S-DES 加解密系统开发手册（点击可跳转至 github）

1. 概述

本系统采用 c/s 架构来实现 S-DES 加解密算法。它支持普通的加解密，ASCII 码的加解密，以及暴力破解功能。

2. 前端页面设计

2.1 主要结构

- **Logo 容器**: 显示系统相关的 logo。
- **标题**: 显示系统的主标题。
- **选项列表**: 用户可以选择二进制、ASCII 或暴力破解。
- **输入部分**: 允许用户输入信息和密钥。
- **信息容器**: 显示开发单位和开发者信息。

2.2 样式和交互

- **背景**: 页面背景采用图片，能够自动调整大小以适应不同的屏幕。
- **提示**: 提供清晰的界面和按钮提示用户进行输入。
- **交互**: 选项列表允许用户选择操作类型。选择暴力破解后，系统将跳转到另一页面。
- **界面 1 的设计**: 下面是本系统的初始界面的 html 文件。选择暴力破解后，系统将跳转到另一页面。

```
<!DOCTYPE html><html><head>

    <style>

            body {

          background-image: url("/static/lan_F.jpg");

          background-size: cover;

          color: white;

          height: 50vh;

          display: flex;

          justify-content: center;

          align-items: center;

          flex-direction: column;
```

```
            margin: 0;
            font-size: 3em;
            font-family: Arial, sans-serif;

        }

        .logo-container {
            position: absolute;
            top: 10px;
            left: 10px;
            display: flex;
            align-items: center;
        }
        .logo-container img {
            height: 50px;
            margin-right: 10px;
        }
        .title {

            font-size: 1em;
            margin-bottom:50px;
            margin-top:200px;
        }
.select-container {
    margin: 20px 0; /* 上下边距为 20px，左右边距为 0 */
    font-size: 40px; /* 字体大小为 16px */
    display: flex; /* 使用 flex 布局以使 label 和 select 在同一行 */
    align-items: center; /* 垂直居中对齐内容 */
}
```

```
.select-container label {
    margin-right: 10px; /* 在 label 和 select 之间增加 10px 的间距 */
}

.select-container select {
    padding: 5px 10px; /* 选择框内部的填充：上下 5px，左右 10px */
    border: 1px solid #cccccc; /* 给选择框一个灰色的边框 */
    border-radius: 4px; /* 边框圆角为 4px */
}
```

```
        .profile-container img {
            width: 100%;
            height: auto;
        }
        /* New CSS for login form */
                /* 应该是灰框*/
        .login-container {
            display: flex;
            flex-direction: column;
            background: rgba(0, 0, 0, 0.5); /* semi-transparent background */
            padding: 10px;  /* reduce padding */
            border-radius: 5px;
            font-size: 0.5em;

              padding: 20px;  /* 增加内边距 */

              width: 600px;  /* 增加宽度 */
```

```
height: auto;  /* 根据内容自适应高度 */




    }
     /* 这个是输入字体控制 */
    .login-container input {
        margin-bottom: 5px; /* reduce margin */
        padding: 5px; /* reduce padding */
        font-size: 0.8em; /* reduce font-size */
        background: transparent; /* make input box transparent */
        color: white; /* change text color to white */
        border: none; /* remove input box border */
    }
     /* 这个是按钮控制 */
    .login-container button {
        padding: 5px;
        background-color: lightblue;
        color: black;
        border: none;
        cursor: pointer;
        font-size: 0.5em; /* reduce font-size */
         margin-bottom:30px;



    }
    /* New CSS for website information */
             /* 这个是底部字体 */
    .info-container {
        position: absolute;
        bottom: 0;
        width: 100%;
```

```
        text-align: center;
        padding: 10px 0;
        background-color: rgba(0, 0, 0, 0.5); /* semi-transparent background */
        font-size: 0.2em; /* adjust font size */
    }
</style>
<script>
    window.onload = function() {
document.getElementById('login-button').onclick = function() {
    var username = document.getElementById('username-input').value;
    var password = document.getElementById('password-input').value;
    var option = document.getElementById('option-list').value; // 获取当前选择的选项

    fetch('/login', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({username: username, password: password, operation: "encryption", option: option})
    }).then(response => response.json())
      .then(data => alert(data.message));
}
// 添加事件监听器检测选择的变化
        document.getElementById('option-list').addEventListener('change', function() {
            if (this.value === "brute_force") {
                window.location.href = "/brute_force_page"; // 假设新的 HTML 页面的路由为"/brute_force_page"
            }
        });
document.getElementById('login-button2').onclick = function() {
    var username = document.getElementById('username-input').value;
```

```
    var password = document.getElementById('password-input').value;
    var option = document.getElementById('option-list').value; // 获取当前选择的选项

    fetch('/login', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({username: username, password: password, operation: "decryption", option: option})
    }).then(response => response.json())
      .then(data => alert(data.message));
}}

</script></head><body>
<div class="logo-container">
    <img src="/static/logo1.png" alt="Logo 1">

    <img src="/static/logo3.png" alt="Logo 3">
</div>
<div class="title">
    S-DES 加解密系统
</div>

   <!-- 选项列表部分 -->
<div class="select-container">
    <label for="option-list">选项：</label>
    <select id="option-list">
        <option value="binary">二进制</option>
```

```
            <option value="asciil">ASCLL</option>
             <option value="brute_force">暴力破解</option>
        </select>
    </div>


          <!-- 文本部分 -->
    <div class="login-container">
        <input id="username-input" type="text" placeholder="信息" required/><!--       <input id="password-input"
type="password" placeholder="密钥" required/>-->
        <input id="password-input" type="text" placeholder="密钥(10bits)" required/>
        <button id="login-button">加密</button>
        <button id="login-button2">解密</button>


    </div>
    <!-- New website information -->
    <div class="info-container">
        <p>所属单位：重庆大学大数据与软件学院  |   分工小组：RNG</p>
        <p>开发人员：吴科明 李泽坤   |  联系方式：1281673219@qq.com</p>
    </div></body></html>
```

- **界面 2 的设计**: 选择暴力破解后，系统将跳转到另一页面。下面是该页面的 html 文件。

```
<!DOCTYPE html><html><head>
    <style>
            body {
          background-image: url("/static/lan_F.jpg");
          background-size: cover;
          color: white;
          height: 50vh;
```

```
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
            margin: 0;
            font-size: 3em;
            font-family: Arial, sans-serif;

        }

        .logo-container {
            position: absolute;
            top: 10px;
            left: 10px;
            display: flex;
            align-items: center;
        }
        .logo-container img {
            height: 50px;
            margin-right: 10px;
        }
        .title {

            font-size: 1em;
            margin-bottom:50px;
            margin-top:200px;
        }
.select-container {
    margin: 20px 0; /* 上下边距为 20px，左右边距为 0 */
    font-size: 40px; /* 字体大小为 16px */
```

```
    display: flex; /* 使用 flex 布局以使 label 和 select 在同一行 */
    align-items: center; /* 垂直居中对齐内容 */
}

.select-container label {
    margin-right: 10px; /* 在 label 和 select 之间增加 10px 的间距 */
}

.select-container select {
    padding: 5px 10px; /* 选择框内部的填充：上下 5px，左右 10px */
    border: 1px solid #cccccc; /* 给选择框一个灰色的边框 */
    border-radius: 4px; /* 边框圆角为 4px */
}
```

```
        .profile-container img {
            width: 100%;
            height: auto;
        }
        /* New CSS for login form */
                /* 应该是灰框*/
        .login-container {
            display: flex;
            flex-direction: column;
            background: rgba(0, 0, 0, 0.5); /* semi-transparent background */
            padding: 10px;  /* reduce padding */
            border-radius: 5px;
            font-size: 0.5em;
```

```
        padding: 20px;  /* 增加内边距 */


        width: 600px;  /* 增加宽度 */
height: auto;  /* 根据内容自适应高度 */



    }
     /* 这个是输入字体控制 */
    .login-container input {
        margin-bottom: 5px; /* reduce margin */
        padding: 5px; /* reduce padding */
        font-size: 0.8em; /* reduce font-size */
        background: transparent; /* make input box transparent */
        color: white; /* change text color to white */
        border: none; /* remove input box border */
    }
     /* 这个是按钮控制 */
    .login-container button {
        padding: 5px;
        background-color: lightblue;
        color: black;
        border: none;
        cursor: pointer;
        font-size: 0.5em; /* reduce font-size */
         margin-bottom:30px;


    }
    /* New CSS for website information */
          /* 这个是底部字体 */
```

```
    .info-container {
        position: absolute;
        bottom: 0;
        width: 100%;
        text-align: center;
        padding: 10px 0;
        background-color: rgba(0, 0, 0, 0.5); /* semi-transparent background */
        font-size: 0.2em; /* adjust font size */
    }
</style>
<script>

    window.onload = function() {



document.getElementById('login-button').onclick = function() {
    var username = document.getElementById('username-input').value;
    var password = document.getElementById('password-input').value;

    fetch('/brute_force', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({username: username, password: password})
    }).then(response => response.json())
      .then(data => alert(data.message));
}}
```

```
</script></head><body>
<div class="logo-container">
    <img src="/static/logo1.png" alt="Logo 1">

    <img src="/static/logo3.png" alt="Logo 3">
</div>
<div class="title">
    S-DES 加解密暴力破解系统
</div>

      <!-- 文本部分 -->
<div class="login-container">
    <input id="username-input" type="text" placeholder="明文" required/><!--       <input id="password-input" type="password" placeholder="密钥" required/>-->
    <input id="password-input" type="text" placeholder="密文" required/>
    <button id="login-button">开始破解</button>

</div>
<!-- New website information -->
<div class="info-container">
    <p>所属单位：重庆大学大数据与软件学院  |  分工小组：RNG</p>
    <p>开发人员：吴科明 李泽坤  | 联系方式：1281673219@qq.com</p>
</div></body></html>
```

3. 加解密算法

3.1 辅助函数

-

`permute(input_str, permutation_table)`:

-
-

根据给定的置换表来进行置换。

-

```
def permute(input_str, permutation_table):
    output_str = ''
    for bit_position in permutation_table:
        output_str += input_str[bit_position - 1]
    return output_str
```

- `left_shift(key, n)`: 将输入字符串的左半部分和右半部分分别左移 `n` 位。

```
def left_shift(key, n):
    left_half = key[:5]
    right_half = key[5:]
    shifted_left = left_half[n:] + left_half[:n]
    shifted_right = right_half[n:] + right_half[:n]
    return shifted_left + shifted_right
```

- `generate_subkeys(key, p10_table, p8_table)`: 根据给定的 P10 和 P8 表生成子密钥。

```
def generate_subkeys(key, p10_table, p8_table):
    p10_key = permute(key, p10_table)
    key1 = permute(left_shift(p10_key, 1), p8_table)
    key2 = permute(left_shift(left_shift(p10_key, 1), 1), p8_table)
    return key1, key2
```

- `f_function(right_half, subkey, sbox0, sbox1, p4_table)`: 进行 S-DES 的 F 函数操作。

```
def f_function(right_half, subkey, sbox0, sbox1, p4_table):
    # Expansion and XOR
    expanded = permute(right_half, EXPANSION_PERMUTATION)
    xored = int(expanded, 2) ^ int(subkey, 2)
    xored_str = format(xored, '08b')

    # S-box substitutions
    s0_input = xored_str[:4]
    s1_input = xored_str[4:]
    s0_row = int(s0_input[0] + s0_input[3], 2)
    s0_col = int(s0_input[1:3], 2)
    s1_row = int(s1_input[0] + s1_input[3], 2)
    s1_col = int(s1_input[1:3], 2)
    s0_output = format(sbox0[s0_row][s0_col], '02b')
    s1_output = format(sbox1[s1_row][s1_col], '02b')
    s_output = s0_output + s1_output

    # Permutation
    return permute(s_output, p4_table)
```

- `ascii_to_binary(ascii_string)`：将 ASCII 字符串转换为二进制。

```
def ascii_to_binary(ascii_string):
    binary_string = ""
    for character in ascii_string:
        binary_string += bin(ord(character))[2:].zfill(8)
    return binary_string
```

- `binary_to_ascii(binary_string)`：将二进制转换为 ASCII 字符串。

```
def binary_to_ascii(binary_string):
    ascii_string = ""
    for i in range(0, len(binary_string), 8):
```

```
        ascii_string += chr(int(binary_string[i:i + 8], 2))
    return ascii_string
```

3.2 加密

`encrypt(plaintext, key)`: 使用给定的密钥对明文进行 S-DES 加密。

```
def encrypt(plaintext, key):
    key1, key2 = generate_subkeys(key, PERMUTATION_P10, PERMUTATION_P8)
    plaintext = permute(plaintext, INITIAL_PERMUTATION)
    left_half = plaintext[:4]
    right_half = plaintext[4:]
    left_previous = right_half
    f_result = f_function(right_half, key1, SBOX0, SBOX1, PERMUTATION_P4)
    right_half1_int = int(left_half, 2) ^ int(f_result, 2)
    right_half1 = format(right_half1_int, '04b')
    f_result = f_function(right_half1, key2, SBOX0, SBOX1, PERMUTATION_P4)
    right_half2_int = int(left_previous, 2) ^ int(f_result, 2)
    right_half2 = format(right_half2_int, '04b')
    return permute(right_half2 + right_half1, INVERSE_INITIAL_PERMUTATION)
```

3.3 解密

`decrypt(ciphertext, key)`: 使用给定的密钥对密文进行 S-DES 解密。

```
def decrypt(ciphertext, key):
    key1, key2 = generate_subkeys(key, PERMUTATION_P10, PERMUTATION_P8)
    ciphertext = permute(ciphertext, INITIAL_PERMUTATION)
    right_previous = ciphertext[:4]
    left_previous = ciphertext[4:]
    f_result = f_function(left_previous, key2, SBOX0, SBOX1, PERMUTATION_P4)
    left_half1_int = int(right_previous, 2) ^ int(f_result, 2)
    left_half1 = format(left_half1_int, '04b')
    f_result = f_function(left_half1, key1, SBOX0, SBOX1, PERMUTATION_P4)
```

```
    right_half1_int = int(left_previous, 2) ^ int(f_result, 2)

    right_half1 = format(right_half1_int, '04b')

    return permute(right_half1 + left_half1, INVERSE_INITIAL_PERMUTATION)
```

3.4 字符串加解密

`encrypt_string(ascii_string, key)`：对 ASCII 字符串进行加密。

```
def encrypt_string(ascii_string, key):

    binary_string = ascii_to_binary(ascii_string)

    encrypted_string = ""

    for i in range(0, len(binary_string), 8):

        plaintext = binary_string[i:i + 8]

        ciphertext = encrypt(plaintext, key)

        encrypted_string += ciphertext

    return encrypted_string
```

`decrypt_string(encrypted_string, key)`：对已加密的字符串进行解密。

```
def decrypt_string(encrypted_string, key):

    decrypted_string = ""

    for i in range(0, len(encrypted_string), 8):

        ciphertext = encrypted_string[i:i + 8]

        plaintext = decrypt(ciphertext, key)

        decrypted_string += plaintext

    return decrypted_string
```

4. 暴力破解

`BruteForceDecrypt` 类：提供了单线程和多线程的暴力破解方法。

- `_brute_force(self, plaintext, ciphertext, start, end)`：单线程暴力破解方法，从 start 到 end 范围内尝试所有的密钥。

```
class BruteForceDecrypt:



    def __init__(self):
```

```
    self.correct_keys = []
    self.lock = threading.Lock()

def _brute_force(self, plaintext, ciphertext, start, end):
    for key in range(start, end):
        key_str = '{0:010b}'.format(key)
        decrypted = decrypt(ciphertext, key_str)
        if decrypted == plaintext and len(key_str) == 10:
            with self.lock:
                self.correct_keys.append(key_str)

def single_thread_brute_force(self, plaintext, ciphertext):
    self._brute_force(plaintext, ciphertext, 0, 2 ** 10)
    return self.correct_keys

def multi_thread_brute_force(self, plaintext, ciphertext):
    threads = []
    for i in range(8):
        start = i * (2 ** 9)
        end = (i + 1) * (2 ** 9)
        thread = threading.Thread(target=self._brute_force, args=(plaintext, ciphertext, start, end))
        thread.start()
        threads.append(thread)
    for thread in threads:
        thread.join()
    return self.correct_keys

def decrypt(self, plaintext, ciphertext):
    start_time = time.time()
    single_result = self.single_thread_brute_force(plaintext, ciphertext)
```

```
        single_time = time.time() - start_time
        self.correct_keys.clear()
        start_time = time.time()
        multi_result = self.multi_thread_brute_force(plaintext, ciphertext)
        multi_time = time.time() - start_time
        return {
            "single_thread": {
                "keys": single_result,
                "time": single_time
            },
            "multi_thread": {
                "keys": multi_result,
                "time": multi_time
            }
        }
```

- `multi_thread_brute_force(self, plaintext, ciphertext)`：多线程暴力破解接口，使用 8 个线程进行破解。

```
def multi_thread_brute_force(plaintext, ciphertext):
    threads = []
    for i in range(8):
        start = i * (2 ** 9)
        end = (i + 1) * (2 ** 9)
        thread = threading.Thread(target=brute_force, args=(plaintext, ciphertext, start, end))
        thread.start()
        threads.append(thread)
    # 等待所有线程完成
    for thread in threads:
        thread.join()
```

5. 使用指南

5.1 S-DES 算法介绍

S-DES（Simplified Data Encryption Standard）是一个供教学而非安全使用的加密算法。它与 DES 的特性和结构类似，但参数小，明文分组为 8 位，主密钥分组为 10 位，采用两轮迭代。

S-DES 加密过程包含两个重要部分：子密码生成过程和 f 函数结构。

子密码生成过程：

- 对初始密钥进行 P10 置换，将置换后的结果分为左右两部分，各 5 位。
- 对左右两部分进行循环左移操作和 P8 置换，得到 K1。
- 再次对上一步结果进行循环左移操作和 P8 置换，得到 K2。

f 函数结构：

- 对右半部分进行 E/P 扩展置换，将其扩展为 8 位。
- 对扩展后的结果与轮密钥进行异或运算，再将异或的结果拆分成 2 个 4 位的块。
- 将这 2 个块分别通过 S 盒代替（S0 和 S1），然后再进行 P4 置换，最后将 P4 置换后的结果与左半部分进行异或，得到 F 函数输出的结果。

5.2 使用步骤

1. 打开 S-DES 加解密系统主页。
2. 选择所需的操作类型：二进制、ASCII 或暴力破解。
3. 输入信息和 10 位的密钥。如果输入不合法会报错。
4. 点击“加密”或“解密”按钮以进行相应操作。
5. 若选择了暴力破解，则系统会尝试所有可能的密钥，直到找到正确的密钥为止。这个过程可能需要一些时间，具体取决于你的计算机性能和你选择的明文和密文对的数量，并在完成后查看找到的正确密钥。
6. 如果你选择了 ASCII 模式，那么你输入的信息应该是一个 ASCII 编码的字符串，而输出也将是一个 ASCII 编码的字符串。请注意，由于 S-DES 算法会对数据进行各种复杂的变换，所以加密后得到的字符串可能包含一些无法打印或无法显示的字符（即所谓的"乱码"）。但这并不影响解密过程。只要知道正确的密钥，就可以使用 S-DES 算法将这个"乱码"字符串解密回原始字符串。

6. 联系信息

若有任何疑问或建议，请联系开发团队：1281673219@qq.com

7. 其他

我们的系统支持浏览器**在线使用**，如果你有需求，请你联系开发团队开启服务器。