T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 451 AĞ PROGRAMLAMA
PROJE RAPORU

G221210045 – MEHMET BOSDANCI
mehmet.bosdanci@ogr.sakarya.edu.tr

G221210383 – METİN AYDIN
metin.aydin4@ogr.sakarya.edu.tr

Dersi Veren : Dr. Öğr. Üyesi MUSA BALTA

2025-2026 Güz Dönemi

Soket Programlama

Bu projenin amacı, TCP/IP protokolü üzerinde çalışan, çoklu istemci desteğine sahip, güvenli bir anlık mesajlaşma sistemi geliştirmektir.

Proje kapsamında, istemci-sunucu (client-server) mimarisi kullanılarak; bireysel, gruplara ve tüm ağa (broadcast) mesajlaşma yetenekleri kazandırılmıştır.

Uygulamanın en temel özelliği **Uçtan Uca Şifreleme (End-to-End Encryption - E2EE)** prensibini benimsemesidir.

Mesajlar sunucu üzerinde şifreli olarak saklanmakta, sunucu yöneticisi dahil olmak üzere gönderici ve alıcı dışında hiç kimse mesaj içeriğini okuyamamaktadır. Ayrıca çevrimdışı (offline) mesajlaşma desteği ile asenkron iletişim sağlanmıştır.

KULLANILAN TEKNOLOJİLER VE YÖNTEMLER

Proje geliştirilirken aşağıdaki teknolojiler ve kütüphaneler kullanılmıştır:

- **Programlama Dili:** Python 3.14.2
- **Ağ Protokolü:** TCP (Transmission Control Protocol) – Veri bütünlüğünü garanti etmek için seçilmiştir.
- **Eşzamanlılık (Concurrency):** asyncio kütüphanesi kullanılarak, tek bir thread üzerinde çok sayıda istemcinin (I/O bound) işlemi asenkron olarak yönetilmiştir.
- **Veritabanı:** PostgreSQL
- **Veritabanı Sürücüsü:** psycopg2
- **Şifreleme: Diffie-Hellman Anahtar Değişimi** ve **Özel Simetrik Şifreleme (Custom Cipher)** algoritmaları matematiksel olarak kodlanmıştır.

SİSTEM MİMARİSİ VE ÖZELLİKLER

Mesajlaşma Modları

Proje, istenen üç farklı iletişim türünü de desteklemektedir:

- **a) Tek Bir Kullanıcıya (Unicast):**

  ‘Alıcı: Mesaj’ formatı ile kişiye özel mesaj gönderilebilir.

- **b) Bir Grup Kullanıcıya (Multicast):**

  ‘Alıcı1, Alıcı2, Alıcı3: Mesaj’ formatı ile dinamik olarak belirlenen bir gruba mesaj gönderilebilir.

- **c) Tüm Kullanıcılara (Broadcast):**

  ‘*: Mesaj’ karakteri kullanılarak aktif ve pasif tüm kullanıcılara mesaj gönderilebilir.

Uçtan Uca Şifreleme (E2EE)

Sistemde güvenlik **Diffie-Hellman** algoritması ile sağlanmıştır.

1. Her istemci giriş yaptığında kendine özel bir Private Key ve Public Key üretir (veya yükler).
2. Public Key sunucu veritabanına kaydedilir.
3. İki kullanıcı mesajlaşacağı zaman, birbirlerinin Public Keylerini kullanarak matematiksel bir **Ortak Sır (Shared Secret)** üretirler.
4. Bu sır kullanılarak mesaj şifrelenir. Sunucu sadece şifreli metni (Ciphertext) görür ve iletir.

Diffi-Hellman Anahtar Değişimi

Algoritmanın çalışma prensibinin daha iyi anlaşılması için küçük sayılarla bir örnek aşağıda sunulmuştur:

1. **Ortak Parametrelerin Belirlenmesi:** Sistem genelinde herkes tarafından bilinen bir asal modül ve üreteç belirlenir.

Üreteç (g) = 3

Modül(p) = 17

2. **Ali'nin İşlemleri:**

   - Ali, kendine rastgele gizli bir sayı seçer: **15** *(Ali'nin Private Key'i)*.
   - Genel anahtarını hesaplar: $3^{15} \bmod 17 = 6$ *(Ali'nin Public Key'i)*.
   - Bulduğu **6** sonucunu herkese açık ağ üzerinden Veli'ye gönderir.

3. **Veli'nin İşlemleri:**

Veli, kendine rastgele gizli bir sayı seçer: **13** *(Veli'nin Private Key'i)*.

Genel anahtarını hesaplar: $3^{13} \bmod 17 = 12$ *(Veli'nin Public Key'i)*.

Bulduğu **12** sonucunu herkese açık ağ üzerinden Ali'ye gönderir.

4. **Ortak Sırrın (Şifrenin) Oluşması:**

Ali: Veli'den gelen genel anahtarı (12) alır, kendi gizli sayısını (15) üs olarak yazar:

$12^{15} \bmod 17 = 10$

Veli: Ali'den gelen genel anahtarı (6) alır, kendi gizli sayısını (13) üs olarak yazar:

$6^{13} \bmod 17 = 10$

Sonuç: Ağ üzerinde sadece **6** ve **12** sayıları dolaşmasına rağmen, hem Ali hem de Veli işlemin sonunda **10** sayısına (Ortak Sır) ulaşmıştır. Mesajlar bu "10" sayısı kullanılarak şifrelenecektir.

Mesajın Şifrelenmesi (Custom Cipher):

- **Öteleme ve Modüler Aritmetik:** Mesajdaki her bir harf (bayt), elde edilen ortak anahtarın sayısal değeri ile toplanır.
- **Döngüsel Yapı:** Eğer toplama sonucu bilgisayarın harf sınırını (256) aşarsa, **modüler aritmetik** işlemi uygulanır.
- **Sonuç:** Örneğin "A" harfi, anahtarla işleme girip anlamsız bir sembole veya sayıya dönüşür. Bu işlemi tersine çevirmek (şifreyi çözmek) için, aynı anahtarla işlemin tersini (çıkarma) yapmak gerekir. Anahtarı bilmeyen biri için bu karakterler anlamsız bir veri yığınıdır.

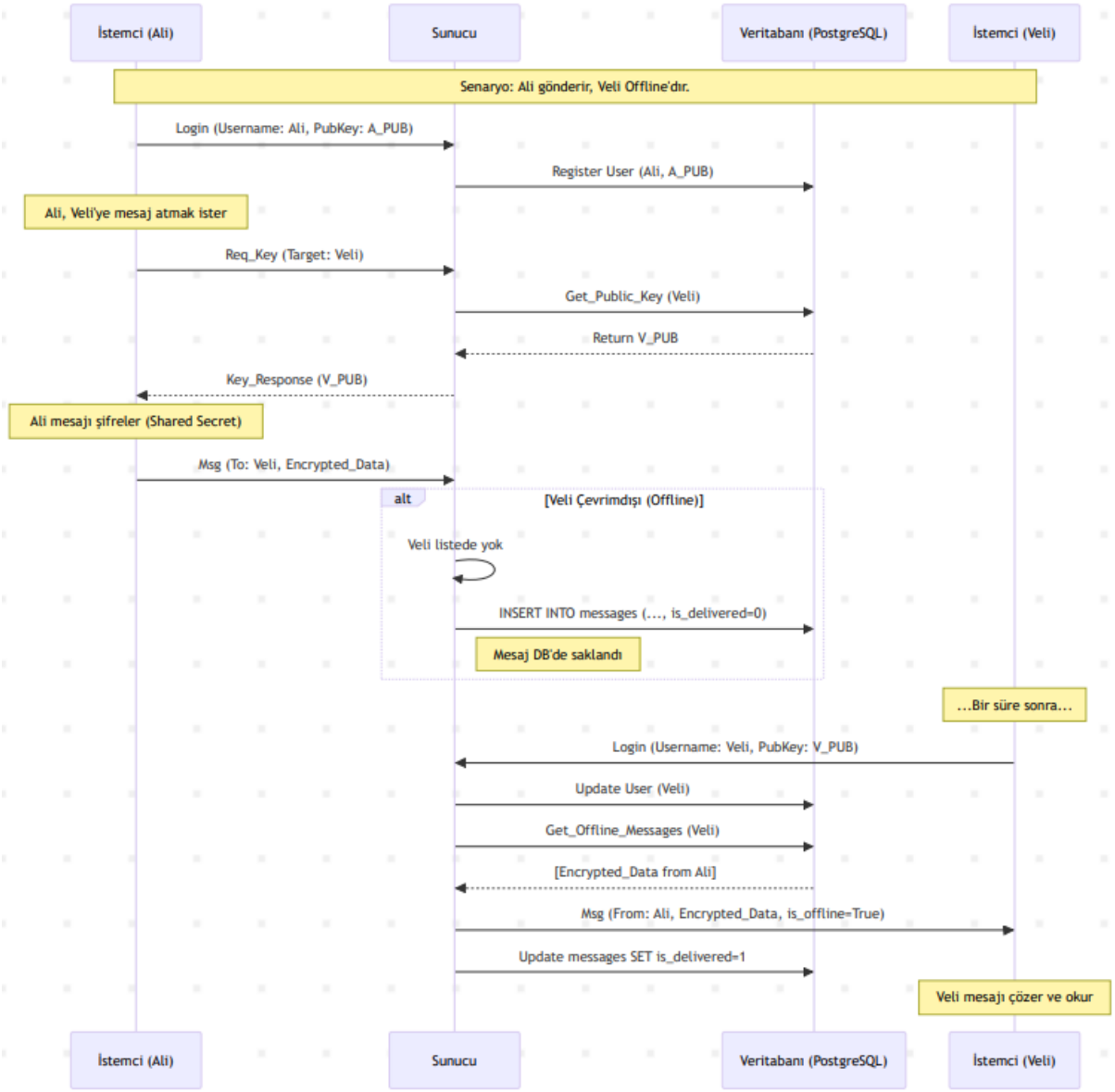
Çevrimdışı (Offline) Mesajlaşma ve Veritabanı

Sunucu, alıcı o an bağlı değilse mesajı kaybetmez.

- Mesajın şifreli hali ve Hash özeti PostgreSQL veritabanındaki **messages** tablosuna kaydedilir.
- Teslim edilme durumu **is_delivered = 0** olarak işaretlenir.
- Alıcı sisteme giriş yaptığı anda, sunucu veritabanını tarar ve bekleyen mesajları iletir.

UML SIRALAMA ŞEMASI (SEQUENCE DIAGRAM)

Aşağıdaki diyagram, **"Ali'nin Veli'ye (Veli Çevrimdışı iken) mesaj atması ve Veli'nin daha sonra girip mesajı alması"** senaryosunu göstermektedir.



Senaryo Adımları:

1. **Login:** Ali sunucuya bağlanır ve Public Key'ini kaydeder.
2. **Key Request:** Ali, Veli'ye mesaj atmak ister ve sunucudan Veli'nin anahtarını ister.
3. **DB Fetch:** Sunucu, Veli offline olsa bile veritabanından Veli'nin Public Key'ini bulup Ali'ye döner.
4. **Encryption:** Ali mesajı şifreler.
5. **Send:** Şifreli mesaj sunucuya gider.

6. **Store (Offline):** Sunucu Veli'yi bulamaz, mesajı Veritabanına kaydeder.
7. **Delivery:** Veli sisteme giriş yapar. Sunucu bekleyen mesajları DB'den çeker ve Veli'ye iletir.
8. **Decryption:** Veli kendi Private Key'i ile mesajı çözer.

WireShark Şifreli TCP Akışı

message alanı uygulanan şifreleme algoritması sayesinde okunamaz durumdaki Hexadecimal veriye dönüşmüştür.