E2EE Messenger

Основная цель проекта

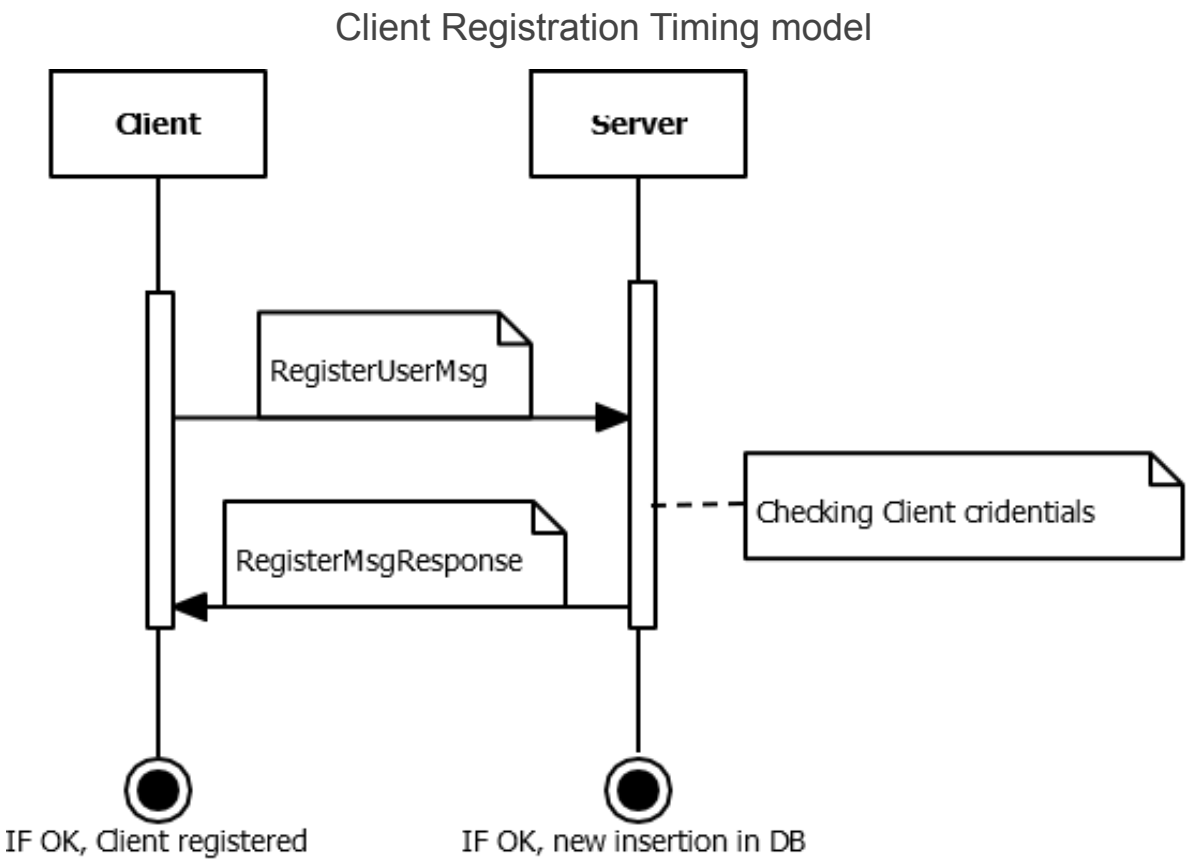
Главной целью проекта можно назвать создание e2ee (End To End Encrypted) мессенджера на основе протокола обмена ключами Diffie-Hellman. Это включает в себя:

- Программная реализация генерации и обмена ключами
- Создание сервера обеспечивающего моментальную отправку сообщений и возможность работать одновременно с большим количеством подключений
- Создание клиентской части для доступа к серверу
- Создание пользовательского интерфейса(в нашем случае это терминальный GUI)
- Организация хранения информации на клиентской и серверной части с использованием БД

Diffie Hellman

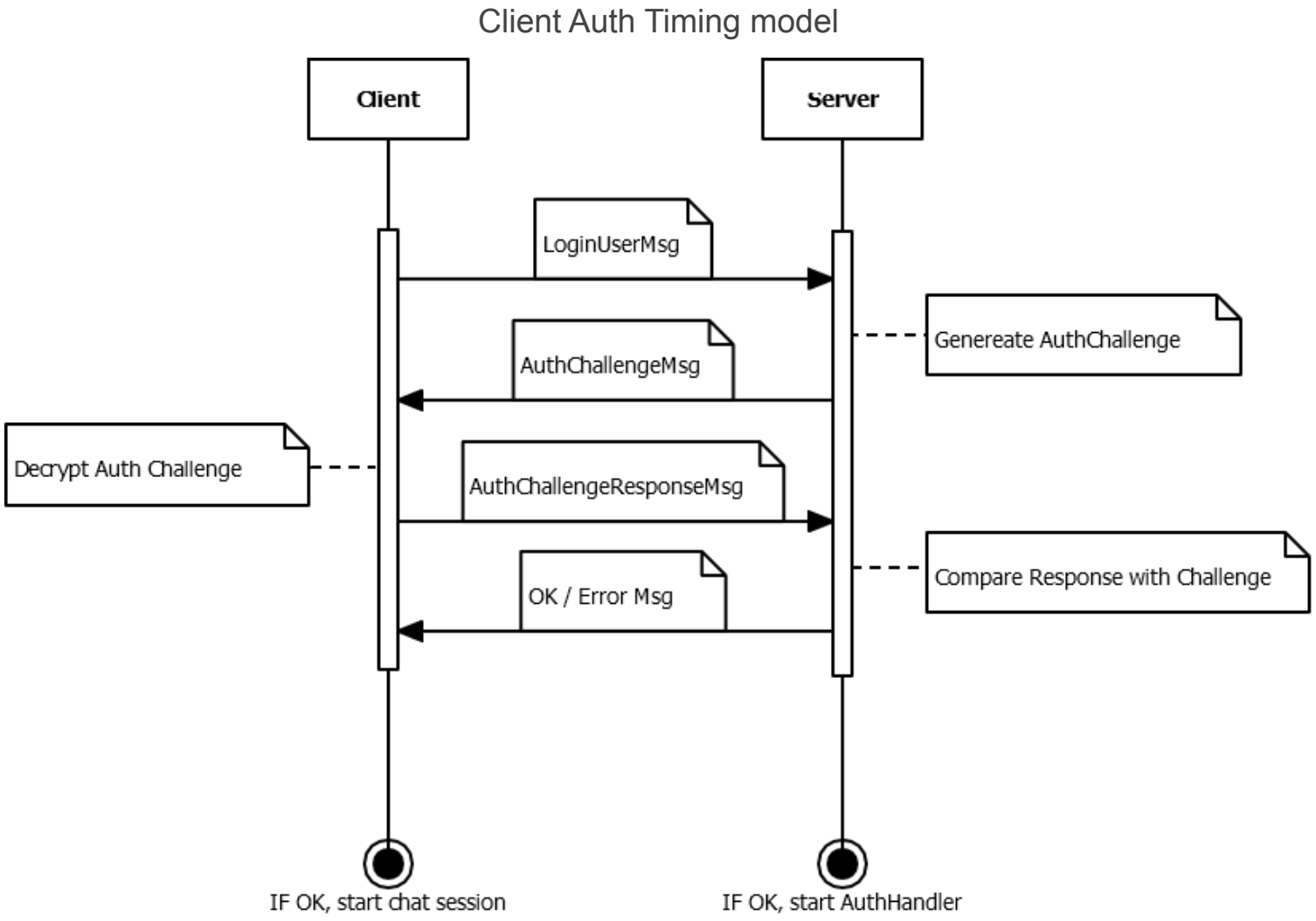
Этот метод обмена ключами широко используется в различных протоколах(TLS, Signal Protocol, SSH). Суть этого протокола заключается в получении общего секретного ключа для двух или более сторон, для его дальнейшего использования в шифровании сообщений. В нашем проекте мы использовали стандартную реализацию в кольце вычетов. Для этого первым делом мы установили p - большое простое число(p>10^232) и g=2 - первообразный корень по модулю p(тоже является простым числом). Теперь рассмотрим процесс обмена ключами между двумя сторонами: Алисой и Бобом.
Алиса и Боб генерируют случайное число a и b соответственно, это их секретные ключи. Открытые ключи вычисляются следующим образом:
A = g^a mod p (открытый ключ Алисы)
B = g^b mod p (открытый ключ Боба)
Далее они обмениваются открытыми ключами по сети и смешивают их со своими секретными:
K = A^b mod p
K = B^a mod p
В итоге они имеют общий ключ, который впоследствии будет использован для симметричного шифрования сообщений с помощью AES(перед использованием в AES, от ключа K берется hash с целью привести его длину к длине блока).
Криптографическая стойкость алгоритма Diffie-Hellmana основана на сложности задачи дискретного логарифмирования.

Client Registration



При регистрации пользователь указывает Username, генерируется пара ключей(PublicKey и PrivateKey). На сервер отправляется сообщение RegisterUser: Username, PublicKey. Сервер проверяет в своей БД предоставленные клиентом данные на уникальность, затем отправляет RegisterUserResponse(Ok Msg, если пользователь успешно зарегистрирован, Error Msg - данные не уникальны). Клиент, получив Ok, уведомляет об этом пользователя и делает запись в БД. Получив Error, пользователь получает уведомление, запись в БД не делается.

Client Authentication

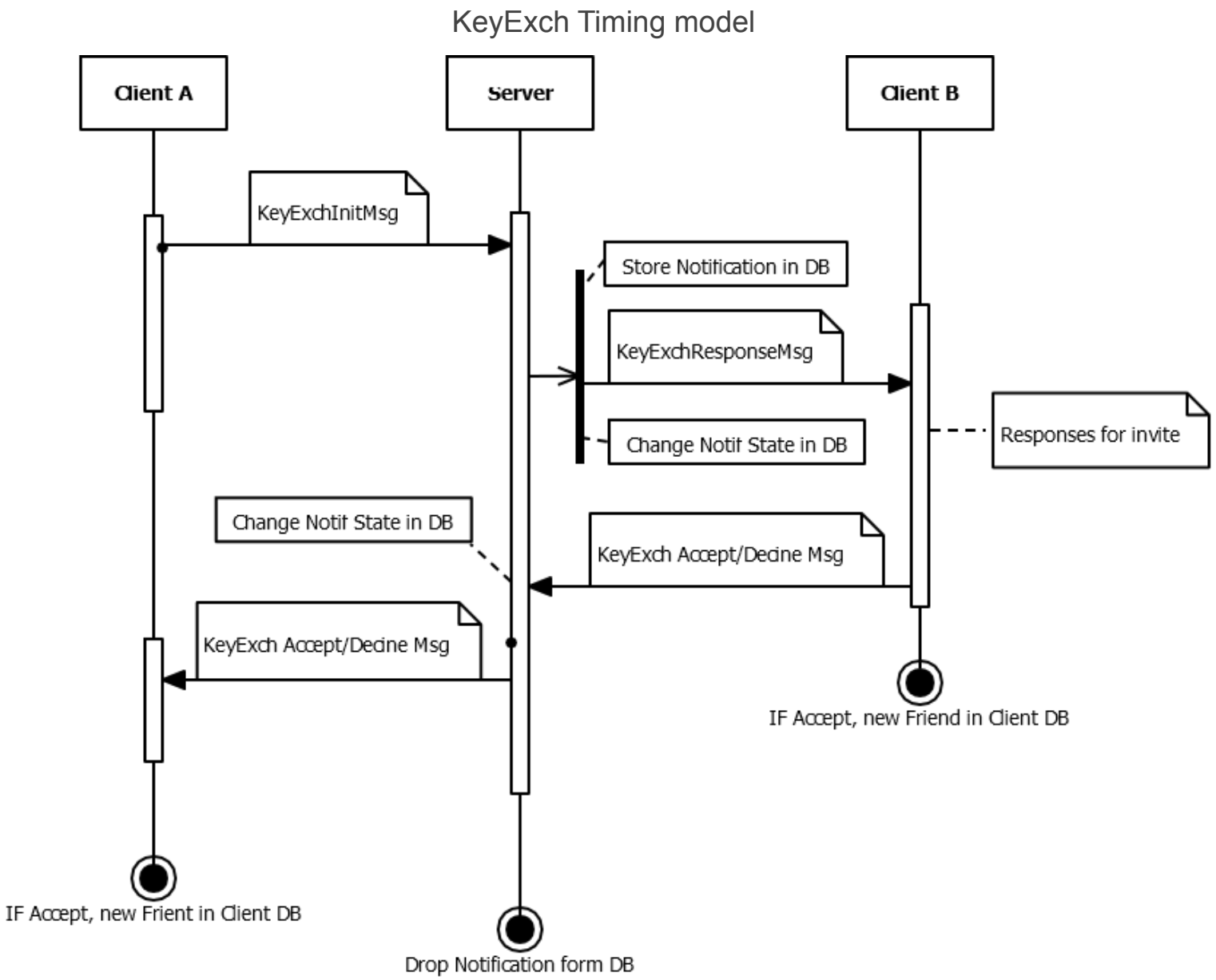


При успешной регистрации, пользователь может пройти процесс аутентификации. Он заключается в следующем:

- Пользователь отсылает на сервер сообщение LoginUserMsg(в нем он указывает Username, под которым он хочет аутентифицироваться)
- Сервер ищет в своей бд пользователя с таким именем и если такая запись существует, пользователю высылается рандомное сообщение зашифрованное с помощью PublicKey пользователя AuthChallengeMsg
- Пользователь должен расшифровать сообщение с помощью своего PublicKey и отправить серверу AuthChallengeResponseMsg
- Пользователь получает сообщение вида Ok/Err об успехе аутентификации

При успешной аутентификации пользователь может начать пользоваться функционалом нашего чата.

Key Exchange



Рассмотрим протокол добавления в друзья или обмена ключами(Key Exchange). Когда пользователь1 в приложении нажимает кнопку Add friend на сервер делается запрос о получении списка всех зарегистрированных пользователей, после этого пользователь выбирает, кого он хочет добавить и на сервер высылается KeyExchInitMsg(в нем содержится имя пользователя2, которого мы хотим добавить). У пользователя1 создается запись в БД(Requests)

- Сервер создает запись в таблице Notifications о запросе в друзья, если пользователь2, которому адресован запрос находится в сети, то запрос будет сразу же перенаправлен KeyExchResponseMsg.
- При успешной отправке запроса меняется состояние в Notification и больше уведомление присылаться не будет

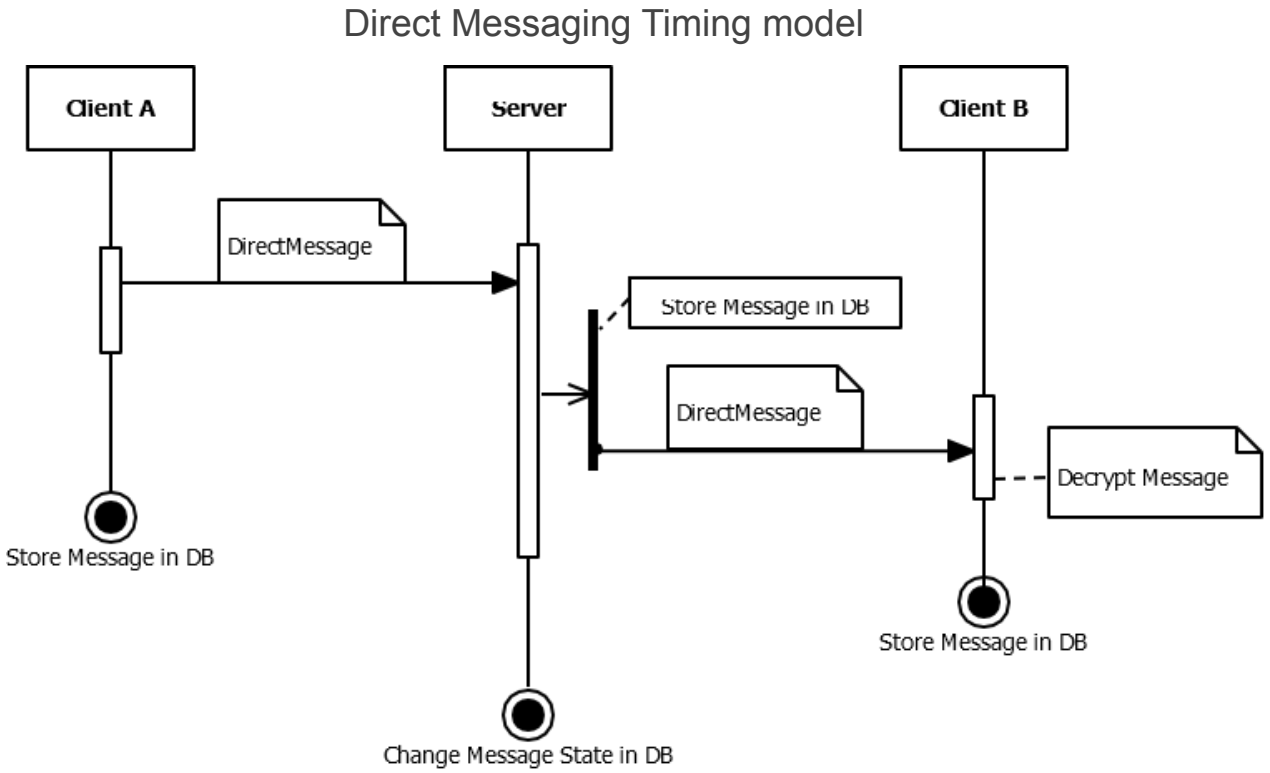
- Пользователь2 получает запрос, создает запись в своей БД(Requests) и реагирует на запрос Accept/Decline:
    - Accept: пользователь2 смешивает ключи(делается запись в таблицу Friends) и уже готов общаться, на сервер отсылается сообщение о том, что он принял invite (KeyExchAcceptMsg)
    - Decline: на сервер отслылается сообщение о том, что он отклонил invite (KeyExchDeclineMsg)

  Пользователь2 удаляет запись из БД(Requests)
- Сервер, получив реакцию на запрос, меняет состояние Notification, пользователю1 отсылается сообщение об успехе его invite(при этом на сервере запись в Notifications удаляется):
    - KeyExchAccept: пользователь1 смешивает ключи(делается запись в таблицу Friends) и он готов общаться.

  Пользователь1 удаляет запись в БД(Requests).

При успешном обмене ключами, пользователи могут начать общаться.

Direct Messaging



Пользователь1 шифрует сообщение для пользователя2 с помощью SharedKey и отправляет его на сервер DirectMessage(указывается имя отправителя/получателя, время отправления, зашифрованное сообщение). Также пользователь2 записывает это сообщение в своей БД(Messages).

- Сервер, получив DirectMessage, делает запись в своей БД(Messages) со статусом(MessageInit) и перенаправляет сообщение пользователю2, когда тот появится в сети
- Пользователь2 появился в сети, сервер перенаправляет сообщение DirectMessage и измениет его статус в своей БД(MessageReceived)
- Пользователь2 расшифровывает сообщение и делает запись в своей БД(Messages).