



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2019-0070076
(43) 공개일자 2019년06월20일

(51) 국제특허분류(Int. Cl.)

H04L 9/08 (2006.01)

(52) CPC특허분류

H04L 9/0861 (2013.01)

H04L 2209/12 (2013.01)

(21) 출원번호 10-2017-0170557

(22) 출원일자 2017년12월12일

심사청구일자 2018년01월11일

(71) 출원인

서강대학교산학협력단

서울특별시 마포구 백범로 35 (신수동, 서강대학교)

(72) 발명자

박수용

서울특별시 마포구 독막로 18길 5 상수 두산 위브 아파트 101동 1001호

이준하

서울특별시 마포구 독막로 38길 32 마포나루타운 102동 202호

박지우

서울특별시 서초구 서초중앙로 24길 33 서초교대 e편한세상 106동 1402호

(74) 대리인

이지연

전체 청구항 수 : 총 1 항

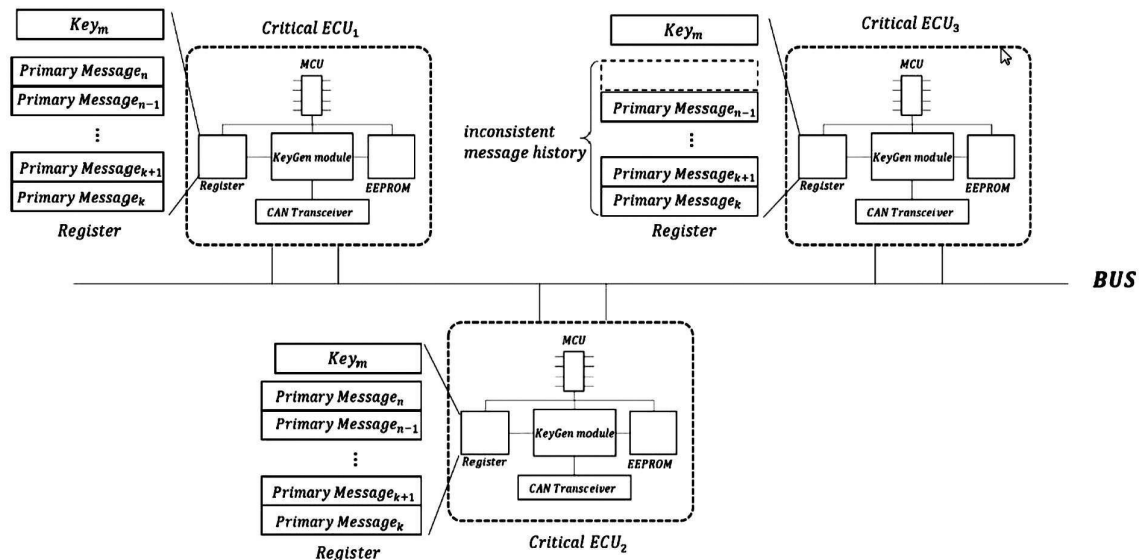
(54) 발명의 명칭 분산 네트워크 시스템의 전자 제어 장치 및 상기 전자 제어 장치의 분산 합의 프로토콜 방법

(57) 요약

자동차의 내부에서는 ECU(Electric Control Unit)이라고 불리는 제어장치들이 버스(bus) 형태의 분산 네트워크로 구성되어 차량을 제어한다. 또한 차량 내부 네트워크는 CAN(Controller Area Network)라는 프로토콜을 기반으로 통신하는 것이 그 특징이다. 그러나 차량 내부에서만 통신하던 네트워크가 커넥티드 카, 자율 주행 등의 등장으

(뒷면에 계속)

대표도 - 도3



로 외부망과 연결되는 경우가 많아지고 있다. 이러한 상황아래 차량 내부 네트워크는 버스라는 구조적 특성상 누구나 접근하여 메시지를 전송할 수 있기 때문에 해킹의 위험에 노출되어 있다. 특히나 차량 해킹은 인명피해로 직결될 수 있는 문제이기 때문에 해커가 전송하는 비 인가된 메시지를 방어할 수 있는 다양한 연구들이 진행되고 있다. 특히 데이터의 위변조 및 무결성에 강점을 보이는 블록체인 기술을 응용한 차량 내부 메시지 인증 기술이 연구가 등장했다. 해당 연구에서는 블록체인 기술의 특징을 활용하여 메시지 내역을 기반으로 암호화 인증 키를 동적으로 생성하고 있다. 그러나 어느 한 ECU가 메시지 수신에 실패 할 경우 메시지 내역이 달라져 차량 내부 네트워크의 인증 키가 일치하지 않는 한계를 가지고 있다. 차량 내부 네트워크와 같이 분산된 환경에서 시스템의 참여자(노드)들이 동일한 값을 유지하기 위한 방법으로는 분산 합의 프로토콜이 대표적이다. 그러나 기존 분산합의 프로토콜에 관한 연구들은 차량 내부 네트워크의 특성(CAN)과 실시간성에 대한 고려가 부족하다. 본 발명에서는 차량 내부 네트워크에서 동작하는 블록체인 기반 동적 키 생성 환경에서 모든 ECU들이 메시지 수신 실패에도 동일한 키를 유지 할 수 있는 분산 합의 프로토콜을 제시한다. 이 프로토콜은 차량 내부 네트워크의 특성을 반영하였고 실시간성을 유지하는 것을 실험적으로 검증하였으며, 또한 분산 합의 프로토콜이 만족해야 하는 속성들에 대한 이론적 검증을 수행하여 모든 ECU들이 프로토콜의 실행으로 동일한 키 값을 가지는 것을 보장하였다.

(52) CPC특허분류

H04L 2209/38 (2013.01)

H04L 2209/84 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호 2017-0-01628

부처명 교육과학기술부

연구관리전문기관 정보통신기술진흥센터

연구사업명 정보통신기술인력양성사업

연구과제명 적응형 블록체인 플랫폼 기술 개발 및 전문 인력 양성

기 여 율 1/1

주관기관 서강대학교 산학협력단

연구기간 2017.06.01 ~ 2017.12.31

명세서

청구범위

청구항 1

차량 내부 네트워크의 블록체인 기반 동적 암호화키 생성 환경에서의 분산합의 프로토콜 방법에 관한 것으로서, 정상 상태, 에러 상태 및 합의 상태로 이루어지는 상태 전이 다이어그램을 구비하며, 상기 합의 상태는 에러 플래그 메시지, 합의 요청 메시지, 합의 메시지를 구비하고, 어떤 ECU가 에러 상태로 전이시 다른 ECU 들에게 에러 플래그 메시지들 전송하고, 에러 플래그를 수신한 ECU들은 키생성 및 메시지 저장을 중지하고 에러 상태로 전이하며, 합의 요청 메시지는 합의를 시작하기 위한 메시지로, primary message의 수신을 실패한 ECU가 복구되어 이후 주기적으로 primary message에 수신 성공시 다른 ECU들에게 합의를 요청하기 위해 합의 요청 메시지를 전송하며, 합의 메시지는 합의 요청 메시지를 수신한 ECU들이 합의를 위하여 사용하는 것을 특징으로 하는 분산 합의 프로토콜 방법.

발명의 설명

기술 분야

[0001] 본 발명은 차량 내부 네트워크의 블록체인 기반 동적 암호화키 생성 환경에서의 분산합의 프로토콜에 관한 것이다.

배경 기술

[0002] 자동차의 내부에서는 ECU(Electric Control Unit)이라고 불리는 제어장치들이 버스(bus) 형태의 분산 네트워크로 구성되어 차량을 제어한다. 또한 차량 내부 네트워크는 CAN(Controller Area Network)라는 프로토콜을 기반으로 통신하는 것이 그 특징이다. 그러나 차량 내부에서만 통신하던 네트워크가 커넥티드 카, 자율 주행 등의 등장으로 외부망과 연결되는 경우가 많아지고 있다. 이러한 상황아래 차량 내부 네트워크는 버스라는 구조적 특성상 누구나 접근하여 메시지를 전송할 수 있기 때문에 해킹의 위험에 노출되어 있다. 특히나 차량 해킹은 인명피해로 직결될 수 있는 문제이기 때문에 해커가 전송하는 비인가된 메시지를 방어할 수 있는 다양한 연구들이 진행되고 있다. 특히 데이터의 위변조 및 무결성에 강점을 보이는 블록체인 기술을 응용한 차량 내부 메시지 인증 기술이 연구가 등장했다. 해당 연구에서는 블록체인 기술의 특징을 활용하여 메시지 내역을 기반으로 암호화 인증 키를 동적으로 생성하고 있다. 그러나 어느 한 ECU가 메시지 수신에 실패 할 경우 메시지 내역이 달라져 차량 내부 네트워크의 인증 키가 일치하지 않는 한계를 가지고 있다.

[0003] 차량 내부 네트워크와 같이 분산된 환경에서 시스템의 참여자(노드)들이 동일한 값을 유지하기 위한 방법으로는 분산 합의 프로토콜이 대표적이다. 그러나 기존 분산합의 프로토콜에 관한 연구들은 차량 내부 네트워크의 특성(CAN)과 실시간성에 대한 고려가 부족하다.

[0004] 본 발명에서는 차량 내부 네트워크에서 동작하는 블록체인 기반 동적 키 생성 환경에서 모든 ECU들이 메시지 수신 실패에도 동일한 키를 유지 할 수 있는 분산 합의 프로토콜을 제시한다. 이 프로토콜은 차량 내부 네트워크의 특성을 반영하였고 실시간성을 유지하는 것을 실험적으로 검증하였으며, 또한 분산 합의 프로토콜이 만족해야 하는 속성들에 대한 이론적 검증을 수행하여 모든 ECU들이 프로토콜의 실행으로 동일한 키 값을 가지는 것을 보장하였다.

[0005] 자동차는 여러 개의 ECU(Electronic Control Unit)로 구성된 분산 시스템이다. 각각의 ECU는 Bus형태로 연결된 자동차 내부 네트워크를 통해 서로 통신하며 동작한다. 최근 커넥티드 자동차, 무인 자율 주행 자동차 등이 활발히 연구됨에 따라 자동차의 내부 네트워크가 외부 인터넷에 연결되어 자동차 내부 네트워크의 해킹에 대한 위험이 높아지고 있다. 자동차 내부 네트워크는 실시간성을 확보하고자 네트워크의 복잡도를 간소화 시키기 위해

bus형태를 취하고 있으며 자동차 내부 네트워크의 특성상 해커가 일단 네트워크에 접근하면 엔진을 제어하는 ECU를 포함한 어느 ECU로든지 메시지를 보낼 수 있기 때문에 그 문제가 더욱 심각하다고 할 수 있다. 따라서 해커가 전송한 비 인가된 메시지를 걸러낼 수 있는 방안에 대한 요구가 증가하고 있다.

[0006] 최근 차량 내부 네트워크에서 비 인가된 메시지를 걸러내기 위해 다양한 연구가 진행되고 있다. 특히 블록체인을 이용한 동적 키 기반의 연구에서는 최근 주목을 받고 있는 블록체인 기술의 핵심 개념 중 분산 원장 개념과 데이터 연결성을 응용하여 송수신되는 메시지를 기반으로 각각의 ECU가 동적으로 키를 생성하고 이를 공유하기 위한 방안을 제시하였다. 해당 연구는 키의 노출이 없고 메시지 간의 연결성을 알지 못하면 키를 생성할 수 없어 인가된 메시지를 보낼 수 없다는 점에서 기존의 연구들보다 우수하다. 하지만 송수신되는 메시지를 기반으로 하기 때문에 이 과정에서 어떤 ECU가 하나의 메시지라도 수신에 실패 했을 경우 해당 ECU는 다른 키를 생성하게 되어 인증을 위한 프로세스 전체가 마비될 수 있다. 즉, 해당 연구가 제시하고 있는 메시지 인증 시스템은 신뢰성의 한계를 가지고 있다.

[0007] < CAN(Controller Area Network) 프로토콜 >

[0008] CAN 프로토콜은 bus 형태로 동작하는 차량 내부 네트워크에서 가장 많이 사용되고 있는 네트워크 프로토콜이다. CAN 프로토콜은 1980년대 자동차 부품회사인 bosch에 의해 처음 등장했으며, CAN-high, CAN-low 두 개의 구리선을 사용하는 bus 형태로 설계되어 있다. 특히 power-train과 같은 차량 엔진 제어 등의 부분에서 많이 응용되고 있는 차량 네트워크 프로토콜이다. 이러한 CAN 프로토콜에서는 통신을 위한 별도의 메시지 프레임이 존재하며 각각 식별자(identifier)를 통해 구분된다. 가장 큰 특징은 메시지 프레임 별로 우선순위(priority)가 존재한다. 이 우선순위에 따라 bus를 점유할 수 있는 권한이 메시지 별로 차등 부여된다. 도 1은 CAN의 데이터 프레임을 보여준다. CAN 메시지 프레임에서 64bit (8bytes)의 데이터 공간에는 임의의 정보를 담아 전송할 수 있으며 이 부분에 실질적인 차량 제어를 위한 메시지가 저장되게 된다.

[0009] 또한 CSMA/CA(carrier sense multiple access / collision avoidance) 방식이 내재되어 있어 우선순위에 따라 한 번에 하나의 메시지가 bus를 통해 전송됨을 보장한다. 데이터 프레임에서 볼 수 있듯이 CRC 또한 내재되어 있어 송수신 데이터의 무결성을 확인할 수 있다.

[0010] <분산 합의 프로토콜(Distributed Consensus) >

[0011] 분산 합의 프로토콜은 분산 컴퓨팅 분야에서 널리 알려진 분야이다. 분산 합의 프로토콜의 목적은 분산 환경에서 동작하는 시스템의 신뢰성(reliability)과 일치성(consistency)을 위하여 서로 다른 n개의 프로세서들이 하나의 값을 선택하기 위함이며 다음과 같은 속성들을 만족하는 프로토콜을 설계하는 것을 목표로 한다.

[0012] 1. 타당성(Validity): 만약 한 프로세서가 특정한 값을 결정한다면, 는 다른 프로세서에 의해 제안된 값이다.

[0013] 2. 동일성(Agreement): 서로 다른 적법한 두 개의 프로세서는 다른 값을 선택하지 않는다.

[0014] 3. 결정성(Termination): 모든 적법한 프로세서들은 최종적으로 합의를 위한 값을 결정한다.

[0015] 분산 합의 프로토콜은 그 적합성(correctness)를 증명하기 위해 위 3가지의 속성을 만족함을 보여야 하며 본 발명에서 제시하는 합의 프로토콜 또한 이론적 증명을 통하여 합의 프로토콜의 속성들을 만족하는지 검증한다.

선행기술문헌

특허문헌

[0016] (특허문헌 0001) 한국등록특허공보 제 10-0871857 호

발명의 내용

해결하려는 과제

[0017] 전술한 문제점을 해결하기 위한 본 발명은 블록체인 개념이 접목된 차량 내부 네트워크의 동적 키 생성 환경에서 실시간성을 유지하는 한편 일치된 암호화 키를 위한 새로운 합의 프로토콜을 제안하는 것을 목표로 한다. 이를 위해 본 발명에서는 아래와 같은 기여를 하고자 한다.

[0018] 1. 인증 시스템의 메시지 누락에 대한 대응을 위한 ECU들의 상태 및 전이조건을 나타내는 상태 다이어그램

(state transition diagram)을 정의한다.

[0019] 2. 메시지 수신 실패가 발생했을 시에만 합의를 수행하는 Lazy한 방식의 프로토콜 제시한다.

[0020] 3. 차량 내부 네트워크의 실시간성을 훼손하지 않고, ECU들 간의 일치된 암호화 키를 가지고 차량 내부 네트워크의 신뢰성을 위한 합의 방법 제시한다.

과제의 해결 수단

[0021] 전술한 기술적 과제를 달성하기 위한 본 발명의 제1 특징에 따른 분산 합의프로토콜 방법은, 차량 내부 네트워크의 블록체인 기반 동적 암호화키 생성 환경에서의 분산합의 프로토콜 방법에 관한 것으로서, 정상 상태, 에러 상태 및 합의 상태로 이루어지는 상태 전이 다이어그램을 구비하며, 상기 합의 상태는 에러 플래그 메시지, 합의 요청 메시지, 합의 메시지를 구비하고, 어떤 ECU가 에러 상태로 전이시 다른 ECU 들에게 에러 플래그 메시지 들 전송하고, 에러 플래그를 수신한 ECU들은 키생성 및 메시지 저장을 중지하고 에러 상태로 전이하며, 합의 요청 메시지는 합의를 시작하기 위한 메시지로, primary message의 수신을 실패한 ECU가 복구되어 이후 주기적으로 primary message에 수신 성공시 다른 ECU들에게 합의를 요청하기 위해 합의 요청 메시지를 전송하며, 합의 메시지는 합의 요청 메시지를 수신한 ECU들이 합의를 위하여 사용한다.

발명의 효과

[0022] 본 발명에서는 차량 내부 네트워크에서 블록체인 메커니즘을 기반으로 한 동적 키 생성 및 메시지 인증 시스템에서 신뢰성과 일치성 위한 합의 프로토콜을 제안하였다. 제시된 합의 프로토콜은 메시지 누락이 발생했을 때에만 동작하는 lazy한 방식의 프로토콜이며 이전 주행에서 저장되었던 EEPROM의 데이터와 3가지의 새로운 메시지 타입을 사용하여 동작한다. 이는 기존 연구들과 비교하여 사용되는 메시지의 타입을 줄여 추가적인 오버헤드를 감소시켰으며 CAN 프로토콜의 특성을 반영하여 차량 네트워크에서 효율적으로 운용될 수 있도록 설계하였다. 더불어 주요 ECU들의 상태를 정의하고 상태에 기반한 합의 프로토콜을 구성하였다.

[0023] 또한 합의 프로토콜이 만족해야 할 모든 속성들인 타당성, 일치성, 결정성 대해 각각 이론적 검증을 수행하였으며 실시간성을 알아보기 위한 방법으로 차량 네트워크와 유사한 환경을 구성하여 실험하였다. 실험의 결과 평균 소요시간은 11ms로 차량 네트워크에서 요구되는 네트워크 요구사항 또한 만족시킴을 보였다. 결론적으로 암호화 키를 이용하는 인증 시스템에서 모든 Critical ECU 들이 같은 암호화 키를 가짐을 보장함으로써 기존 연구에서 제시한 인증 시스템의 신뢰성을 강화하였다.

도면의 간단한 설명

[0024] 도 1은 CAN의 데이터 프레임을 보여준다.

도 2는 본 발명에 따른 시스템 모델의 구성도이다.

도 3은 실패 모델의 시나리오를 바탕으로 하여 작성한 도식이다.

도 4는 메시지 형식들을 도시한 것이다.

도 5는 정형적인 표현을 위한 기호를 정의한 도표이다.

도 6은 ECU들의 상태 전이 다이어그램으로 ECU 상태 간의 관계 및 전이 조건들을 보여준다.

도 7은 본 발명에 따른 제1 알고리즘을 도시한 것이다.

도 8은 본 발명에 따른 제2 알고리즘을 도시한 것이다.

도 9는 본 발명에 따른 제3 알고리즘을 도시한 것이다.

도 10은 롤백, 복제, 회귀방법 간의 비교 분석 결과를 도시한 도표이다.

도 11은 차량 내부 네트워크를 이용한 실험 구성도이다.

도 12는 시간 측정 구간을 도시한 그래프이다.

도 13은 실험 결과를 도시한 도표이다.

도 14는 ECU 종류에 따른 차량 내부 네트워크의 메시지 속도 요구사항을 보여주는 도표이다.

발명을 실시하기 위한 구체적인 내용

- [0025] 분산 시스템에서 신뢰성을 달성하고 네트워크 노드 간의 데이터 불일치성을 극복하기 위한 합의 프로토콜은 무수히 많은 연구들이 수행되었으며, 그 결과로 다양한 방식의 프로토콜들이 제시되었다. 특히 멀티 에이전트 시스템이나 스마트 그리드 등의 시스템 환경을 두루 고려한 합의 프로토콜들이 존재하고 있으나, CAN 프로토콜을 사용하는 네트워크 환경에서 연구된 합의 프로토콜은 많지 않다. 따라서 본 발명에서는 CAN 프로토콜을 사용하는 차량 내부 네트워크 뿐만 아니라 이와 유사한 실시간 분산 시스템 환경에서 응용될 수 있는 합의 프로토콜 연구에 초점을 맞추어 분석하였다. 더불어 전통적인 분산 합의 프로토콜들과의 비교를 통하여 차량 네트워크 환경과 같은 실시간 시스템에서의 통신을 위한 효율성을 제고하였다.
- [0026] < CAN 통신을 사용하는 네트워크 환경에서의 합의 프로토콜 >
- [0027] Lima는 CAN 통신을 기반으로 하는 실시간 시스템에서의 합의 프로토콜을 제시하였다. 해당 연구에서는 CAN 네트워크 환경을 구성하는 프로세서들이 수신한 메시지의 개수와 수신한 순서를 유지하며 동작하는 환경을 가정하였다. 이러한 가정 하에 CAN 프로토콜을 사용하는 네트워크에서 발생 할 수 있는 실패 시나리오들을 정의하였다. 이 연구에서 정의하는 실패 시나리오는 메시지의 수신 실패, 중복 수신 등의 상황을 서술하고 있고 실패 시나리오로 인한 네트워크 불일치를 해결할 수 있는 합의 프로토콜을 제시하였다. Lima의 연구에서 제시하고 있는 합의 프로토콜의 내용은 다음과 같다. 합의를 위한 프로토콜은 임의로 설정된 라운드 별로 진행되며 메시지 불일치가 일어났을 시 모든 프로세서가 하나의 올바른 메시지를 선택하기 위해 자신이 가지고 있는 메시지의 수와 순서로 추정 값을 생성한다. 추정 값이 생성되면 이 값을 다른 프로세서들에게 전송하고 이를 수신한 다른 프로세서들은 자신이 생성한 추정 값과 비교하여 올바른 메시지 하나를 채택하게 된다. 그러나 이와 같은 프로토콜은 통신으로 인한 메시지의 오버헤드와 추정 값을 구하기 위한 연산 시간이 추가적으로 소요된다는 한계점을 가지고 있다.
- [0028] < 메시지 내역을 이용하는 합의 프로토콜 >
- [0029] Azim은 실시간 통신 환경에서 통신 내역과 합의를 기반으로 시스템의 일치성을 유지하기 위한 방법을 제안하였다. Azim의 연구에서는 메시지의 내역을 추적하기 위한 별도의 buffer table을 구성하였다. 이 buffer table은 시스템에서 송수신되는 메시지의 실제 값을 저장하고 있으며, 어떤 변수들을 유지할 것인가는 설계 단계에서 엔지니어가 임의로 정의하여 유지할 수 있도록 하였다. 노드가 메시지를 수신 할 때마다 이 buffer table에 저장되어 있는 값과 수신한 값들 간의 일치성을 검증한다. 최초로 통신이 시작되면 buffer table에서 유지하는 변수들에게 초기값을 할당한다. 이후 메시지를 수신하면 메시지에 담긴 변수와 이에 해당하는 값들로 분해하여 자신이 가지고 있는 buffer table과의 일치성을 확인한다. 만약 특정 변수가 다른 변수에 종속적일 경우 buffer table을 사용하여 종속여부를 확인한다. 이 과정에서 불일치가 발생 했을 경우에 다른 노드들에게 에러 플래그를 송신하게 되는 구조이다. 해당 연구는 시스템의 일치성 유지를 위하여 메시지의 내역을 이용한다는 점에서 본 연구와 유사하지만 CAN 프로토콜의 특성을 반영하지 못하고 있다. 또한 메시지 수신 시점마다 consistency를 위한 추가적인 연산이 필요하고 메시지 값들의 추적을 위한 buffer table을 메시지 저장과는 별도로 유지해야 하는 한계가 존재한다.
- [0030] < 실시간 분산 환경에서의 합의 프로토콜 >
- [0031] Hermant는 이더넷 기반의 분산환경 시스템에서 장애를 탐지하는 방법과 발생한 장애 극복을 위한 합의 프로토콜을 제시하였다. 특히 실시간성을 요구하는 시스템을 위하여 합의에 필요한 시간을 최소 시간으로 보장하는 방식을 제안하였다. 이를 위하여 Hermant의 연구에서는 SRC(Single Round Consensus)라 정의된 Distributed Algorithm을 기반으로 합의 프로토콜을 구성하였다. 이 방법은 시스템을 구성하고 있는 프로세서들이 합의를 이루고자 하는 값을 제안하는 메시지를 전파한다. 네트워크에 전파된 이 메시지들을 각각의 프로세서들이 특정 조건이 만족될 때까지 축적하고 합의를 이루기 위한 값을 도출하기 위해 대기한다. 이러한 과정을 진행 하는 도중에 어떤 하나의 프로세서가 win이라고 정의된 프로세서로 선출이 되게 되는데 이 win프로세서가 전파한 값이 최종적으로 합의를 위한 값으로 정해지게 된다. 그러나 해당 연구는 이더넷 기반의 환경에서 동작하는 프로토콜로 차량 내부 네트워크와 같이 bus 형태를 가지는 환경과 CAN 프로토콜에 대한 특성을 고려하지 않는 문제점을 지니고 있다.
- [0032] < 이더넷 등의 네트워크 통신을 기반으로 하는 분산 환경에서의 합의 프로토콜 >
- [0033] 분산 환경에서 발생하는 시스템의 불일치와 안정성을 해결하기 위하여 혹은byzantine failure라고 알려진 문제를 해결하기 위한 방법으로 합의 프로토콜은 많은 연구가 진행되어 왔다. 대표적으로 lamport가 제안한 paxos

시리즈의 프로토콜이 존재한다. 이 프로토콜은 서로 다른 4개의 메시지 종류를 사용하는 메시지 통신 기반의 합의 프로토콜이며 2PC(Two-Phase Commit)과 3PC(Three-Phase Commit)과 유사하다. Paxos는 크게 준비, 승인의 두 단계로 진행되며 각 진행 단계마다 통신을 위해 사용되는 메시지의 종류가 다르다. 준비 단계에서는 합의를 이루기 위한 값을 임의의 노드가 제안을 하게 되며 이 제안이 합당하다고 판단한 노드들은 이에 대한 응답을 다시 네트워크에 전송하게 된다. 이후 승인 단계에서 최종적으로 합의를 이룬 값을 지정하게 된다. 그러나 paxos 시리즈의 프로토콜은 앞서 서술한 바와 같이 합의를 이루기 위한 통신으로 인한 네트워크에 오버헤드가 크다. 차량 내부네트워크와 같이 bus 형태를 가지는 네트워크 환경에서 오직 합의만을 위한 메시지들이 통신 트래픽의 대부분을 가져갈 경우 실제 차량 제어나 운행에 필요한 메시지들이 bus에 전송되지 못하는 문제점이 발생 할 수 있다.

[0034] 또한 위에서 서술한 Paxos를 기반으로 하여 서버-클라이언트 구조에서 동작하는 합의 프로토콜로 Ongaro가 고안한 Raft 합의 알고리즘이 있다. Raft의 합의 방식은 leader로 정해진 서버가 클라이언트 간의 합의를 이끌어내는 leader-based 방식의 프로토콜이다. 해당 연구는 paxos와 마찬가지로 클라이언트들이 leader를 선출하기 위한 투표 메시지를 포함하여 4가지의 각각 다른 메시지 타입을 정의하여 합의 프로토콜에 사용한다. 투표를 거쳐 leader로 지정된 서버가 중심이 되어 합의를 진행함으로써 paxos보다 간략한 프로세스를 가지고 있다. 그러나 raft consensus는 차량 내부 네트워크와 같은 bus 형태의 네트워크 환경과 CAN 프로토콜의 특성을 고려하지 않았고 서버-클라이언트 구조와 같은 large-scale 컴퓨팅 환경에 적합하도록 고안된 합의 프로토콜이다. 또한 leader 선출 시 최대 150ms~300ms까지의 worst-case가 존재하여 차량 내부 망에서 사용하기에는 한계가 존재한다.

[0035] 이하, 본 발명에 따른 차량 내부 네트워크 환경에서 키 불일치 문제를 해결하기 위한 합의 프로토콜에 대해 상세하게 설명한다.

[0036] 본 발명에서는 차량 내부 네트워크를 위한 시스템 모델을 구성하기 위해 차량 내부 네트워크에서 가장 널리 사용되는 통신 프로토콜인 CAN 프로토콜(Controller Area Network)[8]을 사용한다. 일반적인 CAN 프로토콜 상에서는 ECU들이 bus에 연결된 CAN 트랜시버를 사용하여 통신을 하는 구조이다. CAN 트랜시버는 bus로부터 데이터를 송수신하며 수신 데이터는 ECU에 탑재된 MCU(Micro Controller Unit)에 전달되고 MCU가 실제 차량 제어에 필요한 동작을 수행하는 구조를 가지고 있다.

[0037] CAN 통신 프로토콜을 응용하여 본 발명에서 제안하는 합의 프로토콜이 동작할 수 있는 시스템 모델을 도 2와 같이 구성한다. 도 2는 본 발명에 따른 시스템 모델의 구성도이다. 도 2를 참조하면, 이 시스템 모델은 차량에 탑재되어 있는 ECU들 중에서 엔지니어가 보안이 중요하다고 판단되는 ECU들을 선정하고, 이들 간의 통신에 사용되는 중요 메시지들을 선정하여 각각 Critical ECU, Primary message로 정의한다. Critical ECU로 선정된 ECU들은 하나의 bus를 통해 통신하며, 메시지 인증 및 키 생성을 위한 보안 모듈인 KeyGen Module 을 CAN 트랜시버와 MCU 사이에 추가한다. KeyGen Module 은 블록체인 기반의 동적 키 생성 방법이 탑재되어 있고, 이 모듈은 Primary message, 인증 키의 저장을 위한 EEPROM 및 Register와 연결되어 있다. EEPROM에는 이전 주행 종료 시 가장 마지막으로 생성된 암호화 키와 메시지가 저장되어 있다. Register는 차량의 주행 중 사용하고 있는 인증 키를 비롯하여 키 생성에 기반이 되는 Primary message 들을 저장하고 있다. 또한 차량의 운행 종료 시에는 모든 Critical ECU 들은 각각 자신이 종료시점에 가지고 있던 암호화 키와 메시지들의 동일 여부를 확인하고 EEPROM 에 저장한 후 종료하게 된다.

[0038] 위와 같이 구성된 시스템에서 차량 내부 네트워크에서 동작하는 CAN 프로토콜의 특성을 반영하고자 다음과 같은 시스템 모델을 가정한다.

- [0039] 1.Primary message 는 주기적으로 송신됨을 가정한다.
- [0040] 2. CAN 통신은 fail-safe 모드가 존재하여 수신에 실패하였을 경우, 가장 최근의 수신 값을 다시 처리하도록 설계되어 있다.
- [0041] 3. critical ECU 는 Primary message 의 수신 여부를 주기적으로 확인한다.
- [0042] 4. EEPROM 에 저장되어 있는 데이터는 위·변조 되지 않는다.
- [0043] 5. KeyGen module 은 오직 Primary message 만을 처리하며 그 외의 메시지는 별도의 과정 없이 MCU로 통과 시킨다.
- [0044] 다음, 실패 모델(Failure model)에 대하여 설명한다.

- [0045] 이하, 실패 모델을 구성하여 합의 프로토콜이 동작해야 하는 상황에 대해 정의한다. 시스템의 실패는 메시지의 누락으로 인한 잘못된 키 생성으로 정의한다. 메시지 누락은 Critical ECU 들 중 어느 한 ECU가 주기적으로 수신 되었어야 할 Primary ECU 를 받지 못했을 때 발생하며 이 경우 송수신되는 메시지를 기반으로 동작하는 KeyGen module 에게 치명적인 악영향을 미친다. 왜냐하면 KeyGen module 은 primary message들을 Register에 저장하고 이를 사용하여 메시지 인증을 위한 새로운 암호화 키를 생성하기 때문에 한 ECU가 Primary message 수신에 실패하면 해당 ECU만 전혀 다른 암호화 키를 생성하는 문제가 발생한다. 서로 다른 키를 생성하게 되면 이후 통신에 사용되는 메시지들의 인증에 실패하게 될 것이다. 도 3은 실패 모델의 시나리오를 바탕으로 하여 작성한 도식이다.
- [0046] 도 3을 참조하면, 3번 Critical ECU 는 가장 최근의 Primary message 인 n번째 primary message 수신에 실패하여 다른 Critical ECU 들과 메시지 내역이 불일치함을 알 수 있다. 메시지 내역이 일치해야 동일한 키를 생성할 수 있는 시스템 모델의 특성 상 이처럼 메시지 내역이 Critical ECU 간 불일치 하는 경우 다른 암호화 키를 생성하는 문제가 발생한다. 이하, 이와 같은 키 불일치 문제를 해결하기 위해 본 발명에서 구성한 상태 전이 다이어그램과 합의 프로토콜에 대해 상세히 기술하겠다.
- [0047] < 합의 프로토콜의 상태 전이 다이어그램 >
- [0048] 이하, 전술한 실패 모델의 문제점을 해결하기 위하여 새로운 합의 프로토콜을 정의한다. 우선 합의 프로토콜의 효율적인 동작을 위하여 Critical ECU 의 상태를 아래와 같이 정의한다.
- [0049] 1. 정상 상태(Normal state)
- [0050] 2. 에러 상태(Error state)
- [0051] 3. 합의 상태(Consensus state)
- [0052] Normal state는 ECU들이 정상적으로 primary message 를 주고 받는 상태이다. 이 상태에서는 키 생성을 위한 메시지 저장 및 키 생성 프로세스가 정상적으로 이루어지고 있음을 나타낸다. error state는 primary message 수신에 실패 시 normal state로부터 전이되며 메시지 저장과 동적 키 생성이 정지되어 있는 상태이다. 그러나 키 생성을 위한 프로세스만 정지되었을 뿐 error state에서도 지속적인 메시지 수신 및 처리는 가능하다.
- [0053] Consensus state는 암호화 키에 대해 합의를 하기 위한 상태이며 다른 ECU들과의 메시지 송수신을 통하여 합의 프로세스를 진행한다. 상태 전이 시 메시지 송수신을 통하여 critical ECU들 간의 상태가 동일하게 유지 될 수 있도록 하기 위해 상태 전이 메시지 형식을 정의하였다. 메시지 형식은 CAN 통신에서 사용되는 데이터 프레임 중 8 바이트 크기의 메시지 필드를 사용하여 구성하였다. 메시지의 종류는 1) 한 ECU가 error state로 전이 시 이를 다른 ECU들에게 알리기 위한 메시지와 2) 합의를 요청하는 메시지 및 3) 합의 프로세스 시 사용되는 메시지까지 3가지의 메시지 형식을 정의하였다. 메시지의 타입은 CAN 메시지의 데이터 필드 가장 앞쪽의 4byte의 문자로서 구분된다.
- [0054] 도 4는 메시지 형식들을 도시한 것이다. 도 4의 (a)는 error flag 메시지로 어떤 ECU가 error state로 전이 시 다른 ECU들에게 전송하게 되는 메시지이다. error flag 메시지는 EEPROM 에 저장되어 있는 임의의 값을 이전 키로 암호화 하여 나온 값을 전송한다. error flag를 수신한 ECU들도 마찬가지로 키 생성 및 메시지 저장을 중지하고 error state로 전이한다. (b)는 합의를 시작하기 위한 메시지로 primary message 의 수신을 실패한 ECU가 복구되어 이후 주기적으로 primary message 에 수신 성공 시 다른 ECU들에게 합의를 요청하는 메시지이다. 이 메시지 또한 EEPROM 에 저장되어 있는 임의의 값을 이전 키로 암호화한 값을 담아 전송한다. Request consensus message를 송신한 ECU는 consensus state로 전이하고 이를 수신한 ECU들도 consensus state로 전이하여 합의 프로세스를 시작한다. (c)는 Request consensus message를 수신한 ECU들이 합의를 위하여 사용하는 메시지이다. 각각의 메시지 형식에서 사용되는 EEPROM 의 데이터(EEPROM[θ])는 엔지니어가 시스템의 설계 단계에서 취사선택할 수 있다. 더불어 제시하고자 하는 합의 프로토콜을 설명하기 ECU의 상태 및 합의 프로토콜의 좀 더 정형화된 표현을 위하여 도 5와 같이 기호를 정의한다. 도 5는 정형적인 표현을 위한 기호를 정의한 도표이다.
- [0055] 차량이 운행을 시작하면 Critical ECU들은 최초 상태인 Normal state로 진입하며 메시지의 수신여부, 수신한 메시지 타입에 따라 상태가 전이된다. 도 6은 ECU들의 상태 전이 다이어그램으로 ECU 상태 간의 관계 및 전이 조건들을 보여준다.
- [0056] < 키 일치를 위한 합의 프로토콜 >
- [0057] 이하, 앞서 정의한 ECU의 상태와 메시지 프레임에 바탕으로 하여 합의 프로토콜에 대해 설명한다. 합의 프로토

콜이 동작하기 위해서는 먼저 ECU들의 상태 정보와 암호화 키의 초기화 과정이 선행되어야 한다.

- [0058] 도 7은 본 발명에 따른 제1 알고리즘을 도시한 것이다.
- [0059] 도 7을 참조하면, Algorithm 1은 각각의 ECU들이 시스템 시작 시 수행하는 초기화 프로토콜을 보여준다. 최초 실행 시 가장 먼저 ECU의 상태를 normal state로 설정한다. 이후 EEPROM 으로부터 이전 실행의 종료 직전 저장 하였던 KEYprev 와 KEYcurrent 값을 register 로 읽어온다. 이후 register 에 마찬가지로 이전 실행의 메시지를 옮겨온다. 만약 올바른 ECU들이 초기화 프로토콜을 정상적으로 실행하였다면, 결과적으로는 각각의 register 에는 모두 동일한 키 값과 메시지 내역을 가진 상태가 된다.
- [0060] ECU가 이와 같은 초기화 프로토콜을 정상적으로 수행하고 나면, Algorithm 2와 같은 상태 전이 프로토콜에 의해 동작을 이어간다. 도 8은 본 발명에 따른 제2 알고리즘을 도시한 것이다.
- [0061] 이 프로토콜을 크게 ECU의 primary message 수신 여부와 합의 성공여부에 따라 상태가 변경되며, 자세한 동작을 설명하기 전에 메시지 생성 및 통신을 위한 세 가지의 함수를 정의한다.
- [0062] 1. CreateMsg(type, key, Θ): 매개변수로 입력한 type의 메시지 프레임 생성하며 key를 이용하여 data를 암호화한 후 메시지 프레임에 적재한다. Θ 는 EEPROM 에 저장되어있는 임의의 데이터로 엔지니어가 취사선택 후 적재할 수 있다.
- [0063] 2. Send(M) : 매개변수로 넘어온 메시지 M 를 CAN bus에 전송하는 함수이다.
- [0064] 3. Dec(M, key) : 메시지에서 타입을 정의하는 부분을 제거하고 매개변수로 주어진 key를 사용하여 데이터를 복호화 하는 함수이다.
- [0065] 우선 ECU는 주기적으로 메시지의 수신을 확인한다. 정상적인 수신 확인 되었을 경우, Normal state 상에서는 메시지 타입을 확인한다. 메시지 타입이 M_{PM} 일 경우(메시지 타입이 M_{EF} 이 아닐 경우) 상태 전이 프로토콜은 수신한 메시지를 MCU에게 전송한다. 만일 Normal state에서 수신한 메시지의 타입이 M_{EF} 일 경우는 Dec()함수로 메시지 복호화를 수행한다. 복호화한 메시지의 결과가 자신의 EEPROM 에 저장되어 있는 데이터와 동일하다면, 현재 동작하고 있는 적법한 ECU들 중 어느 하나가 에러를 발생 시킨 것으로 간주되어 register 로 메시지를 저장하지 않고 키 생성 과정을 중단하고 ECU의 상태를 consensus state로 전이한다.
- [0066] 한편, 메시지 수신에 실패하여 error state로 전이된 ECU가 다시금 정상적인 M_{PM} 수신에 성공하면 CreateMsg()함수와 이전 인증 키 및 자신의 EEPROM 에 저장되어 있는 데이터를 사용하여 합의 요청을 위한 메시지 형식(M_{RC})을 생성한 후 bus에 송신한다. 이후 송신을 완료하면 Consensus state로 전이하여 키 불일치에 대한 합의 과정을 시작한다. consensus state로 전이된 ECU들은 consensus protocol을 실행하게 되며 동작 방식은 Algorithm 3과 같다.
- [0067] 도 9는 본 발명에 따른 제3 알고리즘을 도시한 것이다.
- [0068] 프로토콜의 시작과 동시에 이전 키 값인 KEY_prev와 EEPROM[Θ], createMsg()함수를 이용해서 consensus message를 전송하고 수신한 consensus message의 수를 저장하는 변수인 Num_{CM}의 값을 1로 저장한다. 이후 Δ 만큼의 time out을 지정한다. Δ 는 임의의 값이며 최소한 Critical ECU 들 간의 1회 메시지 송수신이 보장되는 시간이 되어야 한다. 이후 ECU는 설정된 Δ 의 시간 동안 다른 ECU들의 consensus message를 처리한다. 메시지의 타입이 M_{CM} 일 경우 Dec()함수를 사용하여 이를 복호화하고 이를 EEPROM[2]의 값과 비교한다. 동일한 결과를 얻을 시 Num_{CM}을 1씩 증가시킨다. 이 값이 Critical ECU 의 수와 같아지면 모든 Critical ECU 들이 동의한 것으로 간주한다. 만일 Δ 이후까지 line 9의 조건을 만족시키지 못하면 자동적으로 consensus protocol을 종료하고 다시 error state로 전이한다. Line 9의 조건이 충족되면 현재 register에 저장되어 있는 값들을 삭제하고 EEPROM 에 저장되어 있던 데이터를 register로 옮긴다. 이 때, EEPROM 에 저장되어 있던 초기 키 값을 현재 키 값으로 할당한다. 이후 normal state로 전이함으로써 프로토콜 수행이 완료된다. 결과적으로 모든 Critical ECU 들이 EEPROM 에 저장되어 있던 이전 주행 메시지 및 암호화 키를 사용하게 된다. 이러한 consensus protocol의 과정은 메시지의 누락으로 인해 발생한 키 생성 프로세스의 에러를 가장 초기 단계로 회귀(recurrence)한다고 볼 수 있다. 에러를 복구하는 방식은 제안한 방법인 회귀 포함하여 크게 1) 롤백(rollback)과 2) 복제(replication)이 추가적으로 고려될 수 있다. 롤백은 에러가 발생하기 이전 상태로 프로세스를 되돌리는 방식이고 replication은 별도의 백업 데이터를 유지하는 방식이다. 그러나 1)과 2)의 방법은 본 발명의 시스템 모델에 적용하기에는 적

합하지 않다. 이를 각 방식들에서 요구되는 시간과 메모리 크기 간의 정량적 비교 및 증명을 위해 아래와 같은 기호를 정의하였다.

[0069] 1. 메시지 전송에 걸리는 지연 시간: N

[0070] 2. 조건문 실행의 걸리는 시간: T_{if}

[0071] 3. 키 생성에 걸리는 시간: T_{key}

[0072] 4. EEPROM으로부터 데이터 전송 시간: t

[0073] 5. Register 크기: s

[0074] 각각의 방법들을 적용 시 합의에 소요되는 시간 및 메모리 크기를 이론적으로 도출한 결과는 도 10과 같다. 도 10은 롤백, 복제, 회귀방법 간의 비교 분석 결과를 도시한 도표이다.

[0075] 도 10의 도표에서 이론적으로 도출한 소요 시간을 비교를 해보자면, 롤백 방식은 총 $3N + T_{if} + T_{key}$ 의 시간이 걸리게 된다. 이에 반해 복제나 회귀의 경우를 보면 $2N + t$ 의 시간이 걸림을 알 수 있다. 여기서 T_{key} 의 값은 t 값보다 명백히 큼을 알 수 있다. 왜냐하면 키 생성에는 별도의 연산이 필요하고 이에 걸리는 시간은 평균 17ms가 걸리는데 반해 t 는 추가적인 연산 과정 없이 단순히 데이터를 EEPROM 에서 읽어오는 시간이기 때문이다. 따라서 시간적으로 비교했을 시 롤백은 적합하지 않은 방법임을 알 수 있다. 롤백을 제외한 복제나 회귀는 동일한 시간이 걸림을 알 수 있으나 복제의 경우 레지스터의 크기가 $s + a$ 로 a 는 복제 데이터를 저장하는데 필요한 추가적인 공간이 요구됨을 알 수 있다. 따라서 이 세 가지의 방법을 미루어봤을 때 회귀 방법이 소요 시간 및 레지스터 사이즈로 볼 때 가장 효율적임을 알 수 있다.

[0076] 이하, 본 발명에 따른 합의 프로토콜의 이론적 검증과 더불어 차량 내부 네트워크와 유사한 환경을 구성하여 실시간성을 훼손하는지를 알아보기 위한 합의 소요 시간을 실험적으로 측정한다. 이를 위해 임베디드 소프트웨어 실험에 널리 사용되는 아두이노와 CAN bus 네트워크의 구축을 위한 아두이노 용 CAN shield를 사용하였다.

[0077] 실험 환경 구성 및 측정 방법은 아래와 같다. 도 11은 차량 내부 네트워크를 이용한 실험 구성도이다. 도 11을 참조하면, Bus 형태를 가지는 차량 내부 네트워크 구성을 위해 아두이노-CAN shield의 쌍을 하나의 ecu처럼 구성한 후 이들을 bus 방식으로 통신하도록 설계하였다. 또한 ecu를 ECUsend와 ECUreceiver 두 개의 그룹으로 구성하였다. ECUsend는 에러 플래그 후 합의 요청 메시지가 수신 되었을 때만 실행되는 합의 프로토콜의 특성 상, 실험을 위한 강제 에러 플래그 및 합의 요청 메시지를 전송하는 ECU이다. ECUreceiver는 에러 플래그, 합의 요청 메시지에 따라 합의 프로토콜을 수행하는 ECU이다.

[0078] 합의에 소요되는 시간을 알아보기 위해 ECUsend가 consensus state로 전이된 시점을 시작으로 하여 다시 normal state로 전이되는 시점을 종료 시간으로 정의하여 측정하였다. 도 12는 시간 측정 구간을 도시한 그래프이다.

[0079] 실험 결과 및 분석은 아래와 같다. 위와 같은 실험 환경에서 δ 구간의 소요 시간을 반복하여 측정한 결과는 도 13과 같다. 도 13은 실험 결과를 도시한 도표이다. 약 19000회의 시도 결과 δ 는 평균 11.62ms의 시간이 소요됨을 측정할 수 있었다. 측정된 시간이 차량 내부 네트워크의 실시간성을 훼손하지 않음을 알아보도록 하겠다. 도 14는 ECU 종류에 따른 차량 내부 네트워크의 메시지 속도 요구사항을 보여주는 도표이다.

[0080] 시스템 모델에서 서술한 바와 같이 Critical ECU 는 엔지니어에 의해 설정되며 주로 차량 제어에 핵심적인 역할을 하는 ECU들이 포함될 것이다. 대표적으로 차량 제어의 핵심적인 역할을 하는 power-train ecu는 최소 10ms에서 최대 10s의 요구사항을 가진다. 합의 프로토콜의 평균 소요시간과 비교해보면 11ms의 시간이 걸리므로 통상 1개~2개 정도의 메시지를 합의 과정으로 인하여 처리하지 않게 된다. 그러나 이는 차량 내부 네트워크에서 충분히 감안할 만한 수준이다. 우선 CAN 프로토콜은 특성 상 fail-safe mode를 지원하고 있어 1~2개의 제어 메시지 생략을 이상 없이 견딜 수 있다. 또한 CAN 네트워크에서 메시지 누락이 발생할 확률은 시간 당 $3.98 \times 10^{-9} \sim 2.94 \times 10^{-6}$ 으로 그 확률이 매우 낮음을 알 수 있다. 즉, 본 발명에서 제시하는 합의 프로토콜은 11ms의 시간이 소요되지만 동작할 확률이 매우 작다. 이는 차량 네트워크의 요구사항 및 메시지 수신 실패 확률로 미루어 볼 때 실시간성을 훼손하지 않는다고 할 수 있다.

[0081] 본 발명에서는 차량 내부 네트워크에서 블록체인 메커니즘을 기반으로 한 동적 키 생성 및 메시지 인증 시스템에서 신뢰성과 일치성 위한 합의 프로토콜을 제안하였다. 제시된 합의 프로토콜은 메시지 누락이 발생했을 때에만 동작하는 lazy한 방식의 프로토콜이며 이전 주행에서 저장되었던 EEPROM의 데이터와 3가지의 새로운 메시지

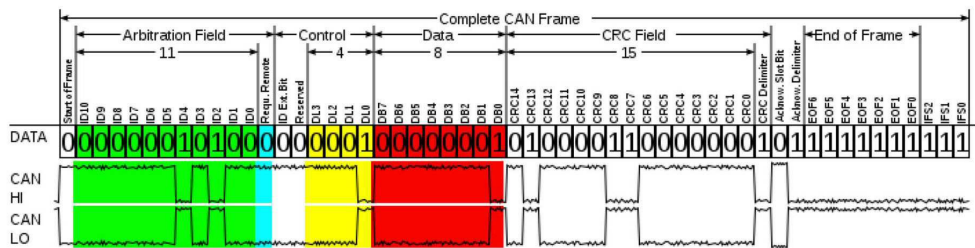
타입을 사용하여 동작한다. 이는 기존 연구들과 비교하여 사용되는 메시지의 타입을 줄여 추가적인 오버헤드를 감소시켰으며 CAN 프로토콜의 특성을 반영하여 차량 네트워크에서 효율적으로 운용될 수 있도록 설계하였다. 더불어 주요 ECU들의 상태를 정의하고 상태에 기반한 합의 프로토콜을 구성하였다.

[0082] 또한 합의 프로토콜이 만족해야 할 모든 속성들인 타당성, 일치성, 결정성 대해 각각 이론적 검증을 수행하였으며 실시간성을 알아보기 위한 방법으로 차량 네트워크와 유사한 환경을 구성하여 실험하였다. 실험의 결과 평균 소요시간은 11ms로 차량 네트워크에서 요구되는 네트워크 요구사항 또한 만족시킴을 보였다. 결론적으로 암호화 키를 이용하는 인증 시스템에서 모든 Critical ECU 들이 같은 암호화 키를 가짐을 보장함으로써 기존 연구에서 제시한 인증 시스템의 신뢰성을 강화하였다.

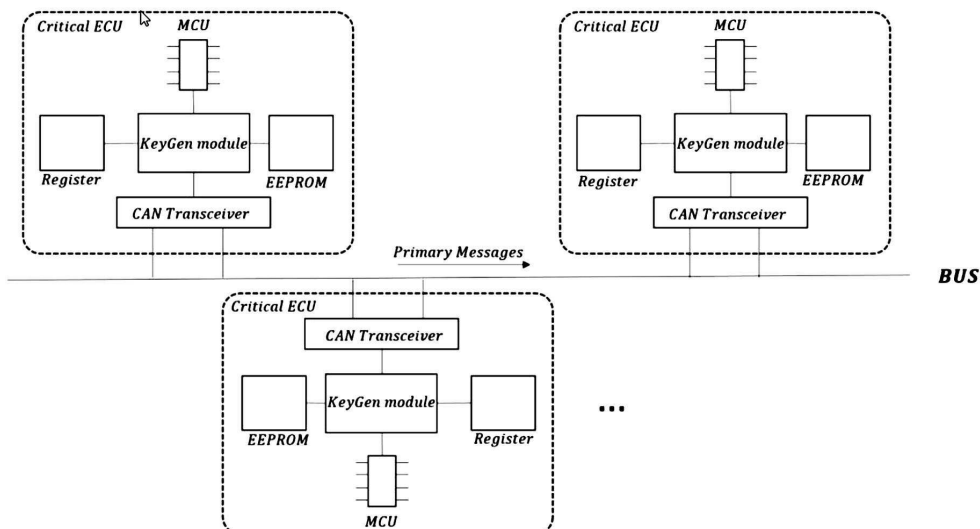
[0083] 이상에서 본 발명에 대하여 그 바람직한 실시예를 중심으로 설명하였으나, 이는 단지 예시일 뿐 본 발명을 한정하는 것이 아니며, 본 발명이 속하는 분야의 통상의 지식을 가진 자라면 본 발명의 본질적인 특성을 벗어나지 않는 범위에서 이상에 예시되지 않은 여러 가지의 변형과 응용이 가능함을 알 수 있을 것이다. 그리고, 이러한 변형과 응용에 관계된 차이점들은 첨부된 청구 범위에서 규정하는 본 발명의 범위에 포함되는 것으로 해석되어야 할 것이다.

도면

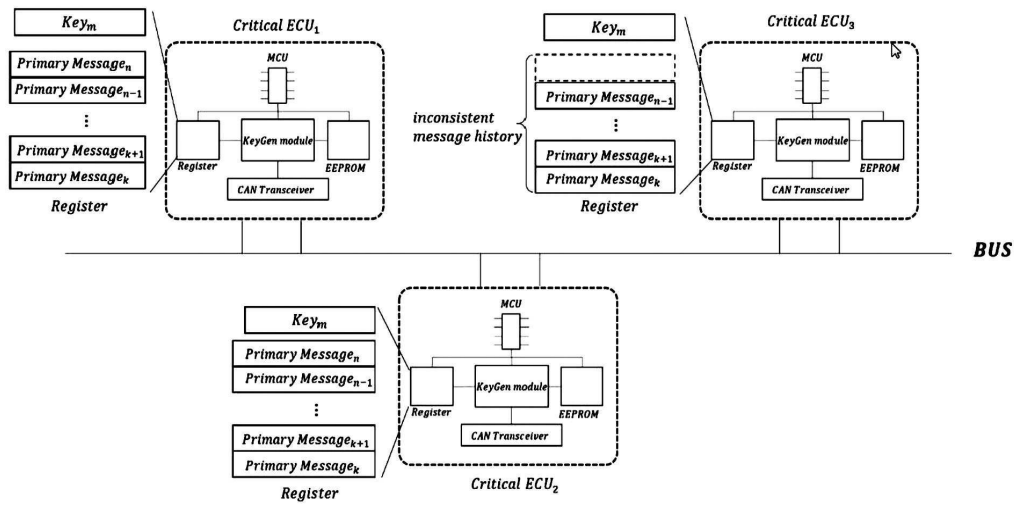
도면1



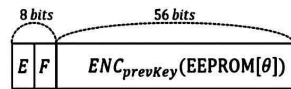
도면2



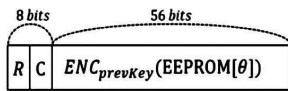
도면3



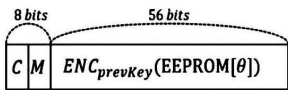
도면4



(a) Error flag message



(b) Request consensus message

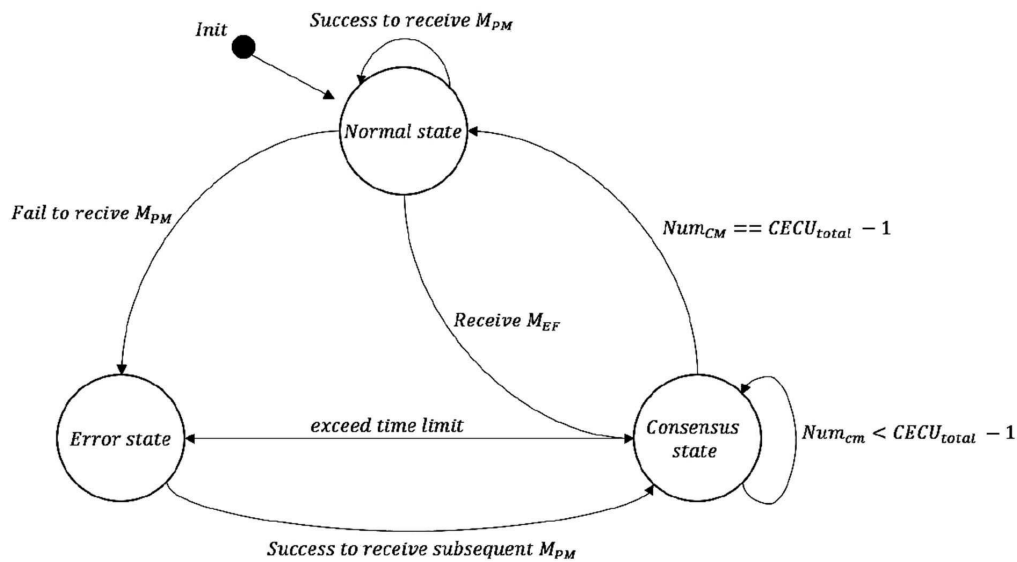


(c) Consensus message

도면5

기 호	정 의
M_{PM}	주요 메시지(primary message)
M_{EF}	에러 플래그 메시지(error flag message)
M_{RC}	합의 요청 메시지(request consensus message)
M_{CM}	합의 메시지(consensus message)
$CECU_{total}$	critical ECU의 수
NUM_{CM}	수신한 합의 메시지의 수

도면6



도면7

Algorithm 1: Initializing protocol

Result: EEPROM data is transferred to Register

- 1 $ECU_STATE \leftarrow NormalState$
- 2 $KEY_{prev} \leftarrow \text{read from } EEPROM$
- 3 $KEY_{current} \leftarrow \text{read from } EEPROM$
- 4 $Register \leftarrow \text{read from } EEPROM$

도면8

Algorithm 2: State transition protocol

```

1  if receive success then
2      if ECU_STATE == Normal State then
3          if MEF is received then
4              if Dec(MEF, KEYprev) == EEPROM[θ] then
5                  lock Register           ▷ Primary message is not stored.
6                  ECU_STATE ← ConsensusState
7              end
8          end
9          else
10             Bypass
11             Continue to protocol
12         end
13     end
14     else if ECU_STATE == Error State then
15         if MPM is received then
16             CreateMsg(MRC, KEYprev, EEPROM[θ])
17             Send(MRC)
18             ECU_STATE ← ConsensusState
19             start consensus protocol
20         end
21     end
22     else if ECU_STATE == Consensus State then
23         if MRC is received then
24             start consensus protocol
25         end
26     end
27 end
28 else
29     lock Register
30     CreateMsg(MEF, KEYprev, EEPROM[θ])
31     Send(MEF)
32     ECU_STATE ← ErrorState
33 end

```

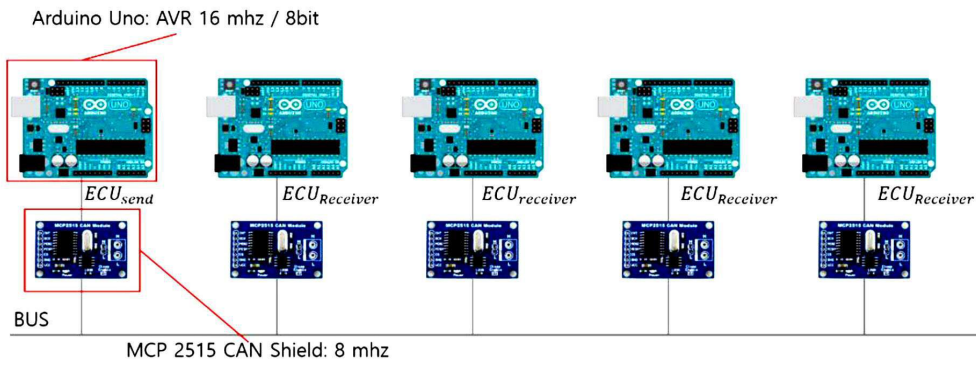
도면9

Algorithm 3: Consensus protocol	
1	$CreateMsg(M_{CM}, KEY_{prev}, EEPROM[\theta])$
2	$Send(M_{CM})$
3	$Num_{CM} \leftarrow 1$
4	$setTimeout(\Delta)$
5	if M_{CM} is received then
6	if $Dec(M_{CM}, KEY_{prev}) == EEPROM[\theta]$ then
7	$Num_{CM} \leftarrow Num_{CM} + 1$
8	end
9	if $Num_{CM} == CECU_{total}$ then
10	$KEY_{prev} \leftarrow \text{read from } EEPROM$
11	$KEY_{current} \leftarrow \text{read from } EEPROM$
12	$Register \leftarrow \text{read from } EEPROM$
13	release $Register$
14	$ECU_STATE \leftarrow NormalState$
15	end
16	end

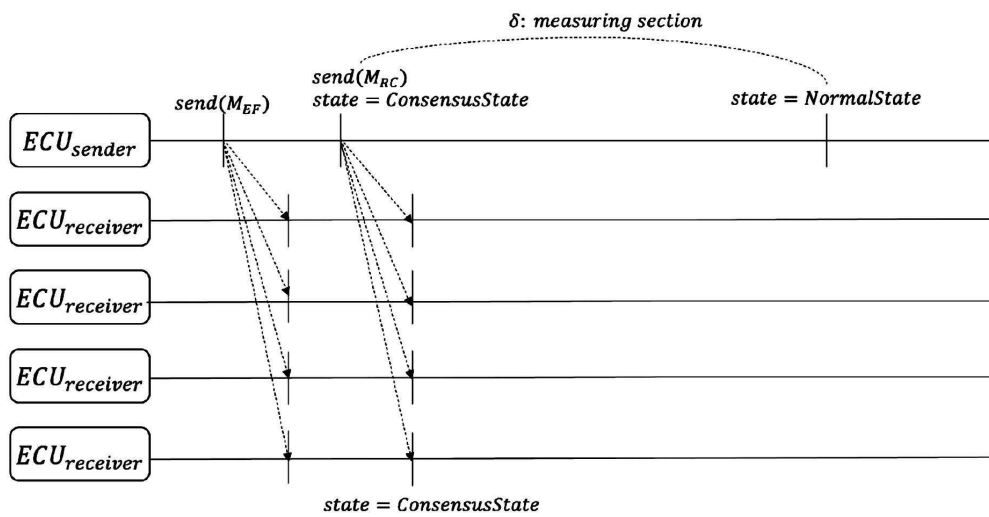
도면10

방법	분석 내용	분석 결과
롤백(rollback)	에러 플래그 메시지 전송: N 롤백 지점 메시지 전송: N 롤백 지점 확인 메시지 전송: N 롤백 지점 비교 시간: T_{if} 키 재생성 시간: T_{key}	전체 소요 시간: $3N + T_{if} + T_{key}$ 레지스터 크기: s
복제(replication)	에러 플래그 메시지 전송: N 합의 메시지 전송: N EEPROM으로부터 데이터 전송 시간: t	전체 소요 시간: $2N + t$ 레지스터 크기: $s + \alpha$
회귀(recurrence)	에러 플래그 메시지 전송: N 합의 메시지 전송: N EEPROM으로부터 데이터 전송 시간: t	전체 소요 시간: $2N + t$ 레지스터 크기: s

도면11



도면12



도면13

시도 횟수 평균 소요 시간		
측정 구간 δ	18930회	11.62ms

도면14

	No. of ECUs	Bandwidth	No. of Messages	Cycle times
Body	14-30	100 Kbits/s	300	50 ms-2 s
Chassis	6-10	500 Kbits/s	180	10 ms-1 s
Power-train	3-6	500 Kbits/s	36	10 ms-10 s