데이터파일의 보호를 위한 스트림 암호방식 설계와 해석

Design and Analysis of Data File Protection based on the Stream Cipher

이경원
충북대학교 컴퓨터공학과

이중한
한밭대학교 정보통신전문대학원

김정호
한밭대학교 정보통신컴퓨터공학부

오창석
충북대학교 전기전자컴퓨터공학부

Kyung-Won Lee (kwlee@nwork.chungbuk.ac.kr)
Dept. of Computer Engineering, Chungbuk National University
Joong-Han Lee (Leexok@mbiznetworks.com)
Graduate School of Information & Communication, Hanbat National University
Jeong-Ho Kim (jhkim@hanbat.ac.kr)
School of Information Communication and Computer Engineering, Hanbat National University
Chang-Suk Oh (csoh@nwork.chungbuk.ac.kr)
School of Electrical and Computer Engineering, Chungbuk National University

중심어 : 스트림 암호방식, 해쉬 함수, 키 교환

Keyword : Stream Cipher, Hash Function, Key Exchange

요 약

개인용 컴퓨터 보급이 일반화되면서 기업체 또는 공공기관 등의 전산 정보체계도 중앙 Host 중심에서 PC 중심의 전산체계로 정착되어가고 있는 추세이다. 그러나 이러한 정보처리 시스템의 중심이 되는 PC는 누구나 쉽게 조작할 수 있다는 용이성에 의해 외부인의 무단 자료 유출 가능성은 항상 내재하고 있으며 자료 보관을 목적으로 하는 보조기억매체에 의한 자료 유출 가능성은 더욱 가중되고 있다. 따라서 본 논문에서는 주요 데이터의 손실 또는 외부 유출을 최소화하는 방안을 제시하고자 한다.

본 논문에서는 데이터 파일의 최종 보호수단은 데이터 자체를 암호화하여 보관하는 것이라는 점에 착안하여 PC와 보조기억매체에 수록되는 데이터의 보호를 위한 암호시스템을 구현하였다. 암호화/복호화 기법은 단일기법만으로 구현하는 것보다는 충분한 보안수준을 유지하기 위해서 Diffie-Hellman 키 교환 프로토콜과 스트림 암호중 대표적인 RC4(Rivest Cipher version 4)와 해쉬 함수의 대표적인 MD5(Message Digest version 5)를 복합적으로 적용하였다.

이상과 같이 구현된 암호시스템에 대한 평가분석으로써 암호복합도 측정, 처리속도 및 패턴매칭 분석을 해본 결과 안전성, 효율성, 유용성 면에서 만족할 만한 결과를 얻었다.

본 암호시스템은 Microsoft사의 Visual C++로 구현된 소프트웨어시스템으로 Windows상의 모든 PC에서 사용 가능한 범용성이 있는 시스템이므로, 최소한의 비용으로 모든 PC에 대한 보안대책을 구현할 수 있다고 생각된다.

Abstract

Recently, as the personal computers are supplied rapidly, they formed the nucleus of the computer system. But, because of the easiness that anyone uses them to go near easily, it is the fact that the security of personal computer is weakness. So, in the paper, I propose the technical method that minimizes the loss and leakage of important data.

This paper implemented a crypto system for security of data file on personal computer and assistance storage medium. The way of encryption/decryption is applied by complexity method which mixed Diffie-Hellman key exchange protocol, a typical RC4(Rivest Cipher version 4) algorithm of stream cipher and a typical MD5(Message Digest version 5) of Hash Function.

For valuation implemented crypto system, three criteria is presented, which are crypto complexity, processing time and pattern matching. And according to analysis the three criteria, the crypto system is verified the security, efficiency and usefulness.

The crypto system is programmed with Visual C++ language of Microsoft. And so, as this is software system, we shall have a technical security system at a minimum cost for all personal computer.

접수번호 : #040223-001
접수일자 : 2004년 2월 23일, 심사완료일 : 2004년 3월 18일

*교신저자 : 이경원, e-mail : kwlee@nwork.chungbuk.ac.kr

Ⅰ. 서론

현재 수많은 컴퓨터들이 인터넷으로 상호 연결되어 정보의 공유가 가능해짐에 따라 개인이 접하는 정보의 양도 급격히 증가하고 있다.

또한, 인터넷을 기반으로 한 다양한 정보와 서비스를 공유하는 PC는 누구나 쉽게 조작할 수 있다는 용이성에 의해 외부인의 무단 자료 유출 가능성은 항상 내재하고 있으며 자료 보관을 목적으로 하는 보조기억매체에 의한 자료 유출 가능성은 더욱 가중되고 있다. 따라서 PC에서의 보안을 위해 대형 시스템에서와 마찬가지로 종합적이고 체계적인 대책이 필요하다. 그러나 PC의 경우 그 특성상 전문 오퍼레이터의 고용, 별도의 안전한 장소 확보 등 대형 시스템과 같은 수준의 보안체계 유지는 곤란하다. 그러므로 PC가 공유하는 환경에서 이에 적절한 체계적인 보안 대책이 다음과 같이 요구된다[1],[2].

첫째가 환경적, 행정적 보안으로써 PC 노출환경에 따른 시스템의 고장 등을 고려한 규율적인 대책과 보안의식, 고취 및 시스템의 사용절차를 규정하는 등의 보안 절차를 말하며, 둘째로 물리적인 보안으로써 외부적인 보호대책을 위한 인원통제, 출입문의 견고한 잠금장치 등을 들 수 있으며, 마지막으로 기술적인 보안 대책으로써 시스템 내부로의 접근을 통제하기 위한 Lock기능이나 인증기능, 그리고 불법적인 자료접근이나 우연한 노출로부터 자료의 비밀성을 유지하기 위한 자료의 암호화 방법 등이 있다.

현재의 컴퓨터 시스템의 기본 설계 이념은 주로 업무의 효율성 향상에만 치중되어 왔기 때문에 이러한 보안상 취약점을 보완하는데 소홀한 것은 사실이다. 따라서 이젠 PC에 대한 제도적, 행정적인 보호 대책보다는 기술적인 보호 대책이 절실한 실정이다.

데이터 보호를 위한 암호시스템에는 하드웨어적 장비를 이용하는 시스템과 소프트웨어적 프로그램을 이용하는 시스템으로 구분할 수 있는데 하드웨어적인 방법은 암복호화시 소요되는 시간을 최소화할 수 있고 강도 높은 보안성을 유지시켜 줄 수 있으나 장비를 갖추는데 비용이 많이 소요되고 장비 사용의 융통성이 부족하다는 단점을 갖고 있다.

이와 반대로 소프트웨어적인 프로그램을 작성하여 사용할 경우에는 암복호화 시간이 많이 소요되고 프로그램 자체의 보호 대책이 강구되어야 하지만 값싸게 활용될 수 있고 컴퓨터 시스템의 어느 곳에서나 융통성 있게 사용될 수 있다는 장점을 갖추고 있다[3].

본 연구에서 구현된 암호시스템은 사용자가 암호화 시스템에 신경 쓸 필요없이 원하는 데이터를 암복호화 되도록 하는 암호화 방식으로, 암호화 알고리즘은 비밀키 암호방식 중 word 단위의 연산을 위주로 하는 S/W 구현용 스트림 암호중 대표적인 RC4를 사용하였으며, 데이터에 대한 무결성을 확인하는 방법으로는 해쉬 함수의 대표적인 MD5(Message Digest version 5 : RFC1321)를 적용하였다[4],[5]. 본 논문에서 구현된 암호시스템의 평가를 암호복합도, 처리속도, 패턴매칭 분석을 시도하였으며, 안정성, 효율성, 유용성 면에서 개선됨을 알 수 있었다.

Ⅱ. 시스템의 기본 구성

1. 설계시 고려사항

본 논문에서는 데이터화일 보호에 적합한 암호시스템을 설계하고자 한다. 암호 시스템을 개발하기 위해서는 대상이 되는 데이터와 주어진 여건들을 면밀히 검토하여야 한다. 고려되어야할 사항들로서는 시스템의 목적, 요구되는 보안수준, 데이터의 양과 활용빈도, 사용자의 특성, 암복호화 소요시간 등이 있다. 특히 본 연구의 대상이 되는 PC는 누구나 수시로 활용 가능한 장비라는 특수성을 감안하여 사용자의 편의성을 우선 고려해야 하며 암복호화시의 처리속도 또한 무시할 수 없는 고려대상이 된다. 본 논문에서는 위와 같은 사항들을 감안하여 다음과 같은 4가지를 고려하여 설계하였다.

◦암호시스템의 효율성

암호시스템에 사용될 암호 알고리즘이 암복호화 작업으로 인해 소요되는 시간 및 저장공간이 감당할 수 없을 정도이거나 PC구조에 대한 근본적인 변경을 초래해서는 안된다. 본 논문에서는 이러한 조건을 잘 만족하는 컴퓨터 논리연산에 가장 부합되는 대수적인 방법을 기본으로 하는 RC4 알고리즘을 이용하여 암호시스템을 설계하였다.

◦투명성(Transparency)

암호시스템이 그 운용이 복잡하고 효율성이 떨어지면 사용자들이 사용을 기피하게 됨으로써 오히려 보안의 위협이 될 수 있다. 아무리 좋은 시스템이 개발되었더라도 사용자에 대한 시스템의 편의성이나 효율성을 배제할 수는 없다. 본 연구에서는 파일을 선택한 후, 마우스의 오른쪽 버튼만 클릭하면 프로그램을 실행할 수 있도록 Window의 Shell을 이용하여 암호시스템을 구현하였다.

◦암호화 기법 적용

암호화는 인가된 데이터의 불법 유출이나 우연한 노출로부터 데이터의 비밀성 보호를 위한 가장 좋은 보호 수단이 된다. 암호화시스템의 설계에 있어서 여러 가지 암호기법 중에서 단일기법만으로 구현하는 것보다는 충분한 보안수준을 유지하기 위해서 여러 가지 기법을 복합적으로 사용하는 것이 필요하다. 그러나 한편으론 소프트웨어적인 암호화 시스템을 구현하기 위해서는 처리속도와 컴퓨터의 메모리용량을 고려하지 않을 수 없다. 본 연구에서 구현하고자 하는 암호화 시스템은 데이터자체를 암호화하여 보관하는 것이라는 점을 고려하여 암호화, 복호화 하는 이용자는 동일 이해집단에 속한다는 전제하에 암호화 알고리즘은 비밀키 암호방식이고, 파일의 기본연산(Insertion, Update, Delete)이 용이하도록 설계하여야 하기 때문에 스트림 암호방식인 RC4를 적용하고 데이터에 대한 무결성을 확인하는 방법으로는 해쉬 함수의 대표적인 MD5를 적용하였다.

◦인증 기능

본 암호화 시스템은 기본적으로 PC에 내장된 데이터파일을 대상으로 수행되므로 PC의 하드디스크에 수록된 데이터의 외부 유출에 대한 대책도 요구된다. 더구나 PC가 사무실내 노출된 환경에서 누구나 사용 가능하다는 용이성으로 인해 외부인의 불법 자료 유출가능성은 항시 상존하고 있는 현실이다. 따라서 PC의 중요한 데이터 파일에 대한 외부 유출방지를 위해서는 작업 권한을 통제할 필요가 있다.

본 암호시스템은 Diffe-Hellman 키 교환 프로토콜과 각각 입력되는 두개의 키를 교환하는 인증을 통하여 데이터파일에 대한 통제 기능을 부여하였다.

2. 시스템의 기본 구성

본 암호시스템은 데이터파일의 최종 보호수단은 데이터자체를 암호화하여 보관하는 것이라는 점에 착안하여 데이터파일을 관리하는데 중점을 두었다. 따라서 시스템의 암호화 알고리즘은 앞에서 살펴본 알고리즘 중에서 데이터파일을 암호화/복호화 하므로 비밀키 암호방식을 이용하였으며, 파일의 기본연산이 용이하도록 설계하여야 하기 때문에 블록암호화 방식을 배제하고 스트림 암호방식을 적용하였다. 따라서 본 논문에서는 암호화 알고리즘은 비밀키 암호방식 중 word 단위의 연산을 위주로 하는 S/W 구현용 스트림 암호중 대표적인 RC4를 사용하였으며 데이터에 대한 무결성을 확인하는 방법으로는 해쉬 함수의 대표적인 MD5를 적용하였다.

시스템은 크게 그림 1과 같이 인증기능, 암호화/복호화 과정, 무결성 인증 과정, 쉘 처리 과정으로 구성되며 암호화/복호화 과정은 2개의 모듈(S-Box를 초기화, RC4 암호화/복호화 과정)로, 무결성 인증 과정은 5개의 모듈(패딩비트의 부가, 길이 표현, MD 버퍼의 초기화, 512비트 블록의 메시지 처리, 출력)로 구성하였다.

암호시스템에 대한 Flow chart는 다음 그림 2와 같다.

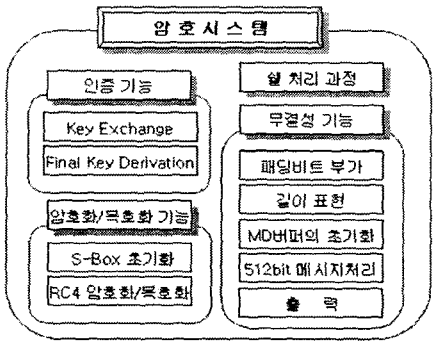

그림 1. 암호시스템의 구성도

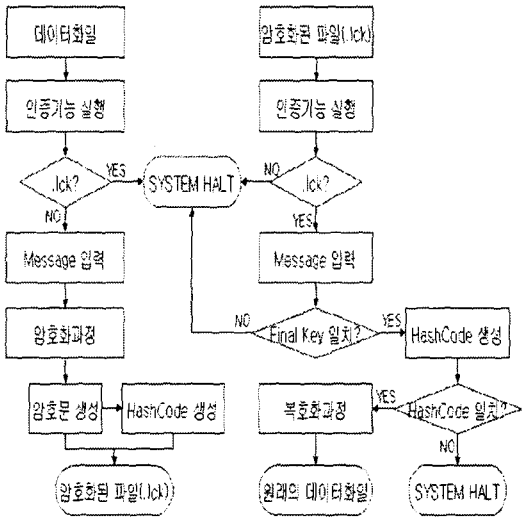

그림 2. 암호시스템의 Flow chart

III. 암호시스템의 설계

1. 인증과정

1.1. Key Exchange 모듈

본 논문에서 구현한 암호 시스템에서는 서로 일치하는 키

를 공유하기 위해 Diffie-Hellman 키 교환 프로토콜을 적용하였다. 처음 접속시, Sender는 접속을 시도하는 Receiver에게 아이디의 입력을 요청한 후 자신이 가지고 있는 사용자 목록과 접속을 시도하는 Receiver의 아이디를 비교한 후, 인가된 사용자일 경우에만 Diffie-Hellman 키 교환 프로토콜을 이용하여 공유 비밀키 DHKey를 생성하게 된다. Sender와 Receiver는 Diffie-Hellman 프로토콜에 의해 키 교환을 하게 되며 그 결과로서 1024비트의 공유 비밀키(DHKey)와 160비트의 인증키를 유도하게 되는데, Key Exchange 과정은 다음과 같으며 이를 그림 3에 나타내었다.

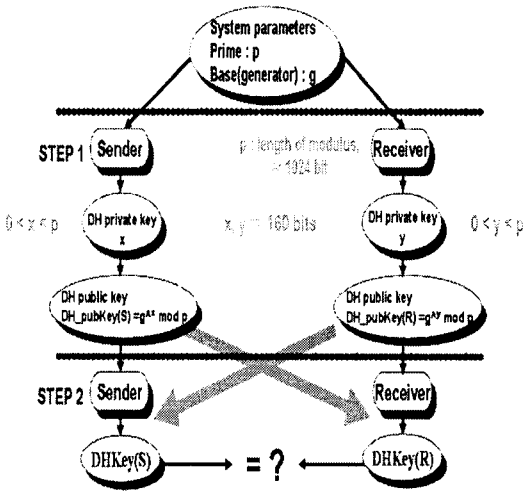

그림 3. Key Exchange 과정

먼저 제 1단계로 Sender와 Receiver는 주어진 시스템 파라미터(Prime Number and Base)에 의해 각자 공개키(Public Key)와 개인키(Private Key)의 키 쌍(Key pair)을 생성한 후, 2단계로 이들 중 공개키(DH_pubKey)를 각각 상대방 측에 전송한다. 마지막 3단계로서 Sender와 Receiver는 서로 교차하여 주고받은 공개키를 Base로 하여 각자가 가지고 있던 비밀키로 승산한 후, 모듈러 연산을 취하면 그 결과 값으로 양측이 동일한 값인 공유 비밀키 DHKey를 얻게 된다.

따라서 Sender와 Receiver는 160비트의 인증키 x, y의 생성에 있어서 난수를 발생시켜 이를 이용하였다. 난수는 전체적인 시스템 보안성을 결정하는 중요한 부분으로서 암호알고리즘은 무작위로 생성된 난수를 필요로 한다.

Sender와 Receiver의 공유비밀키 DHKey의 유도과정은 다음과 같다.

$$DHKey(S) = (DH\_pubKey(R))^{x} \bmod p = (g^{y} \bmod p)^{x} \bmod p = g^{yx} \bmod p$$

$$DHKey(R) = (DH\_pubKey(S))^{y} \bmod p = (g^{x} \bmod p)^{y} \bmod p = g^{xy} \bmod p$$

Sender와 Receiver 사이의 공유 비밀키 DHKey가 유도된 후, Receiver는 Sender에 DHKey를 요청하고 이를 보관하고 있는 DHKey와 비교하여 일치할 경우에만 다음 단계인 Key Derivation을 수행하게 된다. 만일 일치하지 않을 경우에는 처음 상태로 되돌아가게 된다.

1.2. Final key Derivation 모듈

Final Key Derivation 모듈에서는 Sender와 Receiver 사이에서 Diffie-Hellman 키 교환 프로토콜을 통해 유도한 공유비밀키 DHKey를 핵심정보로 이용해서 128 비트의 인증키 AuthKey와 실제적으로 암호화하는데 사용할 암호화 키인 Final Key를 유도하게 된다.

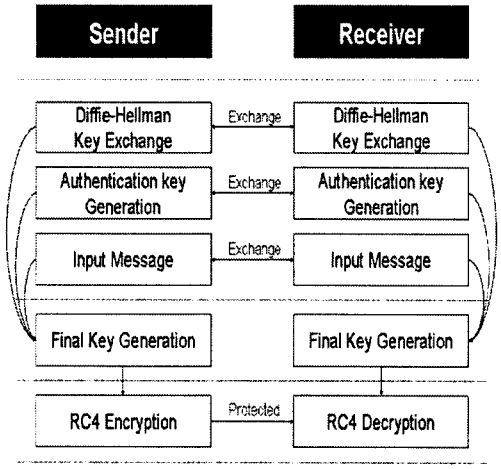

그림 4. Final key Derivation 과정

Final Key Derivation 모듈의 전체적인 처리과정은 그림 4와 같다.

Final Key Derivation 모듈에서는 먼저 DHKey(1024비트)와 Sender와 Receiver가 입력한 Message를 MD5 함수의 입력값으로 하여 AuthKey를 생성하게 되는데, 그 생성과정은 다음과 같다.

$$AuthKey(S) = MD-5[DHKey(S) / InputMessage]$$

$$AuthKey(R) = MD-5[DHKey(R) / InputMessage]$$

Sender와 Receiver의 AuthKey가 생성되고 난 후, Sender는 Receiver 측의 AuthKey를 요구하고, Receiver에서 전송받

은 AuthKey와 Sender가 보관하고 있는 AuthKey를 비교함으로써 사용자 인증 기능을 수행한다. 그 결과 인가된 사용자임이 확인되면 암호화/복호화 과정을 수행하게 되고, 만일 Sender의 AuthKey와 Receiver의 Final key가 서로 일치하지 않을 경우, Sender는 다시 초기 단계로 되돌아가게 된다.

Final key는 RC4 알고리즘으로 데이터를 암호화하는 데 사용할 실제적인 암호화 키로서, 128비트의 AuthKey와 Sender와 Receiver로부터 입력받은 메시지 값을 선택하여 Final key를 유도하게 된다.

2. 암호화/복호화 과정

RC4는 가변키 사이즈 스트림 사이퍼(Variable-key Size Stream Cipher)이며 OFB(Output Feedback)모드로 작동한다. RC4는 8×8 S-Box 즉 S0, S1, …, S255 상태의 S-Box를 가진다[6]. S-Box의 각 상태들은 키 스트림과 카운터 i, j의 연산에 의한 치환 과정에 의해 구성된다. 여기서 카운터 i는 선형치환의 특성을 가지고 j는 랜덤치환의 특성을 가지는 카운터이다[7]. RC4 암호화 알고리즘은 원래 평문(plaintext)과 키 스트림이 입력되면 크게 두 부분의 암호화 과정을 거쳐 암호화가 이루어진다. 이러한 두 부분의 암호화 과정은 그림 5와 같다.

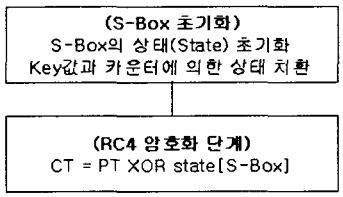

그림 5. RC4의 암호화 과정

2.1. RC4의 S-Box 초기화

S-Box 초기화 과정은 그림 5의 첫 번째 단계에서 이루어진다. 이 과정의 의사 코드(pseudo code)는 다음 그림 6과 같다. 이 과정에서는 S-Box의 256 상태들을 선형적으로 초기화하여 S0=0, S1=1, …, S255=255로 한다. 다음 입력된 키 스트림과 선형적으로 초기화 된 상태(state[256]) 그리고 인덱스인 index1, index2를 이용하여 S-Box의 각 상태들을 치환하여 S-Box의 상태들을 랜덤 하게 구성한다. 이러한 과정에 의해 S-Box의 초기화가 이루어진다[4].

```
void prepare_key(unsigned char *key_data_ptr, int key_data_len, rc4_key *key)
{
    unsigned char index1;
    unsigned char index2;
    unsigned char* state;
    short counter;

    state = &key->state[0];

    /* S box의 초기화 */
    for(counter = 0; counter < 256; counter++)
        state[counter] = counter;

    key->x = 0;
    key->y = 0;

    index1 = 0;
    index2 = 0;

    /* key와 index1, index2를 이용한 S-Box 상태치환 */
    for(counter = 0; counter < 256; counter++){
        index2 = (key_data_ptr[index1] + state[counter] + index2) % 256;
        swap_byte(&state[counter], &state[index2]);

        index1 = (index1 + 1) % key_data_len;
    }
}
```

그림 6. RC4의 S-Box 초기화 알고리즘

2.2 RC4의 암호화 알고리즘

S-Box의 초기화가 이루어진 후 그림 5의 두 번째 단계인 RC4의 암호화 단계에서 실제로 암호화를 수행한다. 이 과정의 의사 코드는 다음 그림 7과 같다.

```
void rc4(unsigned char *buffer_ptr, int buffer_len, rc4_key *key)
{
    /* x와 y는 선형치환과 랜덤치환을 위한 카운터 */
    unsigned char x;
    unsigned char y;
    unsigned char* state;
    unsigned char xorIndex;
    short counter;

    x = key->x;
    y = key->y;

    state = &key->state[0];

    for(counter = 0; counter < buffer_len; counter ++){
        x = (x + 1) % 256;
        y = (state[x] + y) % 256;
        swap_byte(&state[x], &state[y]);

        /* 암호문 생성을 위한 S-Box Index 생성 */
        xorIndex = (state[x] + state[y]) % 256;

        /* 암호문 생성 */
        buffer_ptr[counter] ^= state[xorIndex];
    }
    key->x = x;
    key->y = y;
}
```

그림 7. RC4의 암호화 알고리즘

암호화 과정은 선형치환과 랜덤치환을 위한 카운터 x, y의 덧셈과 나머지 연산, 그리고 치환 연산으로 S-Box 인덱스를 생성한다. 이 인덱스에 의한 S-Box의 상태(state[xorIndex])와 원문을 XOR 연산을 하여 암호문을 생성한다. 즉 '암호문 = 원문 ⊕ 인덱스에 의한 S-Box의 상태'이다[5].

3. 무결성 인증 과정

RC4의 암호화 알고리즘의 무결성을 보완하기 위하여 MD5를 적용한다. MD5는 임의 길이의 메시지를 입력으로 취하고, 128비트 메시지 다이제스트를 출력으로 제시한다. 입력은 512비트 블록으로 처리한다[6].

그림 8은 다이제스트를 만들기 위한 메시지 전체 진행과정을 나타내었으며 해석과정은 본 논문에서 5단계로 제시하였다.

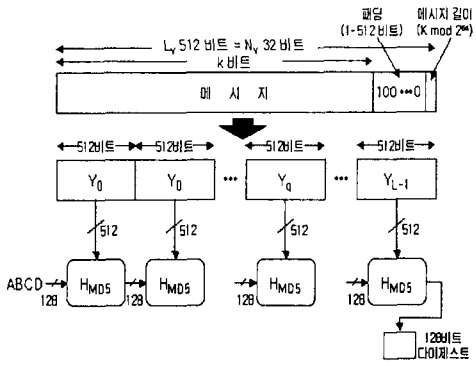

그림 8. MD5를 이용한 Message Digest 생성

1단계. 패딩 비트의 부가

메시지는 비트의 길이가 512를 법으로 하여 448과 합동이 되기 위하여 패딩된다. 따라서 길이는 448 mod 812가 된다. 즉 패딩된 메시지의 길이는 512 비트의 정수배보다 적은 64 비트이다. 메시지가 원하는 길이일지라도 패딩은 항상 부가된다. 예를 들어 만일 메시지의 길이가 448비트라면 512 비트가 패딩되어 960 비트의 길이가 된다. 그러므로 패딩 비트 수는 1~512의 범위이다. 패딩은 하나의 1-비트에 필요한 개수만큼의 0-비트들을 붙여 구성한다.

2단계. 부가된 길이

본래 메시지 길이(패딩 이전)에서의 64비트 길이 표현이 패딩 비트의 부가 단계의 결과에 부가되어진다. 만일 원래의 길이가 264보다 크다면, 길이의 하위 64 비트만이 사용되어진다. 그러므로 필드는 법 264로서 원래의 메시지 길이를 포함한다.

처음 두 단계의 결과로 길이가 512 비트의 정수배가 되는 메시지를 얻는다. 그림 5에서 확장된 메시지는 512 비트 블록 Y0, Y1, ..., YL-1의 계열로서 표현되어지며, 확장된 메시지의 전체 길이는 $L \times 512$ 비트이다. 따라서 결과는 16개의 32비트 단어의 배수가 된다. 그러므로 M[0 ... N-1]을 N이 16의 정수배인 결과 메시지의 단어로 표시하면, $N = L \times 16$ 이다.

3단계. MD 버퍼의 초기화

128비트 버퍼는 해쉬 함수의 중간과 최종 결과를 보관하기 위하여 사용되어진다. 버퍼는 4개의 32비트 레지스터(A, B, C, D)로 표현할 수 있다. 이러한 레지스터들은 다음의 16진수의 값으로 초기화되어진다.

A = 0 1 2 3 4 5 6 7
B = 8 9 A B C D E F
C = F E D C B A 9 8
D = 7 6 5 4 3 2 1 0

4단계. 512비트(16단어) 블록의 메시지 처리

MD5의 핵심은 4개의 라운드 처리로 구성된 모듈이다. 이 모듈은 그림 8에서 H_{MD5}로 표시되어 있고, 이 로직은 그림 9에 나타내었다. 4개의 라운드는 비슷한 구조를 가지나, 각각은 F, G, H, I로 표현되어지는 다른 기약 논리 함수를 사용한다. 그림 9에서 4개의 라운드는 같은 일반적인 함수 구조 f를 가지나, 각각은 서로 다른 기약 함수에 의존함을 나타내기 위하여 f_F , f_G , f_H , f_I로 표시한다.

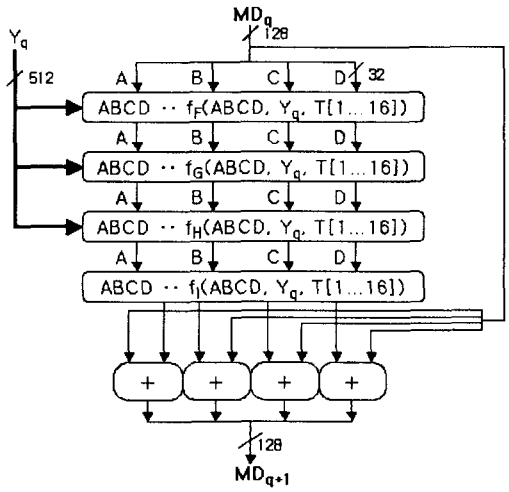

그림 9. 단일 512 비트 블록의 MD5 처리(HMD5)

각 라운드는 처리된 현재의 512비트 블록(Yq)과 128비트 버퍼값 ABCD를 입력으로 취하고, 버퍼의 내용을 갱신한다. 또한 각 라운드는 sin 함수로 구성된 64요소의 표 1의 Table T[1 ... 64]의 1/4를 사용하게 된다. T[i]로 표시하는 T의 i번째 요소는 $2^{32} \times abs(\sin(i))$의 정수 부분과 일치하는 값을 갖는다. 여기서 i는 radian이며 abs(sin(i))가 0과 1사이의 수이므로, T의 각 요소는 32 비트로 표시되어지는 정수이다. 표 1은 입력 데이터에서의 임의의 규칙을 무시한 32 비트 패턴의 랜덤화된 집합과 T의 값을 나타내고 있다.

따라서 블록 Yq에 대하여 알고리즘 Yq와 중간 다이제스트 값 MDq를 입력으로 한다. 즉, MDq는 버퍼 ABCD에 놓인다. 네 번째 라운드의 출력은 MDq에 더해져서 MDq+1을 생성한다. 덧셈은 법 2^{32}의 덧셈을 사용하여 버퍼에 있는 4단어의 각각에 대하여 MDq에서 대응하는 단어와 독립적으로 수행된다.

표 1. MD의 키 요소

(a) 논리 함수의 진리표

b	c	d	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

(b) sin 함수로 구성된 Table T

T[1]=D76AA478	T[17]=F61E2562	T[33]=FFFA3942	T[49]=F4292244
T[2]=E8C7B756	T[18]=C040B340	T[34]=8771F681	T[50]=432AFF97
T[3]=242070DB	T[19]=265E5A51	T[35]=6D9D6122	T[51]=AB9423A7
T[4]=C1BDCEEE	T[20]=E9B6C7AA	T[36]=FDE5380C	T[52]=FC93A039
T[5]=F57C0FAF	T[21]=D62F105D	T[37]=A4BEEA44	T[53]=655B59C3
T[6]=4787C62A	T[22]=02441453	T[38]=4BDECFA9	T[54]=8F0CCC92
T[7]=A8304613	T[23]=D8A1E681	T[39]=F6BB4B60	T[55]=FFEFF47D
T[8]=FD469501	T[24]=E7D3FBC8	T[40]=BEBFBC70	T[56]=85845DD1
T[9]=698098D8	T[25]=21E1CDE6	T[41]=289B7EC6	T[57]=6FA87E4F
T[10]=8B44F7AF	T[26]=C33707D6	T[42]=EAA127FA	T[58]=FE2CE6E0
T[11]=FFFF5BB1	T[27]=F4D50D87	T[43]=D4EF3085	T[59]=A3014314
T[12]=895CD7BE	T[28]=455A14ED	T[44]=04881D05	T[60]=4E0811A1
T[13]=6B901122	T[29]=A9E3E905	T[45]=D9D4D039	T[61]=F7537E82
T[14]=FD987193	T[30]=FCEFA3F8	T[46]=E6DB99E5	T[62]=BD3AF235
T[15]=A679438E	T[31]=676F02D9	T[47]=1FA27CF8	T[63]=2AD7D2BB
T[16]=49B40821	T[32]=8D2A4C8A	T[48]=C4AC5665	T[64]=EB86D391

5단계. 출력

결국 L개의 512 비트 블록이 처리되어지고, L번째 단계로부터의 출력이 128 비트 메시지 다이제스트이다.

하나의 512 비트 블록 처리의 4라운드에서의 각각의 로직을 더 상세히 살펴보면, 각 라운드는 버퍼 ABCE에서 16단계 처리의 계열로 구성된다. 각 단계의 형식은 다음 그림 10과 같다.

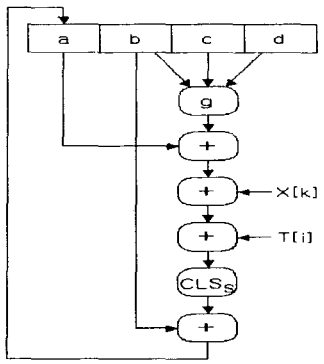

그림 10. MD5의 기본 동작 : [abcd k s i]

그림 10에서 a는 식(1)과 같이 표현 할 수 있다.

$$a \leftarrow b + CLSs(a + g(g, c, d) + X[k] + T[i]) \quad (1)$$

여기서 각 기호는 다음과 같다.

a, b, c, d = 버퍼의 4단어

g = 기약함수 F, G, H, I의 하나

CLSs = 32 비트에서 s 비트 순환 좌측 쉬프트(로테이션)

X[k] = M[q×16 + k] = 메시지의 q번째 512비트 블록에서 k번째의 32비트 단어

T[i] = 행렬 T에서 i번째 32 비트 단어 + = 법 232의 덧셈

4개의 기약 로직 함수의 하나가 알고리즘의 4라운드의 각각에 사용되어진다. 각 기약함수는 3개의 32비트의 단어를 입력으로 취하고 하나의 32 비트 단어를 출력한다[9]. 각 함수는 비트별 논리 연산의 집합을 실행한다. 즉, 출력의 n번째 비트는 3개의 입력들의 n번째 비트의 함수이다. 따라서 본 함수는 다음 표 2와 같이 요약할 수 있다.

표 2. MD5의 기약함수

라운드	기약함수 G	g(b, c, d)
f_F	F(b, c, d)	(b · c)∨(b′ · d)
f_G	G(b, c, d)	(b · d)∨(c · d′)
f_H	H(b, c, d)	b⊕c⊕d
f_I	I (b, c, d)	c⊕(b · d′)

표 2에서 논리 연산자(AND, OR, NOT, XOR)은 심볼(· , ∨, ⊕)로 표시한다. 함수 F는 조건부 함수이므로 만약 b를 만족하면 c이고 그렇지 않으면 d이다. 유사하게 G를 해석하면 만약 d를 만족하면 b이고 그렇지 않으면 c이다. 함수 H는 패리티 비트를 생성한다. 표 1 (a)는 4개 함수의 진리표와 비교하여 최종적인 MD5 과정은 다음과 같이 나타낼 수 있다[3].

MD0 = IV

MDq+1 = MDq + fI[Yq, fH[Yq, fG[Yq, MDq]]]

MD = MDL-1

여기서, 각 기호는 다음과 같다.

IV = 3 단계에서 정의한 ABCD 버퍼의 초기화

Yq = 메시지의 q번째 512 비트 블록

L = 메시지의 블록 수(패딩과 길이 필드 포함)

MD = 최종 메시지 다이제스트 값

```
/* Process each 16-word block. */
For i = 0 to N/16-1 do

  /* Copy block i into X. */
  For j = 0 to 15 do
    Set X[j] to M[i*16+j].
  end /* of loop on j */

  /* Save A as AA, B as BB, C as CC, and D as DD. */
  AA = A
  BB = B
  CC = C
  DD = D

  /* Round 1. */
  /* Let [abcd k s i] denote the operation : a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
  /* Do the following 16 operations. */
  [ABCD  0  7  1]  [DABC  1 12  2]  [CDAB  2 17  3]  [BCDA  3 22  4]
  [ABCD  4  7  5]  [DABC  5 12  6]  [CDAB  6 17  7]  [BCDA  7 22  8]
  [ABCD  8  7  9]  [DABC  9 12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]
  [ABCD 12  7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]

  /* Round 2. */
  /* Let [abcd k s i] denote the operation : a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
  /* Do the following 16 operations. */
  [ABCD  1  5 17]  [DABC  6  9 18]  [CDAB 11 14 19]  [BCDA  0 20 20]
  [ABCD  5  5 21]  [DABC 10  9 22]  [CDAB 15 14 23]  [BCDA  4 20 24]
  [ABCD  9  5 25]  [DABC 14  9 26]  [CDAB  3 14 27]  [BCDA  8 20 28]
  [ABCD 13  5 29]  [DABC  2  9 30]  [CDAB  7 14 31]  [BCDA 12 20 32]

  /* Round 3. */
  /* Let [abcd k s t] denote the operation : a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
  /* Do the following 16 operations. */
  [ABCD  5  4 33]  [DABC  8 11 34]  [CDAB 11 16 35]  [BCDA 14 23 36]
  [ABCD  1  4 37]  [DABC  4 11 38]  [CDAB  7 16 39]  [BCDA 10 23 40]
  [ABCD 13  4 41]  [DABC  0 11 42]  [CDAB  3 16 43]  [BCDA  6 23 44]
  [ABCD  9  4 45]  [DABC 12 11 46]  [CDAB 15 16 47]  [BCDA  2 23 48]

  /* Round 4. */
  /* Let [abcd k s t] denote the operation : a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
  /* Do the following 16 operations. */
  [ABCD  0  6 49]  [DABC  7 10 50]  [CDAB 14 15 51]  [BCDA  5 21 52]
  [ABCD 12  6 53]  [DABC  3 10 54]  [CDAB 10 15 55]  [BCDA  1 21 56]
  [ABCD  8  6 57]  [DABC 15 10 58]  [CDAB  6 15 59]  [BCDA 13 21 60]
  [ABCD  4  6 61]  [DABC 11 10 62]  [CDAB  2 15 63]  [BCDA  9 21 64]

  /* Then perform the following additions. (That is increment each
     of the four registers by the value it had before this block was started.) */
  A = A + AA
  B = B + BB
  C = C + CC
  D = D + DD

end /* of loop on i */
```

그림 11. MD5 알고리즘[13]

III. 암호시스템의 해석 및 평가

본 논문에서 제안한 암호시스템의 구현 결과는 그림 12에 나타내었다.

본 논문에서는 제안한 암호시스템에 대한 평가 분석으로써 암호복합도 측정, 처리 속도 및 패턴매칭 분석을 시도하였으며, 그 결과 안전성, 효율성, 유용성 면에서 DES, LFSR, IDEA와 비교하였다.

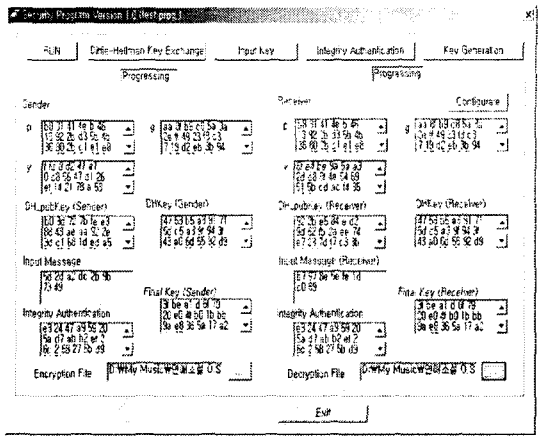

그림 12. 암호시스템 구현 결과

1. 암호복합도 해석

스트림 암호의 안전성은 주기에 의해 결정된다. 표 3은 Galois LFSR과 RC4 스트림 사이퍼의 주기를 보인 것이다.

표 3. Galois LFSR과 RC4 stream cipher의 주기

	Galois LFSR	RC4 Stream cipher
최소주기	$256\times 8bit = 2048$	$256\times(2^{64}-1)\times 64bit \fallingdotseq 1024$
최대주기	$256^2\times 256!\times 8bit$	$256^2\times 256!\times(2^{64}-1)\times 64bit$

표 3에서 Galois LFSR은 최소 2048비트의 짧은 주기를 가질 수 있으나 RC4 스트림 사이퍼는 최소 $2^{56}\times(2^{64}-1)\times 64bit \fallingdotseq 1024$ 비트의 긴 주기를 가진다[7].

결과적으로 RC4 스트림 사이퍼는 키 값에 따라 최소 2048 비트의 짧은 주기를 가지는 Galois LFSR에 비해 최소 주기가 약 1024 비트로 출력 결과의 해독에 의한 공격이 불가능함을 알 수 있다.

2. 처리속도 해석

암호복합도가 높은 암호알고리즘으로 보안성이 충분히 보장된다고 하더라도 처리 시간이 많이 소요되면 사용자의 불편함으로 인해 유용성은 떨어진다고 할 수 있다. 특히 암호시스템은 소프트웨어적인 구현 방법이므로 처리시간의 분석은 중요한 문제이다.

그림 13은 다른 암호 알고리즘과의 수행속도를 비교한 것이다. 그림에서 x 축은 암호 알고리즘의 종류를 나타내고 y축

은 RC4 스트림 암호 알고리즘의 수행 시간을 단위시간으로 표현한 것이다. 실험환경 및 조건은 다음과 같다.

∘실험환경
- CPU : Pentium 800MHz
- RAM : 256M
- Compiler : VC++ 6.0

∘실험조건
- 입력파일 : 한글문서 20개, 영문문서 20개

∘비교대상
- 스트림 암호 : RC4, 64bit Galois LFSR
- 블록 암호 : IDEA, DES

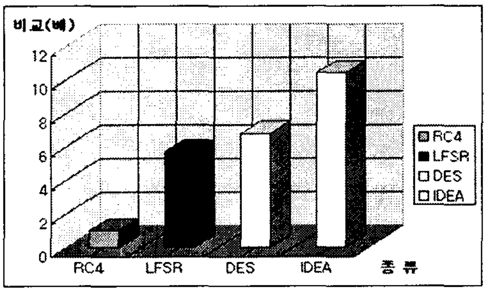

그림 13. 수행속도 비교

본 논문에서 구현한 RC4 스트림 암호를 이용한 암호시스템은 스트림 암호인 Galois LFSR에 비해 4-5배, 일반적인 블록 암호인 DES나 IDEA에 비해 6-9배 정도 고속으로 암호화가 가능함을 알 수 있다.

3. 패턴매칭 해석

대치법과 같은 암호문에서는 키의 길이만큼 주기적으로 같은 암호문이 나올 수 있으며 DES 알고리즘은 같은 키로 동일한 평문을 암호화하면 알고리즘 특성상 항상 동일할 암호문이 출력된다.

본 논문에서 구현한 암호시스템은 RC4와 MD5를 같이 이용되므로 같은 내용의 평문이라도 다른 결과의 암호문이 나타나며 파일내의 같은 문자에 대해서 동일한 암호문이 발생하지 않았음을 알 수 있었다.

암호결과의 분포를 실험하기 위하여 동일한 문자 "A"로 구성된 그림 14와 같은 txt 파일을 암호화하여 나타내었다.

```
00000000h: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 ; AAAAAAAAAAAAAAAA
00000010h: 41 41 41 41 0D 0A 41 41 41 41 41 41 41 41 41 41 ; AAAA..AAAAAAAAAA
00000020h: 41 41 41 41 41 41 41 41 41 41 0D 0A 41 41 41 41 ; AAAAAAAAAA..AAAA
00000030h: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 ; AAAAAAAAAAAAAAAA
00000040h: 0D 0A 41 41 41 41 41 41 41 41 41 41 41 41 41 41 ; ..AAAAAAAAAAAAAA
00000050h: 41 41 41 41 41 41 0D 0A 41 41 41 41 41 41 41 41 ; AAAAAA..AAAAAAAA
00000060h: 41 41 41 41 41 41 41 41 41 41 41 41 0D 0A 41 41 ; AAAAAAAAAAAA..AA
00000070h: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 ; AAAAAAAAAAAAAAAA
```

그림 14. 암호화하기 이전의 txt 파일 내용

그림 14는 데이터파일의 내용을 IDM Software Solutions의 UltraEdit-32 라는 utility를 이용해서 검색한 것으로 왼쪽의 Hexa값 "41h"는 실제 파일의 기록된 ASCII 코드값이고, 오른쪽은 ASCII코드 "41h"에 해당하는 문자 "A"를 나타낸 것이다.

```
00000000h: 11 A5 9A B6 A1 83 53 DB 6D BA 3A 4F F8 90 DB C5 ; .첵땅긕??O?北
00000010h: 51 4F FA F9 26 D4 AC D0 A5 B3 E8 16 07 62 7B 8A ; QO鋏&蓟졸녛..b{?
00000020h: AD BA 94 E7 95 5B 72 03 36 9F 35 F7 FA 24 AC 33 ; ?뷖?r.6?版$?
00000030h: 4C 1F 07 44 8D A0 8B 8D B4 60 1C 79 AD C2 AF E0 ; L..D뜸팥?.y??
00000040h: 6C 25 F3 95 BB 49 AE FD 4E 2D 75 D1 85 78 1A 58 ; l%?뮻?N-u?x.X
00000050h: 0E C7 C7 AB 9D 94 31 03 16 F1 B2 0B E4 75 2A 88 ; .피춸?..周.?*?
00000060h: 41 A1 DD 00 EB 81 3F 5E AF 9F CB F5 AF 16 84 D5 ; A◎.??^갠鍵?뵐
00000070h: B4 81 B9 6E AF BC 93 CF 3A 46 D4 84 AE 75 B2 FD ; 큰뤩?볐:F?칬끗
00000080h: 8D 10 EF 45 5A C0 94 AF C7 EA ED 86 12 11 60 FA ; ??Z뮻?悠?.`?
00000090h: 7B F9 D7 F2 4B DF C4 B3 81 79 FD 84 C7 4B AC D3 ; {翰?蓄콺y??m
000000a0h: 84 4F 08 25 42 31 54 65 6F E1 35 06 F7 B7 C6 E6 ; 졍.%B1Teo?.湯펜
000000b0h: D6 73 62 ED E5 9C EC E5 6E F7 9B D3 25 10 A2 87 ; ?b張뒻???.뷜
000000c0h: B8 82 25 1C D9 98 BB 65 30 BE AB 5E 9E 12 27 17 ; 뵇%.?퍈0쑹^?'.
000000d0h: 62 91 44 F6 F7 88 67 91 E9 EF 85 1B 43 D9 A4 69 ; b몇擢늚뭥?.C侖i
000000e0h: 5E 7A CE 75 D9 A3 FE DC 07 7B BF 2A 08 1E 01 87 ; ^z?名 .{?...?
000000f0h: BC CB E5 7C 5A A4 B9 47 98 0C 06 82 98 0B 6C 05 ; 섯?Z쯔G?.굴.l.
00000100h: B5 39 C9 2A 95 21 64 87 15 95 99 CB C3 4C E8 00 ; ???d?뷜塲L?
00000110h: 9D B7 59 1C AD 58 9E 49 96 82 68 9C A9 02 83 3D ; 뒺Y.춥엘뻴h쓹.?
00000120h: D3 66 9E C3 E4 8B C4 F2 36 2F 48 D1 87 CB 0D 0A ; ?왔?큘6/H??.
00000130h: CB 5A 1D 71 E1 5E FB 4B ED 6B D8 41 C1 27 F3 FC ; ?.q?????冊
```

그림 15. 암호화 결과 1

```
00000000h: 11 A5 9A B6 A1 83 53 DB 6D BA 3A 4F F8 90 DB C5 ; .첵땅긕??O?北
00000010h: 51 4F FA F9 26 D4 AC D0 A5 B3 E8 16 07 62 7B 8A ; QO鋏&蓟졸녛..b{?
00000020h: AD BA 94 E7 95 5B 72 03 36 9F 35 F7 FA 24 AC 33 ; ?뷖?r.6?版$?
00000030h: 4C 1F 07 44 8D A0 8B 8D B4 60 1C 79 AD C2 AF E0 ; L..D뜸팥?.y??
00000040h: 6C 25 F3 95 BB 49 AE FD 4E 2D 75 D1 85 78 1A 58 ; l%?뮻?N-u?x.X
00000050h: 0E C7 C7 AB 9D 94 31 03 16 F1 B2 0B E4 75 2A 88 ; .피춸?..周.?*?
00000060h: 41 A1 DD 00 EB 81 3F 5E AF 9F CB F5 AF 16 84 D5 ; A◎.??^갠鍵?뵐
00000070h: B4 81 B9 6E AF BC 93 CF 3A 46 D4 84 AE 75 B2 FD ; 큰뤩?볐:F?칬끗
00000080h: 8D 10 EF 45 5A C0 94 AF C7 EA ED 86 12 11 60 FA ; ??Z뮻?悠?.`?
00000090h: 7B F9 D7 F2 4B DF C4 B3 81 79 FD 84 C7 4B AC D3 ; {翰?蓄콺y??m
000000a0h: 84 4F 08 25 42 31 54 65 6F E1 35 06 F7 B7 C6 E6 ; 졍.%B1Teo?.湯펜
000000b0h: D6 73 62 ED E5 9C EC E5 6E F7 9B D3 25 10 A2 87 ; ?b張뒻???.뷜
000000c0h: B8 82 25 1C D9 98 BB 65 30 BE AB 5E 9E 12 27 17 ; 뵇%.?퍈0쑹^?'.
000000d0h: 62 91 44 F6 F7 88 67 91 E9 EF 85 1B 43 D9 A4 69 ; b몇擢늚뭥?.C侖i
000000e0h: 5E 7A CE 75 D9 A3 FE DC 07 7B BF 2A 08 1E 01 87 ; ^z?名 .{?...?
000000f0h: BC CB E5 7C 5A A4 B9 47 98 0C 06 82 98 0B 6C 05 ; 섯?Z쯔G?.굴.l.
00000100h: B5 39 C9 2A 95 21 64 87 15 95 99 CB C3 4C E8 00 ; ???d?뷜塲L?
00000110h: 9D B7 59 1C AD 58 9E 49 96 82 68 9C A9 02 83 3D ; 뒺Y.춥엘뻴h쓹.?
00000120h: D3 66 9E C3 E4 8B C4 F2 36 2F 48 D1 87 CB 0D 0A ; ?왔?큘6/H??.
00000130h: CB 5A 1D 71 E1 5E FB 4B ED 6B D8 41 C1 27 F3 FC ; ?.q?????冊
```

그림 16. 암호화 결과 2

또한, 그림 15와 그림 16을 비교해 보면 같은 문자("A")들로 채워진 동일한 파일에 대해서도 전혀 다른 결과의 암호문이 생성되었고 모든 같은 문자들의 평문임에도 불구하고 각각 다르게, 전혀 알아볼 수 없는 형태로 나타났으며 중복률도 거의 5%이내임을 알 수 있다. 특히 본 암호시스템은 패턴매칭 분석에 있어서 아주 강한 보안강도를 지녔다고 할 수 있다.

Ⅳ. 결 론

인터넷을 통한 정보의 제공 및 공유가 증가하고 있는 현실에서 질 높은 서비스와 안정적인 서비스 제공을 목표로 한다.

특히 사용자들이 암호 시스템에 신경 쓸 필요 없이 파일을 선택한 후, 마우스의 오른쪽 버튼만 클릭하면 프로그램을 실행할 수 있도록 Window의 Shell을 이용하여 투명 암호화 방식의 요구와 PC에 내장된 데이터파일을 관리하는데 중점을 두고 데이터파일을 암호화/복호화가 요구된다.

따라서 본 논문에서는 파일에 대한연산이 가능하도록 스트림 암호방식을 채택하였으며, 암호화 알고리즘은 비밀키 암호방식 중 word 단위의 연산을 위주로 하는 S/W 구현용 스트림 암호중 대표적인 RC4를 사용하였고 데이터에 대한 무결성을 확인하는 방법으로는 해쉬함수의 대표적인 MD5를 적용하였으며, 사용자 인증 부분은 Diffie-Hellman Key Protocol를 사용하였다.

본 논문에서 제안한 데이터파일의 보호 시스템은 크게 인증기능, 암호화/복호화 과정, 무결성 인증 과정, 쉘 처리 과정으로 구성되며 인증기능은 Key Exchange 모듈과 Final Key Derivation 모듈로, 암호화/복호화 과정은 2개의 모듈(S-Box 초기화, RC4 암호화/복호화 과정)로, 무결성 인증 과정은 5개의 모듈(패딩비트의 부가, 길이 표현, MD버퍼의 초기화, 512 비트 블록의 메시지 처리, 출력)로 구성하였다.

이상에서 구현한 암호시스템이 실용적으로 사용가능한가에 대한 효율성을 암호복합도 측정, 처리속도 분석, 패턴매칭 분석의 3가지 기준을 설정하여 각각 복잡도 계산 또는 실험에 의해 평가해 보았다. 암호복합도 측정에서는 스트림 암호의 주기를 비교한 결과 출력 결과의 해독에 의한 공격이 불가능함을 알 수 있었고, 다음으로 처리속도를 실험해 본 결과 스트림 암호인 Galois LFSR에 비해 4-5배, 일반적인 블록 암호인 DES나 IDEA에 비해 6-9배정도 고속으로 암호화가 가능함을 알 수 있었다. 또한 패턴매칭 분석에서는 동일한 파일에 대해서 전혀 다른 결과의 암호문이 생성되었고, 중복률도 거의 5%이내임을 알 수 있었다.

따라서 제안한 암호화 방식은 안전성, 효율성, 유용성 면에서 만족할 만한 결과를 얻었다.

한편 PC내의 데이터 파일 보호뿐만 아니라 불법 디스켓에 의한 PC의 하드디스크상의 데이터에 대한 기술적인 보호대책도 마련되었다고 판단되며 또한, 본 암호시스템은 소프트웨어 프로그램으로 Windows상의 모든 PC에서 사용 가능한 범용성이 있는 시스템이므로, 최소한의 비용으로 모든 PC에 대한 보안대책을 구현할 수 있다고 생각된다.

참 고 문 헌

[1] 이임영, 송유진, "현대암호", 생능출판사, 1997.

[2] 박창섭, "암호이론과 보안", 대영사, 1999.

[3] William stalling, "통신망 정보보호", 그린, 1997.

[4] IETF RFC 1321, MD5 Message-Digest Algorithm

[5] R. L. Rivest, "The RC4 Encryption Algorithm," RSA Data Security Inc, Mar 1992.

[6] Douglas R. Stinson, "Cryptography-Theory and Practice," CRC Press, Inc., 1995.

[7] Alfres J. Menezes, Paul C.vaz Oorschot, Scott A. Vanstrone, "Handbook of Applied Cryptography," CRC Press, 1997.

[8] Henk C.A. van Tilborg, "Fundamentals of cryptology," Kluwer Academic Publishers, 2000.

[9] Neal Koblitz, "Algebraic aspects of cryptography," Springer, 1998.

[10] R.J. Anderson, "Fast Attack on Certain Stream Cipher," Electronics Letters, Vol. 29, No. 15, 22 Jul 1993.

[11] 한국전자통신연구원, "암호 알고리즘의 핵심 논리 설계 및 분석 연구", 2000.

[12] E.S. Selmer, "Linear Recurrence over Finite Field," University of Bergen, Norway, 1996.

[13] Bruce schneier, "Applied cryptography," John Wiley & Sons, Inc., 1996.

[14] ANSI/SCTE 41 2001(Formerly DVS 301) POD Copy Protection System, June 2001.

[15] OC-SP-PODCP-IF-I09-030210 OpenCable POD Copy Protection System, CableLabs, February 2003.

이 경 원(Kyung-Won Lee) 정회원

2000년 2월 : 한밭대학교 전자계산학과 (공학사)
2002년 2월 : 한밭대학교 전자계산학과 (공학석사)
2004년 3월 ~ 현재 : 충북대학교 컴퓨터공학과(박사과정)
2001년 12월 ~ 2003년 3월 : (주)아이피에스 기술연구소 전임연구원
2003년 3월 ~ 현재 : 한밭대학교 정보통신컴퓨터공학부 조교
<관심분야> : 암호학, 콘텐츠 보호, 네트워크 보안

이 중 한(Joong-Han Lee) 정회원

1988년 2월 : 서울대학교 산업공학과 (공학사)
2000년 2월 : 포항공과대학교 산업공학과 (공학석사)
2003년 2월 : 한밭대학교 컴퓨터공학과 (박사과정)
1990년 ~ 1994년 : LG전자 대리
1995년 ~ 1996년 : LG그룹 회장실 과장
1997년 ~ 2000년 : LG텔레콤 차장
<관심분야> : 데이터통신, 프로토콜공학, 인터넷보안

김 정 호(Jeong-Ho Kim) 정회원

1980년 2월 : 경북대학교 전자공학과 (공학사)
1983년 2월 : 경북대학교 전자공학과 (공학석사)
1995년 2월 : 단국대학교 컴퓨터공학과 (공학박사)
1983년 ~ 1996년 : 한국전자통신연구원 책임연구원/실장
1989년 8월 : 정보처리기술사
1990년 8월 : 전자기술사
1991년 12월 : 정보통신기술사
1996년 ~ 현재 : 한밭대학교 정보통신컴퓨터공학부 부교수
<관심분야> : 데이터통신, 프로토콜공학, 인터넷보안

오 창 석(Chang-Suk Oh) 종신회원

1978년 2월 : 연세대학교 전자공학과 (공학사)
1980년 2월 : 연세대학교 전자공학과 (공학석사)
1988년 8월 : 연세대학교 전자공학과 (공학박사)
1985년 ~ 현재 : 충북대학교 전기전자컴퓨터공학부 교수
1982년 ~ 1984년 : 한국전자통신연구원 연구원
1990년 ~ 1991년 : 미국 Stanford대학교 객원교수
<관심분야> : 컴퓨터 네트워크, 뉴로 컴퓨터, 정보 보호