НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ № 4

з дисципліни:

«ПРОЕКТУВАННЯ, РОЗРОБКА І РЕАЛІЗАЦІЯ КРИПТОГРАФІЧНИХ СИСТЕМ»

Дослідження систем обміну повідомленнями та IP-телефонії

Виконала:

Студентка групи ФІ-22мн

Калитюк Дар’я

КИЇВ 2023

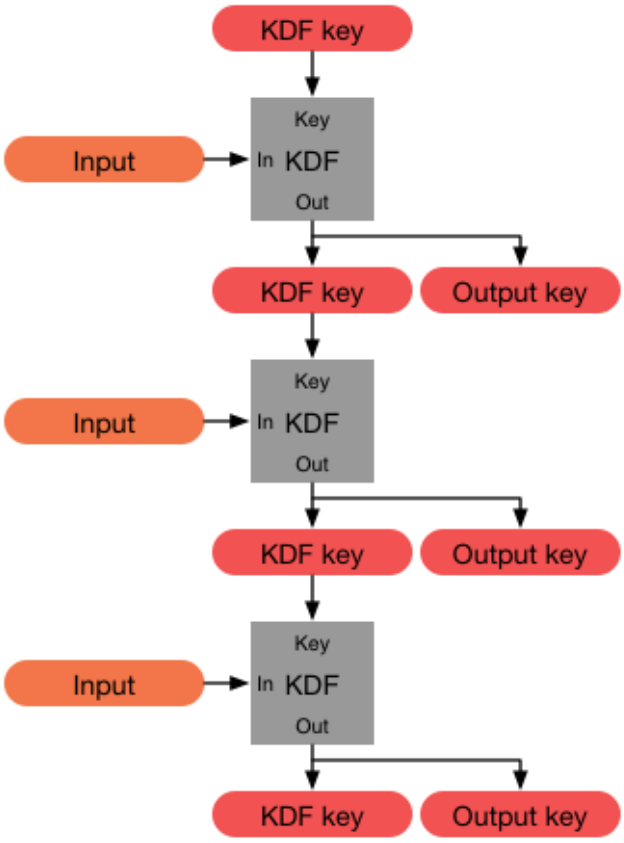
Мета роботи: дослідження особливостей реалізації криптографічних механізмів систем обміну повідомленнями та IP-телефонії.

Хід роботи

Massanger	Encryption used	Keys-Exchange and Cryptographic primitives
Viber	Double ratchet + Salsa20	Pre-keys + Curve25519, HMAC-SHA256, ECDH
WhatsApp	Signal protocol (X3DH + Double ratchet + AES-256)	Pre-keys + Curve 25519, HMAC-SHA256
Skype	TLS/AES-ICM-256	RSA- 2048
Telegram	MTProto 2.0 (AES-IGE-256)	Persistent shared key, generated via DH, KDF, SHA-256

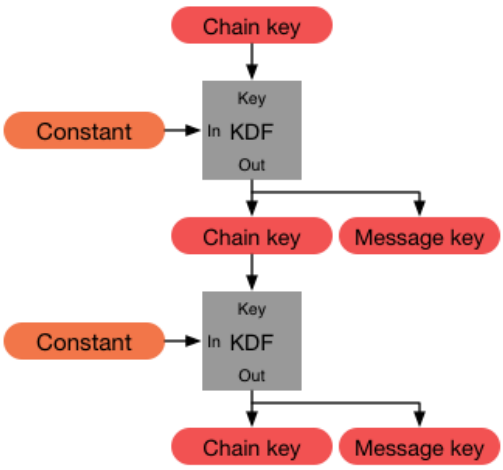
Double ratchet

Ланцюжок KDF — це основна концепція алгоритму Double Ratchet. KDF – криптографічна функція, яка приймає секретний випадковий ключ KDF і деякі вхідні дані та повертає вихідні дані. Вихідні дані неможливо відрізнити від випадкових за умови, що ключ невідомий (тобто KDF задовольняє вимогам криптографічної "PRF"). Якщо ключ не є секретним і випадковим, KDF все одно забезпечує безпечний геш ключа та вхідних даних. Ланцюг KDF означає, що частина вихідних даних із KDF використовується як вихідний ключ, а частина використовується для заміни ключа KDF, який потім можна використовувати з іншим входом.

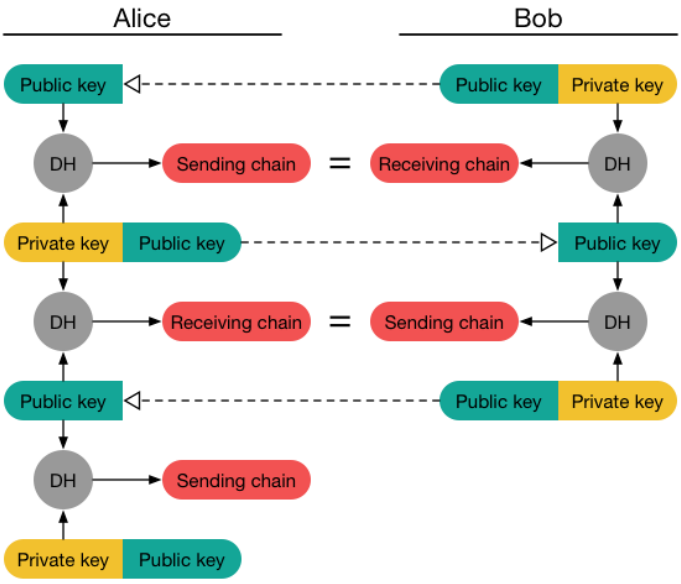


Сеанс між двома пристроями має три ланцюги – кореневий ланцюг, ланцюг відправлення та ланцюг отримання.

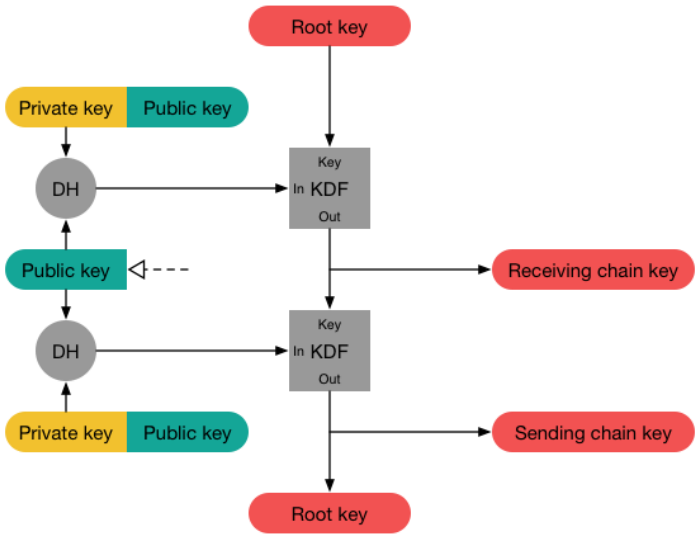
Кореневий ланцюг — це ланцюг KDF, який ініціалізується спільним секретом, який було встановлено за допомогою рукостискання X3DH (WhatsApp) або ECDH (Viber). Обидва пристрої, які беруть участь у сеансі, мають однаковий кореневий ланцюг. На відміну від ланцюжків надсилання та отримання, кореневий ланцюжок ініціалізується лише один раз на початку сеансу. Ланцюг надсилання Аліси дорівнює ланцюжку отримання Боба і навпаки. Ланцюжок надсилання використовується для створення ключів повідомлень, які використовуються для шифрування. повідомлення. З іншого боку, приймальний ланцюг генерує ключі, які можуть розшифрувати вхідні повідомлення.



Одним із видів ratchet є *ratchet із симетричним ключем*. Він бере ключ і деякі вхідні дані та створює новий ключ, а також деякі вихідні дані. Багаторазове повторення процесу створює функціональний ланцюжок виведення ключів (KDF-Chain). Однією з проблем цього алгоритму є те, що він не забезпечує майбутньої секретності. Це означає, що коли зловмисник отримує доступ до одного з ключів KDF у ланцюжку, він може використовувати цей ключ для отримання всіх наступних ключів у ланцюжку з цього моменту. Щоб запобігти цьому, Double Ratchet поєднує ratchet із симетричним ключем і *ratchet DH*, який оновлює ланцюгові ключі на основі вихідних даних Діффі-Геллмана.



Щоб реалізувати ratchet DH, кожна сторона генерує пару ключів DH (відкритий та закритий ключ Діффі-Геллмана), які стають їх поточною парою ratchet ключів. Кожне повідомлення від будь-якої сторони починається із заголовка, який містить поточний відкритий ключ відправника. Коли новий відкритий ключ отримано від віддаленої сторони, виконується етап ratchet DH між локальним секретним ключем та відкритим ключем відправника, який замінює поточну пару ключів із локальної сторони новою парою ключів. Вихідні дані DH, створені під час кожного кроку ratchet DH, використовуються для отримання нових ключів ланцюга надсилання та отримання. Вихідні дані DH використовуються як входи KDF до кореневого ланцюжка, а виходи KDF з кореневого ланцюжка використовуються як ключі ланцюга надсилання та отримання.



Signal protocol та X3DH

Signal protocol — це криптографічний протокол , який забезпечує наскрізне шифрування для голосових і миттєвих повідомлень. Для функціювання Signal protocol потребує наступні набори ключів: IdentityKey (IK), Signed PreKey (SPK) та Set of PreKeys ({OPK}). Якщо Аліса хоче почати спілкування, вона може отримати ключ ідентифікації Боба, підписаний попередній ключ і один із його попередніх ключів і використати їх для створення сеансу. Щоб зберегти криптографічні властивості, рукостискання змінено таким чином:

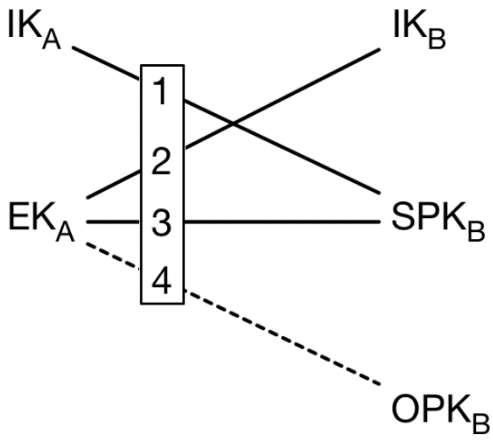
$DH_1 = DH(IK_A, SPK_B)$

$DH_2 = DH(EK_A, IK_B)$

$DH_3 = DH(EK_A, SPK_B)$

$DH_4 = DH(EK_A, OPK_B)$

$S = KDF\left(DH_1 || DH_2 || DH_3 || DH_4\right)$



Де EK_A— ефемерний, випадковий ключ, який генерується Алісою.

Тепер Аліса може отримати ключ шифрування, щоб зашифрувати своє перше повідомлення для Боба. Потім вона надсилає це повідомлення (так зване PreKeyMessage) Бобу разом із деякою додатковою інформацією, як-от її IdentityKey IK_A, публічну частину ефемерного ключа EK_A та ідентифікатор використаного PreKey OPK_B. Коли Боб увійде в систему, він зможе використати цю інформацію для виконання тих самих обчислень (тільки помінявши місцями відкритий і закритий ключі), щоб обчислити S, з якого він отримує ключ шифрування. Тепер він може розшифрувати повідомлення.

Skype

Якщо ви телефонуєте зі Skype на мобільні та стаціонарні телефони, частина вашого дзвінка, яка відбувається через PSTN (звичайну телефонну мережу), не шифрується. Наприклад, у випадку групових дзвінків за участю двох користувачів Skype-to-Skype і одного користувача в PSTN, частина PSTN не шифрується, але частина Skype-to-Skype шифрується.

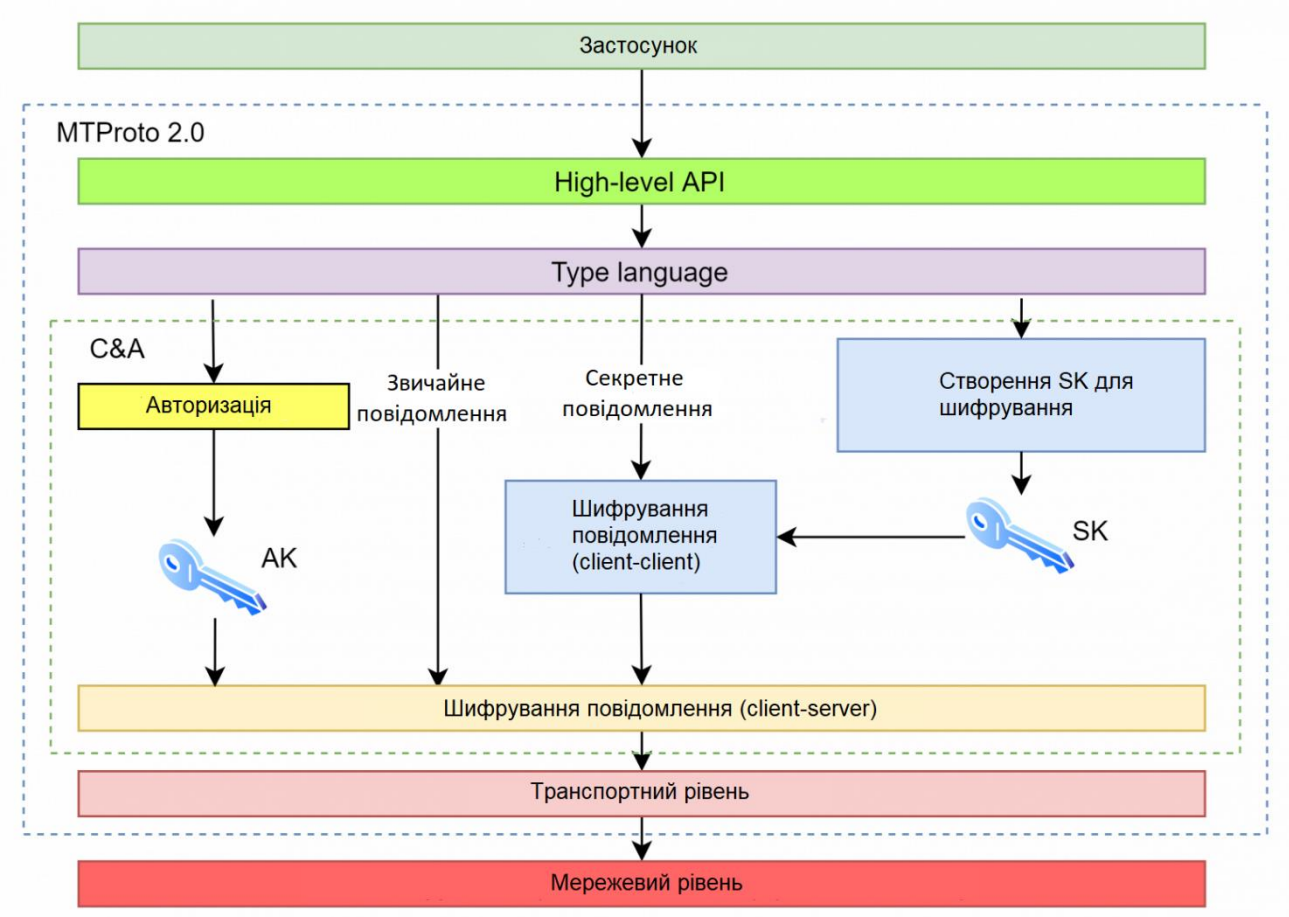
Для миттєвих повідомлень ми використовуємо TLS для шифрування ваших повідомлень між вашим клієнтом Skype і службою або AES-256, для комунікації безпосередньо між двома клієнтами Skype. Голосові повідомлення шифруються під час доставки. Однак після того, як ви прослухали голосове повідомлення, воно передається з наших серверів на вашу локальну машину, де зберігається як незашифрований файл. Відкриті ключі користувача сертифікуються сервером Skype під час входу за допомогою 1536 або 2048-бітних сертифікатів RSA.

MTProto

MTProto – клієнт-серверний набір протоколів, який служить для доступу до сервера з клієнтської програми через незахищене з'єднання. В основі протоколу лежить оригінальна комбінація симетричного алгоритму шифрування AES-IGE, протоколу Діффі-Геллмана для обміну 2048-бітними RSA-ключами між двома пристроями та ряду геш-функцій.

Цей набір можна розділити на 3 основні частини:

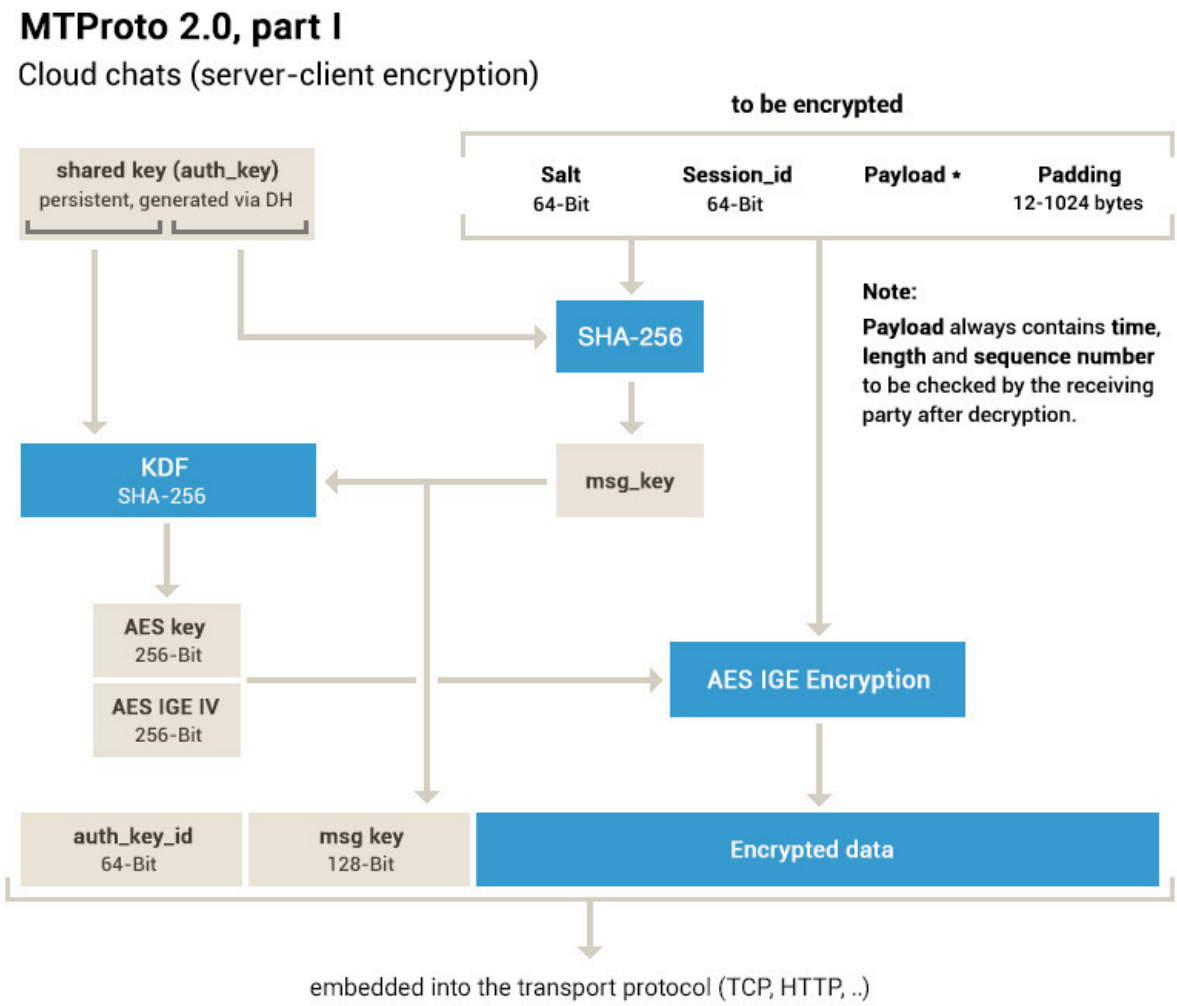
- High-level API and Type language: визначає методи, за допомогою яких запити та відповіді конвертується в двійкові повідомлення.
- Cryptographic and authorization components: визначає, як додаток (клієнт) авторизується на сервері, і методи шифрування повідомлень перед відправкою на транспортний рівень.
- Transport component: визначає, як клієнт та сервер обмінюються повідомленнями за допомогою таких транспортних протоколів, як UDP, TCP, HTTP(S), WebSocket та інші.



MTProto підтримує два режими роботи: client-server encryption (використовується для звичайних чатів) та end-to-end encryption (використовується для так званих секретних чатів).

Client-server encryption

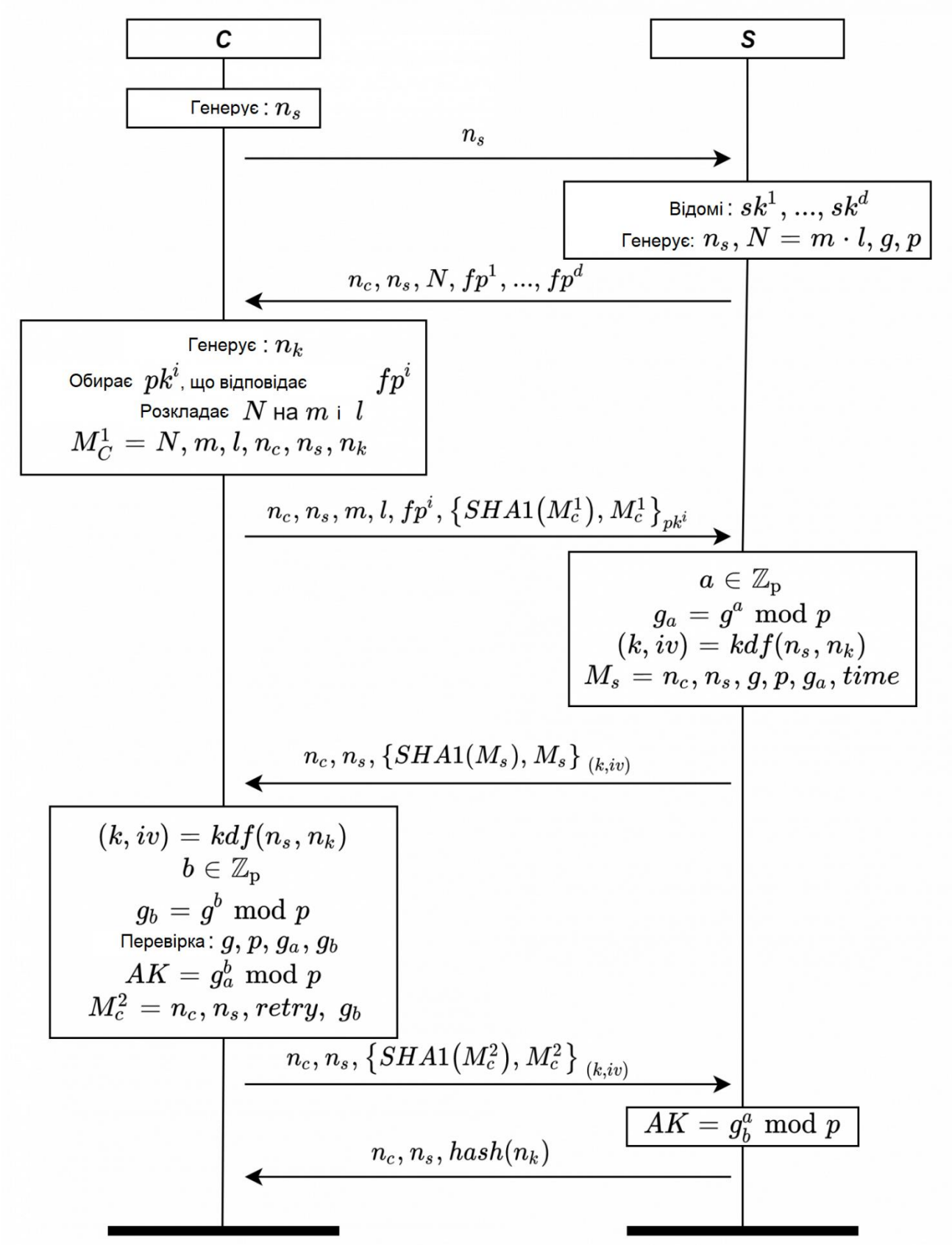
Збирається пакет для шифрування, що складається з server salt, session_id, самого повідомлення (до нього включені час, довжина та порядковий номер, які перевіряються на стороні одержувача) та padding. Далі знаходиться msg_key, 128 середніх біта гешу (SHA-256) від



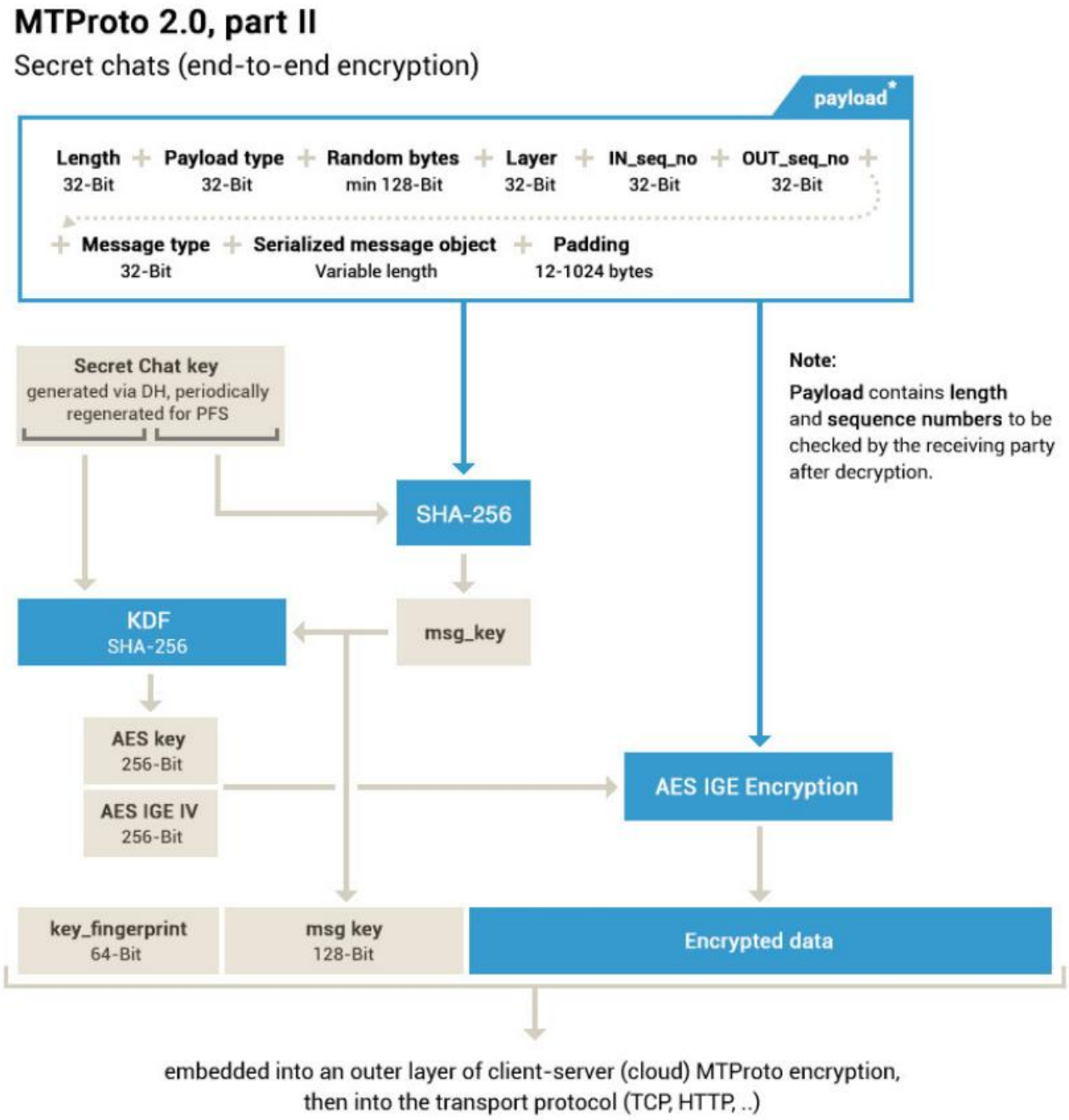
повідомлення з додаванням 32-байтового фрагменту auth_key. Auth_key у комбінації з новознайденим msg_key визначає за допомогою KDF 256-бітний aes_key та 256-бітний початковий вектор aes_iv. Далі знайдені значення aes_key та aes_iv використовуються в алгоритмі AES-IGE для шифрування повідомлення. Наприкінці збирається пакет, що складається з external header та зашифрованого повідомлення.

Генерація auth_key

Для генерації *auth_key* використовуються *RSA, SHA-1, AES* та *Diffie-Hellman*.

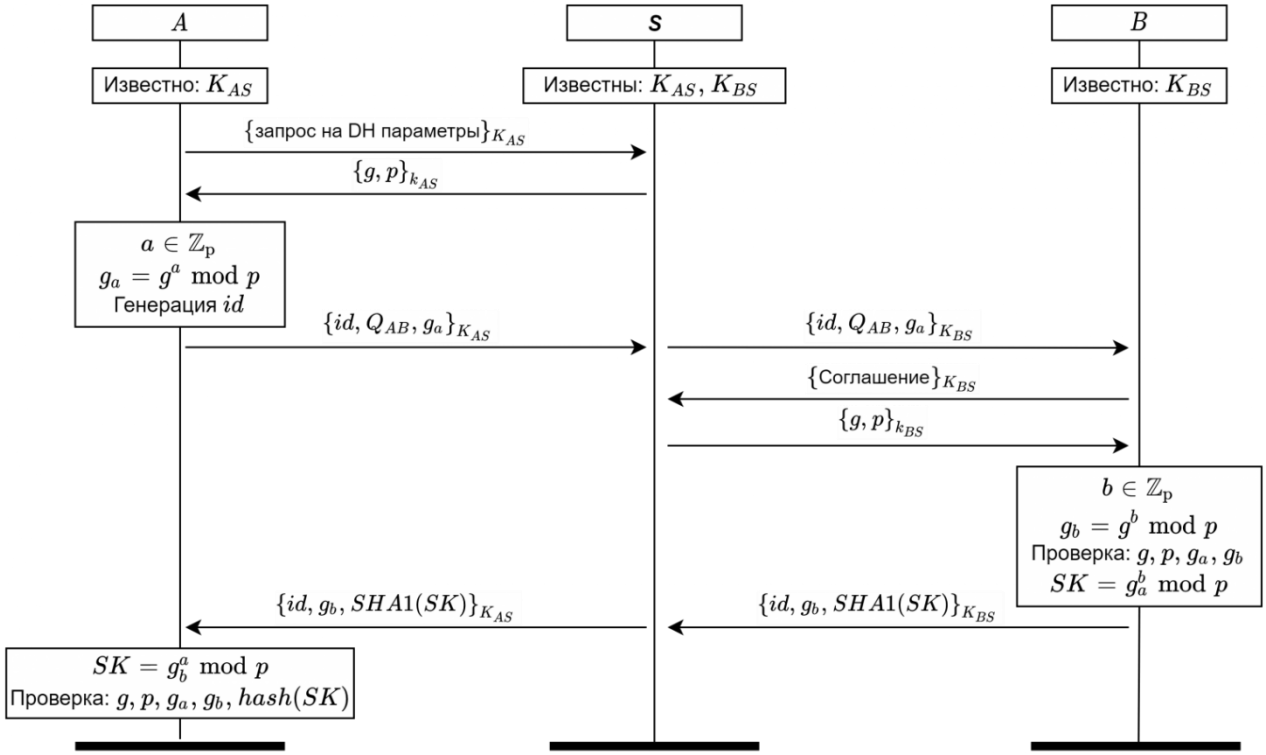

C
S
Генерує : n_s
n_s
Відомі : $sk^1, ..., sk^d$
Генерує: $n_s, N = m \cdot l, g, p$
$n_c, n_s, N, fp^1, ..., fp^d$
Генерує : n_k
Обирає pk^i, що відповідає fp^i
Розкладає N на m і l
$M_C^1 = N, m, l, n_c, n_s, n_k$
$n_c, n_s, m, l, fp^i, \{SHA1(M_c^1), M_c^1\}_{pk^i}$
$a \in \mathbb{Z}_p$
$g_a = g^a \bmod p$
$(k, iv) = kdf(n_s, n_k)$
$M_s = n_c, n_s, g, p, g_a, time$
$n_c, n_s, \{SHA1(M_s), M_s\}_{(k,iv)}$
$(k, iv) = kdf(n_s, n_k)$
$b \in \mathbb{Z}_p$
$g_b = g^b \bmod p$
Перевірка : g, p, g_a, g_b
$AK = g_a^b \bmod p$
$M_c^2 = n_c, n_s, retry, g_b$
$n_c, n_s, \{SHA1(M_c^2), M_c^2\}_{(k,iv)}$
$AK = g_b^a \bmod p$
$n_c, n_s, hash(n_k)$

end-to-end encryption



Тут повідомлення спочатку шифрується з використанням ключа SK та алгоритму, розписаного вище, а потім результат проходить друге коло шифрування з використанням AK та відправляється через сервер іншому клієнту. Важливо зауважити, що секретні ключі, які використовуються в наскрізному шифруванні, змінюються кожні 100 повідомлень або щотижня.

Схема отримання SK є наступною:


A
S
B
Известно: K_{AS}
Известны: K_{AS}, K_{BS}
Известно: K_{BS}
$\{$запрос на DH параметры$\}_{K_{AS}}$
$\{g, p\}_{k_{AS}}$
$a \in \mathbb{Z}_p$
$g_a = g^a \bmod p$
Генерация id
$\{id, Q_{AB}, g_a\}_{K_{AS}}$
$\{id, Q_{AB}, g_a\}_{K_{BS}}$
$\{$Соглашение$\}_{K_{BS}}$
$\{g, p\}_{k_{BS}}$
$b \in \mathbb{Z}_p$
$g_b = g^b \bmod p$
Проверка: g, p, g_a, g_b
$SK = g_a^b \bmod p$
$\{id, g_b, SHA1(SK)\}_{K_{AS}}$
$\{id, g_b, SHA1(SK)\}_{K_{BS}}$
$SK = g_b^a \bmod p$
Проверка: $g, p, g_a, g_b, hash(SK)$