# Towards Efficient Compression of CAN Traffic Logs

András Gazdag[1a], Levente Buttyán[1], and Zsolt Szalay[2]

[1] *Laboratory of Cryptography and System Security, Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics*

[2] *Department of Automotive Technologies, Faculty of Transportation Engineering and Vehicle Engineering, Budapest University of Technology and Economics*

[a] *Corresponding author: agazdag@crysys.hu*

***Abstract***

*In this paper, we propose a compression method that allows for the efficient storage of large amounts of CAN traffic data, which is needed for the forensic investigations of accidents caused by cyber attacks on vehicles. Compression of recorded CAN traffic also reduces the time (or bandwidth) needed to off-load that data from the vehicle. In addition, our compression method allows analysts to perform log analysis on the compressed data, therefore, it contributes to reduced analysis time and effort. We achieve this by performing semantic compression on the CAN traffic logs, rather than simple syntactic compression. Our compression method is lossless, thus preserving all information for later analysis. Besides all the above advantages, the compression ratio that we achieve is better than the compression ratio of state-of-the-art syntactic compression methods, such as zip.*

***Keywords:*** *CAN, network traffic capture, semantic compression, forensic analysis*

## 1 Introduction

Modern vehicles have multiple embedded controllers, called ECUs (Electronic Control Units), connected together by internal communication networks such as the CAN bus (Controller Area Network). ECUs are programmable devices, and many of the vehicles' functions now rely on software running on them, as well as on protocols for exchanging information between ECUs via CAN buses. Often, vehicles also have interfaces, such as the OBD (On-board Diagnostic) port and various wireless interfaces that make it possible to access certain parts of the vehicle's internal network from outside. Such access may be needed for diagnostic and maintenance purposes, for connecting mobile consumer devices to the entertainment unit of the vehicle, or allowing for the transfer of various sensor data in and out of the vehicle. The concept of *connected cars* goes even further by introducing short range wireless connections between vehicles and to the Internet infrastructure, enabling new types of safety and infotainment applications.

All this development means that modern vehicles should be considered as cyber-physical systems, in which special purpose computers control physical processes, and those computers are no longer isolated from the cyber space out there. This introduces an entirely new domain of problems for vehicles, and road safety in general: the domain of *cyber security*. Indeed, ECUs in vehicles can be compromised in similar ways as computers are compromised on the Internet (e.g., exploiting a buffer overflow vulnerability in their software), and due to the increasing level of connectedness, such attacks are now possible to be carried out remotely (i.e., without requiring physical access to the vehicle). The feasibility of such remote attacks has been demonstrated by various research groups recently, showing also their potentially catastrophic consequences [1][2][3].

The possibility of remote cyber attacks on vehicles generated a lot of interest in developing protection measures that either prevent or detect such attacks. One of the proposed approaches is to perform log analysis on recorded CAN traffic and identify intrusions either in real-time or in an off-line manner. While real-time intrusion detection seems to be the ultimate goal, off-line analysis of the logged CAN traffic would still be required for better understanding of how an attack worked and for forensic purposes in case the attack caused some physical damage.

Being able to detect and analyze cyber attacks on vehicles requires continuous collection and recording of the CAN traffic. This can potentially lead to a large amount of data that need to be stored in the vehicle. In this paper, we propose a compression method that allows for the lossless, yet efficient storage of that large amount of data. Compression of CAN traffic logs have other notable advantages: it helps to shorten the time required to off-load the data from the vehicle. Moreover, the large amount of data is not only a problem for storage and communication: it also makes forensic analysis hard, resource intensive, and time consuming. Our compression method allows analysts to perform log analysis on the compressed data, therefore, it contributes to reduced analysis time and effort. We achieve this by performing *semantic compression* on the CAN traffic logs, rather than simple syntactic compression. Besides all these advantages, the compression ratio that we achieve is better than the compression ratio of state-of-the-art syntactic compression methods, such as zip.

Syntactic compression methods operate on the low level byte stream representation of the data. In contrast to this, semantic compression methods interpret the data being compressed and take advantage of its semantic understanding. Semantic compression has generated considerable interest in the recent years. It has been successfully applied in different fields such as general database compression [4], video compression [5], and virtual machine memory compression [6]. What we propose in this paper is a new application area for it.

The rest of the paper is organized as follows: In Section 2, we give an overview on crash data recorders and show that they are not appropriate for supporting forensic analysis of cyber attacks. In Section 3, we provide background information on CAN technology. We describe our new semantic compression algorithm in Section 4, and we evaluate its performance in Section 5. Finally, in Section 6, we conclude the paper.

## 2 Crash data recorder devices

Today, one of the key motivations in automotive development is increasing traffic safety. Since September 2014, a so called Event Data Recorder (EDR) is mandatory for every new passenger car and new light commercial vehicle (LCV) in the US. The purpose of EDR devices is to collect data about the vehicle dynamics and the vehicle status that enable better accident reconstruction. It helps in validating insurance claims, encourages safer driving behavior, and extends the scientific knowledge about real accidents, thus, resulting in safer vehicle design.

The European answer to EDR is the so called eCALL system that will be introduced in all new cars as of April 2018 in the EU. The major difference between the two systems is that EDR devices are offline systems, requiring a later data retrieval, while eCALL is an online system that immediately calls the ambulance (dials 112) in case of emergency. Although EDR is not mandatory in Europe, due to some side effects of the US regulation, approximately 4-6% of the EU vehicles are equipped with an EDR device.

The importance of an EDR-like “black box” increases with the deployment of highly automated functions in road vehicles, as there must be some objective evidence proving who was in charge of control in the vehicle in a critical situation. It is, however, not clear what would be the minimum set of data that needs to be collected in case of automated or highly automated vehicles; accident researchers and automated vehicle experts are currently working together on new regulations in this field.

While EDR devices collect data from the CAN bus, the recording of that data is not continuous in time, but triggered only by certain events that may indicate a forthcoming accident (e.g., events that trigger the airbag). In addition, the data recorded by EDR devices is limited to a short interval in time (typically a few seconds) surrounding the point in time of the accident. Unfortunately, a cyber attack that ultimately leads to an accident may happen long before the accident itself (at least, well beyond a few seconds interval around the time of the accident), and therefore, the data recorded by an EDR device will likely contain no useful information about the cyber attack causing the accident. This means that EDR devices in their current form are inappropriate for supporting forensic investigations related to cyber attacks. The European eCALL system does not even record CAN data, so it clearly cannot be used in case of cyber incidents.

For detecting cyber attacks and for being able to analyze how an attack was executed and what effect it had on the vehicle, one needs to collect and record a continuous flow of CAN traffic data for an extended period of time. In this setting, efficient storage of this potentially large amount of data is crucial, while EDR type devices does not need to store that large amount of data. Thus, although they have seemingly similar goals, EDR devices actually address a problem different from the one that we address in this paper.

## 3 Background – the CAN protocol

The CAN protocol was designed to be simple, causing only a small overhead in the communication. None of the messages contain any authentication information. This nature of the protocol makes it very easy to spoof CAN messages, which, in turn, can lead to numerous other attacks based on the injection of arbitrary messages into the CAN traffic. Understanding CAN messages, however, is a more complex problem, because it requires a priory knowledge of the network and the communicating parties.

In vehicles, the ECUs communicate with each other mostly periodically. While the communication is event based, regular repetition times enable a quite accurate prediction of the upcoming pattern of messages. Another main property of the traffic is that the content of messages are often repeating. Numerous instances of the exact same message or message type can be observed throughout traffic captures.

We confirmed these properties via capturing traffic in real vehicles. Our test vehicles, however, were not premium category vehicles. In premium category vehicles, there are more ECUs, which results in a higher variety of the CAN message types and message contents. Also, the same high variety is expected in autonomous vehicles. Yet, we believe that even in those cases, CAN traffic is rather regular and exhibits features that can be exploited for its efficient compression.

## 4 Compression algorithm

We propose a compression algorithm that takes advantage of the largely periodic nature of the CAN traffic. The high level approach of our algorithm is to separate the traffic into message flows, containing only messages that have the same ID, and then, compressing each message flow separately leveraging the previously identified properties.

As a first step, the traffic is separated into message flows. This is done by filtering messages based on the ID field of the protocol. This step generates separate lists of messages where the only remaining information for each message to be stored is its timestamp in the log and the content of the message. In many cases, the content shows only very low variations, allowing the compression to be even more efficient.

Storing a complete and separate timestamp for each message in a flow would be a waste of storage. Our more efficient approach takes advantage of the periodicity of messages. Theoretically, a new message with the same ID should come at an exact predicted time based on the inter-arrival time of this message type. However, this behavior can be changed by a higher priority message on the CAN bus. If two messages are sent at the same time, then only the one with the higher priority will be sent, shifting the inter-arrival time of the messages with the lower priority. From this point on, the arrival times of this complete message flow will be shifted.

An efficient way to store the timestamp of a message is to store the number of periods (specific for that flow) passed since the last message of the same type and an additional offset value that is induced by either priority causes or measurement distortions. This approach also allows for an efficient description of message flows, where the same message data appears repeatedly from time to time.

```
id:0xb0
start_time:1481492674750066
period:19999
000000001103:0#0,16#-13,16#57,16#10,16#-46,16#52,16#5,16#-45,16#15,16#40,16#4,16#-42,16#131,16#-70,16#-47,16#12,16#51,16#-1
0,16#54,16#189,16#-225,16#-37,16#-2,16#53,16#7,16#-46,16#59,16#-3,16#-43,16#11,16#46,16#11,16#-48,16#50,16#5,16#-49,16#13,1
6#49,16#-1,16#-31,16#40,16#6,16#-34,16#-2,16#57,16#-3,16#-42,16#57,16#-2,16#-40,16#2,16#52,16#6,16#-42,16#49,16#11,16#-53,1
6#16,16#168,16#-119,16#-33,16#40,16#4,16#-42,16#8,16#56,16#-4,16#-40,16#48,16#8,16#-37,16#0,16#50,16#7,16#-41,16#43,16#6,16
#-42,16#16,16#38,16#4,16#-29,16#39,16#12,16#-49,16#0,16#57,16#5,16#-45,16#54,16#2,16#-35,16#-2,16#51,16#11,16#-50,16#61,16#
-8,16#-38,16#12,16#45,16#1,16#133,16#-120,16#7,16#-41,16#-1,16#58,16#-1,16#-37,16#43,16#6,16#-36,16#-1,16#53,16#3,16#-41,16
#169,16#-117,16#-46,16#12,16#49,16#7,16#-43,16#48,16#11,
000000001104:-17583#-4781,16#-178,16#7,16#163,16#-190,16#43,16#80,16#-110,16#42,16#12,16#-47,16#27,16#35,16#-5,16#-38,16#48
,16#5,16#-38,16#132,16#-89,16#20,16#-51,16#54,16#1,16#-41,16#14,16#43,16#2,16#-39,16#50,16#9,16#-47,16#6,16#59,16#-5,16#-42
,16#43,16#10,16#-30,16#-6,16#47,16#4,16#-45,16#61,16#-2,16#-42,16#11,16#47,16#6,16#-45,16#49,16#12,16#-55,16#6,16#58,16#1,1
6#9,16#0,16#7,16#-37,16#-2,16#58,16#-4,16#-42,16#57,16#74,16#-115,16#5,16#48,16#7,16#-49,16#50,16#16,16#-47,16#12,16#44,16#
4,16#-38,16#43,16#5,16#-41,16#4,16#165,16#-105,16#-45,16#57,16#-9,16#-35,16#4,16#52,16#11,16#-49,16#52,16#1,16#-41,16#10,16
#43,16#5,16#-34,16#46,16#11,16#-44,16#73,16#7,16#-31,16#-34,16#47,16#7,
000000001105:-17583#-4999,16#45,16#-5,16#-32,16#47,16#6,16#-40,16#5,16#55,16#35,16#-79,16#56,16#104,16#-81,16#-62,16#52,16#
96,16#-133,16#95,16#162,16#-252,16#16,16#45,16#2,16#-33,16#48,16#7,16#-47,16#4,16#56,16#-2,16#-42,16#48,16#8,16#-31,16#-5,1
6#43,16#10,16#-45,16#58,16#-3,16#-41,16#12,16#48,16#-2,16#-40,16#49,16#10,16#-47,16#5,16#59,16#-3,16#-33,16#34,16#7,16#-28,
16#-7,16#53,16#8,16#-45,16#58,16#-1,16#-45,16#13,16#42,16#9,16#-44,16#49,16#11,16#-50,16#9,16#44,16#31,16#-57,16#43,16#3,16
#-34,16#-4,16#58,16#0,16#-42,16#52,16#7,16#-46,16#3,16#56,16#5,16#-53,16#53,16#13,16#-47,16#9,16#47,16#4,16#-37,16#43,16#6,
16#-38,16#1,16#57,16#-2,16#-40,16#60,16#13,16#-62,16#2,16#59,16#9,16#-49,16#52,16#2,16#-42,16#11,16#46,16#3,16#-33,16#39,16
#9,16#-41,16#4,16#55,16#-10,16#-39,16#53,16#10,16#-38,16#-1,16#50,16#10,16#-47,16#53,
```

***Fig 1*** *A compressed message flow example.*

For each message flow, there are some additional metadata to be stored: the message ID, the first appearance of the flow in the log, and the characteristic period length of the flow. These flow specific metadata should be followed by the message data and then the compressed timestamp for each message. An example of this

compressed format can be seen in Figure 1, where the # sign separates the period number and the offset value in each compressed timestamp.

We validated the lossless property of our algorithm by checking that the SHA-256 hash of the original data matches that of the data that we restore after de-compression.

## 5 Evaluation

We evaluated our algorithm in terms of performance and efficiency. As the most important performance metric, we calculated the compression ratio, and as for efficiency, we also measured the speed of our implementation. We performed our measurements multiple times with different datasets originating from different vehicles. We used vehicles of three different brands all belonging to the low mid-level category built between 2005 and 2010.

We captured traffic with a Raspberry Pi based CAN interpreter[1]. It allowed us to access the raw information on the CAN bus, and we saved every CAN message with a timestamp. We performed traffic captures through the OBD interface where the design of the vehicle allowed for an uninterrupted access to the powertrain CAN bus traffic through this connection. We were able to gather traffic logs of multiple hours in all three types of vehicle that we used in the evaluation.

The measured compression ratios show significant progress in the data sizes. We were able to achieve compression ratios of around 20% using an ASCII representation of the output of our algorithm. In some cases, our results showed an even better result approximating a 16% compression ratio.

Our algorithm runs much faster than the arrival speed of the log data on each of the vehicles we tested. It is capable of efficiently compressing data gathered in a 6-minute time frame in less than 10 seconds. This speed makes our algorithm a good candidate for on-board data compression for local usage of the information or as a preparation for a remote transmission.

## 6 Conclusion

In this paper, we have presented an efficient way to perform lossless compression of CAN traffic logs. Based on our observations of the periodic properties of CAN traffic, we designed a semantic compression algorithm for CAN traffic. With the use of our algorithm, storage efficiency and communication costs can be significantly improved, while keeping the possibility to perform analysis on the compressed data.

## 7 Acknowledgement

The project has been supported by the European Union, co-financed by the European Social Fund. (EFOP-3.6.2-16-2017-00002).

## References

[1] K. Koscher et al., Experimental Security Analysis of a Modern Automobile, IEEE Symposium on Security and Privacy, 2010.

[2] S. Checkoway et al., Comprehensive Experimental Analyses of Automotive Attack Surfaces, Usenix Security Symposium, 2011.

[3] A. Greenberg, Hackers Remotely Kill a Jeep on the Highway - With Me in It, Wired Magazin, July 21, 2015.

[4] H. V. Jagadish et al., ItCompress: an iterative semantic compression algorithm, Proceedings. 20th International Conf. on Data Engineering, 2004, pp. 646-657.

[5] Tao Mei et al., Near-lossless semantic video summarization and its applications to video analysis. ACM Trans. Multimedia Comput. Commun. Appl. 9, 3, Article 16 (July 2013)

[6] Anshul Rai et al., MiG: Efficient Migration of Desktop VMs Using Semantic Compression, USENIX Annual Technical Conference, 2013

[1] http://skpang.co.uk/catalog/images/raspberrypi/pi_2/PICAN2DSB.pdf