T.C.

SAKARYA UNIVERSITY

FACULTY OF COMPUTER AND INFORMATION SCIENCES

GRADUATION PROJECT DESIGN

Rıdvan Al HOURANI

End-to-End ENCRYPTED VOICE CALLING APPLICATION FOR SMARTPHONES

Department of Computer Engineering

Thesis Advisor: Assoc. Prof. Dr. Ünal ÇAVUŞOĞLU

2019-2020 Fall Semester

ÖNSÖZ

Günümüzde internet güvenliği çok önemli bir faktor olduğunu herkese biliniyor, ancak maalesef çok önemli olduğuna rağmen haberleşme araçları ses olsun, mesaj olsun yada başka türlü haberleşme aracı olsun çoğunu güvenlik faktoru doğru ve tam güvenli bir şekilde uygulanılmıyor.

Bu bağlamda ilk önce Allah'a sonra da anneme, babama kardeşlerime ve sayın Dr. Ünal Çavuşoğlu değerli Hocama ve bütün yardım eden arkadaşlarıma Erkam, Furkan ve Ömer özel olarak ve bana kaynak sağlayan kurumlar samimi teşekkürlerimi sunuyorum.

Abstract

Keywords: E2EE, Cryptography, Machine Learning, Mobile Application, React Native.

Speed and safety! In this era, the two most important factors in any application are the rapid transfer of data between backend, user interface, and security. React Native has been used to create hybrid apps with an easy-to-use interface.

I used this cross-platform for mobile devices to design my “End-to-End Encrypted Voice Calling App for Smartphones”, which has the slogan “Type One, Use Everywhere”.

The primary purpose of the app is to create a very secure app. This uses an end-to-end encryption method and an open source signal protocol library to create a voice communication application using different encryption algorithms.

In addition, in the design of the application, not only is it planned to transfer security data in the internet environment, but I have planned to open the application with a face print to increase the user's security in the real world with the ML kit from Google.

While I traveled researching how to build and build this app, I learned the following; Although the cryptography branch is not a broad one, its importance is very great. Encryption algorithms consist of the same concepts on their basis. With change, the performance of algorithms can be improved in terms of security or speed.

There are different encryption algorithms according to my research and the conclusion that I came to in my design, depending on the application that will be designed and the environment in which the application will operate, the encryption algorithm used will play a major role in the performance of the application.

Creating something from scratch isn't always a good idea, so ready-made open source libraries - like the Signal Protocol Library - that exist make it easier for us as a cryptographic designer and take Internet security to a higher level.

Encryption algorithms can and should be combined with current and new technologies such as React Native to create modern high-performance applications.

As a result, the "end-to-end encrypted voice calling application for smartphones" was designed using the React Native Cross platform and signal protocol. SQLite Database and Firebase Cloud were used in the database design. It was decided to use the Google ML kit firebase library of machine learning for face recognition and the RSA algorithm to store the obtained data in a safe way.

We can see how easy it is to implement the end-to-end encryption method. Internet security and the security of people's personal information are of great importance, and people must be seriously aware of this problem, so that rulers and technology companies are forced to use this method.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

RTP	: Real Time Protocol
SRTP	: Secure Real Time Protocol
RSA	: Rivest-Shamir-Adleman Şifreleme Algoritması
AES	: Advanced Encryption Standard
CBC	: Cipher Block Chaining
SHA	: Secure Hash Algorithm
HMAC	: Hash Message Authentication Code
DF	: Diffie-Hellman
3-DF	: 3 Double Diffie-Hellman
EC	: Elliptic Curve
E2EE	:End-To-End Encryption
ML	:Machine Learning

ŞEKİLLER LİSTESİ

ÖZET

Anahtar kelimeler: E2EE, Mobil Uygulama, Makine Öğrenmesi, React Native

Hızlı ve güvenlik! Bu çağda her hangi bir uygulamada en önemli iki faktoru hızlı veri transferi ve UI ve güvenlik konusu. React Native yeni çıkan Cross Platformu mobil cihazları için hızlı arayüzü ile hybrid uygulamalar oluşturarak ve "Bir defe yaz, Her yerde kullan" slogana sahip Android ve İOS için Uçtan Uca şifreleme ile sesli haberleşme mobil uygulaması arayüzünü tasarlamak için kullanılmıştı.

Uygulamayı asıl olan birincil amacı çok güvenliğe sahip bir uygulama oluşturmaktadır. Uçtan Uca şifreleme ve açık kaynak olarak Sinyal Protokolü farklı şifreleme algoritmaları kullanarak bir sesli haberleşme uygulaması oluşturmayı amaçlıyormuş.

Ek olarak sadece internet ortamında güvenlik veri transferi amaçlanılmamış, gerçek dünyada kullanıcının güvenliği arttırmak için uygulamayı yüz izi ile açılması planlamışım.

BÖLÜM 1. GİRİŞ

React native platformu birden fazla farklı işletim sistemi üzerinde aynı kod ile mobil uygulamaları yazmak ve çalıştırmak için cross platform olarak düşünülebilir. Platformu 2013'te ilk defa bir Facebook teknoloji konferansında resmen hakkında konuşuldu ve 2015'te ilk versiyonu ortaya çıktı [1].

1.1. React Native ayırt edilen özelliği

React native platformu ayırt edilen özelliği, onun hızlı çalışan kullanıcı arayüzüdür. Ve kullanması için fazla mükemmel programlama becerileri istememesidir, javaScript diline, HTML ve CSS dayanması yüzündendir. Böylece her hangi bir Front-end tasarımcı, back-end hakkında fazla bilgisine sahip olmayı gerekmeden bir mobile geliştirici olabilir.

React Native cross platformu ile mobile cihazı ve işletim sistemini Android, IOS ve Web uygulamaları arasında bir köprü olarak çalışıyor.

Web view ile spesifik işletim sistemlere yönelik tasarlanan Native uygulamların birleşen React Native platformu hybrid bir platform olarak kabul edilir ve react native'in uygulamaları hybrid uygulamalar sayılır.

1.2. Web view uygulamaları, bir web sitesinin Container

WebView, bir Android cihazında konteyner içinde bir Web sayfasını oluşturmasını ve göstermesi -rendering- sağlayan bir tekniktir. Bu WebView bir mobil uygulaması gibi olur ancak bu durumda uygulamayı telefonun temel özelliklerinden kamera ve GPS gibi özellikleri kullanamaz ama çok hızlı arayüze sahip bir mobile uygulaması olur.

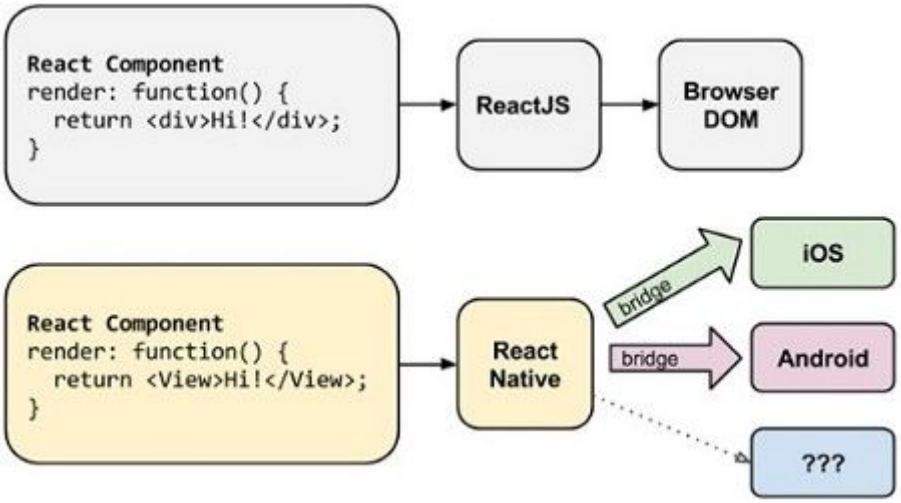
Native uygulamar herhangi bir işletim sisteme tasarlanan bir mobile uygulamasıdır. Native uygulamalar telefonun özellikleri kullanabilir (Kamera, Gbs vb.) ama WebView'ya göre daha yavaş arayüzü vardır. Ve böylece kullanıcının kullanım deneyiminde iyi performansa sahip değildir.

Ve böylece React Native platformu ile tasarlanılan uygulamalar iki özelliğinden faydalanarak (WebView Hızlı arayüzü ve Native uygulamaların kullanabilen telefonunu özellikleri) birleştirerek daha hızlı Native uygulamalar oluşturulur. Bu işlem arka planda JavaScript Thread'leri kullanarak kodlar Json'e dönüştürüp ve cihazın donanımına ve işletim sistemine göndererek derlemesini sağlar.

1.2.1. Asynchronous

Üstelikle React Native bununla yetmedi ve diğer hybrid uygulamalarından ayrıtan şey, bahsettiğim işlemden çift yönlü ve eşzamanlı olmayan (asynchronous) özelliği yapımında kullanarak haberleşme işlemi kat kat daha hızlı olur ve mobil uygulamayı çalışımında daha esnek ve hızlı olur.

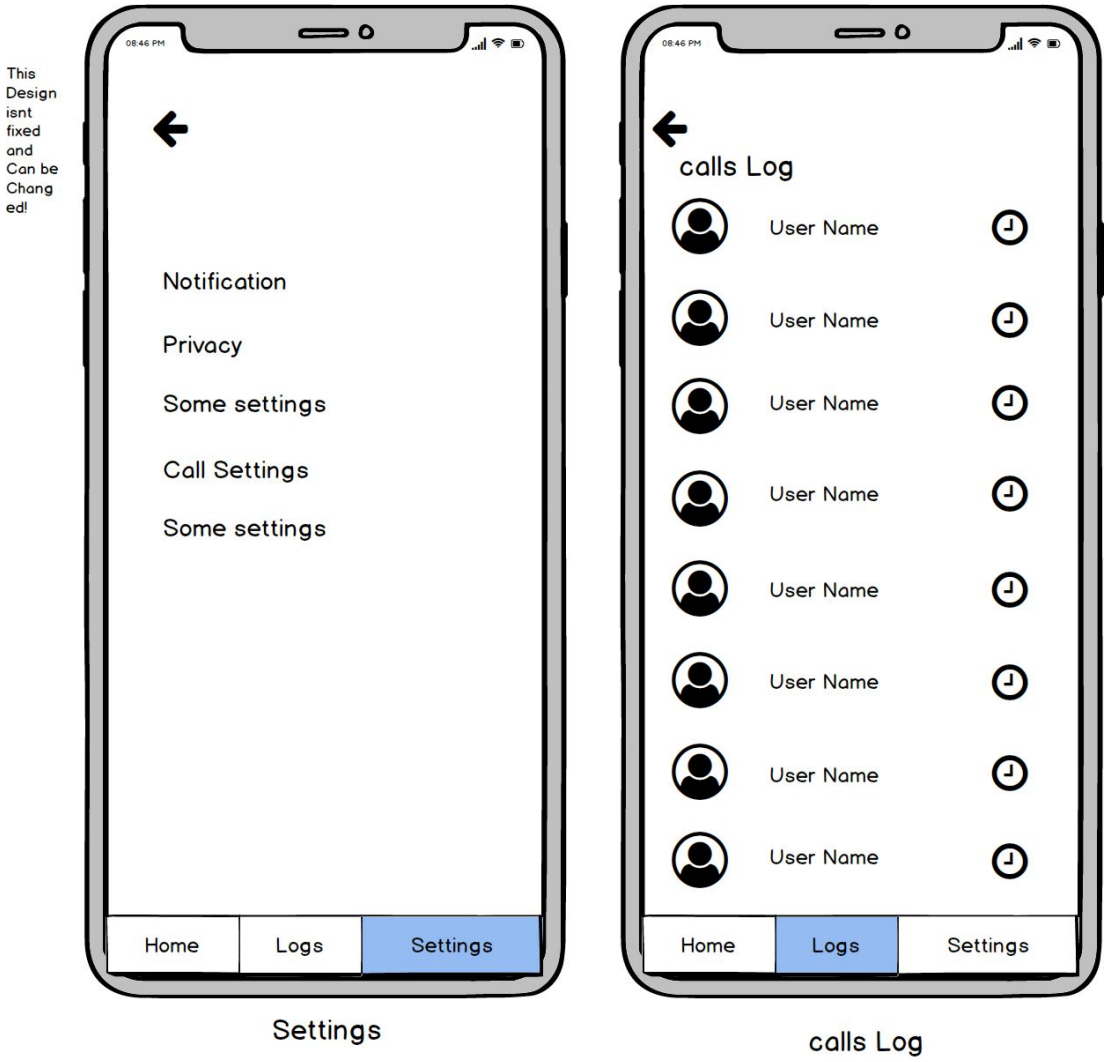
Aşağıdaki resimde ilk önce React ve sonrakinde React Native platformları çalışma aşamaları belirtiyor.



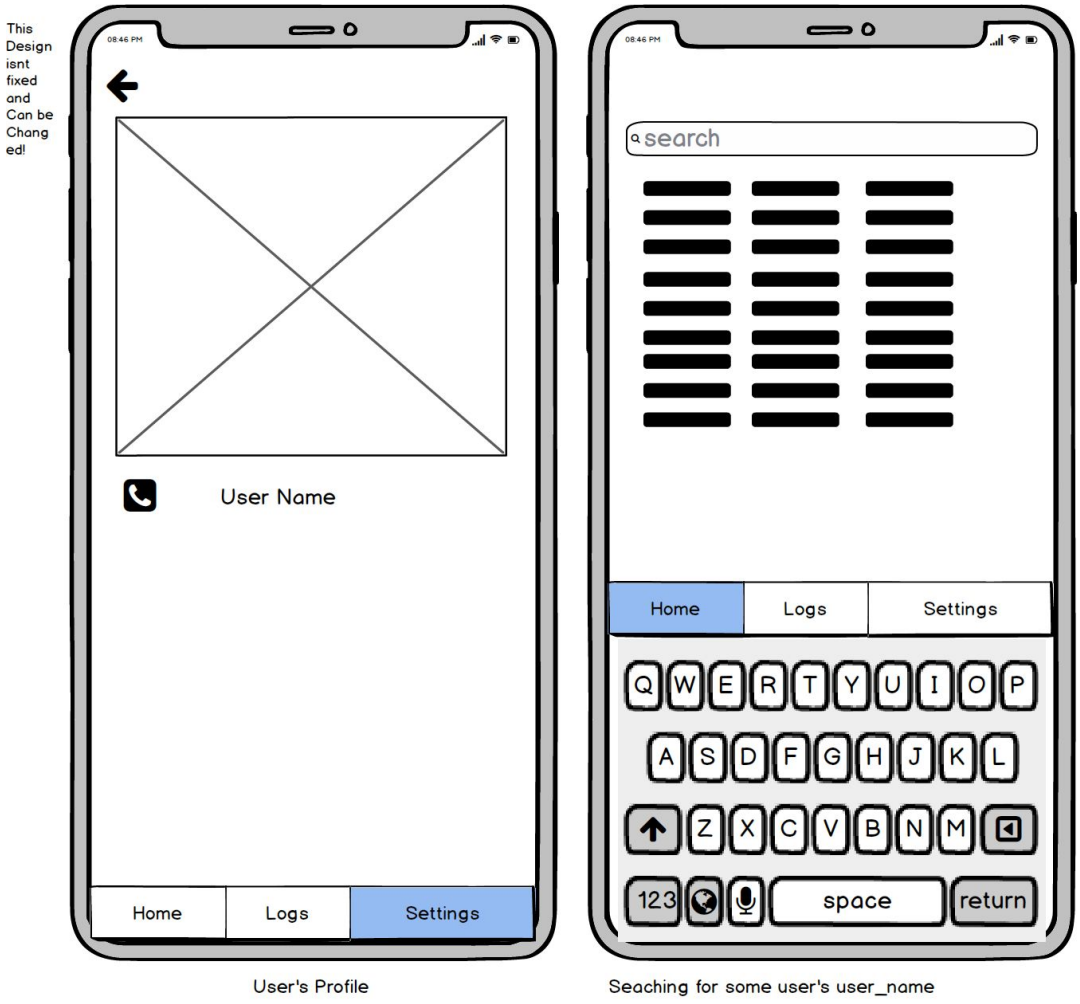
Şekil 1.1. React vs React Native çalışma mantığı

1.3. Uygulamanın ilk arayüz tasarımı (Scratches)

şekil 1.2.


This Design isnt fixed and Can be Changed!
08:46 PM
Notification
Privacy
Some settings
Call Settings
Some settings
Home
Logs
Settings
Settings
08:46 PM
calls Log
User Name
User Name
User Name
User Name
User Name
User Name
User Name
User Name
Home
Logs
Settings
calls Log

şekil 1.3.


This Design isnt fixed and Can be Chang ed!
08:46 PM
User Name
Home
Logs
Settings
User's Profile
08:46 PM
search
Home
Logs
Settings
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
return
Seaching for some user's user_name

şekil 1.4.

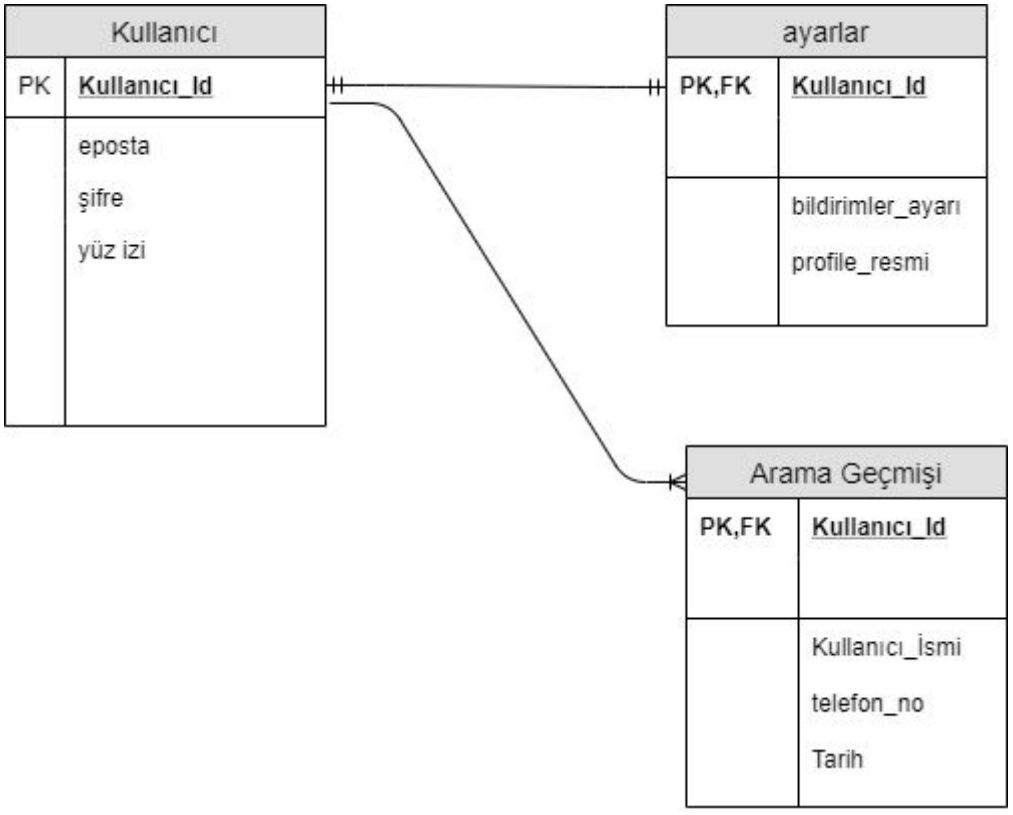
BÖLÜM 2. VERİ TABANI

2.1. Telefon içindeki veri tabanı

SQLite ile bilinen ünlü veritabanı motoru küçük projelere yada fazla veritabanı güce sahip olmayan projelere bir veritabanıdır. SQLite C/C++ ile yazılan veritabanı motoru onlarca programlama dillerine destekliyor. Projemize React Native JS mobil uygulamasına telefon içinde bazı bilgileri saklamak için kullanacağız.

İhtiyacımıza karşılayabilecek SQLite içinde:

Giriş yüz izi,
Arama geçmişi,
Kullanıcı adı ve parolası,
Ayarlar ve benzer,
geçici olarak kullanıcıların profile resimleri ve bilgileri, Tabloları olacak.



Şekil 2.1 Entity relationship diyagramı (telefon içinde veri tabanı)

2.2. Remote Veritabanı (Bulutlu veritabanı)

Remote olarak ta veritabanına ihtiyacimiz gerekecek. Hem Şifreler ve güvenlik yüz resimleri, hem de kullanıcıların ID leri cloud'da saklamalıyız ki kullanıcılar arasında iletişim sağlatabiliriz.

Giriş yüz izi,
Kullanıcıların ID ve şifreleri,
Şifreleme anahtarları,

BÖLÜM 3. ŞİFRELEME ALGORİTMALARI

Uçtan uca şifreleme (E2EE), üçüncü tarafın düz metin iletilere erişmesini engellemek için tasarlanmış güvenli bir iletişim modelidir, böylece sadece gönderici ve alıcı cihazlarında mevcut şifreleme anahtarı sayısından şifrelenmiş metni düz metne çevirebilir ve okuyabilir.

Uygulamamızı sesli haberleşme yapılacağından Real-Time veri şifrelemeyi kullanılmalı.

Real-Time veri transferi ve şifrelemesi (SRTP) sinyal üzerinde yada bit bazı alarak hem hız hem güvenlik sağlanabilir. Uygulamamızda Sinyal protokolu kullanmayı uygun gördüm.

3.1. Signal Protokolü

Signal Protokolü (eskiden TextSecure Protokolü olarak bilinir), sesli aramalar, video görüşmeleri ve anlık mesajlaşma konuşmaları için uçtan-uca şifreleme sağlamak için kullanılabilen federe edilmemiş bir şifreleme protokolüdür. Protokol, Open Whisper Systems tarafından 2013 yılında geliştirildi ve daha sonra Signal olan açık kaynak TextSecure uygulamasında tanıtıldı. Protokol Double Ratchet Algoritması , prekeys ve üçlü Diffie-Hellman (3-DH) el sıkışmayı kombine eder ve Curve25519 , AES-256 ve HMAC- SHA256'yı temel olarak kullanır.

3.1.1. Özellikler

Protokol, gizlilik, bütünlük, kimlik doğrulama , katılımcı tutarlılığı, hedef doğrulama, katılımın reddedilmesi(participation repudiation) ve daha.

3.1.2. Kimlik Doğrulama

Kimlik doğrulama için kullanıcılar, açık anahtar parmak izlerini(public key fingerprint) harici bir kanal aracılığı ile karşılaştırabilir .Bu, kullanıcıların birbirlerinin kimliklerini doğrulamasını ve araya girme saldırısını (MItM) engellemeyi mümkün kılar.

Whisper systems'den üretilen single protocol açık kaynak olarak sağladığı libsignal-protocol-javascript kütüphanesi mobile uygulamamızda kullanacağımız kütüphanedir. Protokol çalışma mantığını Session lardan oluşur.

3.2. Signal Protocol çalışma mantığı

3.2.1. Genel bakış

Ratcheting ileri yönlü gizlilik protokolü hem senkron hem asenkron haberleşme ortamlarında çalışan protokolüdür.

Ratcheting: şifreleme anahtarları düzenli bir şekilde güncelleme sağlanan yöntemdir. Yani bir anahtar birden fazla defa kullanmaya izin verilmeyen bir teknik.

- **PreKeys**

PreKeys anahtarları bir kullanıcının kayıt olduğu zaman -Mobil uygulamada- automatik olarak ECPublicKey oalrak üretilen bir anahtardır. ECPublicKey ve kullanıcını ID'sı beraber server'a yönlendirerek saklanır.

- **Sessions**

Bahsettiğim üzere Single Protocol session'dan oluşur. Bir client bir session oluşturduktan sonra artık aynı session bütün göndereceği yada alacağı mesajların -veri- şifreleyebilir/çözebilir.

Sessions oluşturmak için bu iki yöntemden birisi kullanabilir:

1. **PreKeyBundles**. Bir kimsenin diğer kişi ile iletişim kurmak amacıyla server'den PreKeyBundles alabilir (Retrieving).
2. **PreKeySignalMessages**. Bir alıcı mesaj/veri karşı taraftan (Gönderici) aldığı zaman.

- **State**

İki Client arasında bir session oluştuktan sonra session'un bitine kadar kalması gereken bazı state'ler var:

- Identity State. İki tarafın kullanıcı kimliği belirtilen anahtarı "Pair keys".
- PreKey State. Üretilen PreKeys anahtarları.
- Signed PreKey States. Kullanıcıların Signed PreKey anahtarları.
- Session State. Kullanıcıların oluşturdukları session durumu saklamalı.

- **Gereksinimler**

Libsignal-protocol-javascript kütüphanesi kullanmak için bazı arayüzlere ihtiyaç vardır.

- ArrayBuffer
- TypedArray
- Promise
- WebCrypto with support for:
  - AES-CBC
  - HMAC SHA-256

- **Kullanım**

Projemize dist/libsignal-protocol.js dahil edilmeli

- **Kurulduğunda**

Bir Client uygulamaya kayıt olduğu zaman identity keys, registration id ve prekeys üretilmesi gerekiyor.

```
var KeyHelper = libsignal.KeyHelper;

var registrationId = KeyHelper.generateRegistrationId();

// Store registrationId somewhere durable and safe.


KeyHelper.generateIdentityKeyPair().then(function(identityKeyPair) {

// keyPair -> { pubKey: ArrayBuffer, privKey: ArrayBuffer }

// Store identityKeyPair somewhere durable and safe.

});


KeyHelper.generatePreKey(keyId).then(function(preKey) {

store.storePreKey(preKey.keyId, preKey.keyPair);

});


KeyHelper.generateSignedPreKey(identityKeyPair,
keyId).then(function(signedPreKey) {

store.storeSignedPreKey(signedPreKey.keyId, signedPreKey.keyPair);
```

```
});


// Register preKeys and signedPreKey with the server
```

- **Oturum Oluşturma**

Libsignal Client identity, prekeys, signed prekeys ve oturum durum bilgis saklamak ve sonra ihtiyaç olduğunda da kullanmak için bir veri taban arayüzüne ihtiyacı olacaktır. Örnek olarak github/libsignal/test/InMemorySignalProtocolStore.js var olan arayüz kullanılabilir.[2]

Bu gereksinim sağladıktan sonra aşağıdaki kod bir oturum oluşturma mekanizması anlatıyor:

```
var store = new MySignalProtocolStore();

var address = new libsignal.SignalProtocolAddress(recipientId, deviceId);



// Instantiate a SessionBuilder for a remote recipientId + deviceId tuple.

var sessionBuilder = new libsignal.SessionBuilder(store, address);



// Process a prekey fetched from the server. Returns a promise that resolves

// once a session is created and saved in the store, or rejects if the

// identityKey differs from a previously seen identity for this address.

var promise = sessionBuilder.processPreKey({

registrationId: <Number>,

identityKey: <ArrayBuffer>,
```

```
signedPreKey: {
keyId : <Number>,
publicKey : <ArrayBuffer>,
signature : <ArrayBuffer>
},
preKey: {
keyId : <Number>,
publicKey : <ArrayBuffer>
}
});

promise.then(function onsuccess() {
// encrypt messages
});

promise.catch(function onerror(error) {
// handle identity key conflict
});
```

3.2.2. Şifreleme

Oturum oluştuktan sonra aşağıdaki libsignal kütüphanesinden sağlanan şifreleme fonksiyonu **SessionCipher** kullanabiliriz.

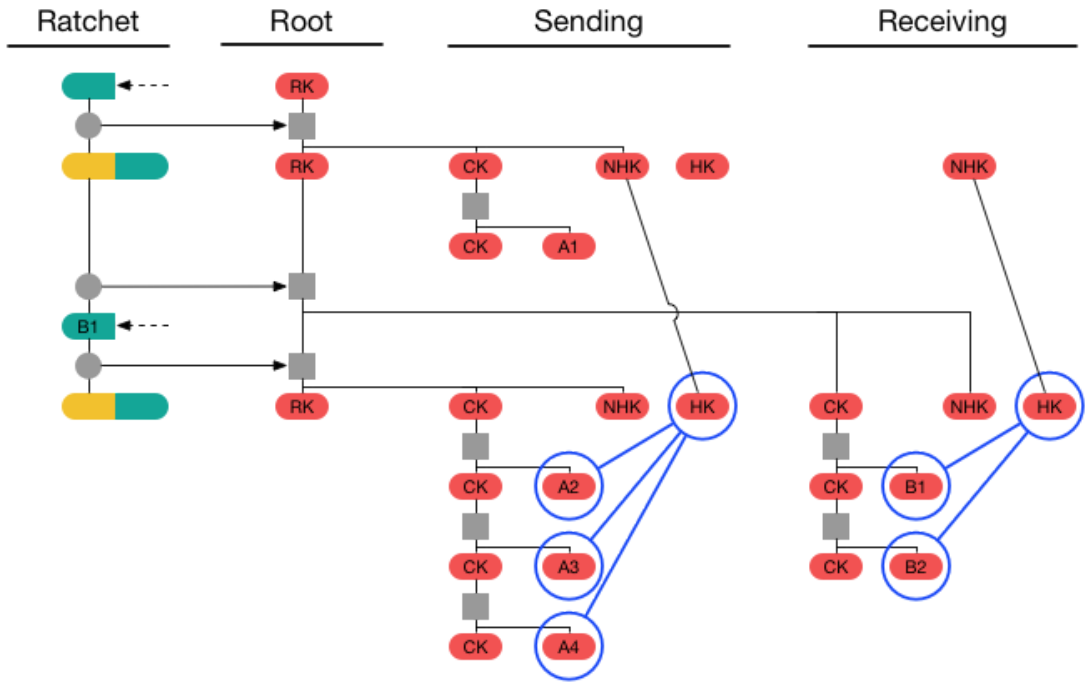
```
var plaintext = "Hello world";

var sessionCipher = new libsignal.SessionCipher(store, address);

sessionCipher.encrypt(plaintext).then(function(ciphertext) {

// ciphertext -> { type: <Number>, body: <string> }

handle(ciphertext.type, ciphertext.body);

});
```

3.2.3. Şifreleme çözme

Ciphertexts iki şekille gelir: WhisperMessage ve PreKeyWhisperMessage.

```
var address = new SignalProtocolAddress(recipientId, deviceId);

var sessionCipher = new SessionCipher(store, address);



// Decrypt a PreKeyWhisperMessage by first establishing a new session.

// Returns a promise that resolves when the message is decrypted or

// rejects if the identityKey differs from a previously seen identity for this

// address.

sessionCipher.decryptPreKeyWhisperMessage(ciphertext).then(function(plaintext) {

// handle plaintext ArrayBuffer

}).catch(function(error) {

// handle identity key conflict
```

```
});


// Decrypt a normal message using an existing session
var sessionCipher = new SessionCipher(store, address);
sessionCipher.decryptWhisperMessage(ciphertext).then(function(plaintext) {
// handle plaintext ArrayBuffer
});
```



Şekil 3.1. Ratcheting

3.2.4. Uçtan Uca şifreleme ile Sesli arama çalışma şekli

1. Şifreli sesli arama yapmak için bir oturum açılmalı. Yukarıdaki anlattığım Oturum açma bölümünde güvenli bir oturum açılır -olmadığı takdirde-.

2. Açılmış oturumdan rastgele 32-byte SRTP anahtarı üretilir.

3. Açılmış oturumdan ikinci cihaza SRTP anahtarı şifreli mesaj olarak gönderiyor. Gönderilen mesaj bir gelen sesli arama sinyali üretir.

4. İkinci taraftaki olan kullanıcıyı gelen sesli aramaya cevap verirse, sesli aramayı onaylanmış olur ve SRTP şifreli sesli arama doğar.

3.3. RTP ve SRTP Nedir?

RTP (Real-time Transport Protocol), gerçek zamanlı ses, görüntü ya da simülasyon verilerinin uçtan uca taşınmasını sağlayan protokoldür. Bu protokol IETF nin Audio-Video Transport çalışma grubu tarafından geliştirildi. RTP geniş ölçüde telefon , video telekonferans uygulamaları ve web tabanlı bas-konuş özellikleri gibi streaming media gerektiren iletişim ve görsel sistemlerde kullanılır.

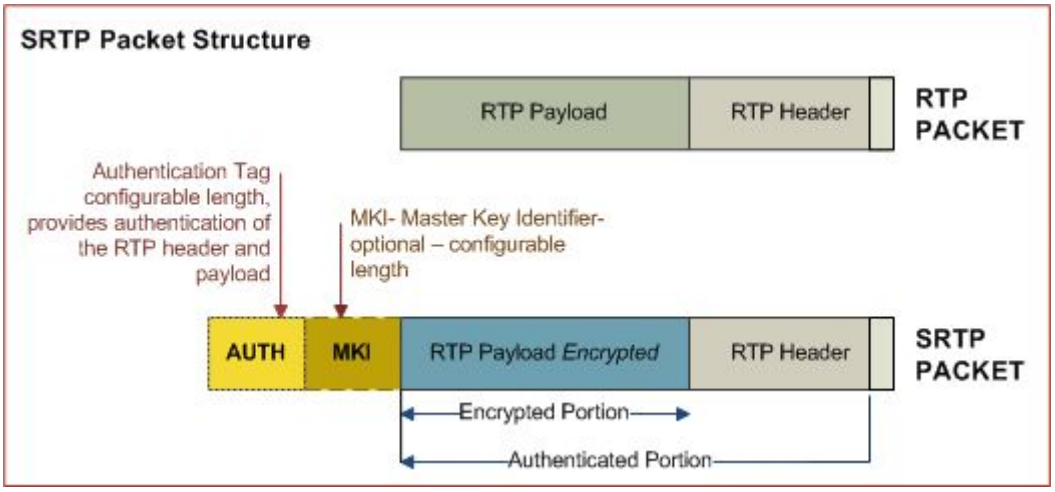
Secure RTP (SRTP) RTP nin güven versiyonudur.

RTP çoklu ortam verilerinin gerçek-zamanlı(real-time), uçtan uca (end-to-end) transferi için tasarlanmıştır. Protokol bir IP network üzerindeki veri iletiminde verilerdeki sıra bozukluğunu tespit eder ve jitter (network üzerinde paketlerin geliş süresindeki, düzenindeki değişiklik) kompanzasyonu için kolaylık sağlar.

Gerçek zamanlı çoklu ortam streaming uygulamaları zamanında bilgileri teslim etmeyi gerektirir ve bu amacı gerçekleştirmek için bazı kaybolan paketleri tolere edebilmelidir. Örneğin audio (ses) uygulamasında kaybolan bir paket ikinci bir paketinin kaybolmasına neden olabilir.

- Protokol Bileşenleri
  - RTP iki alt protokolü tanımlar:
    - Veri Transfer Protokolü: gerçek zamanlı çoklu ortam verisinin transferiyle ilgilenir. Bu protokol tarafından sağlanan bilgi senkronizasyon için tarih bilgisi, kaybolan paketlerin denetimi için sıra numarası ve verinin kodlanmış formatını gösteren payload formatını içerir.

    - Gerçek Zamanlı Kontrol Protokolü: Servis önceliği (QoS) ile ilgili geri bildirimler ve ortam streamleri arasında senkronizasyonu belirtmek için kullanılır.



Şekil 3.2. SRTP Alt yapısı

BÖLÜM 4. KAYIT ve GİRİŞ İŞLEMİ: LOG IN - SIGN UP

Uygulamanın güvenlik seviyesi arttırmak için uygulamayı her açtığında ya da telefon ekranı her kapattıktan sonra ve uygulamayı açmak denediğinde kullanıcıdan giriş LogIn istenecektir.

Giriş işlemi için iki seçenek sunar:

1. E-posta ve şifre
2. Yüz Tanıma

4.1. E-posta ve Şifre

Firebase'in sunduğu hizmetlerden biri login ve signup Android ya da IOS işletim sistemleriyle integrasyon yaparak bir giriş ekranı oluşturabilir.

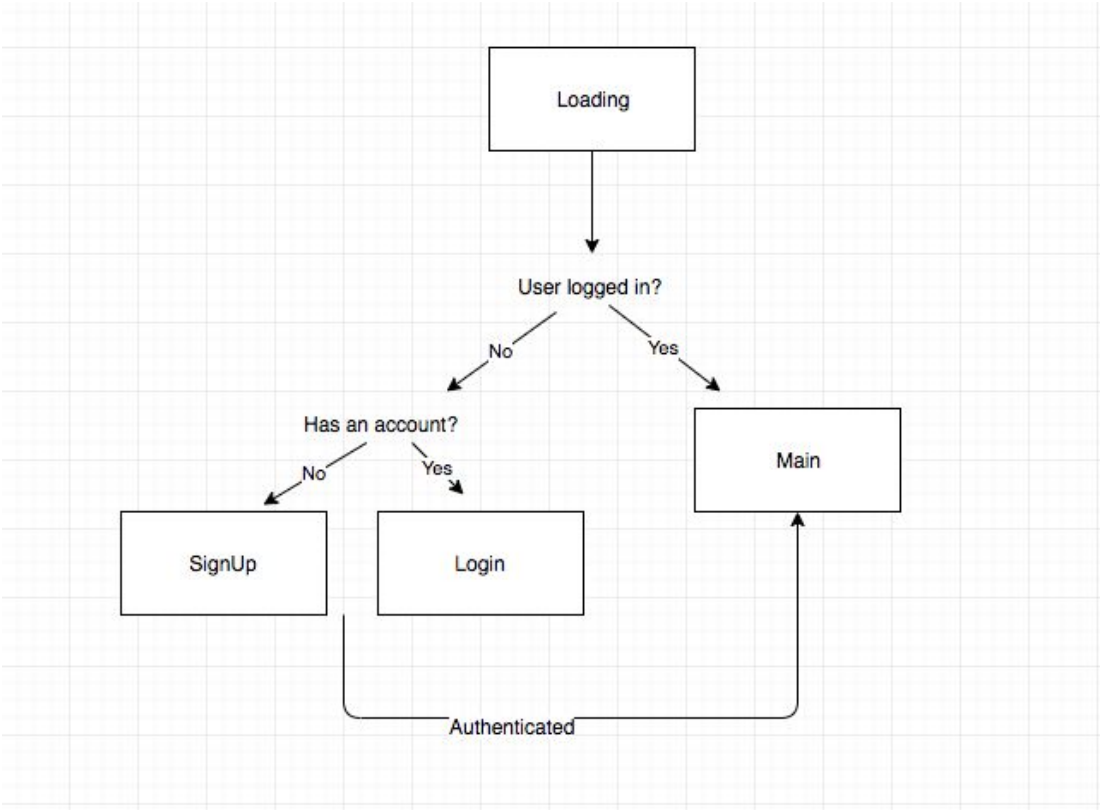
Böyle bir arayüz ve fonksiyon kullanmak ve yapmak için basitçe bir firebase veri tabanın içinde kullanıcı ID sı ve Şifresi saklanarak ve istediğimiz zaman çıkabilme imkanı otomatik olarak sunar.

Kullanıcıların şifreleri firebase tarafından güvenli bir şekilde kendi protokolü ile transfer ediyor.

Firebase projemize entegre şekli gösteren kod

```
import auth from '@react-native-firebase/auth';
import { firebase } from '@react-native-firebase/auth';

export default class signUp extends React.Component {
  state = {email: '', password: '', errorMessage: null};
  signupNow = () => {
    if (this.state.email && this.state.password) {
      firebase
        .auth()
        .createUserWithEmailAndPassword(this.state.email, this.state.password)
        .then(() => this.props.navigation.navigate('Home'))
        .catch(error => this.setState({errorMessage: error.message}));
    } else {
      ToastAndroid.show('Please fill all the fields!', ToastAndroid.LONG);
    }
  };
```

Sign Up code with Firebase React Native



Şekil 4.1. Firebase Database Diagram for Android and IOS

4.2. Yüz Tanıma

Firebase ML Kit sunduğu Makine öğrenme seti kütüphane olarak Firebase veri tabanı gibi indirip projemize entegre edebiliriz. Aşağıdaki kodda ML Kit kütüphanesi kullanarak yazılan java kodu ML Face Detection ve Recognition Kit' i ile yapabileceğimiz işlemler gösteriyor

```
for (FirebaseVisionFace face : faces) {
    Rect bounds = face.getBoundingBox();
    float rotY = face.getHeadEulerAngleY();  // Head is rotated to the right rotY degrees
    float rotZ = face.getHeadEulerAngleZ();  // Head is tilted sideways rotZ degrees

    // If landmark detection was enabled (mouth, ears, eyes, cheeks, and
    // nose available):
    FirebaseVisionFaceLandmark leftEar = face.getLandmark(FirebaseVisionFaceLandmark.LEFT_EAR);
    if (leftEar != null) {
        FirebaseVisionPoint leftEarPos = leftEar.getPosition();
    }

    // If contour detection was enabled:
    List<FirebaseVisionPoint> leftEyeContour =
            face.getContour(FirebaseVisionFaceContour.LEFT_EYE).getPoints();
    List<FirebaseVisionPoint> upperLipBottomContour =
            face.getContour(FirebaseVisionFaceContour.UPPER_LIP_BOTTOM).getPoints();

    // If classification was enabled:
    if (face.getSmilingProbability() != FirebaseVisionFace.UNCOMPUTED_PROBABILITY) {
        float smileProb = face.getSmilingProbability();
    }
    if (face.getRightEyeOpenProbability() != FirebaseVisionFace.UNCOMPUTED_PROBABILITY) {
        float rightEyeOpenProb = face.getRightEyeOpenProbability();
    }

    // If face tracking was enabled:
    if (face.getTrackingId() != FirebaseVisionFace.INVALID_ID) {
        int id = face.getTrackingId();
    }
}
```

Yüz tanıma bilgileri Firebase ile elde etmek

Tam ayarlama bilgileri buradan erişilir : [3]

4.2.1. RSA Algoritması İle Yüz Tanıma Resmi şifreleme ve Çalışma Mantığı

Önceki adımda elimize Yüzün resmi görüntü işleme verileri elde ettiğimizde RSA şifreleme algoritması kullanarak şifrelenecek ve veriler Lokal olarak oluşturduğum veritabanı telefon cihazı içinde saklanacaktır.

Bir kullanıcı yüz tanımıyla giriş denediği an aynı şekilde kullanıcı yüz resmini alınacak ve işlenecek ve sonra da veriler bizim veritabanımızdaki sakladığımız resim verileri ile karşılaştırarak giriş sağlanacaktır ya da reddedilecektir.

4.2.1.1. RSA Algoritması ve Çalışma Prensibi

Bir açık anahtarlı şifreleme yöntemi olan RSA, 1977 yılında Ron Rives, Adi Shamir ve Leonard Aldeman tarafından bulunmuştur. Şifreleme yönteminin adı da bu üç kişinin soy isimlerinin baş harflerinden oluşur.

- **Çalışması:**

1. Yeterince büyük iki adet asal sayı seçilir: Bu sayılar örneğimizde p ve q olsunlar.
2. n=pq hesaplanır. Buradaki n sayısı iki asal sayının çarpımıdır ve hem umumî hem de hususî şifreler için taban (modulus) olarak kabul eder.
3. Totient fonksiyonu hesaplanır. Bu örnek için çarpanların ikisi de asal sayı olduğu için $\varphi(n) = (p-1)(q-1)$ olarak bulunur.
4. Hesaplanan totient fonksiyonu değeri ($\varphi(n)$) ile aralarında asal olan öyle bir e sayısı alınır ki $1 < e < \varphi(n)$ olmalıdır. Bu seçilen e sayısı umumî anahtar olarak ilan edilebilir.
5. d gibi bir sayı hesaplanır ki bu sayı için şu denklik geçerli olmalıdır : $de \equiv 1 \mod (\varphi(n))$. Bu d değeri hususî şifre olarak saklanır.

Şifreleme işlemi: c = m^e mod n

Şifrenin Açılması: m = c^d mod

BÖLÜM 5. SONUÇLAR VE ÖNERİLER

Kriptoloji dalı geniş bir dal olmamasına rağmen önemini epey derecede büyüktür. Şifreleme algoritmaları kendi temelinden aynı kavramlardan oluşur bir değişiklik ile algoritmaların performansı güvenlik açısından yada hız açısından algoritmalar geliştirilebilir.

Araştırmalarıma göre ve tasarımımda ulaştığım sonuca farklı farklı şifreleme algoritmaları vardır ama tasarlanacak uygulamaya göre ve uygulamayı hangi ortamda çalışılacağını hangi algoritma kullanacağımızı etkiler ve ana yol oynar.

Sıfırdan bir şey oluşturmak her zaman iyi olamayabilir bunun için var olan hazır açık kaynak kütüphaneler bizim işimiz kriptocu olarak kolaylaştırır ve internet güvenliği daha üst seviyeye taşır.

Var olan ve yeni teknolojiler React Native gibi eski ve güncellenen şifreleme algoritmalar entegrasyon yapılabılır ve zorundadır.

Sonuç olarak React Native Cross platformu ve Signal Protocol ile uçtan uca şifreleme ile sesli haberleşme mobil uygulaması tasarımı yapıldı. Veritabanı tasarımında SQLite ve Firebase Cloud veritabanı kullanıldı. Ve yüz tanımı için makine öğrenme ML kit hazır firebase kütüphanesi ve elde edilen veriyi güvence telefon içinde saklamak için RSA algoritması kullanmayı karar alındı.

Uçtan Uca şifreleme yöntemi uygulamasını ne kadar basit olduğunu görebiliriz, internet güvenliği ve insanların şahsi bilgi güvenliği büyük derecede önemlidir ve ciddi bir şekilde insanların bu konu hakkında bilinç oluşturulması gerekiyor ki ünlü kurumlar ve devletler bu yöntemi kullanmaya zorlansınlar.

KAYNAKLAR

[1] https://medium.com/kodcular/react-native-nedir-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-ne-olacak-dikkat-flutter-%C3%A7%C4%B1kabilir-96c8aeacbc29

[2] https://github.com/signalapp/libsignal-protocol-java

[3] https://firebase.google.com/docs/ml-kit/android/detect-faces

ÖZGEÇMİŞ

Rıdvan Alhourani, 01.01.1996 de Şam'da doğdu. İlk, orta ve lise eğitimini Jober'de tamamladı. 2014 yılında Al-Saade Fen Lisesi'nden mezun oldu. 2015 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nü kazandı. 2017 yılında SAÜ Bilişim Fakültesinde yazılım stajını yapmıştır. 2015 yılında başladığı Sakarya Üniversitesi Bilgisayar Mühendisliği bölümünde lisans eğitimini sürdürmektedir.

BSM 401 BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI

KONU : Uçtan Uca Şifreleme ile sesli haberleşme uygulaması
ÖĞRENCİLER (G1512.10575/Rıdvan Alhourani):

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma klavuza uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problemin tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımın gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişki modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımın sızma testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN : Doç. DR.Ünal Çavuşoğlu
DANIŞMAN IMZASI: