Synthesizing Safe Timing Bounds for Secure CAN bus in Automotive Applications

ABSTRACT

Security issues in the CAN bus have led to the design of custom authentication protocols. To ensure these protocol behave correctly, a platform independent model is developed and formally verified against desired security properties. However, this model fails to capture the timing behaviors of the authentication protocol, which can be violated when it is integrated on a specific platform. As such, we use a platform dependent, parameterized timed automata model of the protocol and synthesize safe delay bounds that will ensure timing behaviors are preserved. We demonstrate feasibility of our approach by synthesizing delay bounds for the MaCAN authentication protocol.

ACM Reference Format:

. 2018. Synthesizing Safe Timing Bounds for Secure CAN bus in Automotive Applications. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

HyperLTL when traces depend on each other, for example one authentication depends on the authentication of another.

A popular and useful approach to building verified cyber-physical systems (CPS) is to create a platform independent model (PIM) of the intended behaviour of the system. Using this model, various temporal logics can be used to specify and check some properties on that model, such as liveness, deadlock, livelock, etc. While this is an effective method for reasoning on an abstract level, when the CPS is implemented in practice, additional delays are introduced (from hardware and other physical constraints) that are not expressed in the more abstract model. This can lead to models which ensure safety properties that do not hold in practice.

To account for these delays, manufacturers of devices will provide a range of delays that can be expected. These delays can then be integrated into the model to check that saftey properties hold even in the implemented version of the model.

While manufacturer guarantees can often be taken as safe assumptions, such models are not available in many other situations. For example, when embedding platform independent software into a particular system, the timing model may change based on this hardware. Furthermore, as CPS component development becomes more accessible to individuals, there may not be a central manufacturer that can provide a bounded timing model.

for this reason, we address the inverse problem, where these delays are unknown or untrusted, and the user wishes to discover the acceptable range of delays that will still admit the safety properties of interest.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

2 RELATED WORK

As CPS are resource-constrained systems, understanding the impact of any security solutions on control performance, timing, and resources of the system is important. Furthermore, ensuring the solutions respect the semantic gap between design and implementation is crucial for its correct operation. Consequently, Lin *et al.* [4], Pasqualetti and Zhu [5] and Zheng *et al.* [6] proposed frameworks that analyses the impact of security solutions and consider the gap between controller design and implementation. Lin *et al.* [4] analysed the impact of message authentication mechanism with time-delayed release of keys on real-time constraints of a ground-vehicle. Such a security solution was developed to protect Time Division Multiple Access (TDMA)-based protocol, which is used in many safety-critical systems such as automobile and avionics electronic systems because of their more predictable timing behavior. To ensure the increased latencies (due to delayed key release) did not violate timing requirements, an algorithm to optimize task allocation, priority assignment, network scheduling, and key-release interval length during the mapping process from the functional model to the architectural model, with consideration of the overhead was developed. This algorithm combined simulated annealing with a set of efficient optimization heuristics. However, their approach did not consider the impact of their security solution on sampling periods and control performance. Furthermore, they didn't consider presence of a software platform between the security solution and hardware.

Pasqualetti and Zhu's [5] method could analyse control performance, security, and end-to-end timing of a resource-constrained CPS under network (cyber) attack that can compromise systems privacy (confidentiality). They have also quantified interdependency between the three system objectives by means of a minimal set of interface variables and relations. In their work, they have considered an adversary that has complete knowledge of the system model and can reconstruct system states from measurements. As a first step, the physical plant was modeled as a continuous time LTI system. The control input was determined using an output-based control law. A relationship was established to show that the control performance improved with reduced sampling time. Next, resiliency of the encryption method, protecting messages transmitted by sensor to controller was evaluated. It was observed that the encryption method increased the sampling period thereby degrading control performance. While implementing the control function on a CPS platform, the end-to-end delay was calculated by incorporating time incurred during sensing, computation, and communication. During development of the scheduling algorithm for the system, it was ensured that the measured delay was within the sampling period. Based on their analysis, they concluded that the control and the security algorithms should be

designed based on the implementation platform so as to optimize performance and robustness.

Zheng *et al.* [6] quantify the impact of their security solution on control performance and schedulability. They also analyzed the tradeoffs between security level and control performance while ensuring the resource and real-time constraints were satisfied. For demonstration, a CPS with multiple controllers that share computation and communication resources was considered. A controller, which was modelled as a control task, processed information collected from sensors on a shared resource and commanded actuators to carry out task. To prevent attackers from eavesdropping on the communication medium for obtaining system's internal information, messages from sensors were encrypted. The decryption of these messages were modeled as task. Each of these tasks were given an activation period and worst case execution time. In the system, the control tasks competed for computation resources whereas as messages competed for communication resources. Incorporating the message encryption mechanism introduced resource overhead that impacted schedulability and control performance. To avoid this issue, they framed an optimization problem where control performance (a function of control task period) was the objective function and security level, computation resource, communication resource, and end-to-end latency were constraints.

3 MODELING OF SECURITY PROTOCOL FOR CAN

In this section, we first describe the *Message authenticated CAN* (MaCAN) protocol [3], which is a broadcast authentication protocol specifically designed for CAN. Then, we briefly describe the *Parametric Timed Automata* framework that we use to model such a protocol.

3.1 MaCAN protocol

The MaCAN protocol was designed to increase the security of the standard CAN bus. It enables incorporation of authentication message in CAN frames, which can be verified by either a single or a group of receiving nodes (ECUs). The protocol uses a new partitioning scheme that allows addition of CAN-ID (sender's ECU ID), two flag bits (for distinguishing different message types), and destination node's ID to the traditional CAN frame. The CAN-ID contains six bit source ID (src_id) that enables the receiving node to identify the sender's node. The flag bits are mapped to the first two most significant bits of the first byte of the data field and rest six bits are used for storing destination node's ID (dst_id). The remaining seven bytes of the crypt frame data field store the message to be transmitted. The protocol uses different format of messages, details of which can be found in [3].

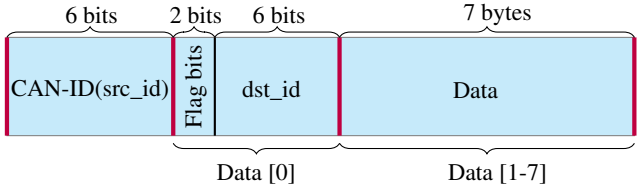


Figure 1: Crypt Frame

The MaCAN protocol requires a *key server* (KS) for distributing key among the nodes and a *time server* (TS) to synchronize internal counter of nodes with the global time. The KS maintains the session keys and stores the symmetric long-term keys (LTK) of all the nodes (ECUs) of the vehicle network. The TS periodically provide precise time to all the nodes and respond to authenticated time request. With this setup, the protocol performs three procedures: (i) session key establishment, (ii) signal authentication, and (iii) time synchronization, for secure message transmission in CAN network.

Session Key Establishment

Prior to the exchange of authenticated messages among ECUs (nodes), the MaCAN protocol requires the sender (ECU_i) and the receiver (ECU_j) nodes to obtain a session key by communicating with the *key server*. Figure. 2 illustrates the key establishment procedure. To send the session keys to the nodes, the *key server* uses the stored LTK of (ECU_i) and (ECU_j).

The procedure starts with ECU_i sending a challenge C_i and ID id_j of the node it wants to communicate with to the *KS* (3.1). The *KS* respond by sending **SK** message that contain the fresh session key SK_{ij}, the challenge C_i, and IDs of ECU_i (id_i) and ECU_j (id_j), encrypted with the LTK $K_{i,LTK}$ of ECU_i (3.2). Subsequently, *KS* sends a request for challenge (**RC**) to ECU_j, notifying it of the session key that has been generated (3.3). After ECU_i decrypts the session key SK_{ij} and verify the challenge value, it sends an acknowledgement message **ACK** to ECU_j. The **ACK** message consist of a 24-bit group field gf that indicates the nodes the sender (ECU_i) is aware of being authenticated and a signed (with SK_{ij}) cipher based message authentication code (*cmac*) that includes the current timestamp T, the group field, and ID of ECU_j (3.4). Subsequently, ECU_j follows an analogous procedure to obtain the session key from the key server and notify ECU_i (3.5, 3.6, 3.7). When ECU_i verifies the authenticity of the **ACK** message received from ECU_j and does not find its ID in the group field, it sends another **ACK** to ECU_j with both id_i and id_j bits in the group field set (3.8).

$$ECU_i \rightarrow KS : \mathbf{CH}, C_i, id_j \tag{3.1}$$
$$KS \rightarrow ECU_i : \mathbf{SK}, \{C_i, id_j, id_i, SK_{ij}\}_{K_{i,LTK}} \tag{3.2}$$
$$KS \rightarrow ECU_j : \mathbf{RC}, id_i \tag{3.3}$$
$$ECU_i \rightarrow ECU_j : \mathbf{ACK}, gf(id_i), \{cmac(T, id_j, gf(id_i))\}_{SK_{ij}} \tag{3.4}$$
$$ECU_j \rightarrow KS : \mathbf{CH}, C_j, id_i \tag{3.5}$$
$$KS \rightarrow ECU_j : \mathbf{SK}, \{C_j, id_j, id_i, SK_{ij}\}_{K_{j,LTK}} \tag{3.6}$$
$$ECU_j \rightarrow ECU_i : \mathbf{ACK}, gf(id_j), \{cmac(T, id_i, gf(id_j))\}_{SK_{ij}} \tag{3.7}$$
$$ECU_i \rightarrow ECU_j : \mathbf{ACK}, gf(id_i|id_j), \{cmac(T, id_j, gf(id_i|id_j))\}_{SK_{ij}} \tag{3.8}$$

Figure 2: MaCAN session key establishment procedure

The MaCAN protocol supports authentication of groups with more than two ECUs. The groups are formed by members with same trust level (e.g. ECUs made by the same manufacturer can form a group) and they share the same session key for message exchange.

Signal Authentication

After establishing the session keys, the ECUs communicate with each other by periodically exchanging messages with signal values (e.g. sensor data of tire pressure is a signal). To receive a signal *sig#*,

the receiving node ECU_i should first send an authenticated signal request **SIG_AUTH_REQ** message to node ECU_j (3.9). The request message is a signed *cmac* that includes the signal number *sig#* that needs to be authenticated, *presc* that specifies the signing behavior ($presc = 0$: request the next message to be authenticated, $presc = 1$: request each following message to be authenticated), the current time stamp T, and the sender and receiver node's IDs. After verifying the *cmac*, ECU_j sends the **AUTH_SIG** message that contains the signed signal value *Signal* to ECU_i (3.10).

$$ECU_i \rightarrow ECU_j : \textbf{SIG\_AUTH\_REQ}, sig\#, presc, \{cmac(T, id_i, id_j, sig\#, presc)\}_{SK_{ij}} \quad (3.9)$$

$$ECU_j \rightarrow ECU_i : \textbf{AUTH\_SIG}, sig\#, Signal, \{cmac(T, id_i, id_j, sig\#, Signal)\}_{SK_{ij}} \quad (3.10)$$

Figure 3: MaCAN signal authentication procedure

Time Synchronization

The ECUs of the MaCAN protocol uses time T as one of the inputs in the *cmac* function. As such, each ECU has an internal clock that is synchronized to the global clock of the *time server (TS)*. The global time is broadcasted periodically in an unauthenticated form by the *TS*. Every ECU receives the time signal and compare it against its internal clock value. If there is a large mismatch between the time values, then the ECU request *TS* for an authentic time value by sending a challenge message **CH** with fwd_id set to 0 and the challenge C_i (3.11). Subsequently, the *TS* responds with the authenticated time **AT** message, which is signed with the ECU's session key SK_{ij} and contains the last broadcasted timestamp value T and the challenge C_i (3.12).

$$ECU_i \rightarrow TS : \textbf{CH}, C_i, fwd\_id = 0 \quad (3.11)$$

$$TS \rightarrow ECU_i : \textbf{AT}, T, \{cmac(T, C_i)\}_{SK_{i,TS}} \quad (3.12)$$

Figure 4: MaCAN time synchronization procedure

As each communication signal (or message) of the network uses the current time value, attack such as replay can be detected and ECUs can be prevented from being affected by corrupted messages.

3.2 Modeling of Security Protocol

Next, we use the formalism of *Parametric Timed Automata* (PTA), which is an extension of the class of Timed Automata (TA) to model the protocol [2]. Unlike TA that uses *constants* in the clock constraints, the PTA allow use of *parameters* i.e. unknown constants within these constraints.

Definition 3.2.1. A *PTA* is a tuple of the form $\mathcal{M} = (\Sigma, L, l_0, X, P, I, E, K)$, where *i*) Σ is a finite set of actions, *ii*) L is a finite set of locations, *iii*) $l_0 \in L$ is the initial location, *iv*) X is a finite set of non-negative real valued $\mathbb{R}^+$ clocks, *v*) P is a finite set of $\mathbb{R}^+$ parameters, *vi*) I is the invariant that assign a guard $g(X)$ - constraint over (X, P) of the form $x \sim p$, where $\sim \in \{<, \leq, \geq, >\}$ - to every $l \in L$, *vii*) E is a set of edges $e = (l, g, a, R, l')$ where $l, l' \in L$ are the source and destination locations, $a \in \Sigma$, g is a guard, and $R \subseteq X$ is a set of clocks to be reset, and *viii*) K is a constraint over P.

Raj: [Implement the model in IMITATOR]

4 SIDE CHANNEL ATTACK

Raj: [Write down the attack model for the protocol]

5 PARAMETER BOUND SYNTHESIS

As described in Sec 1, we are interested in discovering the bounds on the delay that can be introduced to a platform independent model, such that some safety property still holds.

The *bounded integer parameter synthesis problem* from [2], is defined as follows: Given a parametric timed automaton $\mathcal{M}$, a labeling function $\mathcal{L}$, an LTL property (verification condition) ϕ, a lower bound function $lb : P \rightarrow \mathbb{Z}$ and an upper bound function $ub : P \rightarrow \mathbb{Z}$, compute the set of all parameter valuations $v : P \rightarrow \mathbb{Z}$ such that

$$lb(p) < v(p) < ub(p) \wedge (M, v, L) \models \phi$$

In the context of embedded systems, this is an appropriate problem to ask when the delays bounds for a system are given and trusted. However, if the bounds lb and ub are unknown or untrusted, we must instead synthesize these bounds, such that all valuations within those bounds satisfy the property ϕ instead.

5.1 Problem Statement

For this reason, we formulate the *parameter bound synthesis problem*, where the goal is to find upper and lower bounds such that all valuations within those bounds satisfy the verification condition. In the CPS domain, this corresponds to the question "what are the most flexible bounds on the delay my system can induce, such that the overall system is safe?" Formally, given a parametric timed automaton $\mathcal{M}$, a labeling function $\mathcal{L}$, an LTL property ϕ, find the lower bound $lb : P \rightarrow \mathbb{R}$ and upper bound $ub : P \rightarrow \mathbb{R}$ functions such that

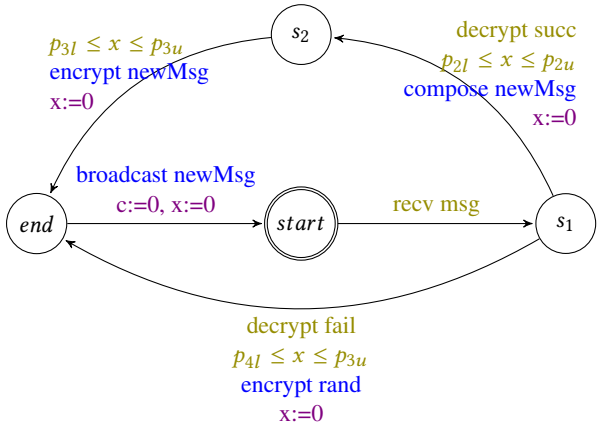
$$\forall v.\ lb(p) < v(p) < ub(p). \quad (M, v, L) \models \phi$$

Since the number of parameters, n, is always finite, we can restate the problem, for the purposes of clarity, as follows

$$\exists\, lb_0,\ \ldots,\ lb_n,\ ub_0,\ \ldots,\ ub_n \in \mathbb{R}.$$
$$\forall p_0 \in [lb_0, ub_0].$$
$$\ldots$$
$$\forall p_n \in [lb_n, ub_n].$$
$$(M, v, L) \models \phi$$

5.2 Toy Example

A simplified version of the *private authentication protocol* from [**?**] as a parameterized automata. Colors indicate; I, the guards, Σ, the set of action, and clock sets.



In this example, we want to find the delays such that the system is safe from side channel timing attacks. In particular, given an attacker can distinguish the trace of an execution by observing a difference, k, in the clocks of those traces, what is the set of allowable values for p_i to ensure that all paths of two traces, π_1 and π_2 are observationally equivalent.

$$\phi = \forall \pi_1 . \forall \pi_2 . G(end_{\pi_1} \wedge end_{\pi_2}) \implies G(|c_{\pi_1} - c_{\pi_2}| \leq k)$$

5.3 Optimizations on bounds

The problem, as stated will find some arbitrary satisfying bounds. In practice,, we may want to consider a sense an optimal solution. **Mark:** [TODO : this doesnt make any sense, rewrite] For example, if the implementation of our system induces a large range for every parameter, we may seek to maximize the minimum range of the bounds. This gives us the most relaxed condition on the bounds so that, if our implementation can meet those bounds, the system is still safe.

$$\mathcal{OP}(lb, ub) = \min(ub(p) - lb(p))$$

It may be the case that we find that the bounds are too tight for the system implementation to guarantee. In this case, we may seek another optimization condition, that satisfies the verification condition and is possible for the implementation to guarantee. For example, our implementation may be able to guarantee that sum of the upper bounds will be less than some constant.

$$\mathcal{OP}(lb, ub, t) = \Sigma_p \; ub(p) < c$$

In a sense, the optimization function $\mathcal{OP}$ is an analog to finding the weakest precondition. The larger the value of the $\mathcal{OP}$, the more system implementations will satisfy the verification condition. We enforce the restriction here that it is always a maximization problem, and the $\mathcal{OP}$ is always linear (TODO is this necessary?).

5.4 Decidability

This problem can be framed as a decision problem in two different ways. First, as a general optimization, "Does there exist a global maximum value for which all valuations within the bounds satisfy ϕ?". This problem is undecidable, as we can reduce this to the unbounded integer parameter synthesis problem, which is known to be undecidable [1]. TODO proof - actually im not sure about this, even over integers.

Second, we can frame the problem as "Does there exist a global maximum value less than k for which all valuations within the bounds satisfy ϕ?" This seems decidable for integers, as we could enumerate every lb and ub, and check each valuation within those bounds. Th complexity of LTL model checking is exponential in the size of the spec. There are only finitely many options for lb and ub, since lb is finitely bounded by $\forall p.0 \leq lb(p) \leq t(p)$.

REFERENCES

[1] Nikola Beneš, Peter Bezděk, Kim G Larsen, and Jiří Srba. 2015. Language emptiness of continuous-time parametric timed automata. In *International Colloquium on Automata, Languages, and Programming*. Springer, 69–81.

[2] Peter Bezdek, Nikola Benes, Vojtech Havel, Jiri Barnat, and Ivana Cerná. 2014. LTL Model Checking of Parametric Timed Automata. *CoRR* abs/1409.3696 (2014). http://arxiv.org/abs/1409.3696

[3] Ondřej Kulatỳ. 2015. Message authentication for CAN bus and AUTOSAR software architecture. *Master's Thesis, Czech Technical University in Prague* (2015).

[4] C. W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli. 2014. Security-aware mapping for TDMA-based real-time distributed systems. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 24–31. https://doi.org/10.1109/ICCAD.2014.7001325

[5] Fabio Pasqualetti and Qi Zhu. 2015. Design and operation of secure cyber-physical systems. *IEEE Embedded Systems Letters* 7, 1 (2015), 3–6.

[6] Bowen Zheng, Peng Deng, Rajasekhar Anguluri, Qi Zhu, and Fabio Pasqualetti. 2016. Cross-layer codesign for secure cyber-physical systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 5 (2016), 699–711.