무선 환경에 적합한 ECC 기반의 인증된 키 합의 프로토콜

정재형°, 윤은준, 류은경, 유기영
경북대학교 컴퓨터공학과 정보보호연구실
{nada°, ejyoon, ekryu}@infosec.knu.ac.kr, yook@knu.ac.kr

An ECC-Based Authenticated Key Agreement Protocol suitable for Wireless Environment

JaeHyoung Jeong°, EunJun Yoon, EunKyung Ryu, KeeYoung Yoo
Department of Engineering, Kyoungpook National University, Daegu, Korea

요 약

최근에 Aydos는 ECC를 기반으로 한 무선 환경에 적합한 인증된 키 합의 프로토콜을 제안하였다. 그러나 그가 제안한 프로토콜은 Sun과 Mangipudi에 의해 각각 몇 가지 암호학적 공격에 취약함을 보였으며, Mangipudi는 더 나아가 그러한 공격에 안전한 개선된 프로토콜을 제안하였다. 하지만 Mangipudi가 제안한 프로토콜은 Sun이 지적한 공격에 대해서 여전히 안전하지 못하다. 본 논문에서는 이러한 모든 공격들에 대해 안전하면서 연산에 있어서 더욱 효율적인 ECC 기반의 인증된 키 합의 프로토콜을 제안한다. 제안하는 프로토콜은 역시 ECC를 기반으로 하고 있으며, 앞서 언급한 Aydos 프로토콜과 Mangipudi 프로토콜 보다 더욱 안전하고 효율적이다.

1. 서 론

최근에 급속한 IT 기술의 발달로 인해 많은 무선 장치들이 생활의 필수품으로 자리 잡고 있다. 이러한 무선 장치들은 낮은 배터리 용량과 적은 메모리 크기를 가지고 있기 때문에, 지금까지 제안된 많은 유선 환경을 위한 프로토콜이 적합하지 않다.

이러한 무선 장치들의 특성으로 인해 알고리즘을 계산하는 과정에서 발생하는 부하가 문제가 될 수 있다. ECC 알고리즘은 훨씬 작은 길이의 키로 다른 알고리즘과 비슷한 강도의 보안성을 제공할 수 있을 뿐만 아니라 적은 크기의 비트를 추가함으로 더 높은 보안성을 제공할 수 있어서 계산의 부담을 줄이는데 용이하기 때문에 무선 장치에 적합하다.

최근에 Aydos[1]는 ECC를 기반으로 무선 환경에 적합한 인증된 키 합의 프로토콜을 제안하였다. 그리고 그는 무선 환경에서 안전하고 효율적인 프로토콜이 되기 위해 만족해야 되는 몇 가지 필수조건들을 정의 했다. 그러나 Sun[4]과 Mangipudi[3]는 Aydos의 프로토콜이 그러한 조건을 모두 만족하지 못한 다는 것을 지적했고, Mangipudi는 그러한 조건을 모두 만족하는 향상된 프로토콜을 제안하였다.

본 논문에서는 Mangipudi의 프로토콜 역시 모든 조건을 만족하지 못한다는 것을 지적하고, 모든 조건을 만족할 뿐만 아니라 연산에 있어서 더욱 효율적인 프로토콜을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 프로토콜의 핵심인 ECDSA 알고리즘에 대해서 설명을 하고, Aydos와 Mangipudi가 제안한 프로토콜의 문제점을 지적한다. 3장에서는 앞 절에서 지적한 문제들을 개선하여 무선 환경에 적합한 ECC 기반의 안전하고 효율적인 프로토콜을 제안하고, 4장에서는 제안된 프로토콜의 분석을 통해서 기존의 프로토콜보다 더욱 안전하고 효율적임을 보인다. 끝으로 5장에서 결론을 맺는다.

2. 관련 연구

본 절에서 먼저 제안하는 프로토콜의 핵심인 ECDSA 알고리즘에 대해 설명을 하고, Aydos와 Mangipudi이 제안한 프로토콜의 문제점들을 지적한다.

2.1. ECDSA

ECDSA(Elliptic Curve Digital signature Algorithm)는 DSA(Digital Signature Algorithm)의 동형으로써, ECC를 기반으로 표준화된 알고리즘이다. ECDSA는 서명 알고리즘과 검증 알고리즘으로 구성되어 있으며, 다음과 같다.

ECDSA Signature Generation

1. Select $k \in_R [1, n-1]$.
2. Compute $kP = (x_1, y_1)$ and convert x_1 to an integer $\bar{x}_1$.
3. Compute $r = \bar{x}_1 \bmod n$. If $r = 0$ then go to step 1.
4. Compute $e = H(m)$.
5. Compute $s = k^{-1}(e + dr) \bmod n$. If $s = 0$ then go to step 1.
6. Return (r, s).

ECDSA Signature Verification

1. Verity that r and s are integers in the interval

$[1, n-1]$. If any verification fails then ("Reject the signature").

2. Compute $e = H(m)$.
3. Compute $w = s^{-1} \bmod n$.
4. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
5. Compute $X = u_1 P + u_2 Q$.
6. If $X = \infty$ then return ("Reject the signature").
7. Convert the x-coordinate x_1 of X to an integer $\bar{x}_1$ Compute $v = \bar{x}_1 \bmod n$.
8. If $v = r$ then return ("Accept the signature"); Else return ("Reject the signature").

2.2. Aydos가 제안한 프로토콜의 문제점

- Forward secrecy : 서버와 사용자의 공개키와 비밀키 쌍을 (*Qu, du*)와 (*Qs, ds*)라고 가정하자. 만약에 한쪽의 비밀키가 깨졌을 때, 공격자는 공개키와 비밀키의 곱을 통해서 상호 합의된 키 *Qk.x*를 얻을 수 있다. 그리고 *Qk.x*를 이용하여 전송된 암호문 C_0 또는 C_1을 복호하여 세션키를 만들기 위해 사용되는 랜덤값 *g*를 얻을 수 있다. 이러한 과정을 통해서 공격자는 쉽게 모든 세션의 세션키를 쉽게 얻을 수 있다. 그러므로 Aydos가 제안하는 프로토콜은 forward secrecy를 제공하지 못한다.

- Non-repudiation : Aydos의 프로토콜은 이 조건을 만족하지 못한다. 왜냐하면 누구든지 CA의 도움 없이 유효한 인증서를 만들 수 있기 때문이다. 예를 들어, 먼저 공격자가 랜덤값 *i* 와 *j* 를 선택한다. 그리고 그는 다음의 계산을 통해 유효한 *r* 과 *s* 를 계산한다.

$$r = (iP * jQ_{ca}).x.$$
$$s = r * j^{-1} \bmod q.$$
$$e = i * s \bmod q.$$

- Key compromised impersonation attack : 사용자 U의 비밀키 *du*가 깨졌다고 가정하자. 이 때, 공격자는 U의 비밀키를 이용하여 다른 사용자 A에게 U인척 할 수 있다. 하지만 U의 비밀키를 이용하여 U에게 A인척 속일 수 없어야 된다. 하지만 Aydos의 프로토콜에서는 익명성의 보장을 위해 서로의 id를 검사하지 않음으로써 간단하게 U의 비밀키를 이용하여 U에게 다른 사용자인척 속일 수 있다.

2.3. Mangipudi가 제안한 프로토콜의 문제점

- Forward secrecy : Mangipudi의 프로토콜에서 유저의 비밀키 *qu*가 세션키를 만드는데 사용되지 않기 때문에 서버의 비밀키 *qs*가 깨졌다고 가정을 한다. 그 외의 과정은 위의 것과 동일하므로 설명은 생략한다.

- Non-repudiatio : Mangipudi의 프로토콜에서도 역시 CA의 도움 없이 누구나 유효한 인증서를 만들 수 있다. 과정은 위의 것과 동일하기 때문에 생략한다.

3. 제안한 프로토콜

앞 절에서 설명한 프로토콜들의 문제를 해결하기 위해서 본 논문에서는 더욱 안전하고 효율적인 프로토콜을 제안한다. 제안하는 프로토콜은 2단계로 구성되어 있다. 먼저 첫 단계인 초기화 단계에서는 서버와 사용자가 안전한 채널을 통해서 오프라인으로 CA(Certification Authority)로부터 인증서를 얻는 과정이다. 이 때 인증서는 CA에 의해 ECDSA 알고리즘을 이용하여 계산되어 진다. 두 번째는 상호 인증단계로써 온라인으로 서버와 사용자가 서로를 인증하면서 세션키를 생성한다.

초기화 단계 (Initialization Phase)

- Server Initialization phase : 서버는 랜덤하게 자신의 개인키 q_s를 선택하고 공개키 $Q_s = q_s \times P$를 계산한다. 그리고 Q_s를 CA에게 보낸다. 그 때 CA는 랜덤하게 k_s를 선택하고 $Q_{cs} = k_s \times Q_s$를 계산한다. 이 때 만들어진 Q_{cs}는 CA가 사용자의 인증서를 만들 때 사용이 된다. 그리고 추후에 서버를 분별하기 위해서 필요한 I_s와 타임 스탬프 T_s를 선택한다. 마지막으로 서버에게 Q_{ca}, I_s, T_s, Q_{cs}를 보내고 CA는 Q_s, T_s, Q_{cs}를 저장한다. 서버는 CA로부터 Q_{ca}, I_s, T_s, Q_{cs}를 받은 후, CA로부터 받은 값들과 함께 Q_s, q_s^{-1}를 저장한다.

- Client Initialization phae : 사용자는 랜덤하게 자신의 개인키 q_u를 선택하고 공개키 $Q_u = q_u \times P$를 계산한다. 그리고 Q_u를 CA에게 보낸다. 그 때 CA는 랜덤하게 k_u를 선택하고 $R_u = k_u \times P$를 계산한다. 그리고 사용자를 위해서 유일한 I_u와 타임 스탬프 T_u를 선택하고, ECDSA 알고리즘을 이용하여 인증서 (r_u, s_u)를 계산한다. 마지막으로 사용자에게 Q_{ca}, Q_s, T_s, I_u (r_u, s_u), T_u를 보낸다. 사용자는 CA로부터 Q_{ca}, Q_s, T_s, I_u (r_u, s_u), T_u를 받은 후, CA로부터 받은 값들과 함께 Q_u, q_u^{-1}를 저장한다.

상호 인증 단계 (Mutual Authentication Phase)

- 사용자는 랜덤하게 k_u를 선택하고 $K_u = k_u \times P$ 를 계산한다. 그리고 K_u의 x-coordinate를 키로 사용하여 (r_u, s_u), T_u, Q_u를 암호화 한다. 이 것을 C_0이라고 하고, $D_u = r_u \times Q_s$와 함께 서버에게 보낸다. 서버는

C_0과 D_u를 받은 후, D_u에 q_s^{-1}를 곱하여 K_u'를 구한다. K_u'를 이용하여 C_0을 복호화 하고, T_u가 유효한지를 검사한다. 만약 유효하다면, 다음과 같이 ECDSA 검증 알고리즘을 이용하여 사용자의 인증서를 계산한다.

$$e_u = H(Q_u.x, T_u, Q_{cs})$$
$$c = s_u^{-1}$$
$$u_1 = c \cdot e_u, \quad u_2 = c \cdot r_u$$
$$R = u_1 \times P + u_2 \times Q_{ca}$$

여기서 R의 x-coordinate와 인증서의 r_u가 똑같다면 인증서는 유효한 것이다. 이제 서버는 자신이 선택한 k_s와 사용자의 공개키 Q_u를 곱하여 D_s를 계산한다. 그리고 이전에 계산한 K_u'와 k_s 곱하여 SK, 즉 둘 사이에 합의된 세션키를 계산한다. 생성된 세션키와 사용자로부터 받은 값을 사용자에게 확인시키기 위해서 $C_1 = H(D_u, SK)$을 계산하고, D_s와 함께 사용자에게 전송한다. C_1과 D_s를 받은 후, 사용자는 D_s에 q_u^{-1}를 곱하여 서버가 만든 세션키 SK와 같은 값을 만든다. 자신이 만든 것과 서버가 만든 것이 같은 값임을 확인 하기 위해서 $C_1' = H(D_u, SK)$를 계산한 후, C_1과 C_1'를 비교한다. 만약 다르다면 프로토콜을 중단한다. 그렇지 않다면, 세션키의 확인을 위해서 $C_2 = H(D_s, SK)$를 계산한 후, 서버에게 보낸다. 서버는 $C_2' = H(D_s, SK)$를 계산하고, 이 값과 사용자로부터 받은 C_2과 비교를 하게 된다. 만약 다르다면 프로토콜을 중단한다.

4. 안정성 분석 및 효율성 비교

본 논문에서 제안한 프로토콜은 ECDSA를 기반으로 세션키를 합의 하기 때문에 forward secrecy가 제공된다. 그리고 초기화 단계에서 CA로부터 얻는 인증서에는 서버만이 알고 있는 값이 들어가기 때문에 공격자는 인증서를 CA의 도움 없이는 만들 수 없다. 따라서 Non-repudiation을 제공한다. Key compromised impersonati-on attack은 서버의 공개키를 이용함으로써 Mangipudi가 제안한 프로토콜과 마찬가지로 간단하게 막을 수 있다. 본 논문에서 제안한 프로토콜의 효율성은 다음의 표 1에 잘 나타나 있다.

제안한 프로토콜은 추가적인 포인트 곱셈과 해쉬 연산이 필요하지만 대칭키 연산이 줄어들었다. 그리고 키 합의 프로토콜은 합의하여 생성한 키에 대한 확인 절차를 가져야 하는데, 이전의 두 프로토콜은 확인 절차를 가지지 않는다. 이를 위해서는 추가적인 메시지 라운드가 필요하다. 하지만 제안한 프로토콜은 3번의 라운드로 키 확인까지 수행한다. 이러한 결과를 통해서 본 논문에서 제안한 프로토콜이 더욱 안전하면서 효율적인 프로토콜임을 알 수 있다.

표 1. 프로토콜의 필요한 연산 비교

	Proposed scheme	Mangipudi scheme	Aydos scheme
Point Mul.	3	2	1
Symmetric Enc.	1	1	1
Symmetric Dec.	-	1	1
Inverse	-	-	1
Mul.	1	-	2
Hash	2	1	1
Message Round	3	3	4
Random Nmber Gen.	1	1	1

5. 결론

본 논문에서는 Aydos와 Mangipudi가 제안한 무선 환경에 적합한 인증된 키 합의 프로토콜의 문제점을 지적하고 그 해결책으로 개선된 프로토콜을 제안하였다. 제안한 프로토콜은 Sun과 Mangipudi이 각각 지적한 문제들을 모두 만족함과 동시에 연산에 있어서 더욱 효율적이며, 작은 길이의 키로 높은 보안성을 제공하는 ECC를 기반으로 하기 때문에 무선 장치와 같은 제한된 자원을 가진 장치를 이용하는 무선 통신에 적합하다.

참고 문헌

[1] M. Aydos, B. Sunar and C. K. Koc, "An elliptic curve cryptography based authentication and key agreement protocol for wireless communication", 2nd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Dallas, Texas, October 30, 1998.

[2] M. Aydos, T. Yanik, and C.K. Koc, "High-speed implementation of an ECC-Based wireless authentication protocol on an ARM microprocessor", IEE Proceedings Communication, vol. 148, no. 5, pp. 273 279, 2001.

[3] K. Mangipudi, N. Malneedi, R. Katti and H. Fu, "Attacks and Solutions on Aydos-Savas-Koc's Wireless Authentication Protocol", in proceedings of Symposium on Network and Security management, the IEEE Global Telecommunications Conference, November 29 December 3, 2004, Dallas, TX.

[4] Hung-Min Sun, "Cryptanalysis of Aydos et al.'s ECC-Based Wireless Authentication Protocol", Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service.