When security affects schedulability of TSP systems: trade-offs observed by design space exploration

Ill-ham Atchadam, Laurent Lemarchand, Hai Nam Tran, Frank Singhoff, and Karim Bigou
Univ. of Brest, UMR 6285, Lab-STICC, F-29200 Brest, France
Email: {ill-ham.atchadam, laurent.lemarchand, hai-nam.tran, frank.singhoff, karim.bigou}@univ-brest.fr

***Abstract*—ARINC 653 introduces the concept of partition that allows time and space isolation in real-time avionic systems. Tasks are assigned to partitions according to various objective functions or constraints such as safety, performance, and security. Some of these objective functions may be conflicting as an improvement of one objective leads to a decrease of another. For example, improving safety by active redundancy may decrease performance. In this paper, we investigate the conflicting aspect between schedulability and security in Time and Space Partitioning (TSP) systems. Many researches have shown that enforcing the security of a system results in an overhead affecting its schedulability. We formulate a design space exploration (DSE) process with a meta-heuristic to explore solutions defined by the tasks to partitions assignment according to security requirements and timing constraints. Experiments are conducted with the Cheddar scheduling analyzer to characterize applications that are concerned by this conflicting issue and to evaluate the trade-offs between schedulability and security.**

Keywords—Real-time Systems, Scheduling and Security Analysis, Design Space Exploration

I. Introduction

ARINC 653 [1] introduces isolation in avionic systems by the concept of partition. Partitions are software units defined to ensure temporal and spatial isolation of applications. Software components are run by tasks assigned to partitions at design time. Partitions are run over operating systems providing a specific runtime that manages the partitions and the communications among them.

Assigning tasks to partitions can be made according to various objective functions or constraints related to safety, performance, and security. These objective functions may be conflicting, i.e. an improvement of one objective leads to a decrease of another. According to security, multi single-level security (MSLS) applications require the isolation of tasks based on their security level. This rule cannot be applied for example when the safety (active redundancy) is involved and the number of partitions is limited and fixed because the safety requires that different instances of the same tasks should not be assigned to the same partition. [2] shows an other example of conflict between schedulability and security on a multi-level system with different clearances users: some users have low security level requests on a part of the system while others need a higher security level on other parts. All user requests have different real-time requirements which raises a schedulability and security trade-off. Today such trade-offs are resolved based on application-specific knowledge and the decision of the designer.

[***Problem statement***] Assigning tasks to partitions to enforce schedulability and security is an NP-hard combinatorial problem with 2 conflicting objective functions. Even with few partitions and tasks, it raises a combinatorial explosion as illustrated in [3], with only 4 partitions, each containing 16 tasks. It leads to numerous partitioning options which may have an impact on the schedulability.

In order to guarantee security, encryption and hashing of data sent through risky communications are examples of solutions. Encryption intervenes to guarantee information confidentiality while hashing helps to guarantee the integrity of data. However, they lead to additional costs which may imply violations of timing constraints. Thus, the conflict between security and schedulability motivates the interest of adopting an optimization method in order to explore possible partitioning.

[***Contribution***] In this paper, we investigate the combinatorial problem associated with the tasks to partitions assignment and the optimization problem between schedulability and security of Time and Space Partitioning (TSP) systems. In [4], we have proposed a method and a tool for the design space exploration (DSE) of real-time systems with the Pareto Archive Evolutionary Strategy (PAES) [5]. In this paper, we show how such meta-heuristic can be used to compute trade-offs between security and schedulability in TSP systems by automating the tasks to partitions assignment. For this purpose, we formalized ARINC 653 tasks to partitions assignment for the PAES meta-heuristic. We extended the Cheddar [6] scheduling analyzer with such partition assignment and several experiments were run to show how security affects schedulability.

For more flexibility, we consider that a subset of critical tasks can miss their deadlines and a subset of critical communications can remain unprotected to minimize the security overhead. These tasks and communications are called low in the sequel. We also consider that resources such as processors and partitions are fixed.

The remainder of this paper is organized as follows. Section II outlines the background and the assumptions taken in our work. Section III depicts our formulation of the DSE problem. Section IV presents experiments to evaluate our approach and to show how security affects schedulability. Finally, section V discusses related work and Section VI concludes the article.

II. Background

In this section, we introduce the background required to understand our approach and the assumptions we take.

A. System model and assumptions

We consider TSP systems such as ARINC 653. ARINC 653 is an avionic standard to ensure time and space isolation through partitioning. We assume r partitions $(P_1, ..., P_r)$ and n periodic synchronous tasks $(\tau_1, ..., \tau_n)$, running on a single processor. Since all tasks and partitions are run on the same processor, we assume no network delay/overhead. With TSP systems, there are 2 levels of scheduling. Partitions are scheduled offline and executed during a cyclic interval called a major time frame. Each partition contains at least one task and is executed at a fixed time frame. Tasks inside each partition are scheduled by a fixed-priority preemptive scheduling policy.

A task τ_i is defined by a sextuple: (C_i, T_i, D_i, IL_i, CL_i, CI_i). C_i, called the capacity, denotes the worst case execution time of τ_i. A task τ_i makes its initial request at time 0, is released periodically every T_i units of time, and must complete before D_i units of time. IL_i and CL_i are respectively integrity (Low, $Medium$ or $High$) and confidentiality ($Unclassified$, $Secret$ or Top_secret) levels of task τ_i. Each task is characterized by a criticality level CI_i ($High$ or Low) according to its degree of tolerance of timing constraint violations.

Intra-partition communications between tasks assigned in the same partition are implemented by ARINC buffer or blackboard. Inter-partition communications are handled by a hypervisor and are implemented by ARINC 653 queuing or sampling ports. Both mechanisms introduce different overheads that are considered in our approach.

B. Security: confidentiality and integrity

We focus on confidentiality and integrity in order to secure a system. Confidentiality ensures that data can only be accessed by authorized actors by means of encryption. Only actors who possess the appropriate key, can decrypt and understand the information. Integrity prevents data from unauthorized modifications. For this purpose, there are hash functions that use mathematical algorithms to transform the data into a hash value. The sender transmits the data with its hash value; then on receipt, the receiver will hash the data and compare its hash value with the one computed by the sender.

In the context of multi-security-level systems, there are two well know security models that define some rules for security analysis. Bell-La Padula (BLP) [7] is intended for confidentiality analysis and based on the No read up-No write down principle. It specifies that a subject at a given confidentiality level is forbidden to read data tagged with a higher confidentiality level. It also cannot write information to a lower confidentiality level. Biba [8] addressed integrity through the No read down-No write up concept. A subject at a given integrity level is forbidden to read data from a lower integrity level and to write data to a higher integrity level.

Our approach of securing a system consists of 2 steps. First, BLP and Biba are used to evaluate the communications in a TSP system, and to identify those that are security violations. Second, those communications that do not respect confidentiality or integrity rules are mitigated by adding security functions. Adding a security function to a task increases its capacity by the capacity of the security function. For communications that violate BLP's rules, we propose to add to the sending (resp. receiving) task an encryption (resp. decryption) function. We assumed a worst-case situation where the encryption key is set up at each task release time. So a function that represents the key set up is added to both sending and receiving tasks. For communications that violate Biba's rules, we add a hash function to the sending and receiving tasks.

The execution times of all the security functions are assumed according to the crypto++ benchmark [9]. We adapted the values of the benchmark to the data size of each considered application and to our CPU frequency. For our experiments, we adopted the Blowfish [10] encryption algorithm and the SHA-256 [11] hash function.

C. Design space exploration and multi-objective optimization

Multi-objective optimization problems (MOOP) [12] require the simultaneous optimization of more than one objective. In some cases, it is almost impossible to optimize one objective without deteriorating another one, i.e. no single solution optimizes simultaneously all the objectives. Then, for DSE, MOOP techniques provide a set of solutions that represent trade-offs between all the design objectives.

Solutions are compared based on the Pareto dominance concept [5] that stipulates that a solution s_1 dominates a solution s_2, if and only if s_1 is better than s_2 for at least one objective; and for any objective, s_1 is not worse than s_2. So s_1 is the dominant (or non-dominated) while s_2 is the dominated. Non dominated solutions are considered as good trade-offs and constitute the Pareto set.

A way of computing the Pareto set (or Pareto front) is to use an exhaustive method by enumerating all possible solutions, evaluating them according to defined objectives, and discarding dominated ones. Unfortunately, it becomes untractable for large-scale problems, because of the combinatorial explosion of the number of solutions to be evaluated. Multi-Objective Evolutionary Algorithms (MOEA) such as PAES have been developed to solve MOOP by finding a set of solutions close to the Pareto front. Inspired by nature, these meta-heuristics represent solutions in chromosomal form and transform them to explore the search space in the neighborhood of corresponding solutions.

In the next section, we detail our DSE approach to enforce temporal and spatial separation under timing constraints and security requirements in TSP systems.

III. Design space exploration formulation

We adopted PAES [5], an MOEA metaheuristic adapted to DSE problems with multiple and conflicting objectives. The algorithm starts with an initial solution (first current solution) contained in an archive. At each iteration of PAES, a new candidate solution is generated from the current one by using a random mutation adapted to the problem. To manipulate solutions especially in the mutation process, the solutions have to be encoded in a chromosomal representation. The algorithm finishes after a prefixed number of iterations and returns an archive that stores the best non dominated candidate solutions that optimize the objectives functions. We now present the specific functions developed to resolve the optimization problem we address.

Tasks	**Write access violation**	**Read access violation**
Confidentiality (security constraints)	Top_secret → Unclassified Secret → Unclassified	Unclassified → Top_secret Unclassified → Secret
Integrity (security constraints)	Low → Medium Low → High	Medium → Low High → Low
Confidentiality (objective functions)	Top_secret → Secret	Secret → Top_secret
Integrity (objective functions)	Medium → High	High → Medium

TABLE I: Communications concerned by security objective functions or constraints

1) Objective functions and constraints: Objective functions are the fitness functions that have to be optimized while constraints are conditions that validate or invalidate a solution.

Our optimization goals rely on 2 major domains: security and schedulability. According to schedulability, we assume that a design should be automatically rejected if one of its high critical tasks does not respect its deadline. This hypothesis represents our schedulability constraints. Other tasks, i.e. with a low criticality level, may be tolerated to miss their deadlines. The number of missed deadlines of low critical tasks can be used as an objective function reflecting the quality of the schedule. We note this function as: $F1 = \#missed\_deadlines$. The number of missed deadlines represents the number of low critical tasks that have the worst case response time higher than their deadline. To assess such a metric, we compute the scheduling simulation of the task set on the feasibility interval [13] with Cheddar. The entry point of Cheddar is a model composed of partitions, tasks, communications between tasks.

The second aspect of our problem is security. As a reminder, we perform security analysis by using BLP and Biba's security models. We consider as security constraints the fact that no task with a confidentiality level higher than $Unclassified$ (resp. integrity level higher than Low) should communicate with another one with $Unclassified$ confidentiality (resp. Low integrity) level. As an example, in a design, if a task tagged $Secret$ ($Medium$) communicates with another one tagged $Unclassified$ (Low), then this design will be automatically rejected. The communications forbidden by security constraints are represented in the 2^{nd} and 3^{rd} rows in Table I. Security-oriented objective functions rely on the remaining communications, i.e. those eventually violating security rules while respecting security constraints. They are represented in the 4^{th} and 5^{th} rows of Table I.

We characterize security with 2 objective functions. First, the number of confidentiality rule violations that represents the number of communications that violate BLP's rule in a TSP system ($F2 = \#Bell\_violations$). Second, the number of integrity rule violations which is the number of communications that violate Biba's rules in a TSP system ($F3 = \#Biba\_violations$). Such metrics are assessed by Cheddar in which both Biba and BLP analysis have been implemented. In order to optimize schedulability and security, we have to minimize the number of low critical tasks that miss their deadlines and also to minimize the number of security (confidentiality and integrity) rule violations.

2) Encoding of solutions: To formulate our optimization problem, we propose a chromosomal representation that will define our design solutions and make them manipulable. Each chromosome is defined as a vector with (n+m) positions where n represents the number of tasks and m represents the number

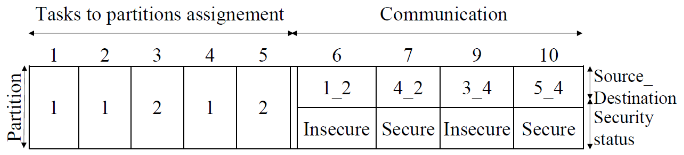


Fig. 1: Example of a normalized chromosome

of communications within the TSP system. The first part of the chromosome models the assignment of tasks to partitions. $chrom[i] = p$ $(1 \le i \le n)$ means that the $i^{th} task$ in the TSP system is assigned to the p^{th} partition.

The second part of the chromosome concerns communications in the TSP system. $chrom[k]$, with $(n+1) \le k \le (n+m)$, represents the k^{th} communication and is made of 2 elements. The first one specifies the task source that initiates the k^{th} communication and the task sink at the other extremity of the communication (e.g. value 1_2: indicates a communication from τ_1 to τ_2). The second element holds the security status of the communication, showing if there are some added security functions (e.g. encryption or/and hashing functions). Two possible values are secure or insecure. Each security function is exclusive to a communication.

We noticed that for the same configuration of tasks to partitions assignment, there are several possible representations. This can be illustrated by a TSP system of 2 tasks (τ_1 and τ_2); each assigned to different partitions (P_1 and P_2). The first possibility is that τ_1 is assigned to P_1 and τ_2 assigned P_2. The other possibility is that τ_1 is assigned to P_2 and τ_2 is assigned to P_1. So for unicity, we opted for a normalization of the chromosome part that represents tasks to partitions assignment by always assigning the task τ_1 to the partition P_1 and the task τ_2 to the partition P_2 if and only if the tasks τ_1 and τ_2 are not supposed to be in the same partition.

Fig. 1 illustrates a normalized chromosome with 5 tasks assigned to 2 partitions and 4 communications. The 6^th position specifies a communication from task τ_1 to τ_2 without calling any security function. The 7^th position reveals that task τ_4 sends data to task τ_2 by using some security functions.

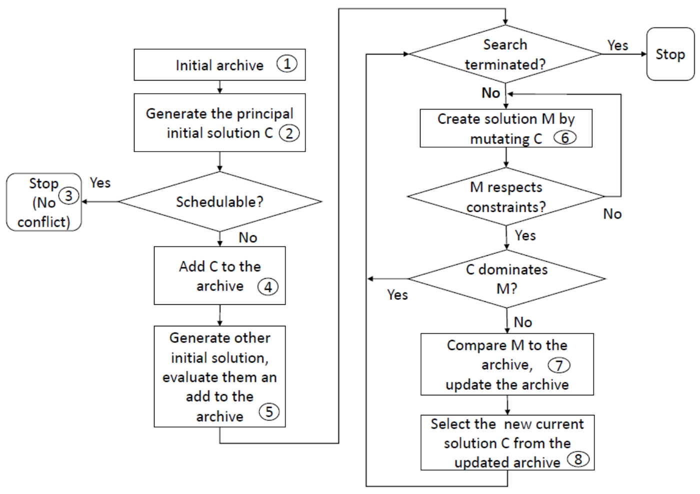
3) Mutation operator: In this section, we present the mutation operator used to generate new solutions in our DSE process based on PAES (step 6 in Fig. 2). We choose a random mutation splitted in 2 parts. The first one concerns tasks to partitions assignment while the second is dedicated to communications. For the first slice, we choose a random task $i (1 \le i \le n)$ and a random partition $p (1 \le p \le r)$. If the task τ_i is not in the partition P_p, then we now assign the task τ_i to the partition P_p. Changing tasks' partition might have an impact on schedulability.

For the second slice of the chromosome, we also choose a random communication $k((n+1) \le k \le (n+m))$ in the set of communications that violate security rules. Then if there are already some added security functions, we remove them. Otherwise, we resolve security issues by adding security functions.

On any mutated solution, we perform schedulability and security analysis with Cheddar. If the solution does not respect schedulability and/or security constraints defined in Section III-1, it is automatically rejected and we proceed with another mutation. Our mutation operator is sketched in Algorithm 1.

Algorithm 1: Mutation algorithm

```
1  Input: A chromosome that represents a solution
   Output: A mutated solution
2  while solution is not mutated do
3      Choose a random $\tau_i$ and a random partition $P_j$
4      if $\tau_i$ is not assigned to $P_j$ then
5          Assign $\tau_i$ to $P_j$
6      Choose a random communication $k$ that may
       violate the security rules
7      if $k$ has no encryption or hashing function then
8          Secure $k$ with encryption and/or hashing
9      else
10         Remove security functions from $k$
11     if the new solution does not respect the scheduling
       and security constraints then
12         Reject the mutated solution
13         Proceed with another mutation
14     else
15         Return the mutated solution
```



Initial archive ①
Generate the principal initial solution C ②
Schedulable?
Yes
Stop (No conflict) ③
No
Add C to the archive ④
Generate other initial solution, evaluate them an add to the archive ⑤
Search terminated?
Yes
Stop
No
Create solution M by mutating C ⑥
M respects constraints?
No
Yes
C dominates M?
Yes
No
Compare M to the archive, ⑦ update the archive
Select the new current solution C from the updated archive ⑧

Fig. 2: Adaptation of PAES

4) Initial solutions and archiving process: PAES uses a local evolution on a single current solution and keeps good found solutions into an archive. At each iteration, it mutates the current solution. The mutated solution is compared to all elements in the archive and eventually becomes the current solution.

The initial current solution we choose is security-oriented: we build a solution that optimizes the security by resolving all the security issues while using one single partition for all the tasks. If the scheduling analysis of that solution reveals that there is no missed deadline, then the optimal solution is found (perfect for both schedulability and security aspects) and DSE (step 3 in Fig. 2) is not useful. To make our PAES method faster and to favor diversity of solutions, we also add extra non dominated solutions in the archive(step 5 in Fig. 2), using various strategies (single or balanced partitions, solving all security issues or not, building MSLS partitioning based on integrity or confidentiality level, etc.).

IV. Test cases and Evaluation

In this section, we evaluate our optimization method on several applications that raise trade-offs between security and schedulability. Each solution is evaluated on schedulability and security. Schedulability is evaluated by the number of tasks that missed their deadlines. Security is evaluated by confidentiality and integrity through the number of communications that violate the BLP's and Biba's security rules.

The purpose of our experiments is both to define situations to which our proposed DSE approach is suited and also to evaluate the quality of the trade-offs provided by the method. Experiment 1 shows a counterexample, where our method is able to show that no conflict arises between objectives. On the contrary, experiment 2 shows a case for which our approach is able to detect conflicts and to propose trade-offs. Experiment 3 demonstrates the effectiveness of the approach for defining good trade-offs between objectives. Finally, experiment 4 evaluates the accuracy of our method by comparing it with optimal solutions.

For all experiments, we assumed that all partitions are identical with the same duration. We also supposed that the tasks are periodically harmonic meaning that for any pair of tasks (τ_i, τ_j) in the task set, τ_i mod τ_j = 0 or τ_j mod τ_i = 0 [14]. We assumed that criticality and security levels of tasks are fixed independently by the designer as inputs of the exploration tool. We also consider the worst case situation by assuming that applications' data sizes are fixed. To evaluate the impact of communication overheads, we performed experiments with and without overhead. We assumed an overhead of 10 us (resp. 280 us) for an intra-partition (resp. inter-partition) communication. We have chosen overhead values that impact the scheduling results for our test cases, i.e. impacts the search space. For lower effective times, this overhead does not need to be taken into account.

A. Flight controller application

This experiment shows that for some applications, conflicts between security and schedulability may not occur. This means that improving security would not necessarily impact the schedulability. Then, there is no need to proceed with a DSE for security.

1) Conditions of experiments: We conduct this experiment with ROSACE [15], a real-time benchmark that describes a longitudinal and multi-periodic flight controller. It is composed of a set of periodic tasks, a total processor utilization of 29% and on average a small size data of 8 bytes. The ROSACE parameters are summarized in table II.

Task parameters are taken from the benchmark in [15]. We fixed the security parameter (confidentiality and integrity levels) values to fit with a worst-case situation: they are fixed to maximize the number of security violations in the application. If we can show that for a ROSACE application with a very high number of security violations, resolving all the security issues could not impact the application schedulability, then we can conclude that for a more realistic ROSACE application with few security violations, no conflict between schedulability and security will exist.

Tasks	C_i	T_i	CL_i	IL_i
Experiment 1: ROSACE				
Aircraft Dynamics	200	5000	Secret	Medium
Va_c, H_c	500	20000	Top_secret	Medium
H_Filter,Az_Filter,Vz_Filter, Q_Filter,Va_Filter	100	10000	Top_secret	High
Altitude_hold,Vz_control, Va_control	100	20000	Secret	Medium
Delta_ec, Delta_thc	500	20000	Secret	High
Engine, Elevator	100	5000	Top_secret	Medium
Experiment 2: JPEG				
Matrix transpose	41	20000	Top_secret	High
Color space conversion	41	20000	Secret	Medium
Wrapper 1, Wrapper 2	625	20000	Secret	Medium
Quantization	270	20000	Top_secret	Medium
Encoder	760	20000	Secret	High
Memory Read/write	41	20000	Secret	Medium
Experiment 3: Autopilot				
Data collection	Uunifast	Uunifast	Top_secret	Medium
Control law computing	Uunifast	Uunifast	Secret	High
Actuator	Uunifast	Uunifast	Top_secret	High
Fault auditor	Uunifast	Uunifast	Secret	Medium
IFBIT	Uunifast	Uunifast	Top_secret	High

TABLE II: Case studies task parameters

For confidentiality and integrity, we adopted respectively a Blowfish algorithm for encryption and SHA-256 for hashing. By assuming that the frequency is 1.2 GHz and the data size is 8 bytes, encryption execution, refreshment encryption key and hash execution times are respectively 0.166 us, 88.83 us and 0.1 us. Those execution times are added to the C_i parameter of the ROSACE tasks. For this experiment, we fixed a maximum of 2 partitions.

2) Results: Our tool starts with an initial solution which is characterized by the resolution of all the security problems (first current solution of the PAES optimization process, see section III-4). The scheduling analysis of the resulting architecture shows that all the tasks met their deadlines. This is due to the fact that initially, ROSACE is characterized by a low total processor utilization of 29%. The method we propose therefore returns that it is not necessary to carry out a DSE for such an application. We note that the addition of the security tasks only increases the total processor utilization to 37%. We explained this result by the low overhead introduced by encryption and hash tasks because they are proportional to the data size of the considered application since ROSACE has small data size (8 bytes).

From this experiment, we conclude that the data size and the initial processor utilization of the application are some of the most important criteria that determine the necessity of the DSE. Our method can directly detect such cases during its initialization phase.

B. Flight controller and multimedia based application

We initiate this experiment first to illustrate the potential conflicts between security and schedulability objectives and thus to show the importance of implementing a DSE approach taking into account both objectives simultaneously using multiple objective optimization techniques such as PAES. Second, to prove its effectiveness, we apply our customized PAES method to find solutions that are good trade-offs between security and schedulability.

1) Conditions of experiments: Critical video applications are now deployed in many current aircrafts. We propose a case study composed of ROSACE and a JPEG multimedia application [16]. Multimedia applications are widely and are well-known for their very high data sizes. Our JPEG application is composed of the tasks mentioned in table II.

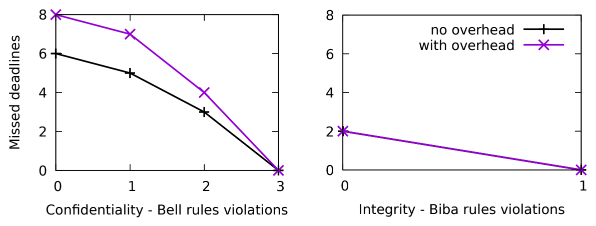


Fig. 3: Schedulability vs. security

For the ROSACE application, we kept the same scheduling parameters fixed in the previous experiment. Concerning the security parameters, we changed the values to significantly reduce the number of security violations compared to the previous experiment. The total processor utilization of this case study (ROSACE + JPEG tasks) is 41%. We adopted, as for the previous experiment, the Blowfish algorithm and the SHA-256 hash function. We assume that the data size is a 4CIF (704x576 pixels) for the JPEG. With 2 bytes per pixel, i.e. the data size is equal to 792 KB. The execution time of the encryption, the refreshment of the encryption key and the hash computations for the JPEG application are respectively 16834 us, 88.83 us and 10173 us. For this experiment, we fixed a maximum of 2 partitions. We proceed with the same process of resolving all the security issues (initial solution for PAES), but we observed that the resulting system is not schedulable (with a total processor utilization that exceeds 100%) and there are 22 tasks that missed their deadlines. So we have to proceed with a DSE to propose a set of solutions that optimize both security and schedulability.

2) Results: After 5000 iterations running during 1.5 hours, the PAES process provides an archive of 6 solutions. The set of points approximating the Pareto front (the values of the objectives functions of each solution in the archive) obtained by PAES is shown in Fig. 3, by couples of objectives.

For a solution represented in Fig. 3 (left) with a number of missed deadlines and a number of confidentiality violations, its values of integrity violations are the one in Fig. 3 (right) associated with the same number of missed deadlines.

In Fig. 3 (left), we observed that there is a solution with no security rule violation but with 6 missed deadlines which increased to 8 when we consider communication overheads. Before the DSE, by resolving all the security issues with all the tasks in the same partition, we obtain a number of missed deadlines equals to 22. We obtain the same result for solutions with no security issues, based on MSLS partitionning. The PAES method proposed a better solution in terms of schedulability by changing the tasks to partitions assignment while preserving security. The experiments also prove that it is possible to obtain a schedulable system by limiting the number of confidentiality issues resolved to 3 (Fig. 3 left) and the number of integrity issues resolved to 1 (Fig. 3 right), and that 2 intermediate trade-offs are also available.

These results show that in some cases, schedulability and security are conflicting objectives, with solutions leading them one by one to optimality and also with trade-offs solutions.

We can conclude that securing applications with large data involves the addition of security computations that may lead to jeopardize schedulability, hence the relevance of our approach.

C. *Synthetic generated architectures*

In the first experiment, we concluded that data size and initial processor utilization are important on the relevance of our approach. We confirmed in the second experiment IV-B that data size may impact its schedulability after its securing. To evaluate the impact of the processor utilization, in this section, we first, perform the PAES algorithm on different architectures, with a maximum of 2 partitions (as in IV-A and IV-B), by varying the total processor utilization of the tasks. Second, we conduct the same experiment, with a maximum of 4 partitions, to evaluate how complexity grows with the number of partitions. Both experiments have the same conditions except the maximal number of partitions.

1) Conditions of experiments: We generate task sets and we run PAES on each set. We generate task sets according to an autopilot system case study [17]. The autopilot system collects data from sensors and then sends commands via actuators to pilot the aircraft. The application is composed of 5 tasks: table II gives their security parameters. To apply our method to a large-scale problem, we run PAES with a set of three identical autopilot applications.

For these experiments, the other task parameters are synthetically generated. We adapted the Uunifast algorithm [18] to generate randomly task capacities according to an uniform distribution with a fixed number of tasks and a given total processor utilization. We guaranteed that the tasks generated are periodically harmonics by fixing a set of periods (each period for a task). Then, we generate with Uunifast each task processor utilization and run the PAES for different values of total processor utilization U (40%, 50%, 60%, 70%, 80%, 90%). We also assume data size of 16 KB. Therefore the execution time of the encryption, the refreshment of the encryption key and the hash computations are respectively 340 us, 88.83 us and 205.52 us.

2) Results of PAES with a maximum of 2 partitions: After running our PAES process, for each value of U (40%, 50%, 60%, 70%, 80%, 90%), the numbers of missed deadlines of the initial solution (tasks in a single partition and all security issues solved) are respectively (7, 7, 7, 8, 8, 12). For each processor utilization value, we run 5000 iterations of PAES and it took 1.4 hours. We can observe that the more the processor utilization increases, the higher number of missed deadlines for this initial solution we have. This enforces our affirmation of the necessity of DSE.

The set of points approximating the Pareto front (the values of the objective functions of each solution in the archive) obtained by PAES are shown in Fig. 4. Both plots show that with a high processor utilization and considering communications overheads, it becomes difficult to solve security issues without deteriorating the schedulability. As an example in this test case, to keep ensuring schedulability, for U=40%, we should tolerate 4 confidentiality and 2 integrity violations while for U=80%, we have to tolerate 6 confidentiality and 5 integrity violations.

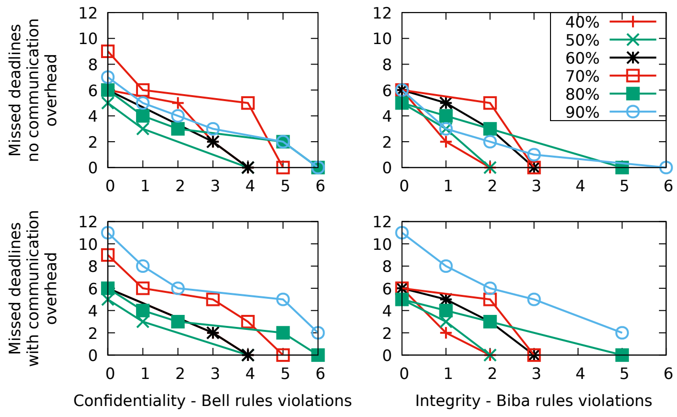


Fig. 4: Schedulability vs. security with and without inter-partition communication overhead, 2 partitions

3) Results of PAES with a maximum of 4 partitions: With 4 partitions, the processing of PAES provides a set of points approximating the Pareto front (the values of the objective functions of each solution in the archive) represented in Fig. 5. For 4 partitions, results are similar to those of the 2 partitions experiment, except that a zero confidentiality errors solution for U=50% and U=60% cases induces 2 more missed deadlines, and the same increase arises for integrity with U=50%. On the contrary, adding 2 partitions allows to decrease the number of missed deadlines in some cases (e.g. zero confidentiality defaults for U=80% and U=90%; zero integrity defaults for U=80%). Those variations are due to the way the scheduling of task/partition is realized, and could be investigated more in detail. This could be a extra degree of freedom for the DSE.

The size of the search space is much more larger for 4 partitions than for 2 partitions: from 1,048,576 when considering a maximal number of 2 partitions, it grows to 2,863,835,840 for 4 partitions. The size of the search space DS corresponds to the number of solutions of the addressed problem. It is computed with the number of tasks to partitions assignment DT and the number of communication parameter configurations DC. The former corresponds to the Stirling number of the second kind $S(n, k)$ [19], that is the number of possibilities to divide n tasks into k (non empty) partitions at most. According to our approach, to represent communications, each of m risky communications has a possibility of 2 values, thus $DC = 2^m$. With n tasks and m risky communications to be mapped into k (eventually empty) partitions, $DS = DT \cdot DC = (\sum_{q=1}^{k} S(n, q)) \cdot 2^m$.

Even if the size of the search space is high with 4 partitions, the optimization process was efficient enough to find comparatively good solutions in the 4 partitions case as compared to the 2 partitions experiment. The combinatorial explosion means that good solutions can be more difficult to exhibit, and that the optimization process has to be reinforced if a higher number of partitions is required. Another point is that, for the moment, partition isolation is not used in the model to reduce security computation. In terms of processor utilization, this 4 partitions experiment confirms the impact of processor utilization in our DSE approach whenever the maximum number of partitions we have.

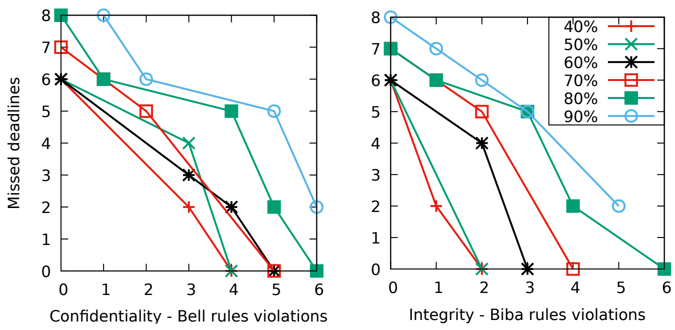


Fig. 5: Schedulability vs. security with inter-partition communication overhead, 4 partitions

In order to identify the limit of processor utilization when no DSE is required, we generated also task sets with U=20%. For such task sets, there is no conflict between schedulability and security. We can conclude that for an application that still has a margin in terms of processor utilization, it is possible to add security without jeopardizing schedulability. So the higher the processor utilization, the more the additional security computation may compromise the schedulability.

D. Evaluation of our PAES tool results

This experiment consists of validating the accuracy of our PAES method by comparing it to the exact Pareto front. For this purpose, we implemented an exhaustive search tool enumerating all the possibilities of solutions.

1) Conditions of experiments: We generate a system similar to the one used in section IV-C composed of 10 tasks from the autopilot case study. We limited the number of risky communication to three over eight communications and we assumed a maximum of 2 partitions. The task set is generated by the Uunifast algorithm for total processor utilization of 90%. We assume that data size is 16 KB which leads to execution time of encryption, refreshment of encryption and hash computation of 340 us, 88.83 us and 205.52 us.

2) PAES tool evalution results: With 10 tasks, 2 partitions and 3 risky communications, the exhaustive research generates a total of 4096 solutions. It took almost 1 hour to produce the exact Pareto front compared to the PAES run limited to 5000 iterations which took almost 2 hours to produce an approximate Pareto front. When the number of partitions increase to 15, the number of solutions rises to 131,072 solutions which may take theoretically almost 32 hours to provide the Pareto front. This shows that the exhaustive method may become quickly unmanageable. The increase of design space has no impact on the duration of PAES while the number of iterations is unchanged. The exact Pareto front set of points (Exhaustive method) and the set of points approximating it (our PAES method) are shown in Fig. 6.

We noticed a gap between both fronts even if they almost look alike. As an example, for a case with no integrity violation, the exhaustive method was able to provide a solution with a number of missed deadlines equals to 3 while the PAES method provides a solution with 4 missed deadlines. This can be easily explained as PAES provides an approximate result: each run of PAES may provide a better or worst solution depending on the generated solutions during the random mutation.

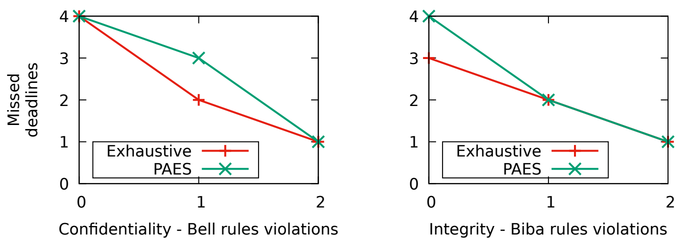


Fig. 6: Schedulability vs. security metrics with inter-partition communication overhead, 2 partitions (Exhaustive vs. PAES)

	[20]	[21]	[22]	[23]	[24]	[25]	[26]	[27]	[4]	[28]	Ours
RTS security	X	X	X	X	X	X	X			X	X
TSP								X			X
Schedulability optimization		X	X			X	X	X	X	X	X
Security optimization		X		X	X	X	X			X	X
Trade-off							X			X	X
MOEA/DSE									X	X	X

TABLE III: Related work

V. Related Work

In this section, we present proposals that integrate methods based on security in real-time systems, focusing on ARINC 653 standard or not, providing DSE and optimization or not. A summary is given in Table III.

Many works have been done to integrate security aspects in real-time systems such as [20]. Some of them after highlighting the conflicting aspect between security and schedulability, have proposed methods to optimize schedulability (resp. security) while guaranteeing as much as possible security (resp. schedulability). For example, [26] optimizes schedulability while partially fixing the security violations. In [22], a security manager is developed for a real-time database that consists of adapting the security levels of components to obtain a significant gain according to real-time performance. In [24], the authors focused first on protecting the real-time database from unauthorized accesses and then investigate the impact of this securing process on the performance of the system in terms of the number of missed deadlines.

To jointly optimize schedulability and security, some researchers had to change the parameters of initial architectures (e.g. security levels of tasks). [21] ensures security while optimizing schedulability by removing from the system any transaction that missed its deadline. Tao Xie and Xiao Qin [25] have developed a real-time scheduling algorithm that helps to improve security. The main goal of their algorithm is to enforce that timing constraints should always be respected while optimizing the security aspect by changing security levels of tasks. They set minimum values for the security levels and then increase them gradually to achieve optimized security. The above works have confirmed the conflict aspect between security and schedulability.

Several researches have been done to investigate tradeoffs in the context of real-time systems by adopting MOEA strategies. About the conflict between security and schedulability, in [28], the authors proposed HYDRA, a DSE for secure multi-core real-time systems based on the assignments of security tasks to cores by changing their parameters (e.g. period) to meet timing constraints and not to disturb the expected execution pattern of the tasks.

To conclude, many researchers have been interested in DSE and multi-objective optimization in the context of real-time systems and security. Fewer have worked on both optimizing security and schedulability, through DSE. Finally, as far as we know, none has considered TSP systems when optimizing security and schedulability through DSE.

VI. Conclusion

In this paper, we propose a DSE approach to compute task partitioning according to security and schedulability objective functions for TSP systems such as ARINC 653. We show that security and schedulability are conflicting objectives as the overhead introduced by security may negatively affect schedulability. The principal contribution of this paper consists in investigating the schedulability and security trade-offs by the mean of a DSE based on a multi-objective optimization.

We proposed a formulation with PAES and implemented it into Cheddar, an open-source scheduling analyzer. We conducted 4 experiments with 3 different applications. From experiment 1, we observe that enforcing security does not raise a schedulability issue when an application has a low processor utilization factor and exchanges small data. Thus, there is no need for DSE in such a situation. This should be typical for many control-command applications, e.g. ARINC 629 [29] characterized by frames of 20 bits. Experiment 2 illustrates an example requiring DSE. It confirms the relevance of data size in the conflict between security and schedulability. It also shows that the overhead of encryption can be partially compensated by changing the tasks to partitions assignment. Experiment 3 shows that there is no multi-objective problem when the processor utilization is less or equal than 20%. Our DSE approach is then able to identify the value of the processor utilization from which schedulability and security trade-offs do not have to be investigated by the designer. It confirms that processor utilization is a key parameter that should be considered to decide if DSE has to be performed or not. It also shows that the number of partitions has a high impact on the size of the search space. Finally, experiment 4 compares the efficiency of the meta-heuristic formulation with the optimal solution. In this paper we assume all partitions are run on the same processor. For future work, we aim to extend our approach with network communication overheads.

References

[1] A. E. E. Committee, "Arinc 653: Avionics application software standard interface, supplement 1," 1997.

[2] I. Greenberg, P. Boucher, R. Clark, E. Jensen, T. Lunt, P. Neuman, and D. Wells, "The secure alpha study (final summary report)," *CS Lab, SRI International*, 1993.

[3] C. Ananda, S. Nair, and G. Mainak, "Arinc 653 api and its application–an insight into avionics system case study," *Defence Science Journal*, vol. 63, no. 2, pp. 223–229, 2013.

[4] R. Bouaziz, L. Lemarchand, F. Singhoff, B. Zalila, and M. Jmaiel, "Multi-objective design exploration approach for ravenscar real-time systems," *Real-Time Systems*, vol. 54, no. 2, pp. 424–483, 2018.

[5] J. Knowles and D. Corne, "The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation," in *Congress on Evolutionary Computation*, vol. 1, 1999, pp. 98–105.

[6] F. Singhoff, J. Legrand, L. Nana, and L. Marcé, "Cheddar: a flexible real time scheduling framework," *ACM SIGAda Ada Letters*, vol. 24, no. 4, pp. 1–8, 2004.

[7] D. E. Bell and L. J. La Padula, "Secure computer system: Unified exposition and multics interpretation," MITRE Corp., Tech. Rep., 1976.

[8] K. J. Biba, "Integrity considerations for secure computer systems," MITRE Corp., Tech. Rep., 1977.

[9] W. Dai, "Crypto++ 5.6.0 benchmarks." [Online]. Available: https://www.cryptopp.com/benchmarks-amd64.html

[10] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *International Workshop on Fast Software Encryption*. Springer, 1993, pp. 191–204.

[11] W. Sun, H. Guo, H. He, and Z. Dai, "Design and optimized implementation of the sha-2 (256, 384, 512) hash algorithms," in *7th International Conference on ASIC*. IEEE, 2007, pp. 858–861.

[12] C. A. C. Coello and C. S. P. Zacatenco, "Twenty years of evolutionary multi-objective optimization: A historical view of the field," *CINVESTAV-IPN Evolutionary Computing Group*, 2005.

[13] J. Goossens, E. Grolleau, and L. Cucu-Grosjean, "Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms," *Real-time systems*, vol. 52, no. 6, pp. 808–832, 2016.

[14] C.-C. Han and H.-Y. Tyan, "A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms," in *Real-Time Systems Symposium*. IEEE, 1997, pp. 36–45.

[15] C. Pagetti, D. Saussié, R. Gratia, E. Noulard, and P. Siron, "The rosace case study: From simulink specification to multi/many-core execution," in *19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2014, pp. 309–318.

[16] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. 18–35, 1992.

[17] K. Zhang, J. Wu, C. Liu, S. S. Ali, and J. Ren, "Behavior modeling on arinc653 to support the temporal verification of conformed application design," *IEEE Access*, vol. 7, pp. 23 852–23 863, 2019.

[18] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.

[19] B. C. Rennie and A. J. Dobson, "On stirling numbers of the second kind," *Journal of Combinatorial Theory*, vol. 7, no. 2, pp. 116–121, 1969.

[20] 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.

[21] Q. N. Ahmed and S. V. Vrbsky, "Maintaining security in firm real-time database systems," in *14th Annual Computer Security Applications Conference (Cat. No. 98EX217)*. IEEE, 1998, pp. 83–90.

[22] S. H. Son, R. Zimmerman, and J. Hansson, "An adaptable security manager for real-time transactions," in *12th Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, 2000, pp. 63–70.

[23] T. Xie and X. Qin, "Enhancing security of real-time applications on grids through dynamic scheduling," in *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 2005, pp. 219–237.

[24] B. George and J. Haritsa, "Secure transaction processing in firm real-time database systems," *ACM SIGMOD Record*, vol. 26, no. 2, pp. 462–473, 1997.

[25] T. Xie and X. Qin, "Improving security for periodic tasks in embedded systems through scheduling," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 3, p. 20, 2007.

[26] S. H. Son, R. Mukkamala, and R. David, "Integrating security and real-time requirements using covert channel capacity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 6, pp. 865–879, 2000.

[27] Q. Xue, Y. Zhu, Y. Wang, K. Mao, H. Wu, M. Li, Y. Mao, and J. Hou, "A scheduling scheme of task allocation in real time multiple-partition embedded avionic," in *IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2017, pp. 41–46.

[28] M. Hasan, S. Mohan, R. Pellizzoni, and R. B. Bobba, "A design-space exploration for allocating security tasks in multicore real-time systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 225–230.

[29] A. Kornecki, J. Zalewski, J. Sosnowski, and D. Trawczynski, "A study on avionics and automotive databus safety evaluation," *Archives of Transport*, vol. 17, no. 3/4, pp. 107–131, 2005.