```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>RSA Encryption and Decryption</title>
    <style>
        /* Basic reset for margin and padding */
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Roboto', sans-serif;
            background-color: #121212;
            color: #e0e0e0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            flex-direction: column;
            font-size: 18px; /* Increased base font size */
        }

        .container {
            background-color: #1a1a1a;
            border-radius: 12px;
            box-shadow: 0 8px 16px rgba(0, 0, 0, 0.3);
```

```
    padding: 40px;
    width: 100%;
    max-width: 1200px; /* Increased the container width */
    display: flex;
    flex-direction: row; /* Arrange elements horizontally */
    justify-content: space-between;
    align-items: flex-start;
    gap: 30px; /* Increased gap between left and right sections */
}

.left-side {
    flex: 1;
}

.right-side {
    flex: 1;
    display: flex;
    flex-direction: column; /* Align the encrypted and decrypted data vertically */
    justify-content: flex-start;
    align-items: flex-start;
}

h1 {
    font-size: 2.5em; /* Increased font size */
    text-align: center;
    color: #ff00ff;
    margin-bottom: 30px; /* Increased bottom margin */
    text-shadow: 0 0 8px #ff00ff, 0 0 16px #ff00ff; /* Adjusted glow effect */
    animation: neon-glow 2s ease-in-out infinite alternate;
}
```

```
label {
    font-weight: bold;
    display: block;
    margin-bottom: 8px; /* Increased margin */
}

textarea {
    width: 100%;
    padding: 15px; /* Increased padding */
    font-size: 1.2em; /* Increased font size */
    margin-bottom: 20px; /* Increased bottom margin */
    border: 1px solid #444;
    border-radius: 8px; /* Increased border radius */
    resize: vertical;
    background-color: #222;
    color: #e0e0e0;
}

button {
    display: inline-block;
    background-color: transparent;
    color: #00ffcc;
    border: 3px solid #00ffcc; /* Increased border thickness */
    padding: 15px 30px; /* Increased padding */
    font-size: 1.2em; /* Increased font size */
    border-radius: 8px; /* Increased border radius */
    cursor: pointer;
    margin: 10px 0; /* Increased margin */
    width: 100%;
    transition: all 0.3s ease-in-out;
    box-shadow: 0 0 12px #00ffcc, 0 0 24px #00ffcc; /* Increased glow effect */
```

```
}

button:hover {
    background-color: #00ffcc;
    color: #121212;
    box-shadow: 0 0 36px #ff00ff, 0 0 48px #ff00ff;
}

button:active {
    transform: scale(0.95);
}

.footer {
    text-align: center;
    font-size: 1em; /* Increased font size */
    color: #777;
    margin-top: 40px; /* Increased margin */
}

.footer a {
    color: #00ffcc;
    text-decoration: none;
}

.footer a:hover {
    text-decoration: underline;
}

@keyframes neon-glow {
    0% { text-shadow: 0 0 8px #ff00ff, 0 0 16px #ff00ff; }
    50% { text-shadow: 0 0 16px #ff00ff, 0 0 32px #ff00ff; }
```

```
        100% { text-shadow: 0 0 8px #ff00ff, 0 0 16px #ff00ff; }
      }
    </style>
</head>
<body>

    <div class="container">
      <!-- Left Side: Text input, Encrypt Button -->
      <div class="left-side">
        <h1>RSA Encryption and Decryption</h1>
        <label for="data">Text to Encrypt:</label>
        <textarea id="data" rows="6">This is a secret message.</textarea><br><br>

        <button onclick="generateKeys()">Generate RSA Keys</button><br><br>

        <button onclick="encryptData()">Encrypt</button><br><br>
      </div>

      <!-- Right Side: Encrypted/Decrypted Data and Footer -->
      <div class="right-side">
        <label for="encrypted">Encrypted Data:</label><br>
        <textarea id="encrypted" rows="6" readonly></textarea><br><br>

        <button onclick="decryptData()">Decrypt</button><br><br>

        <label for="decrypted">Decrypted Data:</label><br>
        <textarea id="decrypted" rows="6" readonly></textarea><br><br>

        <div class="footer">
                </div>
      </div>
```

```
</div>

<script>
    let publicKey, privateKey;

    // Generate RSA keys
    async function generateKeys() {
        const keyPair = await window.crypto.subtle.generateKey(
            {
                name: "RSA-OAEP",
                modulusLength: 2048,
                publicExponent: new Uint8Array([1, 0, 1]), // 65537
                hash: "SHA-256",
            },
            true,
            ["encrypt", "decrypt"]
        );
        publicKey = keyPair.publicKey;
        privateKey = keyPair.privateKey;
        alert("RSA Key Pair Generated!");
    }

    // Encrypt data using RSA
    async function encryptData() {
        const textToEncrypt = document.getElementById("data").value;
        const encoder = new TextEncoder();
        const encodedText = encoder.encode(textToEncrypt);

        const encryptedData = await window.crypto.subtle.encrypt(
            {
                name: "RSA-OAEP",
```

```
        },
        publicKey,
        encodedText
    );

    document.getElementById("encrypted").value = arrayBufferToBase64(encryptedData);
}

// Decrypt data using RSA
async function decryptData() {
    const encryptedText = document.getElementById("encrypted").value;
    if (!encryptedText) return;

    const encryptedData = base64ToArrayBuffer(encryptedText);

    const decryptedData = await window.crypto.subtle.decrypt(
        {
            name: "RSA-OAEP",
        },
        privateKey,
        encryptedData
    );

    const decoder = new TextDecoder();
    const decryptedText = decoder.decode(decryptedData);
    document.getElementById("decrypted").value = decryptedText;
}

// Helper function to convert ArrayBuffer to Base64 string
function arrayBufferToBase64(buffer) {
    return btoa(String.fromCharCode(...new Uint8Array(buffer)));
```

```
        }

        // Helper function to convert Base64 string to ArrayBuffer
        function base64ToArrayBuffer(base64) {
            const binaryString = atob(base64);
            const length = binaryString.length;
            const arrayBuffer = new Uint8Array(length);
            for (let i = 0; i < length; i++) {
                arrayBuffer[i] = binaryString.charCodeAt(i);
            }
            return arrayBuffer;
        }
    </script>

</body>
</html>
```