한국군사과학기술학회지 제13권 제4호, pp. 627~634, 2010년 8월

학술논문 정보기술 부문

해쉬 함수를 이용한 그룹키 합의에 관한 연구

A Study on a Group Key Agreement using a Hash Function

이 준* 김 인 택* 박 종 범*
Jun Lee Intaek Kim Jongbum Park

Abstract

In this paper we suggest a group key agreement protocol among a group consisting more than 3 PKIs. From an 128 bit message, we produce a group key to any length size using a hash function. With a computer experiment we found that PKI's encryption/decryption time is the most dominant part of this procedure and an 160 bit ECC PKI is the most efficient system for distributing an 128 bit message in practical level. We implement this procedure over an unsecure multi user chatting system which is an open software. And we also show that this suggestion could be practically used in military business without a hardware implementation.

Keywords : Group Key Agreement, Fermat Test, KDC, ECC ElGamal

1. 서 론

워 게임과 같이 여러 사람이 공동 임무를 컴퓨터 네트워크에서 수행할 때 자격이 없는 제3자의 참여, 도청, 위장 등 악의적인 공격을 방지할 수 있는 방안은 자격자들만 비밀키를 공유하여 암호화 된 메시지로 통신하는 것이다. 현대암호에서 대칭키 암호는 비밀키 길이와 라운드를 다양하게 정의하여 보안 강도를 높인다. 일반적으로 비밀키 전달은 RSA 혹은 ECC 등의 대표적 공개키를 통해 이루어진다. 공개키 특성상 3인 이상으로 구성된 그룹에 대해서 RSA는 공통 메시지를 전달할 수 있으나 ECC에 비해 암호화 속도가 늦다. ECC는 RSA에 비해 암호화 속도는 빠르나 3개 이상의 ECC로 구성된 그룹에 대해 공통된 메시지를 전달할 수 없다.

본 연구는 3개 이상의 공개키 RSA와 ECC로 구성된 그룹이 공통 메시지를 공개키로 전달받을 수 있는 조건들을 검토하고, 그 결과를 해쉬함수에 이용하여 길이에 제한이 없는 그룹키 합의 방안을 제시한다.

제안된 그룹키 합의는 ECC로 구성된 그룹이 RSA 그룹보다 훨씬 효율적임을 컴퓨터 실험으로 증명한다. 또한 연구된 그룹키 합의 절차를 3인 이상의 다자간 채팅 프로그램에 구현함으로 이론적 제안으로 한정되는 것이 아니라, 실용적 사용이 가능함도 보인다. 본 연구는 워게임, NCW 등 3인 이상이 임무에 따라 유동적으로 그룹이 형성될 경우 1 회용 그룹키를 보안 업무에 필요한 비밀키로 신속히 공유하는데 적용할 수 있다.

† 2010년 4월 30일 접수~2010년 7월 23일 게재승인
* 공군사관학교(Korea Air Force Academy)
책임저자 : 이 준(jlee@afa.ac.kr)

2. 용어 정의

3인 이상의 공개키로 이루어진 공개키 소유자 집합을 **그룹**이라 하자. 그룹 구성원 모두가 동일한 방식으로 같은 세션키를 만드는 것을 **그룹키 합의**라 하고 이 때 동일하게 생산된 세션키를 그룹키(Group Key)라하고 G로 표기한다.

3. 관련연구

가. RSA

RSA 암호는 공개키 $P=\{e,N\}$와 개인키 $P^{-1}=\{d\}$로 이루어진다. RSA는 공개키 N 보다 작은 메시지 m에 대해서 전달 가능하다. n개 RSA로 구성된 그룹에서 구성원 각자가 사용하는 공개키를 첨자 i를 사용하여 $P_i=\{e_i,N_i\}$이라 하고 그에 대응하는 개인키를 $P_i^{-1}=\{d_i\}$라 할 때 아래 식의 M_{RSA}보다 작은 메시지는 구성원 모두에게 공통으로 전달할 수 있다.

$$M_{RSA}=\min\{N_i|1\le i\le n\}$$

따라서 n개의 RSA 공개키로 구성된 그룹에서 메시지 $m\in(0,M_{RSA})$은 공통 메시지로 전달 할 수 있다.

나. ECC

1) ECC 문제점

ECC는 타원방정식 $y^2=(x^3+ax+b)\bmod p$을 이루는 p, a, b와 타원의 두 점 Q와 kQ로 공개키 $P=\{p,a,b,Q,kQ\}$를 구성하며 개인키는 $P^{-1}=\{k\}$ 이다. n개의 ECC로 구성된 그룹에서 각 공개키를 첨자 i를 사용하여 공개키를 $P_i=\{p_i,a_i,b_i,Q_i,k_iQ_i|1\le i\le n\}$라 할 때, 공통의 메시지 m을 전달할 수 있는 범위는 다음과 같다

$$m\in(0,M_{ECC})$$
$$M_{ECC}=\min\{p_i|1\le i\le n\}$$

RSA와 달리 M_{ECC} 보다 작은 메시지에 대해서 모든 공개키가 동일한 메시지를 전달할 수 없다. Hasse 정리[1]에서 타원방정식은 0부터 $p-1$로 구성되는 소수체의 약 반 정도를 메시지로 사용할 수 있다. 또한 타원방정식을 만족하는 점들의 분포가 항상 일정한 것이 아니므로 3개 이상의 공개키로 구성된 타원방정식들을 모두 만족시키는 메시지는 찾을 수 없다. 결론적으로 ECC는 암호화 강도가 비슷한 수준에서 RSA보다 암호화 및 복호화 속도가 빨라 효율적이나 3개 이상의 공개키로 구성된 그룹에서 모든 공개키에 대해서 공통 메시지를 전달 수 없다. 또한 Hasse 정리는 공개키 크기에 무관하므로 키 길이가 확장된 경우에도 ECC로 공통의 메시지 m을 전달할 수 없다.

2) 공유키 존재

$n\ge 3$ 개로 구성된 ECC 공개키에서 M_{ECC}보다 작은 메시지 m은 $2^s>n$ 조건을 만족할 때 메시지 여유공간에 난수를 대입하여 공통 메시지를 전달할 수 있다[2].

여기서 n은 그룹을 구성하는 ECC 공개키 수이며, 공통 메시지 길이가 $|m|$비트, M_{ECC}의 길이가 $|p|$비트일 때 메시지 공간은 $|m|$비트이고, 여유 공간은 $s=|p|-|m|$비트이다.

이 때 메시지 전달 가능성 판단에 소요되는 추가 비용은 서버의 암호화 단계에서 Fermat 판정[3]이 평균적으로 1회 추가되며, 공통 메시지를 전달할 수 있는 알고리즘은 ElGamal[4]이다.

4. 그룹키 설계

가. 그룹키 조건

1) 공개된 설계 방식으로 생산하여 동일한 그룹키를 그룹 구성원이 공유할 수 있어야 한다.
2) 그룹키는 모든 공격에 대해서 그룹이 교신하는 메시지를 보호할 수 있어야 한다.
3) 새로운 그룹 구성마다 신속한 합의가 가능한 세션키로 구성한다.

나. 그룹키 제안

G가 M_{ECC} 혹은 M_{RSA}보다 클 경우 메시지는 나누어 보내야한다. 공개키의 암호화 및 복호화는 많은 시간이 소요되므로 공개키 1회 사용으로 전달된 초기메시지 D로부터 G를 합의하는 방안을 제안한다. 만일 공개키로 전달되는 초기 메시지 D가 대칭키 암호공격에 충분히 견딜 수 있을 경우, D를 사용하여 다음과 같은 절차에 따라 G를 생성한다.

1) 메시지 확대

공개키로 전달된 최초의 메시지 D를 초기 값으로 사용하여 다음과 같이 그룹키 형성에 필요한 길이만큼 해쉬함수 h를 반복하여 H_i를 얻는다.

$$H_i = h(H_{i-1})$$
$$H_0 = D$$

2) 그룹키 길이

그룹키는 대칭키 암호의 비밀키로 사용되므로 길이는 대칭키 암호체계의 비밀키 길이에 의존한다. 비밀키 길이가 $K_1 < K_2 < \cdots < K_{MAX}$로 다양할 경우, H_0 하위부터 $\log_2 K_{MAX}$ 비트를 취한 값을 L이라 할 때 $|G|$는 다음과 같이 결정한다.

$$|G| = \min\{K_i | L < K_i \;\; i = 1, \cdots, MAX\}$$

3) 해쉬함수 반복

해쉬 함수 h의 값 H_i의 길이가 $|H|$비트 일 때 반복 횟수 j는 다음과 같이 결정한다.

$$j = \left\lceil \frac{|G|}{|H|} \right\rceil$$

4) 그룹키 생성

반복되는 해쉬 함수의 마지막 H_j에서 하위비트부터 선택한 t 비트를 H_j^t로 결정하여 다음과 같이 그룹키 G를 생성한다.

$$t = |G| - j|H|$$
$$G = H_1 \| \cdots \| H_{j-1} \| H_j^t$$

다. 초기 메시지 전달

1) 초기 메시지 선택

현재 전자계산기 수준에서 블록암호는 비밀키 길이가 112비트 이상이면 안전함을 주장한다[5]. 통용되는 블록암호 비밀키 길이와 해쉬 함수의 출력은 128비트 이상이다. 따라서 초기 메시지의 길이는 112비트 이상이면 충분하나 처리되는 메시지는 8비트, 16비트, 32비트 단위로 처리하는 것이 편리하므로 128비트가 적합하다.

2) 공개키 선택

1024비트 RSA와 160비트 ECC는 암호화 강도가 유사하며 실용적으로 많이 사용한다.

가) 1024 비트 RSA

1024비트 RSA는 1024비트보다 작은 초기 메시지 128비트 D에 대해서 별도의 조치 없이 암호화하여 그룹 구성원 모두에게 전달할 수 있다.

나) 160 비트 ECC

① 초기 메시지 D가 128비트 인 경우 여유공간은 32비트로 그룹 구성원 수가 2^{32}보다 작은 경우 전달이 가능하다.

② 그룹 구성원의 각 공개키에 대해서 초기 메시지 D의 여유 공간에 임의의 난수 r을 대입한 확장된 메시지 $M = r \| D$가 Fermat 판정을 만족하면 M을 그룹 구성원에게 전달한다. Fermat 판정을 만족하지 않으면 만족할 때까지 난수 r을 교체하여 확장된 메시지 M을 찾는다. 공개키 당 요구되는 Fermat 판정은 평균 2회이다.

③ 확장된 메시지 M가 결정되면 ElGamal 알고리즘으로 메시지를 전달한다.

④ 메시지 M을 전달받은 공개키 소유자는 복호화 후 M에서 r을 제외한 D만 취한다.

5. 프로토콜

그룹키 합의 프로토콜은 다음 세 단계로 나누어진다.

가. 준비

① 그룹서버 S는 자신을 포함한 그룹 구성원 공개키 목록(*G_List*)을 KDC(Key Distribution Center)에 제출한다.

② S는 그룹 구성원에게 KDC에 접속하여 초기 메시지 D를 받은 후 합의된 그룹키로 형성된 그룹에 참여할 것을 알린다.

나. 초기 메시지 분배

① 그룹 구성원은 KDC에 접속하여 자신의 인증서 A를 KDC에 제출한다.

② KDC는 A가 *G_List*에 포함된 경우 그룹 구성원

임을 인증한다.

③ KDC는 인증된 접속자의 공개키 $P \in A$로 초기 메시지 전달 방식에 따라 접속자에게 D를 전달한다.

다. 그룹키(Group Key) 합의

① 그룹 구성원은 KDC로부터 받은 메시지에서 개인키를 사용하여 D를 얻는다.

② D의 하위비트에서 $\log_2 K_{MAX}$비트를 선택하여 그룹키 길이 $|G|$를 결정한다.

③ $|G|$와 $|H|$로부터 해쉬함수 반복회수 j를 얻는다.

④ $G = G \| H_i$를 $i = 1$부터 $j-1$까지 반복하여 결합한 후 H_j^t를 구하여 $G = G \| H_j^t$를 그룹키로 한다.

6. 실 험

공개키로 초기 메시지 D를 전달하는 암호화 및 복호화 시간 측정과 해쉬함수 처리시간을 측정하였다. 초기 메시지는 128비트로 하였다.

가. 실험장비

각 공개키 알고리즘[6,7]은 C++로 작성하였다. 실험에 사용된 수는 16비트 단위로 처리되었으며 연산은 같은 알고리즘을 사용하였고, 공개키에서 요구되는 수의 길이에 따라 배열의 길이만 다르게 설정하였다. 실험은 업무용 PC에서 실행하였다. PC 성능은 Intel Atom CPU 230, 1.6GHz, 0.99 GB RAM이다.

나. 그룹키 합의

1) 1024비트 RSA

512비트 유사 소수 2개를 선택하여 N을 만들고, 또 다른 512비트 유사 소수 하나를 선택하여 공개키 e로 만든 후 개인키 d를 얻었다. 개인키 d의 길이는 1024비트이다.

1024비트 RSA 공개키 당 초기 메시지 100개의 암호화, 복호화 시간을 측정하였다. 또한 다른 공개키를 66개에 대해서 같은 절차로 시간을 측정하여 종합하였다. 암호화에 소요된 총시간은 17459초이고, 복호화에 사용된 총 시간은 35704초이다. 공개키 당 암호화 평균시간은 2.645초이며 복호화 평균시간은 5.410초이다.

2) 160비트 ECC

160비트 ECC 공개키로 100개의 128비트 초기 메시지의 암호화, 복호화 시간을 측정하였다. 암호화 시간은 Fermat 판정 추가시간이 포함되었다. 이와 같은 절차를 250개의 다른 공개키에 대해서 암호화 복호화를 실행한 결과 암호화 총시간은 31599초이고, 복호화 총시간은 13165초이다. 160비트 공개키 당 암호화 평균시간은 1.26초 이고, 복호화 평균시간은 0.527초 이다.

3) 320비트 ECC

320비트 ECC 공개키로 100개의 128비트 초기 메시지의 암호화, 복호화 시간을 측정하였다. 암호화 시간 역시 Fermat 판정 추가시간을 포함하였다. 이와 같은 절차를 98개의 다른 공개키에 대해서 실행한 결과 암호화 총시간은 198865초이고, 복호화 총시간은 82715초이다. 320비트 ECC 공개키 당 암호화 평균시간은 20.29초이며, 복호화 평균시간은 8.44초 이다.

4) 해쉬 함수

해쉬 함수는 MD5[8]를 선택하였다. 같은 메시지에 대해서 해쉬함수를 10,000,000번 적용하여 소요된 시간은 33초이다. 128비트 초기 메시지 D에 대해서 소요되는 해쉬함수 처리시간은 평균 $0.33*10^{-5}$초이다. 암호화 및 복호화 시간에 비교할 때 해쉬함수 처리시간은 상대적으로 작아 무시할 수 있다.

다. 실험 결과

실험결과가 정리된 Table 1에서 암호화 시간은 KDC가 그룹 구성원의 인증서 A의 내부 공개키 P로 초기 메시지 D를 암호화 한 평균시간이며, 복호화 시간은 그룹 구성원이 KDC로부터 받은 암호화된 초기 메시지를 복호화하여 초기메시지 D를 얻는데 필요한 시간이다. 따라서 그룹전체가 그룹키를 소유하는 시간은 KDC 전체 암호화 시간(n*암호 시간)에 마지막 메시지를 받은 구성원의 초기 메시지 복호화 시간을 합한 시간이다. 그룹키 합의는 해쉬함수 처리 시간도 포함되나 실험결과 암호화 및 복호화 시간에 비해 상대적으로 작아 무시한다. 이론적으로 암호화 강도는 320비트 ECC가 가장 높으나, 실험결과 KDC가 초기메시지를 전달하는데 가장 많은 시간이 소요되므로 그룹키 합의에 실용적이라 할 수 없다.

암호화 강도가 비슷한 1024비트 RSA와 160비트 ECC의 초기 메시지 전달능력 비교실험 결과, 160비트

ECC는 암호화 시간에 Fermat 판정 시간이 포함되었음에도 D를 분배하는 데 KDC 부담이 가장 적었다. 실용적 수준에서 160비트 ECC가 1024비트 RSA보다 그룹키 합의에 효율적이다.

Table 1. 그룹키 합의시간 비교(단위 초)

구분	암호 시간	복호 시간	그룹 전체 합의 시간
1024 RSA	2.65	5.410	2.65n+5.41
160 ECC	1.26	0.527	1.26n+0.572
320 ECC	20.29	8.44	20.29n+8.44

라. 분석

공개키 알고리즘 실행시간은 대부분 16비트 곱셈이므로 이를 중심으로 분석한다.

1) RSA의 복호화 키길이는 암호화 키의 2배 이므로 소요시간을 비교할 때 타당하다.
2) ECC의 암호화와 복호화 시간 소요 주된 요인과 분석은 다음과 같다.
   ① ECC ElGamal 암호화는 두 번의 곱셈 mG와 $m(kG)$이 필요하며, 복호화는 한 번의 곱셈 $k(mG)$이 필요하다.
   ② 160비트와 320비트에서 사용되는 각 m의 길이는 2배 이므로 320비트 mG계산은 160비트에 비해 소요시간은 2배가 된다.
   ③ 160비트와 320비트는 배열의 길이가 2배이므로 곱셈계산 소요시간은 4배가 된다.
   ④ 위의 분석에 따라 복호화 시간은 암호화 시간의 약 1/2이 되며, 320비트 ECC 암호화는 160비트의 16배정도 소요되므로 ECC 실험 결과도 타당하다.

마. 안전도

1) 160비트 ECC와 1024비트 RSA는 암호화 강도가 비슷하며 실제적으로 해독이 불가능하다. 실용적 공개키는 1024비트 RSA 혹은 160비트 ECC로 구성된다. 따라서 KDC에서 분배하는 128비트 초기 메시지 D는 그룹 구성원 인증서 공개키로 안전하게 전달되므로 그룹외부에서 초기 메시지 D를 얻을 수 없다.
2) 초기 메시지 길이가 너무 짧은 경우 공개키 암호문은 해독할 수 없더라도 전수공격은 가능하다. 초기 메시지가 128비트이므로 현대 컴퓨터 계산 수준에서 블록암호의 비밀키 전수공격 대해 안전하다.
3) 그룹이 구성될 때마다 그룹서버 S가 대상자들을 KDC에 등록하여 자격자들만 인증된 후 초기 메시지를 받는다. 그룹이 형성될 때마다 KDC로부터 새로운 초기 메시지 D를 받으므로 합의된 그룹키는 다른 그룹의 임무에 재사용될 수 없다.

7. 구 현

KDC에서 초기 메시지를 분배하는 과정과 그룹키 합의를 이용한 다자간 암호문 채팅을 공개 프로그램 위에 구현한 후 Network에서 실험[9]하였다.

가. 공개 소프트웨어

보안장치가 전혀 없는 공개된 다자간 채팅 프로그램[10]을 선택하였다. 채팅 프로그램에 C++를 이용하여 그룹키 합의와 보안 채팅을 구현하였다. 이 때 해쉬함수로 MD5를 선택하고, 160비트 ECC를 공개키 암호체계로 하였으며, 대칭키 암호체계는 ARIA[11]를 사용하였다.

나. 실험 내용

1) 초기 메시지 분배

Fig. 1과 2는 동일한 화면으로 공개키로 암호화된 초기메시지를 분배할 때 사용된다. Fig. 1처럼 서버를 선택하면 KDC 역할을 한다. KDC는 실행 전에 S가 제출한 회의 참석 대상자 인증키 목록 *G-List*를 파일로 갖고 있다. KDC가 준비한 초기 메시지 128비트 D를 다음과 같이 하위 16비트씩 2^{16}진법으로 표기하여 128비트를 기록하였다.

D={ 0xcdab, 0x134f, 0xfdc1, 0x765d, 0xdacf, 0x3308, 0xab64, 0xdb6c }

그룹 구성원은 Fig. 2처럼 클라이언트를 선택하고 KDC의 IP를 기입한다. 클라이언트가 KDC에 접속되면 인증서 A를 KDC에게 제출한다. KDC는 접속한 클라이언트 인증서 A를 받아 그룹 목록 *G-List*에서 일치

하는 인증서 존재를 파악한다. 이 때 사용한 인증서는 연구된 결과[12]를 사용하였다. 일치된 인증서 A가 *G-List*에 있는 경우 KDC는 클라이언트 인증서에 기록된 160비트 ECC 공개키로 초기 메시지 D를 보낼 수 있도록 Fermat 판정을 실시한다. 적합하지 않을 경우 32비트 여유 공간에 임의의 난수를 대입을 반복하여 Fermat 판정이 만족할 때까지 계속한 후 확장된 메시지를 만들어 암호화하여 클라이언트에게 보낸다.

클라이언트는 개인키를 이용하여 KDC가 보낸 메시지를 복호화하고 그 결과 중 하위 128비트를 D로 취해 초기 메시지에서 H_0를 만든다.

Fig. 1. KDC 서버 선택화면

Fig. 2. KDC 클라이언트 선택화면

해쉬 값은 하위부터 8비트씩 128비트를 16진법으로 표기한다.

H_0 = { 0xab, 0xcd, 0x4f, 0x13, 0xc1, 0xfd, 0x5d, 0x76, 0xcf, 0xda, 0x08, 0x33, 0x64, 0xab, 0x6c, 0xdb }

대칭키 ARIA의 키 길이는 K_1=128, K_2=192, K_{MAX} =256이므로 H_0의 하위 8비트($\log_2 K_{MAX}$)를 취해 192 비트로 결정하였다. 해쉬 반복 회수 $j = \lceil 192/128 \rceil = 2$ 이다. H_0를 MD5에 반복 적용한 값은 다음과 같다.

H_1 = { 0x8c, 0x32, 0x9d, 0xc5, 0x05, 0x4c, 0x24, 0x9e, 0x35, 0xa3, 0x2f, 0xc5, 0x5e, 0x63, 0xbf, 0x4b }

H_2 = { 0x10, 0x35, 0xe5, 0xac, 0x5e, 0x07, 0xa4, 0x37, 0x1b, 0xa3, 0x04, 0x7e, 0x2a, 0xeb, 0x83, 0xfe }

이 때 t=192-128=64이며 H_2에서 하위 64비트를 취해 다음과 같이 192비트 G를 얻는다.

$G = H_1 \| H_2^{64} =$ { 0x8c, 0x32, 0x9d, 0xc5, 0x05, 0x4c, 0x24, 0x9e, 0x35, 0xa3, 0x2f, 0xc5, 0x5e, 0x63, 0xbf, 0x4b, 0x10, 0x35, 0xe5, 0xac, 0x5e, 0x07, 0xa4, 0x37 }

2) 보안 채널

G를 얻은 후 그룹 구성원은 접속화면 Fig. 3에서 서버와 클라이언트를 설정한다. 클라이언트를 선택한 경우 업무 서버의 IP를 기입하여 그룹에 참여한다. Fig. 3 이후 모든 클라이언트가 서버 접속부터 채팅 참여까지 교신되는 메시지는 G를 ARIA의 비밀키로 암호화하였다.

Fig. 3. 회의 서버 접속화면

서버와 클라이언트에 ARIA로 암호화된 메시지가 도착하면, 메시지를 그룹키로 복호화하여 화면에 띄운다.

Fig. 4는 그룹 서버이고, Fig. 5는 그룹 클라이언트로 설정된 것이다. Fig. 4와 5는 공개된 프로그램과 외형상 동일하나 기능은 모든 통신 메시지가 ARIA로 암호화 되어 전송되는 큰 차이가 있다. 그룹키가 없는

악의적인 공격자가 스니핑 할 경우 통신 내용은 암호화되어 도청할 수 없다.

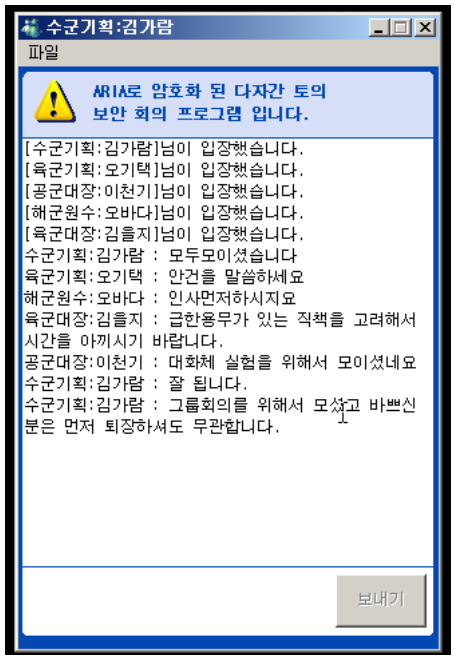

Fig. 4. 그룹 서버

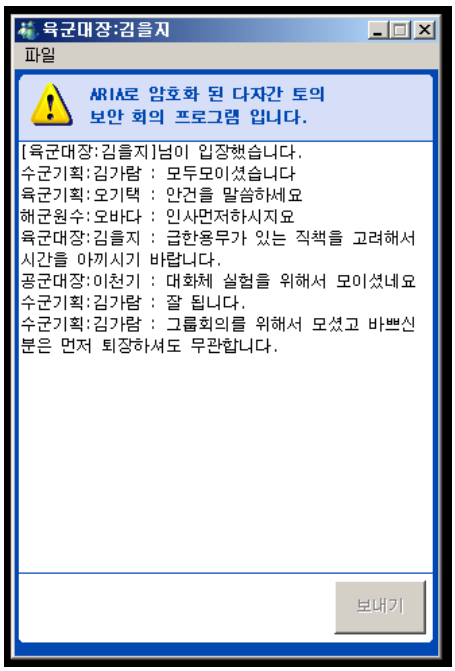

Fig. 5. 그룹 클라이언트

KDC에서 분배한 초기 메시지를 받아 그룹키를 생성한 후 그룹에 참여하는 Fig. 3 이후 Fig. 4 혹은 5의 초기 창으로 전환되는 시간은 거의 실시간 접속으로 소요된 시간은 없었다. Fig. 4와 5에서 대화에서 그룹키를 사용한 메시지의 암호화, 복호화도 실시간으로 이루어졌다.

8. 결 론

본 논문은 3개 이상의 공개키로 이루어진 그룹에서 길이에 제한 없이 그룹 구성원 모두가 비밀키를 신속히 공유하는 그룹키 합의 방법을 제안하고 효율성을 실험하였다. 그룹키 합의에서 해쉬함수 처리시간은 공개키의 암호화/복호화 시간에 비해 무시할 수 있을 정도로 작아 그룹키 합의는 KDC의 초기 메시지 전달능력에 의존함을 실험으로 알았다. 실용적 수준에서 실험한 결과 160비트 ECC는 초기 메시지 전달에 가장 효율적임을 보였다. 제안된 방법으로 생산된 그룹키는 전수공격, 재사용, 도청 등 악의적인 공격에 대해 구조적으로 강함을 보였다. 또한 연구된 절차는 160비트 ECC로 그룹키를 합의하는 과정과 ARIA로 암호문을 교환하는 채팅으로 구현하여 별도의 하드웨어 개발 없이 실용화 할 수 있음도 보였다.

연구 결과는 업무 및 작전에 따라 다양한 그룹이 형성될 때 그룹키를 신속히 공유하는 NCW 등에 적용할 수 있다.

후 기

본 연구 논문은 2009년도 공군사관학교 국고연구비 (KAFA 09-17) 예산지원으로 수행된 결과입니다.

Reference

[1] 김창한, “암호학과 대수학”, 북스힐, p. 356, 1999.
[2] 이 준, 박종범, “ECC 키분배에서 공유키 존재에 관한 연구”, 한국군사과학기술학회지 제12권 제4호, pp. 476~482, 2009.
[3] Johannes A. Buchmann, “Introduction to Cryptography”, Springer, pp. 129~130, 2000.

[4] 강주성 외 6인, “현대 암호학”, 경문사, pp. 170～171, 2000.
[5] 홍성룡, 조정호, “SDR System 적용을 위한 한국형 암호 알고리즘(SEED) 구현 및 성능분석”, 한국정보과학회 2003년도 봄 학술발표 논문집 제30권 제1호, pp. 319～321, 2003. 4.
[6] 이 준, “RSA 공개키 암호의 실용적 구현에 관한 연구”, 공군사관학교, KAFA 02-1-3-9, 2002.
[7] 이 준, “ECC 공개키 암호의 실용적 구현에 관한 연구”, 공군사관학교, KAFA 05-23, 2005.
[8] http://www.ietf.org/rfc/rfc1321.txt
[9] 이 준, “보안성을 보장하는 국방망 회의의 실용적 구현에 관한 연구”, 공군사관학교, KAFA 09-17, 2009.
[10] 김정훈, “TCP/IP 소켓 프로그래밍”, 교학사, pp. 383～453, 2003
[11] http://www.nsri.re.kr/ARIA
[12] 이 준, “국방망 보안 채널 구현에 관한 연구”, 한국군사과학기술학회지 제11권 제3호, pp. 106～114, 2008.