# Playfair Cipher: Theoretical Overview

The Playfair cipher, invented by Charles Wheatstone in 1854 and promoted by Lord Playfair, is a digraph substitution cipher. It encrypts pairs of letters, offering significant advantages over simple monoalphabetic substitution ciphers.

## Features

- Uses a 5x5 grid of letters based on a keyword
- Encrypts digraphs (pairs of letters) instead of single letters
- 25 letters (I and J are typically combined)
- Resistant to frequency analysis attacks

## Key Generation

1. Choose a keyword and remove duplicate letters
2. Fill a 5x5 matrix with the keyword letters first
3. Complete the matrix with remaining alphabet letters
4. Example with keyword "MONARCHY":

$$\begin{array}{ccccc} M & O & N & A & R \\ C & H & Y & B & D \\ E & F & G & I/J & K \\ L & P & Q & S & T \\ U & V & W & X & Z \end{array}$$

## Encryption Rules

For each pair of plaintext letters $(a, b)$:

1. If $a$ and $b$ are in the same row, replace with letters to their right (wrapping around)
2. If $a$ and $b$ are in the same column, replace with letters below (wrapping around)
3. Otherwise, replace with letters on the same row but in the column of the other letter

## Security Considerations

- Stronger than simple substitution ciphers
- Vulnerable to known-plaintext attacks
- Frequency analysis of digraphs can be used for cryptanalysis
- Modern cryptography has rendered it obsolete for secure communication

Listing 1: Playfair Cipher Implementation

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define SIZE 5

void prepare_key(char *key, char matrix[SIZE][SIZE]) {
    char alphabet[26] = "ABCDEFGHIKLMNOPQRSTUVWXYZ";
    int i, j, k, flag = 0;

    for (i = 0; i < strlen(key); i++) {
        if (key[i] == 'J') key[i] = 'I';
        if (!flag) {
            for (j = 0; j < i; j++) {
                if (key[i] == key[j]) {
                    flag = 1;
                    break;
                }
            }
            if (!flag) {
                for (k = 0; k < 25; k++) {
                    if (key[i] == alphabet[k]) {
                        alphabet[k] = '␣';
                        break;
                    }
                }
            }
        }
        flag = 0;
    }

    k = 0;
    for (i = 0; i < SIZE; i++) {
        for (j = 0; j < SIZE; j++) {
            if (k < strlen(key) && key[k] != '␣')
                matrix[i][j] = key[k++];
            else {
                while (alphabet[k - strlen(key)] == '␣')
                    k++;
                matrix[i][j] = alphabet[k - strlen(key)];
                k++;
            }
        }
    }
}

void find_position(char matrix[SIZE][SIZE], char ch, int *row, int *col) {
    int i, j;
    for (i = 0; i < SIZE; i++) {
        for (j = 0; j < SIZE; j++) {
            if (matrix[i][j] == ch) {
                *row = i;
                *col = j;
                return;
            }
        }
    }
```

```
58 }
59
60 void playfair_encrypt(char *plaintext, char matrix[SIZE][SIZE], char *
       ciphertext) {
61     int i, r1, c1, r2, c2;
62     for (i = 0; i < strlen(plaintext); i += 2) {
63         find_position(matrix, plaintext[i], &r1, &c1);
64         find_position(matrix, plaintext[i+1], &r2, &c2);
65
66         if (r1 == r2) {
67             ciphertext[i] = matrix[r1][(c1+1)%SIZE];
68             ciphertext[i+1] = matrix[r2][(c2+1)%SIZE];
69         }
70         else if (c1 == c2) {
71             ciphertext[i] = matrix[(r1+1)%SIZE][c1];
72             ciphertext[i+1] = matrix[(r2+1)%SIZE][c2];
73         }
74         else {
75             ciphertext[i] = matrix[r1][c2];
76             ciphertext[i+1] = matrix[r2][c1];
77         }
78     }
79     ciphertext[strlen(plaintext)] = '\0';
80 }
81
82 void playfair_decrypt(char *ciphertext, char matrix[SIZE][SIZE], char *
       plaintext) {
83     int i, r1, c1, r2, c2;
84     for (i = 0; i < strlen(ciphertext); i += 2) {
85         find_position(matrix, ciphertext[i], &r1, &c1);
86         find_position(matrix, ciphertext[i+1], &r2, &c2);
87
88         if (r1 == r2) {
89             plaintext[i] = matrix[r1][(c1-1+SIZE)%SIZE];
90             plaintext[i+1] = matrix[r2][(c2-1+SIZE)%SIZE];
91         }
92         else if (c1 == c2) {
93             plaintext[i] = matrix[(r1-1+SIZE)%SIZE][c1];
94             plaintext[i+1] = matrix[(r2-1+SIZE)%SIZE][c2];
95         }
96         else {
97             plaintext[i] = matrix[r1][c2];
98             plaintext[i+1] = matrix[r2][c1];
99         }
100     }
101     plaintext[strlen(ciphertext)] = '\0';
102 }
103
104 void display_matrix(char matrix[SIZE][SIZE]) {
105     printf("Playfair␣Cipher␣Matrix:\n");
106     for (int i = 0; i < SIZE; i++) {
107         for (int j = 0; j < SIZE; j++) {
108             printf("%c␣", matrix[i][j]);
109         }
110         printf("\n");
111     }
112     printf("\n");
113 }
```

```
114
115 int main() {
116     char key[] = "KEYWORD";
117     char matrix[SIZE][SIZE];
118     char plaintext[] = "HELLOWORLD";
119     char ciphertext[100], decrypted[100];
120
121     prepare_key(key, matrix);
122
123     playfair_encrypt(plaintext, matrix, ciphertext);
124     playfair_decrypt(ciphertext, matrix, decrypted);
125
126     printf("Key: %s\n", key);
127     display_matrix(matrix);
128     printf("Plaintext: %s\n", plaintext);
129     printf("Ciphertext: %s\n", ciphertext);
130     printf("Decrypted: %s\n", decrypted);
131
132     return 0;
133 }
```