通信情報保護學會論文誌
第 10 券 第 4 號, 2000. 12

IPsec의 키 교환 방식에 대한 안전성 분석*

주 한 규**

Analysis of the IPsec Internet Key Exchange (IKE) Protocol

Hankyu Joo**

요 약

IPsec은 인터넷으로 연결된 컴퓨터 사이의 통신을 보호하기 위한 프로토콜로서 많은 가상 사설망 (VPN, Virtual Private Network)은 IPsec 프로토콜을 사용한다. IKE 프로토콜은 IPsec에서 키 교환을 위해 사용하는 프로토콜이다. 정형화된 분석 방법이 컴퓨터 공학 분야에서 많이 사용되고 있으며, 정형화된 분석 방법을 통하여 시스템의 안전성을 높일 수 있다. 본 연구에서는 IPsec의 키 교환 방식인 IKE 프로토콜을 정형화된 방법으로 분석하여 그 안전성을 파악하였다. 분석 결과에 의하면 전자 서명을 통하여 인증을 수행하는 IKE와 공유키를 이용하여 인증을 수행하는 IKE는 안전하게 키 교환을 할 수 있다. 그러나 공개키를 사용하여 인증하는 경우와 변경된 공개키를 사용하여 인증하는 경우에는 상대방의 공개키에 대한 확신이 필요하며 인증서를 사용함이 안전하다.

ABSTRACT

IPsec is a protocol suite to protect the data communication between computers on internet and many VPNs(Virtual Private Networks) use IPsec protocol. IKE protocol is used to exchange keys in IPsec. Formal analysis method is used increasingly in computer science to increase the reliability of a system. In this paper, the IKE protocol is analyzed formally. This paper shows that IKE with Authentication with Signature and Authentication with Pre-Shared Key is safe, but Authentication with Public Key Encryption and A Revised Method of Authentication with Public Key Encryption are safe only with the assumption that a participant has the correct public key of the correspondent. To make sure that a participant has the correct public key of the correspondent, the usage of certificate is recommended.

keyword : *IPsec, IKE protocol, BAN logic, formal analysis*

Ⅰ. 서 론

인터넷(internet)은 TCP/IP 프로토콜을 이용하여 통신하는 컴퓨터들이 어느 장소에서든지 네트웍을 이용하여 연결된 다른 컴퓨터와 통신할 수 있도록 해준다. 그러나 인터넷이라는 공중망(public network)을 통과하는 데이터는 공공에 노출되어 있어 그 통신 내용에 기밀성이 없으며 침입자에 의해 통신 내용이 도중에 바뀔 수도 있다. 이에 대한 해결책으로 IPsec이 고안되었다. IPsec을 이용함으로써, 인증(authentication), 무결성(integrity), 기밀성(confidentiality), 접근제어(access control), 그리고 재전송 공격(replay attack)에 대한 보호 등의 보안 기능을 가질 수 있다[1].

* 본 연구는 2000년도 한림대학교 학술연구조성비에 의하여 수행되었습니다.
** 한림대학교 정보통신공학부(hkjoo@sun.hallym.ac.kr)

IPsec은 IKE(Internet Key Exchange) 프로토콜[2]을 이용하여 교신하는 두 참가자 사이의 키와 필요한 정보를 교환한다. 따라서 IKE의 안전성은 IPsec의 안전성을 위하여 필수적이다. IKE는 두 단계로 나뉜다. 첫 단계는 ISAKMP SA라고 불리는 키와 정보를 교환하는 단계이며 첫 단계에서 교환된 키와 정보에 의해 암호화된 상태에서 두 번째 IKE 단계가 수행된다. 두 번째 IKE 단계에서 교환된 키와 정보에 의해 실제 교신되는 데이터가 보호된다.

정형적 분석 방법은 시스템의 안전성 분석을 위하여 그 사용이 증가되고 있다[3]. 정형적인 방법을 통해 올바름이 증명된 시스템도 오류가 없다고 보장할 수는 없으나 그렇지 않은 시스템보다 안전도가 증가한다고 할 수는 있다. 정형적 방법은 보안 프로토콜들의 안전성 분석을 위하여서도 사용될 수 있다. 실제 다양한 정형적 방법에 의하여 보안 프로토콜들이 분석되었다. BAN logic을 이용한 분석으로 CCITT X.509, Andrew RPC 등의 키 교환의 오류가 발견되었다[4]. Abstract State Machine를 이용하여 Kerberos가 분석되었으며[5], Murϕ를 이용하여 SSL 3.0이 분석되었다[6]. 또한 Denial of Service 공격 방지를 위한 프로토콜이 정형적인 방법으로 접근되었다[7]. 시스템의 안전성 분석과 마찬가지로, 이들 분석을 통하여 프로토콜의 안전성이 보장되는 것은 아니나 안전성을 증대시킬 수는 있다.

BAN *logic*[4]은 인증에 대한 logic으로 기본적인 키 교환의 안전성을 분석할 수 있다. IPsec의 IKE 프로토콜은 기본적인 BAN logic만을 이용하여 키 교환의 안전성을 분석할 수는 없으나, BAN logic을 확장하여 그 안전성을 분석할 수 있다. 본 고에서는 확장된 BAN logic을 이용하여 IPsec의 IKE 프로토콜의 안전성을 분석한다.

II. IPsec의 IKE 프로토콜

IKE는 크게 두 단계로 나뉜다. 첫 단계는 ISAKMP SA라고 불리는 키와 정보를 교환하는 단계이며 첫 단계에서 교환된 키와 정보에 의해 암호화된 상태에서 두 번째 IKE 단계가 수행된다. 첫 번째 단계는 인증 방법에 따라 네 가지의 방법이 있다 - 서명에 의한 인증(*Authentication with Signature*), 공개키에 의한 인증(Authentication with Public Key Encryption), 변경된 공개키에 의한 인증(A Revised Method of Authentication with Public Key Encryption), 그리고 공유키에 의한 인증(Authentication with Pre-Shared Key). 두 번째 IKE 단계에서 교환된 키와 정보에 의해 실제 교신되는 데이터가 보호된다. 각 키 교환 방법은 다음과 같다.

2.1 서명에 의한 인증

서명에 의한 인증 방법을 사용하는 1 단계 IKE에서는 Diffie-Hellman 키 교환 방법[8]을 이용하여 키 교환을 한다. 키 교환 후 올바른 상대방과 키를 성공적으로 교환하였음을 확인하기 위하여 교환된 정보에 전자 서명을 하여 이를 교환한다. 기본적인 프로토콜은 다음과 같다.

Initiator		Responder
1: HDR, SA	→	
2:	←	HDR, SA
3: HDR, KE, Ni	→	
4:	←	HDR, KE, Nr
5: HDR*, IDii, [CERT,] SIG_I	→	
6:	←	HDR*, IDir, [CERT,] SIG_R

여기에서 SIG_I와 SIG_R은 각각 HASH_I, HASH_R에 전자 서명[9]을 한 것이며 HASH_I와 HASH_R은 각각 다음과 같이 생성된다.

$$SKEYID = prf(Ni\_b \mid Nr\_b, g^{xy})$$
$$HASH\_I = prf(SKEYID, g^{xi} \mid g^{xr} \mid CKY\text{-}I \mid CKY\text{-}R \mid SAi\_b \mid IDii\_b)$$
$$HASH\_R = prf(SKEYID, g^{xr} \mid g^{x\,I} \mid CKY\text{-}R \mid CKY\text{-}I \mid SAi\_b \mid IDir\_b)$$

2.2 공개키에 의한 인증

공개키를 사용하는 1 단계 IKE에서 키 교환은 Diffie-Hellman 키 교환 방법을 사용한다. 새로 생성된 난수가 상대방의 공개키에 의해 암호화되어 전달되며, 올바른 상대방과 성공적으로 키 교환을 하였음을 확인하기 위하여 교환된 난수를 기반으로하여 생성된 키로 교환된 정보를 해쉬하여 그 해쉬값을 교환한다. 다음은 공개키에 의한 인

증 방법을 사용하는 1 단계 IKE의 프로토콜을 보여준다.

Initiator		Responder
1: HDR, SA	→	
2:	←	HDR, SA
3: HDR, KE, [HASH(1),] ⟨IDii_b⟩PubKey_r, ⟨Ni_b⟩PubKey_r	→	
4:	←	HDR, KE, ⟨IDii_b⟩PubKey_r, ⟨Ni_b⟩PubKey_r
5: HDR*, HASH_I	→	
6:	←	HDR*, HASH_R

인증을 위해 사용되는 HASH_I와 HASH_R은 다음과 같이 생성된다.

SKEYID=prf(hash(Ni_b | Nr_b), CKY-I | CKY-R)
HASH_I=prf(SKEYID, g^{xi} | g^{xr} | CKY-I | CKY-R | SAi_b | IDii_b)
HASH_R=prf(SKEYID, g^{xr} | g^{xi} | CKY-R | CKY-I | SAi_b | IDir_b)

2.3 변경된 공개키에 의한 인증

변경된 공개키에 의한 인증 방법을 사용하는 1 단계 IKE에서 난수는 상대방의 공개키에 의해 암호화되어 전달된다. 키 교환은 Diffie-Hellman 키 교환 방법을 사용하며, Diffie-Hellman의 공개정보는 이 난수로부터 생성된 키(Ke_i, Ke_r)에 의해 암호화 되어 있다. 인증 방법은 공개키에 의한 인증 방법과 같다. 다음은 변경된 공개키에 의한 인증 방법을 사용하는 1 단계 IKE의 프로토콜을 보여준다.

Initiator		Responder
1: HDR, SA	→	
2:	←	HDR, SA
3: HDR, [HASH(1),] ⟨Ni_b⟩PubKey_r, ⟨KE_b⟩Ke_i, ⟨IDii_b⟩Ke_i, [⟨Cert-I_b⟩Ke_i]	→	
4:	←	HDR, ⟨Ni_b⟩PubKey_r, ⟨KE_b⟩Ke_r, ⟨IDii_b⟩Ke_r
5: HDR*, HASH_I	→	
6:	←	HDR*, HASH_R

다음은 인증을 위해 사용되는 HASH_I와 HASH_R, 그리고 공개 정보 암호화에 사용되는 키 Ke_i와 Ke_r의 생성을 보여 준다.

SKEYID=prf(hash(Ni_b | Nr_b), CKY-I | CKY-R)
HASH_I=prf(SKEYID, g^{xi} | g^{xr} | CKY-I | CKY-R | SAi_b | IDii_b)
HASH_R=prf(SKEYID, g^{xr} | g^{xi} | CKY-R | CKY-I | SAi_b | IDir_b)
Ne_i=prf(Ni_b, CKY-I)
Ne_r=prf(Nr_b, CKY-R)

Ke_i, Ke_r은 Ne_i, Ne_r로부터 필요한 부분을 취함으로 생성된다.

2.4 공유키에 의한 인증

공유키에 의한 인증 방법을 사용하는 1 단계 IKE 또한 Diffie-Hellman 키 교환 방법을 사용한다. 올바른 상대방과 키 교환이 정상적으로 되었음을 확인하기 위하여 공유키로부터 생성된 키로 교환된 정보를 해쉬하여 그 해쉬값을 교환한다. 다음은 공유키에 의한 인증 방법을 사용하는 1 단계 IKE의 프로토콜을 보여준다.

Initiator		Responder
1: HDR, SA	→	
2:	←	HDR, SA
3: HDR, KE, Ni	→	
4:	←	HDR, KE, Nr
5: HDR*, IDii, HASH_I	→	
6:	←	HDR*, IDir, HASH_R

인증을 위해 사용되는 HASH_I와 HASH_R은 다음과 같이 생성된다.

SKEYID=prf(pre-shared-key, Ni_b | Nr_b)
HASH_I=prf(SKEYID, g^{xi} | g^{xr} | CKY-I | CKY-R | SAi_b | IDii_b)
HASH_R=prf(SKEYID, g^{xr} | g^{xi} | CKY-R | CKY-I | SAi_b | IDir_b)

2.5 2 단계 IKE

2 단계 IKE는 1 단계 IKE에서 교환된 키와 정보에 의하여 보호된 상태에서 진행된다. 2 단계 IKE에

의해 생성된 키와 정보에 의해 실제 교환되는 데이터는 보호된다. 2 단계 IKE에서는 기본적으로 난수를 생성하여 교환하며 교환된 난수와 1 단계 IKE에서 생성된 키를 이용하여 실제 데이터 보호에 필요한 키를 생성한다. 올바른 상대방과 난수 교환이 정확히 이루어 졌음을 확인하기 위하여 해쉬 함수를 사용한다. 해쉬 함수에 사용되는 키는 1 단계 IKE에 의해 생성된 공유키이다. 다음은 2 단계 IKE의 프로토콜을 보여준다.

Initiator		Responder
1: HDR*, HASH_1, SA, Ni [, KE] [, IDci, IDcr]	→	
2:	←	HDR*, HASH_2, SA, Nr [, KE] [, IDci, IDcr]
3: HDR*, HASH_3	→	

인증을 위해 사용되는 해쉬 함수는 다음과 같이 구하여 진다. 해쉬 키인 SKEYID_a는 1 단계 IKE에서 생성되어 공유된 값이다.

HASH_1=prf(SKEYID_a, M-ID | SA | Ni [| KE] [| IDci | IDcr])
HASH_2=prf(SKEYID_a, M-ID | Ni_b | SA | Nr [| KE] [| IDci | IDcr])
HASH_3=prf(SKEYID_a, 0 | M-ID | Ni_b | Nr_b)

Ⅲ. BAN logic

BAN logic은 인증에 대한 logic으로, 이를 이용하여 기본적인 키 교환에 대한 안전성 분석을 할 수 있다. BAN logic의 기본적인 표기법은 다음과 같다.

- believes(P, X): P가 X를 신뢰한다. 참가자 P가 X를 참(true)으로 여기고 행동한다.
- sees(P, X): P가 X를 볼 수 있다. 참가자 P가 X를 포함한 메시지(X가 암호화된 경우 포함)를 받았을 때, P가 X를 판독할 수 있음(필요하면 복호화 한 후)을 의미한다.
- said(P, X): P가 X를 말한 적 있다. 참가자 P가 메시지 X를 전송한 적이 있음을 의미한다. P가 언제 X를 말하였는지는 알 수 없으나 P가 X를 말 한 순간 P는 X를 신뢰했었음을 가정한다.
- controls(P, X): P가 X를 관할한다. 예를 들어 한 참가자가 키를 생성할 경우 그 참가자는 그 키에 대한 관할을 한다.
- fresh(X): X가 새로 생성되었다. 즉 X가 프로토콜의 현재 사용을 위해 사용됨을 의미한다. 새로 생성된 난수가 새로움을 나타내기 위해 종종 사용된다.
- shared_key(P, Q, K): 두 참가자 P와 Q가 K를 키(대칭키)로 공유한다. 이 경우 P와 Q 이외에는 K를 알 수 없다.
- public_key(P, K): P가 K를 공개키로 가진다. 대응되는 비밀키(K^{-1})는 P만이 알 수 있다.
- shared_secret(P, Q, X): P와 Q가 X를 공유된 비밀정보로 가지고 있다. 이 경우 P와 Q 이외에는 X를 알 수 없다.
- $\{X\}_K$: X가 키 K에 의하여 암호화되었다.
- $\langle X\rangle_Y$: X가 Y와 결합되었음을 나타낸다. Y는 X가 누구에 의하여 생성되었는지를 알리기 위해 주로 사용된다.

BAN logic은 다음과 같은 법칙을 가지고 있다.

$$\text{believes}(P, \text{shared\_key}(P, Q, K)), \text{sees}(P, \{X\}_K) \rightarrow \text{believes}(P, \text{said}(Q, X)) \quad (1)$$

즉, 만약 참가자 P가 K를 참가자 Q와의 대칭키로 신뢰하고, K에 의해 암호화된 메시지를 보면, P는 Q가 X를 말한 적이 있음을 믿는다.

$$\text{believes}(P, \text{public\_key}(Q, K)), \text{sees}(P, \{X\}_{K^{-1}}) \rightarrow \text{believes}(P, \text{said}(Q, X)) \quad (2)$$

즉 참가자 P가 참가자 Q의 공개키를 신뢰하고 Q의 비밀키에 의해 암호화된 메시지를 보면, P는 Q가 X를 말한 적이 있음을 믿는다.

이 밖에도 다음과 같은 법칙을 가지고 있다.

$$\text{believes}(P, \text{shared\_secret}(Q, Y)), \text{sees}(P, \langle X\rangle_Y) \rightarrow \text{believes}(P, \text{said}(Q, X)) \quad (3)$$

$$\text{believes}(P, \text{fresh}(X)), \text{believes}(P, \text{said}(Q, X)) \rightarrow \text{believes}(P, \text{believes}(Q, X)) \quad (4)$$

$$\text{believes}(P, \text{controls}(Q, X)), \text{believes}(P, \text{believes}(Q, X)) \rightarrow \text{believes}(P, X) \quad (5)$$

$$sees(P, (X, Y)) \rightarrow sees(P, X) \quad (6)$$

$$sees(P, \langle X \rangle_Y) \rightarrow sees(P, X) \quad (7)$$

$$believes(P, shared\_key(P, Q, K)), sees(P, \{X\}_K) \rightarrow sees(P, X) \quad (8)$$

$$believes(P, public\_key(P, K)), sees(P, \{X\}_K) \rightarrow sees(P, X) \quad (9)$$

$$believes(P, public\_key(Q, K)), sees(P, \{X\}_K^{-1}) \rightarrow sees(P, X) \quad (10)$$

$$believes(P, fresh(X)) \rightarrow believes(P, fresh(X, Y)) \quad (11)$$

Ⅳ. BAN logic의 확장

기존의 BAN logic은 인증서와 전자 서명, 그리고 해쉬(keyed hash(MAC) 포함) 등에 대한 분석 방법을 지원하지 않는다. 따라서 이들을 지원하기 위하여 BAN logic을 확장할 필요가 있다. 본 연구에서는 이들을 지원하기 위하여 다음과 같이 BAN logic을 확장하였다. 확장된 BAN logic은 인증서[10], 전자 서명[9], 그리고 해쉬 함수[11,12]의 기본적인 성질을 정형화된 표기법으로 나타낸다. 다음은 확장된 BAN logic의 표기법이다.

- CERT(Q): Q의 인증서를 나타낸다. 이 인증서는 신뢰할 수 있는 인증 기관의 인증서임을 가정한다.
- $signed(X)_K^{-1}$: X가 비밀키 K^{-1}에 의하여 전자 서명 되었음을 의미한다.
- HASH(X): X의 해쉬(SHA, MD5 등)값을 말한다.
- $MAC(X)_K$: X의 내용에서 키 K에 의하여 생성된 MAC을 의미한다.
- believed(P, X): P가 어느 순간 X를 신뢰했었다. 이는 said(P, X) 보나 약한 상태로 실제 X를 전송하지는 않았으나 X에 대하여 신뢰는 하였음을 의미한다. 예를 들어 Y에 대한 전자 서명을 하여 전송한 경우, 실제 Y의 값을 전송하지는 않았으나 Y를 신뢰하였음을 알 수 있다.

또한 확장된 BAN logic은 다음과 같은 법칙을 가지고 있다.

$$sees(P, CERT(Q)) \rightarrow believes(P, public\_key(Q, K)) \quad (12)$$

P가 Q의 인증서[10]를 받으면, A는 B의 공개키 (K)를 신뢰한다.

$$believes(P, public\_key(Q, K)), sees(P, signed(X)_K^{-1}) \rightarrow believes(P, believed(Q, X)) \quad (13)$$

$$believes(P, public\_key(Q, K)), sees(P, signed(HASH(X))_K^{-1}) \rightarrow believes(P, believed(Q, X)) \quad (14)$$

P가 Q의 공개키를 신뢰하고 Q로부터 전자 서명[9] 된 X를 받은 경우, P는 X가 Q로부터 신뢰되었음을 신뢰한다. Q가 해쉬한 후에 X에 전자 서명한 경우에도 같은 효과가 있다. 그러나 이 경우 P는 Q가 X를 신뢰함을 신뢰하지는 못한다. 그 이유는 X는 언제 Q에 의해 서명되었는지 알 수 없기 때문이다. 또한 이 경우 Q가 X의 원문을 전송하지 않을 수도 있다.

$$believes(P, shared\_key(P, Q, K)), sees(MAC(X)_K), sees(P, X) \rightarrow believes(P, believed(Q, X)) \quad (15)$$

$$believes(P, shared\_key(P, Q, K)), sees(MAC(X,Y)_K), sees(P, X), sees(P, Y) \rightarrow believes(P, believed(Q, (X, Y))) \quad (16)$$

P와 Q가 해쉬를 위한 키를 공유하고 P가 받은 X의 keyed hash(MAC)[11]의 값이 P가 획득한 평문(X)으로부터 생성한 keyed hash(MAC)의 값과 동일하면 P는 Q가 X를 신뢰함을 신뢰한다.

$sees(P, X), sees(P, Y) \rightarrow sees(P, (X, Y))$는 성립하지 않는다. 이는 X와 Y가 독립적인 경우가 있기 때문이다. keyed hash의 경우, X와 Y가 함께 해쉬되었고 X와 Y가 독립적으로 획득되었을 경우, X와 Y가 동시에 전송되었음을 말한다. 실제 X만 Q로부터 전송되었고 Y는 P 자신에 의해 생성되었을지라도 Q는 X와 Y에 대한 신뢰를 한 적이 있으므로 이 법칙은 본 논문의 범위 내에서는 안전하게 사용할 수 있다.

$$believes(P, believed(Q, HASH(X))), sees(P, X) \rightarrow believes(P, believed(Q, X)) \quad (17)$$

believes(P, believed(Q, HASH(X,Y))), sees(P, X),
sees(P, Y) → believes(P, believed(Q, (X, Y))) (18)

어떠한 값 X가 해쉬되었을 경우, 상대방이 HASH(X)에 대한 신뢰가 있음을 신뢰할 수 있으면, 상대방이 X를 신뢰함을 신뢰할 수 있다.

sees(P, shared_key(P, Q, K)),
believes(P, believed(Q, $MAC(X)_K$)), sees(P, X)
→ believes(P, believed(Q, X)) (19)

sees(P, shared_key(P, Q, K)),
believes(P, believed(Q, $MAC(X)_K$)), sees(P, X)
→ believes(P, shared_key(P, Q, K)) (20)

sees(P, shared_key(P, Q, K)),
believes(P, believed(Q, $MAC(X,Y)_K$)),
sees(P, X), sees(P, Y)
→ believes(P, believed(Q, (X, Y))) (21)

sees(P, shared_key(P, Q, K)),
believes(P, believed(Q, $MAC(X,Y)_K$)),
sees(P, X), sees(P, Y)
→ believes(P, shared_key(P, Q, K)) (22)

마찬가지, 특수한 경우로, 상대방과의 MAC(Message Authentication Code)을 위한 공유키를 가지고 있으나 신뢰하지 못하는 경우(이 경우 MAC 키가 상대방에 의해 생성된 것임을 신뢰할 수 없으므로 MAC을 이용하여 인증할 수 없다.)에도, 상대방이 생성한 X의 MAC값에 대한 신뢰가 있으면(예를 들어 전자 서명 된 경우) X에 대한 신뢰가 있다고 할 수 있다. 또한 이 경우에는 자신이 가지고 있는 공유키에 대한 신뢰를 할 수 있다.

이 밖에도,

believes(P, fresh(X)), believes(P, believed(Q, X))
→ believes(P, believes(Q, X)) (23)

said(X) → believed(X) (24)

도 성립한다.

V. IPsec의 분석

IKE의 5 가지 키 교환 방법을 BAN logic을 이용하여 분석하면 다음과 같다.

5.1 서명에 의한 인증

서명에 의한 인증을 사용하는 1 단계 IKE의 경우 분석에 필요한 메시지는 메시지 3, 4, 5, 6 이다. 이를 BAN logic을 위해 정형화하면 다음과 같다.

3. I → R: public_key(I, K_I), N_I
4. R → I: public_key(R, K_R), N_R
5. I → R: $\{ID_I, [CERT.] SIG\_I\}_K$
6. R → I: $\{ID_R, [CERT.] SIG\_R\}_K$

메시지 3과 4에서 K_I와 K_R은 Diffie-Hellman 키교환 방식[8]의 공개 정보이다. 이는 공개키 방식의 공개키에 대응되므로 자신의 공개키로 표현한다. 또한 메시지 5와 메시지 6의 암호화에 사용된 키 K는 I와 R이 공유한 키(shared_key(I, R, K))로 I는 K_I, K_I^{-1}와 K_R을 이용하여 생성할 수 있으며, R은 K_R, K_R^{-1}와 K_I를 이용하여 생성할 수 있다. K_I, K_I^{-1}와 K_R을 이용하여(K_R, K_R^{-1}와 K_I를 이용하여) K를 생성함은 Diffie-Hellman 키교환 방식에 의한 공유키 생성으로 참가자 이외에는 K를 생성할 수 없음을 알 수 있다. 메시지 5와 6의 SIG_I와 SIG_R은 각각 자신의 서명용 비밀키로 HASH_I와 HASH_R을 전자 서명한 것으로 그 서명용 비밀키에 대응되는 공개키는 CERT에 의해 확인 가능하다. HASH_I와 HASH_R은 각각 교환된 정보를 해쉬(MAC)키 K로 해쉬한 것이다. 실제 사용된 해쉬 키 K는 Diffie-Hellman 키교환 방식에 의해 생성된 공유키 K(메시지 5, 6의 암호화 키)와 상이하나 g^{xy}로부터 생성되므로 편의를 위해 동일한 것으로 취급할 수 있다

키 교환이 안전하게 되었음을 알기 위하여 다음을 증명할 수 있어야 한다.

believes(I, shared_key(I, R, K)),
believes(R, shared_key(I, R, K)),
believes(I, believes(R, shared_key(I, R, K))),
believes(R, believes(I, shared_key(I, R, K)))

즉, 키 IKE에 의해 생성된 공유키를 자신이 신뢰해야 하며, 또한 상대방이 생성된 공개키에 대한 신뢰를 가지고 있음을 신뢰할 수 있어야 한다.

증명을 하기 위하여 몇 가지 가정이 필요하다. 이

가정은 합리적인 가정이어야 한다.

believes(I, public_key(I, K_I)),
believes(R, public_key(R, K_R)),
believes(I, fresh(public_key(I, K_I))),
believes(R, fresh(public_key(R, K_R))),
believes(I, controls(R, public_key(R, K_R))),
believes(R, controls(I, public_key(I, K_I)))

위의 가정이 의미하는 바는 다음과 같다. I와 R 모두 자신의 공개키(Diffie-Hellman의 공개 정보)에 대해 신뢰한다. 또한 I와 R 모두 현재의 공개 정보는 현재 세션에 생성되었음을 신뢰한다. 또한 I는 R이 R의 공개정보를 생성함을 신뢰하며 R 또한 I가 I 자신의 공개 정보를 생성함을 신뢰한다.

이러한 가정을 기반으로 다음과 같이 분석될 수 있다.

메시지 3에서 Receiver(R)는 Initiator(I)의 공개키(Diffie-Hellman의 공개 정보)를 획득한다.

sees(R, (public_key(I, K_I), N_I))

이로부터 우리는 sees(R, public_key(I, K_I))를 얻을 수 있으며, 획득된 K_I와 자신의 K_R, K_R^{-1}를 결합하여 shared_key(I, R, K)인 K를 생성할 수 있다.

sees(R, shared_key(I, R, K))

여기에서 shared_key(I, R, K)는 Initiator와 Responder의 공유정보이고 실제 공유키는 이 공유정보에서 생성되나 우리는 여기에서 이를 공유키로 단순화 한다.

동일한 방법으로 메시지 4에서 Initiator는 동일한 shared_key(I, R, K)를 생성할 수 있다.

sees(I, shared_key(I, R, K))

메시지 3과 메시지 4 수행 후, sees(R, shared_key (I, R, K))와 sees(I, shared_key(I, R, K))는 얻을 수 있으나, believes(R, shared_key(I, R, K))와 believes(I, shared_key(I, R, K))는 얻지 못한다. 그러나 이 시점에 우리는 believes(R, shared_key(I, R, K))와 believes(I, shared_key (I, R, K))를 가정하여 암·복호화를 한다. 이 가정은 상대방의 인증을 위하여 사용될 수는 없다. (실제 키가 공유되었는 지 확인하지 못하였으므로 인증에 사용될 수는 없다. 그러나 단순한 암호화와 복호화를 위하여 사용하고 나중에 이들이 실제 공유키임을 증명할 수는 있다.)

메시지 5로부터 다음이 증명될 수 있다.

sees(R, $\{(ID_I, CERT(I), SIG\_I)\}_K$),
  believes(R, shared_key(I, R, K))
    → sees(R, (ID_I, CERT(I), SIG_I))

sees(R, (ID_I, CERT(I), SIG_I)) → sees(R, ID_I)

sees(R, (ID_I, CERT(I), SIG_I)) → sees(R, CERT(I))

sees(R, (ID_I, CERT(I), SIG_I)) → sees(R, SIG_I)

여기에서 인증서에 의한 효과에 의하여 다음을 얻을 수 있다.

sees(R, CERT(I)) → believes(R, public_key(I, K_{IS}))

여기에서 K_{IS}는 I(Initiator)의 서명을 위한 공개키이다.

SIG_I는 K_{IS}를 공개키로하여 그에 대응되는 K_{IS}^{-1}을 이용한 HASH_I에 대한 전자 서명이므로 법칙 (13)에 의하여 다음이 성립한다.

believes(R, public_key(I, K_{IS})), sees(R, SIG_I)
    → believes(R, believed(I, HASH_I))

즉 상대방의 공개키를 신뢰하고 있으며, 그 공개키에 대응되는 비밀키에 의하여 전자 서명을 획득하면 전자 서명된 내용을 상대방이 신뢰한 적이 있음을 알 수 있다.

HASH_I는 (K_I, K_R) 등의 공유 정보를 shared_key(I, R, K)인 K를 키로 이용하여 해쉬한 값이므로, HASH_I를 MAC(K_I, K_R)$_K$로 생각할 수 있다. 따라서 법칙 (21)에 의하여

sees(R, shared_key(I, R, K)),
  believes(R, believed(I, HASH_I)), sees(R, (K_I, K_R))
    → believes(R, believed(I, (K_I, K_R)))

를 얻을 수 있다. 즉 공유된 MAC 키를 가지고 있고 현재 데이터에 대한 MAC을 상대방이 작성하였

음을 신뢰하면, 상대방이 MAC을 작성할 당시 그 데이터를 신뢰하였음을 알 수 있다.

또한 법칙 (20)에 의하여

sees(R, shared_key(I, R, K)),
believes(R, believed(I, HASH_I)), sees(R, (K_I, K_R))
→ believes(R, shared_key(I, R, K))

를 얻을 수 있다. 즉 공유된 MAC 키를 가지고 있고 현재 데이터에 대한 MAC을 상대방이 작성하였음을 신뢰하면, 공유된 MAC 키를 신뢰할 수 있다. 법칙 (11)에 의하여 다음이 성립한다.

believes(R, fresh(K_R)) → believes(R, fresh(K_I, K_R))

즉 R이 K_R이 현 세션에 생성된 자신의 공개키임을 신뢰하므로 (K_I, K_R) 또한 현 세션에 사용되는 데이터임을 알 수 있다.

법칙 (25)에 의하여 다음을 얻을 수 있다.

believes(R, believed(I, (K_I, K_R))),
believes(R, fresh(K_I, K_R))
→ believes(R, believes(I, (K_I, K_R)))

즉, 상대방이 신뢰한 적이 있는 데이터가 현재 세션의 데이터이면 현재 세션에 상대방이 그 데이터를 신뢰하고 있음을 알 수 있다.

K_I, K_R 그리고 K_I^{-1}로부터 shared_key(I, R, K)가 생성되므로

believes(R, believes(I, (K_I, K_R)))
→ believes(R, believes(I, shared_key(I, R, K)))

또한 성립한다.

우리는 메시지 5에서

believes(R, believes(I, shared_key(I, R, K)))

와

believes(R, shared_key(I, R, K))

를 증명하였다. 동일한 방법으로 메시지 6로부터

believes(I, believes(R, shared key(I, R, K)))

와

believes(I, shared_key(I, R, K))

를 증명할 수 있다.

따라서 전자 서명에 의한 IKE 프로토콜은 인증서가 상대방의 공개키가 올바른 공개키임을 보장하면 안전한 키 교환 프로토콜이 됨을 알 수 있다.

5.2 공개키에 의한 인증

공개키에 의한 인증 방법을 사용하는 1 단계 IKE의 경우 분석에 필요한 메시지는 메시지 3, 4, 5, 6 이다. 이를 BAN logic을 위해 정형화하면 다음과 같다.

3. I → R: public_key(I, K_I), $\{ID_I\}_{K_R P}$, $\{N_I\}_{K_R P}$
4. R → I: public_key(R, K_R), $\{ID_I\}_{K_I P}$, $\{N_R\}_{K_I P}$
5. I → R: $\{HASH\_I\}_K$
6. R → I: $\{HASH\_R\}_K$

메시지 3과 4에서 K_I와 K_R은 Diffie-Hellman 키 교환 방식의 공개 정보이다. 이는 공개키 방식의 공개키에 대응되므로 자신의 공개키로 표현한다. 또한 메시지 5와 메시지 6의 암호화에 사용된 키 K는 I와 R이 공유한 키(shared_key(I, R, K))로 I는 K_I, K_I^{-1}와 K_R을 이용하여 생성할 수 있으며, R은 K_R, K_R^{-1}와 K_I를 이용하여 생성할 수 있다. K_I, K_I^{-1}와 K_R을 이용하여(K_R, K_R^{-1}와 K_I를 이용하여) K를 생성함은 Diffie-Hellman 키 교환 방식에 의한 공유키 생성으로 참가자 이외에는 K를 생성할 수 없음을 간주할 수 있다. 메시지 5와 6에서 HASH_I와 HASH_R에서 사용된 해쉬용 키는 암호화에 사용된 키 K와 상이하다. HASH_I와 HASH_R에 사용된 키는 메시지3과 4에서 교환된 N_I와 N_R에 의해 공유된 비밀정보에 의해 생성된다. 메시지 3과 4에서 N_I와 N_R는 각각 상대방의 공개키(K_{RP}, K_{IP})에 의해 암호화되어 전송되므로 I와 R이 상대방의 올바른 공개키(공개키가 노출되지 않은 경우)를 가지고 있다는 보장이 있는 경우, I와 R만이 공유된 비밀정보(해쉬용 키 K_H)를 가질 수 있다.

키 교환이 안전하게 되었음을 알기 위하여 다음을 증명할 수 있어야 한다.

believes(I, shared_key(I, R, K)),
believes(R, shared_key(I, R, K)),

believes(I, believes(R, shared_key(I, R, K))),
believes(R, believes(I, shared_key(I, R, K)))

증명을 하기 위하여 몇 가지 가정이 필요하다.

believes(I, public_key(I, K_I)),
believes(R, public_key(R, K_R)),
believes(I, fresh(public_key(I, K_I))),
believes(R, fresh(public_key(R, K_R))),
believes(I, controls(R, public_key(R, K_R))),
believes(R, controls(I, public_key(I, K_I)))

위의 가정이 의미하는 바는 다음과 같다. I와 R 모두 자신의 공개키(Diffie-Hellman의 공개 정보)에 대해 신뢰한다. 또한 I와 R 모두 현재의 공개 정보는 현재 세션에서 생성되었음을 신뢰한다. I는 R이 R의 공개정보를 생성함을 신뢰하며 R 또한 I가 I 자신의 공개 정보를 생성함을 신뢰한다. 위의 가정 외에도 공개키를 통하여 N_I와 N_R를 교환하기 위하여 다음과 같은 추가적인 가정이 필요하다.

believes(I, public_key(R, K_{RP})),
believes(R, public_key(I, K_{IP}))

이는 I는 R의 공개키 K_{RP}를 신뢰하며 R은 I의 공개키 K_{IP}를 신뢰함을 의미한다. 이 가정 하에 메시지 3과 4에서 송신자는 상대방의 공개키로 암호화된 정보를 생성할 수 있다. 위와같은 가정하에 다음과 같은 증명을 할 수 있다.

메시지 3에서 Receiver(R)는 Initiator(I)의 공개키(Diffie-Hellman의 공개 정보)를 획득한다.

sees(R, (public_key(I, K_I), N_I))

이로부터 우리는 sees(R, public_key(I, K_I))를 얻을 수 있으며, 획득된 K_I와 자신의 K_R, K_R^{-1}를 결합하여 shared_key(I, R, K)인 K를 생성할 수 있다.

sees(R, shared_key(I, R, K))

여기에서 shared_key(I, R, K)는 Initiator와 Responder의 공유정보이고 실제 공유키는 이 공유정보에서 생성되나 우리는 여기에서 이를 공유키로 여긴다.

메시지 3에서 R은 또한 $\{ID_I\}_{KRP}$, $\{N_I\}_{KRP}$를 획득한다. 획득된 $\{N_I\}_{KRP}$로부터 우리는 다음을 얻을 수 있다.

believes(R, public_key(R, K_{RP})), sees(R, $\{N_I\}_{KRP}$)
→ sees(R, N_I)

그리고 획득된 N_I와 자신이 생성한 N_R를 이용하여 해쉬를 위한 공유키 shared_key(I, R, K_H)를 생성한다.

sees(R, shared_key(I, R, K_H))

이와 같은 방법으로 메시지 4에서 Initiator는 동일한 shared_key(I, R, K)를 생성할 수 있다.

sees(I, shared_key(I, R, K))

sees(I, shared_key(I, R, K_H))

메시지 3과 메시지 4 수행 후, sees(R, shared_key(I, R, K))와 sees(I, shared_key(I, R, K))는 얻을 수 있으나, believes(R, shared_key(I, R, K))와 believes(I, shared_key(I, R, K))는 얻지 못한다. 그러나 우리는 believes(R, shared_key(I, R, K))와 believes(I, shared_key(I, R, K))를 가정하여 암호화와 복호화를 한다. 이 가정은 인증을 위하여 사용될 수는 없다. 우리는 believes(I, public_key(R, K_{PR}))과 believes(R, public_key(I, K_{PI}))라는 가정하에 (believes(R, shared_key(I, R, K_H))와 believes(I, shared_key(I, R, K_H))를 얻을 수 있으며, 실제 believes(I, shared_key(I, R, K_H))와 believes(R, shared_key(I, R, K_H))을 보장할 수는 없다. 하지만 I와 R이 각각 올바른 상대방의 공개키를 가지고 있다고 가정하면 I와 R외에는 공유된 K_H를 가질 수 없다. 그리고 실제 해쉬가 검증되는 시점에서 해쉬가 검증되면 I와 R이 K_H를 공유한 것이며, 해쉬가 검증되지 않으면 K_H를 공유하지 못하였거나 정보 교환이 올바로 되지 않은 것이다. 따라서 believes(I, shared_key(I, R, K_H))와 believes(R, shared_key(I, R, K_H))을 가정할 수 있다.

메시지 5로부터 다음이 증명될 수 있다.

sees(R, {HASH_I}$_K$), believes(R, shared_key(I, R, K))
→ sees(R, HASH_I))

HASH_I 는 (K_I, K_R) 등의 교환된 정보를 shared_key(I, R, K_H)인 K_H를 키로 이용하여 해쉬한 값이므로, HASH_I를 MAC(K_I, K_R)$_K$로 생각할 수 있다. 따라서 법칙 (21)에 의하여 다음을 얻을 수 있다.

believes(R, shared_key(I, R, K_H)),
sees(R, HASH_I), sees(R, (K_I, K_R))
→ believes(R, believed(I, (K_I, K_R)))

법칙 (11)에 의하여 다음이 성립한다.

believes(R, fresh(K_R)) → believes(R, fresh(K_I, K_R))

즉 R이 K_R이 현 세션에 생성된 자신의 공개키임을 신뢰하므로 (K_I, K_R) 또한 현 세션에 사용되는 데이터임을 알 수 있다.

법칙 (25)에 의하여 다음을 얻을 수 있다.

believes(R, believed(I, (K_I, K_R))),
believes(R, fresh(K_I, K_R))
→ believes(R, believes(I, (K_I, K_R)))

즉, 상대방이 신뢰한 적이 있는 데이터가 현재 세션의 데이터이면 현재 세션에 상대방이 그 데이터를 신뢰하고 있음을 알 수 있다.

K_I, K_R, 그리고 K_I^{-1}로부터 shared_key(I, R, K)가 생성되므로, 다음을 얻을 수 있다.

believes(R, believes(I, (K_I, K_R)))
→ believes(R, believes(I, shared_key(I, R, K)))

또한 상대방이 데이터를 신뢰하면, 그 데이터의 일부분을 신뢰함을 알 수 있다.

believes(R, believes(I, (K_I, K_R)))
→ believes(R, believes(I, K_I))

상대방이 K_I를 신뢰함을 알고 또한 상대방이 K_I를 생성하였음을 알 수 있으면, 법칙 (5)에 의하여 상대방이 K_I를 신뢰함을 알 수 있다. 따라서 다음을 얻을 수 있다.

believes(R, believes(I, K_I)), believes(R, controls(I, K_I))
→ believes(R, K_I)

K_I, K_R, 그리고 K_R^{-1}로부터 shared_key(I, R, K)가 생성되므로

believes(R, K_I), believes(R, K_R)
→ believes(R, shared_key(I, R, K))

또한 성립한다.

우리는 메시지 5에서

believes(R, believes(I, shared_key(I, R, K)))

believes(R, shared_key(I, R, K))

를 증명하였다. 동일한 방법으로 메시지 6로부터

believes(I, believes(R, shared_key(I, R, K)))

believes(I, shared_key(I, R, K))

를 증명할 수 있다.

따라서 공개키에 의한 IKE 프로토콜은 Initiator와 Responder가 각각 상대방의 올바른 공개키를 가지고 있다는 가정 하에 키 교환을 안전하게 할 수 있다. 그러나 일반적인 공개키 시스템에서 상대방의 올바른 공개키를 항상 가지고 있음을 보장하기에는 어려움이 있다. 이를 해결하기 위해서는 인증서를 이용함이 안전하다고 할 수 있다. 즉 메시지 1과 2에서 필요한 경우 상대방에게 인증서를 보낼 수 있게 함으로써 키 교환의 안전도를 높일 수 있다.

5.3 변경된 공개키에 의한 인증

변경된 공개키에 의한 인증 방법을 사용하는 1단계 IKE의 경우 분석에 필요한 메시지는 메시지 3, 4, 5, 6 이다. 이를 BAN logic을 위해 정형화하면 다음과 같다.

3. I → R: {N_I}$_{KRP}$, {public_key(I,K_I)}$_{Ke\_i}$, {ID_I}$_{Ke\_i}$
4. R → I: {N_R}$_{KIP}$, {public_key(R,K_R)}$_{Ke\_r}$, {ID_R}$_{Ke\_r}$
5. I → R: {HASH_I}$_K$
6. R → I: {HASH_R}$_K$

메시지 3과 4에서 N_I와 N_R는 각각 상대방의 공개

키(K_{RP}, K_{IP})에 의해 암호화되어 전송된다. 이 경우 I와 R이 상대방의 올바른 공개키(공개키가 노출되지 않은 경우)를 가지고 있다는 보장이 필요하다. K_I와 K_R은 Diffie-Hellman 키 교환 방식의 공개 정보이다. Diffie-Hellman의 공개정보인 public_key(I, K_I)와 public_key(R, K_R)은 대칭키 암호화 방식에 의해 암호화되어 전송된다. 실제 Diffie-Hellman 키 교환 방식은 안전하다고 간주하나 이 경우 추가적인 안전성을 주기 위해 암호화하였다. 암호화에 사용된 키는 각각 N_I와 N_R에 의해 생성된다. 또한 메시지 5와 메시지 6의 암호화에 사용된 키 K는 I와 R이 공유한 키(shared_key(I, R, K))로 I는 K_I, K_I^{-1}와 K_R을 이용하여 생성할 수 있으며, R은 K_R, K_R^{-1}와 K_I를 이용하여 생성할 수 있다. 메시지 5와 6에서 HASH_I와 HASH_R에서 사용된 해쉬용 키는 암호화에 사용된 키 K와 상이하다. HASH_I와 HASH_R에 사용된 키는 메시지3과 4에서 교환된 N_I와 N_R에 의해 공유된 비밀정보에 의해 생성된다. 메시지 3과 4에서 N_I와 N_R는 각각 상대방의 공개키(K_{RP}, K_{IP})에 의해 암호화되어 전송되므로 I와 R이 상대방의 올바른 공개키(공개키가 노출되지 않은 경우)를 가지고 있다는 보장이 있는 경우, I와 R만이 공유된 비밀정보(해쉬용 키 K_H)를 가질 수 있다.

이 경우의 분석은 공개키에 의한 인증의 경우와 동일하다.

5.4 공유키에 의한 인증

공유키에 의한 인증 방법을 사용하는 1 단계 IKE의 경우 분석에 필요한 메시지는 메시지 3, 4, 5, 6이다. 이를 BAN logic을 위해 정형화하면 다음과 같다.

3. I → R: public_key(I, K_I), N_I
4. R → I: public_key(R, K_R), N_R
5. I → R: {ID_I, HASH_I}$_K$
6. R → I: {ID_R, HASH_R}$_K$

메시지 3과 4에서 K_I와 K_R은 Diffie-Hellman 키 교환 방식의 공개 정보이다. 이는 공개키 방식의 공개키에 대응되므로 자신의 공개키로 표현한다. 또한 메시지 5와 메시지 6의 암호화에 사용된 키 K는 I와 R이 공유한 키(shared_key(I, R, K))로 I는 K_I, K_I^{-1}와 K_R을 이용하여 생성할 수 있으며, R은 K_R, K_R^{-1}와 K_I를 이용하여 생성할 수 있다. K_I, K_I^{-1}와 K_R을 이용하여(K_R, K_R^{-1}와 K_I를 이용하여) K를 생성함은 Diffie-Hellman 키 교환 방식에 의한 공유키 생성으로 참가자 이외에는 K를 생성할 수 없음을 간주할 수 있다. 메시지 5와 6에서 HASH_I와 HASH_R에서 사용된 해쉬용 키는 암호화에 사용된 키 K와 상이하다. HASH_I와 HASH_R에 사용된 키는 자신들이 공유하고 있는 공유키 K_S이다 (실제 공유키로 Ni, Nr을 해쉬한 값이나 이를 공유키로 취급할 수 있다).

키 교환이 안전하게 되었음을 알기 위하여 다음을 증명할 수 있어야 한다.

believes(I, shared_key(I, R, K)),
believes(R, shared_key(I, R, K)),
believes(I, believes(R, shared_key(I, R, K))),
believes(R, believes(I, shared_key(I, R, K)))

증명을 하기 위하여 몇 가지 가정이 필요하다.

believes(I, public_key(I, K_I)),
believes(R, public_key(R, K_R)),
believes(I, fresh(public_key(I, K_I))),
believes(R, fresh(public_key(R, K_R))),
believes(I, controls(R, public_key(R, K_R))),
believes(R, controls(I, public_key(I, K_I)))

위의 가정이 의미하는 바는 다음과 같다. I와 R 모두 자신의 공개키(Diffie-Hellman의 공개 정보)에 대해 신뢰한다. 또한 I와 R 모두 현재의 공개키는 현재 세션에 생성되었음을 신뢰한다. I는 R이 R의 공개정보를 생성함을 신뢰하며 R 또한 I가 I 자신의 공개 정보를 생성함을 신뢰한다. 이 외에도 공유키를 신뢰한다는 다음의 가정이 필요하다.

believes(I, shared_key(I, R, K_S)),
believes(I, shared_key(I, R, K_S))

메시지 3에서 Receiver(R)는 Initiator의 공개키(Diffie-Hellman의 공개 정보)를 획득한다.

sees(R, (public_key(I, K_I), N_I))

이로부터 우리는 sees(R, public_key(I, K_I))를

얻을 수 있으며, 획득된 K_I와 자신의 K_R, K_R^{-1}를 결합하여 shared_key(I, R, K)인 K를 생성할 수 있다.

sees(R, shared_key(I, R, K))

여기에서 shared_key(I, R, K)는 Initiator와 Responder의 공유정보이고 실제 공유키는 이 공유정보에서 생성되나 우리는 여기에서 이를 공유키로 여긴다. 이와 같은 방법으로 메시지 4에서 Initiator는 동일한 shared_key(I, R, K)를 생성할 수 있다.

sees(I, shared_key(I, R, K))

메시지 3과 메시지 4 수행 후, sees(R, shared_key(I, R, K))와 sees(I, shared_key(I, R, K))는 얻을 수 있으나, believes(R, shared_key(I, R, K))와 believes(I, shared_key(I, R, K))는 얻지 못한다. 그러나 이 시점에 우리는 believes(R, shared_key(I, R, K))와 believes(I, shared_key(I, R, K))를 가정하여 암호화와 복호화를 한다.
메시지 5로부터 다음이 증명될 수 있다.

sees(R, $\{ID_I, HASH\_I\}_K$),
believes(R, shared_key(I, R, K))
→ sees(R, HASH_I)

HASH_I는 (K_I, K_R) 등 교환된 정보를 shared_key(I, R, Ks)인 K_S를 키로 이용하여 해쉬한 값이므로

believes(R, shared_key(I, R, K_S)),
sees(R, HASH_I), sees(R, (K_I, K_R))
→ believes(R, believed(I, (K_I, K_R)))

believes(R, fresh(K_R)) → believes(R, fresh(K_I, K_R))

believes(R, believed(I, (K_I, K_R))),
believes(R, fresh(K_I, K_R))
→ believes(R, believes(I, (K_I, K_R)))

가 성립함을 알 수 있다. (5.1절 참조)
K_I, K_R, 그리고 K_I^{-1}로부터 shared_key(I, R, K)가 생성되므로 다음이 성립한다.

believes(R, believes(I, (K_I, K_R)))
→ believes(R, believes(I, shared_key(I, R, K)))

또한

believes(R, believes(I, (K_I, K_R)))
→ believes(R, believes(I, K_I))

believes(R, believes(I, K_I)), believes(R, controls(I, K_I))
→ believes(R, K_I)

또한 성립한다. (5.2절 참조)
K_I, K_R, 그리고 K_R^{-1}로부터 shared_key(I, R, K)가 생성되므로

believes(R, K_I), believes(R, K_R)
→ believes(R, shared_key(I, R, K))

또한 성립한다.
우리는 메시지 5에서

believes(R, believes(I, shared_key(I, R, K)))

believes(R, shared_key(I, R, K))

를 증명하였다. 동일한 방법으로 메시지 6로부터

believes(I, believes(R, shared_key(I, R, K)))

believes(I, shared_key(I, R, K))

를 증명할 수 있다.
따라서 공개키에 의한 IKE 프로토콜은 Initiator와 Responder가 공유키를 안전하게 유지하면 안전하게 키 교환을 한다고 할 수 있다.

5.5 2 단계 IKE

2 단계 IKE의 경우 BAN logic으로 분석하기 위하여 정형화하면 다음과 같다.

1. I → R: $\{(HASH\_1, SA, N_I)\}_K$
2. R → I: $\{(HASH\_2, SA, N_R)\}_K$
3. I → R: $\{HASH\_3\}_K$

2 단계의 모든 메시지는 1 단계에서 생성한 공유키

K에 의하여 암호화 되어있다. HASH_1은 N_I의 값을 공유키 K로 해쉬한 값이며, HASH_2는 N_I와 N_R의 값을 공유키 K로 해쉬한 값이며, HASH_3도 N_I와 N_R의 값을 공유키 K로 해쉬한 값이다 (실제 조금의 변화가 있으나 그렇게 단순화 할 수 있다). 2 단계에서는 1 단계에 성공적으로 K를 공유 하였음을 가정한다. Ni와 Nr이 안전하게 교환되었음을 보이기 위해 우리는 다음을 증명하여야 한다.

believes(I, N_I),
believes(I, N_R),
believes(I, believes(R, N_I)),
believes(I, believes(R, N_R)),
believes(R, N_R),
believes(R, N_I),
believes(R, believes(I, N_R)),
believes(R, believes(I, N_I))

우리는 다음의 가정을 할 수가 있다.

believes(I, N_I),
believes(R, N_R),
believes(I, fresh(N_I)),
believes(R, fresh(N_R)),
believes(I, shared_key(I, R, K)),
believes(R, shared_key(I, R, K)),
believes(I, controls(R, N_R)),
believes(R, controls(I, N_I))

위 가정에 의하여 다음과 같이 증명될 수 있다.

메시지 1에서 R은 $\{(HASH_1, SA, N_I)\}_K$를 획득한다.

believes(R, shared_key(I, R, K)),
sees(R, $\{(HASH\_1, SA, N_I)\}_K$) → sees(R, N_I)
(believes(R, said(I, N_I))도 획득)

R은 N_I를 볼 수 있다. 이 시점에서 R은 N_I를 신뢰하는 것으로 가정한다 (후에 증명하여야 한다). 메시지 1에서 believes(R, N_I)를 획득한다.

메시지 2에서 I는 $\{(HASH_2, SA, N_R)\}_K$를 획득한다.

believes(I, shared_key(I, R, K)),
sees(I, $\{(HASH\_2, SA, N_R)\}_K$) → sees(I, HASH_2)

believes(I, shared_key(I, R, K)),
sees(I, $\{(HASH\_2, SA, N_R)\}_K$) → sees(I, N_R)

believes(I, shared_key(I, R, K)),
sees(I, HASH_2), sees(I, N_R), sees(I, N_I)
→ believes(I, believed(R, (N_I, N_R)))
believes(I, fresh(N_I)) → believes(I, fresh(N_I, N_R))

believes(I, believed(R, (N_I, N_R)), believes(I, fresh(N_I))
→ believes(I, believes(R, (N_I, N_R)))

believes(I, believes(R, N_I, N_R))
→ believes(I, believes(R, N_I))

believes(I, believes(R, N_I, N_R))
→ believes(I, believes(R, N_R))

believes(I, believes(R, N_R)),
believes(I, controls(R, N_R)) → believes(I, N_R)

메시지 3에서 R은 $\{HASH_3\}_K$를 획득하여 메시지 2에서와 같은 방법으로

believes(R, believes(I, N_I))

believes(R, believes(I, N_R))

believes(R, N_I)

를 얻을 수 있다. 따라서 1 단계 IKE가 안전하다고 가정하면, 2 단계 IKE는 안전하게 키 교환을 수행한다.

Ⅵ. 결 론

본 연구에서는 IPsec의 IKE 프로토콜을 BAN logic을 이용하여 정형적으로 분석하였다. IPsec을 지원하기 위하여 BAN logic은 인증서의 경우와 전자서명, 그리고 keyed-hash(MAC)의 경우를 지원하기 위하여 확장되었다. IPsec의 분석 결과 전자서명을 통하여 인증 하는 IKE 1 단계는 안전하게 키 교환을 한다. 공유키를 사용하여 인증 하는 경우에도 공유한 키가 침입자에게 노출되지 않는 경우 안전하게 키 교환을 할 수 있다. 공개키를 사용하여 인증을 수행하는 경우와 변경된 공개키를 사용하는 경우 상대방의 공개키에 대한 신뢰가 있는

경우에 안전하다. 일반적으로 상대방의 공개키에 대한 신뢰를 항상 할 수는 없으므로 공개키를 사용하여 인증 하는 경우와 변경된 공개키를 사용하여 인증 하는 경우 인증서를 사용함이 안전하다고 할 수 있다.

BAN logic을 이용한 분석은 참가자들 사이의 키 분배를 효율적으로 분석할 수 있으나, 기밀성, Denial of Service 공격 방지 등 그 외의 특성은 분석할 수 없다. 암호 알고리즘, 해쉬, 전자 서명 등의 안전성 또한 BAN logic의 분석 범위를 벗어난다. BAN logic으로 분석될 수 없는 부분은 그들 특성을 지원하는 분석 방법에 의하여 추가적인 분석이 필요하다. BAN logic은 최근의 다양한 프로토콜을 지원하지 못한다. 본 연구에서는 BAN logic을 필요에 따라 확장하여 사용하였다. IPsec의 IKE 프로토콜 뿐만 아니라 다양한 최근의 프로토콜을 지원하도록 BAN logic의 확장이 필요하다.

참 고 문 헌

[1] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," *IETF RFC 2401*, November 1998.

[2] D. Harkins and D. Carrel, "The Internet Key Exchange," *IETF RFC 2409*, November 1998.

[3] Edmund M. Clarke and Jeannette M. Wing, "Formal Methods: State of the Art and Future Directions," *ACM Computing Survey*, Vol. 28, No. 4, Dec. 1996.

[4] Michael Burrows, Martin Abadi and Roger Needham, "A Logic of Authentication," *ACM Transactions on Computer Systems*, 8(1):18-36, February 1990.

[5] G. Bella and E. Riccobene, "Formal Analysis of the Kerberos Authentication System," *Journal of Universal Computer Science*, Vol 3, No 12, pp. 1337~1381, December 1997.

[6] J. Mitchell, V. Shmatikov and U. Stern, "Finite-State Analysis of SSL 3.0 and Related Protocols," *DIMACS Workshop on Design and Formal Verification of Security Protocols*, September 1997.

[7] C. Meadows, "A Formal Framework and Evaluation Method for Network Denial of Service," *Proceedings of the 12th IEEE Computer Security Foundation Workshop*, June 1999.

[8] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, Vol. 22, No. 6, June 1977.

[9] NIST, "Digital Signature Standard," *FIPS 186*, National Institute of Standard and Technology, May 1994.

[10] R. Housley, W. Ford, W. Polk and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," *IETF RFC 2459*, January 1999.

[11] H. Krawczyk, M. Bellare and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," *IETF RFC 2104*, February 1997.

[12] B. Preneel, "MACs and Hash Functions: State of the Art," *Information and Security Technical Report*, Vol. 2, No. 2, pp. 33~43, 1997.

〈著者紹介〉

주 한 규 (Hankyu Joo) 정회원

1988년 2월 : 한림대학교 전자계산학과 졸업
1994년 12월 : Arizona State University 컴퓨터공학과 석사
1998년 12월 : Arizona State University 컴퓨터공학과 박사
1999년 1월~2000년 2월 : 한국전자통신연구원 선임연구원
2000년 3월~현재 : 한림대학교 정보통신공학부 전임강사
〈관심분야〉 컴퓨터공학, 소프트웨어공학, 정보보호