Miracl 라이브러리를 이용한 안전한 1대1 채팅

*이상철 류다은 김수환 김언동 박요한
계명대학교
*zktm9903@gmail.com

Safe one-on-one chat using Miracl library

*Lee, Sang-Cheol Ryu, Daeun Kim, Su-Hwan Kim, Un-Dong Park, Yo-Han
Keimyung University

요약

최근 코로나 사태가 지속되면서 온라인으로 활동하는 경우가 많아졌다. 비대면 채팅, 메신저 사용이 늘면서 개인정보 유출 등 여러 이슈가 발생하고 있다. 이에 따라 정보 보안에 대한 관심이 높아지는 양상을 보인다. 시중에 존재하는 다수의 채팅 서비스들은 대화 내용을 서버 DB에 저장한다. 이러한 방식은 누군가가 다른 사람의 대화 내용에 접근할 수 있다는 가능성을 의미한다. 따라서 서버 DB에 데이터가 남지 않는 소켓 통신 암호화 채팅을 고안하였다. 그 외에도 보안 요소를 추가하기 위하여 외부 라이브러리를 사용하였다. 본 논문에서는 Miracl 라이브러리를 사용하여 안전한 키 교환을 위한 Diffie-Hellman 알고리즘과 평문을 암호화하기 위한 AES 알고리즘을 적용한 1대1 채팅을 제안하고자 한다.

1. 서론

코로나 사태의 연장으로 비대면 교육에 대한 수요가 늘며 메신저, 채팅 프로그램의 수요도 높아졌다. 채팅 프로그램 사용의 증가량은 개인정보 유출 이슈를 발생시키며 보안에 대한 관심도가 늘어나는 양상을 보인다. 2006년 정보보호 실태조사 결과 중 '정보보호 인식 부문'에서 정보시스템 사용자인 응답자의 대부분인 약 98.2% (매우 중요 60.5%, 중요한 편 37.7%)가 정보보호의 중요성을 인식하고 있었다. 정보보호의 인식이 높게 나타난 것은 단순히 당위적인 측면만이 아닌 정보화 역기능에 대한 두려움과 결부된 실질적인 관심인 것으로 나타났다 [1].

채팅 프로그램은 사용자 간 대화 내용의 보안성이 보장되어야 하기 때문에 암호화 알고리즘 선택과 안전한 키 생성, 교환은 중요한 요소가 된다.

본 논문에서는 사용자가 자신의 채팅이 암호화되는 것과 암호화에 필요한 키가 생성되고 교환되는 과정을 직접 확인할 수 있는 1대1 채팅 프로그램을 구현하고자 한다.

2. 관련 연구

2.1 Diffie-Hellman Protocol

상호간의 보안 통신을 위해서는 서로간의 안전한 비밀키 교환이 이루어져야한다. 이를 보장하기 위해서는 안전한 키교환 프로토콜이 사용되어야 한다. 키교환 프로토콜은 안전성을 보장함과 동시에 키의 신선도와 확신에 대한 요구사항을 모두 만족시켜야한다. 본 논문에서는 Diffie-Hellman 프로토콜을 사용하여 안전한 키교환을 하고자 한다.

그림1은 Diffie-Hellman 알고리즘의 수식에 대한 설명이다. Diffie-Hellman 키 교환 방식은 대칭 암호방식에서 공통 비밀키를 교환 배포하는 취약점을 해결하는 방안으로 1976년 디피와 헬만에 의해서 발표되었다 [2].

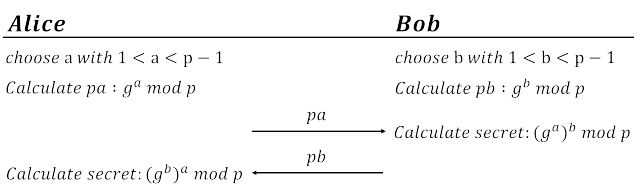

그림 1. Diffie-Hellman Protocol.

Diffie-Hellman 알고리즘은 갈루아 체와 생성원을 이용하여 이산대수 문제를 구현하였다. 기반을 두는 모든 가정은 유한순환군을 기반으로 하며, 그룹의 위수는 인수분해가 어려운 큰 소수를 약수를 가지도록 한다. 군을 생성하는 생성원이 필요하며, 군의 원소 중 하나가 주어졌을 때, 그 군의 원소를 생성하는 생성원의 승수를 찾은 것이 이산대수 문제이다.

각자 랜덤 비밀키를 생성하고 모듈러 연산을 통해 pa, pb를 도출한다. 그 후에 자신의 pa와 상대방의 pb를 공개된 네트워크를 통해 교환한다. 자신의 비밀키와 상대방에게서 받은 pa, pb를 사용하여 모듈러 연산 함수를 통해 대칭키를 도출한다 [3].

만약 공격자가 네트워크 상에서 주고받는 pa, pb를 가로채더라도 이산대수 문제로 클라이언트의 대칭키를 도출해낼 수 없다.

2.2 AES

통신은 공개된 네트워크에서 이루어지기 때문에 공격자가 가로채더라도 아무런 영향을 끼칠 수 없는 값들로 이루어져야 한다. 따라서 정보들을 암호화하여 송수신할 필요가 있다. 본 논문에서는 여러 암호화 알고리즘 중에서 AES 알고리즘을 통해 암복호화를 하고자 한다.

Advanced Encryption Standard (AES) 알고리즘은 빠른 속도와 보안성을 갖춘 암호화 알고리즘이다. Data Encryption Standard (DES) 알고리즘을 대체하기 위해 NIST에서 AES 알고리즘을 권장하고 있다 [2].

AES 알고리즘은 대칭키 방식의 블록 암호 알고리즘이며, Substitution-Permutation Network (SPN) 구조를 가진다. 이 구조는 S-box와 P-box를 이용하여 혼란과 확산을 만족시켜주는 구조이다.

해당 알고리즘은 데이터 블록 개수와 암호키의 사이즈에 따라 10, 12, 14번의 가변적인 수의 라운드를 진행한다. 암호화 과정에서 평문과 암호키는 state 상태로 변환 후 사용한다. 라운드의 구성은 SubBytes, ShiftRows, MixColumns, AddRoundKey로 이루어져 있으며, 마지막 과정인 AddRoundKey에서는 Round Key가 주어진다. 마지막 라운드에서는 MixColumns 과정은 생략된다. 각 과정에 지급되는 키는 key Expansion을 통해 생성된다. 회전, S-box를 이용한 치환, XOR 연산을 통해 매 라운드마다 새로운 키를 지급한다 [3].

복호화는 암호화는 다른 라운드 구성을 가진다. 복호화 라운드 구성은 InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns로 이루어져 있다. 마찬가지로 마지막 라운드에서는 InvMixColumns 과정은 생략된다.

본 논문에서는 AES 암호화를 블록 암호 모드 중에 ECB(Electric CodeBook) 모드로 구현하였다. ECB 모드는 평문 블록과 암호 블록이 일대일로 대응하는 모드이다.

3. 시스템 구조 및 구현

3.1. 시스템 구조

본 논문에서는 Miracl 라이브러리를 이용한 암호화 채팅을 window 데스크톱으로 구현하기 위해 c#의 프레임 워크 WPF를 사용하여 클라이언트를 구성했다. 소켓용 서버를 구축하기 위해서 javascript의 프레임 워크 node.js를 이용하였다. 다음 표 1은 시스템 구현 환경에 대한 설명이다.

표 1. 시스템 구현 환경

항목	환경
클라이언트	WPF
백엔드	node.js
개발 환경	window 10
개발 도구	visual studio, visual studio code

사용자가 채팅 클라이언트를 실행하게 되면 실행과 동시에 자신의 랜덤 비밀키를 생성하고 비밀키를 이용한 모듈러 연산으로 상대방에게 보낼 값을 생성한다. 상대방 클라이언트가 소켓 통신에 접속하게 되면 각자의 랜덤 비밀키를 사용하여 생성한 값을 서로에게 전송한다. 각자 받은 상대방의 값과 자신의 비밀키를 이용해 모듈러 연산을 진행하여 대칭키를 생성한다. 이와 같은 Diffie-Hellman 키 교환을 통해 생성된 대칭키를 AES 암호화의 키로 사용하며, 1:1 채팅을 시작하게 된다. 전체 시스템 구조는 그림 2와 같다.

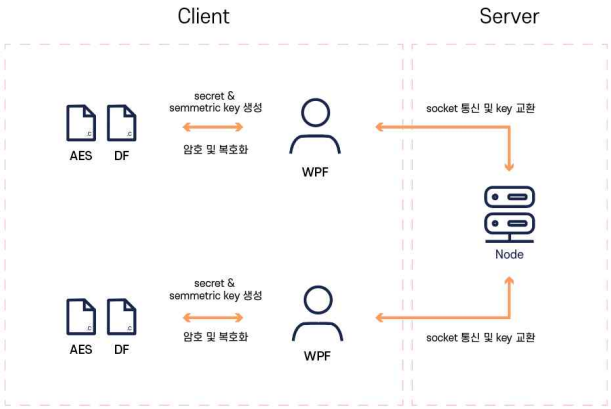

그림 2. 시스템 아키텍쳐.

3.2. 기능 구현

3.2.1. 키 생성 및 교환

Miracl 라이브러리에서는 기존 int 자료형의 크기 문제로 길이를 설정할 수 있는 big 자료형을 이용한다. 먼저, bigbits함수를 통해 160 비트의 랜덤 비밀키 a를 생성한다. 모듈러 연산 함수(powltr)를 통해 pa를 도출한다. a와 pa를 생성하는 코드는 그림 3와 같다.

```
bigbits(160, a);

powltr(3, a, p, pa);
cotnum(a, stdout);
cotnum(pa, stdout);
```

그림 3. a, pa 생성 코드.

비밀키를 사용한 모듈러 연산으로 도출한 pa을 서로 교환한다. 받은 pa와 자신의 a를 모듈러 연산 함수(powmod)를 통해 대칭키를 생성한다. 대칭키를 생성하는 코드는 아래 그림 4과 같다.

```
cinstr(a, myA);
cinstr(pb, yourPB);

powmod(pb, a, p, key);
cotnum(key, stdout);
```

그림 4. 대칭키 생성 코드.

3.2.2. 암호화 및 복호화

Miracl 라이브러리의 AES 객체를 생성한 후, 초기화 함수(aes_init)를 통해 세션을 초기화한다. 초기화 과정에서 필요한 대칭키의 값은 Diffie-Hellman 키 교환을 통해 생성된 대칭키를 사용한다. 해당 AES 객체를 사용하여 각각의 함수(aes_encrypt, aes_decrypt)를 통해 암호화, 복호화를 진행하게 된다. Miracl 라이브러리로 AES 암호화, 복호화를 구현한 코드는 아래 그림 5와 같다.

```
aes_init(&a, MR_ECB, nk, key, NULL);

aes_encrypt(&a, block);
aes_decrypt(&a, block);

aes_end(&a);
```

그림 5. AES 암호화 복호화 코드.

4. 구현 결과

구현한 시스템을 서버 로그와 시간 측정을 통하여 안정성과 성능을 분석하고자 한다. 서버 로그를 분석하여 네트워크 상에서 안전한 정보들이 송수신되는지 확인하고 키 교환과 암호화 과정의 연산속도를 측정하여 성능을 도출한다.

비밀키의 모듈러 연산을 통해 도출한 값과 대칭키로 만든 암호문을 서버를 통해 송수신된다. 공개되어도 상관없는 정보들이 송수신되는 것을 알 수 있다. 주고 받는 정보들에 대한 로그는 그림 6와 같다.

```
Socket IO server listening on port 3000
Conneted socket.id : ZWcZE7hIx39imOovAAAA
Conneted socketCount : 1
Conneted socket.id : psXbnvEx6B_RbULtAAAB
Conneted socketCount : 2
Two clients are accessing.
Client's pa:  6134212946221267337795211779607408071612901
0484430358323312525117523814080762252216289975149969900501
4865028405792625853767273246109433182197298271861343119 9
2704019056424296393409842627625081532672791852981727704092
4774311581706715270071873024019993109687903725642094336 9
9285665795359727858834636358614986847
Client's pa:  8104846461389066112163782258354303134664638
8157097693035584636438133735338353538102919913414308486208
2181314316507042578132452897584517800826471418214728833277
1674693599268106974062387195557943989671519410103233192299
3880836635914919127267596200258306913491173460257965551 4
7958538741387631265191296024127097225
Encryption message from client:  218 225 0 27 106 182 81
82 68 69 63 240 14 117 237 254
Encryption message from client:  231 113 7 79 178 136 128
 78 139 15 50 89 95 175 16 149
```

그림 6. 서버를 통해 송수신되는 값.

널리 쓰이는 암호화 라이브러리인 Miracl을 이용하여 Diffie-Hellman 키 교환과 AES 암복호화를 구현하였다. Diffie-Hellman 키 교환은 비밀키 생성, 대칭키 생성을 나누어 측정하였다. AES 암복호화 또한 암호화, 복호화를 나누어 측정하였다. Miracl에서 제공하는 시간 측정 코드를 사용하였다. 평균 연산 시간은 해당 코드를 10000번 반복하여 평균을 도출하였다. 측정 환경은 다음과 같다 "window 10 pro with memory: 7.6GB processor: Intel(R) Core(TM) i5-10400 CPU @ 2.90GHz type: 64-bit and disk: 222 GB" 다음 표 2는 평균 시간 측정 결과를 나타낸다.

표 2. 연산 속도 측정

항목		평균 시간
Diffie Hellman	비밀키 생성	0.14 ms
	대칭키 생성	0.15 ms
AES	암호화	0.0001 ms
	복호화	0.0001 ms

5. 결론

채팅 프로그램의 수요가 늘면서 보안성에 대한 관심도 늘어나는 양상을 보였다. 따라서 채팅 프로그램의 보안성도 향상되었다. 하지만 사용자들에게는 보안성을 확인할 방법이 없기 때문에 와닿지 않는 문제가 발생한다.

본 논문에서는 Diffie-Hellman 키 교환과 AES 암호화를 사용자가 직접 확인할 수 있는 안전한 1대1 암호화 채팅을 구현하였다. 공개되어도 문제 되지 않는 정보들이 네트워크상에서 전송되고 있는 것을 서버 로그를 통해 확인하였다.

6. 참고문헌

[1] KISA, "2006년 정보보호 실태조사 당신의 정보보호 수준은?," 정보보호뉴스, 2월호, pp. 12-17, 2007.

[2] W. Diffie and M. Hellman. New Directions in Cryptography. IEEE Transactions on Information Theory, 22(6):644-654, November 1976.

[3] N. Mäurer, T. Gräupl, C. Gentsch and C. Schmitt, "Comparing Different Diffie-Hellman Key Exchange Flavors for LDACS," 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC), 2020, pp. 1-10, doi: 10.1109/DASC50938.2020.9256746.

[4] PRERNA MAHAJAN, ABHISHEK SACHDEVA, Dr.. A Study of Encryption Algorithms AES, DES and RSA for Security. Global Journal of Computer Science and Technology, [S.l.], dec. 2013. ISSN 0975-4172. Available at: <https://computerresearch.org/index.php/computer/article/view/272>. Date accessed: 25 oct. 2021.

[5] Dworkin, M. , Barker, E. , Nechvatal, J. , Foti, J. , Bassham, L. , Roback, E. and Dray, J. (2001), Advanced Encryption Standard (AES), Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, [online], https://doi.org/10.6028/NIST.FIPS.197 (Accessed October 25, 2021)