МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

Даянгийн Балжинням

Криптографын зарим алгоритм, программ

(Some algorithms and programs for cryptography)

Програм хангамж(D061302)
Бакалаврын судалгааны ажил

Улаанбаатар

2023 оны 11 сар

МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ
МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ
МЭДЭЭЛЭЛ, КОМПЬЮТЕРИЙН УХААНЫ ТЭНХИМ

Криптографын зарим алгоритм, программ

(Some algorithms and programs for cryptography)

Програм хангамж(D061302)
Баклаврын судалгааны ажил

Удирдагч: _____ Д. Гармаа

Гүйцэтгэсэн: _____ Д. Балжинням (20B1NUM0563)

Улаанбаатар

2023 оны 11 сар

Зохиогчийн баталгаа

Миний бие Даянгийн Балжинням ”Криптографын зарим алгоритм, программ” сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Ажил нь бүхэлдээ эсвэл ихэнхдээ Монгол Улсын Их Сургуулийн зэрэг горилохоор дэвшүүлсэн болно.
- Энэ ажлын аль нэг хэсгийг эсвэл бүхлээр нь ямар нэг их, дээд сургуулийн зэрэг горилохоор оруулж байгаагүй.
- Бусдын хийсэн ажлаас хуулбарлаагүй, ашигласан бол ишлэл, зүүлт хийсэн.
- Ажлыг би өөрөө (хамтарч) хийсэн ба миний хийсэн ажил, үзүүлсэн дэмжлэгийг дипломын ажилд тодорхой тусгасан.
- Ажилд тусалсан бүх эх сурвалжид талархаж байна.

Гарын үсэг: _____

Огноо: _____

ГАРЧИГ

ЗУРГИЙН ЖАГСААЛТ

ХҮСНЭГТИЙН ЖАГСААЛТ

Кодын жагсаалт

УДИРТГАЛ

Энэхүү дипломын ажилд криптографийн янз бүрийн алгоритм, программуудыг системтэйгээр судалсан бөгөөд үндсэн зорилго нь тэдгээрийн үндсэн бүтэц, үйл ажиллагааны механизм, практик хэрэглээг ойлгох явдал юм. Энэхүү судалгааны ажилд уламжлалт болон шинээр гарч ирж буй криптографийн алгоритмуудыг судалж, гүйцэтгэл, аюулгүй байдал, үр ашигтай байдалд үндэслэн харьцуулсан судалгааг хийв.

Энэхүү судалгаанд өгөгдлийн шифрлэлтийн стандарт (DES), дэвшилтэт шифрлэлтийн стандарт (AES), Ривест-Шамир-Адлеман (RSA), эллиптик муруй криптографи (ECC) зэрэг тэгш хэмтэй болон тэгш бус криптограф алгоритмуудыг нарийвчлан судалсан.

Төгсөлтийн ажлын практик хэсэгт хэд хэдэн криптографийн программуудыг боловсруулж харьцуулсан ба орчин үеийн стандартыг хангасан тоон гарын үсгийн системийг үүлэн технологид суурилан бүтээсэн.

Зорилго

Үүлэн технологид суурилсан тоон гарын үсгийн системийг бүтээснээр хэрэглэгчид өөрсдийн цахим гарын үсгээр баталгаажсан файлуудыг интернэт хуваалцах боломжийг бүрдүүлэх гол зорилготой юм.

Зорилт

Бүрэн бүтэн байдал нь хөндөгдөөгүй, эх сурвалж нь тодорхой файлыг хуваалцах боломжийг бүрдүүлэх.

Үндэслэл

Монголд одоогийн байдлаар үүлэн технологид суурилсан тоон гарын үсгийн систем байхгүй байгаа нь хэрэглэгчид энэхүү технологийг ашиглахад төвөгтэй болгож байна. Ихэнх клиент программууд нь зөвхөн Windows үйлдлийн систем дээр ажиллахаар хийгдсэн нь нийцтэй байдлыг хангахгүй байна.

1. ОНОЛЫН СУДАЛГАА

1.1 Тэгш хэмт крифтограф

Тэгш хэмт крифтографт шифрлэлт болон шифр тайлах түлхүүрүүд адил байна. Тэгш хэмт алгоритм нь Тэгш бус хэмт шифрлэлтээс харьцангуй хурдан ажилдаг. Гэвч нууцалсан мэдээллийг тайлж унших түлхүүр болон нууцлах түлхүүр адилхан байх нь харилцагч талууд урьдчилан түлхүүрээ хоорондоо тохиролцох шаардлагыг гаргаж ирдэг. Энэ нь сул тал болох эрсдэлтэй. Хэрвээ гуравдагч этгээд түлхүүрийг олж авбал бүх нууцалсан мэдээллийг үзэх боломжтой болох юм.

Хамгийн түгээмэл хэрэглэгддэг тэгш хэмт шифрлэлтийн алгоритм бол Бельгийн криптографич Жоан Даемен, Винсент Рижмен нарын боловсруулсан Advanced Encryption Standard (AES) юм. AES нь хуучин Data Encryption Standard (DES)-ийг сольсон бөгөөд одоо дэлхий даяар ашиглагдаж байна.[1]

1.1.1 Блокон шифрлэлт

Хэрвээ эх ба шифрлэгдсэн тексүүдийн огторгуй нь ямар нэг $\sum^n$ олонлог байвал тухайн криптографыг блокон шифрлэлт гэнэ. Блокон шифрлэлтэд өгсөн мэдээг тэнцүү *n* урттай хэсгүүдэд хуваан шифрлэдэг.[2]

Блок шифрт энгийн текстийн блокийг бүхэлд нь авч, шифрлэгдсэн текстийн блокийг үүсгэхэд ашигладаг. Блокийн хэмжээг ерөнхийдөө шифрийн алгоритмаар тодорхойлно. Ихэнх блок шифрүүдийн хувьд энэ нь ихэвчлэн 64 эсвэл 128 бит байдаг ба зарим тохиолдолд нууцлалыг нэмэх зорилгоор 256, 512 бит ч байж болдог.

Хоёр төрлийн алгоритм ашиглах ба нэг нь шифр хийхэд нөгөө нь тайлахад ашиглагддаг. Эдгээр нь *n* урттай бит болон *k* бит урттай түлхүүрийг авч *n* бит урттай блок үүсгэнэ. $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. Тайлах алгоритм *D*-г нууцлах функцийн урвуу гэж тодорхойлж

болно.

$D : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$

$\forall k \in \{0,1\}^k, \forall m \in \{0,1\}^n, D(k, E(k,m)) = m$

[3]

1.1.2 Урсгалын шифрлэлт

Урсгалын шифрлэлт гэдэг нь өгөгдлийг урсгал маягаар нэг дор нэг битийг Криптографын алгоритм болон түлхүүрээ ашиглан шифрлэх арга юм. Урсгалын шифрийн давуу тал нь блок шифрлэлтээс харьцангуй хурдан ажиллахаас гадна, хэрэгжүүлэлтэд бага код ордог билээ. Гэсэн хэдий ч орчин үед түгээмэл ашиглагдахаа больсон ба элдэв халдагад түгээмэл өртдөг нь үүнтэй холбоотой. Жишээ нь RC4 гэх Урсгалын шифрлэлтийн алгоритм нь WEB болон WPA хамгаалалтад ашиглагддаг байсан хэдий ч хангалттай сайн хамгаалалт болж чадахгүй байгаа тул, хэрэглээнээс халагдаж байна.

1.2 Өгөгдөл шифрлэлтийн стандарт

1.2.1 DES алгоритм

DES (Data Encryption Standard) нь 1970-аад онд хөгжүүлэгдсэн тэгш хэмт блок шифрлэлтийн алгоритм юм. DES нь 64 бит урттай блок дээр ажиллах ба үүнийг 32-бит урттай хоёр хэсэг L_0, R_0 болгон хувааж, баруун талын 32-бит урттай хэсгийг олон янзын аргаар хувиргаж эцэст нь L_0-тэй XOR үйлдэл хийнэ. Арван зургаан үе хувиргалтын дараагаар L_0, R_0 нийлүүлж 64 бит шифрлэгдсэн блокийг үүсгэнэ.

Шинжүүд

1. Түлхүүрийн урт: DES нь 56 битийн түлхүүрийг ашигладаг бөгөөд анхандаа хангалттай аюулгүй байдлыг хангадаг гэж бодож байсан ч одоо Brute Force халдлагад маш эмзэгт тооцогддог.

2. Symmetric Encryption: DES нь шифрлэлт болон шифрийг тайлахад ижил түлхүүр ашигладаг. Тиймээс түлхүүрийг илгээгч, хүлээн авагч хоёулаа мэдэж, нууцлах ёстой.

3. Блок шифр: DES нь тусдаа бит биш харин өгөгдлийн блокууд дээр ажилладаг. Энэ нь их хэмжээний өгөгдлийг шифрлэх шаардлагатай программуудад тохиромжтой.

4. DES үйлдлүүд: DES нь Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) зэрэг хэд хэдэн үйлдлийн горимыг дэмждэг.

5. DES нь детерминистик: ижил текст болон ижил түлхүүрийн хувьд шифрлэгдсэн текст үргэлж ижил байх болно.

хэдийгээр 3-DES гэж байдаг хэдий ч энэ нь тооцоолол ихээр шаарддаг тул цаашид ашиглагдах нь зогссон.

1.2.2 AES

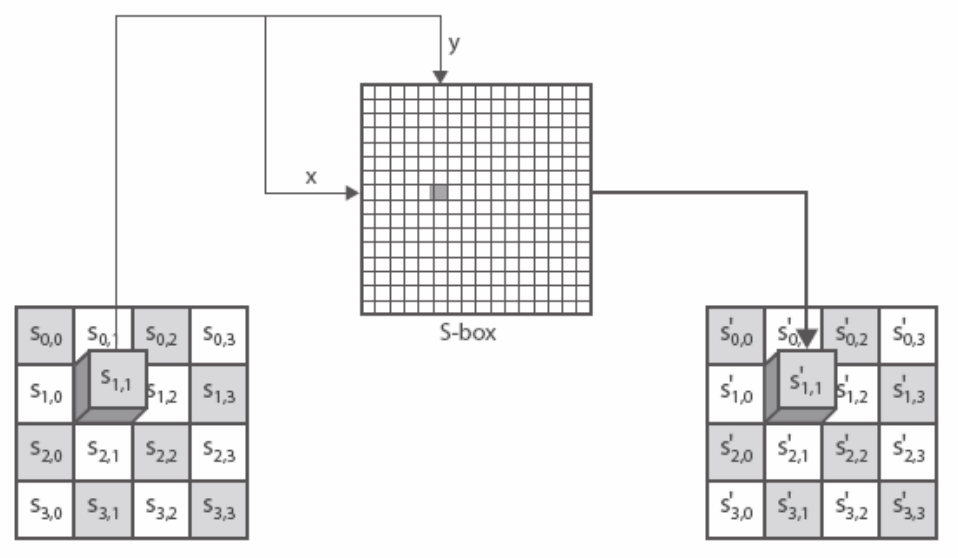
АНУ-ын Стандарт, Технологийн үндэсний хүрээлэн (VIST) 1997 онд өгөгдөл нууцлалын стандарт (DES)-ыг сайжруулах ажлыг эхлүүлж 2001 онд В.Рижмень, Д.Дэймен нарын блокон шифрлэлтийн схемийг дэвшилтэт нууцлалын стандартаар зарласан.[2]

AES нь орлуулах сэлгэлт (substitution-permutation) гэж нэрлэгддэг зарчим дээр суурилдаг бөгөөд программ хангамж болон техник хангамжийн аль алин дээр нь хурдан ажилдаг. Орчин үед шифрлэлтийг хурдан хийх зорилгоор техник хангамж дээр зөвхөн энэ алгоритмд зориулсан хэсэг хүртэл байдаг билээ.

Үндсэн үйлдэл

1. **SubBytes:**

    - Байт болгоны байрлалыг солино

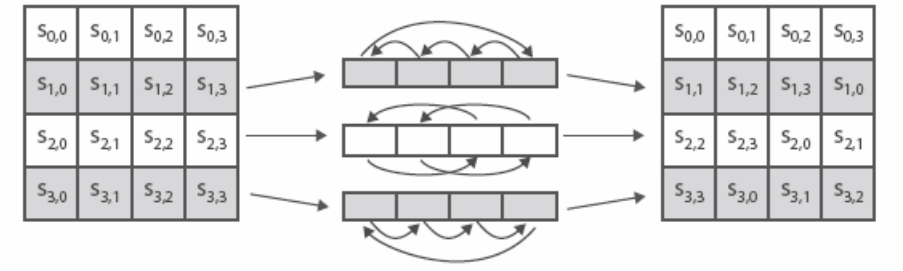


Зураг 1.1: SubBytes үйлдэл

- Тухайн мөр баганын мэдээлэл солигдоно

2. **ShiftRows:**

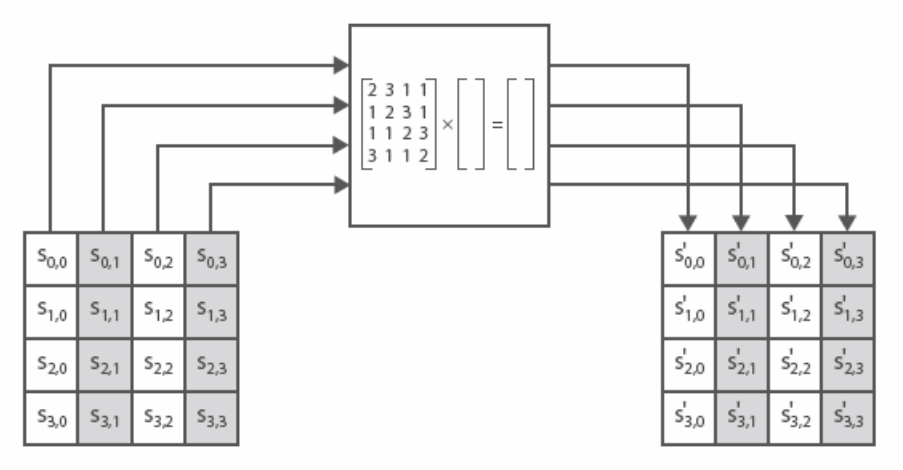
- 1-р мөрийг шилжүүлэхгүй
- 2–р мөрийн байтуудыг зүүн тийш 1 байт шилжүүлнэ
- 3–р мөрийн байтуудыг зүүн тийш 2 байт шилжүүлнэ
- 4–р мөрийн байтуудыг зүүн тийш 3 байт шилжүүлнэ
- Тайлах үйлдлийг хийхдээ баруун тийш шилжүүлэх үйлдлийг хийнэ



Зураг 1.2: ShiftRows үйлдэл

3. **MixColumns:**

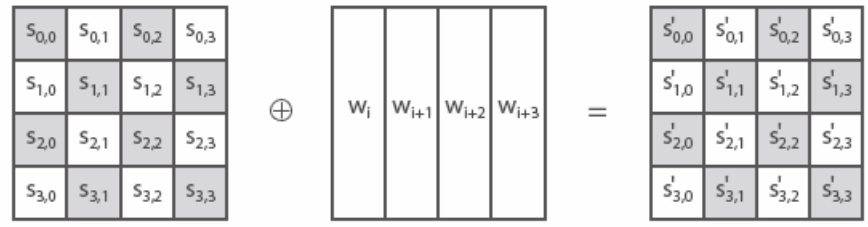
   - Багана бүр тус тусдаа холигдоно
   - Багана болгоны харгалзаа байтууд хоорондоо солигдоно



Зураг 1.3: MixColumns үйлдэл

4. **AddRoundKey:**

   - 128 бит XOR үйлдлийг циклийн түлхүүрт ашиглана
   - Тайлах үйлдэл хийх бол эсрэгээр гүйцэтгэнэ



Зураг 1.4: AddRoundKey үйлдэл

AES-ын нууцлалт

1. шифрлэх блок ба түлхүүрийн урт, мөчлөгийн тоог сонгох. Шифрлэх блок ба түлхүүрийн урт нь 128, 192, 256 байт байж болох бөгөөд мөчлөгийн тоо нь харгалзан 10, 12, 14 байна.

2. Шифрлэх текст, түлхүүрийн матриц *T, W, K*-г үүсгэнэ.

3. Эцсийн мөчлөгөөс бусад мөчлөгийн *T, W, K* матрицуудад **AES**-н үндсэн үйлдлүүдийг дэс дараалан хийнэ. Харин эцсийн мөчлөгт Mix Columns үйлдлийг хийхгүй.

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}$$

1.2.3 RSA

RSA нь анхны тооны өвөрмөц шинж чанарыг ашигладаг тэгш бус хэмтэй шифрлэлтийн арга юм. Анх 1977 онд танилцуулагдсан ба, өнөөг хүртэл хэрэглээнд хэвээр байгаа. Өнөөдрийн дэлхий даяар мөрдөгдөж байгаа стандарт нь хоёр анхны тооны үржвэр болох модулус нь 2048 бит хэмжээтэй байх ёстой. Энэ нь 617 оронтой тоо байна гэсэн үг юм.

- Хоёр анхны тоо болох p болон q сонгоно.
- $n = p * q$ утгыг олно.
- $\phi(n) = (p-1) * (q-1)$ утгыг олно.
- Дараах нөхцөлийг хангах e тоог сонгоно $1 < e < \phi(n)$ ба хиех$(e, \phi(n)) = 1$.
- d нь $d \equiv e^{-1} \mod \phi(n)$ гэж тодорхойлогдоно.

Нийтийн түлхүүр нь (e, n) болох ба хувийн түлхүүр нь (d, n) болно.[5]

2. СИСТЕМИЙН ЗОХИОМЖ

Орчин үеийн үүлийн технологид суурилсан тоон гарын үсгийн систем хийх болсонтойгоор холбоотой хийсэн судалгаанууд.

2.1 Тоон гарын үсгийн стандарт

Хэдийгээр бүх цахим гарын үсэг нь DSS-ийн дүрмийг дагаж мөрдөх ёстой боловч тэдгээр нь бүгд адилхан биш юм. Баримт бичигт гарын үсэг зурахад ашиглаж болох гурван төрлийн тоон гарын үсгийн стандарт байдаг.

1. **Энгийн цахим гарын үсэг (SES) -** Цахим гарын үсгийн хамгийн үндсэн хэлбэр. SES нь баримт бичигт нэмэхэд хурдан бөгөөд хялбар боловч шифрлэлтийн аргаар хамгаалагдаагүй. Өөрөөр хэлбэл, тийм ч аюулгүй биш юм. Үүнд жишээ нь цахим шуудангийн гарын үсэг ордог.

2. **Нарийвчилсан цахим гарын үсэг (AES) -** Хэдийгээр хууль ёсны дагуу хүчингүй боловч AES (Advanced Electronic Signature) нь гарын үсэг зурсны дараа баримт бичигт өөрлчлөлт орсон эсэхийг мэдэх боломжтой крифтографыг ашигладаг. Гэсэн хэдий ч хуулийн дагуу хүчингүй хэвээр.

3. **Qualified advanced electronic signature (QES) -** Цахим хэлбэрээр гарын үсэг зурах хамгийн найдвартай арга. Тоон гарын үсэг гэж нэрлэгддэг шаардлага хангасан цахим гарын үсэг нь аюулгүй байдлын дээд түвшинг хангахын тулд нийтийн түлхүүрийн дэд бүтэц, тэгш бус криптограф, Two Factor баталгаажуулалтыг ашигладаг. Эдгээрийг ашигласнаар, гарын үсэг нь хууль ёсны дагуу хүчийн төгөлдөр болно.

2.2 Адил системийн судалгаа

Tridumkey.mn

Tridimkey нь Монгол улсын бүртгэлийн ерөнхий газраар хүлээн зөвшөөрөгдсөн тоон гарын үсэг олгогч ба байгууллагад зориулж гарын үсэг олгодог нь онцлог санагдсан. Байгууллагад зориулж гарын үсэг авахад бүрдүүлдэг баримтууд.

1. Иргэний үнэмлэх эх хувь эсвэл И-монголиа-ийн иргэний үнэмлэхийн лавлагаа
2. Байгууллагын гэрчилгээ
3. Албан бичиг эх хувь Загвар татах
4. Эзэмшигч өөрийн биеээр ирэх боломжгүй үед итгэмжлэлтэй албан бичиг
5. Анкет

Гэвч сул тал нь энэхүү тоон гарын үсгийн систем нь зөвхөн **Windows** үйлдлийн систем дээр ажилдаг ба Macos эсвэл Linux үйлдлийн систем ашигладаг хэрэглэгчид ашиглах боломжгүй болж байгаа юм.

Monpass.mn

”Таньж баталгаажуулах тоон гарын үсгийн гэрчилгээ: Цахим бизнес, төрийн болон бусад төрөл бүрийн систем, онлайн үйлчилгээнд хандах, бусад цахим гүйлгээ, хэлцэл хийхэд найдвартай таньж баталгаажуулах, захидал харилцааг хөдөлбөргүй баталгаажуулахын тулд тоон гарын үсэг зурах, захидал харилцаа, дамжуулж буй баримт бичгийг шифрлэн дамжуулах, ажилтнууд, хэрэглэгчдийг хялбар таних, бөөний онлайн худалдаа зохион байгуулах гэх мэт зорилгоор ашиглагддаг тоон гарын үсгийн гэрчилгээ – цахим баримт бичиг юм. Энэ гэрчилгээ нь хэрэглэгчийн мэдээлэл, олгосон ГОБ-ын мэдээлэл, хосгүй серийн дугаар болон бусад хосгүй өгөгдлүүд, хүчинтэй хугацаа, тоон гарын үсгийн нийтийн түлхүүр, холбогдох бусад мэдээллийг агуулсан

байх бөгөөд Хувь хүмүүс болон байгууллагын төлөөлөгч хэн боловч ашиглаж болно. Захидал, мэдээлэлдээ тоон гарын үсэг зурахдаа өөрийн тоон гарын үсгийн хувийн түлхүүрийг ашиглах ба харин шифрлэн илгээх бол хүлээн авагчийн нийтийн түлхүүрийг ашиглана.” гэсэн танилцуулагатай байсан ба гүнзгий судалж үзэхэд мөн л хэрэглэгчийн үйлдлийн систем зөвхөн **Windows** байж л тоон гарын үсгийн ашиглах боломжтой байсан юм.

2.3 Системийн шаардлага

Функциональ шаардлага

Table 2.1: Функциональ шаардлага

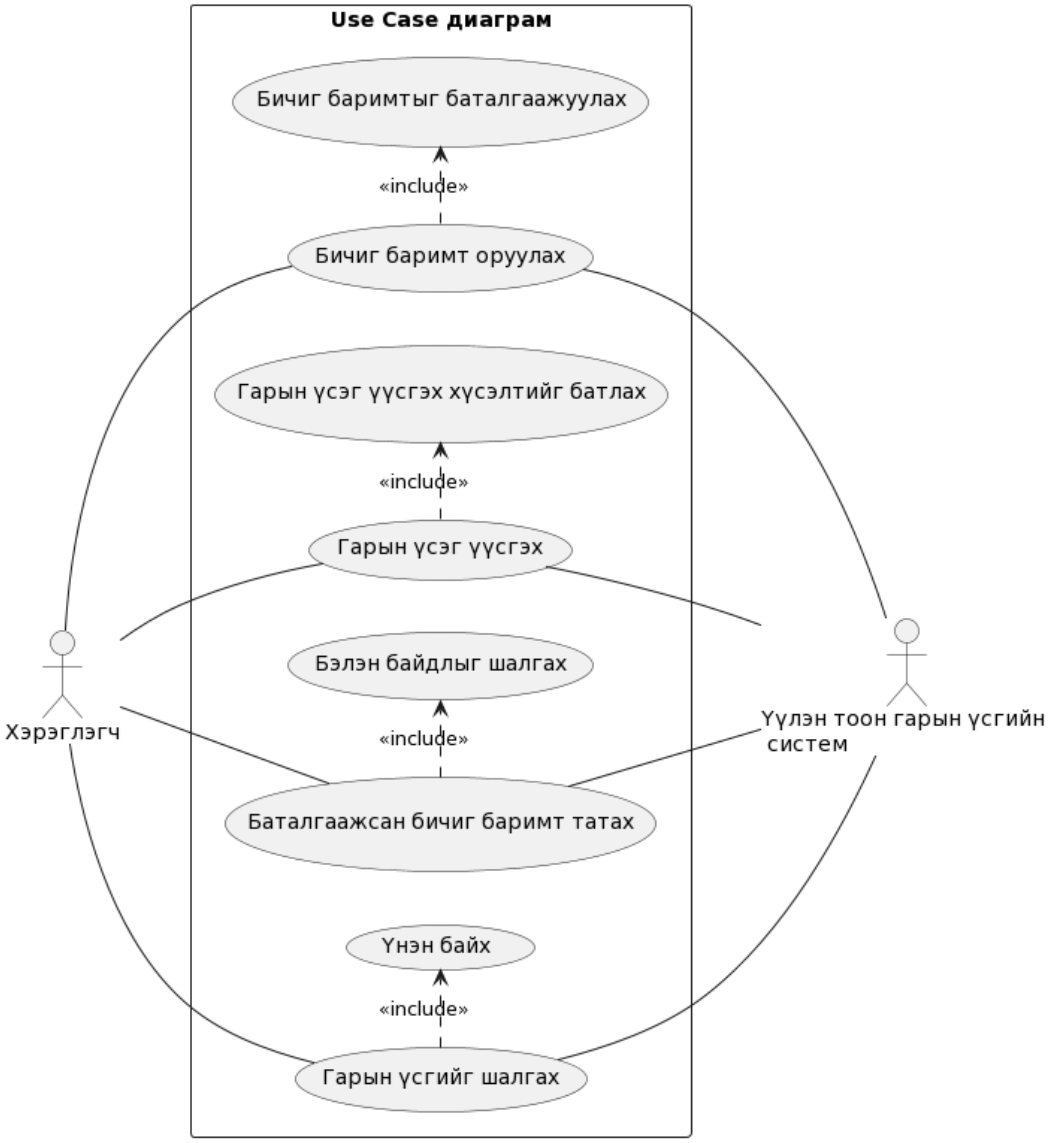
ФШ 100	Систем нь хэрэглэгчийн тоон гарын үсэг үүсгэх чадвартай байх ёстой. Үүнд хэрэглэгч бүрийн өвөрмөц түлхүүрийн хослолыг бий болгох орно.
ФШ 200	Систем нь тоон гарын үсгийг баталгаажуулах функцээр хангах ёстой. Энэ нь гарын үсэг зурсан баримт бичгийг хүлээн авч, гарын үсэг зурсан хүний нийтийн түлхүүрийг ашиглан гарын үсгийг баталгаажуулах ёстой.
ФШ 300	Систем нь хэрэглэгчдэд гарын үсэг зурахын тулд янз бүрийн форматтай цахим баримт бичгүүдийг (жишээлбэл, .doc, .pdf, .xls гэх мэт) байршуулахыг зөвшөөрөх ёстой.
ФШ 400	Систем нь хэрэглэгчдийг баримт бичигт гарын үсэг зурах, баталгаажуулахаас өмнө баталгаажуулах ёстой. Үүнийг хэрэглэгчийн нэр/нууц үг, олон хүчин зүйлийн баталгаажуулалт эсвэл бусад аюулгүй аргуудаар хийж болно.
ФШ 500	Систем нь баримт бичиг байршуулах, гарын үсэг үүсгэх, гарын үсгийн баталгаажуулалт зэрэг хэрэглэгчдийн хийсэн бүх үйлдлийг бүртгэх ёстой.
ФШ 600	Систем нь бусад үйлчилгээтэй нэгтгэх API-г өгөх ёстой. Энэ нь бусад програм хангамж эсвэл үйлчилгээнд энэ үйлчилгээний тоон гарын үсгийн чадварыг ашиглах боломжийг олгоно.
ФШ 700	Веб нь хэрэглэгч бүртгэх боломжтой байх

Функциональ бус шаардлага

Table 2.2: Функциональ бус шаардлага

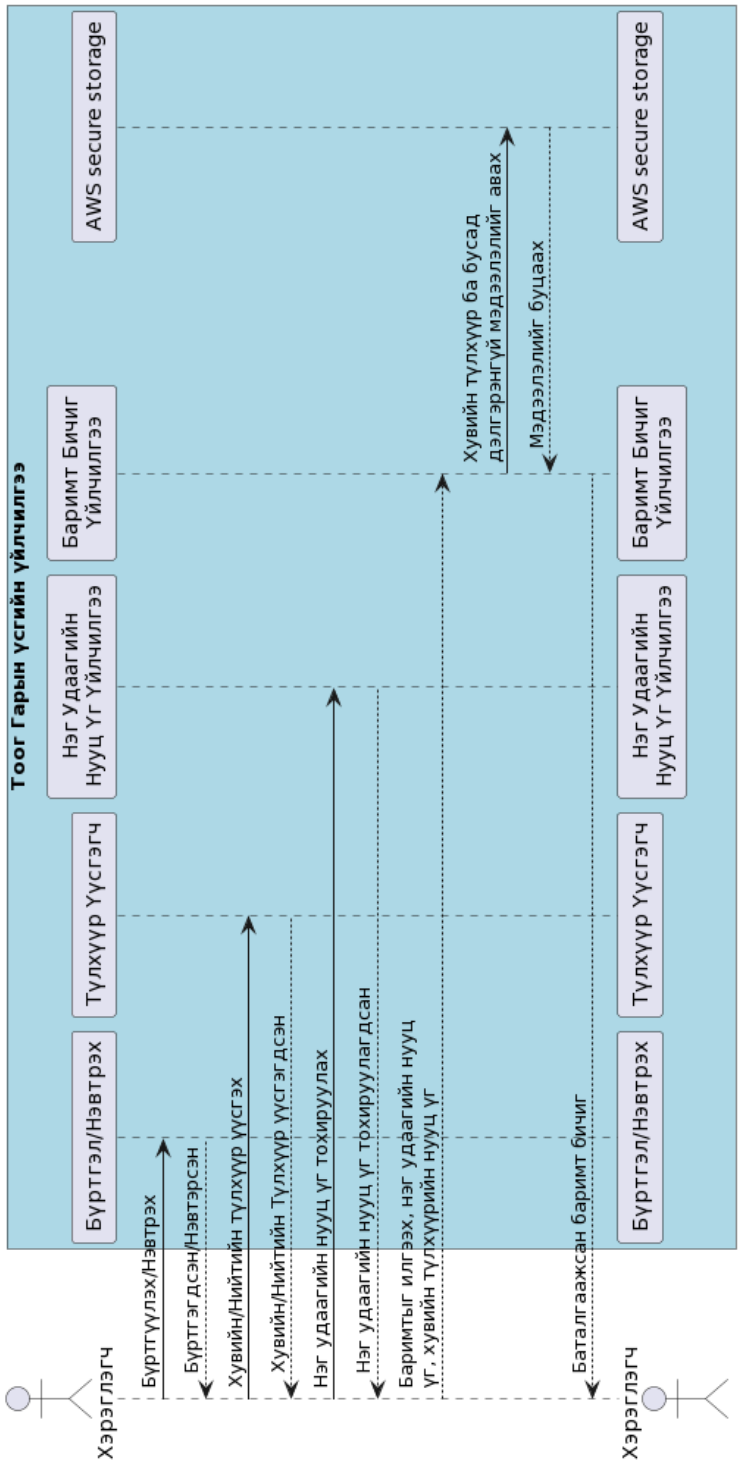
ФБШ 100	Систем нь GDPR эсвэл HIPAA гэх мэт холбогдох бүх мэдээллийн аюулгүй байдал, нууцлалын дүрэм журмыг дагаж мөрдөх ёстой. Гарын үсэг, баримт бичиг зэрэг бүх өгөгдөл шифрлэгдсэн байх ёстой.
ФБШ 200	Систем нь гүйцэтгэлийн бууралтгүйгээр олон тооны хэрэглэгчид болон баримт бичгүүдийг зохицуулах чадвартай байх ёстой.
ФБШ 300	Үүлэн үйлчилгээ нь хамгийн бага зогсолттой, 24/7 цагийн турш ашиглах боломжтой байх ёстой. Үйлчилгээний түвшний гэрээ (SLA) нь дор хаяж 99.9% ажиллах хугацааг баталгаажуулах ёстой.
ФБШ 400	Систем нь хүлээн зөвшөөрөгдсөн тодорхой хугацааны дотор гарын үсэг үүсгэх, баталгаажуулах хүсэлтийг хурдан боловсруулах чадвартай байх ёстой.
ФБШ 500	Систем нь янз бүрийн техникийн чадвартай хэрэглэгчдэд үүнийг үр дүнтэй ашиглах боломжийг олгодог хэрэглэгчдэд ээлтэй интерфэйстэй байх ёстой.
ФБШ 600	Үүлэн үйлчилгээ нь янз бүрийн үйлдлийн систем, хөтөч, төхөөрөмжтэй нийцтэй байх ёстой.
ФБШ 700	Энэ систем нь гамшгийн үед өгөгдөл алдагдахгүй байхын тулд найдвартай нөөцлөх, сэргээх механизмтай байх ёстой.
ФБШ 800	Систем нь Европ дахь eIDAS эсвэл АНУ-ын ESIGN хууль зэрэг тоон гарын үсгийн хууль тогтоомж, дүрэм журамд нийцсэн байх ёстой.

2.4 Use case диаграм



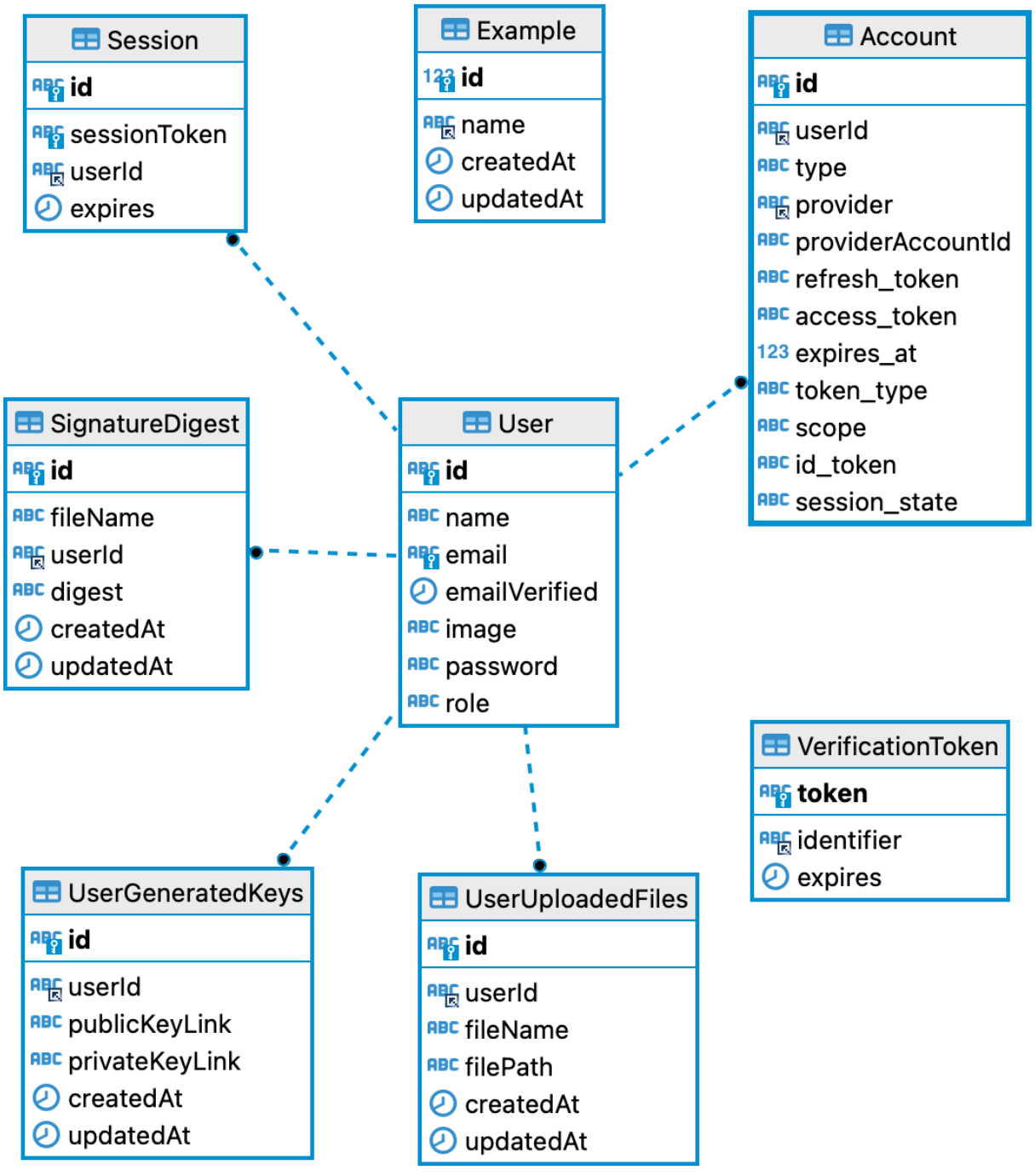
Зураг 2.1: Use case диаграм

2.5 Sequence диаграм

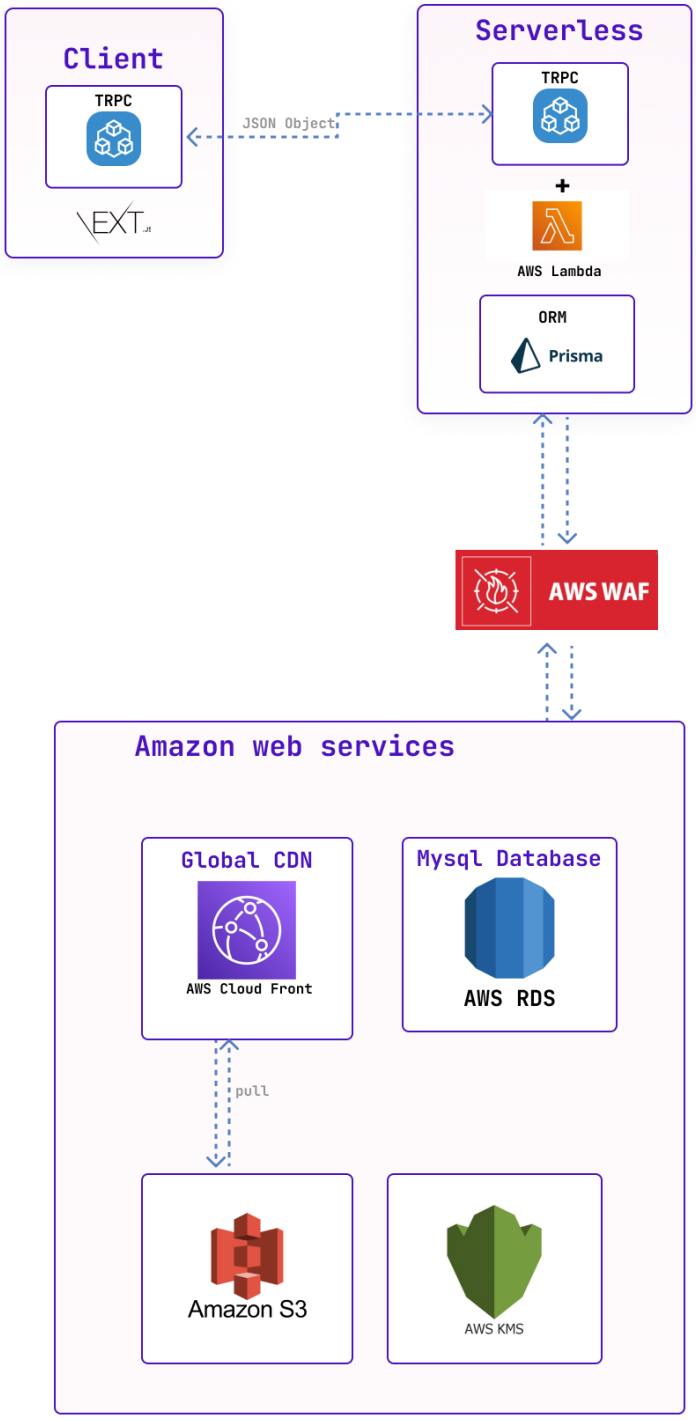


Зураг 2.2: Sequence диаграм

2.6 ER диаграм

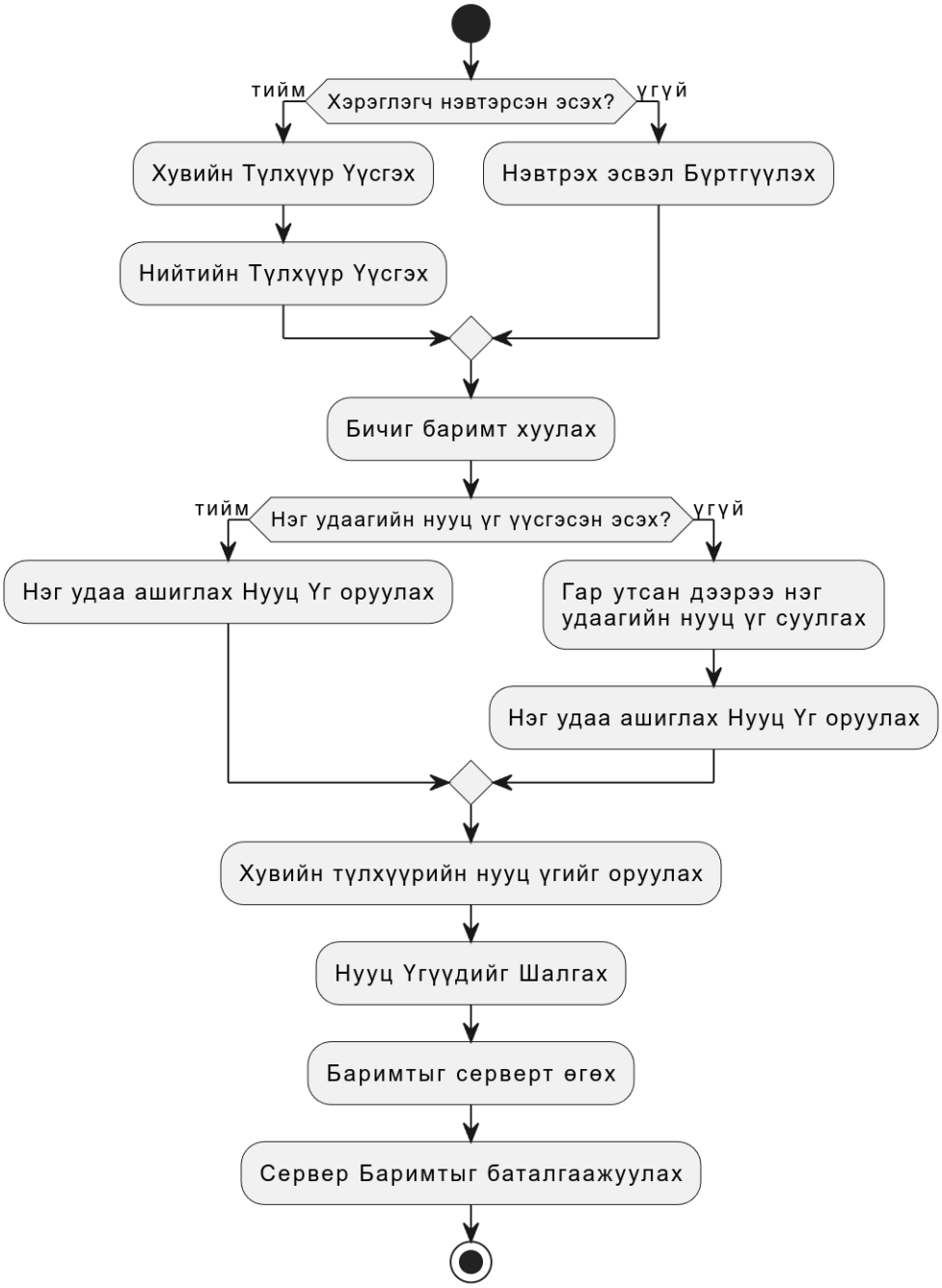


Зураг 2.3: Датабаз диаграм



Зураг 2.4: Архитектур

2.7 Гарын үсэг зурах үйл ажиллагааны диаграм



Зураг 2.5: Гарын үсэг зурах үйл ажиллагааны диаграм

3. ХЭРЭГЖҮҮЛЭЛТ

3.1 Сонгосон технологи

3.1.1 Nextjs & Reactjs

Declarative

React нь хэрэглэгчийн интерактив интерфейс бүтээхийг хялбарчилдаг. Аппликейшны state бүрд зориулсан энгийн бүтэц зохион байгуулахаас гадна, React нь өгөгдөл өөрчлөгдөхөд яг зөв компонентоо өөрчлөн рендер хийдэг. Declarative бүтэц нь кодыг тань debug хийхэд хялбар болгохоос гадна, ажиллагаа нь илүү тодорхой болдог

Компонент-д тулгуурласан

Бие даан state-ээ удирддаг маш энгийн компонент бичиж, эдгээрийг хольж найруулан нарийн бүтэцтэй хэрэглэгчийн интерфейс бүтээ.

Компонентийн логик нь тэмплэйт-ээр бус JavaScript-ээр бичигддэг учраас өгөгдлийг апп хооронд хялбар дамжуулж, DOM-оос state-ээ тусд нь байлгаж чадна.

Nextjs

Netflix, TikTok, Hulu, Twitch, Nike гэсэн орчин үеийн аваргууд ашигладаг энэхүү орчин үеийн фрэймворк нь React технологи дээр үндэслэгдсэн бөгөөд Frontend Backend хоёр талд хоёуланд нь ажилладаг веб аппуудыг хийх чадвартайгаараа бусдаасаа давуу юм. Next.js -ийн үндсэн дизайн нь клиент болон сервер талын аль алиных давуу талыг ашиглаж чаддаг, ямар нэг дутагдалгүй веб сайтыг яаж хамгийн хурдан хялбар бүтээх вэ гэдгийг бодож тусгасан байдаг. Next.js нь сервер талд react компонентуудыг рендерлэн энгийн html, css, json файл болгон хувиргах замаар ажилладаг бөгөөд 2020 оноос олон нийтэд танигдсан JAMStack технологи

болон статик сайт, автоматаар статик хуудас үүсгэх, CDN deployment, сервергүй функц, тэг тохиргоо, файлын системийн рүүтинг (PHP-ээс санаа авсан), SWR (stale while revalidate), сервер талд рендерлэх зэрэг асар олон орчин үеийн шинэхэн технологиудыг бүгдийг хийж чаддаг анхны бүрэн веб фрэймворк гэж хэлж болно.[4]

3.1.2 tRPC (Back End)

Энгийнээр хэлбэл, tRPC нь клиент болон сервер хоорондоо сүлжээгээр харилцаж болох API (Application Programming Interfaces) бүтээх хэрэгсэл юм. Энэ нь хувьсагчийн төрлүүдийг нягт зааж өгч Front-End Back-End хоёрийг холбож ажилладаг. Жишээ нь хэрвээ сервер тал дээр ажиллаж байгаа хөгжүүлэгч, функцын параметр солиход энгийн REST api эсвэл Graphql түүнийг мэдэж чадахгүй юм. Харин tRPC нь шууд алдаа болж харагдах ба хөгжүүлэлтийн орчинд Back-end Front-end хоёр холбогдож ажилдаг гэдгээрээ давуу юм. Ингэснээр хөгжүүлэхэд илүү хялбар, инжинерт илүү ээлтэй болдог билээ. Кодыг А.2

3.1.3 AWS S3 объект агуулах

AWS S3 (Amazon Simple Storage Service) нь Amazon Web Services (AWS) дээрх өгөгдөл, мэдээллийг онлайнаар нөөцлөх, архивлахад зориулагдсан хязгааргүй өргөтгөх боломжтой, өндөр хурдтай, вэб технологид суурилсан үүлэн хадгалах үйлчилгээ юм. Энэ нь вэбийн хаанаас ч хүссэн үедээ ямар ч хэмжээний өгөгдлийг хадгалах, сэргээхэд ашиглаж болно.

3.1.4 AWS KMS

AWS Түлхүүр Удирдлагын Үйлчилгээ (KMS) нь криптографын түлхүүрүүдийг үүсгэх, хянахад хялбар болгодог. Мөн түүнчлэн түлхүүр нь ашиглагдаагүй хадгалагдаж байх үедээ шифрлэгдсэн байдаг. Энэхүү үйлчилгээ нь бусад AWS үйлчилгээнүүдтэй нэгтгэгдсэн тул эдгээр үйлчилгээнд хадгалсан өгөгдлийг шифрлэх, кодыг тайлах түлхүүрүүдэд хандах хандалтыг хянахад хялбар болгодог.

3.1.5 Dockerizing

Орчин үеийн нэгэн гайхалтай технологи бол контайнерчлах юм. Яагаад Docker чухал вэ гэвэл, ямар нэгэн систем хөгжүүлэгчийн компьютер аль эсвэл ямар сервер дээр ажиллаж байгаагаас үл хамааран програм нь өөрийн тусдаа орчинд ажиллах юм. Яг л Virtual machine шиг гэхдээ давуу тал нь Docker host system-ийнхээ цөмийг (kernel)-г ашигладаг учраас маш бага хэмжээгий зай, нөөц ашигладаг.

3.1.6 CI/CD

Мөн сүүлийн үед маш их өргөн түгж байгаа ойлголт бол Continuous Integration/Continuous Deployment. Энэ нь програм хангамж ямар ч нөхцөлд хөгжүүлэлт тасралтгүй явж байх орчноор хангадаг ба системд хэзээ ч тасалдал үүсгэхгүй мөн хүний оролцоог маш бага байлгах давуу талтай.[1] .

3.2 Ажиллагаа

3.2.1 Гарын үсэг зурах

Сервер талд ажиллах

1. Хэрэглэгчийн оруулсан файлыг объект агуулахаас (AWS S3) татаж авах.
2. Файлын бинари (binary) хэсгийг SHA256 алгоритм ашиглан хайш утгыг тооцоолох.
3. Хэрэглэгчийн хувийн түлхүүрийг аюулгүй хадгалах орчноос авах (AWS KMS).
4. Хувийн түлхүүрийг ашиглах хайш утгыг шифрлэх.
5. Шифрлэгдсэн утгыг өгөгдлийг сан руу хадгалах.

[1] Дадлагын ажлаасаа иш татав. https://github.com/b4ljk/internship-report

6. Шифрлэгдсэн утга буюу гарын үсгийг олон улсын стандартын дагуу PDF файл руу нэмэх.

7. Шинээр үүссэн буюу шифрлэгдсэн файлыг объект агуулах руу хуулах.

8. Нэг удаагийн татаж авах холбоосыг хэрэглэгчид өгөх.

Хэрэглэгч талд ажиллах

1. Хэрэглэгчийн өөрийн файлыг оруулах.

2. Файлын бинари (binary) хэсгийг SHA256 алгоритм ашиглан хайш утгыг тооцоолох.

3. Хэрэглэгч хувийн түлхүүрээ оруулах.

4. Хувийн түлхүүрийг ашиглах хайш утгыг шифрлэх.

5. Шифрлэгдсэн утгыг сервер рүү илгээх.

6. Шифрлэгдсэн утга буюу гарын үсгийг олон улсын стандартын дагуу PDF файл руу нэмэх.

7. Хэрэглэгч талд гарын үсэг зурсан файл үүсэх.

3.3 Хөгжүүлэлт

Өгөгдлийн сангийн зохион байгуулалт

Призма нь өгөгдлийн сан болон, код баз хоёрын хялбараар холбоход тусладаг. Үүнийг ORM гэж нэрлэдэг ба давуу тал нь, өгөгдлийг ариутгах, өгөгдлийг сангийн зохион байгуулалт түүхийг хадгалах зэрэг ажлыг инжинер хийх шаардлаггүй болох юм.

```
generator client {
    provider = "prisma-client-js"
}
```

```
4
5  datasource db {
6      provider     = "mysql"
7      url          = env("DATABASE_URL")
8      relationMode = "prisma"
9  }
10
11 model Example {
12     id         Int      @id @default(autoincrement())
13     name       String
14     createdAt  DateTime @default(now())
15     updatedAt  DateTime @updatedAt
16
17     @@index([name])
18 }
19
20 // Necessary for Next auth
21 model Account {
22     id                String  @id @default(cuid())
23     userId            String
24     type              String
25     provider          String
26     providerAccountId String
27     refresh_token     String? @db.Text
28     access_token      String? @db.Text
29     expires_at        Int?
30     ...
```

Код 3.1: Prisma Датабаазын модел

AWS

Амазоны санал болгодог үйлчилгээнүүдийг өөртөө тохирхийг нь ашигласнаар заавал өөрийн серверийг ажлуулах шаардлагүй болно. Мэдээж ашиглахийн тулд AWS дээрээ тохиргоонуудыг хийх ба нууцлалын мэдээллүүдээ код дундаа оруулж үүнийгээ ашиглах юм.

Жишээ нь хэрэглэгчийн оруулсан файлыг 3 хоногийн дараа устана гэсэн тохиргоог AWS дээр хийж өгсөн байгаа.

```
import aws from "aws-sdk";

aws.config.update({
  accessKeyId: process.env.S3_ACCESS_KEY,
  secretAccessKey: process.env.S3_SECRET,
  region: process.env.AWS_REGION,
});

export const s3 = new aws.S3();

export default aws;
```

Код 3.2: AWS нууцлалын хэсэг

```
import { type PresignedPost } from "aws-sdk/clients/s3";
import { s3 } from "~/utils/aws";

export const uploadToSignedUrl = async ({
  signedUploadUrl,
  file,
  setUploadProgress,
  index,
}: {
```

```
  signedUploadUrl: PresignedPost;
  file: File;
  setUploadProgress: React.Dispatch<React.SetStateAction<number[]>>;
  index: number;
}): Promise<void> => {
  return new Promise((resolve, reject) => {
    const formData = new FormData();
    Object.keys(signedUploadUrl.fields).forEach((key) =>
      formData.append(key, signedUploadUrl.fields[key]!),
    );
    formData.append("file", file);

    const xhr = new XMLHttpRequest();
    xhr.open("POST", signedUploadUrl.url, true);

    xhr.upload.addEventListener("progress", (event) => {
      if (event.lengthComputable) {
        const percentComplete = (event.loaded / event.total) * 100;
        console.log(`Upload is ${percentComplete}% done.`);
        setUploadProgress((prev) => {
          const newProgress = [...prev];
          newProgress[index] = percentComplete;
          return newProgress;
        });
      }
    });

    xhr.onload = () => {
```

```
      if (xhr.status > 199 && xhr.status < 300) {
        resolve();
      } else {
        reject(`Error: ${xhr.status}`);
      }
    };

    xhr.onerror = () => {
      reject(`Error: ${xhr.status}`);
    };

    xhr.send(formData);
  });
};
```

Код 3.3: Файл серверлүү урсгалаар илгээх

Хэрэглэгчийн хэсгийн хөгжүүлэлт (Front-end)

Энэ хэсэгт хэрэглэгчийн сервертэй харьцах API хэсэг хийгдсэн ба tRPC нь хэрэглэгчийн талаас серверлүү хүсэлт илгээхдээ хүүк бичих байдлаар ажилдаг. Доор оруулсан код нь API-тэй холбоотойгоор ямар нэгэн алдаа гарвал вэб аппликейшныг тэр чигт нь унагахгүйгээр ямар ч алдааг хэрэглэгчид ойлгомжтой мессеж болгож харуулах код.

```
const queryClient = new QueryClient({
  queryCache: new QueryCache({
    onError: (err) => {
      toast.error(getError(err));
    },
  }),
```

```
  mutationCache: new MutationCache({
    onError: (error) => {
      toast.error(getError(error));
      if (error instanceof TRPCClientError) {
        const err = error.shape as TRPCErrorShape;
        if (err.code === TRPC_ERROR_CODES_BY_KEY.UNAUTHORIZED) {
          modalHandler.setModal(!modalHandler.isModalOpen);
        }
      }
    },
  }),
  defaultOptions: {
    queries: {
      refetchOnWindowFocus: false,
      retry: false,
    },
    mutations: {
      retry: false,
    },
  },
});
```

Код 3.4: Глобал алдааны мэдээллэгч

Сервер хэсгийн хөгжүүлэлт (Back-end)

Middleware нь кодыг эмх цэгцтэй байхад хэрэг болдог ба хэрэглэгчээс хүсэлт ирэхэд сервер хариу өгөхийн яг өмнөхөн ажилдаг хэсэг код билээ. Энэ хэсэгт хэрэглэгчийн мэдээллийг шалгах, нэвтэрсэн үгүйг тодорхойлох зэргийг хийхэд тохиромжтой байдаг.

```
const enforceUserIsAuthed = t.middleware(({ ctx, next }) => {
  if (!ctx.session?.user) {
    throw new TRPCError({
      code: "UNAUTHORIZED",
      message: "User is not logged in".toUpperCase(),
    });
  }
  return next({
    ctx: {
      // infers the `session` as non-nullable
      session: { ...ctx.session, user: ctx.session.user },
    },
  });
});
```

Код 3.5: Middleware

Бүх API нь нэгдсэн байдлаар нэг газар зангидагдаж байх ёстой. Миний хувьд tRPC дээрх бүх API-г root.ts гэдэг файл дотор нэгтгэж сервэрийн кодны үндэс болгож байгаа юм.

```
import { exampleRouter } from "~/server/api/routers/example";
import { createTRPCRouter } from "~/server/api/trpc";
import { authRouter } from "./routers/auth";
import { s3Router } from "./routers/s3";
import { secretKeyRoute } from "./routers/key";
import { signerRoute } from "./routers/signer";
import { otpRoute } from "./routers/otp";

export const appRouter = createTRPCRouter({
  auth_router: authRouter,
```

```
  s3_router: s3Router,
  key_router: secretKeyRoute,
  sign_router: signerRoute,
  otp_router: otpRoute,
});

export type AppRouter = typeof appRouter;
```

Код 3.6: Root

Дүгнэлт

Энэхүү судалгааны ажлаар дэлхий нийтэд ашиглагдаж буй криптографын зарим алгоритмуудыг судалж хэрэгжүүлсэн билээ. Энэхүү судалж суралцсан мэдлэгээ ашиглан практикт олон улсын стандартад нийцсэн үүлэн технологид суурилсан тоон гарын үсгийн системийн бүтээхийг зорилоо. Үр дүнд нь хамгийн орчин үеийн шинэлэг үүлэн технологиудтай танилцсан ба, бүтээгдэхүүний шаардлагыг гаргаж код баазыг үүсгэхээс эхлээд эцсийн хэрэглэгчид хүрэх, чанарын шаардлагыг хангаж ачаалал даахуйц системийг бүтээлээ.

Энэхүү систем нь үүлэн технологид суурилсан гэдгээрээ Монгол улсад анхдагч болж байгаа юм. Цаашлаад блокчейн технологийг ашиглан бүр ч илүү найдвартай, нийтэд нээлтэй систем болох боломжтой гэж харж байна.

Bibliography

[1] Daemen, J., & Rijmen, V. (2002). ”The Design of Rijndael: AES - The Advanced Encryption Standard.” Springer. p.1-2.

[2] Д. Гармаа (2022). ”Криптографын үндэс.” Улаанбаатар хот.

[3] Bellare, Mihir; Rogaway, Phillip (11 May 2005), Introduction to Modern Cryptography (Lecture notes), archived (PDF) from the original on 2023-10-30, chapter 3.

[4] ReactJS, `https://reactjs.org/`

[5] Simmons, G. J. (2022, December 29). RSA encryption. Encyclopedia Britannica. https://www.britannica.com/topic/RSA-encryption

А. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

```ts
import { initTRPC, TRPCError } from "@trpc/server";
import { type CreateNextContextOptions } from "@trpc/server/adapters/next";
import { type Session } from "next-auth";
import superjson from "superjson";
import { ZodError } from "zod";

import { getServerAuthSession } from "~/server/auth";
import { db } from "~/server/db";

interface CreateContextOptions {
  session: Session | null;
}

const createInnerTRPCContext = (opts: CreateContextOptions) => {
  return {
    session: opts.session,
    db,
  };
};

export const createTRPCContext = async (opts: CreateNextContextOptions) => {
  const { req, res } = opts;

  // Get the session from the server using the getServerSession wrapper function
  const session = await getServerAuthSession({ req, res });

  return createInnerTRPCContext({
    session,
  });
};

const t = initTRPC.context<typeof createTRPCContext>().create({
  transformer: superjson,
  errorFormatter({ shape, error }) {
    return {
      ...shape,
      data: {
        ...shape.data,
        zodError:
          error.cause instanceof ZodError ? error.cause.flatten() : null,
      },
    };
  },
});
```

```

export const createTRPCRouter = t.router;
export const publicProcedure = t.procedure;

const enforceUserIsAuthed = t.middleware(({ ctx, next }) => {
  if (!ctx.session?.user) {
    throw new TRPCError({
      code: "UNAUTHORIZED",
      message: "User is not logged in".toUpperCase(),
    });
  }
  return next({
    ctx: {
      // infers the `session` as non-nullable
      session: { ...ctx.session, user: ctx.session.user },
    },
  });
});

export const protectedProcedure = t.procedure.use(enforceUserIsAuthed);
```

Код A.1: tRPC тохиргоо

```
version: '3.1'

services:
  db:
    container_name: thesis_db
    image: mysql:latest
    restart: always
    environment:
      # MYSQL_USER: ${MYSQL_USER}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    volumes:
      - db_data:/var/lib/mysql
    ports:
      - "3306:3306"

volumes:
  db_data:
```

Код A.2: Docker Compose