Applied Number Theory

Korea University
Dept. of Computer Science and Engineering

Joong Heon Kim
Gene Moo Lee

Outline

Contents

- Introduction
- Implementation
- Conclusion and Improvement
- Q & A

Outline

Contents
Introduction
Implementation
Conclusion and Improvement
Q & A

Introduction

- Analyzing the cryptosystems that we learned in this class
- Implementing the systems in computer program

Outline

Contents
Introduction
Implementation
Conclusion and Improvement
Q & A

Implementation

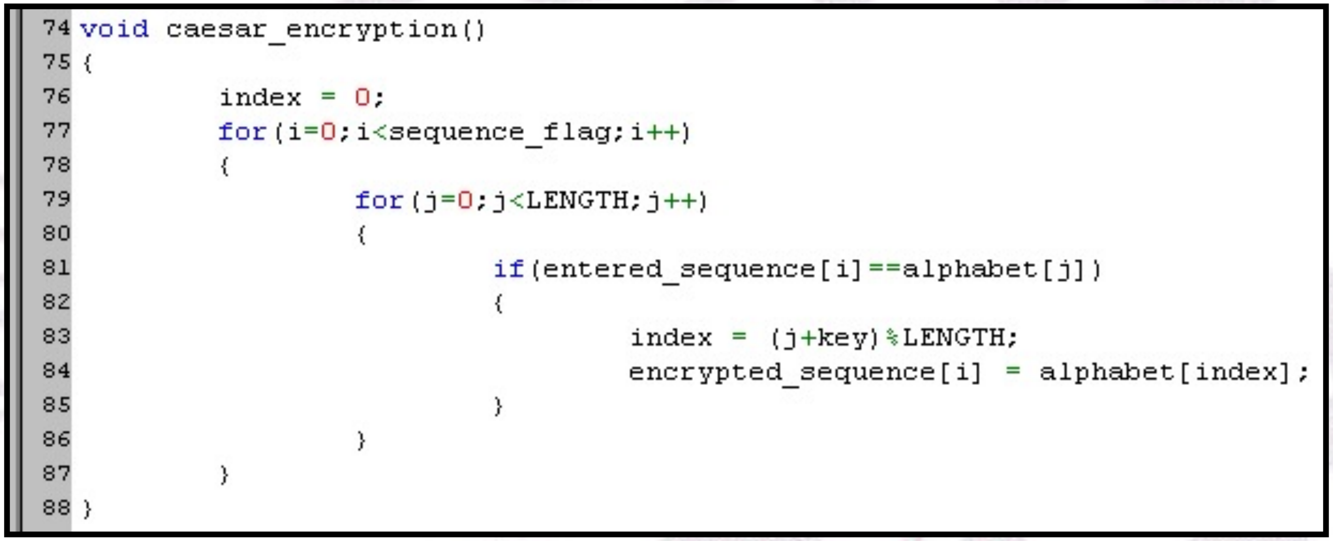
§ Environment :

- Language : Visual C++ 6.0
- Operating System : Windows 2000

§ Programming :

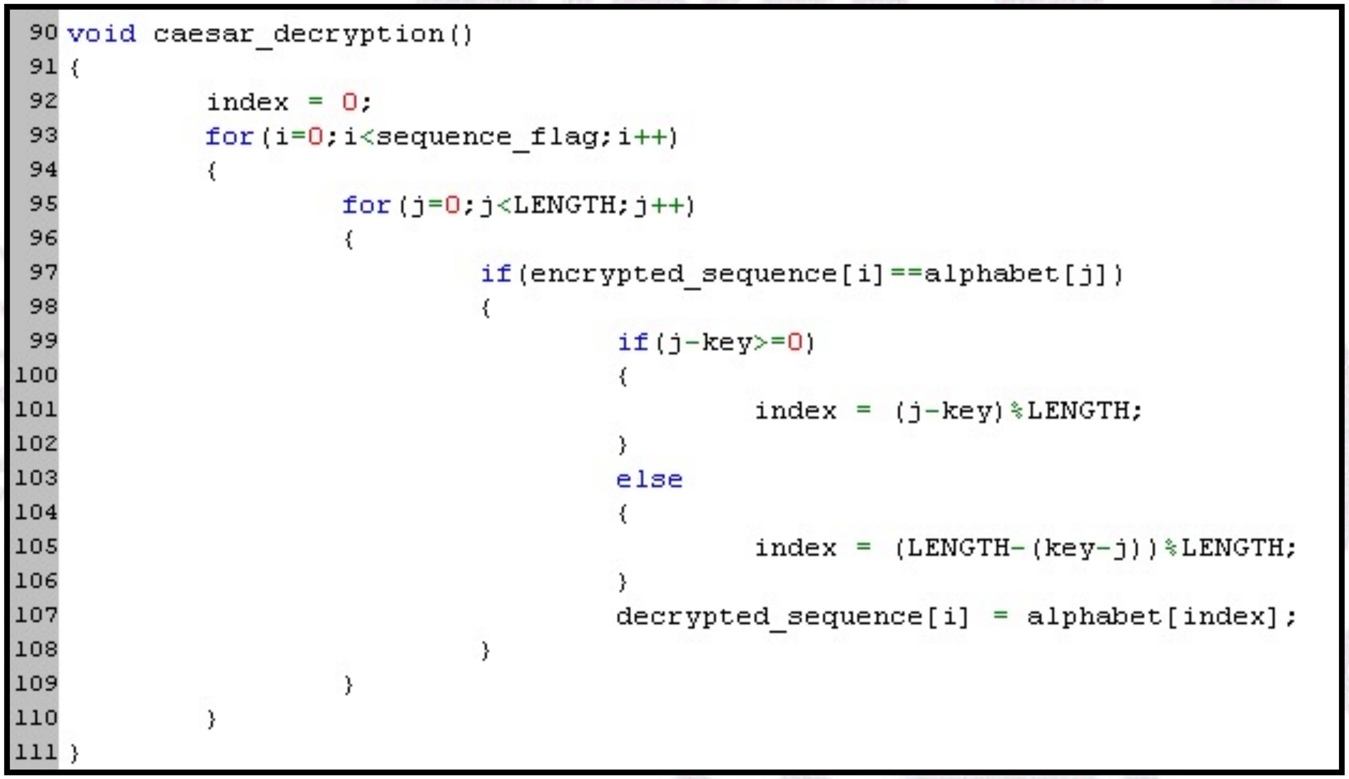
- JH Kim : Caesar, Affine, Vigenere, Autokey
- GM Lee : Hill, Exponentiation, RSA

Caesar Cipher



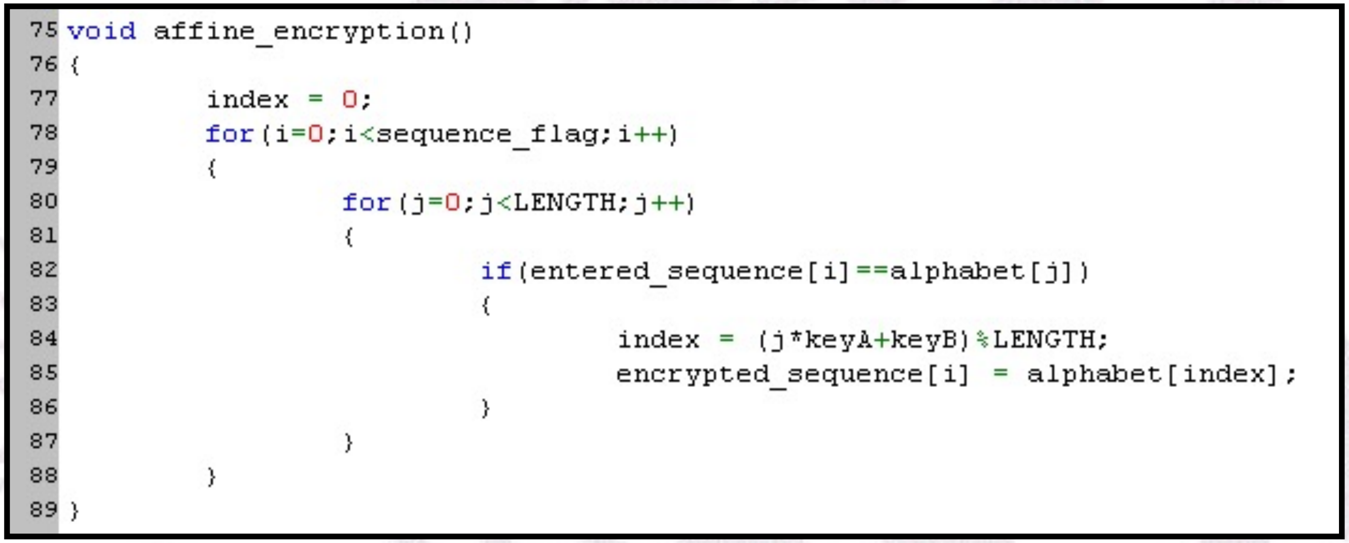
```
void caesar_encryption()
{
        index = 0;
        for(i=0;i<sequence_flag;i++)
        {
                for(j=0;j<LENGTH;j++)
                {
                        if(entered_sequence[i]==alphabet[j])
                        {
                                index = (j+key)%LENGTH;
                                encrypted_sequence[i] = alphabet[index];
                        }
                }
        }
}
```

Caesar Cipher



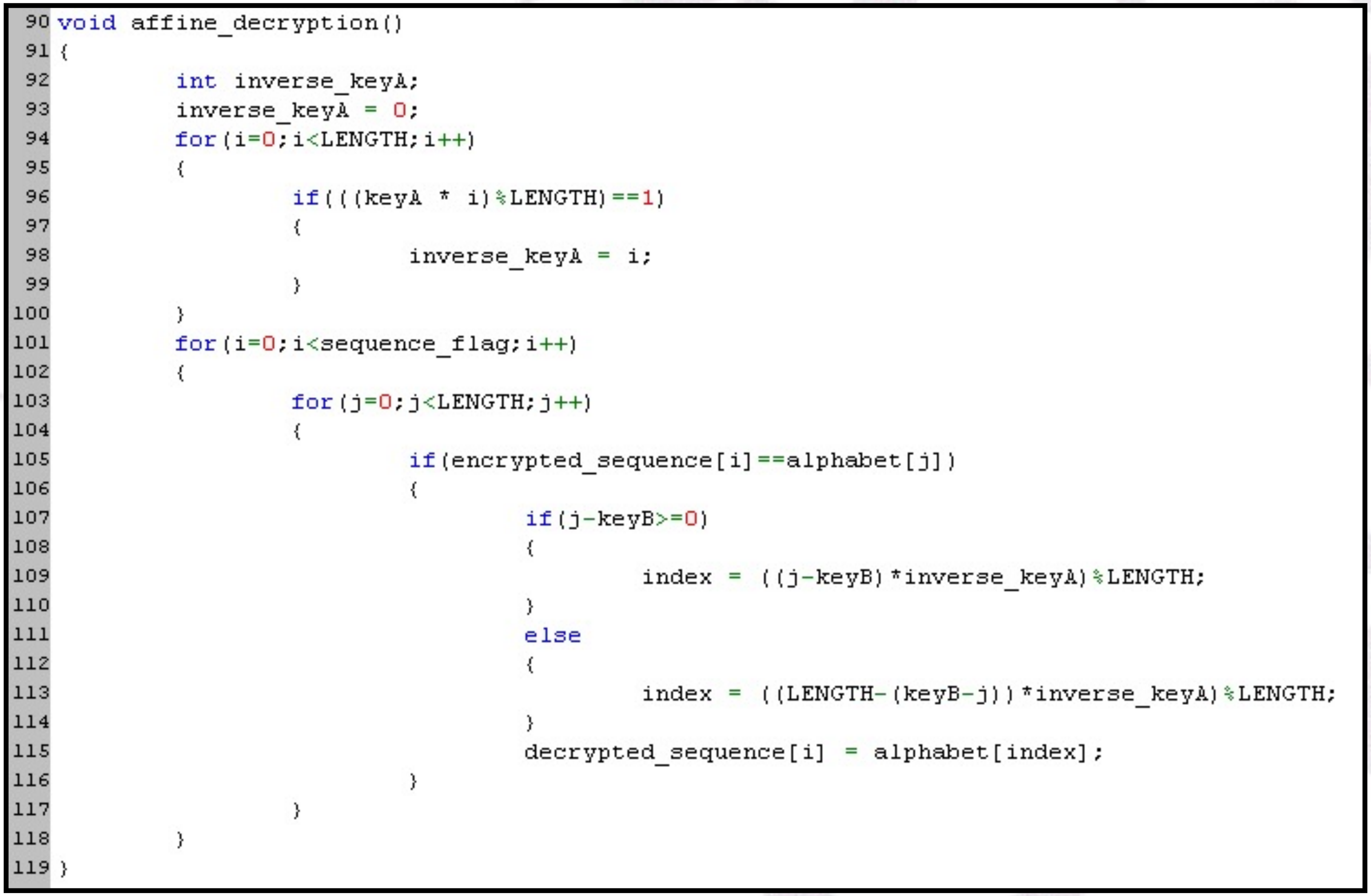
```
void caesar_decryption()
{
        index = 0;
        for(i=0;i<sequence_flag;i++)
        {
                for(j=0;j<LENGTH;j++)
                {
                        if(encrypted_sequence[i]==alphabet[j])
                        {
                                if(j-key>=0)
                                {
                                        index = (j-key)%LENGTH;
                                }
                                else
                                {
                                        index = (LENGTH-(key-j))%LENGTH;
                                }
                                decrypted_sequence[i] = alphabet[index];
                        }
                }
        }
}
```

Affine Cipher



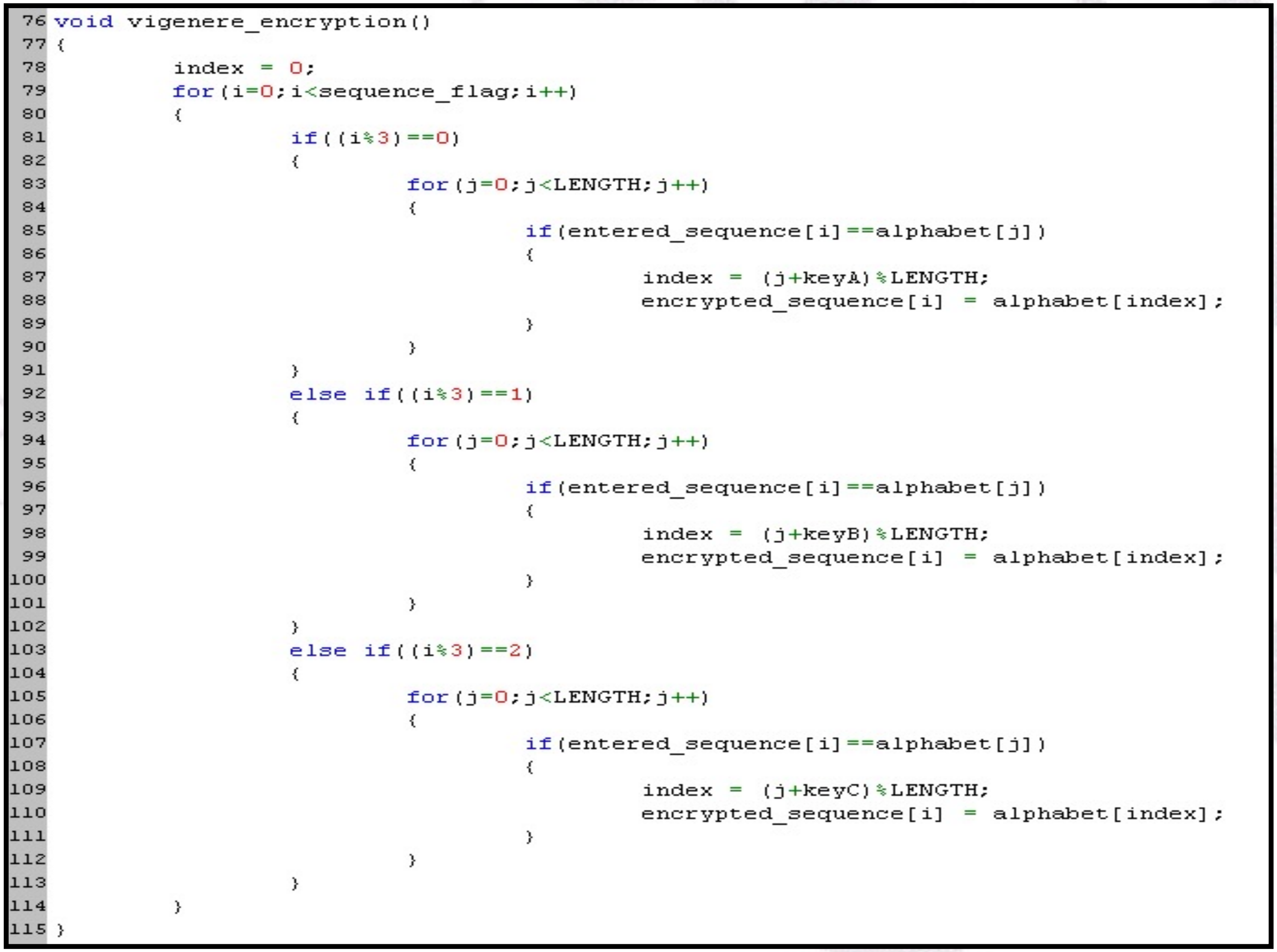
```
void affine_encryption()
{
        index = 0;
        for(i=0;i<sequence_flag;i++)
        {
                for(j=0;j<LENGTH;j++)
                {
                        if(entered_sequence[i]==alphabet[j])
                        {
                                index = (j*keyA+keyB)%LENGTH;
                                encrypted_sequence[i] = alphabet[index];
                        }
                }
        }
}
```

Affine Cipher



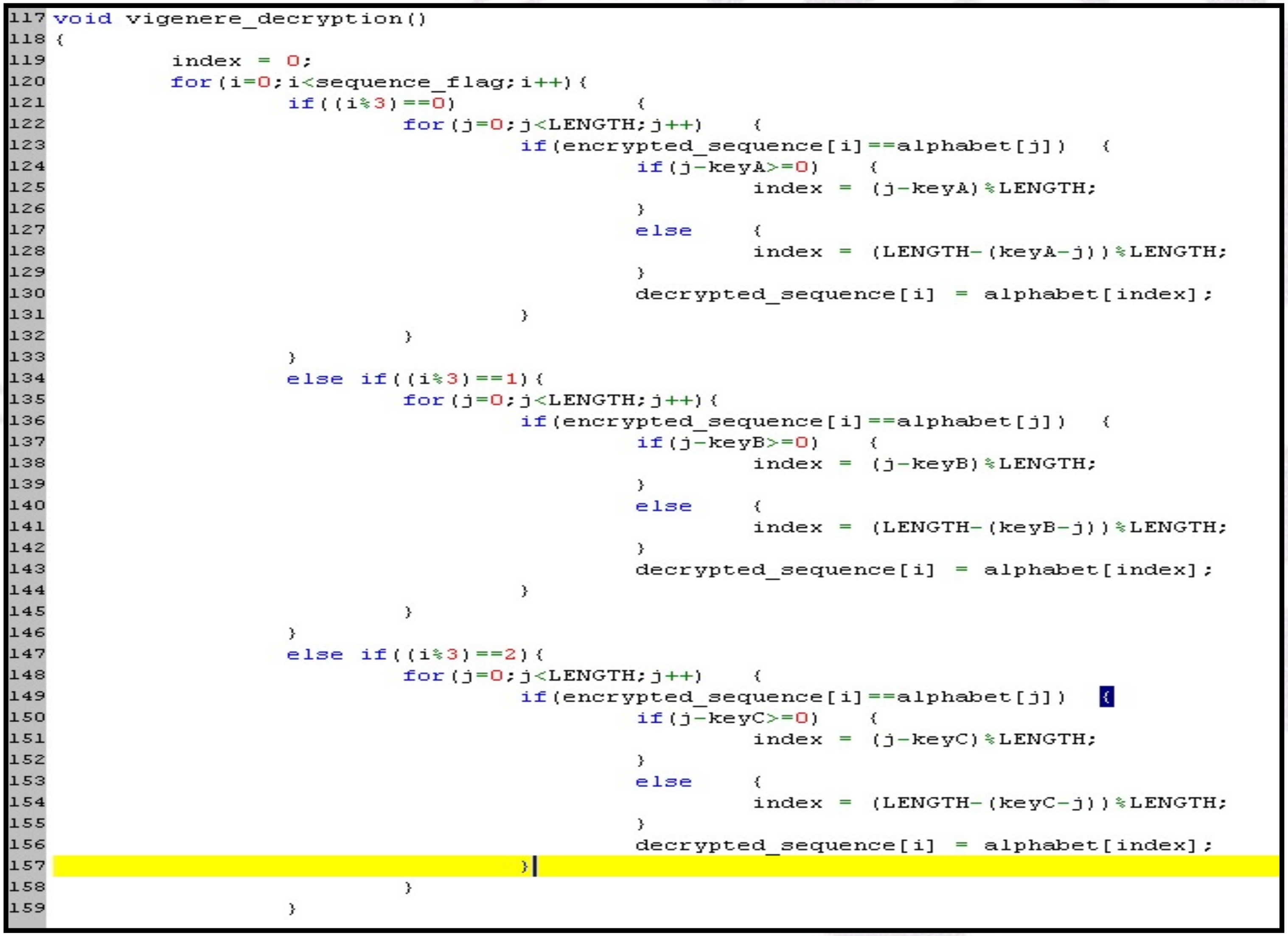
```
void affine_decryption()
{
        int inverse_keyA;
        inverse_keyA = 0;
        for(i=0;i<LENGTH;i++)
        {
                if(((keyA * i)%LENGTH)==1)
                {
                        inverse_keyA = i;
                }
        }
        for(i=0;i<sequence_flag;i++)
        {
                for(j=0;j<LENGTH;j++)
                {
                        if(encrypted_sequence[i]==alphabet[j])
                        {
                                if(j-keyB>=0)
                                {
                                        index = ((j-keyB)*inverse_keyA)%LENGTH;
                                }
                                else
                                {
                                        index = ((LENGTH-(keyB-j))*inverse_keyA)%LENGTH;
                                }
                                decrypted_sequence[i] = alphabet[index];
                        }
                }
        }
}
```

Vigenere Cipher



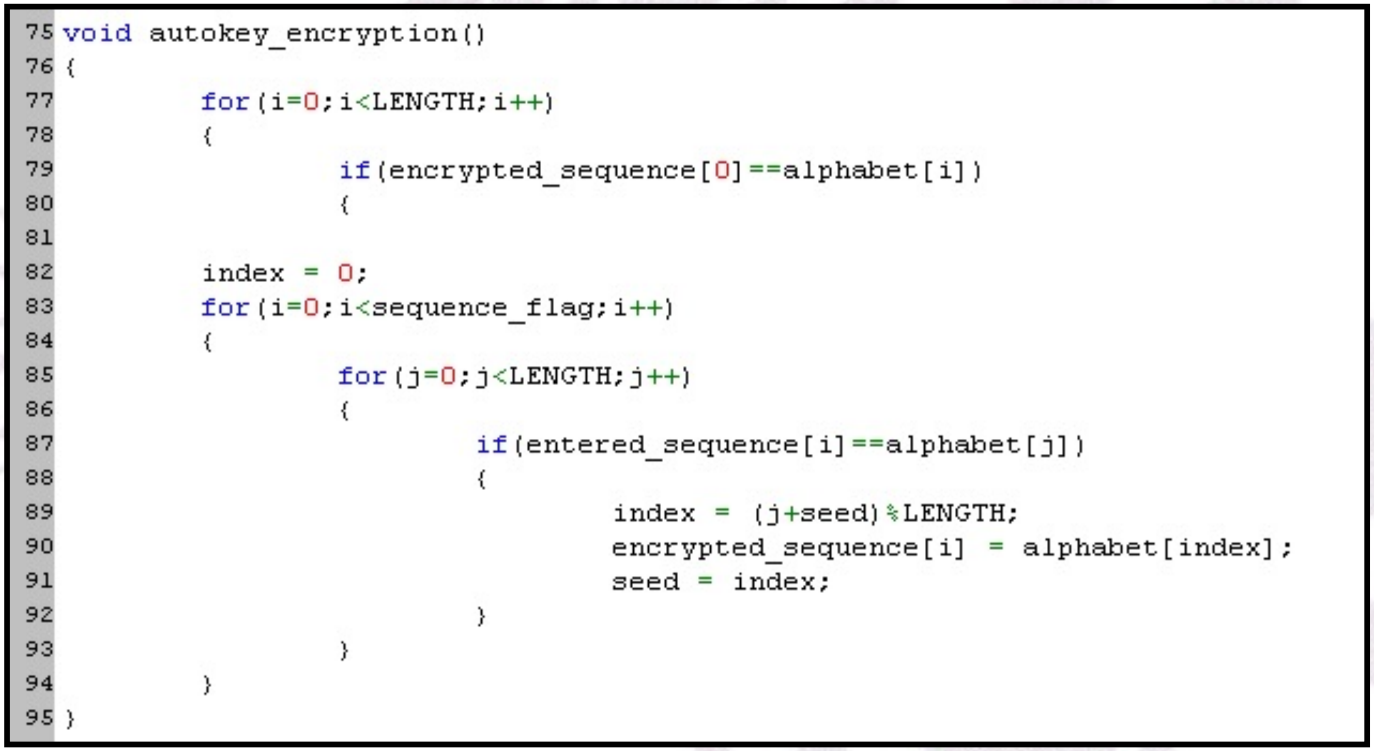
```
void vigenere_encryption()
{
        index = 0;
        for(i=0;i<sequence_flag;i++)
        {
                if((i%3)==0)
                {
                        for(j=0;j<LENGTH;j++)
                        {
                                if(entered_sequence[i]==alphabet[j])
                                {
                                        index = (j+keyA)%LENGTH;
                                        encrypted_sequence[i] = alphabet[index];
                                }
                        }
                }
                else if((i%3)==1)
                {
                        for(j=0;j<LENGTH;j++)
                        {
                                if(entered_sequence[i]==alphabet[j])
                                {
                                        index = (j+keyB)%LENGTH;
                                        encrypted_sequence[i] = alphabet[index];
                                }
                        }
                }
                else if((i%3)==2)
                {
                        for(j=0;j<LENGTH;j++)
                        {
                                if(entered_sequence[i]==alphabet[j])
                                {
                                        index = (j+keyC)%LENGTH;
                                        encrypted_sequence[i] = alphabet[index];
                                }
                        }
                }
        }
}
```

Vigenere Cipher



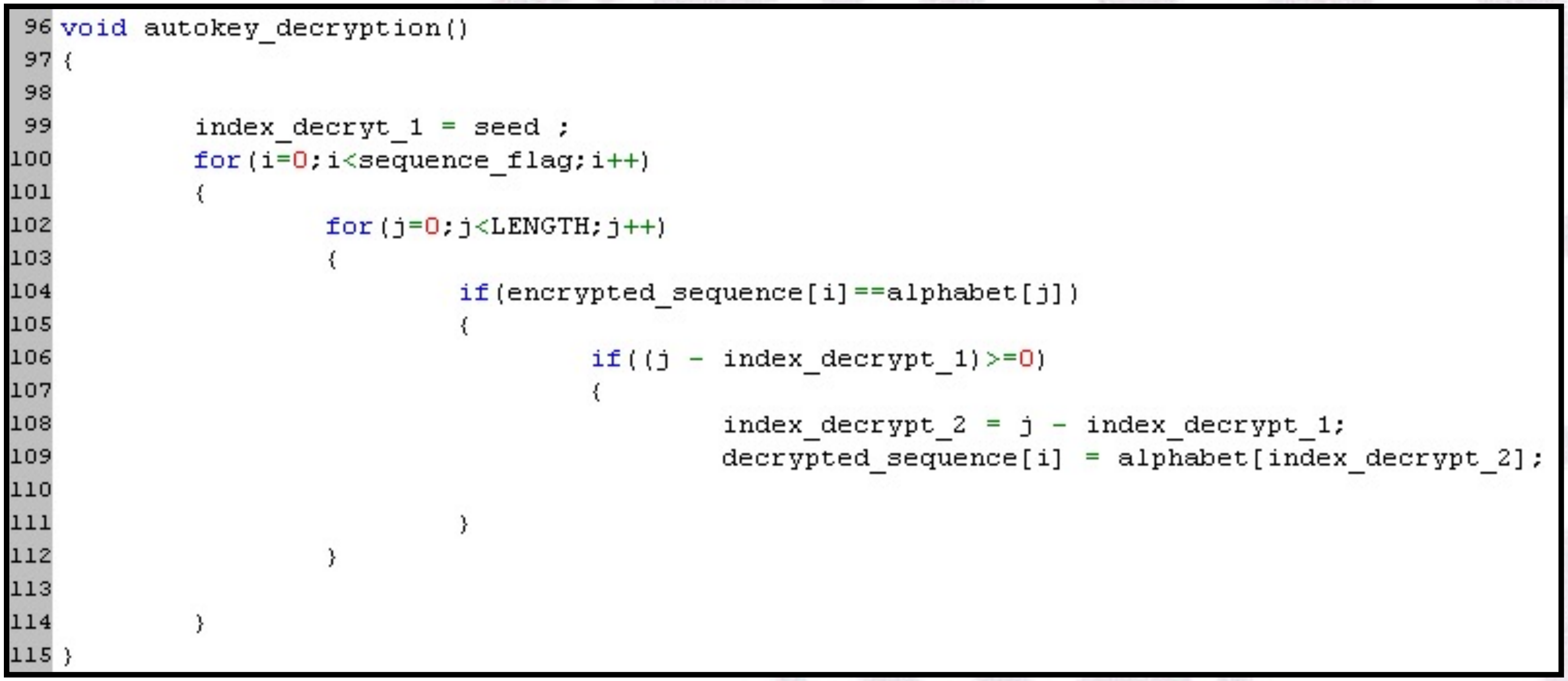
```
void vigenere_decryption()
{
    index = 0;
    for(i=0;i<sequence_flag;i++){
        if((i%3)==0)        {
            for(j=0;j<LENGTH;j++)   {
                if(encrypted_sequence[i]==alphabet[j])  {
                    if(j-keyA>=0)   {
                        index = (j-keyA)%LENGTH;
                    }
                    else    {
                        index = (LENGTH-(keyA-j))%LENGTH;
                    }
                    decrypted_sequence[i] = alphabet[index];
                }
            }
        }
        else if((i%3)==1){
            for(j=0;j<LENGTH;j++){
                if(encrypted_sequence[i]==alphabet[j])  {
                    if(j-keyB>=0)   {
                        index = (j-keyB)%LENGTH;
                    }
                    else    {
                        index = (LENGTH-(keyB-j))%LENGTH;
                    }
                    decrypted_sequence[i] = alphabet[index];
                }
            }
        }
        else if((i%3)==2){
            for(j=0;j<LENGTH;j++)   {
                if(encrypted_sequence[i]==alphabet[j])  {
                    if(j-keyC>=0)   {
                        index = (j-keyC)%LENGTH;
                    }
                    else    {
                        index = (LENGTH-(keyC-j))%LENGTH;
                    }
                    decrypted_sequence[i] = alphabet[index];
                }
            }
        }
```

Autokey Cipher



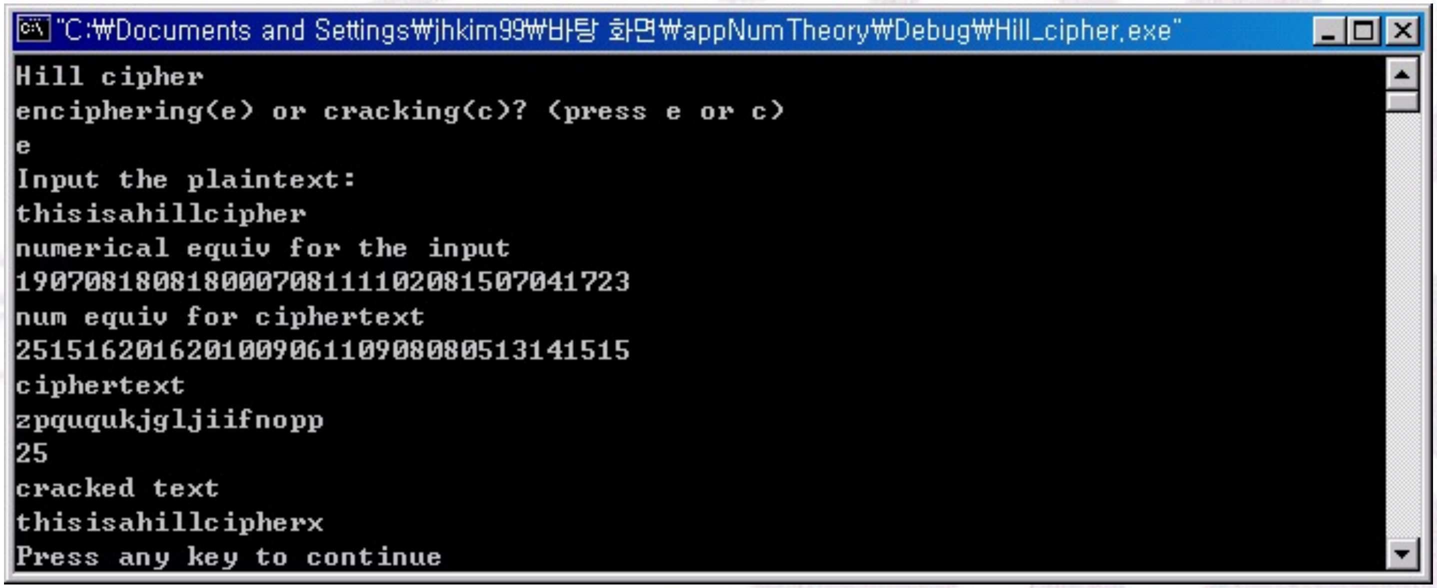
```
void autokey_encryption()
{
        for(i=0;i<LENGTH;i++)
        {
                if(encrypted_sequence[0]==alphabet[i])
                {

        index = 0;
        for(i=0;i<sequence_flag;i++)
        {
                for(j=0;j<LENGTH;j++)
                {
                        if(entered_sequence[i]==alphabet[j])
                        {
                                index = (j+seed)%LENGTH;
                                encrypted_sequence[i] = alphabet[index];
                                seed = index;
                        }
                }
        }
}
```

Autokey Cipher



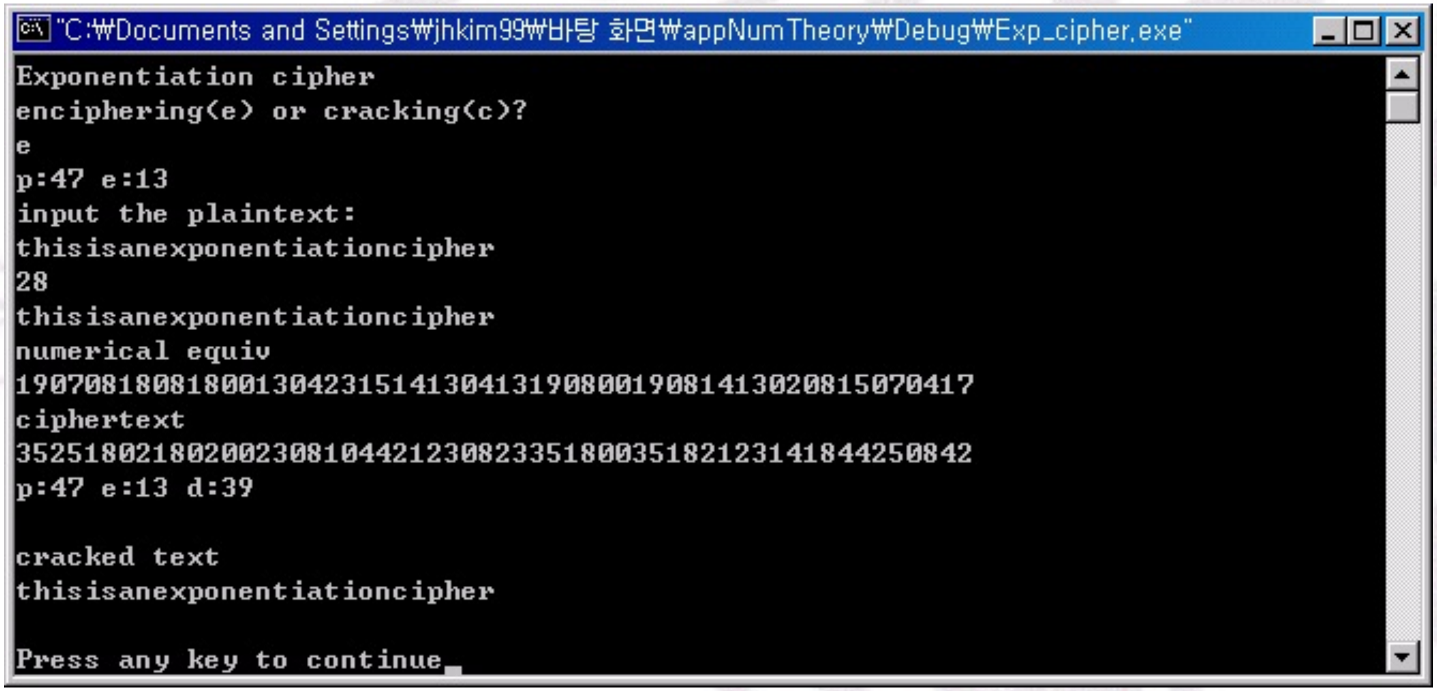
```
void autokey_decryption()
{

        index_decryt_1 = seed ;
        for(i=0;i<sequence_flag;i++)
        {
                for(j=0;j<LENGTH;j++)
                {
                        if(encrypted_sequence[i]==alphabet[j])
                        {
                                if((j - index_decrypt_1)>=0)
                                {
                                        index_decrypt_2 = j - index_decrypt_1;
                                        decrypted_sequence[i] = alphabet[index_decrypt_2];

                        }
                }

        }
}
```

Hill Cipher


"C:₩Documents and Settings₩jhkim99₩바탕 화면₩appNumTheory₩Debug₩Hill_cipher.exe"

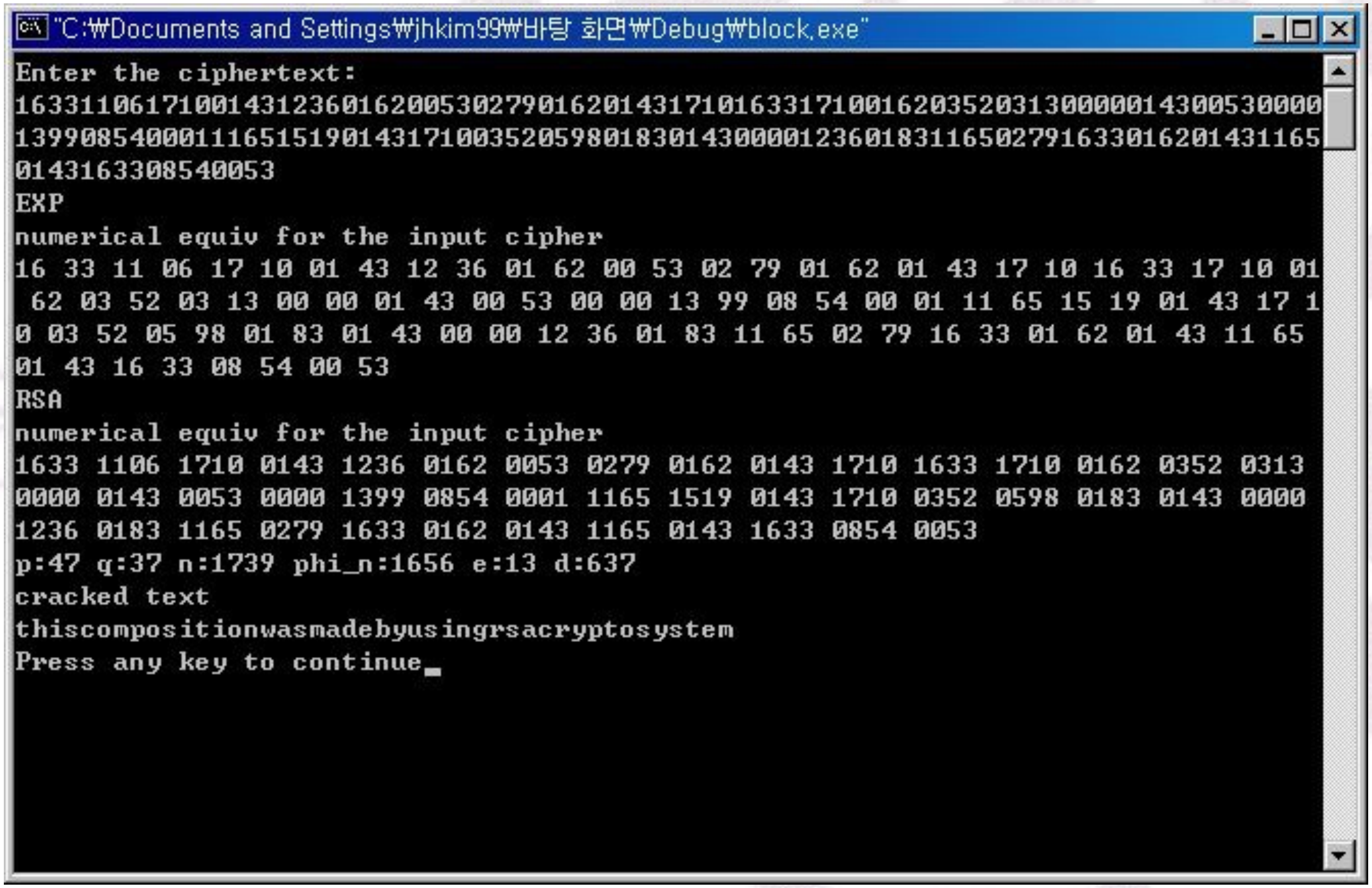
```
Hill cipher
enciphering(e) or cracking(c)? (press e or c)
e
Input the plaintext:
thisisahillcipher
numerical equiv for the input
190708180818000708111102081507041723
num equiv for ciphertext
251516201620100906110908080513141515
ciphertext
zpququkjgljiifnopp
25
cracked text
thisisahillcipherx
Press any key to continue
```

Exponentiation Cipher


"C:₩Documents and Settings₩jhkim99₩바탕 화면₩appNumTheory₩Debug₩Exp_cipher.exe"

```
Exponentiation cipher
enciphering(e) or cracking(c)?
e
p:47 e:13
input the plaintext:
thisisanexponentiationcipher
28
thisisanexponentiationcipher
numerical equiv
19070818081800130423151413041319080019081413020815070417
ciphertext
35251802180200230810442123082335180035182123141844250842
p:47 e:13 d:39

cracked text
thisisanexponentiationcipher

Press any key to continue
```

RSA Cryptosystem



"C:₩Documents and Settings₩jhkim99₩바탕 화면₩Debug₩block.exe"

```
Enter the ciphertext:
1633110617100143123601620053027901620143171016331710016203520313000001430053000
1399085400011165151901431710035205980183014300001236018311650279163301620143116
501431633085400 53
EXP
numerical equiv for the input cipher
16 33 11 06 17 10 01 43 12 36 01 62 00 53 02 79 01 62 01 43 17 10 16 33 17 10 01
 62 03 52 03 13 00 00 01 43 00 53 00 00 13 99 08 54 00 01 11 65 15 19 01 43 17 1
0 03 52 05 98 01 83 01 43 00 00 12 36 01 83 11 65 02 79 16 33 01 62 01 43 11 65
01 43 16 33 08 54 00 53
RSA
numerical equiv for the input cipher
1633 1106 1710 0143 1236 0162 0053 0279 0162 0143 1710 1633 1710 0162 0352 0313
0000 0143 0053 0000 1399 0854 0001 1165 1519 0143 1710 0352 0598 0183 0143 0000
1236 0183 1165 0279 1633 0162 0143 1165 0143 1633 0854 0053
p:47 q:37 n:1739 phi_n:1656 e:13 d:637
cracked text
thiscompositionwasmadebyusingrsacryptosystem
Press any key to continue_
```

Outline

Contents

Introduction

Implementation

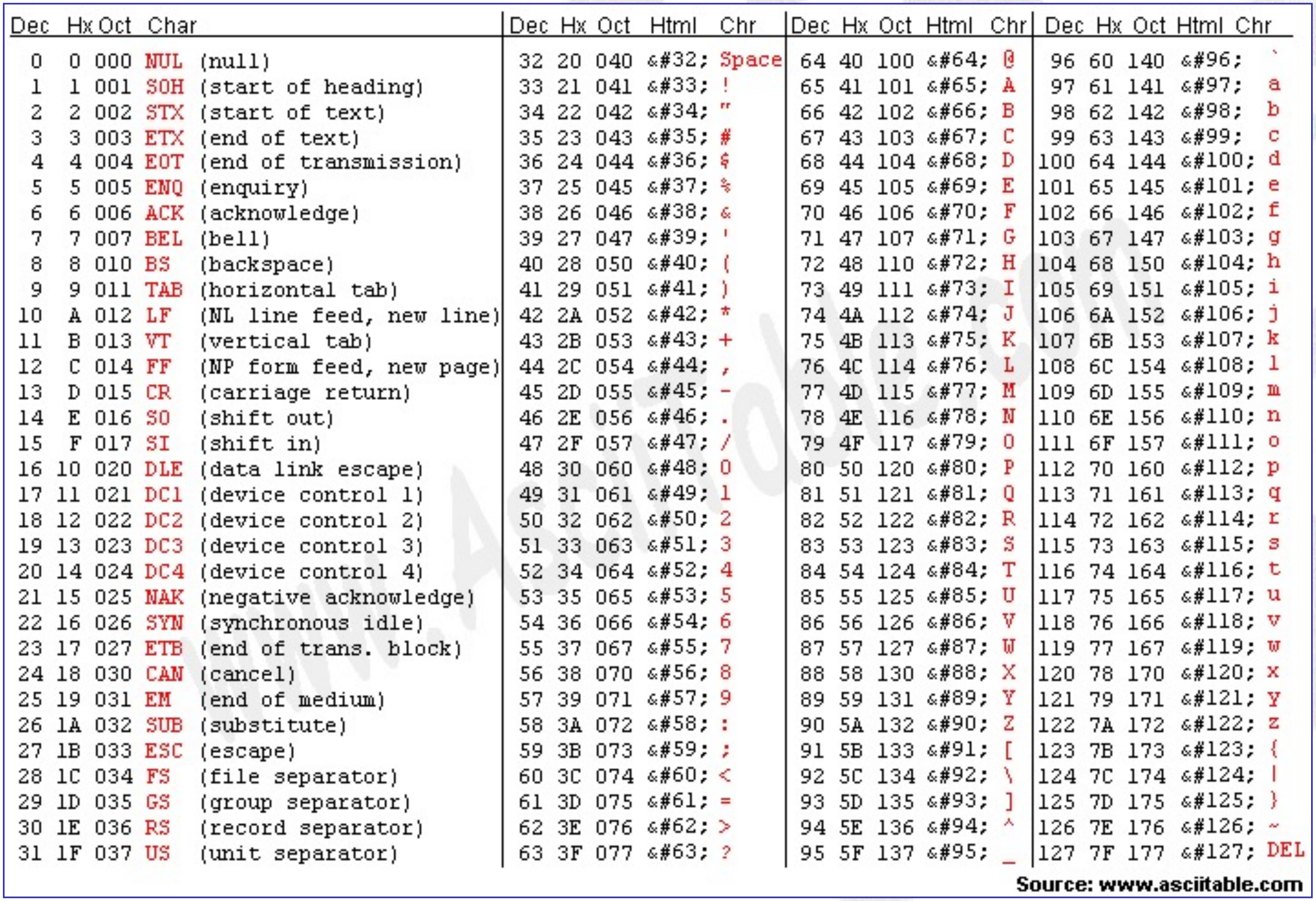
Conclusion and Improvement

Q & A

Readability

- AFRIENDINNEEDISAFRIENDINDEED
- JOONGHEONKIMGENEMOOLEE

Readability



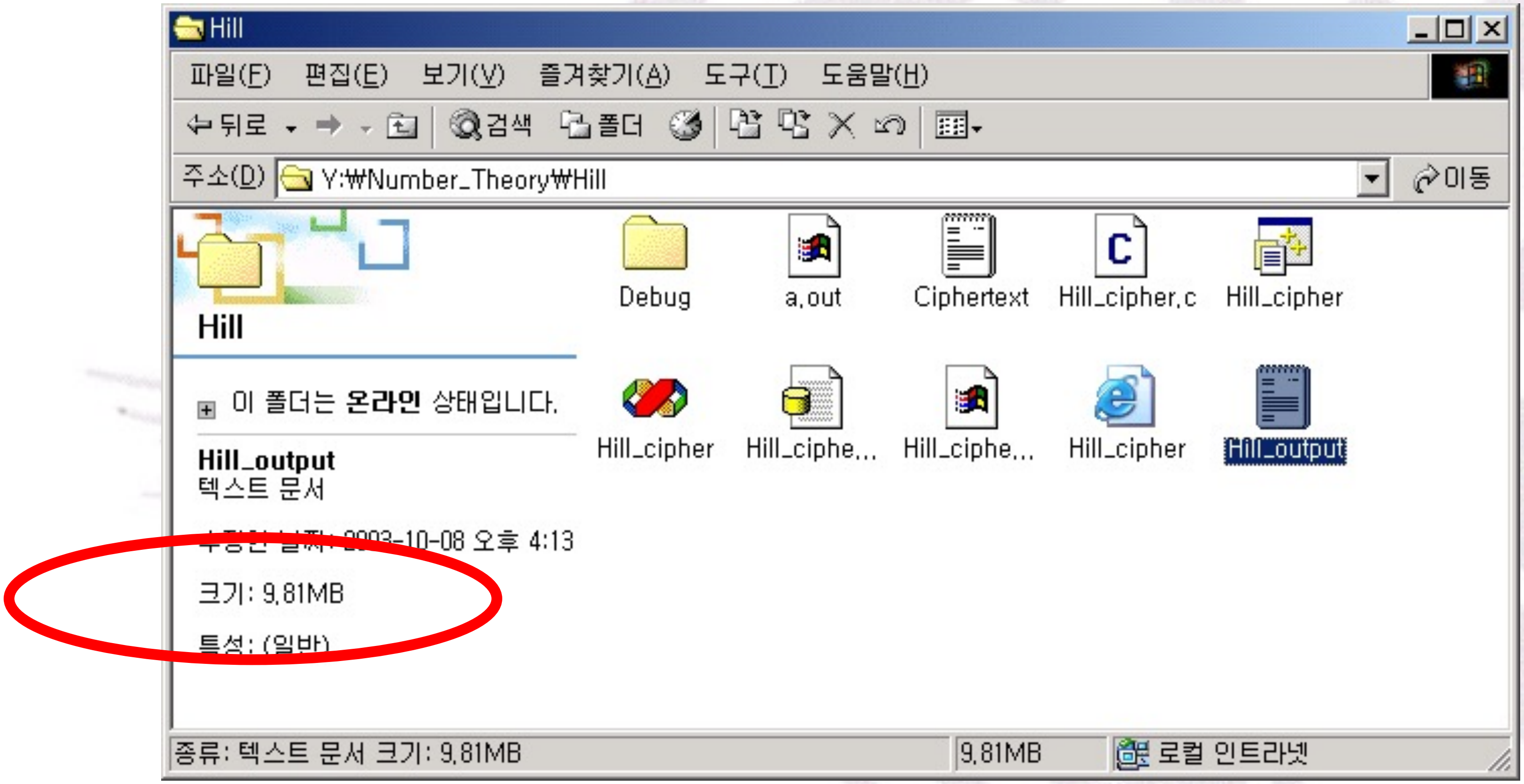
Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX	(end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT	(end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ	(enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK	(acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL	(bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS	(backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB	(horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT	(vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR	(carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO	(shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI	(shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE	(data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1	(device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2	(device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3	(device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4	(device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK	(negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN	(synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB	(end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN	(cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM	(end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB	(substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC	(escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS	(file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS	(group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS	(record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US	(unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

Robustness of Hill

Cipher	Number of Possible Cases
Caesar	26
Affine	312
Vigenere	9
Autokey	9
Hill	157248
Exponentiation	16
RSA	51

Robustness of Hill



Y:₩Number_Theory₩Hill

Hill

이 폴더는 **온라인** 상태입니다.

Hill_output
텍스트 문서

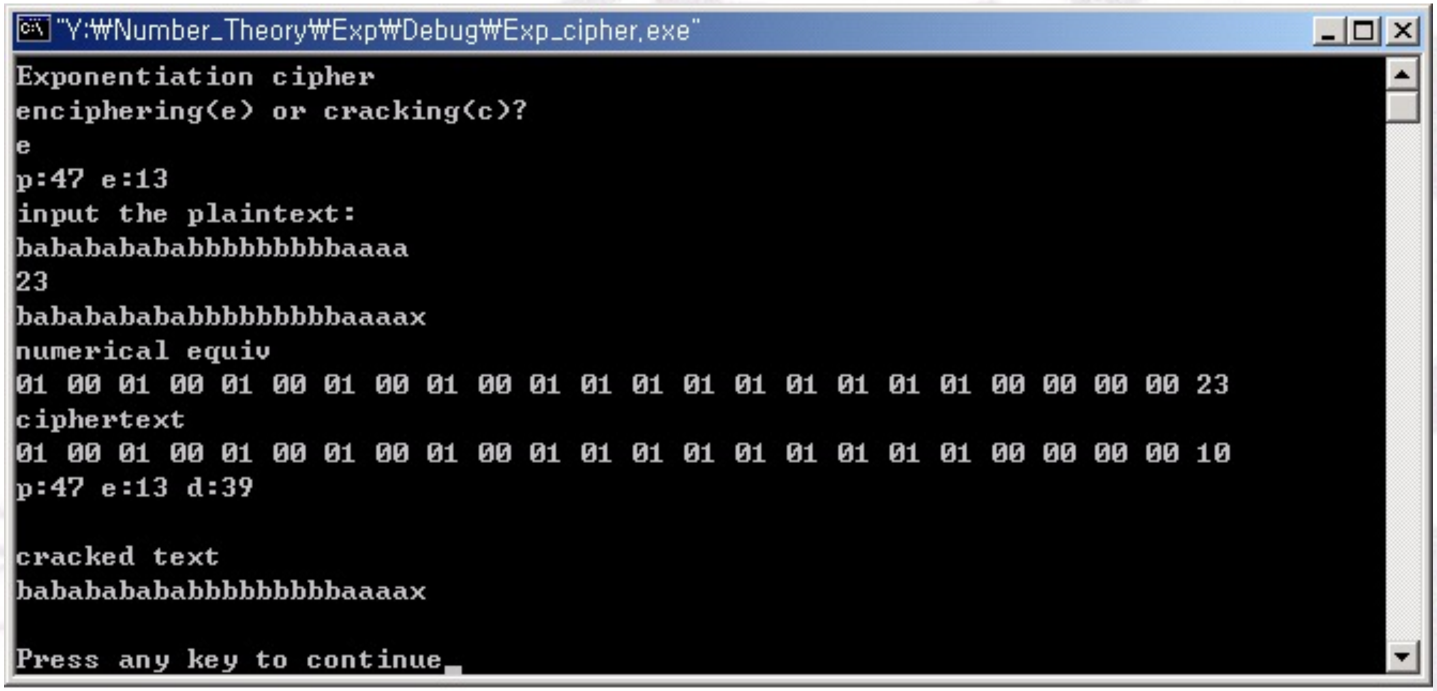
수정한 날짜: 2003-10-08 오후 4:13

크기: 9.81MB

특성: (일반)

종류: 텍스트 문서 크기: 9.81MB | 9.81MB | 로컬 인트라넷

Improving of EXP & RSA



```
Exponentiation cipher
enciphering(e) or cracking(c)?
e
p:47 e:13
input the plaintext:
babababababbbbbbbbbaaaa
23
babababababbbbbbbbbaaaax
numerical equiv
01 00 01 00 01 00 01 00 01 00 01 01 01 01 01 01 01 01 01 00 00 00 00 23
ciphertext
01 00 01 00 01 00 01 00 01 00 01 01 01 01 01 01 01 01 01 00 00 00 00 10
p:47 e:13 d:39

cracked text
babababababbbbbbbbbaaaax

Press any key to continue
```

Using additional cipher system,
Remove the filtered characters.

Outline

Contents

Introduction

Implementation

Conclusion and Improvement

Q & A

Any Questions?