1-라운드 양자간 키 교환 프로토콜
(One-round Protocols for Two-Party Authenticated Key Exchange)

정 익 래 † 이 동 훈 ††
(Ik Rae Jeong) (Dong Hoon Lee)

요 약 양자간 암호학 프로토콜 디자인에서 잘 고려되지 않는 사항 중에 동시 메시지 전송이 있다. 즉, 듀플렉스(duplex) 채널을 사용해서 통신하는 두 파티는 동시에 메시지를 보낼 수 있다. 하지만 대부분의 양자간 키 교환 프로토콜은 두 파티가 교대로 메시지를 보내는 하프 듀플렉스(half-duplex) 채널을 가정해서 디자인되었다. 이 논문에서 우리는 듀플렉스 채널을 사용할 경우에 동시 메시지 전송을 사용해서 좀 더 효율적인 양자간 키 교환 프로토콜을 설계할 수 있음을 보인다.

이 논문에서는 세 개의 안전성 증명 가능한 1-라운드 양자간 키 교환 프로토콜들을 제안한다. 첫 번째 프로토콜은 가장 효율적이며, 세션키 독립성(key independence)을 보장한다. 두 번째 프로토콜은 세션키 독립성과 더불어 전방위 안전성(forward secrecy)을 보장한다. 세 번째 프로토콜은 두 번째 프로토콜과 같은 안전성을 보장하지만, 표준모델(standard model)에서 안전성이 증명된다. 우리가 제안하는 프로토콜들은 최초의 안전성 증명이 가능하면서 전방위 안전성을 제공하는 1-라운드 양자간 키 교환 프로토콜이다.

키워드 : 인증된 키 교환, 전방위 안전성, 라운드 복잡도, Diffie-Hellman 키 교환

Abstract Cryptographic protocol design in a two-party setting has often ignored the possibility of *simultaneous* message transmission by each of the two parties (i.e., using a duplex channel). In particular, most protocols for two-party key exchange have been designed assuming that parties alternate sending their messages (i.e., assuming a bidirectional half-duplex channel). However, by taking advantage of the communication characteristics of the network it may be possible to design protocols with improved latency. This is the focus of the present work.

We present three provably-secure protocols for two-party authenticated key exchange (AKE) which require only a single round. Our first, most efficient protocol provides *key independence* but not forward secrecy. Our second scheme additionally provides *forward secrecy* but requires some additional computation. Security of these two protocols is analyzed in the random oracle model. Our final protocol provides the same strong security guarantees as our second protocol, but is proven secure in the standard model. This scheme is only slightly less efficient (from a computational perspective) than the previous ones. Our work provides the first provably-secure *one-round* protocols for two-party AKE which achieve forward secrecy.

Key words : Authenticated key exchange, Forward Secrecy, Round Complexity, Diffie-Hellman key exchange

1. 서 론

키 교환 프로토콜은 가장 널리 사용되는 암호학 프로토콜 중에 하나이다. 키 교환 프로토콜은 사용자들간에 공통된 세션키를 만들며, 이런 세션키들은 안전하지 않은 네트웍을 통해서도 안전한 통신을 가능하게 해준다. 그러므로 안전한 키 교환 프로토콜은 좀 더 복잡하고 안전한 상위 레벨의 어플리케이션을 위한 기본적인 툴로 사용될 수 있다. 따라서 키 교환 프로토콜의 계산량이나 주고 받는 메시지의 개수 및 라운드 수가 매우 중요하며, 지금까지 많은 연구가 이루어져오고 있다. 이 논문에서는 인증된 키 교환 프로토콜에 국한하여 논의

· 이 논문은 2003년도 한국학술진흥재단의 지원(KRF-2003-041-D00500)에 의하여 연구 되었습니다.
† 정 회 원 : 고려대학교 정보보호센터 연구원
jir@cist.korea.ac.kr
†† 정 회 원 : 고려대학교 정보보호대학원 교수
donghlee@korea.ac.kr
논문접수 : 2005년 5월 11일
심사완료 : 2005년 11월 7일

한다. 인증된 키 교환을 위한 일반적인 방법은 각 사용자가 인증된 공개키-개인키 쌍을 가지고 키 교환을 하는 방식이다.

키 교환의 안전성을 정의하기 위해서 [1]에서 매칭 대화(matching conversation)라는 개념을 명시적으로 정의하였으며, 이후에 키 교환의 안전성 분석을 위해서 널리 사용되었다[2]. 이 매칭 대화는 양자간 키 교환 프로토콜에서 두 사용자가 서로 메시지를 번갈아 주고 받는 것을 가정한다. 달리 말하면, 사용자들이 하프-듀플렉스(half-duplex) 채널을 이용한다고 가정한 것이다. 하지만 많은 경우에 있어서 두 사용자는 동시에 메시지를 주고 받을 수 있다. 즉, 듀플렉스(duplex) 채널로 통신하는 것으로 볼 수 있다. 키 교환 프로토콜이 하프-듀플렉스 채널에서 안전하다면 이 프로토콜은 여전히 듀플렉스 채널에서 안전하다. 하지만 만약 우리가 네트웍의 기능을 이용해서 프로토콜을 설계한다면 더욱 효율적인 프로토콜 설계가 가능할 것이다. 이 논문에서는 듀플렉스 채널에서 가능한 동시 메시지 전송 기능을 사용해서 어떻게 효율적인 키 교환 프로토콜을 설계하는지에 대해서 연구한다. 예를 들어서 전통적인 Diffie-Hellman 키 교환 프로토콜에 대해서 생각해 보자. 통신하는 두 사용자를 Alice와 Bob이라고 하자. 키 교환을 위해서 Alice가 먼저 g^a를 Bob에게 보내고 이 메시지를 받은 Bob은 g^b을 Alice에게 보낸다. 두 메시지를 가지고서 Alice와 Bob은 세션키로 g^{ab}를 계산한다. 이 프로토콜은 하프-듀플렉스 채널에서는 2 라운드가 필요하다. 하지만 듀플렉스 채널에서는 Alcie와 Bob이 동시에 메시지를 보내면 되므로 1 라운드만에 프로토콜 수행을 종료할 수 있다. 방금 본 예에서는 사용자 인증에 대해서는 고려하지 않았다. 만약 인증이 추가된 키 교환 프로토콜에서는 위에서와 같이 프로토콜의 라운드 수를 줄이는 것이 불가능할 수도 있다. 예를 들어 [1,3]의 인증된 키 교환 프로토콜들은 위에서와 같은 방식으로 라운드 수를 줄이는 것이 불가능하다. 우리는 이 논문에서 1 라운드를 필요로 하는 인증된 방식의 키 교환 프로토콜을 설계하며 안전성을 분석한다. 우리가 제안하는 프로토콜은 인터넷과 같은 듀플렉스 채널에서 통신하던 두 사용자가 키를 만들 필요가 있거나 또는 기존의 키를 대체할 새로운 키를 만들 필요가 있을 경우에 일 라운드로 새로운 키를 생성할 수 있다.

1.1. 기존 연구 및 이 논문의 기여도

기존 연구와의 관련성을 논하기 전에 먼저 키 교환 프로토콜에서의 안전성 개념에 대해서 살펴보자. 엄밀한 정의는 다음절에 나온다. 가장 기본적으로 인증된 키 교환 프로토콜은 세션키에 대한 비밀성을 보장해야 한다. 이런 안전성 개념은 어떤 공격에 대해서 안전한지를 말해야 한다. 가장 최소한의 세션키 비밀성은 수동적 도청만 가능한 공격자에 대해 세션키의 비밀성을 유지해야 한다는 것이다. 좀 더 강한 개념으로 키 독립성(Key Independence)이 있다. 이는 세션키들이 공격자에게는 독립적으로 보여야 한다는 것을 의미하며, Denning-Sacco 공격[4]에 대해 안전해야 한다는 것을 말한다. 마지막으로 전방위 안전성(forward secrecy)[5]을 말할 수 있다. 이 개념은 공격자의 간섭없이 만들어진 세션키는 사용자의 비밀키를 가진 공격자라고 할지라도 세션키를 알 수 없다는 것을 말한다. 인증에 대해서 국한하면 암시적 인증성(implicit authentication)과 명시적 인증성(explicit authentication)으로 나누어 볼 수 있다. 암시적 인증성이란 사용자가 의도한 파트너만이 세션키를 만들 수 있음을 의미한다. 명시적 인증성이란 파트너가 이 세션키를 만들었음을 확인할 수 있음을 말한다. 암시적 인증성은 질문-응답(challenge-response)을 사용해서 항상 명시적 인증성으로 바꿀 수 있으므로 우리는 이 논문에서 암시적 인증성만을 고려해서 프로토콜을 설계한다.

최초의 Diffie-Hellman 프로토콜[6]은 수동적 공격에는 안전하지만 인증은 전혀 제공하지 않는다. 인증을 제공하기 위해서 다양한 연구가 행해져 왔다[7-10]. 하지만 키 독립성과 전방위 안전성을 동시에 제공하면서 증명 가능한 프로토콜들은 많지 않다. 그리고 그런 프로토콜들은 명시적 인증성을 제공하기 위해서 부가적인 계산량이나 라운드 수를 필요로 한다. 예로 [2,3,11]에서는 명시적 인증성을 보장하기 위해서 서명이나 MAC을 사용한다. 하지만 어떤 경우에는 명시적 인증성은 필요가 없거나 키를 사용한 프로토콜에서 다른 방식으로 제공할 수 있다. 따라서 명시적 프로토콜이 필요 없다면 더욱 효율적으로 프로토콜 설계가 가능할 수도 있을 것이다.

우리는 먼저 가장 간단한 TS1이라는 프로토콜을 제안한다. 이 프로토콜은 키 독립성은 보장하지만 전방위 안전성은 보장하지 못한다. 이것은 Boyd와 Nieto에 의해서 제안된 프로토콜[12]과 비교할만하다. 두 프로토콜 모두 키 독립성을 보장하지만, TS1이 좀더 효율적이며, 또한 두 사용자가 같은 계산량을 수행한다는 면에서 대칭적이다. 다음으로 우리는 TS2라는 프로토콜을 제안한다. TS2는 키 독립성과 전방위 안전성 두 가지 모두 보장하는 1 라운드 프로토콜이다. 이전까지 제안된 프로토콜 중에서 키 독립성과 전방위 안전성을 보장하는 1 라운드 프로토콜은 없었다. TS2는 사용자마다 단지 3번의 지수 연산을 필요로 할 뿐 다른 서명이나 키 확인(key confirmation) 과정을 필요로 하지 않는다. TS1이나 TS2의 단점으로는 프로토콜의 안전성이 랜덤 오라클

(random oracle) 모델에서 증명된다는 점이다. 우리는 표준모델에서 증명가능하면서 키 독립성과 전방위 안전성을 보장하는 TS3을 또한 제공한다. TS3는 MAC 스킴을 필요로 한다. 키 독립성과 전방위 안전성을 보장하는 증명 가능한 프로토콜들간의 비교가 테이블 1에 나와 있다.[1)]

표 1 관련 프로토콜들간의 비교

	[2[1)],11]	[3]	[13]	TS2	TS3
지수연산	3	4	4	3	3
라운드수	3	3	2	1	1
안전성 모델	랜덤 오라클	표준	표준	랜덤 오라클	표준

2. 키교환의 안전성 모델

우리는 [1]에서 정의된 키 교환 모델을 사용하며, [5]에서와 같이 전방위 안전성을 고려한다. 우리는 키교환 시스템에서의 사용자 수를 N이라 놓고, 각각의 사용자 ID을 P_i로 놓는다. 각 사용자는 공개키-개인키 쌍을 가지고 있으며 키 교환 프로토콜에서 인증을 위해서 사용한다. 우리는 이 논문에서 양자간 키 교환 프로토콜을 고려하며, 이를 위한 모델을 설명한다. 오라클 Π_i^k는 P_i의 k번째 인스턴스를 의미한다. 키 교환 프로토콜이 종료하면 Π_i^k는 세션키 sk_i^k를 생성한다. 세션 식별자 sid_i^k는 세션들 중에서 하나의 세션을 의미할 수 있는 스트링이다. sid_i^k는 Π_i^k가 세션에서 보는 메시지들의 결합이라고 정의할 수 있다. 메시지들의 결합은 사용자들의 ID의 사전순서(lexicographic order)에 의해서 결합할 수 있다. (이 논문에서 우리는 통신하는 두 파티가 동시에 메시지를 보낼 수 있으므로, 시간의 흐름상 먼저 나타나는 순서에 따라서 메시지를 결합 순서를 정할 수 없다.) 오라클의 *파트너*는 오라클이 통신하고 있다고 믿는 상대방의 ID를 의미한다. 만약 $sid_i^k = sid_j^{k'}$이며, Π_i^k의 파트너가 P_j이고 $\Pi_j^{k'}$의 파트너가 P_j이면, Π_i^k와 $\Pi_j^{k'}$은 *매칭*이라고 한다. 모든 안전한 프로토콜은 다음을 만족해야 한다: 만약 두 오라클이 매칭이면, 두 오라클은 같은 세션키를 계산해야 한다. 안전성 개념을 정의하기 위해서 우리는 먼저 공격자의 능력을 정의한다. 우리는 공격자가 일련의 오라클 쿼리(query)를 통해서 통신상에서 흐르는 모든 메시지의 흐름을 제어한다고 가정한다. 공격자의 성공 확률인 어드벤티지를 정의하기 위해서 우리는 실험을 수행하며, 이 실험에서 공격자는 오라클들에게 쿼리를 던지며, 오라클들은 이 쿼리에 응답한다. 오라클 쿼리들은 현실 세계에서 공격자가 행하는 공격을 모델링한다. 이 논문에서 우리는 다음과 같은 쿼리들을 고려한다.

- Initiate(i,j): 이 쿼리는 P_i로 하여금 P_j와 키 교환 프로토콜을 수행하게 한다. P_i는 이 쿼리의 응답으로서 키 교환 프로토콜의 첫 번째 메시지를 공격자에게 돌려준다.
- Send(i,k,M): 이 쿼리는 메시지 M을 오라클 Π_i^k에게 보낸다. Π_i^k는 M을 받고서 프로토콜에 따라서 반응한다. 이 쿼리를 통해서 현실 세계에서의 능동적 공격(active attack)을 모델링할 수 있다.
- Execute(i,j): 이 쿼리는 현실 세계에서의 수동적 공격을 모델링한다. 이 쿼리를 받으면 P_i와 P_j사이에서 키 교환 프로토콜을 공격자의 간섭없이 수행하며, 이 세션에서의 프로토콜 메시지들을 돌려준다.
- Reveal(i,k): 이 쿼리는 기존 세션의 세션키가 알려질 경우를 모델링한다. 즉, Denning-Sacco 공격을 모델링한다. 이 쿼리의 응답으로 세션키 sk_i^k를 돌려준다.
- Corrupt(i): 이 쿼리는 사용자 P_i의 비밀키가 노출되었을 경우를 모델링한다. 하지만 공격자는 P_i의 비밀키는 알 수 있더라도 공격자의 행동까지 컨트롤하지는 못한다. 물론 공격자는 이 비밀키를 가지고서 다른 사용자에게 P_i인척 할 수 있다(impersonation attack).
- Test(i,k) : 이 쿼리는 공격자의 성공 능력을 정의하기 위해서 사용한다. 공격자가 이 쿼리를 던지면 동전던지기를 수행한다. 이 결과를 b라고 하자. 만약 b가 1이면 세션키 sk_i^k를 공격자에게 돌려주고, 아니면 그냥 임의의 값을 키공간에서 뽑아서 돌려준다. 공격자는 하나의 Test 쿼리를 프레쉬(fresh) 오라클에게 던질수 있다. 이제 프레쉬 오라클을 정의한다.

정의 1. 실험이 끝날 때까지 다음의 조건들을 만족하는 오라클 Π_i^k는 프레쉬하다. Π_i^k와 매칭되는 오라클이 있으면 이 오라클을 $\Pi_j^{k'}$라고 하자.

(a) 공격자는 Reveal(i,k)나 Reveal(j,k') 쿼리를 하지 않았다.

(b) 만약 공격자가 Corrupt(i)나 Corrupt(j) 쿼리를 했다면, 공격자는 Send(i,k,*)나 Send(j,k',*) 쿼

1) [2]에서 제안된 4개의 프로토콜 중에서 Protocol 3은 키 독립성을 제공하지 않으며, Protocol 4는 증명되지 않았으므로 표 1의 비교대상에서 제외되었다. Protocol 1과 Protocol 2는 키 독립성과 전방위 안전성을 제공하므로 표 1에서는 이들을 고려한다.

리를 하지 않았다.[2)]

공격자는 실험이 끝날 때 b′을 출력한다. 공격자의 공격 성공 확률은 어드벤티지로 정의되며 다음과 같다: $Adv_A(k)=|2\Pr[b'=b]-1|$. 만약 모든 다항시간 공격자의 어드벤티지가 네글리져블(negligible)하다면 이 프로토콜은 안전한 프로토콜이다. 다음은 키 교환 프로토콜이 얻을 수 있는 안전성들의 종류다.

(1) 프로토콜은 기본적으로 Reveal이나 Corrupt 쿼리를 허용하지 않을 때 안전해야 한다.

(2) 키 독립성(Key Independence): 공격자는 Reveal 쿼리는 할 수 있으나 Corrupt 쿼리는 할 수 없다. 이런 공격자들에 안전한 프로토콜은 키 독립성을 보장한다고 한다.

(3) 전방위 안전성(Forward Secrecy): 공격자는 Reveal 쿼리는 할 수 없으나 Corrupt 쿼리는 할 수 있다. 이런 공격자들에 안전한 프로토콜은 전방위 안전성을 보장한다고 한다.

우리의 정의에서는 전방위 안전성이 키 독립성을 포함한다. 프로토콜 P에 대해 공격자가 할 수 있는 공격 종류에 따라서 어드벤티지를 다음과 같이 표기한다: $Adv_P^{XX}(k,t)=\max_A\{Adv_A^{XX}(k)\}$. 여기서 XX는 KI(키 독립성), FS(전방위 안전성), 또는 KI&FS(키 독립성과 전방위 안전성) 중에 하나이다. k는 안전성 패러미터이며, t는 공격자에게 허용되는 시간이다. 만약 모든 다항시간 t=poly(k)에 대해서 $Adv_P^{XX}(k,t)$가 네글리져블하다면 P는 XX-안전성을 보장한다고 말한다.

3. 1-라운드 키교환 프로토콜

우리가 제시하는 모든 프로토콜들은 다음을 가정한다. 사용자들은 그들의 ID에 의해서 순서대로 정렬될 수 있으며, 만약 P_i가 P_j에 앞설 경우 $P_i<P_j$로 표기한다. k는 안전성 패러미터이며, G는 소수 q를 오더(order)로 가지는 그룹이며, g는 그룹의 제너레이터(generator)이다. H는 임의의 스트링을 k비트 스트링으로 맵핑하는 해쉬 함수다. 각각의 사용자 P_i는 공개키-개인키 쌍 $(y_i=g^{x_i},x_i)$를 가진다.

3.1 프로토콜 TS1

TS1은 대칭적인 프로토콜이다. 따라서 우리는 TS1을 P_i의 입장에서 프로토콜을 기술하지만, 파트너인 P_j도 유사하게 행동한다.

Setup: P_i는 P_j와 세션키를 만들려고 하며, $P_i<P_j$이다.

Round1: P_i는 난수 r_i를 k비트 스트링 공간에서 랜덤하게 뽑아서 P_j에게 보내며, P_j로부터 r_j를 받는다.

세션키 계산: P_i는 세션식별자를 만들기 위해서, 브로드캐스트된 메시지를 P_i와 P_j의 순서에 의해서 결합한다. 즉, 세션 식별자는 $sid_i=r_i\|r_j$이다. P_i는 세션키 $sk_i=H(i\|j\|sid_i\|y_j^{x_i})$를 만든다.

TS1의 실행 예가 그림 1에 나와 있다.

	$P_1(x_1,y_2)$	$P_2(x_2,y_1)$
Round 1	r_1	r_2

$sid=r_1\|r_2$

$sk=H(1\|2\|sid\|g^{x_1x_2})$

그림 1 TS1의 실행 예

다음의 정리는 TS1의 안전성에 관한 것이다.

정리 1. 만약 G에서 CDH(computational Diffie-Hellman) 문제가 어려운 문제면, TS1은 랜덤오라클 모델에서 키 독립성을 보장한다. 구체적으로 G가 CDH 문제에 대해서 (t,ϵ) 안전성을 보장한다면,

$$Adv_{TS1}^{KI}(k,t,q_{re},q_h)\le q_hN^2\epsilon+\frac{q_s^2}{2^k}$$ 이다.

여기서 t는 공격자의 실행시간을 포함한 총 실험시간이며, 공격자는 q_{re} Reveal 쿼리와 q_h 해쉬 쿼리를 던진다. N은 실험에서 최대 사용자의 수이며, q_s는 실험에서 최대 세션의 수이다. CDH 문제에 대한 (t,ϵ) 안전성이란 어떤 공격자도 시간 t안에 ϵ이상의 확률로 CDH문제를 풀 수 없다는 가정이다.

증명. 랜덤 오라클 모델에서는 공격자가 세션키 $sk_i^k=H(i\|j\|sid_i\|y_j^{x_i})$에 대한 정보를 얻을 수 있는 방법이 두 가지 밖에 없다. 첫 번째는 $i\|j\|sid_i\|y_j^{x_i}$를 해쉬 오라클에게 쿼리하는 것이고, 두 번째는 세션 식별자 sid_i^k가 실험에서 반복되는 경우이다. 하지만 sid_i^k가 반복될 확률은 $\frac{q_s^2}{2^k}$로 무시할만큼 작다. 공격자가 첫 번째 경우로 세션키에 대한 정보를 얻을 확률이 클 경우에는, 이 공격자를 사용해서 CDH문제를 풀 수 있음을 보일수 있다.

실험에서 세션 식별자가 반복되는 사건을 col이라고 하자. 그리고 공격자가 $W=g^{x_ix_j}$를 가지고서 $H(i\|j\|*\|W)$ 오

2) 이 정의는 논문들에서 사용되는 전방위 안전성의 두 가지 정의들 중 약한 것으로서 [14]에서 정의된 것이다. 더 강한 전방위 안전성 정의에서는 만약 공격자가 Corrupt(i) 쿼리를 한 후에, Send(i,k,*) 쿼리를 하지 않으면 프레쉬한 오라클로 간주된다.

라클 쿼리를 던지는 사건을 query라고 하자. $Pr_A['b=b \mid \overline{query} \wedge \overline{col}] = \frac{1}{2}$임을 이용해서, $\left|Pr_A['b=b]-\frac{1}{2}\right| \leq \frac{1}{2}\left(Pr_A[query \wedge \overline{col}]+Pr_A[col]\right)$이 됨을 알 수 있다. 생일문제(birthday problem)와 연관해서 $Pr_A[col] \leq \frac{q_s^2}{2^k}$임을 쉽게 알 수 있다. 확률 $Pr_A[query \wedge \overline{col}]$의 상한선을 위해서 먼저 사건 $query \wedge \overline{col}$이 발생하는 경우에 공격자의 성공 확률로 CDH 문제를 푸는 알고리즘 F를 구성할 수 있음을 보인다. F는 입력으로 CDH 문제의 인스턴스인 $(U_1 = g^{u_1}, U_2 = g^{u_2})$을 받는다. F는 공격자에게 키 교환 시스템을 시뮬레이션한다. 시뮬레이션을 할 때 먼저 랜덤하게 두 사용자를 뽑아서 U_1, U_2을 그들의 공개키로 놓는다. F는 해쉬 오라클을 시뮬레이션하며 공격자의 해쉬 쿼리로부터 $g^{u_1 u_2}$을 찾으려고 한다. 좀 더 구체적으로 F는 다음과 같이 행동한다.

1. F는 U_1, U_2을 입력으로 받아서 i^*, j^*을 1부터 N사이에 뽑는다. U_1을 P_{i^*}의 공개키로 놓고 U_2을 P_{j^*}의 공개키로 놓는다. 다른 사용자들의 개인키-공개키는 F 스스로 생성한다.
2. F는 공격자 알고리즘 A에게 사용자들의 공개키를 제공하면서 키 교환 시스템을 시뮬레이션한다. A의 쿼리들에 대해서 다음과 같이 응답한다.
   - 쿼리 H(i||j||sid||W)를 받으면 난수 v를 돌려준다. 만약 $i=i^*, j=j^*$이면 W를 리스트 dh-tuples에 저장한다.
   - 쿼리 Initiate(i,j)를 받으면 난수 r_i를 뽑는다. 만약 이것이 기존의 난수들과 중복되면 실험을 중단한다. 중복되지 않는다면 r_i를 A에게 돌려준다.
   - 쿼리 Send(i,k,M)을 받으면 임의의 난수를 세션키 sk_i^k로 할당한다.
   - 쿼리 Execute(i,j)을 받으면 임의의 난수 r_i, r_j를 만든다. 만약 이것들이 기존의 난수와 중복되면 실험을 중단한다. 중복되지 않으면 다시 임의의 난수를 뽑아서 $sk_i^k = sk_j^{k'}$놓고, r_i, r_j를 A에게 돌려준다.
   - Reveal 쿼리나 Test 쿼리는 기존의 키 교환 실험에서와 같이 수행한다.
3. A가 종료한후에 리스트 dh-tuples에서 하나의 원소를 랜덤하게 뽑아서 출력한다.

사건 query가 일어나기 전까지 F는 A에게 실험을 완벽하게 똑같이 시뮬레이션한다. 따라서 F가 CDH 문제에 대해서 올바른 답을 줄 확률은 $Pr_A[query \wedge \overline{col}]/N^2 q_h$보다 크다. 그러므로 $Pr_A[query \wedge \overline{col}] \leq q_h N^2 e$이 됨을 알 수 있다.

3.2 프로토콜 TS2

우리는 쉽게 TS1은 전방위 안전성을 제공하지 않는다는 것을 알 수 있다. 전방위 안전성은 일회용 Diffie-Hellman 키교환을 TS1에 추가함으로써 얻을 수 있다. TS2는 다음과 같다. 우리는 P_i의 입장에서 프로토콜을 기술하지만, 파트너인 P_j도 유사하게 행동한다.

Setup: TS1과 같다.

Round1: P_i는 난수 α_i를 Z_q로부터 뽑아서, P_j에게 g^{α_i}를 보낸다.

세션키 계산: P_i는 세션식별자를 만들기 위해서, 브로드캐스트된 메시지를 P_i와 P_j의 순서에 의해서 결합한다. 즉, 세션 식별자는 $sid_i = g^{\alpha_i} \| g^{\alpha_j}$이다. P_i는 세션키 $sk_i = H(i\|j\|sid_i\|(g^{\alpha_j})^{\alpha_i}\|y_j^{x_i})$를 만든다.

TS2의 실행 예가 그림 2에 나와 있다.

	$P_1(x_1, y_2)$	$P_2(x_2, y_1)$
Round 1	g^{α_1}	g^{α_2}

$sid = g^{\alpha_1} \| g^{\alpha_2}$

$sk = H(1\|2\|sid\|g^{\alpha_1 \alpha_2}\|g^{x_1 x_2})$

그림 2 TS2의 실행 예

다음의 정리는 TS2의 안전성에 관한 것이다.

정리 2. 만약 G에서 CDH(computational Diffie-Hellman) 문제가 어려운 문제면, TS2는 랜덤오라클 모델에서 키 독립성과 전방위 안전성을 보장한다. 구체적으로 G가 CDH 문제에 대해서 (t,ϵ) 안전성을 보장한다면, $Adv_{TS2}^{KIFS}(k,t,q_{re},q_{co},q_h) \leq q_h(N^2+1)\epsilon + \frac{q_s^2}{q}$ 이다.

여기서 t는 공격자의 실행시간을 포함한 총 실험시간이며, 공격자는 q_{re} Reveal 쿼리와 q_{co} Corrupt 쿼리 그리고 q_h 해쉬 쿼리를 던진다. N은 실험에서 최대 사용자의 수이며, q_s는 실험에서 최대 세션의 수이다. CDH 문제에 대한 (t,ϵ) 안전성이란 어떤 공격자도 시간 t안에 ϵ이상의 확률로 CDH문제를 풀 수 없다는 가정이다.

증명. 랜덤 오라클 모델에서는 공격자가 세션키 $sk_i^k = H(i\|j\|sid_i\|(g^{\alpha_j})^{\alpha_i}\|y_j^{x_i})$에 대한 정보를 얻을 수 있는 방법이 두 가지 밖에 없다. 첫 번째는 $i\|j\|sid_i\|(g^{\alpha_j})^{\alpha_i}\|y_j^{x_i}$를 해쉬 오라클에게 쿼리하는 것이고, 두 번째는 세션 식별자 sid_i^k가 실험에서 반복되는 경우이다. 하지만 sid_i^k

가 반복될 확률은 $\frac{q_s^2}{q}$로 무시할만큼 작다. 여기서 q는 그룹 G의 크기이다. 공격자가 첫 번째 경우로 세션키에 대한 정보를 얻을 확률이 클 경우에는, 이 공격자를 사용해서 CDH문제를 풀 수 있음을 보일수 있다.

실험에서 세션 식별자가 반복되는 사건을 col이라고 하자.

공격자가 Corrupt 쿼리를 던졌던 사용자가 가진 오라클 중 하나나 그의 파트너 오라클에게 Test 쿼리를 던지는 사건을 corrupt이라고 하자. 이런 사건은 프레쉬 오라클의 정의상 Execute 쿼리를 통해서 만들어진 오라클에게 Test 쿼리가 던져진다는 것을 알 수 있다. 실험 도중에 P_i, P_j가 Corrupt되지 않고, 또한 공격자가 $W = g^{x_i x_j}$를 가지고서 $H(i\|j\|U\|V\|X\|W)$ 오라클 쿼리를 던지는 사건을 query1이라고 하자. 공격자가 $H(i\|j\|U\|V\|X\|W)$ 쿼리를 던지며, Execute 쿼리를 통해서 만들어진 오라클 Π_i^k가 존재해서 $sid_i^k = U\|V = g^{\alpha_i}\|g^{\alpha_j}$이며 $X = g^{\alpha_i \alpha_j}$인 사건을 query2라고 하자.

$$Pr_A[b' = b \mid \overline{query1} \wedge \overline{col} \wedge \overline{corrupt}]$$
$$= Pr_A[b' = b \mid \overline{query2} \wedge \overline{col} \wedge \overline{corrupt}] = \frac{1}{2}$$

임을 이용해서,

$$\left| Pr_A[b' = b] - \frac{1}{2} \right| \le \frac{1}{2}(Pr_A[col] + Pr_A[query1 \wedge \overline{corrupt}] + Pr_A[query2 \wedge corrupt])$$

이 됨을 알 수 있다. 생일문제(birthday problem)와 연관해서 $Pr_A[col] \le \frac{q_s^2}{q}$임을 쉽게 알 수 있다. 정리 1의 증명과 똑같은 이유로 인해서 우리는 $Pr_A[query1 \wedge \overline{corrupt}] \le q_h N^2 \epsilon$임을 알 수 있다. 확률 $Pr_A[query2 \wedge corrupt]$의 상한선을 위해서 먼저 사건 $query2 \wedge corrupt$이 발생하는 경우에 공격자의 성공 확률로 CDH 문제를 푸는 알고리즘 F를 구성할 수 있음을 보인다. F는 입력으로 CDH 문제의 인스턴스인 $(U_1 = g^{u_1}, U_2 = g^{u_2})$을 받는다. F는 공격자에게 키 교환 시스템을 시뮬레이션한다. 시뮬레이션을 할 때 F는 U_1, U_2을 랜덤 셀프 리듀서빌러티(random self-reducibility)를 사용해서 모든 Execute 쿼리에 심는다. F는 해쉬 오라클을 시뮬레이션하며 공격자의 해쉬 쿼리로부터 $g^{u_1 u_2}$을 찾으려고 한다. 좀 더 구체적으로 F는 다음과 같이 행동한다.

1. F는 U_1, U_2을 입력으로 받는다. 사용자들의 개인키-공개키는 F 스스로 생성한다.
2. F는 공격자 알고리즘 A에게 사용자들의 공개키를 제공하면서 키 교환 시스템을 시뮬레이션한다. A의 쿼리들에 대해서 다음과 같이 응답한다.
   - 쿼리 H(i||j||U||V||X||W)를 받으면 난수 v를 돌려준다. 만약 $sid_i^k = U\|V$를 가지는 오라클 Π_i^k가 존재하며 이 오라클이 Execute 오라클을 통해서 생성되었으면, 이면 U||V||X를 리스트 dh-tuples에 저장한다.
   - 쿼리 Initiate, Send, Reveal, Test, Corrupt 쿼리는 실험에서와 같이 실행된다.
   - 쿼리 Execute(i,j)을 받으면 임의의 난수 r_i, r_j를 선택하며, $U_1 g^a \| U_2 g^b$을 돌려준다. 그리고 임의의 난수를 뽑아서 $sk_i^k = sk_j^{k'}$놓는다.
3. A가 종료한후에 리스트 dh-tuples에서 하나의 행 (U,V,X)을 랜덤하게 뽑는다. $U = U_1 g^a, V = U_2 g^b$를 만족하는 a,b를 가지고서 $X / U_2^a U_1^b g^{ab}$를 계산해서 출력한다.

사건 query2가 일어나기 전까지 F는 A에게 실험을 완벽하게 똑같이 시뮬레이션한다. 따라서 F가 CDH 문제에 대해서 올바른 답을 줄 확률은 $Pr_A[query \wedge \overline{col}]/q_h$ 보다 크다. 그러므로 $Pr_A[query \wedge \overline{col}] \le q_h \epsilon$이 됨을 알 수 있다.

3.3 프로토콜 TS3

TS2의 안전성은 랜덤 오라클 모델에서 증명되었다. TS3의 안전성은 표준모델에서 증명이 되며, DDH (decisional Diffie-Hellman) 문제에 기반하고 있으며 다음과 같다. 우리는 P_i의 입장에서 프로토콜을 기술하지만, 파트너인 P_j도 유사하게 행동한다.

Setup: TS1과 같다.

Round1: P_i는 MAC 키인 $k_{i,j} = y_j^{x_i}$을 만든다. P_i는 난수 α_i를 Z_q로부터 뽑아서 $\tau_i \leftarrow MAC_{k_{ij}}(i\|j\|g^{\alpha_i})$를 계산한 다음, P_j에게 $g^{\alpha_i}\|\tau_i$를 보낸다.

세션키 계산: P_i는 $k_{i,j}$를 이용해서 τ_j가 올바른 MAC 값인지를 확인한다. 만약 옳지 않다면 세션키는 만들어지지 않는다. MAC 값이 올바르면 세션키 $sid_i = (g^{\alpha_j})^{\alpha_i}$를 만든다. 세션 식별자는 $sid_i = g^{\alpha_i}\|\tau_i\|g^{\alpha_j}\|\tau_j$이다.

TS3의 실행 예가 그림 3에 나와 있다.

다음의 정리는 TS3의 안전성에 관한 것이다.

정리 3. 만약 MAC 스킴이 위조 불가능성을 만족하고, G에서 DDH(decisional Diffie-Hellman) 문제가 어려운 문제면, TS3은 표준 모델에서 키 독립성과 전방위 안전성을 보장한다. 구체적으로 MAC 스킴이 (t, q, ϵ) 위조 불가능성을 보장하며, G가 DDH 문제에 대해서

	$P_1(x_1,y_2)$	$P_2(x_2,y_1)$
Round 1	$g^{\alpha_1}\|\|\tau_1$	$g^{\alpha_2}\|\|\tau_2$

$$k_{1,2}=g^{x_1x_2}$$

$$\tau_1 \leftarrow Mac_{k_{1,2}}(1\|\|2\|\|g^{\alpha_1});\tau_2 \leftarrow Mac_{k_{1,2}}(2\|\|1\|\|g^{\alpha_2})$$

$$sid=g^{\alpha_1}\|\|\tau_1\|\|g^{\alpha_2}\|\|\tau_2$$

$$sk=g^{\alpha_1\alpha_2}$$

그림 3 TS3의 실행 예

(t,ϵ') 안전성을 보장한다면,

$$Adv_{TS3}^{KIFS}(k,t,q_{re},q_{co}) \le (4+2N^2+2q_s^2)\epsilon'+2N^2\epsilon+\frac{2q_s^2}{q}$$

이다. 여기서 t는 공격자의 실행시간을 포함한 총 실험 시간이며, 공격자는 q_{re} Reveal 쿼리와 q_{co} Corrupt 쿼리를 던진다. N은 실험에서 최대 사용자의 수이며, q_s는 실험에서 최대 세션의 수이다. MAC 스킴이 (t,q,ϵ) 위조 불가능성을 보장한다는 것은 어떤 공격자도 시간 t와 MAC 오라클 쿼리횟수 q를 가지고서는 ϵ 이상의 확률로 MAC을 위조할 수 없음을 말한다. DDH 문제에 대한 (t,ϵ') 안전성이란 어떤 공격자도 시간 t안에 ϵ'이상의 확률로 DDH문제를 풀 수 없다는 가정이다.

증명. g^{α_i}가 두 번 반복되는 사건을 col이라고 하자. Execute 쿼리에 의해서 생성된 오라클에 Test 쿼리가 발생한 사건을 Ex라고 하자. 그리고 MAC 위조가 발생한 사건을 Forge라고 하자. 그러면

$$\left|Pr_A[b'=b]-\frac{1}{2}\right| \le Pr_A[col]+Pr_A[Forge]$$
$$+\ Pr_A[b'=b\wedge\overline{col}\wedge Ex]+Pr_A[b'=b\wedge\overline{col}\wedge\overline{Forge}\wedge\overline{Ex}]$$

임을 알 수 있다. 앞에서와 마찬가지로 $Pr_A[col] \le \frac{q_s^2}{q}$이 된다. $Forge_{i,j}$가 사용자 i,j에 대해서 Forge가 발생할 사건이라고 하자. $Pr_A[Forge] \le N^2 Pr_A[Forge_{i,j}]$가 된다. 만약 $k_{i,j}=g^{x_ix_j}$대신에 난수를 사용한다면 $Pr_A[Forge_{i,j}]$는 ϵ'만큼만 영향을 받게 된다. 한편 $k_{i,j}=g^{x_ix_j}$ 대신에 난수를 사용한다면, MAC의 안전성에 의해서 $Pr_A[Forge_{i,j}]$는 ϵ에 의해서 바운드됨을 알 수 있다. 따라서 $Pr_A[Forge] \le N^2(\epsilon+\epsilon')$이 된다. 다음으로 $Pr_A[b'=b\wedge\overline{col}\wedge Ex] \le 2\epsilon'$가 됨을 증명한다. 다음 알고리즘 F는 DDH 인스턴스를 Execute 쿼리의 메시지에 삽입하며, 다음과 같이 행동한다.

1. (g,X,Y,Z)를 입력으로 받아서 랜덤 셀프 리듀서빌러티를 사용해서 많은 (g,X_i,Y_i,Z_i)를 만들 수 있다. (정리 2의 증명 참고.)
2. Execute(i,j) 쿼리를 받으면 (g,X_i,Y_i,Z_i)를 받아서 $X_i\|\|Y_i\|\|\tau'$를 돌려주며, 세션키로 Z_i를 놓는다.
3. 모든 다른 쿼리들에 대해서는 실험에서와 같이 응답한다.
4. 만약 A가 Test 쿼리에서 사용된 동전을 정확히 맞추고 Ex 사건이 일어나면 1을 출력한다. 그렇지 않다면 0 또는 1을 임의로 출력한다. 만약 F의 입력값이 DH 쌍이었다면, F는 완벽히 실험을 시뮬레이션함을 알 수 있다. 이런 경우에 F가 1을 출력할 확률은 $Pr_A[b'=b\wedge\overline{col}\wedge Ex]+\frac{1}{2}(Pr[b'\neq b\vee\overline{Ex}\vee col])$이 된다. 한편 F의 입력값이 DH 쌍이 아닌 랜덤 쌍이었다면 F가 1을 출력할 확률은 정확히 1/2가 된다. 그러므로 두 확률의 차이는 ϵ'에 의해서 바운드됨을 알 수 있다. 다음으로 $Pr_A[b'=b\wedge\overline{col}\wedge\overline{Forge}\wedge\overline{Ex}] \le q_e^2\epsilon'$이 됨을 증명한다. 다음처럼 행동하는 F를 생각해보자.

1. F는 (g,X,Y,Z)를 입력으로 받는다. 다음에 t_1,t_2를 1부터 q_s사이서 뽑는다. F는 모든 사용자들을 위한 공개키-개인키 쌍을 생성한다.
2. F는 A를 실행시키면서 다음과 같이 실험을 시뮬레이션 해준다.
   - Initiate 쿼리는 t_1,t_2번째 쿼리를 제외하고는 정상적으로 수행된다. t_1번째 Initiate(i*,j*)에 대해서 X와 MAC값인 t를 돌려준다. 이때의 오라클을 Π_{i*}^{k}라고 하자. t_2번째 Initiate(j',i')를 받으면 i*과 i'이 같고 j*와 j'이 같은지를 확인한다. 같지 않다면 F는 임의의 값을 출력하고 종료한다. 같다면 Y와 MAC값인 t'을 돌려준다. 이때의 오라클을 $\Pi_{j*}^{k'}$이라고 하자.
   - Send 쿼리는 Send(j*,k,Y*||t*)와 Send(j*,k',M')을 제외하고는 정상적으로 수행된다. 여기서는 두 쿼리에 대해서 비슷하게 처리되므로 첫 번째 Send 쿼리에 대해서만 설명한다. 만약 Y*||t*=Y||t'이라면 Π_{i*}^{k}의 세션키로 Z를 할당하며, 이런 오라클을 good이라고 하자. 만약 t*가 올바르지 않은 값이라면 세션키값을 할당하지 않는다. t*가 올바르지만 Y*||t*가 j*가 이전에 출력한 메시지와 일치하지 않으면 Forge가 발생한 것이므로 F는 종료한다. 마지막으로 Y*||t*가 j*가 출력한 값이 맞으면 F는 올바른 세션키값을 출력할 수 있다.
   - Execute 쿼리는 정상적으로 수행된다.
   - Reveal 쿼리나 Test 쿼리는 다음을 제외하고는 정상적으로 처리된다. 만약 Reveal 쿼리가 good 오라클에게 행해지면 F는 종료한다. 만약 Test 쿼리가

good 오라클이외의 다른 오라클에게 행해지면 F는 종료한다.

- Corrupt(i*)나 Corrupt(j*) 쿼리가 발생하면 F는 종료한다.

3. col이나 Ex 사건이 발생하지 않고, A가 Test 쿼리의 코인을 정확히 맞추면 1을 출력하며, Good 사건이 발생했다고 하자. 다른 경우에는 랜덤 비트를 출력한다.

F가 1을 출력할 확률은 $Pr_A[Good]+\frac{1}{2}(1-Pr_A[Good])$ 이 된다. 만약 F의 입력값이 DH 쌍이었다면 $Pr_A[b'=b\wedge\overline{col}\wedge\overline{Forge}\wedge\overline{Ex}]/q_s^2 \le Pr[Good]$이 된다. 만약 F의 입력값이 랜덤 쌍이었다면 $Pr_A[Good]$의 값은 1/2이 된다. 그러므로 $Pr_A[b'=b\wedge\overline{col}\wedge\overline{Forge}\wedge\overline{Ex}\le q_s^2 e'$ 이 됨을 알 수 있다.

참 고 문 헌

[1] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. Adv. in Cryptology-Crypto '93.

[2] S. Blake-Wilson, D. Johnson, and A. Menezes. Key Agreement Protocols and their Security Analysis. 6th IMA Intl. Conf. on Cryptography and Coding, 1997.

[3] M. Bellare, R. Canetti, and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. STOC '98.

[4] D. Denning and G. M. Sacco. Timestamps in Key Distribution Protocols. Comm. ACM 24(8): 533-536 (1981).

[5] W. Diffie, P. van Oorschot, and M. Wiener. Authentication and Authenticated Key Exchanges. Designs, Codes, and Cryptography 2(2): 107-125 (1992).

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

[7] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An Efficient Protocol for Authenticated Key Agreement. Technical report CORR 98-05, University of Waterloo, 1988.

[8] T. Matsumoto, Y. Takashima, and H. Imai. On Seeking Smart Public-Key Distribution Systems. Trans. of the IECE of Japan, E69, pp.99-106, 1986.

[9] National Security Agency. SKIPJACK and KEA Algorithm Specification. Version 2.0, May 29, 1998.

[10] S. Blake-Wilson and A. Menezes. Authenticated Diffie-Hellman Key Agreement Protocols. Selected Areas in Cryptography, 1998.

[11] R. Ankney, D. Johnson, and M. Matyas. The Unified Model. Contribution to ANSI X9F1, October 1995.

[12] C. Boyd and J.M.G. Nieto. Round-Optimal Contributory Conference Key Agreement. Public Key Cryptography, 2003.

[13] J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. Adv. in Cryptology - Crypto 2003.

[14] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key agreement secure against dictionary attacks. In Proc. of EUROCRYPT '00.

[15] G. Ateniese, M. Steiner, and G. Tsudik. New Multi-Party Authentication Services and Key Agreement Protocols. IEEE J. on Selected Areas in Communications 18(4): 628-639 (2000).

[16] M. Bellare, A. Boldyreva, and S. Micali. Public-Key Encryption in a Multi-User Setting: Security Proofs and Improvements. Adv. in Cryptology - Eurocrypt 2000.

[17] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. Systematic Design of Two-Party Authentication Protocols. IEEE J. on Selected Areas in Communications 11(5): 679-693 (1993).

[18] C. Boyd. On Key Agreement and Conference Key Agreement. ACISP 1997.

[19] E. Bresson, O. Chevassut, and D. Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange-The Dynamic Case. Adv. in Cryptology-Asiacrypt 2001.

[20] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. Adv. in Cryptology-Eurocrypt 2002.

[21] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. ACM Conf. on Computer and Communications Security, 2001.

[22] M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. Advances in Cryptology - Eurocrypt '94.

[23] R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. Adv. in Cryptology - Eurocrypt 2002.

[24] I. Ingemarasson, D.T. Tang, and C.K. Wong. A Conference Key Distribution System. IEEE Trans. on Information Theory 28(5): 714-720 (1982).

[25] M. Just and S. Vaudenay. Authenticated Multi-Party Key Agreement. Adv. in Cryptology-Asiacrypt '96.

[26] V. Shoup. On Formal Models for Secure Key Exchange. Available at http://eprint.iacr.org.

[27] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman Key Distribution Extended to Group Communication. ACM Conf. on Computer and

Communications Security, 1996.
[28] W.-G. Tzeng. A Practical and Secure-Fault-Tolerant Conference-Key Agreement Protocol. Public Key Cryptography, 2000.

정 익 래

1998년 2월에 고려대학교 전산학과 학사 학위 취득. 2000년 2월에 고려대학교 전산학과 석사 학위 취득. 2004년 8월에 고려대학교 정보보호대학원 박사 학위 취득. 그후 2005년 9월까지 Maryland 대학에서 포닥연수를 했고, 현재 고대 정보보호대학원에서 포닥 연구원임. 관심분야는 암호 프로토콜, 암호이론, 계산이론

이 동 훈

1984년에 고려대학교 경제학과 학사 학위 취득. 1987년에 Oklahoma 대학에서 전산학과 석사 학위 취득. 1992년에 Oklahoma 대학에서 전산학과 박사 학위 취득. 2000년부터 고려대학교 정보보호대학원 교수. 관심분야는 암호이론, 암호 프로토콜, 정보이론