이동통신에서의 키 교환 프로토콜 비교 분석

김건우, 류희수

한국전자통신연구원

Comparison with key exchange protocols in mobile communication

Keonwoo Kim, Heuisu Ryu

Electronics and Telecommunications Research Institute

요 약

본 논문에서는 이동통신용 인증 및 키 교환 프로토콜을 분석하였다. Forward secrecy나 signcryption 성질을 만족하기 위해서 프로토콜의 변형이 필요하였고 이 때문에 원래의 보안요구사항이 훼손되지도 않았다. 또한, 클라이언트의 연산능력을 고려한 최근에 제안된 프로토콜에 관해서도 연구하여 연산량과 통신 오버헤드 관점에서 기존의 프로토콜보다 우수함이 분석되었다.

I. 서론

이동환경에서 통신하는 상대개체을 인증하고 메시지를 안전하게 교환하기 위해서는 인증 및 키 교환 프로토콜이 필요하다. 그래서, 클라이언트와 서버는 서로의 신원을 확인하고 두 개체만 소유하는 공통키인 세션키를 분배하여 이후에 교환되는 메시지에 대한 암호화에 사용한다. 이동통신에서의 보안 프로토콜은 오래전부터 많은 연구가 되어왔고, 지금은 공개키 기반의 키 교환 프로토콜에 관해서 많은 설계가 이루어지고 있다. 하지만, 무선이라는 환경의 특징과 클라이언트 계산능력이나 메모리등의 제한으로 유선 네트워크에서와 같은 방법으로 프로토콜을 설계할 수는 없다.

그래서, 본 논문에서는 이동통신 환경에 적합한 공개키 기반의 인증 및 키 교환 프로토콜에 관해서 분석하여 비교한다. 다양한 공개키 기반 프로토콜이 있지만 G. Horn 등에 의하면 ASPeCT 프로토콜이 거의 모든 보안 요구사항을 만족하는 것으로 분석하였다.[1] 하지만, ASPeCT 프로토콜은 Forward Secrecy 성질을 만족하지 않아 본 논문에서는 Forward Secrecy를 충족시키는 몇가지 경우의 변형된 ASPeCT 프로토콜을 예로 들었다. 그리고, 또다른 프로토콜로서 ASIACRYPT 2001 에서 D. S. Wong 등에 의해서 제안된 키 교환 프로토콜을 분석한다.[4] 그래서, Forward Secrecy를 제공하는 ASPeCT 프로토콜과 D. S. Wong의 프로토콜을 연산량이나 통신량 등의 관점에서 비교하여 어느것이 좀더 효율적인 프로토콜인가를 제시한다.

II. Forward Secrecy를 만족하는 이동통신용 키 교환 프로토콜

이 장에서는 기존의 안전하고 효율적인 것으로 검증된 이동통신용 키 교환 프로토콜인 ASPeCT를 Forward Secrecy 성질을 만족하기 위해 변형한 형태의 프로토콜과 ASIACRYPT 2001에서 제안된 저전력의 클라이언트에 적합한 키 교환 프로토콜에 관해서 분석한다.

1. ASPeCT

ASPeCT 프로토콜은 European Commission ACTS project ASPeCT에 의해서 개발되었고, UMTS(Universal Mobile Telecommunication System)에서 사용자와 서버 사이의 공개키 기반 인증 및 키 설정 프로토콜 중의 하나이다. 그림 1에 ASPeCT 프로토콜을 나타내었고, 사용되는 표기는 다음과 같다. 이 논문에서는 과금과 지불 정보에 관한 내용은 언급하지 않는다.

- r_A, r_B : 클라이언트와 서버의 랜덤 nonce
- $Cert_A$, $Cert_B$: 클라이언트와 서버의 공개키 인증서
- g^b : 서버의 공개키
- K : 클라이언트와 서버 사이에 설정된 세션키
- $E_K\{m\}$: 세션키 K를 사용하여 데이터 m에 대한 대칭키 암호화

- $K_A^{-1}\{m\}$: 클아이언트의 개인키를 사용한 데이터 m에 대한 서명
- $h1$, $h2$, $h3$: 해쉬함수

1. A⇕ B : g^{r_A}, TTP_A
2. A⇓ B : r_B, h2(K, r_B, B), B_{Cert}
3. A⇑ B : $\{\{h3(g^{r_A}, g^b, r_B, B)\}_{K_A^{-1}}, A_{Cert}\}_K$

그림 1. ASPeCT 프로토콜

사용자는 랜덤수 r_A 를 생성해서 g^{r_A} 를 계산하여 서버에게 보낸다. 이 메시지를 수신한 서버는 랜덤수 r_B를 생성해서 Diffie-Helman variant를 이용해 세션키 $K = h1(r_B, g^{br_A})$ 를 계산하고 그림 1의 두 번째 flow와 같은 메시지를 사용자에게 전송한다. 그리면 사용자는 세션키 $K = h1(r_B, g^{br_A})$ 를 계산하고 두 번째 flow에서 받은 해쉬값과 자신이 계산한 $h2(K, r_B, ID_B)$ 해쉬값을 비교한다. 이 값이 일치하면 사용자는 세 번째 flow에서와 같은 서명값을 생성해서 서버에게 전송한다. ASPeCT는 공개키 기반의 인증키 설정 프로토콜이 요구하는 보안 요구사항인 사용자와 서버 사이의 상호 개체 인증, 사용자와 서버 사이의 인증된 공개키 교환, 사용자와 서버 사이의 세션키 일치, 암시적인 키 인증 및 키 확인, key freshness 특성 등을 만족한다.

2. Forward Secrecy를 만족하는 ASPeCT 프로토콜 (1)

원래의 ASPeCT 프로토콜은 Forward Secrecy 특성을 만족하지 않는다. Forward Secrecy란 long-term 비밀키가 노출되더라도 이전의 세션키는 복구되지 않아야 하는 성질을 의미한다. 이 문제를 해결하기 위해서는 세션키의 생성 방법이 수정되어야 하는데, D. Park 등이 PKC2000에서 제시한 방법을 이용함으로서 세션키 생성방법을 바꿀 수 있다.[5] 일반적으로 Forward Secrecy를 만족하기 위해서는 원래의 프로토콜에서 약간의 계산량을 더 요구한다.

ASPeCT 프로토콜에서는 세션키로 $K = h1(r_B, g^{br_A})$를 사용하는데, 서버의 long-term 비밀키 b가 노출되면 이전의 세션키는 셔션중에 사용된 g^{r_A}, r_B를 이용해서 구해질 수 있다. 그래서, r_B 대신에 g^{r_B} 를 이용하여 세션키 $K = h1(g^{r_A r_B})$ 로 설정함으로서 Forward Secrecy를 보장할 수 있다. 수정된 프로토콜은 그림 2와 같다. g^{br_A}는 클라이언트에게 서버의 인증을 제공하기 위해서 flow 2의 해쉬함수 $h2(K, g^{r_B}, g^{br_A}, B)$ 안에 포함된다.

1. A⇕ B : g^{r_A}, TTP_A
2. A⇓ B : g^{r_B}, h2(K, g^{r_B}, g^{br_A}, B), B_{Cert}
3. A⇑ B : $\{\{h3(g^{r_A}, g^b, g^{r_B}, B)\}_{K_A^{-1}}, A_{Cert}\}_K$

그림 2. Forward Secrecy를 제공하는 ASPeCT 프로토콜 (1)

3. Forward Secrecy를 만족하는 ASPeCT 프로토콜 (2)

공개키 기반 인증 및 키 공유 프로토콜에서는 기밀성과 인증을 위해 암호화 기법과 디지털 서명을 사용하는데, 기밀성과 인증을 동시에 제공하는 것이 서명 후 암호화를 적용하는 것보다 계산량이나 통신량 측면에서 좀더 효율적이다. 그래서, Y. Zheng은 인증과 기밀성을 동시에 제공하는 signcryption 기법을 키 전송 프로토콜과 키 교환 프로토콜에 적용하였다.[2] 하지만, signcryption 은 Forward Secrecy 성질을 만족하지 못한다. 본 논문에서는 Forward Secrecy를 만족하기 위해 변형된 signcryption 기법을 ASPeCT 프로토콜에 적용한 예를 든다.[3]

1. A⇕ B : $T = g^{r_A} \bmod p$

   B : $K = h1(T^{r_B} \bmod p)$
   $R = h2(T^{x_B} \bmod p)$

2. A⇓ B : $g^{r_B} \bmod p$, h3(K, g^{r_B}, B), R, B_{Cert}

   A : $R = h2(P_B^{r_A} \bmod p)$
   $K = h1(g^{r_B})^{r_A} \bmod p)$
   $m = T \| P_B \| g^{r_B} \| B$
   $r = KH_K(m)$
   $s = r_A / (r+x_A) \bmod q$
   $c = E_K(m, r, s, Cert_A)$

3. A⇕ B : c

   B : $D_K\{c\}$
   $T ?= (P_A g^r)^s \bmod p$
   $KH_K(m) ?= r$

그림 3. signcryption을 이용한 Forward Secrecy를 제공하는 ASPeCT 프로토콜(2)

먼저 클라이언트는 서버에게 $T = g^{r_A}$를 전송한다. 이 메시지를 받은 서버는 랜덤수 r_B 를

만들어서 $g^{r_B} \bmod p$ 를 계산한 다음, 세션키 $K = h1(T^{r_B} \bmod p)$ 와 인증을 위한 값으로 $R = h2(T^{x_B} \bmod p)$ 를 계산한다. 그리고, 두 번째 flow 에서 서버는 자신의 인증서 $Cert_B$와 $g^{r_B} \bmod p$, 인증 응답값 R, 그리고, K, g^{r_B}, B를 해쉬한 값을 클라이언트에게 보낸다. 이 값을 수신한 클라이언트는 서버의 신원을 확인한 다음 R를 계산해서 수신한 R과 비교하며, 서버의 공개키 P_B를 검증한다. 그리하여, 세션키 $K = h1((g^{r_B})^{r_A} \bmod p)$를 계산하고 수신된 $h3(K, g^{r_B}, B)$와 비교한다. 그리고, 서명값 r, s를 구해서 세션키 K를 사용하여 서명값을 암호화하여 전송한다. 세 번째 flow에서 서버가 메시지를 받자마자 c를 복호해서 T를 비교하고 $KH_K(m)$를 계산하여 r과 비교한다.

이 프로토콜에서는 r_A, r_B를 사용한 Diffie-Hellman 키교환 방식을 사용하여 세션키 $K = h1(g^{r_A r_B} \bmod p)$를 설정한다. 그래서, 클라이언트와 서버의 비밀키인 x_A, x_B 가 노출되더라도 양측에서 설정된 세션키는 보호된다.

4. Wong 프로토콜

D.S. Wong과 A. H. Chan은 ASIACRYPT 2001에서 저전력을 가진 클라이언트와 상대적으로 큰 계산능력을 가진 서버 사이의 인증 및 키 교환 프로토콜을 제안하였다. 이 프로토콜은 Bellare-Rogaway 모델의 변형[6]에서 안전한 것으로 증명되었고, 다른 공개키 기반 키 교환 프로토콜과 동일한 scalability를 가지는 반면 통신량이나 효율성 면에서는 우수한 특성을 보이는 것으로 분석된다.

이 스킴에서는, 클라이언트와 서버는 각자의 공개키-비밀키쌍을 소유한다. 서버에 대해서는 비밀키와 공개키로 SK_B, PK_B 를 사용하고, 클라이언트에 대해서는 a, g^a 를 사용한다. 이 프로토콜에서 클라이언트는 단 한번의 공개키 암호화 과정만을 필요로한다. 그리고, TTP로부터 획득하는 Certificate는 다음과 같다.

$$Cert_A = \langle ID_A, g^a, Sig_{TTP}(ID_A, g^a) \rangle$$

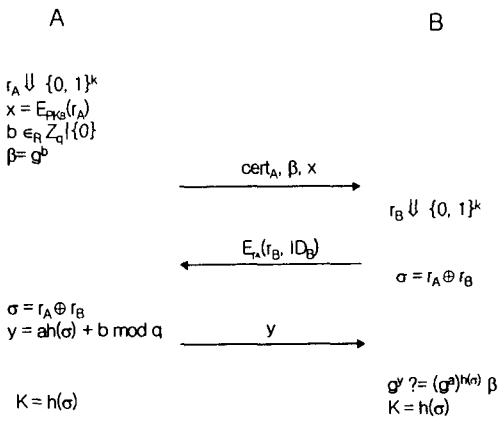

그림 1. Wong 프로토콜

클라이언트는 $r_A \leftarrow \{0,1\}^k$, $b \in_R Z_q \setminus \{0\}$ 를 선택해서 $x = E_{PK_B}(r_A)$와 $\beta = g^b$를 계산하고, 서버에게 $Cert_A$, β, x를 보낸다. 서버는 A의 인증서를 체크하고 $1 < \beta < p$ 와 $\beta^q \equiv 1 \bmod p$ 인지를 검증하다. 이 확인절차는 공개키의 유효성 여부를 확인하는데 매우 중요한 것으로서, small subgroup 공격이나 identity element 공격을 방지하기 위해서는 필수적인 절차이다. 만약 이 검증이 실패한다면 서버는 프로토콜을 종료하고, 검증이 성공하면 x를 복호해서 r_A를 얻는다. 그리고, 서버는 $r_B \leftarrow \{0,1\}^k$를 선택하고 $E_{r_A}(r_B, ID_B)$를 계산하여 그것을 클라이언트에게 보내고, $\sigma = r_A \oplus r_B$ 를 계산한다. flow 2의 메시지를 받은 클라이언트는 메시지를 r_A 을 사용하여 복호해서 서버 identity를 확인하고 $\sigma = r_A \oplus r_B$와 $y = ah(\sigma) + b \bmod q$ 를 계산한다. 그리고나서 서버에게 y를 보낸다. 또한, 클라이언트는 세션키로서 $K = h(\sigma)$ 를 계산한다. h는 σ로부터 세션키 K를 유도하는 키 유도함수이다. 한편 y를 수신한 서버는 $g^y \equiv (g^a)^{h(\sigma)} \beta$ 를 검증해서 검증이 성공한다면 세션키로서 $K = h(\sigma)$를 계산한다.

III. 프로토콜 비교

이 장에서는 2장에서 분석한 프로토콜 중 어느 것이 더 효율적인가를 알아보기 위해 계산량과 통신 오버헤드를 비교하였다. 해쉬함수 계산이나 대칭키 연산은 공개키 지수연산에 비해 아주 적은 계산만 요구하므로 공개키 지수연산의 회수만 언급한다. 사용된 지수 연산회수에 대하여 표 1에 정리하였다.

표 1: 연산량 비교.

	FS_ ASPeCT(1)	FS_ ASPeCT(2)	Wong
A	3 + (1)	2 + (1)	1 + (1)
B	3 + (1)	4 + (1)	4

( ) : precomputation이 가능하다는 의미

지수 연산량 비교만 해보면 클라이언트에서는 Wong, FS_ASPeCT(2), FS_ASPeCT(1) 순으로, 서버에서는 FS_ASPeCT(1), Wong, FS_ASPeCT (2) 순으로 계산량이 적은 것으로 분석된다. Forward Secrecy를 추가한 ASPeCT 프로토콜에서는 signcryption 기법을 사용한 프로토콜이 클라이언트에서 좀더 효율적이었다. 그리고, 세가지 프로토콜 모두 3-pass로 이루어지고, 이동통신용 인증 및 키 교환 프로토콜에서의 요구되는 필요사항을 모두 만족한다. 하지만, 각각의 패스에서 전송되는 통신량은 Wong 프로토콜이 가장 적다.

IV. 결론

인증 및 키 교환은 이동통신에서 상대방을 확인하고 안전한 통신을 하기 위한 필수적인 절차이다. 지금까지 많은 인증 및 키 교환 프로토콜이 제안되었지만 여러가지 공격에 대한 허점이나 보안 요구사항 미비, 과도한 연산요구 등으로 추천할만한 프로토콜로는 받아들이기는 어렵다.

그래서, 본 논문에서는 Forward Secrecy를 만족하는 ASPeCT 프로토콜을 예로 들었다. signcryption 기법을 적용하면 그렇지 않은 경우보다 좀더 효율적이었다. 하지만, Forward Secrecy가 도입되었다고 해서 보안요구사항이나 안전성 이 떨어지는 것은 아니었다. 또한, Wong 프로토콜을 분석하였는데 ASPeCT 프로토콜과는 달리 특별한 조치를 취하지 않고도 프로토콜 자체적으로 Forward Secrecy가 만족된다. 그래서, 연산량이나 통신 오버헤드 관점에서 저전력을 가진 클라이언트에게 아주 효율적인 프로토콜로 판단된다.

참고문헌

[1] G. Horn, K. M. Martin and C. J. Mitchell, "Authentication protocols for mobile network environment value-added services," *IEEE Trans. on Vehicular Technology,* vol. 51, Issue 2, pp. 383-392, March 2002.

[2] Y. Zheng, "Compact and unforgeable session key establishment over an ATM network" In *Proceeding of IEEE INFOCOM'98.* pp. 411-418, 1998.

[3] K. Lee and S. Moon, "AKA protocols for mobile communications," ACISP 2000.

[4] D. S. Wong and A. H. Chen, "Efficient and mutually authenticated key exchange for low power computing devices", ASIACRYPT 2001, LNCS 2248, pp. 272-289, 2001.

[5] D. Park, C. Boyd and S. Moon, "Forward secrecy and its application to future mobile communications," PKC 2000, LNCS 1751, pp. 433-445, 2000.

[6] M. Bellare and P. Rogaway, "Entity authentication and key distribution", CRYPTO 93, LNCS 773, pp. 232-249, 1994.