The influence of the training set size on the performance of the Robust Covariance Estimator as an anomaly detection algorithm on automotive CAN data

Vincent Kieberl
University of Amsterdam
vincent.kieberl@os3.nl

Silke Knossen
University of Amsterdam
silke.knossen@os3.nl

Master Security and Network Engineering
Research Project 1
Assessor: Cees de Laat
Supervisor: Colin Schappin

***Abstract*—Automotive anomaly detection algorithms often rely on CAN frame frequency analysis to determine whether frames on the CAN bus are malicious or not. Related research has found that the Robust Covariance Estimator has potential for use in real-life applications for automotive anomaly detection. This and other related research uses fairly small datasets to conduct their research on. Since datasets collected within seconds or a few minutes may be unable to reflect all different driving situations, this research provides insight into the influence of the amount of training data on the performance of a classification model based on the Robust Covariance Estimator algorithm. Using the PCAN-USB FD adapter, we have collected over 64 minutes of CAN data from a 2006 Audi A4. With this data, tests were performed in which fabrication attacks, suspension attacks and masquerade attacks were simulated. We conclude that, with a few exceptions, the general performance of the Robust Covariance Estimator does not differ significantly when training on more than 5 minutes of CAN data.**

I. Introduction

It has been predicted that by 2030, 50% of the total cost of a car will consist of costs related to electronics systems [1]. As more and more components of vehicles are controlled electronically, the security and potential vulnerabilities of these systems are becoming more important. Most modern automobiles feature an internal network called the Controller Area Network (CAN), which is used to interconnect control systems that are called Electronic Control Units (ECUs). These ECUs manage subsystems such as engine, drivetrain and transmission but also less-critical systems such as Heating, Ventilation, and Air Conditioning (HVAC) and tire pressure monitoring. As CAN was designed in the 1980's by Robert Bosch GmbH, it was meant for closed systems and therefore lacks security features such as encryption and authentication [2]. CAN is a bus system in which nodes are interconnected by a twisted wire pair that makes up a shared backbone, which makes CAN both cost-effective and easy to install and maintain. CAN frames do not contain a field for the destination of a frame. Instead, all frames are broadcast, and CAN nodes use frame filtering to discard frames that are not of use to them by checking a header field called the CAN ID. The CAN ID uniquely identifies a frame type on the network, and also acts as a priority number for that type of frame. A CAN frame sent with a lower CAN ID therefore has higher priority on the network, and another node will stop transmitting on the bus when it senses that another node wishes to send a frame with a lower ID [2] [3]. This is, for example, very useful when prioritizing frames that contain information about the engine or brakes over frames that contain information about the indicator lights.

Newer protocols such as FlexRay, Local Interconnect Network (LIN), Media Oriented Systems Transport (MOST) and Ethernet networks are nowadays sometimes used for subsystems, but their widespread use throughout vehicles is limited due to higher manufacturing costs, insufficient bandwidth and robustness, and complexity [4, 5, 6, 7, 8, 9].

Related research has focused on mitigating potential hazards that occur from attacks performed on ECUs through the CAN bus, for example using Intrusion Detection Systems (IDSs) (see [10, 11, 12]). Research has also shown that for most CAN IDs, the CAN frames occur with a certain frequency (i.e., they occur regularly with roughly the same interval time between frames of that ID throughout the entire dataset) [13, 14, 15]. If an attacker attempts to alter the operation of an ECU by sending forged CAN frames or by prohibiting an ECU to send CAN frames, the frame frequency changes. For this reason, automotive IDSs often rely on CAN frame frequency analysis and therefore on CAN frame timestamps as their primary source of information to determine whether frames on the CAN bus are malicious or not [13].

Schappin [4] has researched the performance of three frequency-based anomaly detection algorithms on automotive CAN data. The performance of these algorithms was measured using 185 seconds of real automotive CAN data in total. The Robust Covariance Estimator (RCE) provided the most promising results on this data. Other research on frequency-based anomaly detection algorithms for the CAN bus also only uses fairly small datasets ranging from 43 seconds to 40 minutes ([16, 13, 11, 17, 10], see Section III-B). Large-scale automotive CAN data acquisition to evaluate anomaly detection algorithms comes with its challenges. Considering

that frequency-based anomaly detection algorithms for automotive CAN data use the CAN frame frequency as their main feature, they require timestamps to have a resolution that reaches into microseconds accuracy [4]. Most generic hardware that can be used to read out the CAN bus (e.g. Arduino devices with CAN shields or generic serial interfaces) buffers frames before sending them to the processing device, where they are timestamped by software. This causes skewed timestamps because the contents of the buffer may be stored in the read-out hardware for a short period of time before being sent to the processing device. In addition, the interval time between frames from the read-out hardware to the processing device may be different than the actual interval time between two CAN frames when they were sent onto the CAN bus. As newer vehicles contain up to 200 separate ECUs [18], the number of CAN frames that are sent over the bus per second can reach into the hundreds. Datasets that are collected within seconds or a few minutes may therefore be unable to reflect all different driving situations. This is the reason we feel it is necessary to verify these algorithms on other and larger amounts of data.
The question we aim to answer with this research is the following:

To what extent does the amount of training data influence the performance of a classification model based on the Robust Covariance Estimator (RCE) as proposed by [4]?

To answer this main question we define the following sub-questions:

1) How can we collect a dataset from a real car that contains over 40 minutes of CAN data with microseconds accuracy?
2) What are the characteristics of the collected CAN datasets from different vehicles?
3) What is the influence of the amount of training data on the performance of a classification model based on the RCE on fabrication, suspension, and masquerade attacks?

II. Ethical considerations

This research involves no personal information. The CAN data used in this research consists of ECU data which is standard vehicle information that does not contain any personal user information. We have only used CAN data collected from cars for which we had permission. Our research has no impact on the security of the cars used as no CAN data is transmitted by us and attacks on the CAN data were simulated offline (see Section V).

III. Related work

A. CAN anomaly detection

Taylor et al. [13] performed research on using a One-Class Support Vector Machine (OCSVM) to detect anomalies in automotive CAN data. They collected approximately 25 minutes of data from a Ford Explorer and used part of this data as the training set for the OCSVM. The training set was then divided into overlapping time windows of 1 second each, with 0.5 seconds increments. This is because some features, such as the mean interval time, can only be computed over a certain range. The overlap was used to take into account slight feature changes over time. Taylor et al. [13] concluded that the mean interval time between two CAN frames was the only reliable feature for detecting inserted frames. The researchers also concluded that the optimal window size was 0.2 seconds with 0.1 second increments. The research did not include non-recurring frames and only evaluated the performance of the algorithm on attacks during which frames were inserted.

B. Robust Covariance Estimator

As briefly noted in Section I, the research conducted by [4] evaluated the performance of three different machine learning algorithms for anomaly detection: a One-Class Support Vector Machine, Isolation Forest and the Robust Covariance Estimator (RCE). The algorithms were tested on three different types of attacks: fabrication attacks, suspension attacks and masquerade attacks. These attacks are further described in Section V. Furthermore, instead of using one time window for all recurring CAN IDs, Schappin [4] split the CAN IDs into three different groups (fast, medium, slow) based on the interval time of the first two occurrences of CAN frames with a specific ID. Research has shown that this frequency does not change drastically over time [13, 14, 15]. This is of particular importance because the RCE uses this frequency to detect anomalies. The paper shows that out of the three evaluated algorithms, the Robust Covariance Estimator produces the most promising results. The algorithms were tested on a real CAN dataset that is available from the University of Tulsa (see [19]), which contains approximately two and a half minutes of CAN data from a 2010 Dodge Ram. The algorithms were also tested on a dataset consisting of five minutes of data that was generated from a CAN simulator, however the Robust Covariance Estimator was unable to produce results on this dataset due to errors which may have been caused by the values in this dataset being too similar. Schappin also attempted to obtain his own dataset using a tool called CANBus Triple, an Arduino-based device, and a tool called CAN Badger. This was however unsuccessful due to the fact that the tools used were unable to log CAN frames with a timestamp resolution in microseconds. Considering that this paper shows promising results of the RCE algorithm, we have used this algorithm in our research to evaluate RCE-based models trained on various amounts of training data.

C. CAN dataset sizes

Markovitz et al. [16] collected their dataset from a 2012 Ford Focus by connecting a CAN to USB interface (Peak System PCAN-USB) to the On-Board Diagnostics version 2 (OBD-2) port of the vehicle, using a D-sub to OBD-2 cable. The OBD-2 system for vehicles provides access to the status of the various vehicle sub-systems and is mostly used for vehicle diagnostics. It has been mandatory in the European Union

since 2001 for all gasoline cars and since 2003 for diesel cars [20]. The OBD-2 connector also contains two pins for CAN access, which theoretically makes it possible to listen in on CAN data traffic. However, it is at the manufacturer's discretion to decide whether to directly connect the internal CAN bus to the OBD-2 connector in the vehicle or not. Some car models feature a CAN gateway or firewall that is located between the ODB-2 CAN pins and the internal CAN network [21]. Using Peak System's PCAN-USB interface, [16] obtained 19 datasets, each consisting of 43 seconds of CAN data. The different datasets used consist of a total of eight different scenarios, such as turning the engine on and off and the operation of lights. In [11], short datasets in the range from 50 to 100 seconds are used. This data was collected by connecting an Arduino board to the OBD-2 port. Olufowobi et al. [17] obtained approximately 18 minutes of CAN data from a real vehicle, of which 40% was used to train their algorithm, and 60% was used as the test set. The largest dataset found is the dataset of around 40 minutes used by [10]. This was collected using an Arduino and Respberry Pi connected to the OBD-2 port of an anonymous vehicle. None of the publicly available CAN datasets that we found contained over 40 minutes of data, as well as timestamps for each CAN frame in microseconds accuracy. A publicly available dataset with these two requirements could be an addition for this field of research. To obtain such a dataset and use it in our research we inspected the previously described methods for collecting CAN data.

IV. Methodology

A. Data acquisition

In this research we have used Peak System's PCAN-USB FD interface [22] to collect CAN data. According to the manufacturer, the device is able to log CAN timestamps with 1 μs accuracy. A similar device used by [16] showed to deliver usable results, which led us to believe that the PCAN-USB FD had the most potential for successful large-scale CAN data acquisition with microseconds timestamp resolution. We connected this interface to the OBD-2 port of a vehicle using a D-sub to OBD-2 cable. To log the data we used the PCAN-Explorer software [22] running on a laptop connected to the PCAN-USB FD. Using this setup we tested six vehicle models to check whether it was possible to log CAN frames: a 2006 Volkswagen Lupo, a 1998 Audi A3, a 2006 Audi A4, a 2017 Volkswagen Golf, a 2018 Volkswagen Golf, and a 2017 Ford Fiesta. The 2006 Volkswagen Lupo and the 1998 Audi A3 did not show any data when the PCAN-USB FD interface was connected to the OBD-2 port. This may imply that the CAN bus is not accessible over the OBD-2 port in these vehicles. For the 2017 and 2018 Volkswagen Golf, the data logs only contained one CAN frame with ID 0x17F00010 repeating every 500 ms. It may be that these vehicles feature a CAN gateway that is connected to the CAN pins of the OBD-2 port, which may transmit useful information when a request frame is sent. This is known as an OBD-2 Parameter ID, standardized in Society of Automotive Engineers (SAE) standard J1979 [23] [24]. We did not, however, verify this as our research requires raw CAN data from the internal CAN bus itself, and because we did not want to affect the operation of the vehicle by transmitting frames on the CAN bus ourselves. We therefore concluded that these cars were not usable for our research. From the 2006 Audi A4 and 2017 Ford Fiesta we were able to log meaningful CAN frames. For both models, we logged CAN datasets of approximately 70 minutes. The characteristics of these datasets are outlined in the next section.

B. Data characteristics

The Audi dataset that we logged ourselves contained 4,705,115 CAN frames in total from 31 unique CAN IDs from a log duration of 3887.01592 seconds (approximately 64.7 minutes). The mean CAN frames per second therefore amounts to 1210.470 CAN frames per second. The mean frequency of the CAN IDs throughout the dataset ranged from 10.063 milliseconds (CAN ID 0x540) to 1.009 second (CAN ID 0x580). All CAN IDs occurred regularly throughout the entire dataset.

The Ford dataset contained 8,935,169 CAN frames in total from 54 unique CAN IDs from a log duration of 4342.73331 seconds (approximately 72.4 minutes). The mean CAN frames per second for this dataset is therefore approximately 70% higher than the Audi with 2057.499 CAN frames per second. A reason for this could be that the Ford Fiesta is a newer vehicle and therefore contains more ECUs that are connected to the CAN bus. These ECUs may send a larger variety information over the bus and will therefore require more CAN IDs to distinguish between various types of information.

The dataset contained two CAN IDs (0x728, 0x720) that both occurred 257 times throughout the dataset in an irregular pattern. Two CAN IDs (0x72F, 0x727) occurred regularly but had a mean interval time between frames of 433.366 seconds. Two other CAN IDs (0x72F, 0x727) also occurred regularly but both had a mean interval time of 205.308 seconds. The mean frequency of the other CAN IDs (the ones that occurred regularly and had a higher frequency than the previously mentioned CAN IDs) had a mean frequency range of 9.561 milliseconds (CAN ID 0x240) to 16.702 seconds (CAN ID 0x728).

Both datasets were recorded while driving in city and motorway settings at various speeds applicable to those driving conditions (max. 135 km/h). All car features were used as they would be used under normal circumstances (e.g. electric windows, indicators, etc.).

Upon inspection of the Ford dataset, we decided not to use the Ford dataset for our experiments. This is because our algorithm requires CAN frames to be sent out regularly, i.e. at approximately the same interval time throughout the dataset.

C. Robust Covariance Estimator (RCE)

The RCE algorithm evaluated by [4] is an one-class classification algorithm. With this type of algorithm, a classifier constructs a model describing the behavior in the environment where the data is obtained from. In our research, the classifier

is trained on real, collected CAN data from a 2006 Audi A4. We are unable to guarantee the car is not hacked before or during the data collection. However, since the probability of this specific vehicle being hacked at that particular time is very small, we assume that the data is collected in a non-attack environment. We assume training a classifier on this data results in a model for the classification of CAN data in a non-attack environment. If the model decides that the monitored data does not belong to this class, it is an anomaly which means an attack on the car would be occurring at that time.

Rousseeuw and Driessen [25] devised the Robust Covariance Estimator which works by fitting a robust covariance estimate to the training data, which can be seen as an estimation of the data distribution. When monitoring new data, it calculates the Mahalanobis distance to decide if the data is normal behavior, or an anomaly. Given the estimation of the data shape, the Mahalanobis distance is the distance between an observation x and the mean of the training data. The higher the distance, the greater the chance an observation is an anomaly. It can be calculated using the following formula:

$$D_M(x) = \sqrt{(x-\mu)' S^{-1} (x-\mu)}$$

In this formula the μ is the mean and S is the covariance matrix of the distribution.

D. Data preprocessing

Our approach concerning the division of CAN IDs into three categories has been the same as used by [4]. We divided the CAN IDs into three different bins based on their mean interval time computed over the first two occurrences. CAN IDs with a mean interval time $\mu_t \leq 50$ ms were categorized as *fast* IDs. Similarly, CAN IDs with $50 < \mu_t \leq 250$ ms were grouped as *medium* IDs. Lastly, CAN IDs with $\mu_t > 250$ ms were classed as *slow* IDs. As proposed by [4], the window sizes and window offsets were different for each ID type. We have used a window size of 0.2 seconds for fast IDs, 0.5 seconds for medium IDs, and a window size of 8 seconds for slow IDs. The window offset was defined as half the window size in all cases.

The RCE algorithm used by [4] only analyses the mean interval time per CAN ID and window. Considering that our algorithm requires a feature matrix as input, the data we have acquired needed preprocessing before it could be used by the algorithm. The input for the algorithm is a matrix containing one feature vector for each window in the data set. Each feature in the vector is the mean interval time for frames with the same CAN ID. A window containing three different CAN IDs would have a feature vector as shown below.

$$FV_{window_n} = \begin{bmatrix} \mu_{ID_1} \\ \mu_{ID_2} \\ \mu_{ID_3} \end{bmatrix}$$

V. Experiments

To train the model we divided the 2006 Audi A4 CAN dataset we acquired into train and test sets. To be able the measure the influence of the amount of training data, we tested classifiers trained on different training set sizes: 2, 5, 10, 20, 30, and 45 minutes of CAN data. For each model three classifiers were trained, one for each ID type. Considering that the algorithm proposed by [4] used classifiers that were trained on approximately two minutes of CAN data, we have also chosen to start testing on this amount of training data to see if the increase has influence on the performance of the RCE.

A. Simulating Attacks

To measure the performance of the RCE we have simulated twelve different attacks by altering the test set. Each attack was performed with one CAN ID from the fast, medium, and slow ID types. For each attack, the same three CAN IDs were used, which were chosen at random before simulating the attacks. Considering an adversary wishes to manipulate vehicle functions, this can be achieved by either injecting a frame with a spoofed ID from a compromised ECU, or suspending a frame transmission of a compromised ECU.

From this perspective, [14] created the so-called adversary model containing three types of attacks: fabrication, suspension and masquerade attacks. Using Python scripts, we simulated these attacks based on this adversary model. In all attacks, we assume an adversary has already compromised an ECU and we assume that there are no other ECUs in the system that send out frames with the CAN ID that we use to simulate our attacks. For each type of attack we created different attack sizes: one frame, 25 frames, and depending on the CAN ID type 2500 (fast), 500 (medium), or 50 (slow) frames. The number of frames of the largest attacks is roughly a third of all the frames for the specific ID type in the test set. Logically, attacks on one frame will have a significantly lower impact on the mean interval time than attacks on more frames. We deem attacks on more frames to be more realistic. For example, let us assume that an attacker wishes to execute an attack in which frames with ID 0x540 are altered. As noted in Section IV-A, the frequency of CAN ID 0x540 is around 10 ms. If the attack has a duration of one second, this suggests that around 100 frames are tampered with.

1) Fabrication attack: In a fabrication attack, the objective is to override a legitimate frame by fabricating a new CAN frame with a spoofed ID and forged data and insert this frame onto the CAN bus [14]. This attack frame would either occur before or after a legitimate CAN frame with the same ID in a CAN data log. We have chosen to only simulate a fabrication attack in which a frame is inserted before a legitimate frame with the same ID. To simulate this type of attack on one frame, we inserted a frame 200 μs before the frame with the same CAN ID. In [4], the value of 200 μs was also used for this attack, and the algorithm was able to detect it. Therefore, we have chosen to also use this value. In a fabrication attack, the attack frame is usually sent with a higher frequency than the legitimate frames containing this ID [14]. To simulate a

fabrication attack of larger size, we added multiple frames with a frequency that is ten times higher than the frequency of the original frame.

2) Suspension attack: The objective of a suspension attack is, for example, preventing the delivery of information to other ECUs [14]. To simulate this type of attack on one frame, we simply removed one frame from the test set. For the larger attack sizes, we removed the specific amount of frames in sequence.

3) Masquerade attack: In a masquerade attack, two ECUs are compromised. However, the objective is that one ECU is used to mask the fact that the other ECU is compromised. To perform an attack on a CAN frame with ID x that is sent from ECU A during normal operation, ECU B first observes the frames with CAN ID x sent by ECU A to determine the frame frequency. After a certain time t, the attacker stops ECU A from sending frames and starts sending frames from ECU B with ID x at the same frequency that ECU A would send frames with ID x [14]. It might seem that the frame frequency does not change in this attack, but one specific factor has to be taken into consideration. All ECUs have an internal clock that accumulates a clock skew over time as CAN does not enforce any clock synchronization between nodes [14]. This effectively means that when a CAN frame is sent from a legitimate ECU periodically, the frame frequency will slowly change in a manner that is characteristic for that particular ECU [14]. When a frame is suddenly sent from another ECU (in this case, ECU B instead of ECU A), that other ECU has a different clock skew and thus a frame frequency that, in time, will slightly vary from the frame frequency of the original ECU. [14] has concluded that it is possible to detect this, and with that, to detect masquerade attacks effectively, whereas traditional automotive IDSs are unable to. To simulate masquerade attacks, we divided the masquerade attack into two type of attacks for which the clock skew is different. Since [14] found that the clock skew is a value between 30 $\mu s\ s^{-1}$ and 460 $\mu s\ s^{-1}$, we used the maximum and minimum clock skew value as the relative clock skew for our two masquerade attack types. When both attacks are detected correctly, we assume the clock skew values in between the maximum and minimum will also be detected correctly. To simulate the masquerade attacks of one frame, we changed the timestamps of one frame by adding the relative clock skew value. For the larger attack sizes, we added the relative clock skew value accumulatively to the original timestamp of the frame. In other words, for the second frame we add the relative clock skew value twice, for the third frame thrice, and so on.

B. Algorithm Evaluation

To evalute the models in our research, we have used the balanced accuracy (bACC) as the main metric. The balanced accuracy is defined as

$$bACC = (\frac{TP}{P} + \frac{TN}{N}) \div 2$$

where TP is the number of true positives returned, P is the total number of true positives, TN is the number of true negatives returned, and N is the total number of true negatives in the dataset. Thus, the balanced accuracy is a weighted value of the average accuracy obtained on either class [26]. This means that if a classifier has equal performance on both classes, the balanced accuracy will effectively reduce to the conventional accuracy. However, if the conventional accuracy on one class is only high because the classifier classifies every sample as belonging to that class, the balanced accuracy will even this out with the misclassification of all samples that do not belong to the class, evening out to a random guess. Note that this metric also takes into account the False Negative Rate (FNR) and the False Positive Rate (FPR), seeing that these are complements:

$$FPR = 1 - TNR$$

and

$$FNR = 1 - TPR$$

VI. Results

The results of the experiments described in Section V are shown in Figure 1. These diagrams contain the balanced accuracy as metric for each attack type and size performed on three CAN ID types. For an overview of all results, including the true positives and negatives and false positives and negatives, see table I in the Appendix.

A. Medium ID type

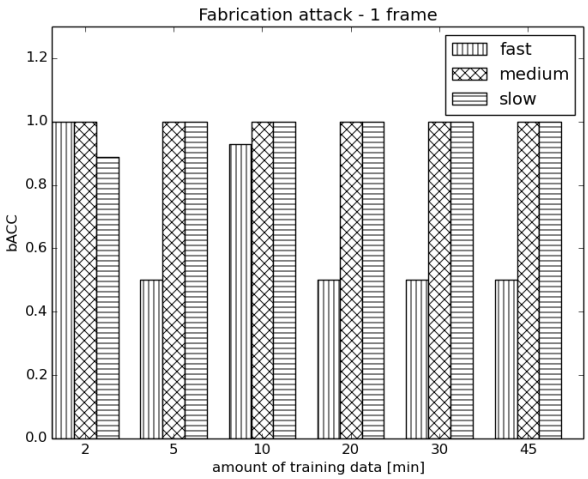
During all attacks, the amount of training data did not have any influence on the balanced accuracy for the medium ID type (see fig. 1).

B. Slow ID type

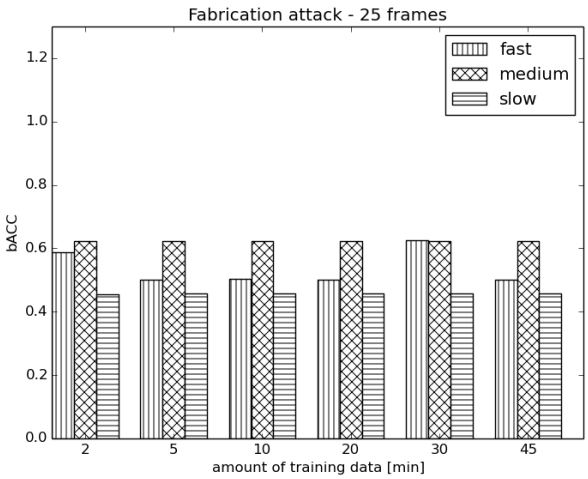
For the slow ID type, the results show the following. For all sizes of the fabrication attacks (figs. 1a to 1c) the balanced accuracy is optimal when the classifiers are trained on five or more minutes of data. The increase in balanced accuracy for the fabrication attack on one frame is significant and results in perfect classification (i.e., $bACC = 1$). However, the increase on larger attacks is negligible and still produces poor classification.

For all sizes of suspension attacks (figs. 1d to 1f), the balanced accuracy is also highest when the classifiers are trained on five or more minutes of data, with one exception. In the suspension attack with 25 frames (fig. 1e), the balanced accuracy is lower when training on 45 minutes of data than training on 5, 10, 20, or 30 minutes. This may be the result of overfitting. Overfitting is the phenomenon in which a model corresponds too closely to training data and is therefore unable to generalize accurately to new observations [27]. In all situations except for this one, the models that are trained on 5 or more minutes of data are able to classify all data perfectly.

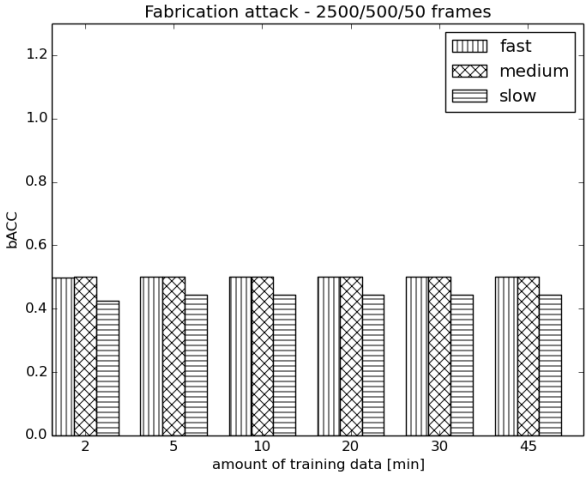
For the masquerade attacks with a large relative clock skew, the balanced accuracy is decreased when training on more



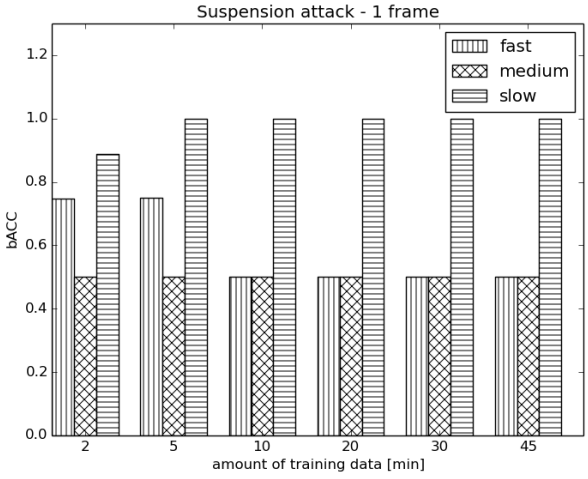
(a)



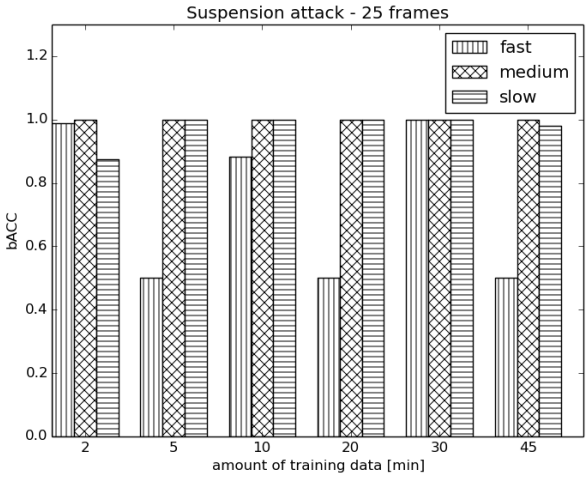
(b)



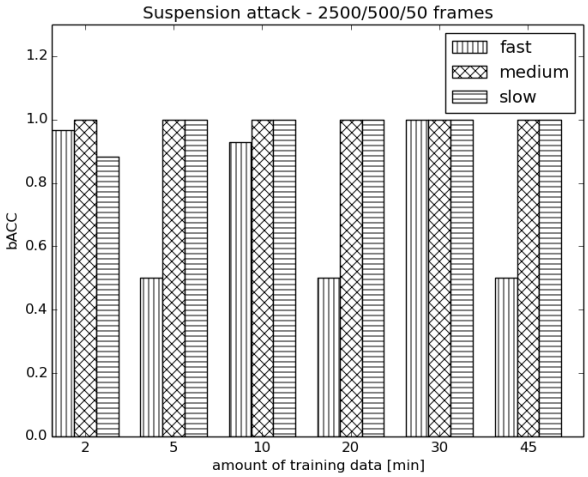
(c)



(d)

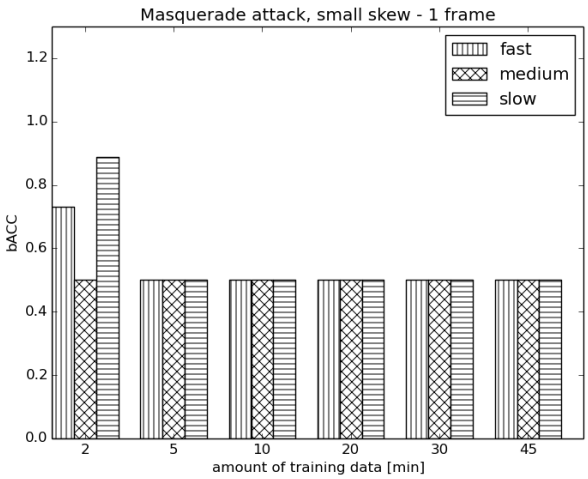


(e)

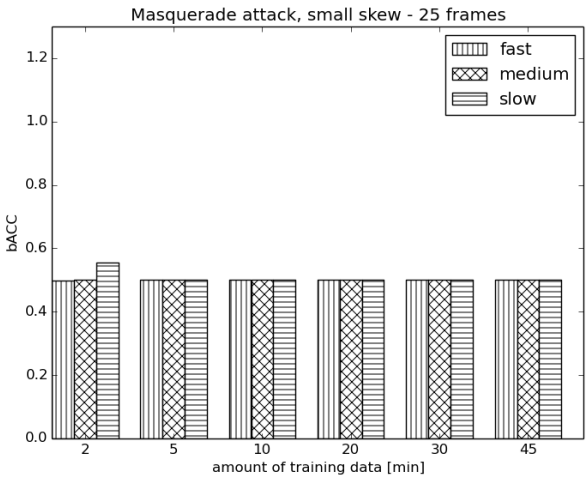


(f)

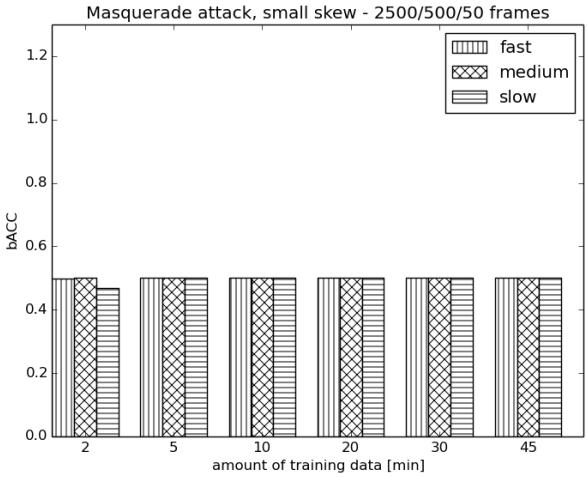
Fig. 1: Results of our experiments for the different attacks.



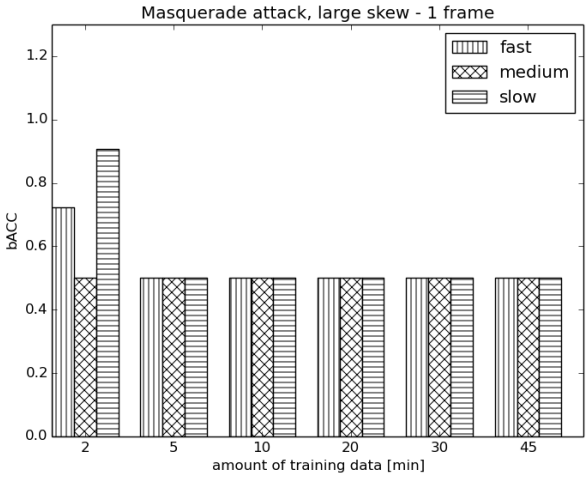
(g)



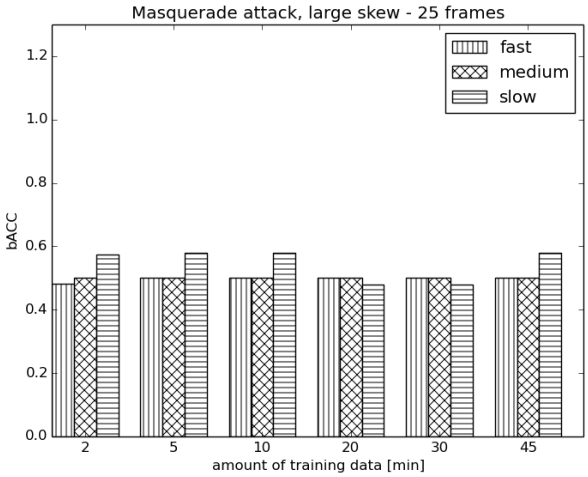
(h)



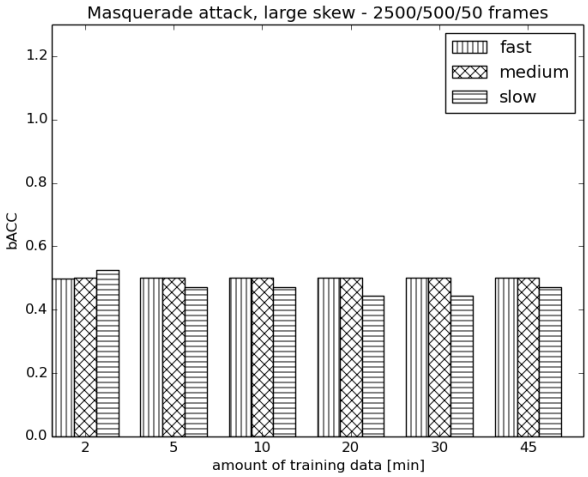
(i)



(j)



(k)



(l)

Fig. 1: Results of our experiments for the different attacks (continued).

than 2 minutes of data for an attack of one frame (fig. 1j). For the attacks on more frames (figs. 1k and 1l), the results are inconclusive. The optimal amount of training data for the attack on 25 frames is 5 or 45 minutes. The optimal amount of training data for the attack on 50 frames is 2 minutes. In both cases the balanced accuracy is approximately 0.5, which means the algorithm performs just as well as a random guess (see Section V-A3).

Finally, for the masquerade attack with a small relative clock skew, the balanced accuracy is at best when training on 2 minutes of data for both an attack of one and 25 frames (figs. 1g and 1h). For the attack on 50 frames (fig. 1i) the performance is at best when training on equal or more than 5 minutes of data, however the increase is negligible.

C. Fast ID type

For the fast ID type, we see the following results. For a fabrication attack on one frame (fig. 1a) the results are inconclusive, with the balanced accuracy at best when training on 2 minutes of data. For an attack on 25 frames (fig. 1b), the results are also inconclusive with the best performance on 30 minutes of training data. The attack of 2500 frames (fig. 1c) result in a balanced accuracy of approximately 0.5 for all different sizes of training data. Only the performance of the model trained on 2 minutes of data classifies all data when the attack of 1 frame is tested perfectly.

For the suspension attacks, the classifier trained on 5 minutes of data produces the highest balanced accuracy for the attack on one frame (fig. 1d). For the attacks of 25 and 2500 frames (figs. 1e and 1f), we see inconclusive results with 30 minutes as optimal training data.

Finally, for both the masquerade attacks with a large and small relative clock skew on one frame (figs. 1g and 1j), the balanced accuracy is decreased the models have been trained on more than 5 minutes of data. For both attack types on 25 and 2500 frames (figs. 1h, 1i, 1k and 1l), the performance is increased when training on more than 5 minutes of data. All results of the masquerade attacks show a balanced accuracy around 0.5 which is considered poor classification since it is as poor as gambling.

VII. Discussion

By evaluating our results we see the amount of training data does not have any effect on any attacks using CAN ID 0x572, which is a medium ID type. For attacks using a different CAN ID type, it depends on the attack type and CAN ID used to determine whether the influence is beneficial or not. When evaluating the performance of various amounts of training data for the fast ID type, we occasionally see inconclusive results. We have seen that some attacks can be perfectly classified (i.e., the bACC = 1 for some experiments). Assuming that this accuracy remains identical when datasets from other vehicles are used, these classifiers could prove valuable to detect these attacks in real-life applications. However, it is to be noted that for other tests with the same classifiers, the balanced accuracy is high (e.g., 0.888), and in these tests all positives (i.e., attack windows) are detected correctly, but there are also negatives that are being detected as positives (i.e., false positives). In a real-life application, the false positives may trigger a warning frame to the driver of a vehicle. Intrusion detection systems require an extremely low false alarm rate, i.e. the number of false positives must be close to zero. For example, let us assume that in some system, one false positive window is classified as an attack window for every 10,000 windows, and the window size is 0.2 seconds. This means that one false positive is classified roughly every 33 minutes. Users of the system will then quickly learn to ignore the warning frames generated by the system, even though the false positive rate is 1 out of 10,000. Hence, a high balanced accuracy does not directly result in a viable real-life application.

There are some limitations to the methodology used in this research. First of all, in this research only three CAN IDs (one fast, one medium, and one slow ID) are used to simulate different attacks. This was done to ensure that the results for the different attacks could be compared as the CAN ID used would not be a variable in the experiments. However, a practical implementation of the RCE algorithm should be able to detect attacks on all possible CAN IDs. Our research can not evaluate this explicitly.

Secondly, this algorithm cannot provide any more information about the attack other than a time frame in which the attack is happening, since the output is a window and its classification. It could be useful to have more information about the attack, for example the CAN ID of the attack frame, or the type of attack. More specific information may be valuable in determining the extent and severity of an attack, and appropriate countermeasures that may be taken to limit the damage caused by an attack. This may be achieved by training a multi-class machine learning algorithm on specific types of attacks, that is then able to output an attack classification.

Furthermore, our algorithm is not able to detect attacks on non-recurring CAN frames because a mean interval time cannot be computed for these frames. This may be solved by using a separate, different classifier for frames that are non-recurring. Features for that classifier may include the CAN data field.

Since the algorithm divides the CAN IDs into three separate categories, boundaries for these categories need to be defined beforehand. Since we have discovered that the interval time ranges for CAN data in an 2006 Audi A4 and 2017 Ford Fiesta are different, this implies that the categories must be determined for each model of vehicle individually. A practical implementation of the algorithm may adopt an approach in which an exploration phase is used in which vehicle-specific data such as the number of CAN IDs, their recurrence and their mean time intervals is collected before selecting appropriate bounds for the categories.

Finally, this algorithm does not utilize the data field inside a CAN frame when classifying a frame as an anomaly. This means attacks will not be detected if an attacker is able to compromise an ECU and send out frames without altering the

frequency of those frames. To successfully execute such an attack, the attacker must also take the different clock skews into account.

VIII. Conclusion

As modern cars feature more interconnected, electronically controlled systems, the potential safety hazards that arise from automotive attacks have become more important as well. Automotive intrusion detection systems can play a significant part in limiting the damage and harm that is caused by attacks on cars if they are able to accurate detect attacks in an early stage.

Our research aimed to evaluate the influence of the amount of training data on the performance of the RCE algorithm when used for automotive CAN anomaly detection. We have researched the possibility of collecting real automotive CAN data with timestamps in microsecond accuracy using the PCAN-USB FD connector, which has proven to be successful for a 2006 Audi A4 and a 2017 Ford Fiesta. Analysis of the characteristics of these collected datasets showed that the data characteristics between both models differ in the number of distinct CAN IDs and the mean time intervals of frames with those IDs. We have performed tests in which fabrication attacks, suspension attacks and masquerade attacks were simulated using the dataset collected from the Audi vehicle. The results show that for medium type IDs, the amount of training data ranging from 2 to 45 minutes of CAN data has no effect on the balanced accuracy of the RCE. For attacks on other ID types, the performance differs between types of attacks. With a few exceptions, the general performance of the RCE does not differ significantly when training on more than 5 minutes of training data.

IX. Future work

For this algorithm to be suitable for real-life applications, it requires more research to improve the detection of different attack types. Further research could look into the possibilities to improve the algorithm by training more classifiers. This could help each classifier focus more on a smaller interval range between frames with the same CAN ID, and therefore detecting anomalies with an higher accuracy. Turning this algorithm into a proof-of-concept, it needs to accept an input stream of data from the CAN bus instead of reading it from a file. Future research could investigate in possible methods to do so.

References

[1] William Chou et al. *Semiconductors – the Next Wave: Opportunities and winning strategies for semiconductor companies*. Tech. rep. Deloitte China, Apr. 2019, p. 14. URL: https://www2.deloitte.com/content/dam/Deloitte/cn/Documents/technology-media-telecommunications/deloitte-cn-tmt-semiconductors-the-next-wave-en-190422.pdf.

[2] *CAN Specification, version 2.0*. Tech. rep. Stuttgart, Germany: Robert Bosch GmbH, Sept. 1991. URL: http://esd.cs.ucr.edu/webres/can20.pdf.

[3] International Organization for Standardization (ISO). *Road vehicles — Controller area network (CAN)*. 2003. URL: http://read.pudn.com/downloads209/ebook/986064/ISO%2011898/ISO%2011898-1.pdf.

[4] H. Hambartsumyan C.N.I.W. Schappin N. Zannone. "Intrustion Detection on the Automotive CAN bus". In: (2017).

[5] *Next Generation Car Network - FlexRay*. Tech. rep. Shanghai, China: Fujitsu Microelectronics (Shanghai) Co.,Ltd., June 2006. URL: https://www.fujitsu.com/downloads/CN/fmc/lsi/FlexRay-EN.pdf.

[6] *FlexRay Automotive Communication Bus Overview*. Tech. rep. Austin, TX, United States: National Instruments Corporation, May 2019. URL: https://www.ni.com/nl-nl/innovations/white-papers/06/flexray-automotive-communication-bus-overview.html.

[7] Hans-Christian von der Wense. "Introduction to Local Interconnect Network". In: *SAE Transactions* 109 (2000), pp. 87–91. ISSN: 0096736X, 25771531. URL: http://www.jstor.org/stable/44699113.

[8] B.T. Fijalkowski. "Media Oriented System Transport (MOST) Networking". In: *Automotive Mechatronics: Operational and Practical Issues: Volume I*. Dordrecht: Springer Netherlands, 2011, pp. 73–74. ISBN: 978-94-007-0409-1. DOI: 10.1007/978-94-007-0409-1_10. URL: https://doi.org/10.1007/978-94-007-0409-1_10.

[9] Peter Hank, Thomas Suermann, and Steffen Müller. "Automotive Ethernet, a Holistic Approach for a Next Generation In-Vehicle Networking Standard". In: *Advanced Microsystems for Automotive Applications 2012*. Ed. by Gereon Meyer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 79–89. ISBN: 978-3-642-29673-4.

[10] H. M. Song, H. R. Kim, and H. K. Kim. "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network". In: *2016 International Conference on Information Networking (ICOIN)*. Jan. 2016, pp. 63–68. DOI: 10.1109/ICOIN.2016.7427089.

[11] Michael R. Moore et al. "Modeling Inter-Signal Arrival Times for Accurate Detection of CAN Bus Signal Injection Attacks: A Data-Driven Approach to in-Vehicle Intrusion Detection". In: *Proceedings of the 12th Annual Conference on Cyber and Information Security Research*. CISRC '17. Oak Ridge, Tennessee, USA: Association for Computing Machinery, 2017. ISBN: 9781450348553. DOI: 10.1145/3064814.3064816. URL: https://doi.org/10.1145/3064814.3064816.

[12] M. Gmiden, M. H. Gmiden, and H. Trabelsi. "An intrusion detection method for securing in-vehicle CAN bus". In: *2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. Dec. 2016, pp. 176–180. DOI: 10.1109/STA.2016.7952095.

[13] A. Taylor, N. Japkowicz, and S. Leblanc. "Frequency-based anomaly detection for the automotive CAN bus".

In: *2015 World Congress on Industrial Control Systems Security (WCICSS)*. Dec. 2015, pp. 45–49. DOI: 10.1109/WCICSS.2015.7420322.

[14] Kyong-Tak Cho and Kang G. Shin. "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection". In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 911–927. ISBN: 978-1-931971-32-4. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/cho.

[15] M. Müter and N. Asaj. "Entropy-based anomaly detection for in-vehicle networks". In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. June 2011, pp. 1110–1115. DOI: 10.1109/IVS.2011.5940552.

[16] Moti Markovitz and Avishai Wool. "Field classification, modeling and anomaly detection in unknown CAN bus networks". In: *Vehicular Communications* 9 (2017), pp. 43–52.

[17] H. Olufowobi et al. "Work-in-Progress: Real-Time Modeling for Intrusion Detection in Automotive Controller Area Network". In: *2018 IEEE Real-Time Systems Symposium (RTSS)*. Dec. 2018, pp. 161–164. DOI: 10.1109/RTSS.2018.00030.

[18] Robert N. Charette. *This Car Runs on Code*. Feb. 2009. URL: https://spectrum.ieee.org/transportation/systems/this-car-runs-on-code.

[19] University of Tulsa Crash Reconstruction Research Consortium. *Dodge CAN Messages*. URL: http://tucrrc.utulsa.edu/DodgeCAN.html.

[20] Council of European Union. *Directive 98/69/EC of the European Parliament and of the Council*. 1998.

[21] F. Sagstetter et al. "Security challenges in automotive hardware/software architecture design". In: *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*. Mar. 2013, pp. 458–463. DOI: 10.7873/DATE.2013.102.

[22] PEAK-System Technik GmbH. "PCAN-USB FD User Manual". In: (2019). URL: https://www.peak-system.com/PCAN-USB.199.0.html?&L=1.

[23] Wolfhard Lawrenz. *CAN System Engineering: From Theory to Practical Applications*. Springer London, 2013, p. 231.

[24] *SAE J1979 E/E Diagnostic Test Modes / ISO 15031-5:2015 Road vehicles - Communication between vehicle and external equipment for emissions-related diagnostics - Part 5: Emissions-related diagnostic services*. Standard. Troy, MI, United States of America / Geneva, Switzerland: Society of Automotive Engineers (SAE) / International Organization for Standardization (ISO), Feb. 2012.

[25] Peter J Rousseeuw and Katrien Van Driessen. "A Fast Algorithm for the Minimum Covariance Determinant Estimator". eng. In: *Technometrics* 41.3 (1999-08-01), pp. 212, 223. ISSN: 0040-1706. URL: http://www.tandfonline.com/doi/abs/10.1080/00401706.1999.10485670.

[26] K. H. Brodersen et al. "The Balanced Accuracy and Its Posterior Distribution". In: *2010 20th International Conference on Pattern Recognition*. Aug. 2010, pp. 3121–3124. DOI: 10.1109/ICPR.2010.764.

[27] Douglas M. Hawkins. "The Problem of Overfitting". In: *Journal of Chemical Information and Computer Sciences* 44.1 (2004). PMID: 14741005, pp. 1–12. DOI: 10.1021/ci0342472. eprint: https://doi.org/10.1021/ci0342472. URL: https://doi.org/10.1021/ci0342472.

APPENDIX

TABLE I: The raw results of our experiments.

t_{train}	Attack	Freq	ID type	n_{train}	n_{test}	$n_{outlier}$	TP	TN	FP	FN	bACC
2	fabrication	1	fast	1193	1199	2	2	1197	0	0	1.00000
5	fabrication	1	fast	2981	1199	2	0	1197	0	2	0.50000
10	fabrication	1	fast	5961	1199	2	2	1030	167	0	0.93024
20	fabrication	1	fast	11922	1199	2	0	1197	0	2	0.50000
30	fabrication	1	fast	17882	1199	2	0	1197	0	2	0.50000
45	fabrication	1	fast	26823	1199	2	0	1197	0	2	0.50000
2	fabrication	1	medium	238	239	2	2	237	0	0	1.00000
5	fabrication	1	medium	596	239	2	2	237	0	0	1.00000
10	fabrication	1	medium	1192	239	2	2	237	0	0	1.00000
20	fabrication	1	medium	2384	239	2	2	237	0	0	1.00000
30	fabrication	1	medium	3576	239	2	2	237	0	0	1.00000
45	fabrication	1	medium	5364	239	2	2	237	0	0	1.00000
2	fabrication	1	slow	29	29	2	2	21	6	0	0.88889
5	fabrication	1	slow	74	29	2	2	27	0	0	1.00000
10	fabrication	1	slow	149	29	2	2	27	0	0	1.00000
20	fabrication	1	slow	298	29	2	2	27	0	0	1.00000
30	fabrication	1	slow	447	29	2	2	27	0	0	1.00000
45	fabrication	1	slow	670	29	2	2	27	0	0	1.00000
2	fabrication	25	fast	1193	1199	4	1	1108	87	3	0.58860
5	fabrication	25	fast	2981	1199	4	0	1195	0	4	0.50000
10	fabrication	25	fast	5961	1199	4	1	904	291	3	0.50324
20	fabrication	25	fast	11922	1199	4	0	1195	0	4	0.50000
30	fabrication	25	fast	17882	1199	4	1	1195	0	3	0.62500
45	fabrication	25	fast	26823	1199	4	0	1195	0	4	0.50000
2	fabrication	25	medium	238	239	4	1	234	1	3	0.62287
5	fabrication	25	medium	596	239	4	1	234	1	3	0.62287
10	fabrication	25	medium	1192	239	4	1	234	1	3	0.62287
20	fabrication	25	medium	2384	239	4	1	234	1	3	0.62287
30	fabrication	25	medium	3576	239	4	1	234	1	3	0.62287
45	fabrication	25	medium	5364	239	4	1	234	1	3	0.62287
2	fabrication	25	slow	29	29	5	1	17	7	4	0.45417
5	fabrication	25	slow	74	29	5	0	22	2	5	0.45833
10	fabrication	25	slow	149	29	5	0	22	2	5	0.45833
20	fabrication	25	slow	298	29	5	0	22	2	5	0.45833
30	fabrication	25	slow	447	29	5	0	22	2	5	0.45833

Table I continued from previous page

t_{train}	Attack	Freq	ID type	n_{train}	n_{test}	$n_{outlier}$	TP	TN	FP	FN	bACC
45	fabrication	25	slow	670	29	5	0	22	2	5	0.45833
2	fabrication	2500	fast	1193	1199	501	1	695	3	500	0.49885
5	fabrication	2500	fast	2981	1199	500	0	698	0	500	0.50000
10	fabrication	2500	fast	5961	1199	501	501	0	698	0	0.50000
20	fabrication	2500	fast	11922	1199	501	0	698	0	501	0.50000
30	fabrication	2500	fast	17882	1199	500	1	698	0	500	0.50100
45	fabrication	2500	fast	26823	1199	500	0	699	0	500	0.50000
2	fabrication	500	medium	238	239	99	1	139	1	98	0.50148
5	fabrication	500	medium	596	239	99	1	139	1	98	0.50148
10	fabrication	500	medium	1192	239	99	1	139	1	98	0.50148
20	fabrication	500	medium	2384	239	99	1	139	1	98	0.50148
30	fabrication	500	medium	3576	239	99	1	139	1	98	0.50148
45	fabrication	500	medium	5364	239	99	1	139	1	98	0.50148
2	fabrication	50	slow	29	29	11	2	12	6	9	0.42424
5	fabrication	50	slow	74	29	11	0	16	2	11	0.44444
10	fabrication	50	slow	149	29	11	0	16	2	11	0.44444
20	fabrication	50	slow	298	29	11	0	16	2	11	0.44444
30	fabrication	50	slow	447	29	11	0	16	2	11	0.44444
45	fabrication	50	slow	670	29	11	0	16	2	11	0.44444
2	suspension	1	fast	1193	1199	2	1	1192	5	1	0.74791
5	suspension	1	fast	2981	1199	2	1	1197	0	1	0.75000
10	suspension	1	fast	5961	1199	2	2	0	1197	0	0.50000
20	suspension	1	fast	11922	1199	2	0	1197	0	2	0.50000
30	suspension	1	fast	17882	1199	2	0	1197	0	2	0.50000
45	suspension	1	fast	26823	1199	2	0	1197	0	2	0.50000
2	suspension	1	medium	238	239	2	0	237	0	2	0.50000
5	suspension	1	medium	596	239	2	0	237	0	2	0.50000
10	suspension	1	medium	1192	239	2	0	237	0	2	0.50000
20	suspension	1	medium	2384	239	2	0	237	0	2	0.50000
30	suspension	1	medium	3576	239	2	0	237	0	2	0.50000
45	suspension	1	medium	5364	239	2	0	237	0	2	0.50000
2	suspension	1	slow	29	29	2	2	21	6	0	0.88889
5	suspension	1	slow	74	29	2	2	27	0	0	1.00000
10	suspension	1	slow	149	29	2	2	27	0	0	1.00000
20	suspension	1	slow	298	29	2	2	27	0	0	1.00000
30	suspension	1	slow	447	29	2	2	27	0	0	1.00000
45	suspension	1	slow	670	29	2	2	27	0	0	1.00000

Table I continued from previous page

t_{train}	Attack	Freq	ID type	n_{train}	n_{test}	$n_{outlier}$	TP	TN	FP	FN	bACC
2	suspension	25	fast	1193	1199	4	4	1168	27	0	0.98870
5	suspension	25	fast	2981	1199	4	0	1195	0	4	0.50000
10	suspension	25	fast	5961	1199	4	4	918	277	0	0.88410
20	suspension	25	fast	11922	1199	4	0	1195	0	4	0.50000
30	suspension	25	fast	17882	1199	4	4	1195	0	0	1.00000
45	suspension	25	fast	26823	1199	4	0	1195	0	4	0.50000
2	suspension	25	medium	238	239	4	4	235	0	0	1.00000
5	suspension	25	medium	596	239	4	4	235	0	0	1.00000
10	suspension	25	medium	1192	239	4	4	235	0	0	1.00000
20	suspension	25	medium	2384	239	4	4	235	0	0	1.00000
30	suspension	25	medium	3576	239	4	4	235	0	0	1.00000
45	suspension	25	medium	5364	239	4	4	235	0	0	1.00000
2	suspension	25	slow	29	29	5	5	18	6	0	0.87500
5	suspension	25	slow	74	29	5	5	24	0	0	1.00000
10	suspension	25	slow	149	29	5	5	24	0	0	1.00000
20	suspension	25	slow	298	29	5	5	24	0	0	1.00000
30	suspension	25	slow	447	29	5	5	24	0	0	1.00000
45	suspension	25	slow	670	29	5	5	23	1	0	0.97917
2	suspension	2500	fast	1193	1199	501	500	653	45	1	0.96677
5	suspension	2500	fast	2981	1199	501	0	698	0	501	0.50000
10	suspension	2500	fast	5961	1199	500	500	600	99	0	0.92918
20	suspension	2500	fast	11922	1199	500	0	699	0	500	0.50000
30	suspension	2500	fast	17882	1199	501	500	698	0	1	0.99900
45	suspension	2500	fast	26823	1199	500	0	699	0	500	0.50000
2	suspension	500	medium	238	239	99	99	140	0	0	1.00000
5	suspension	500	medium	596	239	99	99	140	0	0	1.00000
10	suspension	500	medium	1192	239	99	99	140	0	0	1.00000
20	suspension	500	medium	2384	239	99	99	140	0	0	1.00000
30	suspension	500	medium	3576	239	99	99	140	0	0	1.00000
45	suspension	500	medium	5364	239	99	99	140	0	0	1.00000
2	suspension	50	slow	29	29	12	12	13	4	0	0.88235
5	suspension	50	slow	74	29	12	12	17	0	0	1.00000
10	suspension	50	slow	149	29	12	12	17	0	0	1.00000
20	suspension	50	slow	298	29	12	12	17	0	0	1.00000
30	suspension	50	slow	447	29	12	12	17	0	0	1.00000
45	suspension	50	slow	670	29	12	12	17	0	0	1.00000

Table I continued from previous page

t_{train}	Attack	Freq	ID type	n_{train}	n_{test}	$n_{outlier}$	TP	TN	FP	FN	bACC
2	masq - ss[1]	1	fast	1193	1199	2	1	1155	42	1	0.73246
5	masq - ss	1	fast	2981	1199	2	0	1197	0	2	0.50000
10	masq - ss	1	fast	5961	1199	2	2	0	1197	0	0.50000
20	masq - ss	1	fast	11922	1199	2	0	1197	0	2	0.50000
30	masq - ss	1	fast	17882	1199	2	0	1197	0	2	0.50000
45	masq - ss	1	fast	26823	1199	2	0	1197	0	2	0.50000
2	masq - ss	1	medium	238	239	2	0	237	0	2	0.50000
5	masq - ss	1	medium	596	239	2	0	237	0	2	0.50000
10	masq - ss	1	medium	1192	239	2	0	237	0	2	0.50000
20	masq - ss	1	medium	2384	239	2	0	237	0	2	0.50000
30	masq - ss	1	medium	3576	239	2	0	237	0	2	0.50000
45	masq - ss	1	medium	5364	239	2	0	237	0	2	0.50000
2	masq - ss	1	slow	29	29	2	2	21	6	0	0.88889
5	masq - ss	1	slow	74	29	2	0	27	0	2	0.50000
10	masq - ss	1	slow	149	29	2	0	27	0	2	0.50000
20	masq - ss	1	slow	298	29	2	0	27	0	2	0.50000
30	masq - ss	1	slow	447	29	2	0	27	0	2	0.50000
45	masq - ss	1	slow	670	29	2	0	27	0	2	0.50000
2	masq - ss	25	fast	1193	1199	4	0	1193	2	4	0.49916
5	masq - ss	25	fast	2981	1199	4	0	1195	0	4	0.50000
10	masq - ss	25	fast	5961	1199	4	4	0	1195	0	0.50000
20	masq - ss	25	fast	11922	1199	4	0	1195	0	4	0.50000
30	masq - ss	25	fast	17882	1199	4	0	1195	0	4	0.50000
45	masq - ss	25	fast	26823	1199	4	0	1195	0	4	0.50000
2	masq - ss	25	medium	238	239	4	0	235	0	4	0.50000
5	masq - ss	25	medium	596	239	4	0	235	0	4	0.50000
10	masq - ss	25	medium	1192	239	4	0	235	0	4	0.50000
20	masq - ss	25	medium	2384	239	4	0	235	0	4	0.50000
30	masq - ss	25	medium	3576	239	4	0	235	0	4	0.50000
45	masq - ss	25	medium	5364	239	4	0	235	0	4	0.50000
2	masq - ss	25	slow	29	29	5	2	17	7	3	0.55417
5	masq - ss	25	slow	74	29	5	0	24	0	5	0.50000
10	masq - ss	25	slow	149	29	5	0	24	0	5	0.50000
20	masq - ss	25	slow	298	29	5	0	24	0	5	0.50000
30	masq - ss	25	slow	447	29	5	0	24	0	5	0.50000
45	masq - ss	25	slow	670	29	5	0	24	0	5	0.50000

[1] *masq - ss* refers to the masquerade attack that uses a small clock skew as defined in Section V-A2.

Table I continued from previous page

t_{train}	Attack	Freq	ID type	n_{train}	n_{test}	$n_{outlier}$	TP	TN	FP	FN	bACC
2	masq - ss	2500	fast	1193	1199	501	1	696	2	500	0.49957
5	masq - ss	2500	fast	2981	1199	501	0	698	0	501	0.50000
10	masq - ss	2500	fast	5961	1199	500	500	0	699	0	0.50000
20	masq - ss	2500	fast	11922	1199	501	0	698	0	501	0.50000
30	masq - ss	2500	fast	17882	1199	501	0	698	0	501	0.50000
45	masq - ss	2500	fast	26823	1199	500	0	698	0	500	0.50000
2	masq - ss	500	medium	238	239	99	0	140	0	99	0.50000
5	masq - ss	500	medium	596	239	99	0	140	0	99	0.50000
10	masq - ss	500	medium	1192	239	99	0	140	0	99	0.50000
20	masq - ss	500	medium	2384	239	99	0	140	0	99	0.50000
30	masq - ss	500	medium	3576	239	99	0	140	0	99	0.50000
45	masq - ss	500	medium	5364	239	99	0	140	0	99	0.50000
2	masq - ss	50	slow	29	29	11	3	12	6	8	0.46970
5	masq - ss	50	slow	74	29	11	0	18	0	11	0.50000
10	masq - ss	50	slow	149	29	11	0	18	0	11	0.50000
20	masq - ss	50	slow	298	29	11	0	18	0	11	0.50000
30	masq - ss	50	slow	447	29	11	0	18	0	11	0.50000
45	masq - ss	50	slow	670	29	11	0	18	0	11	0.50000
2	masq - ls[2]	1	fast	1193	1199	2	1	1131	66	1	0.72243
5	masq - ls	1	fast	2981	1199	2	0	1197	0	2	0.50000
10	masq - ls	1	fast	5961	1199	2	2	0	1197	0	0.50000
20	masq - ls	1	fast	11922	1199	2	0	1197	0	2	0.50000
30	masq - ls	1	fast	17882	1199	2	0	1197	0	2	0.50000
45	masq - ls	1	fast	26823	1199	2	0	1197	0	2	0.50000
2	masq - ls	1	medium	238	239	2	0	237	0	2	0.50000
5	masq - ls	1	medium	596	239	2	0	237	0	2	0.50000
10	masq - ls	1	medium	1192	239	2	0	237	0	2	0.50000
20	masq - ls	1	medium	2384	239	2	0	237	0	2	0.50000
30	masq - ls	1	medium	3576	239	2	0	237	0	2	0.50000
45	masq - ls	1	medium	5364	239	2	0	237	0	2	0.50000
2	masq - ls	1	slow	29	29	2	2	26	6	0	0.90625
5	masq - ls	1	slow	74	29	2	0	27	0	2	0.50000
10	masq - ls	1	slow	149	29	2	0	27	0	2	0.50000
20	masq - ls	1	slow	298	29	2	0	27	0	2	0.50000
30	masq - ls	1	slow	447	29	2	0	27	0	2	0.50000
45	masq - ls	1	slow	670	29	2	0	27	0	2	0.50000

[2]*masq - ls* refers to the masquerade attack that uses a large clock skew as defined in Section V-A2.

Table I continued from previous page

t_{train}	Attack	Freq	ID type	n_{train}	n_{test}	$n_{outlier}$	TP	TN	FP	FN	bACC
2	masq - ls	25	fast	1193	1199	4	0	1152	43	4	0.48201
5	masq - ls	25	fast	2981	1199	4	0	1195	0	4	0.50000
10	masq - ls	25	fast	5961	1199	4	4	0	1195	0	0.50000
20	masq - ls	25	fast	11922	1199	4	0	1195	0	4	0.50000
30	masq - ls	25	fast	17882	1199	4	0	1195	0	4	0.50000
45	masq - ls	25	fast	26823	1199	4	0	1195	0	4	0.50000
2	masq - ls	25	medium	238	239	4	0	235	0	4	0.50000
5	masq - ls	25	medium	596	239	4	0	235	0	4	0.50000
10	masq - ls	25	medium	1192	239	4	0	235	0	4	0.50000
20	masq - ls	25	medium	2384	239	4	0	235	0	4	0.50000
30	masq - ls	25	medium	3576	239	4	0	235	0	4	0.50000
45	masq - ls	25	medium	5364	239	4	0	235	0	4	0.50000
2	masq - ls	25	slow	29	29	5	2	18	6	3	0.57500
5	masq - ls	25	slow	74	29	5	1	23	1	4	0.57917
10	masq - ls	25	slow	149	29	5	1	23	1	4	0.57917
20	masq - ls	25	slow	298	29	5	0	23	1	5	0.47917
30	masq - ls	25	slow	447	29	5	0	23	1	5	0.47917
45	masq - ls	25	slow	670	29	5	1	23	1	4	0.57917
2	masq - ls	2500	fast	1193	1199	501	1	694	4	500	0.49813
5	masq - ls	2500	fast	2981	1199	501	0	698	0	501	0.50000
10	masq - ls	2500	fast	5961	1199	500	500	0	699	0	0.50000
20	masq - ls	2500	fast	11922	1199	501	0	698	0	501	0.50000
30	masq - ls	2500	fast	17882	1199	501	0	698	0	501	0.50000
45	masq - ls	2500	fast	26823	1199	500	0	699	0	500	0.50000
2	masq - ls	500	medium	238	239	99	0	140	0	99	0.50000
5	masq - ls	500	medium	596	239	99	0	140	0	99	0.50000
10	masq - ls	500	medium	1192	239	99	0	140	0	99	0.50000
20	masq - ls	500	medium	2384	239	99	0	140	0	99	0.50000
30	masq - ls	500	medium	3576	239	99	0	140	0	99	0.50000
45	masq - ls	500	medium	5364	239	99	0	140	0	99	0.50000
2	masq - ls	50	slow	29	29	11	3	14	4	8	0.52525
5	masq - ls	50	slow	74	29	11	0	17	1	11	0.47222
10	masq - ls	50	slow	149	29	11	0	17	1	11	0.47222
20	masq - ls	50	slow	298	29	11	0	16	2	11	0.44444
30	masq - ls	50	slow	447	29	11	0	16	2	11	0.44444
45	masq - ls	50	slow	670	29	11	0	17	1	11	0.47222