Lab 2: One Time Pad (OTP) Cipher:

Introduction

The One-Time Pad (OTP) cipher is one of the simplest and most secure cryptographic encryption techniques. It involves encrypting a plaintext message with a key that is as long as the message itself, ensuring perfect secrecy when implemented correctly. The key is randomly generated and used only once, making it theoretically unbreakable.

How OTP Works

The OTP encryption process follows these steps:

1. **Generate a Key**: A random sequence of characters, equal in length to the message, is created.
2. **Encrypt the Message**: Each character in the plaintext is combined with the corresponding character in the key using modular arithmetic.
3. **Transmit the Ciphertext**: The encrypted message is sent to the recipient.
4. **Decrypt the Message**: The recipient, who has the same key, performs the inverse operation to retrieve the original plaintext.

Encryption and Decryption

The encryption formula is:

$Ci = Pi \oplus Ki$

Where:

- Ci = is the encrypted character,
- Pi = is the plaintext character,
- Ki = is the key character,

Advantages of OTP

- **Perfect Secrecy**: Since each key is random and used only once, breaking the cipher is impossible.
- **Resistance to Frequency Analysis**: Traditional cryptanalysis methods do not work because the ciphertext appears random.

Disadvantages of OTP

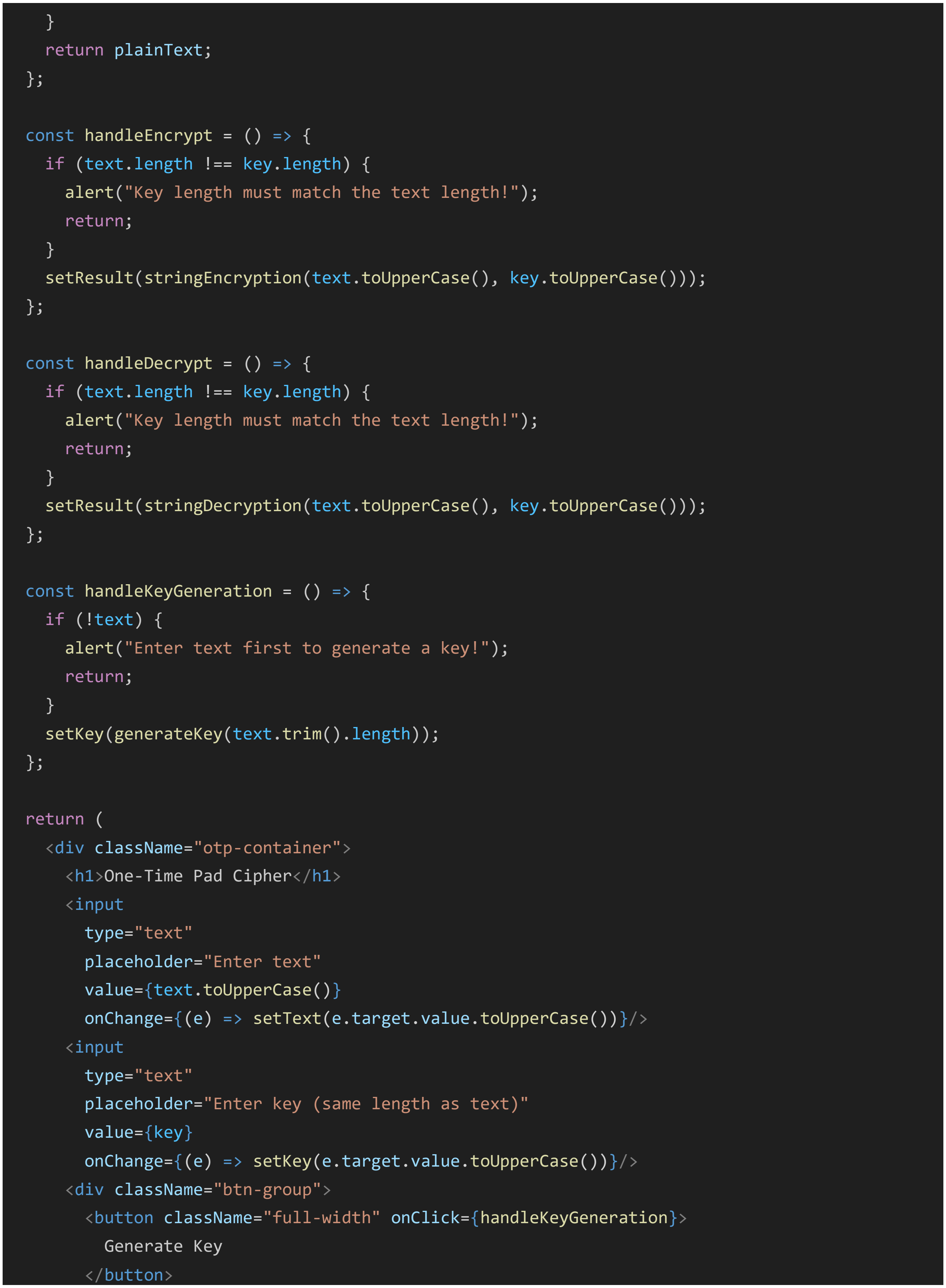
- **Key Management**: Securely sharing and storing a key of the same length as the message is impractical.
- **One-Time Use**: Reusing a key compromises security, making the system vulnerable to attacks.

Implementation in React

Below is a React component that implements the OTP cipher. It provides encryption and decryption functionalities using JavaScript's character encoding methods.

Code:

```
import { useState } from "react";
import "./OTPCipher.css";

const OTPCipher = () => {
  const [text, setText] = useState("");
  const [key, setKey] = useState("");
  const [result, setResult] = useState("");

  const generateKey = (length) => {
    let result = "";
    const characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    for (let i = 0; i < length; i++) {
      result += characters.charAt(Math.floor(Math.random() * characters.length));
    }
    return result;
  };

  const stringEncryption = (text, key) => {
    let cipherText = "";
    let cipher = [];

    for (let i = 0; i < text.length; i++) {
      cipher[i] = text.charCodeAt(i) - "A".charCodeAt(0) + key.charCodeAt(i) -
"A".charCodeAt(0);
      if (cipher[i] > 25) cipher[i] -= 26;
      cipherText += String.fromCharCode(cipher[i] + "A".charCodeAt(0));
    }
    return cipherText;
  };

  const stringDecryption = (cipher, key) => {
    let plainText = "";
    let plain = [];

    for (let i = 0; i < cipher.length; i++) {
      plain[i] = cipher.charCodeAt(i) - "A".charCodeAt(0) - (key.charCodeAt(i) -
"A".charCodeAt(0));
      if (plain[i] < 0) plain[i] += 26;
      plainText += String.fromCharCode(plain[i] + "A".charCodeAt(0));
```

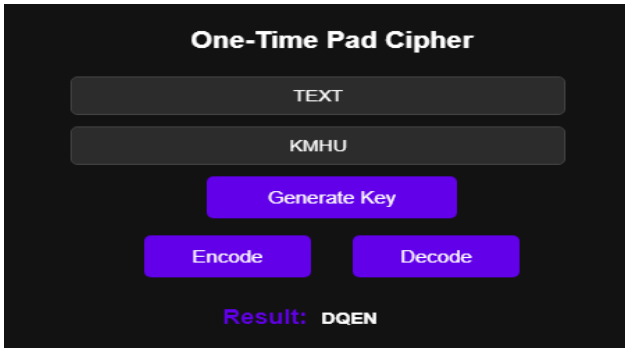


```
  }
  return plainText;
};

const handleEncrypt = () => {
  if (text.length !== key.length) {
    alert("Key length must match the text length!");
    return;
  }
  setResult(stringEncryption(text.toUpperCase(), key.toUpperCase()));
};

const handleDecrypt = () => {
  if (text.length !== key.length) {
    alert("Key length must match the text length!");
    return;
  }
  setResult(stringDecryption(text.toUpperCase(), key.toUpperCase()));
};

const handleKeyGeneration = () => {
  if (!text) {
    alert("Enter text first to generate a key!");
    return;
  }
  setKey(generateKey(text.trim().length));
};

return (
  <div className="otp-container">
    <h1>One-Time Pad Cipher</h1>
    <input
      type="text"
      placeholder="Enter text"
      value={text.toUpperCase()}
      onChange={(e) => setText(e.target.value.toUpperCase())}/>
    <input
      type="text"
      placeholder="Enter key (same length as text)"
      value={key}
      onChange={(e) => setKey(e.target.value.toUpperCase())}/>
    <div className="btn-group">
      <button className="full-width" onClick={handleKeyGeneration}>
        Generate Key
      </button>
```

```
        <div className="row">
          <button  onClick={handleEncrypt}>
            Encode
          </button>
          <button  onClick={handleDecrypt}>
            Decode
          </button>
        </div>
      </div>
      {result && <div className="result-box"><span> Result: </span>{result}</div>}
    </div>
  );
};
export default OTPCipher;
```

Output:

Q: Generate CT using OTP of some PT? (Encode)



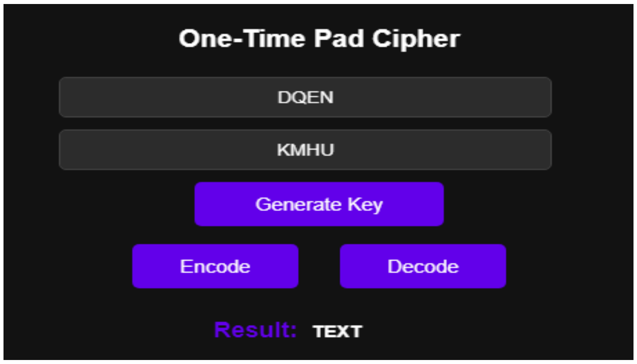
One-Time Pad Cipher

TEXT

KMHU

Result: DQEN

Q: Generate PT using CT and Key? (Decode)



One-Time Pad Cipher

DQEN

KMHU

Result: TEXT

Conclusion

The One-Time Pad is a theoretically secure encryption method but is impractical for widespread use due to key distribution challenges. However, it remains a fundamental concept in cryptography, helping researchers and developers understand the importance of randomness and secure key management in encryption systems.