Optimization of Message Encryption for Distributed Embedded Systems with Real-Time Constraints

Ke Jiang, Petru Eles, Zebo Peng
Department of Computer and Information Science, Linköping University
{ke.jiang, petru.eles, zebo.peng}@liu.se

***Abstract*—In this paper we consider distributed embedded systems in which privacy or confidentiality of the internal communication is critical, and present an approach to optimizing cryptographic algorithms under strict timing constraints. We have developed a technique to search for the best system-affordable cryptographic protection for the messages transmitted over the internal communication bus. Towards this, we formulate the optimization technique in Constraint Logic Programming (CLP), which returns optimal results. However, CLP executions are computationally expensive and hence, we propose an efficient heuristic as an alternative. Extensive experiments demonstrate the efficiency of the proposed heuristic approach.**

I. Introduction

Nowadays, more and more embedded systems are being employed, many of which take advantages of distributed computing due to functional requirements or pursuance for better performance and reliability, such as automated flight control systems and automotive embedded systems. A current high-end sedan, for example, has more than 50 embedded processors and Electronic Control Units (ECUs) connected by various communication infrastructures, e.g., CAN-bus [1].

Security is an important issue in designing many distributed embedded systems, but is seriously overlooked. The key concepts concerning security of distributed embedded system are confidentiality, integrity, availability, authenticity, and non-repudiation, among which confidentiality is very often of central importance. For example, the messages transmitted within an automated flight control system used in military battlefields are highly confidentiality-sensitive. Disclosure of one single message may affect the life of the pilot or even the situation of a whole battle. The security requirements under military context are significant and obvious. Moreover, the requirement of communication secrecy in distributed embedded systems for civilian use, e.g., automotive IT systems, is emerging in recent years. In the following sections, we will discuss distributed embedded system security problems with an emphasis on the automotive domain, but our proposed technique can also be used in other application areas nevertheless.

As more embedded systems are connected to each other and to the Internet, potential security threats scale up dramatically. Although embedded system security has been addressed in some works like [2], [3], [4], security issues in distributed embedded system communication, especially the internal communication, were seriously omitted.

Moreover, new functions, such as drive-by-wire, that potentially increase safety of the passengers, fully depend on the underlying automotive data networks. Meanwhile, severe security threats to the privacy of the drivers and life-crucial in-vehicle control systems arose rapidly, as more critical information is sent over the underlying networks, and in completely unencrypted fashion [5]. Assuming that an attacker obtained access to a car's internal buses, e.g., via wireless communication interfaces, he could easily capture sensitive information of the driver, e.g., pay-for-use account and location based service (LBS) data, and maliciously use such information, e.g., to illegally track the driver.

Most researchers on vehicular networks have focused on protocols and applications, while paying little attention to potential security risks. There are works, e.g., [6], [7], [8], investigating automotive embedded system security, but most of them dealt with external vehicle communication, i.e., vehicle to vehicle (V2V) and vehicle to infrastructure (V2I) communication. Works considering security of in-vehicle communication, e.g., [5], [9], are rare. Wolf et al [5] presented feasible attacks and potential exposures for automotive networks, and proposed an abstract cryptographic architecture for protecting the networks. But the issues of doing encryption under actual resource constraints were not studied. In [9], four practically implemented attack scenarios were described, and the necessity of doing cryptography for protecting the internal bus communication is also raised. However, no concrete protection was proposed. Possible security threats to in-vehicle communication, especially considering privacy and legal aspects, are becoming more serious. Therefore, how to make the internal communication of automotive embedded systems more robust against malicious snooping becomes a pressing topic.

There are several other related works in the area of embedded system security. Xie et al [10] proposed a real-time scheduling algorithm that distributes slack times among an array of security services for a set of periodic tasks according to corresponding calculated security levels. However, no communication related security aspect is considered. In [11], Wollinger et al surveyed several important cryptographic concepts and their relevance to embedded systems. Gebotys [12] presented a table masking countermeasure to resist differential power analysis and differential electromagnetic analysis for secure embedded systems. However, to our knowledge, this is the first work to address the confidentiality aspect of internal bus communication of distributed embedded systems under resource and time constraints.

In this paper, we present a cryptography based technique that can be used to protect confidentiality of internal communication in distributed embedded systems under resource and time constraints. The technique searches for the best system-affordable cryptographic protection for the messages transmitted over the internal communication bus. However, due to the complexity of the problem, finding the optimal solutions, for example, using constraint logic programming, is only possible for small systems. So, a heuristic approach is proposed for solving the problem, which is very efficient in both execution time and results.

The rest of the paper is organized as follows. Section II briefly introduces the main issues of security in internal distributed embedded system communication. Section III and IV present the system model and a motivational example respectively. Section V formulates the design optimization problem raised in our study. Our proposed technique, where the main contribution of this work lied, is described in Section VI. The evaluation of the proposed approach is presented in Section VII. We also apply our approach in a real-life example in Section VIII. The last section concludes the work.

II. Preliminaries

A. Internal communication security in distributed embedded systems

A typical distributed embedded system is composed of a set of processors (or ECUs), controlling various functionalities, and one or several communication networks, e.g., buses, connecting the processors. The employment of buses remarkably reduces the length of wires and increases flexibility. These networks are normally bridged together to achieve better interactions and control. Moreover, more and more internal networks in modern distributed embedded systems have wireless interfaces like Wifi and telematics for external communication,

	Number of rounds				
	8	12	16	20	24
Differential cryptanalysis	2^{56}	2^{117}	2^{190}	2^{238}	2^{299}
Linear cryptanalysis	2^{47}	2^{83}	2^{119}	2^{155}	2^{191}

TABLE I
PLAINTEXT AND CIPHER PAIRS REQUIRED TO ATTACK RC6

which also open up the gate for adversaries to invade the internal networks. If an attacker could eavesdrop the internal communication via such interfaces, he would be able to obtain internally transmitted sensitive information or data, and, e.g., violate the owner's privacy.

One essential problem in the above context is that the processing power of embedded processors and the time constraint of the system limit the possibility of applying strong cryptographic protections. As mentioned in Section I, the communication over automotive buses today is completely unencrypted [5]. Therefore, the transmitted messages can be easily captured and understood. Another reason for this situation is that security issues were seriously neglected in the design of distributed embedded systems. In order to achieve acceptable overall security protection in the context of tight resource limits and stringent timing constraints, this aspect has to be considered during the early stages of system level design and optimization.

B. Cryptography in embedded systems

There are three main approaches in cryptography: public-key cryptography, symmetric-key cryptography and cryptographic hash functions. In public-key cryptosystems, different but related keys are used, including a public key and a private key. They are usually based on the computational complexity of "hard" mathematical problems, e.g., integer factorization problem, and are relatively costly in computation complexity compared with most symmetric key algorithms of equivalent security level. This has limited their use in resource constrained systems like embedded systems. In symmetric-key cryptography, the same key (or trivially related keys) is used for both encryption and decryption. The key represents a shared secret between two or more parties that have access to the confidential information.

In resource constrained systems, public-key algorithms are normally used for occasionally exchanging secret keys for symmetric-key algorithms which will perform the actual message encryption and decryption. By this, the convenience of public-key cryptosystems and the efficiency of symmetric-key cryptosystems are combined, and these systems are called hybrid cryptosystems. In this work, we will concentrate on maintaining confidentiality of internal bus communication by utilizing arguably the most widely used branch of symmetric cryptography, iterated block ciphers.

C. Iterated block ciphers

Iterated block ciphers are constructed by applying a function repeatedly in order to provide better information confusion and diffusion [13] as the number of rounds increases. They are known to be susceptible to cryptanalytic attacks, e.g., linear and differential cryptanalysis, that try to obtain secret information, such as the secret encryption key. However, these attacks require a large number of plaintexts and their encrypted versions from the target system, while the time needed to gather such required information grows exponentially as more rounds are used for encryption. Take a typical iterated block cipher, RC6 [14], for example. The required amount of plaintexts and corresponding encrypted versions for linear and differential cryptanalysis are presented in Table I, which shows that the amount grows exponentially when the number of rounds increases. Thus, the number of rounds used by iterated block ciphers is a determinant value deciding the security strength.

In iterated block ciphers, the more rounds are used for encryption and decryption, the more execution time the procedure will take. For correct message transmission, same number of rounds is used by an encryption and decryption (E/D) process pair. The worst case execution time (WCET) for the E/D pair ce_i and cd_i are as follows.

$$t_{ce_i} = w_E + r_E * x_i \quad (1)$$

$$t_{cd_i} = w_D + r_D * x_i \quad (2)$$

w_E and w_D represent the WCETs for doing the pre-/post-whitening for the algorithm on the encryption and decryption processor E and D, respectively. r_E and r_D represent the WCETs for doing one round on respective processor. Pre-/post-whitening is the initialization and ending operations of the algorithm that take constant time. x_i is the actual number of rounds used by this ith E/D pair ($i \in \{1, 2, ..., n_M\}$, where n_M is the total number of messages to be protected).

RC6 is simple and flexible while providing sound security protection [15] if the design parameters are carefully decided. All the follow-up discussions are based on the assumption of applying RC6 in the system. But our techniques is also applicable to other iterated block ciphers. In RC6, half of the data is encrypted in one round [14], so two rounds are considered as the smallest unit for doing encryption and decryption in following sections.

III. SYSTEM MODEL

We assume that the hardware architecture is constituted of a group of processors interconnected by a statically scheduled TDMA bus. The application is represented as an abstract model which is a directed, acyclic process graph $G(P, E, M)$. Each node $p_i \in P$ represents one process, and E is the set of edges. An edge $e_i \in E$ from p_s to p_t indicates that p_t depends on p_s in the execution flow. The mapping of the application processes to processing resources is given by a function $F : P \rightarrow PE$, where $PE = \{pe_1, pe_2, ..., pe_n\}$ is the set of processors (the bus is also considered as a processing resource). For any process p_i, $F(p_i)$ is the processor to which it is assigned for execution. $m_i \in M$ represents a message on an edge that connects two processes on different processors. These messages are to be sent over the bus, and are illustrated as black dots on the edges in process graphs. The application is constrained by a deadline D, meaning that an execution of the application must complete before the end-to-end delay D. WCET for each process and worst case transmission time for each message are also given.

Fig. 1 depicts an illustrative application graph. The hardware archi-

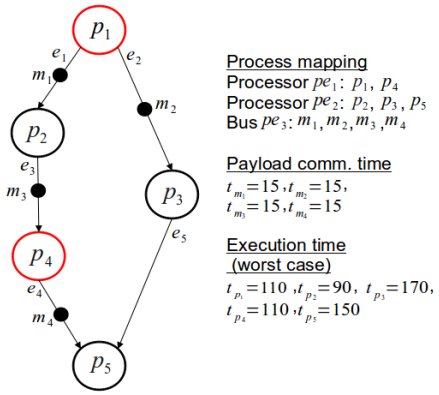

Fig. 1. A simple process graph

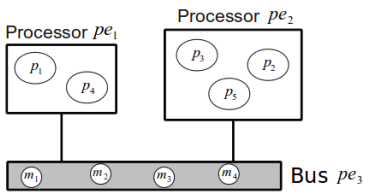

Fig. 2. A hardware architecture example

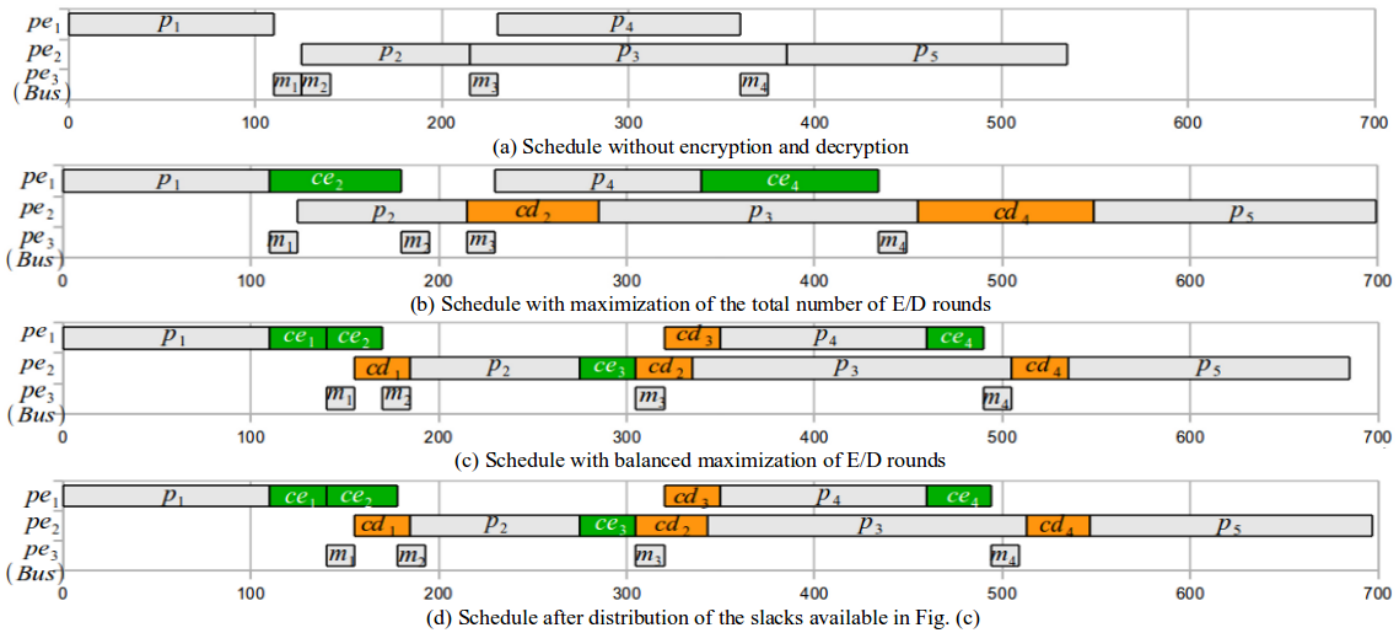

Fig. 3. Schedule of different solutions

tecture is shown in Fig. 2. In this example $P = \{p_1, p_2, p_3, p_4, p_5\}$, $E = \{e_1, e_2, e_3, e_4, e_5\}$, and $M = \{m_1, m_2, m_3, m_4\}$. Processes $\{p_1, p_4\}$ and $\{p_2, p_3, p_5\}$ are mapped to processor pe_1 and pe_2 respectively. The communication messages $\{m_1, m_2, m_3, m_4\}$ are transferred over the bus, denoted as pe_3. The WCETs of the processes are $T(P) = \{110, 90, 170, 110, 150\}$, and the worst case payload communication times of all the messages are assumed to be 15. The global deadline is 700 time units.

IV. Motivational Example

In order to protect the communication against malicious snooping, encryption and decryption should be deployed on the messages. These operations are however computationally expensive, while automotive embedded systems have limited computation resources and stringent timing requirements. So it is indispensable to find a method that provides the best system-affordable cryptographic protection for the communication such that the resources and deadline constraints of the system are not violated. Let us consider the process graph in Fig. 1. The shortest end-to-end delay for this application is 535 time units, and the corresponding schedule is shown in Fig. 3 (a). As can be observed from this schedule, the system currently has time slacks that can be utilized to perform encryption and decryption operations within the deadline of 700 time units. The WCETs of doing pre/post-whitening and one E/D round on the two processors are 2 and 2 time units, respectively, for both processors.

A simple approach to protect the communication of the application would be to maximize the total number of E/D rounds over all messages, which leads to the schedule in Fig. 3 (b). According to this schedule, the number of rounds for m_2 and m_4 are 34 and 46 respectively (a total of 80 rounds), while m_1 and m_3 are not encrypted at all. This, very likely, is not a satisfactory solution, as two messages are left completely unprotected.

Another alternative solution would be to distribute the number of E/D rounds evenly to all messages. The best possible solution in this case is to assign 14 E/D rounds to each message transmitted over the bus, and the corresponding schedule is illustrated in Fig. 3 (c) with a total of 56 rounds. Assigning 16 [1] or more rounds to each message would violate the imposed deadline of 700. Nevertheless, the schedule in Fig. 3 (c) still contains slacks, which means that the number of E/D rounds for certain individual messages can still be increased. Fig. 3 (d) illustrates a solution in which the available slacks has been used to increase the security level of more messages. Now the numbers of E/D rounds for the messages become 14, 18, 14 and 16 (a total of 62 rounds), in which two messages are further encrypted by 4 and 2 more rounds respectively. This is especially useful in the case when different messages are encrypted with different keys. Thus, the system is further protected. The numbers of rounds used by the messages in the solutions mentioned above are presented in Table II.

[1]As discussed in Section II, the number of rounds should be increased by at least 2 when using RC6.

	m_1	m_2	m_3	m_4
Maximization of total rounds (Fig.3b)	0	34	0	46
Even slack distribution (Fig.3c)	14	14	14	14
Further rounds increment (Fig.3d)	14	18	14	16

TABLE II
Number of rounds for the messages for different solutions

V. Design Optimization Problem

The goal of this work is to optimize the protection level of messages transmitted over the bus using iterated block ciphers, considering the available processing power and the imposed timing constraints. The optimization problem is decomposed into two sub-problems that are performed in two steps:

a) Step 1: We firstly want to find the maximal number of rounds N that all E/D processes can perform, i.e. the system can be scheduled within the deadline if N rounds are used by all E/Ds. This step maximize the smallest number of rounds globally (see Fig. 3 (c) and the third row in Table II).

b) Step 2: After step 1, the system may still have extra unutilized slacks due to unbalanced workload and the diversity of the processing power of different processors, as discussed in Section IV. Hence, we can further increase the security strength of the internal communication by utilizing these slacks. By this, the window (the E/D modules running on the smallest number of rounds) revealing the vulnerability of the system is further dwindled. This is particularly effective if different E/D pairs use distinct secret keys to encrypt their communication.

In this step, we want to increase the number of E/D rounds as much as possible, while also trying to provide a certain balance between

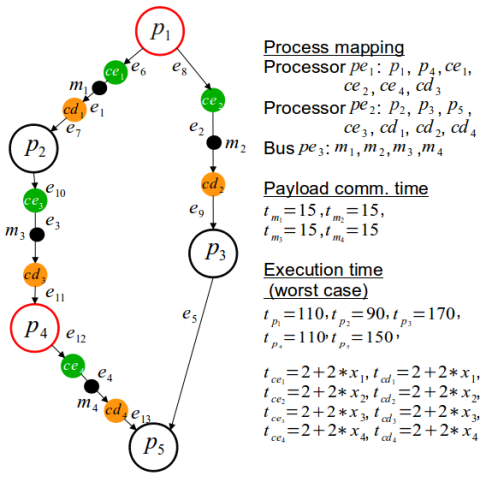

Fig. 4. Reconstructed process graph

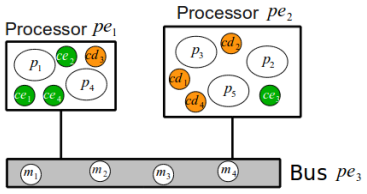

Fig. 5. New process mapping

the individual security degrees of the messages. These two aspects are captured by the following cost function which is driving the optimization in step 2:

$$Cost = \alpha * Avrg(X) - \beta * StandDev(X) \tag{3}$$

in which

$$Avrg(X) = \frac{1}{n} * \sum_{i=1}^{n} x_i \tag{4}$$

$$StandDev(X) = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (x_i - Avrg(X))^2} \tag{5}$$

α and β are the designer-provided weights depending on how balanced the security levels of different messages are desired.

In this paper, we consider that all messages on the network share the same security requirement. However, the proposed approach can be generalized in a straightforward way if messages have different security requirements. For example, in step 2, the number of E/D rounds can be weighted with a coefficient to capture the security demand of the actual message.

VI. Proposed Technique

Let us consider again the application in Fig. 1 and 2. In order to keep the messages confidential, we have to perform message encryption and decryption. Each message will be encrypted before being sent over the bus by the source processor, and decrypted when received by the destination processor. To capture these operations in our representation, we will make the procedure of encryption and decryption explicit. The newly added processes for encryption and decryption are mapped to the same processors as the corresponding sending and receiving processes. For example, encryption task ce_1 is mapped to processor pe_1, while the corresponding decryption task cd_1 to processor pe_2. The reconstructed model from Fig. 1 and 2 and the new hardware mapping are illustrated in Fig. 4 and Fig. 5 respectively.

A. CLP Formulation

We first formulate both the first and second step of our optimization problem using constraint logic programming (CLP). A constraint logic program contains constraints in the body of clauses, and allows users to formulate the problem as a process of constraint satisfaction. Then the CLP solver tries to solve the problem using methods like using branch and bound search. However, finding the best solution using CLP is computationally expensive.

In the following, we present the CLP constraints shared by both steps, which are divided into two sets. The first set contains the dependencies of the process graph, e.g., process ce_1 needs to be executed after process p_1 in Fig. 4. The second set includes the schedulability related constraints, e.g., process p_5 in Fig. 4 must successfully complete its execution before 700 time units. The constraints we used in our CLP formulation are as follows.

Dependency constraints: This set of constraints reflects the structure of the process graph. Each process is allowed to execute only after all its parent processes have terminated, i.e.,

$$\forall p_i \in P \text{ and } \forall p_j \in Parent(p_i),$$
$$StartTime(p_i) \geq StartTime(p_j) + WCET(p_j) \tag{6}$$

Schedulability constraints: There are three related sub-types of constraints in this set, which guarantee that the optimal result obtained is schedulable within the global time constraint, and also that the final schedule is correct.

- Execution time constraints:
  The non-E/D processes have constant WCETs, while the E/D processes have variable WCETs that are related to the optimization variable x_i as stated in equations (1) and (2). So, in order to capture this behavior, we assign fixed values as the WCETs for non-E/D processes, and restrict the WCETs of E/D processes to a domain containing all the possible values that they can take.
- Parallelization and sequence constraints:
  The target applications are mapped to distributed processors. Therefore the processes on different processors can run in parallel, while the executions of different processes on the same processor must not overlap with each other.
- Deadline constraint:
  As all dependencies have been defined, we formulate the deadline constraint as follows: the last task of the application must finish its execution before the deadline D, i.e.,

$$StartTime(p_{last}) + WCET(p_{last}) \leq D \tag{7}$$

Optimization objectives: The two steps have different objectives. In the first step, the objective is to find the maximal number of rounds N that all the E/D pairs can perform. While in the second step, the objective is to maximize the cost presented as equation (3) based on the number N obtained from the first step.

B. Heuristic Approach

The CLP formulation outlined above returns the optimal solution, but cannot scale to large systems due to its computation complexity. In this section, we describe our proposed heuristic for solving the optimization problem which can handle large designs efficiently.

Our heuristic approach is based on two well-known algorithms, list scheduling [16] and simulated annealing (SA) [17]. The former handles the schedulability test, and the latter is in charge of optimizing the cost function.

The first step is achieved by gradually adding two rounds to all E/D processes, whose WCETs are also increased by the WCETs of doing two rounds on corresponding processors, until the system cannot be scheduled within the deadline. The obtained number of rounds, N, is the largest number of rounds that all the E/D processes can perform without breaking the system schedulability. N will be usedby all E/Ds as the initial value for the second step. The pseudocode for the first step is shown in Algorithm 1.

After the first step, some E/D pairs may still have slacks for undertaking extra encryption and decryption rounds. So distributing these slacks in a smart way is crucial for getting closer result of $Cost$

Algorithm 1 Heuristic approach: Step 1

init WCETs of all E/Ds with corresponding w_{pe}
for $i = 2, 4, ..., Bound$ **do**
  increase WCETs of all E/Ds with corresponding $r_{pe} * 2$
  if $ListScheduling(G) > D$ **then**
    return N as $i - 2$ //largest rounds found
  end if
end for

Algorithm 2 Heuristic approach: Step 2

init all x_i with N and $cost$ with $currentcost$
init WCETs of E/Ds with $w_{pe} + r_{pe} * N$ of corresponding processors
while stopping condition is not reached **do**
  while executions on temperature level is not reached **do**
    randomly select edi and the move V to be performed
    if V will lead to $x_{edi} < N$ **then** continue **endif**
    update x_{edi}, $t_{ce_{edi}}$ and $t_{cd_{edi}}$ as V
    if ListScheduling(G) $> D$ **then**
      restore x_{edi}, $t_{ce_{edi}}$ and $t_{cd_{edi}}$ to the previous state
    else
      if $currentcost > cost$ **then**
        $cost = currentcost$
      else
        randomly generate $p \in (0, 1)$
        if $p < exp(-\delta/t)$ **then** $cost = currentcost$
        else restore x_{edi}, $t_{ce_{edi}}$ and $t_{cd_{edi}}$ to the previous state
        end if
      end if
    end if
  end while
  $t = \alpha * t$
end while

to the optimum. We have used a SA based heuristic for the second optimization step. At the beginning, $cost$ and all $x_i (i \in \{1, 2, ..., n_M\})$ are initialized with $currentcost$ and N, and the WCETs of the E/Ds are also initialized correspondingly. Then SA randomly selects one E/D pair edi to manipulate each time. Both increment and decrement manipulations are allowed and randomly decided to be performed on x_{edi}, but decrement moves leading to $x_{edi} < N$ are not allowed. If the updated system can be scheduled by our list scheduling algorithm, then the program proceeds to check whether $currentcost$ is higher than $cost$. Otherwise, this pair edi is restored to the previous state. Now if $currentcost$ is higher than $cost$, then $cost$ is updated to the $currentcost$. If not, a random number $p \in (0, 1)$ is generated and compared with $exp(-\delta/t)$ where t is the current temperature, and δ is the cost decrement caused by this move. The current state and $currentcost$ are kept only in case $p < exp(-\delta/t)$. This makes sure that our search will not be trapped in a local optimum.

The above optimization procedure will terminate when no acceptance occurs in a certain number of consecutive steps. The final $cost$ is the optimum approximation returned by our heuristic approach. The set of decision variables, $x_i (i \in \{1, 2, ..., n_M\})$, hold the numbers of rounds for all E/D pairs. The pseudocode for step 2 is given in Algorithm 2. The SA parameters, including initial temperature, temperature reduction scheme and stopping condition, are set by extensive experiments.

VII. Experimental Results

We have performed experiments on randomly generated process graphs with 10, 15, 20, 30, 40, 50, 60, 70, 80, 90 application processes (not including E/D processes) that are mapped on 2, 3, 3, 4, 4, 5, 5, 6, 6, 7 processors respectively. For each process graph size, 50 different applications are generated. All experiments were performed on a Linux machine having a four-core Intel Xeon CPU with 2.66GHz frequency and 8GB RAM.

The experiments with the CLP formulations were implemented in *ECLiPSe* constraint programming system [18], and were conducted with a timeout setup. If the CLP engine can find the optimal solution of an experiment before the timeout restriction, it returns the optimum for this application. Otherwise, it will be stopped with a timeout notification, without producing the optimum. This setup is used to make the extensive CLP experiments efficient; otherwise, the CLP engine may run for a long time for some cases, and even may not terminate. The computational complexity of the CLP executions is therefore characterized by two parameters. One is the average execution time of those experiments that terminate and return optimal solutions. The other is the percentage of cases that cannot retrieve the optimal solutions due to the timeout restriction. In contrast, the proposed heuristic algorithms, for both step 1 and step 2, always terminate and produce solutions.

The comparison of the average execution time (of both steps together) of the finished CLP experiments and corresponding heuristic executions is illustrated in Fig. 6. For the CLP experiments, of all the ten graph sizes, all the 50 designs retrieved the optimal solutions only on the smallest size 10. For graph size 15 and 20, there were 15 and 28 cases respectively out of 50 that failed in finding the optimums within the timeout limit. While, starting from size 30, the CLP engine cannot find the optimal solutions for any of the CLP executions any longer, so their execution times cannot be presented. As we can see, the execution time of the CLP experiments grows exponentially as the graph size increases. Therefore, finding the optimal solution using the CLP formulation, is only feasible for very small applications. Due to the limit of our experimental time, we set the timeout bounds of the first and second step as 300 and 1800 seconds respectively, which are relatively small, but the trend of complexity growth can still be clearly reflected.

Fig. 6 shows that the execution time of the proposed heuristics is much smaller than that of the CLP solutions. Of course, the execution time grows also aggressively due to the nature of the simulated annealing algorithm, but the average execution time of the heuristics for the biggest graphs (size 90) is only around 900 seconds.

The result comparisons of the two approaches are presented in Fig. 7 and 8 for step 1 and 2, respectively. When comparing the results, the experiments that CLP failed in finding the optimal solutions before timeout are eliminated. So we only analyze the CLP-obtained optimums and their corresponding heuristic-returned results. For example, only 22 experiments of graph size 20 are used to generate the data in Fig. 8. In both Fig. 7 and 8, the yellow bars without hatchings (left ones for size 10 and 15 in Fig. 7, and for size 10 in Fig. 8) indicate that all the CLP experiments found the optimal solutions, while the rest with hatchings present the results of only the finished CLP executions. The red bars are the results obtained by our heuristic approach. Fig. 7 illustrates the global N comparisons for step 1. Averagely, the results produced by the heuristic is 93.83% of the optimal. In the second step, the costs were obtained by setting the weights as $\alpha = 0.7$ and $\beta = 0.3$. As shown in Fig. 8, the final $Cost$ returned by our heuristic approach and the optimal solutions retrieved by CLP are presented, and 94.5% of the optimal result can be achieved by the heuristic approach. From graph size 30, the results of CLP formulations are missed since none of the experiments found the optimal solution within our timeout bound.

VIII. A real-life example

We also experimented with a real life example, an adaptive cruise controller (ACC), similar to the one described in [19]. The ACC application automatically maintains a safe following distance from the preceding vehicle. It also has the possibility of autonomous changes of the maximum speed depending on the speed-limit regulations and helps the driver with the braking procedure in extreme situations.

The ACC application is composed of 22 processes and 15 bus communications, and is constrained by a global deadline (500 time units). The processes are distributed to three processors connected by a bus. The original process graph and the WCET of the processes are depicted in Fig. 9, and the task-to-processor allocation is illustrated in Fig. 10.

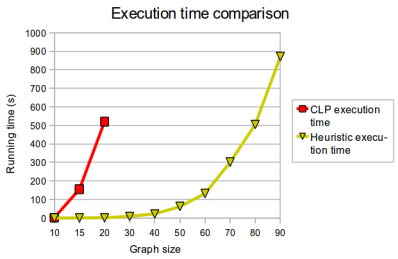

Fig. 6. Average execution time of finished CLP experiments and heuristic approach

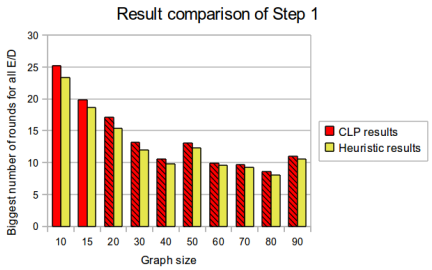

Fig. 7. Result comparison of step 1

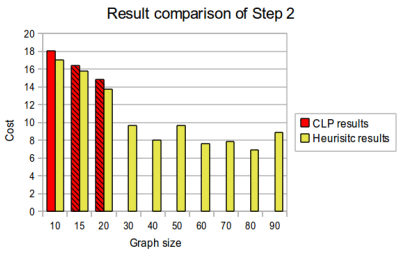

Fig. 8. Result comparison of step 2

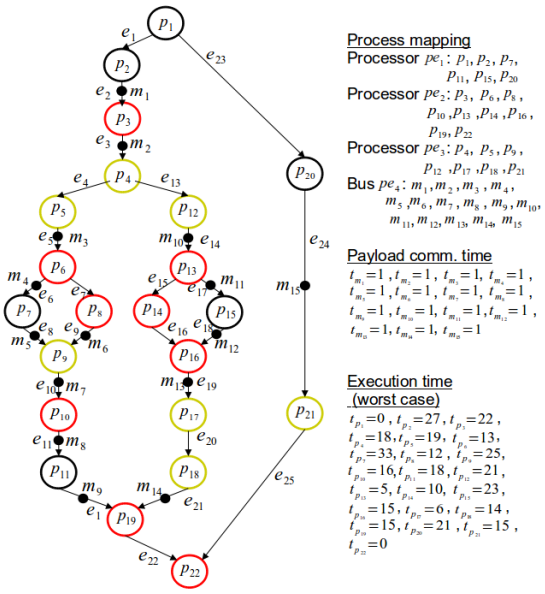

Fig. 9. Adaptive cruise controller

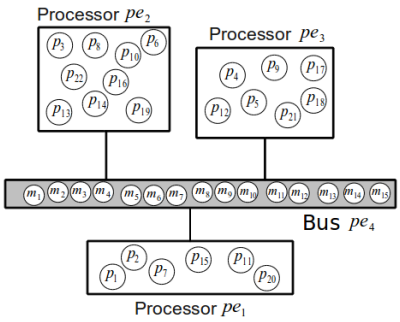

Fig. 10. Hardware architecture of ACC

The results returned by CLP and our heuristic approach are presented in Table III, in which N is the maximal number of rounds that all E/D processes can handle, and $Cost$ is the final optimization result with $\alpha = 0.7$ and $\beta = 0.3$. Time 1 and Time 2 indicate the execution time (in seconds) for the first and second step respectively. The CLP formulation for the second step cannot find the optimal solution using any of the available search methods within our timeout restriction of 1800 seconds. So no results are given in the "$Cost$" and "Time 2" columns of CLP formulation. As can be observed from the table, our heuristic found identical N as the optimum in the first step, and returns the result within a very short period.

	N	$Cost$	Time 1	Time 2
CLP	12	-	9	-
Heuristic	12	10.59	0.03	2.315

TABLE III
RESULT COMPARISON OF ACC

IX. CONCLUSION

In this paper, we have presented an optimization technique for protecting the confidentiality aspect of internal distributed embedded system communication using iterated block ciphers. We proposed a heuristic approach for solving the problem, and ran extensive experiments to prove the time efficiency and result quality. Although we have a focus on the automotive area, this work can also be adopted in other distributed real-time embedded systems where the internal communications need to be secured.

REFERENCES

[1] "CAN in Automation (CiA)," http://www.can-cia.org/.

[2] P. Koopman, "Embedded system security," *Computer*, vol. 37, pp. 95–97, 2004.

[3] P. Kocher, R. Lee, G. McGraw, and A. Raghunathan, "Security as a new dimension in embedded system design," in *DAC '04: Proceedings of the 41st annual Design Automation Conference*. New York, NY, USA: ACM, 2004, pp. 753–760.

[4] D. D. Hwang, P. Schaumont, K. Tiri, and I. Verbauwhede, "Securing embedded systems," *IEEE Security and Privacy*, vol. 4, no. 2, pp. 40–49, 2006.

[5] M. Wolf, A. Weimerskirch, and C. Paar, "Secure In-Vehicle Communication," in *Embedded Security in Cars*, 2006, pp. 95–109.

[6] M. Raya and J.-P. Hubaux, "Securing vehicular ad hoc networks," *Journal of Computer Security*, vol. 15, no. 1, pp. 39–68, 2007.

[7] B. Parno and A. Perrig, "Challenges in Securing Vehicular Networks," in *Proceedings of Workshop on Hot Topics in Networks (HotNets-IV)*, Nov. 2005.

[8] P. Golle, D. Greene, and J. Staddon, "Detecting and correcting malicious data in VANETs," in *VANET '04: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*. New York, NY, USA: ACM, 2004, pp. 29–37.

[9] "Security threats to automotive can networks–practical examples and selected short- term countermeasures," *Reliability Engineering & System Safety*, vol. In Press, Corrected Proof, 2010.

[10] T. Xie and X. Qin, "Improving security for periodic tasks in embedded systems through scheduling," *ACM Trans. Embed. Comput. Syst.*, vol. 6, no. 3, p. 20, 2007.

[11] J. G. Thomas Wollinger and C. Paar, "Cryptography in embedded systems: An overview (invited paper)," in *Proceedings of the Embedded World 2003 Exhibition and Conference*, 2003, pp. 735–744.

[12] C. H. Gebotys, "A table masking countermeasure for low-energy secure embedded systems," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 7, pp. 740 –753, jul. 2006.

[13] C. E. Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical Journal*, 1949.

[14] R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, "The rc6 block cipher," in *in First Advanced Encryption Standard (AES) Conference*, 1998, p. 16.

[15] L. B. W. B. M. D. J. F. E. R. James Nechvatal, Elaine Barker, "Report on the development of the advanced encryption standard (aes)," Tech. Rep., 2000.

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

[17] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi, "Optimization by simmulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[18] *Constraint Logic Programming using ECLiPSe*. Cambridge University Press, 2006.

[19] P. Pop, "Analysis and synthesis of communication-intensive heterogeneous real-time systems," Ph.D. dissertation, Linköping University, 2003.