DDS 보안기술

Security Technology for DDS

소프트웨어 기술의 미래전망 특집

정보흥 (B.H. Chung) 휴먼인식기술연구팀 책임연구원
김정녀 (J.N. Kim) 휴먼인식기술연구팀 팀장
조현숙 (H.S. Cho) 지식정보보안연구부 부장

목 차

CPS(Cyber-Physical Systems)는 고수준의 신뢰성을 기반으로 네트워크를 통해 물리 시스템을 실시간 제어하기 위한 임베디드 시스템이며 무인 주차, 항공, 스마트 그리드와 같이 다양한 의료, 군사, 교통, 로봇제어 분야에 활용 가능한 기술이다. 이러한 환경에서 시스템들 간의 실시간적이고 신뢰성 높은 데이터 통신을 제공하기 위하여 발행/구독 모델에 기반한 실시간 데이터 통신 미들웨어 표준인 DDS(Data Distribution Service)를 사용한다. 그러나, DDS는 임베디드 기기 또는 모바일 기기들이 동적으로 구성된 네트워크에 자유로운 참여, 탈퇴가 가능한 상황에서 실시간 데이터 통신에는 적합하지만 전송되는 데이터 도청, 재전송 등과 같은 다양한 네트워크 공격에는 취약하다. 따라서, 본 고에서는 DDS 보안상의 취약점과 보안강화를 위한 접근 방법에 대해 기술하고, 이후에 DDS 통신채널 보호를 위한 상용제품의 접근법과 안전 통신채널 제공을 위한 멀티캐스트 인증, 암호화에 관련된 접근방법에 대해 기술한다.

Ⅰ. 서 론

CPS(Cyber-Physical Systems)는 고수준의 신뢰성을 기반으로 네트워크를 통해 물리 시스템을 실시간 제어하기 위한 임베디드 시스템이다[1]. 즉, CPS라는 개념은 컴퓨팅의 기술진화적 관점에서 '제어'라는 조작적 개념이 추가된 것으로, 연산 및 컴퓨팅 프로세스를 수행하는 컴퓨터가 물리적인 제어장치들과의 네트워크를 통한 상호교류를 수행하여 물리적 프로세스를 효과적으로 통제할 수 있게 되는 것을 의미한다. 즉, CPS는 지능적 온실 자동제어, 무인 주차 제어, 무인 항공 제어, 스마트 그리드 등에 활용 가능하며 의료, 군사, 교통, 로봇제어, 제조업 등과 같은 다양한 국가 기반시설을 효과적으로 관리할 수 있는 선도적 미래 기술이라는 것이다. 이를 위해서는 데이터 통신상의 실시간성, 신뢰성, QoS(Quality of Service)를 보장할 수 있어야 한다.

임베디드 시스템들 간의 실시간적이고 신뢰성 높은 데이터 통신을 제공하기 위한 데이터 통신 미들웨어 기술로는 Web Service, CORBA(Common Object Request Broker Architecture), JMS(Java Message Service), DDS(Data Distribution Service) 등의 다양한 데이터 통신 미들웨어 기술 규격들이 있다[2]. 이 중에서 DDS는 동적으로 네트워크 데이터 도메인을 형성하고, 각각의 임베디드 기기나 모바일 기기들이 네트워크 데이터 도메인에 자유로운 참여나 탈퇴가 가능한 데이터 통신 환경을 제공할 수 있기 때문에 CPS에 적합한 데이터 통신 미들웨어이다. 그러나, DDS는 동적으로 네트워크가 구성되는 환경에서 실시간적이고 신뢰성있는 데이터 통신은 보장하지만 이 통신채널에 대한 보안성은 고려되어 있지 않기 때문에 메시지 도청, 재전송 공격 등과 같은 네트워크 공격에 취약하다. 따라서, DDS 보안상의 취약점이 무엇인지를 정리하고, 이를 해결하기 위한 보안기술에 대하여 설명한다.

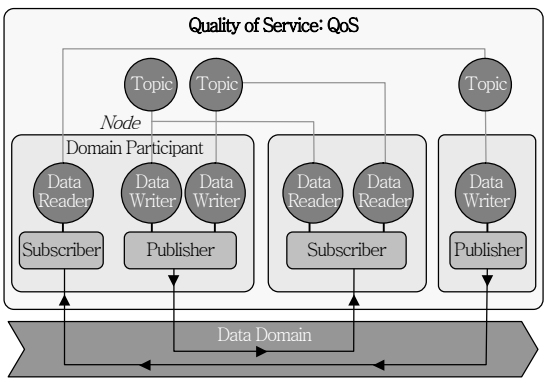
Ⅱ. DDS 취약성 및 보안강화 방법

1. DDS 개요 및 보안 취약성

가. DDS

DDS는 발행(publish)/구독(subscribe) 모델에 기반하여 분산 환경을 위한 데이터 중심 프로그램 모델에 대한 표준화의 필요성에 의해 만들어진 신뢰성있는 실시간 데이터 통신 미들웨어 표준이다[3]. DDS의 목적은 통신에 참여하는 애플리케이션의 위치나 존재에 상관없이 데이터를 전송할 수 있도록 복합한 네트워크 프로그래밍을 단순화하여 분산 어플리케이션 설계 및 구현을 단순화하는 것이다. 이 표준은 DDS API 표준을 서술한 DCPS(DDSv1.2 API Standard)[4]와 네트워크 계층 통신 프로토콜을 서술한 RTPS(RTPS v2.1 Wire Protocol Standard)[5]로 구성되어 있다. DCPS(Data Centric Publisher/Subscriber)는 발행/구독 모델에 기반한 데이터 교환 기능을 제공하는 인터페이스 규격이다.

(그림 1)과 같이 발행자는 전송할 데이터를 생성하고 배포하는 기능을 담당하고 구독자는 이 데이터



(그림 1) DDS 개념적 구조

를 수신하여 활용하게 된다. 이때, 발행자와 구독자는 DDS 내의 동일한 도메인에 참여한 상태여야만 하고 데이터는 DDS의 토픽이라는 개념으로 정의되어야만 한다. 따라서, DDS 규격에서는 토픽 데이터 전달의 신뢰성과 실시간을 위하여 "DURABLITY, HISTORY, RELIABILITY, OWNERSHIP, DEADLINE" 등의 QoS 설정을 할 수 있는 규격을 제시하고 있다. RTPS(Real Time Publish/Subscribe)는 발행/구독 모델에 기반 구현 측면의 데이터 전송 프로토콜로서 UDP(User Datagram Protocol)/IP와 같은 신뢰성 없는 전송계층 위에서도 동작 가능하도록 설계되어 있다. RTPS에서는 DCPS에서 정의한 발행자, 구독자 객체가 실제 통신하기 위해 필요한 디스커버리(discovery), 데이터 코딩 방식, 메시지 포맷 및 교환 방식, 전송절차 등에 대한 사항을 기술하고 있다.

나. DDS 보안 취약성

DDS에 존재하는 보안 취약성은 크게 도메인 분리, 응용 프로그램 무결성, 의도되지 않은 데이터 노출과 파괴, 다양한 네트워크 공격에 대한 취약성들로 분류할 수 있다. 도메인 분리 취약성이란, DDS는 도메인을 기반으로 동작[2]하게 되는데, 도메인을 보호할 수 있는 장치가 마련되어 있지 않다는 것을 말한다. 즉, 공격자가 DDS 응용 프로그램에서 손쉽게 임의의 다른 도메인에 참여할 수 있다는 것이다. 예를 들면, DDS 응용 프로그램들은 상호 간의 통신을 위해서 반드시 같은 DDS 도메인에 참여해야만 한다. 그러나, 이러한 DDS 도메인을 통한 분리는 단지 DDS 라이브러리를 통해서만 강제되고 보안적인 분리기법이라기보다는 기능적 목적으로 가지는 것이기 때문이다. 위 취약성을 해결하기 위한 방법은 DDS 도메인에 참여하는 두 응용 프로그램(발행자, 구독자)을 좀 더 세분화하는 것이다. 예를 들면 현재는 발행자, 구독자의 두 가지 역할만 존재하지만 이를 좀 더 세분화하여 역할을 할당하여 접근을 통제하는 것이다. 또는 이들간의 데이터 통신 방(IPC(Inter-Process Communication), network communication) 등에 대한 접근제한을 설정하여 통제하는 방법이다.

응용 프로그램 무결성 취약성 측면에서는 DDS 운용을 위해서 필요한 다수의 설정파일, 실행파일, 라이브러리, 감사로그 및 응용 프로그램 데이터에 대한 보호가 필요하다. 즉, 이들에 대한 사용자 또는 다른 응용 프로그램에 의한 불법적 접근(corruption or tampering)을 방지해야 한다. 또한, 신뢰된 발행자 또는 구독자를 가장하는 악성 프로그램을 방지할 수 있어야 한다. 마지막으로는 DDS 미들웨어 운용에 필요한 주요한 컴포넌트에 대한 실행 중 보안성을 제공해야 한다. 예를 들면, 다수의 데몬 또는 프로세스들이 연계되어 동작하는 경우 각각의 프로세스들에 대한 불법적인 접근을 효과적으로 차단하여야 한다는 것이다.

의도되지 않은 데이터 노출과 파괴의 위험성이 존재한다. OMG(Object Management Group) DDS 규격 중 발행자, 구독자 객체 규격에 따르면 데이터 교환을 위하여 writer와 reader 사이에 히스토리 캐시(history cache)를 사용하게 되는데, 이들 데이터는 시스템 구현방식에 따라 공유메모리, 임시 파일 등의 형태로 관리되는데 이들 데이터가 평문(plain text)로 취급되게 됨으로써 데이터 노출과 파괴의 위험성이 존재한다.

데이터 전송상의 취약성이 존재한다. OMG DDS 규격 중 데이터 통신에 관련된 RTPS 규격에 따르면, 전송에 관련된 모든 메시지들을 구분하기 위해서 GUID(Globally Unique Identifier) 값만을 정의하고 있다. 따라서, 메시지에 대한 인증 및 검증 절차가 존재하지 않아 도청, 사칭, 잘못된 메시지 고의적 전송

또는 메시지 재전송 등의 다양한 공격에 악용될 가능성이 높다. 예를 들면, 위 규격 중 reader와 writer 사이의 세션 연결을 위한 메시지 중의 하나인 "InfoDestination"메시지에 잘못된 GUID 값을 포함시켜 전송하게 되면 reader는 이 값에 대한 검증을 하지 못하기 때문에 새로운 세션을 연결할 준비를 위해 현재 연결된 세션을 종료하게 된다. 따라서, 이들 메시지들에 대하여 인증 또는 암호화를 수행하여 보호할 필요가 있다.

DoS(Denial of Service), DDoS(Distributed DoS) 등과 같은 다양한 네트워크 공격에 대한 취약성이 존재한다. RTPS 프로토콜 규격에 따르면 데이터 통신을 위한 포트 범위 및 디폴트 포트에 대한 정보는 있지만 이를 보호하기 위한 방법은 없다. 따라서, 이 포트에 대한 분산 서비스 공격을 수행하게 되면 정상적인 데이터 전송 서비스를 수행할 수 없게 된다. 따라서, 이들 포트에 대한 접근제어를 수행하는 방화벽, 침입방지 시스템을 통한 보호가 필요하다.

2. DDS 보안강화를 위한 접근법

본 절에서는 DDS 보안위협을 해결하기 위한 보안기술을 대하여 설명한다. 이를 위해서, 현재까지 DDS 상용제품인 RTI사의 DDS와 PrimsTech사의 OpenSplice에서 제시하고 있는 보안기술에 대하여 설명하고, DDS의 통신채널 보안성 향상을 위한 멀티캐스트 그룹 통신 보안기술에 대하여 설명한다.

가. 접근제어 기술을 통한 보안성 강화

SELinux(Security Enhanced Linux)는 최초에 미국 NSA(National Security Agency)에서 제안된 강제적 접근제어(mandatory access control) 메커니즘으로 현재는 리눅스 커널의 일부분으로서 존재하고, 다수의 리눅스 배포판에 포함되어 있다. SELinux는 호스트 보안, 데이터 분리, 역할기반 접근제어 특성을 가지고 있어, 다양한 위협 상황에 대처할 수 있는 안전한 응용 프로그램 솔루션 개발이 가능하다[6]. 즉, 이러한 SELinux의 특성을 활용하여 DDS 토픽, 발행자, 구독자 등의 객체에 대한 보안성을 강화하려는 것이다. 예를 들어, 응용 프로그램들(발행자와 구독자)이 DDS를 통해 통신을 수행할 때, 이 프로그램들은 다수의 스레드를 가진 단일 프로세스로 동작할 것이고 IPC 또는 표준 네트워킹 프로토콜을 사용하여 통신한다. 이 경우 SELinux는 DDS 도메인 및 이들 프로세스에 보안레벨을 부여하고(type enforcement) 접근 요청 시 부여된 보안 레벨에 따라 접근 허용 여부 결정한다.

나. Ethernet Transport Security 기술을 통한 보안성 강화

DDS는 분산된 발행자, 구독자 응용 프로그램들 간에 표준 네트워킹 프로토콜을 이용한 통신을 수행하기 때문에 IPSec(Internet Protocol Security Protocol), TLS(Transport Layer Security)/DTLS(Datagram TLS) 등과 같은 통신레벨의 보안기술 적용을 제시하고 있다. IPSec은 기밀성과 무결성을 보장하는 운영체제 수준에 적용된 통신보안기술이다. 그러나, TLS/DTLS 등에 비해 비교적 무거운 프로토콜이며 현재까지는 멀티캐스트를 지원하지 않는 단점이 있다. TLS/DTLS은 OSI(Open Systems Interconnection) 7 Layer상의 Transport Layer상에서 동작하며 역시 기밀성과 무결성을 보장하는 통신보안기술이다. 이 역시, IPSec에 비해서는 응용 프로그램 간의 통신상 보안성을 효과적으로 지원할 수 있지만 멀티캐스트를 지원하지 않는 단점이 있다. 또한, 이러한 프로토콜들은 프로토콜을 운영하기 위하여

기본적인 인증키 및 인증서버를 필요로 하고 있으나, 실시간 멀티캐스팅을 사용하는 DDS에 적용하기에는 어려움이 있다.

다. 침입차단 기술을 통한 보안성 강화

이 방법은 DDS는 발행자, 구독자 간의 네트워크 트래픽을 차단하여, 서비스 공격 등과 같은 네트워크 공격으로부터 DDS를 안전하게 보호하기 위한 것이다. 리눅스 환경에서는 방화벽 기능을 제공하는 IP-Tables를 제공하고 있어, 시스템에서 열려진 포트에 대한 접근 허용/거부 여부를 제어할 수 있다.

Ⅲ. DDS 통신채널 보호

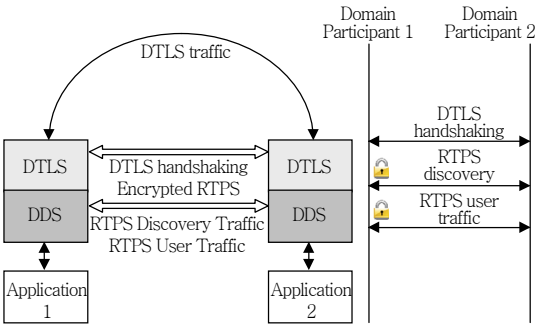
DDS는 데이터 통신상의 실시간성과 신뢰성 보장하며 일반적으로 멀티캐스트 전송 방식을 사용한다. 즉, 동일한 도메인에서 응용 프로그램들 간의 데이터 전송을 최소한의 자원으로 수행하기 위해서 IP 멀티캐스트 기법을 사용한다[7]. 이 방식은 데이터 통신을 수행한 응용 프로그램을 하나의 그룹으로 구성하고, 송신자가 데이터 패킷을 한 번만 전송하여 모든 수신자에게 전달하게 하는 방식이다. 이는 동일 패킷을 n개의 복사본을 만들어 전송하여야 하는 유니캐스트 방식에 비해 장점이 된다. 그러나, 데이터가 멀티캐스트방식으로 전달되기 때문에 메시지에 대한 비밀성을 제공하여 악의적인 데이터 취득을 방지하여야 하고, 이 데이터가 수신자에게 정확히, 변경 없이 전달됨을 보장할 수 있는 무결성을 제공하여야 한다. 따라서, 통신채널 보호가 DDS 보안을 위해서는 가장 중요한 요소이다.

본 절에서는 DDS 통신채널을 보호하기 위한 보안 기술을 대하여 설명한다. 이를 위해서, 현재까지 DDS 상용제품인 RTI사의 DDS와 PrimsTech사의 OpenSplice에 적용된 보안기술에 대해 설명하고, 실시간 멀티캐스트 채널의 비밀성을 보장하기 위한 멀티캐스트 암호화 기술과 데이터 무결성을 보장하기 위한 멀티캐스트 인증 기술에 대하여 설명한다.

1. DDS 제품에서의 통신채널 보호 방법

DDS 통신채널을 보호하기 위해서는 VPN(Virtual Private Network), SSL(Secure Socket Layer) 등과 같은 암호화 통신채널을 연결한 후에 이 암호화 채널을 통해서 DDS 통신을 수행하는 방법과 DDS 프로토콜 규격을 상호인증이 가능하도록 확장하는 방법이 있을 수 있다. 현재까지는 상용 DDS 제품들에서는 전자의 방식을 채용하고 있고, 후자의 방법은 최근에 연구가 진행되고 있다.

DDS 상용제품으로는 “RTI사 DDS”와 “PrimsTech사의 OpenSplice”가 있다. RTI사는 DDS 하부에 DTLS를 채용하여 인증과 데이터 보호를 수행하는 방식으로 통신채널을 보호하는 방법을 제공한다[7]. 또한, PrimsTech사도 각 연결된 노드들 사이에 전용 암호화 채널을 제공하기 위해 “OpenSplice DDS Secure Networking”라는 컴포넌트를 제공하고 있다[8]. 예를 들면, 이들 제품들은 (그림 2)와 같이 TLS(or DTLS)를 통해 두 노드에 대한 상호인증 과정을 수행한다. 이 과정을 통해 양쪽 노드에 공유



(그림 2) DTLS를 이용한 통신채널 보호

비밀키가 분배되고 이후의 트래픽은 이 키를 이용하여 암호화하여 전송하는 방식이다.

다른 접근 방법으로는 통신채널 자체를 암호화하여 보호하는 방식이 아니라, 인증/인가(AA: Authentication and Authorization) 특성을 DDS의 RTPS 프로토콜에 구현하는 방법이다[9]. 그러나, DDS는 프로토콜 스택상에서 트랜스포트와 응용 레이어에 위치하게 되어, 물리적인 네트워크의 구성을 알 수 없어 통신을 수행할 대상을 디스커버리를 통해 알아내야만 된다. 따라서, 이 방법은 RTPS 디스커버리 단계에 인증과정을 구현하게 된다. 그러나, 상호인증 및 메시지 암호화를 위한 키를 관리하는 서버가 존재하지 않는 DDS 운영환경에서는 키를 상호교환 하거나 분배하는 것이 어려운 문제이다. 따라서, 발행자/구독자를 마치 도난당하지 않고 불법적 접근이 불가능한 하드웨어 장비 또는 소프트웨어로 간주하고 사전에 공유 비밀키를 분배해 둔다. 이후에 RTPS 디스커버리에서 상대방을 검색하기 위해서 보내는 메시지에 포함된 정보인 GUID, 난수 값을 이 비밀키로 암호화하여 보낸다.

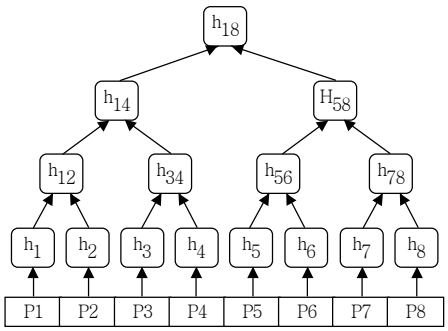
2. 멀티캐스트 인증

멀티캐스트 패킷 스트림을 인증하기 위해서는 패킷에 인증정보를 추가해야 한다. 비디오/오디오와 같은 미리 생산 또는 기록된 콘텐츠에 인증정보를 추가하는 것은 어렵지 않으나, 실시간 스포츠/뉴스 또는 주식정보와 같이 실시간적으로 생성되는 콘텐츠에 인증정보를 생성하고 추가하는 것은 쉽지 않다. 더욱이, 실시간 주식정보와 같은 실시간 데이터의 경우에는 공격자가 정보 배포자를 가장하여 거짓정보를 생성하게 된다면 심각한 상태를 초래할 수도 있다. 따라서, 수신자가 전달받은 콘텐츠의 출처를 검증할 수 있는 소스인증과 수신자가 다시 제3자에게 데이터 송신자를 증명할 수 있도록 하는 부인거부 특성을 제공해야 한다. 일반적으로 소스인증은 위해서는 양자 간에 미리 공유된 비밀키를 통한 MAC(Message Authentication Code)를 사용하고, 부인봉쇄는 비대칭적 암호화인 디지털 시그니처 방식을 사용하여 제공할 수 있다. 그러나, 멀티캐스트 환경에서는 이론적으로 무제한의 신뢰되지 않은 수신인이 존재하고, 패킷 손실이 발생 가능한 통신채널을 통해 무한대의 패킷 스트림이 전달된다. 이는 각각 다자 참여, 데이터 스트리밍 특성이라고 하며 양자 간 인증에 비해 멀티캐스트 인증을 어렵게 하는 요소이다. 다시 말하자면, 양자 간 인증과 다르게 멀티캐스트 인증에서는 수신인을 잠재적인 공격자로 보고 송신자와 수신자 사이의 단순한 대칭적 MAC 방식을 사용하면 안된다는 것이다.

스트림 데이터 인증을 위한 가장 손쉬운 방법은 스트림의 모든 패킷에 공개키/비밀키 기반의 전자서명을 추가하는 것이다. 그러나, 전형적인 1024비트 공개키/비밀키 기반의 전자서명을 모든 패킷에 추가하는 것 자체가 큰 오버헤드이고 이 키들을 계산하기 위한 비용이 크기 때문에 적용하기에는 힘들다. 따라서, 통신 오버헤드를 절감하고 적은 계산비용을 가지는 멀티캐스트 인증을 수행하고, 패킷에 인증정보를 분산하여 추가한 경우, 전송된 패킷 중 일부가 손실되는 경우에도 인증을 수행할 수 있는 방법에 대한 다양한 연구가 진행되었다[10]. 이들 방법 중 대표적인 해시 트리, 해시 체인, TESLA 등에 대해서 설명한다.

가. 해시 트리(Hash Tree)

이 방식은 스트림 패킷들을 일정 크기의 블록으로 구분하고, 블록별 인증정보를 구성한다. 이 인증정보를 각 블록 모든 패킷에 분산시키기 위해서 암호화

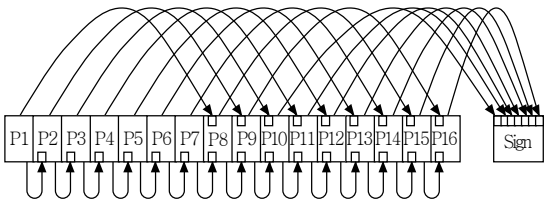


(그림 3) 해시 트리를 이용한 인증정보 관리

해시 함수(H)와 완전한 이진트리를 구성하여 사용하는 방법이다[11]. 즉, (그림 3)과 같이 이진트리가 구성되고, 이 트리의 말단노드는 패킷(P_i)에 대한 해시(h_i)를 가지게 되고, 중간노드는 두 자식노드의 해시 값을 이용한 해시를 수행한 값을 가진다. 따라서, 각 패킷과 블록에 대한 해시 값을 계산하기 위해 필요한 중간 해시 값 집합을 같이 전송하여 개별 패킷의 인증과 패킷이 손실된 경우의 인증도 해결한다. 예를 들면, (그림 3)에서 패킷 P2는 A2 = {h1, h34, h58} 세트와 함께 전송되고, 모든 수신자는 루트 해시 h18을 h18 = H(H(H(h1 | H(P2)) | h34) | h58)로 검증할 수 있다. 블록 해시 값(h18)은 블록 전송이 끝난 후 전송한다. 따라서, 수신자는 블록의 전체 패킷을 버퍼링하고 있어야 한다. 그렇지 않고, 수신된 패킷들을 독립적으로 인증하기 위해서는 h18을 모든 패킷에 추가하면 되는데 오버헤드가 증가한다.

나. 해시 체인(Hash Chains)

이 방식은 블록에서 최대 β 패킷이 갑작스럽게 손실되는 경우에도 유용한 스트림 인증기법으로, 하나 또는 몇 개의 패킷들에 현재 패킷의 암호화 해시를 삽입하여, 블록 내 패킷들 사이에서 (그림 4)와 같이 방향성을 가지는 비순환 그래프를 구성하는 방법이다[12]. 그리고, 마지막 패킷으로는 이 암호화 해시 값에 대한 전자서명을 한 패킷을 전송하고, 수신자는



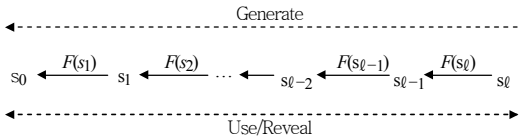
(그림 4) 해시 체인을 통한 인증정보 관리

이 패킷이 도착할 때까지 전송된 패킷 들을 버퍼링한다. 즉, 전송하는 패킷에 대한 암호화 해시 값을 이후에 전송될 하나 이상의 패킷에 추가하여 전송과정에서 불법적으로 변경되지 않았음을 검증하는 것이다. (그림 4)에서 P1의 해시 값은 P2와 P8에 추가되고, P8의 해시 값은 P9과 P16에 추가된다. 이런 식으로 체인 구조를 가지게 되고 마지막 패킷에는 P10에서 16까지 패킷의 해시 값을 모두 추가한 후 전자서명하여 보낸다. 따라서, 공격자가 중간의 패킷들을 변경하여 보낸 경우에도 마지막으로 보내진 전자서명 패킷에 포함된 해시 값들을 이용하여 이전에 보내진 패킷들의 해시 값을 역순으로 검증할 수 있게 된다.

다. TESLA(Timed Efficient Stream Loss-tolerant Authentication)

TESLA는 이전 방식과는 다르게 단방향 해시 함수를 통해 생성된 비밀키를 포함해서 계산된 MAC 값을 패킷에 추가하는 방식이다[13]. 즉, 일회용 비밀키와 같이 개별 패킷을 보낼 때 서로 다른 비밀키를 이용하여 MAC을 계산하여 보냄으로써 공격자가 거짓 멀티캐스트 패킷을 전송할 수 없게 하고, 소스인증을 효과적으로 수행할 수 있게 한다.

이 방식을 간략히 설명하면, 먼저 단방향 해시 함수(F)를 이용하여 길이 l의 비밀키를 생성한다. 즉, 이전 해시 함수의 결과 값을 다음 해시 함수의 시드 값으로 사용한다. 이후에, 송신자와 수신자는 시간 동기화를 수행하고 시간을 일정 시간 간격으로 구분한다. 이때 생성된 비밀키들을 각 시간 간격 경과 시마다



(그림 5) 인증을 위한 키 생성/사용 개념도

생성한 순서의 역순으로 사용한다. 예를 들면, (그림 5)에서 보는 바와 같이 F함수를 이용하여 s_0, s_1, … , s_ℓ을 생성한다. 이때 s_0와 F함수는 통신을 개시하기 전에 송신자와 수신자 간에 사전 공유가 되어 있어야 한다. 이후에 첫 번째 시간 간격에 전송되는 데이터는 S_1으로 계산한 MAC 값과 함께 수신자에게 전송되고, 수신자는 이들을 버퍼링하여 둔다. 다음 번 시간 간격에는 S_1을 수신자에게 공개한다. 그러면, 수신자는 미리 공유된 F와 S_0를 알고 있는 상태에서 $F^i(S_i) = S_0$가 됨을 쉽게 확인하게 된다. S_i를 아는 경우에 S_{i+1}은 다음번 공개되기 전까지는 계산할 수 없기 때문에 공격자가 다음번 패킷을 거짓으로 생성하여 보낼 수 없다. 즉, 송신자만이 정확한 S_i 값으로 MAC 값을 계산하여 보낼 수 있기 때문에 정확한 송신자 인증이 가능하게 된다. 단, 키가 공개되기 전까지의 시간 간격 동안 전송된 패킷들은 수신자가 버퍼링하고 있어야 하며 키가 공개된 후까지 버퍼링된 패킷들에 대한 인증이 미루어진다는 특성을 가지게 된다.

3. 멀티캐스트 암호화

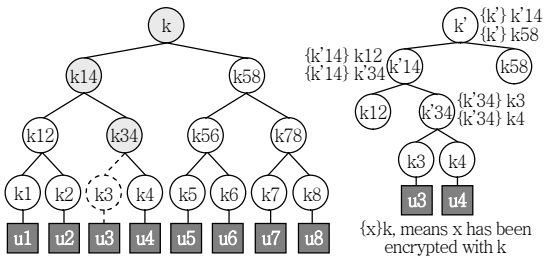
멀티캐스트 암호화는 송/수신자 간 상호 협의된 키를 이용하여 패킷 데이터를 암호화하는 방식으로 가장 중요한 이슈는 일반적으로 키의 비밀성을 유지하면서 안전하게 키 관리/배포하는 것이다. 그러나, 멀티캐스트 환경에서는 그룹 멤버의 참여/탈퇴에 따른 키 비밀성으로 순방향 비밀성(forward secrecy)과 역방향 비밀성(backwards secrecy)이 추가적으로 고려되어야 한다.

멀티캐스트 기반의 그룹통신에서는 전송데이터와 그룹정보를 암호화하는데 사용되는 키(이 키를 그룹키라 한다)를 선정하고, 이 키를 그룹 멤버에게 선별적으로 제공하여 불법적인 접근을 제한하여야 한다. 따라서, 그룹키를 가진 사용자만이 원본 메시지를 복원할 수 있다. 그룹 멤버십에 변경이 발생하면 그룹키에 대한 변경을 수행하여 그룹키의 비밀성을 유지하여야 한다. 이는 새로운 멤버가 합류하기 이전에 교환된 메시지를 해독할 수 없도록 하고(순방향 비밀성), 그룹을 떠나거나 축출된 멤버가 그룹통신에 참여할 수 없도록 막기 위함(역방향 비밀성)이다[14]. 그러나, 변경된 그룹키를 분배하는 것은 매우 쉬운 문제가 아니다. 왜냐하면, 새로운 멤버가 참여하는 경우 이 멤버에게는 새로운 키를 주고, 이전 멤버들에게는 예전 그룹키로 새로운 그룹키를 암호화하여 전달하면 되지만, 멤버가 떠난 후에는 이 멤버가 이전 그룹키를 알고 있기 때문에 이전 키로 새로운 그룹키를 암호화하여 전송할 수 없기 때문이다. 따라서, 멀티캐스트 암호화에서는 그룹키를 안전하게 그룹 멤버에게 배포하고 멤버십 변경에 따라 효과적으로 재분배하기 위한 키 관리방식이 가장 핵심적인 요소이다. 이를 위해 다양한 방법들이 연구되었는데[14] 이를 정리하면 크게 중앙에 키 분배 센터(KDC: Key Distribution Center)를 두어 전체 그룹을 통제하는 중앙집중적 그룹키 관리(centralized group key management), 이 노드에 부하가 집중되는 것을 방지하기 위해 여러 개의 서브 그룹으로 나누고 각 그룹에 서브 KDC를 운영하는 비중앙 집중적 구조(decentralized architectures), 명시적인 KDC가 없고 각각의 멤버가 키 생성에 참여하는 분산 키 관리 프로토콜(distributed key management protocols)로 구분된다. 여기서는 이들 중 대표적인 기법들에 대해서만 설명한다.

가. Logical Key Hierarchy(LKH)

가장 보편적이고 일반적인 키 관리 방식으로 KDC를 두어 그룹키를 관리하며, 각 멤버에게 그룹키를 분배하기 위한 KEK(Key Encryption Key)를 사용하고, 이들을 바이너리 트리구조로 관리한다[15].

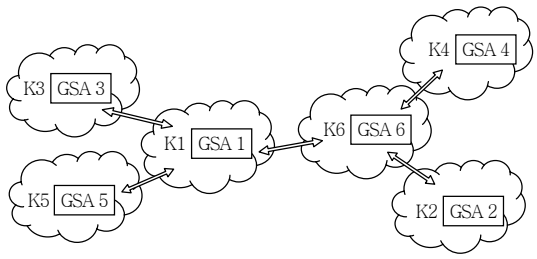
즉, (그림 6)에서 처럼 루트노드는 그룹키이고, 각 멤버에 해당하는 말단노드와 루트노드 사이에 있는 중간노드들은 KEK이다. 만일, 이때 새로운 멤버가 참여한 경우 그룹키의 변경이 필요하며 왼쪽 그림의 진하게 표시된 노드의 KEK 값만 변경하면 된다. 따라서, 오른쪽 그림과 같이({k'34}k3, {k'34}k4, {k'14}k12, {k'14}k'34, {k'}k'14, {k'}k58) 새로이 생성된 여섯 개의 키를 포함한 멀티캐스트 패킷을 전송하면 된다. 여기서 {k'34}k3은 k3으로 k'34를 암호화하였다는 것을 의미한다. 이는 새로운 멤버의 참여, 탈퇴의 경우 새로운 키를 생성, 분배 시 키 생성 및 배포의 부하를 줄이기 위함이다.



(그림 6) LKH를 통한 그룹키 관리

나. Iolus

Iolus는 비중앙 집중적 구조로서 (그림 7)와 같이 커다란 그룹을 작은 서브 그룹으로 나누고 그룹 보안 에이전트(GSA: Group Security Agent)가 각 서브 그룹을 관리하는 구조이다[16]. 이들 GSA들은 다시 그룹화되어 그룹 보안 컨트롤러(GSC: Group Security Controller)가 관리한다. 각 서브 그룹 간에는 서로 독립적인 그룹키를 관리하고 이 그룹 내의 멤버십



(그림 7) Iolus를 통한 그룹 키 관리

변화는 지역적으로 처리된다. 이는 GSC의 과부하는 줄일 수 있지만 서로 다른 GSA 간 통신을 위해서는 서브 그룹 간 데이터 변환과정이 필요하고 이를 위해 GSC가 이 변환을 위한 키를 따로 관리하여야 한다.

다. Group Diffie-Hellman Key Exchange

분산 키 관리 프로토콜로서 기존의 DH(Diffie-Hellman) 키 합의 프로토콜이 확장된 방식이다[17]. DH 방식은 양자 간 공유키를 합의하기 위한 프로토콜이지만, 여기서는 이를 n 멤버가 존재하는 그룹에서 할 수 있도록 확장되었다. 먼저 모든 그룹 멤버는 소수 q와 α를 합의한 후, 첫 번째 멤버가 α^{x1}을 계산하여 다음 멤버에게 전달하고, 이후의 멤버들은 자신만 아는 임의의 숫자를 이용하여 중간 값을 생성하여 전달한다. 이 과정을 수행하고 나면 모든 멤버들은 그룹키 k를 $k=\alpha^{x1\cdots xn} \bmod q$를 통해서 계산하게 된다. 이 방식은 그룹키를 설정하는 시간은 선형적인 장점이 있지만 마지막 멤버에게 도달하기 위한 시간은 지수승에 비례한다는 단점을 가지게 된다.

Ⅳ. 결론

지금까지 효율적인 CPS 구축 운용을 위한 핵심 통신 미들웨어 기술인 DDS의 보안 취약성과 이를 해결하기 위한 보안기술을 설명하였다. DDS는 발간/구독 모델에 기반하여 망이나 서버의 부하를 최소화하

면서 신뢰성있게 동일 도메인 내 다수의 구독자들에게 동시에 실시간으로 동일한 정보를 보낼 수 있는 효과적인 기술이다. 그러나, DDS에는 도메인 분리, 응용 프로그램 무결성, 의도되지 않은 데이터 노출과 파괴, 다양한 네트워크 공격에 대한 취약성들이 있어 CPS 환경에서 효과적으로 사용되기 위해서는 보안성을 높일 필요가 있다. 이를 위해, 상용 제품인 RTI사의 DDS와 PrimsTech사의 OpenSplice에서는 SELinux를 사용한 시스템 보호, TLS/DTLS를 이용한 암호화 통신채널 제공, 방화벽을 통한 네트워크 포트 보호 등의 보안성 강화를 위한 노력을 추진해왔다. 그러나, 이러한 시도는 DDS를 운영하기 위한 노드 시스템의 보안강화 및 정적인 네트워크 환경에서 암호화 통신채널을 제공하는 것이기 때문에, 발행자, 구독자 간에 동적으로 도메인을 구성하여 데이터를 전송하는 DDS 환경에서 운영하기에는 효과적이지 않다. 또한, 실시간 데이터 통신을 안전하게 하기 위해 연구된 멀티캐스트 인증, 암호화는 공유 비밀키 분배를 위한 별도의 인증 및 키 관리 서버를 운영하여야 하기 때문에 DDS에 적용하기에는 어려움이 있다. 따라서, DDS의 보안성을 향상시키기 위해서는 별도의 서버 없이 인증, 암호화를 적은 비용으로 수행할 수 있는 방법을 개발하여 DDS의 프로토콜 규격에 포함시키거나 또는 구현 시 적용하여야 할 것이다.

● 용 어 해 설 ●

분산 데이터 서비스(DDS): Data Distribution Service는 발간/구독 모델에 기반하여 실시간 데이터 통신 미들웨어 표준 기술임.

발행자/구독자: 발행자(publisher)는 DDS에서 토픽는 데이터를 생성하여 배포하는 역할을 담당하는 객체이며, 구독자(subscriber)는 이 토픽 데이터를 구독(subscription)하는 객체를 말함.

약어 정리

AA	Autheriazation and Authorization
CORBA	Common Object Request Broker Architecture
CPS	Cyber-Physical Systems
DCPS	Data Centric Publisher/Subscriber
DDoS	Distributed DoS
DDS	Data Distribution Service
DH	Diffie-Hellman
DoS	Denial of Service
DTLS	Datagram TLS
GSA	Group Security Agent
GSC	Group Security Controller
GUID	Globally Unique Identifier
IPC	Inter-Process Communication
IPSec	Internet Protocol Security Protocol
JMS	Java Message Service
KDC	Key Distribution Center
KEK	Key Encryption Key
LKH	Logical Key Hierarchy
MAC	Message Authentication Code
NSA	National Security Agency
OMG	Object Management Group
OSI	Open Systems Interconnection
QoS	Quality of Service
RTPS	Real Time Publish/Subscribe
SELinux	Security Enhanced Linux
SSL	Secure Socket Layer
TESLA	Timed Efficient Stream Loss-tolerant Authentication
TLS	Transport Layer Security
UDP	User Datagram Protocol
VPN	Virtual Private Network

참 고 문 헌

[1] Lui Sha et al., "Cyber-Physical Systems: A New Frontier," *Lecture Note Comput. Sci.*, Springer, DOI: 10.1007, Apr. 5th, 2009.

[2] 전형국 외, "DDS 미들웨어 표준 기술 동향," 주간기술동향 통권, 1456호, 2010. 7.

[3] OMG, OMG Data Distribution Service. http://www.omg.org

[4] OMG, Data Distribution Service for Real-time Systems Version 1.2. http://www.omg.org
[5] OMG, The Real-Time Publish-Subscribe Wire Protocol: DDS Interoperability Wire Protocol Specification Version 2.1. http://www.omg.org/spec/DDSI/2.1
[6] NSA, Security-Enhanced Linux. http://www.nsa.gov/research/selinux
[7] RTI, RTI Secure WAN Transport. http://www.rti.com
[8] PrimsTech, Secure Networking. http://www.primstech.com/openslice/products/commercial-edition
[9] Fabrizio Ronci and Marco Listanti, "Embedding Authentication & Authorization in Discovery Protocols for Standard Based Publish/Subscribe Middleware: A Performance Evaluation," *J. Commun. Netw.,* vol. 3, no. 1, Feb. 2011, pp. 39-49.
[10] A. Pannetrat and R. Molva, "Efficient Multicast Packet Authentication," *Proc. Netw. Distributed Syst. Security Symp.*, San Diego, CA, Feb. 2003.
[11] C.K. Wong, M. Gouda, and S.S. Lam, "Secure group Communications Using Key Graphs," *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, Feb. 2000, pp. 68-79.
[12] V. Paxson, "End-to-end Internet Packet Dynamics," *IEEE/ACM Trans. Netw.*, vol. 7, no. 3, June 1999, pp. 277-292.
[13] Adrian Perrig et al., "The TESLA Broadcast Authentication Protocol," RSA CryptoBytes, 5(Summer), 2002.
[14] Sandro Rafaeli and David Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Comput. Surveys*, vol. 35, no. 3, Sept. 2003, pp. 309-329.
[15] C.K. Wong, M.G. Gouda, and S.S. Lam, "Secure Group Communications Using Key graphs," *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, Feb. 2000, pp. 16-30.
[16] S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting," *Proc. ACM SIGCOMM*, vol. 27, no. 4, Sept. 1997, pp. 227-288.
[17] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman Key Distribution Extended to Group Communication," *SIGSAC Proc. 3rd ACM Conf. Comput. Commun. Security*, Mar. 1996, pp. 31-37.