JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 20, No. 10 : 1912~1918 Oct. 2016

MPTCP에서 ECDH를 이용한 세션 키 자동생성에 관한 연구

선설희 · 김은기*

The automatic generation of MPTCP session keys using ECDH

Seol-hee Sun · Eun-gi Kim*

Department of Information and Communication Engineering, Hanbat National University, Daejeon 34158, Korea

요 약

MPTCP(Multipath Transmission Control Protocol)는 두 호스트의 연결설정 시, 다수의 TCP 경로를 구성하여 동시에 데이터를 송수신할 수 있다. 따라서 MPTCP는 경로를 추가하려는 호스트의 유효성을 확인하기 위한 인증과정이 필요하기 때문에 초기 연결 설정 시, 키를 교환하여 인증용 토큰을 만든다. 하지만 기존 MPTCP의 토큰은 공개적으로 전송된 키를 그대로 사용하여 생성되기 때문에 MITM(Man In The Middle) 공격에 취약하다. 본 연구에서는 기존 MPTCP 키 교환방식에 ECDH(Elliptic Curve Diffie-Hellman) 키 교환 알고리즘을 적용시켜 기존의 키를 ECDH 공개키로 대체하고, 두 호스트만이 알 수 있는 비밀키를 생성하여 토큰을 만들기 위한 키로 사용하도록 하였다. 또한, 비밀키를 사용하여 데이터의 암호 및 복호화까지 지원하는 방법을 설계하고 구현함으로써 기존 MPTCP에 기밀성을 추가하였다.

ABSTRACT

MPTCP(Multipath Transmission Control Protocol) is able to compose many TCP paths when two hosts connect and the data is able to be transported through these paths simultaneously. When a new path is added, the authentication between both hosts is necessary to check the validity of host. So, MPTCP exchanges a key when initiating an connection and makes a token by using this key for authentication. However the original MPTCP is vulnerable to MITM(Man In The Middle) attacks because the key is transported in clear text. Therefore, we applied a ECDH(Elliptic Curve Diffie-Hellman) key exchange algorithm to original MPTCP and replaced the original key to the ECDH public key. And, by generating the secret key after the public key exchanges, only two hosts is able to make the token using the secret key to add new subflow. Also, we designed and implemented a method supporting encryption and decryption of data using a shared secret key to apply confidentiality to original MPTCP.

키워드 : MPTCP, TCP, ECDH, 네트워크 보안

Key word : MPTCP, TCP, ECDH, Network security

Received 31 August 2016, **Revised** 01 September 2016, **Accepted** 19 September 2016

* **Corresponding Author** Eun-Gi Kim(E-mail:egkim@hanbat.ac.kr, Tel:+82-42-821-1215)
Department of Information and Communication Enginnering, Hanbat National University, Daejeon 34158, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2016.20.10.1912 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

오늘 날, 컴퓨터뿐만 아니라 많은 스마트 기기들은 2개 이상의 네트워크 인터페이스를 갖게 되면서 여러 개의 IP 주소를 갖게 되었다. 하지만, TCP를 사용하는 어플리케이션 레벨의 통신에서 TCP가 엑티브 세션일 때, IP 주소를 바꾸는 것은 불가능 하였다[1]. IETF에서는 이러한 문제를 해결하기 위해 2013년에 MPTCP를 표준화 되었고, MPTCP에서는 기존 TCP에 다수의 경로를 추가하여 데이터를 둘 이상의 경로로 동시에 나누어 전송할 수 있게 하였다[2-4].

호스트들이 다수의 경로를 동시에 사용하기 위해서는 신뢰성이 반드시 뒷받침 되어야 하기 때문에, 인증과정은 필수적이다. 따라서 두 호스트는 경로를 추가할 때 필요한 인증수단인 토큰을 생성하기 위해 초기 연결 설정 시, 키를 교환한다[5].

하지만 기존 MPTCP는 공개적으로 전송된 키를 그대로 사용하여 토큰을 생성하기 때문에, 중간에서 공격자가 호스트들의 키를 알게 된다면 서브플로우를 추가하기 위해 사용되는 인증용 토큰을 똑같이 만들어 낼 수 있다[6]. 따라서 본 연구에서는 기존 MPTCP의 키 교환 방식이 이러한 MITM 공격에 취약하다는 점을 보완하기 위해, ECDH 공개키 교환 알고리즘을 적용하여 기존의 키를 공개키로 대체 하였다. 그리고 두 호스트가 교환한 공개키와 각 호스트에서 갖고 있는 개인키로 만들어진 세션키는 토큰을 만들기 위한 비밀키로 사용되어짐으로써, 공개키를 교환한 두 호스트만이 인증을 위한 토큰을 만들 수 있도록 하였다[7]. 또한 데이터의 기밀성을 보장하기 위해, 공유된 비밀키로 데이터를 암호 및 복호화 할 수 있도록 설계하였고, 이를 리눅스 상에서 구현하고 성능을 분석하였다.

본 논문의 2장에서는 MPTCP, ECDH의 동작과정과 구현 내용에 대하여 자세히 설명하고 3장에서는 성능검증, 4장에서는 결론을 다룬다.

Ⅱ. 본 론

2.1. MPTCP

MPTCP 프로토콜의 계층구조는 하나의 MPTCP 계층아래 다수의 TCP 서브플로우로 구성되고, 이에 따라 IP도 TCP 서브플로우의 개수만큼 나뉜다. 다음 그림 1은 프로토콜의 계층구조를 나타낸다.

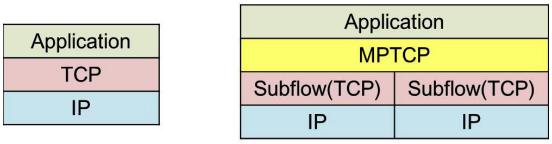

Fig. 1 Comparison of standard TCP and MPTCP stacks

그림1과 같이 MPTCP는 다수의 TCP 서브플로우로 나눠지기 때문에, 연결 설정 과정 또한 TCP 3-way handshakes와 같다. 하지만 MPTCP의 모든 동작은 TCP 옵션으로 표현되기 때문에 SYN, SYN/ACK, ACK 패킷에 MP_CAPABLE 또는 MP_JOIN 옵션이 추가되어진다.

MP_CAPABLE 옵션은 초기 연결 설정 시에만 사용되고, 이 후 서브플로우가 추가 될 때 사용되는 인증용 토큰을 만들기 위한 64비트의 키를 SYN과 SYN/ACK에 각각 포함한다. 단, 연결 설정의 세 번째 ACK를 보내는 호스트는 앞서 교환한 송수신 측의 키를 모두 포함하여 전송한다. 옵션의 구조는 다음 그림 2와 같다.

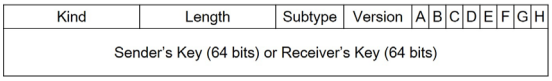

Fig. 2 MP_CAPABLE(Multipath Capable) option format

그림 2에서 A플래그는 체크섬 계산에 대한 여부를 나타내고, B는 확장플래그로서 1로 설정되어지면, C부터 H까지 플래그에 추가로 암호화 알고리즘을 할당 시킬 수 있다. 현재는 H플래그만 HMAC-SHA1로 할당 되어져 있다[5].

MPTCP에서 데이터 전송 시 다수의 경로를 동시에 사용하기 위해서는 서브플로우를 추가해야 한다. 따라서 이미 한 개의 서브플로우가 존재 한다면, 이후 모든 서브플로우는 MP_JOIN 옵션을 포함한 3-way handshakes로 추가될 수 있다. 하지만 서브플로우를 추가하기 위해 인증과정은 필수이기 때문에 MP_JOIN 옵션은 토큰을 포함한다. 토큰은 MP_CAPABLE 옵션에서 협상한 해시 알고리즘 HMAC-SHA1의 결과 값으로, 해시 알고리즘의 입력 값은 초기 연결 설정 시 교환한 수신측의 키로 사용된다. 그림 3은 MP_JOIN의 옵션 구

조를 나타낸다.

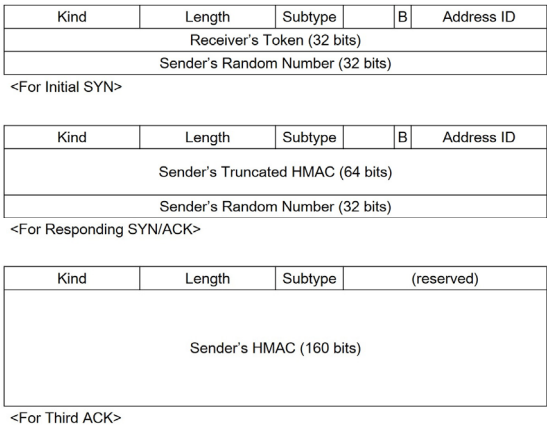

Fig. 3 MP_JOIN(Multipath Join) option format

MP_JOIN 옵션이 포함된 SYN을 수신한 호스트는 자신의 키로 만들어진 토큰이 맞는지 검사 후, 유효하다면 양측의 키와 랜덤 값으로 계산 된 HMAC을 SYN/ACK에 포함시켜 전송한다. 이때, TCP 옵션의 최대 사이즈를 넘지 않기 위해 HMAC은 64비트의 크기만 전송되고 Replay 공격을 예방하기 위한 랜덤 값이 포함된다. 마찬가지로 SYN/ACK를 받은 호스트는 수신한 HMAC의 유효성을 확인 한 후, 새로운 HMAC을 생성하여 ACK를 전송한다. 아래 그림 4는 MPTCP의 초기 연결 설정과 이후 서브플로우를 추가하기 위한 인증 과정을 설명한다[5].

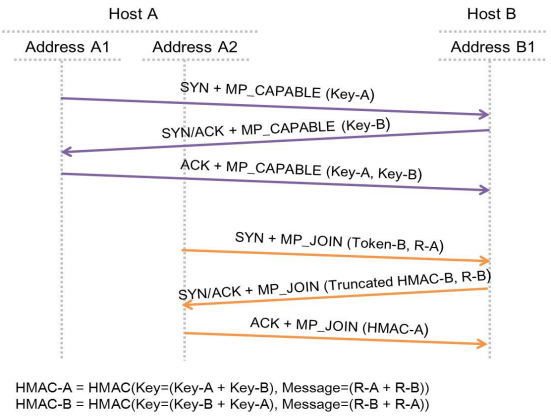

Fig. 4 Initiating an MPTCP connection and associating a new subflow

기존 MPTCP는 서브플로우를 추가 할 때, 호스트의 인증을 위해 토큰을 사용한다. 하지만, 이 토큰은 MP_CAPABLE 옵션에서 공개적으로 전송되어진 키를 그대로 사용하여 만들어 지기 때문에, 초기 연결 설정 시 공격자가 3-way handshakes 패킷에 포함된 키를 알게 되면, 인증수단으로 사용되는 토큰을 똑같이 만들어 낼 수 있다. 따라서 본 연구에서는 MP_CAPABLE에 포함되어 전송된 키를 ECDH 공개키로 대체하고, 이후 양측에서 생성된 세션키가 토큰을 만들기 위한 비밀키로 사용되었다.

2.2. ECDH (Elliptic Curve Diffie Hellman)

ECDH는 타원 곡선 암호화 방식을 적용한 디피 헬만 공개키 교환 알고리즘으로 두 호스트가 보안기능을 제공하지 않는 공개된 채널에서 비밀키를 공유하기 위한 방법이다. 타원 곡선 암호화 방식은 타원곡선 이론에 기반을 둔 공개키 암호방식으로, 줄여서 ECC(Elliptic Curve Cryptography)라 한다.

ECC는 타원 곡선 $y^2 = x^3 + ax + b$라는 식을 만족하는 점들의 집합에서 $4a^3 + 27b^2 \neq 0$의 조건으로 알려진 특정한 점에 대한 무작위 타원 곡선의 이산 로그를 찾는 것이 오래 걸린다는 점에서 착안 되었다. ECC는 다른 암호화 방식에 비해 강력한 보안성을 갖고, 키의 길이가 짧아 암호 및 복호화가 빠르다는 장점이 있다[8].

다음 표 1은 암호화 방식들이 같은 정도의 보안성을 제공하는데 필요한 키 길이를 보여주고 있다[9].

Table. 1 The comparison of key size for cryptography

AES Key Size (Bits)	RSA Key Size (Bits)	ECC Key Size (Bits)	Key Size Ratio (ECC/RSA)
80	1024	160	1:6
128	3072	256	1:12
192	7680	384	1:20
256	15360	512	1:30

본 연구에서는 MPTCP에서 초기 연결설정이 되어있는 두 호스트만이 서브플로우를 추가하기 위한 인증용 토큰을 생성할 수 있도록 ECDH를 적용하였다. 그림 5는 두 호스트가 ECDH 알고리즘을 사용하여 세션키를

공유하는 과정을 나타낸다.

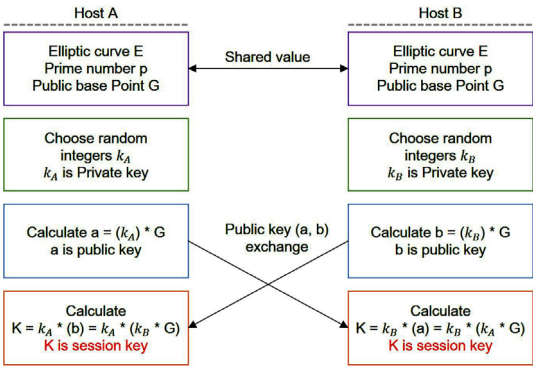

Fig. 5 ECDH key exchange algorithm

위 그림에서 타원곡선 E, 소수 p, 곡선상의 한 점인 G는 두 호스트가 공개적으로 알고 있다. 두 호스트는 각각 임의의 값으로 개인키를 설정하고, G와 곱셈연산을 한다. 이 결과 값은 각 호스트들의 공개키로 사용된다.

본 연구에서는 이 공개키로 초기 연결 설정 시 MP_CAPABLE 옵션에 포함되는 키를 대체하였다. 마지막으로 두 호스트는 공개키 교환 후, 자신의 개인키와 상대방의 공개키를 곱함으로써 양측이 동일한 세션키를 갖게 된다[10]. 본 연구에서는 생성된 세션키를 토큰을 생성하기 위한 비밀키로 사용해, 오직 두 호스트만이 토큰을 생성할 수 있도록 하여 기존 MPTCP의 인증과정을 더욱 안전하게 하였다. 그리고 공유된 비밀키를 사용해 MPTCP 데이터의 암호 및 복호를 할 수 있도록 하여 기존 MPTCP에 기밀성을 추가하였다.

2.3. ECDH를 적용한 MPTCP

MPTCP는 TCP 옵션으로 동작하기 때문에, 기존 MP_CAPABLE 옵션의 키를 ECDH의 공개키로 대체하기 위해서는 TCP옵션의 최대 크기에 대한 고려가 필요하다. 현재 기존 리눅스에서 사용되고 있는 TCP 옵션의 크기는 최대 40바이트 중, MSS(Maximum segment size), TCP SACK Permitted, Timestamps, Window scale의 총 19바이트이다. 이에 따라, 본 연구에서 MP_CAPABLE 옵션이 가질 수 있는 공개키의 길이는 17바이트로 정하였다.

ECDH의 키는 정해진 타원곡선 위의 한 점(x, y)이기 때문에, x 또는 y의 값 중 하나의 값만 알면 전체키를 계산할 수 있다. 이 때, x 또는 y는 압축된 키라 표현하고, 이 키는 계산된 전체 공개키의 값이 홀수 또는 짝수인지를 구분하기 위해 1바이트를 추가로 포함한다. 따라서 실제 압축된 공개키의 크기는 17-1=16바이트로 계산된다. 하지만 MP_CAPABLE 옵션을 포함한 3-way handshakes의 마지막 ACK 패킷은 송수신자의 키를 모두 포함하기 때문에, 8바이트에서 16바이트의 길이로 확장된 키를 모두 포함하게 되면 TCP 옵션의 최대 사이즈를 초과하게 된다. 그러나 실제 리눅스 커널의 MPTCP는 ACK를 수신하는 과정에서 포함된 두개의 키를 별도로 활용하지 않는다. 따라서 본 연구에서는 ACK에 포함되어지는 두 개의 키를 앞서 교환한 공개키의 각각 8바이트 값으로 대체 하였다.

본 연구는 현재 리눅스 커널의 Crypto API에서 ECC를 지원하지 않는 이유로, 어플리케이션 계층에서 Openssl 라이브러리를 사용하여 ECDH 키 생성과 비밀키 계산을 위한 프로그램을 구현하였다[11]. 또한, 이를 Netlink 소켓 API를 사용하여 커널영역과 통신할 수 있도록 하였다. 다음 그림 6은 어플리케이션이 동작하는 유저영역과 커널 영역의 통신과정을 간략히 나타낸다.

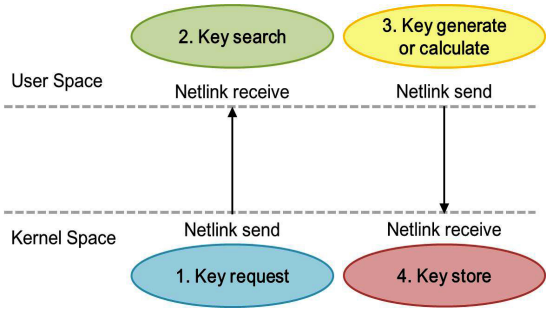

Fig. 6 Communication between user and kernel through netlink socket

어플리케이션의 동작은 커널 영역으로부터 받는 메시지에 따라 수행된다. 우선, 호스트가 SYN 또는 SYN/ACK를 보내기 위해 유저영역으로 키 요청 메시지를 전송하고, 메시지를 수신한 유저영역에서는 키를 생성하여 커널영역으로 전송한다. 그리고 SYN 또는 SYN/ACK에 포함되어진 키를 수신한 호스트는 비밀키를 받기위해 수신한 키와 자신의 키를 함께 유저 영역으로 전송한다. 두 호스트의 키를 모두 받은 어플리케이션은 호스트 키와 관련된 파라메타를 찾는 과정을 거

친 후, 세션키를 계산한다. 마지막으로 비밀키는 세션키의 KDF(Key Derivation Function) 결과 값으로 생성되어진 후에 커널 영역으로 전송된다. 커널 영역에 저장된 비밀키는 서브플로우 추가 시, 토큰을 만들고 MPTCP 데이터를 암호 및 복호화 하는 과정에서 사용된다. MPTCP 데이터의 암호화는 AES 블록 암호화 방식과 CTR 암호화 모드를 사용하였으며, 이는 커널 내에 있는 Crypto API를 사용하여 구현하였다.

Ⅲ. 동작검증 및 성능분석

3.1. 동작검증

본 논문에서는 구현한 MPTCP를 임베디드 보드에 포팅한 후, 와이어 샤크로 패킷을 캡쳐하여 동작을 확인하였다. 그림 7은 MP_CAPABLE 옵션을 포함한 SYN 패킷의 캡처 화면이다.

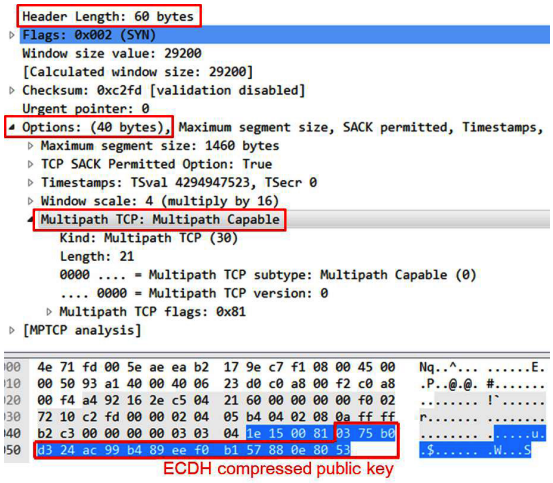

Fig. 7 Capture of SYN including MP_CAPABLE option

그림 7에서 교환되는 17바이트의 ECDH 공개키는 통신하는 각 호스트가 비밀키를 공유하기 위해 사용되고, 비밀키는 서브플로우를 추가하기 위한 인증용 토큰의 생성과 데이터를 암호 및 복호화 하는데 사용된다.

하지만 본 연구에서 데이터의 암호 및 복호화는 커널 내에 있는 Crypto API로 구현되었기 때문에 데이터 패킷을 캡쳐하였을 때, 암호화된 데이터의 내용은 확인할 수 있지만 실제 데이터의 암호 및 복호화 검증은 어플리케이션에서 수신한 데이터를 확인함으로써 이루어질 수 있다. 따라서 본 논문에서는 암호 및 복호화에 대한 동작검증 대신 암호화된 데이터의 전송 RTT(Round Trip Time)를 기존에 데이터 암호화 기능이 없는 MPTCP와 비교하여 성능분석을 진행하였다.

3.2. 성능분석

본 논문에서는 MP_CAPABLE 옵션을 포함한 3-way handshakes에 소요된 시간과 암호화된 데이터를 전송 후, ACK를 받는 RTT시간을 측정하였다. 시간을 측정하기 위해 각각 1000번의 연결 설정과 데이터 전송을 로컬 네트워크 환경에서 실행 하였으며, 사용한 임베디드 보드의 규격은 표 2와 같다.

Table. 2 The specification of test board

CPU	ARMv6-compatible processor rev 6 (v6l)
Memory	1 Gbyte
Kernel version	Linux 3.14.33

그림 8은 기존 MPTCP와 본 연구에서 제안한 MPTCP의 MP_CAPABLE을 포함한 3-way handshakes 과정에서 소요된 시간을 그래프로 나타낸다.

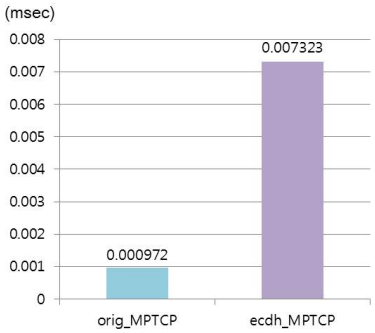

Fig. 8 Time of 3-way handshakes with MP_CAPABLE option

본 논문에서 구현한 MPTCP는 유저 영역에서의 키 생성 시간과 커널 영역과의 Netlink 통신으로 인한 지연으로, 기존 MPTCP의 연결 설정 시간과 약 7배의 속도 차이가 생긴 것을 확인할 수 있다.

다음 그림 9는 데이터를 암호화 시켜 전송한 후, 이를 수신측에서 복호화 한 후 보낸 ACK를 수신 할 때까지

의 RTT를 측정한 결과이다. 데이터는 본 논문에서 사용된 MSS인 1460바이트를 넘지 않도록 하여 16, 816, 1424바이트의 크기로 각각 1000번 전송하였다.

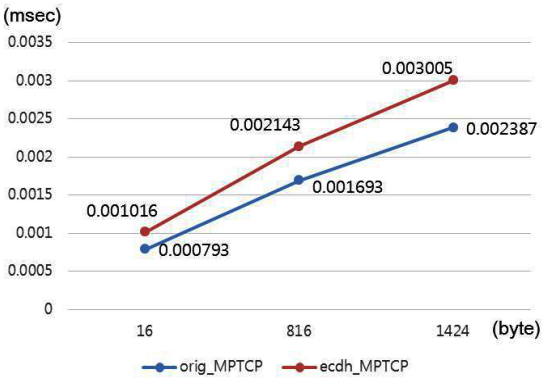

Fig. 9 Data transmission RTT of MPTCP

그림 9의 그래프에서 데이터 전송 RTT는 암호 및 복호화에서 소요되는 시간으로 인해 기존 MPTCP보다 증가한 것을 확인할 수 있다. 또한, 데이터의 암호 방식은 128비트 블록의 단위로 암호 하는 AES를 사용하였기 때문에 데이터의 크기가 커짐에 따라 AES 암호 블록의 개수가 각각 1, 51, 89개로 증가하면서 기존 MPTCP에서 소요된 시간과의 차이가 더 커진 것을 볼 수 있다.

Ⅳ. 결 론

본 연구에서는 기존 MP_CAPABLE 옵션에 포함된 호스트들의 키를 ECDH의 공개키로 대체하여, 두 호스트가 비밀키를 공유할 수 있도록 하였다. 따라서 초기 연결설정 이후 서브플로우를 추가하기 위한 인증 수단인 토큰을 공유된 비밀키로 생성함으로써, 오직 두 호스트만이 서브플로우를 추가할 수 있도록 기존 MPTCP의 인증방식을 보완하였다.

또한, MPTCP에 기밀성을 추가하기 위해 공유된 비밀키를 사용하여 데이터의 암호 및 복호화를 할 수 있도록 하였다. 이로 인해 초기 연결설정 과정과 데이터 전송속도의 시간이 기존 MPTCP보다 증가한 것을 확인할 수 있었다.

추후에는 기존 MPTCP에서 전체 서브플로우의 연결을 종료시키는 MP_FASTCLOSE 옵션이 포함하고 있는 키를 호스트가 공유한 비밀키로 서명함으로서 스푸핑 공격을 예방할 수 있는 방법을 적용시킬 계획이다.

ACKNOWLEDGMENTS

This research was financially supported by the Ministry of Trade, Industry and Energy (MOTIE) through the Promoting Regional specialized Industry (No. R0003847)

REFERENCES

[1] G. Huston, “IP Multi-Addressing and Multipath TCP,” *The Internet Protocol Journal*, vol. 18, no. 2, pp. 2-12, June 2015.

[2] WIKIPEDIA. Multipath TCP [Internet]. Available: http://en.wikipedia.org/wiki/Multipath_TCP.

[3] H. E, Go, J. U. Lee, S. H. Back, and J. H. Hwang “Multipath TCP (MPTCP) standardization and technology development trends,” *Journal of The Korean Institute of Communication Sciences*, vol. 31, no. 9, pp. 9-16, September 2014.

[4] L. T. Tuan, K. S. Kim, J. K. Choe, and S. H. Ro, “A Study on Multi-Path TCP Mobility Management Protocol,” *Journal of KIIT*, vol. 12, no. 6, pp. 109-117, June 2014.

[5] A. Ford, C. Raiciu, M. Handley, *TCP Extensions for Multipath Operation with Multipath Addresses*, RFC 6824, IETF, 2013.

[6] M. Bagnulo, *Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses*, RFC 6181, IETF, 2011.

[7] S. H. Sun, E. G. Kim, “A study on the Key Exchange Using ECDH in MPTCP,” in *Proceeding of the 4th Annual Conference on Engineering and Information Technology*, Kyoto, Japan, pp. 84-89, March 2016.

[8] K. J. Ha, C. H. Seo, D. Y. Kim, “Design of Validation System for a Crypto-Algorithm Implementation,” *Journal of the Korea Information and Communication Society*, vol. 39, no. 4, pp. 242-250, April 2014.

[9] BlueKrypt. (2012). NIST Report on Cryptographic Key

Length and Cryptoperiod [Internet].
Available: http://www.keylength.com/en/4/.

[10] D. K. Too, S. J. Han, "A Study of Key Distribution for Security on VANET," *Journal of the Korea Institute of Information and Communication Engineering*, vol. 16, no. 10, pp. 2192-2198, October 2012.

[11] OpenSSLWiki. Elliptic curve Diffie Hellman [Internet].
Available: http://wiki.openssl.org/index.php/Ellipttic_Curve_Diffie_Hellman.

선설희(Seol-Hee Sun)

2015년 2월 : 한밭대학교 정보통신공학과 (정보통신공학 학사)
2015년 3월 ~ 현재 : 한밭대학교 정보통신전문대학원 석사과정
※관심분야 : 네트워크, MPTCP, 임베디드 S/W, 네트워크 보안

김은기(Eun-Gi Kim)

1989년 2월 : 고려대학교 대학원 전자공학과 (전자공학 석사)
1994년 2월 : 고려대학교 대학원 전자공학과 (전자공학 박사)
1995년 2월 ~ 현재 : 한밭대학교 정보통신공학과 교수
※관심분야 : 컴퓨터 네트워크, 임베디드 S/W, 암호화, 네트워크 보안