Análisis de Supervivencia para la detección y clasificación de anomalías en vehículos

Pablo Menéndez-Trillo
GRADIANT
Estrada do Vilar, 56-58, 36214 Vigo
pmenendez@gradiant.org

Fabián Otero-Vázquez
GRADIANT
Estrada do Vilar, 56-58, 36214 Vigo
fovazquez@gradiant.org

Jacobo de Uña-Álvarez
Universidade de Vigo
Grupo SiDOR, 36310 Vigo
jacobo@uvigo.gal

***Abstract*—Hoy en día la digitalización está presente en todos los ámbitos que nos podamos imaginar, y los vehículos no son una excepción. Hace menos de 50 años la proporción de componentes electrónicas frente a mecánicas en los vehículos era inferior al 1%. Hoy esta cifra se ha elevado al 50%, y seguirá aumentando con la proliferación de los vehículos eléctricos y la aparición de los vehículos autónomos. La comunicación del vehículo con el exterior aporta facilidades para el día a día del conductor (GPS, *bluetooth*, Wi-Fi...), pero también crea vulnerabilidades que pueden ser utilizadas en su contra en forma de ciberataques. En este estudio se describen dichas vulnerabilidades y se propone un novedoso método de detección y clasificación de anomalías basado en el Análisis de Supervivencia, rama de la estadística que estudia modelos en los que la variable respuesta es el tiempo entre dos eventos que marcan el inicio y el final de un suceso.**

Index Terms*—Análisis de Supervivencia; detección de anomalías; clasificación de anomalías; *flooding*; *fuzzy*; *malfunction*; *replay*; *controller area network

Tipo de contribución: *Investigación original*

I. Introducción

Los dispositivos electrónicos de los vehículos están organizados por unidades de control electrónico (ECUs). Para su correcto funcionamiento, cada ECU debe estar en constante comunicación con el resto. El protocolo estándar de comunicación entre ECUs en los vehículos es el Controller Area Network (CAN), barato, ligero, rápido y de sencilla instalación. Se trata de un protocolo basado en mensajes que conecta en paralelo todas las ECUs a través de un bus CAN, dando lugar a un sistema multimaestro en el que ninguna ECU controla a las demás. Cada ECU interacciona con el resto recibiendo y enviando mensajes, formándose una red interna de comunicación. A pesar de las ventajas con las que cuenta el protocolo CAN, también presenta vulnerabilidades que pueden ser utilizadas por los ciberdelincuentes para introducirse y tomar parte en la red interna del vehículo. En 2012, investigadores de la Universidad de Corea mostraron varios procedimientos para ello, con los que consiguieron manipular el panel de control y el sistema de alarma introduciendo un *malware* en una aplicación de inspección de vehículos, [1]. Ese mismo año se estima que más de 300 BMWs fueron robados en Reino Unido eludiendo el sistema antirrobo mediante *smartphones* conectados al puerto On-Board Diagnostic (OBD), [2]. En 2015, un equipo del Keen Security Lab en China mostró cómo conectar un vehículo a una red Wi-Fi maliciosa, tomando el control del sistema de cierre y navegación, [3].

En los últimos años se han desarrollado muchos estudios con el objetivo de detectar ciberataques en vehículos, basados la mayoría de ellos en técnicas de IA y aprendizaje automático. Sin embargo, su procedimiento a nivel estadístico no suele ser riguroso y sus modelos quedan a merced de los datos, sin profundizar en la distribución de las variables de interés. En este estudio propondremos un modelo de detección y clasificación de anomalías en vehículos que se distinguirá de los de la literatura por su rigurosidad estadística. Además, la mayoría de los modelos de detección de anomalías en vehículos de la literatura están basados en el conteo o cálculo del ratio de aparición de los mensajes, variables discretas. Nosotros consideraremos el tiempo entre mensajes, una variable continua. Aunque están muy relacionadas, el tiempo entre mensajes contiene más información, además de permitirnos hacer uso de las técnicas del Análisis de Supervivencia, [4]. Basaremos nuestro estudio en dos bases de datos. Por un lado, *Survival Analysis Dataset for Automobile IDS* (Survival), [5], utilizada en *2019 Information Security R&D Dataset Challenge* en Corea del Sur, que contiene datos sobre la red de mensajes de tres vehículos sometidos a ataques *flooding*, *fuzzy* y *malfunction*. Por otro lado, *Car Hacking Dataset* (Car-Hacking), [6], utilizada en *Car Hacking: Attack & Defense Challenge 2020* también en Corea del Sur y que recoge datos de un vehículo adicional sometido a ataques *flooding*, *fuzzy*, *malfunction* y *replay*.

El resto del artículo se estructura como sigue: en la Sección II se hablará sobre el *background* del problema, exponiendo las vulnerabilidades del protocolo CAN y literatura relacionada con la detección de ataques, introduciendo el Análisis de Supervivencia y definiendo las métricas para evaluar un modelo de clasificación. La metodología seguida para construir los modelos de detección y clasificación se expondrá en la Sección III. En la Sección IV se mostrarán los resultados para ambas bases de datos y en la Sección V se darán las principales conclusiones del estudio.

II. Background y literatura relacionada

A. Protocolo CAN y sus vulnerabilidades

Como ya hemos aniticipado, la comunicación entre los ECUs de un vehículo se hace en forma de mensajes. Cada mensaje está estructurado como se muestra en la Fig. 1, y sus campos se describen a continuación:

- *Start of frame* (SOF): indica el comienzo del mensaje.
- *Arbitration field* (CAN-ID): identificador del mensaje, indica su prioridad dentro de la red. Cuando dos mensajes intentan transmitirse simultáneamente, tendrá prioridad el mensaje con menor ID. Cada ECU tiene asignados varios IDs, pero cada ID solo puede ser utilizado por una ECU.

ISBN:978-84-09-62140-8

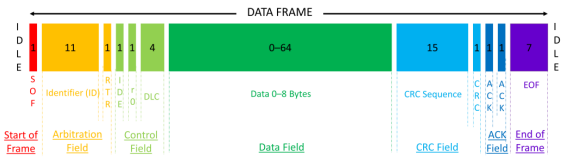

Fig. 1. Estructura de los mensajes del protocolo CAN, [7].

- *Control field* (DLC): indica la longitud del *data field*.
- *Data field*: contiene la información del mensaje, es decir, la información que la ECU emisora quiere transmitir a la ECU receptora.
- *Cyclic redundancy code field* (CRC): también conocido como campo de seguridad, comprueba la validez del mensaje.
- *Acknowledge field* (ACK): comprueba que el mensaje llega a la ECU receptora.
- *End of frame* (EOF): indica el final del mensaje.

Las ventajas del protocolo CAN también dan lugar a vulnerabilidades. Por un lado, el bus CAN es un dominio de difusión, por lo que todas las ECUs tienen acceso a todos los mensajes de la red. Además, la velocidad a la que son enviados y recibidos los mensajes no permite su encriptación, por lo que una ECU maliciosa conectada al sistema puede tener acceso a la red de mensajes. Por otro lado, el bus CAN no cuenta con un sistema de autenticación de mensajes, por lo que nada impide a una ECU transmitir mensajes con IDs que no le pertenezcan. Esto puede ser utilizado por una ECU maliciosa para suplantar la identidad de las ECUs legítimas y enviar mensajes que engañen o colapsen el sistema. Estas vulnerabilidades han dado lugar a la aparición de diferentes ataques sobre la red interna de los vehículos. Todos ellos consisten en la conexión al sistema de una ECU maliciosa que toma ciertas acciones:

Sniffing attack: intercepción de la red de mensajes del vehículo.

Flooding attack (DDoS): inyección de mensajes de máxima prioridad que colapsan la red, impidiendo la correcta transmisión del resto de mensajes.

Fuzzy attack: inyección de mensajes con IDs, DLC y *data fields* aleatorios, pertenecientes y no pertenecientes a la red interna del vehículo, que se transmiten como si de mensajes legítimos se trataran. Para generar estos mensajes los atacantes pueden basarse en la información obtenida a partir de un ataque *sniffing*, o pueden hacerlo sin información previa de la red interna.

Malfunction attack: inyección de mensajes con un ID concreto perteneciente a la red interna, con el *data field* modificado para contaminar el sistema.

Replay attack: inyección de mensajes reales extraídos previamente de la red de mensajes a través de un ataque *sniffing*.

Recientemente se han llevado a cabo diferentes estudios con objetivo la detección de estos ataques. Se han desarrollado modelos basados en la ordenación o frecuencia de los mensajes según su ID: en [8] utilizaron redes neuronales convolutivas, en [9] modelos Long Short-Term Memory (LSTMs), en [10] modelos secuenciales... En ellos se ha observado en mayor o menor medida que la secuencia de IDs en los mensajes es una variable útil para detectar ataques *flooding*, *fuzzy*, *malfunction* y *replay*. También se han desarrollado modelos basados en la comparación del *data field* de los mensajes. Basándose en este enfoque se han aplicado varios modelos de redes neuronales: en [11] modelos LSTM, en [12] autoencoders y en [13] Recurrent Gated Units (GRUs). Algunos de estos modelos han conseguido detectar con precisión varios ataques, aunque en general presentan más desventajas que los modelos de detección basados en los IDs, como por ejemplo un mayor tiempo de respuesta. Además de modelos de aprendizaje automático, también se han desarrollado otros modelos con un enfoque más cercano a la estadística clásica. Por ejemplo, en [14] se presentó un modelo que compara el ratio de aparición de cada ID entre una base libre y en las bases atacadas, detectando ataques cuando estos ratios se salen de ciertos límites. En [7] se hizo un estudio similar, comparando también la desviación típica de los ratios de aparición. En ambos casos se obtuvieron buenas precisiones en la detección de ataques *flooding*, *fuzzy* y *malfunction*. No obstante, estos trabajos profundizan poco en los desarrollos estadísticos formales que son necesarios para una correcta comprensión del problema y estudio de soluciones óptimas; éste es el *gap* que queremos cubrir con el presente trabajo.

B. Análisis de Supervivencia

El Análisis de Supervivencia estudia modelos y métodos en los que la variable respuesta representa el tiempo hasta un evento perfectamente especificado. Tiene aplicaciones en multitud de ramas: en biología y medicina la respuesta suele representar el tiempo desde el nacimiento de un individuo hasta su muerte, el tiempo desde la entrada de un enfermo en un tratamiento hasta su recuperación, o el tiempo desde el diagnóstico de una enfermedad hasta muerte. También es frecuentemente utilizado en ingeniería, donde también es conocido como Análisis de Fiabilidad y suele medir el tiempo de duración de las máquinas desde su instalación, o en economía, donde suele medir el tiempo en paro o la duración de empleo. Es usual en Análisis de Supervivencia encontrarnos con datos censurados por la derecha, para los cuales no se observa el evento final debido a un evento previo que lo impide. Esto puede darse, por ejemplo, si un estudio termina antes del fallecimiento o dada de alta de algún individuo. Estos datos censurados contienen información, no completa pero valiosa, y se han desarrollado técnicas estadísticas para sacar el máximo partido de dicha información.

La función principal del Análisis de Supervivencia que caracteriza la variable temporal es la función de supervivencia, que representa la probabilidad de sobrevivir al tiempo t. Conocer esta función es equivalente a conocer la función de distribución F o la función de densidad f. Si Y denota la variable tiempo de fallo, la función de supervivencia S se define como:

$$S(t) = \mathbb{P}(Y > t), \qquad t \geq 0.$$

El objetivo del Análisis de Supervivencia es estimar, comparar y modelizar curvas de supervivencia. Otra función importante es la función de riesgo λ, que se define como la probabilidad de que un individuo, habiendo sobrevivido a tiempo t, experimente el fallo en el futuro inmediato $[t, t+\Delta t)$. La función de riesgo satisface:

$$\lambda(t) = \frac{f(t)}{S(t)}.$$

Conocer λ es equivalente a conocer f o a conocer S, todas ellas caracterizan la distribución.

A lo largo de este trabajo nos interesará comparar si dos muestras independientes proceden de la misma distribución. Las pruebas más utilizadas son la prueba t de Student bajo la suposición de normalidad e igualdad de varianzas, la prueba de Welch bajo desigualdad de varianzas y el test de Mann-Whitney-Wilcoxon si no se cumple la suposición de normalidad. En Análisis de Supervivencia las pruebas más conocidas son las pertenecientes a la familia log-rank, que aceptan la presencia de datos censurados. Los estadísticos de esta familia de pruebas se construyen en base a la estimación de la función de riesgo λ en cada una de las dos muestras $k = 1, 2$ y en la muestra conjunta, y tienen la forma:

$$U_k = \sum_{i=1}^{D} w(t_i)(o_{ik} - e_{ik}),$$

es decir, miden la diferencia entre los eventos observados y esperados en cada tiempo t_i ponderada por una función peso $w(t_i)$. La función peso determina a qué partes de la distribución se le quiere dar más importancia. Tomando $w(t_i) = 1$ resulta el test log-rank usual, de máxima potencia bajo el supuesto de funciones de riesgo proporcionales $\lambda_2(t) = \theta\lambda_1(t)$. Tomando $w(t_i) = n_i$ (número de individuos a riesgo en t_i) resulta el test de Gehan, una generalización del test de Mann-Whitney-Wilcoxon para muestras censuradas que es el test de máxima potencia bajo el supuesto de diferencias en localización.

C. Evaluación de un método de clasificación

Nuestro modelo será un método de clasificación que agrupará conjuntos de datos en dos clases: **alerta** y **no alerta**. Para evaluar un método de clasificación es necesario conocer las etiquetas reales de los datos. En nuestro caso, cada conjunto de datos sometido a la predicción de su clase es etiquetado previamente como **ataque** o **no ataque**. Estas etiquetas se asignan a ciencia cierta de acuerdo con lo establecido durante la recogida de los datos. Una vez aplicado el método de clasificación, a partir de las observaciones y predicciones se obtiene la matriz de confusión de la Tabla I, donde TN, FP, FN y TP denotan, respectivamente, el número de verdaderos negativos, falsos positivos, falsos negativos y verdaderos positivos. A partir de esta matriz de confusión, existen diferentes métricas que miden la precisión del método. En este artículo utilizaremos la sensibilidad (*true positive rate*, TPR), que mide la proporción de ataques detectados con respecto a los ataques observados, la especificidad (*true negative rate*, TNR), que mide la proporción de no ataques predichos con respecto a los no ataques observados, y el F-score, una medida de la tasa global de aciertos que es más representativa que la precisión usual cuando las clases están desbalanceadas y que da más importancia a la detección correcta de la clase positiva:

$$\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}, \ \text{TNR} = \frac{\text{TN}}{\text{TN+FP}}, \ \text{F} = \frac{2\text{TP}}{2\text{TP+FP+FN}}.$$

Tabla I
MATRIZ DE CONFUSIÓN.

Observado\Predicción	**No alerta**	**Alerta**
No ataque	TN	FP
Ataque	FN	TP

III. METODOLOGÍA: DETECCIÓN Y CLASIFICACIÓN

A. Modelo de detección

Como ya hemos comentado, en este estudio consideraremos dos bases de datos. Por un lado, la base Survival [5] reúne la comunicación entre los dispositivos de tres vehículos: KIA Soul, HYUNDAI Sonata y CHEVROLET Spark, para cada uno de los cuales contamos con cuatro conjuntos de datos: un conjunto de datos libre de ataques y tres conjuntos de datos recogidos bajo la inyección, respectivamente, de ataques *flooding*, *fuzzy* y *malfunction*. Por otro lado la base Car-Hacking [6] cuenta con una base libre y dos bases atacadas, cada una con ataques *flooding*, *fuzzy*, *malfunction* y *replay*, recogidas para un vehículo HYUNDAI Avante CN7. Para más información sobre las bases de datos acudir al Anexo A.

Los cuatro ataques considerados se basan en la inyección de mensajes malignos entre los mensajes libres, por lo que es de esperar que el tiempo entre mensajes cambie entre una conducción libre y una conducción atacada. Esto lo comprobamos en la Fig. 2, donde para cada ataque del KIA Soul se representa la media del tiempo entre mensajes en paquetes de tamaño 4000 a lo largo del tiempo. Ocurriendo lo mismo para el resto de vehículos, esto nos anima a basar nuestros modelos de detección de anomalías en la comparación del tiempo entre mensajes entre las bases libres y las bases atacadas. Así, definimos las variables aleatorias:

$$Y^x = \text{ tiempo entre } x \text{ mensajes en la base libre,}$$
$$Z^x = \text{ tiempo entre } x \text{ mensajes en la base atacada.}$$

Para garantizar que los elementos de las muestras no contienen información repetida, los tiempos se calculan entre posiciones disjuntas. Por ejemplo, si $x = 5$: $Y_1^5 = t_6 - t_1$, $Y_2^5 = t_{11} - t_6, \ldots$ Nuestros modelos se basará en la realización de un contraste de comparación de distribuciones entre las variables Y^x y Z^x, dando señal de alerta si el test rechaza la igualdad de distribuciones.

Para un vehículo y ataque dados, el algoritmo de detección es el que se muestra en la Tabla II. Sus parámetros, mostrados en la Tabla III, merecen algunos comentarios. La longitud de los paquetes L ha de ser suficientemente grande para que las muestras de tiempos entre mensajes tengan un tamaño aceptable para estimar con precisión su función de densidad. El salto entre paquetes j es equivalente al tiempo (ideal) de reacción entre el inicio del ataque y la posible detección del mismo (por ejemplo si $L = 5000$ y $j = 1000$, los paquetes serían $\{1, \ldots, 5000\}, \{1001, \ldots, 6000\}, \ldots$). Lo lógico sería fijar $j = 1$, para lo cual en una situación de máxima precisión la detección sería instantánea. No obstante, esto implicaría realizar tantos contrastes de hipótesis como mensajes en las bases de datos, lo que conlleva un tiempo de ejecución

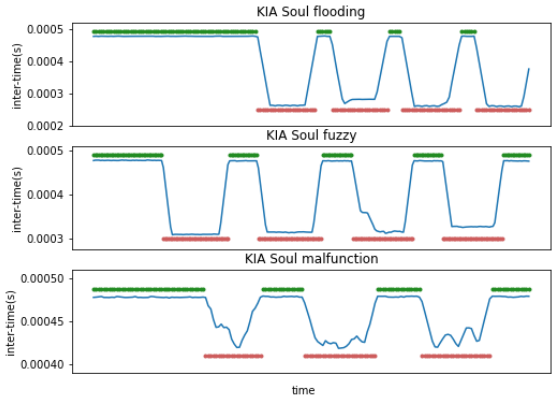

Fig. 2. Tiempo medio entre mensajes en paquetes de 4000 mensajes para el KIA Soul. En verde mensajes libres y en rojo mensajes inyectados.

inaccesible para realizar un estudio exhaustivo del problema (el tiempo de ejecución ronda los 0.003-0.004 segundos por cada contraste de hipótesis). El número de paquetes N viene determinado por L, j y el número de mensajes en la base atacada, M:

$$N = \left\lfloor \frac{M-L}{j} \right\rfloor.$$

Con respecto al parámetro p, en una situación ideal de máxima precisión, un paquete con tan solo un mensaje maligno sería clasificado como alerta. Esto es demasiado ambicioso en la realidad, ya que son necesarios decenas de mensajes malignos para que la distribución de tiempos cambie con respecto a la distribución base. Por ejemplo, fijando $p = 0.06$ estaríamos renunciando a identificar ataques sutiles con una contaminación inferior al 6%. Debemos tener cuidado con no aumentar demasiado este valor, lo que produciría una disminución en la especificidad y no necesariamente un aumento de la sensibilidad. Como test de hipótesis utilizaremos el test log-rank y el test de Gehan. La teoría de la probabilidad nos dice que, bajo H_0, los p-valores se distribuyen como una $\mathcal{U}(0,1)$. Así, fijado α se espera una tasa de error del tipo I del $100\alpha\%$, o lo que es lo mismo, una especificidad de $1-\alpha$. Por otro lado, cuanto mayor sea α más fácil se rechazará H_0 y mayor será la sensibilidad. Así, el parámetro α mide la tensión entre ambas métricas. Finalmente, sería de esperar que la elección de x no influyera significativamente en los resultados. Sin embargo, como veremos en la siguiente sección, este parámetro juega un papel importante en la detección de ataques.

B. Modelo de clasificación

Una vez detectado un ataque resulta altamente interesante estudiar si sus características coinciden con alguno de los ataques conocidos y, en ese caso, clasificarlo. Así, en este apartado propondremos un algoritmo de clasificación en el que entrarán los paquetes alertados del algoritmo de detección. Haremos uso de un atributo de los mensajes que no hemos considerado hasta ahora: su ID. Cada vehículo tiene una red de IDs particular conocida por el distribuidor que denotaremos por I. Para cada $i \in I$ definimos las variables aleatorias:

Y_i^x = tiempo entre x mensajes con ID = i en la base libre,
Z_i^x = tiempo entre x mensajes con ID = i en la base atacada.

Tabla II
ALGORITMO DE DETECCIÓN DE ANOMALÍAS.

Algoritmo de detección

1. Para la muestra de ataque, se crean N paquetes de tamaño L con saltos de tamaño j.
2. Los paquetes que cuenten con al menos un porcentaje p de mensajes malignos son etiquetados como **ataque**. Los que no, son etiquetados como **no ataque**.
3. Se fija el parámetro x.
4. Se calculan los valores que toma la variable Y^x en la base libre.
5. Se calculan los valores que toman las variables Z_n^x para cada uno de los paquetes $n \in \{1, \dots, N\}$.
6. Se realiza un test de comparación a nivel de significación α entre las distribuciones de Y^x y Z_n^x para cada $n \in \{1, \dots, N\}$

$$\begin{cases} H_0: & Y^x =^d Z_n^x \\ H_1: & Y^x \neq^d Z_n^x, \end{cases} \qquad n \in \{1, \dots, N\}.$$

Los paquetes para los que H_0 se rechaza son clasificados como **alerta**. Los que no, son clasificados como **no alerta**.
7. Se obtiene la matriz de confusión entre alertas y ataques y, a partir de ella, se calculan las medidas de precisión de interés: sensibilidad, especificidad y F-score.

Tabla III
PARÁMETROS DEL ALGORITMO DE DETECCIÓN DE ANOMALÍAS.

Parámetro	Función
x	Número de mensajes entre los que se calcula el tiempo.
L	Número de mensajes que componen cada paquete.
j	Salto entre paquetes.
N	Número de paquetes.
p	Porcentaje de mensajes malignos en un paquete a partir del cual es considerado como ataque.
test y α	Contraste de hipótesis utilizado y su nivel de significación.

El algoritmo de clasificación se muestra en la Tabla IV. Este algoritmo clasifica los ataques en función del resultado de los test de comparación de distribuciones que se realizan sobre las variables Y_i^y y Z_i^y, donde y es un parámetro con la misma función que x, pero que ha de tomar valores más pequeños para que las muestras tengan tamaños aceptables. El nivel de significación γ ha de fijarse pequeño, de manera que cuando el test rechace H_0 haya cierta seguridad sobre ello. En el ataque *flooding*, la inyección de mensajes de máxima prioridad impide la comunicación correcta del resto de mensajes y provoca un cambio en la distribución de sus IDs. Es fácilmente identificable estudiando los IDs de máxima prioridad: 1) se detecta algún ID de máxima prioridad no perteneciente a la red vehicular, o 2) se rechaza H_0 para algún ID de máxima prioridad perteneciente a la red vehicular. Durante el ataque *fuzzy* se inyectan mensajes con IDs aleatorios, tanto de la red vehicular como mensajes externos. Aunque el tiempo entre mensajes cambia, el impacto sobre el tiempo entre mensajes de cada ID de la red vehicular es muy pequeño, haciendo que no se rechacen rotundamente los contrastes de hipótesis. Así, el ataque *fuzzy* se identifica si: 1) se detecta algún ID no perteneciente a la red vehicular y que no es de máxima prioridad, o 2) no se rechaza H_0 para ninguno de los IDs de la red vehicular que no son de máxima prioridad. El ataque *malfunction* se centra en un ID en particular de la red vehicular, por lo que la distribución de tiempos entre mensajes

Tabla IV
ALGORITMO DE CLASIFICACIÓN.

Algoritmo de clasificación

1. Se listan los IDs I pertenecientes a la red del vehículo. Se listan los IDs de máxima prioridad J (no necesariamente $J \subset I$). Se listan los IDs hallados en el paquete K.
2. Se fija el parámetro y.
3. Para cada $i \in I$ se calculan los valores que toma la variable Y_i^y en la base libre.
4. Para cada $i \in I$ se calculan los valores que toma la variable Z_i^y en el paquete alertado.
5. Para cada $i \in I$ se realiza un test de comparación a nivel de significación γ entre las distribuciones de Y_i^y y Z_i^y

$$\begin{cases} H_0: & Y_i^y =^d Z_i^y \\ H_1: & Y_i^y \neq^d Z_i^y, \end{cases} \qquad n \in \{1, \ldots, N\}.$$

El ataque es clasificado como ***flooding*** si:
1) Es hallado algún $i \in K$ e $i \in J$ pero $i \notin I$.
2) Se rechaza H_0 para algún $i \in J$ e $i \in I$.

En otro caso, el ataque es clasificado como ***fuzzy*** si:
1) Es hallado algún $i \in (K \backslash J)$ e $i \notin I$,
2) No se rechaza H_0 para ningún $i \in (I \backslash J)$,

como ***malfunction*** si se rechaza H_0 para n o menos $i \in (I \backslash J)$, y ***replay*** si se rechaza H_0 para más de n $i \in (I \backslash J)$. En otro caso, el paquete es clasificado como **anomalía desconocida**.

de dicho ID se ve fuertemente modificada. Sin embargo, no tiene el impacto del ataque *flooding* sobre el resto de IDs, por lo que sus distribuciones rara vez se ven afectadas. Es por eso por lo que clasificamos un ataque como *malfunction* cuando el número de test con rechazo de H_0 es n o menor. Por su parte, el ataque *replay* suele ser más intenso que el ataque *malfunction*, y puede fijarse simultáneamente en más de un ID, rechazando la hipótesis nula en más ocasiones.

IV. RESULTADOS

Comenzando por el modelo de detección, nuestro objetivo final es hallar la combinación de parámetros que maximice las precisiones, es decir, los F-scores. En un estudio preliminar sobre la base Survival hemos observado que las precisiones tienen una fuerte dependencia con el parámetro x. Esto se expone a continuación.

A. Estudio preliminar del modelo de detección: parámetro x

Para hacer este estudio preliminar del efecto del parámetro x fijaremos el resto de parámetros con valores, a nuestro juicio, lógicos:

$$L = 5000, \;\; j = 1000, \;\; p = 0.06, \;\; \alpha = 0.05,$$

(el parámetro $j = 1000$ nos permite un tiempo de reacción de en torno a 0.5 s) y haremos variar $x \in \{1, \ldots, 200\}$. En las Fig. 3 y 4 se muestra la sensibilidad y especificidad en función de x para los tres ataques del Soul. Los resultados se comentan a continuación:

- Para valores pequeños de x se obtienen tanto malas sensibilidades como malas especificidades. Esto nos dice que el tiempo entre mensajes individuales no es una variable que detecte correctamente anomalías debidas a la inyección de mensajes.
- Salvo para el ataque *fuzzy*, el test log-rank presenta inestabilidad en las sensibilidades y, sobre todo, en la especificidades. Se observan patrones para los ataques

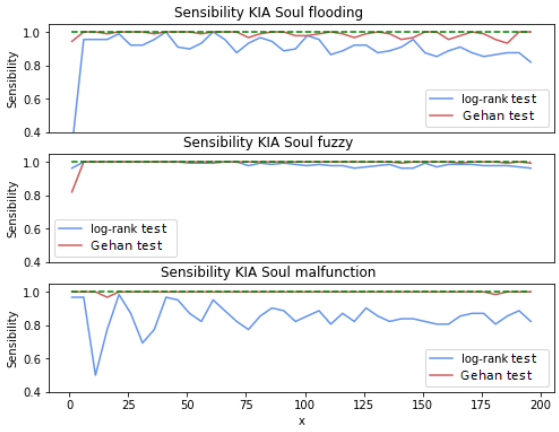

Fig. 3. Sensibilidad en fución de x para los tres ataques del KIA Soul.

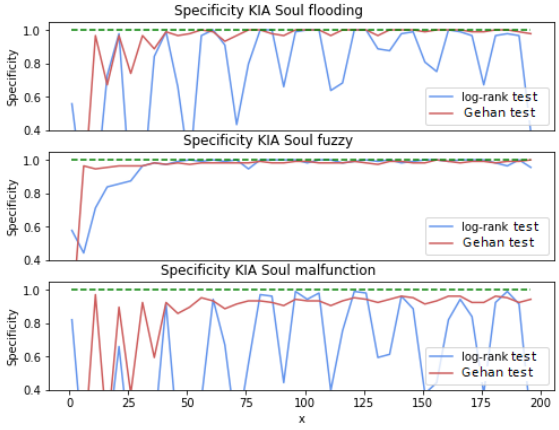

Fig. 4. Especificidad en función de x para los tres ataques del KIA Soul.

flooding y *malfunction*, lo que nos indica que la inestabilidad es originada por la base libre. Por otro lado, el test de Gehan presenta muy buenas sensibilidades, y buenas especificidades a partir de $x = 40$. Existe inestabilidad, aunque mucho menor que para el test log-rank. En cualquier caso, habiendo fijado $\alpha = 0.05$ esperaríamos especificidades de 0.95 (o ligeramente inferiores por haber fijado $p = 0.06$). Sin embargo, esto no ocurre en casi ninguna ocasión. Las posibles razones se comentan más abajo.
- Aunque no se incluyen las gráficas, para el Sonata y el Spark se obtienen resultados muy similares. Para cuantificar estos resultados, en la Tabla V se muestran las medias y desviaciones típicas de los F-scores obtenidos para cada vehículo, cada ataque y cada test, en el rango $x \in \{40, \ldots, 200\}$. Vemos como las desviaciones típicas son considerablemente mayores para el test log-rank. Los resultados para el test de Gehan son muy buenos, tanto en media como en desviación típica. Podría mejorarse el resultado del ataque *malfunction* del Soul. A pesar de ser el ataque con menor impacto sobre los tiempos entre mensajes (Fig. 2), vemos cómo la sensibilidad es muy alta y la disminución del F-score es debida a la especificidad, que nunca llega a ser completamente

Tabla V
MEDIAS Y DESVIACIONES TÍPICAS DE LOS F-SCORES.

		Flooding		*Fuzzy*		*Malfunction*	
		μ	σ	μ	σ	μ	σ
Soul	lr	0.89	0.09	0.99	0.01	0.75	0.15
	Ge	0.99	0.01	0.994	0.003	0.95	0.01
Sonata	lr	0.95	0.03	0.93	0.04	0.94	0.03
	Ge	0.99	0.01	0.99	0.02	0.98	0.01
Spark	lr	0.89	0.08	0.99	0.02	0.93	0.08
	Ge	0.99	0.01	0.99	0.02	0.98	0.01

buena. Es decir, con esta combinación de parámetros, para dicha base de datos se da un número considerable de falsas alertas para todos los x considerados.

¿Por qué el test log-rank presenta tanta inestabilidad en la especificidad con el parámetro x? ¿Por qué las especificidades no se corresponden con el nivel de significación? Hallamos las respuestas en la Fig. 5, donde se muestran las densidades de las variables Y^x y Z^x para $x \in \{10, \ldots, 200\}$ en el Soul. Las variables Z^x son calculadas a partir de los primeros 60 000 mensajes de la base *flooding*, en los que no hay ninguna inyección maligna. Dado un x, cuando más alta sea la densidad de sus tiempos entre mensajes menor será su varianza. Vemos como dependiendo del x la densidad alcanza mayor o menor altura, habiendo picos y valles que se repiten periódicamente. De esto deducimos que los mensajes no son generados de forma individual e independiente, sino que forman parte de lotes de cierto número de mensajes que se generan de forma periódica. Dado que las Z^x son calculadas sobre una conducción libre, esperaríamos que se distribuyeran de igual forma a las Y^x. Sin embargo, vemos cómo existen x para los cuales las densidades de Y^x presentan picos y las de Z^x presentan valles. Para dichos valores de x el test log-rank halla diferencias entre ambas distribuciones y H_0 es frecuentemente rechazada. Esto ocurre por ejemplo para $x = 50$, para el cual en la Fig. 4 vemos como se obtiene una mala especificidad. Para otros x, como $x = 105$, ambas densidades presentan picos, no rechazándose generalmente H_0 y obteniendo muy buena especificidad. Sin embargo, aunque la forma de la distribución cambie, la media permanece invariante para todo el rango de x, por lo que el test de Gehan no halla diferencias significativas. Es por esto que la especificidad presenta una fuerte dependencia con x para el test log-rank y no para el test de Gehan.

De todo esto concluimos tres cosas. Por un lado, el proceso de generación de mensajes legítimos por lotes hace que los tiempos entre mensajes presenten una fuerte correlación y no se cumpla la hipótesis de independencia requerida en los contrastes log-rank y Gehan. Esto podría explicar que las especificidades obtenidas no se correspondan con el nivel de significación fijado. Por otro lado, dicho proceso de generación de mensajes evoluciona en el tiempo, dando lugar a una alta variabilidad de los resultados para el test log-rank. Sin embargo, la media de los tiempos entre mensajes es una variable que caracteriza correctamente una conducción libre de ataques. De hecho, bajo H_0 los tiempos medios son prácticamente idénticos, otra posible razón por la que las especificidades del test de Gehan sean tan altas (por encima de $1-\alpha = 0.95$). Este diferente comportamiento de los tests log-rank y Gehan merece, en cualquier caso, un mayor estudio en el futuro, teniendo también presente que, en otras situaciones prácticas, el patrón de ataques podría diferir del existente en los datos analizados. Además, en situaciones en las que las funciones de riesgo de los tiempos entre mensajes libres de ataque y contaminados se crucen, los citados tests podrían tener baja potencia estadística, requiriéndose tests alternativos.

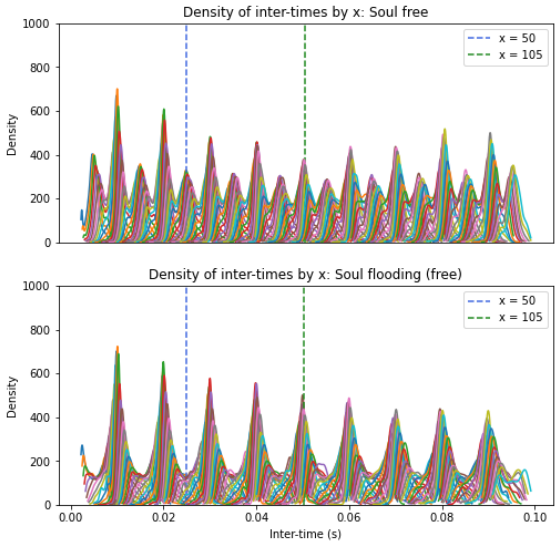

Fig. 5. Para el KIA Soul y para $x \in \{10, \ldots, 200\}$, arriba: densidades de las variables Y^x calculadas a partir de la base libre y, abajo: densidades de las variables Z^x calculadas a partir de los primeros 60 000 mensajes de la base *flooding*.

B. Optimización de parámetros

Habiendo visto y comprendido los resultados, resulta lógico elegir el test de Gehan para la optimización de los parámetros del modelo. Para la base Survival optimizaremos con respecto al F-score tanto para cada vehículo por separado como para los tres vehículos en conjunto, obteniendo una combinación de parámetros que, idealmente, aportará buenas precisiones para los tres ataques de los tres vehículos. Finalmente, aplicaremos esta combinación de parámetros sobre la base Car-Hacking, comprobando así la capacidad predictiva del algoritmo sobre un conjunto de datos diferente al del ajuste. Nuestro modelo tiene demasiados parámetros, por lo que tomaremos ciertas consideraciones sobre ellos y únicamente optimizaremos con respecto a x, el parámetro de mayor influencia en los resultados, y α, el parámetro que mide la tensión entre la sensibilidad y la especificidad.

Los parámetros L y x determinan el tamaño de las muestras de las variables Z_n^x de la forma: $m = \lfloor L/x \rfloor$. Para realizar la optimización fijaremos $m = 50$ y L quedará determinado por x de la forma: $\boldsymbol{L = 50 \cdot x}$. El parámetro j determina el número de contrastes de hipótesis en la ejecución del programa, por lo que influye directamente en el tiempo de ejecución. El test de Gehan nos permite fijar $\boldsymbol{j = 50}$ con tiempos de ejecución razonables. Para $j = 50$ el tiempo de reacción es de aproximadamente 0.025 s. Con respecto a p, hasta ahora hemos fijado $p = 0.06$, con lo cual los paquetes con un porcentaje de mensajes malignos inferior al 6% no eran considerados como atacados. Haciendo esto no estamos

Tabla VI
OPTIMIZACIÓN INDIVIDUAL DE PARÁMETROS.

(x_{opt}, α_{opt})		Sens.	Esp.	F-score	Train F
Soul (51, 0.33)	*Flooding*	0.989	0.992	0.993	0.996
	Fuzzy	0.992	0.987	0.992	0.993
	Malfunction	0.992	0.999	0.992	0.992
	Media	**0.991**	**0.993**	**0.992**	**0.994**
Sonata (49, 0.49)	*Flooding*	0.990	1	0.995	0.992
	Fuzzy	0.988	1	0.994	0.987
	Malfunction	0.985	1	0.992	0.991
	Media	**0.988**	**1**	**0.994**	**0.990**
Spark (80, 0.44)	*Flooding*	0.993	0.724	0.941	0.993
	Fuzzy	1	0.992	0.993	0.991
	Malfunction	1	0.917	0.981	0.988
	Media	**0.998**	**0.878**	**0.972**	**0.991**

Tabla VII
OPTIMIZACIÓN GLOBAL DE PARÁMETROS.

(50, 0.27)		Sens.	Esp.	F-score	Train F
Soul	*Flooding*	0.994	0.841	0.966	0.994
	Fuzzy	0.991	1	0.995	0.993
	Malfunction	0.994	0.961	0.970	0.988
	Media	**0.993**	**0.934**	**0.977**	**0.992**
Sonata	*Flooding*	0.980	1	0.990	0.987
	Fuzzy	0.978	1	0.989	0.979
	Malfunction	0.973	1	0.986	0.983
	Media	**0.977**	**1**	**0.988**	**0.983**
Spark	*Flooding*	0.961	0.801	0.928	0.987
	Fuzzy	1	0.996	0.997	0.991
	Malfunction	0.985	1	0.992	0.980
	Media	**0.982**	**0.932**	**0.972**	**0.986**
	Media total	**0.955**	**0.979**	**0.979**	**0.987**

siendo fieles a la realidad, ya que puede haber ataques que causen problemas con un porcentaje menor de intrusiones. Así, fijaremos $\boldsymbol{p = 0}$ de manera que un paquete con tan solo un mensaje maligno sea considerado como ataque. Esto, puede dar lugar a un incremento en el número de falsos negativos, ya que paquetes con pocos mensajes malignos son difícilmente detectables. Habiendo fijado L, j y p, haremos variar $x \in \{40, \ldots, 200\}$ y $\alpha \in [0, 1]$. Para cada vehículo y cada ataque tomamos como muestra de entrenamiento el 65% inicial de los mensajes y como muestra de test el 35% restante (nos hemos cerciorado de que en todos los casos ambas muestras contienen tanto partes libres como partes atacadas).

Comenzamos optimizando para cada vehículo por separado: para cada vehículo hallamos la combinación (x, α) que maximiza la media de los F-scores de los tres ataques. En la Tabla VI se muestran los valores óptimos de x y α y las sensibilidades, especificidades y F-scores obtenidos en cada caso para la muestra de test, además de los F-scores de las muestras de entrenamiento. A pesar de haber fijado $p = 0$ vemos cómo las sensibilidades obtenidas son muy buenas. Esto se debe a los altos niveles de significación seleccionados, que detectan diferencias muy fácilmente. Esto es un arma de doble filo, ya que puede dar lugar a un aumento en el número de falsos positivos. Por ejemplo, para el ataque *flooding* del Spark, a pesar de obtener muy buena precisión en la muestra de entrenamiento, la especificidad es realmente mala en la muestra de test. Por otro lado, destacan los resultados del Sonata, para el cual no se dan falsas alarmas, y los ataques *fuzzy* y *malfunction* del Spark, para el cual todos los paquetes atacados son alertados. Las especificidades obtenidas de nuevo no se corresponden con los niveles de significación fijados, siendo mucho mejores de lo esperado. Como ya hemos comentado, esto puede tener dos razones. Por un lado, que los tiempos medios comparados sean casi idénticos y el test de Gehan devuelva p-valores muy altos. Por otro lado, que no se cumpla la hipótesis de independencia entre los tiempos entre mensajes y el test de Gehan no sea totalmente válido.

Optimizando para los tres vehículos en conjunto obtenemos $\boldsymbol{(x_{opt}, \alpha_{opt}) = (50, 0.27)}$ y las sensibilidades, especificidades y F-scores para la muestra de test que se recogen en la Tabla VII. Como es lógico, los resultados del F-score empeoran con respecto a las optimizaciones individuales. Para los ataques *flooding* del Soul y del Spark se dan un número considerable de falsas alertas. Las sensibilidades del Soul son

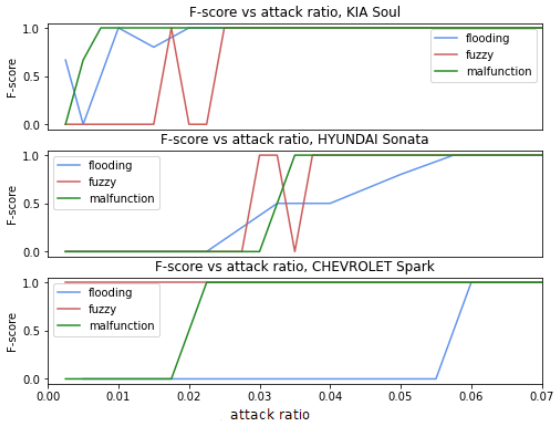

Fig. 6. F-score en función del ratio de ataque para las muestras de test de los tres ataques de los tres vehículos.

muy buenas. Se obtiene un F-score promedio de 0.978, buen resultado teniendo en cuenta la generalidad del modelo. Para estudiar la capacidad de detección, en la Fig. 6 se muestran, para esta combinación de parámetros, los F-scores en función del ratio de ataque de los paquetes, es decir, de la proporción de mensajes malignos. Vemos como basta un ratio de ataque del 6% o incluso inferior para que todos los ataques sean detectados. Dependiendo del efecto que puedan producir los ataques sobre el vehículo puede interesarnos disminuir α para reducir las falsas alertas a cambio de no ser capaces de detectar los ataques con un ratio de ataque pequeño, o aumentar α para detectar cualquier intrusión por pequeña que sea a cambio de dar falsas alertas. Un equilibrio entre estos dos aspectos lo da el F-score, que es lo que nosotros hemos maximizado.

Aplicamos ahora la combinación de parámetros óptima global obtenida ($x = 50$, $\alpha = 0.27$, $L = 2500$, $j = 50$ y $p = 0$) sobre la primera de las bases atacadas de la base Car-Hacking, obteniendo:

$$\textbf{Sens.} = \mathbf{0.845}, \qquad \textbf{Esp.} = \mathbf{0.988}, \qquad \textbf{F-score} = \mathbf{0.906}\ .$$

La especificidad es alta, pero la detección de los ataques no es buena, dando lugar a una baja sensibilidad y por ende a un bajo F-score. Visto que $(x, \alpha) = (50, 0.27)$ no aporta buenas precisiones, optimizamos con respecto a estos parámetros. Para ello utilizamos la segunda base atacada como muestra de

entrenamiento, y la primera como muestra de test, obteniendo la combinación $(\boldsymbol{x_{opt}}, \boldsymbol{\alpha_{opt}}) = (\mathbf{70}, \mathbf{0.45})$ y las métricas:

$$\textbf{Sens.} = \mathbf{0.981}, \qquad \textbf{Esp.} = \mathbf{0.993}, \qquad \textbf{F-score} = \mathbf{0.987}\ .$$

Vemos cómo las precisiones están a la altura de las obtenidas para los tres vehículos anteriores.

De los resultados para las bases Survival y Car-Hacking sacamos dos conclusiones principales. Nuestro modelo es capaz de detectar con alta precisión intrusiones en las redes vehiculares, quedando probado para cuatro vehículos distintos. Sin embargo, no es viable disponer de una combinación de parámetros global que se aplique con precisión sobre todos los vehículos, siendo necesaria una combinación de parámetros propia para cada uno de ellos.

C. Resultados de clasificación

Recordemos que en el algoritmo de clasificación contamos con tres parámetros adicionales: y, que marca el número de mensajes con cada ID entre los que se calcula el tiempo, γ, el nivel de significación de los test de hipótesis para cada ID y n, que indica el número de test rechazados que diferencian a los ataques *malfunction* y *replay*. Tras un estudio preliminar del comportamiento de los IDs en cada ataque, nos parece razonable fijar $\boldsymbol{y} = \mathbf{3}$, $\boldsymbol{\gamma} = \mathbf{0.01}$ y $\boldsymbol{n} = \mathbf{6}$. Con respecto al resto de parámetros (L, j, x, α y p), para cada vehículo utilizamos la combinación óptima obtenida anteriormente. En la Tabla VIII se muestran los F-scores para cada muestra de test. El modelo es capaz de distinguir perfectamente entre los ataques *flooding*, *fuzzy* y *malfunction*, y las falsas alarmas son asignadas en su mayoría al ataque *fuzzy*. Es por esto por lo que para el Soul y el Spark los F-scores aumentan con respecto a los de la Tabla VI para los ataques *flooding* y *malfunction* y dismininuyen para el ataque *fuzzy*, y para el Sonata los F-scores no cambian ya que no se dan falsas alarmas. Con respecto a la base Car-Hacking, de nuevo las falsas alarmas son clasificadas como *fuzzy* y en algún caso se confunden los ataques *malfunction* con ataques *replay*. Al igual que para el algoritmo de detección, estos resultados podrían mejorarse optimizando con respecto a los parámetros y, γ y n.

Tabla VIII
F-SCORES MODELO DE CLASIFICACIÓN.

	Flooding	***Fuzzy***	***Malfunction***	***Replay***	**Media**
Soul	0.995	0.984	0.994		**0.991**
Sonata	0.995	0.994	0.992		**0.994**
Spark	0.996	0.931	1		**0.976**
Avante	0.997	0.971	0.962	0.972	**0.975**

V. Conclusiones

En este artículo hemos propuesto un novedoso modelo de detección de anomalías en la red de comunicación interna de los dispositivos en vehículos. Este modelo se diferencia de los de la literatura existente en dos aspectos. Por un lado, utiliza como variable de interés el tiempo entre mensajes, una variable continua que contiene más información que simples conteos y que permite aplicar técnicas de Análisis de Supervivencia. Por otro lado, el tiempo entre mensajes es calculado sin distinguir su ID, lo que aporta generalidad, simplifica el problema y disminuye los tiempos de ejecución. Los resultados obtenidos nos aportan varias conclusiones. Por un lado, para cada vehículo es posible adaptar el algoritmo propuesto (modificando sus parámetros) para detectar ataques con alta sensibilidad, incluso cuando el ratio de ataque es muy bajo. Sin embargo, esto puede provocar la aparición de falsas alarmas debidas a los diferentes modos de generación de mensajes libres, por lo que es necesario valorar qué es más importante y tomar decisiones a raíz de ello. Estos diferentes modos también impiden la existencia de una combinación de parámetros global que aporte buenas precisiones para cualquier vehículo, por lo que ha de tratarse cada vehículo por separado. El modelo propuesto es un modelo general capaz de detectar anomalías en la red vehicular relacionadas con la inyección de mensajes. En cuanto a los ataques conocidos, hemos propuesto un algoritmo de clasificación que agrupa correctamente los ataques *flooding*, *fuzzy*, *malfunction* y *replay* y que asigna mayoritariamente las falsas alarmas a ataques *fuzzy*. Como estudio futuro, queda pendiente profundizar en cómo la evolución temporal de la generación de mensajes y la correlación entre los elementos de las muestras puede influir en los resultados, principalmente en la detección de falsas alarmas.

References

[1] Crypto World. Smart Car Hacking. (22 de enero de 2013). Accedido el 18 de marzo de 2024. [Video en línea]. Disponible: https://www.youtube.com/watch?v=katVec_SwA4.

[2] J. Torchinsky. “Hackers steal BMWs in 3 minutes using security loophole”. NBC News. Accedido el 18 de marzo de 2024. [En línea]. Disponible: https://www.nbcnews.com/tech/tech-news/hackers-steal-bmws-3-minutes-using-security-loophole-flna868400.

[3] Black Hat — Home. Accedido el 18 de marzo de 2024. [En línea]. Disponible: https://www.blackhat.com/docs/us-17/thursday/us-17-Nie-Free-Fall-Hacking-Tesla-From-Wireless-To-CAN-Bus-wp.pdf.

[4] J. P. Klein y y M.L. Moeschberger, ”Survival analysis: Techniques for censored and truncated data”, 2a ed. *New York: Springer*, 2003.

[5] Hacking and Countermeasure Research Lab. 2020. Survival Analysis Dataset for Automobile IDS. Retrieved August 1, 2021 from https://ocslab.hksecurity.net/Datasets/survival-ids.

[6] Hacking and Countermeasure Research Lab. 2020. Car-Hacking: Attack & Defense Challenge 2020. Retrieved August 1, 2021 from https://ocslab.hksecurity.net/Datasets/carchallenge2020.

[7] J. Khan, D.-W. Lim y Y.-S. Kim, “Intrusion Detection System CAN-Bus In-Vehicle Networks Based on the Statistical Characteristics of Attacks”, *Sensors*, vol. 23, n.º 7, p. 3554, marzo de 2023..

[8] H. M. Song, J. Woo y H. K. Kim, “In-vehicle network intrusion detection using deep convolutional neural network”, *Veh. Commun*., vol. 21, p. 100198, enero de 2020.

[9] M. Jedh, L. Ben Othmane, N. Ahmed y B. Bhargava, “Detection of Message Injection Attacks Onto the CAN Bus Using Similarities of Successive Messages-Sequence Graphs”, *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 4133–4146, 2021.

[10] Kalutarage, H. K., Al-Kadri, M. O., Cheah, M., & Madzudzo, G. ”Context-aware anomaly detector for monitoring cyber attacks on automotive CAN bus”, *Proceedings of the 3rd ACM Computer Science in Cars Symposium* pp. 1-8, octubre de 2019.

[11] Wang, Y., Chia, D. W. M., & Ha, Y., ”Vulnerability of deep learning model based anomaly detection in vehicle network”, 2020 *IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 293-296, agosto de 2020.

[12] H. Narasimhan, V. R y N. Mohammad, “Unsupervised Deep Learning Approach for In-Vehicle Intrusion Detection System”, *IEEE Consum. Electron. Mag.*, p. 103-108, 2021.

[13] V. K. Kukkala, S. V. Thiruloga y S. Pasricha, “LATTE: L STM Self- Att ention based Anomaly Detection in E mbedded Automotive Platforms”, *ACM Trans. Embedded Comput. Syst.*, vol. 20, n.º 5s, pp. 1–23, octubre de 2021.

[14] M. L. Han, B. I. Kwak y H. K. Kim, “Anomaly intrusion detection method for vehicular networks based on survival analysis”, *Veh. Commun.*, vol. 14, pp. 52–63, octubre de 2018.

ANEXO A: BASES DE DATOS

El desarrollo de este estudio se ha apoyado en dos bases de datos públicas que recogen la comunicación entre dispositivos de diferentes vehículos: *Survival Analysis Dataset for Automobile IDS* (Survival), [5], y *Car Hacking Dataset* (Car-Hacking), [6]. Han sido utilizadas en retos de ciberseguridad en Corea del Sur, concretamente en *2019 Information Security R&D Dataset Challenge* y en *Car Hacking: Attack & Defense Challenge 2020* respectivamente. En ambos casos los mensajes entre dispositivos han sido recogidos a través de un puerto intravehicular OBD-II. Por un lado, las bases de datos libres han sido recogidas bajo una conducción 'normal' real de cada vehículo. Por otro lado, los ataques han sido simulados previamente e inyectados periódicamente durante la conducción.

La base Survival recoge datos de tres vehículos: KIA Soul, HYUNDAI Sonata y CHEVROLET Spark. Para cada uno de ellos se incluyen cuatro conjuntos de datos: una base libre y tres bases con ataques *flooding*, *fuzzy* y *malfunction* respectivamente. La base Car-Hacking recoge datos de un vehículo HYUNDAI Avante CN7 y cuenta con una base libre y dos bases atacadas, cada una de ellas con ataques *flooding*, *fuzzy*, *malfunction* y *replay*. En las Tablas IX y X se muestra el número de mensajes libres y mensajes inyectados para cada base de datos, y en la Fig. 7 se muestra, para cada base atacada del Avante, la distribución de los mensajes a lo largo del tiempo. Vemos como los mensajes libres permanecen durante todo el rango temporal, ya que los ataques consisten en la inyección de mensajes malignos entre mensajes libres.

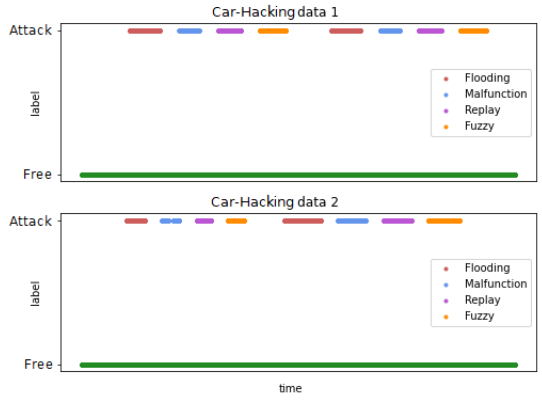

Fig. 7. Distribución de mensajes a lo largo del tiempo para las dos bases atacadas del HYUNDAI Avante CN7.

Tabla IX
NÚMERO DE MENSAJES LIBRES Y MALIGNOS PARA CADA CONJUNTO DE DATOS DE LA BASE SURVIVAL.

Base	**Soul**		**Sonata**		**Spark**	
	Free	*Attack*	*Free*	*Attack*	*Free*	*Attack*
Libre	192 516	0	117 173	0	136 934	0
Flood.	139 788	33 141	109 931	32 422	52 632	22 587
Fuzzy	197 539	39 812	110 332	18 103	32 092	5812
Malf.	155 974	7401	109 511	15 974	38 402	8047

Tabla X
NÚMERO DE MENSAJES LIBRES Y MALIGNOS PARA CADA CONJUNTO DE DATOS DE LA BASE CAR-HACKING.

Avante	*Free*	*Flood.*	*Fuzzy*	*Malf.*	*Replay*
Base libre	179 346	0	0	0	0
Base atacada 1	733 752	38 521	22 620	988	10 509
Base atacada 2	811 532	38 852	22 854	2891	13 266