

## CRYPTOGRAPHY

S. SUNDAR

**N**ews item in 'The Times of India', Mumbai, on June 27, 1999: "The Indian Institute of Technology, Kanpur, has developed a new-generation cipher code for the Indian navy. The breakthrough would provide a technological edge to defence communication. Christened 'Trinetra', the cipher is a modern computer-based code language system which can digitise long, alphabetic messages within seconds."

This article is about cryptography—the science of scrambling data to make it unintelligible to all, except the intended person. It has come to focus recently due to the expanding influence of e-commerce. Almost all the credit card transactions taking place on the Internet or elsewhere make use of cryptography. This concept is also used in 'secure server' transactions, where the data is guarded by the use of cryptography.

Some of the terms used in this article are explained below:

*Plaintext* is the text or data that is to be encrypted. It is one of the inputs to the encryption algorithm.

*Key* is the password used to decrypt the coded message.

*Ciphertext* is the output of the encryption algorithm. It can be transmitted safely as it is unintelligible without the key.

One of the earliest known and simplest methods of encryption is the 'Caesar Cipher' method, which involves replacement of each letter in the message with some other letter or numeral. The principle of 'bit replacement technique' is

that a letter may be replaced by another letter three places (offset or key) down the order; for example, the letter 'a' will become 'd', and so on.

Here, lesser number of keys are possible. Also, the encryption and decryption algorithms are known and the language is understood easily. Because of these factors, an advanced computer using either

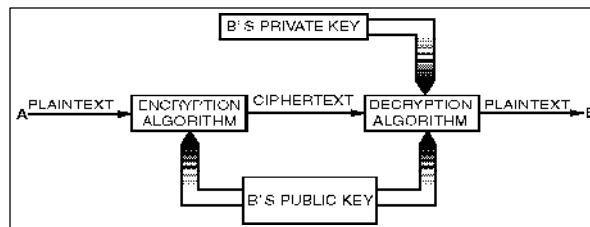


Fig. 2: Public-key encryption

parallel processing or distributed computing can crack this method. This type of cracking is termed as 'Brute Force Attack' in computer parlance.

International Data Encryption Algorithm (IDEA) and Data Encryption Standard (DES) are two algorithms that use the bit-replacement technique explained above. These algorithms, along with compression, are used to send secure e-mail using software programs like PGP (Pretty Good Privacy)—the present standard for encrypted mail. In PGP, the public-key encryption explained in the later part of the article is also used.

### International data encryption algorithm (IDEA)

The Swiss Federal Institute of Technology developed IDEA in 1990. This method uses a 128-bit key; only the powers of two are used as key-size. If the key is too large, the computations involved in encryption and decryption make the working of the algorithm slower; and if it is too small, the algorithm becomes insecure. As with any other bit replacement scheme, there are two inputs, namely, the plain text and the key.

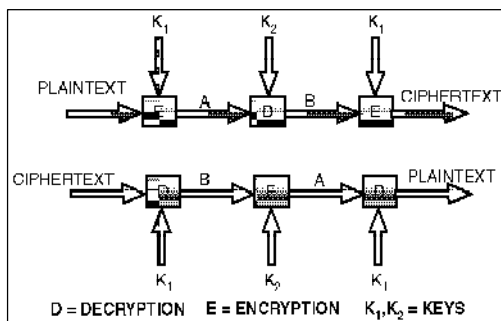


Fig. 1: Triple DES

Here, the number 128 gives the size of the key in bits. This key size gives a key space (total number of combinations of bits in the key) of  $3.4 \times 10^{38}$  keys, which makes the 'Brute force attack' impractical. Higher the key size, tougher it is to crack the ciphertext. In IDEA, eight rounds or iterations of data manipulation are done to generate the final output.

The manipulations may consist of one or a combination of the following three operations:

Bit-by-Bit exclusive-ORing

Addition of integers modulo  $2^{16}$

Multiplication of integers modulo  $2^{16}$

The expression 'x mod y' gives the remainder when x is divided by y. The value of y is referred to as the modulus of the

expression; for example,  $10 \text{ mod } 3 = 1$ . The use of these manipulations in a sequential order produces a complex transformation of the input data, making crypt analysis difficult.

### Data encryption standard (DES)

In DES, the plaintext ought to be of 64 bits and the key of 56 bits. Longer blocks of plaintext are encrypted in 64-bit blocks. Each block of 64-bit input is passed through 16 iterations, with each iteration producing an intermediate 64-bit value.

The iteration is essentially the same complex function that involves a permutation of bits and substitution of one bit pattern by another. A variation of DES, the 'Triple DES' that uses two keys and three executions of the algorithm, has gained considerable success. Triple DES employs an encrypt-decrypt-encrypt sequence. There is no cryptographic significance to the use of decryption for the second stage. It only allows users of triple DES to decrypt data encoded by users of the older single DES.

### Public-key encryption

Another commonly used method for encryption is the public-private key encryption. This method uses two passwords, one termed as *public key* and the other *private key*. The algorithms used to generate keys are based on mathematical numbers generated by the computer on user's request, instead of simple bit operations.

An analogy of public-key encryption is

safe with a special type of lock that can be locked/unlocked with one or two different keys. A left key turns the mechanism to the left and a right key turns it to the right. In the unlocked state, the mechanism is in the centre position. If locked with the right key, the only way to unlock it is by using the left key and vice-versa. If 'A' wants to send a message to 'B', he/she can put the message in the box and lock it

How to distribute the secret key was the most difficult problem for conventional encryption. This problem is eliminated by the public-key encryption, considering that private key is never distributed.

### RSA

Ron Rivest, Adi Shamir, and Len Adleman at MIT (Massachusetts Institute of Tech-

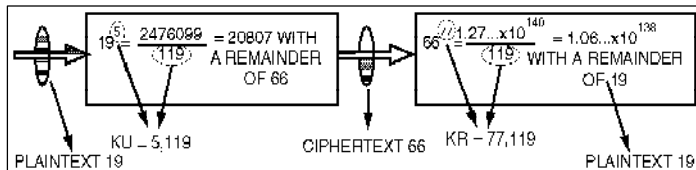


Fig. 3: Example of RSA algorithm

with the right key. On receiving the message, 'B' can unlock the box using the corresponding left key and read the message.

The lock will open only when a particular key combination is used. Suppose a group of people wants to share messages using this method. Then everyone in the group buys one's own safe box and freely supplies the respective right key duplicates but keeps one's left keys to oneself. For communication, one can just write the message, put it in the safe, lock it using the right key of the recipient, and pass it to the recipient who uses one's own left key to unlock it.

Thus, the message can be transmitted safely, since apart from the intended recipient no one else would be having the correct left key.

A similar technique is used in public-key encryption. In place of a right key, a public key is used while encrypting and, in place of a left key, a private key is used for decrypting.

Once a key is lost, the data encrypted using the particular key can't be recovered. So backup copies of all the keys used must be kept necessarily.

Furthermore, these algorithms have the following important characteristics:

- It is computationally unfeasible to determine the decryption key with the knowledge of the cryptography and the encryption key.

- Either of the two related keys can be used for encryption.

With conventional encryption (bit replacement technique as opposed to public-key encryption), a fundamental requirement for two parties to communicate securely was to share a secret key.

only widely accepted and implemented approach to public-key encryption. RSA is a cipher in which the plaintext and the ciphertext are integers between 0 and 'n-1', for some valued 'n'. For a long message, the message is broken into blocks. These blocks are  $\log_2 n$  bits long. Two high-value prime numbers are multiplied to get 'n' so that extracting the value of 'n' becomes very difficult. As the public and private keys are dependent on the value of 'n', guessing this value is very difficult.

Encryption and decryption make use of modular arithmetic. The RSA algorithm for plaintext block M and ciphertext block C with keys 'e' and 'd' is given below:

**Key generation**  
 Select p,q                    p and q are both prime  
 Calculate  $n = p \times q$   
 Calculate  $f(n) = (p-1)(q-1)$   
 Select integer e             $\text{gcd}(f(n),e)=1; 1 < e < f(n)$   
 Calculate d                     $d = e^{-1} \text{ mod } f(n)$   
 Public Key                    KU = {e,n}  
 Private Key                    KR = {d,n}

**Encryption**  
 Plaintext:                    M < n  
 Ciphertext:                     $C = M^e \text{ (mod } n)$

**Decryption**  
 Ciphertext:                    C  
 Plaintext:                     $M = C^d \text{ (mod } n)$

**The RSA Algorithm**  
 $C = M^e \text{ mod } n$   
 $M = C^d \text{ mod } n = (M^e)^d \text{ mod } n = M^{ed} \text{ mod } n$

Both sender and receiver must know the value of 'n' and 'e'; whereas the value of d is known only to the receiver. Thus, this is a public-key encryption algorithm in which the public key is {e,n} and the private key is {d,n}. For public-key encryption, this algorithm will be satisfactory if the following conditions are fulfilled:

filled:

- Values of e, d, and n exist such that  $M = M^{ed} \text{ mod } n$ , for all  $M < n$ .

- It should be relatively easy to calculate  $M^e$  and  $C^d$  for all values of  $M < n$ .

- It should be unfeasible to determine d, even if the values of e and n are known.

The third condition would be satisfied only for large values of e and n.

Another method, the Secure Hash Function (developed by Ron Rivest, 'R' of RSA), is based on public-key encryption. With the advent of Internet, intense research is going on in this field, using diverse technologies which employ distributed networking and the use of the inherently present noise in transmission channel to encrypt data. The five steps involved in this algorithm are detailed below:

**Step 1: Append padding bits.** The message is padded such that the length of the padded message is 64 bits less than an integer multiple of 512 bits. Padding is necessarily done even if the message is already of desired length; for example, even a message of 448 (512-64) bits is made to 960 (512x2-64) bits by padding 512 bits to it. Thus, the number of padding bits lies between 1 and 512. The padding consists of a single -1- bit followed by the necessary number of -0- bits.

**Step 2: Append length.** A 64-bit representation of the length in bits of the original message (before padding) is appended to the result of Step 1. The outcome is a message which is an integer multiple of 512 bits in length. The message is represented in blocks of 512 bits  $Y_0, Y_1, Y_2, \dots, Y_{L-1}, \dots$ . So, the total length of the expanded message is  $L \times 512$ .

**Step 3: Initialise MD buffer.** A 128-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as four 32-bit registers (A, B, C, and D). These registers are initialised to the following hexadecimal values:

A = 01234567  
 B = 89ABCDEF  
 C = FEDCBA98  
 D = 76543210

**Step 4: Process message in 512-bit blocks.** The heart of the algorithm is the module that consists of four rounds of processing. These rounds may have a similar structure, but each may use a different primitive logical function, like AND, OR, etc.

**Step 5: Output.** After all L 512-bit blocks have been processed, the output

from the Lth stage is the 128-bit message digest.

Starting from the simple bit-replacement technique to computer-based code language systems, the science of cryptog-

raphy has indeed come a long way. The Indian system of 'Trinetra' can even digitise and encode the voice of a sender. Cryptographers have come up with various schemes considered to be virtually un-

breakable, only to find out later that they were not so. And thus, public-key encryption remains the standard for secure encryption till date.

## MORSE TUTOR

YUJIN BOBY

**M**orse tutor is a program written in C++. It can be used for learning or sending Morse code. Morse code uses long (dash) and short (dot) sounds to communicate. The dot is the basic unit and the dash is equal to the length of three dots.

Morse code was developed by Samuel Morse in 1897. Due to advantages like overriding noise, static and simple transmitter, and low power requirements, it is still used for communication.

The program provides character mode, line mode, and file mode of operations. In

character mode, separate characters can be practiced. In file mode, stored messages can be sent. This mode also has provision for creating a new message file. The speed can be set between 5 WPM (words per minute) and 20 WPM, and the tone can be varied between 100 Hz and 4,000 Hz using the program. The default speed is 12 WPM and the tone is 700 Hz.

### Program Listing in C++

```
#include<iostream.h>
#include<conio.h>
#include<dos.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
void morse() ;
int mnu_one();
void mode_line() ;
void mode_file() ;
void mode_sett() ;
void mode_char() ;
char in ;
int FREQ=700; // default tone 700 Hz
int DOT=50; // default speed 12 wpm
void main()
{
int choice ;
while (1)
{
choice=mnu_one() ;
if (choice==1) mode_char() ;
else if (choice==2) mode_line () ;
else if(choice==3) mode_file () ;
else if(choice==4) mode_sett () ;
else if(choice==5) break ;
}
clrscr() ;
cout<<endl<<"Thank you for using" ;
highvideo() ; cprintf("MORSE TUTOR") ;
normvideo() ;
cout<<endl<<" 73 & Good Bye..." ;
cout<<endl<<"de" ;
cout<<endl<<"\t\tYUJIN BOBY, Radio
VU3PRX";
cout<<endl<<"\t\tKoilparambil, Arthinkal" ;
cout<<endl<<"\t\tKerala, PIN - 688 530" ;
cout<<endl<<endl ;
}
void morse ()
{
char code [8] ;
switch (toupper(in))
{
case 'A' : strcpy (code, ".-") ; break;
```

```
case 'B' : strcpy (code, "-...") ; break;
case 'C' : strcpy (code, "-.-.") ; break;
case 'D' : strcpy (code, "-..") ; break;
case 'E' : strcpy (code, ".") ; break;
case 'F' : strcpy (code, ".-..") ; break;
case 'G' : strcpy (code, "--.") ; break;
case 'H' : strcpy (code, "....") ; break;
case 'I' : strcpy (code, "..") ; break;
case 'J' : strcpy (code, ".-.-") ; break;
case 'K' : strcpy (code, "-.-") ; break;
case 'L' : strcpy (code, "-..") ; break;
case 'M' : strcpy (code, "--") ; break;
case 'N' : strcpy (code, "-.") ; break;
case 'O' : strcpy (code, "---") ; break;
case 'P' : strcpy (code, "-.-.") ; break;
case 'Q' : strcpy (code, "--.-") ; break;
case 'R' : strcpy (code, ".-.") ; break;
case 'S' : strcpy (code, "...") ; break;
case 'T' : strcpy (code, ".-") ; break;
case 'U' : strcpy (code, "-.-") ; break;
case 'V' : strcpy (code, "...-") ; break;
case 'W' : strcpy (code, "-.-") ; break;
case 'X' : strcpy (code, "-.-.") ; break;
case 'Y' : strcpy (code, "--.-") ; break;
case 'Z' : strcpy (code, "--..") ; break;
case '1' : strcpy (code, ".----") ; break;
case '2' : strcpy (code, "-.-.-") ; break;
case '3' : strcpy (code, "--.-.-") ; break;
case '4' : strcpy (code, "-.-.-.-") ; break;
case '5' : strcpy (code, "-.-.-.-") ; break;
case '6' : strcpy (code, "-.-.-.-") ; break;
case '7' : strcpy (code, "-.-.-.-") ; break;
case '8' : strcpy (code, "-.-.-.-") ; break;
case '9' : strcpy (code, "-.-.-.-") ; break;
case '0' : strcpy (code, "--.--") ; break;
case '.' : strcpy (code, "-.-.-.-") ; break;
case '?' : strcpy (code, "-.-.-.-") ; break;
case '(' : strcpy (code, "-.-.-.-") ; break;
case ')' : strcpy (code, "-.-.-.-") ; break;
case ':' : strcpy (code, "-.-.-.-") ; break;
case '=' : strcpy (code, "-.-.-.-") ; break;
case '/' : strcpy (code, "-.-.-.-") ; break;
case '+' : strcpy (code, "-.-.-.-") ; break;
case '-' : strcpy (code, "-.-.-.-") ; break;
case '+' : strcpy (code, "-.-.-.-") ; break;
default: cout<<endl<<"MSG-Switch::Invalid
```

```
character \";
cout<<in<<"\ "<<endl; break;
}
int len=strlen(code) ;
for (int i=0; i<len; i++)
{
sound (FREQ) ;
if(code [i]=='.') delay(DOT) ;
else
if(code[i]== '-') delay(3*DOT) ;
nosound() ; delay (DOT) ;
}
delay (3*DOT) ;
}
int mnu_one()
{
int ch;
clrscr() ;
gotoxy(30,7) ; cout<<"MORSE CODE TUTOR";
gotoxy(30,8) ;
cout<<"-----";
gotoxy(33,9) ; cout<<"1. CHARACTER MODE";
gotoxy(33,10) ; cout<<"2. LINE MODE" ;
gotoxy(33,11) ; cout<<"3. FILE MODE" ;
gotoxy(33,12) ; cout<<"4. SETTINGS" ;
gotoxy(33,13) ; cout<<"5. EXIT" ;
gotoxy(30,15) ; cout<<"Enter your choice : " ;
gotoxy(25,19) ; cout<<"A Freeware From
VU3PRX" ;
gotoxy(50,15) ; cin>>ch ; return ch ;
}
void mode_line()
{
int flag=1; char line[80] ;
clrscr() ; cout<<"LINE MODE"<<endl ;
cout<<"Type '*' to END sending"<<endl ;
while(flag)
{
gets(line) ;
int len=strlen(line) ;
for (int i=0; i<len; i++)
{
in=line[i] ;
if(in==' ') { delay (6*DOT) ; continue ;
}
else if(in=='*') { flag=0; break;
}
}
```

```

morse() ;
}
}
}
void mode_file()
{
FILE *fp ; char fname[10] ;
clrscr() ;
cout<<"Enter file name : " ; cin>> fname;
if ((fp=fopen(fname,"r")) !=NULL)
{
while((in=getc(fp)) !=EOF)
{
if (in==' ')
{
delay(6*DOT); continue;
}
}
morse() ;
}
else
{
char ans ;
cout<<endl<<"File not found, Create New
(Y/N) ? " ;

cin>>ans ;
if((ans=='y') || (ans=='Y'))
if((fp=fopen(fname,"w")) !=NULL)
while((ans=getchar()) !='\n')
putc(ans, fp) ;
}
fclose(fp) ;
}
}

```

```

void mode_sett() ;
void mode_sett()
{
int cho ;
clrscr() ;
cout<<"1. CHANGE FREQUENCY"<<endl ;
cout<<"2. CHANGE SPEED"<<endl ;
cout<<endl<<"Enter your choice : " ;
cin>>cho ;
if(cho==1)
{
cout<<endl<<"Current Frequency of tone is"
<<FREQ<<"Hz"<<endl ;
cout<<"The valid frequency range is from 100
Hz to 4000 Hz" ;
cout<<endl<<endl<<"Enter the new
frequency : " ;

int tmp ;
cin>>tmp ;
cout<<endl<<endl ;
if ( (tmp>99) && (tmp<4001) )
{
FREQ=tmp ;
cout<<"Now the Frequency is changed to"
<<FREQ<<"Hz"<<endl ;
}
else cout<<"Error : Invalid frequency range" ;
getch() ;
}
if(cho==2)
{
int speed ;
speed=36000/(DOT*60) ;

```

```

cout<<"\nCurrent speed is "<<speed<<
" Words/Minute"<<endl ;
cout<<"The valid Speed is from 5 WPM to 20
WPM "<<endl ;
cout<<"\n\nEnter new speed : " ;
cin>>speed ;
if((speed>4) && (speed<21))
{
DOT=36000/(speed*60) ;
cout<<"\n\nNow the speed is "<<speed<<
"Words Per Minute" ;
}
else cout<<"\n\nError : Speed "<<speed<<
"WPM not allowed\n" ;
getch() ;
}
void mode_char()
{
clrscr() ;
cout<<"CHARACTER MODE"<<endl ;
cout<<"Type '*' to END sending"<<endl ;
while (1)
{
in=getche() ;
if(in==' ') { delay (6*DOT) ; continue;
}
else if(in=='*') break ;
morse () ;
}
}
}

```

# CONNECT-FOUR GAME

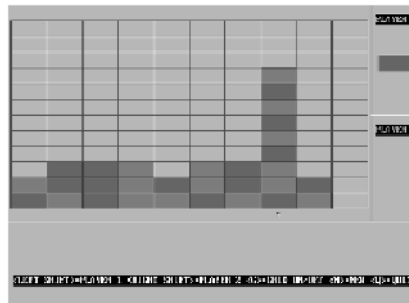
DHAVAL Y. TRIVEDI

**C**onnect-four is a very simple and popular game. Here is the computer version of this game. The game is quite easy and self-explanatory. Online help is provided to avoid any confusion during the game.

This game is executable only in QBASIC and it runs only on a system with VGA monitor or a higher system. A 486 or higher system is recommended, as the game will run very slow on a lower system.

In this game, two different colours are

provided for two players. This game is almost like tick-tack-toe. You have to just



place four pieces of your colour adjacent to each other, as you place in tick-tack-toe. In tick-tack-toe, you can put your piece anywhere. But in connect-four, a piece will be placed at the lowest empty block.

The player who places four pieces of his colour in a row, column, or diagonally will win. The software will automatically detect the winner. All the runtime keys are specified in the online help. Save this file as CONNECT4.BAS. If you want to run this game directly from DOS-prompt, make a file CONNECT4.BAT in your directory as follows:

```

@ECHO OFF
CLS
QBASIC/RUN CONNECT4.BAS
CLS

```

## Program Listing in QBASIC

```

CLEAR , , 8096: CLS : SCREEN 12
10 CLS : SCREEN 12: DIM cur(11): PSET (2, 0): DRAW "g2 e2 f2 h2
d8": DEF SEG = 0: POKE 1047, &H0: DEF SEG
GET (0, 0)-(4, 8), cur: CLS : PAINT (1, 1), 7
LINE (0, 345)-(640, 345), 15: LINE (560, 0)-(560, 345), 15
LINE (572, 80)-(627, 105), 7, BF: LINE (572, 80)-(627, 105), 5, BF:
GOSUB grid:
LINE (560, 175)-(639, 175), 15: LINE (0, 0)-(639, 479), 15, B
X = 30: y = 330: r1 = 325: r2 = 325: r3 = 325: r4 = 325: r5 = 325: r6 =
325: r7 = 325: r8 = 325: r9 = 325:
r10 = 325: turn = 6: gr = 7: PUT (X, y), cur
LOCATE 2, 72: PRINT "PLAYER 1": LOCATE 13, 72: PRINT "PLAYER
2": GOSUB hlp
PLAY "L20": PLAY "O 3CDEFG"
90 KS = INKEYS
KEY 17, CHRS(0) + CHRS(16): ON KEY(17) GOSUB ex: KEY(17) ON: KEY
18, CHRS(0) + CHRS(34): ON KEY(18) GOSUB grid: KEY(18) ON
KEY 15, CHRS(0) + CHRS(42): ON KEY(15) GOSUB p1: KEY(15) ON:
KEY 16, CHRS(0) + CHRS(49): ON KEY(16) GOSUB restart: KEY(16) ON
IF LEN(KS) < 2 THEN 90
K = ASC(RIGHT$(KS, 1))
IF K = 75 THEN x1 = -55: y1 = 330: GOTO 170
IF K = 77 THEN x1 = 55: y1 = 330: GOTO 170
GOTO 90

```

```

170 PUT (X, y), cur: X = X + x1: y = y1
IF X < 30 THEN X = 525 ELSE IF X > 525 THEN X = 30
SOUND 190, .5: PUT (X, y), cur
GOTO 90
p1:
turn = turn - 1
IF X = 30 THEN r1 = r1 - 25: SOUND 990, .5: IF r1 < 25 THEN turn =
6: BEEP: GOSUB play2t: GOTO 90 ELSE LINE
(X - 24, r1 + 24)-(X + 29, r1 + 1), turn, BF
IF X = 85 THEN r2 = r2 - 25: SOUND 990, .5: IF r2 < 25 THEN turn =
6: BEEP: GOSUB play2t: GOTO 90 ELSE
LINE (X - 24, r2 + 24)-(X + 29, r2 + 1), turn, BF
IF X = 140 THEN r3 = r3 - 25: SOUND 990, .5: IF r3 < 25 THEN turn =
6: BEEP: GOSUB play2t: GOTO 90 ELSE LINE
(X - 24, r3 + 24)-(X + 29, r3 + 1), turn, BF
IF X = 195 THEN r4 = r4 - 25: SOUND 990, .5: IF r4 < 25 THEN turn =
6: BEEP: GOSUB play2t: GOTO 90 ELSE LINE
(X - 24, r4 + 24)-(X + 29, r4 + 1), turn, BF
IF X = 250 THEN r5 = r5 - 25: SOUND 990, .5: IF r5 < 25 THEN turn =
6: BEEP: GOSUB play2t: GOTO 90 ELSE LINE
(X - 24, r5 + 24)-(X + 29, r5 + 1), turn, BF
IF X = 305 THEN r6 = r6 - 25: SOUND 990, .5: IF r6 < 25 THEN turn =
6: BEEP: GOSUB play2t: GOTO 90 ELSE LINE
(X - 24, r6 + 24)-(X + 29, r6 + 1), turn, BF
IF X = 360 THEN r7 = r7 - 25: SOUND 990, .5: IF r7 < 25 THEN turn =
6: BEEP: GOSUB play2t: GOTO 90 ELSE LINE
(X - 24, r7 + 24)-(X + 29, r7 + 1), turn, BF
IF X = 415 THEN r8 = r8 - 25: SOUND 990, .5: IF r8 < 25 THEN turn =
6: BEEP: GOSUB play2t: GOTO 90 ELSE LINE
(X - 24, r8 + 24)-(X + 29, r8 + 1), turn, BF
IF X = 470 THEN r9 = r9 - 25: SOUND 990, .5: IF r9 < 25 THEN turn =
6: BEEP: GOSUB play2t: GOTO 90 ELSE LINE
(X - 24, r9 + 24)-(X + 29, r9 + 1), turn, BF
IF X = 525 THEN r10 = r10 - 25: SOUND 990, .5: IF r10 < 25 THEN
turn = 6: BEEP: GOSUB play2t: GOTO 90 ELSE
LINE (X - 24, r10 + 24)-(X + 29, r10 + 1), turn, BF

GOSUB check
GOSUB play1t
320 KS = INKEYS
ON KEY(17) GOSUB ex: ON KEY(18) GOSUB grid
KEY 15, CHR$(0) + CHR$(54): ON KEY(15) GOSUB p2: KEY(15) ON:
ON KEY(16) GOSUB restart: KEY(16) ON
IF LEN(KS) < 2 THEN 320
K = ASC(RIGHT$(KS, 1))
IF K = 75 THEN x1 = -55: y1 = 330: GOTO 400
IF K = 77 THEN x1 = 55: y1 = 330: GOTO 400
GOTO 320
400 PUT (X, y), cur: X = X + x1: y = y1
IF X < 30 THEN X = 525 ELSE IF X > 525 THEN X = 30
SOUND 190, .5: PUT (X, y), cur
GOTO 320
p2:
turn = turn + 1
IF X = 30 THEN r1 = r1 - 25: SOUND 990, .5: IF r1 < 25 THEN turn =
5: BEEP: GOSUB play1t: GOTO 320 ELSE LINE
(X - 24, r1 + 24)-(X + 29, r1 + 1), turn, BF
IF X = 85 THEN r2 = r2 - 25: SOUND 990, .5: IF r2 < 25 THEN turn =
5: BEEP: GOSUB play1t: GOTO 320 ELSE LINE
(X - 24, r2 + 24)-(X + 29, r2 + 1), turn, BF
IF X = 140 THEN r3 = r3 - 25: SOUND 990, .5: IF r3 < 25 THEN turn =
5: BEEP: GOSUB play1t: GOTO 320 ELSE LINE
(X - 24, r3 + 24)-(X + 29, r3 + 1), turn, BF
IF X = 195 THEN r4 = r4 - 25: SOUND 990, .5: IF r4 < 25 THEN turn =
5: BEEP: GOSUB play1t: GOTO 320 ELSE LINE
(X - 24, r4 + 24)-(X + 29, r4 + 1), turn, BF
IF X = 250 THEN r5 = r5 - 25: SOUND 990, .5: IF r5 < 25 THEN turn =
5: BEEP: GOSUB play1t: GOTO 320 ELSE LINE
(X - 24, r5 + 24)-(X + 29, r5 + 1), turn, BF
IF X = 305 THEN r6 = r6 - 25: SOUND 990, .5: IF r6 < 25 THEN turn =
5: BEEP: GOSUB play1t: GOTO 320 ELSE LINE
(X - 24, r6 + 24)-(X + 29, r6 + 1), turn, BF
IF X = 360 THEN r7 = r7 - 25: SOUND 990, .5: IF r7 < 25 THEN turn =
5: BEEP: GOSUB play1t: GOTO 320 ELSE LINE
(X - 24, r7 + 24)-(X + 29, r7 + 1), turn, BF
IF X = 415 THEN r8 = r8 - 25: SOUND 990, .5: IF r8 < 25 THEN turn =
5: BEEP: GOSUB play1t: GOTO 320 ELSE LINE
(X - 24, r8 + 24)-(X + 29, r8 + 1), turn, BF

```

```

IF X = 470 THEN r9 = r9 - 25: SOUND 990, .5: IF r9 < 25 THEN turn =
5: BEEP: GOSUB play1t: GOTO 320 ELSE LINE
(X - 24, r9 + 24)-(X + 29, r9 + 1), turn, BF
IF X = 525 THEN r10 = r10 - 25: SOUND 990, .5: IF r10 < 25 THEN
turn = 5: BEEP: GOSUB play1t: GOTO 320 ELSE
LINE (X - 24, r10 + 24)-(X + 29, r10 + 1), turn, BF

GOSUB check
GOSUB play2t
turn = 6
GOTO 90
END
restart:
GOTO 10
check:
FOR U = 35 TO 325 STEP 25
FOR D = 30 TO 525 STEP 55
IF POINT(D, U) = turn AND POINT(D + 55, U) = turn AND POINT(D
+ 110, U) = turn AND POINT(D + 165, U) = turn THEN GOSUB win
IF POINT(D, U) = turn AND POINT(D, U + 25) = turn AND POINT(D,
U + 50) = turn AND POINT(D, U + 75) = turn THEN GOSUB win
IF POINT(D, U) = turn AND POINT(D + 55, U + 25) = turn AND
POINT(D + 110, U + 50) = turn AND POINT(D
+ 165, U + 75) = turn THEN GOSUB win
IF POINT(D, U) = turn AND POINT(D + 55, U - 25) = turn AND
POINT(D + 110, U - 50) = turn AND POINT(D
+ 165, U - 75) = turn THEN GOSUB win

NEXT
NEXT
RETURN
win:
PLAY "L20": PLAY "O 3GFEDCP8CDEFPG8GFEDCP8CDEFG":
PLAY "L30"
CLS : SCREEN 12: PAINT (1, 1), 7: KEY OFF: FOR I = 1 TO 520 STEP
7: LINE (1, 1)-(I, 479), 5: LINE (640, 1)-(640 - I, 1), 5: LINE (1,
479)-(I, 1), 5: LINE (640, 479)-(640 - I, 1), 5: LOCATE 15, 36:
COLOR 7: PRINT "PLAYER "; turn - 4: LOCATE 17, 39: PRINT
"WINS": NEXT
SLEEP 1
GOTO 10
play1t:
LINE (572, 80)-(627, 105), 7, BF: LINE (572, 250)-(627, 275), 6, BF
RETURN
play2t:
LINE (572, 250)-(627, 275), 7, BF: LINE (572, 80)-(627, 105), 5, BF
RETURN
ex:
CLS : SCREEN 0
LOCATE 11, 35: COLOR 30: PRINT "CONNECT FOUR"
LOCATE 12, 29: COLOR 7: PRINT "BY:- DHAVALY.TRIVEDI,"
LOCATE 13, 30: PRINT "KANCHAN JUNGHA APPT, 101,"
LOCATE 14, 26: PRINT "ROYAL PARK-6, UNLROAD, RAJKOT-5."
PLAY "L20": PLAY "O 3GFEDC"
SYSTEM
grid:
IF gr = 7 THEN gr = 1 ELSE IF gr = 1 THEN gr = 7
LINE (5, 325)-(555, 300), gr, B: LINE (5, 300)-(555, 275), gr, B: LINE
(5, 275)-(555, 250), gr, B
LINE (5, 250)-(555, 225), gr, B: LINE (5, 225)-(555, 200), gr, B: LINE
(5, 200)-(555, 175), gr, B
LINE (5, 175)-(555, 150), gr, B: LINE (5, 150)-(555, 125), gr, B: LINE
(5, 125)-(555, 100), gr, B
LINE (5, 100)-(555, 75), gr, B: LINE (5, 75)-(555, 50), gr, B: LINE (5,
50)-(555, 25), gr, B
LINE (5, 325)-(60, 25), gr, B: LINE (60, 325)-(115, 25), gr, B: LINE
(115, 325)-(170, 25), gr, B
LINE (170, 325)-(225, 25), gr, B: LINE (225, 325)-(280, 25), gr, B: LINE
(280, 325)-(335, 25), gr, B
LINE (335, 325)-(390, 25), gr, B: LINE (390, 325)-(445, 25), gr, B: LINE
(445, 325)-(500, 25), gr, B
LINE (500, 325)-(555, 25), gr, B
RETURN
hlp:
LOCATE 28, 2: COLOR 15: PRINT "<LEFT SHIFT>=PLAYER 1
<RIGHT SHIFT>=PLAYER 2 <G>=GRID
ON/OFF <N>=NEW <Q>=QUIT"
RETURN

```