Period (ms)	Message (kb)
Guidance	100	1000	300
Controller	80	1000	200
Slow Navigation	100	1000	300
Fast Navigation	60	1000	300
Missile Control	500	10000	1000

tasks: *Fast Navigation* and *Slow Navigation*; and *Missile Control* which reads radar and fires missiles. The detailed description of the flight control system can be found in [2] and [32]. There are several versions of the original tasks which are distinguished by the task periods. We randomly chose one version of each task. The system has high security requirement to guarantee the messages between the subsystems not being read or altered by malicious users, and to guarantee the message sender really be the ones claiming to be. The size of the messages that need security service is randomly generated. The parameters of the tasks are shown in Table V.

As in [32], we focus on three groups of services: confidentiality, integrity and authentication. The services in each group are shown in Tables VI–VIII, respectively. We use the overhead provided in [32] where the overhead of the confidential services and integrity services is measured by served data per time unit (kb/ms) and the overhead of the authentication services is measured by service time (ms). The assumption is that no initialization overhead is involved in the encryption mechanisms and the integrity algorithms while the authentication algorithms take constant time. To transfer this to our overhead model, the initialization overhead (I) is zero for all the confidential services and integrity services and the corresponding U value is the inverse of served data per time unit. For example $U_{11} = 1/13.5$, $I_{11} = 0$. The unit data processing overhead (U) is zero for all the authentication services and the corresponding I value equals to the service time (ms). For example $I_{31} = 163$, $U_{31} = 0$. The security level is normalized by the timing performance [32]. The security level can also be derived using some other metrics like the likeness of the mechanism being attacked and the ability to protect the system when a brute force method is used. We assume that the weights for the three groups are 0.5, 0.3 and 0.2

TABLE VI
AIRCRAFT CONTROLLER: CONFIDENTIALITY SERVICES

Group 1 Services	Cryptographic Algorithm	Q level	Overhead (kb/ms)	I	U
S_{11}	IDEA	1	13.5	0	0.074
S_{12}	DES	0.90	15	0	0.067
S_{13}	RC5	0.46	29.35	0	0.034
S_{14}	Blowfish	0.36	37.5	0	0.027
S_{15}	RC4	0.14	96.43	0	0.010

TABLE VII
AIRCRAFT CONTROLLER: INTEGRITY SERVICES

Group 1 Services	Cryptographic Algorithm	Q level	Overhead (kb/ms)	I	U
S_{21}	TIGGER	1	4.36	0	0.229
S_{22}	SHA-1	0.63	6.88	0	0.145
S_{23}	RIPEMD	0.36	12.0	0	0.083
S_{24}	MD5	0.26	17.09	0	0.058
S_{25}	MD4	0.18	23.9	0	0.041

TABLE VIII
AIRCRAFT CONTROLLER: AUTHENTICATION SERVICES

Group 1 Services	Cryptographic Algorithm	Q level	Overhead (kb/ms)	I	U
S_{31}	CBC-MAC-AES	1	163	163	0
S_{32}	HMAC-SHA-1	0.91	148	148	0
S_{33}	HMAC-MD5	0.55	90	90	0

and the minimal group security value (MinGQ) for each group is 0.1.

Table IX shows that the optimal solution can produce the best solution (the combined quality value $\mathrm{CQ} = 0.72$) which is feasible. Maximum policy cannot find a schedulable solution. Minimum policy finds a feasible solution but the combined quality value is very low. If the MinGQ of each group is raised to 0.3, the solution found by the Minimum policy becomes infeasible due to the MinGQ constraint. Random policy sometimes can find a feasible and good solution (such as Rnd2) and sometimes cannot find a feasible solution (Rnd1, for example).

To cope with the security scheme, we need to implement the security mechanisms as program modules and add code to the tasks to invoke the pre-selected security mechanisms in the flight control system. Properly designing the APIs for the security mechanism is therefore important as the APIs serve as

TABLE IX
FOUR SECURITY SELECTION POLICY

	Selected	Satisfy MinGQ?	Schedulable?	CQ
Min	S_{15} , S_{25}, S_{33}	Y	Y	0.23
Max	S_{11} , S_{21}, S_{31}	Y	N	0
Rnd1	S_{11} , S_{21}, S_{33}	Y	N	0
Rnd2	S_{13} , S_{22},S_{33}	Y	Y	0.53
Opm	S_{11} , S_{23}, S_{33}	Y	Y	0.72

the bridge between the tasks in the application and the security mechanism implementation in the underlying systems. For complicated distributed systems, it is worth while investigating the security middleware system to cope with the complexities and specialties of diverse security mechanisms as shown in [33] as the developer might find that the low level security service APIs are complicated to use. With the security middleware which defines high level API for accessing security service and translates the high level service calls to low level security mechanism calls, the users (or the tasks) only need to access high level APIs for managing and configuring multiple security services. As the schedulability test and the optimization of security selection are performed offline, no support is needed for the underlying system for dynamically selecting the security services.

VII. CONCLUSION AND FUTURE WORKS

This paper presents EDF schedulability driven security optimization in real-time systems. A group-based security model is used where the services are partitioned into groups. Services in the same security group provide the same type of security service but of different quality due to the different mechanism used. Services from different groups can be combined to achieve better security. Applying security services to a system incurs overhead to the system. The overhead model of a security service has also been described. A security aware EDF schedulability test has been developed which takes the density of security service into account. Two approaches have been proposed to select the best combination of security services for real-time systems while guaranteeing their schedulability. One is integer linear programming technique and the other is heuristic search technique. The search technique is efficient since pruning and back jumping are used to reduce the search space. Evaluation has been conducted to show the effectiveness of the security optimization techniques. To the best of our knowledge, this is the first work on static security aware schedulability analysis and static security optimization in real-time systems.

Extensive test suites have been performed to compare the two metrics: the combined security value and the schedulability value, using our method and the following three policies (Minimum, Maximum, and Random) to select the security levels for tasks. Experimental results show that the combined security values obtained by our method are substantially higher than those achieved by alternatives for real-time tasks without violating real-time constraints.

Currently, the weight for security group is the same for all the tasks. In fact, different tasks may require different degree of the various types of security services. A task may require more authentication services than confidentiality service and the other task may require more on data integrity than authentication. Further improvement can be made for computing the overhead of each security variant applying to a task. For example, instead of using two parameters (U and I), functions can be defined for each security overhead computation.

The work will also be extended to consider other scheduling policies such as fixed priority scheduling policy in real-time systems.

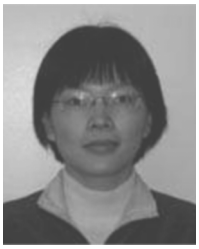
ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their insightful suggestions and comments for improving the quality of this paper.

REFERENCES

[1] *lp_Solve*. [Online]. Available: http://lpsolve.sourceforge.net/.
[2] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin, "QoS negotiation in realtime systems and its application to automated flight control," in *Proc. IEEE Real-Time Technology and Applications Symp.*, 1997, pp. 228–238.
[3] H. Aydin, R. Melhem, D. Moose, and P. Mejia-Alvarez, "Power-aware scheduling for periodic real-time tasks," *IEEE Trans. Comput.*, vol. 53, no. 5, pp. 584–600, May 2004.
[4] A. Burns, "Scheduling hard real-time systems: A review," *Softw. Eng. J.*, vol. 6, pp. 116–128, 1991.
[5] J. J. Chen, H. R. Hsu, and T. W. Kwo, "Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems," in *Proc. IEEE Real Time Technology and Applications Symp.*, Washington, DC, 2006, pp. 408–417.
[6] T. Dean, J. Allen, and Y. Aloimonos, *Artificial Intelligence*. Reading, MA: Addison-Wesley, 1995.
[7] D. P. Eames and J. Moffett, "The integration of safety and security requirements," in *Proc. 18th Int. Conf. Computer Safety, Reliability and Security*, London, U.K., 1999, pp. 468–480.
[8] J. Engblom, A. Ermedahl, M. Sjödin, J. Gustavsson, and H. Hansson, "Towards industry strength worst-case execution time analysis," in *Proc. SNART'99*, Linköping, Sweden, 1999, Linköping Univ..
[9] W. E. Faller and S. J. Schreck, "Real-time prediction of unsteady aerodynamics: Application for aircraft control and manoeuvrability enhancement," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1461–1468, Nov. 1995.
[10] W. A. Halang and A. D. Stoyenko, *Constructing Predictable Real Time Systems*. Norwell, MA: Kluwer, 1991.
[11] C. Irvine and T. Levin, "Quality of security service," in *Proc. 2000 Workshop New Security Paradigms*, Cork, Ireland, 2001, pp. 91–99.
[12] C. Irvine and T. Levin, "Towards a taxonomy and costing method for security services," in *Proc. 15th Annu. Computer Security Applications*, 2007.
[13] C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World*. Upper Saddle River, NJ: Prentice-Hall, 2002.
[14] M. Lin and L. T. Yang, "Schedulability driven security optimization in real-time systems," in *Proc. 1st Int. Conf. Availability, Reliability and Security (ARES 2006)*, Vienna, Austria, Apr. 2006, pp. 314–320.
[15] J. W. S. Liu, *Real-Time Systems*. Englewood Cliffs, NJ: Prentice-Hall, 2000.
[16] J. W.-S. Liu, K.-J. Lin, W.-K. Shih, A. C.-S. Yu, C. Chung, J. Yao, and W. Zhao, "Algorithms for scheduling imprecise computations," *IEEE Comput.*, vol. 24, no. 5, pp. 58–68, May 1991.
[17] B. Mahafza, S. Welstead, D. Champagne, R. Manadhar, T. Worthington, and S. Campbell, "Real-time radar signal simulation for the ground based radar for national missile defense," in *Proc. 1998 IEEE Radar Conf.*, Dallas, TX, May 1998, pp. 62–67.
[18] J. Nilsson and F. Dahlgren, "Improving performance of load-store sequences for transaction processing workloads on multiprocessors," in *Proc. Int. Conf. Parallel Processing*, Washington, DC, Sep. 1999, pp. 246–255.
[19] C. D. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[20] C. Park, "Predicting program execution time by analyzing static and dynamic program paths," *J. Real-Time Syst.*, vol. 5, pp. 31–62, 1993.
[21] P. Puschner and C. Koza, "Calculating the maximum execution time of real-time programs," *J. Real-Time Syst.*, vol. 1, pp. 159–176, 1989.
[22] Q. Quan and X. S. Hu, "Energy efficient DVS schedule for fixed-priority real-time systems," *ACM Trans. Embed. Comput. Syst.*, vol. 6, no. 4, 2007, Article no. 29.

[23] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A resource allocation model for QoS management," in *Proc. IEEE Real-Time Systems Symp.*, San Francisco, CA, Dec. 1997, pp. 298–307.
[24] C. A. Rusu, R. Melhem, and D. Mosse, "Maximizing the system value while satisfying time and energy constraints," *IBM J. Res. Develop.*, vol. 47, no. 5–6, pp. 689–702, 2003.
[25] S. H. Son, R. Mukkamala, and R. David, "Integrating security and real-time requirements using covert channel capacity," *IEEE Trans. Knowl. Data Eng.*, vol. 12, no. 6, pp. 865–879, Dec. 2000.
[26] S. H. Son, R. Zimmerman, and J. Hansson, "An adaptable security manager for real-time transactions," in *Proc. 12th Euromicro Conf. Real-Time Systems*, Stockholm, Sweden, Aug. 2000, pp. 63–70.
[27] S. Son and B. Thuraisingham, "Towards a multilevel secure database management system for real-time applications," in *Proc. IEEE Workshop Real-Time Applications*, New York, May 1993, pp. 131–135.
[28] J. A. Stankovic, "Misconceptions about real-time computing," *IEEE Computer*, vol. 21, no. 10, pp. 10–19, Oct. 1988.
[29] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. Buttazzo, *Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms*, ser. The Springer International Series in Engineering and Computer Science. Norwell, MA: Kluwer, 1998.
[30] S. Suzuki, T. Katane, H. Saotome, and O. Saito, "Electric power-generating system using magnetic coupling for deeply implanted medical electronic devices," *IEEE Trans. Magn.*, vol. 38, no. 5, pt. 1, pp. 3006–3008, Sep. 2002.
[31] T. Xie and X. Qin, "A new allocation scheme for parallel applications with deadline and security constraints on clusters," in *Proc. 7th IEEE Int. Conf. Cluster Computing (Cluster 2005)*, Boston, MA, 2005, pp. 1–10.
[32] T. Xie and X. Qin, "Scheduling security-critical real-time applications on clusters," *IEEE Trans. Comput.*, vol. 55, no. 7, pp. 864–879, Jul. 2006.
[33] T. Xie and X. Qin, "Security middleware model for real-time applications on grids," *IEICE Tran. Inf. Syst. Archive*, vol. E89-D, pp. 631–638, 2006.
[34] T. Xie, X. Qin, and M. Lin, "Open issues and challenges in security-aware real-time scheduling for distributed systems," *Inf. J.*, vol. 9, no. 2, pp. 215–230, 2006.
[35] T. Xie, X. Qin, and A. Sung, "SAREC: A security-aware scheduling strategy for real-time applications on clusters," in *Proc. 34th Int. Conf. Parallel Processing*, Oslo, Norway, 2005, pp. 5–12.
[36] D. Zhu and H. Aydin, "Reliability-aware energy management for periodic real-time tasks," in *Proc. IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS'07)*, Bellevue, WA, 2007, pp. 225–235.



Man Lin (M'08) received the B.E. degree in computer science and technology from Tsinghua University, Beijing, China, in 1994. She received the Lic. and Ph.D. degrees from the Department of Computer Science and Information at Linköping University, Linköping, Sweden, in 1997 and 2000, respectively.

She is currently an Associate Professor of computer science at St. Francis Xavier University, Antigonish, NS, Canada. Her research interests include real-time and embedded system design and analysis, scheduling, power aware computing, and optimization algorithms. Her research is supported by National Sciences and Engineering Research Council, Canada (NSERC) and Canada Foundation for Innovation (CFI).



Li Xu (M'07) received the B.Sc. degree in computer science with first class honors and the M.Sc. degree in computer science from St. Francis Xavier University, Antigonish, NS, Canada, in 2005 and 2008, respectively.

He is currently a system administrator at Atlantic Computational Excellence Network (ACEnet).



Laurence T. Yang (M'97) is with the Department of Math and Computer Science, St. Francis Xavier University, Antigonish, NS, Canada. His research includes high-performance computing and networking, embedded systems, and ubiquitous/pervasive computing and intelligence. His research is supported by National Sciences and Engineering Research Council, Canada (NSERC) and Canada Foundation for Innovation (CFI).



Xiao Qin (S'99–M'04) received the B.S. and M.S. degrees in computer science from Huazhong University of Science and Technology, Wuhan, China, in 1996 and 1999, respectively, and the Ph.D. degree in computer science from the University of Nebraska-Lincoln in 2004.

Currently, he is an Assistant Professor of computer science at Auburn University, Auburn, AL. Prior to joining Auburn University in 2007, he had been with New Mexico Institute of Mining and Technology for three years. His research interests include parallel and distributed systems, real-time computing, storage systems, and fault tolerance.

Dr. Qin received an NSF Computing Processes & Artifacts (CPA) Award and an NSF Computer Systems Research (CSR) Award in 2007. He has been on the program committees of various international conferences, including IEEE Cluster, IEEE IPCCC, and ICPP. He had served as a subject area editor of IEEE Distributed System Online (2000–2001).



Nenggan Zheng received the B.Sc. degree in biomedical engineering from Zhejiang University, Hangzhou, China, in 2002. Currently, he is pursuing the Ph.D. degree at the College of Computer Science and Technology, Zhejiang University.

His research interests include real-time systems and wearable computing.



Zhaohui Wu (SM'05) received the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 1993.

From 1991 to 1993, he was a joint Ph.D. student in the area of knowledge representation and expert system with the German Research Center for Artificial Intelligence (DFKI). He is currently a Professor of computer science with Zhejiang University and the Director of the Institute of Computer System and Architecture. His major research interests include intelligent transport systems, distributed artificial intelligence, semantic grid, and ubiquitous embedded systems. He has authored four books and more than 100 refereed papers.

Dr. Wu is a standing council member of China Computer Federation (CCF). Since June 2005, he has been the vice chair of the CCF Pervasive Computing Committee. He is on the editorial boards of several journals and has served as a PC member for various international conferences.



Meikang Qiu (SM'07) received the B.E. and M.E. degrees from Shanghai Jiao Tong University, Shanghai, China, and the M.S. and Ph.D. degrees in computer science from University of Texas at Dallas.

He is an Assistant Professor of electrical and computer engineering at the University of New Orleans, New Orleans, LA. His research interests include embedded systems, computer and network security, and wireless sensor networks.

Dr. Qiu has been an editor for several journals, a chair, and a TPC member for many international conferences, such as IEEE CSE 2008, IEEE SEC 2008, 2009, IEEE ESO 2008, and IEEE GlobeCom 2008.