sensors

MDPI

Article

Security-Related Hardware Cost Optimization for CAN FD-Based Automotive Cyber-Physical Systems

Yong Xie [1,*], Yili Guo [2], Sheng Yang [2], Jian Zhou [1] and Xiaobai Chen [1]

[1] School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China; zhoujian@njupt.edu.cn (J.Z.); chenxb86@njupt.edu.cn (X.C.)
[2] School of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China; 1922032011@stu.xmut.edu.cn (Y.G.); 1722011058@stu.xmut.edu.cn (S.Y.)
* Correspondence: yongxie@njupt.edu.cn

Abstract: The introduction of various networks into automotive cyber-physical systems (ACPS) brings great challenges on security protection of ACPS functions, the auto industry recommends to adopt the hardware security module (HSM)-based multicore ECU to secure in-vehicle networks while meeting the delay constraint. However, this approach incurs significant hardware cost. Consequently, this paper aims to reduce security enhancing-related hardware cost by proposing two efficient design space exploration (DSE) algorithms, namely, stepwise decreasing-based heuristic algorithm (SDH) and interference balancing-based heuristic algorithm (IBH), which explore the task assignment, task scheduling, and message scheduling to minimize the number of required HSMs. Experiments on both synthetical and real data sets show that the proposed SDH and IBH are superior than state-of-the-art algorithm, and the advantage of SDH and IBH becomes more obvious as the increase about the percentage of security-critical tasks. For synthetic data sets, the hardware cost can be reduced by 61.4% and 45.6% averagely for IBH and SDH, respectively; for real data sets, the hardware cost can be reduced by 64.3% and 54.4% on average for IBH and SDH, respectively. Furthermore, IBH is better than SDH in most cases, and the runtime of IBH is two or three orders of magnitude smaller than SDH and state-of-the-art algorithm.

Keywords: automotive cyber-physical systems; hardware cost; design space exploration algorithm; cyber security; CAN FD

Citation: Xie, Y.; Guo, Y.; Yang, S.; Zhou, J.; Chen, X. Security-Related Hardware Cost Optimization for CAN FD-Based Automotive Cyber-Physical Systems. *Sensors* **2021**, *21*, 6807. https://doi.org/10.3390/s21206807

Academic Editors: Tian Wang, Geyong Min and Md Zakirul Alam Bhuiyan

Received: 2 September 2021
Accepted: 11 October 2021
Published: 13 October 2021

1. Introduction

1.1. Background and Motivations

As security is not considered in in-vehicle networks' specification, the employment of various network interfaces (wireless or wired) in automobiles poses great cyber-security challenges to the safety of automotive cyber-physical systems (ACPS). For example, CAN is the most widely employed in-vehicle network in automobiles, but it is vulnerable to replay attack, masquerade attack, and DoS attack [1]. CAN with flexible data-rate (CAN FD) is proposed in 2011 and viewed as the next generation of CAN technology, which can elevate the bandwidth of the transmission phase to 8 Mbps, but the security vulnerabilities of CAN are not resolved [2]. Many kinds of cyber attacks have been identified in in-vehicle networks such as CAN, CAN FD, and FlexRay, and the potential security vulnerabilities have lead to car recall event [3]. As a result, automotive stakeholders are trying to employ different kinds of security enhancing mechanisms to safeguard in-vehicle networks, such as adding message authentication code (MAC) into message for integrity and availability [4,5] (recommended and specified in AUTOSAR SecOS specification [6]), message encryption for confidentiality [7], intrusion detection, and message ID obfuscation [8]. The authors of [9] give an extensive survey about the security vulnerabilities and proposed security protection mechanisms of in-vehicle networks. However, as it is too time- and memory-consuming

to implement the cryptographic algorithms in software, processors with a dedicated security coprocessor such as hardware security module (HSM) are recommended by the auto industry to be used in the new generation of ECUs [10]. Main automotive chip makers like Infineon and Renesas are already producing multicore processors with HSM, such as Infineon Aurix and Renesas R-Car. The HSM-based security-enhancing approach can reduce the delay overhead to an acceptable level. For example, as the real implementation on Aurix [11] shows that software-based implementation of AES128 consumes 1209 µs, while the HSM-based implementation only needs 62 µs. HSM offer several advantages, such as cryptographic engine (hardware implementation of symmetric/asymmetric cryptographic algorithms and hash functions, such as AES128, ECC256, SHA256, and others), secure key storage, secure log, and others, which would allows the ECU's host core to devote its full power to the other tasks, and offers ECU manufacturers and automakers a powerful plug-and-play security solution that can be easily adapted to their own security requirements. World-renown automotive suppliers are providing HSM-based security solutions that combines Aurix and HSM, such as ESCRYPT's CycurHSM and Elektrobit's zentur HSM.

Although HSM frees the host core from computing-intensive security tasks, it brings considerable hardware cost. Taking Aurix processores from Infineon as an example, the Aurix TC299TP128F300N and Aurix TC299T128F300S have similar performance (in terms of CPU and memory), but the price of Aurix TC299TP128F300N (about 47 dollars) is higher than that of the Aurix TC299T128F300S (about 41 dollars) as it has the HSM, thus the adding of HSM increases the hardware cost by about 15% [12]. However, given that automobiles are mass-produced consumer products which are very cost sensitive, it is not cost-efficient to add HSM in all ECUs. According to the authors of [13], Toyota and Honda only integrate HSM-based security protection mechanisms into a few ECUs implementing safety-critical functions, such as engine ECU, brake ECU, and steering ECU. As a result, it is important to reduce the HSM introduced hardware cost by minimizing the number of ECUs equipped with HSM. In this paper, we formulate an optimization problem to minimize the number of HSMs required for a given ACPS, where both the task mapping, task scheduling and message scheduling are explored subject to both deadline and security constraints.

1.2. Contributions

In this paper, we observe that HSM-based security protection mechanisms introduce considerable hardware cost, which poses great challenges to cost management of car manufacturers. Thus, we formulate a new design space exploration (DSE) problem to reduce the hardware cost for security-enhanced ACPS. To the best of our knowledge, this is the first time that HSM introduced hardware cost is integrated into the DSE of CAN FD-based ACPS. The main contributions of this paper are as follows. (1) We provide a stepwise decreasing-based heuristic DSE algorithm (SDH) which reduces the design space into smaller one based on the decreasing number of HSMs. (2) As the performance of SDH deteriorates as the searching space expands, we provide another interference-balancing based heuristic algorithm (IBH) which can get an equal or even better result with much shorter runtime by comparing with SDH. (3) By comparing with the state-of-the-art algorithm based on both synthetic and real data sets, the efficiency of the proposed algorithms is verified.

This paper is organized as follows. Section 2 surveys the related work. Section 3 presents the system models and key assumptions. In Section 4, the details about the SDH and IBH algorithms are given for the hardware cost optimization problem. Section 5 presents the experimental results, and the paper is concluded in Section 6.

2. Related Work

Cost optimization is one of the key objectives for DSE of embedded systems. In [14], the authors first transfer reliability goal of the application to that of each task, and then resource cost is minimized by heuristically assigning tasks to the processors. In [15,16],

the authors propose to minimize the development cost of embedded systems with genetic algorithm-based and tabu search-based heuristics, respectively. In [17], the authors propose a two-stage solution for function safety risk assessment and development cost optimization of software-defined vehicles, where both the reliability and real-time requirements are considered. However, the above-mentioned works consider resource cost and development cost, and they employ different cost models. Hardware cost is an essential part of the cost of embedded systems, especially for mass-produced ACPS. In [18], the authors present two heuristic algorithms to minimize hardware cost for parallel embedded systems, where both the functional safety and real-time requirements are considered. In [19], the authors combine genetic algorithm and simulated annealing to reduce the hardware cost and energy consumption of embedded products while satisfying the hard real-time and reliability requirements of safety-critical applications. In [20], the authors present three price performance-driven heuristic algorithms for hardware cost minimization of embedded systems, which consider both real-time and reliability requirement. In [21], the authors present a multi-population genetic algorithm towards optimizing both operation time and the number of required processing units for distributed real-time systems. In [22], the authors try to reduce the hardware cost by minimize the number of required processors to schedule an application, where considerably memory requirements and application latency are reduced by comparing with related approaches while meeting the same throughput constraint. In [23], the authors shows how to minimize the number of required processors for feasible running the parallel real-time tasks. However, our work considers to minimize security-enhancing related hardware cost of ACPS by minimizing the number of HSMs, where both security and real-time requirements are considered. In [24], the authors try to minimize the hardware cost for security-aware ACPS, but it considers the FlexRay as the in-vehicle network. Our work considers the CAN FD-based ACPS, and we assume a more general system model with two security levels.

3. System Models and Key Assumptions

3.1. System Model

Figure 1 shows a typical safety-critical electronic subsystem inside the ACPS, where several ECUs are interconnected by CAN FD. We assume that there is a task set in each ECU and tasks are scheduled with static priority-based preemptive scheduling algorithm according to the AUTOSAR specification [25]. Messages are transmitted on the bus to realize the communication and cooperation between communicating tasks which are assigned to different ECUs. ECU set is represented as $ECU = \{E_1, E_2, \ldots E_k, \ldots E_{EN}\}$, where EN indicates the number of ECU in E_k. Each ECU may has two cores: the computing core and HSM core. The computing core is necessary, and it is responsible for the computing tasks, whereas the HSM core is optional, and only those ECUs that include security-critical tasks have HSMs, as security-critical tasks have to send security-critical messages needing to be security-enhanced (such as adding MAC for integrity protection). HSM is used for security protection, such as key storage, MAC generation, message encryption/decryption, and others [26].

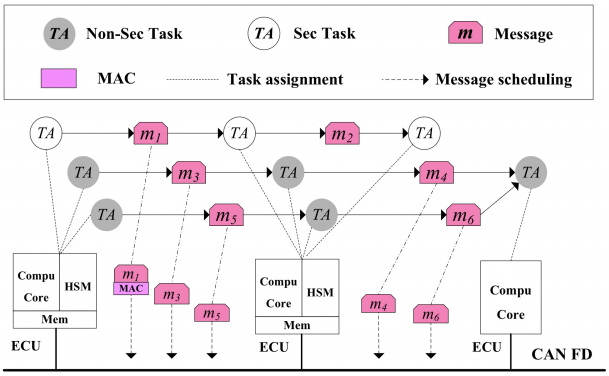


Figure 1. DSE for CAN FD-based ACPS.

3.2. Task Model

Each ACPS application is a directed acyclic graph (DAG) $G =< V, E >$, where V is the set of computing tasks and E is the set of messages. Each message indicates the communication relation between two adjacent tasks, if two adjacent tasks assigned to different ECUs, the corresponding message needs to be transmitted on CAN FD; or else, two adjacent tasks can communicate with each other through shared memory. The ACPS includes functions with different security requirements, functions such as steering control and braking control are with high security requirement, functions such as window control and remote door lock control have low security requirement [27]. Thus, we divide the computing tasks into two subgroups based on their level of security requirement, namely, security-critical tasks and non-security-critical tasks [28]. For those security-critical tasks, they can only be assigned to those ECUs with HSM. While for non-security-critical tasks, they can be assigned to any ECUs. The set of computing task in E_k is indicated as Z_k, where $Z_k = \{T_{k,1}, T_{k,2}, \ldots T_{k,i}, \ldots T_{k,TN_k}\}$, and TN_k indicates the number of tasks in Z_k. The task set of all ECUs is denoted by Z, where $Z = \bigcup_{\forall k} Z_k$, and TN indicates the total number of tasks in Z. $T_{k,i}$ is denoted with a 5-tuple: $T_{k,i} = \{P_{k,i}, C_{k,i}, R_{k,i}, D_{k,i}, Q_{k,i}\}$, which indicate the period (in μs), worst-case execution time (WCET, in μs), worst-case response time (WCRT, in μs), deadline (in μs) and security level, respectively. $Q_{k,i}$ is a binary variable, if $Q_{k,i}$=1, it indicates that $T_{k,i}$ is a security-critical task; otherwise, $T_{k,i}$ is a non-security-critical task. We assume that deadline equals to period for each task, and priorities are assigned to tasks based on their periods, these assumptions are common in ACPS [29–31]. $P_{k,i}$, $C_{k,i}$, $D_{k,i}$, and $Q_{k,i}$ are given, and $R_{k,i}$ is calculated as follows:

$$R_{k,i} = C_{k,i} + \sum_{\forall T_{k,i'} \in hp_{k,i}} \lceil \frac{R_{k,i}}{P_{k,i'}} \rceil \times C_{k,i'} \tag{1}$$

where $hp_{k,i}$ indicates the set of tasks with higher priorities than $T_{k,i}$ in E_k.

3.3. Message Model

Computing tasks of each ACPS functions are allocated to different ECUs, and messages are exchange between different ECUs to realize the communication and cooperation of them, thus CAN FD messages have the same security requirement with their sending tasks. M_k indicates the set of messages sent by ECU_k, where $M_k = \{m_{k,1}, m_{k,2}, \ldots m_{k,j}, \ldots m_{k,MN_k}\}$, and MN_k indicates the number of messages in M_k. The message set of all ECUs is denoted by M, where $M = \bigcup_{\forall k} M_k$, and MN indicates the total number of messages in M. $m_{k,j}$ is indicated with a 5-tuple: $m_{k,j} = \{p_{k,j}, c_{k,j}, r_{k,j}, d_{k,j}, q_{k,i}\}$, which indicate the period (in µs), worst-case transmission time (WCTT, in µs), WCRT (in µs), deadline(in µs) and security level of $m_{k,j}$, respectively. $p_{k,j}$ and $c_{k,j}$ are given, $q_{k,j}$ equals to that of its sending task, and $d_{k,j}$ equals to $p_{k,j}$ [29–31]. As the authors of [32] verified that the rate monotonic priority order is close to the optimal priority order for CAN, thus we assume that the rate monotonic priorities are assigned to CAN FD messages. CAN FD messages are scheduled non-preemptively before transmitting on the bus, thus $r_{k,j}$ is calculated as follows [30,33]:

$$\begin{aligned} &\forall m_{k,j}, m_{k',j'}, r_{k,j} = \\ &block_{k,j} + c_{k,j} + \sum_{\forall m_{k',j'} \in shp_{k,j}} \lceil \frac{r_{k,j}}{p_{k',j'}} \rceil \times c_{k',j'} \end{aligned} \tag{2}$$

where $block_{k,j}$ indicates the blocking delay incurred by low priority messages, $shp_{k,j}$ indicates the set of messages with higher priorities than $m_{k,j}$ in M.

As it shows in Figure 1, we try to reduce the security-related hardware cost by minimize the number of HSMs required for a given CAN FD-based ACPS, where both the task mapping, task scheduling and message scheduling are explored subject to end-to-end deadline and security constraints. Thus, we need to define the end-to-end delay for each function path. A function path PH_l is an ordered interleaving sequence of tasks

and messages, where $PH_l = [T_{l,1}, m_{l,1}, T_{l,2}, m_{l,2}, \ldots, T_{l,n-1}, m_{l,n-1}, T_{l,n}]$, and n indicates the number of tasks in PH_l. PH indicates the set of function paths in ACPS. We assume an asynchronous sampling communication approach is employed in ACPS, thus the end-to-end WCRT of PH_l (we indicate it as PR_l) can be calculated by adding the WCRT of all computing tasks and CAN FD messages on the path, the security-enhancing related delay overhead, as well as the periods of all the messages and their receiving tasks on the path [34]. To sum up, PR_l is calculated as follows:

$$PR_l = \sum_{T_{k,i} \in PH_l \bigwedge m_{k,j} \in PH_l} (R_{k,i} + P_{k,i} + r_{k,j} + p_{k,j} + l_{k,j} \times o_{k,j}) \tag{3}$$

where $o_{k,j}$ indicates the security-enhancing related delay overhead. As messages are processed with the same operations non-preemptively inside the HSM [35], we assume an equal $o_{k,j}$ for all security-critical messages, and it is set according to the real implementation of [11]. For each function path, there is an end-to-end deadline, and we indicate it as PD_l. APCS is schedulable and safe only when all their included function paths meet the deadline constraint.

4. Hardware Cost Minimization Algorithms

4.1. Stepwise Decreasing Based Heuristic Algorithm

Randomized optimization algorithms such as simulated annealing and genetic algorithm are widely employed to solve research problems similar to this paper [24,36]. However, they only evaluate a finite number of design point based on random changes, thus as the design space expands, they are easily trapped into local optimum, and the global optimum can not be assured. To solve this problem, this paper proposes a stepwise decreasing based heuristic algorithm (SDH) based on the basic ideas of SA. The main differences between SDH and SA are as follows: (1) SDH tries to reduce the search space to a smaller one by implementing a stepwise decreasing on the number of HSMs. As the search space is reduced, it is more likely to find a better task assignment result; (2) in each searching loop when the number of HSMs is given, as long as one feasible task assignment result that meets both the end-to-end deadline and security constraint is found, the heuristic searching step is terminated, thus this allows SDH to reduce the number of HSMs quickly. Details of SDH is given in Algorithm 1.

HN indicates the minimal number of HSMs that are required to be attached to ECUs to meet the design constraints of ACPS, HN' is a temporary variable used in each searching loop, where both HN and HN' are initially set as EN. PR_{avg} and PD_{avg} indicate the average end-to-end WCRT and average end-to-end deadline of all function paths, respectively. $flag$ indicates if the task assignment result can meet the deadline constraint or not, and it is initially set as $TRUE$. For SDH, there is an outer while loop (from line 3 to line 27) that realize the stepwise reduction on the number of HSMs, where if the inner while loop can get a feasible task assignment meeting both the real-time and security constraint of ACPS (from line 8 to line 26), the number of HSMs can be reduced by 1, and the outer while loop continues; otherwise, the outer while loop is terminated. Inside the outer while loop, an initial task assignment result is obtained by randomly assigning tasks to ECUs (line 4), and then the average end-to-end WCRT is analyzed (line 5). Next, the inner while loop attempts to decrease the end-to-end WCRT of function paths to meet the end-to-end deadline constraint, which heuristically exchanging tasks assigned to two different ECUs or moving tasks from one ECU to another ECU (line 10). After the timing analysis of the updated task assignment result (line 11), if the end-to-end WCRT if reduced, the new task assignment result is accepted; otherwise, accept it with a certain random probability based on the average end-to-end WCRT difference between current task assignment result and the new task assignment result (from line 12 to line 15). If all function paths can meet the deadline constraints, it means that one feasible task assignment result is found, thus HN' can be reduced by 1, and the outer while loop continues (from line 16 to line 20); or else, $flag$ is set as $FALSE$ to indicate that current searching step can not find a feasible task

assignment result(from line 21 to 23). If the inner while loop cannot find a feasible task assignment result in all the heuristic searching steps, the outer while loop is terminated and HN is returned as the final result.

Algorithm 1 Stepwise Decreasing-Based Heuristic Algorithm.

Input: Z, EN
Output: HN

1: HN=EN, $HN' = EN$;
2: $flag = TRUE$;
3: **while** $(HN' \geq 1) \wedge (flag = true)$ **do**
4: $Result$=Task_Allocation(Z, EN, HN');
5: PR_{avg}=E2EWCRT_Analysis($Result$);
6: $T_{ini} = 3 * EN$, $T_{ter} = .5$,$step\_num = 5 * TN * MN$, $\theta = 0.98$;
7: $T = T_{ini}$;
8: **while** $T > T_{ter}$ **do**
9: **for** $i = 1$ to $step\_num$ **do**
10: $Result'$=Heuristic_Task_Move($Result$);
11: PR'_{avg}=E2EWCRT_Analysis($Result'$);
12: **if** $(PR'_{avg} < PR_{avg}) \vee (exp((PR_{avg} - PR'_{avg})/T) > Rand(0,1))$ **then**
13: $PR_{avg} = PR'_{avg}$;
14: $Result = Result'$
15: **end if**
16: **if** $(PR < PD)$ **then**
17: $flag = TRUE$;
18: $HN = HN'$
19: $HN' = HN' - 1$
20: break;
21: **else**
22: $flag = FALSE$;
23: **end if**
24: **end for**
25: $T = T * \theta$;
26: **end while**
27: **end while**

To sum up, SDH realizes a stepwise decrease of the design space by reducing the number of HSM one by one. For a given system configuration, as long as one viable task assignment solution is found, the searching step is terminated immediately. Thus, SDH can shorten the searching time to find a viable task assignment solution for the given number of HSMs. Furthermore, the proactive reduction of search space taken by SDH makes it easier to find a viable task mapping solution with fewer HSMs.

4.2. Interference Balancing Based Heuristic Algorithm

As SDH is implemented based on the basic ideas of SA, the efficiency of it will deteriorate as the search space expands. Moreover, as the number of HSMs decreases, it becomes increasingly difficult for the heuristic searching to find a feasible task assignment, thus the runtime of SDH increases rapidly. To remedy the above mentioned disadvantages, this paper proposes another interference balancing-based heuristic algorithm IBH. As the number of messages that are scheduled and transmitted on CAN FD depends on the task assignment, and tasks contribute a larger part to the end-to-end WCRT than that of the messages, tasks and their WCRT analysis are primarily considered during the DSE process of IBH. To be more specific, IBH employs the definition of variance to describe difference of interferences caused by the set of tasks assigned to different ECUs. The interference variance is defined as Definition 1 shows.

Definition 1. *The interference variance is the interference differences caused by the set of tasks assigned to different ECUs for a given time interval.*

For a given time interval *len*, the interference variance is calculated as Equation (4) shows.

$$inf\_variance = \sum_{k=1}^{EN} \frac{inf_k^{len} - inf_{avg}^{len}}{EN} \tag{4}$$

where inf_k^{len} indicates the interference caused by the set of tasks assigned to E_k, and the inf_{avg}^{len} indicates the average inf of all ECUs. By assigning task to the ECU that causes the minimal interference variance, the interference caused by those already assigned tasks as well as the end-to-end WCRT of the function paths can be balanced. Details of IBH is given in Algorithm 2.

Algorithm 2 Interference Balancing-Based Heuristic Algorithm.

Input: Z, EN
Output: HN

$HN = EN$, $flag = TRUE$;
Task_Sort(Z);
while $(HN \geq 1) \wedge (flag = true)$ **do**
  $Interval$=Min_Period(Z);
  for $i = 1$ to TN **do**
    $Variance$=$W(1, TN)$;
    if $Q_i == 1$ **then**
      for $k = 1$ to HN **do**
        Result=Task_Allocation(T_i, E_k);
        if T_i is schedulable **then**
          $Variance(1, k)$=Variance_Analysis($Interval$);
        end if
      end for
    else
      for $k = 1$ to EN **do**
        Result=Task_Allocation(T_i, E_k);
        if T_i is schedulable **then**
          $Variance(1, k)$=Variance_Analysis($Interval$);
        end if
      end for
    end if
    ($Min\_Variance$, k)=min($Variance$);
    if $Min\_Variance < W$ **then**
      $flag = TRUE$;
      Allocate T_i to the E_k;
      $Interval$=P_i;
    else
      $flag = false$;
      break;
    end if
  end for
  if system is schedulable **then**
    HN=HN-1;
  else
    break;
  end if
end while

For IBH, HN also indicates the minimal number of HSMs that are required to be attached to ECUs to meet the design constraint of ACPS, $flag$ indicates if each task can be successfully assigned to an ECU where it can meet the deadline constraint, $Interval$ indicates the time interval that the WCRT is analyzed for the tasks, and $Variance$ is an array to store the interference variances when tasks are assigned to different ECUs. HN is initially set as EN, and $flag$ is initially set as $TRUE$. IBH first sorts task with increasing period and size (line 2), and then there is a while loop tries to find a feasible task assignment result (it means that all function paths meet their end-to-end deadline constraint, and the system is schedulable) when the number of HSMs is set as HN (from line 3 to line 37). Inside the while loop, $Interval$ is initially set as the minimal task period (line 4), and next there is a for loop tries to assign each task to the ECU with the minimal variance (from line 5 to line 31). Inside the for loop, each element of $Variance$ is initially set as a big number W (line 6). Moreover, if the current task is a security-critical task, it tries to assign it to the ECUs with HSMs, and the corresponding variance is analyzed(from line 7 to line 14); or else if the current task is a non-security-critical task, it tries to assign it to all ECUs, and the corresponding variance is analyzed(from line 15 to line 21). After the analysis of all possible variances, if the current task can be successfully assigned to ECUs (which means that all assigned tasks are schedulable), it is assigned to the ECU with the minimal variance (from line 22 to line 25), and $Interval$ is updated to the period of the current task (line 26); or else, the current task can not be assigned successfully, which means that it can not find a feasible task assignment result when the number of HSMs is set as HN (from line 27 to line 30). Thus, the for loop is terminated. After the for loop, if all tasks can be assigned properly, HN is reduced by 1, and the while loop continues (from line 32 to line 34); or else, the while loop is terminated (from line 35 to line 36). HN is returned as the minimal number of HSMs that are required to meet both the security and real-time constraints.

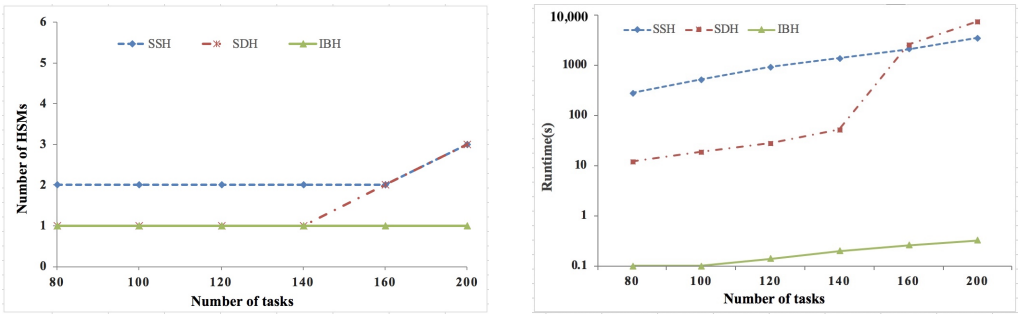
To sum up, IBH also realizes a stepwise decrease of the design space by reducing the number of HSM one by one, the difference between IBH and SDH is that IBH replaces the random searching approach with interference balancing-based approach to find a viable task assignment for the given system configuration. The interference balancing-based approach is easier to find a schedulable task assignment result, as it considers to balance the interference to the unassigned tasks in each step.

5. Experiment Results

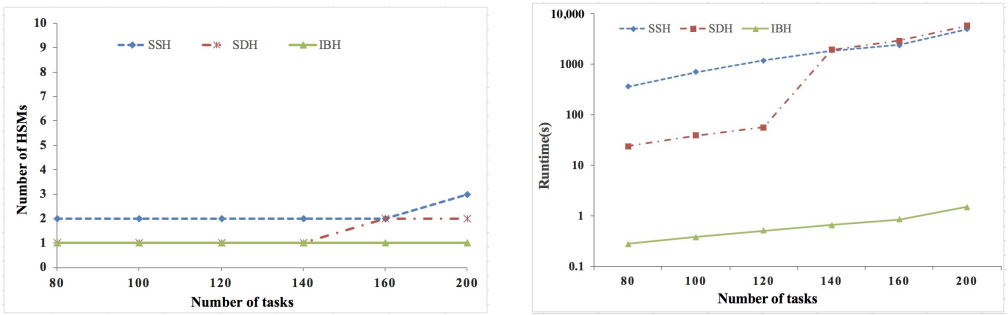
We conducted extensive experiments based on both synthetic and real data sets to verify the proposed SDH and IBH algorithm, and the efficiency of the proposed algorithms is shown by comparing them with the simulated annealing-based heuristic algorithm presented in [24]. This algorithm is state-of-the-art, and we indicate it as SSH. The task graph generator given in [37] is used to generate synthetic ACPS functions, where the parameters of the tasks and messages are generated according to the guidelines on real-world automotive benchmarks [38]. For synthetic data sets, the number of ECUs is set as 6/10/14/18, and the number of tasks is set as 80/100/120/140/160/200. Six real-life ACPS functions given in [15] are adopted to generate the real data sets, where the number of tasks are increased by replicating the ACPS functions. For real data sets, the number of ECUs is set as 6/8/10/12, and the number of tasks is set as 40/60/80/100/120. For both synthetic and real data sets, the percent of security-critical tasks is set as 20%/40%/60%. $o_{k,j}$ is set according to the real implementation given in [35], where $o_{k,j}$=103 μs. We assume that the arbitration phase bit-rate and data phase bit-rate of CAN FD are 500 kbps and 2 Mbps [30,31], respectively. The experiments are conducted on an OS X(v10.13.1) machine running on 2.3 GHz the 7th generation Intel i5 core with 8 GB main memory. The experiment code is implemented in Matlab 2017a.

Figures 2–4 show the experimental results of synthetic data sets, where the percent of security-critical tasks is set as 20%, 40%, and 60%, respectively. Figures 5–7 show the experimental results of real data sets, where the percent of security-critical tasks is also set as 20%, 40%, and 60%, respectively. From the above mentioned results, we can conclude

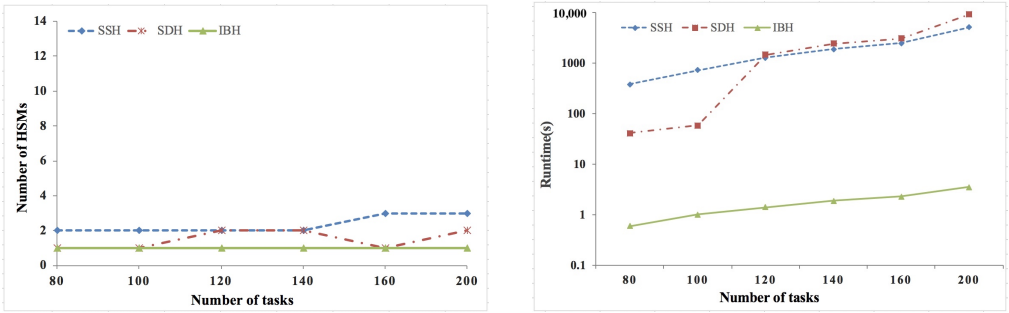
that the proposed SDH and IBH are better than SSH in reducing the security-related hardware cost for ACPS. Especially as the increase about the percent of security-critical tasks, the advantage of SDH and IBH becomes more obvious. For synthetic data sets, the hardware cost can be reduced by 61.4% and 45.6% averagely for IBH and SDH, respectively, and for real data sets, the hardware cost can be reduced by 64.3% and 54.4% averagely for IBH and SDH, respectively. Furthermore, as the expansion of the design space due to the increase of ECUs and tasks, IBH shows a better performance than SDH in reducing the number of HSMs in most cases, and it also gets the same performance with SDH in the other cases. Last but not the least, the runtime of IBH is two or three orders of magnitude smaller than SDH and SSH, which means that SDH is quite extensible.



(**a**) EN = 6

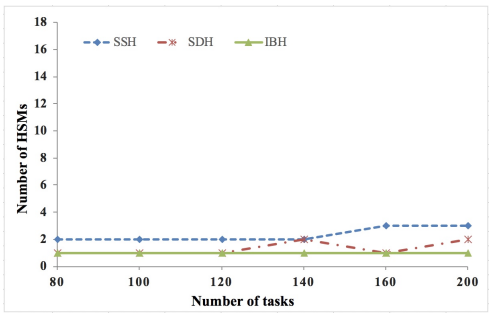


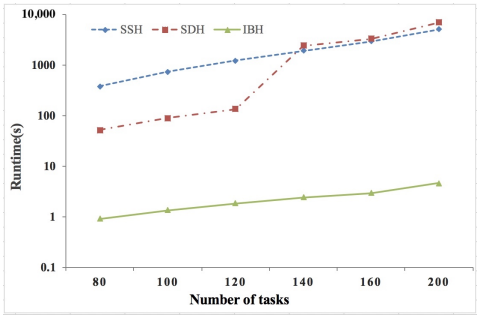
(**b**) EN = 10



(**c**) EN = 14

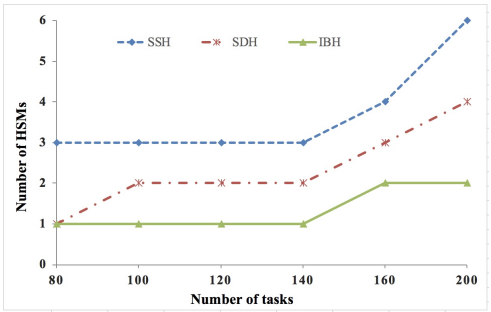
Figure 2. *Cont.*

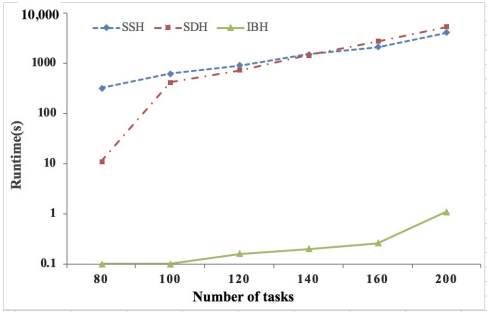




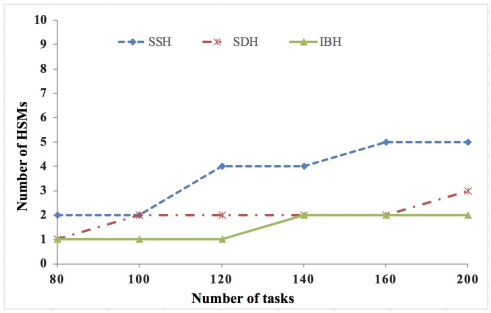
(**d**) EN = 18

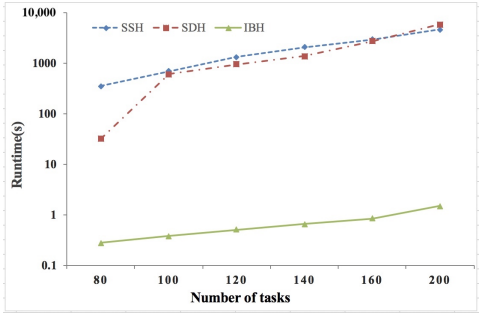
Figure 2. Experimental results of synthetic data set when percent of security-critical task is 20%.



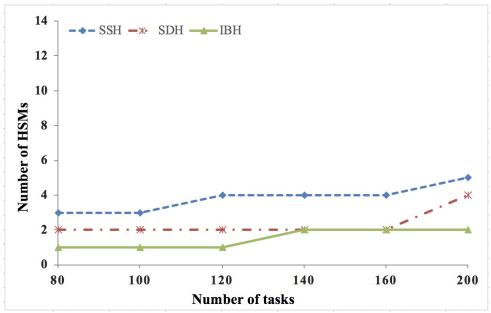


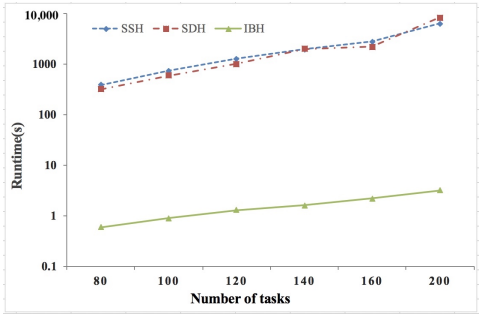
(**a**) EN = 6





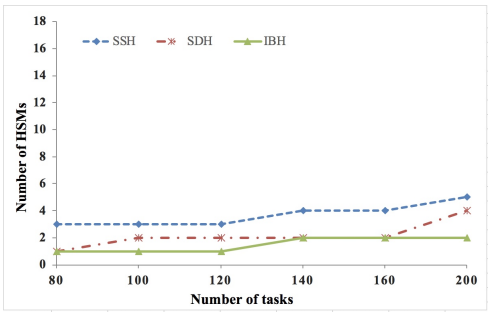
(**b**) EN = 10

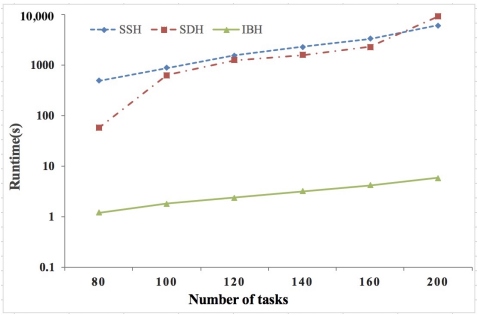




(**c**) EN = 14

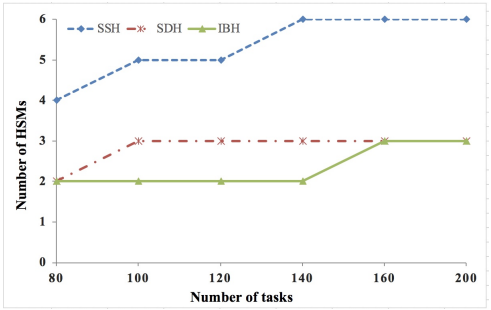
Figure 3. *Cont*.

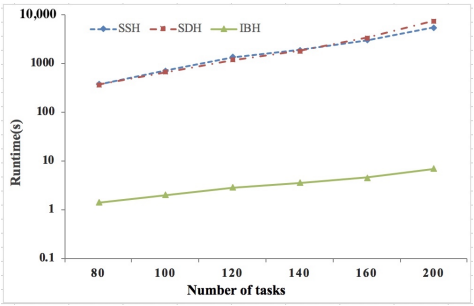




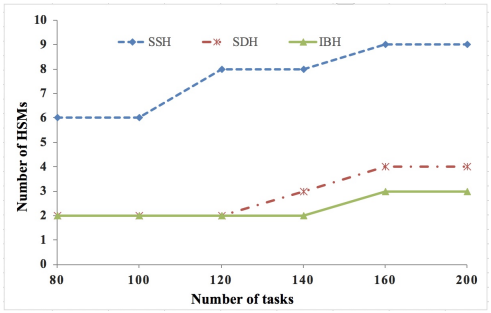
(**d**) EN = 18

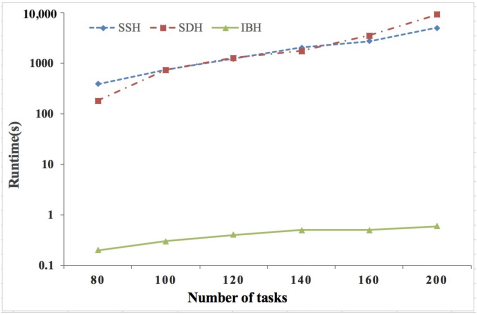
Figure 3. Experimental results of synthetic data set when percent of security-critical task is 40%.



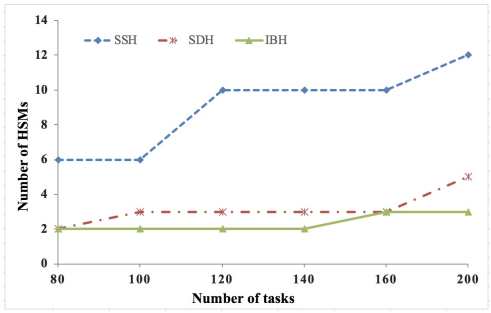


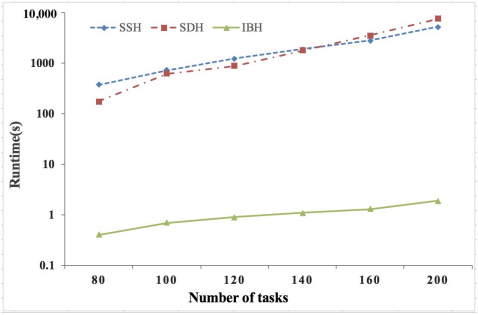
(**a**) EN = 6





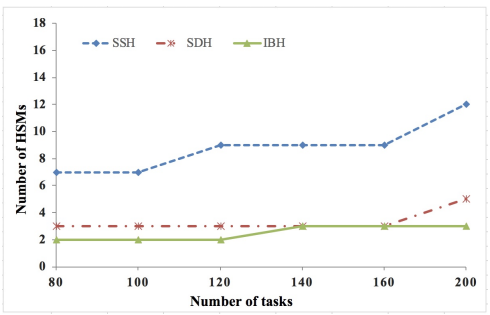
(**b**) EN = 10

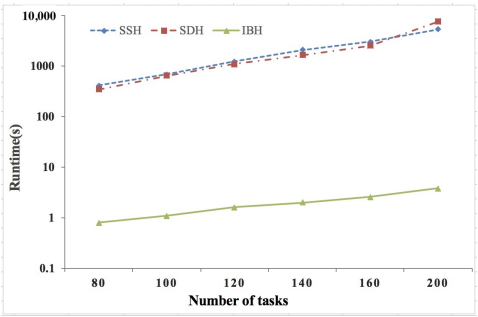




(**c**) EN = 14

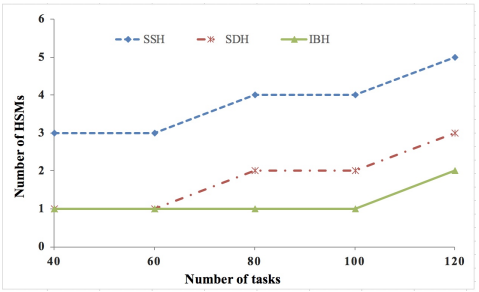
Figure 4. *Cont*.

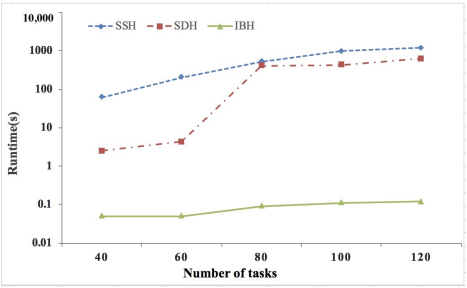




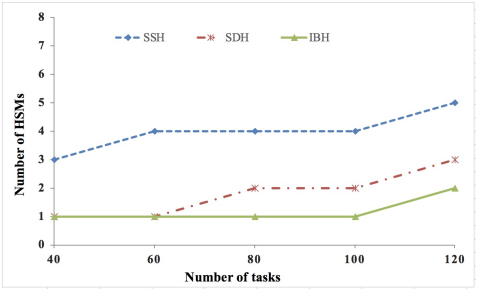
(**d**) EN = 18

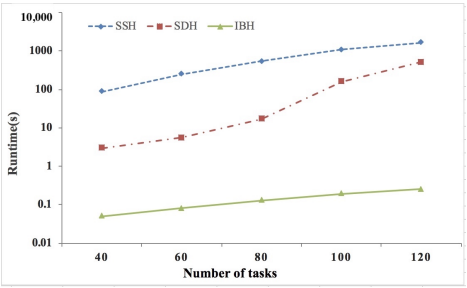
Figure 4. Experimental results of synthetic data set when percent of security-critical task is 60%.



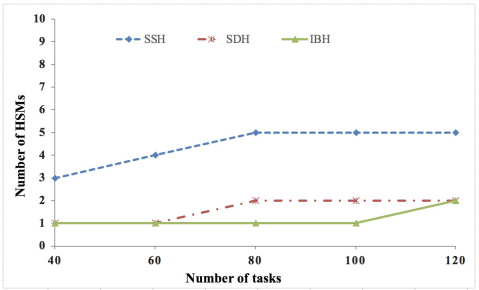


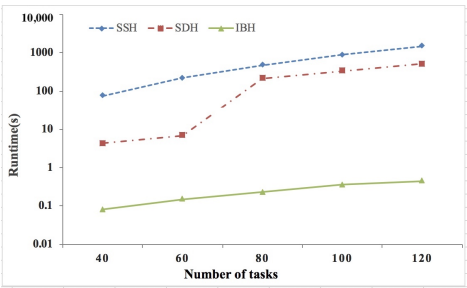
(**a**) EN = 6





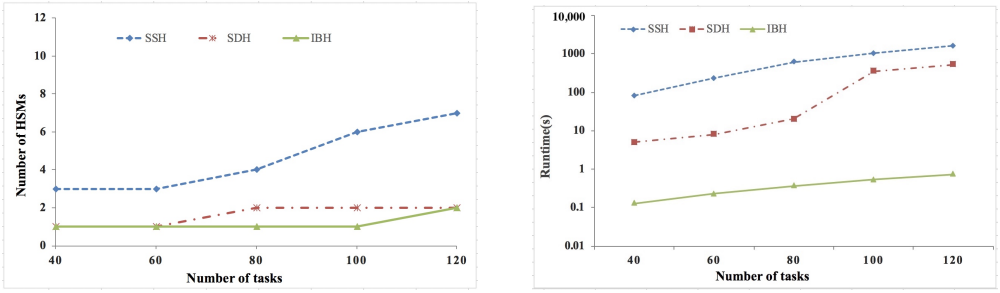
(**b**) EN = 8





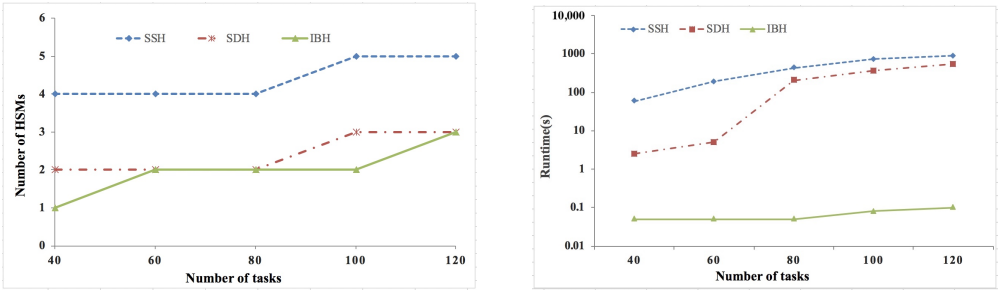
(**c**) EN = 10

Figure 5. *Cont.*

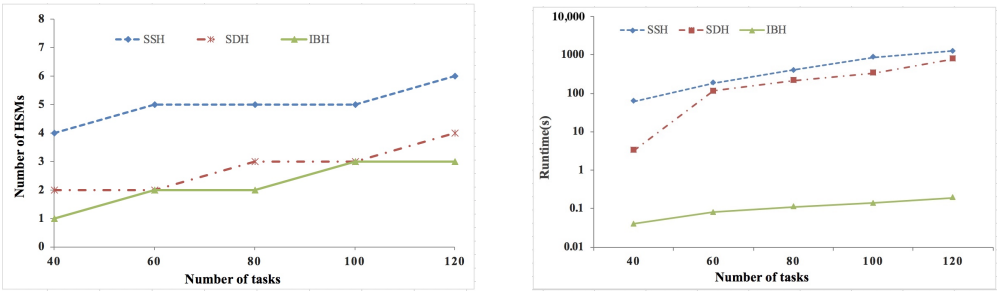


(**d**) EN = 12

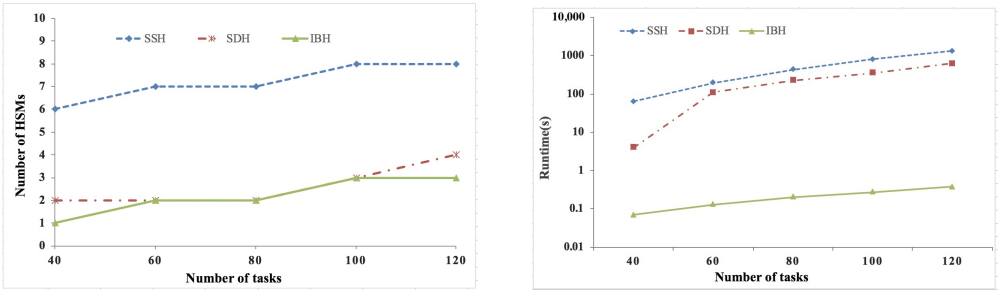
Figure 5. Experimental results of real data set when percent of security-critical task is 20%.



(**a**) EN = 6

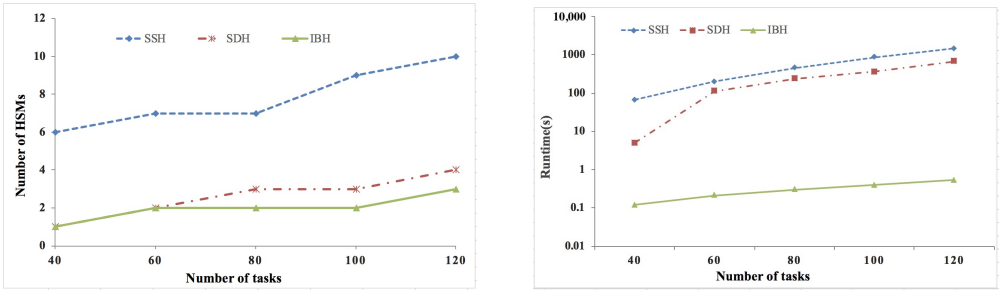


(**b**) EN = 8



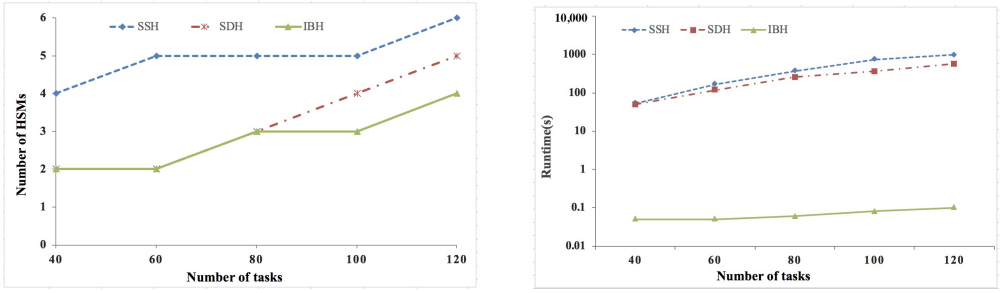
(**c**) EN = 10

Figure 6. *Cont.*

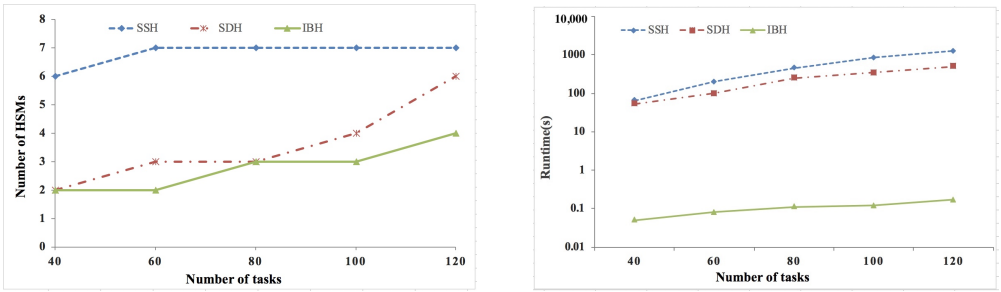


(**d**) EN = 12

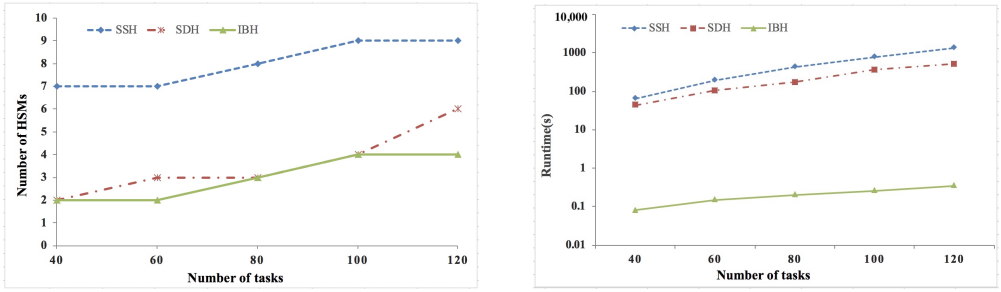
Figure 6. Experimental results of real data set when percent of security-critical task is 40%.



(**a**) EN = 6

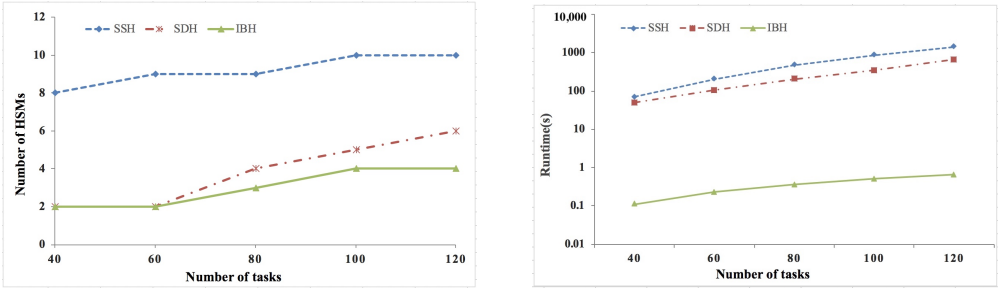


(**b**) EN = 8



(**c**) EN = 10

Figure 7. *Cont*.



(**a**) EN = 12

Figure 7. Experimental results of real data set when percent of security-critical task is 60%.

6. Conclusions

As connectivity between and within automobiles increases, it brings great cyber-security challenges for CAN FD-based ACP. The auto industry recommends employing the HSM-based multicore ECUs to secure the ACPS functions with acceptable delay overhead. However, the introduction of HSM incurs significant hardware cost. In this paper, we try to reduce security-enhancing-related hardware cost by proposing two efficient DSE algorithms, namely, SDH and IBH, which explore the task assignment, task scheduling, and message scheduling to minimize the number of required HSMs. Experiments on both synthetical and real data sets show that the proposed SDH and IBH are superior than the state-of-the-art SSH algorithm, and the advantage of SDH and IBH becomes more obvious as the increase about the percentage of security-critical tasks. Furthermore, IBH is better than SDH, and the runtime of IBH is two or three orders of magnitude smaller than SDH and SSH.

Author Contributions: Methodology, Y.X., J.Z. and X.C.; Software, Y.G. and S.Y.; Writing—review, editing, Y.X.; Algorithm design and paper writing, Y.X.; Algorithm design and experiments, Y.G.; Experiments, S.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant 61872436 and 61972210, in part by the Natural Science Foundation of Jiangsu Province, China under Grant BK20211272, in part by NUPTSF (Grant No.NY220133), and in part by the Research Foundation of NJUPT for "1311 Talents Training Project".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Koscher, K.; Czeskis, A.; Roesner, F.; Patel, S.; Kohno, T.; Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H.; Savage, S. Experimental security analysis of a modern automobile. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010; pp.447–462.
2. Hartwich, F. CAN with flexible data-rate. In Proceedings of the International CAN Conference, Hombach Castle, Germany, 10–11 October 2012; pp.14.1–14.9.
3. With "Recall" Fiat Chrysler Makes Its Car Hack Worse. Available online: www.networkworld.com/article/2953836/security/with-recall-fiat-chrysler-makes-its-car-hack-worse (accessed on 30 November 2016) .
4. Xie, Y.; Liu, L.J.; Li, R.F.; Hu, J.Q.; Han, Y.; Peng, X. Security-aware signal packing algorithm for CAN-based automotive cyber-physical systems. *IEEE/CAA J. Autom. Sin.* **2015**, *2*, 248–257.
5. Xie, Y.; Zeng, G.; Kurachi, R.; Takada, H.; Xie, G.Q. Security/Timing-aware design space exploration of CAN FD for automotive cyber-physical systems. *IEEE Trans. Ind. Inform.* **2019**, *15*, 1094–1104. [CrossRef]
6. AUTOSAR Specification of Module Secure Onboard Communication, Version 4.4.0. Available online: https://www.autosar.org/fileadmin/Releases_TEMP/Classic_Platform_4.4.0/Communication.zip (accessed on 20 November 2020).
7. Munir, A.; Koushanfar, F. Design and analysis of secure and dependable automotive CPS: A steer-by-wire case study. *IEEE Trans. Depend. Secur. Comput.* **2018**, *17*, 813–827. [CrossRef]

8. Xie, G.Q.; Li, R.F.; Hu, S.Y. Security-aware obfuscated priority assignment for CAN FD messages in real-time parallel automotive applications. *IEEE Trans. CAD Integr. Circuits Syst.* **2020**, *39*, 4413–4425. [CrossRef]
9. Xie, Y.; Zhou, Y.; Xu, J.; Zhou, J.; Chen, X.; Xiao, F. Cybersecurity protection on in-vehicle networks for distributed automotive cyber-physical systems: State of the art and future challenges. *Softw. Pract. Exp.* **2021**, *51*, 2108–2127. [CrossRef]
10. Stumpf, F.; Pohl, C.; Hoettges, D.; Klein, T. Introducing HSM-based secure on-board communication in vehicles-challenges and lessons learned. In Proceedings of the Escar Europe, Stuttgart, Germany, 19–20 November 2019; pp. 1–4.
11. Wu, Z.; Zhao, J.; Zhu, Y.; Li, Q. *Research on Vehicle Cybersecurity Based on Dedicated Security Hardware and ECDH Algorithm*; SAE Technical Paper, No. 2017-01-2005; SAE International: Warrendale, PA, USA, 2017.
12. Chip Price. Available online: https://www.arrow.com (accessed on 23 May 2020).
13. Shimizu, N. Fujitsu Develops Platforms for Toyota's, Honda's Cryptographic Technologies. 2018. Available online: http://tech.nikkeibp.co.jp/atcl/nxt/column/18/00213/00003/ (accessed on 12 July 2019).
14. Xie, G.; Chen, Y.; Liu, Y.; Wei, Y.; Li, R.; Li, K. Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems. *IEEE Trans. Ind. Inform.* **2017**, *13*, 1629–1640. [CrossRef]
15. Gan, J.; Pop, P.; Madsen, J. Tradeoff Analysis for Dependable Real-Time Embedded Systems during the Early Design Phases. Ph.D. Thesis, Technical University of Denmark, Department of Informatics and Mathematical Modeling, Lyngby, Denmark, 2014.
16. Tamas-Selicean, D.; Pop, P. Design optimization of mixed-criticality real-time embedded systems. *ACM Trans. Embed. Comput. Syst.* **2015**, *14*, 50. [CrossRef]
17. Xie, G.; Wu, W.; Zeng, G.; Li, R.; Hu, S. Risk Assessment and Development Cost Optimization in Software Defined Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3675–3686. [CrossRef]
18. Xie, G.; Chen, Y.; Li, R.; Li, K. Hardware cost design optimization for functional safety-critical parallel applications on heterogeneous distributed embedded systems. *IEEE Trans. Ind. Inform.* **2018**, *14*, 2418–2431. [CrossRef]
19. Zou, W.; Li, R.; Wu, W.; Zeng, L. Hardware cost and energy consumption optimization for safety-critical applications on heterogeneous distributed embedded systems. In Proceedings of the 24th IEEE International Conference on Parallel and Distributed Systems, Singapore, 11–13 December 2018; pp. 527–536.
20. Xie, G.; Ma, W.; Peng, H.; Li, R.; Li, K. Price performance-driven hardware cost optimization under functional safety requirement in large-scale heterogeneous distributed embedded systems. *IEEE Trans. Ind. Electron.* **2021**, *68*, 4485–4497. [CrossRef]
21. Salimi, M.; Majd, A.; Loni, M.; Seceleanu, T.; Seceleanu, C.; Sirjani, M.; Daneshtalab, M.; Troubitsyna, E. Multi-objective Optimization of Real-Time Task Scheduling Problem for Distributed Environments. In Proceedings of the 6th Conference on the Engineering of Computer Based Systems, Bucharest, Romania, 2–3 September 2019; pp. 1–9.
22. Niknam, S.; Wang, P.; Stefanov, T.P. Resource optimization for real-time streaming applications using task replication. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2018**, *37*, 2755–2767. [CrossRef]
23. Cho, H.; Kim, C.; Sun, J.; Easwaran, A.; Park, J.-D.; Choi, B.-C. Scheduling parallel real-time tasks on the minimum number of processors. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 171–186. [CrossRef]
24. Gu, Z.; Han, G.; Zeng, H.; Zhao, Q. Security-aware mapping and scheduling with hardware co-processors for FlexRay-based distributed embedded systems. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 3044–3057. [CrossRef]
25. Saidi, S.; Steinhorst, S.; Hamann, A.; Ziegenbein, D.; Wolf, M. Future automotive systems design: Research challenges and opportunities. In Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis, Torino, Italy, 30 September–5 October 2018; pp. 1–7.
26. Corbett, C.; Brunner, M.; Schmidt, K.; Schneider, R.; Bannebaum, U. *Leveraging Hardware Security to Secure Connected Vehicles*; SAE Technical Paper, 2018-01-0012; SAE International: Warrendale, PA, USA, 2018.
27. Wyglinski, A.M.; Huang, X.; Padir, T.; Lai, L.; Eisenbarth, T.R.; Venkatasubramanian, K. Security of autonomous systems employing embedded computing and sensors. *IEEE Micro* **2013**, *33*, 80–86. [CrossRef]
28. Nilsson, D.K.; Phung, P.H.; Larson, U.E. Vehicle ECU classification based on safety-security characteristics. In Proceedings of the IET Road Transport Information and Control Conference, Manchester, UK, 20–22 May 2008; pp. 1–7.
29. Joshi, P.; Zeng, H.B.; Bordoloi, U.D.; Samii, S.; Ravi, S.S.; Shukla, S.K. The multi-domain frame packing problem for CAN-FD. In Proceedings of the Euromicro Conference on Real-Time Systems, Dubrovnik, Croatia, 27–30 June 2017; pp. 12:1–12:22.
30. Xie, Y.; Zeng, G.; Kurachi, R.; Peng, X.; Xie, G.Q.; Hiroaki, T. Balancing bandwidth utilization and interrupts: Two heuristic algorithms for the optimized design of automotive CPS. *IEEE Trans. Ind. Inform.* **2020**, *16*, 2382–2392. [CrossRef]
31. Xie, Y.; Zeng, G.; Kurachi, R.; Peng, X.; Xie, G.Q.; Hiroaki, T., Optimizing Extensibility of CAN FD for Automotive Cyber-Physical Systems. *IEEE Trans. Intell. Transp. Syst.* **2021**. [CrossRef]
32. Natale, M.D.; Meschi, A. Scheduling messages with earliest deadline techniques. *J.-Real-Time Syst.* **2001**, *20*, 255–285. [CrossRef]
33. Davis, R.I.; Burns, A.; Bril, R.J.; Lukkien, J.J. Controller Area Network(CAN) schedulability analysis: Refuted, revisited and revised. *J. -Real-Time Syst.* **2007**, *35*, 239–272. [CrossRef]
34. Davare, A.; Zhu, Q.; Natale, M.D.; Pinello, C.; Kanajan, S.; Sangiovanni-Vincentelli, A.L. Period Optimization for Hard Real-time Distributed Automotive Systems. In Proceedings of the 44th Design Automation Conference, San Diego, CA, USA, 4–8 June 2007; pp. 278–283.
35. AUTOSAR Specification of Crypto Service Manager, Version 4.4.0. Available online: https://www.autosar.org/fileadmin/Releases_TEMP/Classic_Platform_4.4.0/Crypto.zip (accessed on 20 November 2020).

36. McLean, S.D.; Craciunas, S.S.; Hansen, E.A.J.; Pop, P. Mapping and scheduling automotive applications on ADAS platforms using metaheuristics. In Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation, Vienna, Austria, 8–11 September 2020; pp. 329–336.
37. Task Graph Generator. Available online: https://sourceforge.net/projects/taskgraphgen/ (accessed on 1 August 2021).
38. Kramer, S.; Ziegenbein, D.; Hamann, A. Real world automotive bench- mark for free. In Proceedings of the 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), Lund, Sweden, 7 July 2015.