

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)
Баклаврын 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

УДИРТГАЛ	1
Зорилго	1
Зорилт	1
Үндэслэл	2
1. ОНОЛЫН СУДАЛГАА	3
1.1 Тэгш хэмт криптограф	3
1.2 Өгөгдөл шифрлэлтийн стандарт	4
2. СИСТЕМИЙН ЗОХИОМЖ	18
2.1 Тоон гарын үсгийн стандарт	18
2.2 Адил системийн судалгаа	19
2.3 Системийн шаардлага	21
2.4 Use case диаграм	23
2.5 Sequence диаграм	24
2.6 ER диаграм	25
2.7 Үйл ажиллагааны диаграм	26
3. ХЭРЭГЖҮҮЛЭЛТ	33
3.1 Сонгосон технологи	33
3.2 Ажиллагаа	35
3.3 Хөгжүүлэлт	36
3.4 PCA (RSA) Хэрэгжүүлэлт	43
3.5 Үр дүн	45
ДҮГНЭЛТ	51
НОМ ЗҮЙ	52
ХАВСРАЛТ	53
А. КОДЫН ХЭРЭГЖҮҮЛЭЛТ	53

ЗУРГИЙН ЖАГСААЛТ

1.1	SubBytes үйлдэл	6
1.2	ShiftRows үйлдэл	6
1.3	MixColumns үйлдэл	7
1.4	AddRoundKey үйлдэл	7
1.5	Хугацааны ээдрээ	10
1.6	RSA ба ECC түлхүүрийн хэмжээнүүдийн харьцуулалт	11
1.7	8 бит өгөгдөл шифрлэлт	13
1.8	64 бит өгөгдөл шифрлэлт	13
1.9	256 бит өгөгдөл шифрлэлт	14
1.10	8 бит өгөгдөл шифрлэлт тайлалт	14
1.11	64 бит өгөгдөл шифрлэлт тайлалт	15
1.12	256 бит өгөгдөл шифрлэлт тайлалт	15
1.13	8 бит өгөгдөл хугацааны харьцуулалт	16
1.14	64 бит өгөгдөл хугацааны харьцуулалт	16
1.15	256 бит өгөгдөл хугацааны харьцуулалт	17
2.1	Use case диаграм	23
2.2	Sequence диаграм	24
2.3	Дататбаз диаграм	30
2.4	Архитектур	31
2.5	Гарын үсэг зурах үйл ажиллагааны диаграм	32
3.1	Нүүр хуудас	46
3.2	Нүүр хуудас, Шөнийн тохиргоо	47
3.3	Тоон гарын үсэг үүсгэх шаардлага	47
3.4	Хүчинтэй гарын үсэгтэй баримт	48
3.5	Хүчингүй гарын үсэгтэй баримт	48

3.6	Нийт баримтын жагсаалт	49
3.7	РСА (RSA) алгоритмын хэрэгжүүлэлт	49
3.8	Цагаас хамаарсан нууц үг тохируулах	50
3.9	Цагаас хамаарсан нууц үг тохируулсны дараа	50

ХҮСНЭГТИЙН ЖАГСААЛТ

1.1	Муйхар хүчний алгоритм ашиглан PCA (RSA) нууцлалыг эвдэх нь [6] .	9
1.2	Аюулгүй байдлын түвшин ба RSA болон ECC түлхүүрийн хэмжээг харьцуулах [8].....	11
1.3	8 бит өгөгдөл – шифрлэлт ба шифр тайлах хугацаа (Секундээр) [8]	12
1.4	64 бит өгөгдөл – шифрлэлт ба шифр тайлах хугацаа (Секундээр) [8] ...	12
1.5	256 бит өгөгдөл – шифрлэлт ба шифр тайлах хугацаа (Секундээр) [8] ..	12
2.1	Функциональ шаардлага	21
2.2	Функциональ бус шаардлага	22
2.3	User хүснэгт.....	25
2.4	Session хүснэгт	26
2.5	VerificationToken хүснэгт	26
2.6	Account хүснэгт	27
2.7	UserUploadedFiles хүснэгт	27
2.8	OtpSecret хүснэгт	28
2.9	SignatureDigest хүснэгт	28
2.10	UserGeneratedKeys хүснэгт.....	29

Кодын жагсаалт

3.1	Prisma Датабаазын модел	36
3.2	AWS нууцлалын хэсэг	38
3.3	Файл серверлүү урсгалаар илгээх	38
3.4	Глобал алдааны мэдээллэгч	40
3.5	Middleware	42
3.6	Root	42
3.7	RSA хэрэгжүүлэлт	43
3.8	Миллер-Рабины тест	44
A.1	tRPC тохиргоо	53
A.2	Docker Compose	54

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 (PCA (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 програмууд нь зөвхөн Windows үйлдлийн систем дээр ажиллахаар хийгдсэн нь нийцтэй байдлыг хангахгүй байна.

1. ОНОЛЫН СУДАЛГАА

1.1 Тэгш хэмт криптограф

Тэгш хэмт криптографт 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 Дамен, 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 Блок шифрлэлт

Хэрвээ эх ба шифрлэгдсэн тексүүдийн огторгуй нь ямар нэг \sum^n олонлог байвал тухайн криптографыг блок шифрлэлт гэнэ. Блок 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.

$E : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$. Тайлах алгоритм D -г нууцлах функцийн урвуу гэж тодорхойлж

болно.

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

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

[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

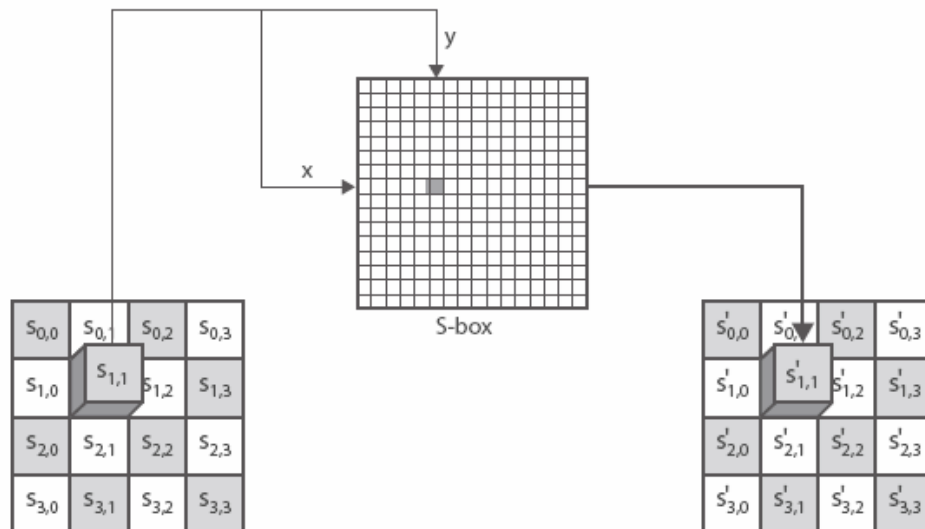
АНУ-ын Стандарт, Технологийн үндэсний хүрээлэн (VIST) 1997 онд өгөгдөл нууцлалын стандарт (DES)-ыг сайжруулах ажлыг эхлүүлж 2001 онд В.Рижмень, Д.Дэймен нарын блокон шифрлэлтийн схемийг дэвшилтэт нууцлалын стандартаар зарласан.[2]

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

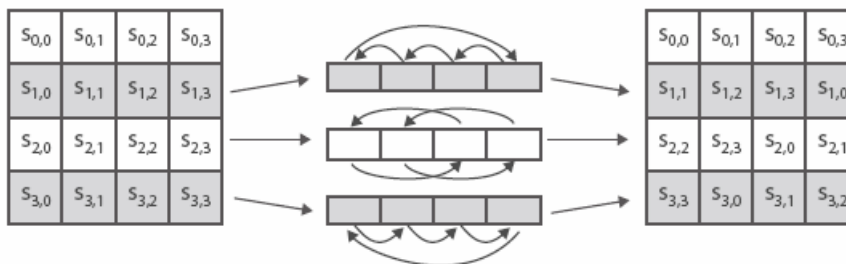


Зураг 1.1: SubBytes үйлдэл

- Тухайн мөр баганын мэдээлэл солигдоно

2. ShiftRows:

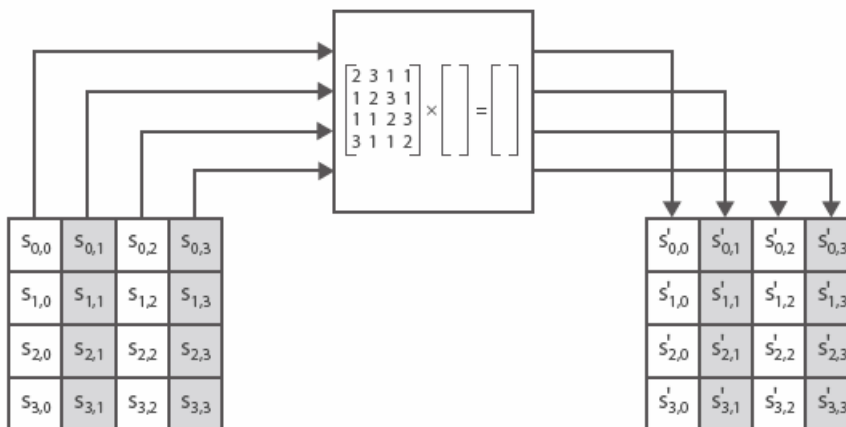
- 1-р мөрийг шилжүүлэхгүй
- 2-р мөрийн байтуудыг зүүн тийш 1 байт шилжүүлнэ
- 3-р мөрийн байтуудыг зүүн тийш 2 байт шилжүүлнэ
- 4-р мөрийн байтуудыг зүүн тийш 3 байт шилжүүлнэ
- Тайлах үйлдлийг хийхдээ баруун тийш шилжүүлэх үйлдлийг хийнэ



Зураг 1.2: ShiftRows үйлдэл

3. MixColumns:

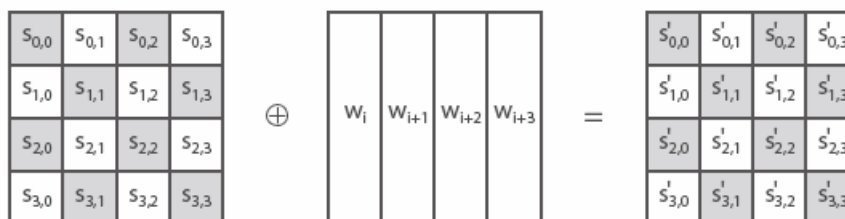
- Багана бүр тус тусдаа холигдоно
- Багана болгоны харгалзаа байтууд хоорондоо солигдоно



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 PCA (RSA)

PCA (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 модулуc 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 $d \equiv e^{-1} \pmod{\phi(n)}$ гэж тодорхойлогдоно.

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

Нууцлал

PCA (RSA) алгоритмын нууцлал маш том хэмжээний анхны тоог хоёр тооны үржигдэхүүн болгон задлах дээр тогтдог ба өнөөгийн бидний машины тооцон бодох чадал хараахан хангалттай биш байгаа юм.

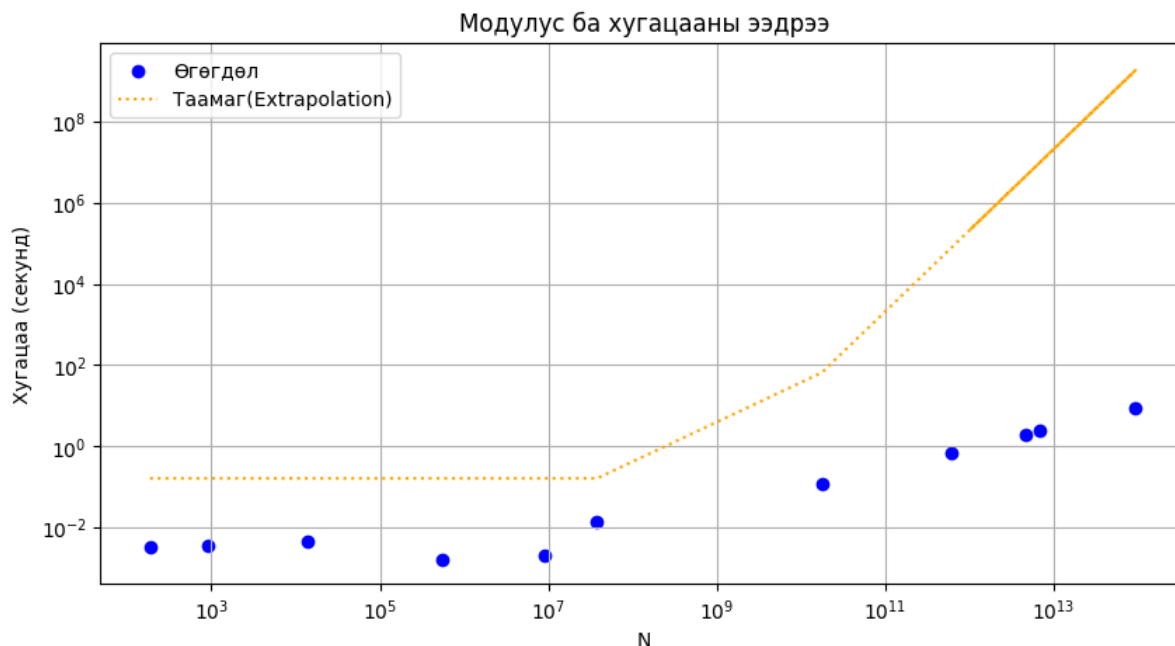
Table 1.1: Муйхар хүчний алгоритм ашиглан PCA (RSA) нууцлалыг эвдэх нь [6]

n	$p \times q$	Оролдого (Хайлт)	Хугацаа (секунд)
187	11×17	5	0.00344800949097
913	11×83	5	0.00358390808105
14041	19×739	8	0.004469871521
557009	653×853	119	0.00167393684387
9192907	937×9811	159	0.00201606750488
37675201	3907×9643	540	0.0139532089233
17614895377	40559×434303	4252	0.117401838303
599855115407	694789×863363	56166	0.712327957153
4684589242027	837533×5593319	66714	1.99000310898
6833740248499	2565161×2664059	187492	2.51408982277
91063247464523	9577907×35324489	188371	8.69724798203

Хамгийн сүүлд үржигдэхүүнд задалж чадсан буюу нууцлал нь амжилттай эвдэгдсэн нь PCA (RSA)-250 буюу 829 бит урттай байгаа юм. Фабрис Будот, Пьеррик Гадри, Ауроре Гилевич, Надия Хенингер, Эммануэль Томе, Пол Циммерманн нараар ахлуулсан судлаачдын баг үүнийг 2020 онд гүйцэтгэсэн. Тооцоололд ойролцоогоор 2700 цөм жил¹ зарцуулагдсан бөгөөд шигших үе шат нь хуанлийн 35 долоо хоног янз бүрийн машинууд дээр хийгдсэн.

PCA (RSA) үржигдэхүүн задлах нь (factoring) цифрийн тоо нэмэгдэх тусам илтгэгч функцээр хугацааны ээдрээ тооцогдох тул одоогийн байдлаар PCA (RSA) 1024, PCA (RSA) 2048 нь

¹Цөм жил гэдэг нь CPU-ний нэг цөмийг бүтэн жил ашигласантай тэнцэнэ.



Зураг 1.5: Хугацааны ээдрээ

хангалттай аюулгүй байгаа бөгөөд дэлхий нийтээрээ ашиглаж байна. Энэ нь дээрх диаграммаас харагдана.

1.2.4 ECC (Эллипс муруйлаг криптограф)

Эллипс муруйлаг криптографи (ECC) нь хязгаарлагдмал талбар дээрх эллипс муруйнуудын алгебрийн бүтцэд суурилсан нийтийн түлхүүрийн криптографийн нэг төрөл юм. Том бүхэл тоонуудын үржвэр дээр суурилдаг RSA-аас ялгаатай нь ECC нь эллиптик муруй дискрет логарифмын бодлогыг (ECDLP) шийдвэрлэхэд хүндрэлтэй байдгаас аюулгүй байдлаа олж авдаг. RSA-аас ECC-ийн мэдэгдэхүйц давуу тал нь түүний үр ашигтай байдал юм; ECC нь RSA-тай ижил түвшний аюулгүй байдлыг RSA-н хажууд асар жижиг хэмжээтэй түлхүүрээр олгодог. Үр ашиг нь илүү хурдан тооцоолол, эрчим хүчний бага зарцуулалт, илүү жижиг хэмжээтэй түлхүүр гэх мэт орох ба ECC нь хөдөлгөөнт төхөөрөмж, ухаалаг карт зэрэг хязгаарлагдмал нөөцтэй төхөөрөмжүүдэд илүү тохиромжтой.

1.2.5 ECC ба RSA харьцуулалт

Битийн аюулгүй байдлын түвшин	RSA бит хэмжээ	ECC бит хэмжээ
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Table 1.2: Аюулгүй байдлын түвшин ба RSA болон ECC түлхүүрийн хэмжээг харьцуулах [8]



Зураг 1.6: RSA ба ECC түлхүүрийн хэмжээнүүдийн харьцуулалт

Table 1.3: 8 бит өгөгдөл – шифрлэлт ба шифр тайлах хугацаа (Секундээр) [8]

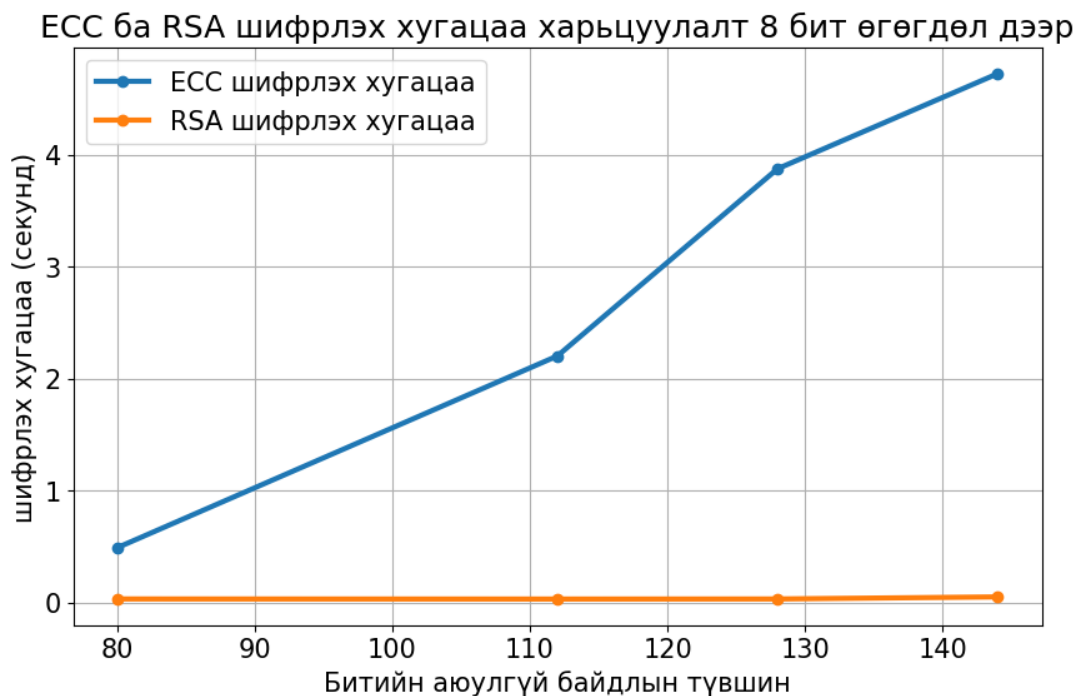
Хамгаалалт	ECC шифр	RSA шифр	ECC тайлах	RSA тайлах	ECC	RSA
80	0.4885	0.0307	1.3267	0.7543	1.8152	0.7850
112	2.2030	0.0299	1.5863	2.7075	3.7893	2.7375
128	3.8763	0.0305	1.7690	6.9409	5.6453	6.9714
144	4.7266	0.0489	2.0022	13.6472	6.7288	13.6962

Table 1.4: 64 бит өгөгдөл – шифрлэлт ба шифр тайлах хугацаа (Секундээр) [8]

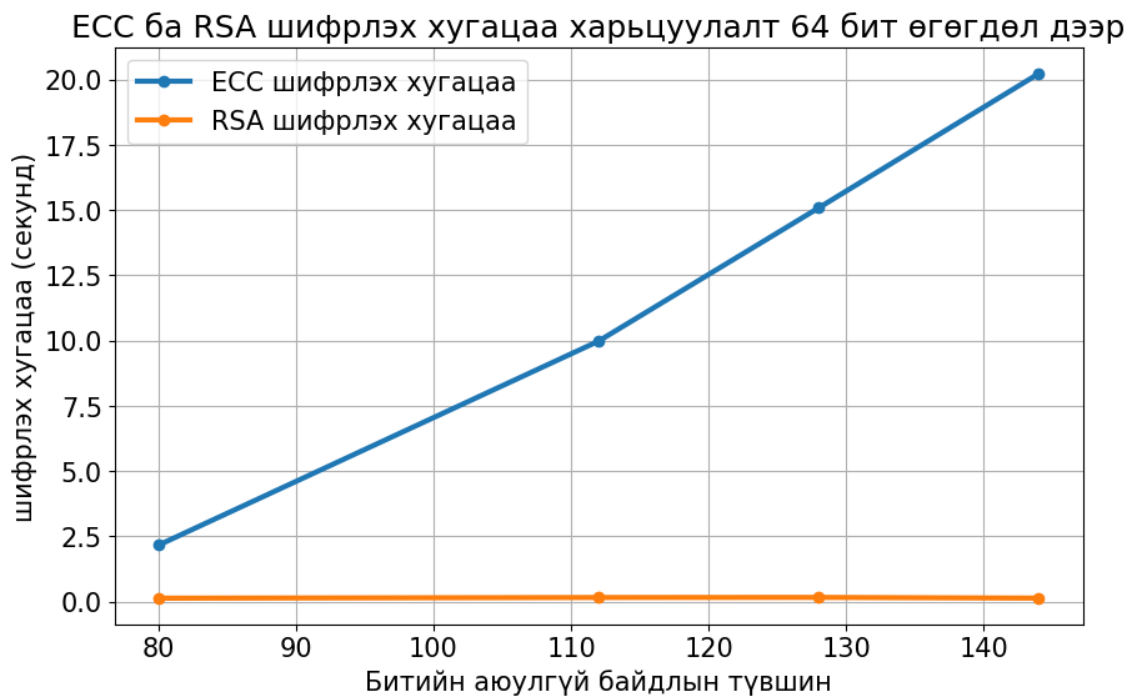
Хамгаалалт	ECC шифр	RSA шифр	ECC тайлах	RSA тайлах	ECC	RSA
80	2.1685	0.1366	5.9099	5.5372	8.0784	5.6738
112	9.9855	0.1635	6.9333	20.4108	16.9188	20.5743
128	15.0882	0.1672	7.3584	46.4782	22.4466	46.6454
144	20.2308	0.1385	8.4785	77.7642	28.7093	77.9027

Table 1.5: 256 бит өгөгдөл – шифрлэлт ба шифр тайлах хугацаа (Секундээр) [8]

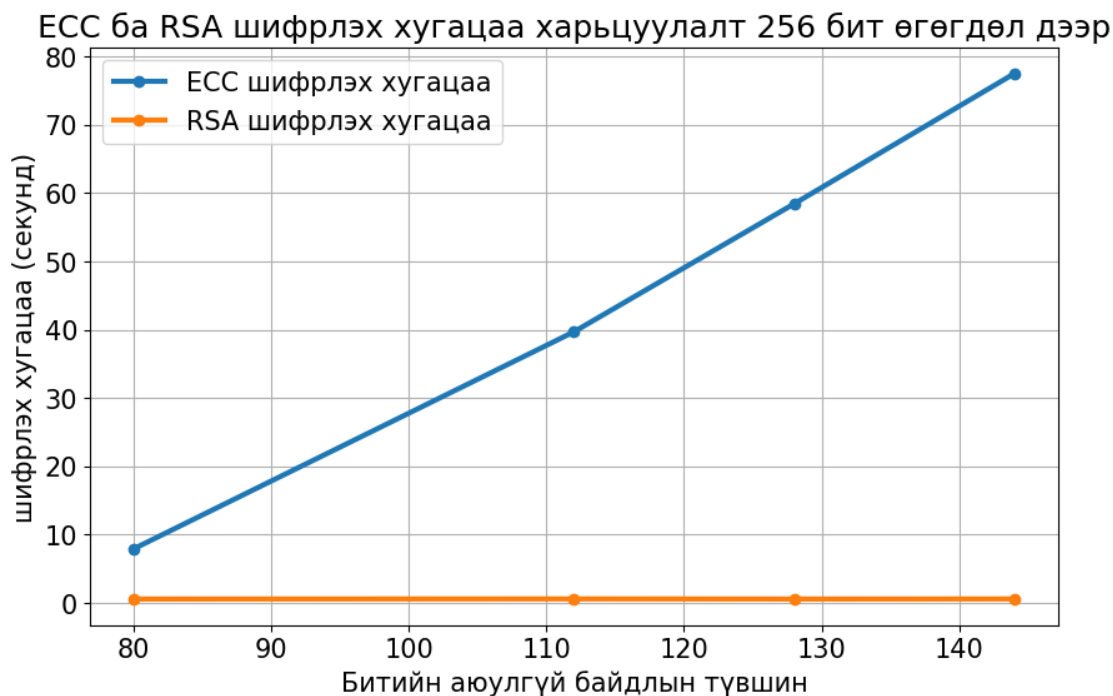
Хамгаалалт	ECC шифр	RSA шифр	ECC тайлах	RSA тайлах	ECC	RSA
80	7.9240	0.5596	22.8851	19.3177	30.8091	19.8772
112	39.7008	0.5815	26.3331	102.0337	66.0339	102.6153
128	58.4386	0.5611	27.4060	209.6086	85.8446	210.1697
144	77.5034	0.5718	32.1522	311.0649	109.6556	311.6368



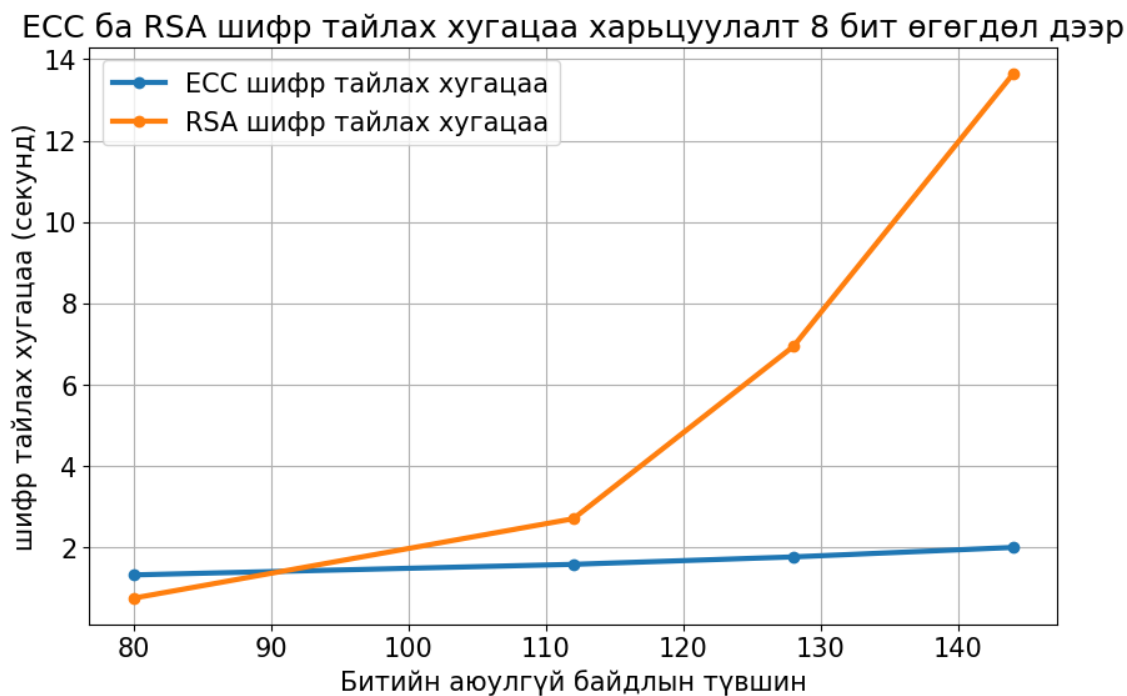
Зураг 1.7: 8 бит өгөгдөл шифрлэлт



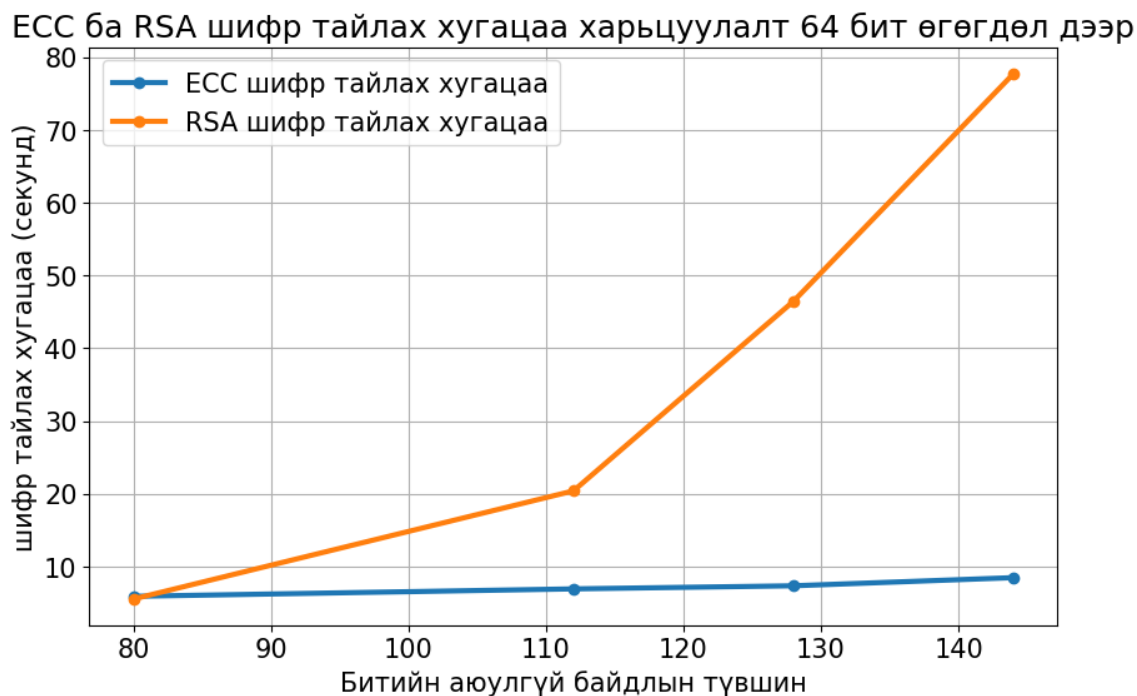
Зураг 1.8: 64 бит өгөгдөл шифрлэлт



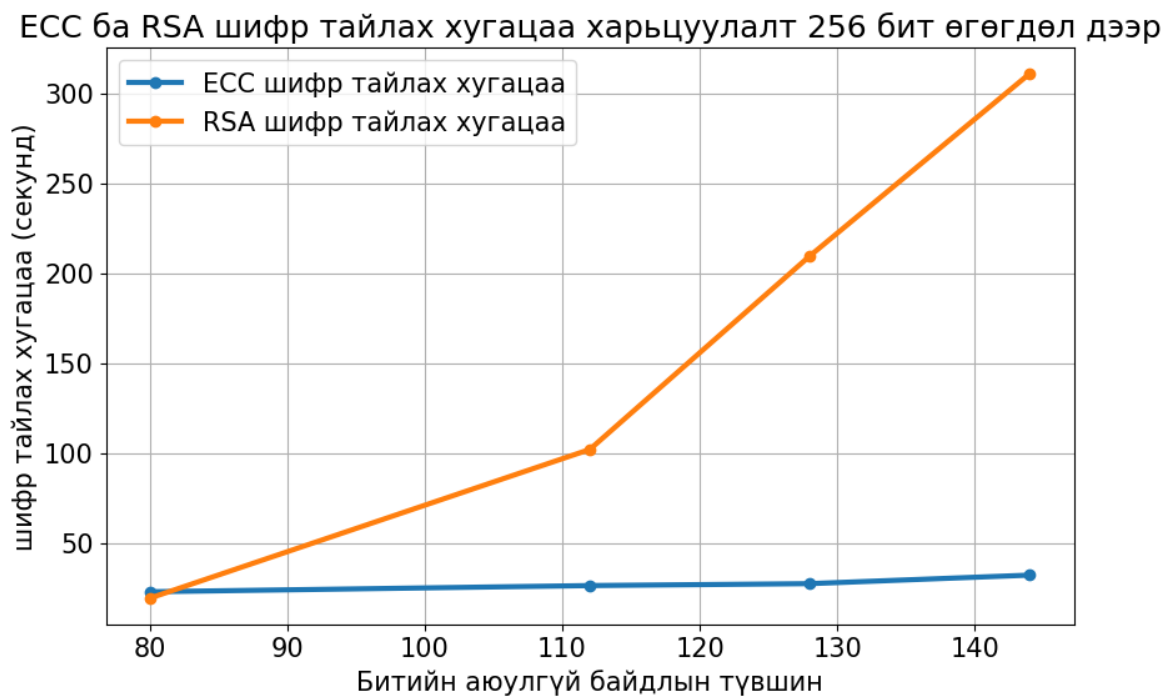
Зураг 1.9: 256 бит өгөгдөл шифрлэлт



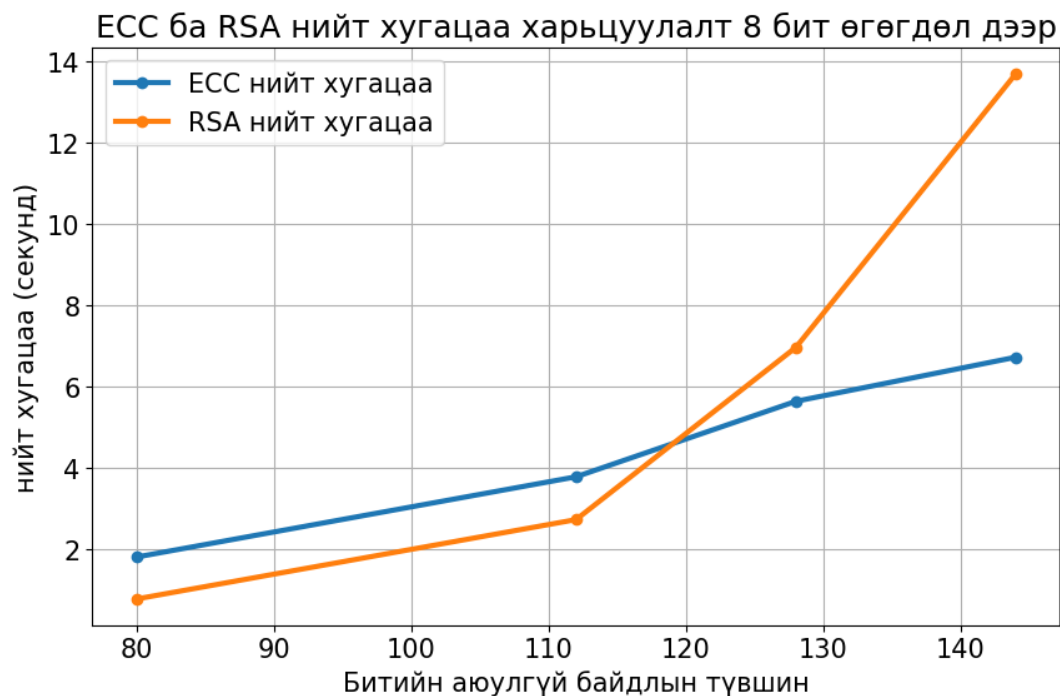
Зураг 1.10: 8 бит өгөгдөл шифрлэлт тайлалт



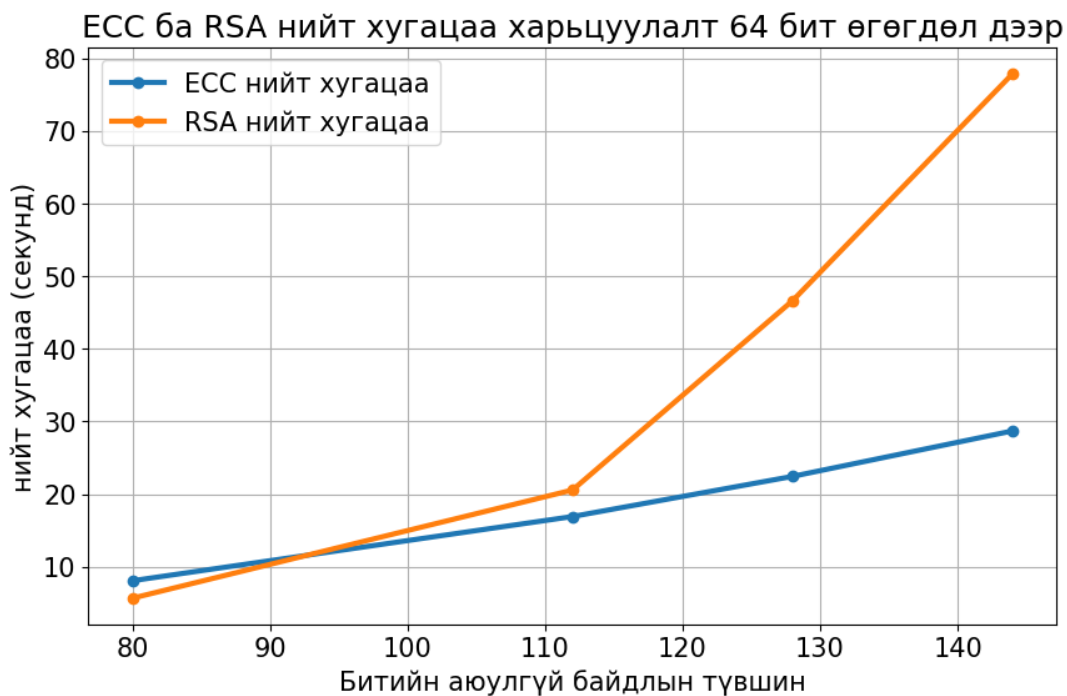
Зураг 1.11: 64 бит өгөгдөл шифрлэлт тайлалт



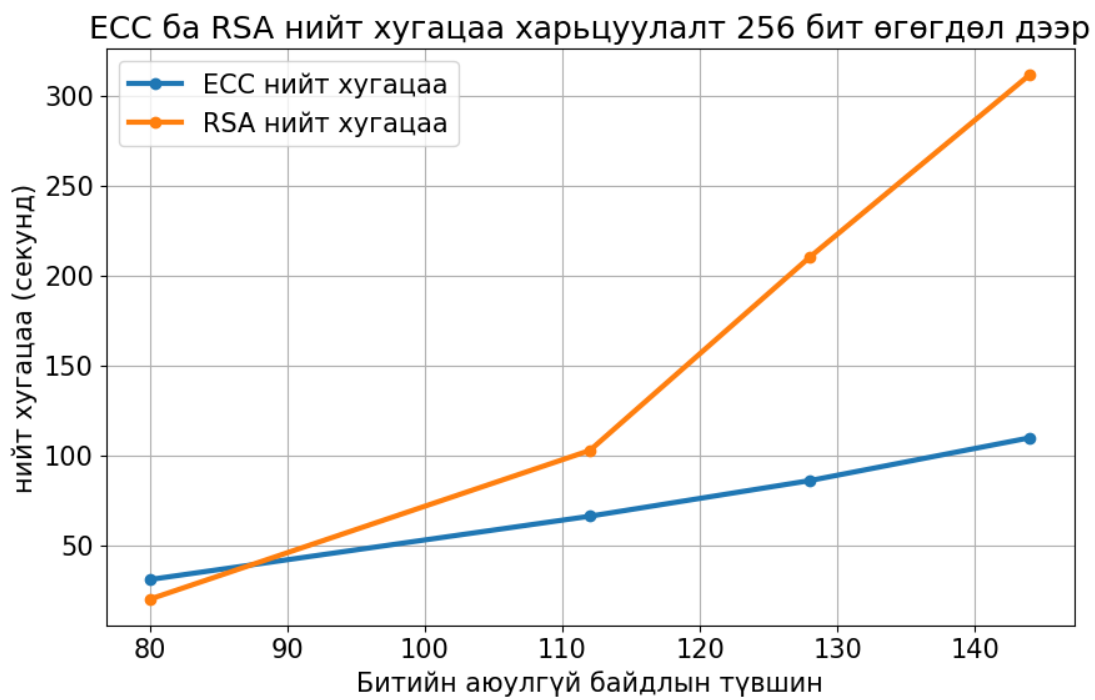
Зураг 1.12: 256 бит өгөгдөл шифрлэлт тайлалт



Зураг 1.13: 8 бит өгөгдөл хугацааны харьцуулалт



Зураг 1.14: 64 бит өгөгдөл хугацааны харьцуулалт



Зураг 1.15: 256 бит өгөгдөл хугацааны харьцуулалт

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 өөрчлөлт орсон эсэхийг мэдэх боломжтой криптографыг 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

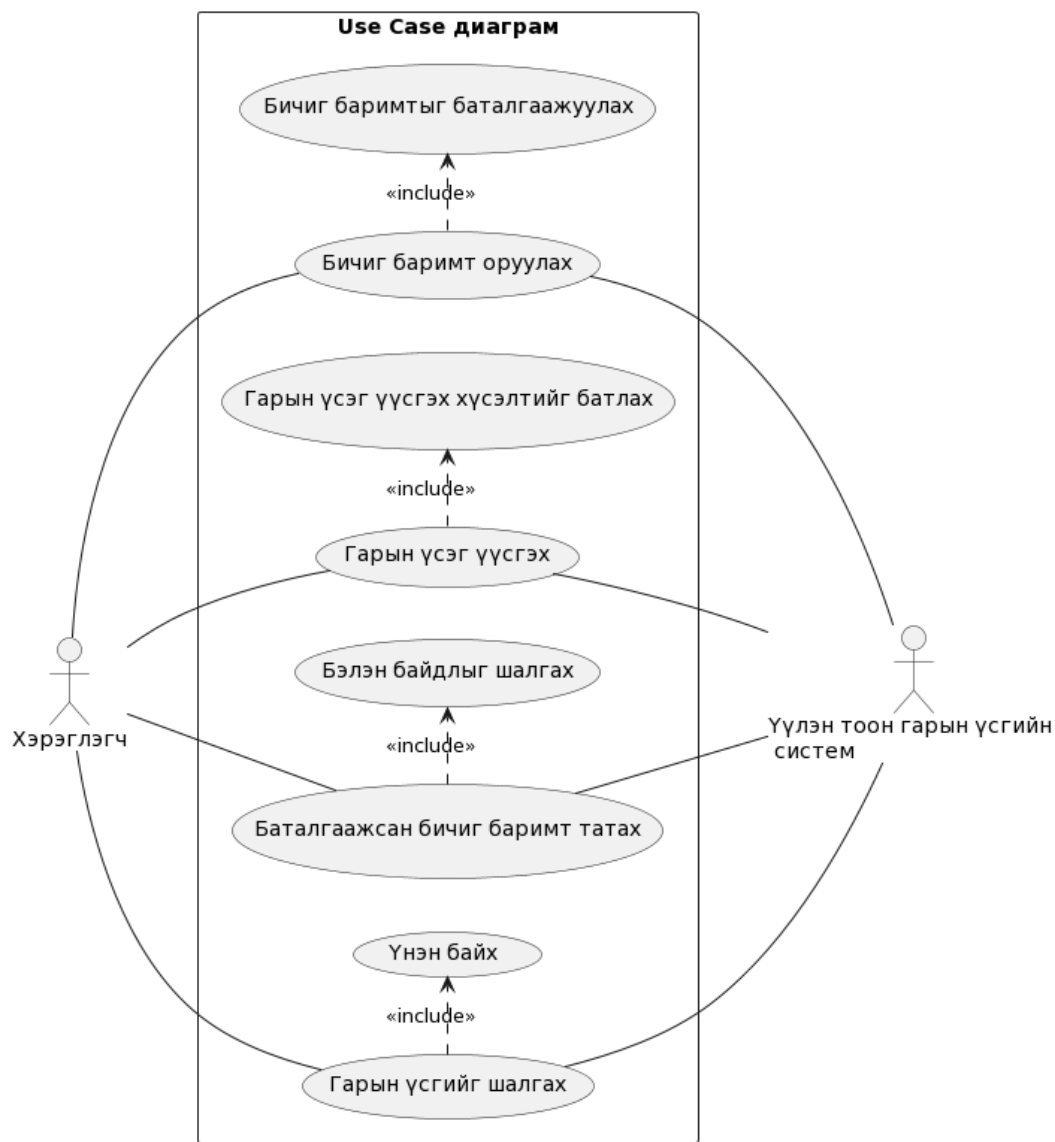
ФШ 100	Систем нь хэрэглэгчийн тоон гарын үсэг үүсгэх чадвартай байх ёстой. Үүнд хэрэглэгч бүрийн өвөрмөц түлхүүрийн хослолыг бий болгох орно.
ФШ 200	Систем нь тоон гарын үсгийг баталгаажуулах функцээр хангах ёстой. Энэ нь гарын үсэг зурсан баримт бичгийг хүлээн авч, гарын үсэг зурсан хүний нийтийн түлхүүрийг 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

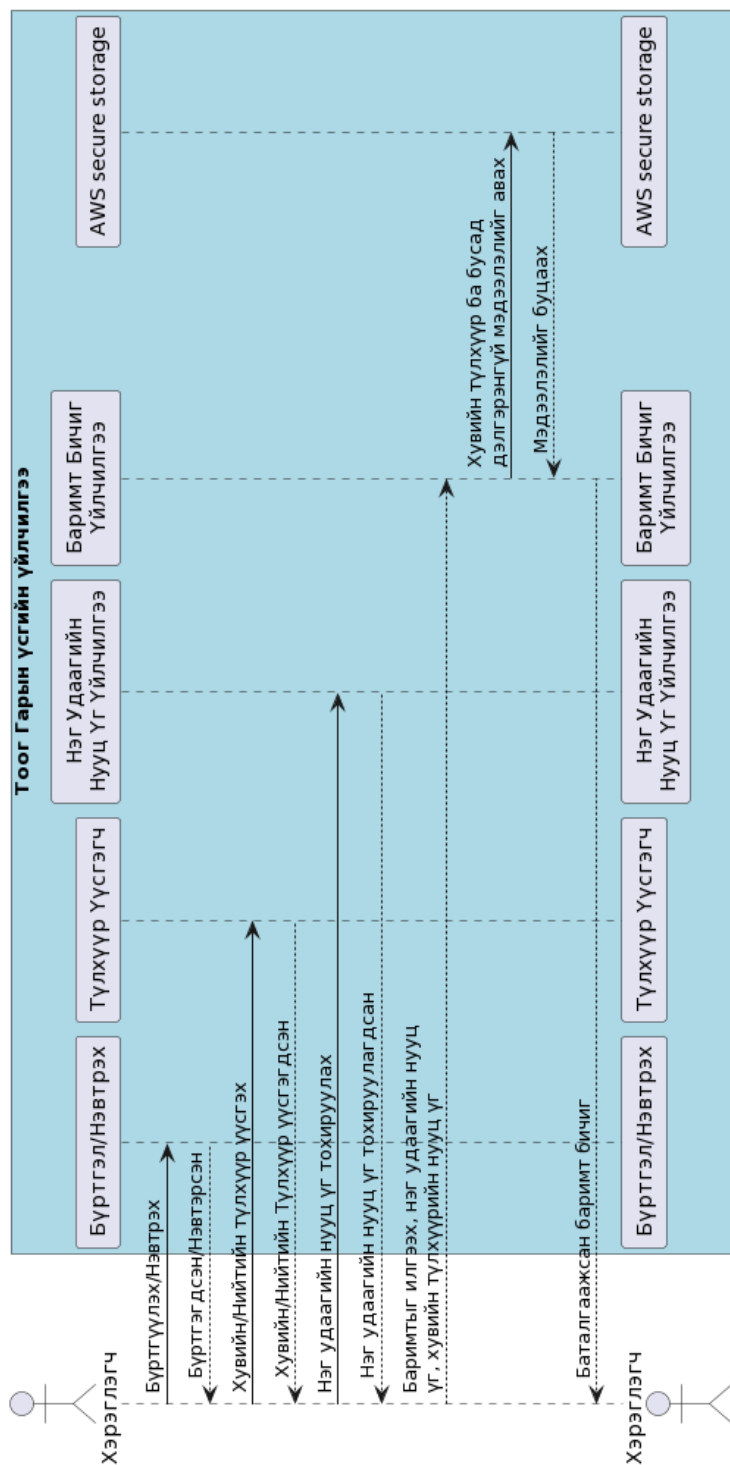
ФБШ 100	Систем нь GDPR эсвэл HIPAA гэх мэт холбогдох бүх мэдээллийн аюулгүй байдал, нууцлалын дүрэм журмыг дагаж мөрдөх ёстой. Гарын үсэг, баримт бичиг зэрэг бүх өгөгдөл шифрлэгдсэн байх ёстой.
ФБШ 200	Систем нь гүйцэтгэлийн бууралтгүйгээр олон тооны хэрэглэгчид болон баримт бичгүүдийг зохицуулах чадвартай 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

Table 2.3: User хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	Varchar	Хэрэглэгчийн дахин давтагдашгүй ID-г хадгална
2	email	Varchar	Хэрэглэгчийн цахим шууданг хадгална
3	password	Varchar	Хэрэглэгчийн нууц үгийг шифрлэж, энэ талбарт хадгална
4	name	Varchar	Хэрэглэгчийн интерфейсээс оруулсан хэрэглэгчийн нэр. Зөвхөн латин үсгийг хадгална.
5	emailVerified	DateTime	Хэрэглэгчийн имэйлийг баталгаажуулсан цагийн тэмдэг
6	image	Varchar	Хэрэглэгчийн байршуулсан зургийн холбоос хадгалагдах бөгөөд зам нь энэ талбарт хадгалагдана
7	role	ENUM	Хэрэглэгчийн нэвтрэлтийн эрх (USER, ADMIN)

Table 2.4: Session хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	Varchar	Нэвтрэлтийн түүхийн өвөрмөц ID
2	sessionToken	Varchar	Гуравдагч этгээдийн токен (Github, Google) байна.
3	userId	Varchar	Хэрэглэгчийн өвөрмөц ID
4	expires	DateTime	Дуусах хугацаа

Table 2.5: VerificationToken хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	identifier	Varchar	Токенд зориулсан өвөрмөц ID
2	token	Varchar	Баталгаажуулалтын Токен
3	expires	DateTime	Дуусах хугацаа

2.7 Гарын үсэг зурах үйл ажиллагааны диаграм

Table 2.6: Account хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	Varchar	Өвөрмөц ID
2	userId	Varchar	Энэ бүртгэлтэй холбоотой хэрэглэгчийн ID
3	type	Varchar	Бүртгэлийн тө
4	provider	Varchar	Аль гуравдагч этгээдийг дамжиж нэвтэрсэн (Github, Google)
5	providerAccountId	Varchar	Хаягийн өвөрмөц ID
6	refresh_token	Varchar	Шинэ токен үүсгэх нууц үг
7	access_token	Varchar	Баталгаажуулах токен
8	expires_at	Int	Дуусах хугацаа
9	token_type	Varchar	Төрөл
10	scope	Varchar	Нэвтрэлтийн эрх
11	id_token	Varchar	Өвөрмөц ID
12	session_state	Varchar	Одоо нэвтрэлттэй байгаа эсэх

Table 2.7: UserUploadedFiles хүснэгт

№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	Varchar	Хэрэгдэгчийн оруулсан файлын өвөрмөц ID
2	userId	Varchar	Файлыг байршуулсан хэрэглэгчийн ID
3	fileName	Varchar	Файлын нэр
4	filePath	Varchar	Файл хадгалагдаж буй зам
5	createdAt	DateTime	Файлыг байршуулсан цаг
6	updatedAt	DateTime	Файлын мэдээлэл хамгийн сүүлд шинэчлэгдсэн цаг

Table 2.8: OtpSecret хүснэгт

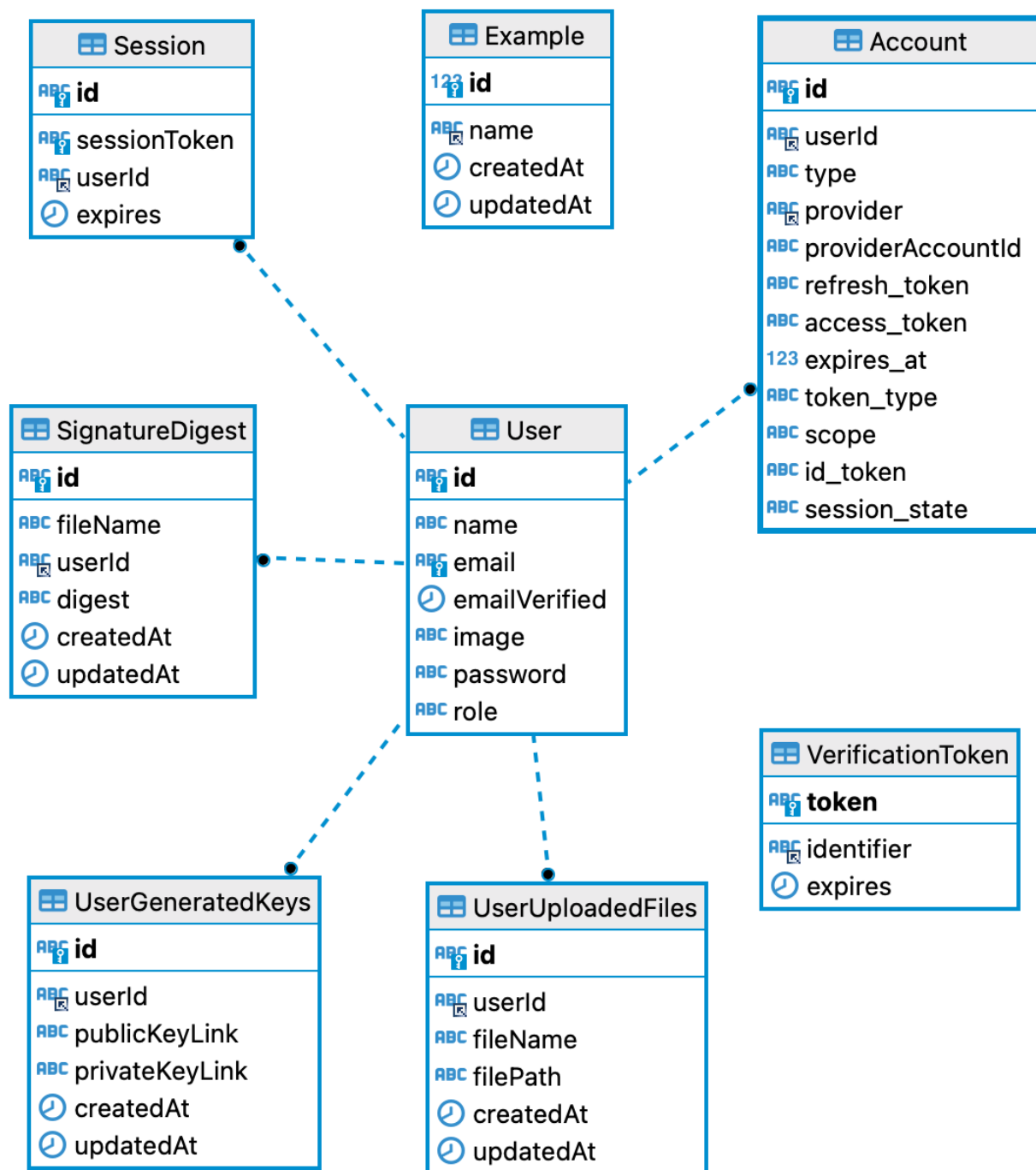
№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	Varchar	Нэг удаагийн нууц үг үүсгэх түлхүүрийн ID
2	userId	Varchar	Холбоотой хэрэглэгчийн ID
3	isVerified	Boolean	ОТР нь баталгаажсан эсэх
4	secret	Text	Нэг удаагийн нууц үгийн баталгаажуулалтад ашигласан нууц
5	createdAt	DateTime	ОТР үүсгэсэн цаг
6	updatedAt	DateTime	Хамгийн сүүлд шинэчилсэн цаг

Table 2.9: SignatureDigest хүснэгт

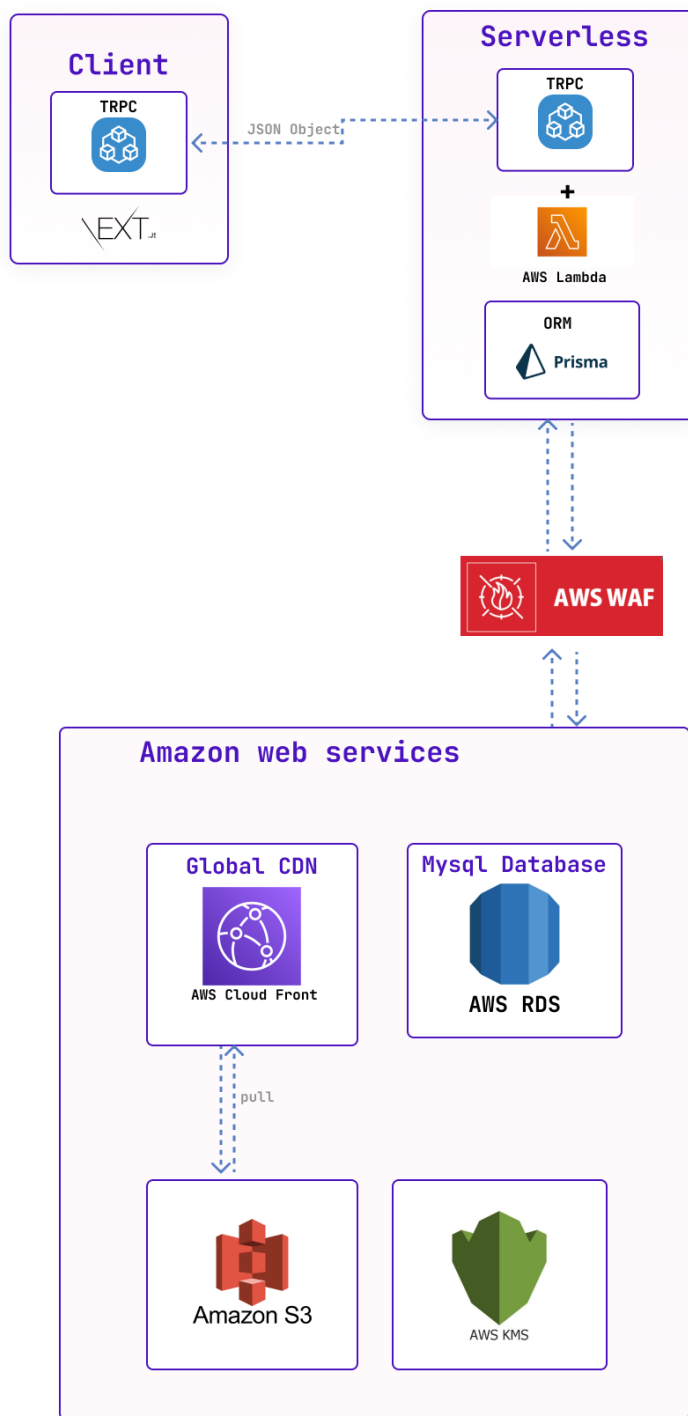
№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	Varchar	Нууц үгийн хайшийн ID
2	fileName	Varchar	Холбогдсон файлын нэр
3	userId	Varchar	Харгалзах хэрэглэгчийн ID
4	digest	Text	Хайшын утга
5	createdAt	DateTime	Үүсгэсэн огноо
6	updatedAt	DateTime	Шинэчилсэн огноо

Table 2.10: UserGeneratedKeys хүснэгт

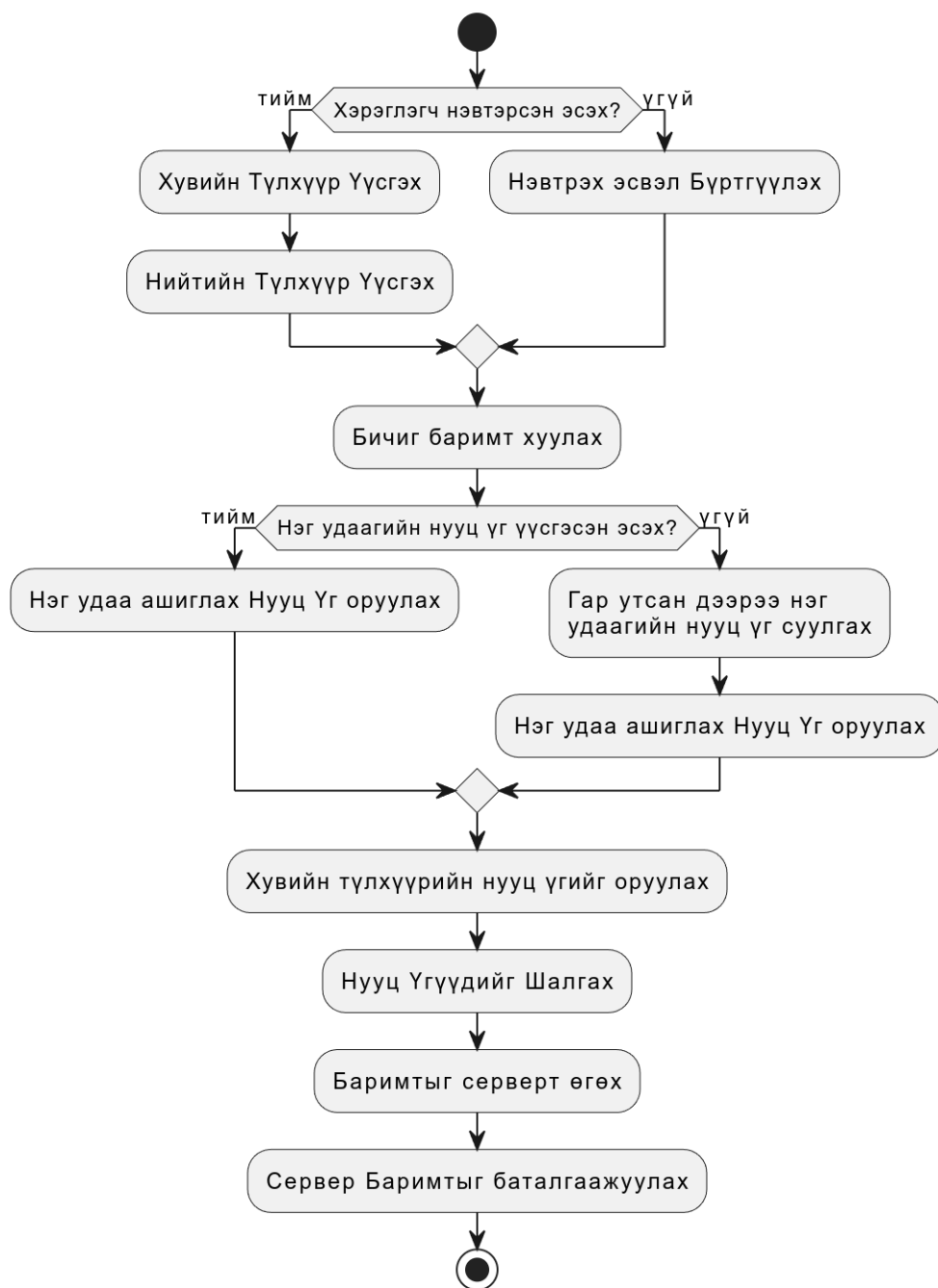
№	Талбарын нэр	Өгөгдлийн төрөл	Тайлбар
1	id	Varchar	Өвөрмөц ID
2	userId	Varchar	Түлхүүрийг үүсгэсэн хэрэглэгчийн ID
3	publicKeyLink	Varchar	Нийтийн түлхүүрийн байршил
4	privateKeyLink	Varchar	Хувийн түлхүүрийн байршил
5	createdAt	DateTime	Үүсгэсэн огноо
6	updatedAt	DateTime	Шинэчилсэн огноо



Зураг 2.3: Датабаз диаграм



Зураг 2.4: Архитектур



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 интерфэйс 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 интерфэйс 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 A.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 контейнерчлах 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 хэмжээний 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 инженер хийх шаардлаггүй болох юм.

```
1 generator client {  
2   provider = "prisma-client-js"  
3 }
```

```
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 шаардлаггүй 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.

```
1 import aws from "aws-sdk";
2
3 aws.config.update({
4   accessKeyId: process.env.S3_ACCESS_KEY,
5   secretAccessKey: process.env.S3_SECRET,
6   region: process.env.AWS_REGION,
7 });
8
9 export const s3 = new aws.S3();
10
11 export default aws;
```

Код 3.2: AWS нууцлалын хэсэг

```
1 import { type PresignedPost } from "aws-sdk/clients/s3";
2 import { s3 } from "~/utils/aws";
3
4 export const uploadToSignedUrl = async ({
5   signedUploadUrl,
6   file,
7   setUploadProgress,
8   index,
9 }): {
```

```
10 signedUploadUrl: PresignedPost;
11 file: File;
12 setUploadProgress: React.Dispatch<React.SetStateAction<number []>>;
13 index: number;
14 }): Promise<void> => {
15   return new Promise((resolve, reject) => {
16     const formData = new FormData();
17     Object.keys(signedUploadUrl.fields).forEach((key) =>
18       formData.append(key, signedUploadUrl.fields[key]!),
19     );
20     formData.append("file", file);
21
22     const xhr = new XMLHttpRequest();
23     xhr.open("POST", signedUploadUrl.url, true);
24
25     xhr.upload.addEventListener("progress", (event) => {
26       if (event.lengthComputable) {
27         const percentComplete = (event.loaded / event.total) * 100;
28         console.log(`Upload is ${percentComplete}% done.`);
29         setUploadProgress((prev) => {
30           const newProgress = [...prev];
31           newProgress[index] = percentComplete;
32           return newProgress;
33         });
34       }
35     });
36
37     xhr.onload = () => {
```

```
38     if (xhr.status > 199 && xhr.status < 300) {
39         resolve();
40     } else {
41         reject(`Error: ${xhr.status}`);
42     }
43 };
44
45 xhr.onerror = () => {
46     reject(`Error: ${xhr.status}`);
47 };
48
49 xhr.send(formData);
50 });
51 };
```

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.

```
1 const queryClient = new QueryClient({
2   queryCache: new QueryCache({
3     onError: (err) => {
4       toast.error(getError(err));
5     },
6   }),
```



```
7   mutationCache: new MutationCache({
8     onError: (error) => {
9       toast.error(getError(error));
10      if (error instanceof TRPCClientError) {
11        const err = error.shape as TRPCErrorShape;
12        if (err.code === TRPC_ERROR_CODES_BY_KEY.UNAUTHORIZED) {
13          modalHandler.setModal(!modalHandler.isModalOpen);
14        }
15      }
16    },
17  }),
18  defaultOptions: {
19    queries: {
20      refetchOnWindowFocus: false,
21      retry: false,
22    },
23    mutations: {
24      retry: false,
25    },
26  },
27 });
```

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.

```

1  const enforceUserIsAuthenticated = t.middleware(({ ctx, next }) => {
2    if (!ctx.session?.user) {
3      throw new TRPCError({
4        code: "UNAUTHORIZED",
5        message: "User is not logged in".toUpperCase(),
6      });
7    }
8    return next({
9      ctx: {
10        // infers the `session` as non-nullable
11        session: { ...ctx.session, user: ctx.session.user },
12      },
13    });
14  });

```

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.

```

1  import { exampleRouter } from "~/server/api/routers/example";
2  import { createTRPCRouter } from "~/server/api/trpc";
3  import { authRouter } from "./routers/auth";
4  import { s3Router } from "./routers/s3";
5  import { secretKeyRoute } from "./routers/key";
6  import { signerRoute } from "./routers/signer";
7  import { otpRoute } from "./routers/otp";
8
9  export const appRouter = createTRPCRouter({
10    auth_router: authRouter,

```

```
11   s3_router: s3Router,
12   key_router: secretKeyRoute,
13   sign_router: signerRoute,
14   otp_router: otpRoute,
15 });
16
17 export type AppRouter = typeof appRouter;
```

Код 3.6: Root

3.4 PCA (RSA) Хэрэгжүүлэлт

PCA нь p , ба q хоёр анхны тооны үржигдэхүүнээр N буюу модулуc тодорхойлогддог. Гэвч 2048 бит мэтийн маш их олон оронтой тоог анхны тоо эсэхийг нь шалгахад нүсэр тооцоолол орох болдог. Иймд Миллер-Рабины тестийг ашиглаж магадлалаар (probabilistic) анхны тоо эсэхийг нь мэддэг юм.

```
1 function RSA(bitSize: number) {
2   const p = generateLargePrime(bitSize);
3   const q = generateLargePrime(bitSize);
4
5   const n = p.multiply(q);
6   const phi = p.minus(1).multiply(q.minus(1));
7
8   let e = bigInt(65537);
9
10  while (!e.greater(phi) || bigInt.gcd(e, phi).notEquals(1)) {
11    e = e.plus(2);
12  }
13 }
```

```
14  const d = e.modInv(phi);
15
16  return {
17    publicKey: {
18      e,
19      n,
20    },
21    privateKey: {
22      d,
23      n,
24    },
25  };
26 }
```

Код 3.7: RSA хэрэгжүүлэлт

Хэдий магадлалаар тооцож байгаа ч k давталтаас хамаарч анхны тоо бус байх магадлал буурна. Жишээ нь $k = 10$ үед $(\frac{1}{4})^{10}$ буюу саяд нэг байх магадлалтай байна.

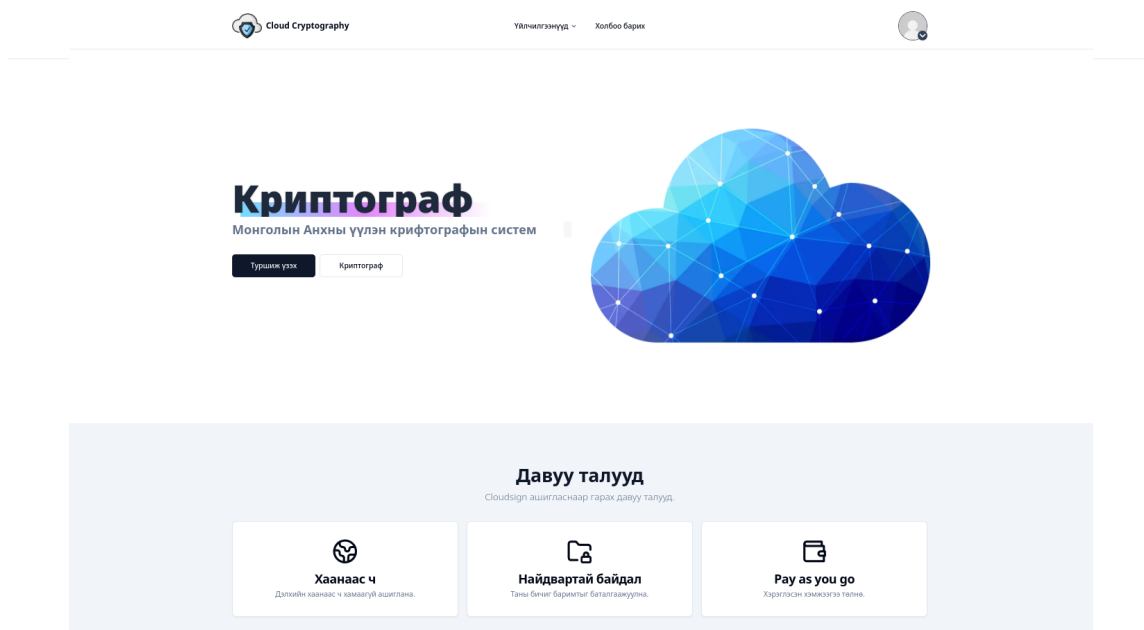
```
1  function isProbablyPrime(n: bigInt.BigInteger, k: number): boolean {
2    if (n.equals(2) || n.equals(3)) return true;
3    if (n.equals(1) || n.isEven()) return false;
4
5    let s = bigInt.zero;
6    let d = n.minus(1);
7
8    while (d.isEven()) {
9      d = d.divide(2);
10     s = s.plus(1);
11   }
```

```
12
13 for (let i = 0; i < k; i++) {
14     const a = bigInt.randBetween(2, n.minus(2));
15     let x = a.modPow(d, n);
16
17     if (x.equals(bigInt.one) || x.equals(n.minus(1))) continue;
18
19     let passed = false;
20     for (let j = 0; j < Number(s) - 1; j++) {
21         x = x.modPow(2, n);
22         if (x.equals(1)) return false;
23         if (x.equals(n.minus(1))) {
24             passed = true;
25             break;
26         }
27     }
28
29     if (!passed) return false;
30 }
31
32 return true;
33 }
```

Код 3.8: Миллер-Рабины тест

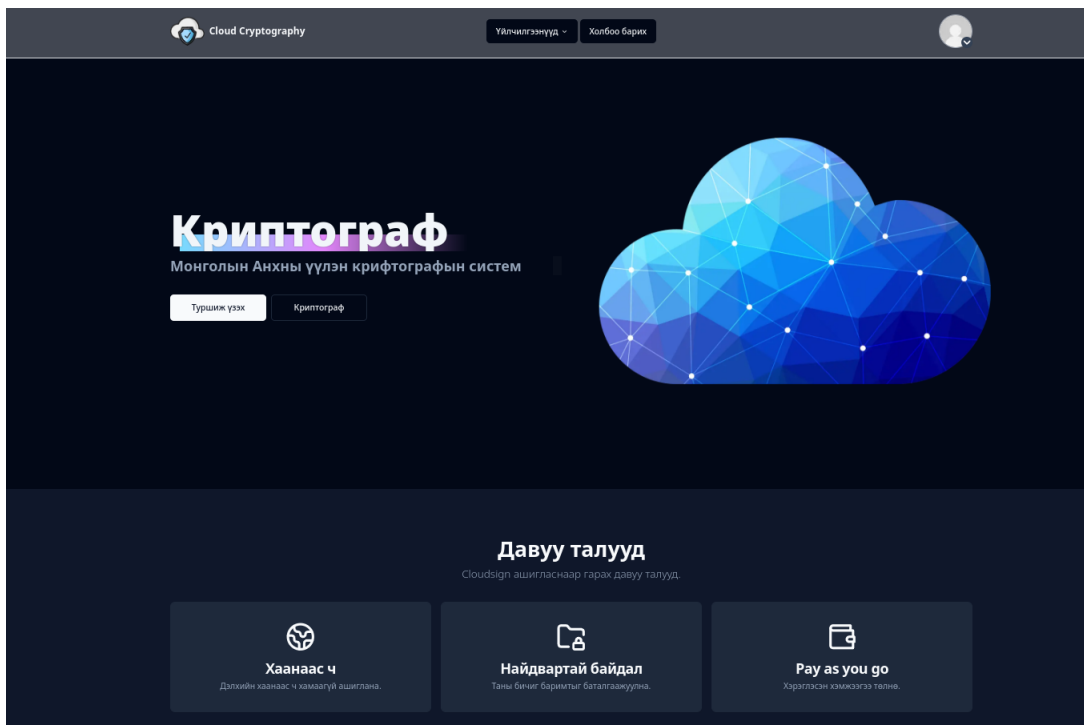
3.5 Үр дүн

Төслийн практик ажлын үр дүнд бүтээгдсэн үүлэн тоон гарын үсгийг системийн интерфейс дараах байдлаар харагдана. Өөрийн хувийн нийтийн түлхүүрийг, нууц үгтэйгээр үүсгэх

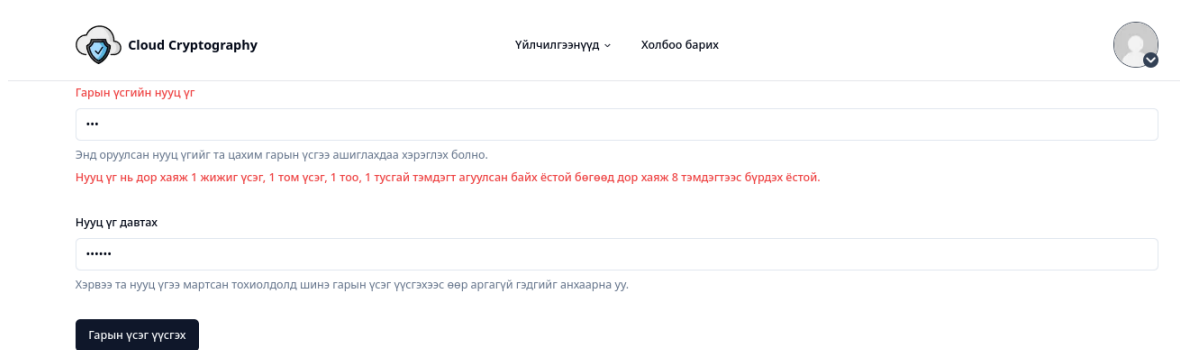


Зураг 3.1: Нүүр хуудас

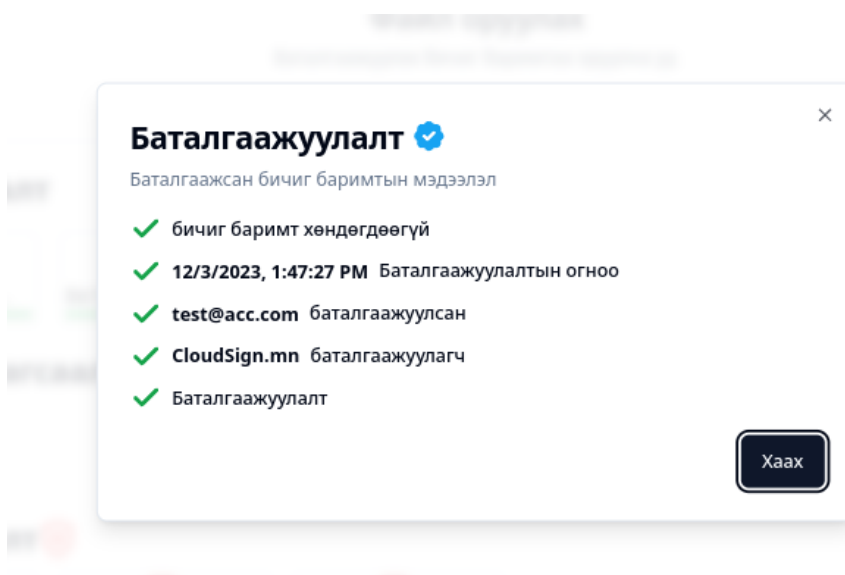
ХЭСЭГ.



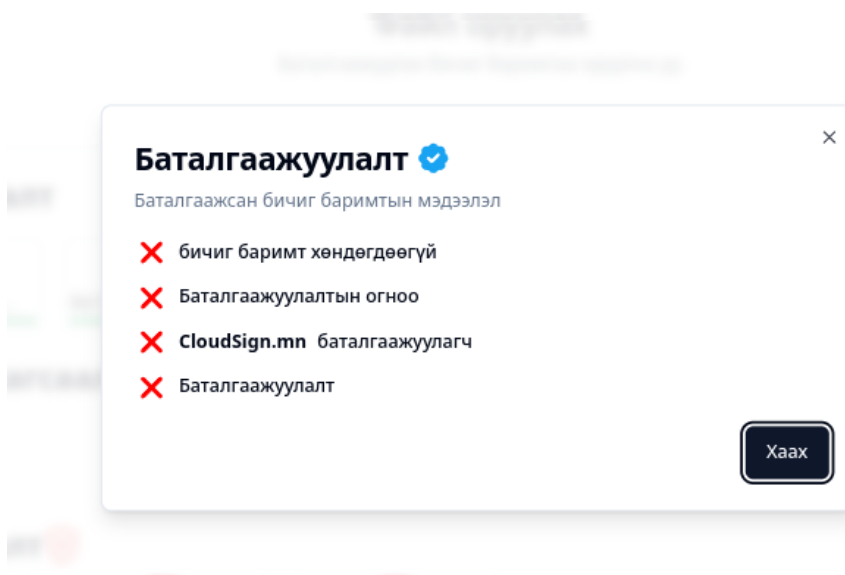
Зураг 3.2: Нүүр хуудас, Шөнийн тохиргоо



Зураг 3.3: Тоон гарын үсэг үүсгэх шаардлага

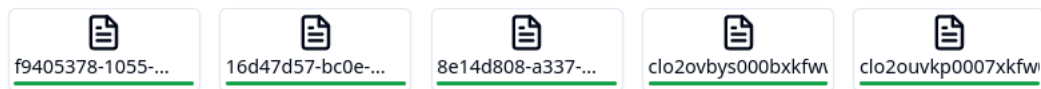


Зураг 3.4: Хүчинтэй гарын үсэгтэй баримт



Зураг 3.5: Хүчингүй гарын үсэгтэй баримт

Оруулсан файлын жагсаалт



Баталгаажсан файлын жагсаалт



Хуурамч файлын жагсаалт



Зураг 3.6: Нийт баримтын жагсаалт

Cloud Cryptography

Үйлчилгээнүүд

Холбоо барих

Криптограф

RSA

Шифрлэх мэдээлэл (Энгийн текст)

Шифрлэх

Шифрлэгдсэн мэдээлэл

Тайлах

Тайлагдсан мэдээлэл

Модулуус (Modulus)

2430667769168421784974992982186229276783464326734914234415365688661482488775272736221884443173152142694972709361687367522892530962605790141444100424662993868488631
5380402167168337710188134134042356074788960619466480577920327652271607022101094558541045044321930018316682449055575322201417319617650264568008526021214951036491550
8145257530446364927663761975881867457234883405235128890085013316752558701337972289039058560099524090325258196778942182894806875831532673301956027111859904584395460
33004018998215176405735512572609556868542341663073198407303580748262613822791259959584978328269652860812721042384694173297304371

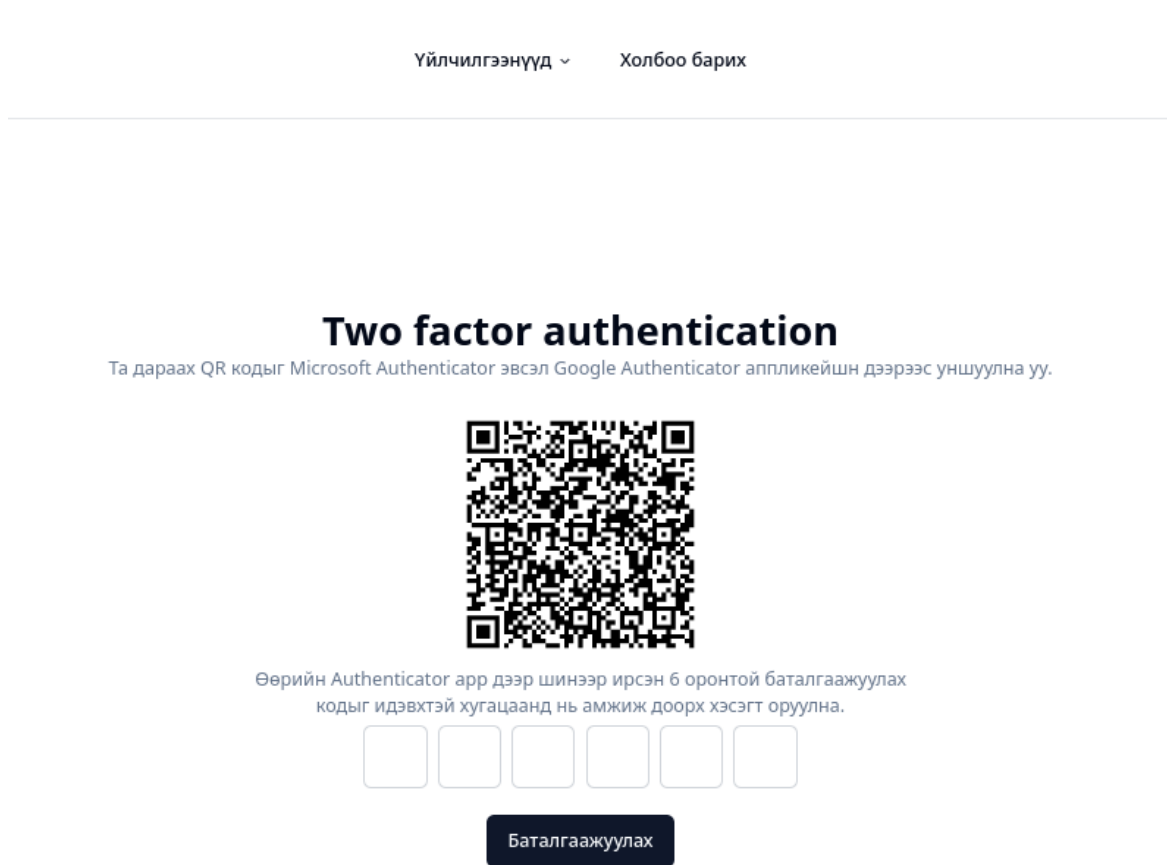
Нууц түлхүүр (Private Exponent)

1055779239987074152646453724268799238713344938843698003478722055881137239524563244360594374194554762500975641714519636962180451235897549858269656602204390114856212
9740089083902184802891046589357412289286661694524353714258647895286474835507283034785062769949650134525715869448411354035836949034832861686150032761586063870340355
1733750894547695821817451106849043985082186851846483698102923397604943054024203192714203247286796441243653320735500599353523880328878655616141444155217295644739163
31100284605278877122597152737619404378548431970858279475257589005178711566657367671412852925732267169159126918973926365805954457

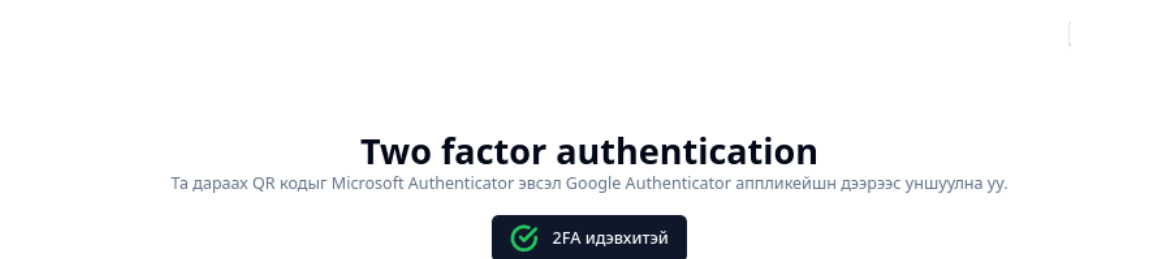
Экспонент (Public Exponent)

65537

Зураг 3.7: PCA (RSA) алгоритмын хэрэгжүүлэлт



Зураг 3.8: Цагаас хамаарсан нууц үг тохируулах



Зураг 3.9: Цагаас хамаарсан нууц үг тохируулсны дараа

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). PCA (RSA) encryption. Encyclopedia Britannica. [https://www.britannica.com/topic/PCA \(RSA\)-encryption](https://www.britannica.com/topic/PCA-(RSA)-encryption)
- [6] Budiman, M. A., & Rachmawati, D. (2018). Using Random Search and Brute Force Algorithm in Factoring the PCA (RSA) Modulus. Retrieved from <https://api.semanticscholar.org/CorpusID:55298868>
- [7] Boudot, F., Gaudry, P., Guillevic, A., Heninger, N., Thomé, E., & Zimmermann, P. (2020, February). A 829-bit factorization. Retrieved from <https://members.loria.fr/PZimmermann/records/factor.html>
- [8] Mahto, Dindayal; YADAV, DILIP. (2017). RSA and ECC: A comparative analysis. International Journal of Applied Engineering Research, Vol. 12, pp. 9053-9061.

A. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

```
1 import { initTRPC, TRPCError } from "@trpc/server";
2 import { type CreateNextContextOptions } from "@trpc/server/adapters/
  next";
3 import { type Session } from "next-auth";
4 import superjson from "superjson";
5 import { ZodError } from "zod";
6
7 import { getServerAuthSession } from "~/server/auth";
8 import { db } from "~/server/db";
9
10 interface CreateContextOptions {
11   session: Session | null;
12 }
13
14 const createInnerTRPCContext = (opts: CreateContextOptions) => {
15   return {
16     session: opts.session,
17     db,
18   };
19 };
20
21 export const createTRPCContext = async (opts: CreateNextContextOptions)
  => {
22   const { req, res } = opts;
23
24   // Get the session from the server using the getServerSession wrapper
  function
25   const session = await getServerAuthSession({ req, res });
26
27   return createInnerTRPCContext({
28     session,
29   });
30 };
31
32 const t = initTRPC.context<typeof createTRPCContext>().create({
33   transformer: superjson,
34   errorFormatter({ shape, error }) {
35     return {
36       ...shape,
37       data: {
38         ...shape.data,
39         zodError:
40           error.cause instanceof ZodError ? error.cause.flatten() :
41             null,
42       },
43     };
44   },
45 });
```

```

45
46 export const createTRPCRouter = t.router;
47 export const publicProcedure = t.procedure;
48
49 const enforceUserIsAuthenticated = t.middleware(({ ctx, next }) => {
50   if (!ctx.session?.user) {
51     throw new TRPCError({
52       code: "UNAUTHORIZED",
53       message: "User is not logged in".toUpperCase(),
54     });
55   }
56   return next({
57     ctx: {
58       // infers the `session` as non-nullable
59       session: { ...ctx.session, user: ctx.session.user },
60     },
61   });
62 });
63
64 export const protectedProcedure = t.procedure.use(enforceUserIsAuthenticated);

```

Код A.1: tRPC тохиргоо

```

1 version: '3.1'
2
3 services:
4   db:
5     container_name: thesis_db
6     image: mysql:latest
7     restart: always
8     environment:
9       # MYSQL_USER: ${MYSQL_USER}
10      MYSQL_DATABASE: ${MYSQL_DATABASE}
11      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
12      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
13     volumes:
14       - db_data:/var/lib/mysql
15     ports:
16       - "3306:3306"
17
18 volumes:
19   db_data:

```

Код A.2: Docker Compose