

Identifier Randomization: An Efficient Protection Against CAN-Bus Attacks



Khaled Karray, Jean-Luc Danger, Sylvain Guilley, and M. Abdelaziz Elaabid

Abstract The Cyber-Physical Architecture of vehicles is composed of sensors, actuators, and electronic control units all communicating over shared communication buses. For historical reasons the internal communication buses, as the Controller Area Network (CAN), do not implement security mechanisms; the communications are assumed to be “trusted.” Recently these trusted relations have been challenged and leveraged to launch cyber-physical attacks against modern vehicles. As a result, it becomes urgent to enhance the security features of vehicles and notably the robustness of the CAN bus which represents an important channel of attacks.

In this work we develop identifier randomization procedures whose aim is to protect the CAN protocol from reverse-engineering, replay, and injection attacks. The idea behind this proposition is to constantly change the message identifiers in a random fashion in a way that both sender and receiver can recover the original message identifier but not the adversary. We present the main challenges of the CAN-ID randomization solution, we highlight the weaknesses of state-of-the-art solutions presented in other scientific papers, and we propose and study candidate solutions

K. Karray (✉)
Télécom ParisTech, Paris, France
e-mail: khaled.karray@telecom-paristech.fr

J.-L. Danger
Télécom ParisTech, Paris, France
Secure-IC S.A.S., Cesson-Sévigné, France
e-mail: jean-luc.danger@telecom-paristech.fr

S. Guilley
Télécom ParisTech, Paris, France
Secure-IC, Paris, France
École normale supérieure, Paris, France
e-mail: sylvain.guilley@telecom-paristech.fr; sylvain.guilley@secure-ic.com

M. Abdelaziz Elaabid
PSA-GROUPE, Paris, France
e-mail: abdelaziz.elaabid@mpsa.com

to overcome these weaknesses. To compare our solutions to state-of-the-art solution, we propose to use the entropy and the conditional entropy as a metrics of security. Results show that the randomization functions that we propose outperform the state-of-the-art solution in terms of both entropy and conditional entropy.

1 Introduction

Two important requirements of today's cars are a high level of safety and connectivity with the outside world. This involves the use of advanced technologies based on a computing infrastructure composed of numerous electronic components—named electronic control units (ECUs)—embedded inside the vehicle. These ECUs are in charge of processing sensed data through embedded sensors and transforming it into commands for the actuators. Communication buses in the automotive domain were introduced as soon as the ECUs embedded in the vehicle have reached a certain level of complexity. This made a point-to-point communication approach no longer viable and impossible to implement and maintain. At that point, the car is a system on its own, as isolated from its external world. The choice of communication buses was not motivated by information security, but rather by safety and robustness issues. The Controller Area Network (CAN) imposed itself as the de facto communication bus for the automotive applications. It implements an approach known as multiplexing, which consists in connecting to the same wires (a bus) a large number of computers. The communication is orchestrated by the protocol. Almost all automotive manufacturers are implementing the CAN bus in their cars. The CAN protocol is used for periodic and event-based messages that allows the ECUs to monitor the vehicle state. It is also used to control and supervise the state of sensors and actuators.

Recently, the CAN protocol has become the center of multiple cyber-security issues [1, 4]. Miller and Valasek [16] showed how the CAN protocol can be an important attack vector that enables remote control of a vehicle. In this context, Hoppe et al. [8] were the first to point out the weaknesses of the CAN bus. These findings were further investigated and confirmed by Koscher et al. [13] and Checkoway et al. [1] that performed frame replay and frame injection attacks on a real vehicle. In these attacks, the attacker physically connects to the CAN network and replays or injects messages on the CAN bus. Given the fact that arbitrary read and write accesses are possible on the CAN network, the attacker who sends the right message with the right identifier and payload cannot be detected. Thus, the attacker message will be processed by all receiving ECUs giving her the ability to anonymously influence and control the behavior of these ECUs. This can greatly impact the safety of passengers.

Even though the CAN standard is used by almost all car manufacturers, each one implements its own messages depending on its own needs and the underlying Cyber-Physical Architecture. Each car manufacturer customizes the set of used message identifiers, payload information, and their periodicity. Hence the attacks, as the injection and replay attacks, are generally specific to a car manufacturer and not always reproducible to another car designed by another company.

The CAN frame does not contain a source and destination, but rather a frame identifier that indicates the “content of the data” conveyed by the frame. The same information is always sent over the same message identifier. For instance, the vehicle speed information coming from the speed sensor is always sent over the same message identifier in order for the receiving ECUs to recognize the message. This makes that some equipment have multi-architectures and are backwards compatible within the same car manufacturer. To overcome this issue, reverse engineering the protocol of a specific manufacturer is an important attack, which gives rise to other attacks against the target architecture. Precisely, the reverse-engineering attack is to identify the message identifiers, their periodicity, and the payload information before injecting messages. Miller et al. [15] show the difficulty to engage in such task. As this step seems particularly tedious given the number of messages used, there are some automatic and statistical tools, as penetration testing tools, that have emerged [19, 21]. These tools can lower the complexity of the reverse-engineering step and make it rather straightforward. Due to these weaknesses, the CAN bus has to be hardened in order to protect the connected car from potentially harmful attacks. Researchers have already proposed possible security countermeasures to protect the CAN bus from some of these attacks. Nevertheless these solutions are not fully satisfactory, as they have flaws which are presented in the next section.

In this chapter we make the following contributions:

- First, we identify state-of-the-art protection mechanisms for the CAN bus and highlight their respective flaws.
- Second, we develop identifier randomization procedures to protect the CAN network from reverse-engineering, replay, and injection attacks, both at software and hardware levels.
- Third, we propose information theory-based metrics to evaluate the proposed methods and to compare them with state-of-the-art solutions.

In what follows, Sect. 2 introduces the main state-of-the-art solutions dedicated to the CAN bus and their flaws. In Sect. 3, we focus on a particular solution, namely, the identifier randomization, and we propose novel randomization strategies both at software and hardware levels, to enhance the security of the CAN protocol. These protections are evaluated with information theoretic metrics. In Sect. 4 we compare the aforementioned randomization strategies with the state-of-the-art solutions. Finally Sect. 5 concludes the chapter.

2 State-of-the-Art CAN Protections

In this section, we identify the current solutions that are proposed to secure the CAN network from different security breaches. We start by explaining the CAN protocol stack; then we expose the principles of protection mechanisms that are payload protection, intrusion detection and prevention, and identifier protection. For each mechanism we identify the flaws and limitations.

2.1 Controller Area Network Overview

The CAN bus is an asynchronous, serial field bus system. It was introduced in 1983 by Bosch company for the networking of control devices in automobiles. The aim of this communication bus was the reduction of cable length and weight. Since 1991, CAN is internationally standardized as ISO 11898 and defines the Layer 2 (data link layer) [10] and Layer 1 (physical layer) [11, 12] of the OSI reference model presented in Fig. 1. The physical layer can be realized in two versions: high-speed CAN (ISO 11898-2) and low-speed CAN (ISO 11898-3). Usually these layers are implemented, respectively, in a CAN transceiver and a CAN controller. A CAN frame (Fig. 2) is composed of multiple fields. It starts with a Start-Of-Frame (SOF) bit, then arbitration field which is the frame identifier (ID), control field that indicates the length of data, data field, followed by a Cyclic Redundancy Check (CRC) field for integrity check and an acknowledgment field, and finally the End-Of-Frame (EOF) field. Each ECU that communicates on the CAN bus is equipped with a CAN controller and a CAN transceiver (Fig. 3). On the application level, whenever the ECU software wants to send a message on the CAN bus, it specifies the ID and the payload to the CAN controller (Layer 2). Then the CAN controller constructs the appropriate CAN frame by adding the remaining fields and sends it to the CAN transceiver (Layer 1) whose role is to physically send the frame on the communication bus.

The CAN bus is event-triggered protocol. Whenever a node needs to send a frame on the bus, it needs to check whether the bus is free; then it starts sending. Sometimes collision between two nodes trying to send information at the same time happens. The CAN protocol gives the priority to one of them according to the arbitration rule. The arbitration in the CAN protocol is decided during the sending of the frame identifier (or arbitration field) and is governed by the following rule.

Data Link (Layer 2)	ISO 11898-1	
Physical (Layer 1)	ISO 11898-2 [CAN High speed]	ISO 11898-3 [CAN Low speed]

Fig. 1 CAN layer model

S O F	Identifier	R T R	I D E	r	DLC	Data	CRC	ACK	EOF
1	11 bits	1	1	1	4 bits	0-8 bytes	16 bits	2	7 bits
	Arbitration Field	Control-Field				Data-Field	Check-Field		

Fig. 2 CAN frame

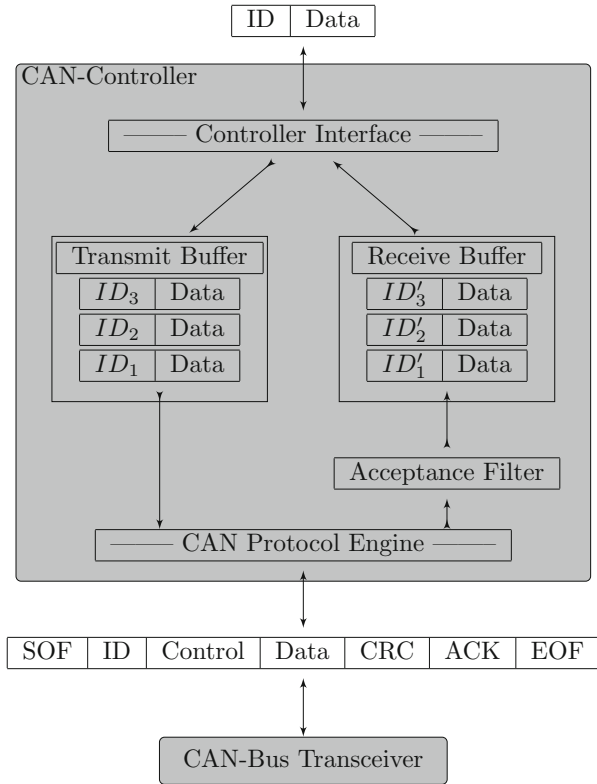


Fig. 3 CAN controller

If one node transmits a dominant bit (“0”) and another node transmits a recessive bit (“1”), then there is a collision and the dominant bit “wins.” Notice that the ID is sent bit by bit starting from the Most Significant Bit (MSB). Whenever there is a conflict between two ECUs trying to send different messages with different IDs, the smaller ID wins the arbitration and will be sent; the other will have to wait for the next frame. To send a frame to other nodes over the CAN bus, the ECU software needs to specify the ID and payload of the frame to the CAN controller (Layer 2). This information is temporarily stored in a buffer waiting to be sent on the bus. The CAN controller constructs the appropriate frame by adding the remaining fields and sends it to the CAN transceiver (Layer 1). To physically send the frame on the bus, the CAN transceiver needs to check if the bus is free (no information is being sent). Sometimes collision between two nodes trying to send frames at the same times happens. The CAN protocol gives the priority to the frame with lower ID.

Usually, for safety reasons, safety-critical signals are assigned to priority message identifiers. The more the signal is safety relevant, the higher the priority is assigned to its identifier.

2.2 *Payload Protection*

Historically, one of the first and obvious family of solutions that were proposed to secure the CAN bus is the payload protection solutions. In fact, if we consider the problem as an authenticity violation, the first step toward overcoming this weakness is to protect the payload with cryptographic mechanisms. Nilsson et al. [18] proposed to send message authentication codes over consecutive CAN frames to authenticate the messages. Hartkopp et al. [7] proposed to use Cipher-Based Message Authentication Code (CMAC) as a symmetric authentication measure between the sender and the receiver.

Flaws

The main limitation of this type of protections is that the produced data is larger than the original data which causes a bandwidth overhead on the communication link. For instance, if we want to use an encryption function to protect the confidentiality of the data, state-of-the-art encryption solutions suggest to use no less than 128-bit encryption key with a block cipher of minimum 64 bits (128 bytes for AES-128). This involves a significant increase of data size, as the data size is of 8 bytes or 16 bytes, even if the required payload is a few bytes. Similarly, if we want to protect the authenticity of the sent data, we have to send a data authentication code along with the original data. While the impact of this transformation could be negligible for only one message, the generalization of the use of such solutions to all the messages will cause a significant network overhead. This will have practical side effects such as increasing the delay of messages and increasing errors on the CAN network.

Furthermore, the payload protection mechanism does protect the confidentiality and integrity of the data, but does not protect against reverse-engineering and frame injection attacks. Since the identifiers are kept unchanged, the attacker can still reverse the messages and their periods. Also the attacker can perform exhaustion attack on the ECUs, by sending messages with correct identifier but wrong payload, thus forcing dummy and useless processing.

2.3 *Intrusion Detection and Prevention Systems*

Another family of protection solutions is known as CAN network Intrusion Detection and Prevention Systems (CAN-IDPS). The role of these systems is to monitor the CAN network for suspicious behavior like frame injection and replay attacks and either physically kill suspicious frame by causing a frame error or by filtering out the suspicious frames. In general, state-of-the-art detection mechanisms can be categorized into two main classes: *rule-based* detection mechanisms and *statistical* detection mechanisms.

Rule-based intrusion detection mechanisms tend to define specific rules of how the network traffic should be. Any message that does not comply with these predefined rules are reported as intrusions. Miller et al. [15] propose to define detection rules of diagnostics messages based on the state of the vehicle and detection rules of periodic messages based on their respective periods. In fact, since the goal of the CAN-IDPS is to protect against injecting extra packets onto the network, and given the fact that most normal frames have predefined frequencies, then if the particular message does not respect its frequency, it should be reported as an intrusion. Taylor et al. [20] propose a detection algorithm based on the comparison of previous and current frame timings to implement this principle. Another example of rule-based detection mechanism is proposed by Marchetti et al. [14] who use the CAN identifier sequence to detect possible injected frames.

Regarding *statistical* detection mechanisms, they define statistical measurements computed over a window of time and used to classify normal and suspicious behavior. In this context, an early work of Müter et al. [17] proposes to use the entropy of the CAN bus as a measurement to classify normal and abnormal behaviors observed on the CAN bus. Dario et al. [2] propose an intrusion detection algorithm that identifies anomalies by computing the Hamming distance between consecutive payloads. This Hamming distance is compared with minimum and maximum thresholds defined during set-up phase.

Flaws

On the one hand, statistical detection measurements do not allow to know the precise CAN frames which have been attacked. They report misbehavior detected on a relatively large time window, which means that the attacker can always inject and replay frames. On the other hand, rule-based detection mechanisms are more effective as they allow only for compliant packets to be accepted. In practice these rules have to be more flexible due to communication imperfections, and the attacker can use this flexibility for her own benefit. If we take the example of frequency-based detection, with a message identifier ID and with a periodicity p , we define a rule that accepts only one packet of identifier ID within a time window p . All the other messages with the same identifier will have to be filtered out or killed. It follows that once the algorithm is synchronized with a legitimate first frame at t_0 , the next frames that arrive at $t_0 + n \times p$ will be accepted; all the others will be blocked. In practice the periodicity is not fixed and is subject to a certain variability of approximately 10%. If the attacker injects a frame close to the legitimate frame, the detection algorithm cannot know which one is the legitimate frame. Besides these flaws, neither statistical nor rule-based IDPS, do not protect against reverse engineering of the protocol.

2.4 Identifier Protection

The identifier protection family is efficient to protect the CAN bus from reverse-engineering, injection, and replay attacks. The idea behind this security principle is to make the identifiers not fixed, but instead constantly changing. In fact, if the identifier is not fixed across vehicles, a large-scale attack that could affect all the cars is no longer possible as the identifiers of messages will change from one vehicle to another. Moreover, if the identifiers are not fixed within the same vehicle, a frame replay attack will not succeed because the identifier is constantly changing, and thus no ECU will catch the replayed frame. A frame injection attack neither will work, because the attacker will have to “predict” what will be the next identifier to be injected in order for the attack to be successful.

Humayed et al. [9] proposed a solution called ID-Hopping to counter DOS (Denial-Of-Service) attacks directed against a specific message. Their method works closely with an intrusion detection mechanism which is needed to identify the existence of an attack against a specific message. Once the attack is detected, the ID-Hopping mechanism is activated. Its role is to assign a new but previously defined identifier as a substitute identifier for the attacked message. Another interesting solution to protect the CAN protocol is to constantly randomize the identifier. The constraint is that both the sender and receiver share the same identifier. Han et al. [5, 6] proposed a candidate randomization function. To the best of our knowledge, this solution is the only one that has been proposed for this purpose in the state of the art.

Flaws

While it effectively protects against replay and injection attacks, this randomization principle is not efficient enough to protect against reverse engineering. In the next section, we expose and analyze in details this solution and propose an enhanced protection.

3 Solutions Based on Randomization and Their Evaluation

In this section we focus on the identifier protection family, precisely those based on randomization functions which present the best security features. We first identify the main characteristics and constraints of the randomization function that have to be used for protection. Then we analyze the state-of-the-art solutions and propose new functions that offer better protection from the security point of view. The evaluation of these different solutions is done by defining formal security metrics coming from the information theory.

3.1 Principle and Formalism

The way the CAN protocol is used today by car manufacturers is the following: each information that needs to be communicated between two ECUs is sent in a CAN frame. Figure 4 illustrated the CAN protocol and the histogram of the identifiers. Each frame has a fixed identifier which is known by the sender and the receiver. It can also be known to the attacker as it is communicated in clear on the CAN bus. The identifiers are fixed during the design phase of the vehicle and respond to priority criteria for safety reasons. The priority level defines the criticality of the information and allows the CAN protocol arbitrates between concurrent messages. It is directly linked to the ID value: the lower the ID, the greater the priority of the associated message.

As explained in Sect. 1, the fact that the same information is always sent over the same frame identifier enables the attacker to reverse the protocol and forge frames that can be accepted by the vehicle ECUs. The attacker first starts with a reverse-engineering step during which she identifies the message identifiers and their frequencies. Then she builds an attack by injecting, or replaying, one or multiple CAN frames.

In order to protect the CAN network from such attacks, we want to change the message identifiers every time the ECU needs to send information. This should be done in a way that the receiving ECUs can recover the original identifier and do not allow the attacker to reverse the protocol or inject messages that can be accepted by other ECUs. To do so, an identifier randomization function F is added in such a way that it takes the original identifier ID and substitutes it with another identifier ID_r that changes at every occurrence m of a new frame on the CAN bus. The index m is the value of a message counter which has to be embedded in every ECU for consistency reasons, as m is not communicated on the CAN bus:

$$ID_r = F(ID, m) \tag{1}$$

At the receiver side, the ID is recovered by using the inverse function of F and F^{-1} s and the value m of the internal counter of the ECU:

$$ID = F^{-1}(ID_r, m) \tag{2}$$

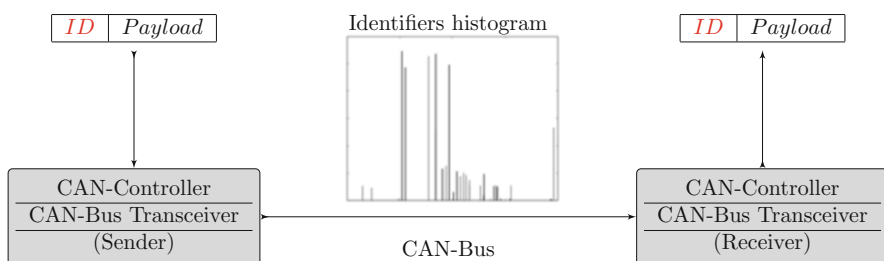


Fig. 4 Controller Area Network with original identifier distribution

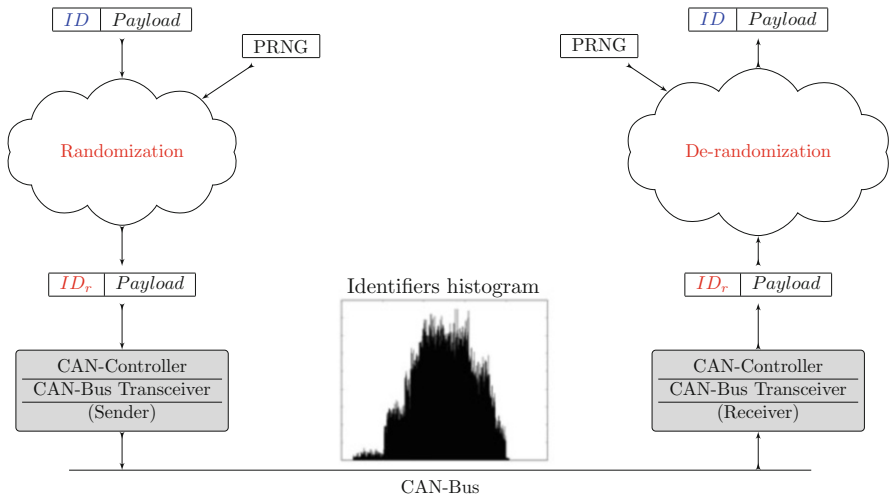


Fig. 5 CAN-ID randomization principle

The randomization function F has to satisfy certain conditions in order to be effective:

- First, F has to be injective and efficiently computable in order for the receiver to recover rapidly the original identifier. Figure 5 shows how this function could be integrated. We can see in this figure that the histogram of randomized identifier ID_r is more spread compared to the one in Fig. 4.
- Second, and for safety reasons, the function F has to be priority-preserving. This means that the priority of message identifiers ID_1 and ID_2 has to be the same as $F(ID_1)$ and $F(ID_2)$, respectively. This boils down to the following condition:

$$ID_1 < ID_2 \Rightarrow F(ID_1, m) < F(ID_2, m) \tag{3}$$

- Third, the priority condition has to be preserved over time. Indeed, the message can go through a transmission buffer before being physically sent to the bus. Consequently, the state of every ECU counter m can be different from the real number of transaction counter on the physical layer. In order to be consistent, the randomization function has to guarantee that the identifiers keep their priority even if the transaction counter m is different. This is expressed by:

$$ID_1 < ID_2, m_1 < m_2 \Rightarrow F(ID_1, m_1) < F(ID_2, m_2), F(ID_1, m_2) < F(ID_2, m_1) \tag{4}$$

- Fourth, the output of the randomization function F has to be unpredictable. An attacker that has some information about previous outputs or identifiers should not be able to predict with high probability the randomized identifier. We achieve

this goal by choosing a randomization function based on a cryptographically secure pseudorandom number generator (PRNG).

3.2 Evaluation Metrics

Many functions with randomization can meet the previous constraints. In order to compare these functions between them, we need to define security metrics that measure the ability of these functions to protect against reverse-engineering and replay/injection attacks. These metrics are based on information theory, which links them to *optimal attacks*, that is, attacks which maximize the likelihood of success [3].

3.2.1 Reverse-Engineering Attack

In the presence of a randomization scheme, the attacker knows that each original identifier has multiple substitute identifiers. The reverse-engineering challenge is to be able to determine for each original identifier the set of substitute identifiers that it could be randomized into. A randomization scheme is perfect if the resulting randomized identifiers are identically distributed over the set of identifiers. From an information theory point of view, the capacity of the attacker to perform this task is related to the entropy of the resulting randomized identifiers. The more the identifiers look random, the harder it is for the attacker to reverse the protocol. Thus we use the entropy as a security metrics to evaluate the protection level of the randomization function against reverse engineering:

$$H(id_r) = \sum_{x \in [0, 2^n - 1]} P(id_r = x) \times \log_2 \left(\frac{1}{P(id_r = x)} \right). \quad (5)$$

3.2.2 Replay and Injection Attacks

In order for the attacker to successfully inject a message on the CAN bus with the presence of a randomization function, she needs to “predict” the next randomized identifier to be sent. If the attacker successfully conducts a reverse-engineering attacks, she should be able to predict the next original identifier to be sent. Knowing the original identifier, the attacker has to predict its randomized version. Since we suggested to combine the randomization function with a cryptographically secure pseudorandom number generator that has a uniform distribution, we suppose that the prediction capability of the attacker is not better than a simple “guess.” Thus, the conditional entropy of the randomized identifier knowing the original identifier can be used as a metrics to evaluate the protection level of the randomization function

against replay and injection attacks:

$$H(id_r|id_o) = \sum_{x \in [0, 2^n - 1]} P(id_r = x|id_o) \times \log_2 \left(\frac{1}{P(id_r = x|id_o)} \right). \quad (6)$$

3.3 The IA-CAN Approach

In [5, 6] Han et al. gave a method for identifier randomization of the CAN protocol called Identity-Anonymized CAN (IA-CAN). Their approach is to mix a part of the identifier (LSB part) and a part of the payload with a random variable generated at sender and receiver sides. Here we want to focus only on the randomization of the CAN identifier. This is motivated by the fact that if the attacker successfully injects an identifier that gets passed through the CAN filter, even if the rest of the payload is not correct, it will nevertheless exhaust the receiver ECU.

If we disregard the payload part of the anonymization in the IA-CAN approach, we can conclude that the randomization function being used is as follows: (We refer the reader to the original paper [5, 6] for further details.)

$$\begin{aligned} f_r : [0, 2^a - 1] \times [2^a, 2^n - 1] &\rightarrow [0, 2^n - 1] \\ r \quad \quad \quad id &\rightarrow id_{MSB(n-a)} + id_{LSB(a)} \oplus r \end{aligned} \quad (7)$$

where

- n is the number of bits of the identifier ($n = 11$ for standard CAN, $n = 29$ for extended CAN).
- a is the number of bits that will be used for randomization ($a < n$).
- r is a random variable in $[0, 2^a - 1]$ generated at both sender and receiver sides.
- id is the original identifier of the message.
- $id_{MSB(a)}$ is the identifier α Most Significant Bit.
- $id_{LSB(a)}$ is the identifier α Least Significant Bit.

We assume that the random number r is uniformly distributed over the randomization interval $[0, 2^a - 1]$. Figure 6 shows the principle of the transformation applied to the identifiers.

Obviously the choice of the variable a is bounded by the total number of original used identifiers N and the minimum of interspace between all identifiers:

$$1 \leq a \leq \lfloor \log_2(\text{Min}_{i,j \in [1,N]} |id_i - id_j|) \rfloor \quad (8)$$

In order to maximize the randomness of the identifiers, we have to choose the maximum possible a : this means that for better security performance, we have to choose:

$$a = \lfloor \log_2(\text{Min}_{i,j \in [1,N]} |id_i - id_j|) \rfloor \quad (9)$$

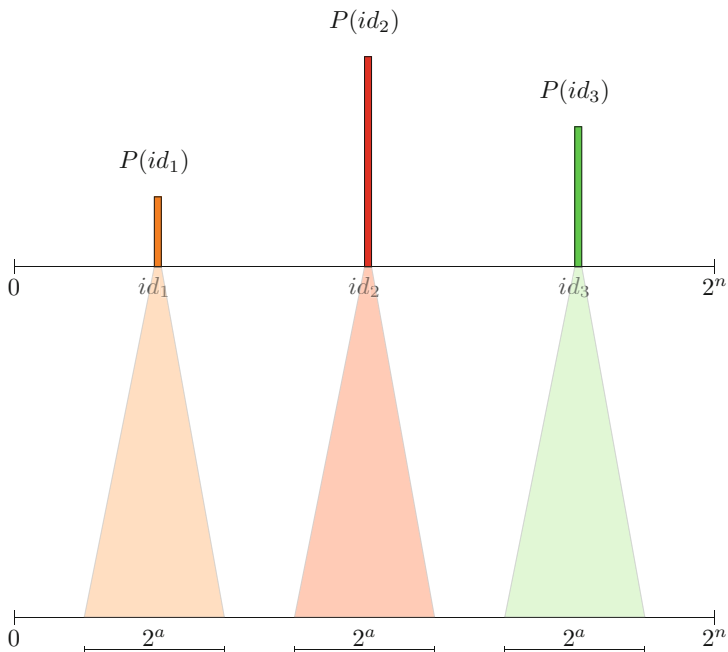


Fig. 6 IA-CAN identifier transformation

3.3.1 Particular Case

A particular case arises when the identifier interspace is constant between all original identifiers. The constant is then $\frac{2^n}{N}$. The upper bound of a is then expressed as the following:

$$\lceil \log_2(\text{Min}_{i,j \in [1,N]} |id_i - id_j|) \rceil = n - \lceil \log_2(N) \rceil \tag{10}$$

To measure the efficiency of this randomization function, we compute the entropy of the randomized identifiers:

$$H_{\text{IA-CAN}}(id_r) = H(id_o) + a \tag{11}$$

And to quantify the attacker capacity to inject new frames, we compute the conditional entropy of the randomized identifiers knowing the original identifiers:

$$H_{\text{IA-CAN}}(id_r | id_o) = a = \lceil \log_2(\text{Min}_{i,j \in [1,N]} |id_i - id_j|) \rceil \tag{12}$$

In case the identifier interspace is constant:

$$H_{\text{IA-CAN}}(id_r | id_o) = a = n - \lceil \log_2(N) \rceil \tag{13}$$

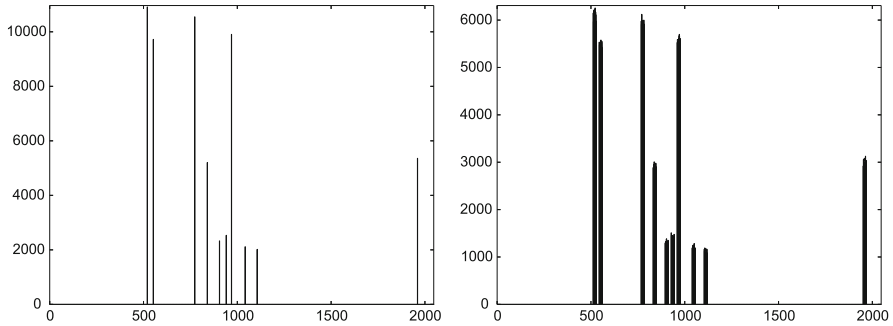


Fig. 7 IA-CAN transformation: Original (left) and randomized (right)

Proof of Eqs. (11) and (12) is presented in the Appendix.

3.3.2 Testing

To test this approach, we made a real acquisition on a vehicle CAN bus, which we used with this randomization procedure to assess its efficiency. Figure 7 shows the identifier histograms before and after randomization. On this particular example, the randomization was done over $a = 4$ bits which means that for each identifier, we allocated $2^a = 2^4 = 16$ substitute identifiers. The computed entropy of the original distribution is $H(id_o) = 3.05$. After randomization, the computed entropy of the randomized identifiers is $H_{IA-CAN}(id_r) = 7.05$. The computed conditional entropy is $H_{IA-CAN}(id_r|id_o) = 4$. We can observe from the randomized identifier distribution of Fig. 7 that the attacker can still distinguish frequencies of the messages. It is also clear from Eq. (11) that the entropy of randomized identifiers depends on the entropy of the original identifiers. Using this information the attacker can deduce the next original identifier to be sent and try to inject a frame within the observed randomization interval.

3.4 Equal Intervals

The first observation that we can make concerning the IA-CAN approach is that there is still room for amelioration in terms of entropy and conditional entropy. In fact, the randomization of IA-CAN is done only on the a least significant bits of the identifier, which makes the added entropy bounded by a which is also bounded by $\log_2(\text{Min}_{i,j \in [1,N]} |id_i - id_j|)$

A first possible improvement is to create a mapping function that assigns to each original identifier a substitute identifier. The set of substitute identifiers should satisfy the equidistance condition, mentioned in the previous section, that

maximizes random space. A second improvement is to change the randomization function from an XOR function to an arithmetic addition in order to increase the randomness and thus the entropy.

If we have N identifiers $id_1 < id_2 < \dots < id_N$, we have to partition the identifier space $[0, 2^n - 1]$ over N intervals $I_i = [inf_i, sup_i]$ such that

- $inf_1 = 0, sup_N = 2^n - 1$
- For each $i \in [1, N - 1] : inf_{i+1} = sup_i + 1$
- For each $i \in [1, N - 1] : sup_i - inf_i = const = \frac{2^n}{N}$

Thus the identifier mapping function Map :

$$\begin{aligned}
 Map : [0, 2^{n-1} - 1] &\rightarrow [0, 2^n] \\
 id_i &\rightarrow inf_i
 \end{aligned}
 \tag{14}$$

The Map function is designed to redefine the distribution of the identifier (by assigning a substitute identifier to the original one) in such a way that the new identifiers maximize the identifier interspace.

Given this new distribution, in the interval $I_i = [inf_i, sup_i]$, we have only one identifier id_i ; we can exploit all the interval to randomize that identifier.

Thus the randomization function:

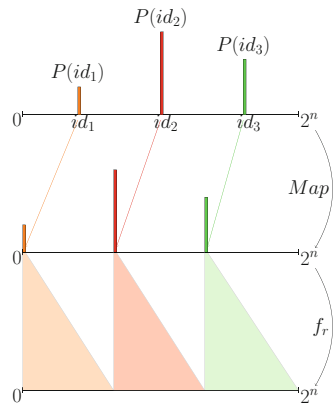
$$\begin{aligned}
 f_r : [0, 2^{n-1} - 1] &\rightarrow [0, 2^n] \\
 id_i &\rightarrow Map(id_i) + r_{[0, sup_i - inf_i]}
 \end{aligned}
 \tag{15}$$

Figure 8 shows the transformation applied to the identifiers. To compare this proposed solution to the previous one, we compute the proposed security metrics:

Entropy:

$$H_{EI}(id_r) = H(id_o) + n - \log_2(N)
 \tag{16}$$

Fig. 8 Equal interval identifier transformation



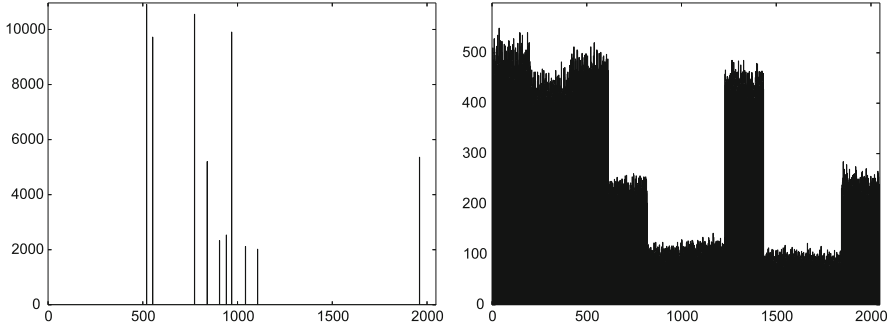


Fig. 9 Equal interval transformation: Original (left) and randomized (right)

Conditional entropy:

$$H_{EI}(id_r|id_o) = n - \log_2(N) \quad (17)$$

Proof of Eqs. (16) and (17) is presented in the Appendix.

In Sect. 4 we show that based on theoretical analysis of the proposed metrics, this randomization function is more secure than the state-of-the-art solution.

3.4.1 Testing

The randomization function is applied to the same identifier distribution used in the previous section. Figure 9 shows the identifier histograms before and after randomization.

We can see that compared to the IA-CAN approach, the equal interval randomization function (15) exploits all the available identifier space. The computed entropy of this particular example is $H_{EI}(id_r) = 10,72$. Compared to the IA-CAN randomization function ($H_{IA-CAN} = 7.05$), the equal interval randomization function generates more entropy. The conditional entropy of the randomized identifier knowing the original identifier is $H_{EI}(id_r|id_o) = 7.67$. We can also observe that compared to IA-CAN ($H_{IA-CAN}(id_r|id_o) = 4$), it has better conditional entropy. In Sect. 4, we formally prove that it is always the case. Nevertheless the attacker can still identify clusters of randomized identifiers that can guide him in the reverse-engineering process even if the probability of a successful injection is slightly smaller than the previous solution.

3.5 Frequency Intervals

The previous methods are not secure enough against reversing the original identifiers and periods. Indeed, given the histogram, the attacker can identify clusters

of identifiers and thus can identify the original identifier and its frequency. In this section we design a new randomization function whose aim is to overcome this limitation. The goal of this function is to make the randomized identifier distribution [histogram] as uniform as possible to improve the entropy of the randomized identifier by still preserving the priority order. In order to do that, a flattening of the peaks has to be done. We choose the randomization interval of each identifier to be proportional to its frequency of appearance on the CAN bus. Thus an identifier that has a high frequency (small period) will appear more frequently on the CAN bus; this identifier will be assigned a large interval of randomization. Similarly, an identifier that has a small frequency (large period) of appearance on the CAN bus will appear less frequently and thus will be assigned a small interval of randomization. In order for this strategy to be possible, we also need a mapping function that assigns substitute identifiers to the original identifier. Then we apply the randomization to the substitute identifier.

Suppose there are N identifiers $id_1 < id_2 < \dots < id_N$, respectively, with a sending frequency of f_1, f_2, \dots, f_N . We have to partition the identifier space $[0, 2^n - 1]$ over N intervals $I_i = [inf_i, sup_i]$ such that:

- $inf_1 = 0, sup_k = 2^n - 1$
- For each $i \in [1, N - 1] : inf_{i+1} = sup_i + 1$
- For each $i \in [1, N - 1] : sup_i - inf_i = \frac{2^n \times f_i}{\sum_{j=1}^N f_j} = P(id_i) \times 2^n$

where $P(id_i)$ is the probability of the identifier id_i to appear on the CAN bus.

We define an identifier mapping function Map that assigns substitute identifiers to the original ones such that the identifier interspace is proportional to the frequency of the smaller identifier:

$$\begin{aligned} Map : [0, 2^{n-1} - 1] &\rightarrow [0, 2^n] \\ id_i &\rightarrow inf_i \end{aligned} \quad (18)$$

The randomization function then assigns a randomized identifier to the substitute identifier. Each identifier is randomized in an interval proportional to its frequency:

$$\begin{aligned} f_r : [0, 2^{n-1} - 1] &\rightarrow [0, 2^n] \\ id_i &\rightarrow Map(id_i) + r_{[0, sup_i - inf_i]} \end{aligned} \quad (19)$$

Figure 10 shows the transformation applied to the identifiers.

To compare this proposed solution to the previous one, the proposed security metrics is computed:

Entropy:

$$H_{FI}(id_r) = n \quad (20)$$

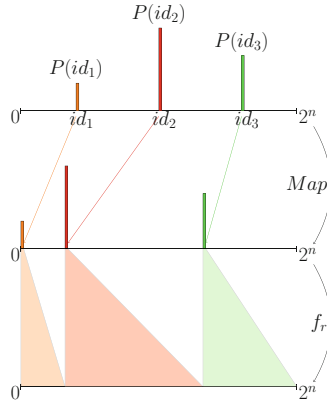


Fig. 10 Frequency interval identifier transformation

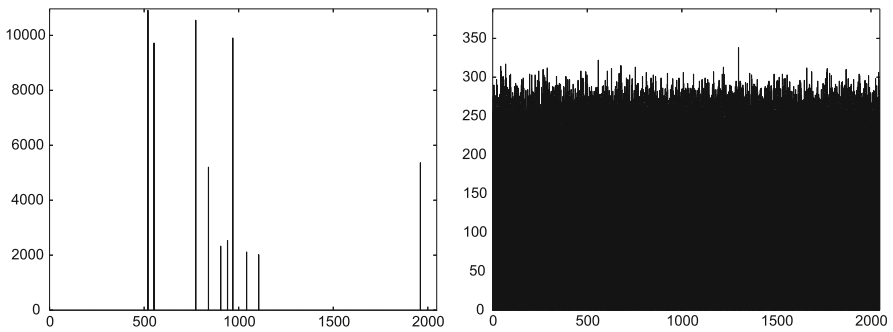


Fig. 11 Frequency interval transformation: Original (left) and randomized (right)

Conditional entropy:

$$H_{FI}(id_r | id_o) = n - H(id) \tag{21}$$

A first observation is that in terms of theoretical entropy, this solution reaches the maximum entropy which is n . Another interesting result is that it gives an enhancement of the conditional entropy as it is shown in Sect. 4. From a theoretical analysis, it is shown in Appendix section “Fixed Mapping Optimality Proof” that the maximum conditional entropy is optimal if the mapping is constant. We will see in the next section that a dynamic mapping will increase the conditional entropy.

3.5.1 Testing

To test this randomization strategy, we apply it to the identifier distribution used for the previous functions. Figure 11 shows the identifier histograms before and

after randomization. The computed entropy for this example is $H_{FI}(id_r) = 10.99$. The computed conditional entropy is $H_{FI}(id_r|id_o) = 7.94$. It is clear from the histogram and the computed entropy that the randomized identifier distribution is more uniform than the previous solutions. A uniform distribution of identifiers is a perfect protection against reverse engineering as it is harder for the attacker to distinguish clusters of identifiers. We can also observe that there is an enhancement in terms of conditional entropy compared to the previous solutions. That is to say, this solution has better security performance against injection attacks.

3.6 Dynamic Intervals

We can now raise the question to increase the conditional entropy obtained with the Frequency Intervals, by using a dynamic mapping. A practical observation of the CAN bus behavior shows that there is a strong dependency between consecutive identifiers: the majority of identifiers will have zero probability to appear right after id_i . This observation involves that using a fixed identifier mapping, after that identifier id_i has been sent, an important part of the allocated space for identifiers will not be used. Hence, if the mapping is changed dynamically after every sending of id_i , and according to the dependency between identifiers, we can increase the conditional entropy.

To construct such randomization function, the Markov matrix can be built to give the probabilities $p_{i,j} = P(id_j^{t+1}/id_i^t)$ of receiving an identifier id_j at iteration $t + 1$ knowing that we received the identifier id_i at iteration t :

$$M = \begin{pmatrix} \cdot & id_1^{t+1} & id_2^{t+1} & \dots & id_j^{t+1} & \dots & id_n^{t+1} \\ id_1^t & p(id_1^{t+1}/id_1^t) & p(id_2^{t+1}/id_1^t) & \cdot & \cdot & \dots & p(id_n^{t+1}/id_1^t) \\ id_2^t & p(id_1^{t+1}/id_2^t) & p(id_2^{t+1}/id_2^t) & \cdot & \cdot & \dots & p(id_n^{t+1}/id_2^t) \\ id_3^t & \cdot & \cdot & \cdot & \cdot & \dots & \cdot \\ \vdots & \cdot & \cdot & \cdot & \cdot & \dots & \cdot \\ id_i^t & \cdot & \cdot & \cdot & p(id_j^{t+1}/id_i^t) & \dots & \cdot \\ \vdots & \cdot & \cdot & \cdot & \cdot & \dots & \cdot \\ id_n^t & \cdot & \cdot & \cdot & \cdot & \dots & \cdot \end{pmatrix} \tag{22}$$

Each time we receive an identifier id_i , immediately after, we have $p(id_j^{t+1}/id_i^t)$ probability to receive id_j . With this in mind, we opt for the Frequency Interval strategy to randomize the upcoming identifiers since it is the optimal strategy that guarantees the maximal entropy. We keep updating the interval partition according to Frequency Interval strategy and to the probabilities in the matrix.

We define the identifier mapping function as the following:

$$\begin{aligned} \text{Map}^{t+1} : [0, 2^n - 1] &\rightarrow [0, 2^n - 1] \\ id_{i+1} &\rightarrow id_i + 2^n \times p_{k,i} \end{aligned} \quad (23)$$

The *Map* function has to be updated every received identifier according to the Frequency Interval strategy. At an instant $t + 1$, knowing that the previous sent identifier is id_k , identifier id_i has an assigned randomization interval of I_i of width $W(I_i) = 2^n \times p_{k,i}$. The resulting randomization function is the following:

$$\begin{aligned} f_r^{t+1} : [0, 2^n - 1] &\rightarrow [0, 2^n - 1] \\ id_i &\rightarrow \text{Map}^{t+1}(id_i) + r_{[0, 2^n \times p_{k,i}]} \end{aligned} \quad (24)$$

3.6.1 Illustrative Example

As an example, consider the following sequence of identifiers appearing on the CAN bus: $[id_2, id_3, id_1, id_2, id_3, id_1, id_2, id_3, id_2, id_1, id_2, id_3, id_2, id_1, id_2]$.

After analyzing the sequence, the following transition matrix can be established:

$$M = \begin{pmatrix} . & id_1^{t+1} & id_2^{t+1} & id_3^{t+1} \\ id_1^t & 0 & 1 & 0 \\ id_2^t & \frac{1}{3} & 0 & \frac{2}{3} \\ id_3^t & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix} \quad (25)$$

This transition matrix is used to define new mapping upon reception of a new identifier. Figure 12 shows the transformation applied to the identifiers after reception of id_2 and then id_3 .

The security metrics obtained with the Dynamic Interval strategy is the following:

Entropy:

$$H_{DI}(id_r) = n \quad (26)$$

Conditional entropy:

$$H_{DI}(id_r^{t+1}|id_o^{t+1}) = \sum_{x \in [0, 2^n]} \sum_{id_j^{t+1}} \sum_{id_i^t} \frac{1}{W(I_{i,j})} P(id_i) \log_2 \left(\frac{1}{\sum_{id_k^t} \frac{1}{W(I_{k,j})} P(id_k)} \right) \quad (27)$$

3.6.2 Testing

To test this randomization strategy, we apply it to the identifier distribution used for the previous functions. The computed entropy for this example is $H_{DI}(id_r) = 11$.

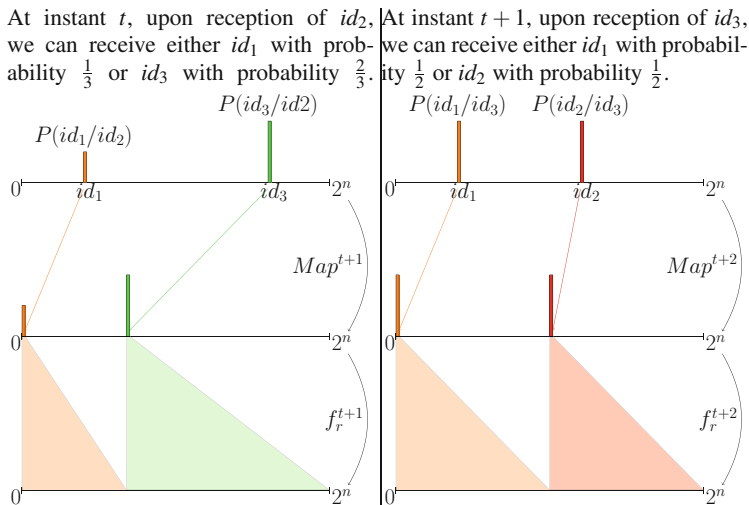


Fig. 12 Dynamic interval identifier transformation at $t + 1$ (left) and $t + 2$ (right)

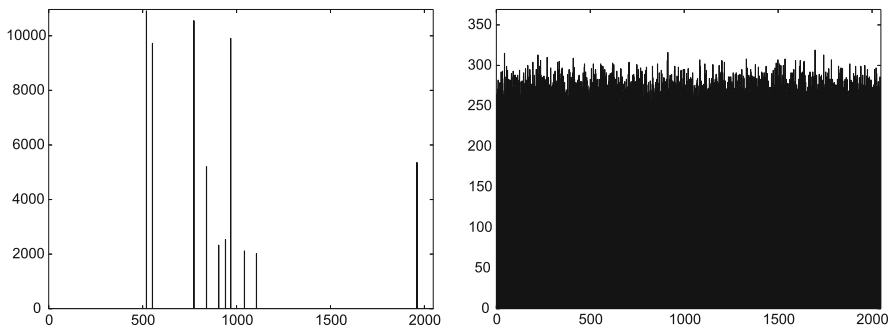


Fig. 13 Dynamic interval transformation: Original (left) and randomized (right)

The computed conditional entropy is $H_{DI}(id_r | id_o) = 10.24$. It is clear from the histogram and the computed entropy that the randomized identifier distribution is uniform as for the frequency interval randomization strategy. Moreover, this method provides a significant enhancement in terms of conditional entropy, compared to the previous solutions (Fig. 13).

3.7 Arithmetic Masking

All of the above-proposed solutions can be applied at software level (Layer 3). This subsection considers a hardware solution which can involve some change in the CAN controllers. The payoff of this choice is to eliminate the third constraint

imposed on Sect. 3.1 that states that the randomization function has to preserve priority over time. Here we consider that the new hardware at physical layer does not have any frame buffer. Hence there is a possibility that all the CAN controllers share the same random variable in a consistent manner. The internal changes of this random variable could be done by a pseudorandom number generator (PRNG) which is initialized identically in every CAN controller at start-up.

The hardware randomization proposal is based on Arithmetic Masking, meaning that the random variable is added arithmetically to the base identifier. The operations are the following:

- First a mapping function is defined. It assigns new substitute identifiers to the original identifiers.
- Then the randomization is performed by adding the random variable to the substitute identifier.
- The random variable is such that it is shared with all CAN controllers and the randomized identifier does not exceed 2^{11} . This allows to preserve the priority between identifiers.

Suppose there are N identifiers $id_1 < id_2 < \dots < id_N$, with a sending frequencies of f_1, f_2, \dots, f_N . A substitute and random identifier is assigned for each original identifier. The identifier mapping function is defined as the following:

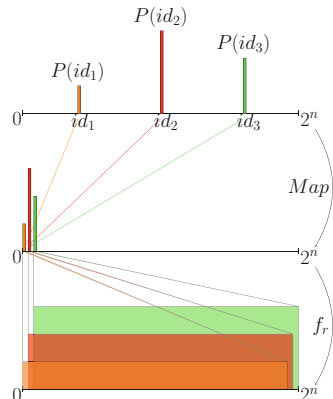
$$\begin{aligned} Map : [0, 2^{n-1} - 1] &\rightarrow [0, 2^n] \\ id_i &\rightarrow i - 1 \end{aligned} \tag{28}$$

The mapping function substitutes the original identifiers with the N first lowest identifiers. The rest of interval $[N, 2^n]$ is used for randomization:

$$\begin{aligned} f_r : [0, 2^n - N] \times [0, 2^{n-1} - 1] &\rightarrow [0, 2^n] \\ r &id_i \rightarrow Map(id_i) + r \end{aligned} \tag{29}$$

Figure 14 shows the transformation applied to the identifiers.

Fig. 14 Arithmetic masking identifier transformation



The security metrics applied to the Arithmetic Masking solution give the following results:

Entropy:

$$\begin{aligned}
 H_{AM}(id_r) &= \log_2(2^n - N + 1) + \frac{1}{2^n - N + 1} \sum_{x \in [0, N-2]} \sum_{i=0}^x P(id_i) \\
 &\quad \times \log_2 \left(\frac{1}{\sum_{i=0}^x P(id_i)} \right) \\
 &\quad + \sum_{i=x+1}^{N-1} P(id_i) \times \log_2 \left(\frac{1}{\sum_{i=x+1}^{N-1} P(id_i)} \right)
 \end{aligned}$$

Conditional entropy:

$$H_{AM}(id_r | id_o) = \log_2(2^n - N + 1) \quad (30)$$

3.7.1 Testing

To test this randomization strategy, we apply it to the identifier distribution used for the previous functions. The computed entropy for this example is $H_{AM}(id_r) = 10.99$. The computed conditional entropy is $H_{AM}(id_r | id_o) = 10.99$. The histogram and the computed entropy show that the randomized identifier distribution is approximately uniform. Moreover, we observe a significant enhancement in terms of conditional entropy compared to the previous solutions (Fig. 15).

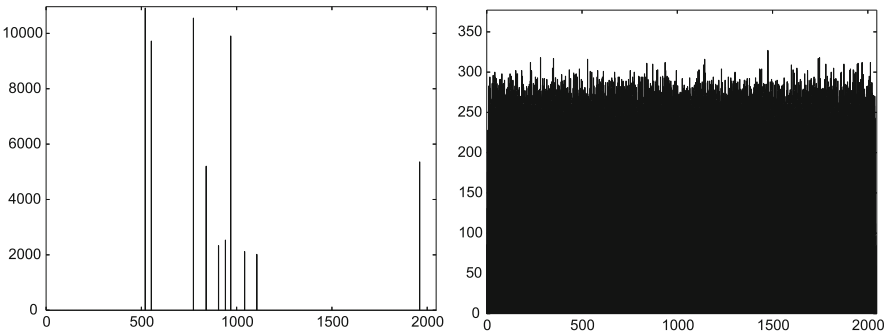


Fig. 15 Arithmetic masking transformation: Original (left) and randomized (right)

4 Comparison

In the previous section, we introduced the state-of-the-art solution for CAN identifier randomization, and we proposed solutions both at software and hardware layers. These solutions were tested on a real identifier trace captured from a real vehicle. In this section a comparison of the proposed solutions is applied to more identifier distributions by using the proposed security metrics.

Four reference identifier distributions are considered. Table 1 summarizes the obtained results. The visual inspection of the histograms indicates that frequency interval and dynamic interval randomization strategies have more uniform distribution than equal intervals and IA-CAN. Hence, these solutions should offer better protection against reverse-engineering attack, at first glance. This observation can be theoretically proven. By comparing the closed-form expressions of the respective entropies, we can establish the following:

$$H_{IA-CAN}(id_r) \leq H_{EI}(id_r) \leq H_{FI}(id_r) = H_{DI}(id_r) \quad (31)$$

Proof

$$H(id_o) \leq \log_2(N)$$

And we can establish from Eqs. (8) and (10) that:

$$\begin{aligned} a &\leq n - \lceil \log_2(N) \rceil \leq n - \log_2(N) \\ &\Rightarrow H(id_o) + a \leq H(id_o) + n - \log_2(N) \\ &\Rightarrow H_{IA-CAN}(id_r) \leq H_{EI}(id_r) \end{aligned}$$

Since







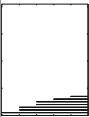
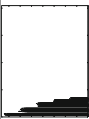
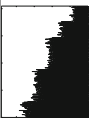



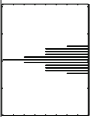





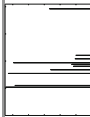


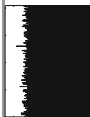


$$H_{DI}(id_r) = H_{FI}(id_r) = n$$

Then:

$$H_{IA-CAN}(id_r) \leq H_{EI}(id_r) \leq H_{FI}(id_r) = H_{DI}(id_r)$$

It is clear that compared to Arithmetic Masking, Dynamic Intervals and Frequency Intervals have better performance in terms of entropy, involving a high robustness against reverse engineering. Comparing the Arithmetic Masking to IA-CAN and Equal Intervals is not trivial. This is mainly because established entropy expressions depend on the entropy of the original identifier distribution. If we consider that the original identifiers have equal probabilities (example of the first distribution), the Equal Interval solution has better entropy ($H_{EI}(id_r) = 10.9997$, $H_{AM}(id_r) = 10.9948$). Theoretically, the entropy of Equal Intervals for this example is maximal.

Table 1 Comparison between different randomization strategies based on identifier histograms (the X axis represents the identifiers space $[0, 2^{11}-1]$, the Y axis represents the occurrence count of identifiers)

Reference distribution	IA-CAN	Equal intervals	Frequency intervals	Dynamic intervals	Arithmetic masking
 $H(i_{d_0}) = 2.80$	 $H(i_{d_r}) = 7.80$	 $H(i_{d_r}) = 10.99$	 $H(i_{d_r}) = 10.99$	 $H(i_{d_r}) = 10.99$	 $H(i_{d_r}) = 10.99$
 $H(i_{d_0}) = 2.68$	 $H(i_{d_r}) = 7.68$	 $H(i_{d_r}) = 10.86$	 $H(i_{d_r}) = 10.99$	 $H(i_{d_r}) = 11$	 $H(i_{d_r}) = 10.99$
 $H(i_{d_0}) = 3.35$	 $H(i_{d_r}) = 8.35$	 $H(i_{d_r}) = 10.88$	 $H(i_{d_r}) = 10.99$	 $H(i_{d_r}) = 11$	 $H(i_{d_r}) = 10.99$
 $H(i_{d_0}) = 3.05$	 $H(i_{d_r}) = 7.05$	 $H(i_{d_r}) = 10.72$	 $H(i_{d_r}) = 10.99$	 $H(i_{d_r}) = 11$	 $H(i_{d_r}) = 10.99$

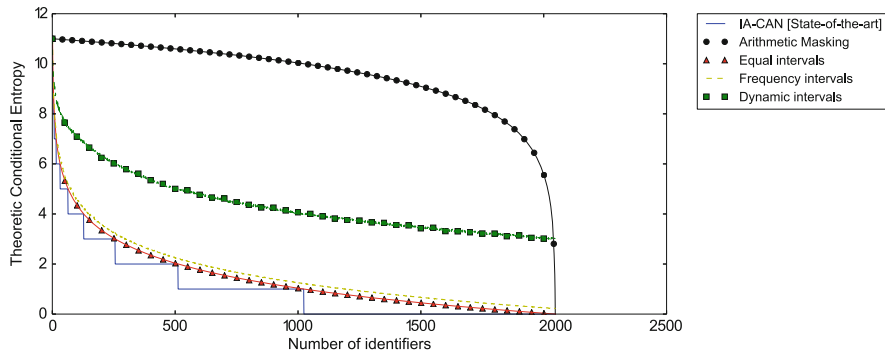


Fig. 16 Conditional entropy $H(id_r | id_o) = f(N)$

On the other side, concerning the second distribution, we can observe that the Arithmetic Masking performs better.

To compare the protection level against replay and injection attacks, the conditional entropy metrics is used. Based on the closed-form expressions established in the previous sections, we draw the curve showing the evolution of the conditional entropy as a function of the total number of identifiers. Figure 16 shows the results. From this graph we can conclude that all the proposed solutions outperform the IA-CAN strategy. Second, it appears that the hardware-based solution, namely, Arithmetic Masking, is the best against replay and injection attacks. However, as discussed previously, the Arithmetic Masking needs to be implemented in the CAN controller between the physical and data link layer, which makes it not easy to deploy. At software level, the Frequency Interval strategy performs the best, both against replay and injection attacks and against reverse engineering.

5 Conclusion

This chapter first presents the state-of-the-art solutions to protect the CAN bus from possible malicious attacks, namely, reverse-engineering, frame injection, and frame replay attacks. It appears that one of the most efficient classes of protection is based on the randomization of the CAN identifiers. Starting from the existing Identity-Anonymized CAN (IA-CAN), three major enhancements based on randomization have been proposed: with Equal Intervals, Frequency Intervals, and Dynamic Intervals. In case it is possible to change the CAN hardware, a randomization based on Arithmetic Masking has also been introduced. The security assessment has been carried out by using security metrics coming from the information theory: entropy (for the reverse-engineering attack) and conditional entropy (for the replay and injection attacks). It has been shown that the enhanced protections provide a significant gain compared to the IA-CAN approach. The entropy obtained from the new randomization solutions is very near the optimum (11 bits), thus presenting

a high robustness against reverse-engineering attacks. The conditional entropy is better achieved with the Arithmetic Masking and the Dynamic Intervals. This last solution has the interest not to modify the hardware of the CAN interface. Overall, the proposed solutions are much better than the existing IA-CAN, as proven by the resulting security gain formally expressed by means of information theory metrics.

Appendix

Let id_o be a random variable representing original identifiers whose outcome is id_1, id_2, \dots, id_N with probabilities $P(id_1), P(id_2), \dots, P(id_N)$. We consider a second random variable id_r representing randomized identifiers whose outcome is in $[0, 2^n - 1]$.

Entropy of Fixed Mapping

The entropy of the fixed mapping solutions (IA-CAN, equal intervals, frequency intervals) is the following:

- IA-CAN: $H_{IA-CAN}(id_r) = H(id_o) + a$
- Equal Intervals: $H_{EI}(id_r) = H(id_o) + n - \log_2(N)$
- Frequency Intervals: $H_{FI}(id_r) = n$

Proof According to the fixed mapping randomization functions (IA-CAN, equal intervals, frequency intervals), each identifier id_i is randomized over a fixed interval I_i of width $W(I_i)$. We begin by computing the probability that the random variable id_r takes the value $x \in [0, 2^n]$:

$$P(id_r = x) = \sum_{i=1}^N P(id_r = x | id_i) \times P(id_i)$$

The conditional probability of id_r knowing the original identifier $id_o = id_i$:

$$P(id_r = x | id_i) = \frac{1_{I_i}(x)}{W(I_i)}$$

$$H(id_r) = \sum_{x \in [0, 2^n - 1]} P(id_r = x) \times \log_2 \left(\frac{1}{P(id_r = x)} \right)$$

$$\begin{aligned}
 &= \sum_{x \in [0, 2^n - 1]} \left[\sum_{i=1}^N P(id_i) \frac{1_{I_i}(x)}{W(I_i)} \right] \times \log_2 \left(\frac{1}{\left[\sum_{j=1}^N P(id_j) \frac{1_{I_j}(x)}{W(I_j)} \right]} \right) \\
 &= \sum_{i=1}^N \sum_{x \in [0, 2^n - 1]} P(id_i) \frac{1_{I_i}(x)}{W(I_i)} \times \log_2 \left(\frac{1}{\left[\sum_{j=1}^N P(id_j) \frac{1_{I_j}(x)}{W(I_j)} \right]} \right) \\
 H(id_r) &= \sum_{i=1}^N \sum_{x \in I_i} P(id_i) \frac{1_{I_i}(x)}{W(I_i)} \times \log_2 \left(\frac{1}{\left[\sum_{j=1}^N P(id_j) \frac{1_{I_j}(x)}{W(I_j)} \right]} \right)
 \end{aligned}$$

Since the intervals I_i are nonoverlapping: $\forall x \in I_i, \forall j \neq i \rightarrow 1_{I_j}(x) = 0$

We can thus simplify the expression: $\forall x \in I_i, \forall j \neq i \rightarrow \sum_{j=1}^N P(id_j) \frac{1_{I_j}(x)}{W(I_j)} = P(id_i) \frac{1_{I_i}(x)}{W(I_i)}$

$$\begin{aligned}
 H(id_r) &= \sum_{i=1}^N \sum_{x \in I_i} P(id_i) \frac{1_{I_i}(x)}{W(I_i)} \times \log_2 \left(\frac{1}{P(id_i) \frac{1_{I_i}(x)}{W(I_i)}} \right) \\
 &= \sum_{i=1}^N \sum_{x \in I_i} P(id_i) \frac{1}{W(I_i)} \times \log_2 \left(\frac{1}{P(id_i) \frac{1}{W(I_i)}} \right)
 \end{aligned}$$

- IA-CAN entropy: $\forall i \in [1, N], W(I_i) = 2^a$

$$H(id_r) = \sum_{i=1}^N \sum_{x \in I_i} P(id_i) \frac{1}{2^a} \times \log_2 \left(\frac{1}{P(id_i) \frac{1}{2^a}} \right) = H(id_o) + a$$

- Equal interval entropy: $\forall i \in [1, N], W(I_i) = \frac{2^n}{N}$

$$H(id_r) = \sum_{i=1}^N \sum_{x \in I_i} P(id_i) \frac{1}{\frac{2^n}{N}} \times \log_2 \left(\frac{1}{P(id_i) \frac{1}{\frac{2^n}{N}}} \right) = H(id_o) + n - \log_2(N)$$

- Frequency interval entropy: $\forall i \in [1, N], W(I_i) = 2^n \times P(id_i)$

$$H(id_r) = \sum_{i=1}^N \sum_{x \in I_i} P(id_i) \frac{1}{2^n \times P(id_i)} \times \log_2 \left(\frac{1}{P(id_i) \frac{1}{2^n \times P(id_i)}} \right) = n$$

□

Conditional Entropy of Fixed Mapping

The conditional entropy of randomized identifiers knowing the original identifiers of the fixed mapping solutions (IA-CAN, equal intervals, frequency intervals) is the following:

- IA-CAN: $H_{IA-CAN}(id_r|id_o) = a$
- Equal Intervals: $H_{EI}(id_r|id_o) = n - \log_2(N)$
- Frequency Intervals: $H_{FI}(id_r|id_o) = n - H(id_o)$

Proof

$$H(id_r|id_o) = H(id_r, id_o) - H(id_o)$$

$$H(id_r, id_o) = \sum_{x \in [0, 2^n - 1]} \sum_{i=0}^N P(id_r = x, id_o = id_i) \log_2 \left(\frac{1}{P(id_r = x, id_o = id_i)} \right)$$

$$P(id_r = x, id_o = id_i) = \begin{cases} P(id_i) \times \frac{1}{w(I_i)}, & x \in I_i \\ 0, & \text{elsewhere} \end{cases}$$

$$H(id_r, id_o) = \sum_{i=0}^N \sum_{x \in I_i} \frac{P(id_i)}{w(I_i)} \log_2 \left(\frac{1}{P(id_i) \frac{1}{w(I_i)}} \right)$$

$$H(id_r, id_o) = H(id_r)$$

$$H(id_r|id_o) = H(id_r) - H(id_o)$$

- IA-CAN conditional entropy : $H(id_r|id_o) = a$
- Equal interval conditional entropy : $H(id_r|id_o) = n - \log_2(N)$
- Frequency interval conditional entropy : $H(id_r|id_o) = n - H(id_o)$

□

Entropy of Dynamic Intervals

Let id_o^t be a Markov chain over the space of original identifiers $(id_1, id_2, \dots, id_N)$. And the matrix M presented in Eq. (25) be its transition matrix. Let id_r be the random variable over $[0, 2^n - 1]$, generated using the dynamic interval randomization strategy applied to id_o^t . We have $H_{DI}(id_r) = n$

Proof

$$\begin{aligned}
 H(id_r) &= \sum_{x \in [0, 2^n - 1]} P(id_r = x) \times \log_2 \left(\frac{1}{P(id_r = x)} \right) \\
 P(id_r = x) &= \sum_i^N P(id_r = x | id_o^t = id_i) \times P(id_o^t = id_i) \\
 P(id_r = x) &= \sum_i^N \sum_j^N P(id_r = x | id_i^t, id_j^{t+1}) \times P(id_j^{t+1} | id_i^t) \times P(id_i^t) \\
 P(id_r = x | id_i^t, id_j^{t+1}) &= \frac{1_{I_{i,j}}(x)}{W(I_{i,j})}
 \end{aligned}$$

where $W(I_{i,j}) = P(id_j^{t+1} | id_i^t) \times 2^n$ is the width of the interval $I_{i,j}$

$$\begin{aligned}
 P(id_r = x) &= \sum_i^N \sum_j^N \frac{1_{I_{i,j}}(x)}{W(I_{i,j})} \times P(id_j^{t+1} | id_i^t) \times P(id_i^t) \\
 &= \sum_i^N \sum_j^N \frac{1_{I_{i,j}}(x)}{P(id_j^{t+1} | id_i^t) \times 2^n} \times P(id_j^{t+1} | id_i^t) \times P(id_i^t) \\
 &= \sum_i^N \sum_j^N \frac{1_{I_{i,j}}(x)}{2^n} \times P(id_i^t)
 \end{aligned}$$

$$\forall x \in [0, 2^n - 1], \sum_j^N 1_{I_{i,j}}(x) = 1$$

$$P(id_r = x) = \sum_i^N \frac{1}{2^n} \times P(id_i^t) = \frac{1}{2^n}$$

$$H(id_r) = \sum_{x \in [0, 2^n - 1]} \frac{1}{2^n} \times \log_2 \left(\frac{1}{2^n} \right)$$

$$H(id_r) = n$$

□

Entropy of Arithmetic Masking

Proof

$$H(id_r) = \sum_{x \in [0, 2^n - 1]} P(id_r = x) \log_2 \left(\frac{1}{P(id_r = x)} \right)$$

$$P(id_r = x) = \begin{cases} \sum_{i=0}^x \frac{P(id_i)}{2^{n-N+1}} & , x \in [0, N-2] \\ \frac{1}{2^{n-N+1}} & , x \in [N-1, 2^n - N] \\ \sum_{i=x-2^n+N}^{N-1} \frac{P(id_i)}{2^{n-N+1}} & , x \in [2^n - N + 1, 2^n - 1] \end{cases}$$

$$\begin{aligned} H(id_r) &= \sum_{x \in [N-1, 2^n - N]} \frac{1}{2^{n-N+1}} \times \log_2(2^n - N + 1) \\ &+ \sum_{x \in [0, N-2]} \left[\sum_{i=0}^x \frac{P(id_i)}{2^{n-N+1}} \right] \times \log_2 \left(\frac{1}{\sum_{i=0}^x \frac{P(id_i)}{2^{n-N+1}}} \right) \\ &+ \sum_{x \in [2^n - N + 1, 2^n - 1]} \left[\sum_{i=x-2^n+N}^{N-1} \frac{P(id_i)}{2^{n-N+1}} \right] \\ &\times \log_2 \left(\frac{1}{\sum_{i=x-2^n+N}^{N-1} \frac{P(id_i)}{2^{n-N+1}}} \right) \\ H(id_r) &= \frac{2^n - 2(N-1)}{2^n - N + 1} \log_2(2^n - N + 1) + \sum_{x \in [0, N-2]} \left[\sum_{i=0}^x \frac{P(id_i)}{2^{n-N+1}} \right] \\ &\times \log_2 \left(\frac{1}{\sum_{i=0}^x \frac{P(id_i)}{2^{n-N+1}}} \right) \\ &+ \left[\sum_{i=x+1}^{N-1} \frac{P(id_i)}{2^{n-N+1}} \right] \times \log_2 \left(\frac{1}{\sum_{i=x+1}^{N-1} \frac{P(id_i)}{2^{n-N+1}}} \right) \end{aligned}$$

$$\begin{aligned}
H(id_r) &= \frac{2^n - 2(N-1)}{2^n - N + 1} \log_2(2^n - N + 1) \\
&\quad + \sum_{x \in [0, N-2]} \frac{1}{2^n - N + 1} \log_2(2^n - N + 1) \\
&\quad + \sum_{x \in [0, N-2]} \sum_{i=0}^x \frac{P(id_i)}{2^n - N + 1} \times \log_2 \left(\frac{1}{\sum_{i=0}^x P(id_i)} \right) \\
&\quad + \sum_{i=x+1}^{N-1} \frac{P(id_i)}{2^n - N + 1} \times \log_2 \left(\frac{1}{\sum_{i=x+1}^{N-1} P(id_i)} \right) \\
H(id_r) &= \frac{2^n - 2(N-1)}{2^n - N + 1} \log_2(2^n - N + 1) + \frac{N-1}{2^n - N + 1} \log_2(2^n - N + 1) \\
&\quad + \frac{1}{2^n - N + 1} \sum_{x \in [0, N-2]} \sum_{i=0}^x P(id_i) \times \log_2 \left(\frac{1}{\sum_{i=0}^x P(id_i)} \right) \\
&\quad + \sum_{i=x+1}^{N-1} P(id_i) \times \log_2 \left(\frac{1}{\sum_{i=x+1}^{N-1} P(id_i)} \right) \\
H(id_r) &= \frac{2^n - N + 1}{2^n - N + 1} \log_2(2^n - N + 1) \\
&\quad + \frac{1}{2^n - N + 1} \sum_{x \in [0, N-2]} \sum_{i=0}^x P(id_i) \times \log_2 \left(\frac{1}{\sum_{i=0}^x P(id_i)} \right) \\
&\quad + \sum_{i=x+1}^{N-1} P(id_i) \times \log_2 \left(\frac{1}{\sum_{i=x+1}^{N-1} P(id_i)} \right) \\
H(id_r) &= \log_2(2^n - N + 1) + \frac{1}{2^n - N + 1} \sum_{x \in [0, N-2]} \sum_{i=0}^x P(id_i) \\
&\quad \times \log_2 \left(\frac{1}{\sum_{i=0}^x P(id_i)} \right) + \sum_{i=x+1}^{N-1} P(id_i) \\
&\quad \times \log_2 \left(\frac{1}{\sum_{i=x+1}^{N-1} P(id_i)} \right)
\end{aligned}$$

□

Conditional Entropy of Arithmetic

The arithmetic masking conditional entropy is:

$$H_{AM}(id_r|id_o) = \log_2(2^n - N + 1)$$

Proof

$$P(id_r = x|id_o = id_i) = \frac{1_{[i-1, 2^n - N + i - 1]}}{2^n - N + 1}$$

$$H_{AM}(id_r|id_o) = \sum_{i=0}^N P(id_i) H(id_r|id_o = id_i)$$

$$H_{AM}(id_r|id_o) = \sum_{i=0}^N P(id_i) \sum_{x \in [i-1, 2^n - N + i - 1]} P(id_r = x|id_o = id_i) \times \log_2 \left(\frac{1}{P(id_r = x|id_o = id_i)} \right)$$

$$H_{AM}(id_r|id_o) = \sum_{i=0}^N P(id_i) \sum_{x \in [i-1, 2^n - N + i - 1]} \frac{1}{2^n - N + 1} \log_2 \left(\frac{1}{\frac{1}{2^n - N + 1}} \right)$$

$$H_{AM}(id_r|id_o) = \sum_{i=0}^N P(id_i) \log_2(2^n - N + 1)$$

$$H_{AM}(id_r|id_o) = \log_2(2^n - N + 1)$$

□

Fixed Mapping Optimality Proof

If we adopt a fixed mapping randomization strategy, the optimal solution in terms of conditional entropy is the frequency interval solutions.

Proof In the context of fixed mapping, we want to find the best decomposition of intervals that maximizes the conditional entropy. We previously showed that the conditional entropy of all fixed mapping solutions can be expressed as $H(id_r|id_o) = \sum_{i \in [1, N]} P(id_i) \times \log_2(W_i)$, where I_i is the randomization interval of id_i of width $W(I_i)$. For the fixed mapping solutions, the intervals are nonoverlapping. Besides the width of each interval I_i is positive ($W(I_i) \geq 0$) and their sum equals 2^n . Thus we define the following problem:

$$\text{Argmax}_{\{I_i\}, i \in [1, N]} H(id_r|id_o) = \sum_i P(id_i) \times \log_2(W_i)$$

Subject to the following constraints:

$$\begin{aligned} h_0 &: \sum_{i \in [1, N]} W_i - 2^n = 0 \\ h_i &: \forall i \in [1, N], -W_i \leq 0 \end{aligned}$$

To find the solution to this problem, we use the Lagrangian multiplier:

$$\mathcal{L}(W_1, \dots, W_N, \lambda_1, \dots, \lambda_N, \lambda_0) = H(id_r|id_o) + \sum_{j=0}^N \lambda_j h_j$$

and solve the equation system: $\frac{\partial \mathcal{L}}{\partial W_i} = 0, \quad \forall i \in [1, N]$

$$\frac{\partial \mathcal{L}}{\partial W_i}(W_1, \dots, W_N, \lambda_1, \dots, \lambda_N, \lambda_0) = \frac{\partial H}{\partial W_i} + \sum_{j=0}^N \lambda_j \frac{\partial h_j}{\partial W_i}$$

$$\forall i \in [0, N] : \lambda_i \times h_i = 0$$

$$\begin{aligned} h_0 &: \sum_{i \in [1, N]} W_i - 2^n = 0 \\ h_i &: \forall i \in [1, N], -W_i \leq 0 \end{aligned}$$

We have: $\frac{\partial H}{\partial W_i} = P(id_i) \times \frac{1}{W_i}$ and $\frac{\partial h_0}{\partial W_i} = 1$ and $\frac{\partial h_j}{\partial W_i} = -1$ if $(i = j)$, 0 otherwise

$$\forall i \in [1, N] : P(id_i) \times \frac{1}{W_i} + \lambda_0 - \lambda_i = 0$$

$$\forall i \in [1, N] : \lambda_i \times h_i = 0$$

Resolving this system of equations gives:

$$\lambda_i = 0, \quad \forall i \in [1, N]$$

$$\lambda_0 = \frac{-1}{2^n}$$

Hence:

$$\Rightarrow \forall i \in [1, N] : W_i = P(id_i) \times 2^n$$

□

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. S. Dario, M. Mirco, C. Michele, Detecting attacks to internal vehicle networks through hamming distance, in *IEEE 2017 AEIT International Annual Conference-Infrastructures for Energy and ICT (AEIT 2017)*, 2017
3. E. de Chérisey, S. Guilley, A. Heuser, O. Rioul, On the optimality and practicability of mutual information analysis in some scenarios. *Cryptogr. Commun.* **10**(1), 101–121 (2018)
4. I.D. Foster, A. Prudhomme, K. Koscher, S. Savage, Fast and vulnerable: a story of telematic failures, in *WOOT*, 2015
5. K. Han, A. Weimerskirch, K.G. Shin, Automotive cybersecurity for in-vehicle communication, in *IQT Quarterly*, vol. 6 (2014), pp. 22–25
6. K. Han, A. Weimerskirch, K.G. Shin, A practical solution to achieve real-time performance in the automotive network by randomizing frame identifier, in *Escar Conference*, Cologne, Germany, 2015
7. O. Hartkopp, R. Schilling, MaCAN - Message authenticated CAN, in *Escar Conference*, Berlin, 2012
8. T. Hoppe, S. Kiltz, J. Dittmann, Security threats to automotive CAN networks—practical examples and selected short-term countermeasures, in *International Conference on Computer Safety, Reliability, and Security* (Springer, Berlin, 2008), pp. 235–248
9. A. Humayed, 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, New York, 2017), pp. 19–26
10. ISO, *11898-1—Road Vehicles—Controller Area Network (CAN)—Part 1: Data Link Layer and Physical Signalling* (International Organization for Standardization, Geneva, 2003)
11. ISO, *11898-2—Road Vehicles—Controller Area Network (CAN)—Part 2: High-Speed Medium Access Unit* (International Organization for Standardization, Geneva, 2003)
12. ISO, *11898-3—Road Vehicles—Controller Area Network (CAN)—Part 2: Fault Tolerant Medium Access Unit* (International Organization for Standardization, Geneva, 2003)
13. 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 *2010 IEEE Symposium on Security and Privacy (SP)* (IEEE, Piscataway, 2010), pp. 447–462

14. M. Marchetti, D. Stabili, Anomaly detection of CAN bus messages through analysis of ID sequences, in *2017 IEEE Intelligent Vehicles Symposium (IV)* (IEEE, Piscataway, 2017), pp. 1577–1583
15. C. Miller, C. Valasek, Adventures in automotive networks and control units. *DEF CON 21*, 260–264 (2013)
16. C. Miller, C. Valasek, Remote exploitation of an unaltered passenger vehicle. Black Hat USA, 2015
17. M. Müter, N. Asaj, Entropy-based anomaly detection for in-vehicle networks, in *2011 IEEE Intelligent Vehicles Symposium (IV)* (IEEE, Piscataway, 2011), pp. 1110–1115
18. D.K. Nilsson, U.E. Larson, E. Jonsson, Efficient in-vehicle delayed data authentication based on compound message authentication codes, in *IEEE 68th Vehicular Technology Conference, 2008. VTC 2008-Fall* (IEEE, Piscataway, 2008), pp. 1–5
19. C. Smith, *The Car Hacker's Handbook: A Guide for the Penetration Tester* (No Starch Press, San Francisco, 2016)
20. A. Taylor, N. Japkowicz, S. Leblanc, Frequency-based anomaly detection for the automotive CAN bus, in *2015 World Congress on Industrial Control Systems Security (WCICSS)* (IEEE, Piscataway, 2015), pp. 45–49
21. Testing CAN Network with help of CANtoolz. <https://www.slideshare.net/AlexeySintsov/testing-can-network-with-help-of-cantoolz>, 2016. Accessed 1 Jan 2018