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

END-TO-END НУУЦЛАЛЫН ПРОТОКОЛ - ТҮҮНИЙ ХЭРЭГЖҮҮЛЭЛТ

End-to-end encryption protocol and implementation

Магистрын судалгааны ажил
Програм хангамж (23m1num0354)

Удирдагч: ..... Доктор Б.Магсаржав

Гүйцэтгэсэн: ..... Э.Цэнгэлт

2024 оны 11 сарын 01

Агуулга

1 Удиртгал

1.1 Ажлын зорилго

Аливаа тусгаар улсын хувьд үндэсний аюулгүй байдлыг хангах асуудал нэн тэргүүнд тавигддаг бөгөөд мэдээллийн технологи үсрэнгүй хөгжиж буй энэ цаг үед мэдээлэл, өгөгдлийг нууцлах, аюулгүй хадгалах, дамжуулах зэрэг асуудлууд нь чухалд тооцогддог билээ. Энэхүү ажлын хүрээнд дээрх асуудлуудын нэгэн шийдэл болох **end-to-end** нууцлалын протокол гэж юу болох, хэрхэн ажиллах туршилтуудыг хийхээс гадна тус нууцлалын протоколыг хэрэгжүүлсэн, програм хангамжийн вэб суурьтай систем бүтээнэ. Уг протокол нь хэрэглэгч хооронд өгөгдлийг нууцлалтай (encrypt) дамжуулах бөгөөд дундын сервэр эсвэл аливаа гуравдагч этгээд (hacker) уг зурвасыг огт тайлж унших боломжгүй бөйх архитектур загвартай протокол юм.

1.2 Ажлын зорилт

Тус ажлын хүрээнд бид дараах зорилтуудыг ерөнхийлөн авч үзнэ.

- Эллиптик муруйн криптографын онол судлах
- Эллиптик муруйн криптографын практик хэрэгжүүлэлтийг судлах
- End-to-end протоколын загвар боловсруулах
- Тухайн протоколын кодыг бичиж, зохих туршилтуудыг хийх
- Вэб болон сервэр програмчилж end-to-end протокол хэрэгжүүлсэн чат систем боловсруулах

1.3 Судалгааны шинэлэг тал

End-to-end протокол тэр дундаа эллиптик муруйн криптографын арга ашиглан нийтийн түлхүүртэй криптосистем үүсгэх, түүнийг бодитоор турших, ашиглах практик манай улсын хувьд дутмаг байдаг. Эсрэгээрээ бусад хөгжингүй болон хөгжиж буй орнууд өөрсдийн нууцлал, аюулгүй байдлын шийдлээрээ мэдээлэл, өгөгдлийн нууцлалаа хангасан програм хангамж, системүүдээ хөгжүүлж ашигладаг. Бидний сайн мэдэх Viber, Messenger, Telegram, Wechat, Signal гэх мэт апп-ууд бүгдээрээ энэхүү протоколыг өөр өөрийн хувилбараар ашиглан хэрэглэгчдийн дамжиж буй өгөгдлийн нууцлал, аюулгүй байдлыг хангаж байдаг. Тиймээс бид энэхүү ажлаараа тухайн протоколыг судлах, өөрсдийн загварыг боловсруулах, турших, практикт хэрхэн хэрэгжүүлэхийг харуулахыг зорьсон.

2 Онолын хэсэг

Хоёр хэрэглэгчийн хооронд нууцлалтай, аюулгүйгээр өгөгдөл дамжуулах асуудлыг математик болон криптографын тусламжтайгаар шийддэг бөгөөд түүний нэг төрлийн арга нь end-to-end нууцлалын протокол юм. Тухайн протоколын шийдэл, аюулгүй байдал нь нийтийн түлхүүртэй криптосистем, бидний хувьд эллиптик муруйн криптосистем байгаа тул энэ хэсэгт эллиптик муруйн криптосистемийн онолын талаар авч үзнэ.

2.1 Эллиптик муруйн математик үндэс

Энэхүү хэсэгт эллиптик муруйн тодорхойлолт, түүний арифметик үйлдэл болоод эллиптик муруйн криптографийн хэрэгжүүлэлтийн тухай авч үзнэ.

2.1.1 Талбарын арифметик

Талбар нь хоёр үйлдэлтэй ба тэдгээрт харгалзах урвуу үйлдэл болон адилтгал элемент бүхий шинж чанартай олонлогоос тогтох алгебрлаг систем юм. Бид талбарын талаар тодорхойлолт болоод ашиглах хүрээнээс хамаарч төгсгөлөг элементтэй талбарын ойлголтыг товч авч үзэх болно.

2.1.2 Талбар

Тодорхойлолт: *Нэмэх (" + "-ээр тэмдэглэх), үржих (" $\cdot$ "-ээр тэмдэглэх) гэсэн хоёр үйлдэлтэй, дараах чанарыг хангах $\mathbb{F}$ олонлогийн бүлийг* ***талбар*** *гэнэ.*

(i) $(\mathbb{F}, +)$ *нь "0" гэж тэмдэглэсэн нэгж элементтэй Абелийн бүлэг байна.*
(Аддитив бүлэг)

(ii) $(\mathbb{F}^*, \cdot)$ *нь "1" гэж тэмдэглэсэн нэгж элементтэй Абелийн бүлэг байна.*
(Мультифликатив бүлэг)
$\mathbb{F}^* = \mathbb{F} \setminus \{0\}$

(iii) $\forall a, b, c \in \mathbb{F} : a \cdot (b + c) = a \cdot b + a \cdot c$ (*Дистрибутив хууль*)

Хэрэв $\mathbb{F}$ талбарын элементийн тоо нь төгсгөлөг бол түүнийг **төгсгөлөг талбар** гэнэ.

2.1.3 Талбарын үйлдэл

$\mathbb{F}$ талбарын хувьд нэмэх, үржих хоёр үйлдэл тодорхойлогдсон. Тэгвэл элементүүдийн "**хасах**" үйлдэл нь нэмэхийн утгаар тодорхойлогдоно: $a, b \in \mathbb{F}, a - b = a + (-b)$, энд $-b$ нь $b + (-b) = 0$ байх $\mathbb{F}$ талбар дахь цорын ганц элемент. ($-b$ элементийг b-ийн *сөрөг* элемент гэнэ). Мөн адилаар, элементүүдийн "**хуваах**"үйлдэл нь үржихийн утгаар тодорхойлогдоно: $a, b \in \mathbb{F}, b \neq 0$ хувьд $a/b = a \cdot b^{-1}$, энд b^{-1} нь $b \cdot b^{-1} = 1$ байх $\mathbb{F}$ талбар дахь цорын ганц элемент. (b^{-1} элементийг b-ийн *урвуу* элемент гэнэ).

Теорем: *Хэрэв $\mathbb{F}$ нь q элементтэй ($|\mathbb{F}| = q$) төгсгөлөг талбар бол q нь анхны тооны зэрэг байна. Ө.х $q = p^m$, $p \in \mathbb{P}$. Иймд p тоог талбарын* ***характеристик*** *гэж нэрлэдэг.*

$q = p^m$ элементтэй төгсгөлөг талбарыг Галуагийн талбар гэж нэрлэдэг ба $\mathbb{F}_{p^m}$ эсвэл $GF(p^m)$ гэж тэмдэглэнэ.
Хэрэв $m = 1$ бол $\mathbb{F}_p$ -ийг анхны талбар, $m \geq 2$ бол $\mathbb{F}_{p^m}$-ийг өргөтгөсөн талбар гэнэ.

Теорем: $\mathbb{F}_q$ *төгсгөлөг талбарын хувьд*

$$\mathbb{F}_q^* = \{1, g, g^2, g^3, ..., g^{q-2}\}$$

байх $g \in \mathbb{F}_q$ элемент оршин байна. Өөрөөр хэлбэл, $(\mathbb{F}_q^*, \cdot)$ нь цикл бүлэг байна. Иймд g-г талбарын ***үүгэгч*** *элемент (примитив язгуур) гэж нэрлэдэг.*

2.1.4 Анхны талбар

p анхны тооны хувьд p модулиар нэмэх, үржих үйлдлээрх үлдэгдлийн бүхэл тоонуудын олонлог $\{0, 1, 2, ..., p-1\}$ нь p эрэмбийн төгсгөлөг талбар болно. Энэ талбарыг $\mathbb{F}_p$ гэж тэмдэглэе. Аливаа a бүхэл тооны хувьд p-д хуваахад гарах r $(0 \leq r \leq p-1)$ үлдэгдлийг $a \ mod p$ гэж тэмдэглэнэ.
$a, b \in \mathbb{F}_p$ тоонуудын анхны талбар $\mathbb{F}_p$ дээрх үйлдлүүд дараах байдлаар хийгдэнэ.

$$a + b \equiv (a + b) \ mod p, \quad a \cdot b \equiv (a \cdot b) \ mod p$$

Жишээ: ($\mathbb{F}_{29}$ -анхны талбар) $\mathbb{F}_{29} \equiv \{0, 1, 2, ..., 28\}$. $\mathbb{F}_{29}$ талбарт дараах арифметик үйлдлүүдийн жишээ биелэнэ:

(i) Нэмэх: $17 + 20 = 8$, энд $37 \ mod \ 29 = 8$.

(ii) Хасах: $17 - 20 = 26$, энд $-3 \ mod \ 29 = 26$.

(iii) Үржих: $17 \cdot 20 = 21$, энд $340 \ mod \ 29 = 21$.

(iv) Урвуу: $17^{-1} = 12$, энд $17 \cdot 12 \ mod \ 29 = 1$.

Цаашид бидний авч үзэх хүрээ ихэвчлэн анхны талбарт хийгдэх болно. Төгсгөлөг талбарын арифметик үйлдлийн үр ашигтай гүйцэтгэл нь эллиптик муруйн системд чухал урьдчилсан нөхцөл юм. Учир нь тухайн талбар дахь муруйн үйлдлүүд түүнийг хэрэглэж гүйцэтгэгдэнэ.

2.2 Эллиптик муруйн үндэс

2.2.1 Эллиптик муруй

$a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$ талбарын элементүүд хувьд

$$E: \ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (1)$$

тэгшитгэлийн шийд байх (x, y) цэгүүдийн олонлогийг $\mathbb{F}$ талбар дээрх **эллиптик муруй** гэнэ. L нь $\mathbb{F}$ талбарын өргөтгөл болог. (1) тэгшитгэлийн шийдийн олонлогийг L дээр авч үзвэл

$$E(L) = \{y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0\} \cup \{\infty\}$$

олонлогийг L дээрх рационал цэгүүд буюу L дээрх эллиптик муруйн гэнэ. (Энд ∞ - төгсгөлгүй алсын цэг). (1)-ийг **Вейерштрассын өргөтгөсөн тэгшитгэл** гэдэг.
Шинж чанарын хувьд дараахыг тэмдэглэе:
- $a_i, \quad i = \overline{1,6}$ коэффициентүүдээс хамаарч (1) тэгшитгэл өөр өөр, харгалзах муруй нь янз бүрийн хэлбэртэй байна.
- $\mathbb{F}$ төгсгөлөг талбар үед уг цэгүүдийн олонлогийг зураглалаар дүрслэхэд амаргүй.
- $\mathbb{F} = \mathbb{Q}, \mathbb{R}, \mathbb{C}$ үед муруйн графикыг зурах боломжтой.

Бүтэн квадрат ялгах аргаар (1)-ийг

$$\left(y + \frac{a_1x}{2} + \frac{a_3}{2}\right)^2 = x^3 + \left(a_2 + \frac{a_4^2}{4}\right)x^2 + \left(a_4 + \frac{a_1a_3}{2}\right)x + \left(\frac{a_3^2}{4} + a_6\right)$$

гэж хувирган бичиж болно. Энэ тэгшитгэлийн баруун гар тал

$$x^3 + \left(a_2 + \frac{a_4^2}{4}\right)x^2 + \left(a_4 + \frac{a_1a_3}{2}\right)x + \left(\frac{a_3^2}{4} + a_6\right) = 0 \qquad (*)$$

тэгшитгэлийн 3 язгуур x_1, x_2, x_3 бол

$$\Delta = [(x_1 - x_2)(x_1 - x_3)(x_2 - x_3)]^2$$

тоог **дискриминант** гэдэг.

Санамж: (1) муруйн хувьд дараах эквивалент өгүүлбэрүүд биелнэ.

(i) $(*)$ куб тэгшитгэл давхар язгуургүй байхыг шаардана.

(ii) $\Delta \neq 0$

(iii) (1) муруйн цэг бүрт зөвхөн ганц л шүргэгч татаж болох *гөлгөр* муруй байна.

Эллиптик муруй бүхэн изоморфийн нарийвчлалтайгаар нэг утгатай тодорхойлогддог болохыг харж болдог. Иймд E эллиптик муруйн (1) тэгшитгэлийг өөртэй нь изоморф хялбар дүрстэй (өөрөөр хэлбэл; ихэнх коеффициент нь тэг байх) муруйд хувиргаж болно. Энэ хувиргалт нь талбарын характеристикаас хамаардаг. Бид энэ удаад $p > 3$ характеристиктэй $\mathbb{F}_p$ анхны талбар дээрх $E(\mathbb{F}_p)$ муруйн тухай асуудлыг авч үзнэ. $\mathbb{F}_p$ талбарт (1) тэгшитгэлийг

$$(x, y) \longrightarrow \left(\frac{x - 3a_1^2 - 12a_2}{36}, \quad \frac{y - 3a_1 x}{126} - \frac{a_1^3 + 4a_1 a_2 - 12a_3}{24} \right)$$

хувиргалтаар $a, b \in \mathbb{F}_p$ байх

$$y^2 = x^3 + ax + b \qquad (2)$$

тэгшитгэлд шилжинэ. Энэ муруйн дискриминант нь $\Delta = -16(4a^3 + 12b^2) \neq 0$ болно. Иймд бид цаашид $\mathbb{F}_p,\ p > 3$ талбарт эллиптик муруйн тэгшитгэлийг (2) хэлбэртэйгээр тодорхойлж болох ба тус муруйн шийдүүдийн олонлогийг

$$E(\mathbb{F}_p) = \{(x, y) \mid y^2 = x^3 + ax + b, \quad a, b \in \mathbb{F}_p\}$$

гэж тэмдэглэе.

2.2.2 Эллиптик муруйн арифметик үйлдэл

$E(\mathbb{F}_p)$ муруйн хоёр цэгийг нэмж муруй дээр орших гурав дахь цэгийг гаргах "**хөвч - шүргэгчийн дүрэм**"/chord-and-tangent rule/ оршин байна. Муруйн цэгүүдийн олонлог $E(\mathbb{F}_p)$ нь уг нэмэх үйлдэлээр ∞ адилтгал элементтэй абелийн бүлэг үүсгэнэ. Энэ бүлэг нь эллиптик муруйн криптографик системийг байгуулахад хэрэглэгддэг.

$E(\mathbb{F}_p):\ y^2 = x^3 + ax + b,\ (p > 3)$ эллиптик муруйн аффин координатаарх бүлгийн үйлдэл нь дараах байдлаар тодорхойлогдоно:

1. (**Адилтгал элемент**): аливаа $P \in E(\mathbb{F}_p)$ хувьд $P + \infty = \infty + P = P$ байна.
2. (**Сөрөг элемент**): $P = (x, y) \in E(\mathbb{F}_p)$ бол $(x, y) + (x, -y) = \infty$ байх $(x, -y) := -P$ цэгийг сөрөг элемент гэнэ. $(\infty = -\infty)$.
3. (**Нэмэх**): $P = (x_1, y_1) \in E(\mathbb{F}_p), Q = (x_2, y_2) \in E(\mathbb{F}_p)$ цэгүүдийн хувьд $P \neq \pm Q$ байг. Тэгвэл цэгүүдийн нийлбэр $P + Q = (x_3, y_3)$ нь дараахаар илэрхийлэгдэнэ:

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2 - x_1 - x_2, \quad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1} \right)(x_1 - x_3) - y_1.$$

4. (**Хоёрчлох**): $P = (x, y) \in E(\mathbb{F}_p),\ P \neq -P$ цэгийн хувьд $P + P = 2P = (x_3, y_3)$ цэг дараахаар илэрхийлэгдэнэ:

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1} \right)^2 - 2x_1, \quad y_3 = \left(\frac{3x_1^2 + a}{2y_1} \right)(x_1 - x_3) - y_1.$$

2.2.3 Цэгийн тоо буюу бүлгийн эрэмбэ

$\mathbb{F}_p$ төгсгөлөг талбар дээрх $E(\mathbb{F}_p)$ муруйн цэгүүдийн тоог $\#E(\mathbb{F}_p)$ гэж тэмдэглээд муруйн **эрэмбэ** гэдэг. Аливаа $x \in \mathbb{F}_p$ бүрийн хувьд Вейерштрасын тэгшитгэлүүд хоёроос олонгүй шийдтэй тул $1 \leq \#E(\mathbb{F}_p) \leq 2p+1$ болно. $\#E(\mathbb{F}_p)$ тоог ойролцоо тодорхойлдог Хассе(Hasse)-ийн теоремийг авч үзье.

Теорем:(Hasse) *E нь $\mathbb{F}_p$ талбар дээрх эллиптик муруй бол дараах биелнэ.*

$$p + 1 - 2\sqrt{p} \leq \#E(\mathbb{F}_p) \leq p + 1 + 2\sqrt{p}.$$

Энэ теоремын ач холбогдол нь p анхны тоо, $|t| \leq 2\sqrt{p}$ байх аливаа t тооны хувьд $\#E(\mathbb{F}_p) = p + 1 - t$ эрэмбэтэй $\mathbb{F}_p$ талбар дээрх E эллиптик муруй оршин байна. Ийм t тоог *муруйн үл*(trace) гэдэг.

2.2.4 Цэгийн үржвэр

k бүхэл тоо ба $P \in E(\mathbb{F}_p)$ цэгийн хувьд өөрийг нь өөр дээр нэмэх буюу хоёрчлох үйлдэл дээр үндэслэн цэгийн k удаа нэмэхийг $P + P + ... + P = kP$ гэж тэмдэглэн цэгийн үржих үйлдэл буюу "***цэгийн үржвэр***"эсвэл "***скаляр үржвэр***"(Point multiplication) гэж нэрлэдэг.

kP нь муруйн өөр нэг цэгт буух ба P-ээр төрөгдсөн $\langle P \rangle$ бүлгийн элемент байна. kP утгыг хэрхэн үр ашигтай тооцоолох нь эллиптик муруйн криптосистемийн гүйцэтгэлд чухал үр нөлөөтэй. Бид k зэргийн хоёртын дүрслэлийг ашиглаж хоёрын зэрэгтүүдээрх нийлбэрийг (хоёрчлох үйлдлийг) давтах замаар цэгийн үржвэрийг олох алгоритмыг авч үзнэ.

$\#E(\mathbb{F}_p) = nh$, энд n анхны тоо, h бага тоо ($n \approx q$), P, Q цэгүүд ижил n эрэмбэтэй ба үржүүлэгч k нь $[1, n-1]$ завсраас санамсаргүй сонгогдсон гэж үзнэ. k тооны хоёртын (бинар) дүрслэлийг $(k_{s-1}, ..., k_2, k_1, k_0)_2$ гэе. $s \approx m = \lceil log_2 p \rceil$
Алгоритм 1 нь k тооны битүүдээр баруунаас зүүн, Алгоритм 2 нь зүүнээс баруун тийш үйлдэнэ.

Algorithm 1 Right-to-left binary method

Оролт: $k = (k_{s-1}, ..., k_2, k_1, k_0)_2,\ P \in E(\mathbb{F}_p)$
Гаралт: kP

1: $Q \leftarrow \infty$
2: **for** from $i = 0$ to $s - 1$ **do**
3: **if** $k_i = 1$ **then**
4: $Q \leftarrow Q + P$.
5: $P \leftarrow 2P$.
6: **return** Q.

2.3 Эллиптик муруйн Дискрет логарифмын бодлого

Эллиптик муруйн дискрет логарифмын бодлогын хүндрэл нь эллиптик муруйд суурилсан бүх криптографик схемүүдийн аюулгүй байдлын амин чухал үндэс нь болдог. Дараах тодорхойлолтоор өгөгдөх бодлогыг **эллиптик муруйн дискрет логарифмын бодлого** (ECDLP) гэнэ.

Тодорхойлолт: *$E(\mathbb{F}_p)$ нь $\mathbb{F}_p$ талбар дээр тодорхойлогдсон эллиптик муруй (бүлэг), $P \in$*

Algorithm 2 Left-to-right binary method

Оролт: $k = (k_{s-1}, ..., k_2, k_1, k_0)_2,\ P \in E(\mathbb{F}_p)$
Гаралт: kP
1: $Q \leftarrow \infty$
2: **for** i from $s-1$ downto 0 **do**
3: $\quad Q \leftarrow 2Q$.
4: $\quad$ **if** $k_i = 1$ **then**
5: $\quad\quad Q \leftarrow Q + P$.
6: **return** Q.

$E(\mathbb{F}_p)$ нь n-эрэмбийн цэг (өө.х $nP = \infty$). $Q \in \langle P \rangle$ дэд бүлэгт харьяалагдах цэг болог. $Q = kP$ байх k тоог Q цэгийн P суурьтай дискрет логарифм гэж нэрлээд формалаар $k = \log_P Q$ гэж тэмдэглэн бичнэ.

ECDLP-ийн хүнд хөнгөнөөс эллиптик муруйн криптографын нууцлалын хүнд, хөнгөн эсэх нь хамаардаг. Хар ухаанаар $Q = kP$ байх k тоог олоё гэвэл $P, 2P, 3P, ...$ гэх мэт P-гийн давтамжуудыг тооцон олж болох мэт харагдаж байвч k-маш том тоо үед энэ тооцоо практик боломжгүй. Ингэхдээ хамгийн багадаа $\frac{n}{2}$-оос цөөнгүй удаа P-ийн давтамжийг бодох хэрэгтэй болдог. Иймд k-тоог өнөөдрийн компьютерийн тооцоолох хүчин чадлаас давсан байхаар сонгосон бол энэ маш хүнд бодлого байх юм.

ECDLP-д довтлох янз бүрийн аргууд байдаг. Тэдгээр довтолгооноос зайлсхийхийн тулд $\#E(\mathbb{F}_p)$ нь том анхны тоо n-д хуваагдахаас гадна практик нууцлалаас хамаарч хамгийн багадаа $n > 2^{160}$ тоо байх ёстой. Бас $\#E(\mathbb{F}_p) = nh$ дахь h- үржигдэхүүн (кофактор) их бага тоо (тухайлбал $h = 1, 2, 3$ бол сайн) байх хэрэгтэй. Энэ нь $\#E(\mathbb{F}_p)$ тоо бараг л анхны тоо байна гэдэг санааг илэрхийлнэ. Тэрчлэн олон довтолгооноос хамаарч $\#E(\mathbb{F}_p) \neq p$ байх ба n анхны тоо $p^k - 1$-ийг $(1 \leq k \leq c)$ хуваахгүй бөгөөд c-тоо хангалттай том, $\mathbb{F}_{p^c}^*$-гийн хувьд ECDLP-ын довтолгоонд оромгүй хүнд байхаар авна, практикт $n > 2^{160}$ үед $c = 20$ байхад боломжтой байдаг.

2.4 Эллиптик муруйн криптосистем

Практикт нийтийн түлхүүрт шифр схем нь нууц түлхүүрт буюу симметр түлхүүрт схемээс илүү удаан ажиллагаатай тул ихэнхдээ зөвхөн бага хэмжээний өгөгдөл (картын дугаар, PIN код, дамжуулагдах симметр түлхүүр /session key/ зэрэг)-ийг шифрлэхэд хэрэглэдэг.
Эллиптик муруйн нийтийн түлхүүрт бүх криптосистемүүд нь ECDLP-д суурилна. Иймд бид сайн криптосистемийг загварчлах нь ECDLP нь найдвартай байх аюулгүй байдлын шаардлагыг бүрэн хангасан сайн домайн параметрийг үүсгэх явдал юм.

2.4.1 Эллиптик муруйн түлхүүрийн хос үүсгэх

Эллиптик муруйн криптосистемд түлхүүрийн хос (d, Q)-г эллиптик муруйн домайн (анхны) параметрүүдийг үндэслэн сонгоно. Энд эллиптик муруйн системийн домайн параметр $\mathcal{D} = (p, a, b, G, n, h)$ нь өмнөх сэдвүүдэд дурдсан бүх шаардлагуудыг хангасан байхаар өөрсдийн найдвартай аюулгүй параметр байдлаар бид үүсгэх нь хамгийн чухал. $G \in E(\mathbb{F}_p)$ цэгээр төрөгдсөн $\langle G \rangle$ бүлгээс санамсаргүйгээр сонгосон Q цэг нь ***нийтийн түлхүүр*** болно. Харгалзах ***далд түлхүүр*** нь $Q = dG$ байх $d \in [1, n-1]$ тоо байна.

Практикт түлхүүрийн хосыг хэрхэн сонгох (Алгоритм 3), үнэн сонгогдсон эсэхийг хэрхэн шалгах (Алгоритм 4) тухай авч үзье.

Algorithm 3 Түлхүүрийн хос үүсгэх

Оролт: $\mathcal{D} = (p, a, b, G, n, h)$ -домайн параметр
Гаралт: нийтийн түлхүүр Q, далд түлхүүр d.

1: **procedure KeyGen**($\mathcal{D}$)
2: $d \xleftarrow{\text{R}} [1, n-1]$ тоог санамсаргүйгээр сонгоно
3: $Q \longleftarrow dG$ утгыг тооцно
4: **return** (Q, d)

Нийтийн түлхүүр Q-ээс хувийн түлхүүр d-г тооцоолох бодлого нь ECDLP болно. Тиймд домайн параметр $\mathcal{D}$-ийг ECDLP нь маш хүнд байхаар сонгох нь хамгийн чухал. Цаашлаад, үүсгэж байгаа d тоо нь хүрэлцээтэй бага, санамсаргүй байх нь чухал бөгөөд энэ нь гуравдагч этгээд өөрийн хайх стратегиэ оновчтой болгох замаар ашигтай байдал үүсгэхээс урьдчилан сэргийлнэ.

2.4.2 Нийтийн түлхүүрийг шалгах

Нийтийн түлхүүрийн шалгалтын зорилго нь тодорхой арифметик шинж чанарыг хангадаг болохыг шалгаж баталгаажуулах явдал юм. Нийтийн түлхүүрийг баталгаажуулснаар түүнд харгалзах хувийн түлхүүр оршин байгаа болохыг харуулж байгаа хэдий ч өөр ямар нэг этгээд уг хувийн түлхүүрийг тооцоолсон эсвэл түлхүүр эзэмшигч үнэхээр уг хувийн түлхүүртэй эсэхийг шалгаж байгаа хэрэг биш юм. Нийтийн түлхүүрийн шалгалт нь Диффи-Хеллман (DH)-ий схемд тулгуурлагдсан протоколуудад нэн чухал хэрэгтэй байдаг. Диффи-Хеллманий протоколд A этгээд нь нөгөө B этгээдээс авсан нийтийн түлхүүр болон өөрийн хувийн түлхүүрийг хольж хамтын нууц түлхүүр k-г гарган аваад дараа нь түүнийгээ симметр протокол (мэдээг нууцлах эсвэл баталгаажуулах г.м.)-д ашигладаг билээ. B этгээд буруу нийтийн түлхүүр сонгосон тохиолдолд k-г хэрэглэх нь A-гийн хувийн түлхүүрийн талаарх мэдээллийг алдаж болзошгүй юм.

Algorithm 4 Нийтийн түлхүүрийг шалгах

Оролт: $\mathcal{D} = (p, a, b, G, n, h)$ -домайн параметр, Q - нийтийн түлхүүр.
Гаралт: Q түлхүүрийн хүчинтэй эсэхийг "**зөвшөөрсөн**"эсвэл "**татгалзсан**"

1: **procedure PubKeyValid**($\mathcal{D}, Q$)
2: $Q \neq \infty$ болохыг шалгах.
3: x_Q, y_Q координатууд $\mathbb{F}_p$ талбарын элементээр дүрслэгдсэн болохыг шалгах.
4: $Q \in E(\mathbb{F}_p)$-г шалгах (ө.х, $y_Q^2 = x_Q^3 + ax_Q + b$).
5: $nQ = \infty$ болохыг шалгах.
6: **if** шалгалт биелэвэл **then**
7: **return** "зөвшөөрсөн"
8: **else**
9: **return** "татгалзсан"

2.4.3 Эллиптик муруйн Диффи-Хэллмэнийн түлхүүр солилцох протокол - ECDH

Бид 2.4.2 дээр эллиптик муруйн түлхүүрийн хос үүсгэсэн. Харин энэхүү протокол нь хэрэглэгчид хэрхэн өөрсдийн нийтийн түлхүүрийн тусламжтайгаар **дундын нууц түлхүүр** (shared secret key) үүсгэх талаар ярилцах юм. Өөрөөр хэлбэл эллиптик муруйн Диффи-Хэллмэнийн түлхүүр солилцох протокол нь хоёр хэрэглэгч хоорондоо харилцах шаардлагатай үед зөвхөн өөрсдийн үүсгэсэн нийтийн түлхүүрийг бие биедээ дамжуулснаар дундын ижилхэн түлхүүр гаргаж авах тухай юм.

2.4.1-ийн Algorithm 3 ашиглан хэрэглэгчид (Q, d) болон (Q', d') гэх хос түлхүүр харгалзан үүсгэсэн гэе. Талууд ижил домайн параметрууд ашиглан өөрсдийн түлхүүрүүдийг үүсгэсэн учраас дундын нууц түлхүүрээ дараах байдлаар тооцоолно.

$$dQ' = dd'G = d'dG = d'Q$$

Өөрөөр хэлбэл эхний хэрэглэгчийн тооцоолсон дундын нууц түлхүүр dQ' болон нөгөө хэрэглэгчийн тооцоолсон дундын нууц түлхүүр d'Q нь хоорондоо ижил байх юм.

2.4.4 Эллиптик муруйн тоон гарын үсэг (ECDSA)

Тоон гарын үсгийн схем нь гараар зурагддаг бичгэн гарын үсгийн тоон хувилбар болох ойлголт юм. Энэ нь мэдээллийн эхийн баталгаажуулалт, бүрэн бүтэн байдал, үл татгалзах байдлыг батлахад хэрэглэгддэг. Гарын үсгийн схем нь ихэвчлэн тухайн тал, түүний нийтийн түлхүүр хоёрыг холбох сертификатад гарын үсэг зурахад итгэмжлэгдсэн сертификатжуулагч талаар хийгдэнэ.
Эллиптик муруйн тоон гарын үсгийн алгоритм (The Elliptic Curve Digital Signature Algorithm - ECDSA) нь тоон талбар дээрх гарын үсгийн алгоритм (DSA)-ын эллиптик муруй дээрх аналоги хувилбар юм. ECDSA нь хамгийн өргөн стандартчилагдсан эллиптик муруйд суурилсан гарын үсгийн схем ба ANSI X9.62, FIPS 186-2, IEEE 1363-2000, ISO/IEC 15946-2 зэрэг стандартуудад тусгагдсан байдаг. ECDSA алгоритмд бид өөрсдийн үүсгэсэн стандарт бус домайн параметрийг ашигласнаар аюулгүй, найдвартай тоон гарын үсгийг ашиглах боломжтой болох юм.
Гарын үсэг зурах нь Алгоритм 5-аар өгөгдөнө. Алгоритмд криптографик хеш функц -ийн гаралт нь n-ээс ихгүй бит урттай байна. Гарын үсэг шалгах нь Алгоритм 6-аар өгөгдөнө:

Algorithm 5 ECDSA гарын үсэг зурах

Оролт: $\mathcal{D} = (p, a, b, G, n, h)$, M -мэдээ, d хувийн түлхүүр.
Гаралт: Гарын үсэг (r, s)

1: **procedure SignECDSA**$(\mathcal{D}, M, d)$
2: $k \xleftarrow{\mathrm{R}} [1, p-1]$
3: $Z = (x_Z, y_Z) \leftarrow kG$ утгыг тооцох;
4: $r \leftarrow x_Z \mod n$
5: **if** $r = 0$ **then** 1-р алхамд шилжих;
6: $e \leftarrow HASH(hlen, M)$
7: $s \leftarrow k^{-1}(e + dr) \mod n$ тооцох;
8: **if** $r = 0$ **then** 1-р алхамд шилжих;
9: **return** (r, s).

Algorithm 6 ECDSA гарын үсэг шалгах

Оролт: $\mathcal{D} = (p, a, b, G, n, h)$, M -мэдээ, Q нийтийн түлхүүр, гарын үсэг (r, s).

Гаралт: Гарын үсгийг "**зөвшөөрөх**"эсвэл "**татгалзах**".

procedure VerifyECDSA$(\mathcal{D}, M, Q)$
  if $r, s \notin \{1, ..., p-1\}$ **then**.
    return "татгалзах".
  $e \leftarrow HASH(hlen, M)$ тооцох.
  $w \leftarrow s^{-1} \mod n$.
  $u_1 \leftarrow ew \mod n$.
  $u_2 \leftarrow rw \mod n$.
  $Z \leftarrow u_1G + u_2Q$. ▷ $Z = (x_Z, y_Z)$
  if $X = \infty$ **then**
    return "татгалзах".
  $v \leftarrow x_Z \mod n$.
  if $v = r$ **then**
    return "зөвшөөрөх".
  else
    return "татгалзах".

Гарын үсгийн шалгалт нь хэрхэн ажиллахыг баталья: M мэдээний хувьд үүсгэгдсэн гарын үсэг (r, s) бол $s \equiv k^{-1}(e + dr)(mod n)$ болно. Эндээс дараах биелнэ.

$$k \equiv s^{-1}(e + dr) \equiv s^{-1}e + s^{-1}rd \equiv we + wrd \equiv u_1 + u_2d(mod n).$$

Иймд $X = u_1G + u_2Q = (u_1 + u_2d)G = kG$ болох ба $v = r$ биелэх юм.

3 Судалгааны хэсэг

3.1 Сигнал (Signal app) протокол

Бид эллиптик муруйн криптосистем болон түүний аюулгүй байдлын талаар онолын хувьд мэддэг боллоо. Харин одоо тухайн криптосистемийг практикт хэрхэн хэрэгжүүлдэг талаар судлах үүднээс сигнал апп-ын **end-to-end** протоколыг товч авч үзье.

Энд **2.4.1** дагуу эллиптик муруйн криптографын арга дээр үндэслэн гаргаж авсан хос (нийтийн түлхүүр, нууц түлхүүр) түлхүүрийг **EC(25519)**, **2.4.4** дагуу эллиптик муруйн гарын үсгийн баталгаажуулалт хийж буй муруй болон түүн дээр үүссэн хос түлхүүрийг **ed25519**, харин **зурвас** гэж нууцлах шаардлагатай өгөгдлийг (message) нэрлэв.

Нийтийн түлхүүрийн төрлүүд

IdentityKey - хэрэглэгчийн бүртгэл үүсгэх үед үүсэж урт хугацаанд ашиглагдах EC(25519) түлхүүр.

SignedPreKey - хэрэглэгчийн бүртгэл үүсэх үед үүсэж дунд хугацаанд ашиглагдах EC(25519) түлхүүр. IdentityKey ашиглан гарын үсгээр баталгаажуулсан байна.

OneTimePreKeys - EC(25519) хэлбэрийн түлхүүрүүдийн дараалал бөгөөд тухайн аг-шинд хэрэглэгдээд устана.

Холболт тогтоох үеийн түлхүүрийн төрлүүд

RootKey - ChainKey үүсгэх үед ашиглах 32 байт урттай түлхүүр.

ChainKey - Зурвас бүрт харгалзах түлхүүр (MessageKey) үүсгэх үед ашиглах 32 байт урттай түлхүүр.

MessageKey – Зурвасыг шифрлэх түлхүүр. 80 байт урттай байх бөгөөд 32 байт нь тухайн нууцлах алгоритмд ашиглагдах түлхүүр, 32 байт нь HMAC-SHA256 алгоритмд ашиглагдах бөгөөд үлдсэн 16 байт нь IV буюу шифрлэх үед ашиглах анхны оролтын утга нь байна.

Криптографын үйлдлүүд

ECDH - Эллиптик муруйн Диффи-Хэллмэнийн түлхүүр солилцох протокол (**2.4.3** дагуу)

HMAC - Криптографын хеш функц дээр суурилсан баталгаажуулалтын алгоритм

SHA256 - Криптографын хеш функц

HKDF - HMAC-SHA256 алгоритм дээр суурилсан түлхүүр үүсгэх функц

Хэрэглэгч тухайн системд бүртгүүлэх үед нийтийн IdentityKey, нийтийн SignedPreKey (баталгаажуулсан тоон гарын үсгийн хамт) болон нийтийн OneTimePreKeys түлхүүрийг сервэрт илгээнэ. Сервэр нь тухайн хэрэглэгчийн илгээсэн түлхүүрүүдийг харгалзах төхөөрөмжийн мэдээллийн хамт хадгална. Энд анхаарах зүйл нь хэрэглэгч зөвхөн өөрийн үүсгэсэн нийтийн түлхүүрүүдийг илгээж байгаа бөгөөд нууц түлхүүрүүд нь өөрт нь үлдэнэ. Өөрөөр хэлбэл **нууц түлхүүрийг сүлжээгээр аль нэг тийш огт дамжуулахгүй**, тухайн төхөөрөмж дээрээ хадгална. End-to-end нууцлалын протоколын **амин гол сүнс** нь ердөө энэ юм. Нууц түлхүүрийг ямар нэг байдлаар сүлжээгээр дамжуулбал энэ нь end-to-end протокол биш болно.

3.1.1 Илгээгч тал (Sender)

Хэрэглэгчид хоорондоо нууцлалтай, аюулгүйгээр өгөгдөл солилцохын тулд эхлээд илгээгч тал нь хүлээн авагч (recipient) талтай холболт тогтоох **дундын нууц түлхүүрийг** (`2.4.3 дагуу`) үүсгэдэг. Мэдээж хүлээн авагч тал нь олон төхөөрөмжөөс орсон үед тухайн төхөөрөмж бүрийн түлхүүрүүд дээр үндэслэн дундын нууц түлхүүртэй болох ёстой бөгөөд үүнийг түр орхиё.

Холболт тогтоох түлхүүр үүсгэхийн тулд тухайн харилцах гэж буй хэрэглэгчийн нийтийн IdentityKey, нийтийн SignedPreKey (баталгаажуулсан тоон гарын үсгийн хамт) болон нийтийн OneTimePreKeys–ийг сервэрээс татаж авна. Дараагаар нь X3DH (triple extended Diffie-Hellman) гэх алгоритмыг ажиллуулдаг бөгөөд доорх тооцооллуудыг хийнэ. Мэдээж SignedPreKey нь тухайн гарын үсгээр баталгаажиж буй эсэхийг шалгах бөгөөд зөвхөн баталгаажсан үед л дараахийг хийхийг анхаарна уу.

Юуны өмнө, илгээгч тал EphemeralKey гэх (түр зуурын) EC(25519) хос түлхүүр үүсгэнэ. Дараагаар нь **дундын мастер нууц түлхүүрийг** (MasterSecretKey) тооцоолно.

$$S1 = ECDH(IdPriKey, SPpubKey)$$

$$S2 = ECDH(EphPriKey, IdPubKey)$$

$$S3 = ECDH(EphPriKey, SPpubKey)$$

$$S4 = ECDH(EphPriKey, OTPreKey)$$

$$MasterSecretKey = HKDF(S1||S2||S3||S4)$$

Энд IdPriKey - IdentityKey-ийн нууц түлхүүр, SPpubKey - SignedPreKey-ийн нийтийн түлхүүр, EphPriKey - EphemeralKey-ийн нууц түлхүүр, IdPubKey -IdentityKey-ийн нийтийн түлхүүр, OTPreKey - OneTimePreKeys-ийн нийтийн түлхүүр болно. Өөрөөр хэлбэл ашиглагдаж буй нийтийн түлхүүрүүд нь харилцах гэж буй хэрэглэгчийн сервэрээс татаж авчирсан нийтийн түлхүүрүүд байх бөгөөд нууц түлхүүрүүд нь өөрийн түлхүүрүүд(private key) байх юм.

Хэрэв OTPreKey ашиглаагүй бол сүүлийн S4-г тооцохгүй.

Илгээгч тал HKDF функцээ ашиглан RootKey болон ChainKey–ийг MasterSecretKey-ээ оролтын утга болгон ашиглаад тооцоолон гаргана. Энд анхаарах зүйл бол илгээгч тал **түр зуурын түлхүүр** (EphemeralKey) ашиглан дундын мастер нууц түлхүүрийг үүсгэсэн бөгөөд хүлээн авагч тал тухайн түр зуурын түлхүүрийг мэдэхгүйгээр дундын мастер нууц түлхүүрээ үүсгэж чадахгүй.

3.1.2 Хүлээн авагч тал (Recipient)

Илгээгч тал дээрх тооцооллыг хийж дундын мастер нууц түлхүүр гарган авснаар хүлээн авагч талыг холбогдох боломжтой байгаа эсэхээс үл хамааран зурвасыг илгээж чадах юм. Зурвасыг илгээхээс өмнө тухайн зурваст ашигласан **нийтийн** EphemeralKey болон хэрэв ашигласан бол OneTimePreKey зэргийг хавсарган явуулна.

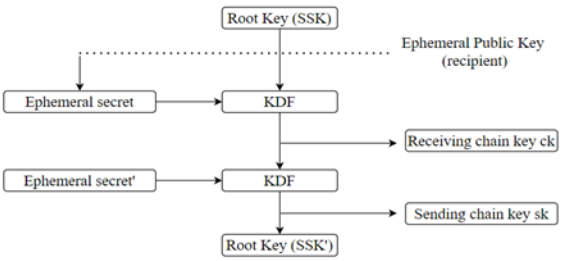
Хүлээн авагч тал зурвас ирсэн үед тухайн зурваст хавсаргасан **нийтийн** түлхүүр, тухайн хэрэглэгчийн нийтийн түлхүүрүүд болон өөрийн нууц түлхүүрийг ашиглан MasterSecretKey буюу дундын мастер нууц түлхүүрийг тооцоолно. Түүний дараагаар HKDF функцээ ашиглан RootKey болон ChainKey-ийг гарган авна. Мэдээж зөв тооцоолол хийгдсэн үед хоёр тал ижил MasterSecretKey гаргаж авсан байх ёстой бөгөөд түүний дараах түлхүүрүүд нь ч (RootKey, Chainkey, MessageKey) мөн ижил байна.

3.1.3 RootKey ашиглан ChainKey тооцоолох

Зурвас илгээгдэх үед тухайн зурваст ашиглагдсан түр зуурын нийтийн түлхүүр (EphemeralKey) хавсаргагдан явдаг. EphemeralKey-ийг хэдий хугацаанд ашиглахаас хамаарч дараах үйлдлүүд хийгдэх буюу RootKey болон ChainKey-ийг EphemeralKey өөрчлөгдөх бүрт дараах байдлаар тооцоолно.

$$EphemeralSecret = ECDH(EphemeralSender, EphemeralRecipient)$$

$$ChainKey, RootKey = HKDF(RootKey, EphemeralSecret).$$



Схем 1. *ChainKey тооцоолох схем.*

3.1.4 ChainKey ашиглан MessageKey тооцоолох

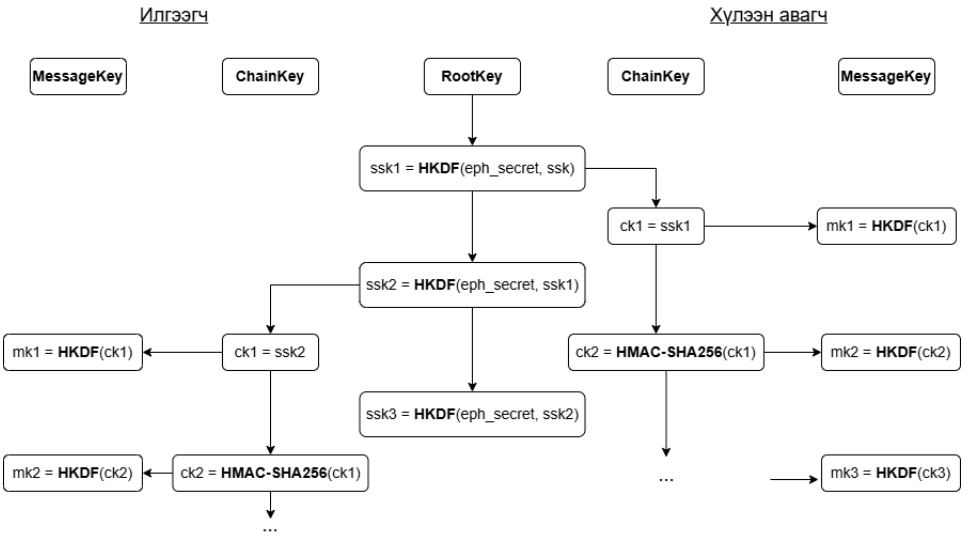
Илгээгч тал нь зурвас илгээх бүртээ ялгаатай MessageKey-ийг дараах байдлаар тооцоолно. Зурвас бүрт ялгаатай MessageKey ашиглах шалтгаан нь криптоанализын элдэв довтолгооноос зайлсхийх зорилготой буюу нууцлагдсан зурваст ямар нэг байдлаар дүгнэлт хийх боломжгүй болгож буй явдал юм. Өөрөөр хэлбэл яг нэг ижил төрлийн зурвасыг удаа дараалан явуулах үед ялгаатай нууцлагдсан зурвасууд бий болгох зорилготой.

$$MessageKey = HMAC\text{-}SHA256(ChainKey, 0x01)$$

Харин ChainKey бүр дараах байдлаар өөрчлөгдөнө.

$$ChainKey = HMAC\text{-}SHA256(ChainKey, 0x02).$$

Одоо тухайн протоколыг ерөнхийлөн харвал дараах байдалтай болно.



Схем 2. *Сигнал протоколын хэрэглэгчид харилцах хэсгийн түлхүүрийн ерөнхий схем.*

3.2 Mazala протокол

Бид эллиптик муруйн криптосистемийг сигнал протокол дээр хэрхэн хэрэгжүүлсэн талаар харлаа. Харин одоо сигнал протоколын үндсэн зарчим дээр суурилсан өөрсдийн протоколын загварын талаар ярилцъя. Энэ нь эллиптик муруйн криптосистем дээр өөрсдийн протоколыг мөн адил загварчлах, хэрэглээнд ашиглах боломжтой гэдгийг харуулах зорилготой юм. Бид энэхүү алгоритмдаа **Mazala** хэмээх нэр өгсөн.

Нийтийн түлхүүрийн төрлүүд

IdentityKey - хэрэглэгчийн бүртгэл үүсгэх үед үүсэж урт хугацаанд ашиглагдах EC(25519) түлхүүр.

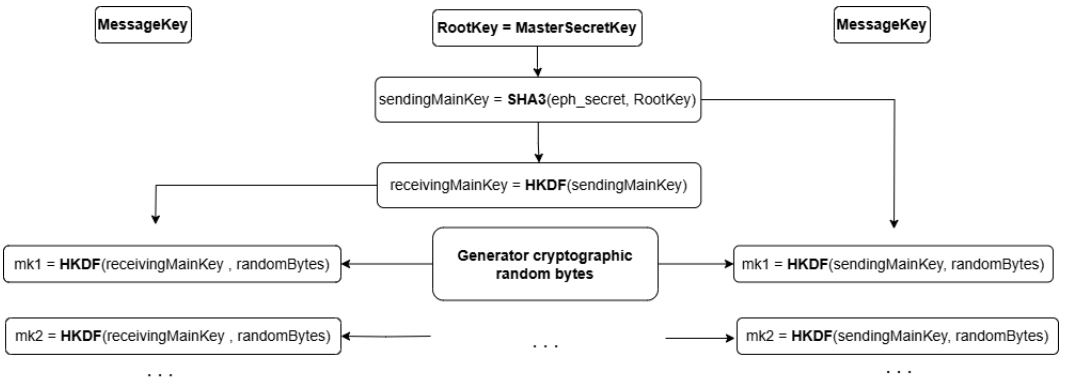
SignedPreKey - хэрэглэгчийн бүртгэл үүсэх үед үүсэж дунд хугацаанд ашиглагдах EC(25519) түлхүүр. IdentityKey ашиглан гарын үсгээр баталгаажуулсан байна.

EphemeralKey - богино хугацаанд ашиглагдах EC(25519) түлхүүр. Долоо хоногт эсвэл сар бүр солих боломжтой.

Тус протокол дээр сигнал протоколын **дундын мастер нууц түлхүүр** (MasterSecretKey) тооцоолох хүртэл ижил буюу дээрх түлхүүрүүдийн хувьд тооцоолно.

Схем 1-д ChainKey гаргаж авах үед EphemeralKey ашиглаж буй хэсэг нь тухайн протоколын хамгийн чухал хэсгийн нэг бөгөөд ямар нэг байдлаар MasterSecretKey-ийг (Схем 1-д SSK гэж тэмдэглэв) гуравдагч этгээд (хакер) мэдсэн ч тухайн зурвасыг унших боломжийг хааж байгаа юм. Өөрөөр хэлбэл MasterSecretKey-ийг зөвхөн мэдээд хангалтгүй, EphemeralKey-ийг ч мөн мэдэх шаардлагатай хэмээн тулгаж буй бөгөөд тухайн EphemeralKey нь тодорхой хугацаанд хэрэглэгчид хооронд өөрчлөгдөж байхаар загварчлагдсаныг бид дээр харсан.

Mazala протоколын хувьд мөн тус шийдлийг өөрсдийн протоколдоо оруулсан бөгөөд харин MessageKey үүсгэхийн тулд дахин шинэ ChainKey үүсгэх, дараагийн MessageKey нь өмнөх MessageKey-ийг ашигласан HKDF функцийн үр дүн байх зэрэг нэг зурвас илгээхийн тулд тооцоологдож буй эдгээр үйлдлүүдийг багасгахыг зорьж дараах байдлаар шийдсэн.



Схем 3. *Mazala протоколын MessageKey үүсгэх ерөнхий схем.*

Тодруулбал, илгээгч тал зурвас илгээх бүртээ криптографын санамсаргүй байдлаар үүсгэгдсэн 16 урттай байтуудыг (randomBytes) ашиглах бөгөөд энэ нь мөн MessageKey бүр дахин давтагдашгүй байна гэсэн зарчимыг дагаж буй юм. Мөн сүлжээгээр дамжиж буй өгөгдлийн хэмжээг багасгах зорилготой.

Энд **SHA3** гэж криптографын хещ алгоритмыг ашигласан бөгөөд тус алгоритм нь сүүлийн үеийн хүчирхэг алгоритмуудын нэг нь юм.

Харин хүлээн авагч тал нь тухайн ирсэн зурвасаас EphemeralKey болон дээрх байтуудыг гаргаж авах бөгөөд цааших үйлдлүүд нь өмнө нь бидний үзсэн протоколын дагуу ажиллана. Гагцхүү бид MessageKey тооцоолох үед хавсаргагдан ирсэн дээрх байтуудыг ашиглах бөгөөд практик хэрэгжүүлэлтийн хувьд үр ашигтай, ажиллах хурдыг сайжруулахыг зорьсон.

Тухайн зурвасыг нууцлах алгоритмыг бид өөрсдийн хувилбараар сонгох бүрэн боломжтой. Одоогийн байдлаар бид AES-256 алгоритмыг түр сонгосон байгаа.

$$EncryptedMessage = AES(message, MessageKey)$$

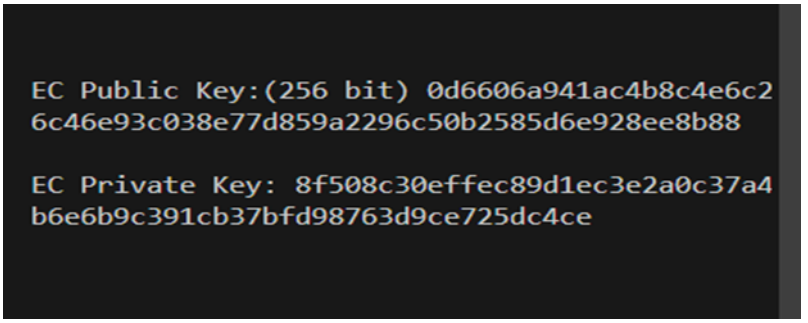
Тус протоколд бид эллиптик муруйн криптосистемийн аюулгүй байдлыг бууруулахтай холбоотой өөрчлөлт огт хийгээгүй бөгөөд чухал хэсгүүд болох MasterSecretKey тооцоолох, EphemeralKey ашиглах зэргийг хэрэгжүүлж буйг давтан хэлье.

4 Хэрэгжүүлэлт - туршилт

Бид **https://www.npmjs.com/package/elliptic** гэх нээлттэй санг өөрсдийн туршилт, хэрэгжүүлэлтдээ ашигласан. Доорх туршилтууд нь Сигнал болон Mazala протоколуудад ашиглагдаж буй гол үйлдлүүдийг дээрх санг ашиглан туршиж буй бөгөөд end-to-end нууцлалтай чат системийн хэрэгжүүлэлтдээ тухайн байдлаар ашиглахыг харуулж буй юм.

4.1 Эллиптик муруйн криптографын хос түлхүүр

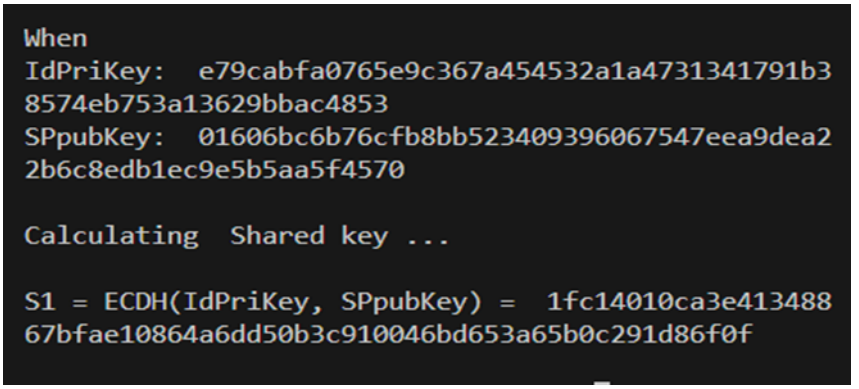
Зураг 1-д EC(25519) хос түлхүүрийг `2.4.1` дагуу үүсгэж 16-тын тооллын системээр дүрсэлж харуулсан бөгөөд нийтийн түлхүүр нь 256 бит урттай байна. Энэ нь IdentityKey, SignedPreKey, EphemeralKey болон OneTimePreKeys нь дараах хэлбэртэй хос түлхүүрүүд байхыг илтгэнэ.



Зураг 1. *EC(25519) дээр үүсгэсэн хос түлхүүр.*

4.2 ECDH тооцоолох

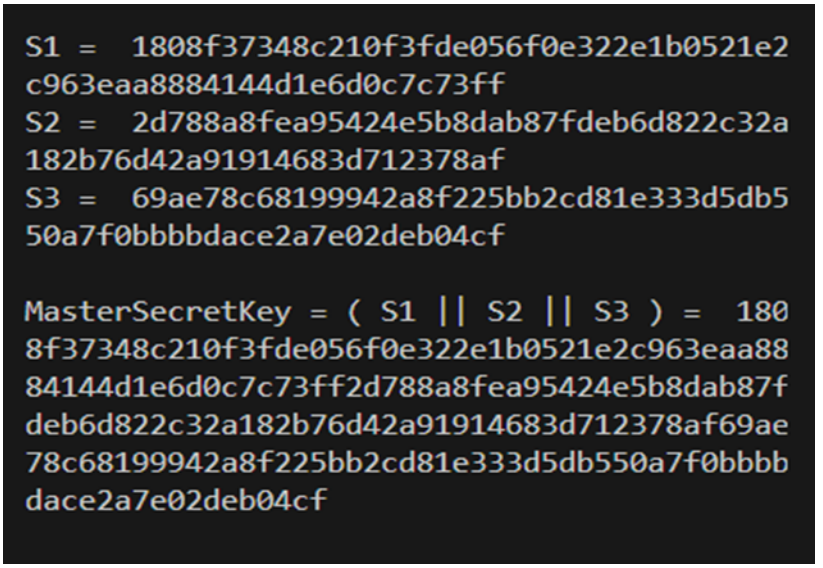
Зураг 2-д IdPriKey буюу IdentityKey-ийн нууц түлхүүр харин SPpubKey нь SignedPreKey-ийн нийтийн түлхүүр бөгөөд **2.4.3** дагуу S1 гэх **дундын нууц түлхүүр** үүсгэж байна. Өөрөөр хэлбэл нэг хэрэглэгч нөгөө талын нийтийн түлхүүрийн тусламжтайгаар тухайн хэрэглэгчтэй харилцах боломжтой дундын нууц түлхүүрээ өөрийн нууц түлхүүрүүдээ ашиглан үүсгэж байна.



Зураг 2. *Дундын нууц түлхүүр үүсгэж байна.*

4.3 Дундын нууц мастер түлхүүр тооцоолох

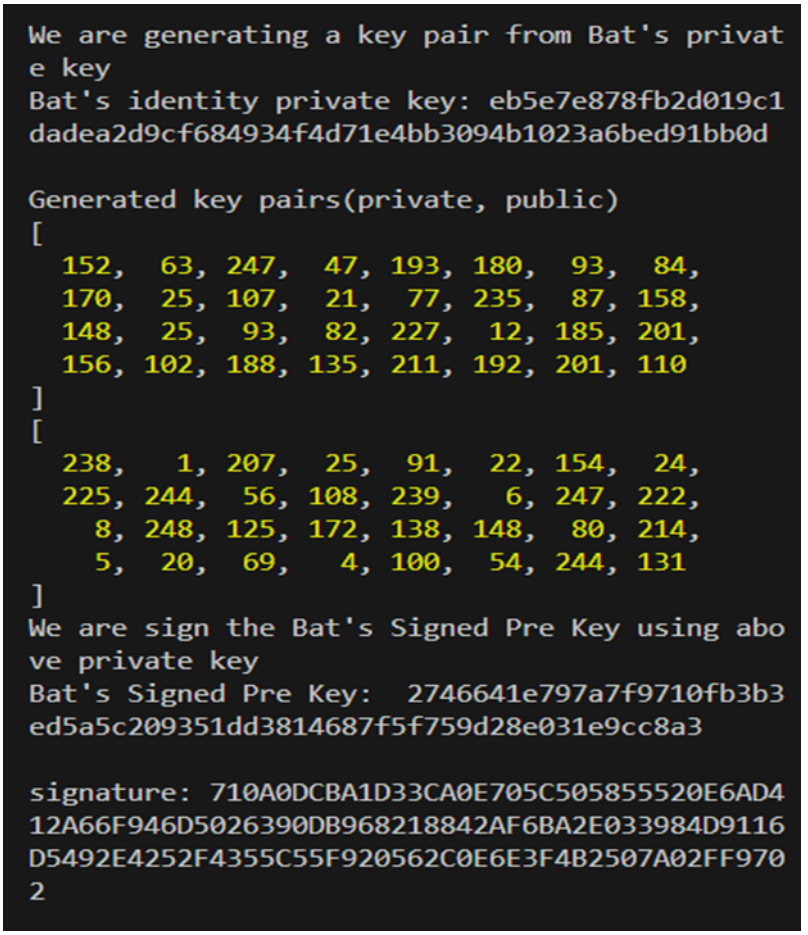
Зураг 3-д бид дээрх аргаар S1, S2, S3-ыг тооцоолсон бөгөөд OneTimePreKeys ашиглаагүй горимоор нь туршсан учир S4 байхгүй байна.



Зураг 3. *MasterSecretKey тооцоолж байна.*

4.4 Гарын үсгээр баталгаажуулах

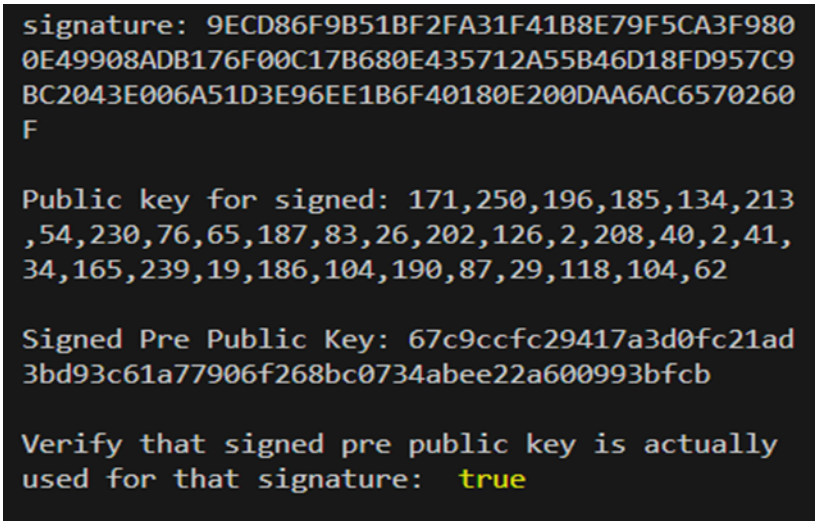
Гарын үсгийн алгоритм нь **ed25519** төрлийн муруй дээр хийгддэг бөгөөд дараах байдалтай байна. Зураг 4-д илгээж буй тал нь өөрийн нууц түлхүүр IdentityKey ашиглан ed25519 төрлийн эллиптик муруй дээр хос түлхүүр үүсгэж байгаа бөгөөд үүссэн нууц түлхүүрээ ашиглан өөрийн ашиглаж буй нийтийн SignedPreKey түлхүүрээ `2.4.3` дагуу баталгаажуулж байна.



Зураг 4. *Өөрийн SignedPreKey-ийг гарын үсгээрээ баталгаажуулж байна.*

4.5 Гарын үсэг шалгах

Зураг 5-д хүлээн авагч нь илгээгчийн нийтийн түлхүүрүүд болон гарын үсгийн тусламжтайгаар тухайн SignedPreKey үнэхээр илгээгчийнх мөн эсэхийг `2.4.3` дагуу шалгаж байна.



Зураг 5. *SignedPreKey-ийг шалгаж байна.*

4.6 HKDF тооцоолох

Сигнал протокол дээр HKDF функцийн оролтын утгууд нь дараах хэлбэртэй байдаг.

- Ikm (input key material) = F || KM, энд KM нь дундын нууц түлхүүр бөгөөд F нь 0xFF хэлбэрийн 32 байт урттай байна.
- Salt – Гаралтын утгын урттай тэнцүү 0x00 утгатай байтууд байна.
- Info – Тухайн системийг төлөөлсөн ASCII тэмдэгт байж болно.

```
F: ffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffff

IKM = F||KM (KM is our master secret key):  f
ffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffff752ea2d76f57b3cd466f889f615
f484e7821b3a88313eeab590c5af1a1ff608e277efb14
0021f97e9fdc37ebccd9202a7896799705757 00b8a58b
6d5a3263d4557d9fffce1934eecec15c3574c6e0ef17e
038a5f03e3e4a2820148761da1bc79

Salt:  0000000000000000000000000000000000000000
0000000000000000000000000

Info:  Testing HKDF

HKDF(IKM, Salt, Info) =  819cce72ad99723545bb
5ab046643309d93ad9133a8ff885af42ca376dba6c72
```

Зураг 6. *Оролтын утгуудын дагуу HKDF тооцоолж байна.*

Бусад криптографын аргууд (HMAC-SHA256, AES-CBC ...) нь нийтлэг хэрэглэгддэг учраас туршаагүй болно.

5 End-to-end нууцлалтай чат систем

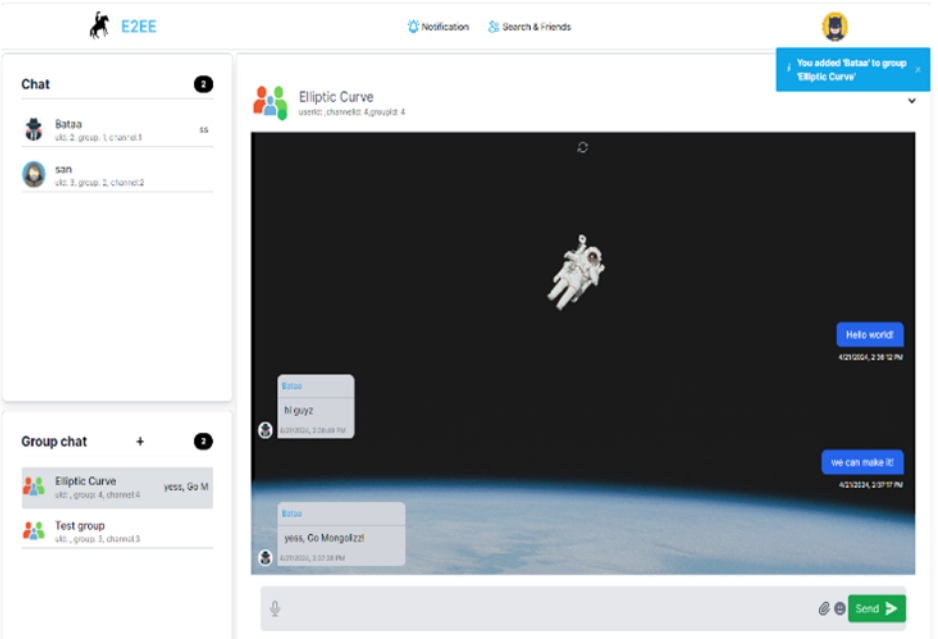
Бид Nextjs, GraphQL, Apollo, Nodejs, Oracle технологиуд ашиглан эллиптик муруйн криптосистемд суурилсан, сигнал протоколын архитектурт үндэслэн өөрсдийн загвараар өөрчилсөн end-to-end нууцлалын **Mazala** хэмээх протоколтой чат системийг бүтээсэн. Энэхүү систем нь https буюу SSL протокол ашиглан сервэр хооронд нууцлагдах(encrypt) бөгөөд бид дунд нь өөрийн протоколоо оруулснаар өгөгдлийн нууцлалын хувьд илүү найдвартай систем болох юм.

Бидний өдөр тутамдаа ашигладаг **https://www.facebook.com**-ын чат систем (Messenger) нь яг энэ зарчимаар ажилладаг.

Одоогоор манай вэб систем нь бусад чат системүүдийн гүйцэтгэдэг үндсэн үйлдлүүдийг

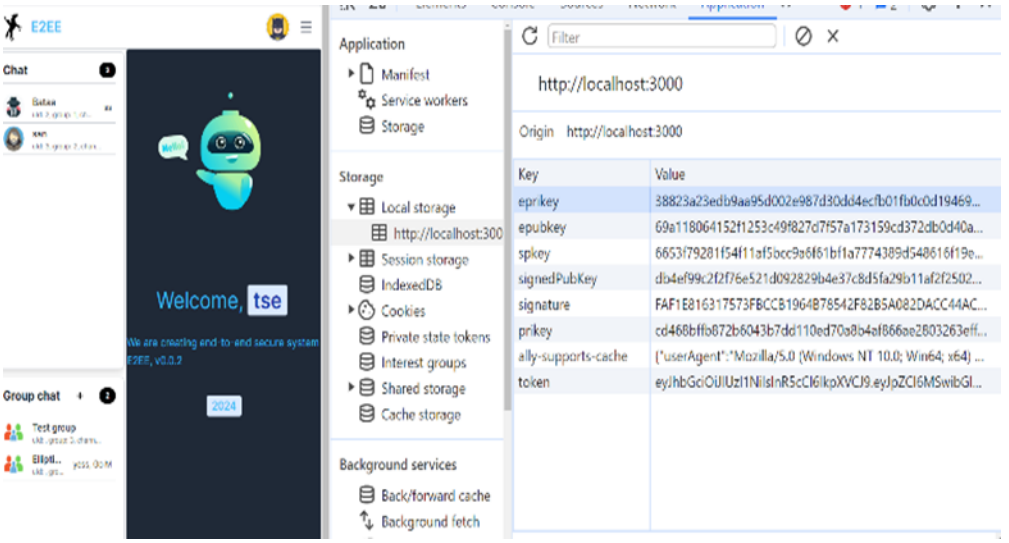
бүгдийг нь гүйцэтгэж буй бөгөөд гар утасны апп болгох хөгжүүлэлт хийж эхэлсэн. Сервэр талдаа бид Nodejs, GraphQL болон Redis ашиглан хэрэглэгчтэй тухайн агшинд нь **web socket** холболт үүсгэж байгаа бөгөөд MySQL, Oracle өгөгдлийн сан ашиглаж байна.

Манай системийн хувьд хэрэглэгчдийн хоорондох хувь чат end-to-end протоколоор нууцлагдаж байгаа бөгөөд групп чатын хувьд end-to-end протокол ажиллах зарчим нь асар төвөгтэй, бусад системүүд (viber, signal, telegram) өөр өөрийн аргаар хэрэгжүүлэх гэж оролдсон байдаг (бүрэн шийдэгдээгүй асуудал) тул бид орхисон болно.



Зураг 7. *Манай систем дээр хэрэглэгчид зурвас солилцож байна.*

Бид хэрэглэгчийг системд харгалзах нэр, нууц үгээр нэвтрэх үед түлхүүрүүдийг нь үүсгэж байгаа бөгөөд тухайн түлхүүрүүдийн тусламжтайгаар өгөгдлийг нууцлах (encryption) үйлдэл хийгдэнэ.



Зураг 8. *Хэрэглэгчийн түлхүүрүүд хөтөч(browser) дээр үүссэн байдал.*

Үнэндээ хэрэглэгчийн эмзэг мэдээлэл, банкны картын мэдээлэл, криптографын түлхүүр зэргийг вэб хөтөч дээр хаана, хэрхэн хадгалах эсэх нь ярвигтай асуудал байдаг бөгөөд бид Зураг 8-д localStorage дээр хадгалсан болно.

Тухайн хэрэглэгчийн бүхий л мэдээллүүд сервэр дээр хадгалагдаж байгаа бөгөөд end-to-end нууцлал нэвтрүүлэх гол шаардлага нь тухайн харилцаж буй хоёр хүнээс бусад хэн ч (сервэр ч мөн адил) тухайн зурвасыг унших боломжгүйд байгаа юм. Учир нь бид хувийн нууц түлхүүрүүдийг нь огт сүлжээгээр дамжуулаагүй бөгөөд хэрэглэгч нэвтрэх үед хөтөч дээр үүсгэж хадгалсныг Зураг 8-д харуулсан билээ.

id	identityKey	signedPreKey	signature	signaturePubKey
1	2e112a3c02ffd38cbba300f43cb8ddf8ee0e64959e764d56a...	5f158807d1518d97633bd388999fc949ee9468a4a698937f98...	BC1AD4189D42ECA464B5822663A01A312D14E188D03D49F73C...	14854496c63fe73fe08dd61600d26b5a54ecc82432ea5f8129...
5	3daf2544c7ee6a59613cdcc87859b36d8c02322a62ddf1906a...	325e7f60af0ccf02fa329c3bbb8be02822a18b6cb35b8a19f0...	D05E2D2037CC69AEF8E4A679F3759C5D6790D276EE24A062AB...	e32265d3a2a53894b3ee0f72220855db5fa7c2948da84fb581...
6	030b55fc76f333de4e434540441e5b9e43be084be1dd221273...	5bffe9949ec6f557f3a55bf15ba4bf075892a9a91a0513d0f7...	FCE7AB4DF7B8A7E8E9C207AE73B6FC6E3D4B3F18D5381577C9...	0c728a7a2d144580585a87751753211d05236fab0e3793b994...
7	561e5c3f142ffecc6903c4598bd8e439d15795dcc6f2999985...	22f6dad68ec4b87e95d7181533080f6b27c9e6fbc9418802fd...	B8E3097524A0EBD161DF9DC03443513649 9ED6F07649C0FBD2...	de4df5dfebf6b3b6174128617db41f6d1d459c68a30a6b0593...
8	16daa59b09ffb14e489b76a5329000e492efff78ff41a42a69...	098278e94a3b77ba9e931d47dc70b6eb949f60e6da61ceca53...	33C8079539C32EEF300A9F3BCEF124ED63747AAEC0BEA913EB...	790c941033587e5dd5ebf4b17c1167301108 0fdaa838cdda2...
9	69264ef5c620ec7335908f5446bf0d254af69d579c41894d45...	2033e7918c155a6dc8f1365b629fd933ea5066aacc3135e1c8...	4270F32BA26B78A57401B022F7732709C85F19F859FE00394...	20e766543dd06ad5bef2a526e2246d79b9fe83f30028dc22bd...
10	62496d43c02031a385f977e6be748a36a354180e9c1306c7cd...	75d6a51c4f2add f4341 1d813123929 7b5bfb456dd18e384aaa1...	C31425452F12EA44E11D414583A06BFAB02E6A1B29B795BE36...	f58bfc488957e6b021e1048ebd720afbd0f99b346a37eddbe2...
11	44d8b4aa610416657caef0e3ad2a0e41af7f6fc66cbc09c30e...	0cc2f71cd190f161ad2df85ce729d76f54f957dcad4f0e24ff...	F2F8A6BBE89A2B3CBE58BA546A2675A3DEFB4351D632D06136...	480756b16f121edfb333e764271cfdeb09ca26181b0463403a...
12	59f758bb89e795a878dec5250a4e908cb3d2b6f7d4702137e...	56e0325f2a68a6a05fc77d43ffad40d936165904d5fd0154ba...	45B5F7C2B4389B9A61620101F332F9D74CEDE282F9688EE6BE...	eeca6f8592db61f0114e5043902bd7ca6cda6ba5ba79193912...
13	2a8518e16752fda8ea40d53305f30eb3f3dc4bf1207c03d258...	68e15c38e8ee370c6d7a70f68cff0ce107b033579b7fa160b2...	FAF1E81631757 3FBCCB1964B78542F82B5A082DACC44ACD17C...	db4ef99c2f2f76e521d982829b4e37c8d5fa29b11af2f25020...

Зураг 9. *Хэрэглэгчийн нийтийн түлхүүрүүд сервэр дээр хадгалагдсан байдал.*

6 Дүгнэлт

Бид эллиптик муруйн криптосистем болон сигнал протоколтой холбоотой үндсэн зарчмуудыг харсан бөгөөд тухайн протоколд хийгдэж буй чухал үйлдлүүдийг дэс дараалан туршиж үзэв. Мөн end-to-end нууцлалын протоколыг зөвхөн онолын түвшинд судлаад өнгөрөх бус практикт хэрхэн ажилладаг, үр шимийг нь бодитой харуулах үүднээс Mazala хэмээх end-to-end протоколыг загварчлан хэрэглэгчид харилцах боломжтой чат системийг бүтээсэн нь бидний томоохон ажил юм.

Цаашдаа тухайн системийн гар утасны хувилбарыг боловсруулах, шаардлагатай хөгжүүлэлтүүдийг нэмэлтээр хийх, хэрэглээнд нэвтрүүлэх алсын хараатай билээ.

Ашигласан ном

[1] D.Hankerson, A.Menezes, S.Vanstone.: Guide to Elliptic Curve Cryptography. Springer-Verlag New York, Inc. (2004)

[2] NTT Corporation. PSEC-KEM Specification, v. 2.2, 2008.

[3] Shoup V. A Proposal for an ISO Standard for Public Key Encryption, v. 2.1, preprint 2001.

[4] Cramer R, Shoup V. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J Computing 2003; 33, 1: 167-226

[5] ANSI X9.63. Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography, American National Standards Institute 2001.

[6] IEEE Std 1363a. Standard Specifications for Public Key Cryptography - Amendment 1: Additional Techniques. Institute of Electrical and Electronics Engineers 2004.

[7] ISO/IEC 18033-2. Information Technology – Security Techniques – Encryption Algorithms – Part 2: Asymmetric Ciphers, International Organization for Standardization 2006.

[8] SECG SEC 1. Elliptic Curve Cryptography, v. 2.0. Standards for Efficient Cryptography Group 2009.

[9] Benjamin Dowling, Cas Cremers, Douglas Stebila, Katriel Cohn-Gordon, Luke Garratt, "A Formal Security Analysis of the Signal Messaging Protocol", July 2019.

[10] Meta, Messenger End-To-End encryption overview, December 06 2023.

[11] Signal, Signal protocol, https://signal.org/docs/.

[12] Microsoft, Skype Private Conversation white paper, June 20, 2018.

[13] WhatsApp, WhatsApp encryption overview technical white paper, September 27, 2023.

[14] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis, The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol, In: IACR Cryptology ePrint Archive 2018 (2018), p. 1037. https://eprint.iacr.org/2018/1037

[15] Darrel Hankerson, Alfred Menezes, Scott Vanstone, Guide to Elliptic Curve Cryptography, 2004.

[16] Elliptic package library, https://www.npmjs.com/package/elliptic

[17] Dion van Dam, Analysing the Signal Protocol, V. Moonsamy, C.N.I.W. Schapin, August 21, 2019.

[18] Wire, Wire Security Whitepaper, July 19, 2021