Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.34, NO.4, Aug. 2024 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2024.34.4.747

CAN 통신 메시지 내의 Checksum Signal 식별 방법 연구*

이 경 연,[1†] 김 형 훈,[2] 이 동 훈,[3] 최 원 석[3‡]
[1,3]고려대학교 (대학원생, 교수), [2]연세대학교 (대학원생)

Checksum Signals Identification in CAN Messages*

Gyeongyeon Lee,[1†] Hyunghoon Kim,[2] Dong Hoon Lee,[3] Wonsuk Choi[3‡]
[1,3]Korea University (Graduate student, Professor),
[2]Yonsei University (Graduate student)

요 약

현대의 자동차는 ECU(Electronic Control Unit)를 통해 전자적으로 제어되면서 운전자에게 안전성과 편의성을 제공해준다. 고급 자동차의 경우 100개 이상의 ECU가 탑재되어 있는 것으로 알려져 있다. 그러나 이러한 컴퓨터 기반의 차량 시스템은 자동차를 사이버 공격에 취약하게 만든다. 이에 대응하여 자동차 제조회사들은 차량용 침입 탐지 시스템(Intrusion Detection System, IDS)과 같은 보안 기술을 개발하고 있다. 자동차 제조회사들만이 자동차 보안 기술을 개발할 수 있는데, 그 이유는 이들만이 차량 내부 네트워크에 대한 데이터베이스(DBC Format File)를 가지고 있으며 이를 기밀로 유지하고 있기 때문이다. 그러나 외부 연구기관들은 DBC Format File과 같은 데이터베이스에 접근할 수 없어 자동차 보안 연구에 어려움을 겪고 있다. 따라서 본 논문에서는 제한된 정보에 따른 한계를 극복하기 위해 차량 내부 네트워크를 분석하여 DBC Format File을 부분적으로 식별할 수 있는 방법을 제안한다. 제안하는 기법은 CAN(Controller Area Network) 트래픽을 분석하여 CAN Frame Data Field의 Signal 중 Checksum Signal의 비트 위치를 식별하고, Lookup Set을 생성해 Checksum값을 계산한다. 더불어 실제 차량에서 수집한 데이터와 공개된 데이터셋으로 해당 방법을 검증하였다.

ABSTRACT

Recently, modern vehicles have been controlled by Electronic Control Units (ECUs), by which the safety and convenience of drivers are highly improved. It is known that a luxury vehicle has more than 100 ECUs to electronically control its function. However, the modern vehicles are getting targeted by cyber attacks because of this computer-based automotive system. To address the cyber attacks, automotive manufacturers have been developing some methods for securing their vehicles, such as automotive Intrusion Detection System (IDS). This development is only allowed to the automotive manufacturers because they have databases for their in-vehicle network (i.e., DBC Format File) which are highly confidential. This confidentiality poses a significant challenge to external researchers who attempt to conduct automotive security researches. To handle this restricted information, in this paper, we propose a method to partially understand the DBC Format File by analyzing in-vehicle network traffics. Our method is designed to analyze Controller Area Network (CAN) traffics so that checksum signals are identified in CAN Frame Data Field. Also, our method creates a Lookup Set by which a checksum signal is correctly estimated for a given message. We validate our method with the publicly accessible dataset as well as one from a real vehicle.

Keywords: Vehicle Security, CAN, CAN reverse engineering, Checksum, Checksum Calculation

Received(06. 24. 2024), Modified(07. 10. 2024), Accepted(07. 18. 2024)
* 이 논문은 2024년 정부(방위사업청)의 재원으로 국방과학연구소의 지원을 받아 수행된 연구임(UI2200575D*)
† 주저자, gyeongyeon@korea.ac.kr
‡ 교신저자, beb0396@korea.ac.kr(Corresponding author)

I. 서 론

Information and Communications Technology (ICT) 기술의 발전은 자동차 산업에도 많은 영향을 미치고 있다. 과거에는 차량의 기능들이 기계적인 방법에 의하여 제어되는 방식이었지만, 현재의 자동차들은 전자제어장치 (Electronic Control Unit, ECU)라 불리는 임베디드 컴퓨터를 통하여 차량의 기능들이 전자적으로 제어되고 있다. ECU들은 자동차 내부에서 네트워크를 형성하여 서로 통신하고 있으며, 최신 차량의 경우 70~100개의 ECU들이 탑재되어 차량의 기능들을 제어하고 있는 것으로 알려져 있다 [1]. 자동차 내부 네트워크에서 ECU들은 일반적으로 Controller Area Network (CAN) 통신 프로토콜을 이용해 통신하고 있다. CAN 통신 프로토콜의 네트워크 Topology는 Bus 형태로 구성되어 있어, 각각의 ECU들은 CAN Bus 상에서 브로드캐스팅 방식으로 CAN 메시지를 송·수신한다.

차량의 많은 기능들을 전자적으로 제어함에 따라 운전 편의성 (Convenience)과 안정성 (Safety)이 많이 향상되었지만, 사이버 공격으로 차량을 악의적으로 제어하는 자동차 해킹에 대한 문제점이 발생하고 있다. 실제로 CAN 통신 프로토콜의 취약점을 이용한 자동차 해킹 위협이 지속해서 등장하고 있는 모습을 볼 수 있는데, 대표적으로 2015년 Charile Miller와 Chris Valasek은 지프 체로키(Jeep Cherokee) 차량이 원격에서 해킹이 가능함을 시연하였다 [2]. 이러한 자동차 사이버 공격에 대응하기 위해 최근 유럽경제위원회 세계 포럼 (Economic Commision for Europe, UNECE)의 WP29에서 R155 사이버보안 규제가 발표되었다 [3]. 이 규제에 따르면, 2022년 7월부터 유럽에서 판매되는 모든 신차는 자동차 사이버보안 위협에 대하여 적절한 보안 기술이 탑재되어야 하고, 2024년 7월부터는 모든 차량에 적용될 예정이다. 해당 규제를 만족하기 위해 자동차 제조회사 및 관련 연구기관에서 차량용 침입 탐지 시스템 (Intrusion Detection System, IDS)과 같은 보안 기술을 개발하고 있다. 침입 탐지 시스템이란, 자동차 내부 네트워크의 트래픽을 분석해 사이버 공격이 발생한 상황을 탐지하는 시스템을 말한다. 또한 자동차 제조회사들은 차량의 상태 정보와 차량을 제어하기 위한 제어 정보 등이 포함된 DBC Format File을 만들어 관리하고 있다. DBC File Format에는 차량을 효율적으로 제어하기 위한 방법이 포함되어 있기 때문에, 자동차 제조회사들은 해당 파일의 비밀 등급을 높게 하여 외부에 공개되지 않도록 관리한다. 이로 인해 DBC Format File에 접근할 수 없는 연구기관에서는 자동차 보안 연구에 많은 어려움을 겪고 있다. 이에 대한 대응책으로 다양한 차종의 DBC Format File이 복원되어 있는 opendbc를 활용할 수 있다 [4]. opendbc는 반자율주행 소프트웨어인 openpilot을 개발하는 comma라는 회사에서 오픈소스로 Github에 공개한 DBC Foramt File이다. opendbc를 IDS 등 다양한 연구에 활용할 수 있지만, 모든 차종이 지원되는 것은 아니며 opendbc만으로는 CAN Data Field의 Signal을 정확하게 식별할 수 없다.

본 논문에서 우리는 DBC Format File에 대한 접근이 자유롭지 않다는 한계를 극복하기 위하여 DBC Format File의 일부 정보를 리버싱하는 방법을 제안한다. 특히, CAN Data Field에 정의되어 있는 여러 유형의 Signal들 중 Checksum Signal로 CAN 메시지 간 관계성을 분석하고 그 결과를 통해 Checksum 값을 계산하는 방법을 제시한다. Checksum Signal 값을 알 수 있다면 IDS 및 Fuzzing 연구를 위해 차량에 메시지를 주입할 때 검증에 통과하는 올바른 Checksum 값으로 주입할 수 있고, 이를 통해 CAN 메시지의 용도를 파악할 수 있다. 현재 악의적인 메시지를 주입해 자동차를 제어하는 자동차 해킹 연구와 이러한 공격을 탐지하는 차량용 침입탐지 시스템에 관한 연구, 그리고 CAN 메시지를 분석하는 CAN 메시지 리버싱 연구가 많이 진행되고 있다. 그러나 Checksum Signal을 계산하는 방법에 대한 연구는 존재하지 않는다. 따라서 본 논문은 최초로 Checksum Signal에 중점을 두며, 본 논문에서 제안하는 기법을 통해 DBC Format File이 없는 상황에서도 차량 사이버공격에 대한 대응 기술 연구를 수행할 수 있을 것이라 생각한다.

II. 관련연구

이번 장에서는 기존의 다양한 자동차 해킹 기법과 이에 대응하기 위해 설계된 차량용 IDS에 대한 관련 연구를 소개하고, CAN Bus를 리버싱하는 기존 연구에 대하여 설명한다.

2.1 자동차 해킹

일반적으로 자동차 해킹은 자동차 내부 네트워크에 제어 메시지를 전송함으로써 차량을 악의적으로 제어한다. 자동차 내부 네트워크에 적용된 통신 프로토콜인 CAN 통신 프로토콜은 메시지 인증에 대한 기능을 제공하지 않는다. 다시 말해, 공격자가 제어 메시지를 자동차 내부 네트워크에 전송한다면, ECU들은 메시지 인증 절차 없이 제어 메시지를 무조건적으로 처리한다. 차량을 제어하는 CAN 메시지를 자동차 내부 네트워크에 전송하여 차량을 악의적으로 제어하는 연구가 2010년 K. Koscher 등에 의하여 처음 발표되었다 [1]. 차량이 정상적으로 작동할 때의 CAN 트래픽을 수집 및 분석하였고, 차량을 제어하는 CAN 메시지를 발견하였다. 이러한 CAN 메시지를 자동차 내부 네트워크에 다시 주입함으로써 차량의 엔진, 계기판, 라디오, 냉난방, 브레이크 등의 기능들을 제어할 수 있었다.

차량에 대한 물리적인 접근이 필요한 위의 방법과 달리, 2011년 S. Checkoway 등은 CAN 네트워크에 간접적으로 접근하여 제어 메시지를 주입할 수 있는 다양한 공격 표면 (Attack Surface) 등을 소개하였다 [5]. 예를 들어, CD 플레이어나 USB 포트와 같은 외부 멀티미디어 인터페이스를 이용해 공격하는 방법이 소개되었다. CAN 네트워크는 멀티미디어 시스템과 상호 연결되어 있으므로 해당 멀티미디어 시스템을 감염시킴으로써 악의적인 제어 메시지를 CAN 네트워크에 주입할 수 있게 된다.

앞서 설명한 자동차 해킹 방법들은 공격대상이 되는 자동차에 직·간접적으로 접근하여 제어 메시지를 주입하기 위한 채널 형성에 대한 사전 작업이 필요하였다. 그러나 2015년 Charile Miller와 Chris Valasek은 자동차 해킹 대상이 되는 차량에 사전 작업 없이 CAN 네트워크에 접근하여 공격에 성공하였다 [2]. 이들은 자동차 제조회사가 운전 편의성을 제공하기 위하여 설치하는 텔레매틱스 기기를 타겟으로 하여, 외부에서 텔레매틱스 기기에 접근할 수 있는 채널을 발견하였다. 또한, 해당 텔레매틱스 기기는 CAN 네트워크와 연결이 되어 있었기 때문에 외부 공격자가 제어 메시지를 CAN 네트워크에 주입할 수 있었고 이를 통해 차량을 악의적으로 제어하는데 성공하였다.

2.2 차량용 침입 탐지 시스템

앞서 설명한 자동차 해킹에 대응하기 위해, CAN 네트워크의 트래픽을 분석하여 공격 여부를 탐지하는 차량용 IDS에 대한 많은 연구가 진행되고 있다. 차량용 IDS는 CAN 트래픽의 다양한 패턴을 분석할 수 있으며 이러한 패턴 특징은 T. Hoppe 등에 의해 처음 소개되었다 [7]. 대부분의 CAN 메시지는 ECU로부터 주기적으로 전송되기 때문에, 차량용 IDS는 CAN 메시지의 주기성을 분석하여 정상 및 비정상 상황을 탐지한다 [8],[9]. 정상 ECU들은 CAN 메시지를 주기적으로 전송하고 있어 공격자가 제어 메시지를 주입하면 해당 CAN 메시지의 주기가 빨라지게 되어 탐지가 가능하다. 그러나 메시지 주기 기반의 차량용 IDS는 CAN 메시지의 전송 주기를 유지하면서 공격을 수행하는 발전된 공격 방법에 대하여 한계점을 갖고 있다.

메시지 주기 기반 IDS의 한계를 극복하기 위해, 행동 특징 (Behavioral Characteristics)을 분석하여 탐지하는 차량용 IDS가 개발되었다 [10][11][12]. ECU들은 고유의 행동 특징을 갖고 있어 이를 이용해서 각각의 ECU를 식별할 수 있다. 만약 식별된 ECU가 제어 메시지를 전송하는 ECU가 아니라면, 해당 ECU는 공격에 사용되고 있는 ECU로 간주할 수 있다. Murvay and B. Groza는 ECU들의 전기적인 특징을 이용해 ECU를 식별하는 방법을 제안하였다 [12]. 하지만 온도변화와 같이 주변의 상황이 변화하면 전기적 성질도 함께 변화하기 때문에 ECU들이 항상 올바르게 식별되는 것을 보장할 수 없다. 또한 해당 기법은 CAN 통신 프로토콜의 Arbitration decision에 대한 고려 없이 설계되었다. Arbitration decision이란 CAN 통신 프로토콜에서 빈번하게 나타나는 현상으로, 다수의 ECU들이 동시에 CAN 메시지를 전송하고자 하여 충돌할 때 메시지 전송의 우선순위를 정해주는 절차이다. 충돌이 발생하면 각 ECU들의 전기적 특징이 섞일 수 있기 때문에 이를 반드시 고려해줘야 한다. Choi 등은 Arbitration decision을 고려하여 전기적 특징을 통해 ECU를 식별하는 방법을 제안하였다 [11]. 그러나 표준 CAN Frame이 아닌 Extended CAN Frame을 활용하였기 때문에 ECU들의 소프트웨어 업데이트가 필요하다는 한계가 존재한다. 이에 따라 ECU들의 소프트웨어 업데이트 없이 식별할 수 있는 기법을 Choi 등이 제안

하였다 [13]. 더불어 ECU들이 가진 고유한 clock skew를 계산하여 ECU들을 식별하는 방법이 Cho 등에 의하여 제안되었다 [10]. 하지만 clock skew 또한 모방 가능하다는 것이 증명되어 공격자가 해당 기법을 우회할 수 있는 방법이 발표되었다 [14].

2.3 CAN Bus 리버싱

효과적인 IDS를 설계하기 위해서는 CAN 메시지 내의 Data Field의 의미를 알아야 한다. 자동차 제조회사들은 해당 정보를 DBC Format File에 정의하고 있다. 그러나 이들은 보안 문제로 인해 DBC Format File을 외부에 공개하지 않아 외부 연구자들로 하여금 CAN 메시지 분석에 어려움을 겪게 한다. 이러한 문제를 극복하기 위해 Bit Flip Rates를 이용하여 CAN Data Field의 Signal 정보를 식별하는 READ 기법이 제안되었다 [15]. READ는 비트 값이 변화하는 정도를 나타내는 Bit Flip Rate을 Data Field 내의 각 비트별로 계산하여 해당 비트가 어떤 Signal인지 식별하는 방법이다. 하지만 READ는 Total Bit Flip Rates, 즉 전체 데이터의 Bit Flip Rates을 계산하므로 데이터셋에 따라 결과가 달라진다는 특징이 있다. 예를 들어 상당히 짧은 시간 동안 동작한 기능에 대해서 Total Bit Flip Rates을 계산한다면 0에 가까운 값이 나올 것이므로, Signal을 정확히 분류하기에 적절치 않다.

이에 따라 Total Bit Flip Rates이 아닌 시계열을 통해 계산한 Bit Flip Rates으로 Signal을 식별하는 기법이 제안되었다 [16]. 연속된 비트의 Bit Flip Rate을 비교하여 두 값의 상관관계가 높다면 같은 Signal로 분류하는 기법으로, READ보다 높은 정확도를 보인다.

Bit Flip Rates뿐만이 아닌 CAN Data Field의 여러 특징을 활용하여 Signal을 식별하는 방법 또한 제안되었다 [17]. 먼저 Byte 단위의 Byte Flip Rates을 계산하여 Signal 분류를 한 다음, Bit Flip Rates을 계산하여 두 값을 비교함으로써 Signal을 READ보다 명확하게 식별하였다. Byte Flip Rates과 Bit Flip Rates 두 값에 각 Signal들에 따라 다른 특징을 보이는 Distinct Value Rate을 곱하여 Threshold를 정함으로써 Signal을 식별했다.

2.4 CAN Data Field 내 Checksum 값 유추

Kim 등은 차량에 이상 작동 현상을 발생시키는 Fuzzing 입력 값을 생성하기 위해 CAN Data Field 내의 Checksum 값을 딥러닝을 이용하여 유추하였다 [20]. DNN (Deep Neural Network) 모델에 Checksum을 제외한 Data Field를 입력 값으로, Checksum을 출력 값으로 설정해 학습시켰고, Fuzzing에 사용할 입력 값을 생성해 그에 맞는 Checksum을 출력하게 만들었다. 모델의 성능으로는 정밀도, 재현율 그리고 F1-score가 모두 98%의 결과를 보였다. 해당 기법을 이용해 Checksum 값을 계산할 수 있지만, 딥러닝 알고리즘 특성상 100%의 성능을 보이지 못해 오탐과 미탐이 생기게 된다. 또한 저자들은 4-bit의 Checksum을 가진 하나의 CAN ID만을 대상으로 학습을 진행했기 때문에, 8-bit, 16-bit의 Checksum의 경우 재학습을 진행해야 한다. 그러나 본 논문에서 제안하는 기법은 CAN 메시지 간의 관계성을 통해 Checksum을 계산하기 때문에 100%의 정확도를 보인다.

III. 배경지식

이번 장에서는 제안하는 기법의 이해를 위한 배경지식에 대해 설명한다.

3.1 자동차 내부 네트워크

자동차 내부에는 여러 개의 ECU가 탑재되어 네트워크를 형성하고 있다. ECU는 특정 기능을 제어함으로써 차량의 상태를 주기적으로 측정한다. 예를 들어, 엔진 기능을 담당하는 ECU는 액셀이 눌렸을 때를 인식하는 센서를 가지고 있다. 해당 ECU는 엔진을 스스로 작동시키거나 다른 ECU에게 메시지를 전송해 작동시킨다. 이러한 자동화된 방법은 차량을 효율적이고 안전하게 제어할 수 있게 한다. 실제로 최신 차량에는 70~100개의 ECU들이 존재하는데, 차량의 샤시(Chassis), 파워트레인(Powertrain), 바디(Body) 등 각각의 ECU들이 담당하고 있는 기능별로 나뉘어 여러 개의 서브 네트워크로 구성되어 있다.

3.2 CAN 통신 프로토콜

ECU는 CAN 통신 프로토콜을 이용하여 상호 통신한다. CAN 통신 프로토콜은 자동차 내부 네트워크의 업계 표준이며, 네트워크 토폴로지는 Bus형태이다. ECU들은 CAN Bus에 연결되어 Data Frame을 이용해 브로드캐스팅 방식으로 메시지를 송수신한다. Data Frame 포맷은 Fig. 1.과 같다. 주요 Field는 메시지를 송신한 ECU를 식별할 수 있는 11bits의 Identifier Field와, Data의 길이를 나타내는 4bit의 DLC (Data Length Code), 0-8byte의 Data Field가 존재한다. 이 중 Data Field는 Counter, Sensor, Checksum 등 여러 개의 Signal로 구성되어 있다.

CAN 통신 프로토콜은 표준 Frame과 Extended Frame 두 가지의 포맷이 존재한다. 표준 Frame 포맷은 11bit의 ID Field를 가지며, Extended Frame 포맷은 29bit의 ID Field를 가진다. 일반적으로 표준 Frame 포맷으로 통신하며, 11bit의 ID Field로 구성되어 있으므로 최대 CAN ID의 수는 2^{11}개가 된다. ID Field는 메시지의 우선순위를 결정할 수 있으며, 서로 다른 ECU들은 같은 ID 값을 사용해 메시지를 송신할 수 없고 하나의 ECU는 여러 개의 ID로 메시지를 송신할 수 있다는 특징이 있다.

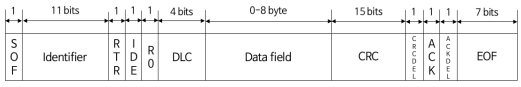

Fig. 1. The data frame format of CAN

3.3 DBC Format File

Vector사는 DBC Format File을 구체적으로 공개했고 CAN 네트워크 통신에 대해 기술하였다 [18]. DBC Format File에는 각 섹션들과 그 속성들에 대한 정보가 존재한다. 이 중 메시지 섹션에는 CAN 네트워크 내 모든 Frame의 이름과 Signal 정보가 아래와 같이 정의되어 있다.

messages = { message } ;
message = BO_ message_id message_name ' : ' message_size transmitter { Signal } ;

첫 번째 줄은 CAN 메시지의 집합이고, 다음 줄은 적어도 하나의 Signal이 정의되어 있는 각 CAN 메시지를 뜻한다. BO_는 CAN 메시지의 객체 타입을 나타내고, message_id는 unsigned integer로 선언되어야 하며 CAN ID와 같은 의미이다. message_size는 DLC와 같으며 마지막 Signal은 CAN 메시지의 Data Field 내에 존재하는 Signal들의 집합이다 [15]. DBC Format File의 전반적인 구조는 Fig. 2.와 같다.

이렇듯 DBC Format File을 통해 CAN 메시지의 정보를 알 수 있고, 특히 Data Field 내에 어떤 Signal이 존재하는지 파악 가능하다. Signal은 차량의 상태 정보를 표현하며 Signal, Counter, Checksum으로 구성되어 있다. 이 중 Checksum Signal은 CAN Data Frame의 15비트인 CRC Field와는 다르게 계산된다. CRC Field의 계산 방법은 CAN 표준 문서에 공개되어 있지만, Checksum Signal의 계산 방법은 자동차 제조회사마다 다른 알고리즘을 사용하는 것으로 파악된다. 그러나 자동차 제조회사들은 DBC Format File을 외부에 공개하지 않아 알고리즘 분석이 어렵다. 이에 대한 대응책으로 Bit Flip Rates 값을 통해 Checksum을 식별할 수 있는데, Bit Flip Rate의 확률 분포가 0.5에 근접하면 Checksum으로 판단한다.

```
DBC_file =
        version
        new_symbols
        bit_timing          (*obsolete but required*)
        nodes
        value_tables
        messages
        message_transmitters
        environment_variables
        environment_variables_data
        Signal_types
        comments
        attribute_definitions
        sigtype_attr_list
        attribute_defaults
        attribute_values
        value_descriptions
        category_definitions (*obsolete*)
        categories           (*obsolete*)
        filter               (*obsolete*)
        Signal_type_refs
        Signal_groups
        Signal_extended_value_type_list ;
```

Fig. 2. The structure of DBC format file

IV. Checksum Signal 비트 위치 식별

이번 장에서 우리는 CAN 메시지의 Data Field 내에 정의된 Signal 중 Checksum 값을 계산하기 위해, 먼저 Checksum Signal의 비트 위치를 식별하는 방법을 두 단계로 나누어 설명한다. 해당 과정은 Fig. 3.의 좌측에 해당된다. 첫 번째로, Bit Flip Rates 계산 결과를 활용하여 Checksum의 최상위 비트 위치를 파악한다. 두 번째로 해당 위치부터 Hash Table을 생성하여 Checksum을 식별한다.

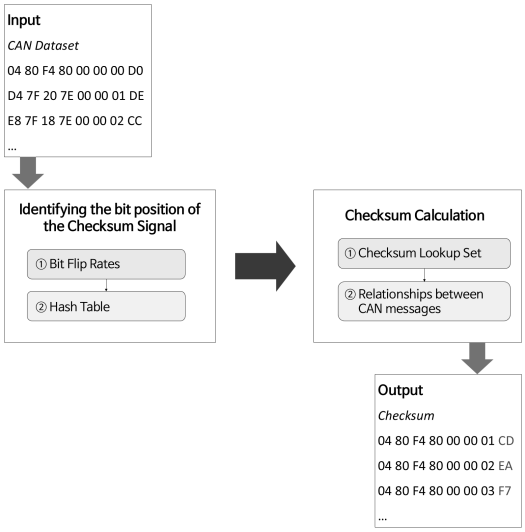

Fig. 3. The process of Checksum Calculation

4.1 Bit Flip Rates

먼저 우리는 Marchetti and Stablili가 제안한 Bit Flip Rates 계산 결과를 기반으로 Signal을 식별하는 READ 알고리즘을 활용한다 [15]. Fig. 4.는 특정 CAN ID의 Bit-Flip Rates 결과를 나타낸 그림인데, 해당 연구에 따라 Signal, Counter, Checksum으로 분류한다. Signal은 엔진 RPM, 자동차 휠 속도 등 각 ECU가 가지는 Physical 기능에 대한 값을 의미하고, Counter는 이전의 CAN 메시지와 비교하여 항상 1씩 증가하는 값을 의미한다. Checksum은 순환 중복 검사(Cyclic Redundancy Check, CRC)와 같은 역할로, Data Field 내의 전송 오류를 감지한다.

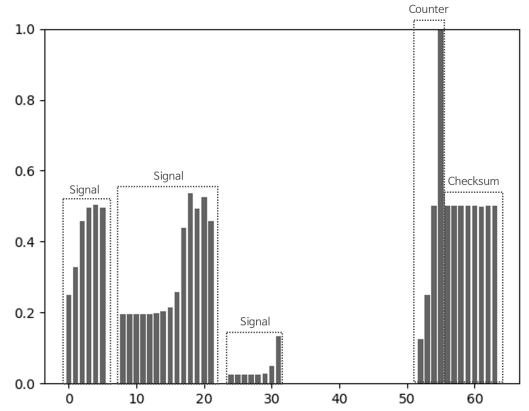

Fig. 4. Signal identification within data field based on Bit Flip Rates

수집한 CAN 트래픽을 CAN ID별로 나누어 S_{ID}로 정의한다. 즉, S_{ID}는 동일한 CAN ID에 대한 CAN 메시지들의 집합이다. S_{ID}에서 i번째 CAN 메시지를 M_i라 하자. 연속된 CAN 메시지 M_{i-1}와 M_i에 대하여, Bit Flip은 Data Field의 각 비트 위치별로 계산된다. 예를 들어 M_{i-1}의 k번째 비트가 0이고 M_i의 k번째 비트가 1이라면 k번째 비트의 Bit Flip 횟수는 1 증가한다. M_{i-1}과 M_i의 각 비트 위치에서 계산되는 Bit Flip 횟수를 BF_i라 정의한다. $i \geq 1$, $M_i \in S_{ID}$일 때, BF_i는 식 (1)과 같다.

$$BF_i = M_{i-1} \oplus M_i \tag{1}$$

각 CAN ID에 대한 CAN 메시지는 항상 동일한 DLC 값을 가지고 있으므로 연속된 M_{i-1}과 M_i에 대하여 비트단위 연산 (Bit-wise Operation)이 가능하다. S_{ID}에 대하여 모든 CAN 메시지들의 모든 Bit Flip 횟수를 우리는 Total Bit Flip이라고 하며, k번째 비트의 Total Bit Flip을 $TotalBF_k$로 정의한다. 또한, Total Bit Flip Rate을 $TotalBFR_k$로 정의한다. $TotalBF_k$과 $TotalBFR_k$은 식 (2)와 (3)과 같다.

$$TotalBF_k = \sum_{i=1}^{|S_{ID}|} k_i \tag{2}$$

```
function getBitFlipRates(S_ID)
  for i ← 1 to |S_ID| do
    M_i ∈ S_ID ← i-th data payload
    BF_i ← M_{i-1} ⊕ M_i
    /* k_i ∈ {0,1} k-th bit in BF_i */
    TotalBF_k ← TotalBF_k + k_i
  TotalBFR ← TotalBF / |S_ID|
  return TotalBFR

function getCounter(TotalBFR)
  ctFrom ← find(TotalBFR == 1)
  if ctFrom == null then
    return null  /* No Count */
  ctTo ← ctFrom
  while(True) do
    if TotalBFR_{ctTo-1} * 2 == TotalBFR_{ctTo}
   then
      ctTo ← ctTo - 1
    else
      /* Boundary of Counter Signal */
      return [ctTo ctFrom]

for all S_ID do
  TotalBFR = getBitFlipRates(S_ID)
  if getCounter(TotalBFR) then
    /* MSB of Checksum Signal */
    return ctFrom + 1
  else
    return null  /* No Checksum */
```

Fig. 5. The algorithm for identifying the position of the MSB of the checksum signal based on total Bit Flip Rate

$$TotalBFR_k = \frac{TotalBF_k}{|S_{ID}|} \quad (3)$$

위와 같이 Bit Flip Rates을 활용하여 식별된 Signal들 중 Checksum을 목적으로 사용되는 Signal이 존재하거나 존재하지 않을 수 있다. 우리는 Checksum을 쉽게 식별하기 위하여, Data Field 내에 Checksum이 Counter와 함께 존재한다고 가정한다. Counter는 Bit Flip Rates 결과를 이용하면 Checksum에 비해 상대적으로 쉬운 식별이 가능하다. Counter는 CAN 메시지마다 1씩 증가하는 패턴을 가지고 있으므로, Counter의 최하위 비트의 Bit Flip Rate는 항상 1이 된다. Counter가 존재하는 CAN ID를 분류한 후 해당 Data Field의 Counter 최하위 비트 다음 위치부터, 즉 Checksum의 최상위 비트부터 Hash Table을 생성한다. 해당 알고리즘의 pseudo-code는 Fig. 5.와 같다.

4.2 Hash Table

Checksum의 최상위 비트를 식별한 후 해당 위치부터 Hash Table을 생성한다. Hash Table이란 Key와 Value로 구성하여 데이터를 저장하는 자료구조이다. 전송 오류 검출을 위해 활용되는 Checksum은 항상 같은 입력값에 대하여 같은 출력값인 Checksum이 계산되어야 한다. 우리는 이러한 특징을 Hash Table에 적용하여 Data Field 내의 Checksum을 식별한다. 예를 들어, S_{ID}에 대한 CAN 메시지들이 64비트 크기의 Data Field를 가진다고 하자. 0부터 Counter의 최하위 비트까지와 Checksum의 최상위 비트 다음 비트부터 63번째 비트까지를 Input으로, Checksum의 최상위 비트를 Output으로 정의하여 각각을 Hash Table의 Key와 Value에 저장한다. S_{ID}에 대한 모든

```
C_ID : The subset of CAN IDs where the
Counter Signal exists
csFrom : MSB of Checksum Signal
hash_table = {}  /* dictionary */

for all C_ID do
  for k ← 1 to (DLC * 8) - (csFrom - 1) do
    for i ← 1 to |C_ID| do
      M_i ∈ C_ID ← i-th data payload
      key ← M_i[:csFrom] + M_i[csFrom+k:]
      value ← M_i[csFrom:csFrom+k]
    if key in hash_table then
      if hash_table[key] != value then
        /* Boundary of Checksum Signal */
        return [csFrom, csFrom+(k-1)]
    else
      hash_table[key] ← value
  return [csFrom, csFrom+(k)]
```

Fig. 6. The algorithm for identifying the checksum signal using Hash Table

CAN 메시지마다 같은 Key 값과 같은 Value값이 된다면 Value 값의 길이를 1비트 증가시키고 Key 값의 길이를 1비트 감소하여 저장한다. 위와 같은 과정을 Checksum의 조건을 만족하지 않을 때까지 반복한다. Fig. 6.은 해당 과정에 대한 pseudo-code이다.

V. Checksum Lookup Set

이번 장에서는 4장을 통해 식별된 Checksum을 기반으로 Checksum Lookup Set을 생성해 Checksum을 계산하는 방법을 제안한다. 대상으로 하는 데이터는 Fig. 7.과 같이 8-bit의 Checksum 앞에 4-bit의 Counter가 존재하는 Data Field이다.

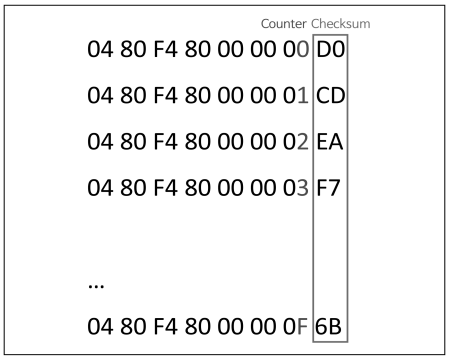

Fig. 7. Counter and Checksum in the Data Field

5.1 Checksum Lookup Set 생성

CAN 메시지 내의 Checksum이 어떤 값으로 구성되어 있는지 확인하기 위해 Checksum Lookup Set을 생성해 값을 분석한다. Checksum Lookup Set이란 Fig. 8.과 같이 Checksum 값으로 구성된 중복 값이 없는 집합으로, 본 논문에서는 총 16개의 Sets가 존재하고 각 Set의 크기 또한 16이며 Set 간 Checksum 값이 겹치지 않는다. 따라서 0x0~0xFF까지의 256개 Checksum이 모두 Checksum Lookup Sets에 존재하게 된다. Checksum Lookup Sets와 각 Set의 값들이 16개인 이유는 0x0에서 0xF로 구성된 16개의 Counter 값을 기반으로 Checksum Lookup Set을 생성했기 때문이다. Fig. 7.로 예를 들자면, '04 80 F4 80 00 00 00 D0'에서 'D0'이 Checksum이고 그 앞의 '0'이 Counter이다. 여기서 Counter를 1씩 더해 '04 80 F4 80 00 00 00 D0'부터 '04 80 F4 80 00 00 0F 6B'까지 16개의 값을 수집한다. 그리고 해당 데이터로부터 수집한 16개의 Checksum 값을 Checksum Lookup Set에 저장한다. 따라서 Checksum Lookup Set의 크기는 16이 된다. 이러한 방식으로 256개의 모든 Checksum 값을 수집한다면 16개의 Checksum Lookup Sets가 만들어지게 된다.

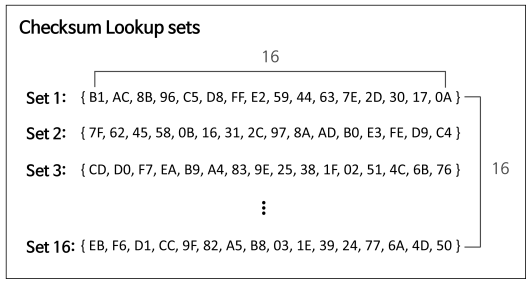

Fig. 8. Checksum Lookup Sets

5.2 같은 Checksum을 가진 다른 CAN 메시지의 Checksum Lookup Set

16개의 Checksum 값으로 구성된 Checksum Lookup Set을 만든 후, 해당 Lookup Set에 있는 Checksum과 하나라도 같은 Checksum 값을 가지고 그 외의 부분은 다른 CAN 메시지의 경우에는 결국 해당 Checksum Lookup Set에 있는 모든 값을 Checksum으로 가지는 것을 발견하였다. 즉, Fig. 9에서 ①번 Data Field의 Checksum Lookup Set을 생성해 놓았다고 할 때, ②번 Data Field의 Counter 값이 0x0일 때 'E2'를 가지는데 이는 ①번의 Checksum Lookup Set에 있는 값이다. 그 후 ②번의 Counter 값을 1씩 더해가면서 16개의 Checksum 값을 찾으면 결국 ②번의 모든 16개의 Checksum 값이 ①번의 Checksum Lookup Set 안에 존재한다는 것을 파악하였다. 순서는 다르지만 같은 Checksum Lookup Set을 가지는 것이다. 이렇게 Checksum 값이 같고 그 외의 부분은 다른 Data Field들은 서로 같은 Checksum Lookup Set을 공유한다는 것을 알 수 있다.

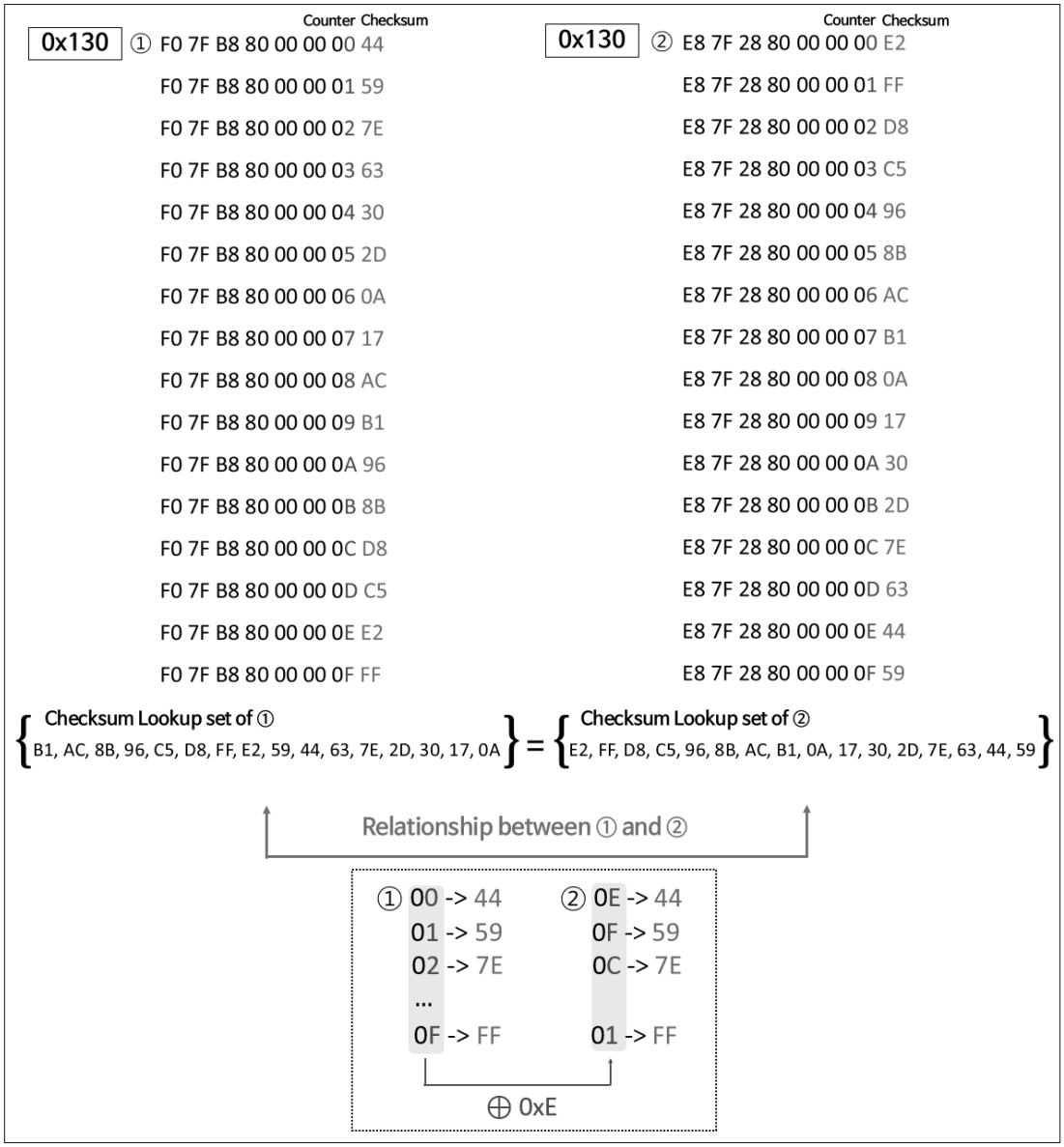

Fig. 9. Relationship between data fields using checksum

5.3 CAN 메시지 간의 관계성을 통한 Checksum 계산

Checksum 값이 같고 그 외의 부분은 다른 Data Field를 가진 CAN 메시지들은 Checksum Lookup Set이 같다는 것을 5.2절에서 발견하였다. 더불어 이들 간의 관계성 또한 파악하여 Checksum 값을 모르더라도 Checksum 값을 계산할 수 있다. Fig. 9.를 보면 ①번에서 Counter가 0x0인 Checksum 값은 '44'인데, ②번에서 Checksum이 '44'인 Counter는 0xE다. 마찬가지로 ①번에서 Counter가 0x1일 때 Checksum 값은 '59'인데, ②번에서 '59'인 Checksum을 가지는 Counter는 0xF다. 나머지 Counter 값도 모두 XOR 0xE만큼 차이나는 것을 알 수 있다. 이렇듯 ②번의 모든 Checksum값을 알지 못하더라도 XOR 0xE만큼의 차이를 이용해 Counter 0xF까

지의 Checksum 값을 유추할 수 있다. 물론 0xE에 해당하는 숫자는 Data Field마다 달라진다. 정리하자면, 16개로 구성된 하나의 Checksum Lookup Set을 알고 있는 상태로 그 Lookup Set의 값 중 같은 Checksum을 가지는 Data Field를 하나라도 구했다면, 해당 Data Field의 나머지 0x0~0xF에 해당하는 Checksum 값을 모두 알 수 있다.

VI. 실험 및 평가

6.1 실험 환경

4장에서는 Checksum을 계산하기 위해 Bit Flip Rates과 Hash Table을 통해 Checksum Signal의 비트 위치를 식별하고, 5장에서는 Checksum Lookup Set을 생성해 Checksum을 계산하였다. 4장과 5장의 기법을 이용해 Checksum을 계산할 수 있는지 평가하기 위하여, 우리는 실제 차량을 1시간가량 주행하면서 CAN Traffic을 수집하였다. Fig. 10.은 실제 차량에서의 CAN Traffic 수집 환경을 보여주고 있다. 실험 차량은 2020년형 아반떼이며, 'Value CAN'을 이용해 PC와 차량을 연결한 다음 'Vehicle Spy'라는 CAN 통신 모니터링 툴로 CAN 메시지 데이터를 수집하였다. 또한, 우리가 수집한 CAN Traffic뿐만 아니라 평가의 공정성을 위하여 인터넷에 공개되어 있는 데이터셋을 활용해 우리가 제안하는 방법론에 대한 평가를 함께 진행하였다 [19], [20]. Table. 1.은 실험 환경 구성에 사용된 데이터셋 및 실험 장비들을 보여준다.

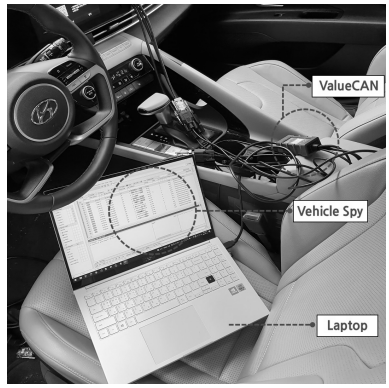

Fig. 10. Experimental setup

Table 1. List of experimental equipment

Category	Name	Description
Experimental vehicle	Avante CN7 (2020)	Experimental vehicle for data collection in CAN
CAN communication monitoring SW	Vehicle Spy	CAN communication traffic analysis software installed on a PC
CAN Adapter	ValueCAN	Hardware connecting a PC and a vehicle for collecting CAN communication traffic
Dataset	Self-Collected Dataset	
	Car-Hacking Dataset[19]	
	Kim et al. [20]	
Laptop	Intel i5 10TH GEN DDR4 8GB Window 10	

6.2 Checksum Signal 비트 위치 식별 검증

수집한 데이터와 공개된 데이터셋을 대상으로 4장의 기법을 이용해 CAN ID별로 Checksum을 식별하였다. 먼저 Bit Flip Rates 계산을 통해 Counter와 Checksum이 모두 존재하는지 확인했다. 두 데이터셋에 대한 Bit Flip Rates 계산 결과는 Fig. 11.과 같다. 데이터셋의 전체 CAN ID 중 일부 ID에 대한 Bit Flip Rates로, Counter와 Checksum이 존재하는 것으로 확인되는 ID가 있는 반면 존재하지 않는 ID도 있다. 그 후 Bit Flip Rates로 Counter와 Checksum이 모두 존재하는 것을 확인한 ID를 대상으로 Checksum의 MSB부터 Hash Table을 생성하여 Checksum 비트 위치를 식별했다. 식별 결과는 Table. 2.와 같다. 직접 수집한 데이터셋에서 Counter와 Checksum이 모두 존재하는 것으로 판단되는 ID는 총 86개의 ID 중 9개였고, 공개된 Car-Hacking 데이터셋에서는 27개의 ID 중 4개였다. 데이터셋의 ID들에는 4-bit와 8-bit인 Checksum이 있는데, 제안하는 Checksum 계산 방법은 8-bit인 Checksum만을

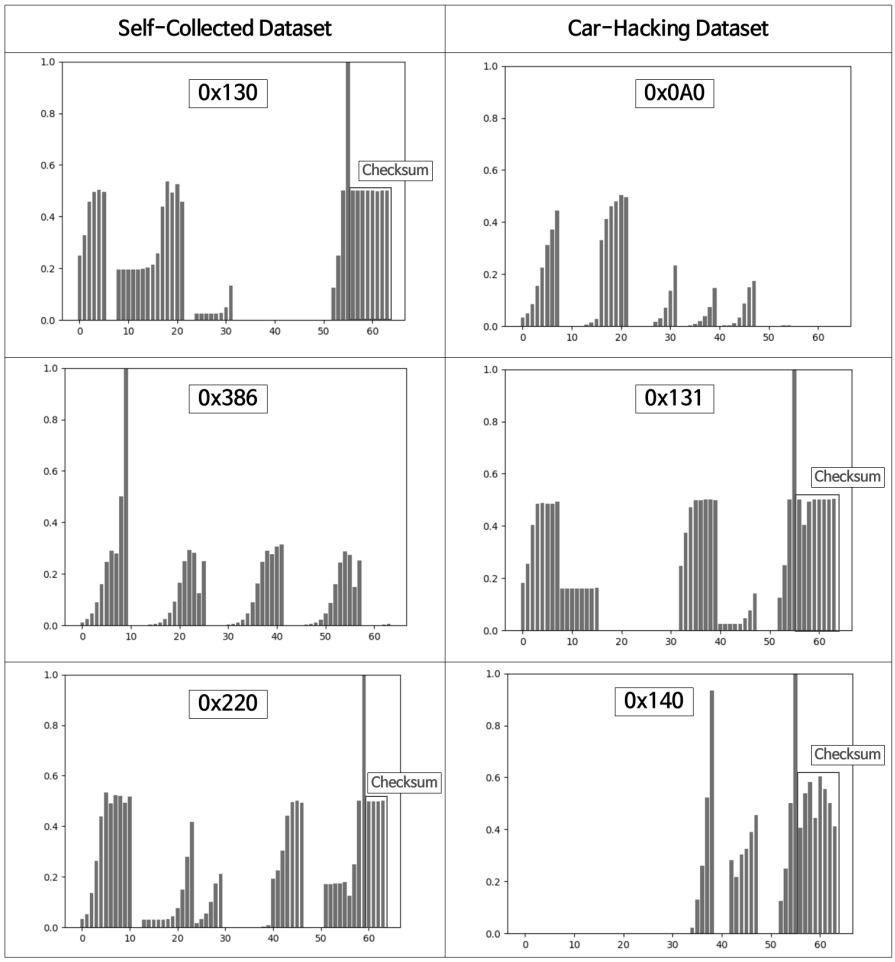

Fig. 11. Results of Bit Flip Rates for the Datasets

Table 2. Results of checksum identification

Dataset	CAN ID	Identified Checksum Bit Positions
Self-Collected Dataset	0x2B0 (48bits)	40, 41, 42, 43, 44, 45, 46, 47
	0x130 (64bits)	56, 57, 58, 59, 60, 61, 62, 63
	0x140 (64bits)	56, 57, 58, 59, 60, 61, 62, 63
	0x164 (32bits)	24, 25, 26, 27, 28, 29, 30, 31
	0x220 (64bits)	60, 61, 62, 63
	0x251 (64bits)	24, 25, 26, 27, 28, 29, 30, 31
	0x381 (64bits)	48, 49, 50, 51, 52, 53, 54, 55
	0x391 (64bits)	56, 57, 58, 59, 60, 61, 62, 63
	0x490 (64bits)	56, 57, 58, 59, 60, 61, 62, 63
Car-Hacking Dataset [19]	0x002 (64bits)	56, 57, 58, 59, 60, 61, 62, 63
	0x130 (64bits)	56, 57, 58, 59, 60, 61, 62, 63
	0x131 (64bits)	56, 57, 58, 59, 60, 61, 62, 63
	0x140 (64bits)	56, 57, 58, 59, 60, 61, 62, 63

대상으로 하므로 이 중 8-bit인 Checksum을 가진 ID로 Checksum 계산 방법을 검증하였다.

6.3 Checksum 계산 방법 검증

6.2절에서 도출한 CAN ID들을 대상으로 Checksum 계산 방법을 검증했다. 같은 CAN ID 뿐만 아니라 서로 다른 CAN ID 사이에서도 검증을 진행했다. Fig. 12.는 서로 다른 CAN ID에서의 검증 과정이다. 먼저, 데이터셋에서 0x130의 ID를 가진 데이터 중 Data Field를 하나 선택하여 하나의 Checksum Lookup Set을 생성했다. 그 후 0x140에서 같은 Checksum 값을 가진 다른 Data Field를 발견하면, 0x140의 Data Field의 Checksum이 생성해 놓은 0x130의 Checksum Lookup Set에 포함되는지 파악했다. 포함되는 것을 확인 후 0x130에서 'F5'의 Checksum 값을 갖고 있는 Data Field를 찾아 Counter끼리 XOR 연산하여 얼마만큼의 차이를 보이는지 확인하였다. 그림에서는 0xB만큼의 차이를 보인다. 따라서 차이를 알 수 있다면, 나머지 15개 Data Field의 Checksum 값도 XOR 0xB 연산하여 계산할 수 있다. 이를 실제 수집한 데이터셋과 Car-Hacking, Kim et al.의 데이터셋에서 모두 검증해 본 결과, XOR 연산하여 계산한 Checksum이 알맞게 나오는 것을 확인할 수 있었다.

VII. 결 론

본 논문에서는 Data Field 내의 Checksum의 비트 위치를 Bit Flip Rates와 Hash Table을 통해 식별했다. 또한 Checksum Lookup Set을 생성하여 Checksum 값이 어떻게 구성되어 있는지 확인하였고, CAN 메시지의 Data Field 간의 관계성을 통해 Checksum을 계산하는 기법을 제안하였다.

다만 제시한 기법에는 한계점이 존재한다. 먼저, Checksum Lookup Set을 생성할 충분한 데이터가 있어야한다. Counter가 0x0부터 0xF까지 모두 존재하는 데이터 추출을 위해서는 방대한 데이터셋이

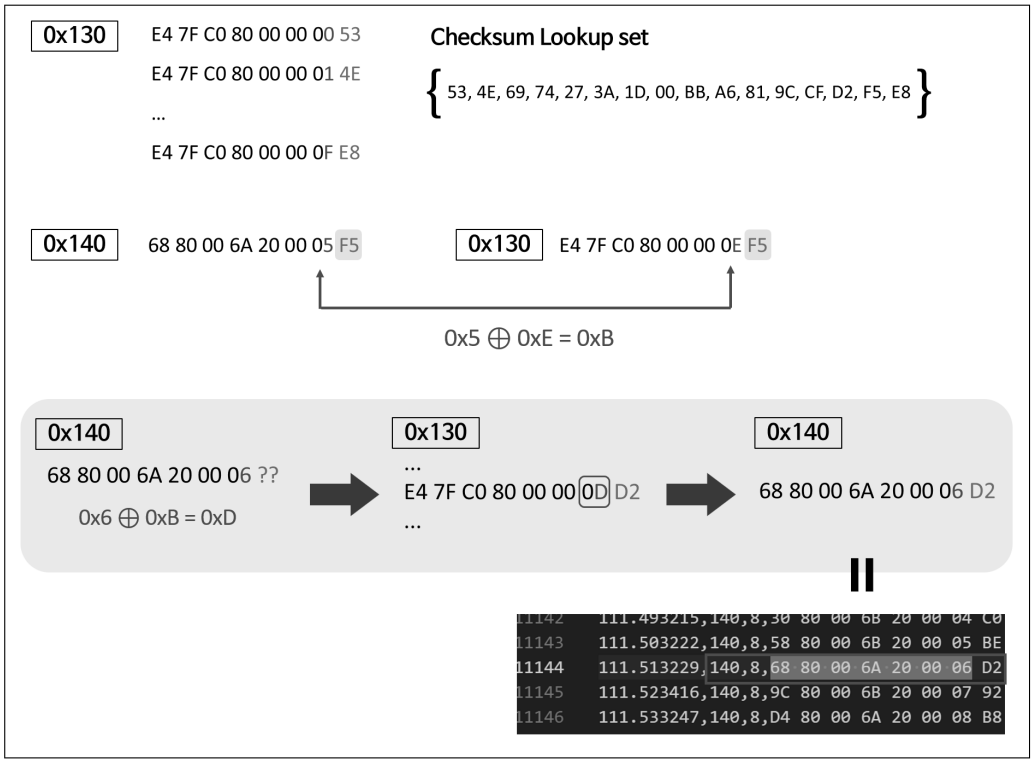

Fig. 12. Verification Process of Checksum Calculation Methods for Different CAN IDs

요구된다. 또한 Counter 값을 기반으로 Data Field 간의 관계성을 파악하므로, Data Field에는 Counter 값이 반드시 존재해야 한다. 더불어 본 논문에서는 Checksum이 8-bit인 경우만을 대상으로 하였다. 크기가 4-bit 및 16-bit인 Checksum은 데이터가 부족하여 충분한 실험과 검증을 하지 못했다. 그러나 같은 Checksum을 가지는 다른 Data Fields와의 관계를 유추하는 본질적인 방법은 같을 것으로 판단된다.

향후 연구에서는 4-bit, 16-bit인 Checksum의 경우에서도 제안하는 기법이 적용되는지 확인하고, 실제 실험 차량과 공개된 데이터셋의 실험 차량인 현대, 기아차뿐만 아니라 다양한 제조회사 차량에서도 데이터를 수집하여 검증할 계획이다.

References

[1] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, and Tadayoshi Kohno, "Experimental Security Analysis of a Modern Automobile," in Proceedings of the 2010 IEEE Symposium on Security and Privacy, pp. 447-462, May. 2010

[2] Wired, "Hackers remotely kill a jeep on the highway-with me in it," https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/, June. 2024

[3] UNECE, "ECE/TRANS/WP.29/2020/79 REVISED," https://unece.org/fileadmin/DAM/trans/doc/2020/wp29grva/ECE-TRANS-WP29-2020-079-Revised.pdf, June. 2024

[4] GitHub, "opendbc," https://github.com/commaai/opendbc, June 2024

[5] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Comprehensive experimental analyses of automotive attack surfaces," in Proc. USENIX Secur. Symp., San Francisco, CA, USA, pp. 1–16, Aug. 2011

[6] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and vulnerable: A story of telematic failures," in Proc. 9th USENIX Workshop Offensive Technol., pp. 1–9, Aug. 2015

[7] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive can networks—Practical examples and selected short-term countermeasures," in Proc. Int. Conf. Comput. Safety, Rel., Secur., pp. 235–248, Sep. 2008

[8] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in Proc. IEEE Intell. Vehicles Symp. (4), pp. 1110–1115, Jun. 2011

[9] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in Proc. 6th Int. Conf. Inf. Assurance Secur. (IAS), pp. 92–98, Aug. 2010

[10] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in Proc. 25th USENIX Secur. Symp., pp. 911–927, Aug. 2016

[11] W. Choi, H. J. Jo, S. Woo, J. Y. Chun, J. Park, and D. H. Lee, "Identifying ECUs using inimitable characteristics of signals in controller area networks," in IEEE Transactions on Vehicular Technology, vol. 67, no. 6, pp. 4757-4770, June. 2018

[12] P.-S. Murvay and B. Groza, "Source identification using signal characteristics in controller area networks," IEEE Signal Process. Lett., vol. 21, no. 4, pp. 395–399, Apr. 2014.

[13] W. Choi, K. Joo, H. J. Jo, M. C. Park and D. H. Lee, "VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion

Detection System," in IEEE Transactions on Information Forensics and Security, vol. 13, no. 8, pp. 2114-2129, Aug. 2018

〔14〕 S. U. Sagong, X. Ying, A. Clark, L. Bushnell and R. Poovendran, "Cloaking the Clock: Emulating Clock Skew in Controller Area Networks," ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), pp. 32-42, Apr. 2018

〔15〕 M. Marchetti and D. Stabili, "Read: Reverse engineering of automotive data frames," IEEE Trans. Inf. Forensics Secur., vol. 14, no. 4, pp. 1083 - 1097, Apr. 2019.

〔16〕 W. Choi, S. Lee, K. Joo, H. J. Jo and D. H. Lee, "An Enhanced Method for Reverse Engineering CAN Data Payload," in IEEE Transactions on Vehicular Technology, vol. 70, no. 4, pp. 3371-3381, Apr. 2021

〔17〕 X. Lin, B. Ma, X. Wang, Y. He, R. P. Liu and W. Ni, "Multi-layer Reverse Engineering System for Vehicular Controller Area Network Messages," IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 1185-1190, May. 2022

〔18〕 Vector, "DBC File Format Documentation," http://mcu.so/Microcontroller/Automotive/DBC_File_Format_Documentation.pdf, June. 2024

〔19〕 HCRL, "Car-Hacking Dataset," https://ocslab.hksecurity.net/Datasets/car-hacking-dataset, June. 2024

〔20〕 H. Kim, Y. Jeong, W. Choi, and H. J. Jo, "An Efficient ECU Analysis Technology through Non-Random CAN Fuzzing," Journal of the Korea Institute of Information Security & Cryptology, 30(6), pp. 1115 - 1130, Dec. 2020

〈저자소개〉

이 경 연 (Gyeongyeon Lee) 학생회원
2023년 2월: 한성대학교 IT융합공학부 졸업
2023년 3월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 자동차 보안, 로봇 보안

김 형 훈 (Hyunghoon Kim) 학생회원
2019년 8월: 한림대학교 컴퓨터공학과 졸업
2021년 8월: 숭실대학교 일반대학원 석사 졸업
2024년 3월~현재: 연세대학교 정보대학원 박사과정
〈관심분야〉 자동차 보안, IoT/CPS 보안, 네트워크 보안

이 동 훈 (Dong Hoon Lee) 종신회원
1983년 8월: 고려대학교 경제학사 졸업
1987년 12월: Oklahoma University 전산학과 석사 졸업
1992년 5월: Oklahoma University 전산학과 박사 졸업
1993년 3월~1997년 2월: 고려대학교 전산학과 조교수
1997년 3월~2001년 2월: 고려대학교 전산학과 부교수
2001년 3월~현재: 고려대학교 정보보호대학원 교수
〈관심분야〉 암호 프로토콜, 암호이론, USN이론, 키 교환, 익명성 연구, PET기술

최 원 석 (Wonsuk Choi) 종신회원
2008년 2월: 서울시립대학교 수학과 졸업
2013년 2월: 고려대학교 정보보호대학원 석사 졸업
2018년 8월: 고려대학교 정보보호대학원 박사 졸업
2020년 2월: 고려대학교 정보보호연구원 연구교수
2023년 2월: 한성대학교 IT융합공학부 조교수
2023년 3월~현재: 고려대학교 정보보호대학원 조교수
〈관심분야〉 센서 보안, 자동차 보안, 로봇 보안, 암호 프로토콜