초경량 환경의 보안 서비스 지원을 위한 보안 API

김원영* · 이영석* · 이재완* · 서창호**

Security APIs for Security Services in Ultra Light-weight Environment

Won-young Kim* · Young-seok Lee* · Jae-wan Lee* · Chang-ho Seo**

이 논문은 산업자원부의 지역혁신인력양성사업의 연구결과로 수행되었음

이 논문은 2007년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임
(No. R01-2007-000-20291-0)

요 약

유비쿼터스 환경에서의 컴퓨팅 기기들은 초경량 컴퓨팅 환경으로서 사용자들이 컴퓨터의 존재를 인지할 수 없도록 사용자 신체나 주변 환경에 내장되며, 크기가 작고, 적은 기능을 가진 많은 컴퓨터를 여러 곳에 분포하여 네트워크로 통신한다. 초경량 컴퓨팅 환경에서는 사용자들에게 서비스를 제공하기 위해 사용자의 정보를 교환하는 일이 많으며, 사용자의 정보를 보호하기 위해서는 보안 기술이 반드시 포함되어야 한다. 본 논문에서는 초경량 컴퓨팅 환경에서 표준화 된 보안 서비스를 제공하는 초경량 환경을 위한 보안 서비스 지원을 위한 API를 연구하고 설계한다. 초경량 환경의 보안 서비스 지원을 위한 API는 일반적인 컴퓨팅 환경에서와 같이 데이터 암호화, 데이터 인증, 키 관리 등의 보안 서비스를 제공하며, 초경량 컴퓨팅 환경에 맞는 RC5와 SHA1 알고리즘 사용, 효율적인 메모리 관리를 위해 각 서비스를 라이브러리 화하여 필요시 마다 라이브러리를 할당/해제 하는 등의 초경량 컴퓨팅 환경의 문제점을 해결할 수 있도록 설계, 구현한다.

ABSTRACT

Computers used for light-weight computing environments are considerably limited in resources and performance running in ubiquitous environment. Because of the limited resources, it is difficult to apply existing security technologies to the light-weight computers. In this paper, light-weight security software is implemented using RC-5 encryption and SHA-1 authentication algorithm which is appropriate for light-weight computing environments. The design of components based on security software of a light-weight computer application and the test-bed for security software are presented. The simulation verifies the correctness of the security software. The architecture of the light-weight and reconfigurable security software for light-weight computer applications is proposed. The proposed security software is small size and provides reconfigurable security library based on the light-weight component and the software manager that configures software platform is loaded with the library at the time it is needed.

키워드

Ultra Light-weight computing, Security Service, API

* 국립군산대학교 전자정보공학부
** 국립공주대학교 바이오정보학과

접수일자 : 2008. 1. 30

Ⅰ. 서 론

반도체와 인터넷의 발달로 인해 환경과 컴퓨터가 하나로 융합이 되어 언제 어디서나 사용자에게 필요한 서비스를 제공하는 유비쿼터스(Ubiquitous) 컴퓨팅 환경의 시대가 다가오고 있다. 유비쿼터스 컴퓨팅 환경에서의 컴퓨터들은 초경량 컴퓨터로써 사용자들이 인지할 수 없도록 사용자 신체 및 주변 환경에 내장되며, 사용자의 물리적 위치나 시간에 상관없이 동일한 컴퓨팅 환경을 제공하고 정보를 이용할 수 있어야 한다. 이와 같은 초경량 컴퓨팅 환경에서의 컴퓨터는 휴대가 용이하거나 외부에 보이지 않아야 하기 때문에 크기가 작아야 하며, 적은 기능을 하는 많은 컴퓨터가 여러 곳에 분포하여 유·무선을 이용하여 통신하기 때문에 이를 일일이 관리하기 어렵다. 때문에 한 번의 설치 이후 이를 오랜 기간 사용하여야 하며 이를 위해서는 연산, 메모리, 통신, 전력 등 컴퓨터의 모든 자원이 제한적이게 된다.

초경량 환경에서의 보안 서비스는 매우 중요한 사항이 된다. 사용자에게 맞는 서비스를 제공하기 위해서는 사용자 정보에 대한 컴퓨터들 간의 통신이 반드시 필요하며, 컴퓨터들 간 통신에서 유선을 사용할 수도 있지만 대부분의 컴퓨터들은 무선을 사용하여 통신하기 때문에 이를 보호하기 위한 방법이 반드시 필요하다.

따라서 초경량 컴퓨팅 환경에 사용되는 보안 서비스는 초경량 컴퓨팅 환경에 맞게 설계되어야 한다. 가능한 적은 메모리를 사용하고 배터리 소모가 적으며, 보안 서비스를 제공하기 위한 시스템의 연산 양이 적어야 한다. 또한 시스템 자원의 절약을 위해 기능에 따라 라이브러리를 달리 함으로써 필요한 기능만을 동적으로 할당하고 사용 후에 해제하는 등의 메모리를 효율적으로 사용하는 초경량 환경만의 보안 서비스 지원을 위한 방법이 필요하다.[1]

본 논문에서는 초경량 환경의 보안 서비스를 위한 보안 소프트웨어를 설계하고 구현한다. 구현된 보안 소프트웨어는 암호화, 인증 등의 보안의 기본 서비스를 제공하는 한편, 초경량 환경에서도 안정적으로 운영될 수 있도록 초경량 환경에 적합한 보안 알고리즘과 자원 제약적인 환경에서 메모리를 효율적으로 사용하도록 각 기능을 컴포넌트 화하여 구현한다.

2장에서는 초경량 컴퓨팅 환경의 보안 요구사항에 대해 살펴본다. 3장에서는 2장의 내용을 토대로 초경량 환경의 보안 서비스 지원을 위한 보안 소프트웨어를 설계하고, 4장에서 구현 결과를 실험하고, 5장에서는 결론과 향후 연구 과제에 대해 논의한다.

Ⅱ. 초경량 환경에서의 보안 요구사항

2.1. 기밀성

기밀성은 허가되지 않은 사용자로부터 데이터를 보호하는 것을 말한다. 중요한 데이터의 경우 허가되지 않은 사람이 데이터를 읽게 될 경우 이를 악의적인 목적으로 사용할 수 있기 때문에 이를 보호하기 위해 암호화라는 방식을 사용한다. 암호화를 사용하는 사람은 키와 암호화 알고리즘을 사용하여 데이터를 암호화 하며, 이 암호화 알고리즘과 암호문의 키를 가진 사람만이 암호를 해독하여 데이터를 볼 수 있다. 스푸핑과 스니핑의 문제를 해결한다.[2]

2.2. 인증

인증은 크게 두 가지로 구분될 수 있다. 내가 데이터를 주고받는 상대가 정당한 사용자인지 확인하기 위한 사용자 인증과 데이터의 무결성. 즉, 주고받는 데이터가 원래 송신자가 보낸 것인지 확인하는 데이터 인증이 있다. 사용자 인증의 경우 보통 데이터를 주고받기 전에 확인하게 되며, 사용자 인증이 완료되고 서로 간 데이터를 주고 받을 때 전송 중에 데이터가 위조, 변조 되었는지 확인하기 위해 데이터 인증을 사용하게 된다.

사용자 인증은 ID나 비밀번호와 같이 미리 저장된 값을 사용자가 입력하여 맞는지 비교하는 방식을 사용하며, 데이터 인증의 경우에는 메시지에 해시 함수 등과 같은 인증 알고리즘을 이용하여 MAC (Message Authentication Code) 즉, 메시지 인증 코드을 만들고, 메시지와 MAC을 함께 전송한다. 수신측에서는 받은 메시지를 송신측과 같은 알고리즘을 이용하여 MAC을 만들고, 수신 받은 MAC 과 생성된 MAC을 비교하여 확인한다. MAC 생성 시 사용하는 알고리즘에 따라 키가 사용될 수 있으며, 키를 사용될 경우 좀 더 높은 보안의 강도를 가진다.

2.3 키관리

키는 크게 대칭키와 비대칭 키 두 가지로 구분되어 사용된다. 대칭키는 암호화를 하는 키와 암호문을 다시 복

호화 하는데 사용되는 키가 같은 키를 말하고, 비대칭키는 암호화에 사용되는 키와 복호화에 사용되는 키가 다른 경우를 말한다. 대칭키의 경우 키를 가진 사람은 모두 암호문을 풀 수 있기 때문에 키에 대한 보안이 필요하며 이를 비밀로 유지하기 때문에 비밀 키라고도 하며, 이와 같이 키를 관리하기 위한 기술이 필요하다. 비대칭 키를 관리하기 위한 대표적인 키 관리 기술로는 인증서를 사용하는 PKI를 들 수 있다. PKI(Public Key Infrastructure)에서 사용되는 키는 공개키와 개인키로 나누어지며, 한 쌍으로 연결되어 있고, 이 키들은 공인된 인증기관을 통해서 인증된다. 하지만, PKI는 그 연산량으로 인해 아직까지 초경량 환경에서 이용하기는 어렵다.

2.4 메모리 관리

초경량 컴퓨팅 환경에서는 제한적인 시스템 자원을 효과적으로 사용할 수 있는 방법이 필요 하다. 암호화나 인증, 키 관리 기술의 경우 사용자에 따라 암호화 기능만을 사용하는 경우와 인증 기능만을 사용하는 경우 또는 두 가지를 모두 사용하는 경우로 나눌 수 있다. 또한, 키 관리 기능을 사용할 경우에는 암호화 기능이나 인증 기능은 필요하지 않지 않기 때문에 일반적인 컴퓨팅 환경과 같이 모든 기능을 메모리에 할당하여 사용할 경우, 메모리의 낭비를 가져오게 된다. 따라서 초경량 컴퓨팅 환경에서의 보안은 이를 고려하여, 각각의 기능마다 라이브러리를 달리하여 어떠한 기능이 필요할 경우 그 기능만을 메모리에 할당하여 사용하고 기능이 사용된 후에는 반납하는 방식을 사용하여 제한적인 시스템을 효율적으로 사용하여야 한다.

Ⅲ. 초경량 환경의 보안 API 설계

일반적인 환경의 보안 요구사항들은 현재 프로그램 개발자들이 프로그램에 보안 기능을 쉽게 적용할 수 있도록 보안 소프트웨어로 개발되어 있다. 이상적으로 모든 프로그램에서 쓰일 수 있는 단 하나의 보안 소프트웨어를 만들어 적용하는 것이 가장 좋은 방법이지만, 실제로 복잡한 형태의 응용 분야를 지원해 주기 위해서는 여러 계층의 소프트웨어가 요구될 수 있다. 현재 개발된 보안 소프트웨어는 GSS-API, GCS-API, Cryptoki, CryptoAPI, CDSA 등이 있으며 본 논문에서는 표준화 된 보안 소프트웨어 중 RSA의 Cryptoki를 위주로 조사하여 설계하였다.[3]

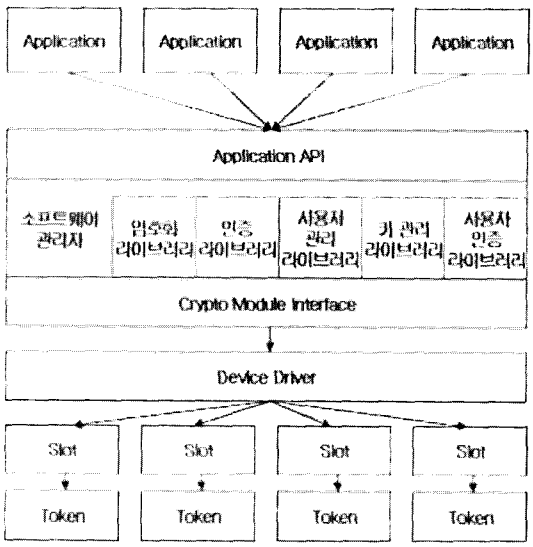

그림 1. 보안 소프트웨어의 구조
Fig. 1. Architecture of security software

그림 1은 초경량 환경의 보안 소프트웨어의 구조를 나타낸 그림으로 전체적인 보안 소프트웨어의 구조는 다수 개의 응용 프로그램과 다수 개의 암호화 모듈로 나누어져 있으며, 보안 소프트웨어는 암호모듈 안의 정보를 읽어서 응용에서 인식할 수 있도록 추상화 된 토큰(token) 정보를 만들고 이를 슬롯(slot)을 인터페이스로 사용한다. 토큰 안에는 객체(object)가 있으며 객체는 데이터, 인증서, 키로 나누어져 있으며, 이를 보안 기능 사용 시에 이용한다.

보안 소프트웨어는 데이터 기밀성을 위한 암호화 기능과 데이터 무결성을 위한 인증 기능, 키 관리를 위한 키 관리 기능, 사용자를 관리하기 위한 사용자 관리 기능과 사용자 인증을 위한 사용자 인증 기능으로 나누어져 있다, 암호화 기능, 인증 기능, 키 관리 기능 등은 사용자만이 사용할 수 있으며, 사용자 관리 기능은 보안 관리자(Security Officer)만이 사용할 수 있도록 세션 기능을 두어 이를 구분하였다. 이 기능들은 각각 필요시마다 따로 호출하여 사용할 수 있도록 라이브러리 화하였으며, 이를 관리하기 위한 소프트웨어 관리자로 나누어져 있다. 각 라이브러리의 기능은 다음과 같다.

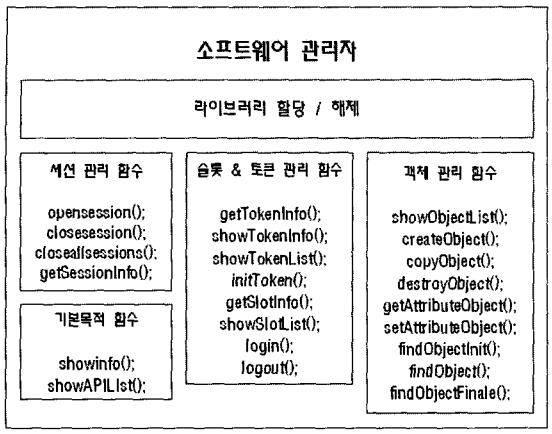

그림 2. 소프트웨어 관리자의 구조
Fig. 2. Architecture of software manager

3.1 소프트웨어 관리자

소프트웨어 관리자의 구조는 그림 2와 같으며 포함하는 API는 기본적인 목적을 위한 함수와 슬롯 및 토큰 관리를 위한 함수, 세션 관리를 위한 함수, 객체 관리를 위한 함수로 나뉘져 있으며, 라이브러리를 동적으로 할당하고 해제하는 부분 또한 이곳에서 관리한다. 소프트웨어 관리자는 사용자의 관계없이 일반적으로 사용되는 함수들을 모아 놓았기 때문에 라이브러리 화하지 않았으며, 보안 API 호출시 모두 메모리에 할당된다.

3.2 암호 라이브러리

암호화 라이브러리는 기밀성을 위해 데이터를 암호화 하는 함수와 암호화 된 데이터를 복호화하는 함수로 구분되어 있으며, 데이터 암호화 라이브러리는 일반 사용자만 사용하는 기능으로 메모리의 효율성을 위해 라이브러리 화하여 응용의 요청에 따라 할당/해제 된다. 데이터를 암호화 하는 기능과 암호화 된 데이터를 복호화하는 기능을 포함하고 있다.

3.3 인증 라이브러리

인증 라이브러리의 구조는 무결성을 위해 메시지 뒤에 덧붙여서 전송하는 MAC을 생성하는 함수와 이를 받은 MAC과 메시지로 MAC을 생성하여 나온 값을 비교하여 검증하는 함수로 나뉘어져 있으며, 응용의 요청에 따라 할당/해제 된다. 데이터를 인증하는 기능과 이를 검증하는 기능을 포함하고 있다.

3.4 키 관리 라이브러리

키 관리 라이브러리는 암호화/복호화와 인증에 사용되는 키를 초기화 하고 변경하기 위한 함수를 가지고 있으며, 키 관리 기능이 포함되어 있다.

3.5 사용자 핀 관리 라이브러리

사용자 핀 관리 라이브러리는 일반 사용자의 PIN (Personal Identification Number)을 초기화 또는 수정하는 함수로 구성되어 있으며, 핀 관리 기능이 포함되어 있다.

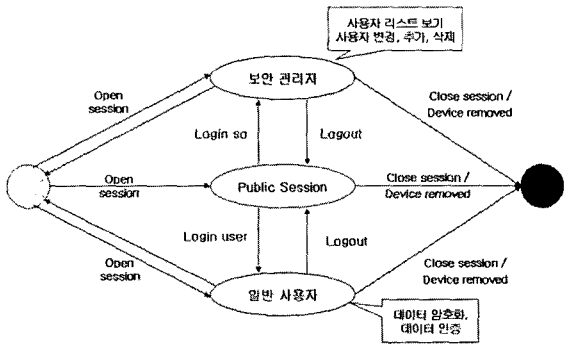

그림 3. 보안 API의 흐름도
Fig. 3. Flow of security API

보안 API의 전체적인 흐름은 그림 3과 같다. 응용에서 암호모듈을 이용하기 위해서는 적어도 하나의 세션을 열어야 하기 때문에 그림의 왼쪽에서 세션을 열게 되면 공용으로 사용되는 세션으로 가게 되고, 여기서 로그인 과정을 거치게 되는데 보안 관리자인지 일반 사용자인지 확인하게 된다. 여기까지의 과정을 위의 소프트웨어 관리자가 하게 되며, 보안 관리자는 사용자의 관한 정보를 바꾸거나 초기화 할 수 있고, 일반 사용자로 로그인 하였을 경우, 보안에 관한 기능 - 암호화, 인증, 키 관리 - 을 사용할 수 있다. 이는 평소에는 사용하지 않다가 사용자(또는 응용)의 요청에 의한 것으로 사용되기 때문에 라이브러리 화하여 메모리를 효율적으로 사용할 수 있도록 하였다.

로그인 상태인 경우에 세션이 열린 상태에서 다른 세션을 열어 작업을 할 수 있기 때문에 보안 관리자와 일반 사용자의 opensession의 상태를 그림 왼쪽과 같이 나타내었다.

3.6 암호화 및 인증 알고리즘

본 논문에 사용되는 알고리즘은 암호화에는 RC5, 인증에는 SHA-1 알고리즘을 사용하였다. RC5 알고리즘의 경우 초경량 환경과 비슷한 환경인 WSN에서 고안된 블록 암호화 알고리즘으로 마이크로프로세서에서 일반적으로 사용하는 XOR, Shift, 모듈러 연산 등의 기본적인 연산을 사용할 뿐만 아니라 메모리 요구량이 낮고, 간단한 알고리즘을 사용하여 속도가 빠르다.[4][5]

표 1. 데이터 메모리 요청
Table 1. Requirement of data memory

	skey	CBC	CFB	OFB	CTR
RC5	92	64	42	42	44
RC6	62	100	62	62	64
Rijndael	16,32	92	54	54	56
MISTY1	4	62	40	40	42
KASUMI	58	62	40	40	42
Camellia	170	148	110	110	112

표 2. 코드 메모리 요청
Table 2. Requirement of code memory

	CBC	CFB	OFB	CTR
RC5	1746	908	836	986
RC6	2576	1324	1252	1402
Rijndael	14716	12688	112616	12766
MISTY1	7132	4222	4150	4300
KASUMI	9702	5446	5374	5524
Camellia	19708	12382	12310	12460

표 1과 표 2는 WSN 환경에서 6가지 블록 암호화 알고리즘에 대한 벤치마킹 결과 중에서 메모리 요구도에 관한 내용으로 다른 알고리즘 보다 낮은 메모리를 요청함을 알 수 있다.[6]

RC5 알고리즘의 암호화 과정은 크게 키 확장 과정과 암호화 과정으로 나뉜다.[4]

키 확장은 암호화 각 과정에 사용될 수 있도록 키를 비트 수와 라운드 수에 맞게 확장하는 과정이다. 키 확장을 위해서는 첫째로 각 라운드 별로 2개의 서브키와 별도의 연산에 적용하기 위해 다음 (1)과 같이 w비트 단위의 t개의 서브키가 필요하다.

$$t=2(r+1) \tag{1}$$

서브키를 이용하여 서브 키 배열 S를 만들고, 배열을 초기화(initialize) 한다(2).

```
S[0], S[1], ···, S[t-1]    /* 서브키 배열 S */
S[0]=Pw=Odd((e-2)*2**W)    /* 초기화 */
for (i = 1 ; i < t ; i++)
S[i]=S[i-1]+Qw(Qw=Odd((phi-1)*2**W))        (2)
```

byte키 배열 K를 초기화된 배열 S의 값과 혼합(mix)하기 위해 word 배열 L로 변환(convert)하고, L과 S의 혼합(mix)연산을 실행하여 워드 단위의 서브 키 배열 S를 생성한다.

암호화 과정에서 평문 2word를 w비트 레지스터 A와 B에 각각 저장하여 암호화한다. 각 라운드에서 좌우 양쪽 단어의 치환, 순열, 키 의존치환 처리 과정을 거치며, 양쪽 단어는 각 반복시마다 갱신된다. 암호화 과정은 (3)과 같은 과정을 통해 진행된다.

```
  A=A+S[0];
 B=B+S[1];
for (i=1 ; i<=R ; i++) {
  A=A^B;
  A=ROTL(A,B,W)+S[2*i];
  B=B^A;
  B=ROTL(B,A,W)+S[(2*i)+1];}        (3)
```

본 논문에서의 인증 알고리즘은 해시 함수 중 SHA-1 알고리즘을 이용하여 구현하였다.[5] SHA-1 알고리즘은 임의의 길이를 가지는 입력 메시지를 512bit 블록 단위로 처리하여 40bit (5byte)의 출력을 낸다. 알고리즘의 과정은 임의의 길이의 메시지 M이 입력으로 들어오면 이 M을 패딩(padding)과정을 통해 512bit의 배수로 만든 후, 512bit 블록 $M_i(1 \le i \le n)$로 나눈다. 각 블록 M_i를 단계 연산 과정을 통해 압축한다. 512bit 단위 블록을 처리하는 압축 함수는 모두 4라운드, 80단계로 구성된다.

본 논문에서 구현한 인증 기술의 메시지 형식은 메시

지 상태를 표현하기 위한 헤더(header) 4바이트와 가변 길이의 데이터, 5바이트의 MAC으로 구분된다.

Ⅳ. 보안 API 실험

4.1 실험 환경

본 논문에서 설계한 보안 소프트웨어에 대한 실험은 테스트 프로그램 실행하고 각 기능에 따라 라이브러리를 호출한 다음 각 라이브러리가 차지하는 메모리의 양을 체크하였다. 테스트에 사용된 컴퓨터는 그림 4와 같으며, 표 3과 같은 사양을 가지고 있다.

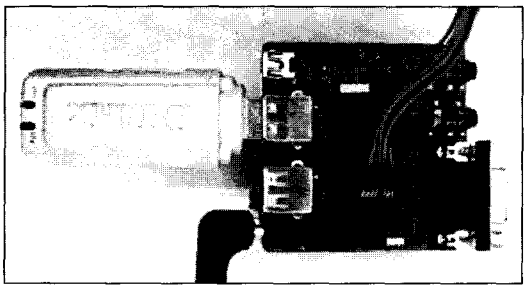

그림 4. 실험에 사용된 초경량 컴퓨터
Fig. 4. Light-weight computer for test

표 3. 초경량 컴퓨터 사양
Table 3. Specification of light-weight computer

CPU	ARM 9 Core Freescale i.MX21(350MHz)
OS	Embedded ARM Linux 3.20-rmk-mx2ads
RAM	64MB

4.2 실험 과정

1. 암호화 모듈에서 정보를 읽어서 토큰을 만들어 슬롯 ID를 부여하고 객체로 만들어 화면에 출력한다.
2. 사용할 객체를 선택한다.
3. 선택된 객체에 로그인 함으로써 사용자를 인증한다. 사용자 인증은 PIN을 이용하여 한다. (일반 사용자의 경우 저장된 PIN 번호를 사용하였고, 보안 관리자의 경우 "000000"을 입력하면 인증되도록 하였다.)
4. 일반 사용자로 로그인 한 경우 암호화, 인증, 키 관리를 선택할 수 있는 화면을 출력하고, 각각의 선택 시 해당 기능을 수행한다.
5. 보안 관리자로 로그인 한 경우 사용자의 PIN을 관리할 수 있는 화면을 출력하고 선택 시 해당 기능을 수행한다.
6. 기능을 수행했을 경우는 다시 기능을 선택할 수 있는 화면으로 되돌아가고 객체를 선택하기 위해서는 객체에 대해 로그아웃을 해야 객체를 선택하는 화면으로 되돌아간다.
7. 각 라이브러리를 호출하여 기능을 수행 전, 수행 시, 수행 후 사용되는 메모리의 양을 측정한다.

테스트에 사용된 각 라이브러리의 크기는 표 4와 같다.

표 4. 라이브러리의 크기
Table 4. Size of library

library	암호화	인증	키 관리	핀 관리
크기	17.7KB	12.9KB	6.7KB	6.8KB

4.2 실험 결과

테스트 프로그램을 시작하면 API 인터페이스와 디바이스 인터페이스, 프로그램에서 기본적으로 사용되는 함수들을 가진 소프트웨어 관리자가 메모리에 할당된다. 소프트웨어 관리자는 초기에 88KB의 메모리 사용량을 갖는다.

이후 사용자 인증 기능인 로그인 과정을 지나 객체를 선택하고 각 기능을 선택할 수 있다. 기능 선택하기 전의 메모리는 92KB이다.

암호화 기능을 선택하였을 경우, 암호화 라이브러리를 호출하며, 암호화를 위해 평문 입력을 대기하게 되고, 평문을 입력하면 암호문을 출력하고, 이 암호문에 대한 평문을 출력하며 암호 라이브러리를 해제하고 기능을 선택할 수 있는 화면으로 돌아간다. 암호화에 사용되는 메모리의 양은 그림 5와 104KB를 나타내며, 프로세스 종료 후에는 그림 6과 같이 암호 라이브러리를 할당하기 전과 같은 크기로 돌아간다.

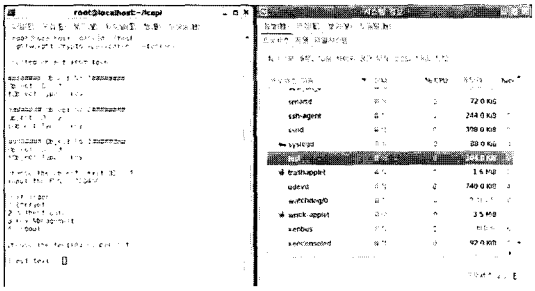

그림 5. 암호 라이브러리 할당
Fig. 5. Load of cryptographic library

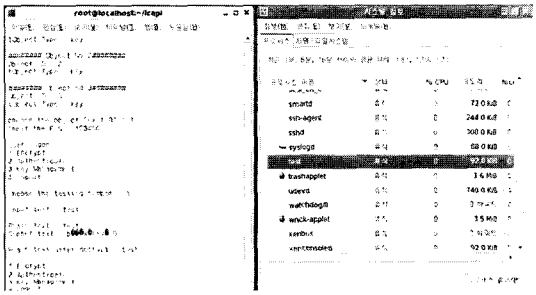

그림 6. 암호 라이브러리 해제
Fig. 6. Unload of cryptographic library

각 라이브러리의 할당/해제를 테스트 한 결과는 표 5와 같으며, 이를 차트로 표현하면 그림 8과 같다.

표 5. 보안 라이브러리 실험 결과
Table 5. Result of security library test

라이브러리	암호	인증	키 관리	사용자 펀 관리
할당 시 메모리 크기	104KB	100KB	96KB	96KB
해제 시 메모리 크기	92KB	92KB	92KB	92KB
사용 메모리	12KB	8KB	4KB	4KB

표 4와 표 5를 보면 각 라이브러리는 그 크기에 비례하여 메모리를 할당받으며, 각 기능을 사용하고 라이브러리를 해제할 경우 할당 받은 메모리만큼 다시 반납하고 할당 전 메모리로 돌아감으로써 제한된 메모리를 효율적으로 사용할 수 있음을 보여준다. 이를 토대로 시스템 자원이 제한적인 환경에서 사용하는 기능이나 기능을 사용하는 빈도 수에 따라 컴포넌트 화하여 필요한 기능만 호출하여 사용하는 방법이 시스템 자원 활용 면에서 매우 효율적이라 생각된다.

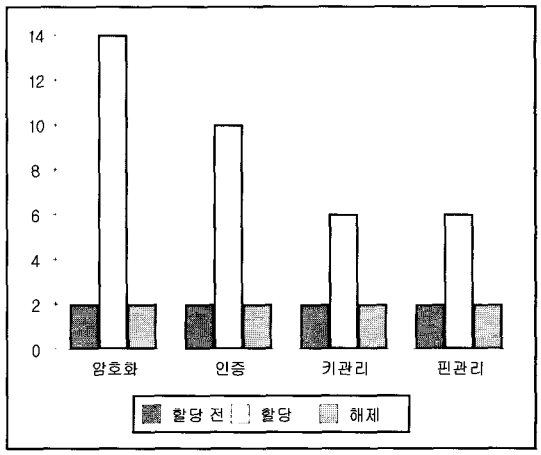

그림 7. 보안 API 실험 결과
Fig. 7. Result of security library test

VI. 결 론

본 논문에서는 초경량 컴퓨팅 환경에서 표준화 된 보안 서비스를 제공하는 초경량 환경을 위한 보안 서비스 지원을 위한 소프트웨어를 연구하고 설계하였다. 초경량 환경의 보안 서비스 지원을 위한 소프트웨어는 일반적인 컴퓨팅 환경에서와 같이 데이터 암호화, 데이터 인증, 키 관리 등의 보안 서비스를 제공하며, 초경량 컴퓨팅 환경에 맞는 알고리즘 사용, 효율적인 메모리 관리를 위해 각 서비스를 라이브러리 화하여 필요시 마다 라이브러리를 할당/해제 하는 등의 초경량 컴퓨팅 환경의 문제점을 해결할 수 있도록 설계, 구현하였다.

구현된 보안 API는 암호화를 위해 RC5 알고리즘과 인증을 위해 SHA1 알고리즘을 사용하였으며, 데이터 암호화 기능, 데이터 인증 기능, 키 관리 기능, 사용자 펀 관리 기능 등을 제공하며, 각각의 기능들을 라이브러리 화하여 필요시 마다 할당/해제 하는 방식을 사용한다. 리눅스를 이용하여 테스트 하였으며, 라이브러리의 할당/해제에 따른 메모리 변화로 초경량 컴퓨팅 환경에 맞는 효율적인 메모리 관리를 보여주었다. 추후 연구로써

RC5 알고리즘과 SHA1 알고리즘 이외에 초경량 환경에 적합한 메커니즘의 추가와 초경량 컴퓨팅 환경에서 적용시킬 수 있는 공개키 방식에 대한 연구가 진행되어야 할 것이다.

참고문헌

[1] 유용덕, 최훈, 김형신, 권영미, Takeshi Nanri "경량 미들웨어를 위한 소프트웨어 구조", 한국차세대PC학회 논문지, Vol.1 No.2, pp.34-42, 2005. 12.

[2] 이영석, 유용덕, 박상현, 최훈, "웨어러블 컴퓨팅 보안", 한국차세대컴퓨팅학회논문지, Vol 2., No. 2, pp. 23-31, 2006. 6.

[3] PKCS #11: Cryptographic Token Interface Standard v2.20, Rsa Labs.

[4] R. Baldwin, R. Rivest "The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms", RFC 2040, Oct. 1996.

[5] D. Eastlake, P. Jones, "US Secure Hash Algorithm 1(SHA1)", RFC 3174, Sep. 2001.

[6] Y. W. Law and J. M. Doumen and P. H. Hartel, "Benchmarking Block Ciphers for Wireless Sensor Networks (Extended Abstract)", 1st IEEE Int. Conf. on Mobile Ad-Hoc and Sensor Systems(MASS 2004), Fort Lauderdale, Florida, Oct. 2004.

저자소개

김 원 영(Won-young Kim)

2007년 군산대학교 전자정보공학부 공학사
2007년~현재 군산대학교 전자정보공학부 석사과정
※관심분야 : 네트워크 보안, Java 네트워크, 모바일

이 영 석(Young-seok Lee)

1992년 충남대학교 컴퓨터공학과 공학사
1994년 충남대학교 컴퓨터공학과 공학석사
2002년 충남대학교 컴퓨터공학과 공학박사
1994년~1997년 LG전자정보통신연구소 연구원
2002년~2004년 한국전자통신연구원 선임연구원
2004년~현재 군산대학교 전자정보공학부 조교수
※관심분야 : 정보보호, 이동컴퓨팅, 컴퓨터네트워크

이 재 완(Jae-wan Lee)

1984년 중앙대학교 전자계산학과 공학사
1987년 중앙대학교 전자계산학과 공학석사
1992년 중앙대학교 컴퓨터공학과 공학박사
1987년~1992년 한국체육과학연구원 선임연구원
1996년~1997년 한국학술진흥재단 전문위원
1992년~현재 군산대학교 전자정보공학부 교수
※관심분야 : 분산시스템, 운영체제, 컴퓨터통신

서 창 호 (Chang-ho Seo)

1990년 고려대학교 수학과 (학사)
1992년 고려대학교 일반대학원 수학과 (이학석사)
1996년 고려대학교 일반대학원 수학과 (이학박사)
1996년~1996년 국방과학연구소 선임연구원
1996년~2000년 한국전자통신연구원 선임연구원, 팀장
2000년~현재 공주대학교 응용수학과(정보보호전공) 부교수
2001년~현재 : 공주대학교 바이오정보학과 부교수
※관심분야 : 암호 알고리즘, PKI, 무선 인터넷 보안 등