論 文
57-4-25

모바일 애플리케이션 프로세서의 JTAG 보안 기법

A JTAG Protection Method for Mobile Application Processors

林 敏 洙* · 朴 琫 一** · 元 東 豪†
(Minsoo Lim · Bongil Park · Dongho Won)

Abstract - In this paper, we suggest a practical and flexible system architecture for JTAG (Joint Test Action Group) protection of application processors. From the view point of security, the debugging function through JTAG port can be abused by malicious users, so the internal structures and important information of application processors, and the sensitive information of devices connected to an application processor can be leak. This paper suggests a system architecture that disables computing power of computers used to attack processors to reveal important information. For this, a user authentication method is used to improve security strength by checking the integrity of boot code that is stored at boot memory, on booting time. Moreover for user authorization, we share hardwired secret key cryptography modules designed for functional operation instead of hardwired public key cryptography modules designed for only JTAG protection; this methodology allows developers to design application processors in a cost and power effective way. Our experiment shows that the security strength can be improved up to 2^{160} x 0.6second when using 160-bit secure hash algorithm.

Key Words : Application processor, JTAG Protection, Security, Authentication, Hash algorithm

1. 서 론

컴퓨터와 통신 기술의 발전에 힘입어 휴대용 개인 정보 시스템의 보급이 확대되고 있다. 이러한 시스템은 금융 업무, 전자상거래 등을 비롯한 사회적인 모든 분야에 폭넓게 이용되고 있으며, 이러한 정보 시스템에 대한 보안은 필수이다. 휴대용 개인 정보 시스템의 정보보호를 위한 가장 효과적인 보안은 암호 알고리즘을 이용하는 방법이다. 암호 기법은 오랜 역사를 가지며 연구되어 왔지만 특정 분야에 그 사용이 국한되어 왔다. 그러나 정보화 사회에서는 암호 기법이 정보 시스템 보안 방식으로 상업적으로 이용되면서 모든 정보 시스템 보안 기술로 자리 잡아 가고 있다. 이로 인해서 암호를 이용한 정보 보호 분야는 많은 발전이 있어 왔으며, 개인 정보기기의 발전과 보급으로 인하여 점차 다양한 분야에 사용되고 있다. 또한 애플리케이션 프로세서는 SoC(System on chip)와 통신 기술의 발전으로 인하여 개인이 휴대할 수 있는 모바일 제품들이 보급되기 시작하면서 휴대용 제품 시장에 최적화되어 개발된 프로세서이다. 애플리케이션 프로세서는 시장의 요구인 저 전력, 고성능, 그리고 휴대의 편이성을 위한 작은 사이즈로 제품을 만들 수 있도록 개발되고 있으며, 개인 업무, 통신, 게임, 그리고 동영상과 같은 멀티미디어 기능을 지원하기 위해 저 전력으로 최적화되어 설계된 IP(Intellectual Property)들을 사용한다.

이러한 암호 알고리즘 기술과 애플리케이션 프로세서의 발전으로 인하여 사용자들은 개인이 편리하게 휴대할 수 있는 핸드폰, PDA(Personal Digital Assistant)등의 휴대용 단말기를 이용하여 개인정보 저장 및 각종 업무에 사용하고 있다. 이와 더불어 애플리케이션 프로세서에서 수행되는 애플리케이션 프로그램의 보안기술도 발전하여 이러한 제품의 보안기술에 대한 악의적인 공격이 현재의 컴퓨팅 파워로는 시간적으로 불가능하기 때문에, 이러한 암호 시스템에 대한 공격은 암호 알고리즘이 아닌 하드웨어 구현상의 취약점에 대한 공격으로 변화하고 있다. 이러한 공격은 회사의 서버와 같은 각종 보안 장치가 완비된 곳에 대한 공격보다는 상대적으로 공격하기 쉬운 모바일 제품에 대한 공격이 될 가능성이 크다. 많은 사용자들이 휴대용 단말기에 중요한 금융 정보 및 개인 정보를 저장하는데 사용하고 있으며, 개인 정보뿐만 아닌 회사 정보 저장에 사용할 수 있다. 그러나 현재 개발되어 있는 대부분의 애플리케이션 프로세서는 JTAG 포트를 통해 프로세서의 내부 및 프로세서와 연결된 디바이스의 내부 정보를 쉽게 접근할 수 있는 취약점을 가지고 있다.

JTAG은 IEEE-1149 표준으로 프로세서 개발자와 프로세서를 이용하여 제품을 개발하는 개발자에게 가장 널리 사용되는 디버깅 인터페이스이다. 프로세서 개발자는 JTAG 포트를 제품 디버깅 용도로 사용할 수 있도록 제공하며, 이를 통하여 프로세서 내부의 CPU(Central Processing Unit) 코어는 물론 여러 IP 들을 접근할 수 있다. 또한 이를 통하여 프로세서와 연결된 외부 메모리 및 기타 장치에 대한 읽기

* 正 會 員 : 三星電子 SOC開發室 責任硏究員
** 非 會 員 : 三星電子 SOC開發室 責任硏究員
† 교신저자, 正會員 : 成均館大學校 情報通信大學院 敎授
E-mail : dhwon@security.re.kr
接受日字 : 2008年 1月 13日
最終完了 : 2008年 2月 11日

또는 쓰기 동작을 수행할 수 있다. 그 외로 JTAG 인터페이스를 사용하여 제품 보드에 장착된 디바이스들 간의 인터페이스가 정상적으로 연결 되었는지를 확인하는 용도로도 사용되며, 애플리케이션 프로세서가 수행하는 코드 및 데이터가 저장되어 있는 부트 메모리에 대한 프로그래밍 기능을 지원한다. JTAG 포트를 통한 프로그래밍 기능은 제품의 기능이 개선되었을 경우, 부트 메모리의 내용을 업데이트하는데 사용되는 중요한 기능이다. 이러한 유용한 기능을 제공하는 JTAG 포트는 보안 측면에서 취약점을 가지고 있다. 일반적으로 민감한 자료는 암호 메커니즘에 의해서 보호되어야 하며, 정당한 사용자 이외에는 JTAG 디버깅을 통해서 프로세서에 접근할 수 없어야 한다. 따라서 악의적인 사용자는 JTAG 인터페이스를 통해 프로세서 또는 프로세서에 연결된 장치의 비밀키 또는 중요한 자료에 접근할 수 없어야 하며, 정당한 사용자만이 적절한 인증 절차를 통해 접근할 수 있도록 하는 방법이 필요하다.

본 논문에서는 JTAG 포트를 통한 실용적인 디버깅 기능을 제공함에 있어, JTAG 보안 등급을 설정하여 정당한 사용자만이 JTAG을 통한 디버깅 기능을 사용할 수 있도록 한다. 정당한 사용자 인증을 위해 애플리케이션 프로세서 부팅시 부트 코드에 대한 해쉬 값을 확인하며, 보안 등급 변경을 위한 권한 인증을 위해 기존의 공개키 기반의 알고리즘이 아닌 비밀키 알고리즘을 사용한 보안 방법을 제안하고자 한다. 또한, 제안된 보안 방법을 이용하여 프로세서 내의 중요한 암호 모듈에 대한 접근 제어 방법도 제안한다. 그리고 제안하는 구조는 USB/UART 등과 같이 일반적인 인터페이스 방법을 통해 컴퓨터상에서 운영되는 보안 프로그램과 쉽게 연결되어 사용할 수 있도록 한다.

제안된 부팅시의 메시지 다이제스트(해쉬) 값을 확인하는 사용자 인증 방법을 사용함으로써 2^{160} x 0.6초의 보안 강도를 기존의 방법에 비해서 향상할 수 있었다. 이는 16-bit 인터페이스를 제공하는 NOR 플래시 메모리를 사용하여 4KB 영역에 대한 블록 삭제, 부트 코드 프로그래밍 그리고 시스템 부팅을 통한 사용자 인증 검증의 과정을 실험하여 얻은 수치이다.

본 논문에서 제안하는 새로운 JTAG 보안 기법을 설명하기 위한 구성은 다음과 같다. 2장에서는 선행 연구에 대해서 분석하며, 3장에서는 제안된 시스템 구조가 어떻게 애플리케이션 프로세서에 구현되어야 하는지를 살펴본다. 4장에서는 메시지 다이제스트 기법을 이용한 사용자 인증 방법에 대해서 자세히 설명하며, 5장에서는 JTAG 보안 등급의 필요성과 어떻게 설정될 수 있는 지를, 그리고 보안 등급을 변경할 수 있는 권한 인증 방법에 대해서 설명한다. 6장에서는 메시지 다이제스트(해쉬) 기법을 이용한 사용자 인증 방법의 안정성에 대해서 실험을 통한 분석과 제안되는 시스템 구조를 구현하기 위해 필요한 하드웨어 암호 알고리즘을 살펴본다. 제안되는 시스템 구조 및 기법은 기존의 방법보다 보안 강도와 하드웨어 암호 알고리즘을 효율적으로 사용할 수 있는 장점을 가진다.

2. 기존 연구

애플리케이션 프로세서는 구현된 제품이 회사 및 개인의 중요한 정보를 담고 있기 때문에 이를 보호하기 위한 방법 및 장치가 필수이다. 이를 위해 애플리케이션 프로세서는 하드웨어 암호 알고리즘을 이용하여 메모리 등에 저장된 중요한 자료를 암호화 하며, 프로세서가 사용하는 부트 코드의 위조 또는 변조를 방지하기 위해서 부팅 시 부트 코드의 무결성을 확인하는 안전한 부팅 메커니즘[2], 그리고 JTAG 포트를 통한 악의적인 접근을 방지하기 위한 JTAG 보호 메커니즘 등을 사용하여 프로세서를 보호한다.

JTAG 포트 인터페이스에 관련된 연구는 JTAG 포트를 통한 위협을 제거하기 위하여 프로세서의 JTAG 포트 인터페이스를 불가능하게 만드는 방법[6]이 제안되었다. 이 방법은 JTAG 포트를 사용할 수 없게 함으로써 외부 위협을 완전히 차단할 수 있는 장점이 있으나, JTAG 포트를 통한 디버깅 기능을 사용할 수 없게 됨으로써 제품 기능 이상 시 내부 프로세서의 상태 정보 등을 쉽게 활용할 수 없는 문제가 있다. 이로 인해 내부 정보를 얻어내기 위한 디버깅 비용 및 소프트웨어 업그레이드를 위한 비용이 상승되는 단점이 있다.

이에 Freescale Semiconductor 사는 JTAG 포트에 대한 보안 등급을 설정할 수 있는 제품을 개발하였다[5]. 이 방법은 퓨즈(electric fuse)를 이용하여 보안 등급을 설정하는 방법으로 JTAG 포트에 대한 여러 보안 등급이 결정할 수 있으며, 일단 결정된 보안 등급은 변경이 불가능한 구조다. 그러나 권한을 가진 사용자들에게 JTAG 포트를 제한 없이 사용할 수 있는 모드가 제공된다. 특히, 권한을 가진 사용자인지를 확인하는 방법으로 challenge and response 프로토콜을 이용하여, 하드웨어로 정해진 코드 값을 확인하는 방법으로 사용자 인증을 한다. 그러나 임의의 값이 아닌 프로세서 내부에 저장된 고정된 코드 값을 이용하여 인증하는 방법은 재전송 공격(replay attack)에 상당히 취약한 방법으로 알려져 있다.

이에 대한 연구[1]은 위의 두 문제에 대한 해결책을 제시한다. 제품의 라이프 사이클 동안 JTAG 보안 등급을 조절할 수 있는 기능을 두어 JTAG 포트를 통한 디버깅 기능이 필요할 경우, 사용자 인증을 통해 보안 등급을 변경할 수 있다. 이 방법은 외부의 위협으로부터 JTAG 포트를 보호하기 위해 높은 보안 등급을 설정하며, JTAG 포트를 통한 디버깅 기능이 필요할 경우 신뢰성 있는 공개키 기반의 암호 알고리즘을 통한 사용자 인증과 보안 등급 변경을 거쳐서 JTAG 포트를 사용할 수 있도록 한다.

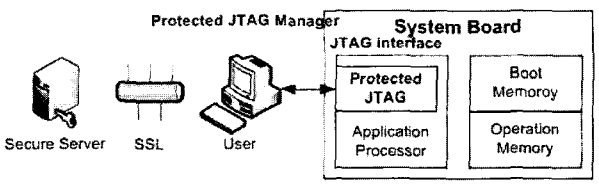

그림 1 JTAG 보안을 위한 기존 연구에 대한 블록도
Fig. 1 *Block diagram of JTAG protection on previous study*

그림 1은 논문[1]의 블록도 이다. 애플리케이션 프로세서의 JTAG 포트 보안 등급 변경이 필요할 경우, 중간에 위치한 사용자(user)는 사용자 컴퓨터의 응용 프로그램인 Protected JTAG Manager을 통해 네트워크 존재하는 Secure server에 접속한다. 이때 사용자 컴퓨터(user)와

Secure server간의 인터페이스는 안정성이 보장된 SSL(Secure Sockets Layer)을 통해서 연결되며, JTAG 포트의 보안 등급을 변경하고자 하는 사용자는 인증을 거쳐 Secure server를 사용할 수 있는 권한을 받은 후에 보안 등급을 변경한다. 보안 등급 변경은 프로세서 내부의 JTAG 포트를 보호하기 위해서 독립적으로 설계된 Protected JTAG과 Secure server간에 공개키 암호 알고리즘인 ECC(Elliptic Curve Cryptography) 알고리즘으로 인증하며, 인증 후에 JTAG 포트의 보안 등급을 변경한다.

논문[1]은 프로세서의 JTAG 포트를 보호하기 위한 인증 방법으로 CPU(Central Processing Unit)와 독립된 Protected JTAG을 위한 하드웨어 ECC 알고리즘을 사용한다. 이 방법은 저 전력을 필요로 하는 모바일 제품에 디버깅 모드에서만 사용되는 하드웨어로 인하여 칩 사이즈 및 파워 소모가 증가될 수 있는 문제가 있다.

3. 시스템 구조

JTAG 보안을 위한 시스템 하드웨어의 구조는 그림 2와 같다. 그림 2에서 시스템 구조는 애플리케이션 프로세서의 보안 기능 및 JTAG 보안을 위해 최소로 필요한 하드웨어 암호 알고리즘 IP에 대한 구조도 이다. 보안 기능을 갖는 애플리케이션의 경우, 부트 코드의 무결성 확인을 위한 해쉬 알고리즘, 자료 암호화를 위한 암호 알고리즘, 그리고 암호기 생성 등을 위한 임의 키 생성기가 필수이다. 이와 같은 구조의 장점은 하드웨어 암호 알고리즘이 보안 기능과 프로세서의 보안 등급 변경을 위해서 사용될 수 있다는 것이다. 또한 암호 알고리즘이 사용할 비밀키 및 안전한 부팅에 사용할 해쉬 값을 저장할 비휘발성 메모리가 필요하다. 이를 위한 저장 매체로서 한 번 기록한 후에는 다시 재 기록할 수 없는 매체보다는 반복하여 기록이 가능한 플래시 메모리가 적합하다. 이는 애플리케이션 프로세서의 보안 등급을 저장하기 위해서도 필수이며, SoC 기술 발전으로 인하여 프로세서 내부에 플래시 메모리를 내장할 수 있다.

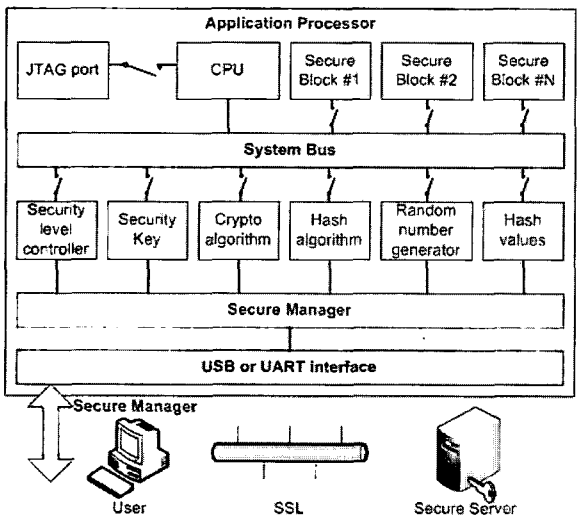

그림 2 제안된 JTAG 보안을 위한 시스템 구조

Fig. 2 System architecture for suggested JTAG protection

논문[1]과 같이 JTAG 포트를 사용하는 사용자 인증 및 암호 등급을 조절하는 방법은JTAG이라는 하나의 인터페이스만을 통해서 사용자 인증 및 보안 등급 변경이 가능하다는 장점이 있으나, 프로세서 회사 또는 제품 개발회사에서 JTAG 포트를 이용한 애플리케이션 프로그램 제작 및 하드웨어를 제작하여야 한다. 그러나 본 논문에서 제안하는 것처럼 이미 보편화 되어 있는 USB 또는 UART 등을 이용한 응용 프로그램 작성이 개발자에게 더 실용적일 것이며, 개발 기간도 단축시킬 수 있을 것이다. 또한 보안 등급을 위한 해쉬 값 확인 또한 개발자에게 부트 코드를 변경하여야 하는 번거로움을 주는 단점이 있으나, 날로 발전하는 컴퓨팅 파워를 사용 불가능하게 한다는 점에서 보안 강도 향상이라는 장점을 가진다. 사용자 인증을 위한 해쉬 값 생성, 비교, 그리고 secure server간의 권한 인증은 애플리케이션 프로세서 내의 전용 하드웨어 로직 또는 프로세서 내부의 ROM 코드로 구현될 수 있으며, 사용자 인증 및 권한 인증의 절차는 응용 프로그램 또는 운영체제와의 연계 없이 프로세서 내에 하드웨어로 구현된 로직에 의해서 진행되어야 한다. Secure server는 제품의 보안 등급 조절 및 제품 관리를 위해 회사 내의 안전한 곳에 위치하여야 하며, 이를 이용할 수 있는 권한을 가진 사용자에게만 접근 및 보안 등급 변경을 위한 권한을 주도록 구현한다.

4. 메시지 다이제스트 기법을 이용한 사용자 인증

애플리케이션 프로세서의 JTAG 포트 보안에 있어서 가장 중요한 기능은 정당한 사용자만이 보안 등급을 변경할 수 있도록 하는 사용자 인증이다. 이 기능을 통해서 정당한 사용자에게 프로세서의 JTAG 포트 인터페이스를 통해 내부 레지스터 또는 외부 메모리를 접근할 수 있는 권한을 주어야 하며, 그 외의 사용자로부터는 프로세서의 중요한 정보를 보호하여야 한다. 또한 컴퓨터의 계산 능력을 활용하기 어려워야 하며, 암호 알고리즘에 대한 새로운 공격방법들로부터 안전한 인증 방법이 필요하다.

따라서 본 논문에서는 컴퓨터의 연산능력을 활용할 수 없는 사용자 인증 방법으로 암호 해쉬(메시지 다이제스트) 알고리즘을 사용한다. 애플리케이션 프로세서를 보호하기 위한 암호화 해쉬는 프로세서의 안전한 부팅을 보장하기 위한 방법으로, 부팅시 부트 메모리 특정 영역의 값을 계산한 해쉬 값과 이미 원본 부트 메모리 특정 영역의 값을 계산하여 프로세서 내부의 플래시 메모리에 저장한 값을 비교하여 부트 메모리의 무결성을 확인한다. 플래시 메모리에 저장된 해쉬 값은 외부로부터 접근할 수 없는 안전한 곳 보관되며, 부트 코드가 위조 또는 변조되지 않았음을 확인하고 부팅을 가능하게 하는 방법이다[2]. 애플리케이션 프로세서가 비교하는 해쉬 값은 제품 개발자에 의해 프로세서 내부의 안전한 장소에 저장되며, 일단 저장된 값은 외부에서 접근이 불가능하다. 또한, JTAG 보안 등급 변경을 위한 사용자 인증을 위해서, 애플리케이션 프로세서는 부트 코드에 대한 해쉬 값을 계산 및 비교하여야 하며 이는 절대적인 시간이 소요되는 작업이다. 즉, 부트 코드의 해쉬 값을 계산하기 위해 소요되는 메모리 접근 시간은 악의적인 사용자 등에 의해서 조작 또는 변경될 수 없다. 따라서 본 논문에서는 정당한 사용자

인증을 위한 방법으로 부트 코드의 해쉬 값이 정당한 값인지를 확인하는 방법을 사용자 인증 방법으로 제안한다. 이 방법은 컴퓨터의 강력한 컴퓨팅 능력을 활용할 수 없으므로 사용자 인증을 위한 악의적인 시도는 시간적으로 불가능하다. 또한 사용자 인증을 위해 부트 메모리에 저장된 부트 코드는 최종 사용자를 위해 제품에 적용된 부트 코드가 아닌 JTAG 포트의 보안 등급을 변경하기 위한 사용자 인증용 부트 코드이다. 따라서 JTAG 보안 등급 변경을 위한 부트 코드 및 이에 대한 해쉬 값은 제품 개발자에 의해 생성, 관리되므로 외부 사용자에게 노출되지 않는다. 특히 해쉬 값은 애플리케이션 프로세서의 안전한 곳에 보관되며, 이 값을 외부에서 읽을 수 있는 방법이 제공되지 않아야 하며, 악의적인 부트코드가 수행될 수 없도록 부팅시 해쉬 값을 먼저 검토하여 실행 여부를 판단하는 부트 코드 또는 하드웨어가 프로세서 내부에 구현되도록 한다.

그림 3은 애플리케이션 프로세서의 부팅 순서에 대한 순서도로, 애플리케이션 프로세서는 내부의 안전한 장소에 프로세서에서 필요로 하는 모드 별로 각각의 해쉬 값을 가지며, 3가지 모드에 대한 예이다. 이는 표 1과 같이 제품용으로 일반 사용자를 위하여 개발된 코드에 대한 해쉬 값, JTAG 포트 보안등급 변경을 위한 사용자 인증용 해쉬 값, 그리고 애플리케이션 프로세서의 제품별 암호 알고리즘의 접근 제어를 위한 사용자 인증용 해쉬 값으로 나뉘진다. 본 논문에서는 JTAG 보안 등급 변경용, 그리고 제품에 따른 보안 알고리즘 접근 제어 변경용과 같이, 필요로 하는 기능 등에 따라 다른 해쉬 값을 사용하게 함으로서 각 보안 기능에 따른 보안을 유지할 수 있는 방법을 제안한다. 애플리케이션 프로세서는 부팅시 부트 코드의 해쉬 값을 계산하여 프로세서 내부의 안전한 곳에 저장된 해쉬 값과 비교하여 어느 모드로 진입할지 결정한다.

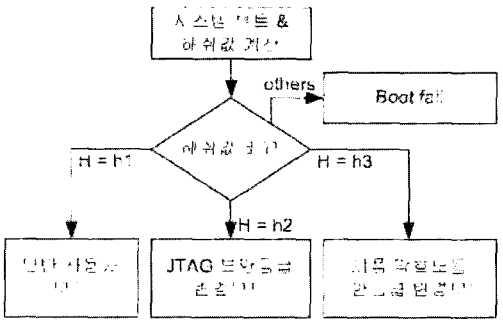

그림 3 부트 메모리의 해쉬 값을 이용한 보안 모드 인식도
Fig. 3 Protection mode recognition diagram using hash values of boot memory.

표 1과 같은 모드 변경을 위한 해쉬 값 참조는 보안을 필요로 하는 각 기능에 대해서 정당하지 못한 사용자로부터 그 기능을 은닉시킬 수 있는 장점이 있으며, 이를 공격하게 위해 부트 메모리의 부트 코드를 변경하는 작업은 많은 시간과 비용이 소요되는 작업이다. 표 1에 설명된 JTAG 보안 등급 및 제품별 보안 알고리즘에 대한 접근 제어는 5.1에서 자세히 설명하기로 한다.

표 1 부트 코드에 대한 해쉬 값의 예
Table 1 Example hash values for boot codes

해쉬 값	해쉬 값에 따른 프로세서의 기능 모드
해쉬 값 #1	최종 사용자를 위한 제품용 해쉬 값
해쉬 값 #2	JTAG 포트 보안 등급 설정 및 변경용
해쉬 값 #3	제품 보안알고리즘 접근제어 설정, 변경용
기타	부트코드 위조, 변조로 판단하여 부팅불허

5. 권한 인증

5.1 하드웨어 암호 알고리즘의 차별적인 사용 인가

우리 주변에는 많은 임베디드 시스템 제품들이 사용되고 있다. 어떤 제품은 보안 알고리즘을 필요로 하지 않으며, JTAG 포트에 대한 접근 제한도 필요하지 않는다. 그러나 모바일 환경에서 사용되는 휴대전화, PDA 등의 개인 단말기 등의 제품의 경우는 타인 또는 악의적인 사용자에게 노출되어서는 안 되는 개인 정보 및 제품 정보를 포함하고 있어, 외부의 악의적인 공격으로부터 보호되어야 한다. 또한 제품이 다양해짐에 따라 제품 별로 다른 보안 등급을 필요로 하며, 어떤 제품은 사용기간에 따라서 다른 보안 등급을 필요로 한다.

제품의 라이프 사이클은 디자인부터 시작하여 개발, 테스트를 통하여 최종 제품으로 조립되어 사용자에게 전달된다. 보안 제품 경우는 개발단계부터 시작하여 최종 사용자에게 전달될 때까지 서로 다른 보안 등급들이 필요하다. 일반적으로 최종 사용자에게 접근할수록 외부의 악의적인 공격을 받을 위협의 가능성이 크기 때문에 더 높은 보안 등급을 필요로 한다. 또한 제품의 라이프 사이클 동안 제품 결함으로 인해 프로세서 내부를 디버깅할 수 있도록 보안 등급을 낮출 수 있는 방법이 필요하며, 이러한 보안 등급의 조절은 인증된 사용자에게만 허락되어야 하며 그 외의 사용자로부터 보호되어야 한다.

표 2 JTAG 사용자 권한 등급 레벨 예
Table 2 *Authentication level examples for JTAG protection*

보안 등급	보안 등급에 따른 설명	등급 변경
1	JTAG 사용, 프로세서 모든 기능사용 가	가능
2	JTAG 사용, 외부 디바이스 접근가능, 암호기능 사용불가	가능
3	JTAG 사용가능, 암호기능 및 외부 디바이스 사용불가	가능
4	JTAG 포트 인터페이스 기능사용 불가	가능
5	JTAG 포트 인터페이스 기능사용 불가	불가

표 2에 설명된 JTAG 사용자 권한 등급은 논문[1]을 통해서 설명된 것과 유사한 여러 개의 조절 가능한 보안 레벨들을 두어 사용자의 접근 권한을 조절할 수 있다. 이는 프로세서 개발자, 제품 개발자 그리고 제품 사용자간의 JTAG 접근 권한을 조절할 수 있는 기능이다. 일반적으로 프로세서 및 제품 개발자는 JTAG 포트를 사용하여 프로세서 내의 모든 곳을 제한 없이 접근할 수 있는, 보안 등급이 가장 낮은 보안 1등급에서 프로세서 및 제품을 개발하며, 최종 사용

자에게는 JTAG 포트를 통한 디버깅 기능을 사용할 수 없는 보안 등급 4 또는 5로 배포된다. 보안 등급 2 또는 3의 경우는 JTAG 포트를 통한 디버깅 기능을 사용할 수 있으나, 보안 기능과 관련된 하드웨어 암호 알고리즘 및 중요한 정보를 저장하고 있는 자원에 대한 접근은 제한된 등급이다. 이러한 등급은 암호 기능과 관련이 없는 개발자에게 제품의 중요한 정보를 유출하지 않으면서 제품에서 보호되어야 할 자원과 관련 없는 부분에 대한 기능 검증 및 디버깅을 진행할 수 있는 유용한 등급이다. 특히 보안 등급 5의 경우는 제품 개발이 완료되었을 경우 사용할 수 있는 등급으로 JTAG 포트를 이용한 디버깅 기능 및 보안 등급 변경이 불가능하다.

본 논문에서는 애플리케이션 프로세서의 사용용도 및 사용자 권한 등급에 따라 시스템의 하드웨어로 구현된 암호 알고리즘과 중요한 정보를 저장하고 있는 레지스터 또는 메모리에 저장된 자원들에 대한 접근을 차별화함으로써, 암호 관련 모듈들이 불법적으로 이용될 수 있는 것을 차단하는 기법을 제안한다. 암호 알고리즘을 가진 애플리케이션 프로세서가 사용되는 제품은 그 사용 분야가 암호기능을 필요로 하는 제품에 한정된다. 이는 보안 기능이 필요하지 않은 목적으로 보안기능을 포함하고 있는 프로세서를 구입한 악의적인 사용자가 프로세서 내부의 중요 정보 및 제품의 취약성 분석용으로 사용할 수 있는 가능성이 있기 때문이다. 이와 같이 보안 기능을 갖는 애플리케이션 프로세서가 가지는 문제점은 보안 기능을 필요로 하지 않는 제품에 사용될 경우 악용될 수 있다는 것이다. 그리고 제품을 개발하는 업체에 따라서 필요한 하드웨어 암호 알고리즘 등의 사양이 서로 다를 수 있다. 따라서 본 논문에서는, 다음 표 3와 같은 제품 관점에서 애플리케이션 프로세서가 갖고 있는 하드웨어 암호 알고리즘에 대한 접근 제어 기능을 제안한다. 이 접근 제어 기능을 통해서 앞에서 설명된 보안 기능을 갖는 애플리케이션 프로세서가 가진 제한성에서 벗어날 수 있다. 또한, 표 3에서 설명된 것 외의 기능을 지원하는데 유용하게 사용될 수 있다. 한 예로, 애플리케이션 프로세서는 입출력 포트를 사용하는 많은 IP를 포함하고 있으므로 사용할 수 있는 입출력 포트의 수가 제한되어 있다. 따라서 UART (Universal Asynchronous Receiver / Transmitter)와 USB(Universal Serial Bus) 인터페이스가 동일한 입출력 포트를 공유하고 있을 경우, 적용될 제품의 기능에 따라서 사용할 수 있는 IP를 고정하여 프로세서의 동작 모드를 결정할 수 있다. 이러한 기능은 동일한 프로세서를 사용하여 각각 다른 기능을 필요로 하는 제품을 제공하는 데 있어서 효과적이다.

표 3 제품 관점의 하드웨어 암호 모듈에 대한 접근 제어
Table 3 Access control about hardware cryptography modules

접근등급	보안 등급에 따른 설명	등급 변경
등급 1	하드웨어 암호 모듈 사용불가	불가
등급 2	하드웨어 암호 모듈 사용불가	가능
등급 3	일부 하드웨어 암호 모듈 사용가능	불가
등급 4	일부 하드웨어 암호 모듈 사용가능	가능
등급 5	모든 하드웨어 암호 모듈 사용가능	불가
등급 6	모든 하드웨어 암호 모듈 사용가능	가능

표 3는 하나의 애플리케이션 프로세서를 이용하여 보안 기능이 필요한 제품과 필요하지 않은 제품에 필요한 접근 제어에 대한 한 예이다. 이는 JTAG 포트를 통한 디버깅 기능이 보안 등급에 따라 사용 기능이 제한되는 것과 같이, 애플리케이션 프로세서의 제품 별로 접근 제어를 통해 하드웨어 암호 알고리즘을 사용할 수 있는 권한이 제한된다. 이러한 접근 제어는 하드웨어 암호 알고리즘을 갖는 애플리케이션 프로세서가 암호 알고리즘을 필요로 하지 않는 제품에 사용될 수 있도록 한다(등급 1과 2). 또한 애플리케이션 프로세서의 하드웨어 암호 알고리즘은 필요한 일부 기능만 제품에 사용될 수 있다(등급 3과 4). 이러한 기능을 사용함으로써, 여러 개의 프로세서를 개발하는 소요되는 시간, 비용, 그리고 자원을 줄일 수 있다. 이러한 프로세서의 접근 제어 등급은 구현된 애플리케이션 프로세서가 가지고 있는 하드웨어 보안 알고리즘에 따라 더 세분화될 수 있다.

5.2 보안 등급 변경을 위한 권한 인증 기법

본 논문에서는 JTAG 포트 보안을 위해서 부트 메모리에 저장된 부트 코드에 대한 해쉬 값을 이용하여 사용자 인증을 진행하며, 보안 등급을 변경하기 위해 사용자 권한 인증을 진행한다. JTAG 포트의 디버깅 등급을 변경하기 위해서, 사용자는 JTAG 보안 등급 변경을 위한 부트 코드를 프로그램 하여 부팅을 진행한다. 부팅을 통한 사용자 인증 후, 사용자 권한 인증을 위한 절차를 진행한다. 본 논문에서는 논문[1]과는 달리 사용자 인증을 위한 해쉬 값을 확인하는 방법으로 컴퓨터의 강력한 계산 능력을 사용할 수 없도록 하며, 또한 사용자 권한을 확인하는 과정에서 JTAG 포트가 아닌 USB 또는 UART 인터페이스를 사용한다. 일반적으로 JTAG 포트를 이용한 인터페이스는 케이블 이외에 인터페이스를 위해 사용자 컴퓨터와 프로세서 간에 별도의 하드웨어가 필요하며, 이를 위한 툴 지원이 필수이다. 또한 논문[1]에서 제안된 프로세서의 CPU와 독립된 하드웨어로 구현된 "Protected JTAG"을 구현하기는 많은 시간과 비용이 소모되는 작업이다. 이 보다는 이미 개발 환경이 갖추어 있으며, 쉽게 인터페이스가 가능한 USB 또는 UART 포트를 이용하여 인터페이스를 지원하는 것이 프로세서와 제품 개발자에게 더 실용적이며, 유연한 개발환경을 제공한다.

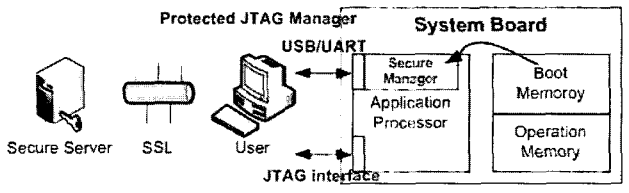

그림 4 제안하는 JTAG 보안 시스템의 블록 다이어그램
Fig. 4 Block diagram of suggested JTAG protection system

보안 등급을 변경하기 위하여 사용자(user)는 그림 4과 같이 네트워크에 연결된 Secure sever에 접속하게 된다. 보안 등급을 변경하고자 하는 사용자(user)는 Secure server로부터 사용자 및 권한 인증을 진행하며, 이 뒤의 일련의 절차는 사용자 간섭 없이 진행된다. 애플리케이션 프로세서는

Secure Server와 연결된 후 제품 ID와 고유한 Chip ID를 서버에 전달한다. 서버는 제품 ID와 Chip ID를 이용하여 저장된 DB에서 해당 애플리케이션 프로세서의 비밀키를 준비한다. 보안 등급 변경은 Secure sever와 애플리케이션 프로세서간의 여러 번의 challenge and response 프로토콜을 통해서 상대방이 정당한 사용자인지를 검증 한 후에 보안 등급을 조절한다.

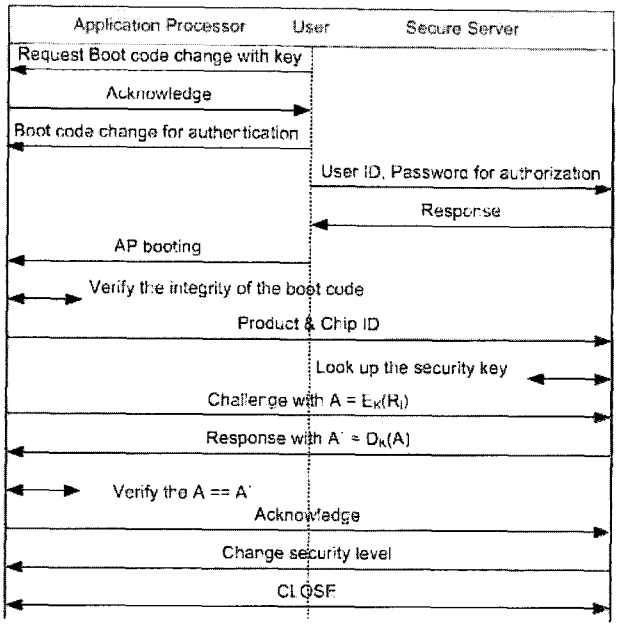

그림 5 JTAG 보안을 위한 프로토콜
Fig. 5 Suggested protocol for JTAG protection

이에 대한 프로토콜은 그림 5와 같다. 사용자 인증을 위해 사용된 challenge and response 프로토콜은 공개키 기반의 알고리즘이 일반적이다. 그러나 본 논문에서는 암호키는 프로세서와 Secure server가 모두 알고 있다는 점과 모든 암호 기능을 갖는 애플리케이션 프로세서의 경우, 공개키 암호 알고리즘보다는 비밀키 암호 알고리즘을 모두 내장하고 있으므로 비밀키 알고리즘을 이용한 인증 방법을 제안한다. SSL을 이용한 인증 과정에서 사용자 컴퓨터의 공개키 암호 알고리즘을 이용하여 Secure server와의 보안 프로토콜 협상을 진행하며, 프로세서가 필요로 하는 Secure server에 대한 인증에 대해서는 Secure server와 사용자 컴퓨터 간에 설정된 비밀키 암호 알고리즘에 의해 캡슐화 되어 진행이 가능하다. 이와 같은 방법을 사용하면 프로세서는 사용자 인증을 위해 공개키 암호 알고리즘을 사용하지 않아도 된다. 따라서 그림 5의 프로토콜은 사용자 컴퓨터와 secure server 간에 설정된 암호 프로토콜에 캡슐화 되어 처리된다. 애플리케이션 프로세서에서 Secure server가 정당한 사용자임을 확인하는 방법으로 임의 값에 대한 암호화 및 복호화 된 값을 가지고 확인하므로 비밀키는 외부로 유출되지 않는다. 또한, 프로세서는 secure server에 의해 생성 또는 secure server가 가지고 있는 키와 Chip ID를 이용하여 생성할 수 있는 고유한 비밀키를 사용하므로 특정 프로세서의 키가 유출되었다고 해서 다른 프로세서의 보안에 영향을 주지 않는다.

6. 분 석

6.1 해쉬 값을 이용한 사용자 인증 방법의 안정성

JTAG 포트를 이용하여 디버깅 기능을 사용하기 위하여 암호화 해쉬 기법을 이용한 사용자 인증의 안정성을 분석하기 위해서 부트 메모리로 널리 사용되는 NOR Flash 메모리를 이용한 사용자 인증 시간을 산출하였다. 표 4는 보안 등급 변경을 위해 필요한 NOR Flash 메모리의 부트 타이밍 값으로 NOR 플래시 메모리의 스펙 값[9]과 실험값이다. 이때 해쉬 값을 계산하기 위해 외부 16 비트 데이터 인터페이스로 연결된 NOR 플래시 메모리의 4KByte 영역에 대해서만 계산한다고 가정할 경우, 아래 표 4와 같이 최소 약 0.6초가 소요로 측정되었다. 이 값은 플래시 메모리 프로그램과 부트 코드를 DRAM 영역에 저장한 후에 애플리케이션 프로세서의 CPU를 이용한 NOR 플래시 메모리에 대한 섹터 삭제, 4KB 부트 코드 프로그래밍, 부팅을 통한 사용자 인증 시간을 계산한 것으로, 사용자 인증이 실패하였을 경우 다시 부트 메모리를 삭제하고 프로그램하기 위한 코드를 수행할 수 없기 때문에 실질적인 시간은 이 보다 더 소요될 것으로 예상된다. 또한 메모리의 특성으로 인하여 프로그래밍할 수 있는 회수가 최소 1,000,000(1,000K)번으로 제한되어 있다. 이는 160 비트 해쉬 값을 가지는 SHA1 암호화 해쉬 알고리즘을 사용한다고 가정할 경우, 현실적으로 해쉬 값을 모르는 상태에서 전수검사(2^{160} x 0.6초)로 해쉬 값에 대한 충돌 쌍을 찾기는 불가능하다.

표 4 사용자 인증을 위한 1회 공격 시간
Table 4 One-time attack time against user authentication

조 건	AP tPOR + Block Erase + 4KB Word Program + 4KB Read Access
스펙 값	2.001 + 1000 + 16.38 + 0.143 = 1018.524ms
실험 값	2.001 + 580 + 14.525 + 4.250 = 600.776ms

tPOR: Power on Reset Timing

표 5 NOR 플래시 메모리 타이밍 테이블
Table 5 NOR flash memory timing table

Function	스펙 값 Typical	실험값	Unit
Sector Erase (16KB)	1000	580 ~ 620	ms
4KB Program	16.38 = 4K/2 x 8us	14.525	ms
Read Access	70	2,075 = 83-cycle x 25ns	ns
4KB Read on boot	143.36 = 4K/2 x 70ns	4,249.5 = 4K/2 x 2.075us	us
Erase/Program Endurance	1,000K	-	cycle

표 5는 표 4에서 사용한 각 타이밍 값에 대한 메모리 스펙 값과 실험 및 시뮬레이션을 통한 수치이다. 사용된 메모

리는 AMD사의 Am29LV800D NOR 플래시 메모리[9]로 삼성사의 S3C2443 애플리케이션 프로세서를 이용하여 실험한 섹터 삭제 시간 및 4KB의 프로그래밍 시간을 구하였으며, 부팅시에 소요되는 4KB의 읽기 소요시간은 시뮬레이션을 통해서 산출하였다. S3C2443에 적용된 정적 메모리(SRAM) 컨트롤러의 리셋 값은 32 사이클이며, 한 워드인 32-bit을 읽기 위해 소요되는 시간은 내부 latency로 인해서 166 사이클이 소요되는 것으로 그림 6과 같이 시뮬레이션 되었다. 또한, S3C2443이 지원하는 oscillator 사양이 최대 40MHz이므로 한 사이클 당 25ns기준으로 계산하였다.

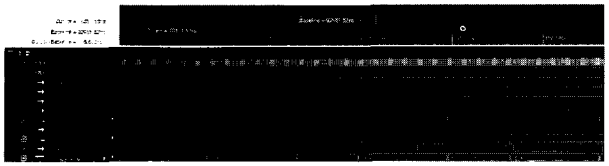

그림 6 NOR 플래시 메모리 부트 타이밍 다이어그램
Fig. 6 NOR flash memory boot timing diagram

그림 7과 표 6은 S3C2443 애플리케이션 프로세서의 tPOR 타이밍이다. 이는 부팅시의 사용자 인증이 실패할 경우, 부팅을 다시 실행시키기 위한 방법으로 사람이 직접 리셋 하는 방법은 시간이 많이 소요되므로, 파워를 리셋 하는 방법을 사용한다고 가정할 경우 최소로 필요한 값이다.

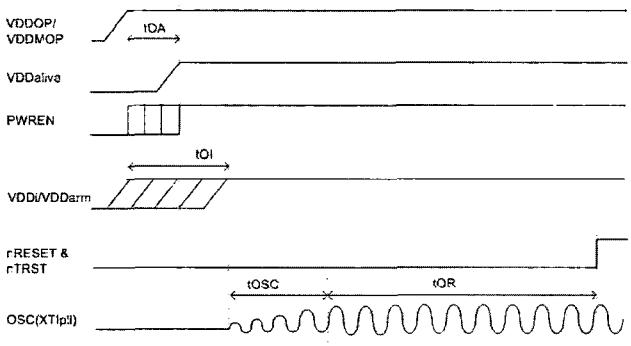

그림 7 Power on reset 타이밍
Fig. 7 Power on reset timing

표 6 Power On reset 타이밍 스펙
Table 6 Power on reset timing specifications

Symbol	Description	Min	Unit
tOA	VDDOP/VDDMOP to VDDalive	0	ms
tOI	PWREN to VDDi/VDDarm	0	ms
tOSC	VDDi/VDDarm to Oscillator stabilization	2	ms
tOR	Oscillator stabilization to PnRESET & PnTRST high	1	us

6.2 보안 기능을 위한 하드웨어 암호 알고리즘

모바일 제품의 경우, 보안 기능을 지원하기 위한 하드웨어 암호 알고리즘을 지원하는데 제한이 있다. 표 7는 하드웨어로 구현된 암호 알고리즘의 gate count와 그에 따른 성능을 보여준다. 암호 알고리즘이 사용되는 예를 들면, ARM사의 TrustZone을 이용한 안전한 부팅(Secure boot) 방법에서는 부트 코드의 무결성을 증명하기 위한 방법으로 해쉬 알고리즘을 사용 한다. 또한 중요한 데이터를 암호화하기 위해 비밀키 암호 알고리즘과 비밀키 생성을 위한 PRNG(Pseudo Random Number Generator) 또는 TRNG(True Random Number Generator)을 사용한다. 그리고 기능상으로 필요할 경우 공개키 알고리즘인 RSA를 사용하는 방법이 존재한다.

정보보호 제품에서 논문[1]에서 제시된 ECC 공개키 알고리즘 보다는 RSA 공개키 알고리즘이 하드웨어 형태로 널리 지원되고 있으며[10], 공개키 알고리즘의 경우 연산 속도가 비밀키 암호 알고리즘보다 느린 문제로 인해 문서 암호화보다는 전자서명과 사용자 인증에 주로 사용되고 있다. 따라서 문서 암호화의 경우는 비밀키 암호 알고리즘이 보편적으로 사용되며, 이는 JTAG 포트의 보안을 위해서 프로세서에서 필요로 하지 않는 공개키 알고리즘을 사용하여 제작비용과 개발 기간을 증가 시킬 수 있다. 또한 표 7를 참조하면, 공개키 암호 알고리즘은 비밀키 암호 알고리즘과 비교하여 칩 면적(gate count)을 많이 요구하기 때문에 모든 애플리케이션 프로세서에 사용하기가 쉽지 않으며, 사용 빈도가 많지 않기 때문에 소프트웨어로 구현되어 사용되기도 한다. 논문[1]은 프로세서의 CPU를 활용하지 않고 독립적인 하드웨어인 Protected JTAG을 두어 외부로 어떠한 정보도 나가지 않는 장점을 제공한다. 이 장점은 안전한 부팅(secure boot) 방법에서 사용하는 프로세서 내부 ROM을 사용하여 JTAG 보안을 위한 일련의 순서를 내장하는 방법을 사용함으로써 프로세서 외부로 어떤 정보도 유출되지 않도록 할 수 있다. 또한, 사용되는 ROM에 저장되는 코드도 JTAG 보안 등급과 동일하게 보안 등급을 설정하여 외부로 유출되지 않게 관리할 수 있으며, ROM에 저장된 코드가 유출되더라도 코드의 역할은 데이터의 흐름 제어이므로 비밀키 및 프로세서의 중요한 정보는 밖으로 유출되지 않는다. 이러한 방법은 논문[1]에서 제안한 하드웨어로만 설계된 모듈을 사용하는 것보다, 프로세서 내부에 프로그램된 ROM을 사용하여 제어할 수 있으므로 유연한 개발환경을 제공한다. 유연하게 제어할 수 있는 CPU를 사용함으로써 개발 시간과 비용, 그리고 전용의 Protected JTAG을 구현하는 데 따르는 위험을 줄일 수 있으며, 내부 암호 알고리즘에 의해서 보호되는 ROM을 사용함으로써 프로세서 외부로 어떤 정보도 유출되지 않으며, 기능상의 변경이 필요할 경우 ROM 프로그램만 변경하면 되므로 하드웨어를 다시 설계할 필요가 없다.

표 7 H/W IP에 따른 gate count 및 성능비교표[8]
Table 7 Gate count and performance according to H/W IP

Crypt Accelerator	Gate count	Throughput
DES/3DES	16.6KGates @ 190MHz	DES(16bits/cycle), 3DES(5.3bits/cycle)
AES	35.5KGates @ 150MHz	1.78 bits/cycle with 128-bit key
MD5 and SHA-1	23.0KGates @ 250MHz	MD5(7.8 bits/cycle), SHA-1(6.3 bits/cycle)

TRNG	5.3KGates @ 433MHz	0.19 bits/cycle
Public-Key (RSA, ECC)	236KGates @ 215MHz	218,211 cycles for RSA-1024
	137KGates @ 220MHz	395,121 cycles for RSA-1024
	55KGates @ 240MHz	1,559,822 cycles for RSA-1024
	24KGates @ 240MHz	3,919,167 cycles for RSA-1024

7. 결 론

본 논문에서는 모바일 환경의 개인용 휴대 단말기에 널리 사용되는 애플리케이션 프로세서의 JTAG 포트를 보호하기 위한 기본 시스템 구조와 방법에 대해서 제안하였다. 컴퓨터의 강력한 계산 능력을 사용할 수 없도록 애플리케이션 프로세서 부팅시의 해쉬 값을 확인하는 방법을 사용자 인증 방법으로 사용하여 160-bit 해쉬 함수 사용 시 보안 강도를 2^{160} x 0.6초를 더 향상하였으며, 효율적인 하드웨어 자원의 활용을 위해 종래의 JTAG 포트 보안을 위한 전용의 하드웨어 공개키 알고리즘이 아닌, 응용 프로그램용으로 제작된 하드웨어 비밀키 알고리즘을 공유할 수 있는 사용한 권한 인증 방법을 제안하였다. 또한, 제안된 해쉬 값을 확인하는 사용자 인증 방법을 JTAG 포트 사용 권한 제어뿐만이 아닌, 애플리케이션 프로세서 내부의 하드웨어 모듈의 접근 제어 방법에 사용하는 방법과 효율적인 프로세서 개발을 위해서 전용 하드웨어가 아닌 내부 ROM 프로그래밍과 UART 또는 USB 포트 인터페이스를 이용한 개발 방법론을 새로 제안하였다. 제안된 시스템 구조를 통해서 신뢰할 수 있는 디버깅 방법과 중요 정보를 보호할 수 있으며, 프로세서를 적은 인적, 물적 자원으로 구현 가능할 수 있도록 하며, 프로세서를 보호하는데 있어서 기능성과 확장성 면에서 유연한 방법론을 제공한다. JTAG 디버깅 기능 보안과 안전한 부팅 방법을 통하여 시스템 부팅까지 안전하게 프로세서를 보호할 수 있으며, 프로세서를 사용하는 제품 개발자에게 다양한 응용분야의 정보 보호 제품을 신속히 개발할 수 있는 가능성과 정보 보호를 위한 신뢰성 있는 제품을 만들 수 있는 기반을 제공할 수 있다. 향후 연구 목표로는 제안하는 보안 시스템을 구현하여 기존의 보안 기법과 성능을 비교해 볼 예정이다.

참 고 문 헌

[1] R.F. Buskey, B.B. Frosik, "Protected JTAG", Proceedings of the 2006 International Conference Workshops on Parallel Processing, pp. 405-414, 2006.

[2] ARM, Designing with TrustZone - Hardware Requirements, Retrieved August, 2007, from http://www.arm.com/pdfs/TrustZone_Hardware_Requirements.pdf

[3] B. Schneier, Applied cryptography, 1996.

[4] 원동호, 현대 암호학, 도서출판 그린, 2006.

[5] A. Ashkenazi, Security Features in the i.MX31 and i.MX31L Multimedia Applications Processors, Retrieved August, 2007, from http://www.freescale.com/files/32bit/doc/white_paper/IMX31SECURITYWP.pdf

[6] J. Grand, "Practical Secure Hardware Design for Embedded Systems", presented at the Embedded Systems Conference, San Francisco, California, March 29, 2004.

[7] IEEE Standard Test Access Port and Boundary-Scan Architecture, IEEE Standard 1149.1, 2001.

[8] Proven Security IP for Next-Generation SOCs, Retrieved November, 2007, from http://www.safenet-inc.com/solutions/dev/intProp.asp.

[9] Am29LV800D datasheet, Retrieved November, 2007, from http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/Am29LV800D_00_A4_E.pdf

[10] 주학수, 주홍돈, 김승주, "고속 암호연산 프로세서 개발현황", 한국정보보호학회지, 1598-3978, 제12권3호, pp. 48-56, 2002

저 자 소 개

임 민 수 (林 敏 洙)

1973년 3월 24일생. 1997년 서경대학교 컴퓨터과학과 졸업. 2000년~2001년 실리콤텍(주) 주임연구원, 2007년~현재 성균관대학교 대학원 반도체디스플레이공학과 석사과정, 2001년~현재 삼성전자 SOC개발실 책임연구원

Tel : 031-209-5715
Fax : 031-209-0837
E-mail : minsoo.lim@samsung.com

박 봉 일 (朴 琫 一)

1972년 2월 28일생, 1994년 한국과학기술원 전기 및 전자공학과 졸업. 1996년 동 대학 전기 및 전자공학과(석사). 2002년 동 대학 전자 전산학과(공박). 2002년~2004년 (주)Dynalith 책임연구원. 2004년~현재 삼성전자 SOC개발실 책임연구원.

Tel : 031-209-4165
Fax : 031-209-0837
E-mail : bongil.park@samsung.com

원 동 호 (元 東 豪)

1949년 9월 23일생. 1976년~1988년 성균관대학교 전자공학과(학사, 석사, 박사). 1978년~1980년 한국전자통신 연구원 전임연구원. 1985년~1986년 일본 동경공업대 객원연구원. 1988년~ 2003년 성균관대학교 교학처장, 전기전자 및 컴퓨터공학부장, 정보통신 대학원장, 정보통신기술연구소장, 연구처장. 1996년~1998년 국무총리실 정보화추진위원회 자문위원. 2002년~ 2003년 한국정보보호학회장. 현재 성균관대학교 정보통신공학부 교수, 한국정보보호학회 명예회장, 정보통신부지정 정보보호 인증기술연구센터 센터장

Tel : 031-290-7107

Fax : 031-290-7686

E-mail : dhwon@security.re.kr