```
import React, { useState } from 'react';

import { useEffect } from 'react';



const ImageEncryptionTool = () => {

const [selectedImage, setSelectedImage] = useState<File | null>(null);

const [encryptedImage, setEncryptedImage] = useState<string | null>(null);

const [decryptedImage, setDecryptedImage] = useState<string | null>(null);

const [key, setKey] = useState<number>(1);



const handleImageChange = (event: React.ChangeEvent<HTMLInputElement>) => {

  if (event.target.files) {
```

```
    setSelectedImage(event.target.files[0]);
  }
};

const handleEncrypt = () => {
  if (selectedImage) {
    const reader = new FileReader();
    reader.onload = (event) => {
      if (event.target) {
        const imageDataUrl = event.target.result as string;
        const image = new Image();
```

```
image.onload = () => {
  const canvas = document.createElement('canvas');
  canvas.width = image.width;
  canvas.height = image.height;
  const ctx = canvas.getContext('2d');
  if (ctx) {
    ctx.drawImage(image, 0, 0);
    const pixels = ctx.getImageData(0, 0, canvas.width, canvas.height);
    for (let i = 0; i < pixels.data.length; i += 4) {
      pixels.data[i] = (pixels.data[i] + key) % 256;
      pixels.data[i + 1] = (pixels.data[i + 1] + key) % 256;
```

```
          pixels.data[i + 2] = (pixels.data[i + 2] + key) % 256;
        }
        ctx.putImageData(pixels, 0, 0);
        setEncryptedImage(canvas.toDataURL());
      }
    };
    image.src = imageDataUrl;
  }
};
reader.readAsDataURL(selectedImage);
}
```

```
};



const handleDecrypt = () => {
  if (encryptedImage) {
    const image = new Image();
    image.onload = () => {
      const canvas = document.createElement('canvas');
      canvas.width = image.width;
      canvas.height = image.height;
      const ctx = canvas.getContext('2d');
      if (ctx) {
```

```
    ctx.drawImage(image, 0, 0);

    const pixels = ctx.getImageData(0, 0, canvas.width, canvas.height);

    for (let i = 0; i < pixels.data.length; i += 4) {

      pixels.data[i] = (pixels.data[i] - key + 256) % 256;

      pixels.data[i + 1] = (pixels.data[i + 1] - key + 256) % 256;

      pixels.data[i + 2] = (pixels.data[i + 2] - key + 256) % 256;

    }

    ctx.putImageData(pixels, 0, 0);

    setDecryptedImage(canvas.toDataURL());

  }

};
```

```
    image.src = encryptedImage;
  }
};

return (
  <div className="container mx-auto p-4 pt-6 mt-10">
    <h1 className="text-3xl font-bold mb-4">Image Encryption Tool</h1>
    <input
      type="file"
      onChange={handleImageChange}
      className="block w-full text-sm text-gray-500 file:mr-4 file:py-2 file:px-4 file:rounded-full file:border-0 file:text-sm file:font-semibold file:bg-blue-50 file:text-blue-700 hover:file:bg-blue-100"
```

```
/>

{selectedImage && (

  <button

    onClick={handleEncrypt}

    className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded mt-4"

  >

    Encrypt

  </button>

)}

{encryptedImage && (

  <div className="mt-4">
```

```
    <h2 className="text-2xl font-bold mb-2">Encrypted Image</h2>
    <img src={encryptedImage} alt="Encrypted Image" />
    <button
      onClick={handleDecrypt}
      className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded mt-4"
    >
      Decrypt
    </button>
  </div>
)}
{decryptedImage && (
```

```
  <div className="mt-4">
    <h2 className="text-2xl font-bold mb-2">Decrypted Image</h2>
    <img src={decryptedImage} alt="Decrypted Image" />
  </div>
)}
<div className="mt-4">
  <label className="block text-sm font-medium text-gray-700">Key</label>
  <input
    type="number"
    value={key}
    onChange={(e) => setKey(parseInt(e.target.value))}
```

```
        className="block w-full text-sm text-gray-500 p-2 rounded border border-gray-300 focus:outline-none focus:ring-blue-500 focus:border-blue-500"

      />

    </div>

  </div>

);

};




export default ImageEncryptionTool;
```