Proceedings

2nd Workshop on the Security and Dependability of Critical Embedded Real-Time Systems (CERTS) 2017

co-located with the IEEE Real-Time Systems Symposium (RTSS)

Paris, France

Dec 5, 2017

Table of Contents

Message From the Chairs

This is the second iteration of the CERTS 2017 workshop, held in conjunction with IEEE RTSS. This edition takes place in Paris, France. The technical program includes five peer-reviewed papers and a keynote address by Matthias Schunter from Intel labs, Germany.

The aim of this workshop is to bring researcher and practitioners from a variety of domains, viz., real-time and embedded systems, security, dependability and cyber-physical systems to name just a few. The idea is to foster a community that looks at all of these topics and develops techniques, algorithms, policies and frameworks to improve the security and dependability of critical systems.

CERTS owes its success to a variety of people. We would like to thank the steering committee that consists of: Paulo Esteves-Verissimo, Marcus Volp, Antonio Casimiro and Rodolfo Pellizzoni. We would also like to thank the technical program committee members for taking the time to review and provide feedback for the papers. In addition, we would also like to thank Frank Mueller, general chair RTSS, Isabelle Puaut, program chair for RTSS 2017 and Song Han, workshops chair for RTSS 2017. Finally, the workshop cannot be a success without the authors who submitted their papers and the attendees of the workshop.

We hope that you will enjoy the CERTS 2017 workshop and it will foster work in many new directions.

Marisol Garcia Valls
Universidad Carlos III de Madrid
Co-chair

Sibin Mohan
University of Illinois at Urbana-Champaign
Co-chair

Workshop Organizers and Program Committee

Program Chairs

Sibin Mohan, *University of Illinois at Urbana-Champaign*
Marisol García Valls, *Universidad Carlos III Madrid*

Steering Committee

Marcus Völp, *SnT – University of Luxembourg*
Paulo Esteves-Verissimo, *SnT – University of Luxembourg*
Antonio Casimiro, *University of Lisboa*
Rodolfo Pellizzoni, *University of Waterloo*

Program Committee

Lui Sha, *University of Illinois at Urbana-Champaign*
Christian Esposito, *University of Naples Federico II*
Hans Reiser, *Universität Passau*
Danny Dolev, *The Hebrew University of Jerusalem*
Antônio Augusto Fröhlich, *Federal University of Santa Catarina*
Zbigniew Kalbarczyk, *University of Illinois at Urbana-Champaign*
Miroslav Pajic, *Duke University*
Ravi Prakash, *University of Dallas*
Sasikumar Punnekat, *Maelardalen University*
Guillermo Rodriguez-Navas, *Maelardalen University*
José Rufino, *Faculdade de Ciencias da Universidade de Lisboa*
Elad Schiller, *Chalmers University of Technology*
Bryan Ward, *MIT Lincoln Laboratory*
Heechul Yun, *Kansas University*
Saman Zonouz, *Rutgers University*

Technical Papers

Facing the Safety-Security Gap in RTES: the Challenge of Timeliness

Marcus Völp, David Kozhaya and Paulo Esteves-Verissimo
Critical and Extreme Security and Dependability Group (CritiX)
Interdisciplinary Center for Security, Reliability and Trust (SnT)
University of Luxembourg
L-2721 Luxembourg
Email: <name>.<surname>@uni.lu

***Abstract*—Safety-critical real-time systems, including real-time cyber-physical and industrial control systems, need not be solely correct but also timely. Untimely (stale) results may have severe consequences that could render the control system's behaviour hazardous to the physical world. To ensure predictability and timeliness, developers follow a rigorous process, which essentially ensures real-time properties a priori, in all but the most unlikely combinations of circumstances. However, we have seen the complexity of both real-time applications, and the environments they run on, increase. If this is matched with the also increasing sophistication of attacks mounted to RTES systems, the case for ensuring both safety and security through aprioristic predictability loses traction, and presents an opportunity, which we take in this paper, for discussing current practices of critical real-time system design. To this end, with a slant on low-level task scheduling, we first investigate the challenges and opportunities for anticipating successful attacks on real-time systems. Then, we propose ways for adapting traditional fault- and intrusion-tolerant mechanisms to tolerate such hazards. We found that tasks which typically execute as analyzed under accidental faults, may exhibit fundamentally different behavior when compromised by malicious attacks, even with interference enforcement in place.**

I. Introduction

In the past, real-time systems were closed and, despite telemetry, largely disconnected. They executed simple, controller-like tasks that read sensor inputs, feed them into a model of the controlled plant, and produce appropriate actuator signals for maintaining safe and energy-efficient operation. Simplicity brought predictability and hence safety in all but the most rare circumstances. Safety violations, including late control decisions, are only tolerated if it can be shown that either their likelihood of occurrence is sufficiently low to practically never occur over the lifetime of the fleet or if their consequences are marginal. In particular, safety assurance criteria demand replication only if the above properties could not be achieved with singular systems, but not as precaution against attacks.

Unfortunately, the coverage of the assumed level of safety behind the above classical, accidental fault-prevention driven development process, degrades when one assumes malicious (and thus intentional) faults. Firstly, because these faults are no longer stochastic, and in consequence, a hazard is much less a residual probability (of some defect being activated), and much more a likelihood, once a defect known, and accessible to an attacker. Secondly, because both these two latter conditions have "improved" over the later years: vulnerability diagnostic tools have improved, and highly-skilled adversaries target these systems, as parts of critical information infrastructures.

It can reasonably be argued that such attacks were infeasible in the past, e.g., due to a simplicity-enforced lack of exploitable vulnerabilities, and/or to limited connectivity of real-time systems. However, the same cannot be said about current critical application scenarios, autonomous or cooperative driving, for instance, being a blatant example. In fact, we have recently elaborated on the threat surface of the cooperative-driving ecosystem [1], revealing what we call the safety-security gap:

Vulnerabilities rarely triggered through combinations of natural events may well cause serious harm when exploited by adversarial teams.

Having said this, two important factors single out critical real-time applications from general IT ones. First, both the attack and the necessary defense are dictated by the environment dynamics, making the slow and imprecise human-in-the-loop approach to current IT security, infeasible, or ineffective at best. Second, Cyber-physical Systems (CPS) may cause severe impact upon failure, both to humans or to resources. Rather than discharging threats with "unlikelihood" arguments, we believe it is time to meet them with paradigms that can come to exhibit a power and an effectiveness commensurate to the adversarial power we begin to witness.

It seems intuitive that the decreased coverage of the level of safety, which we have discussed earlier, could be regained, if systems, albeit in the presence of defects and other faults that may now be explored by attacks — with considerable reachability and likelihood of activation — could still achieve a similar level of failure avoidance as in the past, through *automatic means*. Fault tolerance as a general predicate seems to have been performing up to the task, in the scope of accidental faults. Now, we would need both fault and intrusion tolerance.

Fortunately, the fault and intrusion tolerance body of knowledge (commonly called BFT, for 'Byzantine Fault Tolerance')

This work is partially supported by the Fonds National de la Recherche Luxembourg (FNR) through PEARL grant FNR/P14/8149128.

has had a dramatic development, and already gives us a few preliminary solutions and insights to mitigate these threats in an automatic way, at least if the system at hand is not real-time. Replication and voting mask the actions of a minority of compromised replicas behind a majority of healthy replicas, reaching consensus [2]. Reactive rejuvenation of known or suspected compromised replicas and occasional proactive rejuvenation counteract exhausting the set of healthy replicas and defy stealthy adversaries and detection flaws [3]. Rejuvenation is of particular importance if adversaries persist in their attack with the goal to eventually exceed the tolerance threshold of up to f compromised replicas. Last but not least, diversification ensures that adversaries cannot benefit from knowledge gained during previous attack runs [4].

However, most of this research is concerned with asynchronous or partially synchronous systems, and further research is required on the extension of the paradigm to encompass real-time behavior. Namely, because the impact of the behavior of compromised components on system timeliness, is not well understood. We give a contribution in this position paper, at the specific level of task and component scheduling. We argue that intrusion tolerance mechanisms, despite their proven guarantees for facing attacks, lose their effectiveness if applied without a good understanding of the interaction between system tasks and components. To this end, we first revisit the traditional real-time system development process to highlight additional complications that arise when a subset of tasks may have been compromised.

In Section IV , this paper sketches our vision of a real-time BFT architecture featuring replication of "critical" components or sub-systems as the key to face faults and compromises through intrusions. More importantly, it shows in Section III, that traditional simple replication mechanisms may fall short of achieving their mission, in real-time systems. The reason lies in unanticipated interference: due to lower level system operations, tasks have access to different parts of common resources, which in case of malicious behaviour allows attackers to sabotage the whole resource. As such, replication, under contemporary interference analysis, may not yield the desired fault-/intrusion-tolerance. This paper concretely investigates such interferences, focusing on memory and cache interferences in multi-core systems, and highlights pitfalls of intrusion-agnostic analysis of task behavior deviation. We identify the challenges when facing the timeliness threats of compromised tasks, and sketch intrusion tolerance solutions for partially interfering-controllable resources. To our surprise, anticipating replication not only for fault but also for intrusion tolerance, though it complicates system analysis, it also bears opportunities to actually simplify the resulting scheduling problem, possibly even leading to more optimistic response times.

II. The Safety-Security Gap of the Cooperative-Vehicle Ecosystems

Lima et al. [1] identified threats to the cooperative-vehicle ecosystem, concluding that autonomous driving without cooperation is doomed to fail in the interim phase where both fully autonomous and human-controlled vehicles share the road.

For one, while driving, humans base their decisions on a variety of implicit protocols, interpreting driving styles, eye contacts, subtle movements and similar indicators as signs to evade an opposing car or to break aggressively (e.g., because the driver observed a scared look in a mother's face when chasing her child to prevent him from running onto the street). Cooperation can partially close this communication gap through explicit communication, until research incorporates these implicit protocols.

The second aspect, despite the need for cooperation, lies in the increased threat surface of autonomous and to a larger extent also cooperative vehicle ecosystems. Vehicles must defend against attacks on global V2I, I2I communication infrastructures [5], [6], against V2V attacks, but also against the classical in-vehicle communication networks such as CAN [7], [8] and Flexray [9]. Already today, diagnostics and infotainment access expose these safety-critical networks, with an often non-redundant gateway being the last line of defense against remote attacks.

Vulnerabilities in the software of this gateway, but also in other components such as the complex scenery detection tasks required for autonomous driving, put safe operation at risk. Similarly vulnerable, but more exposed are road-side units and cloud-based authentication mechanisms which are required in cooperative scenarios to distinguish authentic from fake events.

In addition to the above cyber attacks, autonomous cars (but likewise CPS and IoT systems) are also exposed to attacks against their plant and environment sensing capabilities [10], [11], a matter which although accidental already took their first life toll [12]. Sensor fusion and cross-validation amongst vehicles may be one solution to mitigate this threat. However, mitigation strategies of this kind heavily rely on reliable V2V and V2I communication, which is easily blocked through jamming in current substrates if cyber attacks are accompanied by physical attacks. In fact, Serageldin et al. [13] show that jamming becomes over-proportionally effective at higher DSRC bandwidths, leaving only low bandwidth solutions tolerant to such attacks.

Clearly, analyses fall short of correctly valuing safety threats if they anticipate only accidental faults and their likelihood, but not coordinated attacks to cause these faults. In particular, natural occurrences of combinations of independent faults are extremely rare and as such often overlooked or misinterpreted in terms of risk. However, adversaries in control of the system may easily trigger such combinations and thereby exploit the safety-security gap in security-agnostic analyses, a conclusion which is also shared by Hamad et al. [14] in their attack-tree based security analysis of automated obstacle avoidance.

III. Timeliness Threats

Clearly, the fundamental prerequisites for meeting the above challenges include (1) limiting the interference that compromised tasks can have on other tasks, and (2) developing mechanisms for enforcing these limits in a trustworthy manner.

Otherwise, any compromised task would be able to exceed the bounds to jeopardize the timeliness of its critical counterparts.

A. Memory Isolation and Cooperative-Scheduling

An intuitive consequence of the need to limit interference suggests isolation as a key factor for reducing the attack surfaces inside real-time systems. This implies reducing the ability of an adversary to compromise further components once it has successfully compromised one.

Clearly, in systems without sufficient memory protection, adversarial control may spread from one task to others until critical system components become compromised.

Common practice of embedded real-time systems today is to execute code directly from flash images, so one might argue against code-level compromises. However, examples like return-oriented programming [15] and similar techniques have demonstrated how programs can be compromised without altering their code. Also, over-the-air update capabilities demand for mechanisms to replace pre-installed code. Not to mention that higher-level tasks of autonomous driving, such as scenery detection and trajectory planning, exceed today's on-die flash capacity and require instruction caches or scratch-pad memory (i.e., modifiable storage) to keep up with their performance requirements.

Notice that it is purposeful to enforce strong isolation between tasks, even if one task produces a result, which is an essential input to the other. Strongly isolating these dependent tasks slows down adversaries, who are forced in that case to attack by either breaking the isolation or through the communication interfaces between tasks. Moreover, when we later introduce replication, dependent tasks in a chain may be replicated separately rather than the whole chain. This way, each dependent segment benefits from a majority of healthy replicas in the previous segment.

Now, for the same reasons that we have to disqualify real-time systems with lack of strong isolation, we must also disqualify cooperative scheduling and schedulers without time-slice enforcement. In these systems, tasks are expected to voluntarily relinquish control over allocated resources. However, compromised tasks, deviating from their analyzed behavior, may never relinquish such resources, thereby falsifying analyzed resource bounds.

Fortunately, memory isolation and enforced schedules are already state-of-the-art in many (though evidently, until recently, not all [16]) automotive systems. In particular the lower control levels run on physically isolated microcontrollers or on well isolating RTOSs. However, the same care must be exercised for the complex autonomous driving counterparts, in particular due to the imminent threat of legacy OS compromises.

B. Resource Bound Analysis

So far, car manufacturers refrained from using modern, superscalar, multicore processors with their innumerous latency hiding mechanisms. However, the performance demands of higher autonomy levels [18] and the expected data rates that need to be processed for cooperative driving may change the picture. Let us therefore investigate more closely what impact a deviation of a compromised task from its analyzed behavior can have in modern multicore architectures that are not specifically equipped with QoS mechanisms to ensure minimal guarantees for critical tasks (e.g., by employing AMBA 5's bandwidth-control mechanisms [19]).

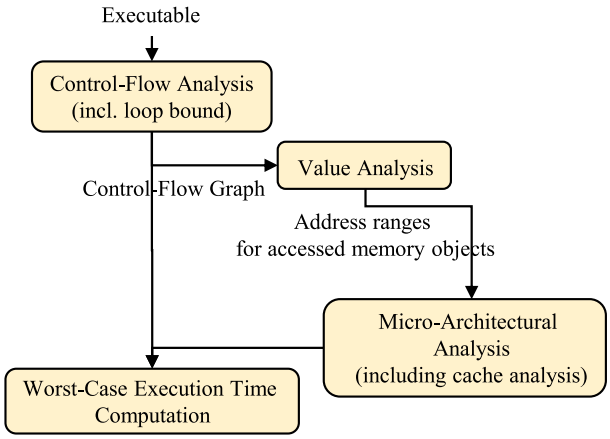


Fig. 1. Path and cache analysis for WCET estimation (adopted from [17] Fig.2).

Traditionally, resource bound analysis is concerned with finding the worst-case combination of task execution paths that lead to the worst-case overall execution time and hence the maximal interference tasks may have on each other. It is therefore tempting to subject task groups to such an analysis and run them with the obtained interference bounds.

When interference bounds can be controlled completely, following this approach preserves safety even under attack. However, not all resources, through which tasks may interfere with others, can be controlled to the required degree. For these resources, WCET analysis must not only compute the worst-case combination of execution paths of analyzed tasks, but also combinations where a subset of tasks execute in the worst possible pattern.

Compromised tasks may exhibit arbitrary execution patterns over the resources they can get hold of. That is, given a statically allocated set of resources R_i, a compromised task τ_i may access the resources in R_i in a pattern that maximizes interference on other tasks. In systems with dynamic resource allocation, as they will be required for more demanding applications, compromised tasks τ_i may request further resources, expanding R_i to the set of acquirable resources R_i^{max}, and then construct a worst-case interference pattern. In this situation, resource bound analysis has to anticipate arbitrary execution over the extended set R_i^{max}.

1) Caches: Let us exemplify the consequences of this observation with the example of resource bound analyses for processor data caches [17].

Caches are near core memories split into multiple pairs of tag and data RAM (called ways) and equipped with a logic to transparently resolve cache hitting and missing memory

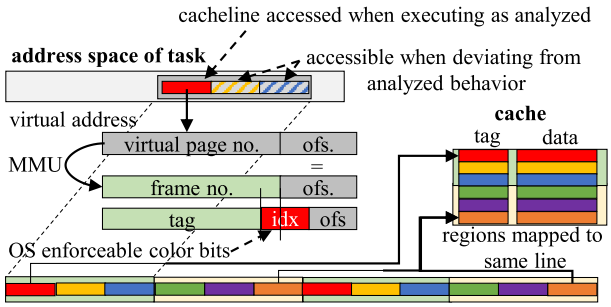


Fig. 2. OS controllable cache colors. Although only cachline-size regions of the same color collide in the cache, OS interference control is limited to those index bits that range into the frame number. Interference by compromized applications can therefore be much higher than when behaving as analyzed.

accesses. Extracting from the accessed address the lower most bits after the cacheline offset (i.e., the *index*), the cache logic determines the row in all ways and compares the tag RAM against the remaining higher order bits (the *tag*) to determine hits (exactly one match) or misses (no match). Since replacement is only among cachelines of the same row, it is possible to color memory locations by the rows in which they will be inserted in the cache. The bottom part of Figure 2 illustrates this coloring and the split of the physical address into tag, index and offset. Given cl large cachelines and an associativity (no. of ways) a, a cache of size $s = n \cdot a \cdot cl$ has n colors.

Processors come with multiple levels of caches, some separately caching code and data, others both at the same time. Lower levels (e.g. L1) are typically private (i.e., exclusively used by a single core) while later levels (e.g. L2) act as victim caches to keep evicted cachelines from L1 or as possibly inclusive shared caches for all cores on the same die (e.g., L3). Inclusive means data cached in lower level caches is also cached in L3 and if evicted in the latter, must also leave the lower levels. Write-through caches update lower cache levels and RAM immediately, write-back defers these updates and, in case of L2 victim caches also the cacheline allocation, to the time of eviction.

Figure 1 shows the building blocks of a worst-case execution time (WCET) pipeline. Starting from the executable binary, the control flow graph and loop bounds are extracted, which are then fed into a further value analysis for determining address ranges for all accessed variables and other memory objects. With these ranges, a micro-architectural analysis is invoked, which includes a cache analysis. The cache analysis itself proceeds by abstractly tracking the locations that may and must hit in the cache along the replacement policy (e.g., sorted by age in case of least-recently-used (LRU) replacement) and by merging the abstract states at control-flow join points.

Figure 3 shows this merging for a 4-way set associative LRU cache after executing the following code from empty caches:

```
d = 0;
if (a > 0) { b = 1; c = 2; }
else { d = 1; e = 2; }
```

Assuming all variables are in cachelines of the same color, d and a must hit in the cache with age 4 and 3, respectively (maximum age to conservatively bound cache hits), while c, e may hit with age 1, d, b with age 2 and a with age 3 (minimum age to conservatively bound cache misses, potential write backs, and cross core evictions due to L3 cache inclusiveness).

2) Cache analysis under attack: From the above observation, it is tempting to extract the interference pattern of a task from the addresses it accesses and to compute from this cache related preemption delays (see e.g., [20]) and similar interference bounds. Assume for example two tasks τ_1 and τ_2 whereby the scheduler executes both tasks on the same core while allowing τ_2 to preempt each job of τ_1 once. If τ_2 accesses at most two cachelines of a particular color, the worst case interference that may happen to τ_1's memory accesses of this color are two evictions plus the write-back of two possibly dirty lines (see right part of Fig. 3).

The same scenario when executing τ_2 on a different core of the same die and with an inclusive last-level cache effectively reduces to two the ways available to τ_1 for this color. This is because τ_2 may repeatedly access the two cachelines to evict the memory cached by τ_1. Of course, further analysis of τ_2's access pattern allows exploiting more fine grain interleavings, e.g., allowing τ_1 a number of subsequent accesses in between any two of τ_2's accesses.

Unfortunately, the operating system (OS) can enforce cache colors only at the granularity of the smallest page size [21] when allocating page frames for an application and only at the cost of having to support paging, a feature necessarily required by more resource demanding applications, but, due to predictability concerns, rarely supported in real-time operating systems (RTOSs) [22]. This lack of control stems from paging, which allows the OS to define only those index bits that are part of the page number (i.e., above the page offset). Assume $ps = k \cdot cl$ holds for the size of the smallest page ps. Then, because k cachelines fit this page, the number of enforceable colors is reduced to n/k (e.g., yellow and green in Fig. 2).

Rephrasing the above statement slightly differently, a task τ_i analyzed to access memory in the sequence of colors S_i^c may exhibit arbitrary sequences S_i' of colors c when compromised, where c agrees with a color in S_i^c in the index bits above the page offset. In particular, compromised τ_i may access memory that it did not access when executing as analyzed.

Although not yet quantified in adversarial settings, the comprehensive benchmarks in [23]–[25] give a first indication on the impact that compromised tasks can have when executing outside analyzed behavior, in particular when interfering through shared implicit last-level caches. In addition, these works suggest hardware and software-level solutions to partially mitigate cross-core interference though shared caches. For example, Kim et al. [24] introduces hardware way- and set-partitioning mechanisms in MC^2 for last-level caches (LLCs), Kenna et al. [23] discuss page coloring for LLCs and Mancuso et al. [25] introduce colored lockdown.

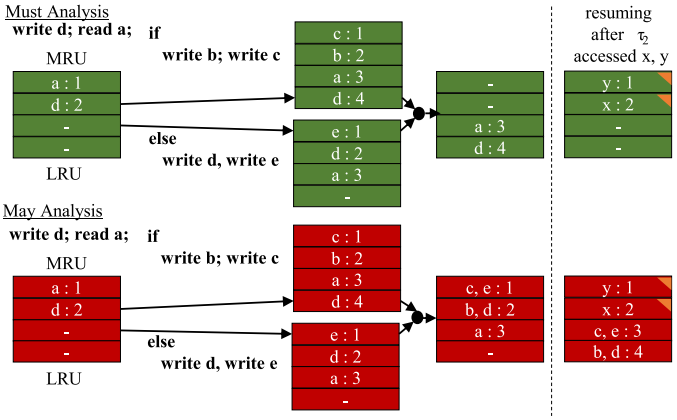


Fig. 3. Merging of abstract cache states at the join point after the conditional branch.

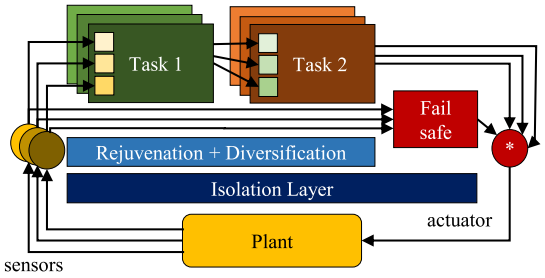


Fig. 4. Intrusion Tolerant Architecture for Complex Safety-Critical Real-Time Systems

Krüger et al. [26] work on counteracting these threats through schedule randomization by probabilistically increasing the distance between attacking and attacked tasks.

3) Memory, Busses and Pipelines: Coloring not only applies to caches, but extends also down to memory banks for limiting DRAM refresh interferrence. However, like caches, color-based control over DRAM banks only overapproximates the banks that compromised tasks may access, which lets us expect a similar discrepancy between analyzed and compromised behavior.

Yun et al. [27] introduce a performance-counter based framework to enforce memory bus bandwidths. However, again, counters only have throttling capabilities, hence they may not change access patterns at finer granularity. In particular buses like the CPU / GPU interconnects, which allow bursts, may therefore exhibit large discrepancies between analyzed and compromised behavior.

Last but not least, most of the above arguments implicitly assumed a timing-anomaly free pipeline to enable the above analyses in the first place. Modern out-of-order, speculative and superscalar pipelines, as required for more demanding tasks such as scenery analysis, defy to a large extent predictability and can, through careful exploits, be turned into time consuming monsters.

IV. Towards an Intrusion Tolerant Architecture

Observing the possibly devastating effect compromised tasks can have on timeliness, leave alone correctness, we sketch in this section possible architectures to tolerate intrusions. Our focus is thereby on correctness matters, leaving timeliness opportunities from replication for the next section.

Figure 4 shows a birds eye view on our envisaged fault and intrusion tolerant real-time architecture. Exemplified are two complex tasks τ_1 and τ_2 controlling the plant, which may be the controlled physical system or another system of the same structure with lower-level control tasks. For example, for drones [28], a common architectural pattern is to couple the rotors and elevators with a flight stabilizing controller which in turn receives signals from a more powerful, decoupled system running more complex tasks such as flight planning and autonomous landing.

A. Isolation

In Section III-A, we have already seen the essential need for memory isolation to slow down adversaries. Candidates for this isolation layer are real-time microkernels [29]–[31] but but also hardware solutions are imaginable where replicas run on dedicated soft cores or on hard cores in ASICs. The remaining ingredients, which we discuss in the following, are voting, sensor fusion, fall-back to simplex actuator control and rejuvenation and diversification of replicas.

B. Replication and Voting

Tolerating complexity-induced vulnerabilities demands replicating τ_1 and τ_2 such that up to f of their replicas can be compromised. The remaining replicas should continue to reach consensus on the values that replicas read from the replicated sensors, on the inputs they receive from τ_1 and on the outputs τ_2 forwards to the plant actuators. Unfortunately, classical BFT consensus protocols, such as PBFT [2] or derived hybrid protocols [32], [33], operate on an asynchronous (i.e., time agnostic) system model. Applying them in a synchronous system setting is not trivial, despite bounded message transmission times and bounded execution times, as simply summing up the bounds through all protocol steps easily leads to intolerable worst case execution times.

For example, PBFT, MinBFT and CheapBFT achieve consensus by a leader proposing the next client request to vote on. However, in the presence of a faulty leader, this causes downtimes until the remaining replicas agree on a new leader. Leaderless protocols avoid this complication, however, they generally require more replicas. For example, PBFT requires $n = 3f + 1$ replicas to tolerate up to f faults. Hybrid protocols reduce this number to $2f+1$ respectively $f+1$ active plus f passive replicas. Leaderless protocols require $5f + 1$ replicas (or introduce further complexities [34] to maintain $n = 3f + 1$). An exception to this rule is BFT-TO by Correia et al. [35]. building on top of a trusted ordering wormhole, leaderless BFT-TO requires only $2f + 1$ replicas. If leader-based protocols are used in real-time systems, leader change must be bounded and anticipated in the schedule.

Fortunately, most real-time tasks are triggered by the physical system triggering events such as alarms or timers. Reliable invocation of replicas may therefore avoid voting in different orders. However still, faulty replicas (including sensors) may lie about their values and in particular they can lie differently to different client replicas. These inconsistent faults (also called 'equivocation') are prototypical of the Byzantine fault phenomenon, and are avoided by either extended number of players and rounds of communication, or through cryptographic means such as signing messages for authenticity and unforgeability. Hybrid protocols make use of trustworthy sequencers (e.g., monotonic counters) which apply cryptographic means to ensure that only a single vote can be given for a single instance. However, naturally, cryptograpy means come at non-negligible costs, which are not tolerable in low latency real-time systems.

To avoid these costs, we instead propose to exploit the tight coupling between components and to introduce hybrid components that capture sensor and task values in a manner that prevents overwriting during the same instance. For sensors, capture/compare units suggest themselves as they also capture the timing of the sampled event, thereby preventing mixing of too time distant reads, provided of course the timing source is trustworthy. OS controlled FIFO buffers and synchronous IPC [30] achieve the same for intra task communication.

C. Sensor Fusion

Marzullo [36] shows that $3f+1$ interval-type sensor values are required to agree on an interval that contains the true value, Schmid and Schossmaier [37] refine this result to Lipschitz continuous intervals (i.e., small changes in the proposed intervals lead only to small changes in the agreed upon result), and Rushby proved correct both results in the theorem prover PVS [38]. The continuous nature of the controlled plant allows intra-task communication of largely diverse replicas to operate in a similar manner, provided tasks can project the intermediate results onto the control points of the other. The consequences of course are that voting is no longer on identical values but on semantically equivalent, yet possibly different control commands.

D. Simplex and Actuators

Even though replication and voting reduce dependability from a single instance to a majority of assumed healthy replicas operating in consensus, a residual risk of common mode failures remains, in particular if replicas exceed a certain size and complexity. Common mode failures are caused by systematic vulnerabilities in all replicas. They allow adversaries to exceed the tolerance threshold f, which BFT protocols require to maintain safe operation. It is therefore also crucial to explore simplistic fail-safes that reinstantiate safety in the rare situations when consensus-reaching values are wrong.

Bak et al. [39] and Veríssimo et al. [40] propose hybrid architectures wherein a simple controller monitors and returns the system to a stable state if more complex controllers fail to provide correct and timely results.

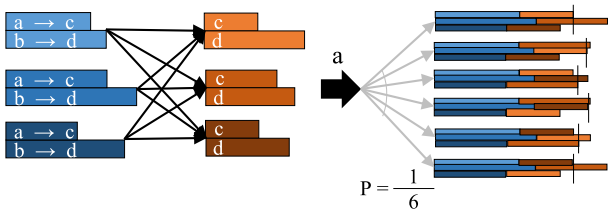


Fig. 5. Stochastic independent scheduling of dependent replicated tasks. Shown are two tasks (blue - left) and (orange - right) with clearly dependent execution times: an input of b leads to longer execution times in all replicas. However, they complete for input a in one of the patterns shown on the right with probability 1/6. In case all replicas respond correctly, the vertical bars indicate majority completion time.

E. Rejuvenation and Diversification

Diversification and rejuvenation are essential ingredients for maintaining a majority of healthy replicas and for reducing the threat of common mode failures. Diversification prevents adversaries from accumulating knowledge how to attack the system, as long as all replicas are rejuvenated periodically and faster than adversaries can compromise more than f replicas [3]. Rejuvenated replicas are down and must be compensated by additional replicas. The combination of the above requirements means diversification must be automatic (e.g., through obfuscating compilation [41]) and the real-time system needs access to a continuous stream of such diverse replicas. That is, they must either receive a continuous flow of updates from infrastructure components or must perform the compliation online in the same system. The flow/compilation task is thereby weakly real-time with an attacker speed determined periodicy, which means n new updates must be produced within this period or the fail safe must kick in (and possibly disrupt system continuity).

F. Actuators

Las but not least, when it comes to the actuation of the physical plant, agreement must be reached which control signal is applied to the plant. In the absence of consensus, this means actuators must fall back to the fail-safe while discarding the diverging replica decisions. Replication agnostic actuators must be driven by a trusted-trustworthy component condensing the multitude of proposed values.

V. Timeliness Challenges and Opportunities

The correctness challenges of the previous section also imply timeliness challenges. For example, all local downtimes and recovery mechanisms must be bounded and scheduled to guarantee (weak) timeliness of all tasks. In particular, WCET bound preserving or at least WCET bound providing compilation of diverse replicas is a challenging task that has yet to be solved. Many other questions remain open, as pointed by our reviewers: How to prove classical replication techniques to not cause interference themselves? How can sensor fusion assist in detecting interference in modern multicore architectures? How can that interference be incorporated in the system so that it can be tolerated? And how can rejuvenation and stochastic

scheduling be deployed safely without becoming a major source of interference? However, replication also brings some opportunities to simplify the WCET analysis and scheduling problem.

For example, in a multicore system, the same non-replicated schedule can be reused when critical components are replicated. Rather than allocating replicas to fixed cores (e.g., replica r_i to core i) it may be more beneficial to randomize this allocation in order to exploit the stochastic independence of replicas of the same task, even if tasks are dependent. Consider the example in Figure 5 with two replicated tasks τ_1 and τ_2, which are I/O dependent as illustrated in Figure 4. Even though the output of τ_1 influences the execution performed in τ_2, different replicas have different execution times for producing this output (in the replicas τ_1^i of τ_1) and for consuming this output (in the replicas τ_2^j of τ_2). Randomizing the allocation of τ_2^j relative to τ_1^i leads to a distribution of combined execution times for this input/output pair of which, as it is common for replicated settings, only a correct majority of $2f + 1$ (respectively $f + 1$) replicas must reach agreement before the task's deadline D_i. The other replicas need only to complete by T_i (i.e., $T_i - D_i$ time units later for deadline constrained tasks) or not at all if the task itself is stateless.

VI. Acknowledgments

The authors like to thank the anonymous reviewers for their insightful comments and suggestions to improve the paper, raising in part additional challenges, which we took the freedom to include in the above description.

VII. Conclusions

Revisiting the threat surface of safety-critical real-time systems, we have identified several challenges but also opportunities, for applying fault and intrusion tolerance techniques, which not only mitigate accidental faults but also protect the system against targeted attacks, and do both in an automatic, unattended way. We analyzed the situation at the specific level of task and component scheduling, and studied problems arising when malicious fault and intrusion tolerance must take a timeliness and scheduling perspective into account. Major challenges and hence directions for future work include limited interference controls when tasks deviate from analyzed behavior, definition of safe fail stops to compensate common mode failures in complex replicated tasks, and analysis aware obfuscating compilation.

References

[1] A. Lima, F. Rocha, M. Völp, and P. Esteves-Verissimo, "Towards safe and secure autonomous and cooperative vehicle ecosystems," in *CPS-SPC*, ACM, Vienna, Austria, Oct. 2016.

[2] M. Castro and B. Liskov, "Practical byzantine fault tolerance," 1999, pp. 173–186.

[3] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Verissimo, "Highly available intrusion-tolerant services with proactive-reactive recovery," *IEEE Trans. on Parallel & Distributed Systems*, pp. 452–465, 2009.

[4] P. Larsen, A. Homescu, S. Brunthaler, and M. Franz, "Sok: Automated software diversity," in *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, ser. SP '14, Washington, DC, USA: IEEE Computer Society, 2014, pp. 276–291.

[5] J. R. Douceur, "The Sybil attack," in *Peer-to-peer Systems*, Springer, 2002, pp. 251–260.

[6] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," *IEEE Jour. on Selected Areas in Communications*, vol. 24, no. 2, pp. 370–380, 2006.

[7] K. Pazul, "Controller area network (CAN) basics," *Microchip Techn. Inc*, 1999.

[8] J. Staggs, "How to hack your mini cooper: Reverse engineering CAN messages on passenger automobiles," *Institute for Information Security*, 2013.

[9] F. Consortium *et al.*, "Flexray communications system-protocol specification," *Version*, vol. 2, no. 1, pp. 198–207, 2005.

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[11] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," in *Black Hat Europe*, 2015.

[12] *Tesla's autopilot has had its first deadly crash*, https://www.wired.com/2016/06/teslas-autopilot-first-deadly-crash/, Accessed: 2016-07-05.

[13] A. Serageldin, H. Alturkostani, and A. Krings, "On the reliability of dsrc safety applications: A case of jamming," in *2013 International Conference on Connected Vehicles and Expo (ICCVE)*, Dec. 2013, pp. 501–506.

[14] M. Hamad, M. Nolte, and V. Prevelakis, "Towards comprehensive threat modeling for vehicles," in *1st Workshop on Security and Dependability of Critical Embedded Real-Time Systems*, M. Völp, P. Esteves-Verissimo, A. Casimiro, and R. Pellizzoni, Eds., co-located with the IEEE Real-Time Systems Symposium 2016, IEEE, Porto, Portugal, Dec. 2016, pp. 31–36.

[15] E. Buchanan, R. Roemer, H. Shacham, and S. Savage, "When good instructions go bad: Generalizing return-oriented programming to RISC," in *Proceedings of CCS 2008*, P. Syverson and S. Jha, Eds., ACM Press, Oct. 2008, pp. 27–38.

[16] C. of Oklahoma County, *Bookout vs toyota*, http://www.safetyresearch.net/Library/Bookout_v_Toyota_Barr_REDACTED.pdf, Accessed: 2016-07-22.

[17] M. Lv, N. Guan, J. Reinecke, R. Wilhelm, and W. Yi, "A survey on static cache analysis for real-time systems," *Leibnitz Transactions on Embedded Systems*, vol. 3, no. 1, pp. 1–48, Jun. 2016.

[18] C. Hayes, *Driving along at full speed for autonomous vehicles*, Jan. 2016.

[19] ARM, *AMBA5 ahb protocol specification*, Oct. 2015.

[20] Z. Zhang and X. Koutsoukos, "Cache-related preemption delay analysis for multi-level inclusive caches," in *2016 International Conference on Embedded Software (EMSOFT)*, Oct. 2016, pp. 1–10.

[21] J. Liedtke, H. Hartig, and M. Hohmuth, "Os-controlled cache predictability for real-time systems," in *Proceedings Third IEEE Real-Time Technology and Applications Symposium*, Jun. 1997, pp. 213–224.

[22] *Free RTOS*.

[23] R. Mancuso, R. Dudko, E. Betti, M. Cesati, M. Caccamo, and R. Pellizzoni, "Real-time cache management framework for multi-core architectures," in *19th IEEE International Conference on Real-Time and Embedded Technology and Applications Symposium (RTAS 2013)*, Philadelphia, PA, USA.

[24] B. Ward, J. Herman, C. Kenna, and J. Anderson, "Making shared caches more predictable on multicore platforms," in *25th Euromicro Conference on Real-Time Systems*, Jul. 2013, pp. 157–167.

[25] N. Kim, B. Ward, M. Chisholm, C.-Y. Fu, J. Anderson, and F. Smith, "Attacking the one-out-of-m multicore problem by combining hardware management with mixed-criticality provisioning," *Real-Time Systems, special issue of outstanding papers from the 22nd IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2016)*, vol. 53, no. 5, pp. 709–759, Sep. 2017.

[26] K. Krüger, M. Völp, and G. Fohler, "Improving security for time-triggered real-time systems against timing inference based attacks by schedule obfuscation," in *Euromicro Conference on Real-Time Systems (ECRTS) - Work-in-progress Session*, 2017.

[27] H. Yun, R. Mancuso, Z.-P. Wu, and R. Pellizzoni, "PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms," in *Intl. Conference on Real-Time and Embedded Technology and Applications Symposium (RTAS)*, IEEE, 2014.

[28] P. Vivekanandan, G. Garcia, H. Yun, and S. Keshmiri, "A simplex architecture for intelligent and safe unmanned aerial vehicles," in *International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, IEEE, 2016.

[29] D. Hildebrand, "An architectural overview of qnx," in *Proceedings of the Workshop on Micro-kernels and Other Kernel Architectures*, 1992, pp. 113–126.

[30] J. Liedtke, "On micro-kernel construction," in *Proceedings of the 15th ACM Symposium on Operating System Principles*, 1995, pp. 237–250.

[31] (). Fiasco, [Online]. Available: https://os.inf.tu-dresden.de/fiasco/.

[32] G. S. Veronese, M. Correia, A. N. Bessani, L. C. Lung, and P. Veríssimo, "Efficient byzantine fault-tolerance," *IEEE Trans. Computers*, vol. 62, no. 1, pp. 16–30, 2013.

[33] T. Distler, C. Cachin, and R. Kapitza, "Resource-efficient byzantine fault tolerance," *IEEE Trans. Computers*, vol. 65, no. 9, pp. 2807–2819, 2016.

[34] F. Borran and A. Schiper, "A leader-free byzantine consensus algorithm," in *Int. Conf. on Distributed Computing and Networking (ICDCN)*, 2010.

[35] M. Correia, N. F. Neves, and P. Verissimo, "Bft-to: Intrusion tolerance with less replicas," *The Computer Journal*, vol. 56, no. 6, pp. 693–715, Jun. 2013.

[36] K. Marzullo, "Tolerating failures of continuous-valued sensors," *ACM Transactions on Computer Systems*, vol. 8, no. 4, pp. 284–304, Nov. 1990.

[37] U. Schmid and K. Schossmaier, "How to reconcile fault-tolerant interval intersection with the lipschitz condition," *Dirstibuted Computing*, vol. 14, no. 2, pp. 101–111, May 2001.

[38] J. Rushby, "Formal verification of marzullo's sensor fusion interval," SRI International, Tech. Rep., Jan. 2002.

[39] S. Bak, D. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha, "The system-level simplex architecture for improved real-time embedded system safety," in *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, IEEE, 2009, pp. 99–107.

[40] P. Verissimo and A. Casimiro, "The timely computing base model and architecture," *IEEE Transactions on Computers*, vol. 51, no. 8, pp. 916–930, Aug. 2002.

[41] R. Pucella and F. B. Schneider, "Independence from obfuscation: A semantic framework for diversity.," in *19th IEEE Work. on Computer Security Foundations*, 2006, pp. 230–241.

IDHCC: A Security-Enhanced ID Hopping CAN Controller Design to Guarantee Real-Time

Wufei Wu[1,2], Ryo Kurachi[2], Gang Zeng[3], Yutaka Matsubara[2], Hiroaki Takada[2], Renfa Li[1]
[1]College of Computer Science and Electronic Engineering, Hunan University, Changsha, China.
[2]Graduate School of Informatics, Nagoya University, Nagoya, Japan.
[3]Graduate School of Engineering, Nagoya University, Nagoya, Japan.
wufeiwu@hnu.edu.cn, kurachi@nces.is.nagoya-u.ac.jp, sogo@ertl.jp, yutaka@ertl.jp, hiro@ertl.jp, lirenfa@hnu.edu.cn

***Abstract*—Controller Area Network (CAN) is the most widely used protocol for safety critical applications in current vehicle electronic systems. The security enhancement of CAN is a multi-constrained and cost-sensitive optimization problem, our aim is to propose a real-time and security mechanism. First of all, we propose a novel ID (identify) hopping CAN (IDH-CAN) mechanism to address both security and safety constraints. Second, to improve the security performance of CAN, we design and implement the IDH-CAN controller (IDHCC) on FPGA, which works as a hardware firewall in the data link layer to isolate the applications from the physical layer. Third, our simulation and practical evaluations demonstrate the effectiveness of this approach in defense reverse engineering, targeted DoS and replay attacks without violating design constraints and highlight the importance of considering security together with other metrics during the design stages for automotive real-time applications.**

***Index Terms*—Autonomous; Vehicle Security; ID hopping; Controller Area Network (CAN); Real time; Schedulability**

I. Introduction

The security attacks against CAN bus have attracted increasing research attention[1]. As mentioned in [1][2] and [3], CAN does not offer a security mechanism, such as data frame authentication or encryption, but ECUs connected by CAN are usually safety-critical (i.e., anti-lock braking system (ABS) and electronic stability program (ESP)) components that need to guarantee strict end-to-end latencies. Attacks against a CAN bus will lead to privacy disclosure and even life and property threats in extreme circumstances. As defined by ISO 26262 (road vehicles functional safety) [4], CAN clusters in vehicles need a high automotive safety integrity level (ASIL). If the CAN bus has no security mechanism for data transmission, then the attacker can easily monitor or even control the vehicle. The situation becomes worse when the vehicle is autonomous vehicle[5].

To ensure automobile information security, some standards were established. SAE J3061[6] was published in January 2016 by the Society of Automotive Engineers (SAE) international, which defined the process framework of a security lifecycle for the security lifecycle of cyber-physical vehicle systems. The software specification in the AUTOSAR Secure Onboard Communication (SOC) module[7] was developed to create resource-efficient and practicable authentication mechanisms for critical data transport among ECUs. This specification is based on the assumption that symmetric authentication approaches with message authentication code (MAC) were used (e.g., a CMAC[8] based on AES[9] with an adequate key) [10][11] . However, given the limited data field of CAN (maximum of 64 bits), the MAC (256 bits) needs to be divided into four parts, and each part needs to be included in the subsequent messages. This process consumes additional time and bandwidth.

In this study, we propose a hardware-based ID hopping CAN controller (IDHCC) an conduct several experiments to test its security performance under the design constraints. The main contributions of this paper are as follows:

1) A security enhancement CAN mechanism based on ID hopping called IDH-CAN was designed. This mechanism protects the network from sniffing as well as targeted DoS attacks and replay attacks.
2) A hardware-based CAN controller intellectual property (IP) for IDH-CAN called IDHCC was designed and implemented on a field-programmable gate array (FPGA). This controller works on the data-link layer and is compatible with the existing upper layer protocols and applications, such as SAE J1939.
3) The security performance of the proposed mechanism was analyzed. Several different security mechanisms were compared with IDH-CAN.

The rest of the paper is organized as follows. Section II introduces the research background and related work. Section III describes attack model of our method. Section IV describes our proposed solution as well as the basic idea and details of IDHCC, including its operation, synchronous method and error recovery mechanism. Section V discusses the hardware implementation of IDHCC. Section VI evaluates IDHCC from resource consumption and security performance. Section VII concludes the paper.

II. Background and Related Works

A. Security research on CAN

Different security mechanisms have been designed to protect CAN from attacks and meet its security demands. The existing CAN security methods can be divided into three categories. The methods in the first category are authenticated by digital signature and MAC, which will consume limited payload of CAN message frame [12][13] and computing

resources of node. Compare to the encrypted CAN messages, due to the constraints of CAN system (e.g., bandwidth and real-time), ID hopping are more suitable for CAN message. The second category includes CAN+ [14][15], a CAN-specific authentication model that requires additional messages. The third category is anonymous IDs. In ESCAR Europe 2015, Han et al. introduced a new solution called the identity-anonymity CAN (IA-CAN) protocol [16]. The major drawback of IA-CAN is cannot guarantee the worst-case response time (WCRT) of CAN messages, because the ID is randomly generated (ID field of CAN is used for message arbitration). Therefore, the IA-CAN is not suitable for time-sensitive systems.

B. Related Work

The most relevant work to our method is the ID hopping mechanism proposed by Humayed et.al [17], the authors used a special ID from the gateway as a parameter of hopping synchronization and then implemented ID hopping using a software way. Therefore, the switching time is unavoidable. Our approach differs from the ID hopping mechanism in three ways. First, they choose software implementation, we implement with hardware, therefore, our method does not require additional computation time. Second, different ways of hopping synchronization, special ID was used in [17] to achieve synchronization, it means that the additional message is unavoidable. But the message counter was used in our researches. Due to the CAN protocol's broadcast nature, message counters is a parameter that is shared instantaneously by nodes on CAN bus. Therefore, our method requires less additional messages. Third, two methods have different ways of generating $ID\_hopping\_table$. In [17], the new IDs are equal to the previous IDs add offset after hopping in the runtime. However, in our method, to increasing the diversity of IDs, $ID\_hopping\_table$ is generated in the phase of system design, and stored in a tamper-proof storage (e.g. in a SHE or HSM secure memory) to ensure that only authenticated nodes can access the table.

III. ATTACK MODEL

In our threat model, we assume attackers can access CAN bus. The available methods, including but not limited to: Bluetooth, OBD_II, WiFi, physical access and USB. The attacker can be easy to reverse of CAN messages in real vehicles, because there is no message authentication in CAN bus. Once an attacker gains access to the CAN, the attacker can sniffing, spoofing, replay, and denial-of-service (DoS) attacks. To make up for this deficiency, references to [18], we constructed an attack model as shown in Figure 1, among them, the high-priority DOS attack is the type of attack that we currently cannot resolve. Most of the available security measures for CAN focus on protect the message payload, but our threat model focuses on improving the security of the ID field. The motivation of this paper is to provide the countermeasures for sniffing and reverse engineering of CAN messages by integrating the CAN-ID masquerade techniques.

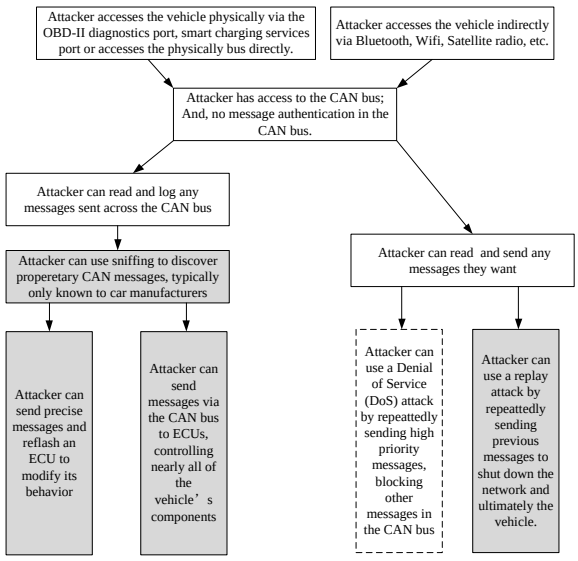


Fig. 1. An attack model for the CAN bus.

This means that our approach not only increases the difficulty of reverse-engineering, but also can resist most attacks against CAN, except for high-priority DoS attacks.

IV. PROPOSED MECHANISM

In this section, we present a security mechanism under real-time constraints, which is based on the randomly change of sending CAN-ID to protect sniffing and replay of CAN messages. To simplify the complexity of the problem, we make the following assumptions which are reasonable in practical applications[19]: 1) the sender and receiver of each message have been assigned to specific ECUs, and 2) the allocation of the ID for the CAN message has been completed from the application layer viewpoint.

A. ID hopping CAN

Figure 2 presents an overview of the ID hopping mechanism for CAN based on the message counter and $ID\_hopping\_table$. Two kinds of IDs are available in IDH-CAN, App_IDs is used as the CAN-ID of CAN message in application software, while the physical layer ID is defined as Phy_IDs, which can be obtained by looking up the $ID\_hopping\_table$. $ID\_hopping\_table$ (with maximum 16 pages Phy_IDs) is generated in the design phase based on App_IDs and will be written on the IDH-CAN controller. The idea is that every IDHCC will have multiple translation tables ($ID\_hopping\_table$), where each page maps "natural" message IDs (App_IDs) to other IDs (Phy_IDs). The IDs of sent messages are translated via the currently active mapping page before they are placed on the wire. Similarly the IDs of the received messages in which an ECU is interested are converted via the reverse mapping page (in the receive "filter"). All the ECUs switch from page to page according to a global message count, because CAN is a broadcast network. Those messages with a Phy_ID, which is un-mapped in

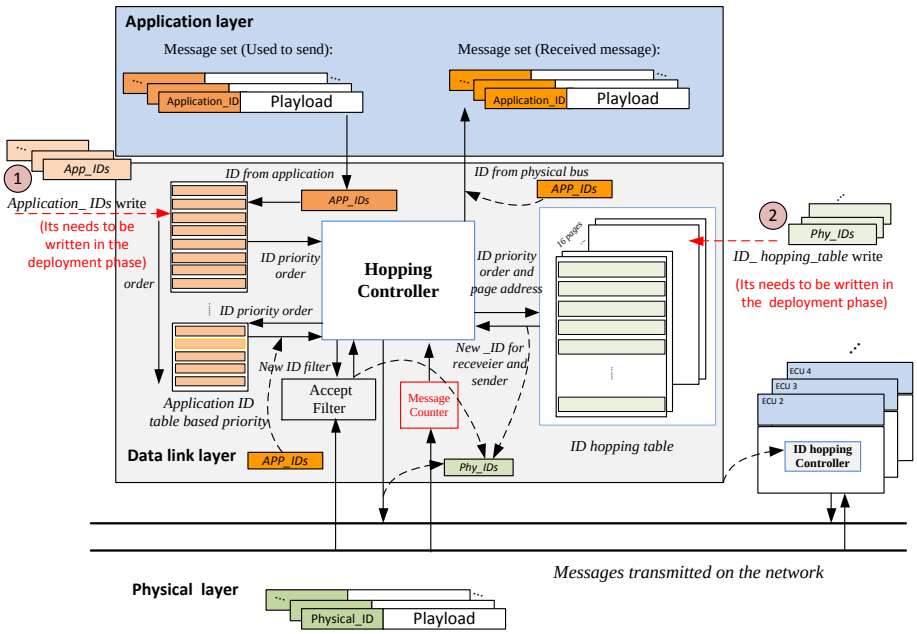


Fig. 2. Overview of ID hopping mechanism for CAN based on the message counter and $ID\_hopping\_table$.

$ID\_hopping\_table$ will be discarded (allegedly making it difficult for the attacker to inject messages). An important aspect is that each hopping must maintain the relative numerical order of message App_IDs (given that the ID serves as a priority in CAN, switching the relative order of two IDs will change the WCRT of the messages).

As shown in Figure 3, the ID hopping tables are stored in IDH-CAN controllers of transmitter node and receiver node respectively, the size of the table equal to $Phy\_IDs * pages\_number$ (e.g., $4 * 11$ in this examples). After ID hopping is performed, one page of $ID\_hopping\_table$ is choose as Phy_IDs in IDH-CAN controllers based on message counter, the priority order of Phy_IDs is guaranteed to fixed according to the order of App_IDs. For the sending node, the IDHCC maps the App_ID to the Phy_ID based on the priority order of the message. Therefore, the actual transmitted message ID becomes more diverse. For receiving nodes, the

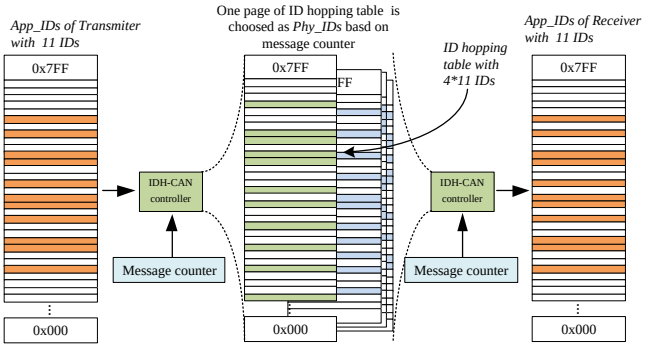


Fig. 3. An examples of IDH-CAN.

IDHCC maps the Phy_ID to the App_ID by looking up the $ID\_hopping\_table$ based on the priority order of the message. Meanwhile, the receiving ECUs use the current page of $ID\_hopping\_table$ to get the new filter registers to filter the incoming data frames before verifying data integrity. Invalid data frames are filtered without requiring any additional runtime computations, because our ID hopping techniques are implemented in CAN controller hardware.

Arbitration ID hopping in this study is synchronized by the message counter, when 8 messages are transmitted over the network, there will be a hopping in the CAN network. We will improve security by designing the hopping rule in our feature work (et.al, different number of messages as the hopping cycle or different hopping distance can be set). Therefore, while all message relative priority order is fixed for authentication nodes, there is no way to determine the relative priority sequence of messages for non-authentication nodes. Because they don't know the hopping rules.

For gateway environment: Message counter in different CAN clusters is different. But multiple CAN clusters interconnection through the gateway is common for automotive electronic system. For messages that need to cross gateways, the hopping mechanism and available range of IDs in different CAN clusters can be different. For example, if messages from cluster 1 need transmit to cluster 2. First of all, the Phy_IDs_1 from cluster 1 maps to its App_IDs_1 in gateway node, and then App_IDs_2 maps to App_IDs_2. Then, according to the target cluster 2's hopping mechanism and ID hopping tables, the App_IDs_2 changed to Phy_IDs_2. At last, The message is transmitted. Namely, the cross-gateway messages need to be mapped twice: cluster 1 (Phy_IDs_1)$\rightarrow$ gateway($App\_IDs\_1\rightarrow$ App_IDs_2

)→ cluster 2 (Phy_IDs_2), which means that the message counters are not need to synchronized between different CAN cluster.

B. Schedulability analysis

Consequently, CAN message usually has a hard deadline constraint, which is denoted by D_m. The tasks on the receiving nodes may have multiple timing requirements on the message, but in such a hard real-time system, we assume that D_m is the tightest time constraint. Therefore D_m must be met in the security enhancement solution. A message is said to be schedulable if and only if its worst-case response time is less than or equal to its deadline ($R_m <= D_m$). The system is schedulable if and only if all of the messages in the system are schedulable.

As described in [20], the message response time R_m is composed of three terms, namely the release jitter J_m, the queue delay $\mathcal{W}_m$, and the maximum transmission time C_m.

$$R_m = J_m + \mathcal{W}_m + C_m \tag{1}$$

The maximum transmission time C_m is determined by the message payload s_m (1 to 8 bytes), the ID format (11 or 29 bit), and the bit time. The queue delay $\mathcal{W}_m$ is determined by two factors, namely the blocking factor B_m due to non-preemptive message transmission and the interference due to higher priority messages (denoted by the set $hp(m)$).

$$\mathcal{W}_m^{n+1} = \max\left(B_m, C_m\right) + \sum_{\forall k \in hp(m)} \lceil \frac{\mathcal{W}_m^n + J_k + \tau_{bit}}{T_k} \rceil C_k \tag{2}$$

A simple upper bound on the blocking factor B_m is given by the transmission time of the longest message on the network.

$$B_m = \max_{\forall k \in lp(m)} \{C_k\} \tag{3}$$

Based on formulas (2) and (3), the available time for the ID hopping in the CAN bus system can be computed as

$$C_m = \left(g + 8S_m + 13 + \lfloor \frac{g + 8S_m - 1}{4} \rfloor\right) \tau_{bit} \tag{4}$$

where S_m is the data field of CAN that is assumed to be 8 bytes. g (i.e., the worst-case bit-stuffing length) is equal to 34 for standard frame format or 54 for extended format (29-bit ID field), and τ_{bit} is the bus bit rate. For 11-bit identifiers at 1 Mbps bit rate, C_m is equal to 0.136ms. Given the limited available time for ID hopping the hardware implementation plan for the ID hopping mechanism is selected in this study.

Previous works on the security enhancement of CAN did not consider the problem of schedulability analysis. For example, given the $\mathcal{W}_m$ in CAN+ and the fact that Identifys Anonymous CAN (IA CAN) has been changed, their schedulability analysis model need to be changed inevitably. Given that the IDH-CAN is implemented, its results guarantee the invariability of the schedulability analysis model of CAN introduced in [21].

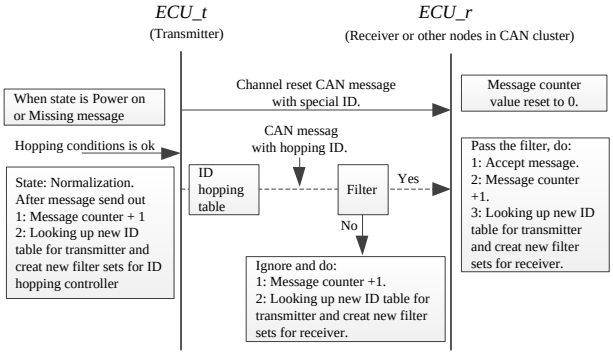


Fig. 4. Process diagram of IDH-CAN.

C. Operation of IDH-CAN

This section explains how IDH-CAN works in automotive electronic system development. For ease of understanding, a process diagram of IDH-CAN is presented in Figure 4. The deployment of IDH-CAN is divided into five steps from the perspective of the designer.

- **Step 1:** According to the functional requirements of the system, the number of message set N and its priority order $P\{p_1, p_2, ..., p_n\}$ can be determined. The number of messages used in one cluster is not large and is usually within 256 or less [22].
- **Step 2:** Each message in the CAN cluster is assigned a ID as a priority parameter. A total of 2048 possible message IDs can be placed in one CAN cluster. According to C_{2048}^N, the available ID combinations for Phy_IDs are very large.
- **Step 3:** According to the relationship between ECUs and messages, after Step 2, the acceptable message set named RX_set can be determined for each ECU and each ID page at different message counter moments. With RX_set, the filter registers for each ECU at each ID pages can be obtained. New ID hopping page for IDH-CAN can be obtained by adding the registers to the ID hopping pages generated in Step 2.
- **Step 4:** In the design or upgrade phase of the vehicle electronic system design, the $ID\_hopping\_table$ and the sorted App_IDs are written to the ID hopping controller (details will be introduced in Section V).
- **Step 5:** Startup or restart the system.

D. ID hopping synchronization

Synchronization is especially important for ID hopping CAN and anonymous ID. Through the CAN analysis and CAN controller implementation, In this study, the message counter is used for ID hopping synchronization (select one page from the $ID\_hopping\_table$ as a new one). Where the ACK signal in the data link layer can be used to count the message counter, which can be used as an ID hopping synchronization parameter. The ID hopping is performed on a cycle of 8. That is, after 8 messages are transmitted over the bus, each IDHCC get this shared parameter, and ID hopping

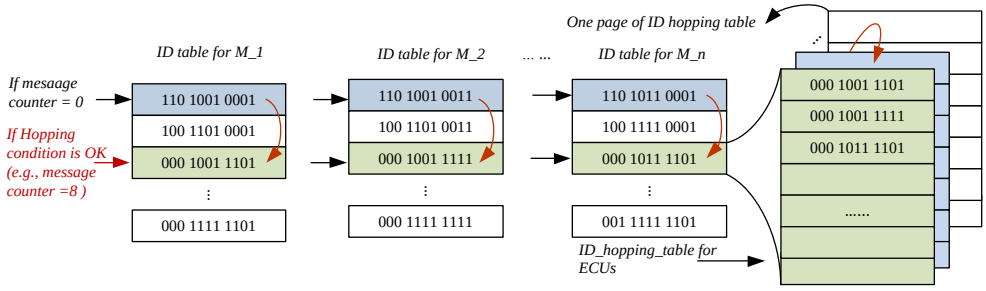


Fig. 5. ID hopping synchronization from the perspective of the message and ECU.

occurs synchronously. Because the relative priority order of the message is constant in each page, the priority order is used as a parameter to the message mapping.

As shown in Figure 5, from the viewpoint of the message, the ID of the message at each moment will correspond to a new ID based on the $ID\_hopping\_table$ and priority order (according to the lookup Application IDs table) of the message. However, from the perspective of the ECU, each combination can be treated as one ID page. The $ID\_hopping\_table$, as a 3D array, can be written to the IDHCC. Taking into account the limited register resources and cost factors, the table size in this study is divided to $4, 8$ and 16 pages with 256 IDs per page.

E. Startup and recovery mechanism

Synchronize of hopping described in this study is based on the message counter. The failure of the node to monitor the message will lead to synchronization failure (i.e., missing messages). Although, according to the characteristics of CAN, this phenomenon rarely occurs. But, a robust recovery mechanism for IDH-CAN is necessary, especially for an autonomous vehicle system. In this study, a startup message with special ID (such as $0x01$) is used. In the case where a network idle is detected and message counter equal to the threshold of hopping (e.g., 8 in this study), a startup message carrying a byte load (used to select the next page from $ID\_hopping\_table$) is sent by one of the nodes. When the other node receives the message, it will carry on the ID hopping, realize the error recovery and start.

V. Hardware-based implementation

A. Hardware-based implementation

To implement IDH-CAN, we present an ID hopping CAN controller design and discuss its implementation. This security enhancement CAN controller is based on the OpenCores open source IP core community CAN controller designed by Mohor [23]. The IP core uses the Verilog hardware description language to create the SAJ1000 controller. The supporters of CAN 2.0A/B protocol can also support CAN standard frames (11-bit identification code) and extended frames (29-bit identification code). CAN 2.0A/B has 64 bytes receive FIFO, and the rate can up to 1 Mbit/s. As shown in Figure 6, four modules are present in the top-level module of the IDHCC, namely $ID\_hopping\_module$, $can\_registers$, can_btl, and can_bsp. $ID\_hopping\_module$ has two single modules, namely $app\_id\_priority\_table$ and $ID\_hopping\_table$. The $app\_id\_priority\_table$ gets from the priority order sorted App_IDs. With message counter synchronization, the ID hopping controller controls the process of hopping.

Hopping controller: According to the message counter, the ID hopping controller (part of the CAN controller) converts App_IDs to Phy_IDs for the transmission message. Under the action of the controller, receive filter registers can be obtained from the new page of $ID\_hopping\_table$. The ID hopping controller looks up the $ID\_hopping\_table$ with different message counter values for different messages. The hopping controller module is connected to the can_bsp module via a 32-bit bus.

Application IDs table: An application IDs table named $app\_id\_priority\_table$ contains the message IDs from the application layer software that need to be sent or received. The IDs in the table is sorted based on their priority. Accordingly, this table will be used to find the priority of the messages and to obtain the message IDs based on their priority. The maximum size of the application IDs table in this paper is 256.

ID hopping table: To keep the order of priority, we need to guarantee that each ID combination has a relative priority order. The $ID\_hopping\_table$ contains the message ID for the physical layer message that will be transmitted on the CAN bus. One page of the ID table is defined as $ID\_field\_length * ID\_depth$(maximum 256), and the data in each page of table is sorted by priority.

Accept filter: According to the message set that the node needs to receive, the acceptance code and acceptance mask registers can be obtained at the same time. IDHCC has four acceptance code and acceptance mask registers, respectively. These registers will be written to the $can\_registers$ module, and work in the CAN_acf module. The receiving nodes are included in the CAN_acf module to decide whether to accept or discard a received data frame.

B. Data flow in IDH-CAN

To clearly describe the proposed IDH-CAN mechanism, we describe the data flow in IDH-CAN in Figure 7.

For transmitter link: After the message is transferred to the CAN controller by reading registers, the priority of the

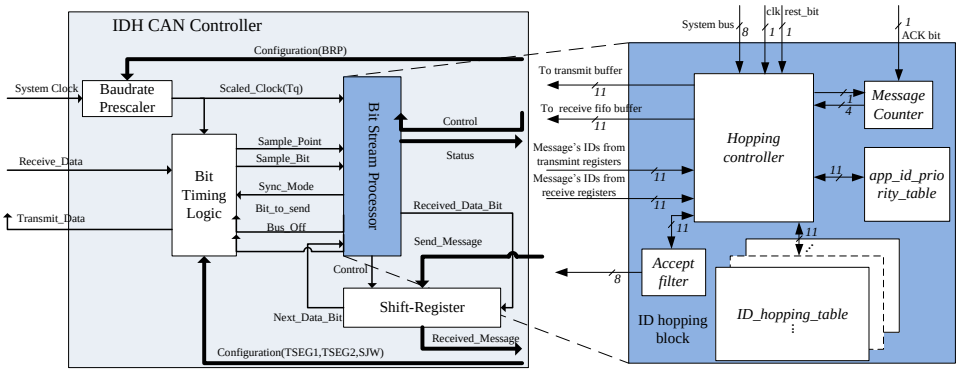


Fig. 6. Structure diagram of the ID hopping CAN controller.

message can be obtained by looking up the priority order of $app\_id\_priority\_table$, which be written to IDHCC in design phase too. The App_IDs can then be converted to Phy_IDs by looking up the $ID\_hopping\_table$ for sending (synchronized by the message counter). Therefore, the ID transmission on the physical layer actually will be different.

For receiver link: When a new message is transported on the CAN bus, if the ID of the message passes the accepting filter, then message will be processed by the ECU. First, the relative priority of the message can be obtained by looking up the $ID\_hopping\_table$ based on the Phy_ID and message counter, and the App_IDs can be obtained by looking up the application IDs table with the relative priority. Second, the ID for the application software layer is obtained.

C. Waveform simulation

Before deploying to the FPGA platform, we design a test bench to verify the correctness of the IDHCC design. The environment is ModelSim ALTERA STARTER EDITION 10.1d, and the test includes sending and receiving tests for the IDHCC. The first step of this test bench is to write IDs for both $app\_id\_priority\_table$ and $ID\_hopping\_table$ in the IDHCC. The simulation test can be divided into the following aspects.

In the sending state, as shown in Figure 8, the ID that needs to be sent is 00000000000. However, the transmitted ID waveform on the bus is converted to 10101010100 as the

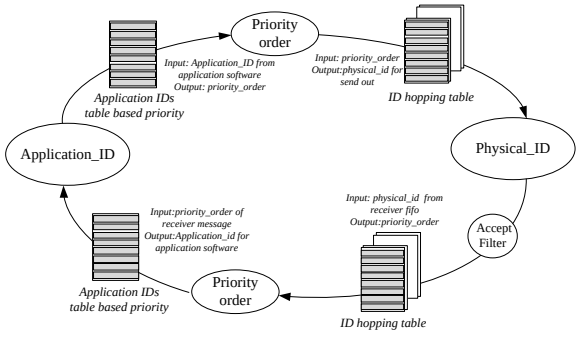


Fig. 7. Data flow in IDH-CAN.

```
task send_frame_basic;    // CAN IP core sends frames
  begin
    //write register for message transmitter send out
    write_register(8'd10, 8'h00); // Writing ID[10:3] = 0x00000000
    write_register(8'd11, 8'h08); // Writing ID[2:0] = 0x001, rtr = 0, length = 8
    write_register(8'd12, 8'h56); // data byte 1
    write_register(8'd13, 8'h78); // data byte 2
    write_register(8'd14, 8'h9a); // data byte 3
    write_register(8'd15, 8'hbc); // data byte 4
    write_register(8'd16, 8'hde); // data byte 5
    write_register(8'd17, 8'hf0); // data byte 6
    write_register(8'd18, 8'h0f); // data byte 7
    write_register(8'd19, 8'hed); // data byte 8
```

The application layer's ID is written to the register

Fig. 8. The message ID is written to the register.

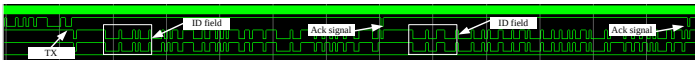


Fig. 9. The waveform on the physical CAN bus.

waveform shows in Figure 9. If we keep sending the same ID, then the waveform will be different. Hopping synchronization occurs based on the ACK signal.

The simulation test shows that the IDHCC can work correctly and that the IDs transmitted on the physical layer change according to the message counter (trigger via the ACK signal).

VI. Evaluation

The results in Section V-C show that IDH-CAN can work in the hopping module correctly. To evaluate the effectiveness of our solution, we evaluate IDH-CAN from three aspects, namely resource consumption, performance, and security effectiveness. Several numerical examples are presented to confirm the proposed mechanism.

A. Resource consumption

Given that the automotive electronics design is cost sensitive, the new IDHCC must minimize the occupation of hardware resources. As described in Table I, the comparison with normal CAN controller [23] shows that IDH-CAN has improved security with slightly increased hardware resources.

B. Performance analysis

The development platform employed in this study is Altera DE0-Nano with EP4CE22F17C6. The experimental results on

TABLE I
COMPARISON OF RESOURCE CONSUMPTION

	CAN controller	**ID hopping CAN controller**
Device	EP4CE22F17C6	EP4CE22F17C6
Total logic elements	2,313 / 22,320 (10 %)	2,362 / 22,320 (11 %)
Total combinational functions	1,927 / 22,320 (9 %)	1,996 / 22,320 (9 %)
Dedicated logic registers	1,175 / 22,320 (5 %)	1,175 / 22,320 (5 %)
Total pins	19 / 154 (12 %)	30 / 154 (19 %)

TABLE II
PERFORMANCE COMPARISON OF CAN SECURITY ENHANCEMENT METHODS

	MAC[12] [13]	**CAN+ [15][24][11] [14]**	**IA-CAN[16]**	**ID hopping[17]**	**Proposed**
Computational complexity	High	Middle	Low	Low	Low
Time delay	Middle	High	No	Low	No
Play load consumption	High	No need	Middle	No need	No need
Additional messages	No need	Need	No need	Need	One
Schedulability analysis	Easy	Complex	Complex	Complex	Easy

the FPGA platform show that the designed controller can always reach 25 MHz.

Communication response time: Given that synchronization is achieved using the message counter, the start message is fixed to the highest priority and fixed message counter cycles. The experiments show that IDHCC can work at 1 Mbps. The number of instructions in one cycle required for the IDHCC to complete the hopping operation is 3 step. Theoretically, the time required for the IDHCC to complete the hopping process is 240 µs, which is less than C_m 0.136 ms as described in Section IV-B.

Schedulability analysis: IDH-CAN does not need additional payloads to achieve ID hopping techniques, and the schedulability analysis model of CAN messages is guaranteed because of the fixed priority order and the constant message time model. Therefore, IDH-CAN is compatible with the existing designs in the vehicle environment.

C. Success rates of attacks

We analyze the security performance of IDH-CAN from three aspects, namely, targeted DoS attacks, replay attacks, and reverse engineering.

Targeted DoS attacks. Targeted DoS attack which uses special messages for targeted ECU. Our method can against this kind of DoS attacks by ID hopping mechanism. For a physical attack, when the ID jump occurs, the ID in the previous page of $ID\_hopping\_table$ will become illegal in the next page of $ID\_hopping\_table$.

Replay attacks. In a system equipped with IDHCC, N_pages denotes the pages of the $ID\_hopping\_table$, while $N_{num}\_App\_IDs$ denotes the number of IDs in application layer. The success rate of the replay attacks against the IDH-CAN system can be computed as

$$\frac{1}{N_{num}\_Phy\_IDs} \tag{5}$$

where $N_{num}\_Phy\_IDs$ is the total number of IDs in the physical CAN bus with diversity of Phy_IDs. For normal CAN, $N_{num}\_Phy\_IDs$ is equal to $N_{num}\_App\_IDs$, but in IDH-CAN, $N_{num}\_Phy\_IDs$ is equal to $N_{num}\_App\_IDs * N\_pages$. Therefore, the theoretical success rate of replay attacks against IDH-CAN is computed as

$$\frac{1}{N_{num}\_App\_IDs * N\_pages} \tag{6}$$

Obviously, this probability is less than that recorded in normal CAN (which is $\frac{1}{N_{num}\_App\_IDs}$). In IDH-CAN, the diversity and non-formatting of the messages transmitted on the network significantly increase the probability for reverse engineering and violent attacks. An attacker cannot determine the priority order of the message.

Reverse engineering: Data collection analysis is often the first step in the attack process[14]. In [22], the authors find that the CAN messages collectively have low entropy with an average of 11.436 bits. Moreover, in-vehicle ECUs usually use IDs ranging form $0x000$ to $0x5FF$ while the automotive diagnostic tool uses IDs ranging from $0x700$ to $0x7FF$[14]. In summary, the ID structure of the existing CAN protocol is fixed, has a poor diversity, and is prone to reverse engineering. To compare diversity of IDs between normal CAN, this study provides the entropy analysis of IDH-CAN with SAE Benchmark based message set, comprised of both time-triggered and event-triggered messages. Figure 10 shows the result of comparison between IDH-CAN, normal CAN and proposal in [17]. The results of the entropy analysis showed that IDH-CAN has bigger entropy in each number of message set. Therefore, IDHCC can play a role in preventing reverse engineering.

Table II presents some of the advantages of IDHCC: first, compared with MAC, IDH-CAN does not need to consume data-field and additional messages and does not take up

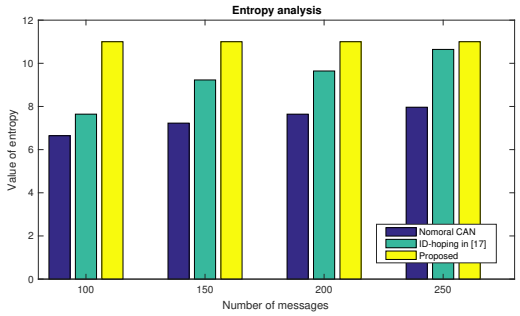


Fig. 10. Entropy comparison.

additional computing resources and time, although IDH-CAN controller is needed. Second, compared with CAN +, IDH-CAN only need one additional message for synchronization of message counters, which means higher effective bandwidth and utilization. Third, compared with IA-CAN, our proposed method is easy to achieve the schedulability (WCRT) and randomization of CAN messages. Fourth, compared with the ID hopping mentioned in [17], the IDs in IDH-CAN are more diverse and have less processing time in software.

VII. Conclusion

In this paper, a non-ISO IDH-CAN controller IP core was designed and implemented based on the traditional controller standard. Therefore, this controller can be used with different MCUs and in the FPGA platform. Moreover, if ECUs are equipped with our IDH-CAN controller, then the schedulability analysis model for CAN messages is applicable. This unchanged model is especially important for security-critical systems (such as the automotive domain). In addition, we fully consider the compatibility of this controller with the previous designs and the upper layer protocols and software.

This research contributes to the literature by guaranteeing that the CAN bus can defend the system from anti-engineering, sniffing, targeted DoS attacks and replay of messages attacks without requiring much bandwidth resources, additional messages and long calculation time. A future research direction would be how to improve the security performance of IDHCC. For example, the hopping rule in IHDCC can be setting by using a secret key to defend the application layer from attacks at same time.

Acknowledgment

This work was partially funded by China Scholarship Council under Grant Number 201606130063 and the National Natural Science Foundation of China under Grant Number 61672217.

References

[1] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*. San Francisco, 2011.

[2] J. Petit and S. E. Shladover, "Potential cyberattacks on automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 546–556, 2015.

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

[4] "Road vehicles-functional safety, iso 26262," 2011.

[5] B. Zheng, H. Liang, Q. Zhu, H. Yu, and C.-W. Lin, "Next generation automotive architecture modeling and exploration for autonomous driving," in *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*. IEEE, 2016, pp. 53–58.

[6] C. Schmittner, Z. Ma, C. Reyes, O. Dillinger, and P. Puschner, "Using sae j3061 for automotive security requirement engineering," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2016, pp. 157–170.

[7] C. Bernardeschi, G. Dini, and G. Del Vigna, "Security modeling and automatic code generation in autosar," 2016.

[8] M. J. Dworkin, "Recommendation for block cipher modes of operation: The cmac mode for authentication," *Special Publication (NIST SP)-800-38B*, 2016.

[9] N. F. Pub, "197: Advanced encryption standard (aes)," *Federal information processing standards publication*, vol. 197, no. 441, p. 0311, 2001.

[10] C. Szilagy and P. Koopman, "A flexible approach to embedded network multicast authentication," 2008.

[11] A. Hazem and H. Fahmy, "Lcap-a lightweight can authentication protocol for securing in-vehicle networks," in *10th escar Embedded Security in Cars Conference, Berlin, Germany*, vol. 6, 2012.

[12] C.-W. Lin and A. Sangiovanni-Vincentelli, "Cyber-security for the controller area network (can) communication protocol," in *Cyber Security (CyberSecurity), 2012 International Conference on*. IEEE, 2012, pp. 1–7.

[13] D. K. Nilsson, U. E. Larson, and E. Jonsson, "Efficient in-vehicle delayed data authentication based on compound message authentication codes," in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*. IEEE, 2008, pp. 1–5.

[14] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle can," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 993–1006, 2015.

[15] P. Mundhenk, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, and S. Chakraborty, "Lightweight authentication for secure automotive networks," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 2015, pp. 285–288.

[16] A. W. Kyusuk Han and K. G.Shin, "Apractical solution to achieve real-time performance in the automotive network by randomizing frame identifier." Embedded Security in Cars (escar) Europe, October 26, 2015.

[17] A. Humayed and B. Luo, "Using id-hopping to defend against targeted dos on can," in *Proceedings of the 1st International Workshop on Safe Control of Connected and Autonomous Vehicles*. ACM, 2017, pp. 19–26.

[18] H. Ueda, R. KURACHI, H. TAKADA, T. MIZUTANI, M. INOUE, and S. HORIHATA, "Security authentication system for in-vehicle network," *SEI Technical Review*, no. 81, 2015.

[19] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1204–1223, 2005.

[20] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (can) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[21] R. I. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Schedulability analysis for controller area network (can) with fifo queues priority queues and gateways," *Real-Time Systems*, vol. 49, no. 1, pp. 73–116, 2013.

[22] E. Wang, W. Xu, S. Sastry, S. Liu, and K. Zeng, "Hardware module-based message authentication in intra-vehicle networks," in *Proceedings of the 8th International Conference on Cyber-Physical Systems*. ACM, 2017, pp. 207–216.

[23] M. Igor, "Can protocol controller," *https://opencores.org/acc,view,igorm*, 2009.

[24] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "Canauth-a simple, backward compatible broadcast authentication protocol for can bus," in *ECRYPT Workshop on Lightweight Cryptography*, vol. 2011, 2011.

A Byzantine Fault-Tolerant Key-Value Store for Safety-Critical Distributed Real-Time Systems

Malte Appel[†*], Arpan Gujarati[*], and Björn B. Brandenburg[*]
[*]Max Planck Institute for Software Systems (MPI-SWS), Germany
[†]Saarland University, Germany

I. Motivation

From modern cars to airplanes to industrial plants, many applications that must execute in a timely manner are deployed on distributed systems. In case of safety-critical applications, like the anti-lock braking system of a car, the underlying system must tolerate inadvertent environmentally-induced faults to guarantee user safety. Since such systems often operate at high frequencies, fault-induced failures have to be masked through active replication. Furthermore, before such a system is deployed, it typically has to be analyzed w.r.t. its runtime, safety guarantees, *etc.* This is required for common safety-certification standards such as the DO-178C standard for aviation or the ISO 26262 standard for automotive systems.

To ease the development of such systems, our goal is to design a fault-tolerant middleware on which real-time control applications can be effortlessly replicated, that respects real-time and low-latency requirements, and whose reliability can be analyzed *a priori* for the purpose of safety certification.

II. Model and Assumptions

We assume a distributed system consisting of multiple networked processing elements (PEs) that hosts one or more distributed real-time control applications. An application *fails* if the control loop output, *i.e.*, its final physical actuation, is incorrect due to failures in one of the intermediate stages of the control loop, as explained next.

We consider failures caused by transient soft errors and/or permanent errors due to environmental conditions (such as electromagnetic interference (EMI), thermal effects, *etc.*) and manufacturing defects. In particular, we assume that failures are environmentally induced and not malicious.

The aforementioned failure sources may result in program-visible *Byzantine PE failures*, *i.e.*, PEs may behave arbitrarily, resulting in the delivery of incorrect or inconsistent outputs to other PEs, or in no outputs at all. For example, a PE may end up sending differing messages during a broadcast to its neighbors, say, due to two PEs interpreting the same signal differently owing to a soft error [1], or due to inconsistencies in the underlying network protocol, as in CAN [2].

In contrast to PEs, the network connecting the distributed PEs is assumed to be both *synchronous* and *reliable*, *i.e.*, message delivery times are bounded and message deliveries are ordered. Any network failures are attributed to PE failures, *e.g.*, transient network partitions or delayed message transmissions are considered as PE omission failures.

We assume that the PEs are reliably synchronized using a high-precision clock synchronization protocol, such as [3].

III. Problem Statement

Byzantine failures include both *value failures*, *e.g.*, incorrect computation or inconsistent message deliveries, and *timing failures*, *e.g.*, crashes or message omissions. Value failures may lead to incorrect system behavior, *e.g.*, when wrong inputs are delivered to an actuator, it performs an incorrect action. Crashes of critical components may lead to immediate system failure. Omission failures may lead to a delay or complete lack of reaction. Thus, depending on the extent of value failures and the duration of timing failures, they can have catastrophic consequences in a safety-critical real-time application.

Existing Byzantine fault tolerance (BFT) protocols (see §IV) mitigate the effects of Byzantine failures, but focus on soundness while compromising timeliness. A majority of them were designed primarily for large-scale, predominantly throughput-oriented distributed systems, and thus these protocols (occasionally) exhibit unpredictable, long execution times unsuitable for high-frequency real-time control applications.

This work targets the problem of providing BFT in a predictable, preferably short, time suitable for applications with activation frequencies as high as 10 kHz. In particular, an ideal implementation of a BFT real-time control application and the underlying distributed system must guarantee the following correctness properties despite Byzantine failures.

- *Validity:* If a correct (non-faulty) task of the control application receives or reads a value, that value should have been sent or written by a correct task.
- *Freshness:* If a correct task of the control application receives or reads a value, that value should have been sent or written less than $X\ ms$ ago, where X is application-specific.
- *Agreement:* If a correct task has state S at the end of a control-loop iteration, then all correct replicas of the task have state S at the end of the control-loop iteration.
- *Timely termination:* During each control-loop iteration, the control loop should perform the intended action (the final plant actuation) before the end of that iteration.

Example domains that have such requirements include control systems in ships, avionics, air traffic control, *etc.* [4].

In addition to guaranteeing these correctness properties, any BFT mechanism added to an otherwise certified application should also be *analyzable*. That is, given the peak soft error rates in different system components, it should be possible

Table I: BFT protocols tolerating f failures

Name	Network model	Hosts	Type
BChain-3 [5]	partial synchrony	$3f+1$	chain
Zyzzyva [6]	partial synchrony	$3f+1$	broadcast
PBFT [7]	weak synchrony	$3f+1$	broadcast
Q/U [8]	asynchronous	$5f+1$	quorum
HQ [9]	asynchronous	$3f+1$	quorum
Aliph-Chain [10]	asynchronous	$3f+1$	quorum/chain/broadcast
Ben-Or *et al.* [11]	synchronous	$4f+1$	randomized; quorum
Mostéfaoui *et al.* [12]	asynchronous	$3f+1$	randomized; broadcast
AER [13]	asynchronous	$3f+1$	randomized; quorum
Patra *et al.* [14]	asynchronous	$3f+1$	randomized; broadcast
HoneyBadgerBFT [15]	asynchronous	$3f+1$	randomized; broadcast

to quantify the overall system reliability (*e.g.*, in terms of its mean time to failure) for safety-certification purposes.

IV. Existing BFT Protocols

Since the Byzantine Generals problem was proposed by Lamport *et al.* [16], many BFT protocols have been proposed with the objective of ensuring some (if not all) of the correctness properties listed in §III. Representative protocols for different network models and different design types are summarized in Table I and discussed in brief below.

Broadcast-based BFT protocols always involve all replicas in the agreement process. The client broadcasts its proposal to all replicas [17] or to a designated primary replica that multicasts the proposal to the backups [6, 7]. In contrast, quorum-based BFT protocols require only a representative subset of replicas (the *quorum*) to form an agreement [8, 9]. Both broadcast- and quorum-based BFT protocols achieve relatively low end-to-end latency, but at the cost of large bandwidth consumption. It is thus challenging to incorporate them in distributed real-time systems that often use low-bandwidth networks for cost-efficiency and predictability.

Chain-based protocols arrange replicas in a chain. Clients send their value to the head of the chain and receive a reply only after the message has moved through either a part of [5] or the complete chain [10]. This provides higher throughput, but results in higher latency (compared to broadcast or quorum-based protocols). Depending on the latency requirements and the underlying network, such protocols can be prohibitive for real-time control applications.

The protocols discussed above are *deterministic*, which implies that the number of rounds required for agreement is lower-bounded by $f+1$ when tolerating up to f failures [18]. To improve upon this performance metric, *non-deterministic* or *randomized* BFT protocols were proposed [19, 20], which reduce the number of required rounds, but may violate one of the correctness properties listed in §III with low probability. For example, in the $(1-\epsilon)$-*terminating* protocol by Patra *et al.* [14], a correct task terminates with probability $(1-\epsilon)$, and protocols like AER [13] use *almost-everywhere* Byzantine agreement where agreement is guaranteed for all but $O(\log^{-1} n)$ correct tasks. Protocols such as the one proposed by Patra *et al.* [14] are favorable if ϵ is reasonably low and does not significantly affect the overall system reliability.

None of the protocols discussed above, however, guarantees freshness and timely termination (as stated in §III). For example, in a hard real-time application, if a value satisfying validity and agreement is delivered to a task after its deadline, it has nonetheless zero utility. It is thus better to receive a correct value (or maybe a value that is correct with high probability) on time, or to not receive it at all. To realize this, the BFT protocol must be aware of the timeliness requirements of all values that it handles. Similarly, to ensure freshness, it must be aware of the application-specific lifetime of each value.

V. Proposed Solution

We propose to build a BFT key-value store (KVS) that will act as a middleware for distributed real-time applications, that satisfies the correctness properties listed in §III, and that is analyzable. We first give an overview of the system design, and then explain the rationale behind our design.

Overview. The KVS provides a `write(k,v,t)` API for *publishing* a value `v` for key `k` at time `t` and a `read(k,t)` API for reading the latest published value `v` (that is published not earlier than `t`) for key `k` (see Listing 1 for an example). The time parameter `t` is application-specific and inspired by the logical execution time paradigm [21, 22]. For a write, it determines the absolute time at which the write should be published, *i.e.*, made visible to subsequent read requests for key `k`, and for a read, it determines the freshness requirement of the returned value, *i.e.*, values published earlier than time `t` are not returned. The middleware underlying the read and write APIs consists of one local data store per PE, which coordinate using a BFT protocol to tolerate Byzantine failures.

Freshness and timely termination. The time parameter `t` in the read and write APIs allows the programmer to convey application-specific freshness and timeliness requirements to the KVS. The agreement protocol disseminates any written value `v` by time `t` to enable timely termination of the control loop, where `t` must be sufficiently far in the future to allow the execution of the agreement protocol. For a read request, the value is served by the local data store. If no fresh value exists locally, a valid value is requested from other data stores with a consensus protocol. If still no fresh value exists (*i.e.*, there is no fresh value in the system), the read returns a default value. Thus, by using the time parameter `t`, the KVS guarantees both freshness and timely termination for the control application.

Validity. The agreement protocol guarantees that a valid value is stored in every local data store, and read correctly by the client, if the value or the read operation is not affected by failures on the client PE. Furthermore, to reduce the likelihood of invalid reads due to failures on the client side (say, when the published value in the local data store is corrupted just before being read), the local data store computes and stores a checksum for each published value. With this, the KVS has the option of invoking the consensus protocol to retrieve the value from other local data stores in case of a checksum mismatch.

Agreement. The agreement property requires that replicas have a uniform state at the end of every control-loop iteration.

Listing 1 Example PID controller programmed over KVS

```
1: procedure PERIODICTASKACTIVATION
2:     freshness ← timeOfLastActivation()
3:     currentPos ← KVS.read("sensorDataKey", freshness)
4:     error ← KVS.read("targetPosKey", freshness) − currentPos
5:     integral ← KVS.read("integralKey", freshness) + error
6:     derivate ← error − KVS.read("errorKey", freshness)
7:     newPos ← (KVS.read("kpKey", freshness) ∗ error) +
                (KVS.read("kiKey", freshness) ∗ integral) +
                (KVS.read("kdKey", freshness) ∗ derivative)
8:     time ← timeOfNextActivation()
9:     KVS.write("errorKey", error, time)
10:    KVS.write("integralKey", integral, time)
11:    KVS.write("controlValueKey", newPos, time)
```

The KVS guarantees this by requiring that any stateful values used by the application are written to and read from the KVS (as illustrated in Listing 1). Multiple writes for the same key that should be published at the same time are resolved transparently by the KVS middleware. Applications can specify a key-level policy at configuration time, such as majority voting, averaging, median, *etc.*, that decides how the KVS processes differing values (say, noisy, but correct, values published by replicated sensor tasks). As a key benefit, this approach makes replication effortless for the application developer, since it suffices to instantiate the application (*e.g.*, the PID controller code in Listing 1) on an arbitrary number of hosts for replication, without any changes to the code. Furthermore, since all application state is persisted in the KVS, crashed applications or PEs can be trivially restarted.

Analyzability. The proposed design reduces the application failure domain to the KVS, *i.e.*, failures are attributed to the KVS implementation and not to the application code. It abstracts away any BFT mechanisms from the programmer and decouples it from the application logic, which makes it easier to reason about and formally model the KVS. In particular, a layered design consisting of a separate application layer, KVS layer, clock synchronization layer, networking layer, *etc.*, enables independent analysis of the worst-case reliability bounds for each layer while assuming that other layers are reliable, and then composition of these bounds to yield an overall system reliability bound.

Coordination protocol. The process of choosing and evaluating an appropriate BFT protocol for the coordination of data store replicas is still in progress. Since we focus on control applications, we concentrate on protocol latencies rather than their throughput. For fail-operational systems, protocols that completely degrade in performance as soon as failures occur are unacceptable. We plan to avoid using protocols such as Zyzzyva [6] that improve performance through speculative execution at the cost of unpredictable revert actions. However, a predictable version of such protocols, with manageable latencies, might be interesting. Clement *et al.*'s [23] work on *robust BFT*, which favors an equal performance in both failure and non-failure cases over optimizations benefiting only the failure-free scenario, is particularly interesting in this regard. Some of the non-deterministic protocols achieve much lower latencies and thus seem appealing, but they introduce a small risk of violating the agreement property. If this probability is reasonably low, randomization might be the favorable solution, but for the moment, we leave the possible incorporation of non-deterministic protocols as future work.

Next steps. Once the KVS is designed and implemented, we will conduct rigorous fault-injection experiments through the injection of bit flips in arbitrary memory locations (including the OS kernel), since environmental EMI sources are not restricted to the specific parts of the memory used by the KVS process. Finally, we aim to analyze the reliability of the BFT KVS to derive a safe bound on the mean time to failure of applications hosted on this platform, given bounds on the peak rates of soft and permanent errors in all PEs.

REFERENCES

[1] K. Driscoll, B. Hall, H. Sivencrona, and P. Zumsteg, "Byzantine fault tolerance, from theory to reality," in *SafeComp*, 2003.

[2] G. M. Lima and A. Burns, "A consensus protocol for CAN-based systems," in *RTSS*, 2003.

[3] "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–300, July 2008.

[4] D. Locke, *Applications and System Characteristics*. Boston, MA: Springer US, 2002, pp. 17–26.

[5] S. Duan, H. Meling, S. Peisert, and H. Zhang, "BChain: Byzantine replication with high throughput and embedded reconfiguration." in *OPODIS*, 2014.

[6] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: Speculative Byzantine fault tolerance," in *SOSP*, 2007.

[7] M. Castro, B. Liskov *et al.*, "Practical Byzantine fault tolerance," in *OSDI*, 1999.

[8] M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-scalable Byzantine fault-tolerant services," in *SOSP*, 2005.

[9] J. Cowling, D. Myers, B. Liskov, R. Rodrigues, and L. Shrira, "HQ replication: A hybrid quorum protocol for Byzantine fault tolerance," in *OSDI*, 2006.

[10] R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 BFT protocols," in *EuroSys*, 2010.

[11] M. Ben-Or, E. Pavlov, and V. Vaikuntanathan, "Byzantine agreement in the full-information model in $O(\log n)$ rounds," in *STOC*, 2006.

[12] A. Mostéfaoui, H. Moumen, and M. Raynal, "Signature-free asynchronous binary Byzantine consensus with $t < n/3, O(n^2)$ messages, and $O(1)$ expected time," *JACM*, vol. 62, no. 4, p. 31, 2015.

[13] N. Braud-Santoni, R. Guerraoui, and F. Huc, "Fast Byzantine agreement," in *PODC*, 2013.

[14] A. Patra, A. Choudhury, and C. P. Rangan, "Asynchronous Byzantine agreement with optimal resilience," *Distributed Computing*, vol. 27, no. 2, pp. 111–146, 2014.

[15] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," in *CCS*, 2016.

[16] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM TOPLAS*, vol. 4, no. 3, pp. 382–401, 1982.

[17] G. Bracha, "Asynchronous Byzantine agreement protocols," *Information and Computation*, vol. 75, no. 2, pp. 130–143, 1987.

[18] M. J. Fischer and N. A. Lynch, "A lower bound for the time to assure interactive consistency," *Information processing letters*, vol. 14, no. 4, pp. 183–186, 1982.

[19] M. O. Rabin, "Randomized Byzantine generals," in *FOCS*, 1983.

[20] M. Ben-Or, "Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols," in *PODC*, 1983.

[21] T. A. Henzinger, B. Horowitz, and C. M. Kirsch, "Embedded control systems development with Giotto," in *LCTES*, 2001.

[22] C. M. Kirsch and A. Sokolova, "The logical execution time paradigm," in *Advances in Real-Time Systems*. Springer, 2012, pp. 103–120.

[23] A. Clement, E. L. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, "Making Byzantine fault tolerant systems tolerate Byzantine faults." in *NSDI*, 2009.

Lower-Bounding the MTTF for Systems with (m, k) Constraints and IID Iteration Failure Probabilities

Arpan Gujarati, Mitra Nasri, and Björn B. Brandenburg
Max Planck Institute for Software Systems (MPI-SWS), Germany
{arpanbg, mitra, bbb}@mpi-sws.org

***Abstract*—We derive a sound lower bound on the *mean time to failure* of periodic systems with (m, k) constraints. We assume that upper bounds on the failure probabilities of each system iteration, *e.g.*, a job or a runtime activation of a periodic task, or a single actuation cycle of a control loop, are known and that they satisfy the IID assumption. Our analysis leverages prior work on the well-studied *a-within-consecutive-b-out-of-c:F* system model.**

I. Introduction

For safety certification, the reliability of a safety-critical system must be carefully analyzed before deployment. Typically, this is done using generic analyses such as *fault tree analysis* (FTA) [18] and *failure mode and effects analysis* (FMEA) [19], or domain-specific analyses (*e.g.* [5, 16]). The analyzed reliability is reported using metrics such as the *mean time to failure* (MTTF) or the *failures-in-time* (FIT) [20], or sometimes simply using a failure probability.

Many periodic safety-critical systems are subject to (m, k) constraints, *i.e.*, at least m iterations out of any k consecutive iterations must be correct [10]. For example, many real-time systems can tolerate a few deadline misses [11] and well-designed, robust control applications remain functional despite a few missed or incorrect actuations [4], thanks to the underlying physics. However, accurate and sound reliability analysis of systems with (m, k) constraints is still an open problem.

Prior works focus on the reliability analysis of individual iterations, *e.g.*, analyzing the probability of a deadline miss or a faulty message transmission in any iteration, as in [3, 8, 16], and then extrapolate overall reliability guarantees for the system, *e.g.*, by analyzing the probability of *no* deadline misses or *no* faulty message transmissions. For systems with (m, k) constraints, this zero-tolerance hard real-time approach towards reliability analysis results in excessively pessimistic reliability bounds, and consequently in cost-inefficient designs that under-utilize system resources.

In this work, we propose a new reliability analysis for systems with (m, k) constraints. We assume that (an upper bound on) the failure probability for each iteration is known in advance and that these iteration failure probabilities satisfy the *IID assumption* (*i.e.*, they are identical for each iteration and are independent of other failed iterations). For instance, this assumption is satisfied by Broster et al.'s analysis [3] of the probability of a timely transmission of a Controller Area Network (CAN) message, or our prior work [8] on the reliability analysis of replicated CAN messages.

We then leverage existing results on the reliability analysis of the well-studied *a-within-consecutive-b-out-of-c:F* system model [12] to derive the probability that the system violates its (m, k) constraint for the first time during the n^{th} iteration. Finally, using these probabilities, we characterize the system reliability in terms of a lower bound on its MTTF.

This work is part of an ongoing project on reliability analysis of CAN-based *networked control systems* (NCS) with replicated tasks that are characterized using (m, k) constraints. We are currently developing an analysis to derive IID failure probabilities for each iteration of a control loop in the NCS, which, together with the analysis presented in this paper, will help quantify the overall reliability of the NCS.

II. System Model

Let P_F (for **F**ailure) denote an upper bound on the probability that a single iteration of the system fails, and let $P_S = 1 - P_F$ (for **S**uccess). We assume that $0 < P_F < 1$ and that P_F satisfies the IID assumption. Since reliability analyses, such as the ones in [3, 8], often analyze the worst-case scenario for *any* iteration, the resulting iteration failure probabilities satisfy this assumption.

Let S denote the system being analyzed and T denote its activation period. S *fails* as soon as it violates the (m, k) constraint. That is, it fails during the n^{th} iteration if the n^{th} iteration fails and it is the $(k-m+1)^{\text{th}}$ failed iteration in the last k iterations (thus violating the (m, k) constraint), and if the (m, k) constraint has not been violated before. This implies that S cannot fail during the first $k - m$ iterations.

The MTTF of a system is defined as its *expected lifetime*. That is, for a system S with an (m, k) constraint, MTTF is the average time that it takes for S to violate its (m, k) constraint. It can be computed using the well-known definition $MTTF = \int_0^\infty t \times f(t)\, dt$ [12, §2.2], where $f(t)$ denotes the *probability density function* (*p.d.f.*) of S, *i.e.* the probability that S violates its (m, k) constraint for the first time at time instant t. The objective of this paper is to derive a lower bound on the MTTF of system S, given its iteration failure probability P_F.

III. Overview

The proposed analysis consists of four steps. In Step 1, we formulate the probability that S violates its (m, k) constraint for the first time in its n^{th} iteration. In Step 2, we define a lower bound on this probability, since obtaining an exact value is computationally hard. In Step 3, we lower-bound the *p.d.f.*

of S, which is required for computing its MTTF. Finally, in Step 4, we derive a lower bound on the MTTF using the lower bound on the *p.d.f.* Steps 1-4 are explained in detail below.

Step 1. S violates its (m, k) constraint for the first time in its n^{th} iteration if the following conditions hold:

E_1: The n^{th} iteration must fail.

E_2: Exactly $k-m$ iterations must fail out of the $k-1$ iterations between the $(n-k+1)^{\text{th}}$ and the $(n-1)^{\text{th}}$ iteration.

E_3: Fewer than $k-m+1$ iterations fail out of any k consecutive iterations, among the first $n-1$ iterations.

Thus, given E_1, E_2, and E_3, the probability that S violates its (m, k) constraint for the first time in its n^{th} iteration is lower-bounded by $P(E_1) \times P(E_2) \times P(E_3)$.

Step 2. From §II, $P(E_1) = P_F$. Summing over all possible combinations of $k-m$ iteration failures in $k-1$ consecutive iterations, $P(E_2) = \binom{k-1}{k-m} P_F^{(k-m)} P_S^{(m-1)}$. But obtaining the exact value of $P(E_3)$ is computationally challenging, since it requires evaluating all possible combinations of failed and successful iterations among the first $n-1$ iterations.

Thus, we approximate $P(E_3)$ using the well-studied a-within-consecutive-b-out-of-c:F system [12, §11.4], which consists of c ($c \geq a$) linearly ordered components, and which fails iff at least a ($a \leq b$) components fail among any b consecutive components. That is, in terms of the (m, k) model, an a-within-consecutive-b-out-of-c:F system fails if it violates the $(b-a+1, b)$ constraint. We refer to this system model as an a/Con/b/c:F system, for brevity. We model E_3 as an a/Con/b/c:F system where $a = k-m+1$, $b = k$, and $c = n-1$, and lower-bound $P(E_3)$ using a reliability lower bound $R_{LB}(a, b, c)$ of this system, which we will introduce in §IV.

From the above definitions of $P(E_1)$, $P(E_3)$, $P(E_3)$, if $n > k-m$, a lower bound on the probability that S violates its (m, k) constraint for the first time during its n^{th} iteration is:

$$g_{LB}(n) = \binom{k-1}{k-m} P_F^{(k-m+1)} P_S^{(m-1)} \times R_{LB}(k-m+1,\, k,\, n-1). \quad (1)$$

Step 3. Recall from §II that T denotes the period of system S. Accordingly, any time t such that $(n-1)T < t \leq nT$ corresponds to the execution of the n^{th} iteration of S. Thus, the sum of the *p.d.f.* of system S at all time instants in $((n-1)T,\, nT]$ is lower bounded by $g_{LB}(n)$, *i.e.*,

$$\int_{(n-1)T}^{nT} f(t) \geq g_{LB}(n). \quad (2)$$

In addition, $f(t) = 0$, *i.e.*, the system is reliable, for all $t \leq (k-m)T$ since by definition of the (m, k) constraint, the system can fail only after $k-m$ iterations.

Step 4. A lower bound on the MTTF is derived using the expression $MTTF = \int_0^\infty t \times f(t)\, dt$ and Eq. 2. However, since $g_{LB}(n)$ in Eq. 2 is defined in terms of $R_{LB}(k-m+1, k, n-1)$, a recursive expression with complex definitions of its subproblems (see §IV), symbolic integration to lower bound the MTTF is infeasible, even with tools such as Mathematica [1].

Instead, we propose a numeric, but sound approach to lower-bound the MTTF that relies on computing the value of $g_{LB}(n)$ at finitely many data points (see §V).

IV. The a/Con/b/c:F System Model

We assume that the system consists of IID components.[1] We first define a lower bound on the reliability of an *a/Con/b/c:F* system, *i.e.*, a lower bound on the probability that the system does not fail, using prior results and then prove that this lower bound decreases with increasing c if certain conditions hold.

A. Reliability of an a/Con/b/c:F system

Let $R(a, b, c)$ denote the exact reliability of an a/Con/b/c:F system. A brute-force approach to compute $R(a, b, c)$ requires enumerating all combinations of failed/not-failed components, selecting the combinations for which the system does not fail, and then adding the event probabilities for these reliable combinations. However, since the number of combinations that need to be checked are exponential in c, the brute-force approach is infeasible, particularly since c can easily exceed 10^{50} (see §VI for details).

We instead use the results of Sfakianakis et al. [17] to derive a lower bound on $R(a, b, c)$, denoted $R_{LB}(a, b, c)$, for large values of c. Sfakianakis et al.'s analysis breaks the problem into smaller subproblems for which exact analyses are available. Their analysis, as well as the exact analyses for different types of subproblems, are explained in detail in [12].

Table I summarizes the relevant results in [12] for different values of a, b, and c. Cases 1 and 2 are trivial: if $a = 0$, the system is always unreliable, and if $a = 1$, the system is reliable only if none of the c components fail. Cases 3, 5, 6, and 7 correspond to special cases where c is small (less than or equal to either $2b$ or $4b$) and for which exact reliabilities can be computed. Cases 4 and 8 correspond to large, unbounded values of c and are resolved using Sfakianakis et al.'s recursive analysis. A generic lower bound $R_{LB}(a, b, c)$ is defined by combining all of these cases.

B. $R_{LB}(a, b, c)$ decreases with increasing c

The MTTF analysis in §V depends on the property that $R_{LB}(a, b, c)$ *decreases with increasing* c. This property trivially holds for cases $a = 0$ and $a = 1$, as seen from the definitions of $R_1(a, b, c)$ and $R_2(a, b, c)$ in Table I. However, proving the property for cases $a = 2$ and $a > 2$ is non-trivial. We discuss the more general case $a > 2$ below. Case $a = 2$ is not discussed due to space constraints, but is handled similarly.

Notice that case $a > 2$ corresponds to multiple cases (5-8) in Table I. In fact, because of the recursive definitions for some of these cases, case $a > 2$ actually depends on the remaining cases as well, which makes it hard to prove that $R_{LB}(a, b, c)$ decreases with increasing c. Instead, we prove a weaker property: we show that *if* $R_{LB}(a, b, c)$ decreases with increasing c for small values of c (*i.e.*, for $c \leq 2b$), *then*

[1] We use the terms *iteration* and *component* interchangeably. We use the term *component* in this section since it is consistent with the terminology used in the existing literature on the a/Con/b/c:F model.

#	Case	Definition	Type	Source
1	$a = 0$	$R_1(a,b,c) = 0$	Exact	–
2	$a = 1$	$R_2(a,b,c) = P_S^c$	Exact	–
3	$a = 2 \wedge c \leq 4b$	$R_3(a,b,c) = \sum_{i=0}^{\lfloor \frac{c+b-1}{b} \rfloor} \binom{c-(i-1)(b-1)}{i} P_F^i P_S^{c-i}$	Exact	[12, §11.4.1] (Eqs. 11.9 and 11.10)
4	$a = 2 \wedge c > 4b$	$R_4(a,b,c) = R_3(a,b,b+t-1)(R_3(a,b,b+3))^u$ where $t = (c-b+1) \bmod 4$ and $u = \lfloor \frac{c-b+1}{4} \rfloor$	LB	[12, §11.4.1] (Eq. 11.16)
5	$a > 2 \wedge c \leq 2b \wedge a = b$	$R_5(a,b,c) = \begin{cases} 1 & 0 \leq c < a \\ 1 - P_F^a - (c-k)P_F^a P_S & a \leq c \leq 2a \end{cases}$	Exact	[12, §9.1.1] (Eqs. 9.2, 9.9, and 9.20)
6	$a > 2 \wedge c \leq 2b \wedge a \neq b \wedge c \leq b$	$R_6(a,b,c) = \sum_{i=c-a+1}^{c} \binom{c}{i} P_S^i P_F^{c-i}$	Exact	[12, §7.1.1] (Eq. 7.2)
7	$a > 2 \wedge c \leq 2b \wedge a \neq b \wedge c > b$	$R_7(a,b,c) = \sum_{i=0}^{a-1} \binom{b-s}{i} P_F^i P_S^{b-s-i} M(a',s,2s)$ where $s = c-b$ and $a' = a-i$, and $M(a',s,2s) = \begin{cases} 1 & a' > s \\ R_2(a',s,2s) & a' = 1 \\ R_3(a',s,2s) & a' = 2 \\ R_5(a',s,2s) & a' > 2 \wedge a' = s \\ R_7(a',s,2s) & a' > 2 \wedge a' \neq s \end{cases}$	Exact	[12, §11.4.1] (Eq. 11.14)
8	$a > 2 \wedge c > 2b$	$R_8(a,b,c) = R_\phi(a,b,b+t-1)(R_\phi(a,b,b+3))^u$ where $t = (c-b+1) \bmod 4$ and $u = \lfloor \frac{c-b+1}{4} \rfloor$, and $R_\phi(a,b,c) = \begin{cases} R_5(a,b,c) & a = b \\ R_6(a,b,c) & a \neq b \wedge a \leq b \\ R_7(a,b,c) & a \neq b \wedge a > b \end{cases}$	LB	[12, §11.4.1] (Eq. 11.16)

TABLE I. **Type** indicates whether the reliability definition for that respective case is an exact value or a lower bound.

$R_{LB}(a,b,c)$ also decreases with increasing c for larger values of c (*i.e.*, for $c > 2b$). Since b is typically relatively small, *i.e.*, $b = k$ (recall Step 2 from §III), the *if* condition can be easily checked for specific values of a, b, c and p through exhaustive enumeration.

Lemma 1. *For $c \geq a$ and $a > 2$, if $R_{LB}(a,b,c)$ is monotonically decreasing for $c \in \{a, \ldots, 2b+1\}$, then $R_{LB}(a,b,c)$ is also monotonically decreasing for $c \geq 2b+1$, i.e.,*

$$\begin{aligned} &\textit{if} \quad \forall c \leq 2b: \; R_{LB}(a,b,c) \geq R_{LB}(a,b,c+1), \\ &\textit{then} \quad \forall c > 2b: \; R_{LB}(a,b,c) \geq R_{LB}(a,b,c+1). \end{aligned} \tag{3}$$

Proof. The proof has three steps. In the first step, we simplify the *if* condition in Eq. 3 for the case $c < 2b$; and then in the second and the third step, we use it to prove the *then* condition in Eq. 3 for cases $(c-b+1) \bmod 4 = 3$ and $(c-b+1) \bmod 4 < 3$, respectively.

Step 1 [$c < 2b$]. Since $a > 2$ and $c < 2b$ imply that $c+1 \leq 2b$, the *if* condition in Eq. 3 can be simplified using the definition of $R_\phi(a,b,c)$ from Case 8 in Table I as follows.

$$\begin{aligned} R_{LB}(a,b,c) &\geq R_{LB}(a,b,c+1) \\ &\equiv R_\phi(a,b,c) \geq R_\phi(a,b,c+1). \end{aligned}$$

Step 2 [$c > 2b$ and $(c-b+1) \bmod 4 = 3$].

$$\frac{R_{LB}(a,b,c)}{R_{LB}(a,b,c+1)}$$

{since $a > 2$ and $c > 2b$, both terms $R_{LB}(a,b,c)$ and $R_{LB}(a,b,c+1)$ are resolved using case 8 in Table I; thus, from $R_8(a,b,c)$'s definition, and letting $x = c-b+1$}

$$= \frac{R_\phi(a,b,b+(x \bmod 4)-1)(R_\phi(a,b,b+3))^{\lfloor \frac{x}{4} \rfloor}}{R_\phi(a,b,b+((x+1) \bmod 4)-1)(R_\phi(a,b,b+3))^{\lfloor \frac{x+1}{4} \rfloor}}$$

{since $x \bmod 4 = 3$ implies $(x+1) \bmod 4 = 0$}

$$= \frac{R_\phi(a,b,b+2)(R_\phi(a,b,b+3))^{\lfloor \frac{x}{4} \rfloor}}{R_\phi(a,b,b-1)(R_\phi(a,b,b+3))^{\lfloor \frac{x+1}{4} \rfloor}}$$

{since $x \bmod 4 = 3$ implies $\lfloor \frac{x+1}{4} \rfloor = \lfloor \frac{x}{4} \rfloor + 1$}

$$= \frac{R_\phi(a,b,b+2)(R_\phi(a,b,b+3))^{\lfloor \frac{x}{4} \rfloor}}{R_\phi(a,b,b-1)(R_\phi(a,b,b+3))^{\lfloor \frac{x}{4} \rfloor + 1}}$$

{dividing numerator and denominator by $(R_\phi(a,b,b+3))^{\lfloor \frac{x}{4} \rfloor}$}

$$= \frac{R_\phi(a,b,b+2)}{R_\phi(a,b,b-1)R(a,b,b+3)}$$

{since $R_\phi(a,b,b-1) \leq 1$ (being a probability)}

$$\geq \frac{R_\phi(a,b,b+2)}{R_\phi(a,b,b+3)}$$

$\{$since $2 < a \leq b \implies 2 < b \implies b+2 < 2b$; from the *if* condition in Eq. 3 and from Step 1, $R_\phi(a,b,b+2) \geq R_\phi(a,b,b+3)\}$

$\geq 1.$

Step 3 [$c > 2b$ and $(c-b+1) \bmod 4 < 3$].

$$\frac{R_{LB}(a,b,c)}{R_{LB}(a,b,c+1)}$$

$\{$since $a > 2$ and $c > 2b$, both terms $R_{LB}(a,b,c)$ and $R_{LB}(a,b,c+1)$ are resolved using case 8 in Table I; thus, from $R_8(a,b,c)$'s definition, and letting $x = c-b+1\}$

$$= \frac{R_\phi(a,b,b+(x \bmod 4)-1)(R_\phi(a,b,b+3))^{\lfloor \frac{x}{4} \rfloor}}{R_\phi(a,b,b+((x+1) \bmod 4)-1)(R_\phi(a,b,b+3))^{\lfloor \frac{x+1}{4} \rfloor}}$$

$\{$since $x \bmod 4 < 3$ implies $\lfloor \frac{x+1}{4} \rfloor = \lfloor \frac{x}{4} \rfloor\}$

$$= \frac{R_\phi(a,b,b+(x \bmod 4)-1)(R_\phi(a,b,b+3))^{\lfloor \frac{x}{4} \rfloor}}{R_\phi(a,b,b+((x+1) \bmod 4)-1)(R_\phi(a,b,b+3))^{\lfloor \frac{x}{4} \rfloor}}$$

$\{$dividing numerator and denominator by $(R(a,b,b+3))^{\lfloor \frac{x}{4} \rfloor}\}$

$$= \frac{R_\phi(a,b,b+(x \bmod 4)-1)}{R_\phi(a,b,b+((x+1) \bmod 4)-1)}$$

$\{$since $x \bmod 4 < 3$ implies $(x+1) \bmod 4 = 1 + x \bmod 4\}$

$$= \frac{R_\phi(a,b,b+(x \bmod 4)-1)}{R_\phi(a,b,b+(x \bmod 4))}$$

$\{$since $x \bmod 4 < 3 \implies b+(x \bmod 4)-1 < b+2$, and since $2 < k \leq b \implies 2 < b \implies b+2 < 2b$, we have $b+(x \bmod 4)-1 < 2b$; thus, from the *if* condition in Eq. 3 and from Step 1, $R_\phi(a,b,b+(x \bmod 4)-1) \geq R_\phi(a,b,b+(x \bmod 4))\}$

$\geq 1.$ $\square$

In the next section, while describing the proposed MTTF analysis, we assume that $R_{LB}(a,b,c)$ *decreases with increasing* c. When applying the proposed analysis (*e.g.*, in §VI), for every use of $R_{LB}(a,b,c)$, we check that the *if* condition in Lemma 1 holds in order to justify this assumption.

V. MTTF Analysis

Recall the definition of $g_{LB}(n)$ from §III (Eq. 1). Since $R_{LB}(a,b,c)$ decreases with increasing c and since $g_{LB}(n)$ is defined in terms of $R_{LB}(k-m+1,k,n-1)$, $g_{LB}(n)$ also decreases with increasing n.

Assume that the value of the function $g_{LB}(n)$ is known (*i.e.*, computed) at finitely many data points d_0, d_1, d_2, …, d_D, such that each $d_i \in \mathbb{N}$ and $k-m+1 = d_0 < d_1 < d_2 < \ldots < d_D$. Using time instants d_0T, d_1T, d_2T, …, d_DT corresponding to the start time of iterations d_0, d_1, d_2, …, d_D, and the property that $g_{LB}(n)$ is decreasing with increasing n, we derive a lower bound on the MTTF as follows.

Lemma 2.

$$MTTF \geq \sum_{i=0}^{D-1} \Big(d_iT \times g_{LB}(d_{i+1}) \times (d_{i+1} - d_i) \Big) \tag{4}$$

Proof.

$$MTTF = \int_0^\infty t \times f(t)\, dt$$

$\{$splitting $(0,\infty)$ into a finite number of subintervals $(0, d_0T]$, $(d_0T, d_1T]$, …, $(d_{D-1}T, d_DT]$, and (d_DT, ∞); and dropping the integrals for subintervals $(0, d_0T]$ and (d_DT, ∞) since we are interested in lower-bounding the MTTF$\}$

$$\geq \sum_{i=0}^{D-1} \int_{d_iT}^{d_{i+1}T} t \times f(t)\, dt$$

$\{$since for all $t \in (d_iT, d_{i+1}T]$, $t \geq d_iT\}$

$$\geq \sum_{i=0}^{D-1} \left(d_iT \times \int_{d_iT}^{d_{i+1}T} f(t)\, dt \right)$$

$\{$splitting each subinterval $(d_iT, d_{i+1}T]$ into multiple subintervals $(d_iT, (d_i+1)T]$, $((d_i+1)T, (d_i+2)T]$, …, $((d_{i+1}-1)T, (d_{i+1})T]$, each of length $T\}$

$$= \sum_{i=0}^{D-1} \left(d_iT \times \left(\sum_{j=0}^{d_{i+1}-d_i-1} \int_{(d_i+j)T}^{(d_i+j+1)T} f(t)\, dt \right) \right)$$

$\{$since $\int_{(d_i+j)T}^{(d_i+j+1)T} f(t)\, dt \geq g_{LB}(d_i+j+1)$ (from Eq. 2)$\}$

$$\geq \sum_{i=0}^{D-1} \left(d_iT \times \left(\sum_{j=0}^{d_{i+1}-d_i-1} g_{LB}(d_i+j+1) \right) \right)$$

$\{$since $g_{LB}(n)$ is decreasing with increasing n, for each integer j in the interval $[0,\ d_{i+1}-d_i-1]$, $g_{LB}(d_i+j+1) \geq g_{LB}(d_i+d_{i+1}-d_i-1+1) = g_{LB}(d_{i+1})\}$

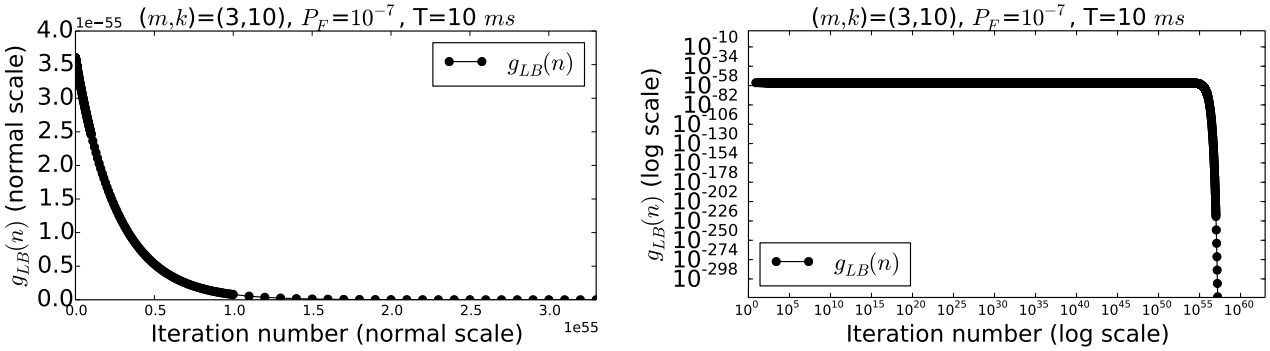
$$\geq \sum_{i=0}^{D-1} \left(d_iT \times \left(\sum_{j=0}^{d_{i+1}-d_i-1} g_{LB}(d_{i+1}) \right) \right)$$

$\{$simplifying the innermost summation$\}$

$$= \sum_{i=0}^{D-1} \Big(d_iT \times g_{LB}(d_{i+1}) \times (d_{i+1} - d_i) \Big) \quad \square$$

Let $MTTF_{LB}$ denote the lower bound derived in Lemma 2. If $D \ll d_D$, $MTTF_{LB}$ can be computed quickly. If the individual data points d_0, d_1, d_2, …, d_D are appropriately chosen, then the computed $MTTF_{LB}$ is sufficiently close to the exact MTTF. We revisit the choice of data poitns in §VI.

Next, we discuss how to estimate the MTTF using simulations. We use a *biased coin toss* experiment, where the biased coin comes up with heads with probability P_S, and tails with probability $P_F = 1-P_S$. *Tails* denotes that the system iteration is incorrect, and *heads* denotes that the system iteration is correct. In each trial, the coin toss is repeated until tails is



(a) $MTTF_{LB} = 2.34 \times 10^{55}\, ms$

(b) Same as (a), but with log-scale axes

Fig. 1: **(a)** $g_{LB}(t)$ for $m = 3$, $k = 10$, and $P_F = 10^{-7}$, where $D = 5050$ and $d_D = 9.90 \times 10^{57}$. **(b)** $g_{LB}(t)$ for the same parameters, except that both the x- and y-axes are log-scale.

encountered $k-m+1$ times among the last k consecutive coin tosses. If Ω denotes the total number of trials and ω_i denotes the number of coin tosses during the i^{th} trial, averaging ω_i over Ω trials, *i.e.*, $\hat{\omega} = (\sum_{i=1}^{\Omega} \omega_i)/\Omega$, gives the expected number of iterations required to violate the (m, k) constraint. Using $\hat{\omega}$, the MTTF is estimated as $MTTF_{sim} = \hat{\omega} \times T$.

$MTTF_{sim}$ cannot be used to safely lower-bound S's reliability because it may over-approximate the reliability. However, it is a useful baseline for evaluating $MTTF_{LB}$. By comparing $MTTF_{LB}$ with $MTTF_{sim}$, we determine how much accuracy we loose by sampling D data points when deriving $MTTF_{LB}$.

VI. Evaluation

The objective of this section is twofold. First, we discuss the method used to choose the data points d_0, d_1, d_2, …, d_D. Second, we present results from a comparison of $MTTF_{LB}$ and $MTTF_{sim}$ for different values of m, k, and P_F.

A. Choosing d_0, d_1, d_2, …, d_D

In Fig. 1(a), we illustrate the function $g_{LB}(n)$ for $m = 3$, $k = 10$, and $P_F = 10^{-7}$. As expected based on §IV-B, $g_{LB}(n)$ decreases with increasing n. Since $MTTF_{LB}$ depends on $g_{LB}(n)$, the key idea is to ensure that points d_0, d_1, d_2, …, d_D are sufficient to trace the shape of function $g_{LB}(n)$, and that the magnitude of $g_{LB}(n)$ is negligible beyond $n = d_D$.

The first point d_0 was set to $(k-m+1)$, as mentioned in §V. To compute the last point d_D, *i.e.*, the point at which $g_{LB}(n)$ becomes negligible, we observed the logarithm of function $g_{LB}(n)$ for $n \in \{1, 10^1, 10^2, 10^3, \ldots\}$. That is, we plotted the function $g_{LB}(n)$ on a logarithmic scale for both the x- and y-axes as in Fig. 1(b), and then determined the time instant at which the curve starts falling rapidly (*e.g.*, $d_D \approx 10^{55}$ in Fig. 1(b)). The intermediate points $d_1, d_2, \ldots, d_{D-1}$ were chosen such that the step size $d_{i+1} - d_i$ between any two consecutive points d_i and d_{i+1} **(i)** is small enough to closely track the function $g_{LB}(n)$, and **(ii)** yet still proportional to the order of magnitude of d_i, to avoid evaluating an exponential number of points. For example, while generating Fig. 1, the step size was 1 for $n \in (10, 100]$ and 10^{52} for $n \in (10^{53}, 10^{54}]$.

B. $MTTF_{LB}$ versus $MTTF_{sim}$

To compare $MTTF_{LB}$ and $MTTF_{sim}$, we chose specific parameters that ensure that the simulation completes within reasonable time. In particular, we avoided parameters for which $MTTF_{LB}$ was very high (typically, configurations with a very small P_F), since the number of rounds in each simulation trial for such parameters would likely be very high as well. In Fig. 3, we illustrate the results for each $P_F \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$, $k \in \{5, 7, 10\}$, and m such that $k - m + 1 = 3$, *i.e.*, $m \in \{3, 5, 8\}$ (respectively).

The simulations were run on a 16-core Intel Xeon E5-2667 v2 machine. For each $P_F \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$, we ran $640,000$, $64,000$, $6,400$, and 640 simulation trials, respectively. To ensure that the number of trials for each simulation was sufficient, we also computed the 99% confidence interval for each $MTTF_{sim}$ (shown as error bars in Fig. 3). In general, the smaller the P_F, the higher was the time to finish a single simulation trial. The average times required to complete a *single* simulation trial and the analytical lower bound $MTTF_{LB}$ for different values of P_F are illustrated in Fig. 2 below. While the former grows exponentially in $log\, P_F$, the latter grows linearly in $log\, P_F$.

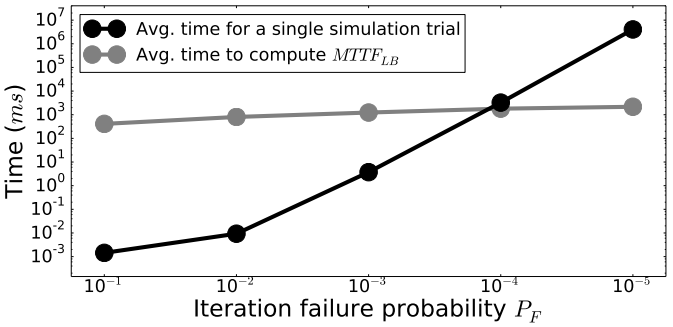


Fig. 2

We draw the following conclusions from the experiments. **(1)** For each configuration, $MTTF_{LB}$ and $MTTF_{sim}$ are roughly of the same order of magnitude, which indicates that the proposed method is sufficiently accurate. Note that while evaluating system reliability, the order of magnitude of the

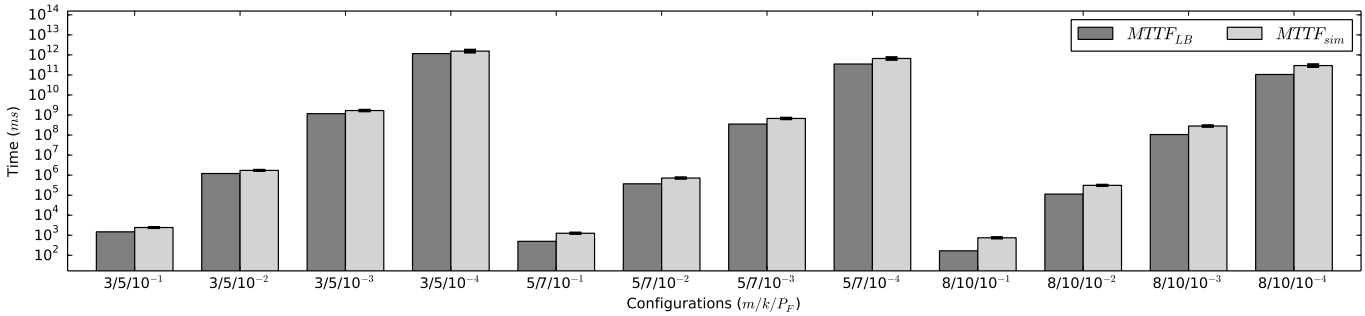


Fig. 3: The period of the system was $T = 10\,ms$. For $MTTF_{sim}$, the 99% confidence intervals are shown as error bars.

reliability metric (in this case, MTTF) is typically more important than minor differences in absolute value. **(2)** $MTTF_{LB}$ is always less than $MTTF_{sim}$. This was expected since we use a lower bound on the *p.d.f.*; $MTTF_{LB}$ is hence also a lower bound on the exact MTTF. **(3)** $MTTF_{LB}$ can be computed significantly faster than $MTTF_{sim}$ for low failure probabilities, and scales to parameters yielding very high MTTFs.

Overall, the experiments show that the proposed analysis provides a fast method to compute a safe bound on the MTTF of system with (m, k) constraints. The analytical results are comparable to the simulation results, but unlike simulation, they are provably sound and scalable.

VII. Conclusion

We have proposed an analysis to derive a safe lower bound on the MTTF of a system with (m, k) constraints. MTTF is one of the standard metrics for measuring system reliability. While closed-form MTTF analyses can be derived for well-known distributions and simple system models [12, Ch. 4], the MTTF analysis for complex systems subject to different types of failures is often difficult, requiring non-trivial techniques (*e.g.*, [9, 13–15]). To the best of our knowledge, for systems with (m, k) constraints, or for the a/Con/b/c:F system model used in this paper, there exists no prior work that safely lower-bounds the system MTTF. Recent works by Eryilmaz et al. [7] and Eryilmaz and Kan [6] derive approximate MTTFs.

As mentioned before, we plan to use the presented MTTF analysis to quantify the overall reliability of a CAN-based NCS with replicated tasks that are characterized using (m, k) constraints, in the presence of environmentally-induced transient failures. We are currently developing an analysis to derive IID failure probabilities for each iteration of a control loop of the NCS. The IID property is guaranteed by the fact that we consider worst-case scenarios w.r.t. the occurrence of faults and interference, and since the iteration failure probability is obtained independently of whether earlier iterations failed, which justifies the IID assumption made in this work.

As future work, to obtain a more general analysis, we will consider systems with multiple (m, k) constraints (*e.g.*, separate constraints for delayed and incorrect messages, or for modeling short-term and long-term behavior) and systems with different flavors of (m, k) constraints (*e.g.*, out of any k *consecutive* iterations, less than m iterations may fail) [2].

References

[1] "Wolfram Mathematica: Modern Technical Computing," available at https://www.wolfram.com/mathematica/.

[2] G. Bernat, A. Burns, and A. Liamosi, "Weakly hard real-time systems," *IEEE transactions on Computers*, vol. 50, no. 4, pp. 308–321, 2001.

[3] I. Broster, A. Burns, and G. Rodriguez-Navas, "Timing analysis of real-time communication under electromagnetic interference," *Real-Time Systems*, vol. 30, no. 1-2, pp. 55–81, 2005.

[4] K.-H. Chen, B. Bönninghoff, J.-J. Chen, and P. Marwedel, "Compensate or ignore? meeting control robustness requirements through adaptive soft-error handling," in *ACM SIGPLAN Notices*, vol. 51, no. 5. ACM, 2016, pp. 82–91.

[5] J. B. Dugan and R. Van Buren, "Reliability evaluation of fly-by-wire computer systems," *Journal of Systems and software*, vol. 25, no. 1, pp. 109–120, 1994.

[6] S. Eryilmaz and C. Kan, "Dynamic reliability evaluation of consecutive-k-within-m-out-of-n:F system," *Communications in Statistics-Simulation and Computation ®*, vol. 40, no. 1, pp. 58–71, 2010.

[7] S. Eryilmaz, C. Kan, and F. Akici, "Consecutive k-within-m-out-of-n:F system with exchangeable components," *Naval Research Logistics (NRL)*, vol. 56, no. 6, pp. 503–510, 2009.

[8] A. Gujarati and B. Brandenburg, "When is CAN the weakest link? a bound on failures-in-time in CAN-based real-time systems," in *RTSS*. IEEE, 2015, pp. 249–260.

[9] P. Gupta and M. Sharma, "Reliability and MTTF evaluation of a two duplex-unit standby system with two types of repair," *Microelectronics Reliability*, vol. 33, no. 3, pp. 291–295, 1993.

[10] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m, k)-firm deadlines," *IEEE transactions on Computers*, vol. 44, no. 12, pp. 1443–1451, 1995.

[11] F. Kluge, M. Neuerburg, and T. Ungerer, "Utility-based scheduling of (m, k)-firm real-time task sets." in *ARCS*, 2015, pp. 201–211.

[12] W. Kuo and M. J. Zuo, *Optimal reliability modeling: principles and applications*. John Wiley & Sons, 2003.

[13] D. Pandey and M. Jacob, "Cost analysis, availability and MTTF of a three state standby complex system under common cause and human failures," *Microelectronics Reliability*, vol. 35, no. 1, pp. 91–95, 1995.

[14] H. Pham, A. Suprasad, and R. Misra, "Reliability and MTTF prediction of k-out-of-n complex systems with components subjected to multiple stages of degradation," *International Journal of Systems Science*, vol. 27, no. 10, pp. 995–1000, 1996.

[15] M. Ram and S. Singh, "Availability, MTTF and cost analysis of complex system under preemptive-repeat repair discipline using gumbel-hougaard family copula," *International Journal of Quality & Reliability Management*, vol. 27, no. 5, pp. 576–595, 2010.

[16] M. Sebastian and R. Ernst, "Reliability analysis of single bus communication with real-time requirements," in *PRDC*. IEEE, 2009, pp. 3–10.

[17] M. Sfakianakis, S. Kounias, and A. Hillaris, "Reliability of a consecutive k-out-of-r-from-n:F system," *IEEE Transactions on Reliability*, vol. 41, no. 3, pp. 442–447, 1992.

[18] M. Stamatelatos, W. Vesely, J. Dugan, J. Fragola, J. Minarick, and J. Railsback, "Fault tree handbook with aerospace applications," 2002.

[19] D. H. Stamatis, *Failure mode and effect analysis: FMEA from theory to execution*. ASQ Quality Press, 2003.

[20] S. Stanley, "MTBF, MTTR, MTTF & FIT explanation of terms," *IMC Networks*, 2011.

SEEDSTRAINER: An Approach to Improve the Hit Ratio of Malicious Candidate URLs

Yasuyuki Tanaka
Institute of Information Security (IISEC)
2-14-1 Tsuruyacho, Kanagawa-ku,
Yokohama, Kanagawa, 211-0835, Japan
Email: dgs155102@iisec.ac.jp

Atsuhiro Goto
Institute of Information Security (IISEC)
2-14-1 Tsuruyacho, Kanagawa-ku,
Yokohama, Kanagawa, 211-0835, Japan
Email: goto@iisec.ac.jp

***Abstract*—Currently, increasing Internet use is plagued by malicious activity; drive-by download attacks have become a particularly serious problem. To counter these malicious sites, blacklisting is widely used as a multilayer defense mechanism in modern Internet security techniques. Blacklisting on network side is especially effective for protecting critical embedded systems or Internet of thing devices because it is not necessary to change the configuration or to use system resources for protection. To make an accurate blacklist, it is necessary to check malicious candidate Uniform Resource Locators (URLs), for example, using client honeypots. Because there are numerous malicious candidate URLs and limited crawling resources, efficient crawling is necessary. In this paper, we propose SEEDSTRAINER, an approach that improves the efficiency of crawling. SEEDSTRAINER creates high-hit-ratio malicious candidate URL lists using open feeds, open intelligence, and machine learning. With SEEDSTRAINER, the hit ratio was improved by 12.5 times. In addition, we reveal the type of information distributed and the update statuses of various open feeds.**

I. INTRODUCTION

Internet usage is becoming increasingly plagued by malicious activity, and drive-by downloads have become an especially serious problem. Grier et al. described the *exploit-as-a-service* model, a malware distribution ecosystem based on drive-by download attacks [1]. In this model, attackers pay for an exploit kit or service to "do the dirty work" of exploiting a victim's browser. For attackers, this is easier than building their own malware distribution network. According to an Internet security threat report from Symantec [2], the number of new unique malicious web domains has decreased, in response to the *exploit-as-a-service* model. It is more difficult to identify these malicious infrastructures and shut them down. A typical drive-by download attack consists of a *landing site*, an *exploit site*, and a *malware download site*. The *landing site* is often set up on a legitimate site, and it redirects the victim to the *exploit site*. On the *exploit site*, attack codes are downloaded and executed; then, malware is downloaded from the *malware download site* and executed. In this way, victims' computers become infected.

To counter these malicious sites, blacklisting is widely used as a multilayer defense mechanism in modern Internet security techniques. Blacklisting on the network side is especially effective for protecting critical embedded systems or Internet of things (IoT) devices because it is not necessary to change the configuration or to use system resources. For example, Microsoft provides the SmartScreen Uniform Resource Locator (URL) Filter [3] to protect users from the sites that are reported to host phishing attacks or distribute malicious software. Google Safe Browsing [4] is installed by default in popular web browsers such as Firefox, Safari, and Chrome. The above products or services are based on blacklisting technology. In general, to make an accurate blacklist, it is necessary to check malicious candidate URLs, for example, using client honeypots. Google Safe Browsing blacklists malicious candidate URLs after checking them using a crawler[5]. Because there are numerous malicious candidate URLs and limited crawling resources, efficient crawling is essential[6]. An additional problem, because we are not Microsoft or Google, is where to collect candidate URLs. In this paper, we propose an approach that improves efficiency of crawling. We create high-hit-ratio malicious candidate URL lists using open feeds, open intelligence, and machine learning. Because our approach is based on open available information, anybody can test it. Our contributions are summarized as follows:

- We developed an approach to improve the hit ratio of malicious candidate URLs. The hit ratio was improved by 12.5 times.
- We observed various open feeds and reveal the types of information they distribute and their update statuses.

II. RELATED WORK

There are two fields of study to counter malicious URLs. One is the detection procedure, which decides whether a given webpage is malicious, and the other is the finding procedure, which efficiently finds malicious candidate URLs on the Internet.

In the detection field, various client honeypots have been proposed. Client honeypots are divided into two categories, high-interaction honeypots and low-interaction honeypots. High-interaction client honeypots use real browsers to access a malicious site and analyze the file downloads and executions [7], [8]. Low-interaction client honeypots use browser emulators [9]. In this experiment, we use a low-interaction client honeypot as our detection method.

In finding field, Stokes proposed WebCop to discover malicious entry of URLs related to malware distribution by tracing

links or linked links of malware distribution URLs [10]. Akiyama proposed a method to efficiently discover an unknown malicious URL by inspecting the structural neighborhood of known malicious URLs [11]. Zhang proposed PoisonAmplifier, a method of discovering unknown malicious URLs that conducts search engine optimization (SEO) by extracting a character string unique to the SEO and performing a keyword search on a known malicious URL performing that SEO [12]. Invernizzi proposed EvilSeed to discover unknown malicious URLs using hyperlink, URL structure, SEO, domain registration information, and Domain Name System (DNS) query information [13]. Ma proposed a method to classify unknown URLs as benign or malicious binary using supervised machine learning [14]. Our study is different from these related studies because we focus on improving the hit ratio of malicious candidate URLs.

III. System overview

To improve the hit ratio of malicious candidate URLs, we use open feeds, open intelligence, and machine learning. Figure 1 shows the overview of our system. First, we collect the latest malicious URLs/domains from various open feeds. Second, we expand the latest malicious URLs by querying malicious domains with open intelligence. Finally, we select the truly malicious candidate URLs using machine learning. In general, malicious candidate URLs are called *seeds*.

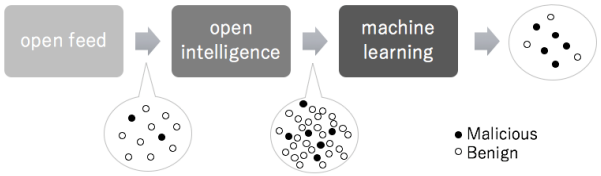


Fig. 1. Overview of SeedStrainer

A. Open feeds and open intelligence

To collect malicious candidate domains/URLs, we use publicly available feeds. Open feeds publish malicious URLs on their webpages. Some open feeds manually check URLs, whereas others automatically check the URLs. For example, urlQuery[21] is an automatic service that receives URLs submitted by contributors and publishes the results of visiting those URLs on their Web site. MDL [22] volunteers manually check URLs, and MDL provides a public forum for analysts. The Information provided (e.g., IP address lists, domain lists, and URL lists) and the update frequency vary depending on the open feeds; therefore, we periodically scanned the webpages of the open feeds.

To expand the malicious candidate domains/URLs, we used a publicly available open threat intelligence service. Such services accumulate information on threats and provide search functions. However at least in free we can not obtain all information which each open intelligence has, in general. It is necessary to use search key such as malicious candidate IP address or domain name. Therefore we use IP address or domain name which we obtained from open feed as search key. We chose to use VirusTotal [25] because it has more information available than other services and they provide an application programming interface (API) function for free. These information is called Open-Source Intelligence (OSINT) in general. However we want to show a different characteristic of open feed and open intelligence, therefore we defined each. Note that since bogus information may be included in OSINT, it is necessary to determine truly malicious or not by a method as shown in Fig 5, before finally using as a blacklist.

Figure 2 shows the detailed data flow of SeedStrainer. To collect the latest malicious domains, we crawled each open feed every few hours. We made a daily unique domain list for three consecutive days (Dom_20170603) and extracted only the new domain list (Domseed_20170605) by taking a difference of the previous days (Dom_20170604 and Dom_20170605). Next, we obtained each domain report using VirusTotal API and extracted the latest URLs from each report.

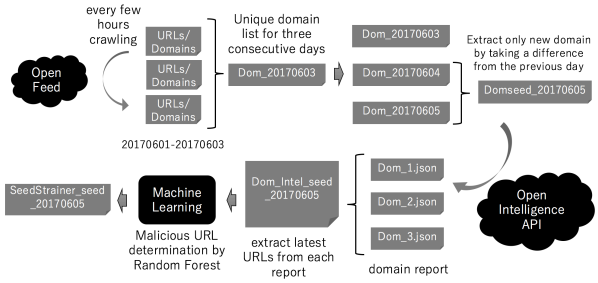


Fig. 2. Detailed data flow of SeedStrainer

B. Machine learning

In Figure 2 we determine malicious URLs with a high hit ratio (SeedStrainer_seed_20170605) from the extracted list (Dom_Intel_seed_20170605) using machine learning. In this experiment we use the Random Forest algorithm for the machine learning because it is high classification performance, excellent scalability, easy to use. Random Forest is one of ensemble learning method where classifiers are composed by collecting multiple classifiers [15]. Table I shows the list of features that we use. We divided the URLs into their domain, path, and query. Next, we calculated their length, entropy, and n-gram (n=2).

TABLE I
Features

Features		
Domain Length	Path Length	Query Length
Domain Entropy	Path Entropy	Query Entropy
Domain n-gram	Path n-gram	Query n-gram

IV. Evaluation

A. Investigation result for the open feeds

To determine which feed was best for SeedStrainer, first we observed each feed. Table II shows the information type

and the number of information items for each feeds. The observation period was 2 months from February 1, 2017, to March 31, 2017. In Table II, *All* indicates the daily average of the total number of items of information we could gather. *Latest* indicates the daily average of the number of latest information items. Note that we use same method in Figure 2 to obtain the latest URLs. Cruzit and urlQuery provide larger amounts of information than other feeds. However, the *Latest* number of Cruzit is low. Figure 3 shows the time series variation of the total number of URLs for the 2 months, demonstrating that the total number of URLs for each feed monotonically increases. Figure 4 shows the time series variation of the number of latest URLs for the 2 months. Overall, the numbers of *all* and *latest* information items for urlQuery are larger than those of the other feeds. Table III shows the other feed information from May 1, 2017, to June 30, 2017. From this result, we selected Vxvault, Dshiled, urlQuery, Ponmocup, and Alienvault.

TABLE II
FEED INFORMATION

Feed name	Info. type	All	Latest
SSLBL[16]	IPaddress	300.2	3.41
Cruzit[17]	IPaddress	6589.9	4.42
Vxvault[18]	URL	601.2	3.03
Dshiled[19]	domain	237.8	2.34
Macl0de[20]	URL	458.1	0.017
urlQuery[21]	URL	69443	687

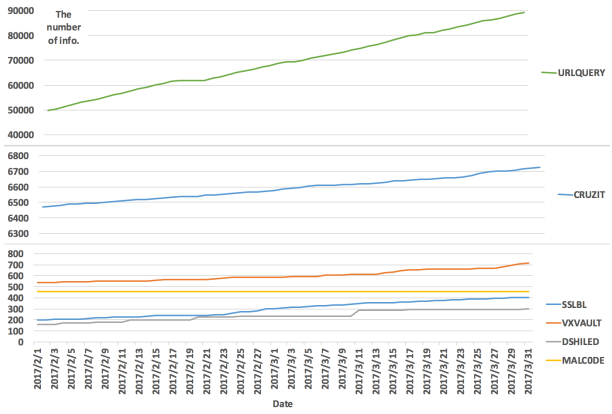


Fig. 3. The number of *All* information items for each feed

TABLE III
OTHER FEED INFORMATION

Feed name	Info. type	All	Latest
MDL[22]	URL	1789.4	0.033
Ponmocup[23]	URL	779.4	6.79
Alienvault[24]	URL	3931	12.05

B. Environment

Figure 5 shows the environment we used to evaluate the hit ratio of the malicious candidate URLs. We use two components, a client honeypot and an antivirus software package.

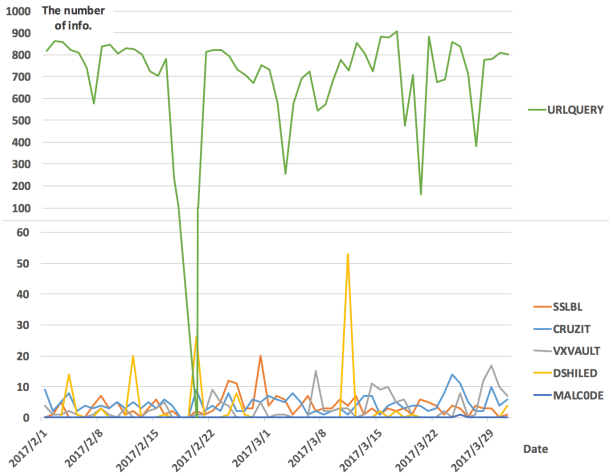


Fig. 4. The number of *Latest* information items for each feed

We used thug [9] as a client honeypot to crawl the seed list of malicious candidate URLs. Thug is a client honeypot that emulates a real web browser, fetches and executes any internal or external JavaScript, follows all redirects, downloads files just like any browser would, and collects the results. We used four antivirus software packages, Trend Micro [26], Symantec [27], McAfee [28], and Kaspersky [29]. In this experiment, if at least one antivirus software package identified the file as malicious, the source URL was set to malicious. These four antivirus applications were selected because they hold the top market shares in Japan. Some antivirus applications return false-negatives; however, the antivirus applications that hold top market share return fewer false-positives than less popular applications. Note that the signatures of the antivirus applications were always updated to the latest version.

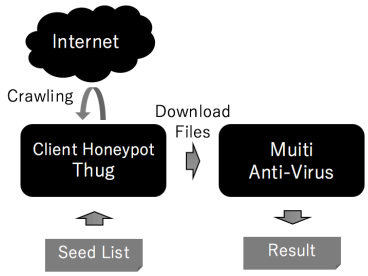


Fig. 5. Evaluation environment

C. Result

We define the hit ratio as the percentage of truly malicious URLs to the number of the original seed list. In Figure 5, the hit ratio is the percentage of the *seed list* to the *result*. Formally, we define the *hit ratio* as follows:

$$Hit\ ratio = Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Note that, in general, since hit rate is called as recall, we denote the *hit ratio* as precision to avoid any misunderstanding.

First, we calculated the hit ratio of the original open feed. Because urlQuery provides more information than the other seeds according to Section IV-A, we calculated the average hit ratio of urlQuery for April 2017, a period of 1 month. Next, we calculated the hit ratio of the prior stage of the machine learning process (the hit ratio of Dom_Intel_seed_20170605 in Figure 2). Table IV shows these values. Note that the hit URL and the input URL indicate the number of each type of URL. Furthermore we show standard deviation (SD) of urlQuery and Prior ML for April 2017, a period of 1 month.

TABLE IV
HIT RATIO

Seed name	hit URLs	input URLs	hit ratio	SD of hit ratio
urlQuery	1,040	20,340	0.0511	0.0149
Prior ML	517	6,731	0.0768	0.0547

Finally, we calculated the hit ratio of SEEDSTRAINER (the hit ratio of SeedStrainer_seed_20170605 in Figure 2). Table V shows the number of URLs and the observation period. Dataset SS is the SEEDSTRAINER seed. For comparison, we prepared Dataset SS_UQ, which was made only with urlQuery seed and machine learning. These datasets contain malicious or benign labels according to the evaluation environment (Figure 5). Note that, because we had to consider our evaluation environment performance, we restricted the number of URLs acquired from open intelligence; therefore the number of data in the SS dataset is smaller than that in the SS_UQ dataset.

TABLE V
DATASET

Dataset Name	URLs	Observation period
SS	24,567	March 30, 2017 - July 25, 2017
SS_UQ	83,270	March 30, 2017 - July 25, 2017

We divided each dataset into two for training and testing. Table VI shows the details of the training and test data. We used old data as training data and new data as test data to predict the future from the past data. We predicted the label of the test data using the Random Forest algorithm with the features in Table I. Figure 6 shows the hit ratio of each feature that we used. The hit ratio is highest when all features are used (i.e., length, entropy, and n-gram: hit ratio = 0.637 (SS)). Of entropy and length, length has the higher hit ratio. Of domain, path, and query, domain has the highest hit ratio. Overall, compared with the original hit ratio in Table IV (hit ratio = 0.0511, urlQuery)), the hit ratio of SEEDSTRAINER improved significantly, by 12.5 times (all features: hit ratio = 0.637 (SS) in Figure 6). Note that observation periods have a little difference of SS and urlQuery dataset, however, standard deviation of urlQuery (see Table IV) is very small, therefore, we think there is no impact on the result.

TABLE VI
TRAINING AND TEST CONDITIONS

Dataset Name	URLs	Observation period
SS_train	12,000	March 30, 2017 - April 27, 2017
SS_test	12,567	April 28, 2017 - July 25, 2017
SS_UQ_train	41,500	March 30, 2017 - May 24, 2017
SS_UQ_test	41,770	May 25, 2017 - July 25, 2017

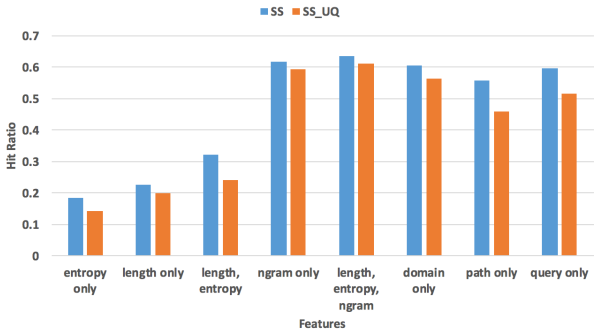


Fig. 6. Hit ratio results

V. DISCUSSION

A. Size of the dataset

Because we needed to consider our evaluation environment performance, we restricted the number of URLs we acquired from open intelligence sources (see Section IV-C). However, in theory, it is possible to expand more with SEED STRAINER; therefore, an experiment with a large dataset is desired. We used only one open intelligence source because it was able to expand sufficiently. If more open intelligence sources are added, more seeds with high hit ratios could be created.

B. Evaluation indicator

In our case, because the number of candidate URLs is very large, it is desirable to have many malicious URLs within the selected item. Therefore, we use the hit ratio (precision) as an evaluation indicator. According to situation evaluation indicator should be chosen. However, we believe that this indicator is important, at least to evaluate the efficiency of crawling.

C. Ethical issues

We could not discern an alternative way of confirming the statuses of the Web sites over the Internet. To reduce the amount of unnecessary traffic in our active measurements, we accessed each Web site once a day rather than consecutivly scanning them over a short period. We downloaded files from the Web sites using a benign procedure, namely by sending an HTTP GET request. No additional intrusive traffic, such as penetration testing and vulnerability scanning, was introduced by our methods.

VI. CONCLUSIONS

To counter malicious sites, blacklisting is widely used as a multilayer defense mechanism in modern Internet security

techniques. Blacklisting on the network side is especially effective for protecting critical embedded systems or IoT devices because it is not necessary to change the configuration or to use system resources for protection. To make accurate blacklist, it is necessary to check malicious candidate URLs, for example, using client honeypots. Because there are numerous malicious candidate URLs and limited crawling resources, efficient crawling is necessary. In this paper, we proposed SEEDSTRAINER, an approach that improves the efficiency of crawling. SEEDSTRAINER creates high-hit-ratio malicious candidate URL lists using open feeds, open intelligence, and machine learning. The hit ratio improved by 12.5 times when using SEEDSTRAINER. In addition, we revealed the types of information distributed and the update statuses of various open feeds.

REFERENCES

[1] C. Grier, L. Ballard, J. Caballero, N. Chachra, C. J.Dietrich, K. Levchenko, P. Mavrommatis, D. McCoy,A. Nappa, A. Pitsillidis, N. Provos, M. Z. Rafique, M. A. Rajab, C. Rossow, K. Thomas, V. Paxson, S. Sav-age, and G. M. Voelker. Manufacturing Compromise:The Emergence of Exploit-as-a-Service. In *Proc. of the ACM Conference on Computer and Communications Security(CCS)*, 2012.

[2] 2015 Internet Security Threat Report, Volume 20. https://www.symantec.com/content/dam/symantec/docs/security-center/archives/istr-15-april-volume-20-en.pdf, 2017.

[3] Microsoft: SmartScreen URL Filter. https://technet.microsoft.com/en-us/library/jj618329(v=ws.11).aspx, 2017.

[4] Google Safe Browsing. https://developers.google.com/safe-browsing/, 2017.

[5] T. Gerbet, A. Kumar, and C. Lauradoux. A Privacy Analysis of Google and Yandex Safe Browsing. In *Proc. of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2016.

[6] Y. Tanaka, M. Akiyama, and A. Goto. Analysis of Malware Download Sites by Focusing on Time Series Variation of Malware. ELSEVIER, Journal of Computational Science, 2017.

[7] C. Seifert and R. Steenson. Capture-HPC. https://projects.honeynet.org/capture-hpc.

[8] M. Akiyama, M. Iwamura, Y. Kawakoya, K. Aoki and M. Itoh. Design and Implementation of High Interaction Client Honeypot for Drive-by-Download Attacks. IEICE TRANS. COMMUN., 2010.

[9] A. dellaera. Low-interaction honeyclient Thug. https://www.honeynet.org/node/827.

[10] J. Stokes, R. Andersen, C. Seifert, and K. Chellapilla. WebCop: Locating Neighborhoods of Malware on the Web. In *Proc. of USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2010.

[11] M. Akiyama, T. Yagi, and M. Itoh. Searching Structural Neighborhood of Malicious URLs to Improve Blacklisting. In *Proc. of Applications and the Internet (SAINT)*, 2011.

[12] J. Zhang, C. Yang, Z. Xu, and G. Gu. PoisonAmplifier: A Guided Approach of Discovering Compromised Websites through Reversing Search Poisoning Attacks. In *Proc. of Research in Attacks, Intrusions, and Defense (RAID)*, 2012.

[13] L. Invernizzi, S. Benvenuti, M. Cova, P. M. Comparetti, C. Kruegel, and G. Vigna. EvilSeed: A Guided Approach to Finding Malicious Web Pages. In *Proc. of the 2012 IEEE Symposium on Security and Privacy*, 2012.

[14] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. In *Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2009.

[15] L. Breiman. Random Forests. Journal of Machine Learning. 2001.

[16] SSL blacklist. https://sslbl.abuse.ch/blacklist/, 2017.

[17] Cruzit. http://www.cruzit.com/, 2017.

[18] Vxvault. http://vxvault.net/ViriList.php, 2017.

[19] Dshiled. https://www.dshield.org/, 2017.

[20] Malc0de. http://malc0de.com/database/?&page=1, 2017.

[21] urlQuery. https://urlquery.net/, 2017.

[22] MDL : malwaredomainlist. https://www.malwaredomainlist.com/, 2017.

[23] Ponmocup. http://security-research.dyndns.org/pub/malware-feeds/ponmocup-infected-domains-CIF-latest.txt, 2017.

[24] Alienvault. https://www.alienvault.com/, 2017.

[25] VirusTotal. https://www.virustotal.com, 2017.

[26] Trend Micro ServerProtect for linux. http://www.trendmicro.co.jp/jp/business/products/splx/, 2017.

[27] Symantec Endpoint Protection. https://www.symantec.com/ja/jp/theme.jsp?themeid=endpointsecurity_lineup_sep, 2017.

[28] McAfee Endpoint Protection Suite. https://www.mcafee.com/jp/products/endpoint-protection-suite.aspx, 2017.

[29] Kaspersky Security. http://home.kaspersky.co.jp/store/kasperjp/ja_JP/pd/ThemeID.37143200/productID.5064647600, 2017.