

1. Giriş

Bu proje kapsamında, Java programlama dili ve ağ socket programlama teknikleri kullanılarak güvenli bir mesajlaşma sistemi tasarlanmıştır. Geliştirildiğimiz program, kullanıcı ad/parola doğrulaması yapar, parolanın bir görsel içerisine **LSB** Steganografi yöntemiyle gizleyerek bir doğrulama mekanizması sunar. Ayrıca, mesajlaşma trafiği **DES** algoritması ile şifrelenerek, sunucu üzerinden güvenli bir şekilde iletilmektedir.

Proje, istemci-sunucu mimarisi üzerine inşa edilmiş olup, göklu istemci desteği ve çevrimdışı mesajlaşma özelliğiyle günümüz mesajlaşma sistemlerine benzer yapıdadır.

2. Sistem Tasarımı

Bu bölümde, projenin isteklerini karşılayan on temel fonksiyonun teknik detayları ve algoritmik altyapısı açıklanmıştır.

2.1. Göklu İstemci Desteği

Sunucu mimarisi, aynı anda birden birden fazla istemcinin bağlanabilmesi üzerine kurgulanmıştır. Bu yapı için Java'nın ServerSocket sınıfı ile 5555 portu dinlenmeye alınmış ve Multi-threading yapı kullanılmıştır.

*Gerçekleşimi: Ana sunucu döngüsü (while(true)), her yeni bağlantı isteği (accept) geldiğinde, bu bağlantıyı yönetmek için bağımsız çalışan bir ClientHandler iş parçası oluşturur.

*Sonuç: Bu sayede, örneğin Ötken, Gağalay ve Alazattin aynı anda sunucuya bağlanıp işlem yapabilirken birbirlerinin veri trafiğini veya sunucunun genel işleyişini bloke etmezler.

2.2 Grafik Arayüz ve Kayıt İşlemi

Kullanıcı deneyimini kolaylaştırmak amacıyla Java Swing kütüphanesi kullanılarak bir kayıt formu (RegisterForm) tasarlanmıştır.

- **Girdi Yöntemi:** Kullanıcıdan "Kullanıcı Adı" (username), "Parola" (password) metin olarak alınır.

- **Görsel Seçim:** Steganografi işleminin uygulanabilmesi için kayıpsız sıkıştırma sunan PNG formatındaki dosyaların seçilmesine izin veren JFileChooser entegre edilmiştir. Bu arayüz güvenlik parametrelerinin kullanıcı dostu bir şekilde sisteme girilmesini sağlar.

2.3 Steganografi: LSB ile Anahtar Gizleme

Projenin en kritik güvenlik katmanlarından biri olan veri gizleme işlemi, LSB algoritması ile istemci istemci tarafında gerçekleştirilir. Amaç, parolanın açık metin olarak dolanmasını engellemektir.

- **Algoritma:** Seçilen resmin pikselleri, BufferedImage sınıfı ile taranır. Her bir pikselin RGB değerinin son biti, bit düzeyinde (bitwise) işlemle değiştirilir.

- **AND operatörü** (pixel & 0xFFFFFFFE) ile son bit sıfırlanır.

- **OR operatörü** (1 bit) ile parolanın ilgili biti piksele yazılır.

- **Güvenlik Artırımı:** Pikseller sırayla değil, Collections.shuffle kullanılarak karıştırılmış bir sırayla seçilir. Bu sayede görsel üzerinde oluşabilecek desen bozulmalarının önüne geçilir.

2.4. Sunucu Tarafında Anahtar Çıkartma

İstemci tarafından gönderilen "Stego-Resim" sunucuya ulaştığında, sunucu bu resmi işleyerek içindeki gizli anahtarı ayırır.

- **Süreç:** Sunucu şifreleme aşamasında kullanılan tatum değeri ile aynı piksel sırasını üretir. Her pikselin son biti (pixel & 1) okunarak bir araya getirilir ve 8 bitlik gruplar halinde karakterlere dönüştürülür.

- **Eşleştirme:** Çıkarılan anahtar, kullanıcının sisteme girdiği kullanıcı adıyla eşleştirilerek sunucu belleğindeki (ConcurrentHashMap) güvenli kasa saklanır.

2.5. Aktif Kullanıcı Listesi ve Durum Yönetimi

Sunucu, sisteme bağlı olan kullanıcıları anlık olarak takip eder.

- 'Canlı Liste': `activeClients` isimli Thread-safe bir harita yapısı kullanılarak online kullanıcılar hafızada tutulur.

- 'Görselleştirme': Sunucu, periyodik olarak veya kullanıcı listesinde değişim olduğunda, tüm istemcilere güncel listeyi gönderir. İstemci arayüzünde sunucudan gelen veriye göre aktif kullanıcılar Yeşil, çevrimdışı kullanıcılar kırmızı renk ile dinamik olarak gösterilir.

2.6. Çevrimdışı (Offline) Mesajlaşma Mimarisi

Sistem, alıcı o an sisteme bağlı olmasa bile mesajın kaybolmamasını sağlayan "Depola ve İlet" modelini uygular.

- 'Mekanizma': Bir istemci mesaj gönderdiğinde, sunucu hedef kullanıcının `activeClients` listesinde olup olmadığını bakar.

- 'Depolama': Eğer hedef kullanıcı çevrimdışı ise, mesaj o kullanıcıya özel ayrılmış bir mesaj kuyruğuna (`OfflineMessages Map` yapısı) eklenir ve RAM üzerinde bekletilir.

2.7. Çevrimiçi Olunca Mesajları İletme

Kullanıcı sisteme giriş yapıp kimlik doğrulamasını tamamladığı anda sunucu bu kullanıcı adına bekleyen mesaj olup olmadığını kontrol eder.

- 'Mesaj İletimi': `handleRegister` metodu tetiklendiğinde, eğer kuyruğa bekleyen mesajlar varsa, bu mesajlar sırayla kullanıcının socket hattına yazılır. Mesajlar iletildikten sonra sunucu kuyruğundan silinerek veri tekrarı önlenir.

2.8. İstemci Tarafında DES Şifreleme

İstemci, mesajı ağ kablosuna bırakmadan önce kendi belirlediği anahtarla şifreler.

• Kriptografi: Java Kriptografi Mimarisi kullanılarak DES algoritması uygulanır. Blok şifreleme modu olarak ECB ve dolgu yöntemi olarak PKCS5Padding tercih edilmiştir.

• İşlem: Kullanıcının resme girdiği anahtar (secret key), mesaj metnini şifreli bir 'baget' dizisine dönüştürmek için kullanılır.

2.9. Şifreli Mesajın İletimi

Ağ trafiği üzerinde hiçbir zaman açık metin veri dolaşmaz.

• Protokol: İstemci, şifrelediği veriyi SENDA HEDEF-KULLANICI ŞIFRELI-VERI formatında paketleyerek sunucuya iletir. Bu paketleme sayesinde sunucu, mesajın içeriğini görmeden kimden geldiğini ve kime gideceğini protokol başlığından alır.

3.10. Sunucu Tarafında Desifreleme ve Yönlendirme

Sunucu, bir köprü görevi görür.

• Desifreleme: Sunucu, gelen paketin kimden geldiğini bildiği için, o kişinin baget aşamasında resimden çıkardığı anahtarı kullanarak şifreli mesajı çözer ve açık metni elde eder.

• Yeniden Şifreleme: Mesajın alıcısı farklı bir anahtara sahip olduğu için sunucu elde ettiği açık metni bu kez alıcının anahtarı ile tekrar şifreler. Böylece mesaj, göndericiden alıcıya güvenli bir dönüşüm geçirerek iletilmiş olur.

3. Senaryo Analizleri

3.1. Eş zamanlı Çevrimici İletişim (Online-Online)

Kullanıcı A (gönderici) ve kullanıcı B (alıcı) aynı anda sisteme bağlıdır.

Akış: A mesajı gönderir → İstemci: A mesajı kendi anahtarıyla şifreler → Sunucu alır ve B'nin anahtarıyla tekrar şifreler → kullanıcı B mesajı alır ve anahtarıyla çözer.

3.2. Asenkron İletişim ve Veri Saklama

Kullanıcı A çevrimici, kullanıcı B çevrimdışı

Akış: Kullanıcı A mesajı gönderir. Sunucu, B'nin aktif olmadığını tespit eder. Mesajı, B'nin kuyruğuna ekler. B sisteme giriş yapınca, sunucu kuyruktaki mesajları B'ye iletir.

3.3. Oturum Bazlı Anahtar Değişimi

Kullanıcı B, sunucuda bekleyen mesajları varken sisteme farklı görsel/parol ile sisteme giriş yapar.

Akış: Sistem B'nin yeni anahtarını kabul eder ve iletişime izin verir. Ancak sunucuda bekleyen eski mesajlar eski anahtar ile şifrelenmiş için B bu eski mesajları açarken şifre çözüme hatası alır.

Bu durum hata olarak değerlendirilmelidir. Bir saldırgan B'nin kullanıcı adını ele geçirirse dahi, B'nin orijinal görseline sahip olmadan geçmiş mesajlarını okuyamayacağını garanti eder.

4. Sistem Test Logları

Geliştirildiği program çalışma zamanında ürettiği log kayıtları, incelenerek, algoritmaların doğru çalıştığı aşağıdaki vutlerle doğrulanmıştır.

4.1 Steganografi (LSB) İşlen Kontı

SteganoManager sınıfı tarafından resim işlenirken stegano-debug.txt dosyasına kaydedilen piksel değişim örnekleri aşağıdadır.

```
[14:20:15] --- [ENCODE BAŞLADI] key: mySecretKey --- [14:20:15] [DEĞİŞİM]  
Piksel(12,45) | Eski: LSB:0 → Yeni: LSB 1 | Gözlenen Bit 1 [14:20:15] [DEĞİŞİM]...
```

4.2 DES Analizi

[CLIENT LOG] Kullanıcı Girdisi: "Mehabo" [CLIENT LOG] Des ile şifrelenen: "yT5+dfg9Klo="

[Network] Gönderilen paket: SEND | özen | yT5+dfg9Klo=

[SERVER LOG] Gelen şifreli (paket): "yT5+dfg9Klo=" [SERVER LOG] (Plain text): "Mehabo"

[SERVER LOG] : (Özkan'ın şifreledi: "Ab3Kp9Zmsk="

4.3 Offline

[SERVER LOG] : (özen) kontrol ediliyor. [SERVER LOG] [STOK] özen offline. Mesaj kuyruğı alındı. (özen sisteme giriş yapar.) [SERVER LOG] [KAYIT] özen | Anahat: özen123)

[SERVER LOG] özen için bekleyen mesajlar iletilirdi.

5. Sonuç

Bu proje kapsamında, Java teknolojileri kullanarak Steganografi'nin kriptografi tekniklerini birleştiren, güvenli bir anlık mesajlaşma sistemi başarıyla geliştirilmiştir.

425 492 Alastin Uysal
449 821 Çağrı Turan
449 823 Özen Teber