情報保護學會論文誌
第12券 第4號, 2002. 8

소프트웨어 불법복제방지를 위한 보안칩*

오 명 신**, 한 승 조***

The Secure Chip for Software Illegal Copy Protection

Myung-shin Oh**, Seung-jo Han***

요 약

현재는 유형적인 물질의 가치보다 무형적인 정보의 가치가 중요시되고 있는 시대이다. 특히 하드웨어보다는 소프트웨어 제품이 훨씬 급성장하고 있지만 소프트웨어 불법 복제는 정보화시대의 가장 큰 역기능으로 이슈화 되어있다. 그러나 현재 상용화되고 있는 소프트웨어 불법복제방지제품(락)들은 복제방지에 대한 강도가 약하기 때문에 쉽게 락이 크랙되어 복제방지 기능을 할 수 없는 것들이 대부분이다. 논자는 [1,2,3]에서 DES 암호알고리즘을 대체 할 수 있는 112비트 키 길이를 갖는 96비트 블록 Cipher를 제안한 바 있으며, [3.4]에서 칩으로 하였다. 따라서 본 논문은 [1,2,3]에서 제안한 96비트 블록 Cipher와 복제방지에 필요한 보안모듈을 ASIC화하여 소프트웨어 복제방지를 위한 전용 보안칩을 설계 및 구현하며, 보안칩과 연동하여 동작되는 자동블록보호기법을 설계한다.

ABSTRACT

Software has been developed very fast as information has become important value. Illegal software copy has been the main problem of developing software business. Recently used protecting lock system for software copy has not guaranteed perfectly from easily cracked-defense system. This paper, therefore, proposes 96-bit block cipher with 112-bit length to replace a DES(Data Encryption Standard) algorithm. Security chip by ASIC(Application Specific Integrated Circuit) security module is presented for software copy protection. Then, an auto block protecting algorithm is designed for the security chip. Finally, controlling driver and library are built for the security chip.

Keyword : *crypto-chip*

Ⅰ. 서 론

인터넷의 보급과 하드웨어의 고속 고용량 화에 힘입어 소프트웨어의 개발 능력은 이제 국가적인 경제가치의 척도가 되었으며, 무공해 고부가가치 산업 및 고용창출 효과에 의해 주요 선진국들에서는 국가적인 지원 아래 하이테크 고급인력들을 경쟁적으로 유치하여 소프트웨어 개발 경쟁력을 높이고 있는 실정이다. 현재 전 세계적으로 소프트웨어 시장은 수십조 달러에 달하고 있으나 유통되는 소프트웨어의 38%가 불법복제품으로 조사되었고 이에 대한 복제방지 대책도 다각도로 연구되고 있다. 실제 복제방지에 들어가는 비용은 소프트웨어 구입비용의 10%에 달하고 있고 그 만큼 복제방지가 미흡할 경우 복제방지 투자비용의 두 배가 넘는 손실을 입게 되는 것이다. 따라서 차세대 복제방지 락은 보다 더 싼

* 본 논문은 '98년 조선대학교 교내 연구지원에 의해 연구되었음.
** 전남대학교병원 의공학과(ohm99@hitel.net)
*** 조선대학교 전자정보통신공학부(sjbhan@mail.chosun.ac.kr)

가격에 완벽에 가까운 복제방지 비율과 보다 더 유연성 있는 응용능력을 요구하고 있다.

소프트웨어 복제방지 락은 소프트웨어나 하드웨어로 구현하게 되는데, 락을 탑재한 소프트웨어는 락과의 정해진 방법으로 통신(lock checking)을 함으로써 락의 유무나 오류 등을 결정하게 된다. 소프트웨어로 구현된 복제방지보다는 하드웨어로 제작된 락의 강도가 높다. 그러나 하드웨어 락은 소프트웨어 제품의 가격에 큰 영향을 끼칠 정도로 고가이기 때문에 고가의 소프트웨어에만 적용되고 있는 실정이다. 또한 기존의 상용 하드웨어 락은 사용상의 불편함과 기능의 한정성을 가지고 있어서 소프트웨어 불법복제 방지측면에서 보면 개발업자들에게는 환영을 받지만 소프트웨어를 사용하는 최종사용자 입장에서는 외면 받기 십상이었다. 따라서 소프트웨어 복제방지를 위해서는 보안 강도가 높고 저가이며, 사용이 편리하고 다기능을 갖는 하드웨어 락이 필요한 실정이며, 개발비용에 포함되는 락의 비용보다는 역기능 방지에 강력한 암호알고리즘을 이용한 견고한 락의 개발이 중요하다.[4]

따라서 본 논문에서는 [1,2,3]에서 제안한 96비트 블록 암호알고리즘과 복제방지에 관련한 보안모듈을 ASIC화하여 소프트웨어 복제방지를 위한 전용 보안칩을 설계 및 구현하며, 보안칩과 연동하여 동작되는 자동블록보호기법을 설계한다.

II. 소프트웨어 복제 이론

현재 국내에서 개발 시판 중인 락은 하드웨어 락이 대부분을 차지하고 있다. 그러나 하드웨어 락은 락의 제조비용에 대비 복제방지 능력이 소프트웨어 락의 복제방지 능력과 별로 차이가 없기 때문에 정식 프로그램을 구매하는 것보다 무단복제를 많이 사용하는 나라에서는 소프트웨어 락이나 기능제한(Evolution Version, Trial Version)을 적용한 사용자 등록제(User Registration) 등의 방법을 이용하여 사후관리의 차별화로 구매를 유도하는 경향이 두드러지고 있다.[5]

국내외의 프로텍션 제조회사들도 제조비용 대비 복제방지 능력을 높이기 위한 방법을 여러 가지로 강구하고 있다. 그 중 한 가지가 실행파일이나 데이터 파일을 암호화하고 이 암호화에 사용된 키를 락에서 가져오는 방식을 사용하는 것이다. 이 방식은 NSTL(National Software Testing Laboratories)에서 우수 평가를 받은 방식이기는 하지만 락 체크의 빈도에 프로그램의 수행시간이 많은 영향을 받는 것이 단점으로 지적되고 있다. 또한 보안등급이나 유연성 면에서 높이 평가를 함에도 불구하고 소프트웨어에 의한 제어구조이기 때문에 기계어 수준의 디버깅으로 불법복제가 가능하다는 것이 단점이다.[6]

현재 보편화되어 있는 2가지의 락의 구조를 살펴보면, 첫 번째 구조는 단순히 물리적인 락이 존재하는 것만을 검사하는 것으로써 락으로 전송하는 데이터나 락으로부터 반환되는 데이터의 형태가 불변인 락의 구조를 갖는 것이 있고, 두 번째 구조는 초기의 락이 쉽게 에뮬레이션이 가능한 점을 보완하기 위하여 정적인 제어가 아닌 동적인 제어를 하도록 하는 데이터 형태가 변형되는 락의 구조가 있다.[4,7]

다음은 동작 방식에 따른 특징을 기술한 것이다.

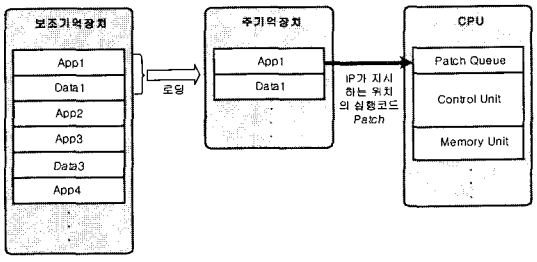
S/W Lock은 크게 하드웨어의 물리적인 특성을 이용한 경우, 사용자로부터 정식사용자임을 직접 등록하게 하는 사용자 등록제 그리고 실행코드에 Time Stamp 등을 삽입하는 방법으로 나뉜다.[8,9]

H/W Lock은 크게 가장 고전적인 방법인 락의 유무검사 방법, 락에 정보를 저장해 놓고 실행시 그 정보를 읽어오는 방법 그리고 락의 연산기능을 이용한 원본 실행코드의 변형으로 나뉜다.[6,10]

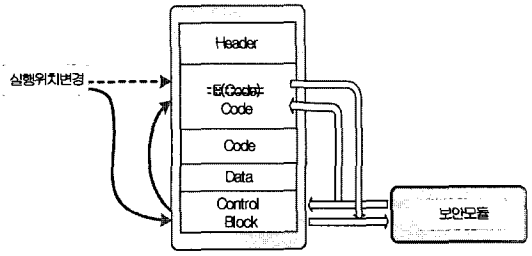
III. 복제방지기법의 설계

3.1 실행코드 보호기법 설계

컴퓨터는 기억장치가 보조기억장치와 주기억장치로 분리되어 존재한다. 평상시 보조기억장치에 저장되어 있는 프로그램 및 데이터들은 그 실행 시점에 주기억장치로 로딩된다. 주기억장치에 로딩된 프로그램은 IP가 지시하는 위치의 실행코드를 CPU 내의 Queue에 넣고 분석 처리한다. [그림 1]은 그 과정을 나타낸 것이다.[8,11]



[그림 1] App.의 메모리 로딩 및 실행과정



〔그림 2〕 자동 블록 보호된 프로그램의 실행도

본 논문에서 제안한 자동블록보호기법은 보조기억장치에 암호화 된 상태로 저장되어 있다가 프로그램이 실행을 위해 로딩될 때 암호화 되어있는 실행프로그램을 복호화 하여 실행하는 기술이다. 실행 종료 후에는 메모리에서 제거(Release)되어 복호화된 코드가 제거된다. 이때 암복호화 되는 코드는 실행코드 전체가 대상이 될 수 있다.

본 설계는 프로그램 동적 디코딩 기술과 함께 맞물려 동작하게 되며, 보안모듈을 통한 복호화 과정 없이는 실행자체가 안될 뿐만 아니라, 실행코드 원본을 노출시키지 않도록 설계한다.

자동블록보호기법을 이용한 보호된 프로그램의 실행도는 〔그림 2〕와 같다.

자동블록보호기법의 설계에서는 다음의 3단계 과정이 필요하다.

〔1단계〕 프로그램의 헤더를 변경하여 실행 시작위치 등의 필요정보를 기록한다.
〔2단계〕 컨트롤 블록을 추가한다.
〔3단계〕 원 프로그램의 코드영역을 암호화한다.

이러한 설계를 위해 필요한 세부 기법에는 PE (Portable Executable) File 분석 및 변형기법, 주소 독립적 Control Block 제작 기법, 코드의 암호화 기법 등이다.

PE File 분석 및 변형기법은 윈도우즈의 기본 실행파일 포맷인 PE file을 분석함으로써 본 설계에서 제어블록을 추가하는 등의 필수적인 기법이다. 이때 제어블록을 대상 실행파일에 붙인 후 정상적으로 동작하게 하려면 기계어 단계의 어드레스가 자동 재계산 가능한 동적 구조여야 하는데 이때 필요한 기법이 주소 독립적 Control Block 제작 기법이며, 대상 실행파일의 실행 코드영역을 암호화 해놓았다가 자동 복호화 하는 기법이 코드의 암호화 기법에 해당한다.

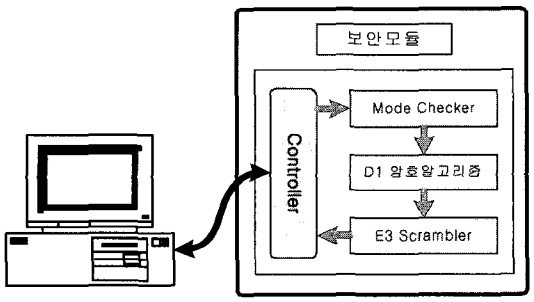
3.2 동작구조 설계

본 논문에서는 소프트웨어 복제방지 기법 중 하드웨어가 담당해야 하는 기능으로 크게 3가지를 포함하였다. 디버깅 여부를 판단할 수 있는 Mode Checker, 데이터의 암복호화를 담당할 D1 암호알고리즘, 디버거에 의한 모니터링을 방지하기 위한 E3 Scrambler가 그것이다.

〔그림 3〕과 같이 먼저 PC로부터 보안칩에 의한 실행코드의 복호화 요청시 Hash 코드를 포함하여 입력 복호화 요청을 하게 되며 Mode Checker가 1차로 위변조 여부를 파악한다. 그리고 등록되어 있는 인증키에 의해 복호화 요청된 실행코드를 복호화 한 후 스크램블러를 이용하여 재차 암호화하여 PC로 전송한다. 이때 스크램블러에 사용되는 키는 Random Number를 이용하여 조합된 키가 사용되며, 이 값은 PC와 Sync된다.

그 결과 Linear cryptoanalysis에 대한 Mode Checker의 방어와 통계분석 Hacking에 대한 Scrambler의 방어가 이루어진다.

그 이외에 Controller에 의해 PC와 time stamp를 교환함으로써 Softice 등과 같은 debugging 툴로부터 강하게 저항할 수 있다.

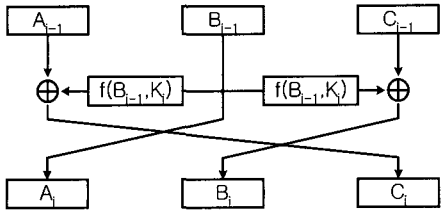


〔그림 3〕 소프트웨어 보호를 위한 동작 구조

3.3 암호알고리즘의 설계

DES를 확장한 KSE96은 16라운드의 각 서브블록동안 서로 다른 f 함수를 적용하도록 하여 암호학적 강도를 높이는 방법을 이용한다. 이것을 위하여 96비트 한 블록의 데이터를 32비트의 3개의 서브블록으로 분할함으로써 비대칭으로 만든다.[1,2,3]

$$B_i = C_{i-1} \oplus f(B_{i-1}, K_i)$$
$$C_i = A_{i-1} \oplus f(B_{i-1}, K_i) \quad (1)$$



〔그림 4〕 KSE96의 구조 (1 라운드)

이것을 초기조건 $C_i = B_{i-1}$와 같이 단순화시켜 적용하여 보면, 완전히 같은 특성으로 반대편에 위치함을 알 수 있다. 식 (1)에 의한 조합은 대칭적이며, 따라서 KSE96 알고리즘에서 〔그림 3〕과 같이 적용하였다. 암호화와 복호화는 각각의 관계를 서로 비교함으로써 동치임을 확인할 수 있다. 〔그림 4〕로부터 암호화 공식을 유도하면 다음과 같다.

$$\begin{aligned} A_i &= B_{i-1} \\ B_i &= C_{i-1} \oplus f(B_{i-1}, K_i) \\ C_i &= A_{i-1} \oplus f(B_{i-1}, K_i) \end{aligned} \quad (2)$$

식 (2)로부터, XOR 특성과 $f(B_{i-1}, K_i) = f(A_i, K_i)$를 이용하면 다음의 복호화 식을 얻는다.

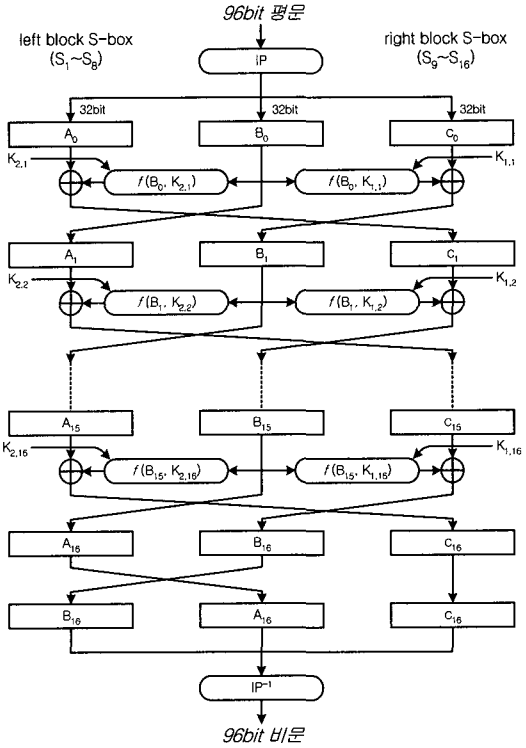
$$\begin{aligned} B_{i-1} &= A_i \\ C_{i-1} &= B_i \oplus f(A_i, K_i) \\ A_{i-1} &= C_i \oplus f(A_i, K_i) \end{aligned} \quad (3)$$

A와 B를 서로 바꾸면 다음과 같은 관계를 얻을 수 있다.

$$\begin{aligned} A_{i-1} &= B_i \\ B_{i-1} &= C_i \oplus f(B_i, K_i) \\ C_{i-1} &= A_i \oplus f(B_i, K_i) \end{aligned} \quad (4)$$

이 식을 이용한 KSE96의 전체 구조를 도식화하면 〔그림 5〕와 같다.

KSE96은 differential cryptoanalysis에 보다 견고하게 하기 위하여 f 함수의 수를 증가시켰다. 증가된 수치는 3개의 서브블록과는 차이가 있다. 본래의 DES에 사용된 f 함수는 R_0에서 L_{16}까지 그리고 L_0에서 R_{16}까지 변환되는 과정에서 8번 반복된다. 반면에 개선된 DES의 f 함수는 A_0에서 C_{16}까지 처리될 때 11번 반복된다. 이것은 같은 E(Bit-selection table), P(Permutation table) 그리고 S-box



〔그림 5〕 KSE96의 암호화 알고리즘

에서 KSE96이 본래의 DES에 비해 differential cryptoanalysis에 보다 더 견고함을 보여준다.

S-Box의 설계는 알고리즘의 강도에 결정적인 영향을 미친다. DES에 있어서 비선형적인 모듈러-2 알고리즘이 단지 하나의 단계에 불과한 반면 실제적인 알고리즘의 암호강도는 S-Box에 의존하고 있다.[12] S-Box의 엔트리를 개선하기 위해 DES는 잘 분석된 엔트리를 사용한다. 이 분석은 평균비트확률($P_{i,j}$)과 그 상호계수[12]에 의해 나타나는 SAC(Strict Avalanche Criterion)[13]에 근거하고 있다.

3.4 스크램블러의 설계

본 논문에서는 스크램블러의 암호강도 보장을 위해 DES-CBC를 사용한다. 단, 키를 사용함에 있어 소프트웨어 복호화 시작시 최초 개시 이전에 PC로부터 Random Number를 부여받아 이를 이용하여 스크램블 키를 생성함으로써 소프트웨어의 매 실행시 다른 스크램블 데이터 결과를 수신하게 된다.

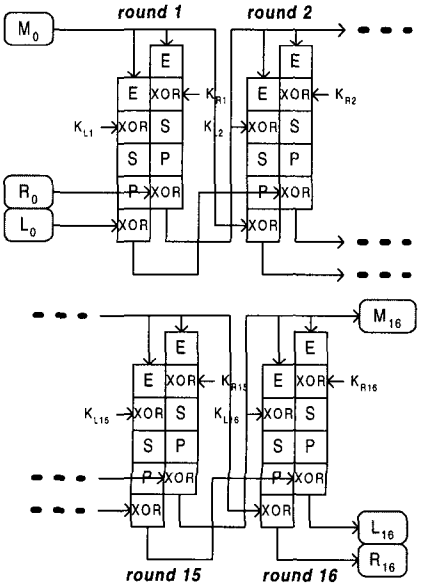
RN = D2KR(C2)
C3 = E3RN(P)

Ⅳ. 복제방지전용 보안칩 설계

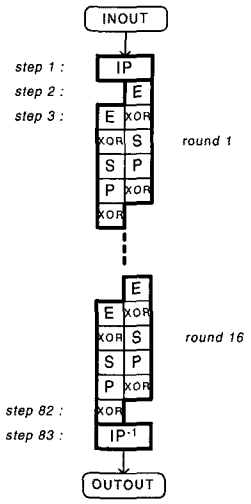
4.1 96비트 블록 암호 알고리즘

[1,2,3]에서 제안한 96비트 블록암호알고리즘을 복제방지를 위한 암호알고리즘으로 사용하였으며 이 암호알고리즘을 KSE96이라 칭한다. KSE96은 크게 초기치환(IP), 16회 반복 라운드 그리고 역초기치환(IP^{-1})으로 나눌 수 있으며, 각 라운드는 좌우측이 각각 E, XOR, S, P 그리고 XOR의 5단계로 구성되어 있다. 또한 96비트의 데이터 입력을 갖고 있으며, 3개의 32비트 블록들로 분할되어 16회 단일 라운드를 수행하게 된다. 이때 좌측 서브프로세싱블록과 우측 서브프로세싱블록은 서로 독립적으로 동작한다. KSE96의 각 단계를 펼쳐놓게 되면 [그림 6]과 같은 구조를 이룬다.

좌측 서브프로세싱블록과 우측 서브프로세싱블록은 서로 독립적으로 동작하므로 우측보다 좌측을 1스텝씩 지연시킴으로써 치환이나 대치연산에서 사용되는 테이블을 32개에서 16개로 줄일 수 있다. 각 라운드는 좌우측이 각각 E, XOR, S, P 그리고 XOR 등 5단계로 구성되어 있으나 1단계 지연을 합하여 6개의 단계를 갖는다. 이때 16라운드를 전부 선형으로 연결하면 각각의 단계에서 1단계는 다음 단계와 중복되어도 되기 때문에 16라운드 전체는 5×16+1 개의 단계를 갖게 되고 여기에 IP와 IP^{-1}을 추가하여 [그림 7]과 같이 총 83단계의 파이프라인 구조를 가지게 된다.

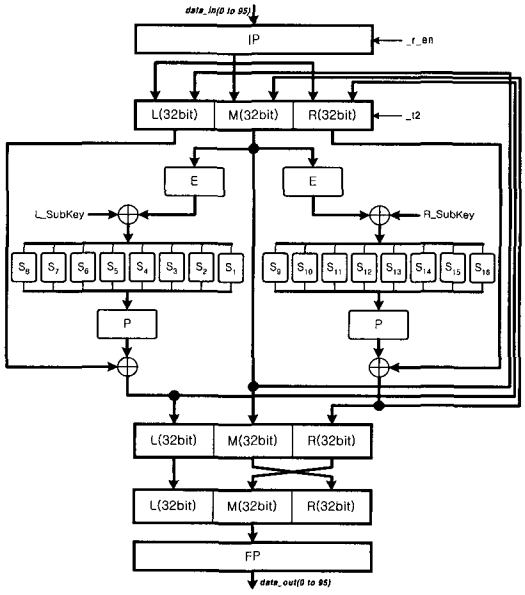


[그림 6] KSE96의 선형구조



[그림 7] KSE96의 파이프라인구조

그러나 16개의 라운드를 파이프라인 구조로 설계하게 되면 16개의 E, S 그리고 P 테이블이 필요하게 되어 칩으로 구현함에 있어서 내부 네트와 면적에서 제작이 난해하다. 따라서 본 개발에서는 16개의 라운드를 단일라운드 반복법으로 수정 설계하고 내부 컨트롤 블록에 의해 제어를 할 수 있는 구조로 재설계하였다. 단일 라운드 반복법은 각각 2개씩의 E, S 그리고 P 테이블이 소요되며, 좌우측 서브프로세싱 블록은 테이블 공유는 하지 않는다. 단일라운드 반복법을 이용한 KSE96 블록도는 [그림 8]과 같다.



[그림 8] KSE96 라운드 블록의 블록도

4.2 칩의 구조

본 논문에서 설계한 칩은 크게 5가지의 기능별 블록을 갖는다.

첫째, KSE96 암호화 블록을 이용하여 실행코드를 보호한다. 보조저장매체에 저장되어 있는 실행코드는 프로그램의 개발자에 의해 미리 암호화되어 있으며 최종사용자는 락에 내장되어 있는 복호화 블록에 의해 프로그램의 실행단계에서 복호화 작업을 수행함으로써 인증된 사용을 하게 된다. 이를 위하여 핵심칩에는 KSE96 알고리즘을 내장한 Block Cipher를 갖는다.

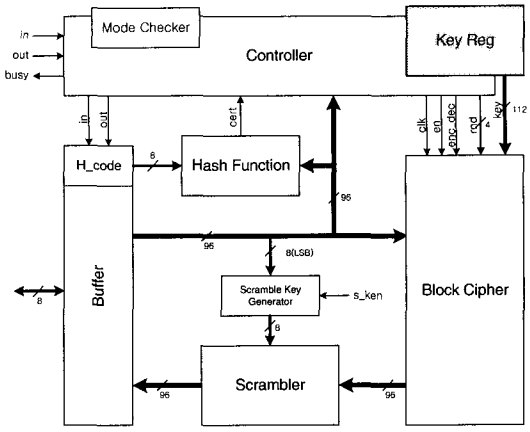
둘째, 실행코드의 변경을 검사한다. 복제방지 된 프로그램이 실행을 위하여 메모리에 적재되어 있을 때 가장 취약한 부분은 적재된 프로그램의 변경이며, 이것을 방지하기 위한 방법으로 실행코드의 변경을 감시하기 위한 코드를 삽입하는 방법이 사용된다. 본 논문에서 코드의 변경감시는 Hash 함수에 의한 출력 코드의 변경 검사기법을 이용하였다. 복제방지 된 프로그램은 보조 메모리에 적재될 때에 저장된 코드에 대한 Hash값을 함께 저장하고 있게 되며 핵심칩 내에서 보조저장매체에 존재하는 Hash값과 칩 내부에서 계산된 Hash값을 비교 수행함으로써 코드의 변경 유무를 검사하게 된다.

셋째, Block Cipher에 의해 복호화 된 실행코드는 PC로 전송되어지기 전에 다시 암호화되어진다. 이는 락으로부터 반환되는 값이 일정할 경우 락을 하드웨어적으로 에뮬레이션 하는 것이 가능하기 때문이며, 이를 방지하기 위한 방법으로 Scrambler를 사용하였다.

넷째, 락을 제어하는 소프트웨어를 보호하기 위한 PC의 실행 모드 검사기를 칩에 포함한다. 모든 프로그램은 디버거에 의해 실행 시점에서 노출이 되어지며, 실행코드의 변경도 가능해지게 된다. 따라서 이를 방지하기 위한 디버거 검사기를 칩에 내장함으로써 디버거에 의한 노출을 방지할 수 있다.

락 제어용 프로그램은 복제방지 된 프로그램의 실행시 PC의 동작환경 중 디버거와 관계된 정보를 락으로 전송하게 되며, 칩 내부의 모드 검사기는 이를 검사하여 디버거가 존재하지 않는 환경에서만 락이 동작을 하게 제어한다.

다섯째, 외부 입출력을 위한 버퍼를 갖는다. 설계된 칩은 외부의 8비트 입출력 포트를 갖고 있다. KSE96 암호알고리즘은 96비트 블록이며 이를 위하여 칩의 내부는 96비트로 구성되어 있다. 따라서 외부와의 입출력을 위한 8 to 96비트 컨버트를 위한 버퍼가 필요하다.



〔그림 9〕 핵심칩의 블록도

이와 같은 5개의 블록과 이를 제어하기 위한 controller 그리고 key register block이 〔그림 9〕에 보여지고 있다.

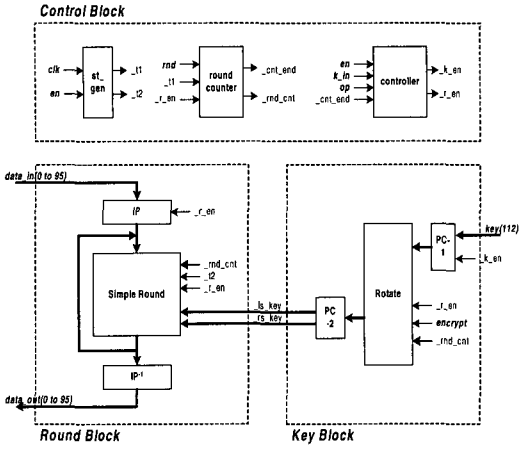
4.3 칩의 설계

4.3.1 KSE96

KSE96은 Round Block, Key Block 그리고 Control Block 등 크게 3부분으로 나누어진다. Round Block은 KSE96 알고리즘 중 1개의 라운드를 가지고 있으며 2분주 된 Clk에 의해 16회 반복 수행하게 된다. Key Block은 16라운드에서 사용될 Kl과 Kr을 생성하는 블록으로 데이터의 입력 전에 미리 키 레지스터로 입력받게 되며 새로운 키가 입력되기 전까지 계속 유지된다. 입력된 키는 라운드블록의 반복에 맞추어 서브키를 생성해서 출력해 준다. Control Block은 Key Block과 Round Block이 유기적인 동작을 할 수 있도록 외부 제어선의 제어하에 내부 각 블록들을 세부적으로 제어하는 블록이며, 데이터의 반복에 따른 현재의 루프를 카운트하는 기능 등을 갖는다.

〔그림 10〕은 KSE96 알고리즘의 블록도이다.

라운드 블록은 16라운드의 KSE96 알고리즘을 실질적으로 처리하는 블록으로서 데이터는 96비트로 입력받아 96비트로 출력한다. _r_en이 상승에지일 때 초기순열(IP)은 데이터 버스로부터 값을 입력받아 초기순열을 수행한 후 Round Processor의 입

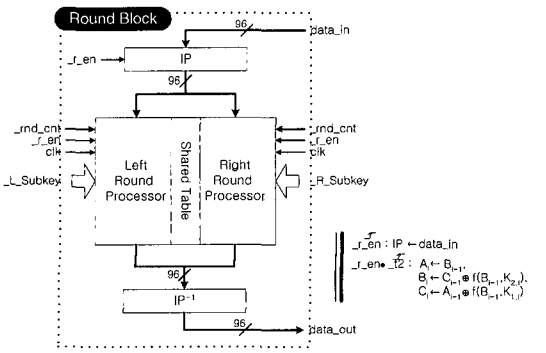


〔그림 10〕 KSE96 알고리즘의 블록도

력으로 보낸다. Round Processor는 좌측과 우측으로 분리되어 있으며 Processor 내에서 사용되는 각종 테이블을 서로 공유한다. Round Processor는 단일 라운드 블록만으로 구성되어 있으며 16회 반복수행을 하게 된다. 반복라운드의 수행시 좌우측 각각의 Round Processor에는 현재의 수행 라운드 횟수에 맞는 서브키가 입력된다. 서브키는 키 블록으로부터 전송되며, 각각 48비트로 구성되어 있으며, 〔그림 11〕은 라운드블록의 블록도이다.

Round Processor는 E(Bit-selection table), mod 2 (XOR), S-box, P(Permutation table) 그리고 mod 2 (XOR)의 5단계로 구성되어 있으며, _r_en이 액티브일 때 클록(clk)의 상승에지에서 제공된 서브키를 이용하여 5단계 1라운드씩 암호화를 수행한다. 5단계의 각 라운드가 16회 반복 수행된 데이터는 역초기순열(IP^{-1})을 거쳐 96비트의 데이터 버스를 통해 출력된다.

키 블록은 입력된 128비트의 키를 PC-1을 통해

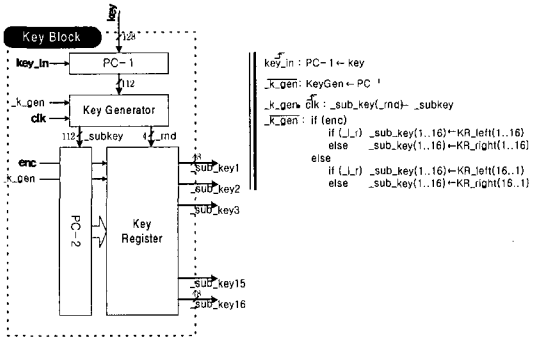


〔그림 11〕 라운드블록의 블록도

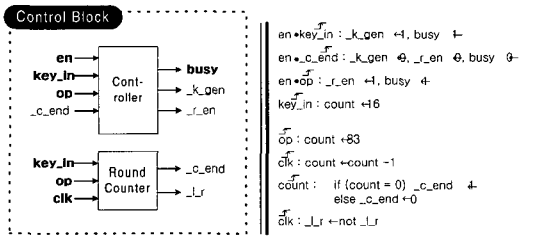
패리티 비트를 제거하여 112비트로 줄인 후 키 스케줄에 따라 라운드별로 좌측 쉬프트를 수행한 후 PC-2를 거쳐서 각 라운드별 서브키를 생성한다. 키 레지스터는 16라운드에 해당하는 좌측 서브키와 우측 서브키를 보관하고 있게 된다. 키의 입력은 외부 제어선인 key_in에 의해 이루어진다. enc 제어시그널은 키가 출력될 때 현재의 동작이 암호화에 따른 정순 출력인지 복호화 처리에 따른 역순 출력인지를 지시하며, 키블록은 〔그림 12〕와 같다.

라운드 블록에서 사용되는 키는 다음의 새로운 키가 입력될 때까지 내부 레지스터에 유지되며, 키 제너레이터는 _k_gen이 액티브일 때 클록(clk)에 따라 내부 카운터가 1부터 16까지 증가하면서 서브키를 생성하여 키 레지스터로 전송하게 된다.

컨트롤 블록은 크게 2가지의 기능을 갖는다. 첫째, 외부의 제어선으로부터 입력을 받아 전체 블록들을 제어하는 기능과, 둘째, 입력된 데이터가 몇 번째 단계를 수행 중인지를 카운트하여 최종 데이터가 중간단계에 잔류하지 않도록 하는 기능으로 분류되며, 컨트롤 블록의 블록도는 〔그림 13〕과 같다.



〔그림 12〕 키블록의 블록도



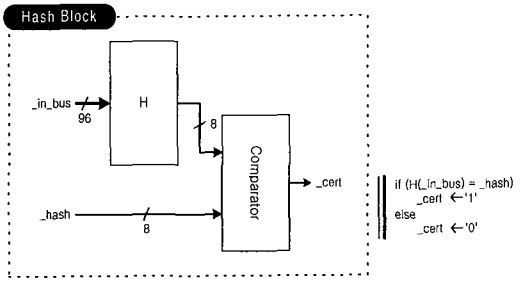
〔그림 13〕 컨트롤 블록의 블록도

4.3.2 Hash Block

Hash 블록은 암호화되어 있던 실행코드의 불법적인 변경을 감시하기 위한 블록으로 _in_bus에 입력

된 코드의 H 값과 _hash에 입력된 값을 비교한 후 같으면 _cert에 '1'을 출력한다.

〔그림 14〕는 Hash Block의 블록도이다.

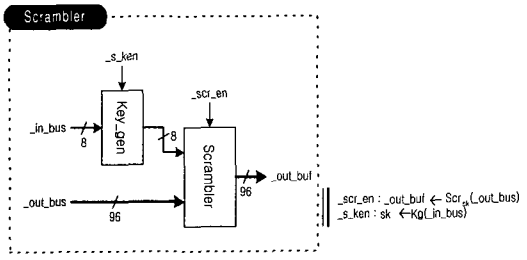


〔그림 14〕 Hash Block의 블록도

4.3.3 Scrambler

Block cipher에서 암호화되어진 데이터는 PC로 전송되어지기 전에 Scramble 과정을 거치게 된다. 키는 Key Generator에 의해 생성되며, _s_ken이 active될 때 _in_bus를 입력으로 하여 SK(Scramble Key)를 생성한다. 생성된 SK는 _scr_en 이 active 될 때 _out_bus의 데이터를 Scramble 하는데 사용되어지며 _out_buf의 출력을 얻게 된다.

〔그림 15〕는 Scrambler의 블록도이다.

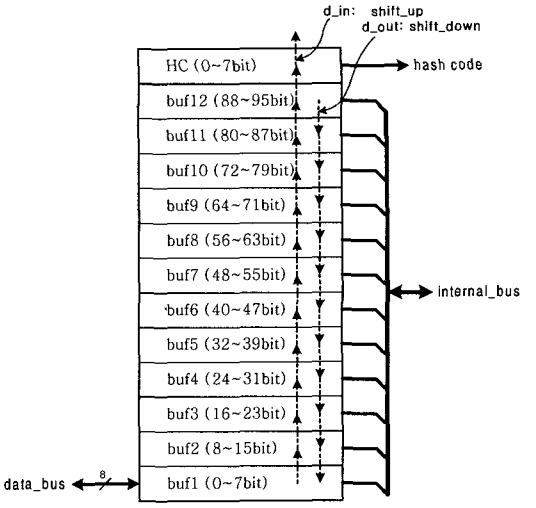


〔그림 15〕 Scrambler의 블록도

4.3.4 Buffer

설계된 칩은 내부 96비트, 외부 8비트의 입출력을 갖는다. 따라서 데이터의 입출력을 위한 버퍼가 필요하며 8 to 96비트 컨버터 역할을 하게 된다. 버퍼는 hash code를 위한 8비트를 포함하여 13개의 8비트 버퍼 블록으로 나뉘어진다. 또한 외부 입출력 버스는 buf1과 연결되어 있고, 데이터의 입출력시 13개의 버퍼는 쉬프트를 수행함으로써 내부 버퍼를 채우게 된다. 설계된 버퍼는 _d_in의 상승에지에서 하위버퍼에서 상위버퍼로 쉬프트 되며, _d_out의 상승에지에서 하위버퍼로 쉬프트 된다.

〔그림 16〕은 버퍼의 블록도이다.



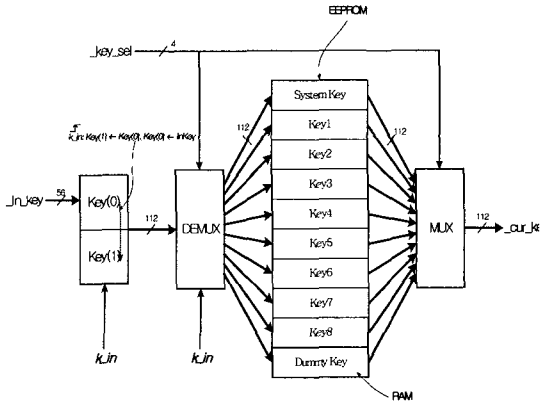
〔그림 16〕 Buffer의 블록도

4.3.5 Controller

Controller는 각각의 블록들의 유기적인 동작을 위한 제어를 담당하며, 그 외에 외부의 제어워드를 해석하고 키 레지스터를 포함하고 있으며, 디버그 모드를 감지하기 위한 모드 검사기를 갖고 있다.

키 레지스터는 총 10개의 키 공간을 내장하고 있으며, 그중 1개는 키의 고유한 ID를 갖게 되고, 8개는 외부에서 writing이 가능한 ROM으로 구성되며, 1개는 임시키를 위한 RAM의 형태를 갖는다. 키의 선택은 _key_sel에 의해 이루어지며 키의 입력과 출력에서 공통이다. 키의 입력은 _k_in의 상승에지에서 _In_key의 값을 입력으로 받게 되며, 키의 출력은 내부의 10개의 키 중 _key_sel에 의해 선택되어진 후 _cur_key로 출력한다.

키 레지스터의 블록도가 〔그림 17〕에 나타나 있다.



〔그림 17〕 Key Register의 블록도

4.3.6 입출력부

본 암호칩은 듀얼포트 구조로 설계되어야 하며 칩의 제어를 위하여 4개의 제어신호가 필요하다.

〔표 1〕은 제어신호 및 데이터선의 종류와 기능에 대한 설명이다.

내부의 버퍼는 최대 17바이트이며 최하위 8비트는 제어워드로 사용된다. 제어워드는 버퍼에 실린 16바이트의 데이터에 대한 해석에 사용된다.

〔표 1〕 고속동작 구조를 위해 요구되는 제어신호선

제어신호	기능	방향	기타
Done	Buffer Ready	I	High Active
Rst	Chip Initialize	I	High Active
WR	Write	I	Low Active
RD	Read	I	Low Active
DataIn [0:8]	Data Input bus	I	
DataOut [0:8]	Data Output bus	O	

4.3.7 제어워드(Control Word)

제어워드는 칩의 환경을 세팅하는 명령어 그룹과 암복호화를 수행하기 위한 명령어 그룹으로 나뉜다. 설계된 칩은 IBM PC 환경에서 동작할 경우 칩과의 통신내용을 디버거(Debugger)를 통하여 모니터를 할 수 있는 단점을 보완하기 위하여 디버거의 특징을 판단하여 칩의 동작유무를 결정하는 함수를 내장하고 있다. 또한 칩으로부터 되돌아오는 암복호화된 데이터의 분석을 방지하기 위하여 스크램블러(Scrambler)를 내장하고 있으며, 이 두 가지의 기능은 필요에 따라 ON/OFF 할 수 있다.

〔표 2〕 제어워드

	Control Word	Function	Op. Cycle
E	0x20	Internal Key의 생성	20
	0x08	External Key의 세팅	3
	0x10	Scramble Key의 세팅	3
	0x04	Module ID 반환	3
	0x02	Scrambler On/Off	1
	0x01	Debugger Detection	3
O	0xa0	Internal Key를 이용한 Encryption	83
	0x88	Internal Key를 이용한 Decryption	83
	0x90	External Key를 이용한 Encryption	83
	0x84	External Key를 이용한 Decryption	83

E : 환경설정을 위한 명령어 그룹
O : 암호 연산을 위한 명령어 그룹

〔표 2〕는 제어워드에 대한 설명과 수행에 필요한 cycle이다.

V. 실험 및 고찰

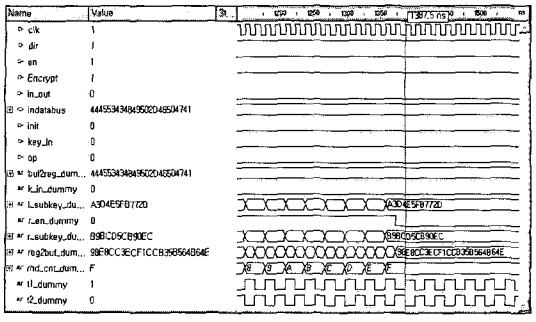
5.1 시뮬레이션

KSE96은 데이터의 입력을 96비트로 하고 112비트의 키를 필요로 하는 블록 암호화 알고리즘으로 암복호화 과정을 검증하기 위하여 다음과 같은 조건에서 시뮬레이션을 수행하였다.

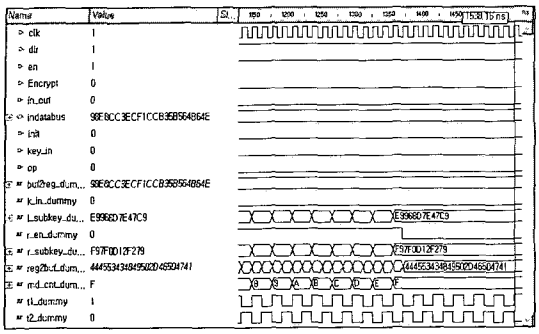
- Key : "InfoCommSystem"
  (49 6e 66 6f 43 6f 6d 6d 53 79 73 74 65 6d)
- Data : "DESCHIP-FPGA"
  (44 45 53 43 48 49 50 2d 46 50 47 41)
- Clock : 350 MHz
- Simulator : Active-VHDL

적용된 키는 고정으로 유지하고 입력데이터는 96비트를 83회 반복해서 입력하였다. 83개의 스테이지를 모두 거쳐 암호화된 출력데이터를 다시 복호화 과정의 입력으로 넣어 여기서 출력된 복호화 된 데이터가 원본과 일치함을 확인하였다. 암호화 및 복호화 시뮬레이션 결과는 〔그림 18〕, 〔그림 19〕와 같다.

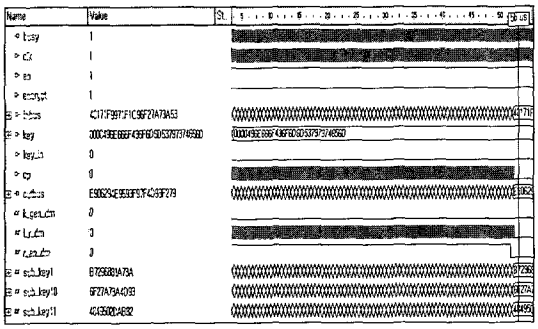
또한 대형 스트림 데이터에 대한 시뮬레이션 결과가 〔그림 20〕에 나타나 있다.



〔그림 18〕 KSE96의 암호화 시뮬레이션



〔그림 19〕 KSE96의 복호화 시뮬레이션



〔그림 20〕 Stream 데이터의 시뮬레이션

5.2 칩의 제작

Verilog-XL을 사용하여 Functional Simulation, Pre Simulation 그리고 Post Simulation을 거쳐 칩을 구현하였다. 구현된 칩은 Test Vector에 의해 92.9%의 Gate Toggle율로 80,000 step을 동작해 봄으로써 동작 피크시의 오류도 최소화하였다.

칩의 구현을 위하여 본 논문에서는 대만의 UMC 공정을 사용하였다. 제작된 칩은 0.45㎛ 공정의 Gate Array Rule을 따랐으며, 실 Gate 수가 26,728 gate였다. 칩의 설계단계에서 동작주파수의 평균을 33MHz로 설계하였으나, 최대 동작 주파수는 50MHz까지 오류 없이 동작하였다. 50MHz의 동작 주파수는 현재 보편화되고 있는 100MHz 이상의 주변 동작 주파수에 비해 낮으며, 이는 Design Rule을 Full Custom으로 하거나 현재의 0.18㎛ 공정 등 물리적인 요소에 의한 속도 향상으로 1G이상의 동작속도를 유도할 수 있다.

〔그림 21〕은 제작된 칩의 사진이다.

〔그림 21〕 구현된 보안칩사진

칩의 주요 스펙은 다음과 같다.

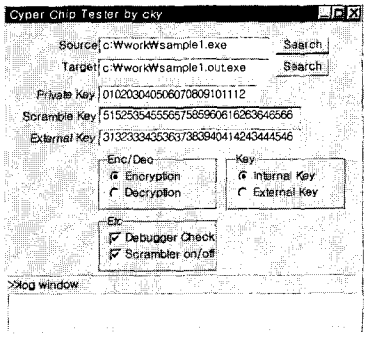
- Fab. : 대만UMC
- Package Type : 28TSOP
- Design Rule : 0.45㎛ G/A
- Count : 26,728 gate
- Op. Volt : -0.3~6.0V
- Est. Current : 2mA
- Temp. : -55~150℃
- Op. Frequency : 1~50MHz

5.3 검증

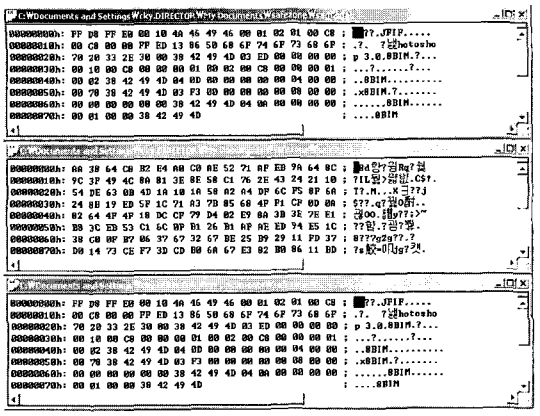
제작된 칩의 검증을 위하여 다음의 과정을 테스트할 수 있는 프로그램을 제작하였다.

- Key 관리
- 내부키의 생성
- 내부키를 이용한 데이터의 암·복호화
- 외부키의 입력 및 저장
- 외부키를 이용한 데이터의 암·복호화
- 디버거 체크 루틴 동작검증
- 스크램블러의 동작 검증

내부키는 칩내에 저장되어 있는 고유 ID와 외부에서 주어지는 Key를 조합하여 생성하여 놓기 때문에 직접 검증이 불가능하나 내부키를 이용한 데이터의 암호화 결과와 소프트웨어 시뮬레이션 암호화 결과를 비교하여 검증하였다. 디버거 체크 루틴의 검증은 실제로 시스템에 디버거를 띄운 상태에서 동작시킴으로써 칩의 반응을 모니터링 하였으며, Scrambler의 동작 검증은 제작된 칩으로부터 반환되는 스크램블 된 데이터를 소프트웨어적으로 디스크램블 해 봄으로써 오류 유무를 검사하였다.



〔그림 22〕 테스트 프로그램



〔그림 23〕 복제방지를 위한 실행파일과 암호화파일 결과 비교

〔그림 22〕는 소프트웨어 복제방지전용 보안칩에 대한 테스트프로그램의 결과이고, 〔그림 23〕은 테스트용 프로그램으로 복제방지를 위한 실행파일과 암호화로 락을 걸었을 때의 결과 비교이다.

Ⅵ. 결 론

인터넷이나 무선전화, FAX 등과 같이 전송매체에 의한 정보의 양은 급격히 증가하고 있으며, 프라이버시의 침해나 허가되지 않은 접근에 의한 공격도 급격히 증가하고 있다. 또한 정보의 노출뿐만 아니라 S/W 같은 무형 재산권에 대한 침해는 사회적 경제적 피해효과가 상당히 크다.

본 논문에서는 급속히 증가하는 프로그램 및 정보의 불법복제의 방지를 위한 전용 보안칩을 개발하기 위하여 보다 안정성 있고 빠른 속도의 암호화 알고리즘을 도출하고, 하드웨어로 제작하게 될 때 얻게 되는 장점을 최대한 고려하여 암호화부 및 콘트롤부 등을 FPGA로 테스트 후 ASIC으로 제작하였으며, 이를 구동하기 위해 S/W복제방지전용보안칩 사용을 위한 자동블록보호기법을 설계하였다. 또한 복제방지전용보안칩 제어용 드라이버 및 라이브러리를 제작하였다.

보안칩 내에 96비트 블록암호알고리즘 내장해 암복호화를 수행함으로써 불법복제를 방지하도록 하였으며, 이러한 기법들이 하드웨어로 설계되었기 때문에 크랙커에 의해서 프로그램이 크랙되는 것이 불가능하도록 하였다. 또한 보안칩의 무력화를 방지하기 위하여 통신포트로의 정보 교환시 보안기법(Scramble)에 의해 데이터의 인터셉트(Monitoring)를 불가능하게 하였다.

본 논문의 결과는 컴퓨터 프로그램에 대한 보호가 시급한 현시점에서 강력한 복제방지기능을 상품화할 수 있을 뿐 만 아니라, 불법복제의 방지를 위한 전용 보안칩에 대한 라이브러리함수와 디바이스 드라이버를 설계한다면, 데이터의 보호 및 사용자 인증 기능에도 적용할 수 있다. 또한 급속히 초고속화되어가고 있는 컴퓨터 통신망에서 비인가자에 대한 전송정보의 보호에도 효과적으로 적용되어질 수 있을 것이다.

참 고 문 헌

〔1〕 S. J. Han, H. S. Oh, "Design of Extended-DES cryptography," Proceeding of the IEEE International Symposium on Information Theory, pp. 353~359, July 1995.

〔2〕 S. J. Han, "The Improvement Data Encryption Standard(DES) Algorithm," Proceedings of ISSSTA '96, IEEE, pp. 1167~1171, Sept. 1996.

〔3〕 S. J. Han, "Improved-DES Cryptosystem Design," Journal of Kiss, Vol. 24, No. 1, pp. 57~67, Jan. 1997.

〔4〕 A. Herzberg and S. S. Pinter, "Public protection of software," In Advances in Cryptology, Proc Crypto 85. H. C. Williams, Ed., pp. 158, 1986.

〔5〕 A. Herzberg, and G. Karmi, "On software protection," In Proc. Fourth JCIT, pp. 388, Apr. 1984,

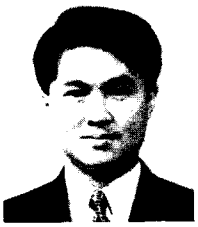
〔6〕 David Aucsmith, "Tamper resistant software," Information Hiding-Proceedings of the First International Workshop, pp. 317~333, 1996.

〔7〕 G. B. Purdy, G. J. Simmons, and J. A. Studier, A software protection scheme. In Proc. 1982 Symp. Security and Privacy, Oakland. CA, pp. 99~101. Apr. 1982.

〔8〕 D. Chaumm, "Design concepts for tamper responding systems," In Advances in Cryptology: Proc. Crypto 83. D. Chaum. Ed. New York: Plenum. pp. 387~390. 1984.

[9] J. Voelker and P. Walich, "How disks are padlocked," IEEE Spectrum. pp. 32~39, June 1986.
[10] M. G. Arnold and Mark D. Winkel, "Computer systems to inhibit unauthorized copying, unauthorized usage, and automated cracking of protected software," U.S. Patent No. 4 558 176. issued Dec. 1985.
[11] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," Revised, Oct. 1995.
[12] J. B. Kam & G. I. Davida, "Structured Design of Substitution Permutation Encryption Networks," IEEE Trans. on Comput., Vol. 28, No. 10, pp. 747~753, Oct. 1989.
[13] A. F. Webster, & S. E. Tavares, "On the design of S-boxes," CRYPTO '85, 1985.

〈著者紹介〉



오 명 신 (Myung-shin Oh)
1987년 : 조선대학교 전자공학과 졸업
1998년 : 조선대학교 전자공학과 석사
2002년 현재 : 동대학원 박사과정
1986년~7월 현재 : 전남대학교병원 의공학과
〈관심분야〉 영상압축, PACS, S/W불법복제방지시스템

한 승 조 (Seung-jo Han) 정회원
1980년 : 조선대학교 전자공학과 졸업
1982년 : 조선대학교 전자공학과 석사
1994년 : 충북대학교 전자계산학과 박사
1986년 6월~1987년 3월 : 뉴올리언즈대학 객원교수
1995년 2월~1996년 1월 : 텍사스대학 객원교수
2000년 12월~2002년 3월 : 버클리대학 객원교수
1998년 3월~현재 : 조선대학교 전자정보통신공학부 교수
〈관심분야〉 통신보안시스템설계, S/W불법복제방지시스템, ASIC 설계