Signal протокол

2024 он

Signal протокол

Signal протокол нь 2 эсвэл түүнээс олон тооны хэрэглэгчид хоорондоо зурвас илгээх болон хүлээн авах үед зурвасын нууцлалыг хангахад зориулж зохиогдсон, нийтэд нээлттэй протокол юм. Хэрэглэгчдийн хооронд дамжиж буй текст, яриа, зураг, бичлэг гэх мэт бүх төрлийн өгөгдлийг **end-to-end** шифрлэлтийн зарчмын дагуу шифрлэж дамжуулах процессыг энэхүү протоколын дагуу хийж гүйцэтгэдэг.

- Whatsapp
- Facebook messenger
- Skype
- Wire
- Signal
- Google Allo

Signal протоколын үе шатууд

1. Бүртгэлийн үе шат
2. Түлхүүр тохиролцож холболт тогтоох үе шат (X3DH)
3. Зурвас илгээж харилцах үе шат (Double Ratchet)

Хэрэглэгчид хамгийн эхэнд бүртгүүлж нэвтрэх үед хэрэглэгч бүрд хос түлхүүрүүд үүсгэгдэнэ.

- Урт хугацааны түлхүүр болох `identity key`
- Дунд хугацааны түлхүүр болох `signed prekey`
- Богино хугацааны түлхүүр болох `one-time prekey`

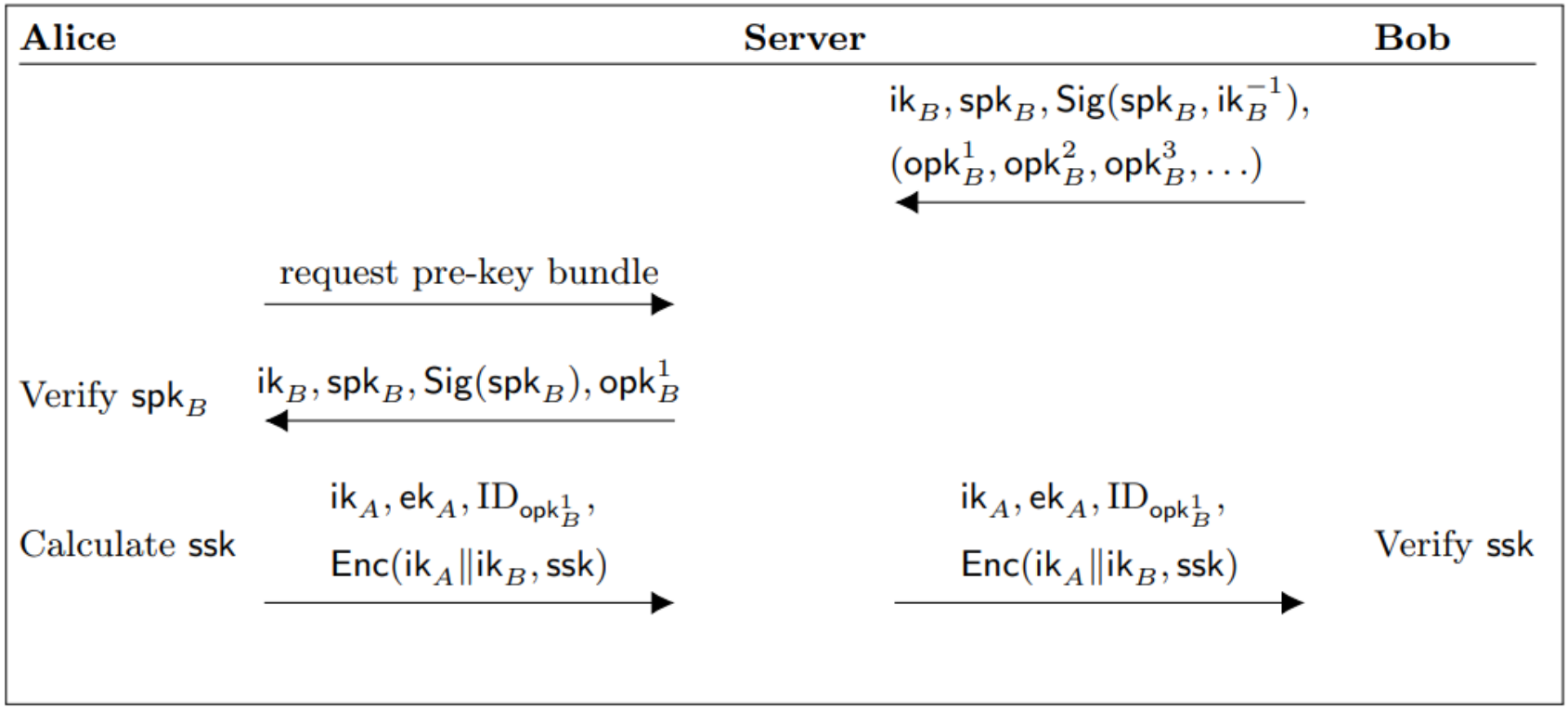
Эдгээр нь бүгд хос түлхүүрүүд байх бөгөөд бүх нийтийн түлхүүрүүдийг серверлүү илгээнэ. Мөн `signed prekey`-д тоон гарын үсэг зурж баталгаажуулан серверт илгээнэ.

Тэмдэглэгээ	Тайлбар
ik_A	Алисын нийтийн `identity key`
ek_A	Алисын нийтийн `ephemeral key`
ik_B	Бобын нийтийн `identity key`
spk_B	Бобын нийтийн `signed prekey`
opk_B	Бобын нийтийн `one-time prekey`

Жишээ: ik_A^{-1}

Функц	Тайлбар
$DH(x, y)$	Диффи Хеллманы түлхүүр солилцох функц
$KDF(x)$	Шинэ түлхүүр үүсгэн авах нэг чигт функц
$Enc(x, y)$	Шифрлэгч функц
$Sig(x, y)$	Тоон гарын үсэг зурах функц

X3DH буюу extended triple Diffie-Hellman



$$k_1 = DH(ik_A^{-1}, spk_B)$$

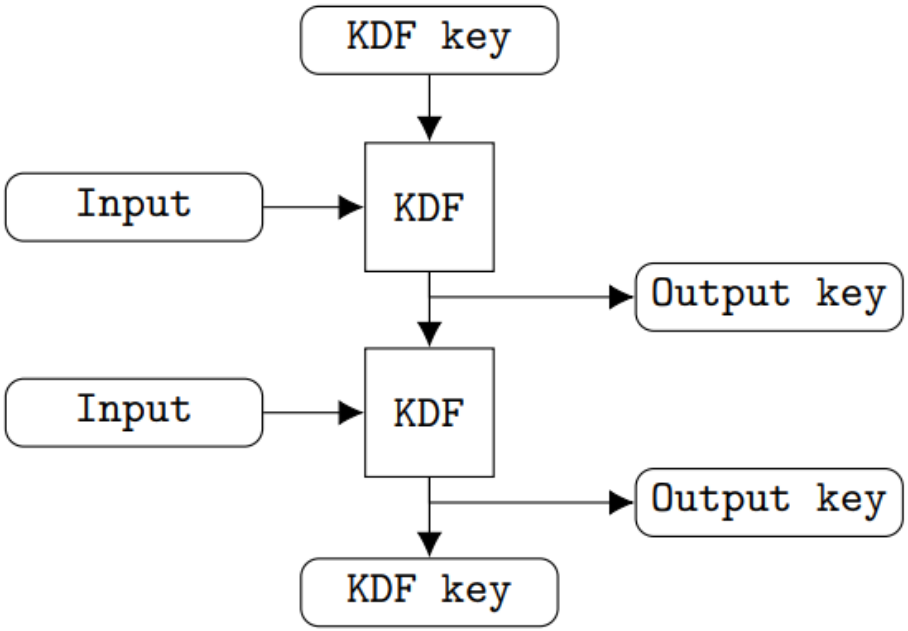
$$k_2 = DH(ek_A^{-1}, ik_B)$$

$$k_3 = DH(ek_A^{-1}, spk_B)$$

$$k_4 = DH(ek_A^{-1}, opk_B^x)$$

$$ssk = KDF(k_1||k_2||k_3||k_4)$$

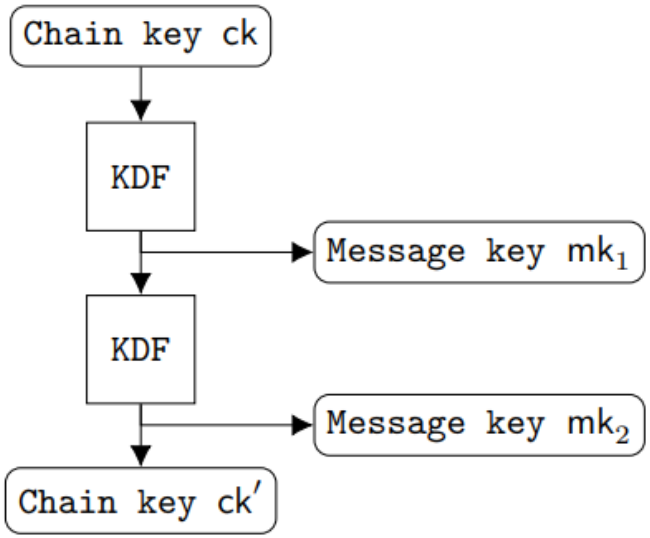
KDF гинж



- resilience
- forward secrecy
- future secrecy

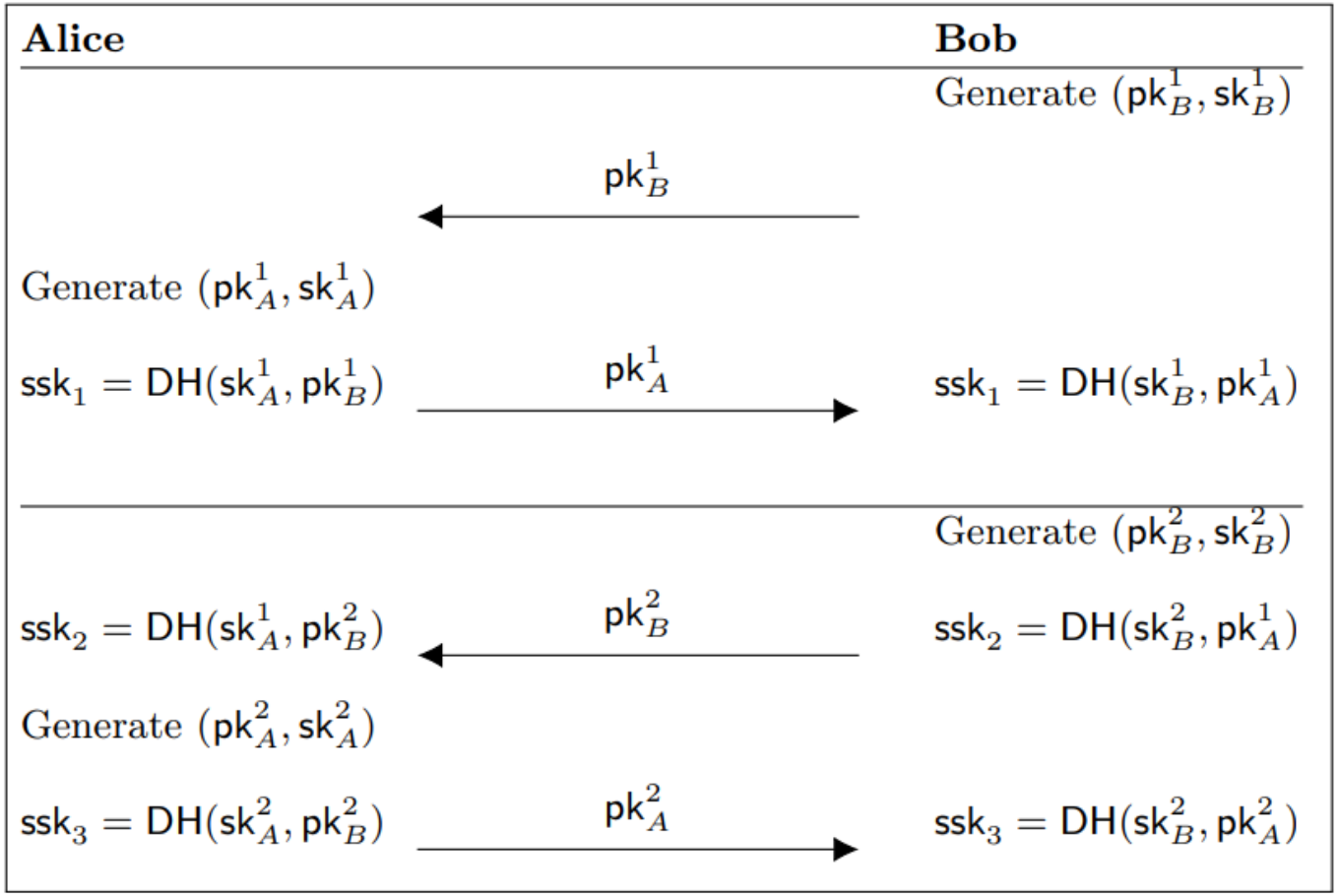
Symmetric ratchet

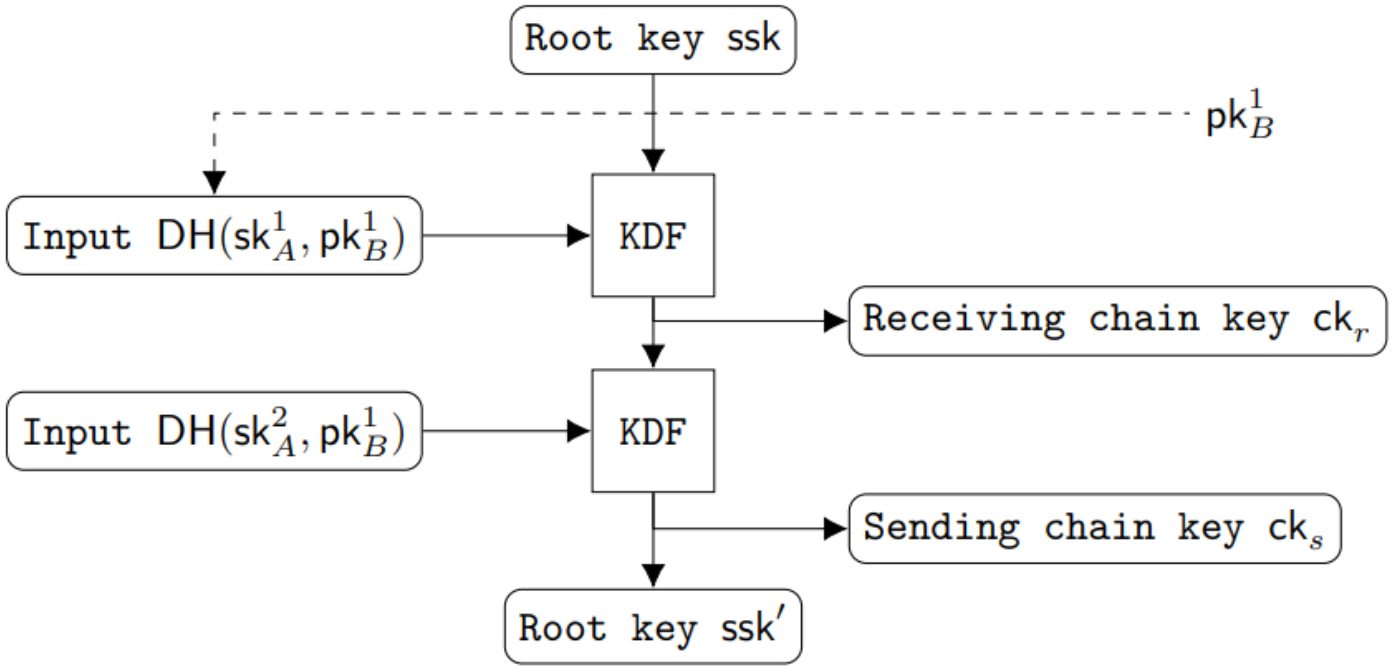
Double Ratchet алгоритм нь Diffie-Hellman ratchet болон Symmetric ratchet хоёроос тогтоно.

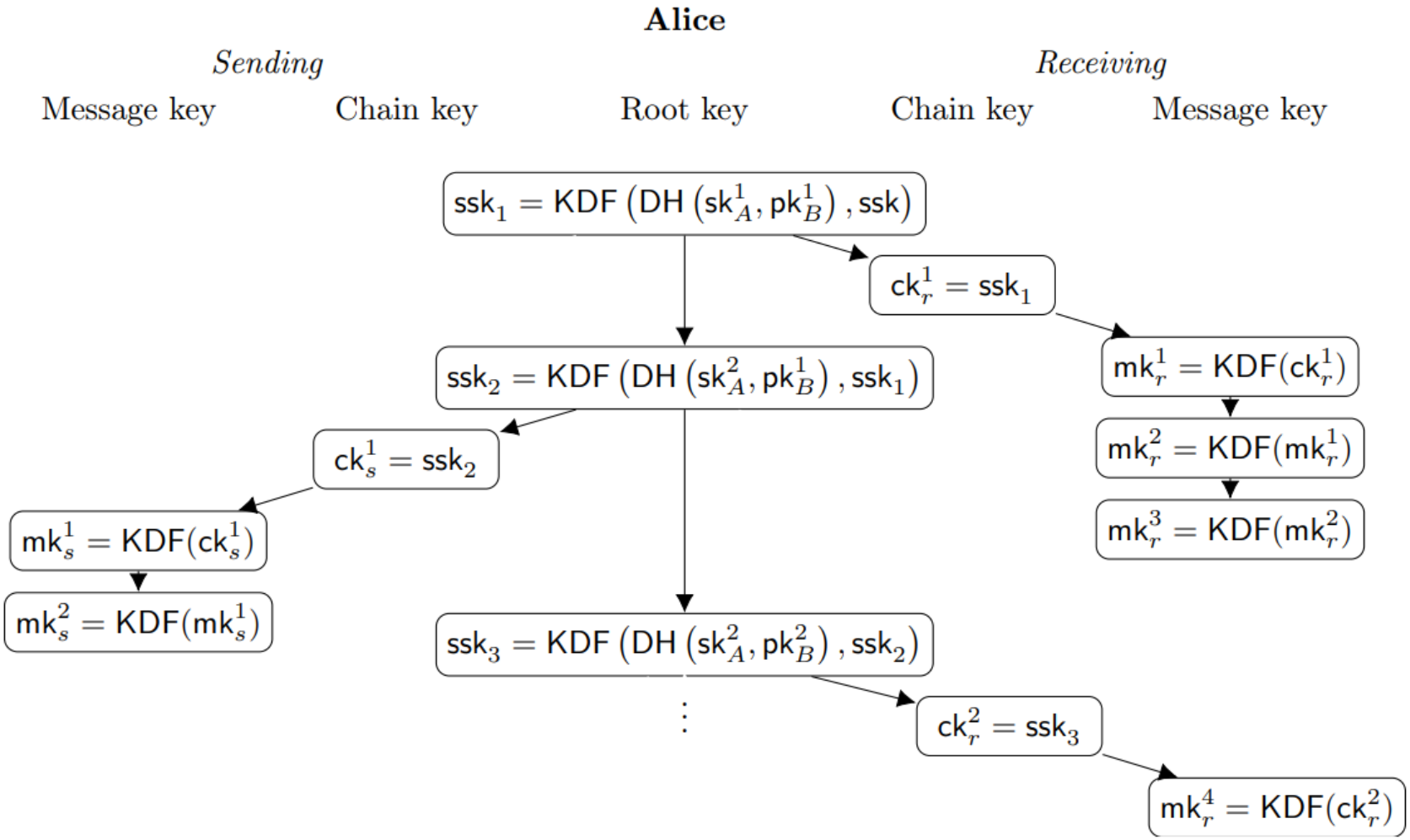


- resilience
- forward secrecy
- ~~future secrecy~~

Diffie-Hellman ratchet




Root key ssk
pk_B^1
Input $DH(sk_A^1, pk_B^1)$
KDF
Receiving chain key ck_r
Input $DH(sk_A^2, pk_B^1)$
KDF
Sending chain key ck_s
Root key ssk'


Alice
Sending
Receiving
Message key
Chain key
Root key
Chain key
Message key
$\mathsf{ssk}_1 = \mathsf{KDF}\left(\mathsf{DH}\left(\mathsf{sk}_A^1, \mathsf{pk}_B^1\right), \mathsf{ssk}\right)$
$\mathsf{ck}_r^1 = \mathsf{ssk}_1$
$\mathsf{ssk}_2 = \mathsf{KDF}\left(\mathsf{DH}\left(\mathsf{sk}_A^2, \mathsf{pk}_B^1\right), \mathsf{ssk}_1\right)$
$\mathsf{mk}_r^1 = \mathsf{KDF}(\mathsf{ck}_r^1)$
$\mathsf{ck}_s^1 = \mathsf{ssk}_2$
$\mathsf{mk}_r^2 = \mathsf{KDF}(\mathsf{mk}_r^1)$
$\mathsf{mk}_r^3 = \mathsf{KDF}(\mathsf{mk}_r^2)$
$\mathsf{mk}_s^1 = \mathsf{KDF}(\mathsf{ck}_s^1)$
$\mathsf{mk}_s^2 = \mathsf{KDF}(\mathsf{mk}_s^1)$
$\mathsf{ssk}_3 = \mathsf{KDF}\left(\mathsf{DH}\left(\mathsf{sk}_A^2, \mathsf{pk}_B^2\right), \mathsf{ssk}_2\right)$
$\mathsf{ck}_r^2 = \mathsf{ssk}_3$
$\mathsf{mk}_r^4 = \mathsf{KDF}(\mathsf{ck}_r^2)$

Sesame алгоритм

Sesame алгоритм нь синхрон биш болон олон төхөөрөмжийн тохиргоонд зориулагдан хөгжүүлэгдсэн. Энэхүү алгоритмын хэрэгцээ нь дараах 3 жишээнээс харагдана.

1. Хэрэв Боб олон төхөөрөмжөөс ханддаг бол Алис өөрийнхөө зурвасыг төхөөрөмж бүрт нь зориулж шифрлэх ёстой.
2. Алис Боб 2 нэгэн зэрэг session холболт тогтоосон үед цаашид алийг нь ашиглахаа тохиролцох ёстой.
3. Боб session холболтын хугацаа дууссантай холбоотойгоор өөрийнхөө төлвийг шинэчлэх эсвэл backup-аас сэргээж чаддаг байх ёстой.

Нэршил	Тодорхойлолт
`UserID`	Хэрэглэгчийг тодорхойлно (нэвтрэх нэр эсвэл утасны дугаар)
`DeviceID`	Төхөөрөмжийг тодорхойлно (хэрэглэгч бүрт цор ганц байна)
`SessionID`	Холболтыг тодорхойлно
`UserRecord`	`DeviceRecord`-уудын олонлогийг агуулна
`DeviceRecord`	Идэвхитэй холболт болон идэвхгүй холболтуудын жагсаалтыг агуулна
`MessageRecord`	Нэг зурвасын мэдээллийг агуулна
`MAXLATENCY`	`UserRecord` эсвэл `DeviceRecord` устахаас өмнөх хугацаа

Сервер нь тухайн төхөөрөмжид ирсэн зурвасуудыг агуулдаг төхөөрөмж бүрт харгалзах шуудангийн хайрцагтай байна. Төхөөрөмжнөөс өөр төхөөрөмжийн шуудангийн хайрцаглуу зурвас илгээвэл илгээж буй хэрэглэгчийн `UserID`, `DeviceID` сервер дээр хадгалагдана.

Хэрэв харгалзах хэрэглэгч эсвэл төхөөрөмж устсан байвал `UserRecord` эсвэл `DeviceRecord`-ийг хуучирсан гэж тэмдэглэнэ. Энэхүү хуучирсан бичлэгүүд нь `MAXLATENCY` хугацаа өнгөрч дуусахад устах болно.

Зурвас илгээхдээ өгөгдсөн `UserID`-ийн хувьд зурвасыг шифрлэнэ. Хэрэглэгчийн хуучраагүй төхөөрөмж бүрд шифрлэгдэх болно. Хэрэглэгч шифрлэсэн зурвасыг хүлээн авагчийн `UserID` болон `DeviceID`-ын хамт серверт илгээнэ. Зурвасыг хүлээн авагч тал илгээгчийн `UserID` болон `DeviceID`-ийг серверээс авна. Хэрэв илгээгчийн төхөөрөмжтэй холболт үүсээгүй бол төхөөрөмжийн нийтийн түлхүүрийг зурвасын толгой хэсгээс салган шинээр `DeviceRecord` үүсгэнэ.

Мушгирсан Эдвардийн муруй (twisted Edwards curve)

K төгсгөлөг талбар ба $d \in K,\ a, d \notin \{0, 1\},\ x, y \in K$ байг.

$$x^2 + y^2 = 1 + dx^2y^2 - \texttt{Эдвардын муруй}$$

$$ax^2 + y^2 = 1 + dx^2y^2 - \texttt{Мушгирсан Эдвардын муруй}$$

$$(0, 1) - \texttt{нейтрал элемент}$$

$$(x_1 + y_1) + (x_2 + y_2) = \left(\frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2} \right)$$

Монтгомерийн муруй

K төгсгөлөг талбар ба $A, B \in K,\ B(A^2 - 4) \neq 0$ байг.

$$By^2 = x^3 + Ax^2 + x - \texttt{Монтгомерийн муруй}$$

Лемм: Эдвардын муруй бүр Монтгомерийн муруйтай бирациональ эквивалент байна.

Мушгирсан Эдвардын муруйн $P = (x, y)$ цэгийг ашиглахын оронд түүний y координат болон s тэмдгийн битийг ашиглая.

```
convert_mont(u):
    u_masked = u (mod 2^|p|
    P.y = u_to_y(u_masked
    P.s = 0
    return P
```

```
calculate_key_pair(k):
    E = kB
    A.y = E.y
    A.s = 0
    if E.s == 1:
        a = -k (mod q)
    else:
        a = k (mod q)
    return A, a
```

XEdDSA

```
hash_i(X):
    return hash(2^b - 1 - i || X)
```

```
xeddsa_sign(k, M, Z):
    A, a = calculate_key_pair(k)
    r = hash_1(a || M || Z) (mod q)
    R = rB
    h = hash(R || A || M) (mod q)
    s = r + ha (mod q)
    return R || s
```

```
xeddsa_verify(u, M, (R || s)):
    if u >= p or R.y >= 2^|p| or s >= 2^|q|:
        return false
    A = convert_mont(u)
    if not on_curve(A):
        return false
    h = hash(R || A || M) (mod q)
    R_check = sB - hA
    if bytes_equal(R, R_check):
        return true
    return false
```

Асуулт, хариулт