# Using ID-Hopping to Defend Against Targeted DoS on CAN

Abdulmalik Humayed
University of Kansas
Lawrence, Kansas 66045
ahumayed@ku.edu

Bo Luo
University of Kansas
Lawrence, Kansas 66045
bluo@ku.edu

## ABSTRACT

With the exponential growth of automotive security research, new security vulnerabilities and attacks have been revealed and new challenges have emerged. In recent years, various attacks ranging from replay attacks, through false information injection, to Denial of Service (DoS), have shown how fragile automotive security is. As a result, a number of security solutions have been proposed that rely on techniques like encryption and firewalls. However, most proposals require performance and computational overheads that would become an additional burden rather than a solution. In this paper, we propose a new automotive network algorithm, called ID-Hopping, that aims to prevent targeted DoS attacks in which attackers target certain functions by injecting special frames that would prevent a car's normal operations. We aim to raise the bar for attackers by randomizing the expected patterns in the automotive network. Such randomization hinders the attacker's ability to launch targeted DoS attacks. We built a testing platform and implemented the randomization mechanism to evaluate the algorithm's effectiveness. Based on the evaluation, the algorithm holds a promising solution for targeted DoS, and even reverse engineering, which automotive networks are most vulnerable to.

## CCS CONCEPTS

**•Security and privacy →Embedded systems security; •Computer systems organization →Embedded and cyber-physical systems;**

## KEYWORDS

Security, Smart Cars, Denial of Service, DoS, ID-Hopping, Original IDs, Alternative IDs, CAN

**ACM Reference format:**
Abdulmalik Humayed and Bo Luo. 2016. Using ID-Hopping to Defend Against Targeted DoS on CAN. In *Proceedings of 2017 1st International Workshop on Safe Control of Connected and Autonomous Vehicles (SCAV 2017), Pittsburgh, PA USA, April 2017 (SCAV 2017),* 8 pages.
DOI: http://dx.doi.org/10.1145/3055378.3055382

## 1 INTRODUCTION

Nowadays, smart cars are more intelligent than ever before. Manufacturers embed numerous microcomputers to enhance the cars' safety, comfort, and entertainment features. Examples are collision avoidance systems, and active trace control (safety features); remote start and parking assistance systems (comfort features); and on-board Internet and satellite radio (entertainment features).

The recent advancements in smart cars, however, also introduced new security issues that car manufacturers have not anticipated. Numerous reports and research papers have shown the inadequacy of security in smart cars [1, 9, 13]. These security issues are consequences of the unexpected behaviors that result from the interactions between the many heterogeneous components every smart car consists of [2, 7]. This includes Commercial-Off-The-Shelf (COTS) products and components implemented by third parties. In addition, the increased wireless channels exposed the two-decades-old unconnected cars to numerous security problems.

With complex cyber-physical interactions, as well as increased communication channels, security attacks become imminent. Therefore, security solutions have been proposed to add security layers to smart cars. For example, cryptography-based solutions have been proposed to provide authentication, confidentiality, and integrity measures [17, 21–23, 25]. Intrusion Detection Systems (IDS) have been proposed to detect internal attacks [10, 11, 15, 16, 18, 20] and protection solutions against external attacks [3, 4, 19, 24].

Although these solutions improve security in smart cars, there is a very little attention given to DoS attacks. Automotive networks typically deploy a network protocol called Controller Area Network (CAN). This protocol has inherent security weaknesses such as the lack of authentication and confidentiality, weakness of integrity checking, and lack of nodes' identification. Network nodes, called Electronic Control Units (ECUs), cannot identify each other. Instead, each frame has a unique ID, called *arbitration ID*, that signifies what the frame means and what priority it has. Due to the lack of security measures, an attacker can easily flood the network with a high priority ID to constantly dominate the network and prevent legitimate ECUs from using it. This is considered a DoS attack because ECUs cannot use the network. In addition, an attacker can easily use any ID and spoof other ECUs with illegitimate frames carrying an ID belonging to a legitimate ECU.

In this paper, we consider a special type of DoS where an attacker targets a certain ECU, or a set of ECUs, to prevent it from sending particular frames. For example, when the Anti-Brake Systems (ABS) sends a sensing data frame that is needed to mitigate a potential accident, an attacker would send a frame, or a set of frames, to ensure that the ABS's frame cannot use the network, and therefore the ABS fails. This scenario is safety-critical and proper solutions must be proposed.

We propose the ID-Hopping algorithm where we design a mechanism that hinders an attacker's ability to send malicious frames that would make targeted DoS attacks possible. In addition to prevention, if an attack is attempted, our mechanism detects it and


*SCAV 2017, Pittsburgh, PA USA*

DOI: http://dx.doi.org/10.1145/3055378.3055382

reacts immediately. The mechanism generates a set of alternative IDs that each ECU should use when an attack is detected. Our proposal is similar to Identity-Anonymized CAN (IA-CAN) proposed by Han et. al. [5], where they propose a mechanism that requires all communicating ECUs to calculate anonymous IDs before an ID is sent or received. Our proposal is different in that we only require communicating ECUs to share a carefully calculated value (*offset*) such that it can be used when an attack occurs so the IDs are changed based on that value.

We evaluated the ID-Hopping mechanism by simulating the process of generating alternative IDs from the offset. Two critical conditions our proposal ensures when alternative IDs are generated: 1) alternative IDs are unique to the original IDs and 2) IDs priorities are not compromised.

We believe that the ID-Hopping mechanism holds a promising solution to a variety of DoS attacks. In addition, if we deploy the mechanism to work natively in the network, we believe it should prevent reverse engineering attacks on IDs.

## 2 BACKGROUND

All ECUs are connected to a bus-topology network that has several subnetworks. Each subnetwork consists of a number of interconnected ECUs that perform certain functions. The most common protocol is Controller Area Network (CAN), which is a typical bus network through which ECUs can intercommunicate, monitor sensors, and control actuators.

### 2.1 CAN Protocol

CAN is the most common protocol because it has been mandated to be deployed in all cars in the US since 2008 [9]. CAN protocol mainly provides two services: 1) at the physical layer, it allows the transmission of frames as voltages that do not get influenced by interfering magnetic fields, and 2) at the data link layer, each frame is formatted in a well-defined format so ECUs can exchange messages in a meaningful manner. After that, a higher level protocol is needed to handle the data contained in the frames and deal with the semantics. These layers are specified by ISO 11898-1 through 11898-5.

*2.1.1 CAN Subnetworks.* Typically, CAN networks are divided into three subnetworks: 1) powertrain, 2) comfort, and 3) infotainment. The powertrain subnetwork consists of ECUs that monitor and control operations related to the engine, brakes, and other critical operations. Whereas the comfort subnetwork consists of ECUs that open/close windows, control HVAC, and adjust seats, to name a few.

In addition to the in-car bus network, smart cars have a number of wireless interfaces that are of two kinds: short-range (Bluetooth, Remote Keyless Entry, Tire Pressure Monitoring System, and RIFD keys) and long-range (cellular channels) [1]. Fig. 1 shows a typical example of CAN network consisting of three subnetworks. If an ECU in *network 1* wants to communicate with another ECU in *network 2*, the central gateway forwards the message.

*2.1.2 Arbitration IDs.* CAN protocol has a special approach to handle addresses of senders/recipients of messages. There is not an IP-like addressing. Instead, the protocol uses an 11-bit field, and

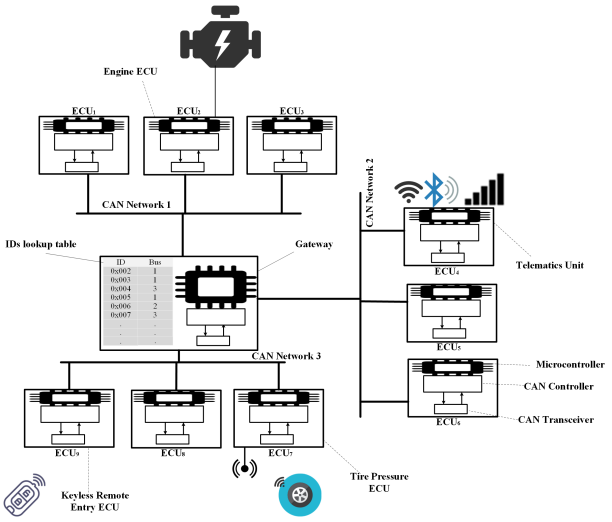


**Figure 1: Overview of a Modern In-Vehicle Network**

could be extended to 29 bits, that is called arbitration ID. Each CAN frame starts with an ID that determines the frame's purpose and priority. The purpose is what the frame means to ECUs, whereas the priority determines the frame's ability to win arbitration over using the network when another frame collides with it in the bus. The frame with the lower value ID will get higher priority and therefore dominate the network. All frames must have unique IDs in order to avoid errors caused by two frames transmitting simultaneously as a result of dominating the bus because of their identical IDs.

A frame gets the right to access the bus if the ID has the lowest value. When a frame occupies the bus, all ECUs receive it and only the interested ECUs accept it. An ECU only accepts frames that it is configured to accept by recognizing their IDs. The choice of arbitration IDs is proprietary and differs from an Original Equipment Manufacturer (OEM) to another. In addition, an arbitration ID is not the sender or the recipient's address. Rather, it is only an indicator of a message's content and purpose.

### 2.2 ECUs

Each ECU is composed of three components: a microcontroller, CAN controller, and CAN transceiver. The microcontroller is responsible for high-level functions such as calculating the speed, sending an airbag command, and warning the driver about oil pressure. Whereas the CAN controller is responsible for outgoing/incoming CAN frames to/from the CAN bus. The CAN transceiver converts the frames to/from physical-level bits.

Typically, each ECU is configured to use a certain set of IDs for its outgoing frames that would make sense for certain ECUs, and another set for incoming frames from other ECUs. For example, in Table.1, $ECU_1$ broadcasts a frame with an ID `0x002`. $ECU_2$ is configured to accept frames with the ID `0x002`.

## 3 DOS ATTACKS

Due to the way ID arbitration works in CAN networks, DoS is very feasible. There are a number of DoS reported attacks. For

| $ECU_1$ | | $ECU_2$ | |
|---|---|---|---|
| Send | Receive | Send | Receive |
| 0x002 | 0x005 | 0x005 | 0x002 |
| 0x003 | 0x006 | 0x006 | 0x009 |
| 0x004 | 0x007 | 0x007 | 0x011 |

**Table 1: IDs in ECUs 1 and 2**

example, Koscher et al. [9] disabled CAN communication to and from the Body Control Module (BCM) which resulted in a sudden drop from 40 to 0 MPH on the speedometer. In addition, this attack also resulted in freezing the whole Instrument Panel Cluster (IPC) in its current state. For example, if the speedometer was at 60 MPH before the attack, and the driver increases the speed, there will be no change in the speedometer. Hoppe et al. [6] demonstrated other forms of DoS attacks. One of which is where the attack prevents passengers from closing any opened window. Another is to disable the warning lights. The authors also performed another DoS attack on the theft alarm system, so that it will not go off during a burglary. DoS attacks can take on different forms whose impacts vary in safety-criticality such as traditional DoS, random DoS, and targeted DoS.

## 3.1 Types of DoS

*3.1.1 Traditional DoS.* An attacker could simply flood the network with frames that have the lowest IDs, i.e., `0x000`, such that their dominance of the network is guaranteed. Solving this kind of DoS is not possible at the data-link and application layers. Rather, a physical design modification is needed so that extra checks could be introduced before a high priority frame continuously dominates the network. In addition, the detection of this traditional DoS is not difficult. When ECUs are not able to transmit nor receive frames, they could revert to fail-safe mode as a safety feature in CAN [5]. Our proposal is designed towards a selective class of DoS attacks that is subtler and more difficult to detect.

*3.1.2 Random DoS.* The attacker in this type of DoS does not target a specific ECU, rather he would randomly send frames with randomly selected IDs aiming to disturb ECUs' normal operations. Although fuzzing the network with random IDs might result in unintentional DoS, the attack is not targeted against a particular ECU.

*3.1.3 Targeted DoS.* An attacker could be interested in sabotaging certain ECUs for different reasons. For example, an attacker could target the airbag system to fail when a collision occurs. This could cause life-threating consequences to the driver and passengers. Targeted DoS is a subtle attack and challenging to detect, let alone prevent. Therefore, targeted DoS is the focus of this paper.

## 3.2 Attacker Model

In this paper, we consider a variant of DoS attacks where an attacker targets a certain ECU and aims to prevent its frames from using the network and eventually from arriving to the desired destinations. For example, in Fig.2, the engine's ECU ($ECU_2$) sends its periodic CAN frame containing some information to be displayed to the

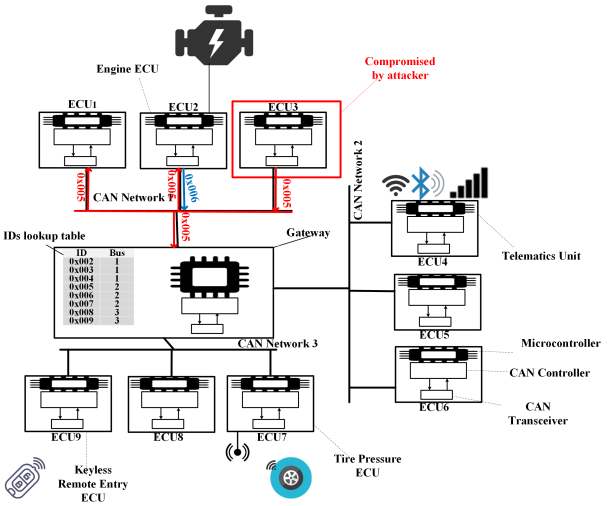


**Figure 2: Scenario of a targeted DoS attack**

driver, such as speed and temperature. The frame has an arbitration ID with a value of `0x006`.

At the time $ECU_2$ is sending this frame, it has the highest priority in *CAN network 1*. Therefore it should successfully occupy the bus and get transmitted to the gateway, which in turn forwards the information to the instrument panel cluster's ECU in another CAN network.

However, an attacker is assumed to have compromised $ECU_3$ and configured its controllers to monitor the network. $ECU_3$ gets triggered and floods the network with a frame whose ID has a lower value such as `0x005` whenever a frame with the ID `0x006` is observed in the network. This attack scenario could miss the first frame with the ID `0x006`, but the remaining frames are going to be affected. This will force $ECU_2$ to retract from sending `0x006` frame due to the network's occupancy with a higher priority ID, i.e., `0x005`. The latter frame has higher priority as it has the lowest value, and thus can dominate the bus. This attack ensures that $ECU_2$ cannot use the bus resulting in an unfair usage of resources by other ECUs, in particular, the attacker's. This attack is often called *fairness/starvation attack*.

# 4 ID-HOPPING MECHANISM

## 4.1 Requirements

The mechanism should satisfy the following requirements:

(1) No modifications needed for current CAN protocol.
(2) The mechanism must preserve the same level of priority for current IDs configured by OEMs.
(3) Minimum to no performance overhead on the network.

## 4.2 Overview

When an ECU becomes target of a DoS attack, it should detect that it is under attack and then switch its frames' ID to the proposed ID-Hopping mechanism. We propose configuring ECUs with two types of IDs: 1) IDs used during normal operations and 2) IDs used when an ECU or more is under a targeted DoS attack. The first type

is currently deployed in all cars, whereas the latter is our proposal towards thwarting targeted DoS attacks against certain ECUs.

For the sake of simplicity, let's say that $ECU_1$ is sending a frame with ID `0x002`. An attacker immediately floods the network with a frame that has `0x001` ID in order to win the arbitration and cause the `0x002` frame to retract. After a few attempts sending the `0x002` frame, $ECU_1$ notices that the frame loses the arbitration repeatedly due to the existence of a higher ID. Also, a host-based or gateway-based mechanism should be implemented to recognize that the frame with `0x001` ID only gets triggered whenever $ECU_1$ sends `0x002` ID. We assume that one or both of these two patterns are to be detected through a trained detection mechanism for DoS attacks. Details of this mechanism should be further demonstrated in our future work.

To overcome the DoS attack, $ECU_1$ should switch to ID-Hopping mode where it uses an alternative set of IDs in order to confuse the attacker. Now, $ECU_1$ sends a frame with `0x009` ID which carries the same data as the previous frame, except that it uses a different ID as shown in Table 2. This should thwart the attacker from targeting the ECUs until he learns the alternative set of IDs. The attacker needs close monitoring and reverse engineering in order to uncover the alternative IDs. When this happens, another DoS attack could be launched during the ID-Hopping mode. Therefore, the gateway and ECUs are already prepared to hop to a new set of alternative IDs that have been already generated.

| ID | Alt ID | Receive | Alt ID |
|---|---|---|---|
| `0x002` | `0x102` | `0x005` | `0x105` |
| `0x003` | `0x103` | `0x006` | `0x106` |
| `0x004` | `0x104` | `0x007` | `0x107` |

**Table 2: $ECU_1$ IDs**

ECUs that receive frames from $ECU_1$ need to be aware of the alternative IDs in order for the car to function normally. We achieve that by having the alternative IDs in every ECUs' lookup table as long as the ECU is configured to receive the original ID before an attack takes place. Each ECU calculates alternative IDs with the offset sent by the gateway. For example, Table 3 shows that $ECU_2$ has the alternative ID `0x00C` as well as the original ID `0x005`.

Finally, when ECUs switch to the ID-Hopping mode, the alternative IDs must not compromise the original priorities of the original IDs. ID-Hopping mode must preserve the overall priority across all IDs. In other words, each original ID will have an equivalent alternative ID in terms of priority. So if two original IDs 0x100 and 0x101 are used in the alternative mode, they will be 0x111 and 0x112. Hence, the priority is still preserved even though different IDs are used. This can be achieved when all ECUs, including the gateway, use the alternative IDs when an ECU or more are under attack. If at least one ECU does not switch to the ID-Hopping mode, this requirement is violated. Therefore, the gateway needs to ensure that all ECUs are in the ID-Hopping mode to avoid priority violations.

## 4.3 The Algorithm

In the gateway, the ID-Hopping algorithm takes a list of used IDs as input to generate the alternative IDs. Each resulting ID is unique

| ID | Alt ID | Receive | Alt ID |
|---|---|---|---|
| `0x005` | `0x105` | `0x002` | `0x102` |
| `0x006` | `0x106` | `0x003` | `0x103` |
| `0x007` | `0x107` | `0x04` | `0x104` |

**Table 3: $ECU_2$ IDs**

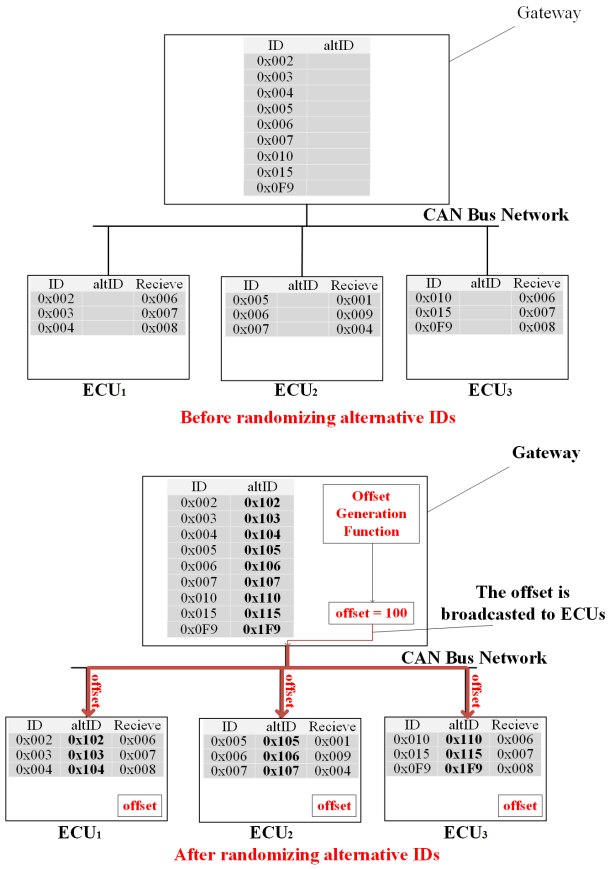


**Figure 3: Overview of the network before and after the ID-Hopping takes place**

and not used by any ECU in the network. This generation is the result of adding a carefully selected offset to the currently used IDs resulting in a new list of IDs that preserve the same level of priority as their original counterparts. Then each ECU receives the offset, in a secure way, and then adds it to every original ID it has, resulting in an alternative ID corresponding to each original ID. Finally, when an ECU detects a targeted DoS attack, ID-Hopping is activated and all ECUs switch to the alternative IDs. We assume that ECUs share a secret key with the gateway so it can broadcast the offset securely.

*4.3.1 Phase One: IDs Generation.* When an attack is detected, either by the gateway or an ECU, the ID-Hopping mode gets activated. The gateway calculates a randomly generated offset and uses it for generating an alternative ID for every used ID. The assigned

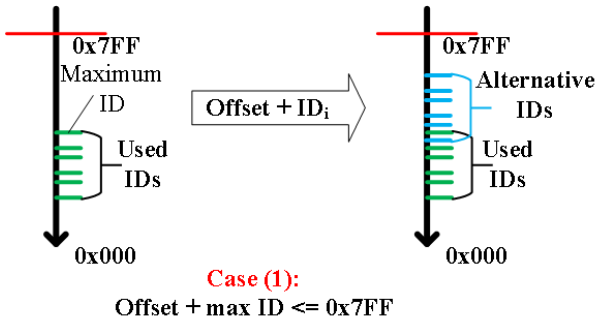


**Figure 4: Alternative IDs' Generation: Case 1**

IDs will not compromise the existing priorities configured by the car's manufacturer. In the deployed algorithm, the basic idea is to shift the values of current IDs by a random offset. The offset itself must satisfy three conditions: 1) $offset \geq$ `0x001`, 2) $offset + maximum \leq$ `0x7FF`, and 3) $usedID_i + offset \neq usedID$.

These three conditions ensure that no alternative ID results in an out-of-bound value, i.e., more than `0x7FF`, which is upper bound for an 11-bit ID. Once the random offset is generated, each ECU generates the alternative IDs by adding this offset to each of the original IDs. The offset is added even with IDs that are not targeted in order to maintain the overall posture of priorities across the network that the manufacturer has intended.

Before adding the offset to the original IDs, we check whether the value of the highest ID is `0x7FF` . If it is not, we proceed with the described procedure above as shown in Fig.4. However, if the maximum ID is indeed `0x7FF`, this means there will be at least an alternative ID out-of-bound and will have more than 11-bit value. Therefore, the algorithm makes some additional steps before generating the alternative IDs as shown in the pseudocode of Algorithm 1.

Firstly, we take the lowest available set of IDs and add the random offset to each one. Although it could be enough to use these low value available IDs as the alternative ID, we need to avoid detectable patterns in the alternative IDs such as consecutive numbers. This is the case because our algorithm deals with IDs that have already been assigned by the manufacturer in a proprietary manner. Therefore, we cannot predict how they have been assigned. There is one case though where the resulting alternative IDs are in sequence. Such pattern is expected if the value of the highest original ID is `0x7FE`, which is the upper bound `0x7FF` $- 1$. The algorithm can only generate $offset = 1$ because it is the only available number that satisfies the two conditions discussed above. This can easily be changed with the offset requirements to ensure it has a value greater than one.

*4.3.2 Phase Two: IDs Assignment.* When the gateway generates the alternative IDs, all ECUs need to know about the alternatives of their IDs before an attack takes place. We considered two potential methods to distribute the alternative IDs: 1) sending a list of the corresponding alternative IDs to each individual ECU or 2) sending the generated offset instead, and then each ECU calculates the alternative IDs by adding the offset to each original ID. The first approach takes $n$ transmissions, where $n$ is the number of alternative IDs to be sent to each ECU. The latter approach only requires one transmission because the gateway broadcasts the offset and all ECUs receive it simultaneously. Therefore, we chose the latter as it significantly requires less network usage.

One might ask how the offset is sent to ECUs. What if an attacker pretends to be a legitimate ECU, which could potentially allow him to receive the offset and easily deduce the alternative IDs of the

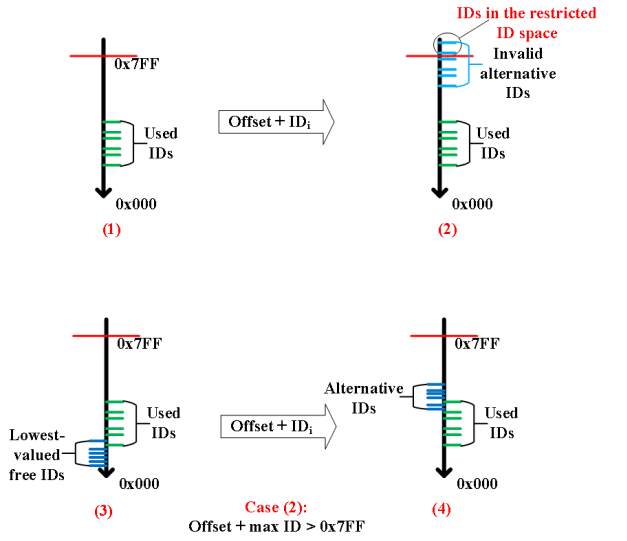


**Figure 5: Alternative IDs' Generation: Case 2**

**Algorithm 1** Phase 1 of ID-Hopping algorithm

The **RandomGen** function computes the offset based on the used IDs.
The **getMax** function returns the highest ID in the passed list.
The **getLowestIDs** function returns the available IDs with the lowest values.
The **getOffset** function returns the offset that is: 1) $offset \geq 1$, 2) $offset + max \leq 0x7FF$ and 3) $ID_i + offset \neq usedID_i$
Input: *usedIDs*: a list of all used IDs by the ECUs.
Output: *offset*: the value to be sent to ECUs
Variables: **max:** the highest value of the used IDs, **altIDs:** a list of the alternative IDs, **rand:** a random integer, **lowIDs:** a list containing the lowest available IDs

**RandomGen**(*usedIDs*):

```
1:  ids ← usedIDs
2:  max ← getMax(ids)
3:  if max = 0x7FF then
4:      lowIDs ← getLowestIDs(ids)
5:      ids ← lowIDs
6:      go to 2
7:  else
8:      offset ← getOffset(ids)
9:  end if
10: return offset
```

targeted original IDs? One of the most straightforward ways is to establish a secure connection between the gateway and ECUs. The encryption could be asymmetric, i.e., with a public/private key pair, or symmetric, i.e., with a shared secret key. The latter is a more efficient approach that does not introduce network overhead, especially if the encryption is only used for sending the offset. Therefore, we assume that an encryption mechanism is in place for sending the offset.

Because encryption operations are relatively expensive and could add network overhead, we limit the encryption to only this operation, i.e., sending the offset to ECUs once every time alternative IDs are generated. This operation is required at least once when the car starts, and once again when an attack is detected. Therefore, when each ECU receives the encrypted offset, encryption is no longer needed and the network should not suffer from any overhead.

*4.3.3 Phase Three: ID-Hopping Mode.* When an ECU detects that one (or more) of its IDs is no longer able to occupy the network, the ECU assumes that it is under a targeted DoS attack. Therefore, it resorts to the ID-Hopping mode where only alternative IDs are used. Receiving ECUs and the gateway will receive the same frame but with an alternative ID. As a result, they notice that an alternative ID is used, and hence they switch to ID-Hopping mode as well. The gateway broadcasts a message to all ECUs, including the unaffected ones, telling them to switch. The message is encrypted and containing a flag that indicates that ID-Hopping mode is activated. Therefore, all receiving ECUs switch to the ID-Hopping mode. The reason we force all ECUs in the network to switch is to preserve the IDs' priorities assigned by the car's manufacturer. If we only allow one alternative ID to be used, the overall IDs' priority order might become compromised.

# 5 SECURITY ANALYSIS

## 5.1 Attack Prevention

We implemented a targeted DoS attack with the testing platform explained in Sec.6 and the ID-Hopping successfully detected the attack and made all ECUs switch to their alternative IDs and the network behaved normally. The mechanism proved its effectiveness to prevent targeted DoS attacks.

## 5.2 Collision-Free

Our evaluation showed that when all ECUs switch to the alternative IDs, the network is collision-free. This is because the ID-Hopping mechanism guarantees that there is no case where two ECUs have the same alternative IDs unless they had the same original ID assigned by the OEM.

## 5.3 Priority Preservation

ID-Hopping successfully maintains the priorities given to the assigned original IDs as long as all ECUs switch to the alternative IDs mode. This is because the mechanism is designed to carefully calculate an offset that guarantees two conditions: 1) original IDs priorities are preserved and 2) alternative IDs are unique.

## 5.4 Protocol-Independent

Compared to Han et al.'s work [5] which needs to be configured differently across different CAN variants, ID-Hopping is independent of the deployed CAN variants because it takes all bits of the ID field as the priority bits.

## 5.5 Limitations

*5.5.1 Original Frames are intact, except their IDs.* One limitation in the ID-Hopping mechanism is that when ECUs switch to the ID-Hopping mode, the original frames remain with no modifications. In other words, the only change is in the IDs. This poses a risk for the attacker to cross match the contents of the frames and infer the alternative ID for the attacked frame. A possible solution is to modify the contents of the original frame, as well as the ID, possibly by shifting the values by the same offset used for IDs.

*5.5.2 Attacker retrieves the offset.* It could be possible for an attacker to compromise an ECU and gain access to the encrypted offset. We assume that such an attack needs physical access to the ECU in order to succeed. To prevent such an attack we need to rely on tamper-proof ECUs that are physically-protected. Furthermore, since the original IDs hop once every time the nodes switch to the ID-Hopping mode, it is possible to deduce the offset from analyzing the change pattern in the alternative IDs.

*5.5.3 Potential limited CAN ID space.* Because free CAN ID space is only known for OEMs, our mechanism might fall short when implemented in a network with little or no free CAN ID space. A potential solution that could be considered is the use extended CAN frames, where we can use 29-bit IDs rather than 11-bit. Better yet, we could improve our design to use 11-bit IDs by default, and 29-bit IDs when space is limited. Therefore, the mechanism can scale when free ID space is limited or the number of ECUs is large.

# 6 EVALUATION

Our evaluation objectives are twofold. First, to evaluate the offset generation and distribution phases and how the gateway and ECUs calculate the alternative IDs. Second, to evaluate the effectiveness of the ID-Hopping during an attack.

Fig. 6 shows our test platform which consists of five BeagleBone Black (BBB) microcontrollers, five CAN Capes, a breadboard, a switch, a multimeter, and an external LED. Four of the BBBs simulate regular ECUs, and the fifth simulates a gateway. The communication between the microcontrollers is realized by the simulated CAN bus in the breadboard. The switch is not relevant to smart cars. Rather it is only used for accessing and configuring the BBBs for our test scenarios. The implementation has been done using *python-can*[1], the python implementation of *SocketCAN* [8].

The evaluation shows that the algorithm successfully generates alternative IDs that substitute the original IDs. Also, the gateway and ECUs successfully switched to the ID-Hopping mode when an attack was detected. Furthermore, the alternative IDs maintain the same priority level as in the original IDs. The current version of ID-Hopping is designed for a single CAN network and the size of IDs is 11 bits. We plan to perform a more realistic evaluation on a testing bench and test the mechanism with the extended 29-bit ID.

[1] http://python-can.readthedocs.io/en/latest/index.html

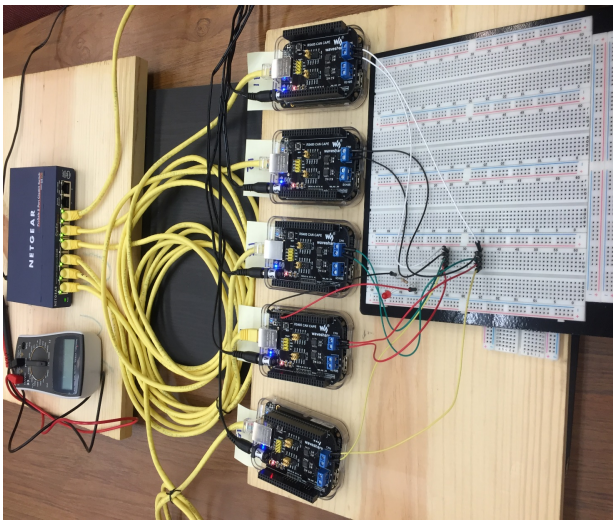


**Figure 6: Testing Platform**

## 7 RELATED WORK

Solutions that rely on cryptography have been proposed to provide authentication, confidentiality, and integrity measures [17, 21–23, 25]. In addition, IDS also have been proposed such as in [10, 11, 15, 16, 18, 20]. Furthermore, controls preventing attacks initiated from external devices have been introduced in [3, 4, 19, 24]. Most of the proposed solutions require ECUs to verify each arriving frame before deciding to accept it for further processing or drop it. This in itself poses ECUs to DoS attacks when an attacker floods the network with frames that have certain IDs but incorrect data. When this happens, a receiving ECU would check the ID first and then verify other measures such as the authenticity and integrity of the frame. This adds an unnecessary overhead in terms of the number of operations needed at the receiving side.

The most relevant work to ID-Hopping mechanism is the Identity-Anonymized CAN (IA-CAN) that Han et. al. have proposed [5]. They introduce a solution that provides authentication with minimal overhead as opposed to other CAN-based authentication approaches, e.g., [21, 22, 25], that require relatively more overhead.

Therefore, Han et. al. [5] propose an ID-anonymization approach that lets ECUs only accept legitimate frames based on their anonymous IDs. The IA-CAN anonymizes IDs in the sense that they have no meaning for an eavesdropper. With some reverse engineering skills and patience, an attacker could infer CAN IDs and then launch various attacks such as sending false information and safety-critical command injection. IA-CAN prevents such inference because the IDs are changed randomly, from the attacker's viewpoint, so that only legitimate ECUs can send and receive frames using IDs that only they can know.

IA-CAN works in three stages: 1) an ID is generated for each frame before it is sent and uses the shared secret key between the sender and receiver, 2) each receiving ECU has already pre-computed the ID, 3) when a frame with the expected ID arrives, the receiving ECU simply filters it using an XOR operation. Each ID is used only once, which prevents replay attacks. The authors have four assumptions in order for their mechanism to work: 1) they assume there is only one CAN network where an attacker exists, 2) ECUs share a secret key before the communication takes place, 3) keys are securely pre-stored in a tamper-proof hardware, 4) there is a time ECU for synchronization.

In addition, an important feature in IA-CAN is ID priority preservation. The authors maintain the priority bits intact in the arbitration field, and anonymize/randomize the remaining bits. The arbitration bits vary depending on the deployed standard. For example, SAE J1939 uses the first 3 bits of the 29-bit ID. In our proposal, we maintain the priority by considering all bits in the ID field. In other words, we treat the whole ID as priority bits. Therefore, regardless of the used variant of CAN protocol, ID-Hopping should be applicable.

Using the same IDs across a large number of cars that use the same platform exposes them to large-scale attacks that might affect a significant number of cars. As a matter of fact, Miller and Valasek [14] had the capability of attacking any car in the US that belongs to a certain fleet of cars that uses the same platform. They were able to uncover a vulnerability in a car that allowed them to take remote control over it by exploiting its cellular communication channel. Once they got into the car's internal network, they could inject CAN frames with IDs that they have already reverse engineered and know what their purposes are. Because all cars of this fleet use the same set of IDs in their CAN network, the attack was possible to be launched on more than a million cars across the US.

Therefore, another work along the lines of IDs randomization is proposed by Lukasiewycz et. al. [12]. They introduce an approach, called *Security-aware Obfuscated Priority Assignment*, that would prevent large-scale attacks on a fleet of cars that use the same platform and hence the same CAN IDs. Unlike our proposal and IA-CAN, this mechanism is aimed at a fleet of cars instead of an individual car. In addition, the IDs' priority assignment will be fixed in each car from the time it is manufactured. This might protect against large-scale attacks, but not a targeted attack on one car. They use Quadratically Constrained Quadratic Program (QCQP) solving to generate obfuscated CAN IDs across different cars using the same platform. Unlike Lukasiewycz's et. al. [12] proposal, our ID-Hopping mechanism does not need to be integrated during the design phase. Instead, we claim that it should be applicable to existing CAN networks with fewer modifications, namely the shared secret key between ECUs and the central gateway in order to authenticate whenever IDs need to be changed.

## 8 FUTURE WORK

Our future plans include improving ID-Hopping to work with 29-bit CAN IDs and tackling multiple simultaneously targeted DoS against single/multiple ECUs. In addition, the current version of ID-Hopping performs a single hop every time an attack is detected. We plan to improve the mechanism to handle multiple hops. We would design more attack scenarios so that an attacker would be able to prevent non-periodic frames (we used a periodic speed update frame in our scenario) and inject false data to render the targeted ECU unavailable.

Further work needs to be done to ensure the synchronization of the gateway and ECUs so that they are all aware of the ID-Hopping mode. In addition, we need to investigate the possible collateral damage that could affect ECUs other than the targeted one.

## 9 CONCLUSIONS

While CAN is prone to various attacks due to the lack of security measures, smart cars are increasingly exposed to new communication channels. This makes it critical to design short-term and long-term security solutions. Our work serves as a short-term solution that works with current CAN networks so that only minimal modification is needed to the current infrastructure deployed in today's smart cars. Long-term solutions take time to be adopted and require security considerations from early design phases.

In this paper, we propose ID-Hopping mechanism to defend against targeted DoS on CAN networks. We design the algorithm to provide ECUs with an efficient approach to generate alternative IDs that can be used when a DoS takes place. Of particular significance is that these alternative IDs maintain the same priorities of the original IDs regardless of the CAN variant used. In addition, the gateway only needs to send a single frame containing an offset that is carefully calculated to ECUs so they can calculate their alternative IDs.

We evaluated ID-Hopping with a testing platform consisting of five microcontrollers, simulating ECUs and a gateway, and a breadboard, simulating the CAN bus. The evaluation shows successful alternative ID generation, DoS detection, and prevention.

## ACKNOWLEDGMENTS

We would like to thank Dr. Yasser Shoukry and Dr. John Gibbons for their valuable discussions and ideas, and anonymous reviewers for their insightful feedback and suggestions to improve this paper.

This work was supported in part by the US National Science Foundation under NSF CNS-1422206 and NSF DGE-1565570.

## REFERENCES

[1] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings*. http://static.usenix.org/events/sec11/tech/full_papers/Checkoway.pdf

[2] Jeffrey Cook. 2006. Cyber-physical systems and the twenty-first century automobile. In *NSF Workshop on Cyber-Physical Systems*. 1–3.

[3] Andrea Dardanelli, Federico Maggi, Mara Tanelli, Stefano Zanero, Sergio M Savaresi, R Kochanek, and T Holz. 2013. A security layer for smartphone-to-vehicle communication over bluetooth. *IEEE embedded systems letters* 5, 3 (2013), 34–37.

[4] Kyusuk Han, Swapna Divya Potluri, and Kang G Shin. 2013. On authentication in a connected vehicle: secure integration of mobile devices with vehicular networks. In *Cyber-Physical Systems (ICCPS), 2013 ACM/IEEE International Conference on*. IEEE, 160–169.

[5] Kyusuk Han, André Weimerskirch, and Kang G Shin. 2014. Automotive Cybersecurity for In-Vehicle Communication. In *IQT QUARTERLY*, Vol. 6. 22–25.

[6] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann. 2008. Security Threats to Automotive CAN Networks — Practical Examples and Selected Short-Term Countermeasures. In *SAFECOMP*.

[7] Abdulmalik Humayed and Bo Luo. 2015. Cyber-physical security for smart cars: taxonomy of vulnerabilities, threats, and attacks. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*. ACM, 252–253.

[8] M Kleine-Budde. 2012. SocketCAN–The official CAN API of the Linux kernel. In *Proceedings of the 13th International CAN Conference (iCC 2012), Hambach Castle, Germany CiA*. 05–17.

[9] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. 2010. Experimental Security Analysis of a Modern Automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*. 447–462. DOI: http://dx.doi.org/10.1109/SP.2010.34

[10] Ulf E Larson, Dennis K Nilsson, and Erland Jonsson. 2008. An approach to specification-based attack detection for in-vehicle networks. In *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 220–225.

[11] Congli Ling and Dongqin Feng. 2012. An Algorithm for Detection of Malicious Messages on CAN Buses. In *2012 National Conference on Information Technology and Computer Science*. Atlantis Press.

[12] Martin Lukasiewycz, Philipp Mundhenk, and Sebastian Steinhorst. 2016. Security-Aware Obfuscated Priority Assignment for Automotive CAN Platforms. *ACM Trans. Des. Autom. Electron. Syst.* 21, 2, Article 32 (Jan. 2016), 27 pages. DOI: http://dx.doi.org/10.1145/2831232

[13] Charlie Miller and Chris Valasek. 2013. Adventures in Automotive Networks and Control Units. A SANS Whitepaper. (August 2013). http://illmatics.com/car_hacking.pdf

[14] Charlie Miller and Chris Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA* (2015).

[15] Michael Müter and Naim Asaj. 2011. Entropy-based anomaly detection for in-vehicle networks. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 1110–1115.

[16] Michael Müter, André Groll, and Felix C Freiling. 2010. A structured approach to anomaly detection for in-vehicle networks. In *Information Assurance and Security (IAS), 2010 Sixth International Conference on*. IEEE, 92–98.

[17] Hisashi Oguma, XAkira Yoshioka, Makoto Nishikawa, Rie Shigetomi, Akira Otsuka, and Hideki Imai. 2008. New attestation based security architecture for in-vehicle communication. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*. IEEE, 1–6.

[18] Satoshi Otsuka, Tasuku Ishigooka, Yukihiko Oishi, and Kazuyoshi Sasazawa. 2014. *CAN Security: Cost-Effective Intrusion Detection for Real-Time Control Systems*. Technical Report. SAE Technical Paper.

[19] Florian Sagstetter, Martin Lukasiewycz, Sebastian Steinhorst, Marko Wolf, Alexandre Bouard, William R Harris, Somesh Jha, Thomas Peyrin, Axel Poschmann, and Samarjit Chakraborty. 2013. Security challenges in automotive hardware/software architecture design. In *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 458–463.

[20] Hendrik Schweppe. 2012. *Security and privacy in automotive on-board networks*. Ph.D. Dissertation. Télécom ParisTech.

[21] Anthony Van Herrewege, Dave Singelee, and Ingrid Verbauwhede. 2011. CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus. In *ECRYPT Workshop on Lightweight Cryptography*, Vol. 2011.

[22] Qiyan Wang and Sanjay Sawhney. 2014. VeCure: A practical security framework to protect the CAN bus of vehicles. In *Internet of Things (IOT), 2014 International Conference on the*. IEEE, 13–18.

[23] Marko Wolf and Timo Gendrullis. 2012. Design, implementation, and evaluation of a vehicular hardware security module. In *Information Security and Cryptology-ICISC 2011*. Springer, 302–318.

[24] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee. 2015. A practical wireless attack on the connected car and security protocol for in-vehicle CAN. *Intelligent Transportation Systems, IEEE Transactions on* 16, 2 (2015), 993–1006.

[25] Tobias Ziermann, Stefan Wildermann, and Jürgen Teich. 2009. CAN+: A new backward-compatible Controller Area Network (CAN) protocol with up to 16× higher data rates.. In *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09*. IEEE, 1088–1093.