Resource Allocation of Security-Critical Tasks with Statistically Guaranteed Energy Constraint

Wei Jiang[1,3] Ke Jiang[2] Yue Ma[1]
[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China
[2]Department of Computer and Information Science, Linköping University
[3]Department of Informatics and Mathematical Modeling, Technical University of Denmark
Email: weijiang@uestc.edu.cn, ke.jiang@liu.se, yue_ma_880131@hotmail.com

ABSTRACT—**In this paper, we are interested in resource allocation for energy constrained and security-critical embedded systems. Tasks in such systems need to be successfully executed under certain energy budget and be robust against serious security threatens. Different to former energy minimal scheduling problem, we introduce a new optimization problem for a set of tasks with energy constraint and multiple security choices. We present a dynamic programming based approximation algorithm to minimize the security risk of the system while statistically guaranteeing energy consumption constraints for given energy slack ratio. The proposed algorithm is very efficient in both time and space dimensions, and achieves good solutions. Extensive simulations demonstrate the superiority of our algorithm over other approaches.**

I. INTRODUCTION

With the trend of being connected by wireless or wired networks, embedded systems are facing more and more severe security threatens from network attacks or vulnerability of themselves. Meanwhile, embedded systems are popular to be used in safety or reliability critical areas. All these reasons make the need of protecting security-sensitive data in embedded systems to be emerging [1]. Since the snooping, spoofing and alternating of security-critical data lead to system failure or crash, resulting in great loss of finance, life and even disaster to the earth, we take these kinds of systems as Security-Critical Embedded Systems (SCES), e.g. flight control systems [2], satellite communication systems, radar tracking systems. All such systems have strict security requirements. To make sure the successful implementation of SCES, a series of security service, i.e. integrity, confidentiality, authentication, and auditing service, need to be considered and protected to ensure security of the systems. With most suitable security services in place, SCES could be effectively protected with less system resources.

One major barrier against implementing SCES is the energy consumption, as security protections usually demand a significant amount of energy. Moreover, most of SCES are battery-powered and even under no nursing state. Quick energy consumption or early exhaustion of battery may lead to failure of critical task, resulting in unexpected loss, such as the energy incurred failure of Mar's Path Finder, NASA Spatial systems [3]. Hence, design of energy-efficient SCES is another key problem. Considering security and energy together has obviously become of great challenge, and an imperative work.

Resource allocation algorithm is a key approach to obtain high performance of embedded systems. Unfortunately, traditional approaches for real-time embedded applications are trying to assign most proper CPU resource to tasks with guaranteed timing requirements [4]. These researches ignore to consider both security and energy factors together in system-level design, which cannot be competent for security-critical embedded applications.

In this paper, we identify one common problem lying in many SCES, namely how to allocate resource to ensure security performance without sacrificing the energy constraints. Concretely, we are going to devise an allocation algorithm for a set of security-critical tasks with the purpose of minimizing the total security risk while satisfying the given energy budget. We propose specific policies to solve the problem with low time and space complexity, which is suitable for SCES.

The primary contributions are in three folds. Firstly, a novel security- and energy-sensitive application for SCES is presented. Secondly, we formulate the design optimization problem that minimizes the security risk under predefined energy budget. Finally, we propose an approximation algorithm, which provides statistically guarantee of energy constraint with average error of zero. And the algorithm has polynomial time and bounded space complexity.

The remainder of this paper is organized as follows. Section II describes the relative work. Section III presents the system model and system problem. Section IV presents our approximation scheduling mechanism. Section V demonstrates the simulation results, and Section VI concludes this paper and future work.

II. RELATED WORK

Energy related problems of embedded systems have been well studied from different perspectives. Based on DVS and DPM technology, there are many works on task scheduling to reduce the energy consumption of embedded systems like [5, 6]. Some work tried to estimate the energy consumption of program by the energy model in instruction level [7], or by macro modeling method [8]. Recently, energy characteristics of cryptographic algorithms also gained attention. Reference [9] measured the energy consumption of several chippers

978-0-7695-4824-1/12 $26.00 © 2012 IEEE
DOI 10.1109/RTCSA.2012.34

by measuring apparatus, and then established a mathematic model to denote the relation between security and energy cost. Detailed analysis has been done to study the energy characteristics of various ciphers under real embedded devices running Linux [10] and μC/OSII [11].

To improve the security protection of mission-critical systems, security-aware resource allocation becomes a promising topic. For security-critical distributed systems, authors of [2] proposed a heuristic algorithm to assign security services to real-time tasks under distributed cluster environments. Based on a job failure model on trust level, ref. [12] researched on scheduling algorithms for independent jobs over grid computing environments, and proposed a space-time genetic strategy. In [13], authors proposed a task assignment algorithm for heterogeneous distributed security-critical systems. A communication mechanism that maximizes the security protection of the internal communication in distributed real-time systems is presented in [14]. In [15], they devised a critical-path based resource allocation mechanism to improve the security quality of homogeneous parallel applications. Considering the security problem of embedded system, authors of [16] proposed a heuristic to maximize the security quality of periodic real-time tasks. Based on group-level security service model, ref. [17] designed two allocating approaches using integer linear programming technique and depth-first search. However, all the aforementioned works did not consider the energy factors of SCES, which is critical to be studied in SCES.

In this paper, we propose a polynomial approximation scheme which obtains near-optimal solution for specified slack bound and also statistically approximates given constraint with zero error. To the best of our knowledge, this is the first work that defines a security- and energy-aware resource allocation problem.

III. SYSTEM MODEL AND OPTIMIZATION PROBLEM

In this section, we first provide an example to motivate the need for studying the security- and energy-aware resource allocation. Then, the model of tasks and energy consumption will be given. We also introduce a risk metric to quantify the security protection strength of each task and finally formulate our optimization problem.

A. Modeling of security and energy-aware Application

In this paper we consider battery-powered embedded systems, which run security-critical tasks under restricted energy. An illustrative system is depicted in Fig. 1, which is an Unmanned Aerial Vehicle (UAV). The UAV is sending messages to a service center, which is assumed to have infinite computation resources for the sake of simplicity. Each task generates a message that is sent over an unsafe region. In order to make the communication secure (i.e. to protect the confidentiality and integrity requirements of the messages), we need to carry out cryptography, e.g. like RC5, DES and SHA-2 on the tasks besides their normal executions. Therefore, the energy consumption of each task consists of two parts from its normal execution and extra security protection, which will be discussed in more details in Sec. III(B). However, the system can only provide limited energy budget for all tasks. So, how to allocate resources to protect different messages becomes a clear design trade-off. In another word, our purpose is to provide best security protection for the whole system while meeting certain energy budget.

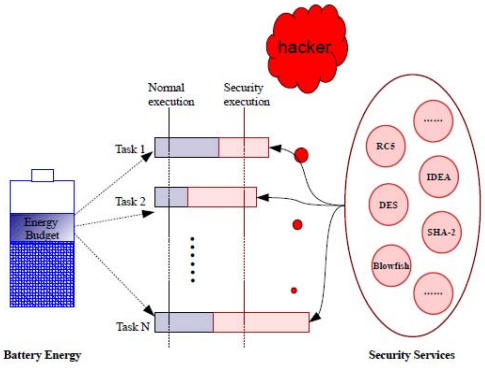

Fig.1 Motivating application

Given a set of security- and energy-aware tasks running on SCES, each task T_i is modeled by a tuple $T_i=\{BE_i, D_i, S_i, S_i^{DM}, V_i, SR_i\}$. BE_i denotes the basic energy consumption caused by execution of its non-security part. D_i is the data size which needs to be protected by security service. S_i and S_i^{DM} are the chosen and designated security level of T_i respectively. If S_i^{DM} is achieved, this task is assumed to be absolutely secure. V_i is the security impact value of T_i, and represents the relevant importance of the message generated by the task, e.g., the task transmitting pictures of enemy's military base has a higher security impact than the task transmitting the current weather condition. SR_i is the security risk of T_i, which means the expected security loss of the security protection. In this paper we only focus on security-critical tasks. However, it is easily generalized to non-security tasks with $D_i = 0$.

B. Energy consumption of security tasks

Implementing additional security service leads to better security protection, but also results in sacrifice of other protections. For example, the IDEA encryption on one task may incur degradation of protecting other tasks.

In general, there are several classes of security services, e.g. confidentiality and integrity services. For confidentiality algorithms, there exists a series of protection algorithms, e.g. DES, 3DES, IDEA, AES, CAST, RC2, RC4, RC5, Blowfish. It is the same case for integrity services, e.g. MD2, MD4, MD5, SHA, SHA1, and HMAC. Authors in [10] have measured energy consumption characteristics of most security algorithms

in a PDA-based Linux embedded platform. Inspired by their work, authors of [11] conducted lots of work to measure energy costs of security services in a real-time operating system based on ARM9 platform, in which their obtained results of energy cost will be more significant for security-critical real-time systems. Table I lists the obtained results for eight mostly used algorithms protecting confidentiality. Precisely quantifying the security strength of security algorithms is still an open and hard problem. Different metric will result in different security level assignment and newly developed algorithm may have higher level while lower overhead like AES-128 in Table I. So we enumerate the level according to their reasonable security strengths [2] in this paper, but we can also replace it with more reasonable data. In next section, we will introduce a risk-driven metric to quantify the security quality.

TABLE I. ENERGY COST OF CONFIDENTIALITY ALGORITHMS

Algorithm	Key Setup (μ J)	Encrypt (μ J/B)	Sec. level
RC4	54.54	1.96	1
BLOWFISH	69.36	4.6	2
RC5	54.18	2.8	3
DES	54.28	9.3	4
IDEA	54.56	5.7	5
SKIPJACK	55.88	7.95	6
3DES	54.28	25	7
AES-128	53.97	7.0	8

If task T_i generates D_i Bytes data with confidentiality requirement, then the energy consumption of T_i can be formulated as Eq. (1). Energy is composed of normal part (BE_i) and security part ($En\_confidentiality$). BE_i is supposed to be given by the task user. $KeySetup(S_i)$ is energy consumption of the key setup procedure. $Conf(S_i)$ is the mapping function of security level S_i to unit energy consumption of confidential encryption. Taking S_i=3 for example, $KeySetup(S_i)$ and $Conf(S_i)$ are thus energy consumptions of RC5 algorithm according to Table I.

$$\begin{aligned} En_i &= BE_i + En\_Confidentiality \\ &= BE_i + KeySetup(S_i) + Conf(S_i) \cdot D_i \end{aligned} \quad (1)$$

C. Security Risk of each task

To quantify the security quality of tasks, it is necessary to introduce security risk model. In [15] and [17], linear security profit model, which is the sum of weighted security levels, is assumed to evaluate security quality of security-critical tasks. Obviously, linear model is impractical to depict the security quality of security-critical tasks. Ref. [12] proposed a trust level based job failure model to test the failure probability of security-critical tasks. Based on the job failure model, we establish a security risk model. Firstly, the failure probability of security-critical task T_i with security level S_i is defined as

$$P_i^{risk} = \begin{cases} 1-\exp^{-\lambda_i (S_i^{DM} - S_i)}, & \text{if } S_i < S_i^{DM} \\ 0, & \text{if } S_i \geq S_i^{DM} \end{cases} \quad (2)$$

where λ_i is the security risk coefficient of T_i, which can be adjusted by the designer based on different risk scenarios. Eq. (2) demonstrates that if the assigned security level is greater or equal to the demanded security, we assume no failure will happen when security attacks are mounted. Inversely, the task has the probability to fail, and bigger security demand gap leads to higher probability.

Since security risk is product of security breach probability and consequence of security breach [18], we model the security risk (SR) of T_i as Eq. (3).

$$SR_i = V_i \cdot P_i^{risk} = \begin{cases} V_i \cdot (1-\exp^{-\lambda_i (S_i^{DM} - S_i)}), & \text{if } S_i < S_i^{DM} \\ 0, & \text{if } S_i \geq S_i^{DM} \end{cases} \quad (3)$$

D System problem description

In this paper, we consider a task set of N tasks running in an energy-constrained security-critical embedded system. System managers give the energy budget ($En\text{-}budget$) for the execution of these tasks. In other words, if their execution leads to energy violation, then it may result in the failure of other components in the system. In special situations, it demands energy consumption not exceed $(1+\beta)En\_budget$, where β is recognized as energy slack ratio. The system problem is to assign most suitable security services for tasks with the minimum security risk, and satisfying the given energy budget. Thus, Overall Security Risk (OSR) minimizing problem can be formulated as

$$\text{Min } OSR = \sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} SR(T_i) = \sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} \cdot V_i \cdot P_{ij}^{risk} \quad (4)$$

$$\text{S.T. } \begin{cases} \sum_{i=1}^{N} En_i \leq (1+\beta) En\_budget \\ \sum_{j=1}^{M} x_{ij} = 1,\ x_{ij} = \{0,1\} \end{cases}$$

where, x_{ij} denotes whether task T_i is assigned security protection on j-th level. Clearly, the optimization problem is a Multiple Choice Knapsack Problem, which is NP-hard. It will take high computation overhead to get optimal solution, thus we try to find low-complexity assignment algorithm, which can obtain good solution while satisfying the predefined constraints efficiently.

IV. RANDOM SCALED APPROXIMATION ALLOCATION POLICY

The overall security risk minimizing problem is NP-hard, so it needs efficient methods to solve it. In this paper, we transform the problem to a multiple stage decision-making procedure, and use Dynamic Programming to address it.

A Multi-stage decision-making procedure

Considering a set of N tasks, the security risk minimization problem can be taken as an N-stage decision-making procedure. The decision variable for the i-th stage is S_i, that needs to be assigned. Thus the purpose can be transformed to find a vector $\overline{S} = (S_1, S_2, ..., S_N)$ with minimum system security risk while satisfying the energy budget constraint.

We denote a three-tuple $(\xi_{ik}, \gamma_{ik}, S_{ik})$ to describe the k-th state in i-th stage. ξ_{ik} and γ_{ik} are two values in this state, which present the accumulated energy consumption and the accumulated security risk for the first i tasks respectively. S_{ik} is the specific value of security decision variable S_i in this state. The State Set Ω_i for i-th stage is defined as

$$\Omega_i = \{(\xi_{i1}, \gamma_{i1}, S_{i1}), (\xi_{i2}, \gamma_{i2}, S_{i2}), ..., (\xi_{ii^{\max}}, \gamma_{ii^{\max}}, S_{ii^{\max}})\}$$

where $i^{\max}$ is the number of all states for i-th stage. Given the Ω_{i-1} in (i-1)-th stage, we can obtain the state set Ω_{ik} by adding energy consumption and security risk of task i under security level $S_{ik} \in [1, S_i^{DM}]$ to all states of Ω_{i-1},

$$\begin{aligned}\Omega_{ik} &= \Omega_{i-1} \oplus (En_i(S_{ik}),\ SR(S_{ik}), S_{ik}) \\ &= \{(\xi_{i-1,1} + En_i(S_{ik}),\ \gamma_{i-1,1} + SR(S_{ik}),\ S_{ik}), \\ &\quad (\xi_{i-1,2} + En_i(l_{ik}), \gamma_{i-1,2} + SR(S_{ik}),\ S_{ik}),\ ...\ \}\end{aligned}$$

Thus, the state set in i-th stage is

$$\Omega_i = \bigcup_{k=1}^{|S_i|} \Omega_{ik}$$

We set $|S_i|$ as the number of available options for task T_i. Obviously, in the first stage, there are only $|S_1|$ states, and the number of states in i-th stage is the production of $|S_i|$ and the number of states in stage i-1. The maximal state space is as follows,

$$SS = \prod_{i=1}^{N} |S_i| \tag{5}$$

From Eq. (5), the state space grows exponentially as the number of tasks and security choices grows. Thus it is infeasible to get the optimal solution using Dynamic Programming algorithm for large system designs. So, we must find efficient methods to reduce the solution space.

Inspired by the approximation algorithm of Knapsack problem [19], grouping the energy consumption into a series of discretized integers is a good approach to reduce the decision states on each stage. Setting δ as the group factor, then the energy consumption of each task can be divided to an integer by En_i / δ. Thus, in each stage, we just keep the state with minimal security risk when several states have the same energy integer. Bigger δ gives smaller scaled energy integer and smaller number of states in each decision-making stage.

Table II Matrix of Decision states

0	1	2	…	M
T_1	$(\xi_{1,1}, \gamma_{1,1}, S_{1,1})$	$(\xi_{1,2}, \gamma_{1,2}, S_{1,2})$	…	$(\xi_{1,M}, \gamma_{1,M}, S_{1,M})$
T_2	$(\xi_{2,1}, \gamma_{2,1}, S_{2,1})$	$(\xi_{2,2}, \gamma_{2,2}, S_{2,2})$	…	$(\xi_{2,M}, \gamma_{2,M}, S_{2,M})$
…	…	…	…	…
T_{N-1}	$(\xi_{N-1,1}, \gamma_{N-1,1}, S_{N-1,1})$	$(\xi_{N-1,2}, \gamma_{N-1,2}, S_{N-1,2})$	…	$(\xi_{N-1,M}, \gamma_{N-1,M}, S_{N-1,M})$
T_N	$(\xi_{N,1}, \gamma_{N,1}, S_{N,1})$	$(\xi_{N,2}, \gamma_{N,2}, S_{N,2})$	…	$(\xi_{N,M}, \gamma_{N,M}, S_{N,M})$

Due to the number of states in each stage can be maximized to a constant $M = \lceil En\_budget / \delta \rceil$, we use a two-dimension table to demonstrate all states as Table II.

B Random Scaled Policy

In this paper, we introduce a random scaling approximation mechanism. For each task T_i, the energy consumption is scaled by En_i / δ. If the obtained result is not an integer, we randomly scale it to the closest smaller integer or to the closet bigger integer according to Eq. (6). The purpose of randomly scaling is to let the average error equal to zero statistically. This gives better solution while having the same complexity of down-scaling or up-scaling methods [6]. We will prove its advantages in later sections.

$$En_i^{\delta} = \begin{cases} \left\lfloor \dfrac{En_i}{\delta} \right\rfloor, & \text{probability of } \rho_1 = En_i/\delta - \left\lfloor \dfrac{En_i}{\delta} \right\rfloor \\ \left\lceil \dfrac{En_i}{\delta} \right\rceil, & \text{probability of } \rho_2 = 1 - \rho_1 \end{cases} \tag{6}$$

For N tasks, we can obtain that the totally discretized energy consumption as the following equation.

$$OEn^{\delta} = \sum_{i=1}^{N} En_i^{\delta} \tag{7}$$

And the total error for N tasks is

$$OErr = \sum_{i=1}^{N} Err_i = \sum_{i=1}^{N} (En_i - \delta \cdot En_i^{\delta}) \tag{8}$$

Theorem 1. Given the energy slack ratio β and the energy budget En_budget, the maximum value of grouping factor δ is $\delta = \beta \cdot En\_budget / (N+1)$.

Proof. Clearly, Random Scaling (RS) policy has the total error between Up Scaling (US) and Down Scaling (DS) policy. For US policy, the real energy consumption has the negative error to scaled energy, and the overall error ($OErr$) is as follows,

$$\begin{aligned} OErr^{US} &= \sum_{i=1}^{N} Err_i = \sum_{i=1}^{N} (En_i - \delta \cdot En_i^{\delta}) \\ &= \sum_{i=1}^{N} (En_i - \delta \cdot \left\lceil \frac{En_i}{\delta} \right\rceil) \\ &\geq \sum_{i=1}^{N} (En_i - \delta \cdot (\frac{En_i}{\delta} + 1)) \\ &= \sum_{i=1}^{N} (-\delta) \\ &= -N\delta \end{aligned}$$

For DS policy, the real energy consumption has the positive error, and the error can be inferred by the following equation.

$$\begin{aligned} OErr^{DS} &= \sum_{i=1}^{N} Err_i = \sum_{i=1}^{N} (En_i - \delta \cdot En_i^{\delta}) \\ &= \sum_{i=1}^{N} (En_i - \delta \cdot \left\lfloor \frac{En_i}{\delta} \right\rfloor) \\ &\le \sum_{i=1}^{N} (En_i - \delta \cdot (\frac{En_i}{\delta} - 1)) \\ &= \sum_{i=1}^{N} (\delta) \\ &= N\delta \end{aligned}$$

Thus for RS policy, the total error can be ranged as follows,

$$-N\delta \le OErr \le N\delta \tag{9}$$

That is, RS can lead the real energy consumption to at largest $N\delta$ energy error comparing to the scaling energy. According to the description of 2-dimensional decision-making table, the maximum energy value is $\lceil En\_budget/\delta \rceil$ which has the most δ error to En_budget. Thus, the maximum error of the proposed policy comparing to the given energy budget is $(N+1)\delta$. Hence, to guarantee the loosely energy constraint, we need to satisfy the following inequality.

$$(N+1)\delta \le \beta \cdot En\_budget$$

Then we obtain,

$$\delta \le \frac{\beta \cdot En\_budget}{N+1} \tag{10}$$

According to inequality (10), we can get the conclusion of Theorem 1. ***End of proof***.

Theorem 1 shows that we can only take the grouping factor δ within $[0, \beta \cdot En\_budget/(N+1)]$ by giving energy slack ratio β. If δ is bigger than $En\_budget/(N+1)$, then the energy slack ratio cannot be guaranteed.

Lemma 1. The accumulated error of N tasks energy consumption is statistically zero.

Proof. This is because RS rounds the energy to ceiling or to floor integer according to certain probability. Tasks can have positive and negative errors, and then positive errors and negative errors cancel out one another in the whole task set. Thus the accumulated error is minimized to zero statistically.

Because the energy value of each task with RS policy is a random variable, then we can deduce the expected or mean error of each task. Without loss of generality, we take T_i as an example, and assume that En_i is not the integer times of δ. The expected error of T_i is inferred as following.

$$\begin{aligned} \mathfrak{e}(Err_i) &= (En_i - \delta \left\lceil \frac{En_i}{\delta} \right\rceil) \cdot \rho_1 + (En_i - \delta \left\lfloor \frac{En_i}{\delta} \right\rfloor) \cdot \rho_2 \\ &= (En_i - \delta \left\lceil \frac{En_i}{\delta} \right\rceil) \cdot (\frac{En_i}{\delta} - \left\lfloor \frac{En_i}{\delta} \right\rfloor) \\ &\quad + (En_i - \delta \left\lfloor \frac{En_i}{\delta} \right\rfloor) \cdot (1 - \frac{En_i}{\delta} + \left\lfloor \frac{En_i}{\delta} \right\rfloor) \\ &= (En_i - \delta \left\lceil \frac{En_i}{\delta} \right\rceil) \cdot (\frac{En_i}{\delta} - \left\lfloor \frac{En_i}{\delta} \right\rfloor) \\ &\quad + (En_i - \delta \left\lfloor \frac{En_i}{\delta} \right\rfloor) \cdot (\left\lceil \frac{En_i}{\delta} \right\rceil - \frac{En_i}{\delta}) \\ &= 0 \end{aligned}$$

Then the accumulated error of N tasks is,

$$\mathfrak{e}(OErr) = \sum_{i=1}^{N} \mathfrak{e}(Err_i) = 0 \tag{11}$$

Thus we get the conclusion of Lemma 1.

Lemma 2. The standard deviation of RS approximating mechanism is bounded by

$$\sigma_{OErr} = \frac{\beta\sqrt{N} \cdot En\_budget}{2(N+1)} \tag{12}$$

Proof. According to the definition of variance of random variable, we can have the variance of energy error of Tasks T_i as follows,

$$\begin{aligned} \mathfrak{D}(Err_i) &= \sum_{k} (Err_i^k - \mathfrak{e}(Err_i))^2 \cdot \rho(Err_i^k) \\ &= (En_i - \left\lceil \frac{En_i}{\delta} \right\rceil \delta - \mathfrak{e}(Err_i))^2 \cdot \rho_1 \\ &\quad + (\frac{En_i}{\delta} - \left\lfloor \frac{En_i}{\delta} \right\rfloor \delta - \mathfrak{e}(Err_i))^2 \cdot \rho_2 \\ &= (En_i - \left\lceil \frac{En_i}{\delta} \right\rceil \delta)^2 \cdot (\frac{En_i}{\delta} - \left\lfloor \frac{En_i}{\delta} \right\rfloor) \\ &\quad + (En_i - \left\lfloor \frac{En_i}{\delta} \right\rfloor \delta)^2 \cdot (1 - \frac{En_i}{\delta} + \left\lfloor \frac{En_i}{\delta} \right\rfloor) \\ &= (En_i - \left\lceil \frac{En_i}{\delta} \right\rceil \delta)^2 \cdot (\frac{En_i}{\delta} - \left\lfloor \frac{En_i}{\delta} \right\rfloor) \\ &\quad + (En_i - \left\lfloor \frac{En_i}{\delta} \right\rfloor \delta)^2 \cdot (\left\lceil \frac{En_i}{\delta} \right\rceil - \frac{En_i}{\delta}) \\ &= \delta^2 (\frac{En_i}{\delta} - \left\lceil \frac{En_i}{\delta} \right\rceil) \cdot (\frac{En_i}{\delta} - \left\lfloor \frac{En_i}{\delta} \right\rfloor) \cdot (\left\lfloor \frac{En_i}{\delta} \right\rfloor - \left\lceil \frac{En_i}{\delta} \right\rceil) \\ &= \delta^2 (\left\lceil \frac{En_i}{\delta} \right\rceil - \frac{En_i}{\delta}) \cdot (\frac{En_i}{\delta} - \left\lfloor \frac{En_i}{\delta} \right\rfloor) \\ &= \delta^2 (1 - \rho_1) \cdot \rho_1 \end{aligned}$$

For $\mathfrak{D}(Err_i) = \delta^2(1-\rho_1)\cdot\rho_1$, the derivative of it is

$$\begin{aligned} &\mathfrak{D}'(Err_i) = \delta^2(1-2\rho_1), \\ &\{\mathfrak{D}'(Err_i) = 0, \text{ when } \rho_1 = 1/2\ \} \end{aligned} \tag{13}$$

Then we obtain the maximum variance value of Err_i as

$$\mathfrak{D}^{\max}(Err_i) = \delta^2(1-\frac{1}{2})\cdot\frac{1}{2} = \frac{1}{4}\delta^2 \quad (14)$$

Thus, the total variance of N tasks is

$$\mathfrak{D}^{\max}(OErr) = N\cdot\mathfrak{D}^{\max}(Err_i) = \frac{1}{4}N\delta^2 \quad (15)$$

Therefore, the maximal standard variance of N tasks can be inferred by Theorem 1 and Eq. (15)

$$\sigma_{OErr} = \sqrt{\mathfrak{D}^{\max}(OErr)} = \frac{1}{2}\delta\sqrt{N} = \frac{\beta\sqrt{N}\cdot En\_budget}{2(N+1)} \quad (16)$$

C RS-based Dynamic Programming Algorithm

Based on RS policy and the 2-dimensional states presentation, we propose a Dynamic Programming based security-aware assignment algorithm. The main purpose of Random Scaling based Dynamic Programming (RSDP) is to assign most suitable security level to each task with minimum system security risk. The pseudo-code of the algorithm is given in Fig. 2. The approximation scheme works by reducing the number of decision states in each stage. For initialization, RSDP calculates the grouping factor δ in line 1, and initializes the 2-dimension decision-making matrix by lines 2-5. From line 5 to line 16, it is the upgrading procedure of decision states for every decision-making stage. Based on each non-zero risk state in (i-1)-th stage, RSDP calculate every possible state for i-th stage (lines 7-14). If the immediately generated state has lower security risk comparing to prior state in i -th stage with the same energy cost, RSDP replaces the old state by the new one with smaller security risk. After all the decision states in every stage have been renewed, RSDP selects the state with smallest security risk in the N-th stage as line 17-18. Security risk of selected state is

1. Calculate the grouping factor $\delta = \beta\cdot En\_budget/(N+1)$
2. Initialize matrix $\mathbb{Z}_{N\times\lceil En\_budget/\delta\rceil}$
3. For each element in $\mathbb{Z}_{N\times\lceil En\_budget/\delta\rceil}$
4. $z_{i,j} = (\xi_i, \gamma_i, S_i) = (j, 0, 0)$
5. For $(i = 1, i < N, i++)$
6. While $z_{i-1,j} = (\xi_{i-1}, \gamma_{i-1}, S_{i-1}) \neq (j,0,0)$ Do
7. For $(S_i^{\#} = 1, S_i^{\#} < S_i^{DM}, S_i^{\#}++)$
8. $\xi_i^{\#} = \xi_{i-1} + En_i^{\delta}$
9. $\gamma_i^{\#} = \gamma_{i-1} + SR_i$
10. IF $\xi_i^{\#} > \lceil En\_budget/\delta\rceil$
11. Then break;
12. IF $(\gamma_i^{\#} < \gamma_i \mid \xi_i^{\#} = \xi_i)$
13. Then $(\xi_i, \gamma_i, S_i) = (\xi_i^{\#}, \gamma_i^{\#}, S_i^{\#})$
14. EndFor
15. EndWhile
16. EndFor
17. For j=1 to $\lceil En\_budget/\delta\rceil$
18. Find $z_{N,j}^{*}$ with minimal security risk γ_N
19. For N to 1
20. Mark the final decision vector $\overline{S} = (S_1, S_2, ..., S_N)$

Fig. 2 The pseudo code of RSDP

the final solution of the optimization problem. To obtain the final security level, RSDP goes back from N to 1st stage to get the vector of security assignments (lines 19-20).

Theorem 2. RSDP is a polynomial time complexity algorithm with $O(RSDP) = N^2\cdot|S_i|/\beta$. And the space complexity is bounded up to $\lceil En\_budget/\delta\rceil$.

Proof. From Fig. 2, we can deduce the time complexity of RSDP. It takes $O(1)$ by line 1, and $O(\lceil En\_budget/\delta\rceil)$ by lines 3-4. Based on each state in (i-1)-th stage, the states upgrading procedure of i-th stage (lines 7-13) will cost time of $|S_i|$; and there exists at most $\lceil En\_budget/\delta\rceil$ states in (i-1)-th stage (line 6). Thus, for all N stages, the state upgrading procedure will take $O(N\cdot\lceil En\_budget/\delta\rceil\cdot|S_i|)$, (seeing lines 5-16). For lines 17 to 20, it takes $O(N)$ time. Therefore, we obtain the complexity of RSDP as,

$$\begin{aligned} O(RSDP) &= O(1) + O(N\cdot\lceil En\_budget/\delta\rceil\cdot|S_i|) + O(N) \\ &= O(N\cdot\lceil En\_budget/\delta\rceil\cdot|S_i|) \\ &= O(N(N+1)\cdot|S_i|/\beta) \\ &= O(N^2\cdot|S_i|/\beta) \end{aligned}$$

Therefore, the time complexity of RSDP is polynomial in the number of tasks N, security choices $|S_i|$ and $1/\beta$.

On each stage, we only need to store $\lceil En\_budget/\delta\rceil$ states at most, so it is the upper bound of memory space for our proposed RSDP algorithm.

Theorem 3. The proposed algorithm is a polynomial complexity algorithm with guaranteed energy slack ratio and zero error violation of energy budget statistically.

Proof. The grouping factor of RSDP is set as $\delta = \beta\cdot En\_budget/(N+1)$, seeing Fig. 2. Hence, according to Theorem 1, we can obtain that RSDP can satisfy the given energy slack ratio. According to Lemma 1, we can get to know that the energy error is statistically zero, thus RSDP can have zero error violation of given energy budget. According to Theorem 2, we know that RSDP is polynomial complexity. Therefore, we can have the conclusion that RSDP is a polynomial complexity algorithm with guaranteed energy slack ratio, and zero error violation of ideal energy budget statistically.

V. SIMULATION RESULTS

In this section, we conduct experiments to verify the performance of the proposed algorithm. We implement a simulator using C# programming language to implement the security assignment procedure. For evaluation purposes, we compare our RSDP algorithm with three other methods, named USDP, Greedy and SEAS. The performance metrics in our experiments are overall security risk, real energy consumption, and energy error ratio. Energy error ratio is the ratio of real energy consumption error and energy budget.

- **USDP:** It utilizes Up-Scaling approximation scheme like reference [6], but we focus on risk minimizing

problem not energy minimal problem.

- **Greedy:** For this scheme, we assign the security level to tasks in a greedy fashion. It provides the currently highest security level to tasks step by step according to their natural order until all available energy slack is depleted.
- **SEAS:** It is a Security and Energy Aware Scheme, which is similar to SASES [16]. SEAS escalates the security level by comparing risk-energy ratios among tasks just like the benefit-cost ratio used in SASES.

We conduct four groups of simulations. For all the tasks, the basic energy consumptions are randomly in range of 0.1J to 0.2J, the security demands are randomly assigned between levels 6 and 8 for confidentiality protection. The impact value of each task is randomly generated between 1 and 10. In this paper we consider the energy of security part is nearly the same order of basic energy cost. Thus, we set the size of sensitive data ranges from 10KB to 40KB uniformly based on the energy costs of security services in Table I, For simplification, the security coefficient λ is set to 1 for all tasks.

A Impacts of energy slack ratio on energy and risk

The goal of this simulation is to evaluate the performance under different energy slack ratio. We set the energy budget to 4.5 Joule. The number of tasks is fixed as 20. Other task parameters are generated randomly based on the range described formerly. We vary the energy slack ratio from 0.01 to 0.2 with step increment of 0.01. The simulation results of overall security risk, real energy consumption and energy error are shown in Fig. 3, 4 and 5 respectively. For security risk in Fig. 3, we get to know that RSDP has the minimal security risks, while SEAS incurs maximal security risks under different energy slack ratio. Specifically speaking, USDP, Greedy and SEAS obtain increased security risk averagely up to 2.8%, 5.1% and 8.2% of RSDP. From Fig. 4, we can notice that the real energy consumption of RSDP is randomly changed around the energy budget value due to the random scaling policy. For USDP, it is decreasing with bigger β, which is because that bigger β means bigger δ, and the negative error of USDP is getting greater. One interesting thing is that all of the three test metrics of SEAS and Greedy are fixed under different β due to the fact that β is affecting the searching procedure in neither of them.

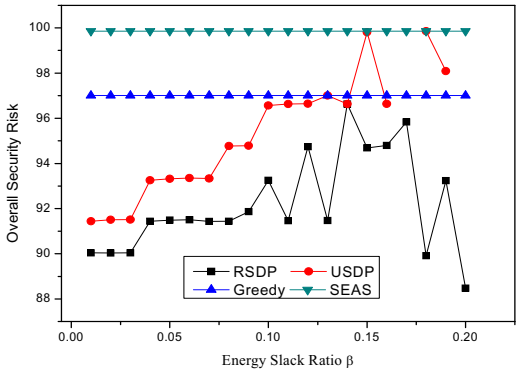

Fig. 3 Security risk under various slack ratios

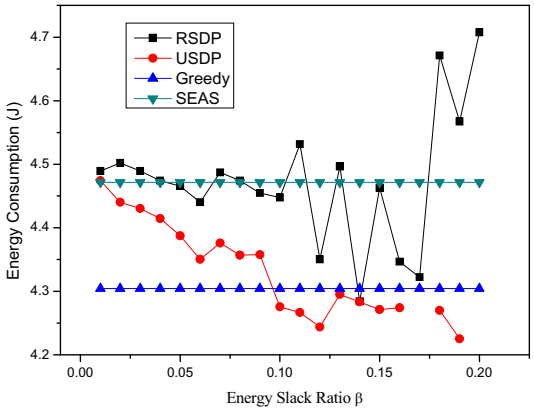

Fig. 4 Energy consumptions under various slack ratios

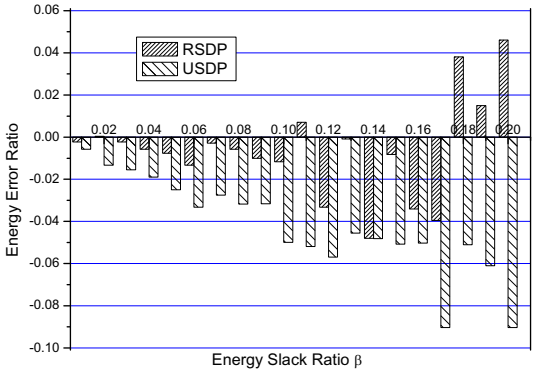

Fig. 5 Energy error ratios under various slack ratios

Fig. 5 illustrates the energy error ratios. RSDP can guarantee the given energy slack ratio even if it is very small, e.g. 0.01 and 0.02. Another interesting thing is that USDP always gets negative error ratio and RSDP can obtain smaller error ratio when it is negative. For example, when β is set to be 0.12 and 0.16, the energy error ratio of RSDP is only 0.033 and 0.034, while USDP is 0.057 and 0.099. This is because RSDP utilizes random scaling policy which cancels out the positive and negative errors. As depicted in the figures, RSDP is the best algorithm among the four that get the lowest risk with little energy budget error.

B Impact of delta on security risk and energy error

In the second group of simulation, we are going to evaluate the impact of grouping factor δ on the performances of the four algorithms. The energy budget and the number of tasks are also set to 4.5 Joule and 20 respectively. In this part, we don't deduce δ by the energy slack ratio β, but we directly vary the grouping factor δ from 2000 uJ to 40000 uJ with step increment of 2000 uJ instead. The results of total security risk, real energy consumption and energy error are shown in Fig. 6, 7 and 8 respectively. For security risk in Fig. 6, the overall security risk of RSDP is the lowest among the four algorithms. RSDP's security risk is averagely 4.5% less than USDP. Greedy and SEAS get more risk which is also fixed in this experiment. We also find that USDP

cannot get the solution when δ is 3600 uJ and 3800 uJ. This means the schedulability of RSDP is more robust than USDP.

From Fig. 7 and 8, we get the same situations as the first experimental group. The real energy consumption of USDP is decreasing with bigger δ. This is because that bigger δ results in greater approximation error. The real energy consumption of RSDP is not changing as great as USDP either. Fig. 8 illustrates the energy error ratios of RSDP and USDP. The energy error ratios of Greedy and SEAMS are fixed as 0.0435 and 0.0063, so we omitted them in Fig. 8. Clearly, both of RSDP and USDP get bigger energy error ratios if δ increases. RSDP can also obtain smallest error relatively. For example, when δ is set to 3000 uJ, the energy error ratio of RSDP is only 0.0215, while USDP is 0.0481. Given β =0.05, we can get the maximal δ as 10714 uJ according to Theorem 1. From Fig. 8, we can see that energy slack ratios are all within 0.05 when δ is from 2000 uJ to 10000 uJ. This proves the correctness of Theorem 1. Based on the results above, we can conclude that RSDP is a better algorithm to get less risk but with smaller energy budget error.

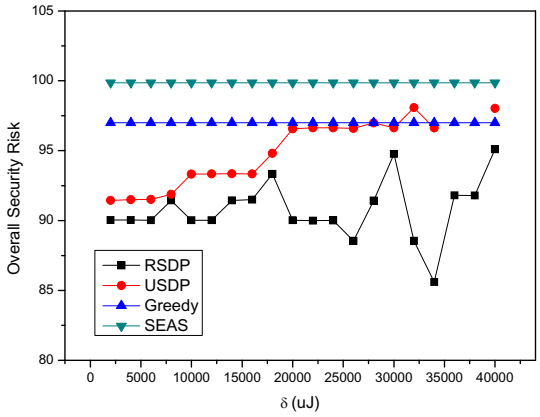

Fig. 6 Impact of group factor δ on overall security risk

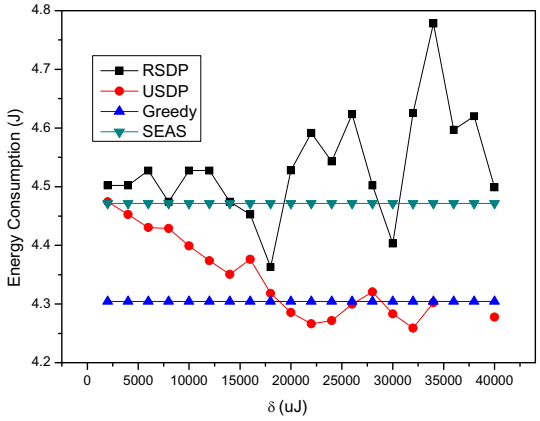

Fig. 7 Impact of group factor δ on energy consumption

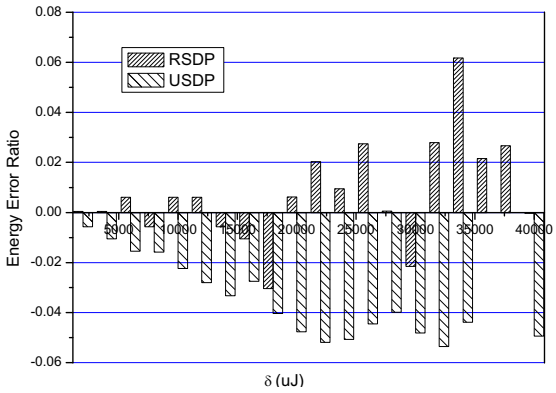

Fig. 8 Impact of group factor δ on energy error ratio

C *Impact of energy budget on security risk and energy*

In this group of simulation, we make efforts to verify the performances under various energy budgets. We set the energy slack ratio β to 0.04. The energy budget varies from 3.5 J to 9.2 J with the step increment of 0.3 Joule. The rest simulated parameters are generated as in the second experiment. Fig. 9, 10 and 11 are numerical results of these four algorithms under different energy budget respectively. From Fig. 9, we find some interesting observations. When the energy budget is small like 3.5 J and 3.8 J, no algorithm can get the solution for the task set. This is because the energy budget is too small to provide the minimal energy consumption when all tasks select the lowest security levels. On the other hand, when energy budget is up to 6.8 J, all the three algorithms consume the same energy. The reason is that the energy budget is enough to provide each task with their designated security level. In the range of 4.1 J to 6.5 J, the real energy consumptions of all algorithms are increasing with energy budget. SEAS consumes more energy than USDP and Greedy methods, while RSDP has the random value up or down the budget.

As can be seen in Fig. 10, the risk values of all algorithms have the same trend as the first two experiments between 4.1 J and 6.5 J. RSDP gets the minimal security risk, while SEAS incurs the greatest which is 25.7% more than RSDP on average when energy budget is between 4.1 J and 6.5 J. This means SEAS is not a good scheme to handle risk minimizing problem identified in this paper. The security risks of Greedy and USDP is between RSDP and SEAS. From Fig. 11, we can obtain the energy error ratio results. From 7 J, the energy error ratio is greatly decreased below zero. This is because the energy consumptions of all algorithms are fixed while the energy budget is increased step by step. In the range of 4.1 J to 6.5 J, RSDP and USDP can guarantee the energy slack ratio of 0.04, but Greedy sometime violates the requirement. We can also notice that RSDP is better in error ratios aspect, i.e. it is closer to zero. Therefore, RSDP analyzes the tradeoff between the security risk and energy consumption efficiently, which tries to make full use of the energy with statistically zero error while getting minimal security risk.

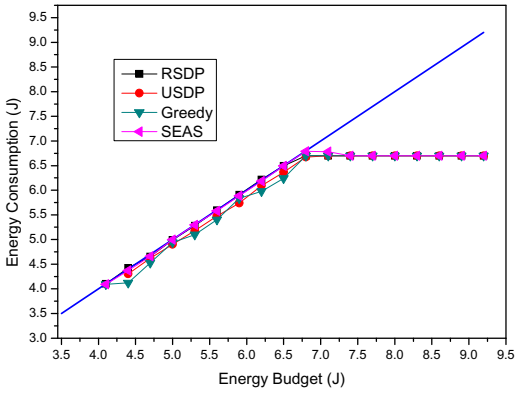

Fig. 9 Impact of energy budget on energy consumption

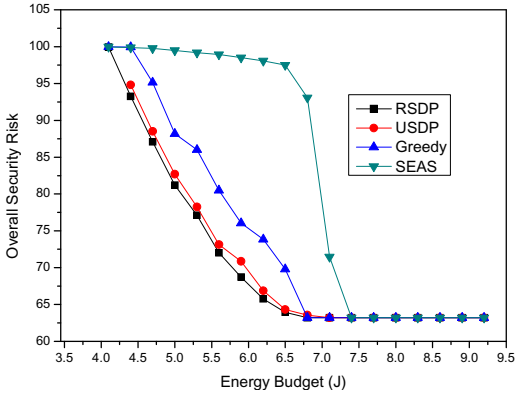

Fig. 10 Impact of energy budget on security risk

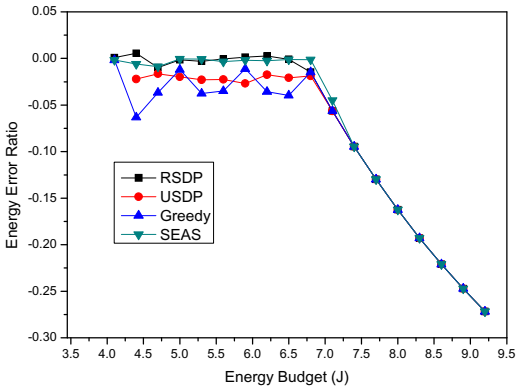

Fig. 11 Impact of energy budget on energy error

D *Impact of task number on security risk and energy*

In the last group of simulation, we test the impact of application sizes on the performance of these algorithms. The energy budget and the energy slack ratio are set to 15 Joule and 0.04 respectively. We vary the task number from 5 to 100 with step increment of 5.

Table III Energy error ratio of each algorithm

Algorithm	RSDP	USDP	Greedy	SEAS
Error Ratio	0.00021	0.0208	0.01123	0.00057

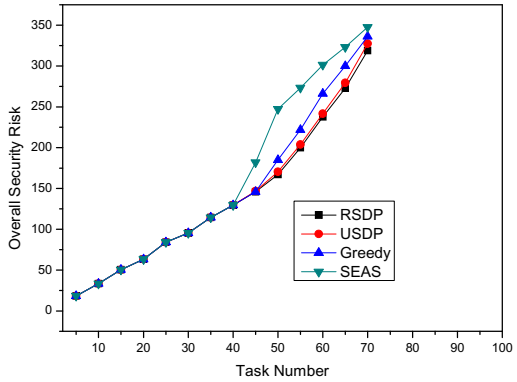

Fig. 12 Impact of task numbers on security risk

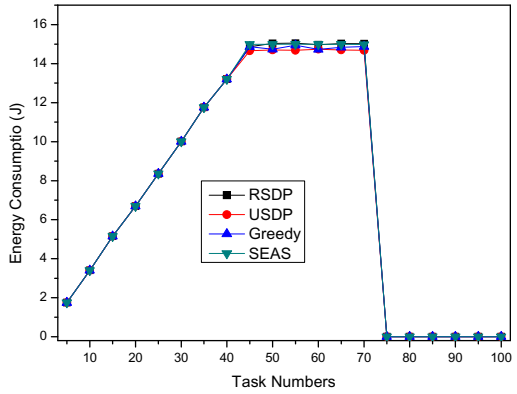

Fig. 13 Impact of task numbers on energy expenditure

The results of total security risk and energy consumption are shown in Fig. 12 and 13 respectively. From which, we can find that the real energy consumption of all algorithms have the same value when the task number is in range of (5, 45). Since 15 Joule energy budget can provide these tasks with highest security levels, the security risks of all algorithms are the same when tasks number is in range of (5, 45), as in Fig. 12. But it is not schedulable when there are more than 80 tasks, seeing Fig. 13. The reason is that 15 J cannot support these tasks even with minimal security level assignments. This is also why there is no risk in Fig. 12 when tasks number more than 75. When tasks number are in (50, 75), the energy budget is between the minimal and maximal energy consumption of these tasks. Hence, the proposed algorithm makes the most suitable choices for each task to find the corresponding security level. In the range of (50, 75), RSDP and USDP have the same pattern as the first three experiments, which means RSDP is also the best approach to get the lowest security risk and can guarantee the energy slack ratio. The average energy error ratios of all algorithms are listed in Table III. Clearly, the error ratio of RSDP is closest to zero among these algorithms.

E *Simulation conclusion*

Based on the detailed analysis of the former

simulations, we have following conclusions. 1）RSDP can always get the minimal security risk, while the risks of Greedy and SEAS are much higher, which means both of Greedy and SEAS are not competent for security risk-driven optimization problem. 2) RSDP is a polynomial complexity algorithm, which can get good solution with low time overhead. 3) RSDP has the same time complexity as USDP, but it can achieve less risk and statistically satisfy the given constraint with zero error. 4) Clearly, our randomly scaling policy based dynamic programming mechanism can be easily used for other constrained optimization applications which have statistical guaranteed requirements.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we investigate one general resource allocation problem in security- and energy-critical embedded systems. The problem seeks to find an offline resource allocation policy such that the expected security risk against security threatens is minimized, and the total energy budget is met. This problem is a knapsack problem which can be easily proved to be a NP-hard problem, thus we propose an approximation algorithm to find the near-optimal solution within statistically guaranteed energy constraint. The proposed algorithm has low complexity in both execution time and memory space, which is very suitable for use in resource limited embedded systems. Finally, extensive experiments demonstrate the superiority of the proposed algorithm.

We are interested in two potential future works. The first is to consider the systems which have additional real-time requirement in our scheduling framework. The second is to consider security as a dimension of constraints (as the system design problem in [20]), and try to obtain the minimal energy cost of the whole system.

VII. ACKNOWLEDGEMENT

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

VIII. REFERENCES

[1] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *ACM Transactions on Embedded Computing Systems,* vol. 3, no. 3, pp. 461-491, 2004.

[2] T. Xie and X. Qin, "Scheduling security-critical real-time applications on clusters," *IEEE Transactions on Computers,* vol. 55, no. 7, pp. 864-879, 2006.

[3] N. Shankaran, N. Roy, D. C. Schmidt, X. D. Koutsoukos, Y. Chen, and C. Lu, "Design and performance evaluation of an adaptive resource management framework for distributed real-time and embedded systems," *EURASIP Journal on Embedded Systems,* 2008.

[4] L. Sha, T. Abdelzaher, K. E. Arzen, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-Time Systems,* vol. 28, no. 2, pp. 101-155, 2004.

[5] V. Chaturvedi and G. Quan, "Leakage conscious dvs scheduling for peak temperature minimization," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2011, pp. 135-140.

[6] X. Zhong and C. Z. Xu, "System-wide energy minimization for real-time tasks: lower bound and approximation," *ACM Transactions on Embedded Computing Systems,* vol. 7, no. 3, pp. 1-24, 2008.

[7] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: a first step towards software power minimization," in *Readings in hardware/software co-design*, 2001.

[8] A. Muttreja, A. Raghunathan, S. Ravi, and N. K. Jha, "Automated energy/performance macromodeling of embedded software," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 26, no. 3, pp. 542-552, 2007.

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

[10] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "A study of the energy consumption characteristics of cryptographic algorithms and security protocols," *IEEE Transactions on Mobile Computing,* vol. 5, no. 2, pp. 128-143, 2006.

[11] Z. Guo, W. Jiang, N. Sang, and Y. Ma, "Energy Measurement and Analysis of Security Algorithms for Embedded Systems," in *IEEE/ACM International Conference on Green Computing and Communications*: IEEE Computer Society, 2011, pp. 194-199.

[12] S. Song, K. Hwang, and Y. K. Kwok, "Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling," *IEEE Transactions on Computers,* vol. 55, no.6, pp. 703-719, 2006.

[13] T. Xie and X. Qin, "Performance evaluation of a new scheduling algorithm for distributed systems with security heterogeneity," *Journal of Parallel and Distributed Computing,* vol. 67, no.10, pp. 1067-1081, 2007.

[14] K. Jiang, P. Eles, and Z. Peng, "Optimization of message encryption for distributed embedded systems with real-time constraints," in *14th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS11)*, 2011, pp. 243-248.

[15] T. Xie and X. Qin, "Security-aware resource allocation for real-time parallel jobs on homogeneous and heterogeneous clusters," *IEEE Transactions on Parallel and Distributed Systems,* vol. 19, no.5, pp. 682-697, 2008.

[16] T. Xie and X. Qin, "Improving security for periodic tasks in embedded systems through scheduling," *ACM Transactions on Embedded Computing Systems* vol. 6, no.3, pp. 1-20, 2007.

[17] M. Lin, L. Xu, L. T. Yang, et al, "Static security optimization for real-time systems," *IEEE Transactions on Industrial Informatics,* vol. 5, no.1, pp. 22-37, 2009.

[18] B. Karabacak and I. Sogukpinar, "ISRAM: information security risk analysis method," *Computers & Security,* vol. 24, no.2, pp. 147-159, 2005.

[19] H. Kellerer and U. Pferschy, "Improved dynamic programming in connection with an FPTAS for the knapsack problem," *Journal of Combinatorial Optimization,* vol. 8, no.1, pp. 5-11, 2004.

[20] K. Jiang, P. Eles, and Z. Peng, "Co-Design Techniques for Distributed Real-Time Embedded Systems with Communication Security Constraints," in *Design Automation and Test in Europe (DATE2012)*, 2012, pp. 947-952.