Journal of the Korea Academia-Industrial cooperation Society
Vol. 15, No. 5 pp. 3086-3092, 2014

http://dx.doi.org/10.5762/KAIS.2014.15.5.3086
ISSN 1975-4701 / eISSN 2288-4688

보안 기능을 지원하는 파일 전송 프로토콜의 설계 및 구현

안재원[1], 최범진[1*], 옥성진[1], 강정하[1], 김재영[2], 김은기[1*]
[1]한밭대학교 정보통신학과, [2]한국전자통신원구원 융합기술연구소

Design and implementation of file transfer protocol supporting security functionalities

Jae-Won Ahn[1], Beom-Jin Choi[1*], Sung-Jin Ok[1], Jung-Ha Kang[1], Jae-Young Kim[2], and Eun-Gi Kim[1*]
[1]Dept. of Information and Communication Engineering, Hanbat National University
[2]IT Convergence Technology Research Lab., Electronics and Telecommunications Research Institute

요 약 FTP는 한 호스트에서 다른 호스트로 파일을 전송하기 위한 프로토콜로써, 데이터를 평문으로 전송하기 때문에 기밀성이 보장되지 않는다. 현재 보안 기능이 제공되는 FTP로 FTPS와 SFTP가 있다. FTPS는 SSL/TLS 암호화 프로토콜 기반에서 동작하는 FTP이다. SFTP는 SSH를 통해 파일을 전송하는 프로토콜이다. 따라서 FTPS는 SSL/TLS, SFTP는 SSH와 같은 추가적인 시스템이 반드시 필요한 단점이 있다. 본 논문에서는 추가적인 암호화 시스템 없이 FTP 내에서 보안 기능을 지원하는 Secured FTP를 제안하였다. Secured FTP는 FTP 내에서 Diffie-Hellman 알고리즘을 이용하여 공유된 비밀 키를 생성하고 AES-Counter 알고리즘을 이용하여 FTP 데이터를 암호화 및 복호화 하도록 설계되었다. Secured FTP를 Linux 운영체제에서 구현하였고 시험을 통하여 비밀 키가 정상적으로 교환되고 FTP 데이터가 암호화되어 전송되는 것을 확인하였다.

Abstract The FTP that provides file transfer capabilities to/from another station cannot provides data confidentialities. The FTPS and SFTP can support a security functionalities. The FTPS needs a SSL layer and SFTP use a functions of SSH. And therefore the FTPS or SFTP needs an additional modules such as SSL or SSH. In this paper, we propose a new Secured FTP protocol that can support the security functions without extra security system. The Secured FTP uses Diffie-Hellman key agreement algorithm for shared secret key generation and AES-Counter algorithm for data encryption algorithm. Our designed Secured FTP is implemented in Linux environments and the proper operations of implemented Secured FTP is verified.

Key Words : AES, Confidentiality, Diffie-Hellman key exchange, FTP, Security

1. 서론

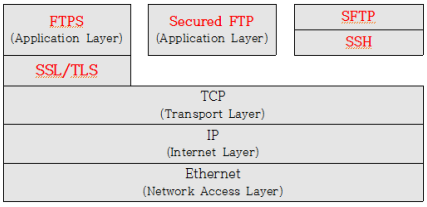
최근에 많은 사람들은 인터넷을 통해 수많은 데이터를 주고받고 있다. 응용 데이터의 종류에 따라서 데이터를 송수신할 수 있는 다양한 응용 프로토콜이 존재한다.

FTP(File Transfer Protocol)는 파일을 전송하는 대표적인 파일 전송 프로토콜로써 대용량의 파일을 송수신할 때 많이 사용되고 있다. FTP는 파일 송수신만을 위해 설계된 프로토콜이기 때문에 동작 방식이 매우 단순하다[1]. 이러한 단순한 프로토콜 구조로 인해 보안상 취약한 점이 있다[2]. 공격자가 FTP를 통해 전송되는 데이터를 가로챌 때 데이터의 정보가 그대로 노출된다. 한 예로,

본 연구는 교육부와 한국연구재단의 지역혁신인력양성사업(No. 201301590001) 및 국토교통부 물관리연구사업의 연구비지원(12기술혁신C01)에 의해 수행된 연구결과임.
*Corresponding Author : Beom-Jin Choi(Hanbat National Univ.)
Tel: +82-10-9929-2138 email: bj20428@naver.com
Received January 15, 2014 Revised February 14, 2014 Accepted May 8, 2014

FTP 사용자의 로그인 정보인 사용자 계정(Identification)과 비밀번호(Password)가 그대로 노출된다. 이는 보안상 많은 문제점들을 발생시킬 수 있다. 이러한 취약점을 보완하기 위해 제안된 것이 FTPS(FTP over SSL)와 SFTP(SSH FTP)이다. FTPS는 보안 기능을 제공하기 위하여 SSL/TLS(Secure Sockets Layer/Transport Layer Security) 암호화 프로토콜 기반에서 동작하도록 설계된 FTP이다[3]. SFTP는 SSH(Secure Shell)를 통해 동작하는 FTP이다[4]. 본 논문에서는 SSL/TLS와 SSH 같은 추가적인 시스템 없이 FTP 자체에서 보안기능을 지원하는 Secured FTP를 제안한다. Secured FTP는 FTP 내에서 Diffie-Hellman 알고리즘을 이용하여 양측만이 알 수 있는 공유된 비밀 키를 생성하고 AES(Advanced Encryption Standard)-Counter 알고리즘을 이용하여 FTP 데이터를 암호화 및 복호화 하도록 설계되었다.

FTPS, SFTP, 그리고 Secured FTP의 계층 구조를 Fig. 1에 나타냈다.



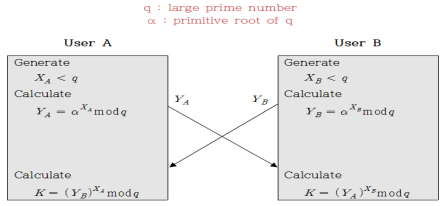
[Fig. 1] Hierarchical structure of FTPS, SFTP, and Secured FTP

본 논문의 구성은 다음과 같다. 2장에서는 Diffie-Hallman 알고리즘, AES 알고리즘, 그리고 Counter Mode에 대하여 기술하였다. 3장에서는 Secured FTP의 설계 및 구현에 대하여 기술하였다. 4장에서는 Secured FTP 동작 검증과 실험 및 평가에 관한 내용을 기술하였다. 마지막으로 5장에서는 결론을 다룬다.

2. 관련연구

2.1 Diffie-Hellman 알고리즘

Diffie-Hellman 알고리즘은 두 종단이 안전하지 않은 채널을 통해 공통의 키를 생성할 수 있는 알고리즘이다. 즉, Diffie-Hellman 키 교환을 통해 두 종단에서만 사용할 수 있는 비밀 키를 생성할 수 있다. 두 종단이 Diffie-Hellman 키 교환을 하기 위하여 p, α를 공유해야 한다. p는 큰 소수이고 α는 p의 원시근이다. 각 종단에서 p보다 작은 임의의 키를 생성한다. 이는 각 종단의 개인 키다. 각 종단은 p, α, 개인 키를 이용하여 각자의 공개 키를 생성한다. 이후 각 종단은 공개 키를 서로에게 전송해주고 상대방의 공개 키와 자신의 개인 키, p를 이용해 비밀 키를 생성한다[5]. 아래의 Fig. 2는 Diffie-Hellman 알고리즘의 키 교환을 나타낸다.

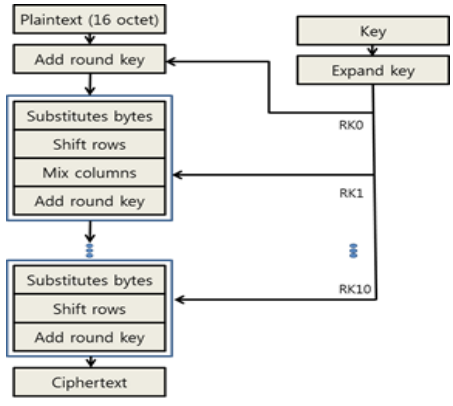


[Fig. 2] Diffie-Hellman key exchange

Fig. 2에서 X는 p보다 작은 임의의 개인 키이고 Y는 공개 키이다. K는 공개 키를 교환한 후 이를 이용해 만드는 비밀 키이다. 이 비밀 키는 AES 알고리즘의 키로 사용된다.

2.2 AES 알고리즘

AES는 국제 표준 대칭 키 암호 알고리즘으로서 128, 192, 256비트 길이의 키를 사용하여 평문(Plaintext)을 16바이트 블록 단위로 나누어 암호화한다. 암호화를 수행하는 과정은 Fig. 3과 같다.



[Fig. 3] AES algorithm

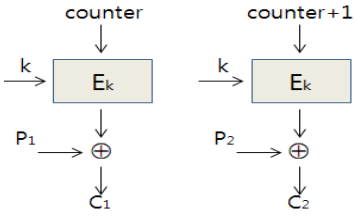
AES는 128, 192, 256비트의 키의 길이에 따라 각각 10, 12, 14라운드를 수행한다. AES의 각 라운드는 Substitutes Bytes, Shift Rows, Mix Columns, Add Round Key 순으로 이루어져 있다[1].

- ◆ Substitutes bytes: 각 바이트를 S-Box를 이용해 치환
- ◆ Shift rows: 행 단위로 왼쪽 순환 시프트
- ◆ Mix columns: 열(column) 단위로 행렬과 곱
- ◆ Add round key: 각 라운드 키와 XOR

각 라운드가 수행되는 동안 16바이트의 입력 메시지는 4x4 형태의 바이트 단위로 처리된다. 모든 라운드가 끝나면 16바이트의 암호문(Ciphertext)을 얻을 수 있다[1].

2.3 Counter Mode

Counter Mode에서는 1씩 증가하는 카운터를 암호화한 비트열과 평문 블록(P)과의 XOR을 취한 결과가 암호문 블록(C)이 된다. Counter Mode를 사용한 암호화 과정은 Fig. 4와 같다.

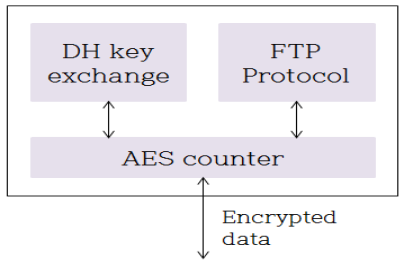


[Fig. 4] Operation of AES Counter Mode

3. 본론

3.1 Secured FTP 설계

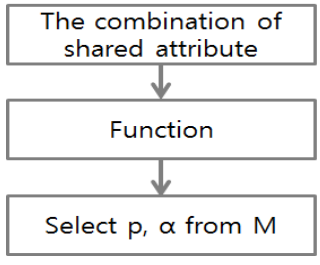
FTP 서버와 클라이언트는 네트워크상에서 데이터를 송수신한다. 연결 설정 과정에서 클라이언트가 서버에게 연결 요청을 하면 Diffie-Hellman 키 교환 방식을 통하여 암호화에 사용할 키를 생성한다. 키를 생성한 이후에 모든 데이터는 AES-counter 모듈을 통과하여 암호화되어 송수신된다. AES-counter 모듈은 송신하는 모든 FTP 데이터를 암호화하고 수신되는 모든 FTP 데이터를 복호화한다. AES-counter는 암호화와 복호화가 같은 구조로 이루어져 있다. 따라서 데이터 송수신 시에 같은 AES-counter 모듈을 사용한다. Fig. 5는 Diffie-Hellman 키 교환 알고리즘과 AES-counter 암호화 알고리즘을 접목한 Secured FTP 서버와 클라이언트의 블록도이다.



[Fig. 5] Block diagram of Secured FTP

Diffie-Hellman 키 교환 시에 서버와 클라이언트는 동일한 p, α를 가지고 있어야 한다. 서버와 클라이언트가 같은 p, α를 가지려면 한쪽 종단에서 p, α를 생성하여 다른 종단으로 전송하는 방법이 있다.

p는 매우 큰 소수이기 때문에 p를 생성할 때 많은 시간이 걸린다. 이 문제점을 해결하기 위해 서버와 클라이언트는 공통으로 인식할 수 있는 공유속성을 사용하여 각각 p를 생성한다. 공유속성은 서버와 클라이언트의 IP 주소, Port 번호 등이다. 서버와 클라이언트가 공유속성을 이용하여 p, α를 생성하기 위한 전체적인 구조는 Fig. 6와 같다.

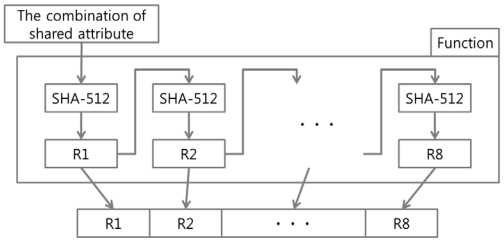


[Fig. 6] Whole structure for a creation of p, α

공유속성의 조합(The combination of shared attribute)에서는 공유속성을 연결하는 작업을 한다. 예를 들면, 통신을 하는 양쪽 단의 IP 주소, Port 번호를 연결하는 것이다.

Function에서는 공유속성의 조합을 입력으로 받고 이

것을 통해 최대 4096비트 길이의 결과 값을 내는 작업을 한다. Fig. 7는 Function의 동작을 나타낸다.

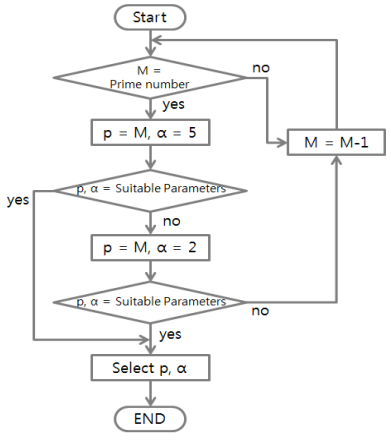


[Fig. 7] Operation of Function

Function의 동작은 최초에 공유속성의 조합을 입력으로 받고 이 입력이 해시 함수(SHA-512)의 입력으로 들어와 512비트 길이의 결과를 출력한다. 이 값을 R1이라고 한다. R1을 다시 해시 함수의 입력으로 넣고 512비트 길이의 결과를 출력한다. 이 값을 R2라고 한다. R1 뒤에 R2를 연결한다. 위의 과정을 R8이 나올 때까지 반복하며 최종적으로 4096비트 길이의 결과를 얻게 된다.

4096비트의 왼쪽부터 m비트를 선택하고 이를 M이라고 한다. m은 p의 비트수를 의미한다.

M에서 p, α를 선택하는 알고리즘은 Fig. 8에 나타냈다.

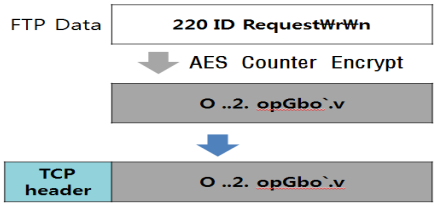


[Fig. 8] Algorithm for selecting p, α from M

먼저 M이 소수인지를 판단한다. M이 소수가 아니면 M을 1씩 감소시키는 연산을 계속한다. M이 소수라면 p=M, α=5로 할당한다. p, α가 Diffie-Hellman 파라미터로 적절하다고 판단되면 이 p, α를 선택한다. 반대로 p, α가 Diffie-Hellman 파라미터로 적절하지 않다면 α=2로 할당한다. p, α가 Diffie-Hellman 파라미터로 적절하다고 판단되면 이 p, α를 선택한다. 그렇지 않으면 M에서 1을 빼는 연산을 수행하여 M이 소수인지를 판단하는 과정부터 다시 시작한다.

서버와 클라이언트는 위의 과정을 거쳐 선택된 동일한 p, α를 사용하여 공개키를 생성한다. 서버와 클라이언트는 생성된 공개키를 상대방에게 전송해주고 암호화에 사용할 대칭적인 비밀 키를 생성한다.

클라이언트와 서버 간에 비밀 키가 생성되면 이후 전송되는 패킷들은 암호화되어 송수신된다. 평문 형식의 FTP 데이터가 암호화되어 전송되는 과정을 Fig. 9에 나타냈다.



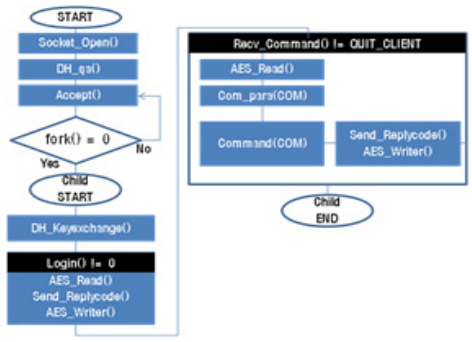
[Fig. 9] Encrypted FTP data

FTP에서 전송되는 응답 코드를 포함한 모든 데이터는 TCP 데이터가 된다. 서버와 클라이언트 간에 패킷 송수신 시 FTP 데이터 전체가 암호화되고 TCP 헤더가 추가되어 전송된다. 이는 임의의 공격자가 서버와 클라이언트 사이에서 송수신되는 패킷을 가로채도 FTP 데이터가 모두 암호화되어 있기 때문에 내용을 알아볼 수 없게 된다.

3.2 Secured FTP 서버-클라이언트 구현

본 논문에서 설계된 Secured FTP를 Linux OS 기반으로 구현하였다. Fig. 5의 Secured FTP의 블록도에서 Diffie-Hellman 키 교환 기능과 AES-Counter 기능을 구현하였고 FTP 프로토콜이 정의된 RFC 959를 참고하여 암호화 관련 기능과 연동될 수 있도록 FTP 기능을 구현하였다.

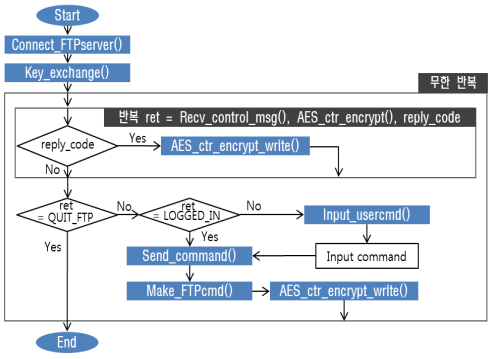
Fig. 10는 구현된 Secured FTP 서버 프로그램의 동작을 나타낸다.



[Fig. 10] Operation of Secured FTP server

- Socket_Open():제어연결에 사용될 Port Open
- DH_qa(): q와 α 생성
- DH_Keyexchange(): Diffie-Hellman 키 교환 수행
- Login(): 로그인 동작
- AES_Read(): read(2)한 데이터 복호화
- AES_Write():데이터 암호화 후 write(2) 수행
- Send_Replycode(): 응답코드 전송
- Recv_Command(): Client로부터 명령어 수신
- Com_Para(): 명령어 해석
- Command(): 명령어 수행

Fig. 11은 구현된 Secured FTP 클라이언트 프로그램의 동작을 나타낸다.



[Fig. 11] Operation of Secured FTP server

- Connect_FTPserver(): 서버로 연결 요청
- Key_exchange(): Diffie-Hellman 키 교환 수행
- Recv_control_msg(): 서버의 응답 수신, 응답 및 반환값에 따라 적절한 동작 수행
- AES_ctr_encrypt(): 암호화된 서버의 응답 복호화
- AES_ctr_encrypt_write(): 데이터 암호화 및 전송, 암호화된 데이터 수신 후 데이터 복호화 및 출력
- Input_usercmd(): 명령어 입력
- Send_command(): 명령어 송신
- Make_FTPcmd(): 사용자 명령어를 FTP 명령어로 변환

4. 동작 검증 및 고찰

4.1 Secured FTP 동작 검증

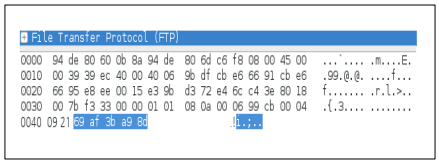
네트워크상에서 Secured FTP가 전송하는 모든 데이터들은 암호화되기 때문에 외부로 데이터의 정보가 노출되더라도 데이터의 기밀성이 보장된다.

Fig. 12은 클라이언트 콘솔 창에서 명령어를 암호화하여 서버에게 전송하고 서버로부터 명령어의 결과를 수신한 화면을 나타낸 것이다.

```
myftp> pwd
comm=[pwd] para=[]
Encrypted message is :          [69] [af] [3b] [a9] [8d]
200 OK /home/ahnjae/Sfrtp9
```

[Fig. 12] Terminal of Secured FTP client

Fig. 13는 스니핑 프로그램인 WireShark를 이용하여 Fig. 12에서 전송한 패킷을 스니핑한 화면이다.



[Fig. 13] Sniffing packet

Fig. 12에서 클라이언트는 pwd\r\n을 AES-counter 모듈을 통과시켜 얻은 암호화된 값 [69] [af] [3b] [a9] [8d]을 서버로 전송한다. [Fig. 13]를 통해 암호화된 값이 전송되었음을 확인할 수 있다. 서버는 수신된 [69] [af] [3b] [a9] [8d]을 AES-counter 모듈로 통과시켜 다시 pwd\r\n 값을 얻어내고 pwd 명령의 결과를

AES-counter 모듈을 통과시켜 암호화하여 클라이언트로 전송한다.

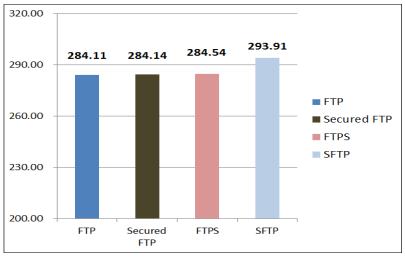
4.2 실험 및 평가

FTP, SFTP, FTPS, 그리고 Secured FTP의 성능을 비교한다. 서버와 클라이언트를 동작시키기 위해서 2대의 PC를 사용했다. Table 1는 서버와 클라이언트를 동작시킨 PC들의 스펙을 보여준다.

[Table 1] Test system information

	Server	Client
OS	Fedora18	Fedora19
Processor	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHZ	
Memory	8GB	

위의 환경에서 FTP, SFTP, FTPS, 그리고 Secured FTP를 동작시켜 데이터 전송 속도를 측정했다. [Fig. 14]는 각 클라이언트가 서버로부터 300MB의 데이터를 다운로드 받은 평균 시간을 측정한 것이다.



[Fig. 14] Average data transfer rate

SFTP는 내부적으로 여러 개의 부계층(sublayer)으로 구성되어지며, 하나의 클라이언트가 실행될 때 클라이언트 프로세스(process)와 SSH 클라이언트 프로세스가 생성되어, 전송되는 데이터가 두 개의 프로세스를 경유하기 때문에 데이터 전송 속도가 더 느린 것을 알 수 있다.

기존 FTP는 암호화 과정이 없기 때문에 다른 FTP들에 비해 속도 면에서 조금 더 빠르다는 것을 알 수 있다. 그 다음으로 Secured FTP, FPTS 순으로 데이터 전송 속도가 빠른 것을 확인했다. 그러나 기존 FTP의 데이터 전송 속도와 큰 차이가 나지는 않는다. 암호화를 하는 과정에서 처리 시간을 많이 소요하지 않기 때문이다.

구현한 Secured FTP가 다른 FTP와 전송 속도는 크게 차이나지는 않지만 프로그램의 이식성과 가벼운 무게를 갖는 장점이 있다. FTPS는 SSL/TLS가 반드시 필요하고 SFTP는 SSH가 지원되는 환경에서만 가능하다. 하지만 본 논문에서 제안한 Secured FTP는 그 프로그램 자체만으로 암호화까지 지원해 주기 때문에 프로그램의 이식성이 높은 동시에 프로그램이 가볍다.

5. 결론

본 논문은 FTP가 데이터를 평문으로 전송하여 데이터의 기밀성이 보장되지 않는 점을 보완하고자 하였다. 추가적인 시스템을 사용하지 않고 FTP 자체에서 Diffie-Hellman 알고리즘을 이용하여 공유된 비밀 키를 생성하고 AES-Counter 암호화를 지원해 주는 Secured FTP를 제안하였다. Diffie-Hellman 알고리즘은 서버와 클라이언트가 서로의 공개 키를 교환하도록 한다. 이를 통해 둘만이 알 수 있는 암호화에 사용할 비밀 키를 생성한다. AES-counter 암호화 모듈은 키 교환을 통하여 생성된 비밀 키를 사용하여 FTP에서 전송하는 모든 데이터를 암호화한다. 이를 통해 FTP 데이터의 기밀성을 보장할 수 있다.

본 논문에서 제안한 Secured FTP는 SSL/TLS와 SSH 같은 추가적인 시스템을 사용하지 않고 FTP 내부에서 암호화 기능을 지원하기 때문에 프로그램의 이식성이 높고 무게가 가벼운 장점이 있다.

Reference

[1] J. Postel, J. Reynolds "File Transfer Protocol (FTP)", RFC 959, IETF, October 1985

[2] Behrouz A. Forouzan, “TCP/IP Protocol Suite”, 4rd Ed., p.643, McGraw Hill, 2007

[3] Rolf oppliger, “SSL and TLS Theory and Practice”, pp.75-79, ARTECH HOUSE, 2009

[4] Micheal stahnke, “Pro OpenSSH”, p.84, Apress, 2006

[5] Behrouz A. Forouzan, Sophia Chung Fegan, "Data Communications and Networking", 4rd Ed., pp.943-945, 952-954, McGraw Hill, 2007

안 재 원(Jae-Won Ahn) **[준회원]**

- 2014년 2월 : 한밭대학교 정보통신공학과 (정보통신공학 학사)

<관심분야>
컴퓨터 네트워크, 암호화, 네트워크 보안

강 정 하(Jung-Ha Kang) **[정회원]**

- 2001년 8월 : 한밭대학교 정보통신공학과 (정보통신공학 석사)
- 2002년 1월 ~ 2012년 4월 : 휴메이트 책임연구원
- 2005년 8월 ~ 현재 : 한밭대학교 정보통신공학과 (정보통신공학 박사 과정)

<관심분야>
컴퓨터 네트워크, 무선통신, 암호화, 네트워크 보안

최 범 진(Beom-Jin Choi) **[준회원]**

- 2014년 2월 : 한밭대학교 정보통신공학과 (정보통신공학 학사)

<관심분야>
컴퓨터 네트워크, 암호화, 네트워크 보안

김 재 영(Jae-Young Kim) **[정회원]**

- 1992년 2월 : 연세대학교 대학원 전자공학과 (전자공학 석사)
- 1996년 8월 : 연세대학교 대학원 전자공학과 (전자공학 박사)
- 1996년 9월 ~ 1999년 2월 : (주) 대우전자
- 1999년 3월 ~ 현재 : 한국전자 통신연구원 책임연구원

<관심분야>
에너지IT, 암호화, 센서무선통신

옥 성 진(Sung-Jin Ok) **[준회원]**

- 2014년 2월 : 한밭대학교 정보통신공학과 (정보통신공학 석사)

<관심분야>
컴퓨터 네트워크, 암호화, 네트워크 보안

김 은 기(Eun-Gi Kim) **[정회원]**

- 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