ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ԿՈՒՐՍԱՅԻՆ ԱՇԽԱՏԱՆՔ



Ինստիտուտ ՝ ՏՀՏԷ

Խումբ ՝ ՏՏ219-2

Առարկա ՝ ԾՀՊ

Թեմա ՝ Գաղտնագրում և Վերծանում

Ուսանող ՝ Լ․ Մաթոսյան

Դասախոս ՝ Ա․ Մարտիրոսյան

**Բովանդակություն**

Ընդհանուր նկարագրություն ․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․ 3

Ներածություն ….……………….......……………………………………………………….….…...... 4

Գաղտնագրում ………...................…………..………………………….…………………………… 4

Ալգորիթմները ………………………….........……………………………………………....………․.. 6

Ծրագրի մասին հակիրճ ․․․․․․․․․․․․․․․․․․․․․…………………․․․․․․․․․․․․․․․……………...․․․․․․․․․․․․․․․․․․․ 11

Ծրագրի կատարման քայլերը ․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․……….․․․․․․․․․․․․․․․․․․……………..․ 12

Սերվերի դերը համակարգում ․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․․ 20

**Ընդհանուր նկարագրություն**

Ծրագրավորման լեզու - React(JSX), Redux, CSS, Node.js (JavaScript)

Ծրագրավորման միջավայր - Visual Studio Code

Օգտագործվող գրադարաններ - CryptoJS, React, Codepen.io

Ծրագրային համակարգի նկարագրությունը` Ցանցային

Client 🡨 ---------------------- > Client

Client 🡨 ---------------------- > Server

Ծրագրային նախագծի անվանումը, նկարագրությունը`

## Web page with encryption and decryption

***Ներածություն***

***Գաղտնագրում***

Կոդավորումը տվյալների խառնման միջոց է, որպեսզի միայն լիազորված կողմերը կարողանան հասկանալ տեղեկատվությունը: Տեխնիկական առումով դա մարդու կողմից ընթեռնելի պարզ տեքստի վերածելու գործընթացն է անհասկանալի տեքստի, որը նաև հայտնի է որպես գաղտնագրված տեքստ: Ավելի պարզ ասած, կոդավորումը վերցնում է ընթեռնելի տվյալներ և փոխում դրանք այնպես, որ պատահական երևան: Կոդավորումը պահանջում է գաղտնագրային բանալի՝ մի շարք մաթեմատիկական արժեքների, որոնց շուրջ համաձայն են գաղտնագրված հաղորդագրություն ուղարկողը և ստացողը:

**Ինչպե՞ս է աշխատում կոդավորումը:**

Կոդավորումը մաթեմատիկական գործընթաց է, որը փոխում է տվյալները՝ օգտագործելով գաղտնագրման ալգորիթմ և բանալի: Պատկերացրեք, եթե Ալիսը ուղարկի «Բարև» հաղորդագրությունը Բոբին, բայց նա իր հաղորդագրության յուրաքանչյուր տառը փոխարինի այբուբենի երկու տեղից հետո գրված տառով: «Բարև»-ի փոխարեն նրա ուղերձում այժմ գրված է «Ջգննք»: Բարեբախտաբար, Բոբը գիտի, որ բանալին «2» է և կարող է վերծանել իր հաղորդագրությունը «Բարև»:

Ալիսն օգտագործեց չափազանց պարզ կոդավորման ալգորիթմ՝ Բոբին ուղղված իր հաղորդագրությունը կոդավորելու համար: Ավելի բարդ գաղտնագրման ալգորիթմները կարող են ավելի շատ խառնել հաղորդագրությունը.



Թեև կոդավորված տվյալները պատահական են թվում, գաղտնագրումն ընթանում է տրամաբանական, կանխատեսելի ձևով, որը թույլ է տալիս կողմին, որը ստանում է գաղտնագրված տվյալները և ունի ճիշտ բանալին՝ վերծանելու տվյալները՝ վերածելով դրանք պարզ տեքստի: Իսկապես անվտանգ կոդավորումը կօգտագործի այնքան բարդ ստեղներ, որ երրորդ կողմը դժվար թե կարողանա ապակոդավորել կամ կոտրել գաղտնագրված տեքստը կոպիտ ուժի միջոցով, այլ կերպ ասած՝ գուշակելով բանալին: (Ալիսի գաղտնագրման առաջին մեթոդը շատ արագ կխախտվի):

Տվյալները կարող են գաղտնագրվել «հանգիստ վիճակում», երբ դրանք պահվում են կամ «փոխանցման մեջ», մինչդեռ դրանք փոխանցվում են այլ տեղ:

**Ի՞նչ է բանալին ծածկագրության մեջ:**

Կրիպտոգրաֆիկ բանալին նիշերի շարան է, որն օգտագործվում է գաղտնագրման ալգորիթմում տվյալների փոփոխման համար, որպեսզի դրանք պատահական երևան: Ֆիզիկական բանալիի նման, այն կողպում է (գաղտնագրում) տվյալները, որպեսզի միայն ճիշտ բանալի ունեցողը կարողանա բացել (գաղտնազերծել) դրանք:

**Որո՞նք են գաղտնագրման տարբեր տեսակները:**

Գաղտնագրման երկու հիմնական տեսակներն են՝ սիմետրիկ կոդավորումը և ասիմետրիկ կոդավորումը: Ասիմետրիկ գաղտնագրումը հայտնի է նաև որպես հանրային բանալիների կոդավորում:

Սիմետրիկ գաղտնագրման մեջ կա միայն մեկ բանալի, և բոլոր հաղորդակցվող կողմերն օգտագործում են նույն (գաղտնի) բանալին ինչպես կոդավորման, այնպես էլ վերծանման համար: Ասիմետրիկ կամ հանրային բանալին գաղտնագրման մեջ կա երկու բանալի՝ մեկ բանալի օգտագործվում է գաղտնագրման համար, իսկ մեկ այլ բանալի՝ ապակոդավորման համար։ Ապակոդավորման բանալին պահվում է մասնավոր (այստեղից՝ «մասնավոր բանալի» անունը), մինչդեռ գաղտնագրման բանալին համօգտագործվում է բոլորի համար (այստեղից՝ «հանրային բանալին» անվանումը): Ասիմետրիկ գաղտնագրումը հիմնարար տեխնոլոգիա է TLS-ի համար (հաճախ կոչվում է SSL):

**Ինչու՞ է գաղտնագրումը կարևոր:**

Գաղտնիություն. Կոդավորումը երաշխավորում է, որ ոչ ոք չի կարող կարդալ հաղորդակցությունները կամ տվյալները հանգիստ վիճակում, բացառությամբ նախատեսված ստացողի կամ տվյալների օրինական սեփականատիրոջ: Սա կանխում է հարձակվողներին, գովազդային ցանցերին, ինտերնետ ծառայություններ մատուցողներին և որոշ դեպքերում կառավարություններին գաղտնի տվյալներ որսալու և կարդալուց՝ պաշտպանելով օգտատերերի գաղտնիությունը:

Անվտանգություն. գաղտնագրումն օգնում է կանխել տվյալների խախտումները, անկախ նրանից՝ տվյալները փոխանցվում են, թե հանգստանում են: Եթե կորպորատիվ սարքը կորչի կամ գողանա, և դրա կոշտ սկավառակը պատշաճ կերպով կոդավորված է, այդ սարքի տվյալները դեռ ապահով կլինեն: Նմանապես, կոդավորված հաղորդակցությունները հնարավորություն են տալիս հաղորդակցվող կողմերին փոխանակել զգայուն տվյալներ՝ առանց տվյալների արտահոսքի:

Տվյալների ամբողջականությունը. Կոդավորումը նաև օգնում է կանխել վնասակար վարքագիծը, ինչպիսիք են ուղու վրա հարձակումները: Երբ տվյալները փոխանցվում են ինտերնետով, կոդավորումը երաշխավորում է, որ այն, ինչ ստացողը ստանում է, ճանապարհին չի դիտվել կամ կեղծվել:

**Կանոնակարգեր.** Այս բոլոր պատճառներով, արդյունաբերության և պետական կանոնակարգերից շատերը պահանջում են, որ ընկերությունները, որոնք մշակում են օգտվողների տվյալները, պահպանեն այդ տվյալները գաղտնագրված: Կարգավորող և համապատասխանության ստանդարտների օրինակներ, որոնք պահանջում են գաղտնագրում, ներառում են HIPAA, PCI-DSS և GDPR:

**Ի՞նչ է գաղտնագրման ալգորիթմը:**

Գաղտնագրման ալգորիթմը այն մեթոդն է, որն օգտագործվում է տվյալների գաղտնագրման տեքստի վերածելու համար: Ալգորիթմը կօգտագործի գաղտնագրման բանալին, որպեսզի փոխի տվյալները կանխատեսելի ձևով, այնպես որ, չնայած գաղտնագրված տվյալները պատահական կհայտնվեն, դրանք կարող են հետ վերածվել պարզ տեքստի՝ օգտագործելով վերծանման բանալի:

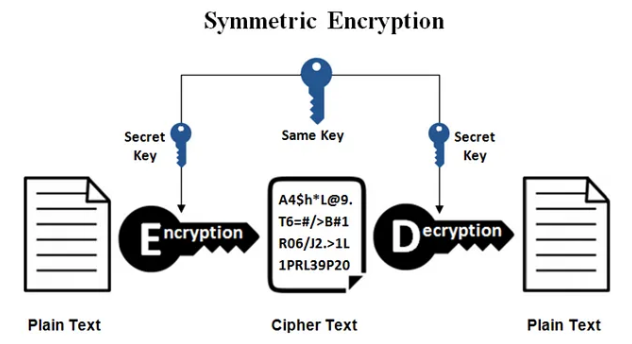
**Որո՞նք են գաղտնագրման ընդհանուր ալգորիթմները:**

Սովորաբար օգտագործվող սիմետրիկ գաղտնագրման ալգորիթմները ներառում են.

* **AES**
* **3-DES**
* **SNOW**

Սովորաբար օգտագործվող ասիմետրիկ գաղտնագրման ալգորիթմները ներառում են.

* **RSA**
* **Էլիպսային կորի ծածկագրում**

****

**Ալգորիթմները, որոնք օգտագործվելու են ծրագրի մեջ`**

1. **AES**

Ընդլայնված գաղտնագրման ստանդարտը (AES) 2001 թվականին ԱՄՆ Ստանդարտների և տեխնոլոգիաների ազգային ինստիտուտի (NIST) կողմից հաստատված էլեկտրոնային տվյալների գաղտնագրման հստակեցում է: AES-ն այսօր լայնորեն օգտագործվում է, քանի որ այն շատ ավելի ուժեղ է, քան DES-ը և եռակի DES-ը, չնայած ավելի դժվարին: իրականացնել.

Հիշելու կետեր

AES-ը բլոկային ծածկագիր է:

Բանալու չափը կարող է լինել 128/192/256 բիթ:

Կոդավորում է տվյալները յուրաքանչյուրը 128 բիթանոց բլոկներով:

Դա նշանակում է, որ այն վերցնում է 128 բիթ որպես մուտք, իսկ 128 բիթ կոդավորված ծածկագրված տեքստ է թողարկվում որպես ելք: AES-ը հիմնված է փոխարինման-փոխակերպման ցանցի սկզբունքի վրա, ինչը նշանակում է, որ այն իրականացվում է մի շարք կապակցված գործողությունների միջոցով, որոնք ներառում են մուտքային տվյալների փոխարինում և խառնում:

Գաղտնագրման աշխատանք.

AES-ը գործողություններ է կատարում տվյալների բայթերով, այլ ոչ թե բիթերով: Քանի որ բլոկի չափը 128 բիթ է, ծածկագիրը միաժամանակ մշակում է մուտքային տվյալների 128 բիթ (կամ 16 բայթ):

Փուլերի քանակը կախված է բանալին երկարությունից հետևյալ կերպ.

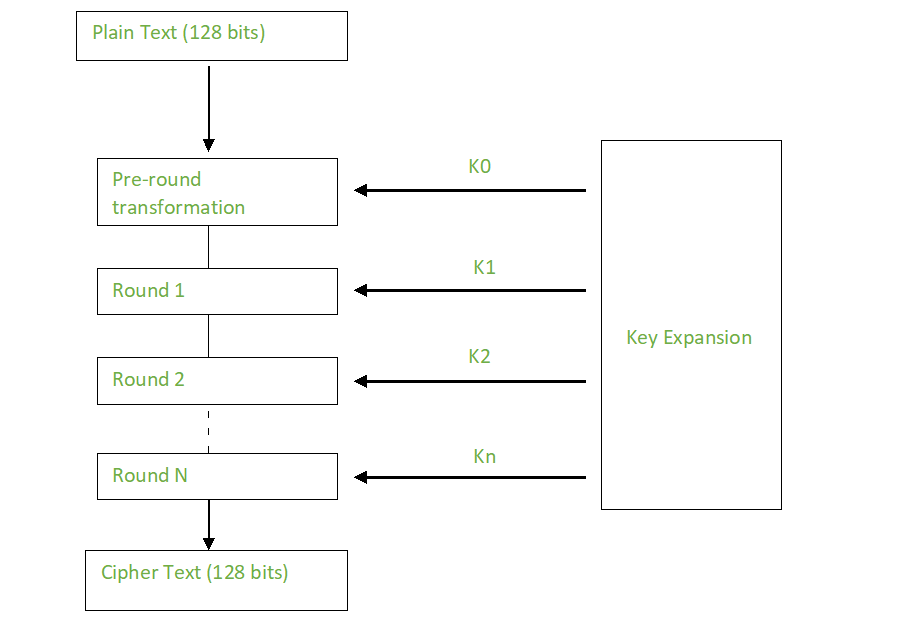
128 բիթանոց բանալի – 10 փուլ

192 բիթանոց բանալի – 12 փուլ

256 բիթանոց բանալի – 14 փուլ

**Կլոր ստեղների ստեղծում.**

Key Schedule ալգորիթմն օգտագործվում է բանալիից բոլոր կլոր ստեղները հաշվարկելու համար: Այսպիսով, սկզբնական բանալին օգտագործվում է բազմաթիվ տարբեր կլոր բանալիներ ստեղծելու համար, որոնք կօգտագործվեն գաղտնագրման համապատասխան փուլում:



**Կոդավորումը.**

AES-ը յուրաքանչյուր բլոկ դիտարկում է որպես 16 բայթ (4 բայթ x 4 բայթ = 128) ցանց սյունակի հիմնական դասավորության մեջ:

Յուրաքանչյուր փուլը բաղկացած է 4 քայլից.`

* Ենթաբայթեր
* ShiftRows
* MixColumns

Ավելացնել կլոր բանալի

Վերջին փուլը չունի MixColumns փուլ:

1. **Blowfish**

Blowfish-ը գաղտնագրման տեխնիկա է, որը մշակվել է Բրյուս Շնայերի կողմից 1993 թվականին՝ որպես DES ծածկագրման տեխնիկայի այլընտրանք: Այն զգալիորեն ավելի արագ է, քան DES-ը և ապահովում է գաղտնագրման լավ արագություն՝ մինչ օրս հայտնաբերված ոչ մի արդյունավետ կրիպտովերլուծության տեխնիկա: Այն առաջին, անվտանգ բլոկային գաղտնագրերից է, որը ենթակա չէ որևէ արտոնագրի և, հետևաբար, ազատորեն հասանելի է բոլորի համար օգտագործելու համար: Դա սիմետրիկ բլոկային ծածկագրման ալգորիթմ է:

* **Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

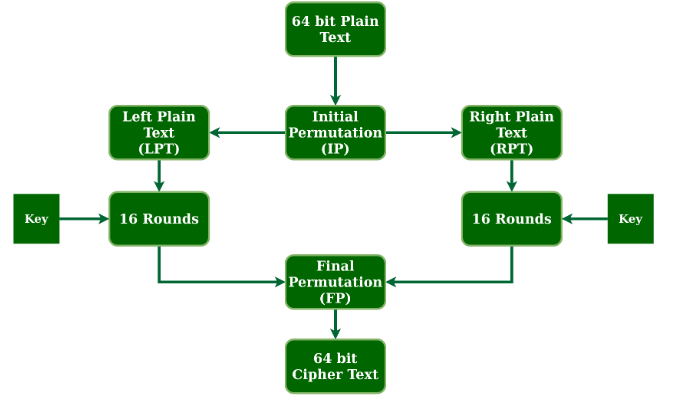
  Автоматически созданное описание**բլոկի չափը՝ 64 բիթ
* keySize: 32-bit-ից 448-bit փոփոխական չափը
* ենթաբանալների քանակը՝ 18 [P-array]
* raund-ների քանակը՝ 16
* փոխարինման տուփերի քանակը՝ 4 [յուրաքանչյուրն ունի 32 բիթանոց 512 մուտք]

1. **DES**

Տվյալների գաղտնագրման ստանդարտը (DES) 56-բիթանոց բանալի երկարությամբ բլոկային ծածկագիր է, որը նշանակալի դեր է խաղացել տվյալների անվտանգության մեջ: Տվյալների գաղտնագրման ստանդարտը (DES) խոցելի է գտնվել շատ հզոր հարձակումների նկատմամբ, հետևաբար, DES-ի ժողովրդականությունը փոքր-ինչ նվազել է: DES-ը բլոկային ծածկագիր է և գաղտնագրում է տվյալները յուրաքանչյուրը 64 բիթանոց բլոկներում, ինչը նշանակում է, որ 64 բիթ պարզ տեքստը մուտքագրվում է որպես DES, որը արտադրում է 64 բիթ ծածկագրված տեքստ: Նույն ալգորիթմը և բանալին օգտագործվում են կոդավորման և վերծանման համար՝ չնչին տարբերություններով: Բանալու երկարությունը 56 բիթ է:

Հիմնական գաղափարը ներկայացված է ստորև.

Մենք նշել ենք, որ DES-ն օգտագործում է 56-բիթանոց բանալի։ Փաստորեն, սկզբնական բանալին բաղկացած է 64 բիթից: Այնուամենայնիվ, նախքան DES գործընթացը սկսելը, ստեղնի յուրաքանչյուր 8-րդ բիթը հանվում է 56-բիթանոց բանալի արտադրելու համար: Այսինքն՝ 8, 16, 24, 32, 40, 48, 56 և 64 բիթային դիրքերը հանված են:



1. **MD5**

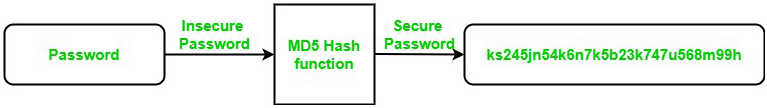
MD5-ը կրիպտոգրաֆիկ հեշ ֆունկցիայի ալգորիթմ է, որն ընդունում է հաղորդագրությունը որպես ցանկացած երկարության մուտքագրում և այն փոխում է 16 բայթ ֆիքսված երկարության հաղորդագրության: MD5 ալգորիթմը նշանակում է հաղորդագրություն-դիջեստ ալգորիթմ: MD5-ը մշակվել է որպես MD4-ի կատարելագործում, առաջադեմ անվտանգության նպատակներով: MD5-ի (Digest չափը) ելքը միշտ 128 բիթ է: MD5-ը մշակվել է 1991 թվականին Ռոնալդ Ռիվեստի կողմից:

MD5 ալգորիթմի օգտագործումը.

Այն օգտագործվում է ֆայլերի նույնականացման համար:

Վեբ հավելվածում այն օգտագործվում է անվտանգության նպատակներով: օր. Օգտագործողների անվտանգ գաղտնաբառը և այլն:

Օգտագործելով այս ալգորիթմը, մենք կարող ենք պահպանել մեր գաղտնաբառը 128 բիթ ձևաչափով:



1. **Rabbit**

Изображение выглядит как текст, диаграмма, снимок экрана, План

Автоматически созданное описаниеRabbit ալգորիթմը հակիրճ կարելի է նկարագրել հետևյալ կերպ. Այն վերցնում է 128-բիթանոց գաղտնի բանալի և 64-բիթանոց IV (ցանկության դեպքում) որպես մուտքագրում և յուրաքանչյուր կրկնության համար ստեղծում է 128 կեղծ պատահական բիթների ելքային բլոկ՝ ներքին վիճակի բիթերի համակցությունից:

**Ծրագրի մասին հակիրճ**

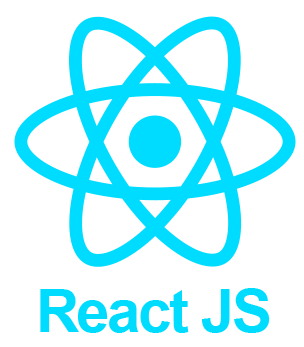
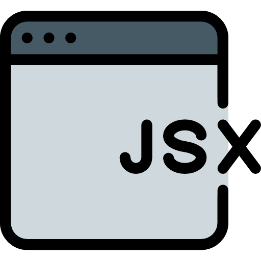
Սկզբի համար պետք է ստեղծենք React հավելված տերմինալի միջոցով․



Գրեթե ամբողջ ֆունկցիաները գրվում է jsx-ում, որը արդեն իր մեջ ներառում է ծրագրի մարմինը՝ HTML-ը։

Ստեղծում ենք մեկ էջանոց հավելված, որտեղ հաճախորդի և սերվերի միջև անցումը(routing) կատարվում է առանց էջի թարմացումների: Այդ գործողությունը իրականացնելու համար օգտագործում ենք Router:

Այժմ նշեմ այն գործիքները որնոնք օգտագործել եմ հավելվածը պատրաստելու մեջ՝



Изображение выглядит как снимок экрана, Графика, Цвет электрик, Прямоугольник

Автоматически созданное описание

Հավելվածի մեջ լինելու է հնարավորություն, որպեսզի հաճախորդը կարողանա ընտրել թե ինչ տեսակի ծածկագրում է ցանկանում իրականացնել՝ թաքնագրություն թե գաղտնագրություն։

Ընտրելով գաղտնեգրություը կբացվի ցուցակ, որտեղ պետք է ընտրի թե կոնկրետ որ ալգորիթմով է ցանկանում գաղտնեգրել տեքստը։

Այս գործողությունները կատարելու համար պետք է տողադրենք Handler-ներ, որոնք պետք է տեսնեն մեր տվյալների վիճակի փոփոխությունները։ Կօգտագործենք Redux-ը մեր տվյալների վիճակները թարմացնելու համար։

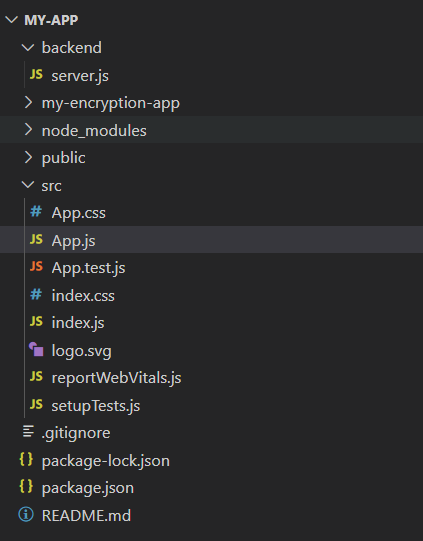
Հավելածը կունենա գաղտնագրման մի քանի ձևեր, որը ընտրելու է հենց user-ը։

**Ծրագրի կատարման քայլերը**

Առաջին քայլը մեր React App ը ստեղծելու հատվածն է։ Մուտք ենք գործում Visual Studio Code, բացում ենք այնտեղի տերմինալը և մեջը գրում ենք հետևյալը՝



Տվյալ գործողությունը կատարելուց հետո ստանում ենք հետևյալը՝



Այնուհետև մուտք ենք գործում App.jsx ֆայլ, որտեղ գրեթե բոլոր գործողությունները և հրամանները գրելու ենք։

Սկզբի համար իմպորտ պետք է անենք use state ը, դինամիկ փոփոխականները հայտարարելու համար։

Իմպորտ ենք անում նաև CryptoJs-ը, որի մասին կխոսեմ մի փոքր ուշ և App.css-ը հետագայում մեր էջին գեղեցիկ տեսքով ապահովելու համար։



Հիմա անցնենք նույն ֆայլում գտնվող return-ի հատված, որտեղ ներմուծելու ենք html կոդը, մեր էջի կառուցվածքը ապահովելու համար։ Մեր էջը կունենա մեկ ընդհանուր կոնտեյնեռ, որը իր հորթին բաղկացաց կլինի մի քանի փոքր կոնտեյնեռներից։

Առաջին div թագի մեջ կդնենք կլասս container անունով, որի մեջ ներառում ենք մի քանի div թագեր նրանց տալով կլասսներ՝ dropDown, button, card, content անուններով։

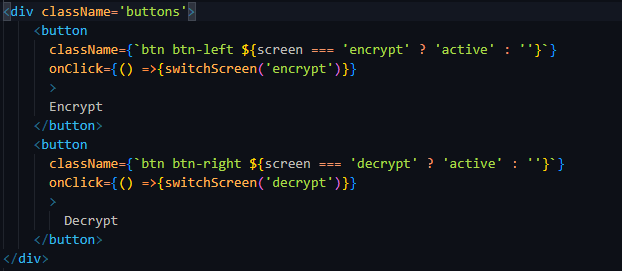
Սկզբից ռեալիզացնենք dropDown ի հատվածը որը իրենից ներկայացնում է Select, որի միջոցով օգտատերը կկարողանա ընտրել թե որ գաղտնագրման ալգորիթմով է ցանկանում կատարել գաղտնագրումը։

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Select-ին տալիս ենք option բոլոր ալգորիթմերի անունները պահելու համար։

Հիմա անցնենք Button-ներին․ Տեղադրվում է երկու Button նախատեսված գաղտնագրման և վերծանման համար։ Նրանց տալիս ենք կլասս և onClick վերջինիս շարժը նկատելու համար։

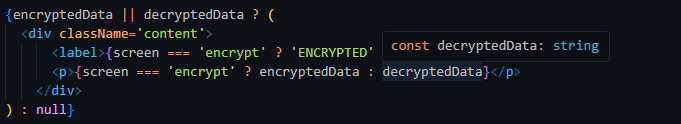


Անցնենք card-ին, որը իր մեջ կներառի տեքստ ներմուծելու վայրը, և ստեղնը։ Կարճ ասաց դա այն բանի համար է, որպեսզի մենք ներմուծենք տեքստը և սեղմելով ստեղնի վրա մեր տեքստը գաղտնագրվի։



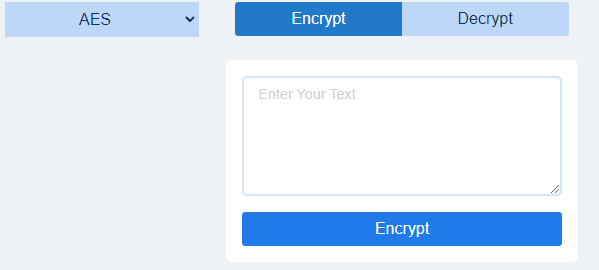
Textarea-ին տալիս ենք OnChange, ներմուծված տեքստի փոփոխությունը նկատելու և թարմացնելու համար, Button-ին տալիս ենք OnClick, ստեղնին դիպչելը նկատելու համար։

Անցնենք մեր էջի վերջին սխեմային, որը content-ն է։ Նրա էությունը հետևյալն է․ տեքստը ներմուծելուց և գաղտնագրում ստեղնը սեղմելուց հետո մեր գաղտնագրված արդյունքը տպվում է էկրանին, կոնկրետ այս դեպքում հենց content-ում։

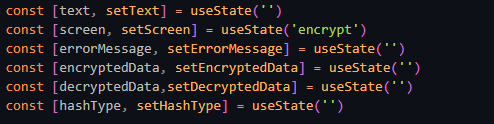


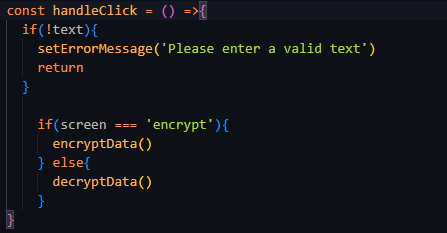
ԵՎ այս դեպքում կախված վիճակից, content-ը կամ կերևա էկրանին կամ ոչ։ Դա կախված է card-ում գտնվող ստեղնից։

Այս բոլոր քայլերը կատարելուց հետո կունենանք հետևյալ տեսքը՝



Տվյալ նկարում content հատվածը չի երևում, քանի որ մեր կոդը դեռ ֆունկցիոնալ հատվածով ապահոված չէ։ Այժմ անցնենք js-ական հատվածին ֆունկցիոնալությունը լրացնելու համար։ Սկզբի համար հայտարարենք փոփոխականները որոնք փոփոխվելու են կախված որևէ բանից: Հայտարարելու ենք արդեն import արած use state ի միջոցով:

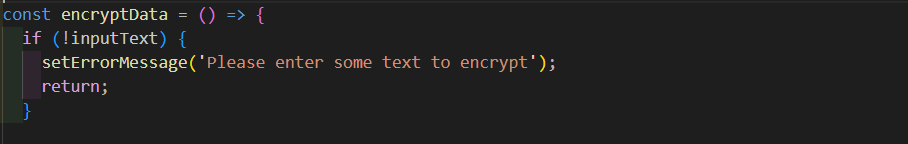


Բոլոր փոփոխականներին հայտարարելիս տալիս ենք նրանց սկզբնական արժեքները։ Այժմ եկեք պատրաստենք տեքստ մուտքագրելու և ստեղնի հատվածը։ Սարքում ենք ֆունկցիա handleClick անունով, որը if-երի միջոցով ստուգում է ամեն բան բավարարում է մեր տված պայմաններին թե ոչ, նախ առաջին if-ով ստուգում է թե արդյոէ կա որևէ ներմուծված տեքստ թե ոչ, եթե չկա ապա ցուցադրում է error message, եթե կա շարունակում է քայլերը։ Երկրորդ if-ը արդեն ստուգում է ներմուծված տեքստը համապատասխանում է պայմաններին թե ոչ։

Հաջորդ ֆունկցիան ստեղծենք hashTypeFuction անունով, որը նախատոսված կլինի select-ը ընտրությունը թարմացնելու համար։



Այժմ ռեալիզացնենք գաղտնագրման և վերծանման ֆունկցիաները։ Ստեղծենք երկու ֆունկցիա encryptData և decryptData անուններով։ Մեզ այժմ պետք կգա արդեն import արած CryptoJS-ը։ Երկու ֆունկցիաներն էլ գրում ենք try/catch ով, catch-ի ժամանակ error դուրս բերելով։





Սա գաղտնագրման ֆունկցիան է, որը աշխատում է դինամիկ, այսինքն կախված թե user-ը որ գաղտնագրման ալգորիթմն է ընտրել, գաղտնագրումը կատարվում է հենց այդ ալգորիթմով և նույննել կատերվում է վերծանման համար։



Գաղտնագրման և վերծանման համար ընտրել եմ ամենահայտնի 5 ալգորիթմները․՝

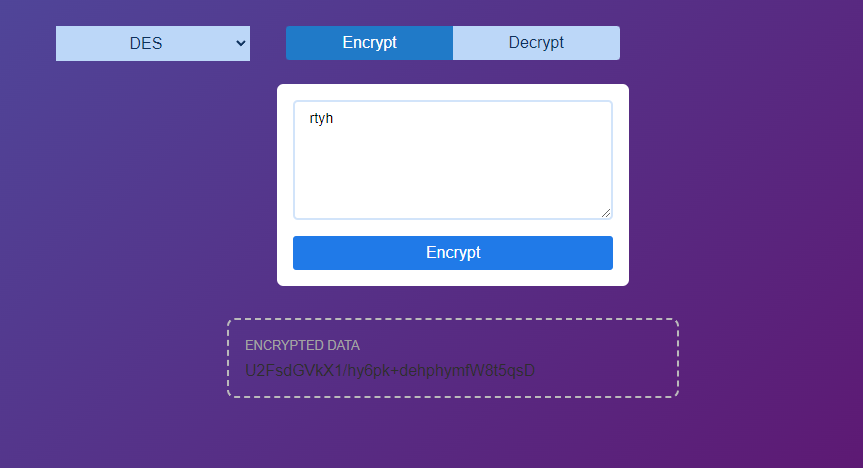
* AES
* Blowfish
* DES
* MD5
* Rabbit

Այժմ ավելացնենք վերջին 2 շտրիխները և մեր կոդի ֆունկցիոնալությունը

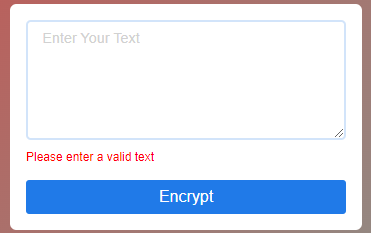
պատրաստ է։ Ավելացնենք հետևյալ ֆունկցիան՝



Այժմ կարող ենք աշխատացնել մեր ծրագիրը, որը կունենա հետևյալ տեսքը։ Բայց աշխատացնելուց առաջ նշեմ որ ավելացրել եմ css գեղեցիկ դիզայն տալու համար։



Իսկ սխալ մուտքագրման դեպքում error message ը հայտվում է էկրանին հետևյալ ձևով՝



**Սերվերի դերը համակարգում**

Իմ ծրագրում սերվերը հանդիսանում է միջնորդային շերտ, որը ապահովում է տվյալների անվտանգ փոխանակումը հաճախորդի (բրաուզերի կամ React հավելվածի) և հետին համակարգի (backend) միջև։ Այս սերվերը հիմնված է Node.js տեխնոլոգիայի վրա՝ օգտագործելով Express.js գրադարանը, որը թույլ է տալիս արագ և հարմար ստեղծել API վերջնակետեր։

Օգտագործվող գրադարաններ և սերվերի աշխատանքը՝



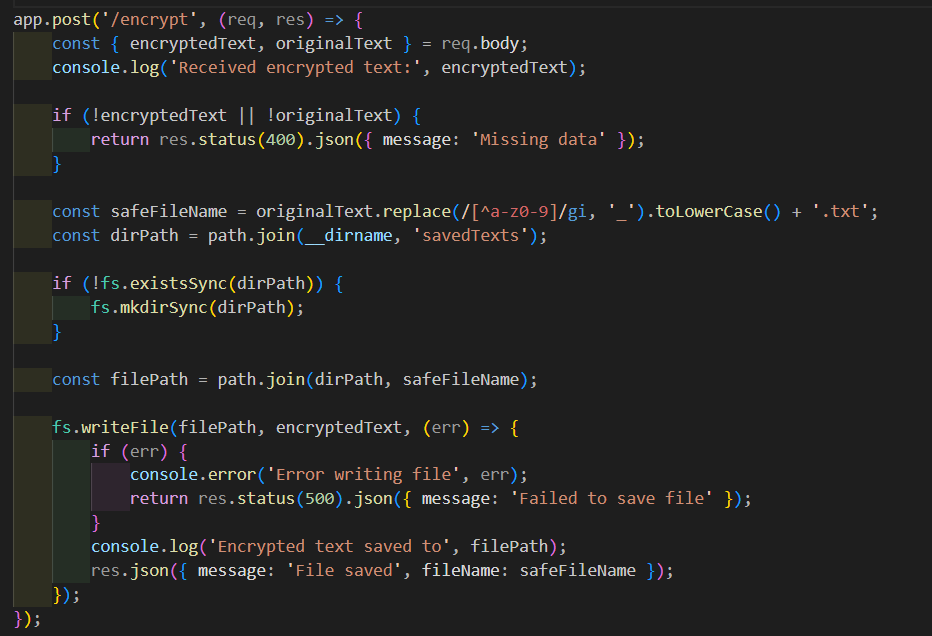
**Սերվերի հիմնական գործառույթները**

1. **Տվյալների ընդունում հաճախորդից**

Երբ օգտատերը մուտքագրում է տեքստ և ընտրում է գաղտնագրման ալգորիթմը, React հավելվածը գաղտնագրում է տեքստը և POST հարցման միջոցով ուղարկում սերվերին՝ ուղարկելով encryptedText կամ decryptedText դաշտով JSON տվյալներ։

1. **Հարցումների մշակում**

Սերվերը ստանում է հարցումը, վերլուծում փոխանցված տվյալները և համապատասխան արձագանք է ուղարկում։ Այս փուլում հնարավոր է կատարել տարբեր լրացուցիչ գործողություններ՝ օրինակ տվյալների պահպանում, վերլուծություն կամ թարմացում։





1. **Անվտանգության ապահովում**

Սերվերը կարող է նաև ստուգել տվյալների վավերությունը, վերահսկել թույլտվությունները և ապահովել, որ փոխանցվող տեղեկատվությունը պաշտպանված լինի արտաքին միջամտություններից՝ օգտագործելով SSL կոդավորում, API բանալիներ կամ այլ մեթոդներ։

**Հաղորդակցություն React հավելվածի հետ**

React հավելվածը fetch() ֆունկցիայի միջոցով կապ է հաստատում սերվերի հետ։ Օգտագործողը գործողություն է կատարում (գաղտնագրում կամ վերծանում), և արդյունքում այդ տվյալները ուղարկվում են հետևյալ հասցեներով.

* Գաղտնագրելու համար՝ http://localhost:5000/encrypt
* Վերծանելու համար՝ http://localhost:5000/decrypt

Սերվերը ստանում է տվյալները, մշակվում են համապատասխան վերջնակետերում (routes), և հաճախորդին վերադարձվում է արձագանք՝ հաջողության կամ սխալի մասին։

