Praca zaliczeniowa dla przedmiotu Bezpieczeństwo Systemów Informatycznych – Julia Kućmierczyk

Projekt jest tworzony w technologiach React + Typescript, staram się o ocenę 5.

Projekt posiada nawigację między poszczególnymi szyframi która wygląda następująco:

Klikając w poszczególny przycisk nawiguje się między rozwiązaniami szyfru.

- Projekt można znaleźć w przeglądarce pod poniższym linkiem:

https://security-cipher.vercel.app/

- Kod źródłowy znajduję się w repozytorium na platformie Github:

https://github.com/JuliaKucmierczyk/security-cipher

julka kućmierczyk

Streszczenie

Komponent Szyfru Cezara pozwala użytkownikowi na szyfrowanie i deszyfrowanie tekstu z wykorzystaniem polskiego alfabetu. Jest to implementacja klasycznego szyfru przesuwającego, który przesuwa każdą literę o ustaloną liczbę miejsc w alfabecie.

Opis Komponentu

Komponent Szyfru Cezara umożliwia użytkownikowi wprowadzenie tekstu i wybranie przesunięcia, które zostanie zastosowane do szyfrowania lub deszyfrowania tekstu. Użytkownik może wprowadzić dowolny tekst w języku polskim, a komponent poprawnie obsłuży specyficzne znaki diakrytyczne.

Funkcje Szyfrujące i Deszyfrujące

- **Szyfrowanie (encrypt)**: Funkcja przyjmuje tekst i liczbę, która reprezentuje przesunięcie. Każda litera w tekście jest przesuwana w alfabecie o liczbę miejsc równą przesunięciu. Jeśli przesunięcie przekroczy koniec alfabetu, to kontynuuje od początku.
- **Deszyfrowanie (decrypt)**: Funkcja działa odwrotnie do szyfrowania, przesuwając litery wstecz o podaną liczbę miejsc.

Interfejs Użytkownika

Elementy Wejścia

- **Pole tekstowe**: Umożliwia użytkownikowi wprowadzenie tekstu, który ma być zaszyfrowany lub odszyfrowany.
- **Pole numeryczne**: Umożliwia użytkownikowi wprowadzenie liczby, która określa przesunięcie w szyfrze.

Przyciski

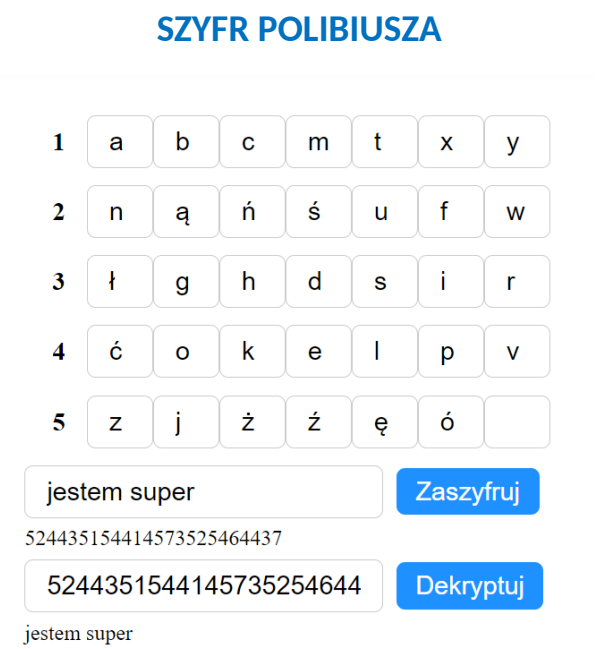
- **Zaszyfruj**: Po kliknięciu szyfruje wprowadzony tekst zgodnie z podanym przesunięciem.
- **Odszyfruj**: Po kliknięciu deszyfruje wprowadzony tekst przy użyciu tego samego przesunięcia.

Wyświetlanie Wyników

- **Kontener Wyniku**: Pokazuje wynik szyfrowania lub deszyfrowania wprowadzonego tekstu.

Logika Walidacji

Komponent zawiera walidację danych wejściowych, aby upewnić się, że wszystkie pola są wypełnione zanim przyciski zostaną użyte. Dodatkowo, komponent informuje użytkownika, jeśli przesunięcie nie jest podane lub jeśli pole tekstowe jest puste.



Streszczenie

Komponent Szyfru Polibiusza w React pozwala użytkownikowi na szyfrowanie i deszyfrowanie tekstu przy użyciu zdefiniowanej przez użytkownika tablicy Polibiusza. Użytkownik może wprowadzić dowolne litery do tablicy 5x7, a następnie użyć tej tablicy do zaszyfrowania i odszyfrowania tekstu.

Opis Komponentu

Struktura Tablicy

Tablica szyfru Polibiusza jest tworzona jako siatka 5x7, gdzie użytkownik ma możliwość wprowadzenia własnego zestawu znaków, tworząc unikalny szyfr.

Szyfrowanie

Funkcja **encrypt** przejmuje wprowadzony tekst i przekształca go w serię cyfr odpowiadających współrzędnym każdej litery w tablicy Polibiusza. Każda litera jest zastąpiona przez parę cyfr - pierwsza cyfra reprezentuje wiersz, a druga kolumnę w tablicy.

Deszyfrowanie

Funkcja **decrypt** przetwarza ciąg cyfr z powrotem na tekst jasny. Oczekuje serii cyfr podzielonych w pary, gdzie każda para odpowiada współrzędnym w tablicy Polibiusza.

Walidacja

Komponent zapewnia walidację danych wejściowych:

- Wszystkie pola w tablicy muszą być wypełnione.
- Wartości w polach tablicy muszą być unikatowe.

Jeśli te warunki nie są spełnione, komponent wyświetla ostrzeżenie i zatrzymuje proces szyfrowania lub deszyfrowania.

Dodatkowy warunek: użycie jakiegoś działania matematycznego na liczbach

Jeśli liczba jest liczbą parzystą zostanie poddania działaniu potęgowania.

Interfejs Użytkownika

Wprowadzanie Tekstu

- **Pole tekstowe do wprowadzenia tekstu**: Użytkownik wpisuje tekst, który chce zaszyfrować.
- **Pole tekstowe do wprowadzenia zaszyfrowanych danych**: Użytkownik wpisuje zaszyfrowane dane (cyfry) do odszyfrowania.

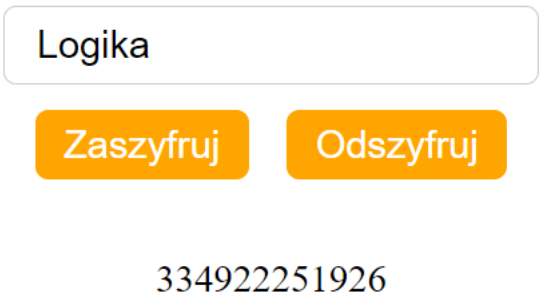
Przyciski

- **Zaszyfruj**: Po kliknięciu, tekst wprowadzony przez użytkownika jest szyfrowany.
- **Odszyfruj**: Po kliknięciu, wprowadzone zaszyfrowane dane są deszyfrowane.

Wyświetlanie Wyników

- **Pole wynikowe dla zaszyfrowanego tekstu**: Wyświetla zaszyfrowaną wersję wprowadzonego tekstu.
- **Pole wynikowe dla odszyfrowanego tekstu**: Wyświetla odszyfrowany tekst.

SZYFR HOMOFONICZNY



Streszczenie

Szyfr Homofoniczny jest metodą kryptograficzną, która przypisuje każdej literze tekstu jawngo jeden z wielu możliwych homofonów (kodów) w celu zaszyfrowania. W odróżnieniu od tradycyjnych szyfrów, które przypisują każdej literze jeden stały symbol, szyfr homofoniczny używa wielu symboli do reprezentowania jednej litery, co utrudnia kryptoanalizę poprzez analizę częstotliwości.

Opis Projektu

Projekt implementuje interfejs użytkownika (UI) oraz logikę szyfrowania i deszyfrowania w języku React z TypeScriptem. Użytkownik może wprowadzić dowolny tekst, który zostanie zaszyfrowany z użyciem wcześniej zdefiniowanego zestawu homofonów dla polskiego alfabetu.

Interfejs Użytkownika

Interfejs zawiera następujące elementy:

- **Pole tekstowe**: Użytkownik wprowadza tekst do zaszyfrowania lub odszyfrowania.
- **Przycisk "Zaszyfruj"**: Inicjuje proces szyfrowania wprowadzonego tekstu.
- **Przycisk "Odszyfruj"**: Inicjuje proces deszyfrowania wprowadzonego ciągu cyfr.
- **Pole wynikowe**: Wyświetla zaszyfrowany tekst jako ciąg cyfr lub odszyfrowany tekst jako ciąg znaków.

Logika Szyfrowania

Szyfrowanie odbywa się przez iterację po każdej literze tekstu wejściowego i wybór losowego homofonu z predefiniowanego zestawu odpowiadającego tej literze. Zaszyfrowany tekst jest reprezentowany jako ciąg cyfr, z których każda grupa odpowiada jednej literze.

Logika Deszyfrowania

Deszyfrowanie wymaga odwrotnej operacji do szyfrowania, co oznacza przekształcenie każdej grupy cyfr z powrotem na odpowiadającą literę. Jest to możliwe tylko wtedy, gdy znamy dokładny homofon użyty do zaszyfrowania danej litery, co w praktycznym zastosowaniu wymagałoby dodatkowego mechanizmu śledzenia użytych homofonów lub użycia deterministycznego algorytmu do ich wyboru.

Zastosowanie

Szyfr Homofoniczny jest szczególnie przydatny w sytuacjach, gdzie chcemy zniwelować skuteczność analizy częstotliwości w próbach złamania szyfru. Może być używany do celów edukacyjnych, pokazując zaawansowany sposób ochrony przed prostą analizą statystyczną.

Implementacja

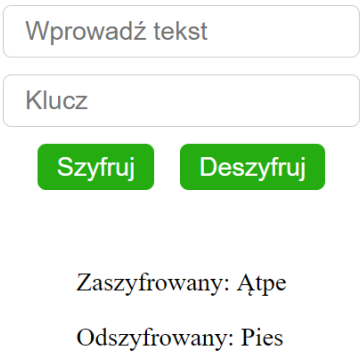
Implementacja szyfru homofonicznego w tym projekcie zakłada użycie następujących technologii i metod:

- **React** jako biblioteka do budowy interfejsu użytkownika.
- **TypeScript** dla typowania statycznego i lepszej kontroli nad kodem.
- **JSON** do przechowywania mapowania homofonów.

Podgląd pliku homophones.json


```
{
    "a": ["101", "321", "633", "859", "527",
    "850" , "162", "228", "926"],
    "ą": ["39", "02"],
    "b": ["91", "47"],
    "c": ["128", "035", "741", "532"],
    "ć": ["10", "01"],
    "d": ["512", "313", "814"],
    "e": ["088", "075", "898", "959", "239",
    "992", "911", "723"],
    "ę": ["17", "18"],
    "f": ["19", "20"],
    "g": ["21", "22"],
    "h": ["23", "24"],
    "i": ["25", "00", "77", "99", "69", "41",
    "32", "23"],
    "j": ["557", "378"],
    "k": ["30", "31", "1", "03"],
    "l": ["32", "33"],
    "ł": ["34", "35"],
    "m": ["36", "37" , "11"],
    "n": ["38", "39", "40", "401", "42", "43"],
    "ń": ["151", "324"],
    "o": ["44", "45", "46", "47", "48", "49",
    "50", "51"],
    "ó": ["912", "842"],
    "p": ["777", "75", "768"],
    "q": ["274", "530"],
    "r": ["51", "562", "364", "285", "523"],
    "s": ["253", "554", "349", "974"],
    "ś": ["55", "56"],
    "t": ["587", "588", "465", "445"],
    "u": ["59", "60", "61"],
    "y": ["611", "612"]
```

SZYFR TRITHEMIUSZA Z KLUCZEM



Streszczenie

Szyfr Trithemiusza to klasyczna metoda szyfrowania, która jest ulepszoną formą szyfru Cezara. Polega na zastosowaniu przesunięcia liter w alfabecie, które zwiększa się z każdą kolejną literą tekstu jawnego. W tej specjalnej wersji szyfru dodano klucz startowy, który definiuje początkowe przesunięcie dla pierwszej litery tekstu.

Opis Projektu

Projekt realizuje interfejs użytkownika (UI), który pozwala na szyfrowanie i deszyfrowanie tekstu zgodnie z zasadami szyfru Trithemiusza z wykorzystaniem **polskiego** alfabetu. Użytkownik wprowadza tekst do zaszyfrowania, podaje pojedynczy znak jako klucz, a następnie może zaszyfrować lub odszyfrować tekst.

Składniki interfejsu

- **Pole tekstowe dla tekstu jawnego**: Użytkownik wprowadza tekst, który chce zaszyfrować.
- **Pole tekstowe dla klucza**: Użytkownik wprowadza pojedynczą literę, która będzie użyta jako klucz startowy szyfru.
- **Przycisk "Szyfruj"**: Po naciśnięciu, wprowadzony tekst jest szyfrowany z wykorzystaniem podanego klucza.
- **Przycisk "Deszyfruj"**: Po naciśnięciu, zaszyfrowany tekst jest deszyfrowany z powrotem do formy jasnej, używając tego samego klucza.
- **Pole wyjściowe dla zaszyfrowanego tekstu**: Wyświetla wynik szyfrowania.
- **Pole wyjściowe dla odszyfrowanego tekstu**: Wyświetla wynik deszyfrowania.

Funkcje Kodu

Kod realizuje dwie główne funkcje:

- **trithemiusEncrypt(text: string, key: string): string**: Szyfruje podany tekst, wykorzystując podany klucz. Każda litera tekstu jawnego jest przesuwana o wartość zależną od jej pozycji oraz pozycji klucza w alfabecie.
- **trithemiusDecrypt(encryptedText: string, key: string): string**: Odszyfrowuje tekst, stosując odwrotny proces do szyfrowania.