

# 컴퓨터보안 보고서

## 실습과제 #02

강좌 명: 컴퓨터보안

교수님: 정준호 교수님

학과: 컴퓨터공학과

학년: 3학년

학번: 2017112138

이름: 정여준

# 1. 시저 암호

## <정의>

간단한 치환암호의 일종이다. key값에 해당하는 값만큼의 일정한 거리만큼 밀어서 다른 알파벳으로 치환하는 방식이다. 약 기원전 100년경에 만들어져 로마의 장군인 카이사르가 동맹군들과 소통하기 위해 만든 암호이다.

## <복호화 과정>

key값에 해당하는 값만큼의 일정한 거리만큼 당겨서 원래 알파벳으로 되돌린다.

## <문제점>

보안성이 안 좋다. 그냥 알파벳을 키 값만큼 당기면 암호가 풀리기 때문이다. 또한 중복되는 단어가 나열되는 경우에 유추하기가 매우 쉬워진다는 단점이 있다.

## <소스코드>

```
#include<iostream>
#include<string>
using namespace std;

char caesar(char c,int key) //시저 암호화
{
    if (isupper(c)) //대문자일 경우
        c = (((c - 65) + key) % 26) + 65; //26글자안에서 몇칸 이동하는지
    else if (islower(c))
        c = (((c - 97) + key) % 26) + 97; //26글자 안에서 몇칸 이동하는지

    return c;
}

char decryption(char c,int key) //복호화
{
    if(isupper(c)) //대문자일 경우
    {
        if (c - key < 65) //아스키코드를 기준으로 키 값을 뺐을 때 A(65)보다 작아질
            경우
                c = 91 - (key - (c - 65)); //65를 기준으로 더 작아진 만큼 Z(90)+1에서
                뺀다.
        else
            c = c - key;
    }
    else if (islower(c)) //소문자일 때
    {
        if (c - key < 97) //아스키코드를 기준으로 키 값을 뺐을 때 a(97)보다 작아질
            경우
                c = 123 - (key - (c - 97)); //97을 기준으로 더 작아진 만큼
                z(122)+1에서 뺀다.
```

```

        else
            c = c - key;

    }

    return c;
}

int main()
{
    int key = 0;
    string input; //평문
    string output; //암호문
    string output2; //복호문
    cout << "평문을 입력해주세요: ";
    getline(cin, input);
    cout << "키 값을 입력해주세요: ";
    cin >> key;

    for (int i = 0; i < input.length(); i++)
    {
        output += caesar(input[i],key);
    }
    cout << "암호문: " << output << endl;

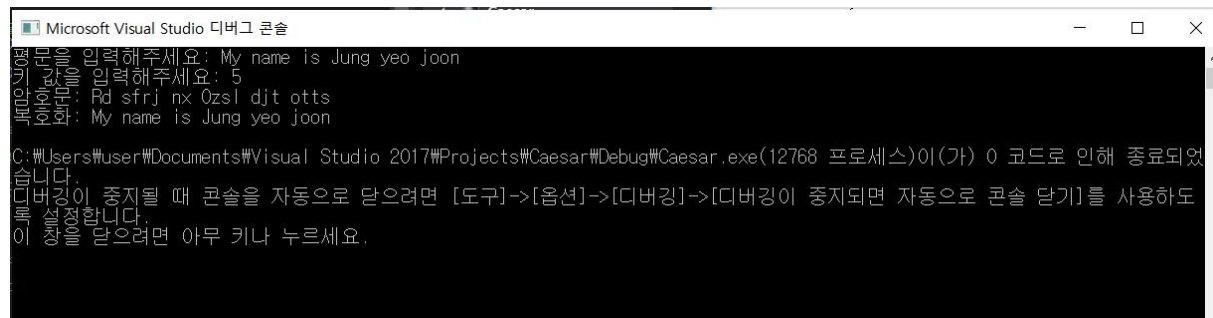
    for (int i = 0; i < output.length(); i++)
    {
        output2 += decryption(output[i],key);
    }
    cout << "복호화: " << output2 << endl;
}

```

### <소스코드 리뷰>

일단 평문을 입력하고 키 값도 입력받는다. 입력받은 평문에 대해서 키 값으로 받은 값만큼 씩 뒤로 밀어서 암호문을 생성한다. 여기서 키 값을 더했을 때 26을 초과하는 경우가 있다. 즉, 아스키코드로 A를 10진수로 나타내면 65이고 Z는 90인데 알파벳 + 키 값이 90을 넘게 되면 아예 다른 문자가 나온다는 문제가 있다. 이러한 문제를 해결하기 위해 순환적으로 만들어야 하는데 이를 위해 (문자의 아스키코드에서의 10진수로 나타낸 수 - 65) + key 값을 26으로 나눈 나머지를 65에 더해준다. 이를 소문자와 대문자인 경우를 나눠서 실행해준다. 복호화의 경우에도 소문자와 대문자를 나눠서 순환적으로 만들어준다.

## <결과>



```
Microsoft Visual Studio 디버깅 콘솔
명문을 입력해주세요: My name is Jung yeo joon
키 값을 입력해주세요: 5
암호문: Pd sfrj nx 0zsl djt otts
복호화: My name is Jung yeo joon

C:\Users\user\Documents\Visual Studio 2017\Projects\Caesar\Debug\Caesar.exe(12768 프로세스)이(가) 0 코드로 인해 종료되었습니다.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구]->[옵션]->[디버깅]->[디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요.
```

## 2. 단일 치환 암호

### <정의>

알파벳 26문자를 무작위로 나열한 집합과 원래의 알파벳 26문자를 서로 1대1 대응시킴으로써 암호문을 생성해내는 방법. 송신자와 수신자는 치환표를 공유해야 한다. 전사공격으로는 해독이 불가능하다.

### <복호화 과정>

치환표를 기준으로 1대1 대응되는 문자를 찾아서 원래의 문자로 변환한다.

### <문제점>

평문에 등장하는 문자의 빈도가 암호문으로 바뀐 뒤에도 그대로 드러난다는 것이 단일 치환 암호의 단점 중 하나이다. 그래서 빈도 분석이란 암호해독방법을 사용하여 해독할 수 있다. 그래서 암호문이 길수록 빈도분석을 하기 용이하다. 그래서 긴 암호문을 만들 때 이 방법을 사용하면 해독하기 다소 쉬울 수 있다. 또한 같은 문자가 연속해서 나타나거나 단어의 단락을 알게 된다면 단서가 된다. 이 때문에 해독의 속도가 갈수록 빨라진다.

### <소스코드>

```
#include<iostream>
#include<string>
using namespace std;

int main()
{
    string input; //평문
    string output; //암호화한 문장
    string output2; //복호화한 문장

    //치환표 key1 문자열과 key2문자열에서 같은 인덱스에 있는 값으로 치환
    string key1 = "abcdefghijklmnopqrstuvwxyzABCDEFGH IJKLMNOPQRSTUVWXYZ";
    string key2 = "wyhfxumthvsngenbrdzlqapcokiWYHFXUMTJVSGENBRDZLQAPCOKI";

    cout << "암호화할 평문을 입력하세요: ";
    getline(cin, input);
    int index = 0; //치환표에서 어느 인덱스에 있는 문자인지 알려줄 정수형 변수

    //암호화
    for (int i = 0; i < input.length(); i++)
    {
        for(int j=0; j<key1.length();j++){
            if (input[i] == key1[j]) //평문에 있는 알파벳을 key1에서 찾은 후
                //인덱스 값 저장
```

```

        {
            index = j;
            break;
        }
    }
    if (input[i] == ' ')
        output += ' ';
    else
        output += key2[index]; //치환표를 활용해서 암호문 생성
}
cout << "암호화한 문장: " << output << endl;

//복호화
for (int i = 0; i < output.length(); i++)
{
    for (int j = 0; j < key2.length(); j++)
    {
        if (output[i] == key2[j]) //암호문에 있는 알파벳을 key2에서 찾은 후
        {
            index = j;
            break;
        }
    }

    if (output[i] == ' ')
        output2 += ' ';
    else
        output2 += key1[index]; //치환표를 활용해서 복호화
}
cout << "복호화한 문장: " << output2 << endl;
}

```

인덱스 값 저장

## <소스코드 리뷰>

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
W	Y	H	F	X	U	M	T	J	V	S	G	E	N	B	R	D	Z	L	Q	A	P	C	O	K	I

치환표를 생성하는데 두개의 길이가 같은 문자열로 생성한다. 일단 첫번째 문자열에는 원래의 알파벳 순서대로 알파벳들을 나열해준다. 두번째 문자열에는 각 알파벳에 1대1로 대응시킬 알파벳을 입력해준다. 이제 평문을 입력받는다. 평문에 나오는 알파벳을 첫번째 문자열을 통해서 몇 번째에 위치하는지 알아내고 그 값을 정수형 변수에 저장해 둔다. 그리고 두번째 문자열에서 저장되어 있는 값에 위치하는 알파벳으로 변환해준다. 복호화 과정은 암호문에 나오는 알파벳을 두번째 문자열을 통해서 몇 번째에 위치하는지

알아내고 그 값을 다시 정수형 변수에 저장해 둔다. 그리고 첫번째 문자열에서 저장되어 있는 값에 위치한 알파벳으로 다시 변환해준다.

#### <실행결과>

```
Microsoft Visual Studio 디버그 콘솔
암호화할 평문을 입력하세요: My name is Jung yeo joon
암호화한 문장: Ek nwex hl Vanm kxb ybbn
복호화한 문장: My name is Jung yeo joon

C:\Users\user\Documents\Visual Studio 2017\Projects\SimpleSubstitution\
) 0 코드로 인해 종료되었습니다.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구]->[옵션]->[디버깅]->[
록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요.
```

### 3. 다중 치환 암호

#### <정의>

단일 치환 방식에서 알파벳의 빈도 정보를 파악해서 어느정도 유추할 수 있는 문제점이 있었는데 이러한 빈도정보를 무력화시킬 방법으로 등장한 암호방식이다. 평문에 등장하는 문자의 빈도와 암호문에 등장하는 문자의 빈도를 상이한 암호 알고리즘이다.

#### <복호화 과정>

비즈네르 암호로 하면 키워드를 공유해서 키워드에 해당하는 키 값을 얻어내고 얻어낸 키 값을 통해서 시저암호에서 복호화 한대로 동일하게 복호화를 하면 된다.

#### <문제점>

예시를 들어보자.

평문: My dog is cool I like dogs better than cats. Dogs do not have hair so dog is good.

키워드: dog

암호문: pmjruovquozrowqhrujghhzhfzkotfozvruijgrbuwvgysndwxvcjruov

2개의 특정 문자열 ruj가 등장한다. ruj가 존재하는 두 구간은 키의배수고 키의 길이는 이 구간의 약수인 것을 알 수 있다. 하지만 단순히 우연히 이루어진 것이라고 생각할 수도 있다. 그리고 알파벳 두개가 겹치는 단어는 같은 단어일 확률이 낮다. 그래서 위와 같이 세 글자 이상의 단어가 두 번 반복된다면 평문에서 똑 같은 단어가 2번 이상 사용됐을 수 있겠다고 유추할 수 있다. 또한 알파벳 3개 이상으로 이루어진 특정 문자열로 키의 길이를 어느정도 예측할 수 있다는 문제점이 있다.

#### <소스코드>

```
#include<iostream>
#include<string>
using namespace std;

char caesar(char c, int key) //시저 암호화
{
    if (isupper(c)) //대문자일 경우
        c = (((c - 65) + key) % 26) + 65; //26글자안에서 몇칸 이동하는지
    else if (islower(c))
        c = (((c - 97) + key) % 26) + 97; //26글자 안에서 몇칸 이동하는지

    return c;
}

char decryption(char c, int key) //복호화
```



```

{
    if (isupper(c)) //대문자일 경우
    {
        if (c - key < 65) //아스키코드를 기준으로 키 값을 뺀을 때 A(65)보다 작아질
        경우
            c = 91 - (key - (c - 65)) % 26; //65를 기준으로 더 작아진 만큼
            Z(90)+1에서 뺀다.
        else
            c = c - key;
    }
    else if (islower(c)) //소문자일 때
    {
        if (c - key < 97) //아스키코드를 기준으로 키 값을 뺀을 때 a(97)보다 작아질
        경우
            c = 123 - (key - (c - 97)) % 26; //97을 기준으로 더 작아진 만큼
            z(122)+1에서 뺀다.
        else
            c = c - key;
    }

    return c;
}

int main()
{
    string input; //평문
    string key = "yeo"; //비즈네르암호에서 키워드
    string key2 = "abcdefghijklmnopqrstuvwxyz";
    string output; //암호문
    string output2; //복호화한 문장

    cout << "평문을 입력하세요: ";
    getline(cin, input);
    int index = 0;
    int k = 0;

    //암호화
    for (int i = 0; i < input.length(); i++)
    {
        if (input[i] == ' ')
            output += ' ';
        else {
            index = i % key.length(); //인덱스는 key의 길이가 3일 때
            0,1,2,0,1,2,0,...순으로 바뀐다.
            for (int j = 0; j < key2.length(); j++)
            {
                if (key[index] == key2[j]) //키워드를 알파벳 표에서의
                인덱스를 구한다.
                    k = j; //구한 값을 k에 대입 k가 키 값이 된다.
            }
            output += caesar(input[i], k); //키 값을 찾았으면 시저암호와 동일한
            방법으로 키만큼 이동하여 암호문 생성
        }
    }
}

```

```

cout << "암호문: " << output << endl;

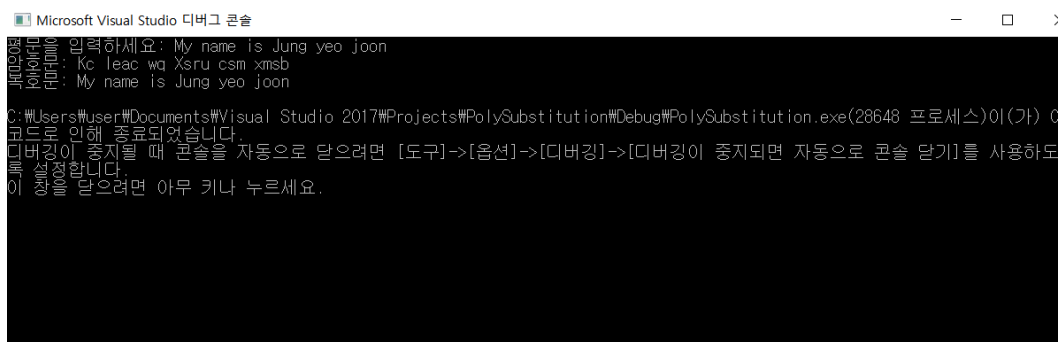
//복호화
for (int i = 0; i < input.length(); i++)
{
    if (output[i] == ' ')
        output2 += ' ';
    else {
        index = i % key.length(); //인덱스는 key의 길이가 3일 때
0,1,2,0,1,2,0,...순으로 바뀐다.
        for (int j = 0; j < key2.length(); j++)
        {
            if (key[index] == key2[j])//키워드를 알파벳 표에서의 인덱스를
구한다.
                k = j;//구한 값을 k에 대입 k가 키 값이 된다.
        }
        output2 += decryption(output[i], k);// 키 값을 찾았으면 시저암호와
동일한 방법으로 복호화
    }
}
cout << "복호문: " << output2 << endl;
}

```

### <소스코드 리뷰>

내가 원하는 키워드를 생성한다. 그리고 알파벳을 순서대로 입력한 문자열을 생성한다. 이제 평문을 입력하고 반복문을 통해 암호문을 생성한다. 암호문을 생성할 때 평문을 입력받은 문자열의 인덱스 값을 키워드의 길이로 나눈 나머지를 정수형 변수에 저장한다. 그리고 키워드를 저장한 문자열에서 저장한 인덱스에 위치한 문자를 알파벳과 비교해서 자리를 알아낸다. 그 자리를 정수형 변수에 저장한다. 이 값이 시저암호에서의 키 값의 역할을 한다. 이제 시저암호에서 암호화하는 방법과 동일한 방법으로 저장한 키 값을 이용해 암호화한다. 복호화 방법은 암호화했을 때의 방법과 동일하게 인덱스 값과 키 값을 찾아낸다. 그 이후 시저암호에서 복호화 한 방법과 동일하게 복호화를 해준다.

### <실행결과>



```

Microsoft Visual Studio 디버그 콘솔
평문을 입력하세요: My name is Jung yeo joon
암호문: Kc leac wo Xsru csm xmsb
복호문: My name is Jung yeo joon
C:\Users\User\Documents\Visual Studio 2017\Projects\PolySubstitution\Debug\PolySubstitution.exe(28648 프로세스)이(가) C
코드로 인해 종료되었습니다.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구]->[옵션]->[디버깅]->[디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도
록 설정합니다.
이 창을 닫으려면 아무 키나 누르세요.

```

## 소감

이번 과제를 통해서 각 암호 프로그래밍에 대한 정의와 복호화 과정 암호화 과정 단점 등을 알 수 있었다. 일단 시저암호의 경우는 이미 알고 있었던 내용이었기 때문에 어떠한 메커니즘으로 암호화가 이루어지고 복호화가 이루어지는지 알고 있었다. 하지만 코드로 구현해보는 것은 처음 이어서 다시 한번 알고 있었던 내용을 이용해 코드를 짜면서 조금 더 깊게 이해할 수 있었다. 그리고 단일 치환 암호 같은 경우에는 수업시간에 처음 본 방법이었다. 1대1대응하는 값으로 변환된다는 점에서 시저암호와 유사한 것 같았다. 하지만 시저암호처럼 규칙이 있는 것이 아닌 임의로 정한 알파벳과 1대1로 대응시키므로 전사공격에 의해서 풀 수 없는 암호가 된 것이 특징인 것 같다. 하지만 평문에서 반복된 알파벳은 암호문에서도 반복되기 때문에 빈도수에 의해서 해독이 가능하다는 단점이 있다는 것도 내가 직접 코드를 구현해서 암호화하고 복호화 하면서 확인할 수 있었다. 그리고 다중 치환 암호를 프로그래밍해보면서 일단 다중 치환 암호에 어떤 종류가 있는지 알 수 있었고 각각이 어떤 방식으로 암호화되고 복호화 되는지 알 수 있었다. 또 분석하기는 힘들지만 어느정도 남아있는 문제점도 알아낼 수 있었다. 이렇게 3가지 암호 프로그램을 직접 코드로 구현해보면서 각 암호화 방법에 대해서 더 자세히 알 수 있었고 직접 코드를 짤기 때문에 암호화가 이루어지는 메커니즘과 복호화가 이루어지는 메커니즘 또한 더 자세하게 알 수 있어서 좋았습니다.