Design Optimization of Energy- and Security-Critical Distributed Real-Time Embedded Systems

Xia Zhang, Jinyu Zhan, Wei Jiang*, Yue Ma
School of Information and Software Engineering
University of Electronic Science and Technology of China
Email: zhangxia19870317@gmail.com,
{zhanjy, wejiang}@uestc.edu.cn, yue_ma_880131@hotmail.com
*Corresponding Author

Ke Jiang
Department of Computer and Information Science
Linköping University
Email: ke.jiang@liu.se

***Abstract*—In this paper, we approach the design of energy- and security-critical distributed real-time embedded systems from the early mapping and scheduling phases. Modern Distributed Embedded Systems (DESs) are common to be connected to external networks, which is beneficial for various purposes, but also opens up the gate for potential security attacks. However, security protections in DESs result in significant time and energy overhead. In this work, we focus on the problem of providing the best confidentiality protection of internal communication in DESs under time and energy constraints. The complexity of finding the optimal solution grows exponentially as problem size grows. Therefore, we propose an efficient genetic algorithm based heuristic for solving the problem. Extensive experiments demonstrate the efficiency of the proposed technique.**

***Keywords*—Distributed Real-time System; Security; Energy; Mapping; Scheduling; System Design.**

I. Introduction

Nowadays, more and more embedded real-time systems take advantage of distributed computing to achieve better performance such as higher throughput and better reliability. In addition, the adoption of new network interfaces in modern Distributed Embedded Systems (DESs) strengthens the system functionality and flexibility. Although beneficial in many cases, the interaction with outside world also opens up unsecure access to the systems [1], leading to higher possibility of exhibiting the internal communication. So how to securely transfer sensitive information and control messages over the internal communication infrastructure becomes an emerging problem.

The DESs which must be robust against security attacks are referred to as Security-Critical Distributed Real-time Embedded Systems (SCDRES). In this paper, we focus on protecting the confidentiality, the central factor of information security, of the internal communication. Then, cryptography is the promising method for undertaking the protection [2]. However, introduction of cryptographic protection leads to significant time overhead, that may further leads to deadline violations. In real-time system, deadline misses result performance degradations or even serious catastrophe. For example, deadline misses of the engine control task in unmanned aerial vehicles may cause crashes [3]. Moreover, employment of security protections also strains energy supply. Hence, providing sound security protections under stringent real-time and energy constraints becomes a challenge.

Many algorithms have been proposed for handling scheduling problems existed in distributed real-time systems. The authors in [4] proposed a distributed approach to handle the scheduling problem of distributed system through local deadline assignments. In [5], researchers presented a Tabu Search based algorithm for designing mixed-critical applications on distributed and cost-constrained architecture. Unfortunately, all of these techniques ignored security and energy requirements.

More recently, researchers have made progresses in security-aware task scheduling for distributed real-time systems. The authors of [6] presented a heuristic approach to search for the best system-affordable cryptographic protection for the messages transmitted over the internal communication bus. Two resource allocation schemes for scheduling parallel applications on clusters with timing and security constrains are proposed in [7]. However, both works ignored the effect of communication volume on system security, and the energy constraint existed in most SCDRES. In [8], the authors presented a series of mathematical models to capture the relationships between power consumption and security. Besides, they formulated and solved a security maximization problem under real-time and energy constraints using dynamic programming. In [9] and [10], the authors proposed an energy efficient adaptive risk control and security management mechanism for centralized distributed systems and a resource allocation technique for optimizing the security protection in energy constrained mono-processor systems, respectively.

In this paper, we focus on security optimization problem of parallel applications in homogenous SCDRES with deadline and energy constraints. We find that communication traffic reduction and security reinforcement are the two solutions to the design problem. Traffic reduction can be achieved by better task-to-processor mapping. In order to improve security protection, security service strengths are quantified, and corresponding time and energy consumption are derived. Then the problem is formulated, and related constraints are addressed. Due to the complexity of the problem, a Genetic Algorithm (GA) based heuristic, Energy and Security-aware Schedule Algorithm for Communication-sensitive Applications (ESACA), is proposed.

The rest of the paper is organized as follows. In Section II, we present our system model. The design optimization problem is formulated in Section III. Section IV illustrates a motivating example. We describe our security optimization algorithm in Section V. Simulations and experiments are

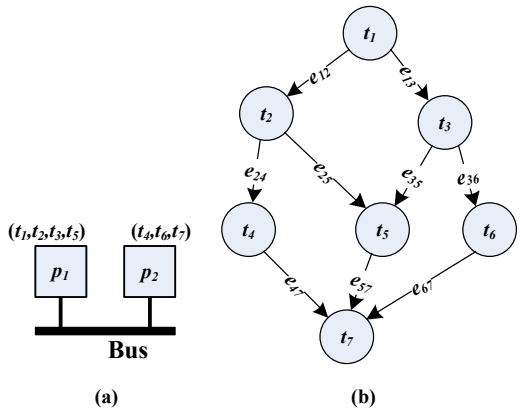

Fig. 1. A hardware architecture and a simple application example

conducted and analyzed in Section VI. Finally, we conclude the paper in Section VII.

II. SYSTEM MODEL

A. System architecture

In this paper, we consider homogenous DESs, which consist of multiple processors for tasks execution and a common bus for data transmission. For the sake of simplicity, we make the following assumptions:

- All the processors are homogeneous. That is, the processing capacity and power consumption are the same.
- Available energy of each processor is limited, namely, the system must work under strict energy constraint.
- Messages transmitted over the bus should be encrypted before sending and decrypted after reception.
- We assume that the hardware is temper-proof. Therefore, communication within a processor is secure, and then can be ignored.
- Only one message can access to the bus at a time.

The set of processors in the system is denoted as $P = \{p_1, p_2, \ldots, p_\rho\}$. All the processors are connected to a bus. An example of two nodes hardware platform is depicted in Fig. 1(a).

There are one or more applications running in the system. An application, modeled as a directed acyclic graph (DAG), is composed of several interdependent tasks. An example of 7 tasks is revealed in Fig. 1(b). Each node in DAG represents a task. Each edge denotes a data dependency, referred to as a message, from tail to head. A task can only be scheduled when all the messages from its preceding tasks are received.

An application is formally represented as $G(T, E, M, W)$. $t_i \in T$ is a task having a designer specified WCET τ_i. E is a $n \times n$ 0-1 matrix. $e_{ij} \in E$ denotes the message from t_i to t_j. If there is a message transmitted from t_i to t_j, then $e_{ij} = 1$, otherwise $e_{ij} = 0$. M and W are both $n \times n$ matrixes. $m_{ij} \in M$ and $w_{ij} \in W$ are the size and weight of message e_{ij}, respectively. $m_{ij} = w_{ij} = 0$, if $e_{ij} = 0$. A higher value of w_{ij} implies a higher security requirement of e_{ij}. All the tasks in an application have a common deadline D which is not allowed to be violated.

All tasks of an application will be mapped to processors, i.e. nodes in Fig. 1(a). 0-1 matrix A stands for the mapping of tasks to processors. The size of the matrix is $\rho \times n$. For $a_{ij} \in A$, if t_j is mapped to p_i, then $a_{ij} = 1$, otherwise $a_{ij} = 0$. If two tasks which communicate with each other are assigned to different processors, their communication information will be transferred over the bus. Otherwise, the communication actions will be accomplished inside the processor. The transmission delay in processors can be ignored, but the communication delay on a bus cannot be neglected. The bus transmitting time χ_{ij} of message e_{ij} can be formulated as:

$$\chi_{ij} = \frac{m_{ij}}{B} \tag{1}$$

where χ_{ij} denotes the transmission delay between t_i to t_j. B is the bandwidth of the bus.

Meanwhile, we define a 0-1 matrix Π sized $n \times n$ to represents the bus communication. $\pi_{ij} = 1$ means that the message from t_i to t_j is transmitted over the bus. $\pi_{ij} = 0$ implies that message from t_i to t_j never appears on the bus, no matter whether the communication actually exists or accomplished inside the processors. The value of Π can be derived if A and E are specified by the designer. And the equation can be formulated as follow:

$$\Pi = \Psi(A, E), \quad \Psi : \pi_{ij} = e_{ij}(\sum_{k=1}^{\rho} \pi_{ij\_k})/2 \tag{2}$$

$$\pi_{ij\_k} = (a_{ki} + a_{kj}) mod 2 \tag{3}$$

$$s.t.\ 1 \leq i, j \leq n \tag{4}$$

For each $e_{ij} = 1$, if the sending task and receiving task are both assigned to p_k, the message would be internal processor communication($\pi_{ij\_k} = 0$). If only one of the two tasks is assigned to p_k, the message would be transmitted over the bus ($\pi_{ij\_k} = 1$). If the two tasks are assigned to other processors, we have π_{ij_k}=0, which can be obtained in Eq. (3). To obtain π_{ij}, all of π_{ij_k} ($k = 0, 1, \ldots, \rho$) must be investigated, as Eq. (2).

TABLE I. PLAINTEXTS REQUIREMENTS TO ATTACK RC6

Number of Rounds	8	12	16	20	24
Required Plaintexts	2^{47}	2^{83}	2^{119}	2^{155}	2^{191}

B. Models of message security requirements

As already discussed, encryption must be applied in SC-DRESs for security protections. In this paper, we assume the use of RC6, which is a fully parameterized symmetric cypher, and one of the important members in iterated block cypher (IBC) family. IBC, that is arguably the most widely used cryptography, encrypt a plaintext block using several iterations. In each iteration, the same transformation is applied to the data using a sub key. The number of iterations of a selected IBC is one determinant of its security performance and resource consumption. Then, there are two problems that we must pay attention to: how to evaluate the security performance of different rounds and how do the rounds affect energy consumption?

Lots of cryptanalysis have been conducted since the publication of RC6. Rivest et. al [11] have done some work to analyze the security performance of RC6 against cryptanalytic

attacks, such as linear and differential cryptanalytic attacks. Cryptanalytic attacks require a large number of plaintexts, the number of which is regarded as an indication of its security strength. So, more required plaintexts mean more secure. Required amount of plaintexts grows as encryption rounds increase. For linear cryptanalysis, the required plaintexts are less than that of differential cryptanalysis. This implies that linear cryptanalysis is more effective at breaking RC6. The number of rounds and required plaintexts by a successful linear cryptanalytic attack are presented in Table 1.

The correlation between plaintext amount and number of rounds is formulated as:

$$\log_2 N_{plaintext} = \alpha l \tag{5}$$

where α, $N_{plaintext}$, and l are a constant, the number of required plaintexts, and encryption rounds, respectively.

A simple approach to improve security protection is to maximize the total number of encryption rounds of all messages transmitted over the bus. However, this approach is not suitable for our problem, because communication reduction and rounds maximization are two contradictory requirements. Therefore, we take the Degree of Security Gap (DSG) r of a message as the security performance metric formulated as follows:

$$r = \frac{l^{max} - l}{l^{max}} \tag{6}$$

l^{max} is the maximum rounds of RC6 required by the system, and l is the actual rounds used to encrypt and decrypt the message. As can be noticed, the value of r is bounded between 0 and 1. Based on Eq. (5) into (6), we can obtain:

$$r = \frac{l^{max} - (\log_2 N_{plaintext})/\alpha}{l^{max}} = 1 - \xi \cdot \log_2 N_{plaintext} \tag{7}$$

$$\xi = \frac{1}{l^{max}\alpha} \tag{8}$$

We can see that ξ becomes a constant. Eq. (7)indicates that DSG decreases as plaintext amount increases. Therefore, to obtain high security, r should be minimized.

Then an $n \times n$ matrix R is introduced to describe all the message protections. $r_{ij} \in R$ is 0 if there is no message from t_i to t_j. Otherwise, r_{ij} can be calculated using Eq. (7). Similarly, an $n \times n$ matrix L is introduced to describe the rounds assignments to messages. $l_{ij} \in L$ denotes the encryption rounds of the message from t_i to t_j. If there is no message from t_i to t_j, then $l_{ij} = 0$. Given L and E, R can be derived as:

$$R = \frac{l^{max}E - L}{l^{max}} = E - \frac{1}{l^{max}}L \tag{9}$$

Note that R and L are for all messages in an application, no matter whether they are bus communications or not.

C. Time overhead of security protection

With the increase of encryption rounds, execution time grows. RC6 consists of three procedures: key schedule, encryption, and decryption [12]. The execution times that are important to us are the times consumed by the encryption and decryption procedures. The encryption and decryption procedures take roughly the same amount of time. So for the sake of simplicity, we assume that they share the same WCET. Similar to the Eq. [6], the WCET c_U of encrypting/decrypting a block are

$$c_U = c_I + l \cdot c_L \tag{10}$$

where c_I is the WCET of the initial procedure. c_L denotes the WCET for one round encryption/decryption operation. Then time consumption when encrypting/decrypting a block can be reveal by an $n \times n$ matrix C^U, which is defined as:

$$C^U = c_I \cdot E + c_L \cdot L \tag{11}$$

$c^U_{ij} \in C^U$ denotes the WCET of encrypting/decrypting one block of message e_{ij}.

D. Energy Consumption

Energy consumption is an important performance indication of embedded systems. When executing tasks, processors run at a full speed with maximal power. And they run at a lower speed with less power in idle time to reduce energy consumption. We assume v_r stands for the power of processors at run time, while v_s denotes that at idle time. They should satisfy the constraint $v_r > v_s$. A processors energy consumption γ is formulated as follow:

$$\gamma = \tau_r v_r + \tau_s v_s \tag{12}$$

where τ_r and τ_s are the running time duration consisted of task execution time and encryption/decryption time and the idle time duration, respectively. Assuming that the application starts at time 0, τ_r and τ_s should satisfy:

$$\tau_r + \tau_s = D \tag{13}$$

Based on Eq. (12) and (13), we can get:

$$\begin{aligned} \tau_r v_s + \tau_s v_s \le \tau_r v_r + \tau_s v_s \le \tau_r v_r + \tau_s v_r \\ \Rightarrow D \cdot v_s \le \gamma \le D \cdot v_r \end{aligned} \tag{14}$$

If energy constraint is not introduced, then energy consumption has a maximum and a minimum value as seen in Eq. (14). However, in embedded systems, total available energy is usually very limited. So it is necessary to employ an energy up-boundary γ_b on each processor. And γ_b must satisfy the constraint $\gamma_b \ge D \cdot v_s$. If γ_b is lower than $D \cdot v_s$, it is impossible to get a valid schedule satisfying all constraints. When γ_b is higher than $D \cdot v_r$, energy limit can be ignored because every schedule scheme fulfilling the deadline requirement can satisfy the energy constraint.

III. PROBLEM FORMULATION

A. Problem Identifying

In this section, we formally formulate our design optimization problem. Before going further, we first introduce the DAG reconstruction method [6] that we used. In order to make the system more explicit for analyzing and more flexible for scheduling, we abstract the encryption and decryption explicitly from their corresponding sending and receiving tasks. The abstracted encryption and decryption processes should be mapped to the same processors as their parents. Fig. 2(a) is an illustrative DAG having two tasks (t_1 and t_2) and a message (e_{12}). The two tasks are mapped to different processors. So encryption (ec_{12}) and decryption (ed_{12}) processes must be performed to protect the message (e_{12}). The reconstructed

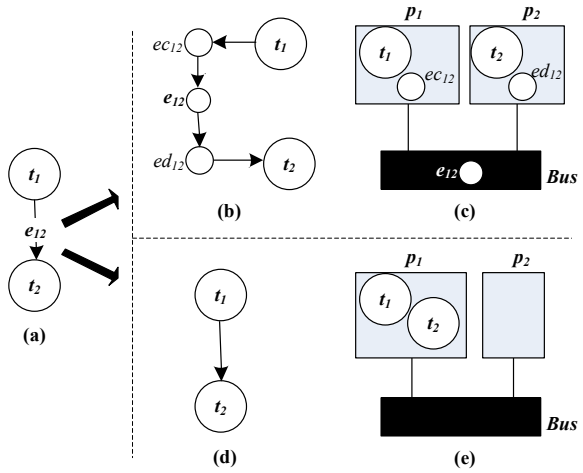

Fig. 2. DAG reconstruction under two mapping cases

DAG is presented in Fig. 2(b). The bus is seen as a processor, and undertakes the messages, as shown in Fig. 2(c). If t_1 and t_2 can be mapped to the same processor, message e_{12} is ignored, and encryption and decryption are not launched. So the DAG will be transform into Fig. 2(d) with corresponding mapping in Fig. 2(e). The reconstruction procedure can be expanded to multiple applications.

From above analysis, we find that communication would result in more resource overhead. And different task mapping leads to different communication volume, namely different overhead. Therefore, a mapping with less communication could save more resource for security reinforcement. So task mapping and message encryption/decryption are the means for pursuing high security protection.

DSG of an Application (DSGA) is defined as the sum of DSG of all the messages transmitted over the bus. And lower DSGA implies higher security performance. Both communication volume reduction and security reinforcement would have the value of DSGA diminished. Therefore, the objective can be formulated as minimizing DSGA as much as possible by obtaining a good task mapping and a sound encryption strategy (rounds assignment). At the same time, several constraints such as dependency constraint, real-time constraint, and energy constraint must be considered.

B. Objective

Our design objective of DSGA Γ is formulated as follows

$$\Gamma = \eta^T \Theta \eta \tag{15}$$

η is a vector sized n whose elements are all 1. Θ is a $n \times n$ matrix having the message protections. $\theta_{ij} \in \Theta$ indicates DSG of e_{ij}. If there is no communication from t_i to t_j, or the communication are accomplished inside a processor, $\theta_{ij} = 0$. The function Θ is presented as follows:

$$\Theta = f_{multiply}{}^1(W, R, \Pi) \tag{16}$$

[1]Assuming there are $n+1$ matrixes which are B_1, B_2, , B_n, H, whose elements are presented as b_{1ij},b_{2ij},,b_{nij}, h_{ij}. Then $f_{multiply}$ is defined as:

$$H = f_{multiply}(B_1, B_2, \ldots, B_n), \quad f_{multiply} : h_{ij} = b_{1ij} \cdot b_{2ij} \cdot \ldots \cdot b_{nij}$$

Π, which has been mentioned in the previous section, is used for removing the internal processor communication. Then based on Eq. (2), (9) and (16), (15) can be transformed as:

$$\begin{aligned} \Gamma &= \eta^T f_{multiply}(W, R, \Pi)\eta \\ &= \eta^T f_{multiply}(W, E - \frac{1}{l^{max}} L, \Psi(A, E))\eta \end{aligned} \tag{17}$$

All the variables except L and A are already known in advance. Thus, DSGA depends on the value of the two variables. We combine L and A to a two-tuple solution $s(A, L)$. In the paper, our objective is to find the optimal solution s^* that achieves the minimal Γ^*.

C. Constraints

Given a solution, the application can be scheduled by List Scheduling (LS) [13] approach, in which the dependency constraint is inherently guaranteed. Using LS, we can obtain schedule length and energy consumption easily, and then check whether the constraints are satisfied or not. The encryption/decryption time overhead are calculated by

$$\begin{aligned} C &= f_{multiply}(M, C^U, \Pi) \\ &= f_{multiply}(M, c_I E + c_L L, \Psi(A, E)) \end{aligned} \tag{18}$$

C is an $n \times n$ matrix which represents the encryption time overhead. $c_{ij} \in C$ is the encryption time of message e_{ij}. If e_{ij} does not exist or only exists inside a processor, then $c_{ij} = 0$. As already discussed, the decryption time is equal to encryption time. Therefore, C also indicates the decryption time overhead.

In LS approach, multiple messages will be transmitted over the bus, and conflictions have to be addressed. By using the DAG reconstruction approach mentioned before, we can transform an application into a simple DAG whose edges just represent the dependencies. Besides, in the LS approach, bus is seen as a processor where messages are undertaken. In this way, the LS schedule objective is just to assign a set of tasks to a group of processors. And message conflicts are solved in a concise way. Based on reconstructed DAG, the dependency constraint is defined as follows.

Dependency constraints.Each task is allowed to be executed only after all its preceding tasks are finished. Then the dependency constraint is

$$\tau_{i\_s} \geq max\{\tau_{j\_s} + \tau_i | t_j \in Perants(t_i)\} \tag{19}$$

τ_{i_s} and τ_{j_s} stands for the start time of t_i and t_j, respectively. $Parents(t_i)$ denotes the set of t_i's preceding tasks. After that, LS returns the schedule length τ_f, and the deadline D needs to be satisfied.

$$\tau_f = ListSchedule(G, C, A) \tag{20}$$

Real-time constraint: all the tasks of an application should be finished before the deadline:

$$\tau_f \leq D \tag{21}$$

Available energy is limited, too, and is the same for all processors. Energy consumption can be calculated by Eq. (12). Assuming $y = \{\gamma_1, \gamma_2, \ldots, \gamma_\rho\}$ is a vector whose elements presents the energy consumption of each processor. And it can be calculated by the following function:

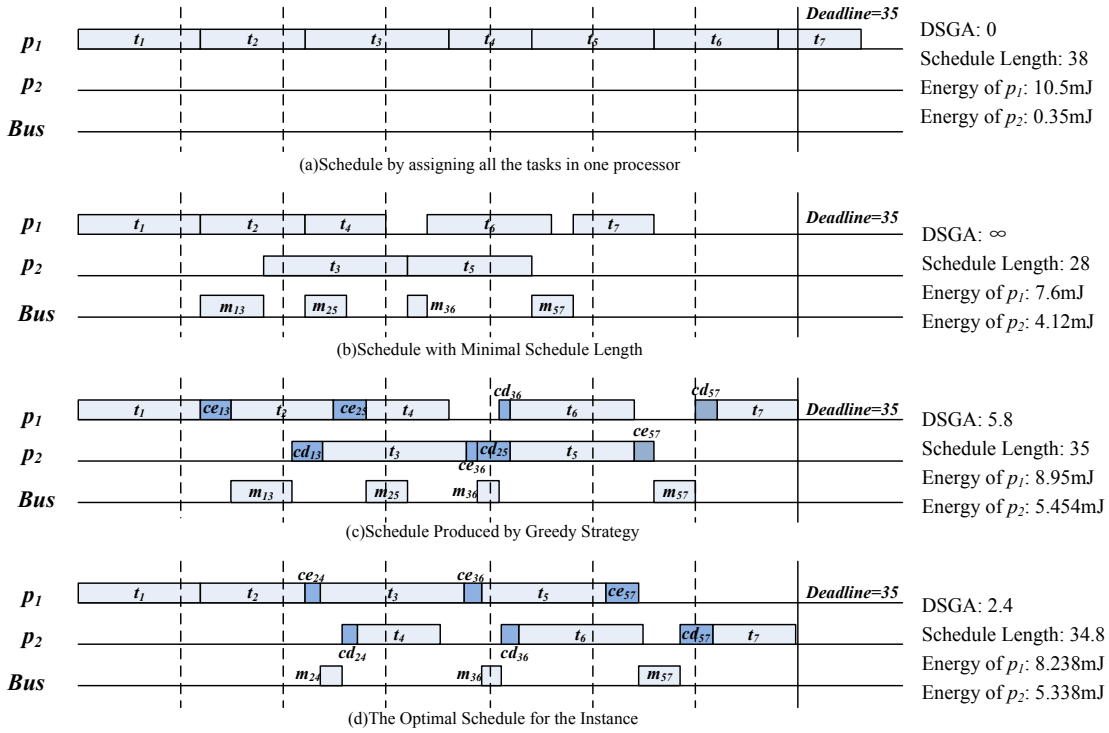

Fig. 3. Schedule examples under different schemes

$$y = \begin{bmatrix} {\tau_r}^{(1)} \\ {\tau_r}^{(2)} \\ \vdots \\ {\tau_r}^{(\rho)} \end{bmatrix} v_r + \begin{bmatrix} {\tau_s}^{(1)} \\ {\tau_s}^{(2)} \\ \vdots \\ {\tau_s}^{(\rho)} \end{bmatrix} v_s \tag{22}$$

Base on Eq. (13), following function can be obtained:

$$y = \begin{bmatrix} {\tau_r}^{(1)} \\ {\tau_r}^{(2)} \\ \vdots \\ {\tau_r}^{(\rho)} \end{bmatrix} (v_r - v_s) + D v_s \eta \tag{23}$$

For each ${\tau_r}^{(i)} (i = 1, 2, \ldots, \rho)$:

$$\begin{aligned} {\tau_r}^{(i)} &= \sum_{j=1}^{n} (\tau_j + \sum_{k=1}^{n} (c_{jk} + c_{kj})) \\ &= \begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{bmatrix}^T \begin{bmatrix} \tau_1 + \sum_{k=1}^{n} (c_{1k} + c_{k1}) \\ \tau_2 + \sum_{k=1}^{n} (c_{2k} + c_{k2}) \\ \vdots \\ \tau_n + \sum_{k=1}^{n} (c_{nk} + c_{kn}) \end{bmatrix} \\ &= \begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{bmatrix}^T (\begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{bmatrix} + C\eta + C^T \eta) \end{aligned} \tag{24}$$

Let $\Lambda = \{\tau_1, \tau_2, \ldots, \tau_n\}$ denotes WCETs of all tasks in an application.

$$\begin{bmatrix} {\tau_r}^{(1)} \\ {\tau_r}^{(2)} \\ \vdots \\ {\tau_r}^{(\rho)} \end{bmatrix} = A(\Lambda + C\eta + C^T \eta) \tag{25}$$

Therefore:

$$y = A(\Lambda + C\eta + C^T \eta)(v_r - v_s) + D v_s \eta \tag{26}$$

Having obtained energy consumption, we can know if energy constraint can be satisfied.

Energy constraint: energy on each processor cannot violate energy upper bound:

$$\forall \gamma_i \in y \quad \gamma_i \leq \gamma_b \tag{27}$$

With a further investigation in Eq. (20) and (25), it can be observed that schedule length and energy consumption depend on $s(L, A)$. And dependency constraint must be obeyed in LS procedure. Therefore, to minimize DSGA, a reasonable solution that satisfies the constraints must be obtained.

IV. MOTIVATIONAL APPLICATION

In this section, we present an illustrative example having one application. For multiple parallel applications, it can be transformed into a single application by adding two dummy tasks (entry task and exit task) [14]. Our aim is to find a reasonable solution to obtain optimal security performance with deadline and energy limitation guaranteed.

Let us consider the example having seven tasks that are

mapped to the hardware platform with two processors (p_1 and p_2) and one bus. The parameter settings are given below. Because most of the following matrixes are sparse, we only list those elements that are non-zero.

- WCET of tasks: $\Lambda = \{6, 5, 7, 4, 6, 6, 4\}$
- Size of messages: M=$\{m_{12}, m_{13}, m_{24}, m_{25}, m_{35}, m_{36}, m_{47}, m_{57}, m_{67}\}$=$\{4,3,1,2,3,1,1,2,1\}$
- Weight of messages:W=$\{w_{12}, w_{13}, w_{24}, w_{25}, w_{35}, w_{36}, w_{47}, w_{57}, w_{67}\}$=$\{1,2,1,3,1,1,1,2,2\}$
- Deadline: $\tau_d = 35$
- Energy Limitation: $\gamma_b = 9mJ$
- Bus bandwidth: $\beta = 1$
- Maximum rounds: $l^{max} = 20$

The tasks can be scheduled in several different ways, which are illustrated in Fig. 3. We assume that if there is no message transmitted on the bus, DSGA is 0. And if at least one message on the bus is not encrypted, DSGA is ∞.

To prevent confidentiality attacks, we could have all the communication within a processor. That means that all the interdependent tasks are allocated to a single processor, as seen in Fig.3(a). However, this schedule violates the deadline and energy constraints.

Fig.3(b) is another possible schedule. In this case, both deadline and energy constraints are satisfied, but security protection is not employed and these messages are transferred over the bus in plaintext, thus, vulnerable to malicious eavesdrops. If we look closer at the schedule, we can find that there exists idle time and energy budget on both processors, which can be used to increase security performance.

We can use greedy strategy to increase encryption rounds as in Fig. 3(c). In this scheme, CPU time and energy are fully used and any increase of encryption rounds will lead to constraints violation. However, if we look at the DSGA, this is still not a satisfying solution.

Fig. 3(d) is actually the optimal schedule, which gives the minimum DSGA with respect to the deadline and energy constraints. Comparing with Fig.3(c), communication traffic is reduced due to the remapping of tasks to processors. Meanwhile, encryption rounds are assigned in a different way. So, to obtain the best schedule scheme, task mapping and rounds assignment are two factors that must be considered. However, obtaining task mapping and increasing encryption round separately will have solutions fall into local optimum and may not give acceptable results. Therefore, the two parts of solutions must be optimized synthetically.

Solutions consist of task mapping and rounds assignment. There are ρ^n different task mapping and $(l^{max})^k$ different rounds assignment of an application. Here k denotes the message number of an application. Therefore, for an application and a given hardware architecture, there are $\rho^n(l^{max})^k$ different solutions that grows exponentially as n and k grow. When the problem size become bigger, it becomes impossible to obtain the optimal solution. Thus, heuristic approaches are becoming a choice for this problem.

V. PROPOSED TECHNIQUE

Due to the complexity of the problem, we propose an efficient algorithm, ESACA, for solving this problem. ESACA is a GA based heuristic that takes DAGs and hardware platform specification as input, and returns the task-to-processor mapping and rounds assignments as output.

Algorithm 1 ESACA

```
1 begin
2     initial G;
3     population initialization;
4     while stop criteria is not satisfied do
5       evolution of A;
6       evolution of L;
7     end while
8     obtain the best solution s*;
9 end
```

GA was developed based on the emulation of natural selection and evolution, and famous for its robustness and global searching ability [15]. GA is widely used in solving combinatorial optimization problems. A group of solutions, which is called population, is evolved from generation to generation for pursuing better solutions. In our algorithm, an acceleration technique [16] and elite strategy are imported to achieve quick convergence.

In ESACA, s is employed as chromosome, which also is regarded as an individual in a population. Operations on chromosomes such as crossover and mutation fall into two categories: operations on L and operations on A. According to the classification, the evolution procedure in ESACA is designed as a two-stage procedure. The main procedure of ESACA is presented in Alg. 1.

First, an application is input into the algorithm. Then, a group of initial population treated as the start point of evolution procedure is generated. Line 4 to 7 is the two-stage evolution procedure, and A and L are optimized alternatively. The best solution is obtained in line 8. In the following sections, we will discuss the procedures in more details.

A. Population Initialization

Before evolution procedures, initial population must be generated in some way. Traditionally, initial population is generated randomly. However, bad solutions can be generated and increase time overhead for finding acceptable solutions. To obtain quick convergence, an acceleration technique is introduced to obtain the initial populations with good quality.

The core of the acceleration technique is composed of two parts: a scheduler and a greedy searching procedure. Firstly, initial task mapping is produced without considering the security protection. Then, the task mapping A and the rounds assignment L_{min} with the lowest protection level are combined to form the initial solution of greedy searching procedure. A and L are randomly adjusted to explore better solutions. After a number of iterations, some solutions are kept as the initial population.

Let us assume that the population size is μ. Then the pseudo code of the population initialization procedure is shown in Alg. 2. In line 2, A is obtained by the scheduler. Initial solutions of the procedure are generated in line 3. From line 4 to line 13, initial population is generated greedily.

B. Two-stage Evolution Procedure

This procedure is the core step of ESACA. In the first stage, the population evolves on task mapping. Similarly, the population evolves on rounds assignment in the second stage.

Algorithm 2 Population Initialization

```
1  begin
2    A = Schedule(G);
3    s_initial = (A, L_min);
4    while population size not reach μ do
5        s = s_initail;
6        while stop criteria is not satisfied do
7            s' = adjust(s);
8            if Γ(s') < Γ(s), τ_f(s') < D and (∀γ ∈ y(s')) < γ_b then
9              s = s';
10           end if
11       end while
12       s is assigned to initial population;
13   end while
14end
```

Algorithm 3 Two-stage Evolution

```
1  begin
2      x = 1;
3      evaluation;
4      while stop criteria is not satisfied do
5          Φ'(x) = selection(Φ(x));
6          Φ''(x) = crossover_A(Φ'(x));
7          Φ(x + 1) = mutation_A(Φ''(x));
8          x = x + 1;
9          evaluation;
10     end while
11     obtain s_best in Φ(x);
12     for each s in Φ(x) do
13         s = (A_best, L);
14     end for
15     evaluation;
16     while stop criteria is not satisfied do
17         Φ'(x) = selection(Φ(x));
18         Φ''(x) = crossover_L(Φ'(x));
19         Φ(x + 1) = mutation_L(Φ''(x));
20         x = x + 1;
21         evaluation;
22     end while
23     obtain s_best in Φ(x);
24     for each s in Φ(x) do
25         s = (A, L_best);
26     end for
27 end
```

And the two stages are operated alternately. There are four main operations in each stage: evaluation, selection, crossover and mutation. In evaluation, fitness of every solution is calculated. Selection is a process in which high quality solutions in current population is selected for following operations, i.e. crossover and mutation. New solutions are produced from crossover and mutation. Let $\Phi(x)$ denotes the population of the x-th generation, and $s_{best} = (A_{best}, L_{best})$ indicates the best solution in a generation. Alg. 3 presents pseudocode of the procedure.

Lines 3-14 is the first stage, and line 15-26 is the second stage. At the end of each stage (lines 11-14 and lines 23-26), elite strategy is applied: A_{best} or L_{best} are obtained, and replace A or L of each solution. In this way, good task mapping and rounds assignment are reserved for following evolutions. Now we discuss the four main operations one by one.

1) Evaluation: Here we evaluate the fitness of each individual, and the fitness function is formulated as

$$f(s) = \varepsilon_1 \Gamma + \varepsilon_2 \triangle d + \varepsilon_3 \triangle y \tag{28}$$

$$\triangle d = \begin{cases} \tau_f - D & \tau_f > D \\ 0 & else \end{cases} \tag{29}$$

$$\triangle y = \sum_{i=1}^{\rho} \triangle \gamma_i, \quad s.t. \ \triangle \gamma_i = \begin{cases} \gamma_i - \gamma_b & \gamma_i > \gamma_b \\ 0 & else \end{cases} \tag{30}$$

Three criteria are employed in the function. The first one is DSGA of the solution. The next two are the deadline and energy constraints. In the system, the result is not allowed to violate any constraints. But solutions with deadline misses or energy limit violations are possible to evolve into better global solutions. So we consider these excesses of schedule length and energy consumption as penalty factors in fitness function, rather than simply discard them. Solution with smaller fitness value have stronger adaptability. ε_1, ε_2 and ε_3 are the weights of the three criteria.

2) Selection: After selection, good solutions are likely to be kept to guarantee the evolution quality. When a solution is selected, they are copied to mating pool for crossover and mutation. A random selection approach, namely roulette-wheel selection, is applied in this paper, and solution s_i's probability of being selected is:

$$p(s_i) = \frac{f(s_{max}) - f(s_i)}{\sum_{j=1}^{\mu} f(s_{max}) - f(s_j)} \tag{31}$$

Here s_{max} denotes the solution with maximal fitness in current population. When a solution is selected, they are copied to mating pool for crossover and mutation operation.

3) Crossover: Crossover combines two parents to create new individuals, which may take good characteristics of their ancestors. As already mentioned, the operation objects in the two stages are to find good task mapping and rounds assignments. Take stage 1 for instance. Firstly, a crossover point i on A is randomly selected. Then the columns behind the i-th column of two parents ($A^{(1)}$ and $A^{(2)}$) are swapped to form offspring ($A^{(3)}$ and $A^{(4)}$). It means all the tasks ranging from $i+1$ to n exchange their task mapping. The crossover operation is formulated as follows:

$$\begin{cases} A^{(3)} = A^{(1)} I_c(i) + A^{(2)} I_c'(i) \\ A^{(4)} = A^{(2)} I_c(i) + A^{(1)} I_c'(i) \end{cases}, \ s.t. \ I_c(i) + I_c'(i) = I \tag{32}$$

where, I is an unit matrix, and

$$I_c(i) = \begin{bmatrix} I_{i \times i} & 0 \\ 0 & 0 \end{bmatrix}, \quad I_c'(i) = \begin{bmatrix} 0 & 0 \\ 0 & I_{(n-i)\times(n-i)} \end{bmatrix} \tag{33}$$

The crossover operation on L is similar. The only difference is that when a crossover point i is selected, all the rows behind the i-th row of two parents are exchanged to create offspring. That is, all the tasks ranging from $i+1$ to n exchange the rounds assignment of the messages they send. This is formulated as follows:

$$\begin{cases} L^{(3)} = I_c(i) L^{(1)} + I_c'(i) L^{(2)} \\ L^{(4)} = I_c(i) L^{(2)} + I_c'(i) L^{(1)} \end{cases} \tag{34}$$

4) Mutation: Mutation operator plays an important role in GA, keeping the variety of the population and preventing results from converging to local optimums. Mutation operation will be employed on task mapping or rounds assignment of a randomly selected solution with the two parts:

- *Task mapping mutation*. A randomly selected task is relocated to another processor. Assuming i is the

mutation point, which is actually the id of the selected task. A randomly selected j is the id of the new processor that t_i is mapped to. Then mutation of A is denoted as:

$$A' = AI_m(i) + \hat{\imath}^T(j)\hat{\imath}(i) \tag{35}$$

$I_m(i)$ is a $n \times n$ diagonal matrix having element $\iota_{ii} = 0$ and ι_{11}=ι_{22}=...=$\iota_{(i-1)(i-1)}$=$\iota_{(i+1)(i+1)}$=...=$\iota_{nn} = 1$. $\hat{\imath}(i)$ and $\hat{\imath}(j)$ are unit vectors whose i-th and j-th elements are 1, respectively.

- *Rounds assignment mutation*. The encryption rounds of a message are increased or decreased. The step size $\triangle l$ of this alteration is determined in advance. Assuming a message e_{ij} is randomly selected. Then the mutation operation is denoted as:

$$L' = L \pm \triangle l \cdot \hat{\imath}^T(i)\hat{\imath}(j) \tag{36}$$

Solutions for crossover and mutation are randomly selected based on user provided rates, i.e. a crossover rate δ_c and a mutation rate δ_m.

C. Brief Summary

The whole optimization procedure terminates when the stopping condition is reached, which in this paper, is the number of optimization iterations. And the best solution is returned. Like traditional GA, ESACA is sensitive to the value of parameters, such as μ, iterations, δ_c and δ_m. Therefore, these parameters must be fine-tuned by extensive experiments to achieve the best performance.

VI. EXPERIMENTAL RESULTS

In this section, we present the experiments that we conducted for evaluating our proposed technique. All the simulations are implemented using C#, and performed on a Windows machine having a dual-core Intel Pentium CPU with 2.22GHz frequency and 2GB RAM. Three security-aware scheduling algorithms for parallel applications (TAPADS, Min-Greedy and SA-Heuristic) are also implemented and used for comparisons. TAPADS[7] is a security-aware scheduling approach that maximizes security of applications by greedily increasing tasks' or messages' security level step by step without considering communication reduction. Min-Greedy is a greedy approach as the method in Fig. 3(c). SA-Heuristic [6] is an optimization algorithm trying to balancing resource utilizations.

We carried out three groups of experiments. The execution time of ESACA, Min-Greedy and SA-Heuristic are relatively long because of their complexity. TAPADS and SA-Heuristic focus on security optimization under constraints of deadline, but do not consider energy limit. The parameters that are used in the experiments are given in Table 2. Other parameters such as task numbers and deadlines are specifically defined in each group.

A. Evaluation on Different Graph Size

In this set of experiments, applications of different sizes are generated, and corresponding deadlines are determined, as seen in Fig. 4. These applications are mapped to 2, 2, 3, 3, 4, 4, 4, 5, and 5 processors respectively. For each graph size, 5 applications are generated, and for each application, 10 test cases are conducted on the four algorithms. The available energy of all the processors is 70mJ. In the rest of the paper, we call the energy consumption of the processor which is the most among all processors as maximum energy in brief. The simulation results of DSGA, maximum energy and schedule length are presented in Fig. 4.

TABLE II. PARAMETER SETTING

Parameters	Value
Encryption Rounds	1∼20
Message Weight	1∼5
Message Size	1∼20KB
WCET of Tasks	1∼20ms
CPU Power in Running Time	300mW
CPU Power in Slack Time	10mW

From Fig.4(a), we get to know that ESACA is the best in security optimization in most cases. TAPADS could get results in the shortest time, but has the worst performance. ESACA obtains the lowest DSGA on applications sized 10, 20, 30, 40, 50 and 60. The optimization performance of ESACA is almost the same with SA-Heuristic on size 70 and 80, and worse on size 90. However, from Fig. 4(b), we can find that, from application size 50, 60, 70, 80 and 90, energy consumption of TAPADS and SA-Heuristic on a single processor is already larger than the energy bound, as they do not consider energy during optimization. When application sizes are relatively small, energy consumption is relatively low, hence can be ignored. With the increase of application sizes, the energy constraint impacts more. Since ESACA and Min-Greedy must consider the deadline and energy constraints at the same time, the result qualities of ESACA and Min-Greedy are weakened when application size rises.

TAPADS and Min-Greedy are greedy approaches, and have high probability of falling into local optimums. SA-Heuristic and ESACA are both based on randomness, and have global searching ability. But SA-Heuristic does not consider energy limit. Therefore, compositing security and energy aspects, ESACA still obtains higher performance compared to the other algorithms.

It can be observed from Fig. 4(c) that all the algorithms meet the deadline requirements. On applications sized 70-90, schedule length of SA-Heuristic and ESACA are smaller than deadline, while that of TAPADS and Min-Greedy are almost near deadline. Thats because when application size rise, the search space is enlarged. Its possible for random algorithms such as SA and GA to obtain sub-optimal solution and utilize resources insufficiently.

B. Evaluation of Energy Limit Impacts

In this set of experiments, application size and deadline are fixed as 30 and 160, respectively. The energy limit is ranged from 35mJ to 55mJ. The results are shown in Fig. 5.

We find that ESACA and Min-Greedy can always fulfill the energy requirement, while TAPADS and SA-Heuristics energy consumptions violate the constraints when the energy limit is lower than 45mJ. As seen in Fig. 5(a), ESACA's DSGA trends down when energy constraint is higher. And so does Min-Greedy, although not obvious. That's because more energy budget means more resource for security reinforcement. ESACA and SA-Heuristics security performances are much

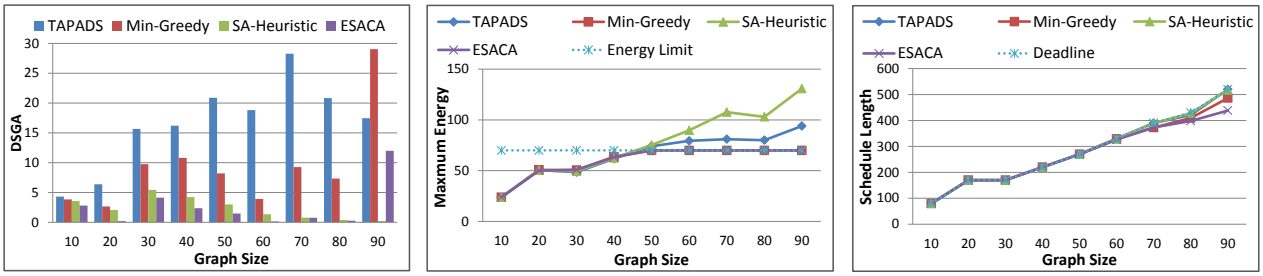

Fig. 4. Performance Corresponding to Different Graph Size

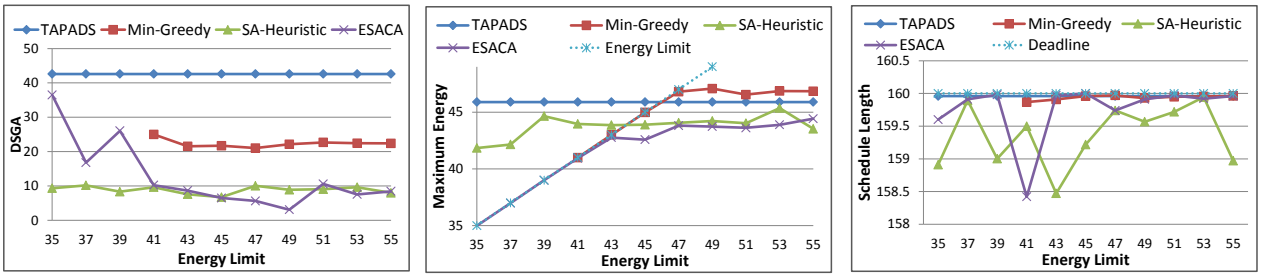

Fig. 5. Impact of Energy Limit on Algorithms' Performance

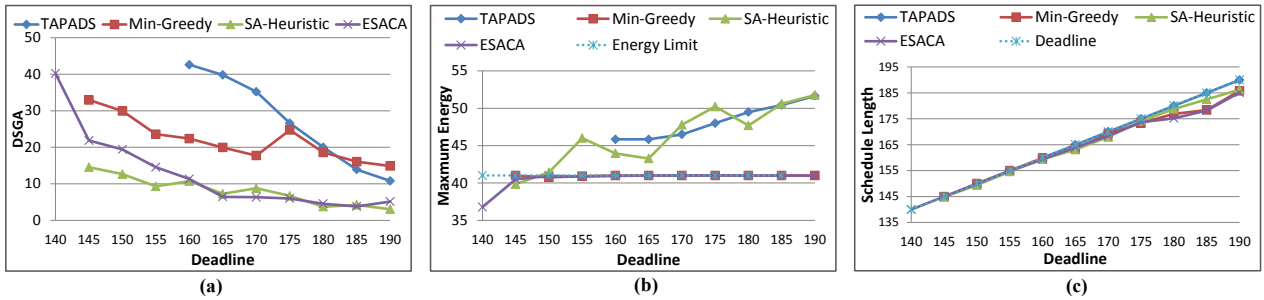

Fig. 6. Impact of Deadline on Algorithms' Performance

higher than the other two. When the energy limit lowers than 41mJ, Min-Greedy cannot get a valid solution. In general, ESACA can always meet the energy constraint, and give high security performance, even when available energy is limited.

C. Evaluation of Deadline Impacts

In this set of experiments, application size and energy limit are fixed as 30 and 41, respectively. The deadline is between 140 and 190. The results are presented in Fig. 6.

With the increase of deadline, DSGA of all algorithms are decreasing. That means when the time limits are extended, the performances of all the algorithms become better. We can see from the figures that no solution can be obtained by these algorithms except ESACA when deadline is tight. For Min-Greedy and SA-Heuristic, no solution is derived when deadline is 140. And TAPADS can get solutions only if deadline is longer than 160. ESACA shows high quality dealing with tight time constraints. Solutions obtained by SA-Heuristic and ESACA are better in general.

When deadline ranges from 145 to 155, the security performance of ESACA is lower than SA-Heuristic. If there is an application with enough tasks and tight deadline, energy limit tends to drive the tasks to be mapped to all CPUs evenly. It potentially increases the parallelism and communication possibility, but decreases the security of the system. That's why DSGA of ESACA increases when deadline is tight. SA-Heuristic is not restricted by energy limit, so it can obtain better security performance in this case.

In Fig. 6(b), we can notice that ESACA and Min-Greedy respect the energy budget. TAPADS and SA-Heuristics violate the energy limit when deadline is longer than 150. Therefore, ESACA obtain the best results with respect to security and energy. SA-Heuristic and ESACA does not fully utilize the processors. Meanwhile, when the deadline restriction is fully relieved, energy limit becomes dominating impacting the schedule length. Thus, Min-Greedy cannot fully use CPU time if deadline is too long.

VII. CONCLUSION

To provide sound security protections for SCDRES, especially to protect the confidentiality aspect of the internal communication, we must take security considerations into our scope from the early system design and optimization phases. In this paper, we focus on the design optimization problem of secure SCDRESs with energy and real-time constraints. Communication traffic reduction and message cryptographic

reinforcement are the key methods in the design procedure. DSGA is used to capture the security strength of the system, and must be minimized to achieve the best protection. At the same time, deadline and energy constraints must be satisfied. Due to the large complexity of finding optimal solutions, a GA based heuristic is presented for solving the problem efficiently. Furthermore, acceleration techniques are employed to obtain quick convergence. Extensive experiments are conducted on our heuristic together with three comparative approaches. Experimental results have demonstrated the efficiency of our proposed technique.

ACKNOWLEDGMENT

This work was partly supported by the National Natural Science Foundation of China under Grant No. 61003032, the Fundamental Research Fund for the Central Universities of China under Grant No. ZYGX2011J061, and the Research Fund of National Key Laboratory of Computer Architecture under Grant No. CARCH201104.

REFERENCES

[1] J. Wilbrink, D. Nativel, and T. Morin, "Networked networks and embedded microcontroller architectures," *Industrial & Military Applications*, vol. 4, no. 4, pp. 16–19, Feb. 2006.

[2] J. Guajardo, T. Wollinger, and C. Paar, "Cryptography in embedded systems: an overview," in *Proc. the Embedded World 2003 Exhibition and Conference*, 2003, pp. 735–744.

[3] V. Bonifaci, G. D'Angelo *et al.*, "Scheduling real-time mixed-criticality jobs," in *Proc. IEEE Transactions on Computers*, vol. 6, no. 8, Aug. 2012, pp. 1140–1152.

[4] S. Hong, T. Chantem, and X. S. Hu, "Meeting end-to-end deadlines through distributed local deadline assignments," in *Proc. IEEE Real-Time Systems Symposium*, Nov. 2011, pp. 183–192.

[5] D. Tămas-Selicean and P. Pop, "Design optimization of mixed-criticality real-time applications on cost-constrained partitioned architectures," in *Proc. IEEE Real-Time Systems Symposium*, Nov. 2011, pp. 24–33.

[6] K. Jiang, P. Eles, and Z. Peng, "Optimization of message encryption for distributed embedded systems with real-time constraints," in *Proc. of Design and Diagnostics of Electronic Circuits & Systems*, Apr. 2011, pp. 243–248.

[7] T. Xie and X. Qin, "Security-aware resource allocation for real-time parallel jobs on homogeneous and heterogeneous clusters," in *Proc. IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 5, May 2008, pp. 682–697.

[8] R. Chandramouli, S. Bapatla, and K. P. Subbalakshmi, "Battery power-aware encryption," in *Proc. ACM Transactions on Information and System Security*, vol. 9, no. 2, May 2006, pp. 162–180.

[9] Y. Ma, W. Jiang, N. Sang, and X. Zhang, "Arcsm: A distributed feedback control mechanism for security-critical real-time system," in *Proc. IEEE International Symposium on Parallel and Distributed Processing with Applications*, Jul. 2012, pp. 379–386.

[10] W. Jiang, K. Jiang, and Y. Ma, "Resource allocation of security-critical tasks with statistically guaranteed energy constraint," in *Proc. Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Aug. 2012, pp. 330–339.

[11] S. Contini, R. L. Rivest, M. J. B. Robshaw, and Y. L. Yin, "The security of the rc6 block cipher," Aug. 1998. [Online]. Available: http://www.rsa.com/rsalabs/node.asp?id=2512

[12] R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, "The rc6 block cipher," in *Proc. Advanced Encryption Standard (AES) Conference*, Aug.

[13] P. Eles, Z. Peng, P. Pop, and A. Doboli, "Scheduling with bus access optimization for distributed embedded systems," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 8, no. 5, pp. 472–491, 2000.

[14] V. Izosimov, P. Pop, P. Eles, and Z. Peng, "Design optimization of time- and cost-constrained fault-tolerant distributed embedded systems," in *Proc. Automation and Test in Europe Conference and Exposition (DATE)*, Mar. 2005, pp. 864–869.

[15] D. E. Golgberg, *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

[16] K. Z. Gkoutioudi and H. D. Karatza, "Multi-criteria job scheduling in grid using an accelerated genetic algorithm," *Journal of Grid Computing*, vol. 10, no. 2, pp. 311–323, Jun. 2012.