한국정보전자통신기술학회논문지(jkiiect)'15-12, Vol.8 No.6
논문 15-08-06-520 http://dx.doi.org/10.17661/jkiiect.2015.8.6.520

ETSI 표준 키 교환 프로토콜의 성능 분석

이영석*, 최 훈

Performance Analysis of Key Exchange Protocols on ETSI Standard

Young-Seok Lee*, Hoon Choi

요 약 ETSI(European Telecommunications Standards Institute) EN 301 790에서는 광대역 위성망에서 제공되어야 할 두 가지 보안 서비스를 기술하고 있는데, 하나는 사용자(개체) 인증 서비스이고, 다른 하나는 사용자들 간의 데이터의 흐름이나 사용자와 관리자간의 데이터의 흐름을 악의적으로 공격하여 허락되지 않도록 불법적인 접근으로부터 데이터를 보호하는 기밀성 서비스이다. 특히, 기밀성 서비스를 제공하기 위해 ETSI에서는 NCC(Network Control Centre)와 RCST(Return Channel Satellite Terminal) 사이에 Main Key Exchange, Quick Key Exchange, 그리고 Explicit Key Exchange 세 가지 키 교환 프로토콜을 제시하고 있다. 본 논문에서는 ETSI 표준의 키 교환 프로토콜을 분석하고, 성능 분석과 성능 평가를 통하여 키 교환 프로토콜의 효율성과 장단점을 분석한다.

Abstract The key exchange protocols are very crucial tools to provide the secure communication in the broadband satellite access network. They should be required to satisfy various requirements such as security, key confirmation, and key freshness. In this paper, we present the security functions in ETSI(European Telecommunications Standards Institute), and analyze the specification of the security primitives and the key exchange protocols for the authenticated key agreement between RCST(Return Channel Satellite Terminal) and NCC(Network Control Centre). ETSI key exchange protocols consists of Main Key Exchange, Quick Key Exchange, and Explicit Key Exchange. We analyse the pros and cons of key exchange protocols based on performance analysis and performance evaluation.

Key Words : ETSI, Key exchange protocol, NCC, RCST, Authentication

1. 서론

위성 통신 환경은 무선으로 데이터를 송 · 수신하기 때문에, 악의의 공격자가 제약 없이 네트워크에 공격 할 수 있다. 그러므로 위성 통신은 도청 및 데이터의 위조, 변조와 같은 다양한 공격 기법에 쉽게 노출된다. 위와 같은 문제를 해결하기 위해 인증, 기밀성, 무결성, 부인방지 등의 정보보호 서비스를 제공해야 한다. 이러한 서비스를 제공하기 위해 많이 사용하는 방법은 키를 이용한 암 · 복호화 및 인증이다. 하지만 기존 무선 통신 환경에서 안전하게 키를 설립하는 것은 어려운 일이며, 기존에 제안된 프로토콜들은 중간자 공격 등의 공격에 노출되어 있는 문제점이 있다[1]. 이러한 문제점을 해결하기 위해 ETSI(European Telecommunications Standards Institute, 유럽 전기통신 표준기구)에서는 위성망에서의 보안을 제공하는 EN 301 790 표준을 제정하였다[2].

그림 1은 광대역 위성망의 참조 모델을 보여준

This research was supported by the Ministry of Education (MOE) and National Research Foundation of Korea(NRF) through the Human Resource Training Project for Regional Innovation (No. 2013H1B8A2032180)
*Corresponding Author : Department of Information Communication Engineering, Kunsan National University (leeys@kunsan.ac.kr)
Received December 02, 2015 Revised December 10, 2015 Accepted December 20, 2015

다.그림 1에서 NCC(Network Control Center)는 망 관리 센터를 의미하며, RCST(Return Channel Satellite Terminal)는 사용자의 위성 단말을 의미한다[3].

ETSI EN 301 790에서는 그림 1과 같은 광대역 위성망에서 제공되어야 할 두 가지 보안 서비스를 기술하고 있는데, 하나는 사용자(개체) 인증 서비스이고, 다른 하나는 사용자들 간의 데이터의 흐름이나 사용자와 관리자간의 데이터의 흐름을 악의적으로 공격하여 허락되지 않도록 불법적인 접근으로부터 데이터를 보호하는 기밀성 서비스이다[4].

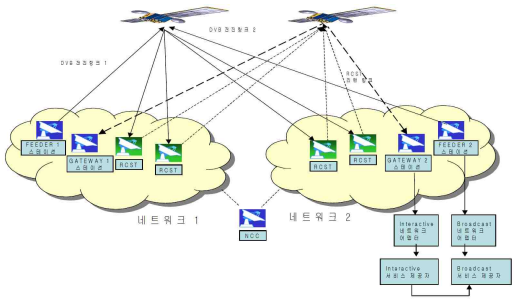

그림 1. 위성 Interactive 망 참조 모델
Fig. 1. Satellite Interactive Network Reference Model

특히, 기밀성 서비스를 제공하기 위해 ETSI에서는 Main Key Exchange, Quick Key Exchange, 그리고 Explicit Key Exchange 세 가지 키 교환 프로토콜을 제시하고 있다.

본 논문에서는 ETSI 표준의 키 교환 프로토콜을 분석하고, 성능 분석과 성능 평가를 통하여 ETSI 키 교환 프로토콜의 효율성과 장단점을 분석한다.

본 논문은 다음과 같이 구성된다. 2장에서는 ETSI 표준 키 교화 프로토콜의 기술한다. 3장에서는 ETSI 키 교환 프로토콜의 성능을 분석하고, 4장에서는 시뮬레이션을 수행하여 ETSI 키 교환 프로토콜의 성능을 평가한다. 5장에서 결론을 맺는다.

2. ETSI 표준 키 교환 프로토콜

2.1 Main Key Exchange(MKE) 프로토콜

MKE은 Diffie-Hellman을 사용하여 쿠키의 값과 무관하게 NCC와 RCST 사이의 비밀 값을 만든다. MKE의 쿠키 값은 RCST를 NCC로부터 인증을 받도록 한다. 그리고 쿠키 값의 갱신을 위하여 새로 만들어진 공유 값을 선택적으로 사용하게 된다. 마지막으로, MKE은 페이로드 스트림 데이터 처리를 위한 암호문을 위해 사용하는 공유 비밀키를 유도한다.

MKE은 Diffie-Hellman 에 사용되는 m,g,X 값과 랜덤 임시 문자열인 nonce1을 포함한 메시지를 NCC가 전송하면서 초기화가 된다. RCST은 자신의 Diffie-Hellman 값인 Y와 랜덤 임시 문자열 nonce2와 인증 값 auth를 포함한 메시지로 응답한다. NCC와 RCST은 인증 값을 계산하기 위하여 동일한 계산을 한다. 아래의 수식은 인증 값 auth의 계산식을 나타낸다.

· auth = H(cookie, nonce1 ~ nonce2)

auth값은 RCST가 NCC에게 전달되고 NCC은 이를 확인한다. 정확한 auth 값을 계산하기 위해서 쿠키 값을 사용하기 때문에 단말기의 인증이 가능하다. NCC와 RCST는 각자 Diffie-Hellman 값을 사용하여 동일한 비밀 값인 s를 계산한다. 아래의 수식은 비밀값 s의 계산식을 나타낸다.

· s = g^(x*y) mod m

unsigned integer 값은 big-endian 바이트 정렬 방식을 적용하여 Diffie-Hellman 파라미터 크기에 의하여 정해진 길이의 바이트 문자열로 생성된다. 이 값을 이용하여 임시 공유 비밀 문자열(temp)을 생성한다. 아래의 수식은 temp의 계산식을 나타낸다.

temp = H(encode(s), nonce2 ~ nonce1)

만약 쿠키 값을 갱신할 경우, n=1,2,...의 섹션의 새로운 쿠키값(new cookie)을 계산한다. 아래의 수식은 newcookie(n)의 계산식이다.

· newcookie(n) = H(temp ~ (unsigned char)1 ~ (unsigned char)n, "")

문자열 값은 총 길이가 쿠키의 길이와 같거나 클 때까지 계산하여 순차적으로 결합한다. 그 후, 결합한 문자열의 처음 20 바이트를 쿠키 값으로 사용하게 된다. 페이로드 스트림의 암호화를 위한 세션 키는 위 방식과 비슷한 방식으로 계산하게 된다.

· key(n) = H(temp ~ (unsigned char)2 ~ (unsigned char)n, "")

이 과정에서 많은 수의 섹션을 계산하여 키의 길이 이상의 충분한 값을 생성해야 한다. 이러한 세션키는 "쿠키 생성과 동일한 방식"으로 계산한다. 그림 2는 MKE 프로토콜의 흐름도를 나타낸다. NCC와 RCST가 가지고 있다고 가정되는 값은 괄호로 표시된다.

그림 2의 절차에서 확인 할 수 있듯이 MKE 프로토콜은 새로운 세션을 설정할 때 마다 수행되는 프로토콜이다. 사용자 인증을 위하여 auth(x)를 사용하며 쿠키 값을 이용하게 된다. 키의 주기적인 갱신을 위해 임시 Diffie-Hellman 공개키, 개인키를 생성하여 교환하게 된다.

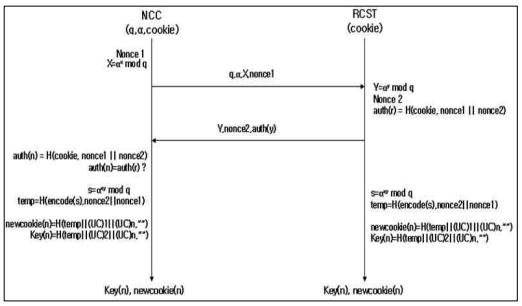

그림 2. Main Key Exchange 절차
Fig. 2. Main Key Exchange Procedure

2.2 Quick Key Exchange(QKE) 프로토콜

QKE는 현 상태의 쿠키 값을 이용하여 NCC로부터 RCST의 인증을 받고 페이로드 스트림 데이터 처리 암호문을 위해 사용하는 공유 비밀키를 생성한다. QKE는 NCC가 랜덤 임시 문자열인 nonce1을 포함한 메시지를 보내면서 초기화가 시작된다. RCST는 랜덤 임시 문자열인 nonce2와 인증 값 auth를 포함한 메시지로 응답한다. auth 값은 MKE와 같은 방법으로 계산되며 이는 RCST의 인증을 위해 사용된다. NCC와 RCST는 각각 임시 공유 문자열인 temp를 계산한다.

· temp = H(cookie ~ (unsigned char)3, nonce2 ~ nonce1)

이 값을 이용하여 MKE 와 같은 방법으로 페이로드 암호화 키를 생성한다. 그림 3은 QKE 프로토콜의 흐름도를 나타낸다. 마찬가지로 괄호의 값은 NCC와 RCST가 갖고 있는 값을 나타낸다.

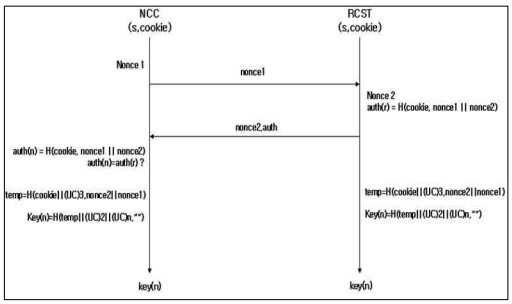

그림 3. Quick Key Exchange 절차
Fig. 3. Quick Key Exchange Procedure

MKE 프로토콜과의 큰 차이점은 기존에 갖고 있는 쿠키와 비밀 값을 이용하여 사용자 인증 기능을 수행한다는 점이다. 새로운 비밀 값을 계산하기 위한 과정이 없기 때문에 Diffie-Hellman 과정이 수행 되지 않는다. QKE의 프로토콜의 큰 기능은 빠른 시간 안에 인증을 위한 것이다.

2.3 Explicit Key Exchange(EKE) 프로토콜

EKE는 NCC가 RCST에게 결정된 세션 키를 전송한다. 세션 키는 쿠키 값으로부터 얻은 임시 키를 통해 암호화되며 페이로드 스트림 데이터 처리 암호문을 위해 사용된다.

NCC는 랜덤 임시 문자열 nonce1 과 바이트 단위 문자열 encryptedkey 값을 RCST에게 보내면서 임시 키를 전달하게 된다. 이 과정에서 바이트 단위의 문자열 값의 길이는 페이로드 암호화를 위해 사용되는 키와 같다. RCST는 랜덤 임시 문자열 nonce2와 인증 값 auth를 포함한 메시지

로 응답한다. auth 값은 MKE 과 같은 방식으로 계산하며 이는 RCST를 인증하기 위해 사용된다. NCC와 RCST는 각가 임시 비밀 문자열 temp를 계산한다. 아래의 계산식은 temp의 계산식을 나타낸다.

· temp = H(cookie ~ (unsigned char)4, nonce1)

temp 값은 MKE 과 같은 방식으로 임시 키의 섹션을 계산하기 위해 사용된다. NCC은 encryptedkey 값을 얻기 위해서 임시 키 문자열 섹션을 사용하여 세션키와 XOR을 하고, RCST는 세션 키의 값을 얻어 내기 위하여 XOR를 한번 더 하게 된다. 그림 4는 EKE의 흐름도를 나타낸다.

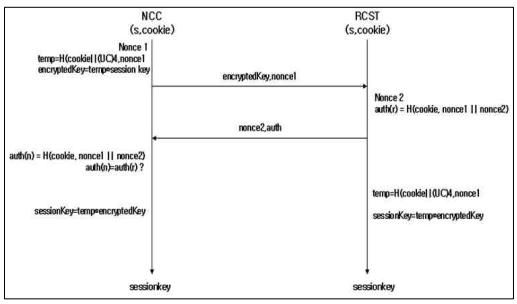

그림 4. Explicit Key Exchange 절차
Fig. 4. Explicit Key Exchange Procedure

EKE는 브로드캐스트를 위하여 사용할 수 있는 프로토콜이다. 이러한 이유는 NCC가 결정한 세션 키를 연결된 RCST 들에게 전달하기 때문이다. QKE와 마찬가지로 갖고 있는 쿠키 값을 이용하여 인증 기능을 제공한다.

3. ETSI 키 교환 프로토콜 성능분석

3.1 메모리 사용량 분석

ETSI 키 교환 프로토콜의 NCC가 저장해야 하는 저장량을 비교하면, MKE 같은 경우 Diffie-Hellman 기반 공개키를 생성하기 위한 소수 a, 소수 q, 생성되는 공개키 Pncc, RCST가 생성하는 공개키 Prcst , NCC가 생성하는 난수 Rncc , RCST가 생성하는 난수 Rrcst , NCC의 인증 값 Hncc , RCST의 인증 값 Hrcst , 비밀 값 s, temp 값 t, 기존 쿠키 값 Cold, 새로 생성되는 쿠키 값 Cnew , Key 값 k 으로 인하여 log(a) + log(q) + log(Pncc) + log(Prcst) + log(Rncc) + log(Rrcst) + log(Hncc) + log(Hrcst) + log(s) + log(t) + log(Cold) + log(Cnew) + log(k) = log(aqPnccPrcstRnccRrcstHnccHrcststColdCnewk) 의 저장량을 필요로 한다.

QKE 같은 경우 비밀 값 s, 쿠키 값 Cold, NCC가 생성하는 난수 Rncc , RCST가 생성하는 난수 Rrcst , NCC의 인증 값 Hncc , RCST가 생성하는 인증 값 Hrcst , temp 값 t, Key 값 k 으로 인하여 log(s) + log(Cold) + log(Rncc) + log(Rrcst) + log(Hncc) + log(Hrcst) + log(t) + log(k) = log(sColdRnccRrcstHnccHrcsttk) 의 저장량을 필요로 한다.

EKE의 경우는 비밀 값 s, 쿠키 값 Cold , NCC가 생성하는 난수 Rncc , RCST가 생성하는 난수 Rrcst , temp 값 t, 비밀 값과 temp값을 이용하여 생성되는 encryptedkey 값 x, NCC의 인증 값 Hncc , RCST의 인증 값 Hrcst , 새로운 비밀 값 e로 인하여 log(s) + log(Cold) + log(Rncc) + log(Rrcst) + log(t) + log(x) + log(Hncc) + log(Hrcst) + log(e) = log(sColdRnccRrcsttxHnccHrcste) 의 저장량을 필요로 한다.

RCST의 관점으로 ETSI 키 교환 프로토콜의 메모리 사용량을 비교하면, MKE의 경우 기존 쿠키 값 Cold , Diffie-Hellman 기반 공개키를 생성하니 위한 소수 a, 소수 q, NCC가 생성하는 공개키 Pncc , RCST가 생성하는 공개키 Prcst , NCC가 생성하는 난수 Rncc , RCST가 생성하는 난수 Rrcst , RCST의 인증 값 Hrcst , 비밀 값 s, temp 값 t, 새로운 쿠키 값 Cnew , Key 값 k 으로 인하여 log(ColdaqPnccPrcstRnccRrcstHrcststCnewk) 의 저장량을 필요로 한다.

QKE 같은 경우 비밀 값 s, 쿠키 값 Cold , NCC가 생성하는 난수 Rncc , RCST가 생성하는 난수 Rrcst , RCST의 인증 값 Hrcst , temp 값

t, Key 값 k로 인하여 log(s) + log(Cold) + log(Rncc) + log(Rrcst) + log(Hrcst) + log(t) + log(k) = log(sColdRnccRrcstHrcsttk) 의 저장량을 필요로 한다.

EKE 의 경우 비밀 값 s, 쿠키 값 Cold , NCC가 생성하는 난수 Rncc , RCST가 생성하는 난수 Rrcst , RCST의 인증 값 Hrcst , temp 값 t, encryptedKey 값 x, 새로운 비밀 값 e로 인하여 log(s) + log(Cold) + log(Rncc) + log(Rrcst) + log(Hrcst) + log(t) + log(x) + log(e) = log(sColdRnccRrcstHrcsttxe) 의 저장량을 필요로 한다. ETSI 기반 키 교환 프로토콜의 메모리 사용량이 표 1에 보여진다[6].

표 1. ETSI 키 교환 프로토콜 메모리 사용량 비교
Table 1. Comparison of Memory Usage Amount

구분	MKE	QKE	EKE
NCC	$\log(aqP_{ncc}P_{rcst}R_{ncc}R_{rcst}H_{ncc}H_{rcst}stC_{old}C_{new}k)$	$\log(sC_{old}R_{ncc}R_{rcst}H_{ncc}H_{rcst}tk)$	$\log(sC_{old}R_{ncc}R_{rcst}txH_{ncc}H_{rcst}e)$
RCST	$\log(aqP_{ncc}P_{rcst}R_{ncc}R_{rcst}H_{rcst}stC_{old}C_{new}k)$	$\log(sC_{old}R_{ncc}R_{rcst}H_{rcst}tk)$	$\log(sC_{old}R_{ncc}R_{rcst}H_{rcst}txe)$

- a : 소수
- q : 소수
- Pncc : NCC의 공개 키
- Prcst : RCST의 공개 키
- Rncc : NCC가 생성하는 난수
- Rrcst : RCST가 생성하는 난수
- Cold : 기존의 쿠키 값
- Cnew : 새로 생성되는 쿠키 값
- Hncc : NCC가 생성하는 인증 값
- Hrcst : RCST가 생성하는 인증 값
- s : 비밀 값
- t : temp 값
- k : key 값
- x : encrypted key
- e : 새로운 비밀 값

3.2 통신량 분석

MKE의 통신량을 살펴보면 Diffie-Hellman 기반 소수 값 a, 소수 값 q, NCC가 생성하는 공개키 Pncc , NCC가 생성하는 난수 Rncc , RCST가 생성하는 공개키 Prcst, RCST가 생성하는 난수 Rrcst , RCST의 인증 값 Hrcst 로 log(a) + log(q) + log(Pncc) + log(Rncc) + log(Prcst) + log(Rrcst) + log(Hrcst) = log(aqPnccRnccPrcstRrcstHrcst) 이다.

QKE의 통신량을 살펴보면 NCC가 생성하는 난수 Rncc, RCST가 생성하는 난수 Rrcst, RCST가 생성하는 인증 값 Hrcst 로 log(Rncc) + log(Rrcst) + log(Hrcst) = log(RnccRrcstHrcst) 이다.

EKE의 통신량은 NCC가 생성하는 난수 Rncc, RCST가 생성하는 난수 Rrcst, RCST가 생성하는 인증 값 Hrcst, encryptedKey x로 log(Rncc) + log(Rrcst) + log(Hrcst) + log(x) = log(RnccRrcstHrcstx) 이다. 다음 표 2는 통신량 비교를 보여준다.

표 2. ETSI 키 교환 프로토콜 통신량 비교
Table 2. Comparison of Communication Amount

구분	MKE	QKE	EKE
NCC-RCST	$\log(aqP_{ncc}R_{ncc}P_{rcst}R_{rcst}H_{rcst})$	$\log(R_{ncc}R_{rcst}H_{rcst})$	$\log(R_{ncc}R_{rcst}H_{rcst}x)$

3.2 연산량 분석

연산량에서 ETSI 키 교환 프로토콜의 NCC가 연산해야 하는 연산량을 비교하면, MKE 프로토콜은 NCC가 생성하는 난수 발생 연산 1회, Modulo 연산 2회, 지수 연산 2회, 해쉬 함수 연산 4회로 r+2m+2e+4h의 연산량을 필요로 한다. QKE 프로토콜의 경우 난수 발생 연산 1회와 해쉬 함수 연산 3로 r+3h의 연산량을 필요로 한다. EKE은 난수 발생 연산 1회, 해쉬 함수 연산 2회, XOR 연산 2회로 r+2h+2x의 연산량을 갖는다.

RCST가 연산해야하는 연산량을 비교하면, MKE 프로토콜은 난수 발생 연산 1회, Modulo 연산 2회, 지수 연산 2회, 해쉬 함수 연산 4회로 r+2m+2e+4h의 연산량을 필요한다.

QKE 프로토콜의 경우 난수 발생 연산 1회와 해쉬 함수 연산 3회로 r+3h의 연산량을 필요로

한다. EKE은 난수 발생 연산 1회, 해쉬 함수 연산 2회, XOR 연산 1회로 r+2h+x의 연산량을 갖는다.

표 3. ETSI 키 교환 프로토콜 연산량 비교
Table 3. Comparison of Calculation Amount

구분	MKE	QKE	EKE
NCC	r+2m+2e+4h	r+3h	r+2h+2x
RCST	r+2m+2e+4h	r+3h	r+2h+x

- r : 난수 발생 연산
- m : Modulo 연산
- e : 지수 연산
- h : 해쉬 연산
- x : Exclusive-OR 연산

4. ETSI 키 교환 프로토콜 성능평가

4.1 시뮬레이션

ETSI 키 교환 프로토콜의 성능평가를 위해 시뮬레이션을 수행하였다. 시뮬레이션은 NS-2를 사용하여 수행되었고, 위성의 환경 옵션은 NS-2 위성 시뮬레이션 설정 시 많이 사용하는 옵션으로 이용하였다. 위성 채널은 기본 제공 Channel/Sat을 따라 설정하였으며 Uplink와 Downlink의 대역폭은 2Mb, Queue의 형식은 Droptail 형식을 사용하였다.

NCC와 RCST의 역할을 위해 2개의 터미널을 설정하였으며, 하나의 터미널은 서울의 위치로 설정하여 위도 33.5°, 경도 126.9°, 다른 터미널은 LA의 위치인 위도 33.7°, 경도 84.4°로 설정하였다.

전송 계층(Transport Layer)는 TCP로 설정하였으며, 응용 계층 프로토콜은 CBR(Constant Bit Rate) 트래픽 타입으로 설정하였다. 에러 모델은 여러 가지 확률 분포(probability distribution)에 따라 패킷에 에러가 발생되도록 설정하였다. MAC과 link 계층 사이에 수신경로에 대한 에러 모델을 추가 하였으며, 위성환경에서 많이 사용되는 설정 값으로 추가하였다.

각각의 키 교환 프로토콜은 총 4단계로 네트워크 전송이 이루어지게 된다. 2개의 터미널에서 데이터 전송 시 중간에 정지 위성을 경유하기 때문에 src 터미널에서 위성, 위성에서 dst 터미널의 과정이 2번씩 있게 된다. 메모리 사용량, 통신량, 그리고 연산량 비교 그래프는 본 논문의 3장에 기술된 ETSI 키 교환 프로토콜 성능분석을 참고하여 수행된 결과를 보여준다. 시뮬레이션을 위한 파라미터의 크기는 ETSI 방식에 따라 결정하였으며, 표 4에 보여진다.

표 4. 성능평가를 위한 파라미터 값
Table 4. Parameter Value for Performance Evaluation

구분	함수명	키 크기(bit)	출력 크기(bit)
키 교환	Diffie-Hellman	512	512
해쉬 알고리즘	HMAC-SHA1	-	160
암호 알고리즘	DES	40	64
의사난수 생성기	-	-	64

4.2 성능 평가 결과

그림5와 같이 MKE의 메모리 사용량은 DIffie-Hellman 에서의 키 생성을 위해 소수 값과 이를 이용하여 생성하는 NCC와 RCST의 공개키를 모두 가지고 있고, 새로운 쿠키 값과 비밀 값을 저장하기 때문에 330Byte로 가장 크고, QKE은 MKE와 다르게 Diffie-Hellman 의 과정이 없기 때문에 소수 및 공개키를 저장하지 않으며, 새로운 쿠키 값을 생성하지 않기 때문에 180Byte 정도로 사용량이 적은 것으로 보인다.

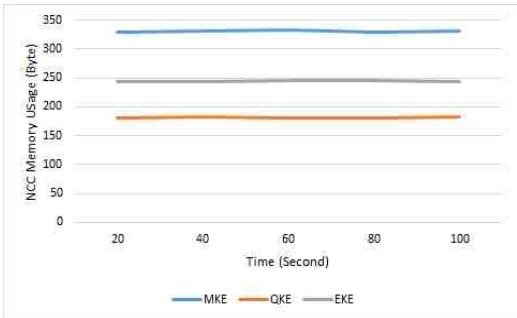

그림 5. NCC의 메모리 사용량 비교
Fig. 5. Comparison of Memory Usage Amount for NCC

RCST의 메모리 사용량 그래프는 그림 6에 보여진다. NCC와 마찬가지로 MKE의 사용량은 Diffie-Hellman를 이용한 공개키 생성 및 공개키 때문에 310 Byte 로 가장 크며, Quick Key

Exchange의 메모리 사용량은 RCST와 마찬가지로 Diffie-Hellman을 사용하지 않기 때문에 160 Byte로 가장 작다.

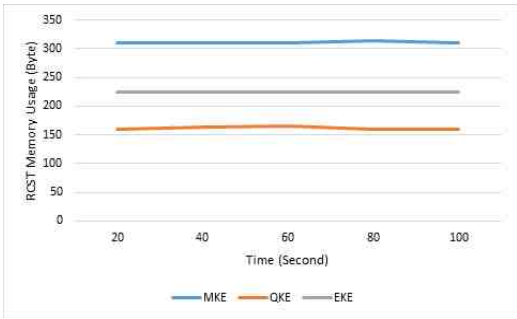

그림 6. RCST의 메모리 사용량 비교
Fig. 6. Comparison of Memory Usage Amount for RCST

통신량에 대한 비교 그래프는 그림 7에서 보듯이, 1000초 까지 키 교환을 했을 경우, MKE는 Diffie-Hellman 과정을 위한 소수 값과 NCC와 RCST의 공개키로 다른 프로토콜에 비해 약 172 KB로 제일 크며, EKE는 QKE와 다르게 encrypted key를 통신하기 때문에 약 113 KB, QKE는 DIffie-Hellman 과정을 하지 않고, encrypted key도 통신하지 않기 때문에 약 74 KB로 보여 진다. 패킷의 수는 QKE가 많지만, 패킷의 크기가 작기 때문에 통신량에서는 반대로 다른 프로토콜들에 비해 적게 보여 진다.

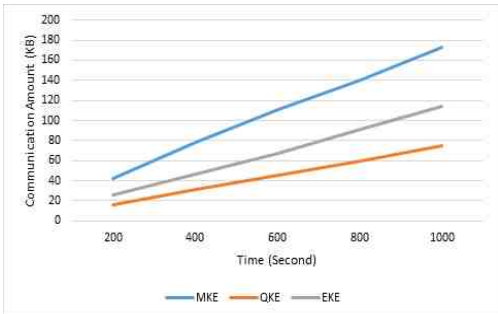

그림 7. 통신량 비교
Fig. 7. Comparison of Communication Amount

연산량에 대한 그래프는 그림 8에서 볼 수 있듯이, 3개의 프로토콜이 큰 차이를 없다. 1000초에서 MKE는 약 9600번, EKE는 약 9300번, QKE는 약 8500번의 연산을 보여준다. 1번의 프로토콜을 시작할 때 연산량만을 볼 경우는 MKE가 Diffie-Hellman의 공개키 생성 과정 때문에 Modulo 연산이 추가되어 가장 많고, QKE는 Modulo 연산이 필요 없기 때문에 제일 작지만, 그림과 같은 결과가 나타나는 이유는 패킷의 전송 시 수행해야하는 연산과 패킷의 전송 빈도 수에 차이이다.

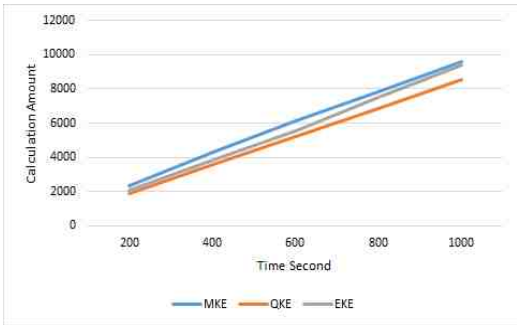

그림 8. 연산량 비교
Fig. 8. Comparison of Calculatin Amount

5. 결론

본 논문에서는 ETSI 표준안에서 지정된 세가지 키 교환 프로토콜을 살펴보고, ETSI 키 교환 프로토콜의 성능 분석과 성능 평가를 통하여 세가 프로토콜의 효율성과 장단점을 제시하였다. 세가지 프로토콜 모두 Diffie-Hellman 알고리즘의 기반을 두고 있지만, Diffie-Hellman 알고리즘의 근본적이 단점인 중간자 공격에 취약하다는 것을 알 수 있다. 또한, 성능평가를 통해 QKE는 메모리 사용량과 통신량, 연산량이 가장 적지만, 시간에 따라 연속적으로 실행할 경우 적은 패킷으로 인하여 통신량과 연산량이 다른 프로토콜에 비해 급격히 증가하여 불안정하다고 볼 수 있다. 하지만 MKE는 메모리 사용량과 통신량, 연산량이 다른 프로토콜에 비해 높지만 연속적으로 프로토콜을 실행할 경우 안정성이 다른 프로토콜에 비해 높다고 할 수 있으며 각각 프로토콜이 수행될 때의 연산량에 따라 Delay Time이 달라지는 것도 볼 수 있다.

본 연구를 기반으로 ETSI 표준 키 교환 프로

토콜의 장단점과 문제점을 파악하는 것이 가능하며, 광대역 위성망에 적합한 새로운 프로토콜을 제안하는데 기여할 수 있다.

REFERENCES

[1] Zhong Yantao and Ma Jianfeng, “A Highly Secure Identity-Based Authenticated Key-Exchange Protocol for Satellite Communication”, JOURNAL OF COMMUNICATIONS AND NETWORKS, VOL. 12, NO 6, 2010

[2] ETSI EN 301 790 V1.2.2 : “Digital Video Broadcasting(DVB) : Interaction channel for satellite distribution systems” 2000.12, European Standard (Telecommunications series).

[3] Sergio PARDO MARTINEZ, “Estudio de algoritmos basados en DAMA para la distribuction de ancho de banda en redes DVB-RCS”, 2010

[4] Heung-Ryong Oh,Heung-Youl Youm, “Key Exchange Protocols for Domestic Broadband Satellite Access Network,” Korea Industrial of Information Security & Cryptology, Vol. 14, No. 3, pp.13-24, 2004.6.

[5] Sam Tran and Tuan Anh Nguyen, “Encryption/Decryption functions for NS-2”, CSCI 5931 Project 2, 2005

[6] Young-seok Lee, “Device Authentication Method for Safe Internet of Things Environments,” Korea Institute of Information, Electronics, and Communication Technology, Vol.8, No.1, pp. 51-58. 22015. 1.

저자약력

이 영 석(Young-Seok Lee) [종신회원]

- 1992년 2월 : 충남대학교 컴퓨터공학과(학사)
- 1994년 2월 : 충남대학교 컴퓨터공학과(석사)
- 2002년 2월 : 충남대학교 컴퓨터공학과(박사)
- 2002년 3월 ~ 2004년 8월 : 한국전자통신연구원 선임연구원
- 2004년 9월 ~ 현재 : 군산대학교 컴퓨터정보통신공학부 교수

<관심분야> 정보보안, 사물인터넷, 이동컴퓨팅

최 훈(Hoon Choi)

- 1983년 2월 : 서울대학교 컴퓨터공학과(학사)
- 1990년 2월 : Duke Univ. 전산학과(석사)
- 1993년 2월 : Duke Univ. 전산학과(박사)
- 1983년 3월 ~ 1996년 8월 : 한국전자통신연구원 선임연구원
- 1996년 9월 ~ 현재 : 충남대학교 컴퓨터공학과 교수

<관심분야> 모바일 컴퓨팅/분산 시스템 미들웨어, 운영체제