ISSN(Print): 2287-5891

http://dx.doi.org/10.3745/KTCCS.2013.2.1.051

Design and Implementation of EAI(Enterprise Application Integration) System for Privacy Information

Kim Yong Deok† · Jun Moon Seog††

ABSTRACT

This paper describes the design and implementation of the PKI-based EAI system which is used for delivery of sensitive personal information between business systems. For this purpose, we propose a key exchange protocol with some key process : Diffie-Hellman Schema is used to provide forward secrecy, public key-based digital signature is used for EAI Server authentication, data integrity. In addition, in order to minimize the performance impact on the overall EAI systems. The EAI server was designed simply to be used only as a gateway. This paper shows the implementation of Korea public key authentication algorithm standard and a symmetric encryption algorithm for data encryption.

Keywords : EAI, PKI, Certification, Key Exchange

개인정보 보호를 위한 EAI 시스템 설계 및 구현

김 용 덕† · 전 문 석††

요 약

본 논문에서는 업무시스템 간 민감한 개인정보 전달을 위해 PKI기반 EAI 시스템을 설계, 구현하고 이에 대하여 기술한다. 이를 위해 업무시스템이 EAI 서버를 통해 안전하게 데이터를 연계하기 위해 EAI 특성에 맞는 키 교환 프로토콜을 제안한다. 키 교환을 위해서 전방향 안정성에 널리 사용되는 Diffie-Hellman기법을 적용하였으며, 키 교환 주체에 대한 인증 및 자료의 무결성을 위해 전자서명방식을 혼용한다. 또한 전체 EAI 시스템의 성능에 대한 영향을 최소화하기 위해 인증, 키 교환 및 암/복호화를 업무시스템에서만 이루어지도록 구성하고, EAI 서버는 단순히 연계통로로만 사용하도록 구성한다. 그리고 인증 및 암/복호 알고리즘은 국내표준을 준용할 수 있도록 EAI 시스템을 설계, 구현한다.

키워드 : EAI, PKI, 인증서, 키교환

1. 서 론

인터넷의 활성화와 비즈니스에서 IT기술의 도입이 가속화되면서 기업들은 업무 간 연계를 위해서 각각의 업무시스템을 Peer-to-Peer 형식으로 연결하기 시작했지만, 방대한 데이터에 대한 안정성, 성능 그리고 모니터링 등 여러가지 고려사항이 여전히 존재하였다. 이를 해결하기 위해 어플리케이션 간 연동만을 전문으로 하는 아키텍처와 솔루션이 필요하게 되었는데, 그것이 EAI(Enterprise Application Integration)이다. 기업들은 EAI를 통해 다수의 어플리케이션에 분산되어 있는 데이터 등을 전사적 차원에서 공유할 수 있을 뿐 아니라 비즈니스 프로세스를 단순화하고 자동화함으로써 기존 시스템의 효율성을 재고할 수 있으며 새로운 어플리케이션 개발 시 시간과 비용을 줄일 수 있게 되었다[1].

최근 개인정보침해에 대한 사회적인 이슈로 민감한 정보를 저장·전송할 경우 안정하게 암호화기술을 적용하거나 이에 상응하는 조치를 취할 것을 강제하고 있다[2]. 때문에 EAI 서버를 통해 업무시스템 사이에 민감한 정보를 전송할 경우 이에 부합한 보호조치를 고려해야 한다. 일반적으로 EAI는 중앙의 EAI 서버가 모든 메시지나 어플리케이션 간 커뮤니케이션을 관리하는 구조로 송신시스템은 EAI 서버로 메시지를 전달만 하고 메시지에 대한 변환 및 라우팅은 EAI 서버가 담당을 한다.

다음은 EAI에 보안을 적용한 현황이다. 첫째 기존 EAI 시스템 내 정보보호는 송신시스템과 EAI 서버, EAI 서버와 수신시스템 사이를 SSL/TLS 등으로 각각 분리하여 보안을 적용하였다. 둘째, 기존 EAI 시스템이 일반적으로 내부

† 정 회 원 : 숭실대학교 컴퓨터학과 박사과정
†† 종신회원 : 숭실대학교 컴퓨터학과 교수
논문접수 : 2012년 10월 19일
수 정 일 : 1차 2012년 11월 19일, 2차 2012년 12월 14일
심사완료 : 2012년 12월 14일
* Corresponding Author : Jun Moon Seog(mjun@ssu.ac.kr)

자들이 공유하는 동일 망 내에 위치하는 다수의 서버들을 연계 프레임워크로 묶는 것이기에 고정키 방식을 많이 적용하였다. 셋째, 모든 메시지는 EAI 서버에 의해 변환 및 라우팅되기 때문에 중간공격자(Man in the Middle)공격에 취약한 구조를 갖고 있다. 최근 EAI 적용 업무범위가 금융권 등 외부연계로 확대되는 점을 고려하여 중요정보를 안전하게 교환하기 위해 업무시스템 간 단대단 보안(End-to-End)을 적용해야 하며, 동적 키 관리 및 멀티 도메인 인증이 반영되어야 한다. 또한 보안을 적용할 경우 중간 중계자인 EAI 서버를 포함한 통합된 보안구조를 갖도록 해야 한다.

따라서 본 논문에서는 업무시스템이 EAI 서버를 통해 안전하게 데이터를 연계하기 위해 EAI 특성에 맞는 키 교환 프로토콜을 제안한다. 키 교환을 위해서 전방향 안정성에 널리 사용되는 Diffie-Hellman기법을 적용하였으며, 키 교환 주체에 대한 인증 및 자료의 무결성을 위해 전자서명방식을 혼합하였다.

본 논문의 구성은 2장에서 본 논문의 근간이 되는 EAI 및 보안 기술에 대해 살펴본다. 3장에서는 본 논문에서 제안하는 EAI 모델에 맞는 키 교환 프로토콜을 제시한다. 4장에서는 구현한 결과 및 성능을 분석하고, 5장에서 결론을 맺는다.

2. 관련 연구

2.1 EAI(Enterprise application Integration)

EAI란 단일 기업 내 이기종 어플리케이션을 통합하는 것으로 새로운 미들웨어를 이용하여 비즈니스 프로세스를 중심으로 기업 내의 각종 어플리케이션을 통합하는 과정 및 관련 기술로[3], 비즈니스 환경에서 서비스를 연결, 중재 및 관리하기 위한 기반을 제공한다. 기술적인 측면에서 EAI는 기존 어플리케이션을 변경하지 않고 어플리케이션 간 데이터 공유와 비즈니스 프로세스를 통합하는 프로세스로, 최근에는 기존 EAI와 B2Bi(Business to Business Integration)에 머무르지 않고 기업 외부의 응용 프로그램까지 포함하고 있다[4]. EAI의 특징은 모든 메시지가 EAI 서버를 매개체로 하여 메시지 변환 및 업무시스템으로 라우팅된다. 또한 어플리케이션 측면에서 EAI 서버는 TTP(Trust Third Party, 신뢰기관)의 역할을 수행한다고 볼 수 있다. 일반적인 EAI 유형은 다음과 같이 분류한다[5].

EAI유형 중 Hub & Spoke방식은 중앙의 Hub가 모든 메시지나 어플리케이션 간 커뮤니케이션을 모니터링 할 수 있으며, Spoke로 구성되는 송신시스템 또는 수신시스템에 대한 감지가 용이하기 때문에 최근 많이 사용된다[6].

2.2 공개키 기반 인증

공개키 암호 알고리즘(Public-Key Crypto Algorithm)은 암호화 하거나 복호화 하는데 쓰이는 두 개 키가 한 쌍으로 존재하는 알고리즘으로[7], 하나의 키는 누구든지 사용할 수 있게 공개하고 다른 하나는 외부에 공개되지 않도록 보관한다. 이때 공개하는 키를 공개키(public key)라고 하며 자신만이 볼 수 있도록 보관하는 키를 개인키(private key)라고 한다.

Table 1. Features of EAI type[6]

구분	P2P	Hub & Spoke	Bus
적용 대상	연계 어플리케이션이 3개 이하	연계 어플리케이션이 3개 이상	연계 어플리케이션이 3개 이상
라우팅	연계대상이 고정되어 있는 경우	연계대상이 복잡하고 동적으로 변경	연계대상이 고정되어 있는 경우
데이터 변환	가능	가능	가능
장단점	-연계 어플리케이션이 많을 경우 변경시간과 비용이 많이 소요	-변경비용이 적게 소요 -시스템 성능에 부하를 적게 함	-어플리케이션이 데이터를 버스로 보내는 방법을 알아야 함

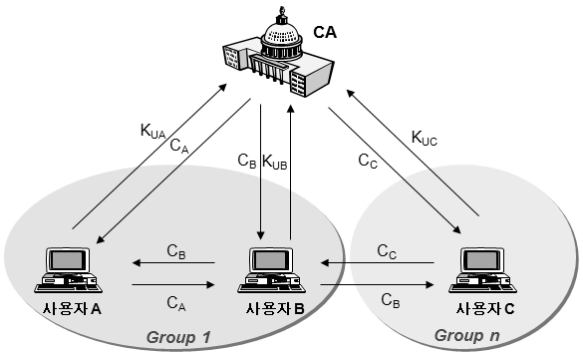

Fig. 1. PKI based Authentication Process

공개키 기반구조(Public Key Infrastructure)는 기본적으로 암호시스템 및 서명시스템에서 요구되는 사용자 공개키에 대한 무결성과 인증성을 보장하기 위하여 인증기관에 의하여 발행되는 인증서에 바탕을 두고 있다[8]. 따라서 인증기관에서 발급한 인증서를 기반으로 전자서명을 검증한다는 것은 Fig. 1과 같이 동일 도메인뿐만 아니라 멀티 도메인 간 전자서명 주체에 대한 인증과 서명정보에 대한 무결성을 함께 제공한다는 것을 의미한다.

2.3 키 교환 프로토콜

공개키 암호 알고리즘은 대칭키 암호 알고리즘에 비해 암/복호하는 시간이 오래 걸리기 때문에 정보를 암/복호하는 경우 대부분 속도가 빠른 대칭키 알고리즘을 이용한다. 그러나 비밀키는 사용하기 전에 이용 주체들이 비밀키를 사전에 공유해야 하므로 키 관리에 문제가 발생한다[4, 10]. 따라서 통신하려는 두 통신 엔티티에 동일한 암호화 키를 제공하기 위해 키 교환 프로토콜이 필요하다[9]. 키 교환 프로토콜에서 요구되는 안정성은 키 기밀성(Key secrecy), 전방향 안전성(Forward secrecy) 그리고 기지 키 공격에 대한 안정성(Known-key secrecy이다[11, 13]. 키 기밀성은 공격자가

비밀키에 대한 어떠한 정보도 얻을 수 없어야 함을 의미하고, 전방향 안전성은 어떠한 공격자라도 개인키 노출 이전에 교환한 비밀키에 대한 어떠한 정보도 얻을 수 없어야 함을 의미한다. 그리고 기지 키 공격에 대한 안전성은 여러 세션의 비밀키들이 노출되어도 노출되지 않은 세션의 키 비밀성에 영향을 주지 않아야 함을 의미한다.

인증과 키 교환이 동시에 이루어지는 인증된 키 교환 프로토콜(AKE: Authenticated Key Exchange)중 SIG-DH 프로토콜은 전방향 안전성을 위해 널리 사용되는 Diffie-Hellman 기법에 중간 공격자(Man in the middle)공격에 취약하다는 문제점을 보완하기 위해 전자서명을 추가한 방식으로[11, 12] Cannetti와 Krawczyk에 의해서 제안되었다.

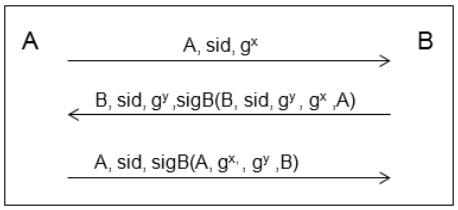

Fig. 2. SIG-DH protocol

step 1 : 사용자 A는 랜덤 값 x를 이용해서 g^x를 계산해서 사용자 B에게 전달한다.

step 2 : 사용자 B는 자신의 랜덤 값 y를 생성하여 g^y를 계산 한 후, g^x, g^y, sid, A, B를 자신의 개인키로 서명하여 A에게 전달한다.

step 3 : 사용자 A도 A, g^x, g^y, B를 개인키로 서명하여 사용자 B에게 전달한다. 이 때 생성되는 세션 키는 g^{xy}이다.

국내의 경우 금융권에 적용 가능한 키 교환 프로토콜로 Diffie-Hellman기법에 전자서명을 혼합한 키 교환 프로토콜인 FKE1(Financial Key Exchange)이 제안되었다[13].

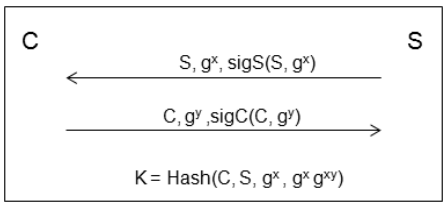

Fig. 3. FKE1 protocol

step 1 : 사용자(C)가 서버(S)에 접속을 하면 서버는 랜덤 값 x를 이용해서 g^x를 계산한 후 자신의 개인키로 서명하여 사용자에게 전단한다.

step 2 : 사용자는 서버의 인증서를 통해 서명검증을 한다.

step 3 : step 2가 완료되면 사용자는 랜덤 값 y를 생성하여 g^y를 계산 한 후 자신의 개인키로 g^y를 서명하여 서버에게 전달한다. 생성되는 세션 키는 step 3에서 상호 전달된 모든 정보를 해쉬한 값이다.

2.4 EAI 시스템에 보안기술 적용상의 문제점

EAI 시스템에 기존 키 교환 프로토콜을 적용하기 어려운 점은 다음과 같다.

첫째, 업무시스템 간 모든 메시지는 EAI 서버를 통해 전달되는 구조이기에 중간 공격자과 같은 악의적인 공격에 취약하다. 따라서 업무 시스템들이 비밀키 교환을 할 경우 EAI 서버를 포함해서 구성해야 한다. 때문에 EAI 서버를 인증할 수 있는 방안이 마련되어야 한다.

둘째, EAI 시스템은 정보를 정확하게 전달하는 것이 주요 역할이기 때문에 EAI 서버를 통한 송신시스템과 수신시스템 간 메시지 전달은 일반적으로 2 Pass 프로토콜로 이루어진다. 즉, 송신시스템이 메시지를 EAI 서버에 전달하면 EAI 서버는 메시지 정보를 기반으로 해당 메시지를 수신시스템에 전달하고 그 결과를 송신시스템에 돌려주는 구조이다. 따라서 보안 프로토콜을 적용할 경우 2 Pass 프로토콜 이내에서 트랜잭션이 완료되도록 설계해야 한다.

셋째, EAI 시스템은 데이터가 동시에 대량으로 전달될 수 있는 구조이다. 따라서 일부 민감한 정보를 전자봉투 방식으로 암호화 하여 전달하면 매 암호 트랜잭션마다 비 대칭키 연산을 해야 하므로 이를 고려한 키 교환 프로토콜이 요구된다.

넷째, 최근 EAI 시스템을 도입한 기업에서 정보보호를 위해 EAI 서버를 경유하지 않고 업무시스템에 직접 접속하는 것을 보안 정책상 통제하고 있다. 이는 키 교환을 위해 송신시스템이 수신시스템에 직접 접속하지 못한다는 것을 의미한다. 따라서 키 교환에 EAI 서버를 포함하여야 한다.

3. 보안성이 적용된 EAI 시스템의 설계

3.1 EAI 보안모델

본 논문에서는 2.4절과 같은 EAI만의 특성을 반영하여 EAI 시스템을 설계한다. 이를 위해 EAI유형 중 일반적으로 많이 사용되는 Hub & Spoke방식에 보안을 적용한다. EAI 서버는 업무시스템들 사이에서 메시지에 대한 변환 및 라우팅을 담당하는 TTP 역할을 수행하기 때문에 안전한 키 교환을 위해 EAI 서버가 키 교환에 참여하도록 구성한다. 키 교환과정에서 신뢰기반을 마련하기 위해 EAI 서버를 신뢰한다는 정보(인증서 포함)를 모든 업무시스템이 갖고 있어야 한다. 하지만 보안 적용에 따른 EAI 서버 영향을 최소화하기 위해 메시지 암/복호화 과정에는 EAI 서버를 단순 전달자 역할 만을 하도록 구성한다.

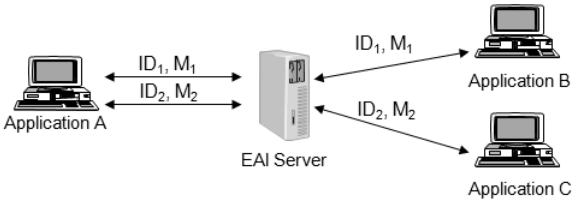

Fig. 4. EAI Model

또한 Fig. 4와 같이 키 교환을 포함한 모든 메시지는 EAI 서버가 인식할 수 있는 구조로 설계한다. 특히 EAI 서버가 메시지 변환 및 라우팅을 결정할 수 있도록 메시지를 유일하게 구분할 수 있는 ID를 모든 메시지에 포함한다.

3.2 EAI 키 교환 프로토콜

본 논문에서 제안한 키 교환 프로토콜인 EAI-KE(EAI Server based Key Exchange)는 2.3절의 다른 프로토콜과 동일하게 전 방향 안정성을 위해 많이 사용되는 Diffie-Hellman 알고리즘을 사용한다. 서론에서 언급한 바와 같이 최근 EAI를 적용하는 업무범위가 금융권 등 외부연계로 확대되고 있어 멀티 도메인 내 업무시스템 간 인증이 고려되어야 하기에 Fig. 1과 같이 인증기관에서 발급한 인증서를 활용한 전자서명 기법을 Diffie-Hellman 알고리즘에 혼용한다. 또한 기지 키 공격에 대한 안정성을 위해 송신시스템과 EAI 서버, EAI 서버와 수신시스템 간 메시지 전달 및 비밀키 생성 과정에 교환된 랜덤 값이 포함되도록 설계한다. 본 논문에서 사용하는 기호는 Table 2와 같다.

Table 2. Syntax of extended processes

표 기	설 명
Hash(m)	Θ를 안전성 파라미터라 하고 H:{0, 1}*→{0, 1}Θ 인 해쉬 함수
SignA(m)	사용자 A의 개인키로 메시지를 전자서명
VerifyA(m)	사용자 A의 공개키로 전자서명 값을 검증

키 교환을 위한 사전작업으로 EAI 서버를 포함한 모든 참여자는 인증기관으로부터 인증서를 발급받아 자신의 저장 공간에 보관해야 한다. 또한 TTP인 EAI 서버를 위해 업무시스템들은 EAI 서버 인증서를 함께 저장하여 관리해야 한다. 이 인증서를 키 교환과정에서 EAI를 인증할 때 사용한다.

키 교환 프로토콜의 세부 단계는 아래와 같다.

step 1 : Diffie-Hellman 공개 파라미터 p와 g에 대해 p는 소수, g는 p의 원시근이라 할 때 업무시스템(Application) A는 {1, p}내에서 자신의 개인키 값 x를 생성하고 이를 통해 공개값인 $P_A = g^x \bmod p$를 생성한다. 그리고 키 교환 세션을 구분하는 랜덤 값 rd1 값을 생성한다. 이 후 인증기관을 통해 발급한 개인키로 서명한 값 $S_A = Sign^A(rd1, p, g, P_A)$를 계산한다. 모든 정보가 생성되면 메시지를 유일하게 구분하는 ID와 S_A 서명값을 검증하기 위한 인증서(C_A)를 포함하여 전체 메시지(ID, rd1, p, g, P_A, S_A, C_A)를 EAI 서버에게 전달한다.

step 2 : EAI 서버는 키 교환 세션을 위해 랜덤 값 rd2를 생성한 후 자신의 개인키로 서명한 값 $S_E = Sign^E(rd1, rd2)$를 생성한다. 그리고 인터페이스 ID를 기준으로 라우팅정보를 조회한다. 랜덤 값(rd2)과 서명값(S_E)을 포함한 전체 메시지(ID, rd1, p, g, P_A, S_A, C_A, rd2, S_E)를 라우팅정보를 기반으로 업무시스템 B에 전달한다.

step 3 : 업무시스템 B는 EAI가 보낸 서명값(S_E)을 EAI 인증서(C_E)를 이용해 검증하여 신뢰할 수 있는 EAI 서버인지를 확인한다. 이후 어플리케이션 A가 보낸 서명값(S_A)을 인증서 C_A로 검증한다. 검증이 성공하면 업무시스템 B는 {1, p}내에서 자신의 개인키 값 y와 공개값인 $P_B = g^y \bmod p$를 생성한다. 그리고 인증기관을 통해 발급한 개인키로 서명한 값 $S_B = Sign^B(rd1, rd2, P_A, P_B)$를 계산한다. 이 후 인터페이스를 유일하게 구분하는 ID와 S_B 서명값을 검증하기 위한 인증서(C_B)를 포함하여 전체 메시지(ID, rd1, rd2, P_B, S_B, C_B)를 EAI 서버에게 전달한다. 키 교환과정이 종료되면

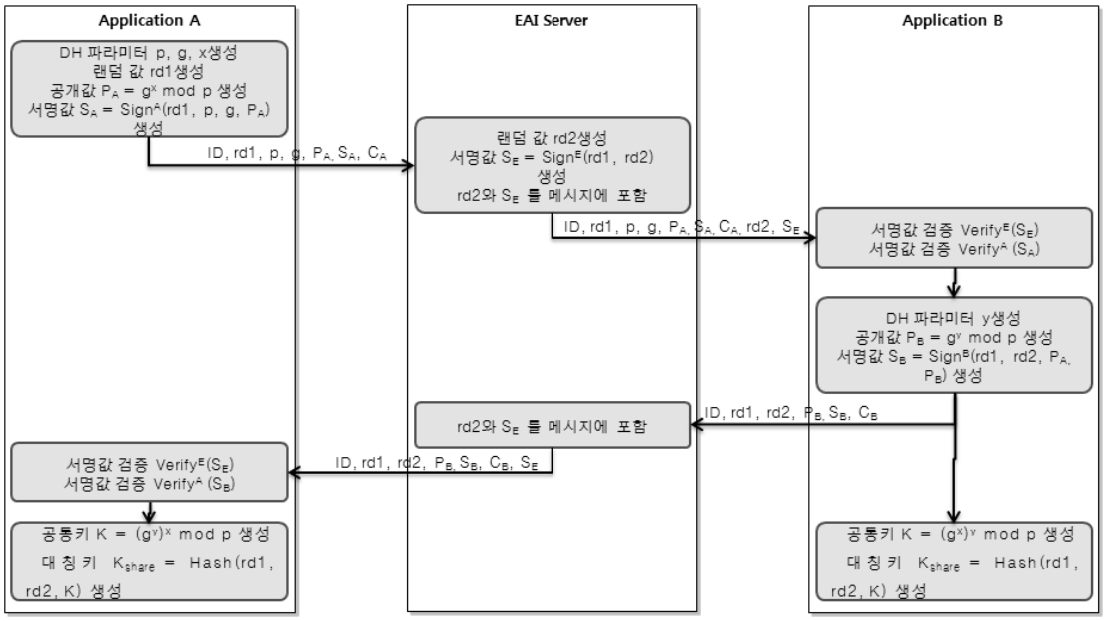

Fig. 5. EAI-KE Key Exchange Protocol

업무시스템 B는 공통키인 $K = (g^x)^y \bmod p$를 생성한다.

step 4 : EAI 서버는 키 교환 세션을 위해 step 2에서 생성한 서명값 S_E을 기존 메시지에 추가하고 전체 메시지(ID, rd1, rd2, P_B, S_B, C_B, S_E)를 업무시스템 A에 전달한다.

step 5 : 업무시스템 A는 EAI가 보낸 서명값을 EAI 인증서(C_E)를 이용해 검증하여 신뢰할 수 있는 EAI 서버인지를 확인한다. 그리고 어플리케이션 B가 보낸 서명값(S_B)을 인증서 C_B로 검증한다. 키 교환과정이 완료되면 업무시스템 A는 공통키인 $K = (g^y)^x \bmod p$를 생성한다.

step 5 이후 어플리케이션 A와 B는 정보를 암호하기 위한 비밀키로 K_{share} = Hash(rd1, rd2, K)를 생성한다.

전반적으로 2.3절의 다른 키 교환 프로토콜과 큰 차이점이 없으나 EAI-KE은 EAI 서버가 프로토콜에 참여되면서 2개의 통신세션을 연결하는 정보와 EAI 서버를 인증하기 위한 정보를 포함한 것이 특징이다.

업무시스템 A의 입장에서 EAI 서버가 연결대상이지만 실제 키 교환대상은 업무시스템 B이다. 때문에 EAI 서버는 업무시스템 A와 EAI 서버 그리고 EAI 서버와 업무시스템 B를 연결하는 정보를 키 교환 프로토콜에 포함한다. step 2에서 업무시스템 A와 EAI 서버와의 세션 정보 값인 rd1과 EAI 서버와 업무시스템 B를 위한 세션 정보 값인 rd2를 EAI 서버가 전자서명하면서 2개의 세션이 연결되었다는 것을 보증한다. 이를 step 3과 step 5에서 EAI 서버의 서명값(S_E)을 검증하면서 EAI 서버 인증 및 세션이 동일함을 확인하도록 한다. 또한 step 3에서 업무시스템 B가 전자서명을 할 때 랜덤 값 rd1과 rd2를 함께 포함한 것을 step 5에서 업무시스템 A가 업무시스템 B 인증하는 과정에서 2개의 세션이 동일함을 함께 확인하도록 한다. 그러나 키 교환 과정에 EAI 서버의 성능에 영향을 최소화하도록 step 2에서 생성한 서명값(S_E)을 step 4에서 재사용 하였으며, 실제 키 교환의 주체인 업무시스템 A와 B가 생성한 키 교환정보는 EAI 서버가 관여하지 않는다.

4. EAI 시스템 구현 및 평가

4.1 EAI 시스템 구현 및 결과

보안을 고려한 EAI 시스템을 구현하기 위한 환경은 Table 3과 같다.

Table 3. EAI System Development Environment

항목		세부 내역
개발언어		Java(JDK 6)
구현서버		x86 계열의 CPU 2.5GHz 4개, 8GB 메모리
소프트웨어	EAI S/W	Oracle Service Bus 11gR1
	WAS	Web Logic 11g
암호 알고리즘	대칭키 알고리즘	SEED
	비대칭키 알고리즘	키 교환(DH), 서명(RSA 1024)
	해쉬함수	SHA-1

구현 시나리오는 Table 4와 같이 사용자 사번을 수신시스템에 전달하면 해당 사번에 해당하는 사용자 이름과 주민번호를 송신시스템에 돌려주도록 구성하였다. 모든 메시지는 평문이지만 개인정보인 주민번호는 암호화하도록 했고, 이를 위해 키 교환을 수행하게 하였다.

Table 4. Scenario Packet

구분	세부내용	input	output	비고
company id	사용자 사번	→	←	
name	사용자 이름		←	
ssn	주민번호		←	암호화 형태로 전달 필요

구현결과 Fig. 6의 경우와 같이 메시지 부분 중 ssn값이 암호화되었음을 확인하였다.

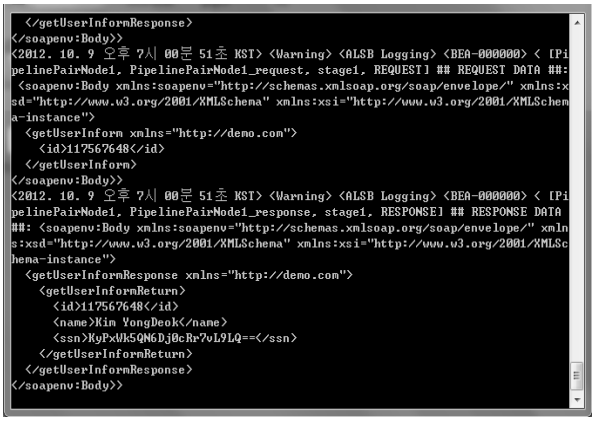

Fig. 6. encrypted message log

또한 송/수신 업무시스템에서 데이터를 암/복호하는 시간을 측정한 결과는 Table 5와 같다. 일반적으로 개인정보의 길이를 5 Kbyte 이내로 가정할 경우업무시스템에서 암/복호하는데 소요되는 시간은 15 ~ 16 millisecond정보로 측정되었다. 즉, 전송시간을 제외한 암/복호 속도는 느끼지는 못할 정도로 매우 빠르게 수행됨을 확인할 수 있다.

Table 5. Speed benchmarks for data transmission (unit : millisecond)

Data Size	총소요 시간 (송신서버⇔ EAI 서버 ⇔ 수신서버)	네트워크 전송시간	암호화 시간 (암호 + 복호)
1KB	140	125	15
5KB	141	125	16
100KB	234	202	32
500KB	1405	1265	140
1MB	59978	59119	859

* 총소요 시간 = 네트워크 전송 시간 + 암호화 시간

4.2 키 교환 프로토콜 평가

본 논문에서는 전체적인 키 교환 프로토콜을 평가하기 위해 SIG-DH, FKE1 및 EAI-KE를 동일하게 EAI 환경에 적용하여 산출하였다.

1) 알고리즘 지수승 연산 측면

국내 금융권 암호 통신 프로토콜에서 가장 많은 비용이 드는 연산은 지수승(exponentiation) 연산이다. 때문에 금융권에서는 서버를 기준으로 지수승 연산을 최소화하는 프로토콜 설계를 요구하고 있다[11]. 따라서 본 논문에서도 EAI 시스템이 금융권을 포함해 많은 분야에서 사용되고 있는 점을 고려하여 키 교환 프로토콜에 대한 성능평가를 위해 지수승 연산수를 사용하였다. 또한 국내 인증기관에서 대부분 사용하는 RSA알고리즘 기반 전자서명에도 지수승 연산이 사용되므로 전자서명/검증 횟수를 성능평가에 포함하였다. Table 6은 본 논문에서 제안한 EAI-KE와 2.3절의 다른 키 교환 프로토콜을 비교하여 성능영향요소를 키 교환 주체인 송신시스템(Client), 수신시스템(Server) 및 EAI 서버로 구분하여 지수승 연산 수와 전자서명/검증 횟수에 대해 정리하였다.

Table 6. Performance Factors of Key Exchange Protocols

키 교환 프로토콜	통신횟수	송신시스템		EAI 서버		수신시스템	
		지수승	서명/검증	지수승	서명/검증	지수승	서명/검증
SIG-DH	3	2	2	-	-	2	2
FKE1 (클라이언트 인증)	2	2	2	-	-	2	2
EAI-KE	2	2	3	0	1	2	3

성능평가 결과 SIG-DH 및 FKE1 프로토콜은 송신시스템과 수신시스템에서 2번의 지수승과 2번의 전자서명/검증 작업이 각각 수행되었으나 EAI-KE는 송신시스템과 수신시스템에서 2번의 지수승과 3번의 전자서명/검증 작업이 각각 수행되었다. 또한 EAI 서버에서도 1번의 전자서명/검증 작업이 별도로 수행되었다. 이는 1회의 키 교환 프로토콜이 완료되면 SIG-DH와 FKE1 프로토콜에 비해 지수승 연산수는 동일하나 3번의 전자서명/검증 작업이 추가로 수행되었다. 그러나 추가 작업이 특정서버에 집중되지 않고 송신시스템, 수신시스템 그리고 EAI 서버가 1번씩 수행하는 점은 성능분산 측면에 긍정적이라 할 수 있다.

2) 통신 파라미터 측면

Table 7는 송신시스템과 수신시스템을 기준으로 송/수신되는 모든 프로토콜 파라미터를 비교한 결과이며 시스템구분자 및 Diffie-Hellman 공개 파라미터 p와 g는 공통항목이기에 제외하였다. EAI-KE는 다른 프로토콜과 다르게 EAI 서버에서 랜덤값과 서명값을 추가로 생성한다. 이를 길이로 산출을 하면 대략 3KB정도 송/수신 메시지가 각각 생성된다.

Table 7. Key Exchange Protocol Parameters

키 교환 프로토콜	송신시스템				수신시스템			
	랜덤값	공개키값	서명값	인증서	랜덤값	공개키값	서명값	인증서
SIG-DH	3	2	2	2	3	2	2	2
FKE1 (클라이언트 인증)	0	2	2	2	0	2	2	2
EAI-KE	3	2	3	2	4	2	3	2

EAI-KE 프로토콜이 알고리즘 지수승 연산과 통신 파라미터가 증가하는 것은 2.4절에서 언급한 것과 같이 EAI 만의 특성을 키 교환 프로토콜에 반영하다 보니 추가로 발생되는 비용이다. 즉, EAI 서버를 중심으로 업무시스템 A와 EAI 서버 그리고 EAI 서버와 업무시스템 B를 연결하는 정보를 키 교환 프로토콜에 포함하고 정보에 대한 무결성과 행위자인 EAI 서버를 인증하기 위한 작업이 추가되었기 때문이다.

3) 프로토콜 안정성 측면

EAI-KE는 SIG-DH및 FKE1 프로토콜과 동일하게 전 방향 안정성을 위해 많이 사용되고 있는 Diffie-Hellman 알고리즘에 전자서명 기법을 혼용하여 구성하였기 때문에 프로토콜 안정성 측면에서 키 기밀성(Key secrecy), 전방향 안전성(Forward secrecy) 그리고 기지 키 공격에 대해 안정성(Known-key secrecy)이 검증되었다[11,12,13]. 그러나 EAI 서버를 키 교환과정에 포함한 부분에 대해서는 추가 분석이 필요하다.

a) 키 기밀성 측면 : 공격자가 키 교환과정에서 참여자로 가정해 키 교환을 하는 경우를 대비해 기존 프로토콜에서는 전자서명을 통해 이를 해결하였다. 그러나 EAI 환경에서는 2.4절에서 언급한 것과 같이 업무시스템 간 모든 메시지는 EAI 서버를 통해 전달되는 구조이기에 중간 공격자가 EAI 서버를 가정하여 키 교환에 참여할 경우 SIG-DH및 FKE1은 공격에 취약한 문제점을 안고 있다. 그러나 EAI-KE는 키 교환에 EAI 서버의 전자서명을 통해 이를 해결하였다. 그리고 이를 위한 전제조건으로 2.2절 공개키 기반구조에서 인증기관을 신뢰하여야 전체 신뢰체계가 마련되는 것과 유사하게 3.1절에서 키 교환에 대한 신뢰기반을 위해 EAI 서버를 신뢰한다는 정보를 모든 업무시스템이 갖도록 하였다.

b) 전방향 안전성 측면 : EAI-KE는 키 교환을 할 때 마다 EAI 서버가 생성한 랜덤 값 rd2가 매번 변경되기 때문에 rd2를 포함하여 생성한 비밀키 값(K_{share})도 매번 다르게 생성된다. 때문에 공격자가 개인키 노출 이전에 교환한 비밀키에 대해 어떠한 정보도 얻을 수 없게 된다.

c) 기지 키 공격에 대한 안전성 측면 : 전방향 안전성 측면과 동일하게 EAI 서버가 생성한 랜덤 값 rd2가 매 키 교환과정마다 변경되기 때문에 비밀키 값(K_{share})도 매번 다르게 생성된다. 따라서 다른 세션에서 비밀키들이 노출되어도 노출되지 않은 세션의 키 비밀성에 영향을 주지 않는다.

d) 업무시스템에서 개인키의 재사용 측면 : 많은 업무시스템이 연계가 되는 EAI 특성상 여러 가지 이유로 업무시스템에서 비밀키를 신규로 생성하지 않고 기존에 생성한 키를 재사용할 수 있다. 이 경우 EAI-KE는 3.2절 step 1과 step 2 내용에서와 같이 송신시스템에서 생성한 랜덤 값 rd1과 EAI 서버가 생성한 랜덤 값 rd2가 매번 변경이 되기 때문에 최종적으로 생성되는 비밀키 값(K_{share})은 매번 다르게 생성됨을 확인할 수 있다. 때문에 업무시스템에서 개인키를 재사용하는 경우에도 비밀키에 대한 기밀성이 보장된다.

e) 2개 세션 동일성 검증 측면 : 3.2에서 서술한 바와 같이 업무시스템 A의 입장에서 EAI 서버가 연결대상이지만 실제 키 교환대상은 업무시스템 B이다. 때문에 EAI 서버가 키 교환과정에 포함되면서 생성된 2개의 세션이 동일하다는 것을 검증할 필요가 있다. EAI-KE는 업무시스템 A와 EAI 서버 그리고 EAI 서버와 업무시스템 B를 연결하는 정보를 키 교환 프로토콜에 포함하고 이를 검증한다. 즉, 업무시스템 A 측면에서는 step 5에서 서명값(S_A) 검증 단계에서 rd1, 서명값(S_B) 검증 단계에서 rd1 과 rd2 그리고 서명값(S_E) 검증 단계에서 rd1과 rd2가 변경되지 않았음을 비교하여 2개의 세션이 동일함을 확인한다. 업무시스템 B 측면에서는 step 3에서 서명값(S_A) 검증 단계에서 rd1 그리고 서명값(S_E) 검증 단계에서 rd1과 rd2가 변경되지 않았음을 비교하여 세션이 동일함을 확인한다.

5. 결 론

본 논문에서는 민감한 개인정보를 EAI 서버를 통해 어플리케이션에 전달하기 위해 EAI 시스템에 기존 키 교환 프로토콜을 적용하기 어려운 점을 분석하고 EAI만의 고려사항을 반영하여 보안을 적용하였다. 이를 위해 EAI 특성에 맞는 키 교환 프로토콜을 제안하였다. 전체 EAI 측면에서 기존 인터페이스에 키 교환을 위한 인터페이스를 추가로 개발해야 하나 전체 인터페이스 수량에 비해 극히 미미한 수준으로 EAI 서버에 영향을 거의 주지 않는 것을 확인하였다.

알고리즘 지수승 연산 측면에서 EAI-KE는 SIG-DH 및 FKE1 프로토콜과 동일하게 송신시스템과 수신시스템에서 2번의 지수승과 2번의 전자서명/검증 작업을 각각 수행하였으나 3번의 전자서명/검증 작업을 추가로 수행되었다. 그리고 프로토콜 파라미터 측면에서는 EAI-KE가 EAI 서버에서 생성되는 랜덤값과 서명값으로 인해 다른 프로토콜에 비해 대략 3KB정도 메시지가 길어졌으나 이는 EAI만의 특성을 키 교환 프로토콜에 반영했기 때문에 추가된 작업으로 판단되며 전체 EAI 인터페이스 횟수에 비해 키 교환 인터페이스 횟수가 극히 적음을 고려한다면 수용할 수 있는 범위일 것이다. 프로토콜 안정성 측면은 EAI-KE가 2.3절의 다른 키 교환 프로토콜과 동일하게 키 기밀성(Key secrecy), 전방향 안전성(Forward secrecy) 그리고 기지 키 공격에 대한 안정성(Known-key secrecy)이 제공됨을 확인하였다. 그러나 SIG-DH, FKE1은 EAI 서버를 키 교환과정에 포함하였을 경우 중간공격자 공격에 취약한 반면 EAI-KE는 안전함을 확인하였다. 그리고 업무시스템에서 생성된 개인키를 다른 세션에서 재사용하는 경우에도 추가적으로 안정성이 제공되고, 업무시스템 A와 EAI 서버 그리고 EAI 서버와 업무시스템 B로 분리된 2개의 세션이 동일함도 추가로 확인하였다.

끝으로 비즈니스 환경에서 서비스를 연결, 중재 및 관리하기 위한 기반 인프라인 EAI에 본 논문에서 제시한 보안 모델을 적용한다면 기업의 정보보호 및 대고객 서비스 측면에 안정적인 서비스를 제공할 수 있을 것이다. 그리고 향후 공유된 키를 재사용하는 부분을 개선하여 키 교환을 더욱 줄일 수 있다면 EAI 서버 및 업무시스템 성능향상에 도움이 될 것이다.

참 고 문 헌

[1] Eun-Ok Ha, Yoon-Ho Kim, “Design and Implementation of Web-based Monitoring System for an EAI Environment”, CALS/EC Vol.14, No.3, pp.2, 2009.

[2] Ministry of Pubic Administration and Security, Private Information Protection Law, 2012.

[3] Young-Bae Kim, “A case study on the effective Enterprise Application Integration”, Yonsei University, pp.8, pp.49, 2007.

[4] Sung-Doke lee, Dong-Soo Han, “A Web Services based e-Business Application Integration Framework”, Journal of computing science and engineering Vol.11, No.6, pp.1, 2005.

[5] Ho-Ki Nam, Sang-Min Park, Jong-Hyun Kim, Sung-Ah Jung, “The Implementation of Enterprise Application Integration System in ERP Environment”, Journal of Korea Safety Management & Science, Vol.12, No.3, pp.3, 2010.

[6] 하은욱, 김윤호, “EAI환경에서의 웹기반 모니터링 시스템의 설계와 구현”, CALS/EC Vol.14, No.3, pp.110, 2009.

[7] Kyung-Jae Ha, Cheol-Gon Moon, “Development of Web-Based mail-System using the Public-Key Encryption Algorithm”, Journal of KIISE, pp.2, 2000.

[8] 염흥렬, “IETF공개키 기반구조 및 PKI기반 응용 표준화 동향”, Journal of KIISC, Vol.14 No.2, pp.2, 2004.

[9] Seung-Han Cho, Chong-sun Hwang, “Design of Key Distribution Protocol using a New Secrecy-Analysis Method”, Journal of KIISE, Vol.18, No.2, pp.1, 1991.

[10] Wikipedia, Key Exchange [Internet], http://en.wikipedia.org/wiki/Key_exchange

[11] Ran Canetti, Hugo Krawczyk, “Analysis of key-exchange protocols and their use for building seure channels”, Springer-Verlag, Eurocrypt'01, pp.453-473, 2001.

[12] 변진욱, “강화된 키 교환 프로토콜의 안전성 모델에 관한 연구”, Journal of KIISC, Vol.20, No.2, pp.81-83, 2010. 4.

[13] Seon Jong Kim, Jeong Ok Kwon, “Secure Key Exchange Protocols against Leakage of Long-term Private Keys for Financial Security Servicers”, Journal of KIISC, Vol.19, No.3, pp.120-129, 2009. 6.

[14] Yoon-Jin Lee, Jae-Guen lee, In-june Jo, “Enhanced Diffie-Hellman Key Distribution using Mobile-phone”, International journal of maritime information and communication sciences , pp.2564-2567, Vol.134, No.12, 2009. 12.
[15] Rakesh Jaiswal, “Security Concerns in EAI”, Wipro Technologies, pp.3-5, 1998.
[16] Chul Sur, Young-Ho Park, Kyung-Hyune Rhee, “A Multi-receiver Certificates Encryption scheme and Its Application”, Journal of Korea Multimedia Society, pp.1, Vol.14, No.6, 2011.
[17] Wei Dai, Crypto++ 5.6.0 Benchmarks [Internet], http://www.cryptopp.com/benchmarks.html
[18] Matjaz B Juric, Professional J2EE EAI, WROX, 2002
[19] E. Rescorla, Diffie-Hellman Key Agreement Method, IETF RFC 2631, 1999.
[20] Jeremy Westernman, "SOA today, Introduction to service-oriented Architecture, DBReview.
[21] William stallings, Cryptography and Network security, Prentice Hall, 1998.
[22] Jong-Hun Park “BPM and Enabling Technology”, BPM Solution Korea Conference, 2004.
[23] Whitfield Diffie and Martin Hellman, “New Directions in Cryptography,” IEEE Transaction on Information Theory, Vol.1T-11, No.6, November, 1976.

김 용 덕
e-mail : dragongood@naver.com
1997년 인천대학교 전자계산학과(학사)
2002년 숭실대학교 정보통신융합학과 (석사)
2003년~현 재 숭실대학교 컴퓨터학과 박사과정
관심분야 : 정보보호, EAI, 계정관리, 권한관리, 암호학

전 문 석
e-mail : mjun@ssu.ac.kr
1981년 숭실대학교 컴퓨터학과 학사
1986년 University of Maryland 전산과 (석사)
1989년 University of Maryland 전산과 (Ph. D.)
1989년 Morgan State University 전산수학과 조교수
1989년 New Mexico State University 부설 Physical Science Lab. 책임연구원
1991년~현 재 숭실대학교 컴퓨터학부 정교수
관심분야 : 정보보호, 전자여권, 전자상거래, 암호학