DOI: 10.3745/KIPSTD.2008.15-D.5.699

ECC 알고리즘 기반 모바일 웹 서비스 시스템의 성능 향상

김 용 태[†] · 정 윤 수[††] · 박 길 철[†††]

요 약

인터넷의 대중화로 웹에 대한 의존도가 높아지고 사용자가 증가로 인하여 웹 서비스 성능과 커뮤니케이션의 보안 문제가 중요한 이슈가 되고 있다. 기존의 웹 서비스 기술은 동시 클라이언트 개수의 제한과 서버 처리량의 감소 그리고 평균 응답 시간을 증가시켜 웹 어플리케이션 서버의 성능을 감소시킨다. 그리고 커뮤니케이션 보안을 위한 메시지 암호화 작업과 초기 handshake 비용은 연결에 필요한 계산 시간을 증가시켜 전송 속도와 서버 성능을 감소시킨다. 따라서 본 논문에서는 보안 요구사항을 만족하고 동시에 웹 서비스를 위하여 서버의 과부하를 개선하고 웹 서버 아키텍처의 신뢰성과 안전성을 위하여 타원곡선 암호화(ECC) 알고리즘을 이용하여 암호화 처리를 수행하고, 향상된 성능과 지연 처리에 필요한 기술을 제공하는 개선된 모바일 웹 서버를 제안한다.

키워드 : 웹 서비스, SOAP, Apache, SSL 프로토콜, ECC 알고리즘

Performance Enhancement of ECC Algorithm-based Mobile Web Service System

Yong-Tae Kim[†] · Yoon-Su Jeong[††] · Gil-Cheol Park[†††]

ABSTRACT

By the dependence on Web from popularization of internet and increasing number of users, web services capability and security problem of communication is becoming a great issue. Existing web services technology decrease the capability of web application server by limiting the number of synchronous client, decreasing the processing load and increasing average response time. The encryption process to secure communication and the early expense of handshake decrease transmission speed and server capability by increasing the calculation time for connecting. Accordingly, this paper executes an encryption procedure by elliptical encryption algorithm to satisfy secure demands, improve the overload of server for web services and get reliability and security of web server architecture and proposes an improved mobile web sever which provides better ability and the techniques for deferred processing.

Keywords : Web Services, SOAP, Apache, SSL Protocol, ECC Algorithm

1. 서 론

웹 기반의 XML 기술을 이용하는 웹 서비스는 정보 기술 변화의 하나이다. XML 발표 이후 웹은 단순한 HTML 정보 교환 형태에서 비즈니스 어플리케이션 영역으로 확대되고, 인터넷 사용자의 증가로 네트워크 트래픽 증가와 서버의 처리 지연 등 많은 문제가 발생한다. 그리고 웹 서비스는 아파치(Tomcat)와 멀티쓰레드 패러다임 기반으로 클라이언트의 종료까지 모든 요구를 체크하는 부담과 쓰레드에 의한 동시 클라이언트 개수의 제한으로 서버 처리량 감소와 평균 응답 시간의 증가로 웹 어플리케이션 서버의 성능을 감소시킨다[1].

오늘날 웹 서비스의 중요성이 부각되면서 많은 비즈니스 어플리케이션의 상업적 목적을 위하여 웹 서비스의 종류가 다양해지고 사용자도 폭넓게 확대되고 있다. 시간과 공간의 제약이 없는 여러 가지 장점에도 불구하고 웹 서비스는 트랜잭션 관리와 보안에 취약하므로 신뢰성과 안전성을 보장하는 안전한 통신이 필요하다. 그러나 비즈니스의 안전한 웹 서비스를 위한 메시지 암호화 작업은 서버의 성능에는 부정적인 영향을 준다.

그러므로 본 논문에서는 보안 요구 사항을 만족하면서 동시에 웹 서비스의 장점인 통합과 커뮤니케이션 지원을 위하

※ 본 연구는 지식경제부 지역혁신센터 사업인 민군겸용 보안공학 연구센터 지원으로 수행되었음.
† 정 회 원 : 한남대학교 멀티미디어학부 강의전담 교수
†† 정 회 원 : 충북대학교 대학원 전자계산학 이학박사(교신저자)
††† 종신회원 : 한남대학교 멀티미디어학부 교수
논문접수 : 2008년 5월 19일
수 정 일 : 2008년 7월 22일
심사완료 : 2008년 8월 13일

여 효과적인 방법으로 서버의 과부하를 개선하고, 웹 서버 아키텍처의 신뢰성과 안전성 향상을 위해 XML 문서에 대한 정보 보호 그리고 향상된 성능과 지연 처리에 필요한 기술을 제공하는 모바일 웹 서버를 제안한다. 그리고 세션 정보 관리를 결합하여 기존 시스템(Tomcat 5.5에서 구현)과 제안한 모바일 웹 서버 아키텍처를 평가한다.

본 논문은 다음과 같이 구성한다. 2장은 웹 서비스, HTTP/S 프로토콜 그리고 웹 서비스 보안에 대하여 기술한다. 3장에서는 본 논문에서 제안한 모바일 웹 서비스 시스템인 SMWSS의 설계에 대하여 기술하고, 4장은 제안한 웹 서버 시스템의 실행 환경과 테스트 결과를 기존 시스템과 비교 평가한다. 마지막으로 5장에서 결론과 본 논문과 연관된 향후 연구에 대하여 기술한다.

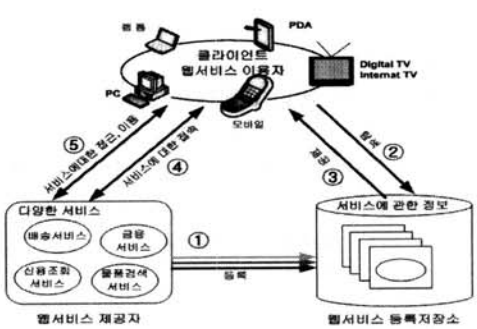
2. 관련 연구

2.1 웹 서비스의 개요

웹 서비스는 XML(eXtensible Markup Language) 표준에 의한 인터넷(HTTP) 기반의 자료 교환에 대한 컴포넌트이며, 서비스 교환, 기술(description), 등록(registration)과 발견(discovery) 단계로 구성한다. 다양한 하드웨어와 소프트웨어 사이의 자원 공유가 가능한 개방형 표준인 XML과 인터넷 프로토콜을 결합한 새로운 패러다임에 의한 분산 컴퓨팅 기술로 인터넷 접근이 용이하고, 서비스 제공자(Service Provider), 서비스 저장소(Service Registry), 그리고 서비스 요청자(Service Requester) 등의 결합으로 구현한다[2].

현재 대부분의 웹 서버에서 사용되는 아파치는 멀티쓰레딩 기능의 추가로 확장성과 유연성을 위하여 프로세스와 쓰레드의 혼합 실행이 가능하다[4]. 그리고 요청 처리를 위해 쓰레드의 실시간 생성으로 인하여 다수의 동시 접속에 정상적인 동작이 불가능하다. 따라서 기존의 다양한 연구 결과는 [5]에서 나타나는 것처럼 아파치 웹 서비스의 성능 결과는 다른 웹 서비스와 비교하면 하락함을 나타낸다.

웹 서비스의 특징은 독립성, 개방성, 확장성, 유연성을 가지며 상호운용성을 위하여 UDDI(Universal Description, Discovery, and Integration), WSDL(Web Services Description Language), SOAP(Simple Object Access Protocol), HTTP와 개방형 표준에 근거하고 텍스트 기반의 XML을 사용하여 확장성과 유연성이 높고 서로 다른 시스템과 상호 연동이 단순하고 개발이 용이하다. 반면, 웹 서비스의 문제점은 트랜잭션 관리와 보안에 취약하다. 또한 톰캣 서블릿 엔진의 사용으로 추가적인 포트의 필요에 의해 서버의 보안과 관리에 부가적인 요소가 발생한다. 이러한 단점 보안을 위해 웹 서버에 부가적인 웹 서비스 모듈을 직접 구현/추가하는 작업이 필요하다.



(그림 1) 웹 서비스 작동 과정에 대한 개념도

웹 서비스 보안을 위한 표준은 SSL(Secure Socket Layer) 프로토콜이며 HTTPS(Secure HTTP) 프로토콜을 위해 웹(HTTP)에 부가적으로 사용된다. SSL 프로토콜은 인터넷 통신의 두 어플리케이션 사이에 프라이버시와 신뢰성을 제공한다. SSL 프로토콜은 공개키와 비밀키 암호 기법을 결합하여 두 컴퓨터 사이의 메시지를 암호화한다. 메시지 암호화 작업은 서버의 처리량을 감소시킬 뿐만 아니라 평균 응답 시간을 증가시키면서, CPU 작업을 낭비하기 때문에 웹 어플리케이션 서버의 성능에 부정적인 영향을 나타낸다[3].

2.2 웹 서비스 보안

웹 서비스의 상호운용성에 의한 접근 용이성은 웹 서비스 보안의 중요성을 더욱 부각시킨다. 웹 서비스의 보안은 통신 프로토콜의 전송 계층, 메시지(SOAP 또는 XML) 그리고 서비스 수준의 보안으로 구분한다. 웹 서비스를 위한 통신 프로토콜은 HTTP, HTTPS, SMTP((Simple Mail Transfer Protocol), FTP와 같은 인터넷 프로토콜들을 사용한다.

HTTP 기본 권한은 평문의 패스워드와 신원 정보에 의해 접근을 제어하는 간단한 인증 프로토콜이므로 암호화 작업과 전송 계층의 보안을 결합하여 강한 보안을 제공하는 프로토콜이 필요하다. HTTP 인증 프로토콜은 방화벽 충돌의 문제가 없지만 암호화 과정 결핍으로 공격에 취약하므로 추가적인 보안 절차가 필요하다.

HTTPS 프로토콜은 HTTP 인증 단점의 해결 방법으로 SSL(Secure Socket Layer)을 사용한다. HTTP 어플리케이션 계층의 서브 계층에 SSL을 추가하므로 클라이언트와 웹 서버는 보안 채널을 통해 트랜잭션의 프라이버시 통신이 가능하며, 메시지 송수신을 위한 모든 트래픽에 공개키와 비밀키에 의한 다양한 알고리즘으로 암호화 작업을 수행하고, 디지털 증명서(X.509)와 결합하여 상호 인증이 가능하다[6].

SSL 프로토콜은 SSL 레코드 프로토콜과 서버와 클라이언트의 인증과 암호 알고리즘, 키를 분배하는 SSL handshake 프로토콜로 구분된다. SSL 프로토콜은 데이터 암호화와 압축으로 안전한 전송을 담당하고 SSL handshake 과정으로 서버와 클라이언트가 공개키에 의한 상호 인증과 빠른 암호화, 복호화를 위한 대칭적인 키를 생성한다. SSL의 문제점은 대용량의 인증 자료 요구로 트랜잭션에 영향을 미치며 트랜잭션 각 단계의 수행이 불가능한 경우도 있다. 그리고 SSL 암호화는 SSL 연결을 위한 계산 시간 증가로 전송 속도가 감소하여 웹 서비스 성능에 영향을 준다.

(그림 2)는 SSL handshake 프로토콜 사용으로 클라이언트와 서버가 데이터 전송 이전의 인증과 암호 알고리즘과 키를 교환하는 과정을 나타낸다.

XML 암호화는 문서의 기밀성을 제공하며, 기존의 암호화

(그림 2) SSL handshake의 키 교환 과정

표준에 의해 문서 전체 또는 일부분의 암호화 작업이 가능하다[7]. XML 암호화는 암호화한 결과를 XML을 사용하여 표현한다. 즉, 암호화 결과가 XML 형태의 암호화된 데이터이다. 따라서 XML의 장점을 유지하며 암호화 기능을 이용한다. 또한, 다양한 암호화 알고리즘에 의해서 어플리케이션 변경없이 XML 기반의 암호화가 가능한 장점을 가진다.

그러나 XML 문서의 특성에 의한 문서 구조를 분석하여 유효성 검사를 수행하므로 일반 문서의 암호화보다는 시간이 많이 소요되지만 구조적인 문서, 부분적 암호화 등 일반 문서보다 많은 장점과 비즈니스 어플리케이션에서 XML의 이용 증가와, 지속적인 성능 개선으로 XML 문서의 암호화 시간도 많이 개선되고 있다.

2.3 타원 곡선 암호(ECC) 알고리즘

현재 대부분의 시스템은 소인수 분해의 난이도에 의한 RSA를 사용하고 1024비트 이상이므로 스마트 카드나 모바일 환경에는 적합하지 않다. ECC 알고리즘은 기존의 암호 알고리즘인 RSA, Diffe-Hellman, DAS, ElGamal보다 적은 비트(최소 160비트 정도)로 대등한 보안 수준을 보장하며, ECC와 RSA의 키 사이즈 비율(1:7)이 커질수록 동등한 안정성 유지를 위한 두 알고리즘의 키 사이즈 격차는 더욱 커진다. ECC 연산의 대부분이 덧셈으로 암호화와 복호화의 수행 시간도 짧고 H/W와 S/W 구현이 쉬운 장점을 갖는다.

유한체 $GF(2^m)$의 ECC 알고리즘은 GF(p)의 ECC보다 계산 속도가 빠르고 효율적이고 유한체인 $GF(2^m)$상에서의 스칼라 곱셈(scalar multiplication) 알고리즘 구현의 용이성 때문에 많이 사용한다. 하드웨어 설계를 위하여 유한체 기반의 알고리즘을 사용하는 경우 덧셈 연산은 XOR 연산과 동일하여 반올림에 따른 오차가 없고 설계가 용이하며 빠른 연산이 가능하다[8,9].

모든 ECC 응용 시스템에서 요구하는 핵심 연산은 스칼라 곱셈으로 Q=kP를 계산한다. 소수체 $GF(2^m)$상에서의 타원 곡선은 $y^2+xy=x^3+ax^2+b$의 형태이며 (x, y)의 포인트로 구성된다. 여기서 a, $b \in GF(2^m)$이며 $4a^3+27b^2 \neq 0 (\mathrm{mod}\ p)$이다. 효율적인 스칼라 곱셈의 연산을 위하여 가장 일반적인 shift and add method, Double and Add 등의 알고리즘을 사용한다[10,11].

3. SMWSS 프레임워크의 설계 및 구현

3.1 제안 시스템의 설계

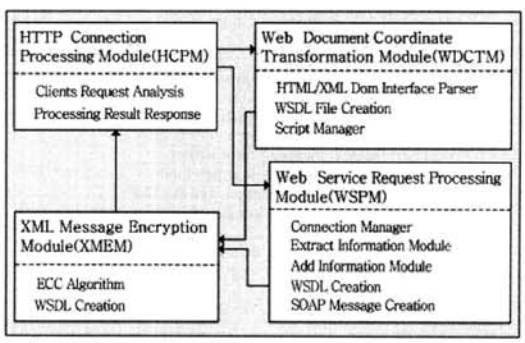
본 논문에서 제안한 모바일 웹 서비스 시스템 SMWSS (Secure Mobile Web Service System)은 웹 서비스의 접속 시간 단축, 지연 처리 향상 그리고 보안을 위하여 웹 서버에 부가적인 웹 서비스 모듈을 직접 구현한다. 웹 서비스 구현을 위해서 표준의 WSDL 문서 사용과 SOAP 메시지를 사용하고, AXIS 모듈을 그대로 사용하면서 톰캣의 오버헤드 감소를 위하여 직접 SOAP 요구와 응답 메시지를 처리하는 시스템을 설계/구현하는 방법을 선택한다.

SMWSS의 구현을 위해 클라이언트가 웹 서비스를 요청하는 SOAP 메시지에서 필요한 정보를 추출하는 기능, 웹 서비스의 처리 결과를 SOAP 메시지로 생성하여 클라이언트로 전달하는 기능 그리고 웹 서비스를 위한 WSDL 생성 기능, 그리고 웹 서비스의 성능 향상을 위한 추가 모듈과 보안을 위한 메시지 암호화 처리 모듈이 필요하다. 따라서 SMWSS 구성 요소로 다음과 같은 모듈을 서버에 직접 구현한다.

- SOAP 요구 메시지 처리기(SOAP Request Message Processor)
- WSDL 전처리기(WSDL Preprocessor)
- 메시지 암호화 처리기(Message Encryption Processor)

제안 시스템(그림 3)이 전형적인 표준 시스템과 가장 중요한 차이는 톰캣 서블릿 엔진을 포함하지 않는다. 서블릿 엔진 대체를 위해서 WSDL 전처리기를 사용하여 WSDL 파일은 WSDL 전처리기에 의해 직접 생성하고 SOAP 요구 메시지는 SOAP 요구 메시지 처리기에서 처리한다.

본 논문에서 제안한 SMWSS 모바일 웹 서비스 시스템은 (그림 3)과 같다. 본 논문에서 제안한 웹 문서 변환기는 클라이언트에서 사용하는 WSDL 파일을 자동으로 생성하기 위하여 HTML/XML 파서를 이용하여 요청된 메시지를 WSDL로 변환한다. org.w3c.dom 라이브러리를 이용하고 AXIS에 독립적으로 실행하며, Dom의 Document 객체를 사용한다. 웹 문서 변환기는 웹 서비스 접근과 실행에 필요한 메시지 규격, 프로토콜, 웹 서비스의 위치 정보 등 웹 서비스가 지원하는 기능을 수행한다. WSDL의 내용은 서비스에 대한 내용, 서비스에 대한 접근 방법, 서비스의 위치 등을 포함한다.



(그림 3) SMWSS 모바일 웹 서비스 시스템의 구조

3.2 웹 서비스 요구 메시지 처리기

SOAP 메시지 처리기는 클라이언트의 SOAP 메시지 형태의 요구를 분석하는 모듈과 SMWSS의 수신 결과를 규격화된 SOAP 메시지로 구현하는 모듈이 필요하다. 따라서 SOAP 메시지 처리기는 웹 서비스 요청을 수신하는 접속부와 SOAP 메시지 분석기 그리고 SOAP 메시지 생성기로 구성한다. (그림 4)는 클라이언트가 서버에 요청하는 경우의 처리 과정을 나타낸다.

SOAP 메시지 분석기는 요청 서비스명, 입력값, 세션아이디가 포함된 클라이언트의 SOAP 형태의 XML 문서인 SOAP 요청 메시지를 접속부가 SOAP 메시지 분석기로 전송하면 클라이언트의 요청 형태를 분석한다. 요청 서비스의 WSDL명을 추출한 후 수신 SOAP 메시지의 분석 정보에서 서비스명과 입력값의 추출과 정보를 생성하여 서비스 처리부에 서비스 처리를 요청한다.

서비스 처리부의 처리 결과를 SOAP 메시지 생성부로 전송하면 응답 SOAP 메시지 생성을 위하여 응답 헤더 정보와 응답 SOAP 메시지의 정보를 생성하고, 정상적인 서비스 처리 결과는 서비스 결과값과 세션 아이디를 추가하고, 비정상인 경우는 실패 정보를 추가하여 SOAP 메시지를 생성하고 접속부로 전송하면, 접속부는 다시 클라이언트로 전송한다.

3.3 타원곡선에 의한 XML 메시지 암호화 처리기 설계

본 논문의 XML 메시지 암호화 연산은 XML 문서 중 일부 보안이 필요한 부분에 ECC 암호 알고리즘을 이용한 암호화 방식을 사용한다. ECC를 적용한 XML 메시지는 웹 서비스의 URI(Uniform Resource Identifier)에서 서비스 데이터의 추출과 SOAP 메시지를 생성한다. XML 메시지는 SOAP 메시지와 ECC 알고리즘의 시그너처를 이용하여 메시지를 암호화하여 웹 서비스를 보호한다. 원본 URI의 서비스 데이터는 HTTP 프로토콜을 통해 접근 가능한 원격의 웹 서비스를 위한 파일로 사용한다. ECC 암호화 알고리즘을 이용한 XML 암호화 작업이 기존 암호화 처리와 차이점은 XML 문서의 구조적 특징에 의해 보안이 필요한 부분만을 암호화하면서 최종 암호화 처리 결과가 이진 코드가 아닌 XML 메시지 형태를 따른다. SOAP와 XML 보안사이의 관계는 (그림 5)와 같다.

(그림 5)와 같은 XML 메시지 암호화 처리를 위하여 WSDL에서 생성된 정보를 암호화한 후 암호화된 문서는 SOAP로 전달되고 ECC 시그너처와 함께 전체를 암호화하여 XML 서비스를 보호한다. 각 항목에 대해 각기 다른 키가 사용되며, 이러한 처리 과정은 메시지 전체가 아닌 일부 데이터를 위한 키를 수신한다.

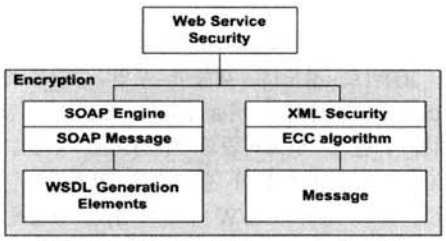
XML 메시지에 대한 ECC 암호화 과정에 대한 알고리즘은 (그림 6)과 같다. 클라이언트의 요구에 대한 웹 서비스 데이터를 추출하고 SOAP 메시지 처리기가 SOAP 메시지를 생성하면 암호화된 XML 형식의 메시지를 생성하는 암호화 알고리즘을 적용한다.

메시지 암호화 모듈은 XML 형식의 메시지의 문서 구조와 비교하여 유효한(well-formed) XML 문서인 경우는 XML 문서에서 암호화 작업을 수행하는 부분적인 element들을 검출하고, 비밀키 암호 알고리즘을 사용하여 암호화한다. 암호화의 비밀키는 해당 그룹에 대한 세션키이다. 그리고 요소들의 암호화를 완료하면 XML 메시지에서 요소들과 교체를 위해 인코딩한다. 마지막으로 암호화한 내용을 복호화할 때 필요한 정보를 암호화한 내용과 같이 추가한다. 이 과정을 통해 보안이 필요한 부분만 암호화 XML 메시지로 작성한다.

XML 메시지의 복호화 과정은 암호화 과정과 유사하다. (그림 7)은 XML 메시지 복호화 모듈의 프로세스 순서를 나타낸다. 메시지 복호 모듈은 XML 메시지가 유효한 XML 문서 구조 여부를 검사하여 유효한 문서인 경우는 XML 메

```
Procedure Server()
{  Procedure Connection()
    {   connection_acceptance
    analysis of request a kind;
    If (kind of request = request web service) then
          {call SOAP_Message_Analysis;
            Wait for result return SOAP message;
            transmission transaction result to Client;  }
    }
    Procedure SOAP_Message_Aalysis()
    {   request analysis;
    an extract of WSDL name;
    creation of SOAP message information
    an extract of service name;
    an extract of input value
    request of service transaction to SOAP_Message_Creation;
    call SOAP_Message_Creation();  }
    Procedure SOAP_Message_Creation()
    {   creation of response header information
    creation of SOAP message information
       If (transaction result = normal) then
        {      an addition of service result value
          an addition of session ID;  }
        else
              an addition of a failure information
    creation of a result SOAP message;
    call Connection()
       transmissiontransaction result to Connection  }
}
```

(그림 4) 클라이언트의 웹 서비스 요청 처리 과정



(그림 5) SOAP과 XML Security의 관계

```
Procedure Encrytion()
{    Accept XML message;
 Parsing well-formed XML message;
 Whlie(parsing result valid) then
    {   Extract the element to be encryted
        If (extract result = exit) then
        {
        Encrypt using private key;
        Encode;
        Create Encryptiom Information element;
        Replace the element;
        }
      else {
             encrypted XML message;
             end;
           }
 }
 Exception error;
}
```

(그림 6) XML 메시지의 암호화 과정

```
Procedure Decrytion()
{  Accept encryted XML message;
Parsing well-formed XML message;
Whlie(parsing result valid) then
   {  Extract the element to be decryted
      If (extract result = exit) then
      {
      Decode;
      Decrypt
      Delete Encryptiom Information element;
      Replace the element;
    }
    else
    {
       Decrypted XML message;
       end;
    }
}
Exception error;
}
```

(그림 7) XML 메시지의 복호화 과정

시지에서 복호화할 element들을 검출하고, 검출한 element들을 디코딩한다. 인코딩 XML 메시지에서 element의 복호화를 위해 세션키를 사용한다. 복호화를 완성하면 암호화한 element와 관련된 정보들은 복호화한 element로 대체한다.

4. 실험 환경 및 성능 평가

4.1 실험 환경 및 방법

본 논문에서는 웹 서비스를 위한 어플리케이션 서버의 평가를 위하여 3가지 다른 웹 서비스를 구현한다. 전통적인 오리지널 서버(Tomcat 5.5)와 본 논문의 제안 시스템(Tomcat 5.5 제거)을 각각 구현하여 테스트를 실행했다. 웹 서비스 엔진은 2GB RAM, Intel Xeon 3.0GHz Windows 2000 서버와 2GB RAM, XEON 2.4GHz의 클라이언트와 Java SDK 1.5 그리고 Tomcat 5.5와 AXIS을 이용하여 개발하고 테스트한다.

클라이언트의 실행을 위한 실험 환경은 먼저 시스템에 접근하는 사용자는 동시에 200명으로 가정하고, 4대의 중계기를 준비하여 각각의 중계기는 클라이언트 시뮬레이션 프로그램에 의해 최대 50개의 클라이언트를 쓰레드를 생성하고 여러 개의 SOAP 요구 메시지를 1초에서 10초 간격으로 반복 요청한다.

실험을 위한 콘텐츠 서버는 다음사이트와 야후사이트(http://www.daum.net(dictionary, http://kr.yahoo.com (stock))이고, 각각의 서버의 타임 아웃은 30초이다.

본 논문에서 제안한 SMWSS 시스템의 성능 평가를 위해 실험에 사용된 3개의 웹서비스 시스템은 다음과 같다.

- 전통적인 기존 방식을 적용한 시스템(AXIS와 Tomcat 5.5 사용)
- 기존 방식을 개선한 시스템(Tomcat 5.5 제거, AXIS 적용, ECC 미적용)
- SMWSS 시스템(Tomcat 5.5 제거, AXIS, ECC 적용)

4.2 결 과

본 논문의 실험은 서버 성능의 테스트를 위하여 다수의 문자를 전송한다. 다수의 문자 전송은 초기 TCP, HTTP 연결을 설정하고 SOAP 메시지로 문자를 전송한다. <표 1>은 실험에 사용된 각각의 시스템에 대한 실험 시간과 실험 시간에 대한 요청의 개수를 나타낸다. 본 논문의 실험은 모바일 웹 서비스 시스템의 실행 평가에 초점을 맞췄다.

<표 2>는 ECC 알고리즘의 성능 측정을 위하여 ECC와 RSA 알고리즘의 처리 시간 비교를 위하여 각각의 모듈을 구현하여 실험을 실시하였다.

<표 2>의 결과는 ECC 암호화 모듈이 RSA 암호화 모듈보다 컴퓨팅 오버헤드를 적게 요구하는 것을 의미한다. 따라서 ECC가 더욱 효율적이고 메모리 사용도 적게 요구함을 나타낸다. 그러므로 시스템 성능 향상은 전체 시스템의 성능 향상과 누적되는 연결 신호 처리의 향상을 의미한다.

제안 시스템의 실험 시간이 표준 시스템의 실험 시간보다 짧지만 전체 요청수는 표준 시스템보다 많고 타임아웃 수는 표준 시스템보다 적었다. 제안된 시스템에서의 초당 평균 요구 개수는 17.94인 반면에 표준 시스템에서는 9.64이다. 표준 시스템에서는 많은 연결 에러가 발생한다.

(그림 8)은 RSA와 ECC 암호모듈을 표준 시스템과 제안 시스템에 적용한 후 클라이언트의 응답 지연 시간을 측정한 결과이다. (그림 8)의 제안 시스템은 ECC의 적용 유·무에 따라 개선 시스템과 제안 시스템으로 평가하였다. 제안 시스템에 ECC을 적용 유·무에 따른 응답 지연 시간의 차이는 전체 시스템의 동작 처리시간으로 보면 크게 영향을 미치지 않는 수치이다. (그림 8)의 결과에서 보면 개선 시스템과 제안 시스템이 표준 시스템보다 클라이언트의 응답 지연시간이 선형적으로 증가하였다. 이 같은 결과는 문맥 전환 비용의 감소에 의한 결과를 의미한다.

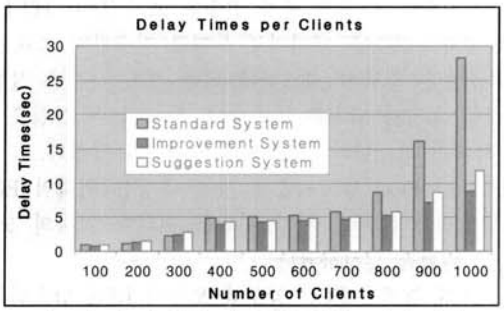
(그림 9)는 표준 시스템과 제안 시스템의 동작과정 중 클라이언트의 초당 요청처리 개수를 나타낸다. 각각의 시스템의 요청 처리 개수는 개선 시스템과 제안 시스템에서 일정한 수준을 유지하지만 표준 시스템에서는 증가하는 클라이언트에 대하여 성능은 반대로 감소한다. 이와 같은 결과는 <표 2>에서 정의한 ECC와 RSA의 암호와, 복호화 그리고 키 생성의 처리시간 차이로 인해 발생된다. ECC를 사용할 경우 RSA보다 키 생성에 많은 처리시간이 필요하지만 암호화와 복호화의 처리시간은 RSA보다 평균 2~5% 적게 소요된다. (그림 9)의 결과처럼 클라이언트의 초당 요청처리 개

〈표 1〉 각 실험 시스템에 대한 실험 시간과 요구 횟수

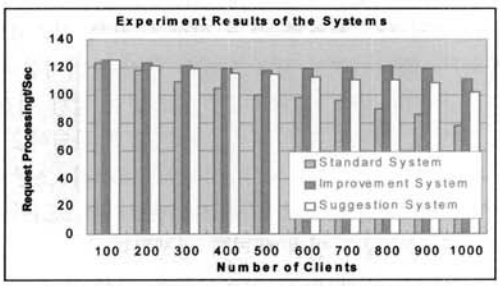
테스트 정보	표준 시스템	개선 시스템	제안 시스템
실험 시간	46,200	29,400	33,400
총 요구 횟수	445,422	524,573	504,625

〈표 2〉 ECC와 RSA의 처리 시간 비교

Error Type	ECC	RSA1024
Encryption	43.244	101.297
Dncryption	46.828	241.526
Key generation	110.34	7.578



(그림 8) 각 실험 시스템에 대한 지연 시간



(그림 9) 클라이언트의 초당 요청처리 개수

수는 RSA를 사용하는 표준 시스템보다 ECC를 사용한 제안 시스템의 초당 요청처리 개수가 높게 나타난다.

5. 결 론

본 논문은 확장성과 유연성을 갖춘 웹 서비스 보안 기술 연구를 통해 비즈니스 어플리케이션의 정보 보호 문제의 해결을 제시하고, 안전한 웹 서비스 환경에 알맞은 모바일 웹서비스 아키텍처를 제안하고 구현하였다.

실험 결과 모바일 환경에서 제안 시스템은 전통적인 아파치 웹 서비스보다 6.7%, 개선된 웹 서비스보다 9.7%의 성능 향상을 얻었다. 또한 동시 요청이 50개 이상의 경우 접속 성능은 약 38% 향상 되었다. 따라서 실험 결과는 접근의 수행을 처리하는 SOAP 요구에서 표준 웹서비스 구현보다 성능이 향상되었음을 증명하였다. 본 연구는 매우 작은 연결 오류를 가지고 있으므로 전형적인 웹서비스 구현보다 ECC 보안 모듈을 추가하여도 효율적인 서비스 요구 처리가 가능하다. 향후 연구는 I/O 시간을 감소시키며 개선된 시스템과의 성능 격차를 감소시키는 알고리즘에 관한 연구가 필요하다.

참 고 문 헌

[1] J. Guitart, V. Beltran, D. Carrera, J. Torres, and E. Ayguad'e. "Characterizing secure dynamic web applications scalability," In 19th International Parallel and Distributed Processing Symposium, Denver, Colorado(USA). April 4-8, 2005.

[2] 김용태, 박길철, 김석수, 이상호, "웹 서비스의 서버 구조 단순화를 통한 웹2.0 웹서비스 성능 향상," 정보처리학회논문지D, Vol.14-D No.04. pp.421-426, 2007.

[3] V. Beltran, D. Carrera, J. Guitart, J. Torres, and E. Ayguad'e, "A Hybrid Web Server Architecture for Secure e-Business Web Applications," LECTURE NOTES IN COMPUTER SCIENCE(LNCS), Springer Berlin/Heidelberg, pp.366-377, 2005.

[4] Jakarta Project, Apache Software Foundation, Tomcat, http://jakarta.apache.org/tomcat.

[5] D. Davis and M. Parashar, "Latency Performance of SOAP Implementations," Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pp.407-412, 2002.

[6] 정 지훈, 웹 서비스, 23장 웹 서비스의 미결과제3, 한빛미디어, 2007.

[7] 최동희, 박석, "접근제어 정책구현을 위한 역할 기반 XML 암호화," 정보보호학회 논문지 Vol.15 No.1, pp.3-15, 2005.

[8] 김 의석, 정용진, "컨텐츠 보호를 위한 DTCP용 타원곡선 암호(ECC) 연산기의 구현," 한국통신학회논문지, pp.176-184, Vol.30 No.3C, 2005.3.

[9] Ying Liu Yeap, T.H. O'Brien, W., "Securing XML Web Services with Elliptic Curve Cryptography," Electrical and Computer Engineering, 2007. CCECE 2007, pp.974-977, 2007.

[10] J. Kim, Y. Kim and Y. Jeong, "Implementation of a pipelined Scalar Multiplier using Extended Euclid Algorithm for Elliptic Curve Cryptography(ECC)," 한국정보보호학회, pp.17-30, Oct. 2001.

[11] Chen, Zhiqun, Java Card Technology for Smart Cards: Architecture and Programmer's Guide, pp.129-148, Addison-wesley, 2002.

김 용 태

e-mail : ky7762@hannam.ac.kr
1988년 숭실대학교 전자계산학과(석사)
2008년 충북대학교 전산학과(이학박사)
2002년~2006년 가림정보기술 이사
2006년 3월~현 재 한남대학교 멀티미디어학부 강의전담 교수
관심분야 : 모바일 웹서비스, 정보보안, 센서 웹, 모바일 통신보안, 멀티미디어

정 윤 수

e-mail : bukmunro@gmail.com
2000년 2월 충북대학교 대학원 전자계산학(이학석사)
2008년 2월 충북대학교 대학원 전자계산학(이학박사)
관심분야 : 센서 보안, 암호이론, 정보보호, Network Security, 이동통신보안

박 길 철

e-mail : gcpark@hannam.ac.kr
1986년 숭실대학교 전자계산학과(석사)
1998년 성균관대학교 전자계산학과(박사)
2006년 UTAS, Australia 교환교수
1998년 8월~현 재 한남대학교 멀티미디어학부 교수
관심분야 : multimedia and mobile communication, network security