Security-Aware Mapping for CAN-Based Real-Time Distributed Automotive Systems

Chung-Wei Lin*, Qi Zhu†, Calvin Phung†, and Alberto Sangiovanni-Vincentelli*
University of California, Berkeley, Berkeley, CA*, University of California, Riverside, Riverside, CA†
E-Mails: cwlin@eecs.berkeley.edu, qzhu@ee.ucr.edu, phungc@cs.ucr.edu, alberto@eecs.berkeley.edu

Abstract—**Cyber-security is a rising issue for automotive electronic systems, and it is critical to system safety and dependability. Current in-vehicles architectures, such as those based on the Controller Area Network (CAN), do not provide direct support for secure communications. When retrofitting these architectures with security mechanisms, a major challenge is to ensure that system safety will not be hindered, given the limited computation and communication resources. We apply Message Authentication Codes (MACs) to protect against masquerade and replay attacks on CAN networks, and propose an optimal Mixed Integer Linear Programming (MILP) formulation for solving the mapping problem from a functional model to the CAN-based platform while meeting *both the security and the safety requirements*. We also develop an efficient heuristic for the mapping problem under security and safety constraints. To the best of our knowledge, this is the first work to address security and safety in an integrated formulation in the design automation of automotive electronic systems. Experimental results of an industrial case study show the effectiveness of our approach.**

I. Introduction

Cyber-security has become a pressing issue for automotive electronic systems. In [4], [8], [9], [14], the authors demonstrated that modern vehicles can be attacked from various interfaces, including direct or indirect physical access, short-range wireless access, and long-range wireless channels. Furthermore, the authors showed that by infiltrating one automotive Electronic Control Unit (ECU) through those interfaces, the adversary can gain access to other ECUs via communication buses, including safety critical components such as brakes and engines, and conduct various attacks such as eavesdropping, spoofing, injection, and message replay. Such security challenges are expected to grow for automotive electronic systems, with the dramatic increase of their communication to external entities (in particular through telematics) and the resource sharing among different functionalities. Rather than assuming the attack interfaces will not be breached, it is important to harden the in-vehicle communications with security mechanism to limit the damages from compromised ECUs.

In in-vehicle communications, the Controller Area Network (CAN) is the most used protocol for safety critical applications, and it will likely be used for a long time to come in the future. It has been the most attractive protocol for attackers [6], [10], [12]. There is no direct support for secure communication in CAN, and the limitations on bus bandwidths and message lengths make it very challenging to embed security in CAN without hindering the safety applications [18]. Several approaches have been proposed to add Message Authentication Codes (MACs) in CAN data frames to provide message authentication [5], [11], [12], [16], [17]. In [12], a delayed data authentication method is proposed to detect and possibly recover from injection and modification attacks based on compound MACs transmitted over multiple messages. In [5], [11], [17], security mechanisms based on MACs and counters are proposed specifically for CAN. For instance, the mechanism in [11] can protect against the *masquerade attack*, where an attacker sends a message in which it claims to be a node other than itself, and the *replay attack*, where an attacker sends a message that it has received from the CAN bus without any modification.

However, adding MAC bits to an existing design may not lead to optimal or even feasible systems because

- There might not be enough space in messages for the required MAC bits because of the message length limitation (only 64 bits for payload in basic CAN protocol [1]).
- Adding MAC bits increases the message transmission time (in particular if MACs are truncated and transmitted over multiple messages), which may cause the violation of timing constraints and affect system safety.

Some extensions of CAN provide longer message length [2], [20]. For instance, the CAN with Flexible Data-Rate (CAN-FD) protocol [2] can allow messages with 64 bytes payload. However, the problems above may still exist if the MAC bits are added in an ad-hoc fashion or *after* the other parts of the design are done. Therefore, to achieve a secure and safe design, it is crucial to *address security together with other objectives* such as latency and utilization during the design space exploration of the system.

In this paper, we propose an integrated Mixed Integer Linear Programming (MILP) formulation to address both the security and the safety requirements during the exploration of the mapping from the functional model to the CAN-based architecture platform. The mapping *design space* we explore includes the allocation of tasks onto ECUs, the packing of signals into messages, the sharing of MACs among multiple receiving ECUs, and priority assignment of tasks and messages. The *security constraints* are set to prevent direct and indirect attacks on the MACs. We extend the security mechanism in [11] by allowing multiple receiving ECUs to share one MAC in a message. This provides more design flexibility under limited resources, while also requires quantitative measurement of the security cost in our formulation. The *safety constraints* are defined on the end-to-end latency deadlines for safety-critical functional paths. We extend the formulation proposed in [19] for the modeling of path latencies, by adding the consideration of MAC bits ([19] optimizes the path latencies without considering any security constraint).

To the best of our knowledge, this is the *first* work to address security and safety in an integrated formulation in the design automation of automotive electronic systems. Based on the optimal MILP formulation, we further propose a three-step algorithm that gradually solves the mapping problem in three simplified MILPs.

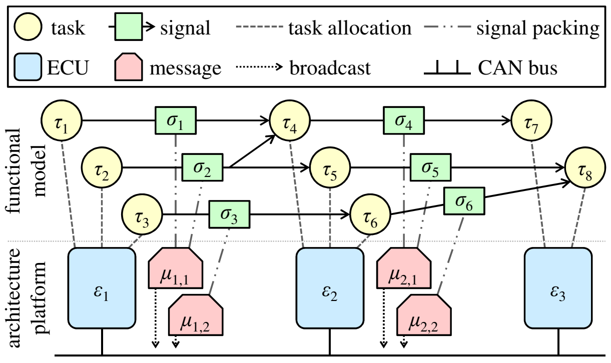

Fig. 1. System model of a CAN-based distributed system.

This approach balances optimality and efficiency, and enables solving complex industrial-size problems. We also propose a greedy heuristic algorithm for more efficiency. We apply our algorithms to an industrial case study and demonstrate their effectiveness in addressing both system security and safety.

The paper is organized as follows: Section II introduces the system model, the security model and constraints, and the end-to-end latency constraints. Section III presents the MILP formulation for the entire mapping problem, and the three-step MILP-based algorithm. Section IV introduces the greedy heuristic algorithm. Section V shows the experimental results, and Section VI concludes the paper.

II. System Model and Constraints

A. System Model

The mapping problem addressed in this paper is based on the Platform-Based Design paradigm [3], [7], [15], where the functional model and the architecture platform are initially captured separately and then brought together through a mapping process. As shown in Figure 1, the architecture model is a distributed CAN-based platform that consists of $n^{\mathcal{E}}$ ECUs, denoted by $\mathcal{E} = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{n^{\mathcal{E}}}\}$, and a CAN bus that connects all the ECUs. Each ECU ε_k can send $n_k^{\mathcal{M}}$ messages, denoted by $\mathcal{M}_k = \{\mu_{k,1}, \mu_{k,2}, \ldots, \mu_{k,n_k^{\mathcal{M}}}\}$. ECUs are assumed to run AUTOSAR/OSEK-compliant operation systems that support preemptive priority-based task scheduling [13]. The bus uses the standard CAN bus arbitration model that features non-preemptive priority-based message scheduling [1]. The functional model is a task graph that consists of $n^{\mathcal{T}}$ tasks, denoted by $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_{n^{\mathcal{T}}}\}$, and $n^{\mathcal{S}}$ signals, denoted by $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots, \sigma_{n^{\mathcal{S}}}\}$. Each signal σ_i is between a source task src_{σ_i} and a destination task dst_{σ_i}. Tasks are activated periodically and communicate with each other through signals.

A path π is an ordered interleaving sequence of tasks and signals, defined as $\pi = [\tau_{r_1}, \sigma_{r_1}, \tau_{r_2}, \sigma_{r_2}, \ldots, \sigma_{r_{k-1}}, \tau_{r_k}]$. $src(\pi) = \tau_{r_1}$ is the path's source and $snk(\pi) = \tau_{r_k}$ is its sink. Sources are activated by external events, while sinks activate actuators. The worst case end-to-end latency incurred when traveling a path π is denoted as l_π, which represents the largest possible time interval that is required for the change of the input (or sensed) value at the source to be propagated and cause a value change (or an actuation response) at the sink.

During mapping, the functional model is mapped onto the architecture platform, as shown in Figure 1. The design space of task allocation, signal packing, priority assignment and key sharing is explored with respect to a set of design objectives and constraints. For instance, a path deadline d_π may be set for path π as an application requirement, and we use $\mathcal{P}$ to denote the set of time-sensitive paths with such deadline requirements. There are also utilization constraints on ECUs and the CAN bus, payload size constraints on messages, and constraints on security costs. The details of the security and end-to-end latency constraints are introduced in the following sections, and all constraints are formulated in the MILP formulation in Section III.

B. Security Mechanism

We focus on using MACs to protect against two types of attacks within a CAN network—*masquerade attack* and *replay attack*, both from a compromised ECU to other ECUs on the same bus. A masquerade attack is the case that an attacker (a compromised node) sends a message in which it claims to be a node other than itself. A replay attack is the case that an attacker sends a message that it has received from the CAN bus without any modification.

We leverage the security mechanism proposed in [11] to protect against these two types of attacks. In [11], a shared secret key is distributed between *any* pair of nodes within a CAN network during the initialization and used to compute MACs during the operation. To protect against masquerade attacks, a message is sent with MACs (one for each receiver) so that each receiver can authenticate it by checking if the corresponding MAC sent with the message is equal to the MAC computed by itself. To further protect against replay attacks, a message is also sent with a counter so that a receiver can check if the message is fresh or not. Due to the limited size of the payload in a CAN message, only the least significant bits of the counter is sent with the message based on trade-off analysis, and reset mechanisms are provided to avoid out-of-sync counters [11]. It should be noted that the mechanism does not address Denial-of-Service (DoS) attacks which is believed to require additional hardware for CAN network.

C. Security Constraints and Key Distribution

MACs provide authentication to protect against masquerade and replay attacks, and security constraints should be set at the design time to assure there are enough MAC bits to prevent *indirect attack* on the MAC bits.

Definition 1: An **indirect attack** is the scenario that an attacker does not have the shared secret key between a sender and a receiver so that it can only guess a MAC and attempt to make a message accepted by the receiver.

If there is no prior information and the guess of MAC is purely random, the successful probability of an indirect attack is 2^{-L}, where L is the number of bits of the MAC. In our design formulation, a minimal number of bits is required for each MAC, based on the importance of the signals in the message and the importance of the receivers.

The security mechanism in [11] uses a dedicated shared key for any pair of nodes, in which case only indirect attacks need to be addressed. However, using such pair-wise key distribution may require a significant number of MAC bits when there are multiple receivers for a message, and may not be feasible. In this work, we extend the mechanism in [11] by defining the notion of *receiving group* to allow multiple receivers share one MAC in

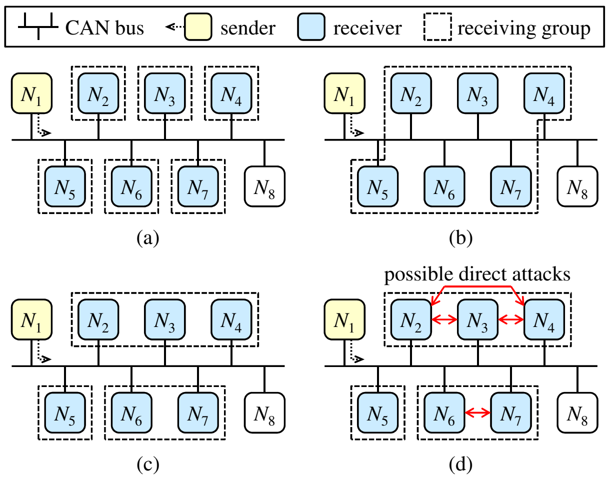

Fig. 2. Given a message sent by node N_1 and received by N_j ($2 \leq j \leq 7$), (a) the pair-wise key distribution, where 6 MACs are required to be sent with the message, and there is no possible direct attack; (b) the one-key-for-all key distribution, where only 1 MAC is required, but there are possible direct attacks between any pair of receivers; (c) another key distribution, where 3 MACs are required, and (d) there are some possible direct attacks.

a message, *i.e.*, using the same secret key. This provides more design flexibility but has the risk from *direct attack*.

Definition 2: A **receiving group** of a message is a set of receivers sharing one secret key with the sender of the message.

Definition 3: A **direct attack** is the scenario that an attacker gets the shared secret key between a sender and a receiver so that it can pretend as the sender and send a message to the receiver successfully.

In pair-wise key sharing as in [11], each receiving group contains only one receiver. The example in Figure 2 (a) shows that one MAC is used for each receiver in the message (6 receivers in total). There is no possibility for direct attack in this case. However, some MACs will not have enough bits available for preventing indirect attack (assuming 32 bits in the message payload are reserved for all MAC bits, then some MACs will have fewer than 6 bits, which means the successful probability of an indirect attack is higher than 3%).

The problem of limited MAC length can be relieved by allowing multiple receivers to share one MAC. A straightforward solution is to use one-key-for-all key distribution, where all receivers are in the same receiving group and use the same key (therefore the same MAC), as illustrated in Figure 2 (b). This will provide more bits for preventing indirect attack, but it may induce direct attacks—once one ECU in a receiving group is compromised, it can conduct direct attacks on all other ECUs in the same receiving group through masquerade attacks on the message.

In our design formulation, we explore the grouping of receivers into different receiving groups to trade off between direct attack risk and indirect attack risk, based on the total available MAC bits in a message, how critical a message is falsely accepted by a receiver, and how likely an existing node may be compromised. For instance, as illustrated in Figure 2 (c) and (d), if N_5 is extremely critical, then no other receiver will be assigned in its receiving group, and there will be no possible direct attack toward it. On the other hand, if N_6 and N_7 are trusted that they are very difficult to be compromised, then they can be assigned in the same receiving group because the probability of a direct attack between them is very low. We assume the factors that affect direct and indirect attack risks are quantitatively measured and given as parameters in the design inputs, and we set constraints in our formulation to restrict these risks.

D. End-to-End Latency and Response Times

An important aspect in our approach is to make sure the design with security mechanism still meets the end-to-end latency constraints along functional paths, which directly affect the safety of the system. Assuming an asynchronous sampling communication scheme, the worst-case end-to-end latency of a path π can be computed by adding the worst-case response times of all the tasks and global signals on the path, as well as the periods of all global signals and their destination tasks on the path, similarly as in [19]:

$$l_\pi = \sum_{\tau_i \in \pi} r_{\tau_i} + \sum_{\sigma_i \in \pi \wedge \sigma_i \in \mathcal{S}_G} (r_{\sigma_i} + T_{\sigma_i} + T_{dst_{\sigma_i}}), \qquad (1)$$

where r_{τ_i} and r_{σ_i} are the response times of task τ_i and signal σ_i, respectively; T_{τ_i} and T_{σ_i} are the periods of τ_i and σ_i, respectively; $\mathcal{S}_G$ is the set of all global signals. The key for calculating end-to-end latency and resource scheduling is to compute the response times of tasks and messages (the response time of a signal is equal to the response time of the message to which the signal is packed into), and they have been defined in [19].

III. MILP-Based Mapping Algorithm

A. Definitions

The notations of the indices, elements, sets, quantities are listed in Table I. The notations of the constant parameters are listed in Table II, and we assume these parameters are given as design inputs. $R_{i,j}$ is decided by how critical $\sigma_{i,j}$ is. $R_{i,j,k',k''}$ depends on how likely $\varepsilon_{k'}$ may be taken control by a malicious attacker and how much the computation of $\varepsilon_{k''}$ depends on $\sigma_{i,j}$. $L_{i,j}$ includes the payload data length and also the length of its corresponding counter, which is decided in advance by checking the given bound of the probability of a false rejection [11]. $L'_{i,j,k}$ is decided by checking the given bound of the probability of a false acceptance [11]. $R_{i,j}$ and $R_{i,j,k',k''}$ address the security risk of a direct attack, and $L'_{i,j,k}$ addresses the security risk of an indirect attack. The current maximum allowed security risk is defined at signal-level, but it can also be defined at receiver-level or at system-level with minor modifications. Finally, the notations of the decision variables are also listed in Table II.

B. Constraints

In this section, we introduce the various constraints on allocation, security cost, and end-to-end latency[1].

1) Allocation and Packing Constraints:

$$\forall i, \qquad \sum_k a_{i,k} = 1; \qquad (2)$$

$$\forall i,j,k, \qquad a_{i,k} + a_{j,k} + s_{i,j} \neq 2. \qquad (3)$$

[1]If there is no specific mention, the ranges of variables are $1 \leq i,j \leq n^{\mathcal{T}}$, $1 \leq k \leq n^{\mathcal{E}}$, $1 \leq l \leq n^{\mathcal{M}}_k$, $1 \leq g \leq n^{\mathcal{G}}_{k,l}$, and $1 \leq h \leq n^{\mathcal{P}}$. If a constraint is trivial for all tasks, signals, ECUs, messages, receiving group, or paths, then its "$\forall$" may be omitted in the following paragraphs.

TABLE I. The notations of indices, elements, sets, and quantities.

i, i', j, j'	the index of a task.
k, k', k''	the index of an ECU.
l, l', l''	the index of a message sent from an ECU.
m	the index of a multicast signal from a task.
g	the index of a receiving group of a message.
h	the index of a path.
τ_i	the i-th task.
$\sigma_{i,j}$	a signal between τ_i and τ_j.
ε_k	the k-th ECU.
$\mu_{k,l}$	the l-th message of ε_k.
$\Gamma_{k,l,g}$	the g-th receiving group of $\mu_{k,l}$.
π_h	the h-th path.
$\mathcal{T}$	the set of tasks.
$\mathcal{S}$	the set of signals.
$\mathcal{E}$	the set of ECUs.
$\mathcal{M}$	the set of messages.
$\mathcal{G}_{k,l}$	the set of receiving groups of $\mu_{k,l}$.
$\mathcal{P}$	the set of paths.
$\mathcal{T}^{<}_{i,m}$	the set of receiving tasks of the m-th multicast signal of τ_i.
$n^{\mathcal{T}}$	the number of tasks.
$n^{\mathcal{S}}$	the number of signals.
$n^{\mathcal{E}}$	the number of ECUs.
$n^{\mathcal{M}}_k$	the number of messages of ε_k.
$n^{\mathcal{G}}_{k,l}$	the number of receiving groups of $\mu_{k,l}$.
$n^{\mathcal{P}}$	the number of paths.

Equation (2) guarantees that τ_i is allocated to exactly one ECU[2]. Equation (3) guarantees that $s_{i,j} = 1$ if and only if there exists k such that $a_{i,k} = a_{j,k} = 1$, satisfying the definition of $s_{i,j}$.

$$\forall \sigma_{i,j} \in \mathcal{S}, k, \qquad \sum_{l} t_{i,j,k,l} = a_{i,k}(1 - a_{j,k}); \tag{4}$$

$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \qquad t_{i,j,k,l} \le v_{k,l}; \tag{5}$$

$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \qquad t_{i,j,k,l} T^{\mu}_{k,l} \le T^{\sigma}_{i,j}; \tag{6}$$

$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \qquad t_{i,j,k,l} T^{\sigma}_{i,j} \le T^{\mu}_{k,l}. \tag{7}$$

Equation (4) guarantees that $\sigma_{i,j}$ is packed into exactly one message from ε_k, if its source ECU is ε_k and its target ECU is not ε_k. Equation (5) guarantees that $v_{k,l} = 1$ if there exists a signal packed into $\mu_{k,l}$. Equations (6) and (7) guarantee that the period of a signal is equal to the period of the message in which the signal is packed into ($T^{\sigma}_{i,j} = T^{\mu}_{k,l}$ if $t_{i,j,k,l} = 1$).

$$\forall i, k, l, m, \forall \tau_j, \tau_{j'} \in \mathcal{T}^{<}_{i,m}, \; t_{i,j,k,l} = t_{i,j',k,l}; \tag{8}$$

$$\forall i, k, l, m, \forall \tau_j \in \mathcal{T}^{<}_{i,m}, \; t_{i,j,k,l} = \sum_{\tau_{j'} \in \mathcal{T}^{<}_{i,m}} u_{i,j',k,l}. \tag{9}$$

Equation (8) guarantees that each branch of a multicast signal is mapped to the same message. Equation (9) guarantees that exactly one branch of a multicast signal adds its length to the message.

2) Security Constraints:

$$\forall \sigma_{i,j} \in \mathcal{S}, k', k, l, \qquad a_{j,k'} + t_{i,j,k,l} - 1 \le w_{k',k,l}; \tag{10}$$

$$\forall k', k, l, \qquad \sum_{g} x_{k',k,l,g} = w_{k',k,l}; \tag{11}$$

$$\forall k', k, l, g, \qquad x_{k',k,l,g} \le y_{k,l,g}. \tag{12}$$

Equation (10) guarantees that $\varepsilon_{k'}$ is a receiver of $\mu_{k,l}$ if there exists a signal $\sigma_{i,j}$ such that τ_j is mapped to $\varepsilon_{k'}$ and $\sigma_{i,j}$ is mapped to $\mu_{k,l}$. Equation (11) guarantees that each receiver is in exactly one receiving group. Equation (12) guarantees that $y_{k,l,g} = 1$ if there exists a signal mapped to $\mu_{k,l}$ and the signal is in the receiving group $\Gamma_{k,l,g}$.

[2]In some cases, a task τ_i can only be allocated to a specific ECU ε_k. Then, $a_{i,k}$ should be assigned to 1 directly, and $a_{i,k'}$ is assigned to 0 if $k' \neq k$.

TABLE II. The notations of constant parameters, binary variables (their values are 1 if the conditions are true), and real variables.

T^{τ}_i	the period of τ_i.
$T^{\sigma}_{i,j}$	the period of $\sigma_{i,j}$.
$T^{\mu}_{k,l}$	the period of $\mu_{k,l}$.
A	the transmission rate of the CAN bus.
$B_{k,l}$	the blocking time of $\mu_{k,l}$.
$C_{i,k}$	the computation time of τ_i on ε_k.
D_h	the deadline of π_h.
$R_{i,j}$	the maximum allowed security risk of $\sigma_{i,j}$.
$R_{i,j,k',k''}$	the security risk if $\varepsilon_{k'}$ and $\varepsilon_{k''}$ share the corresponding secret key of $\sigma_{i,j}$.
$L_{i,j}$	the data length of $\sigma_{i,j}$.
$L_{k,l,g}$	the reserved MAC length of $\Gamma_{k,l,g}$.
$L'_{i,j,k}$	the required MAC length of $\sigma_{i,j}$ if $\sigma_{i,j}$ is received by ε_k.
M	a large constant for linearization.
H	total length of non-payload part of a message
P	maximum length of payload part of a message
$a_{i,k}$	τ_i is mapped to ε_k.
$s_{i,j}$	τ_i and τ_j are mapped to the same ECU.
$t_{i,j,k,l}$	$\sigma_{i,j}$ is mapped to $\mu_{k,l}$.
$u_{i,j,k,l}$	$\sigma_{i,j}$ adds its length to $\mu_{k,l}$.
$v_{k,l}$	$\mu_{i,j}$ is non-empty.
$w_{k',k,l}$	$\varepsilon_{k'}$ is a receiver of $\mu_{k,l}$.
$x_{k',k,l,g}$	$\varepsilon_{k'} \in \Gamma_{k,l,g}$.
$y_{k,l,g}$	$\Gamma_{k,l,g}$ is non-empty.
$z_{k',k'',k,l}$	$\varepsilon_{k'}$ and $\varepsilon_{k''}$ are in the same receiving group of $\mu_{k,l}$.
$p_{i,j}$	τ_i has a higher priority than τ_j.
$p_{k,l,k',l'}$	$\mu_{k,l}$ has a higher priority than $\mu_{k',l'}$.
r^{τ}_i	the response time of τ_i.
$r^{\mu}_{k,l}$	the response time of $\mu_{k,l}$.
$r^{\sigma}_{i,j}$	the response time of $\sigma_{i,j}$.
$b_{k,l}$	the total length of $\mu_{k,l}$.
$c_{k,l}$	the computation time of $\mu_{k,l}$.

$$\forall k', k'', k, l, g, x_{k',k,l,g} + x_{k'',k,l,g} + z_{k',k'',k,l} \neq 2. \tag{13}$$

Equation (13) guarantees that $z_{k',k'',k,l} = 1$ if and only if there exists g such that $x_{k',k,l,g} = x_{k'',k,l,g} = 1$, satisfying the definition of $z_{k',k'',k,l}$.

$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \qquad \sum_{k',k''} t_{i,j,k,l} \times w_{k',k,l} \times w_{k'',k,l} \times z_{k',k'',k,l} \times R_{i,j,k',k''} \le R_{i,j}; \tag{14}$$

$$\forall \sigma_{i,j} \in \mathcal{S}, k', k, l, g, \qquad a_{j,k'} \times t_{i,j,k,l} \times x_{k',k,l,g} \times L'_{i,j,k'} \le L_{k,l,g}. \tag{15}$$

Equation (14) guarantees that the security risk (cost) is not larger than the maximum allowed security risk (cost). Equation (15) guarantees that the required MAC length is not larger than the reserved MAC length.

Note that the impact of ECUs being compromised is considered in risk parameters $R_{i,j,k',k''}$. As mentioned before, $R_{i,j,k',k''}$ depends on how likely $\varepsilon_{k'}$ may be taken control by a malicious attacker and how much the computation of $\varepsilon_{k''}$ depends on $\sigma_{i,j}$. Such relation can also be modeled *explicitly* by first introducing parameters R_k as the possibility of ε_k being compromised and then modeling $R_{i,j,k',k''}$ as a linear function of R_k and other factors. In this work, we focus on addressing the masquerade and replay attacks on security-critical messages and assume $R_{i,j,k',k''}$ are given.

3) End-to-End Latency Constraints: For end-to-end latency constraints, we first model the priority assignment, and then compute the task and message response times, and finally set up the latency constraints on paths.

$$p_{i,j} + p_{j,i} = 1; \tag{16}$$
$$p_{i,j} + p_{j,j'} - 1 \leq p_{i,j'}; \tag{17}$$
$$p_{k,l,k',l'} + p_{k',l',k,l} = 1; \tag{18}$$
$$p_{k,l,k',l'} + p_{k',l',k'',l''} - 1 \leq p_{k,l,k'',l''}. \tag{19}$$

Equations (16), (17), (18), and (19) guarantee that the priority assignment is feasible.

$$\forall i, \qquad r_i^\tau = \sum_k a_{i,k} \times C_{i,k} + \sum_j \sum_k a_{i,k} \times a_{j,k} \times p_{j,i} \times \left\lceil \frac{r_i^\tau}{T_j^\tau} \right\rceil \times C_{j,k}. \tag{20}$$

Equation (20) computes the task response time of τ_i.

$$b_{k,l} = H + \sum_{\sigma_{i,j} \in \mathcal{S}} u_{i,j,k,l} L_{i,j} + \sum_g y_{k,l,g} L_{k,l,g}; \tag{21}$$
$$b_{k,l} \leq H + P; \tag{22}$$
$$c_{k,l} = \frac{b_{k,l}}{A}. \tag{23}$$

Equation (21) computes the total length of $\mu_{k,l}$, taking into account of the data payload, the counter, and the MAC length. Equation (22) guarantees that the total message length does not exceed the limit. Equation (23) computes the computation time of $\mu_{k,l}$.

$$\forall k, l, \qquad r_{k,l}^\mu = B_{k,l} + c_{k,l} + \sum_{k',l'} v_{k',l'} \times p_{k',l',k,l} \times \left\lceil \frac{r_{k,l}^\mu - c_{k,l}}{T_{k'l'}^\mu} \right\rceil \times c_{k',l'}. \tag{24}$$

Equation (24) computes the message response time of $\mu_{k,l}$.

$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \qquad r_{k,l}^\mu - M(1 - t_{i,j,k,l}) \leq r_{i,j}^\sigma; \tag{25}$$
$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \qquad r_{i,j}^\sigma \leq r_{k,l}^\mu + M(1 - t_{i,j,k,l}); \tag{26}$$
$$\forall \sigma_{i,j} \in \mathcal{S}, \qquad r_{i,j}^\sigma \leq M(1 - s_{i,j}). \tag{27}$$

Equations (25), (26), and (27) compute the signal response time of $\sigma_{i,j}$. If $\sigma_{i,j}$ is mapped to $\mu_{k,l}$, then $r_{i,j}^\sigma = r_{k,l}^\mu$; otherwise, if it is not mapped to any message (its source ECU and target ECU are the same), $r_{i,j}^\sigma = 0$.

$$\sum_{\tau_i \in \pi_h} r_i^\tau + \sum_{\sigma_{i,j} \in \pi_h} (r_{i,j}^\sigma + (1 - s_{i,j})(T_{i,j}^\sigma + T_j^\tau)) \leq D_h. \tag{28}$$

Equation (28) computes the path latency of π_h and guarantees that its deadline is satisfied.

4) Conversion to Linear Constraints: In above formulation of the constraints, there are four cases where the formulation is not linear. The first three cases can be converted into equivalent linear formulations based on their specific representations. The fourth case is more general and can be converted into equivalent linear formulations by introducing a large constant M, similarly as in [19]. The details of the conversions to linear constraints are explained as follows.

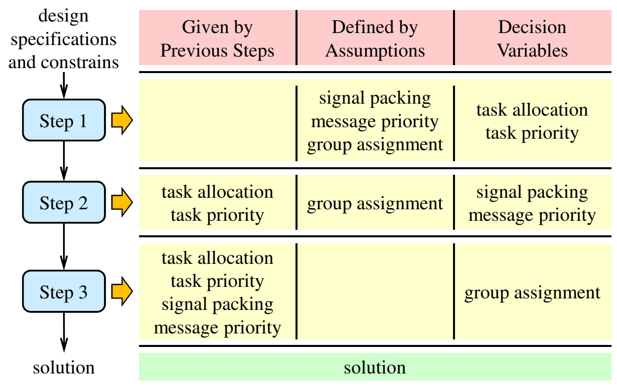
design specifications and constrains
Given by Previous Steps
Defined by Assumptions
Decision Variables
Step 1
signal packing message priority group assignment
task allocation task priority
Step 2
task allocation task priority
group assignment
signal packing message priority
Step 3
task allocation task priority signal packing message priority
group assignment
solution
solution

Fig. 3. The flow of three-step MILP-based algorithm, where "group assignment" means "receiving group assignment."

Inequalities of summations of three binary variables in Equations (3) and (13): if α, β, and γ are binary variables, then we replace the constraint $\alpha + \beta + \gamma \neq 2$ by three constraints: $\alpha + \beta - \gamma \leq 1$, $\alpha - \beta + \gamma \leq 1$, and $-\alpha + \beta + \gamma \leq 1$.

Ceiling functions in Equations (20) and (24): if α is a function and $\lceil \alpha \rceil$ exists in a constraint, then we replace $\lceil \alpha \rceil$ by an integer variable β and add one constraint: $0 \leq \beta - \alpha \leq 1$, which is a linear constraint if α is a linear function.

Multiplications of two binary variables in Equations (4), (14), (15), (20), and (24): if α and β are binary variables and $\alpha \times \beta$ exists in a constraint, then we replace $\alpha \times \beta$ by a binary variable γ in the constraint and add one constraint: $\alpha \times \beta = \gamma$. Next, $\alpha \times \beta = \gamma$ can be replaced by equivalent constraints: $\alpha + \beta - 1 \leq \gamma$, $\gamma \leq \alpha$, and $\gamma \leq \beta$. In fact, if $1 \leq i \leq n$, α_i is a binary variable, and $\prod_{1 \leq i \leq n} \alpha_i$ exists in a constraint, then we can replace $\prod_{1 \leq i \leq n} \alpha_i$ by a binary variable γ in the constraint and add one constraint: $\prod_{1 \leq i \leq n} \alpha_i = \gamma$. Next, $\prod_{1 \leq i \leq n} \alpha_i = \gamma$ can be replaced by equivalent constraints: $\sum_{1 \leq i \leq n} \alpha_i - (n-1) \leq \gamma$ and $\forall i, \gamma \leq \alpha_i$.

Multiplications of one binary variable and one non-integer variable in Equations (20) and (24): if α is a binary variable, β is a non-integer variable, and $\alpha \times \beta$ exists in a constraint, then we replace $\alpha \times \beta$ by a non-integer variable γ in the constraint and add one constraint: $\alpha \times \beta = \gamma$. Next, $\alpha \times \beta = \gamma$ can be replaced by equivalent constraints: $0 \leq \gamma \leq \beta$ and $\beta - M(1-\alpha) \leq \gamma \leq M\alpha$.

C. Objective Function

The objective function can be defined to minimize the summation of the end-to-end latencies of selected paths:

$$l_{\pi_h} = \sum_{\tau_i \in \pi_h} r_i^\tau + \sum_{\sigma_{i,j} \in \pi_h} (r_{i,j}^\sigma + (1 - s_{i,j})(T_{i,j}^\sigma + T_j^\tau)); \qquad \min \sum_{\pi_h \in \mathcal{P}} l_{\pi_h}, \tag{29}$$

where l_{π_h} is the path latency of π_h and $\mathcal{P}$ is the set of selected paths. The objective function can also be defined to minimize the total security risk with minor modifications.

D. MILP-Based Algorithm

The MILP formulation introduced in Section III provides an optimal solution but has high complexity. To address complex industrial-size problems, we propose a three-step algorithm, where each step solves part of the mapping problem in a simplified MILP formulation (derived from the original optimal MILP). The flow of the algorithm is shown in Figure 3.

In Step 1, we assume that (1) each message is only reserved for one signal, (2) a MAC with maximum required length for the signal is included in each message, and (3) the priorities of the messages are assigned by the Rate Monotonic policy, *i.e.*, messages with smaller periods have higher priorities. Based on these assumptions, we simplify the MILP formulation introduced above and optimize the task allocation and the task priority.

In Step 2, having the task allocation and the task priority from Step 1, we optimize the signal packing and the message priority in a simplified MILP formulation, with the assumption that a MAC with maximum required length for the signal is included in each message.

In Step 3, having the task allocation, the signal packing, and the task and message priorities from previous two steps, we optimize the receiving group assignment. For each message, its length is minimized, and its receiving group assignment satisfies the constraints of security risks (from the perspective of protecting against a direct attack) and required MAC lengths (from the perspective of protecting against an indirect attack).

The formulation of Step 1 is motivated by the observation that task allocation and priority typically have the most significant impact on path latencies and resolving them significantly simplifies the problem for the following steps. The division of Step 2 and Step 3 further reduces the complexity. In addition, if designers are given an existing mapped system and wants to improve the security level by exploring different key sharing strategies, the MILP formulation in Step 3 can be directly applied.

IV. Heuristic Algorithm

Algorithm 1 Heuristic for security-aware mapping

```
1: For each pair $(\tau_i, \tau_j)$, $\mathcal{S}_{i,j} = \{\sigma_k | \sigma_k \in \mathcal{S} \wedge src_{\sigma_k} = \tau_i \wedge dst_{\sigma_k} = \tau_j\}$
2: For each pair $(\tau_i, \tau_j)$, $w_{\tau_i,\tau_j} = \sum_{\sigma_k \in \mathcal{S}_{i,j}} \frac{W_l \left(T_{\sigma_k} + T_{\tau_j}\right)}{T_M} + \frac{W_s L_{\sigma_i}^{MAC}}{L_M^{MAC}}$
3: Set a priority queue $Q_w$ that consists of all $w_{\tau_i,\tau_j}$
4: while $Q_w \neq \emptyset$ do
5:     $w_{\tau_i,\tau_j} = extract\_max(Q_w)$
6:     if $mergeable\_tasks(\tau_i, \tau_j)$ then
7:         Assign $\tau_i$, $\tau_j$ to the same ECU
8: for all $\tau_i$ that has not been assigned an ECU do
9:     Assign $\tau_i$ to the feasible ECU $\varepsilon_k$ that provides smallest $C_{\tau_i,\varepsilon_k}$
10: Assign each signal into its own message
11: while $\exists$ message $m_i$ and $m_j$ s.t. $mergeable\_msgs(m_i, m_j)$ do
12:     Merge $m_i$ and $m_j$
13: Assign priorities to tasks and messages using Rate Monotonic policy
14: Calculate the response times for each task and message
15: Calculate the end-to-end latency for each path
16: Calculate the design objective
```

For comparison with the MILP-based algorithm, we also propose a greedy heuristic algorithm, as shown in Algorithm 1. First, we calculate a weight between each task pair (τ_i, τ_j) that represents an estimation of how much benefit we can gain by making the signals between the two tasks local signals (*i.e.*, mapping the two tasks onto the same ECU). This estimation depends on the potential gain from reducing the path latency and the potential gain on security (local signals are not at risk of masquerade and replay attacks), as shown in line 2. W_l and W_s are weights that can be tuned to give more emphasis to either latency or security. T_M is the largest period of all tasks and signals, and L_M^{MAC} is the largest required MAC length of all signals. $L_{\sigma_i}^{MAC}$ is the required MAC length for signal σ_i.

Following the descending order of w_{τ_i,τ_j}, we will try to cluster the tasks and assign them onto the same ECU (lines 3–7). In the $mergeable\_tasks$ function, we check whether we can assign two tasks onto the same ECU without violating utilization constraints (if the two tasks are already assigned to different ECUs, we check whether we can assign all the tasks from these two ECUs onto one ECU). Once task allocation is decided, we pack the signals into messages (lines 10–12) in a greedy fashion, *i.e.*, we will merge two messages as long as the combined message satisfies size constraints and we merge MACs as long as the security constraints permit. Finally, we assign priorities based on the Rate Monotonic policy.

V. Experimental Results

We obtained the test case that is used in [19]. The test case supports advanced distributed functions with end-to-end computations collecting data from 360-degree sensors to the actuators, consisting of the throttle, brake and steering subsystems and of advanced Human-Machine Interface devices. The architecture platform consists of 9 ECUs connected through a single CAN [1] or CAN-FD [2] bus with the speed 500kb/s. The functional model consists of 41 tasks and 83 signals. For the safety requirements, 171 paths are selected with deadlines 300ms or 100ms. For the security requirements, 50 signals are selected with required MAC lengths ranging from 30 bits to 10 bits for CAN and from 128 bits to 64 bits for CAN-FD (with longer message length, CAN-FD is able to provide more MAC bits and therefore more secure communications). The maximum allowed security risk of each signal is simplified so that no more than 2 ECUs can be assigned to the same receiving group, *i.e.*, $2 \leq \frac{R_{i,j}}{R_{i,j,k',k''}} < 3$ in Equation (14). The program is implemented in C/C++. CPLEX 12.5 is used as the MILP solver. The experiments were run on a 2.5-GHz processor with 4GB RAM. We compare our MILP-based algorithm with the heuristic algorithm in Section IV and non-integrated approaches applying the pair-wise key distribution in [11] and the one-key-for-all key distribution.

A. Comparison with the Greedy Heuristic Algorithm

The results are listed in Table III. For the basic CAN protocol, our MILP-based algorithm can find a solution satisfying all the design constraints. In Step 1, within 3,600 seconds, the objective of the best solution found by the solver is 11,070.61ms. The largest latencies among the paths with deadlines 300ms and 100ms are 127.92ms and 90.72ms, respectively. In Step 2, the program ends in 600 seconds, and the objective is 11,069.88ms. The largest latencies among the paths with deadlines 300ms and 100ms are 127.82ms and 90.62ms, respectively. In Step 3, the program ends in few seconds, and the objective is 11,069.62ms. The largest latencies among the paths with deadlines 300ms and 100ms are 127.79ms and 90.59ms, respectively. There is little improvement on the latency objective at Steps 2 and 3 because the message response times are much smaller compared to the task and message periods that contribute to the path latencies in our model. However, Steps 2 and 3 can significantly reduce the bus load from 76.92kb/s to 45.57kb/s and 31.52kb/s, respectively.

In comparison, the objective of the greedy heuristic is 23,114.50ms, while satisfying all the design constraints. Its

TABLE III. The objective (the latencies summation of selected paths), maximum latencies, load, and runtime of each step of the MILP-based algorithm, where "Max L_{300}" and "Max L_{100}" are the largest latencies among the paths with deadlines 300ms and 100ms, respectively.

Protocol	Step X	Results after Step X				
		Objective (ms)	Max L_{300} (ms)	Max L_{100} (ms)	Bus Load (kb/s)	Runtime (s)
CAN	1	11,070.61	127.92	90.72	76.92	$3,600$
	2	11,069.88	127.82	90.62	45.57	< 600
	3	11,069.62	127.79	90.59	31.52	< 10
CAN-FD	1	11,075.08	128.56	91.22	211.74	$3,600$
	2	11,073.67	128.39	91.05	176.47	< 600
	3	11,071.69	128.14	90.80	98.33	< 10

runtime is 1.4 seconds, but the value of the objective is much worse than the one obtained with the MILP formulation. This is because the exploration is stuck at a local minimum, which is a common problem for heuristic algorithms (we set different weights W_l and W_s for latency and security in Algorithm 1 but the results do not change significantly).

For the CAN-FD protocol, we can also find a solution satisfying all the design constraints. The objectives, the largest latencies, and the bus loads are increased because the required MAC lengths are much longer (128 bits to 64 bits). Similarly, Steps 2 and 3 reduce the bus load from 211.74kb/s to 176.47kb/s and 98.33kb/s, respectively, showing the effectiveness of signal packing and our flexible key distribution scheme. On the other hand, the greedy heuristic cannot find a feasible solution in this case (with bus speed at 500kb/s) due to the much longer required MAC lengths. In fact, we find that the heuristic can only find a feasible solution when we increase the bus speed to 4Mb/s.

B. Comparison with Non-Integrated Approaches

We also tried two experiments in which we do not consider security constraints explicitly at Steps 1 and 2 (*i.e.*, solving a traditional mapping problem with only timing constraints), and then explore the addition of MAC bits and the key distribution at Step 3. In the first experiment, at Steps 1 and 2, we constrain that all messages should have at most 32 bits used for data while packing signals, *i.e.*, leaving 32 bits available for MAC bits. Then, at Step 3, we find that there is no feasible solution for either the pair-wise key distribution in [11] or the one-key-for-all key distribution. The reason is that the pair-wise key distribution requires more than 32 MAC bits for certain messages, while the one-key-for-all key distribution leads to too high security risks for some messages. In the second experiment, at Steps 1 and 2, we do not set any constraint on the number of bits for data, *i.e.*, they may use as many as all 64 bits in the payload. Then, at Step 3, we find that there is no feasible solution for the pair-wise key distribution, the one-key-for-all key distribution, or our flexible key distribution scheme. This is because some messages use almost all 64 bits, so no MAC can be added to those message. The results from these two experiments demonstrate that it is necessary to consider security together with other metrics during mapping; otherwise, it may be difficult or even impossible to add security measurement later.

VI. Conclusion

In this paper, we proposed an approach to address both the security and the safety in the design space exploration of automotive electronic systems. We presented an MILP formulation that explores task allocation, signal packing, MAC sharing, and priority assignment while meeting both security and safety constraints. Experimental results of an industrial case study showed that our approach can effectively explore the design space to meet the system security and safety requirements.

Acknowledgment

This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP), a Semiconductor Research Corporation program sponsored by MARCO and DARPA. This work was supported in part by the Industrial Cyber-Physical Systems Center (iCyPhy).

References

[1] Bosch, CAN Specification, version 2.0, 1991.

[2] Bosch, CAN with Flexible Data-Rate White Paper, version 1.1, 2011.

[3] L. Carloni, F. D. Bernardinis, C. Pinello, A. Sangiovanni-Vincentelli, and M. Sgroi, "Platform-based design for embedded systems," Embedded Systems Handbook, CRC Press, 2005.

[4] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," USENIX Conference on Security, 2011.

[5] B. Groza, S. Murvay, A. Van Herrewege, and I. Verbauwhede "LiBrA-CAN: a lightweight broadcast authentication protocol for Controller Area Networks," International Conference on Cryptology and Network Security, pp. 185–200, 2012.

[6] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN networks—practical examples and selected short-term countermeasures," International Conference on Computer Safety, Reliability, and Security, pp. 11–25, 2008.

[7] K. Keutzer, S. Malik, A. R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, "System level design: orthogonolization of concerns and platform-based design," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, no. 12, pp. 1523–1543, 2000.

[8] P. Kleberger, T. Olovsson, and E. Jonsson, "Security aspects of the in-vehicle network in the connected car," IEEE Intelligent Vehicles Symposium, pp. 528–533, 2011.

[9] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," IEEE Symposium on Security and Privacy, pp. 447–462, 2010.

[10] F. Koushanfar, A. Sadeghi, and H. Seudie, "EDA for secure and dependable cybercars: challenges and opportunities," ACM/IEEE Design Automation Conference, pp. 220–228, 2012.

[11] C.-W. Lin and A. Sangiovanni-Vincentelli, "Cyber-security for the Controller Area Network (CAN) communication protocol," ASE International Conference on Cyber Security, pp. 344–350, 2012.

[12] D. K. Nilsson, U. E. Larson, and E. Jonsson, "Efficient in-vehicle delayed data authentication based on compound message authentication codes," IEEE Vehicular Technology Conference, pp. 1–5, 2008.

[13] OSEK/VDX, OS Specification, version 2.2.3, http://www.osek-vdx.org, 2006.

[14] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar. "Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study," USENIX Conference on Security, 2010.

[15] A. Sangiovanni-Vincentelli, "Quo vadis, SLD? Reasoning about the trends and challenges of system level design," Proceedings of the IEEE, vol. 95, no.3, pp. 467–506, 2007.

[16] C. Szilagyi, "Low cost multicast network authentication for embedded control systems", Ph.D. thesis, Electrical and Computer Engineering Department, Carnegie Mellon University, 2012.

[17] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "CANAuth—a simple, backward compatible broadcast authentication protocol for CAN bus," Workshop on Embedded Security in Cars, 2011.

[18] M. Wolf, A. Weimerskirch, and C. Paar, "Security in automotive bus systems," Workshop on Embedded Security in Cars, 2004.

[19] Q. Zhu, Y. Yang, M. Di Natale, E. Scholte, and A. Sangiovanni-Vincentelli, "Optimizing the software architecture for extensibility in hard real-time distributed systems," IEEE Transactions on Industrial Informatics, vol. 6, no. 4, pp. 621–636, 2010.

[20] T. Ziermann, S. Wildermann, and J. Teich, "CAN+: a new backward-compatible Controller Area Network (CAN) protocol with up to 16x higher data rates," ACM/IEEE Design Automation and Test in Europe, pp. 1088–1093, 2009.