TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GÜVENİLİR İŞLETİM ORTAMI TEKNOLOJİSİ KULLANILARAK MOBİL KİMLİK UYGULAMASI GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Kaan KÜÇÜK

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. Kemal BIÇAKCI

NİSAN 2019

Fen Bilimleri Enstitüsü Onayı

……..
Prof. Dr. Osman EROĞUL
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksininlerini sağladığını onaylarım.

…….
Prof. Dr. Oğuz ERGİN
Anabilimdalı Başkanı

TOBB ETÜ, Fen Bilimleri Enstitüsü'nün 151111029 numaralı Yüksek Lisans Öğrencisi **Kaan KÜÇÜK** 'ün ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"GÜVENİLİR İŞLETİM ORTAMI TEKNOLOJİSİ KULLANILARAK MOBİL KİMLİK UYGULAMASI GELİŞTİRİLMESİ"** başlıklı tezi **11/04/2019** tarihinde aşağıda imzaları olan jüri tarafından kabul edilmiştir.

Tez Danışmanı : **Prof. Dr. Kemal BIÇAKCI** .....
TOBB Ekonomive Teknoloji Üniversitesi

Jüri Üyeleri : **Prof. Dr. Ali Aydın SELÇUK (Başkan)** .....
TOBB Ekonomi ve Teknoloji Üniversitesi

Prof. Dr. Kemal BIÇAKCI .....
TOBB Ekonomi ve Teknoloji Üniversitesi

Dr. Öğretim Üyesi Özgür ERGÜL .....
Çankaya Üniversitesi

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Kaan KÜÇÜK

ÖZET

Yüksek Lisans

GÜVENİLİR İŞLETİM ORTAMI TEKNOLOJİSİ KULLANILARAK MOBİL KİMLİK UYGULAMASI GELİŞTİRİLMESİ

Kaan Küçük

TOBB Ekonomi ve Teknoloji Üniveritesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Kemal Bıçakcı

Tarih: Nisan 2019

Nüfus cüzdanlarının yerine kullanılmaya başlanan yeni kimlik kartları elektronik ortamda da kimlik doğrulama amacıyla kullanılabilecek teknolojiye sahiptir. Öte yandan, akıllı kart okuyucusu gereksinimi, yazılım kütüphanesinin mobil işletim sistemlerine uyumlu olmaması gibi nedenlerden ötürü yeni kimlik kartlarının akıllı telefonlar üzerinden kullanımı çok kolay değildir. Bu çalışmada, söz konusu teknik zorlukları aşmak için Android ortamında çalışan bir Mobil Kimlik çözümü tanıtılmaktadır. Çözümümüz, iki temel bileşenden oluşmaktadır. Mobil telefon uygulaması, Güvenilir İşletim Ortamı teknolojisini kullanarak açık anahtar ve özel anahtar çifti üretir. Özel anahtar, Güvenilir İşletim Ortamından çıkmaz ve bu sayede Zengin Android ortamına ilişkin tehditlere karşın korunmuş olur. Masaüstü uygulaması ise, mobil telefondan gelen açık anahtarı kullanıcının kimlik kartında mevcut özel anahtar ile imzalatır (bir anlamda, sayısal sertifika üretmiş olur) ve üçüncü partiler kimlik doğrulama amacıyla mobil telefon uygulamasının ürettiği sayısal imzaları doğrularken açık anahtar ile kullanıcı arasındaki bağın güvenli bir şekilde kurulmasını sağlar. Geliştirdiğimiz çözümün, başta E-devlet uygulamaları olmak üzere mevcut mobil kimlik doğrulama çözümlerine güvenli ve kullanışlı bir alternatif olduğu düşünülmektedir.

Anahtar Kelimeler: Kimlik kartları, Mobil kimlik, Mobil güvenlik, Kimlik doğrulama, E-Devlet, Sayısal imza, Güvenilir bilişim, Açık-Anahtar kriptografisi, Açık-Anahtar altyapısı.

ABSTRACT

Master of Science

DEVELOPING A MOBILE IDENTITY APPLICATION BY USING

TRUSTED EXECUTION ENVIRONMENT

Kaan Küçük

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Computer Engineering Programme

Supervisor: Prof. Kemal Bıçakcı

Date: April 2019

New identity cards in Turkey, which are started to be used instead of old ones, have the technology to be used for authentication purposes also in electronic environments. On the other hand, it is not straightforward to use the new ID cards on smartphones due to the smart card reader requirements and compatibility issues with respect to software library. This study introduces a Mobile Identity solution that works in the Android environment to overcome these technical challenges. Our solution consists of two basic components. The mobile phone application generates a public key and private key pair using Trusted Execution Environment technology. The private key does not exit the Trusted Execution Environment and is thus protected from threats from the Rich Android environment. With the desktop application, the public key obtained from the mobile phone is signed with the private key on the user's identity card (in a sense, a digital certificate is generated) and the third parties could authenticate digital signatures generated by the mobile phone application for authentication purposes by ensuring the secure connection between the public key and the user. The solution we have developed is considered to be a secure and usable alternative to existing mobile authentication solutions, especially for e-government applications.

Keywords: Identity cards, Mobile identity, Mobile security, Authentication, E-Government, Digital signing, Trusted computing, Public-Key cryptography, Public-Key infrastructure.

TEŞEKKÜR

Bu çalışmanın konusunun belirlenmesinde ve çalışmanın gerçekleştirilme sürecinde bilgisini, tecrübesini ve değerli zamanını esirgemeyerek her fırsatta yardımcı olan değerli hocam Prof. Dr. Kemal BIÇAKCI'ya, çalışmamın başarıya ulaşması için sunduğu kaynaklar ile yardımcı olan TÜBİTAK UEKAE E-Kimlik Birimi'ne, benden desteğini hiçbir zaman esirgemeyen, gösterdiği sabır ile her zaman yanımda olan sevgili eşim Rabia KÜÇÜK'e ve hayatım boyunca eğitimime eşsiz bir özen gösterip beni bugünlere getiren sevgili annem Nesime KÜÇÜK'e sonsuz teşekkürler.

İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

KISALTMALAR

CSR : Certificate Signing Request
GİO : Güvenilir İşletim Ortamı
JVM : Java Virtual Machine
NFC : Near Field Communication
OP-TEE : Open Portable Trusted Execution Environment
RSA : Rivest-Shamir-Adleman açık anahtarlı şifreleme yöntemi
TCKK : Türkiye Cumhuriyeti Kimlik Kartı

RESİM LİSTESİ

Sayfa

1. GİRİŞ

Vatandaşlara dağıtımının 2023 yılında tamamlanması planlanan çipli yenil nesil kimlik kartları açık anahtar altyapısı kullanan güvenli akıllı kartlar hükmündedir ve bu kartlar ile elektronik ortamda bir verinin sayısal imzalama işlemini yapabilmekteyiz. Bir akıllı kart okuyucusu ve akıllı kartın sunduğu uygulama programlama ara yüzü ile gerçekleştirebildiğimiz imzalama işlemi, imzalanan verinin bütünlüğünü koruduğu gibi elektronik ortamda sahteciliğin de önüne geçerek veri güvenliğimizi sağlamamıza yardımcı olur. Sayısal imzalama, kimlik doğrulama amaçlı geliştirilen kriptografik protokollerde de yoğun olarak kullanılan bir teknolojidir. Veri güvenliğimize ve kimlik doğrulama işlemine sağladığı katkıya rağmen, her imzalama işlemi için gerekli olacak akıllı kart okuyucusu ve imzalama işlemini gerçekleştirecek program ve programın çalışacağı bir bilgisayar zorunluluğu, kullanılabilirlik problemlerine ve işlem gerçekleştirme zamanında kayıplara yol açacağından bilhassa mobil platformlarda akıllı kartların kullanımı günümüzde çokça tercih edilmemektedir.

Teknolojik gelişmeler ile beraber, mobil platform kullanımı günbegün artmaktadır [1]. Artan mobil platform kullanımı ile beraber birçok kamu veya özel kurum-kuruluşlar kendi sistemlerini mobil platformlara entegre etmektedir. Hızlı işlem yapabilme ve her an bilgiye erişim kolaylığı sağlayan mobil cihazlarımıza entegre olan sistemler arttıkça da geleneksel anlamda bilgisayar kullanımı azalmaktadır. Artan mobil cihaz kullanımı ve mobil platformlarda yapılan iş ve işlemler bize mobil cihaz ve uygulama güvenliğinin önemini göstermektedir [2].

Günümüzde mobil cihazlarımızda kullanılabilen mobil kimlikler (mobil imzalar), mobil operatörler aracılığı ile dağıtılan açık anahtar altyapısı kullanan SIM kartlar üzerinden yapılabilmektedir. Elektronik (mobil) imza üretebilen ve özel anahtarınızın tutulduğu özel SIM kartlar aracılığıyla, verilerimizi imzalayabilmekteyiz. Fakat, bu imzalama işleminde, imzalama ve doğrulama yapacak olan tüm sistemler üçüncü şahıs olarak mobil operatöre entegre edilmek zorundadır. Bu da sistem geliştiricilere ve kullanıcıya fazladan iş yükü anlamına gelmektedir [2]. Ayrıca, SIM kart tabanlı çözümlerin işlemci ve bellek sınırlamaları gibi bilinen diğer dezavantajları mevcuttur.

Açık anahtar altyapısı kullanan kimlik kartlarımızı daha verimli ve kolay şekilde kullanılabilmesini hayatımızın her alanında, heran ve her yerde kolayca kullanmaya başladığımız mobil cihazlarımıza entegre ederek sağlayabiliriz. Fakat, oluşturulacak mobil kimliğin cihaz üzerinde güvenle saklanabilmesi ve üçüncü şahıslara karşı gizliliğinin sağlanabilmesi problemi karşımıza çıkmaktadır. Bu çalışmamızda bu probleme Güvenilir İşletim Sistemi teknolojisi tabanlı açık-kaynak OP-TEE ortamı ve Android İşletim Sistemi üzerinde çalışan bir mobil kimlik uygulaması geliştirerek bir çözüm getirdik. Böylelikle, farklı sistemlere gerek kalmadan akıllı kartlarımız ile mobil kimliklerimizi entegre ederek, bu kimlikleri mobil cihazlarımızda güven ile saklayabilecek bir altyapı kurulmuş olmaktadır [3, 4].

1.1 Tezin Amacı

Günümüzde, elektronik ortamdaki sistemlere (Web sitesi, masaüstü uygulama vb.) giriş için gerekli kimlik doğrulamasını, açık anahtar altyapısını kullanan kimlik kartlarımız üzerinden yapabilmekteyiz. Bu durumda kimlik doğrulama için gerekli standart kart okuyucusu zorunluluğu, akıllı kimlik kartlarının mobil cihazlarda kullanımını zora sokmaktadır. Mobil ortamlarda yapılan iş ve işlemlerin öneminin artması ile akıllı kimlik kartları üzerinden kimlik doğrulama işleminin yapılamaması mobil ekosistemin eksikliği olacaktır. Çalışmamızda, Güvenilir İşletim Ortamı teknolojisinin sunduğu güvenlik özellikleri kullanılarak akıllı kimlik kartı üzerinden oluşturulan mobil kimlik ile yeni nesil kimlik kartlarımızı mobil ekosisteme kazandırılması amaçlanmıştır.

1.2 Literatür Araştırması

Literatürde, çalışmamızın çıkış noktası olan açık anahtar altyapısına sahip akıllı kimlik kartlarının mobil cihazlar üzerinde mobil kimliklere dönüştürülmesi işleminde tavsiye edilen adımlar ve mobil kimliği oluşturan yeni türetilmiş sertifikaların güvenli bir şekilde Güvenilir İşletim Ortamı'nda nasıl saklanabileceğine yönelik yapılmış çalışmalar bulunmaktadır. Literatürdeki bu çalışmalar, Mobil Kimlik Uygulamamızdaki bazı iş akışlarının kararının verilmesinde yön vermiştir. Ayrıca, bu çalışmaların eksik yönleri de incelenmiştir. İncelemelerimiz sonucunda, bu eksik yönler çalışmamızda göz önünde bulundurarak kendi fikirlerimizi katarak

tamamlandığı gibi, katılmadığımız bilgilerde yeni sunduğumuz çözümler ile neden katılmadığımızı göstermiştir.

Kaynak [5]'te, Ulusal Standartlar ve Teknoloji Enstitüsü (National Institute of Standards and Technology, United States)' ne göre açık anahtar altyapısına sahip bir kimlik kartı açık anahtarından türetilecek yeni bir sertifikanın, mobil kimlik olarak kullanılmak üzere nasıl türetileceği, bakım ve yenileme işlemlerinin nasıl yapılacağı ve güvenliği sağlanmış bir şekilde mobil cihaz üzerinde nasıl saklanacağı konularında tavsiyeler sunulmuştur. Bu tavsiyelerin içerisinde Güvenilir İşletim Ortamı'nın sunduğu kaynaklar ve güvenlik özelliklerinden bahsedilmemiştir. Çalışmamızın temel teknolojisi Güvenilir İşletim Ortamı'nın sunduğu özellikler ile varolan bu tavsiyelere uyum sağlanabileceği görülmüştür.

Kaynak [5] göre, akıllı kimlik kartından türetilmiş kişisel kimlik doğrulama güvence (Credential) bilgisinin kullanılması, bir mobil cihazı mobil kimlik olarak etkinleştirmenin en olası yoludur. Akıllı kimlik karttan türetilmiş kimlik doğrulama güvence bilgisini mobil cihaza vermek yerine, kimlik kartının temaslı veya temassız ara yüzünü kullanarak, mobil cihazla birlikte kullanılması pratik olabilir. Bir mobil cihazına akıllı kart okuyucusunun entegre edilmesi için cihazın yeterince büyük olmaması, mobil cihaz ve akıllı kimlik kartı arasındaki haberleşmenin alternatif yollar ile sağlanması gerekliliği doğurmaktadır. Bu tarz alternatif yollar günümüzdeki uygun cihazlarda bulunmaktadır. Mobil cihazların içine akıllı kart okuyucusunu entegre etmek yerine, mobil cihazın USB ara yüzüne kart okuyucusu takılarak kimlik kartı kullanılabilir. Ayrıca, yakın temas teknolojisine sahip akıllı kartın, bir çok mobil cihaza entegre olmuş NFC modülü ile de kullanılması mümkündür. Fakat her kimlik kartı temassız iletişime sahip bir teknolojiye sahip değildir. Nitekim çalışmamızda kullanılan kimlik kartı temassız iletişim gerçekleştirebilecek bir donanıma sahip değildir. Bu durumda kullanıcı her defasında USB arabirimi üzerinden akıllı kart üzerinde işlem yapabilecektir. Fakat, akıllı kimlik kartın temassız iletişim donanıma sahip olsada sürekli kart kullanımı (NFC teknolojisi olmaması durumda ek olarak kart okuyucu kullanımı) son kullanıcıya ek, kullanımda pratik olmayan durumlar çıkaratacaktır. Çalışmamızda olduğu gibi literatürde de tavsiye edilen, bu tip pratik olmayan durumlarda akıllı kimlik üzerinden türetilmiş bir kişisel kimlik doğrulama güvence bilgisininin oluşturulabileceğidir.

Ulusal Standartlar ve Teknoloji Enstitüsü'ne göre kişisel kimlik doğrulama altyapısı ve uygulamaları ile birlikte çalışabilirliği sağlamak için, türetilmiş kişisel kimlik doğrulama güvence bilgilendirmesinin temeli olarak açık anahtar altyapısı teknolojisi seçilmiştir. Türetilmiş bir kimlik doğrulama sertifikası için açık anahtar altyapısının kullanılması güvence düzeyini artıracağı belirtilmiş ve sertifika için X.509 açık anahtar sertifika politikası tavsiye edilmektedir [5]. Bu tavsiyeye göre çalışmamızda akıllı kimlik kartından türetilecek kimlik doğrulama bilgisi için mobil cihazımızda açık anahtar altyapısını kullanan RSA şifreleme algoritması kullanılmıştır. Mobil cihaz üzerinde ürettiğimiz RSA anahtar çifti, türetilecek kimlik doğrulama güvence bilgisinde kullanılmıştır. Ayrıca türetilmiş kişisel kimlik doğrulama güvence bilgisi için X.509 sertifika politikası kullanılmıştır.

Çizelge 1.1 : Tavsiye edilen hayat döngüsü gereksinimlerinin çalışmamız koşulları ile karşılaştırılması.

	Tavsiye Edilen	Çalışmamız	Karşılaştırma
İlk Yayınlama	Türetilmiş kimlik doğrulama sertifikası bir kurum, kuruluş veya ağ üzerindeki sisteme başvuru üzerine elde edilir.	Akıllı kimlik kartı sahibi, çalışmamızdaki uygulamaları kullanarak türetilmiş kimlik doğrulama sertifikasını elde eder.	Çalışmamızda GİO teknolojisinin sunduğu güvenlik özellikleri ile harici bir kurum, kuruluş veya sisteme gerek yoktur.
Bakım	Türetilmiş sertifikanın bakım işlemleri için yine harici bir kurum, kuruluş veya sisteme sahiptir.	Türetilmiş kimlik doğrulama sertifikasının bakım çalışmalarını kişi mobil cihazı üzerinden yapabilir.	GİO teknolojisi ile yeniden kimlik doğrulama sertikası üretme, değiştirme, iptal bakım etkinliklerini kişi çalışmamızda kolaylıkla gerçekleştirir.
Akıllı Kimlik Kartı ile Bağlantı	Türetilmiş sertifikanın bağlı olduğu akıllı kimlik kartı üzerindeki değişikleri kontrol edecek bir mekanizmanın bulunması gerekmektedir.	Türetilen sertifika akıllı kimlik kart üzerinde yapılan imzalama işlemi ile gerçekleştirilir. Akıllı kimlik kartının üzerindeki değişikliklerde türetilen sertifika çalışmayacaktır.	Çalışmamızda akıllı kimlik kartından türetilen sertifikanın üretilme işlemi içinde bir kontrol mekanizması zaten barındırır.

Kaynak [5]'e göre türetilmiş kişisel kimlik doğrulama güvence bilgisinin hayat döngüsü için gereksinimler ve çizelge 1.1'de bahsedilen karşılaştırmaların detayları aşağıdaki gibidir:

1-) İlk Yayınlama: Elde edilen akıllı kimlik kartındaki kimlik doğrulama anahtarı kullanılarak başvuru sahibinin kimliğinin doğrulaması yapılır. Kimlik doğrulaması akabinde başvuru sahibine türetilmiş kimlik doğrulama güvence bilgisi verilir. Bu kimlik bilgisi MikroSD, USB harici bellek kartları gibi bir donanımsal veya yazılımsal güvenlik belirtecinde bulunur. Bu belirteçler vasıtasıyla, mobil cihazlarımıza bu kimlik bilgisi sertifika olarak yerleştirilebilinir. Ayrıca, başvuru sahibi ilk talebini SSL/TLS protokolü ile korunmuş bir ağ üzerinden de yapabilir.

Çalışmamızda, akıllı kimlik kartından mobil kimlik doğrulama sertifikasının ilk oluşturulması için harici bileşenlere gerek yoktur. Güvenilir İşletim Ortamı'nın sunduğu güvenlik olanakları kullanılır. Tüm kriptografik işlemler ve güvenli depolama alanının izole olarak mobil cihaz üzerinde çalışması bize üçüncü bir şahsın gerekliliğini ortadan kaldırıyor.

2-) Bakım: Türetilmiş kişisel kimlik doğrulama güvence bilgisi, asimetrik kriptografik kimlik bilgilerine uygulanabilir tipik bakım etkinlikleri gerektirebilir. Bu bakım etkinlikleri, yeniden anahtarlama, değiştirme ve iptal etme işlemlerini içerir. İşlemleri talep eden kullanıcı bunu şahsi olarak veya SSL/TLS protokolleri ile korunmuş bir ağ üzerinden de gerçekleştirebilir. Eğer türetilmiş kimlik doğrulama güvence bilgisi, özel anahtarın dışa aktarılmasına izin vermeyen bir donanım şifreleme belirtecinde oluşturulmuş ve depolanmışsa türetilmiş kimlik doğrulama sertifikanın iptali isteğe bağlıdır. Diğer tüm donanımsal ve yazılımsal belirteçlerde bulunan sertifikanın iptal edilmesi zorunludur. Akıllı kimlik kartının kaybolması, çalınması veya hasar görmesi de dahil olmak üzere, kullanıcı kimlik kartını yeni bir kimlik kartı ile değiştirdiğinde, türetilmiş kimlik doğrulama güvence bilgisi etkilenmez. Bu durumda türetilmiş kimlik doğrulama bilgisi, bağlı olduğu sistemlere giriş için kullanılmaya devam edebilecektir.

Çalışmamızda, mobil kimlik sertifikasının geçerlilik süresinin dolması, çalınma olasılığının düşünülmesi veya herhangi bir durumda ötürü yenilenme ve iptal edilmesi istendiğinde gerekli çalışmalar yapılmış ve uygulamamız bu tarz isteklere cevap verebilecek nitelikte geliştirilmiştir. Fakat, akıllı kimlik kartının çalınması veya

yenilenmesi durumda mobil kimlik cihazın açık anahtarının akıllı kart üzerinde imzalanmasından oluşturulduğundan mobil kimliğinde iptal edilerek tekrar oluşturulması gerekecektir. Yine birinci madde de olduğu gibi, bakım işlemleri için kullanıcının bir kurum veya kuruluşa istekte bulunması gerekmez. Güvenilir İşletim Ortamı'nın sunduğu özellikler ile kullanıcı bunu kendi başına yapabilir.

3-) Akıllı Kimlik Kartı ile Bağlantı: Bir türetilmiş kişisel kimlik doğrulama belgesi vere kişi yalnızca başvuru sahibin kimlik kartı ile ilgili bilgilere, kimlik kartını veren kuruluştan edindiği taktirde başvuru sahibine türetilmiş kimlik doğrulama belgesini verecektir. Özellikle türetilmiş kimlik doğrulama belgesini veren kurum veya kuruluş, kimlik kartını sonlandırılıp sonlandırılmadığını veya türetilmiş kimlik doğrulama bilgilerinde görünecek olan kişi hakkında bilgi değişip değişmediğini belirlemek için kimlik kartı veren kuruluşla periyodik olarak kontrol etme mekanizmasına sahip olacaktır. Bu kontrol mekanizma ve kimlik kartına sahip olan kişinin bilgilendirme durumları da kaynak [5]'te bildirilmiştir.

Çalışmamızda oluşturulan mobil kimlik, Güvenilir İşletim Ortamı'nda üretilen RSA anahtar çiftinin açık anahtarının akıllı kimlik kartı üzerinde imzalanması ile oluşturulmaktadır. Bu nedenle, kullanıcı kişinin bilgisine gerek kalmadan akıllı kimlik kartı ve mobil kimliği ile bağlantı kurulmuş olur. Ayrıca, bir kontrol mekanizması veya bilgilendirme çalışması yapmaya gerek yoktur. Mobil kimliğin doğrulanması uzak sunucuda akıllı kimlik kartına ait açık anahtar ile olacağından kimlik kartının herhangi bir nedenle yenilenmesi durumunda mobil kimlik ile artık doğrulama yapılamayacaktır. Çalışmamızda uzak sunucunun akıllı kimlik kartının bağlı olduğu sistem içerisinde olduğunu ve akıllı kimlik kartının yenilenmesi durumda uzak sunucuda yer alan kimlik kartının açık anahtarının da yenilendiğini varsayıyoruz.

Tüm bu bilgiler ışığında, akıllı kimlik kartlarının direk olarak mobil cihazlar ile kullanılamaması durumunda, türetilmiş kişisel kimlik doğrulama güvence bilgisinin hayat döngüsü ve güvenlik tavsiyelerini çalışmamız için tartışmış bulunmaktayız. Çalışmamızda, tavsiye edilen kimlik doğrulama bilgisinin oluşturulması, bakımı ve kimlik kartı ile olan bağlantısı konularındaki fikirler ve güvenlik tedbirleri uygulanmış fakat Güvenilir İşletim Ortamı'nın devreye alınmasıyla türetilmiş kimlik doğrulama bilgisinin taşınmasındaki donanımsal veya yazılımsal belirteçler mobil cihaza ait olan Güvenilir İşletim Ortamı olarak belirlenmiş ve ilk yayınlama ve bakım çalışmalarında bahsedilen kurum veya kuruluşlar devreden çıkarılabilmiştir.

Güvenilir İşletim Ortamı teknolojisinin hayatımıza girmesi ile beraber, akıllı kimlik kartlarımızı direk olarak mobil cihazın sunduğu donanımlar sayesinde her defasında kullanmaya gerek kalmadan ve var olan mobil cihazlar için türetilmiş kişisel kimlik doğrulama güvence bilgilerinin üçüncü sahışlardan alınan mobil kimliklere gerek kalmadığı görülmüştür [6]. Güvenilir İşletim Ortamı sayesinde, kriptografik özelliklerine göre açık anahtar altyapısına sahip RSA anahtar çiftleri üretebiliyoruz ve bu anahtar çiftlerini mobil cihazımızda güvenli olarak saklayabilmekteyiz. Güvenilir İşletim Ortamı teknolojisinin mobil cihaz üzerinde, literatürde Zengin İşletim Sistemi (Örneğin Android İşletim Sistemi) ile tam bağımsız olarak donanım seviyesinden ayrı bir şekilde çalışması ürettiğimiz mobil kimliklerin yüksek güvenlikle saklanabildiğini garanti eder. Böylece, Güvenilir İşletim Ortamı teknolojisi ile oluşturulmamış mobil kimlik çözümlerinin dezavantajlarını ortadan kaldırılabildiği gibi, kullanıcıya daha gizli, hızlı, kullanılabilirliği yüksek ve bilgi bütünlüğünün daha fazla korunduğu mobil kimlik çözümlerinin olabileceği görülmüştür.

Kaynak [6], kişilerin devlet iş ve işlemlerinde (Vergi ödeme, oy kullanma, polis hizmetleri vb.), bankacılık, sağlık, finans ve e-ticaret sektörlerinde mobil kimlik doğrulama sistemlerini kullanmaya başladığını göstermektedir. Örneğin Estonya, 202 yılında vtandaşlara hem ulaşım hem de elektronik oylama için bir kimlik bilgisi ile birlikte kişisel kimlik bilgileri sunan akıllı kimlik kart tabanlı dijital kimlik sistemi kullanmıştır. Estonya, 2007'de akıllı kimlik kartlarının cep telefonlarında SIM kartlarla artırılmasını veya değiştirilmesini sağlayan bir mobil kimlik sistemini kullanmaya başladı. Görüldüğü üzere, mobil cihazlar ile yapabildiğimiz bir çok iş ve işlemlerin yanı sıra kimlik kartımız olmayada başlıyor.

Geleneksel kimlik doğrulama ehliyet numarası, pasaport numarası ve nüfus cüzdanı numarası ile yetkili makamlarca kimlikler ile yapılırdı. Yeni dijital kimlik doğrulama teknikleri yukarıda bahsedilen benzersiz numaralara dayanmak yerine açık anahtar altyapısını kullanarak bir güven zinciri oluşturur. Kaynak [6], bu güven zincirinin bir parçası olan açık anahtar altyapısını kullanan akıllı kimlik kartlarından türetilmiş kimlik doğrulama sertifikalarının mobil cihazlarda güvenle nasıl saklanabileceğini incelemiştir. Buna göre bir türetilmiş sertifika mobil cihaz üzerinde Zengin İşletim Ortamı, Güvenilir İşletim Ortamı veya Güvenli Eleman (Secure Element) mekanizması kullanarak saklanabilir. Fakat, bu iki ortamın ve güvenli eleman mekanizmasının kendi aralarında kullanımı son kullanıcı açısından sunduğu

kullanılabilirlik ve güvenlik düzeylerini farklılaştırıyor. Çünkü, Zengin İşletim Ortamı üzerinde yapılacak "root" adı verilen hak yükseltme işlemi, yetkisiz kişilerce türetilmiş kimlik doğrulama sertifikasının çalınmasına sebep oluyor. Ayrıca, Güvenli Eleman mekanizma kullanılarak sağlanacak bir güvenlik, Güvenli Eleman Uygulama Geliştirme Ara Yüzü'nün sunduğu kısıtlı kullanım alanı ve özellikler ile kullanılabilirliği azaltıyor. Bunlara karşılık olarak, Güvenilir İşletim Ortamı'nın sunduğu geniş depolama alanı, güvenlik özellikleri ve kriptografik işlemler türetilmiş sertifikaların mobil cihazlarda güvenle saklanabilmesinin önünü açmaktadır. Güvenilir İşletim Ortamı, Güvenli Eleman mekanizmasına göre geniş bir geliştirme ortamı ve son kullanıcı kullanabilirliğini artırdığı gibi Zengin İşletim Ortamı'nda "root" işlemi yapılsa bile Güvenilir İşletim Ortamı'nın sunduğu güvenilir depolama, güvenilir kullanıcı ara yüzü ve kriptografik işlemler özellikleri ile açık anahtar altyapısı kullanan akıllı kimlik kartlarımızdan yeni sertifikalar türetebilmeyi ve mobil cihaz üzerinde donanım seviyenden başlayan güvenlik mekanizması ile bu sertifikaları saklayabildiğimiz görülmüştür.

2. YÖNTEM ve MATERYAL

Açık anahtar altyapısı kullanan akıllı kartlar günümüzde kimlik kartı, ehliyet gibi yüksek güvenlik gerektiren belgelerde yaygın olarak kullanılmaya başlanmıştır. Beraberinde, birçok sistemin de akıllı kartlarımız ile entegre olmaya başladığını görmekteyiz. Böylelikle sistemler kimlik doğrulama ve yetkilendirmesini bu akıllı kartlar vasıtasıyla yapabilir hale gelmiştir [5]. Fakat, geliştirilecek uygulamalarda akıllı kartların her işlemde akıllı kart okuyucusu tarafından okunması gerekliliği işlemi zor hale getirebilmektedir. Bu çalışmada, mobil platformlar üzerinde bu gerekliliği belli bir süre için sadece bir sefere mahsus kılan bir mobil kimlik oluşturarak, oluşturulan mobil kimlik ile ilgili hassas verileri güvenilir bir ortamda saklayabilme sağlanmıştır. Bu doğrultuda, kullanıcılar için geliştirilen mobil kimlik uygulaması ile kullanımı daha kolay ve daha güvenilir bir çözüm sunulmuştur [7].

Çalışmamızda, aşağıda detaylandırılacağı üzere, ilk olarak uygulamamızın temellerini oluşturacak mobil işletim sistemleri ve güvenilir işletim ortamları incelenmiştir. Ayrıca, bahsedilen işletim ortamlarının uyumlu bir şekilde çalışacağı donanımlar üzerinde inceleme yapılmıştır.

2.1 Mobil İşletim Sistemi

Günümüzde en yaygın olarak kullanan iki mobil işletim sistemi vardır. Android ve iOS. Geliştirilen mobil kimlik uygulamamız için mobil işletim sistemi olan Android İşletim Sistemi seçilmiştir. Çünkü Android İşletim Sistemi açık kaynaklı bir mobil platformdur [8]. Bundan dolayı, yazılım ve donanım geliştiricilere geniş bir çalışma alanı sunabilmektedir. Böylelikle, güncel olarak Android destekleyen donanımlar ve Android ile uyumlu çalışabilen Güvenilir İşletim Ortamı teknolojileri geliştirilebilmektedir. Apple markasının ürünü olan iOS İşletim Sistemi'nin kullanılmaması, iOS İşletim Sistemi'nin kapalı bir proje olmasından, güncel Güvenilir İşletim Sistemleri'ne uygun olmamasından ve iPhone haricinde farklı bir donanıma kurulum imkânı sağlamamasından dolayıdır.

2.2 Güvenilir İşletim Ortamı

Güvenilir İşletim Ortamı teknolojisinin özellikleri ve gereksinimleri "Global Platform" adlı güvenli dijital servisler ve cihazlar için standartları belirleyen, kâr amacı gütmeyen bir kuruluş tarafından belirlenmiştir [9]. Güvenilir İşletim Ortamı, kriptografik anahtarlar gibi hassas verilerin yalıtılmış ve güvenilen bir ortamda depolanmasını, işlenmesini ve korunmasını sağlayan, bağlı olduğu cihazın ana işlemcisinin güvenli tarafıdır. Böylelikle çalışmamızda zengin işletim sistemine karşı muhtemel saldırılarda hassas verimiz için koruma sağlanmaktadır [10]. Bir Güvenilir İşletim Ortamı teknolojisinin sağlayabildiği güvenlik özellikleri aşağıdaki gibidir [11, 12]:

- İzole Çalışabilme: Her Güvenilir İşletim Ortamı teknolojisi çalıştırabilen ana işlemcilerin iki ayrı tarafı vardır. Donanımsal olarak da birbirinden izole olarak çalışan bu taraflar terminolojide "Güvenli Dünya" ve "Normal Dünya" olarak adlandırılır. Bu özellik, uygulamanın çalışma sürecinde ilgili verilerin gizliliğini ve bütünlüğünü korur. Bu sayede, mobil kimlik uygulamamıza ait anahtarların yetkisiz erişimlere karşı korunması sağlanmış olur.

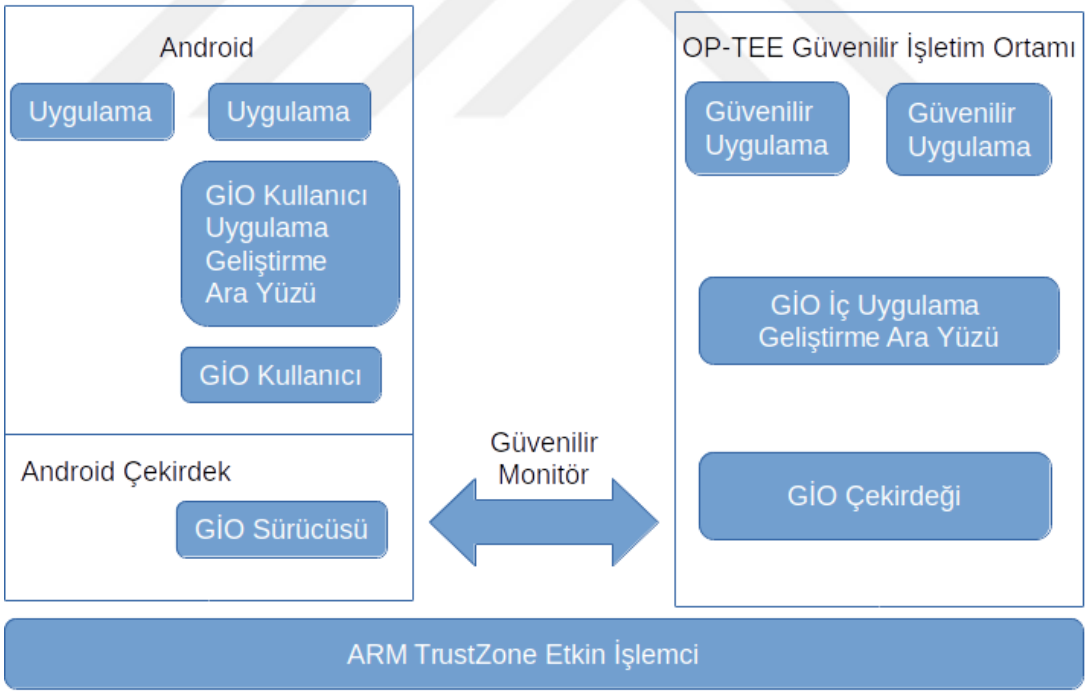
- Güvenli Depolama: Bir Güvenilir İşletim Ortamı cihaz üzerinde depolanan verilerin bütünlüğünün ve gizliliğinin sağlanmasına güvence vermelidir. Çalışmamızda, mobil kimlik için kullanılan anahtar çiftleri uygulamanın temel amacına ve kullanıcılara göre hem önemli hem de hassas bir veridir. Bu hassas verinin cihaz üzerinde depolanması Güvenilir İşletim Ortamında sağlanmaktadır.

- Uzaktan Teyit: Bir tüzel kişiliği veya onun özelliklerini üçüncü bir tarafa kefil etme yeteneğidir. Güvenilir İşletim Ortamı teknolojisi bir yazılım veya işletim sistemin güvenilir olduğu bir servise güvence vermek için de kullanılabilir.

- Güvenli Provizyon: Bir cihazda veya cihaz üzerinde çalışan belirli bir yazılım bileşenine güvenli bir şekilde veri gönderme kabiliyetidir. Bu uygulama geliştiricilere, uygulamalarını hazırlama ile ilgili güncellemeleri yapma olanağı sağlar. Bir Güvenilir İşletim Ortamı, geliştirilecek uygulamaların kullanıcıya ait cihaza yüklenmesi veya güncellenmesi durumlarında gerekli güvenlik mekanizmalarına sahip olmalıdır. Uygulamanın bütünlüğü ve kriptografik verilerin gizliliği konusunda güvence vermelidir.

- Güvenilir Yol: Bir Güvenilir İşletim Ortamı Teknolojisi, kullanıcıların cihazdaki uygulamalarla etkileşimde bulanabilecekleri güvenli bir yol sağlamalıdır. Örneğin, güvenli dünya tarafında çalışacak uygulamaya tuş takımı ile bir girdi verilecekse, uygulama ve tuş takımı arasında üçüncü parti uygulamaların ulaşamayacağı güvenli bir yol sunmalıdır.

Günümüzde aktif olarak güncellemeleri devam eden açık kaynak kodlu veya ticari birkaç Güvenilir İşletim Ortamı bulunmaktadır. Çalışmamızda, Güvenilir İşletim Ortamı Teknolojisi olarak detayı altbölüm 2.3'te verilecek olan ARM TrustZone mimarli bir altyapıya uygun OP-TEE İşletim Ortamı seçilmiştir. OP-TEE, Global Platform organizasyonunun standartlarına göre yazılmış açık kaynak kodlu bir Güvenilir İşletim Ortamı teknolojisidir. Geliştirilmesine güncellemeler ile devam edilen OP-TEE projesi farklı Linux tabanlı birçok işletim sistemi ile uyumlu çalışabildiği gibi, farklı cihazlar üzerine kolayca kurulup çalıştırılabilmektedir. OP-TEE destekli güncel ve uygun donanımlar bulunabilmesi kolaylığı ile OP-TEE çalışmamız için en uygun Güvenilir İşletim Ortamı olmaktadır [3].



Şekil 2.1 : ARM TrustZone işlemci üzerinde Android ve OP-TEE yapısı.

Şekil 2.1'de, detayları altbölüm 2.3'de anlatılacak olan ARM TrustZone mimarisine sahip bir donanım üzerinde Android ve OP-TEE işletim ortamlarının nasıl bir şekilde konumlandığını görmektesiniz. Her iki işletim ortamı işlemci üzerinde ayrı olarak

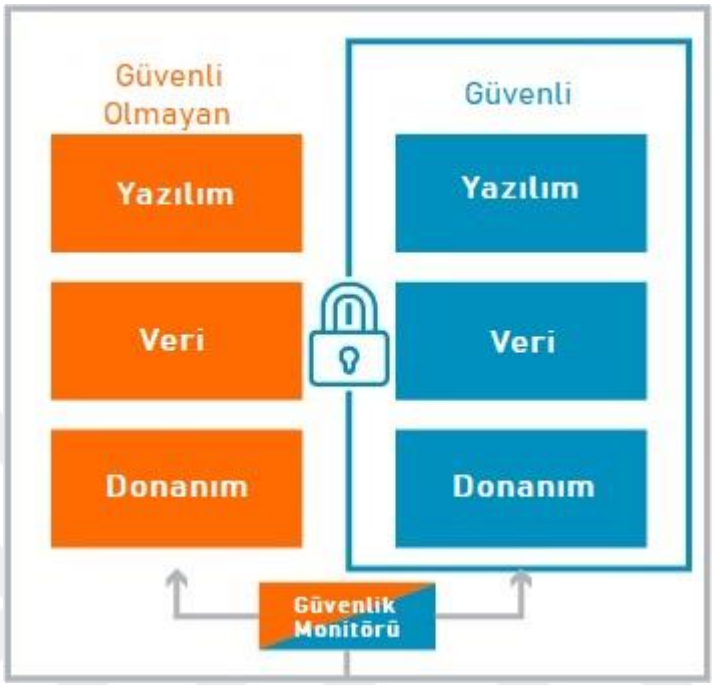
çalışmaktadır. Burada ki dikkat edilmesi gereken husus, gerekli haberleşmenin sağlanması için GİO Kullanıcı Sistemi, GIO Sürücüsü'nün Android İşletim Ortamı'nda ve her iki dünya arasında geçişi sağlayan Güvenilir Monitör'ün bulunması gerekliliğidir. OP-TEE Güvenilir İşletim Ortamı beraber çalışacağı uyumlu Zengin İşletim Ortamı'na göre bu üç sistemi bize sağlamaktadır. OP-TEE projesi Android ile uyumludur ve her iki işletim ortamının cihaz üzerinde kurulması için gerekli imajlarda tüm sistem hazırdır. Geliştiricinin, Android'in OP-TEE'yi tanıması için ayrıca bir işlem yapmasına gerek olmadığı gibi gerekli imajları uyumlu bir cihaza kurması yeterlidir.

Şekil 2.1'ye tekrar baktığımızda, bahsedilen haberleşmenin güvenilir uygulama ve Android uygulaması arasında olduğunu görmekteyiz. GİO'nun sunduğu iş ve işlemler için bir güvenilir uygulamayı kullanacak Android uygulama, GİO Kullanıcı Sistemi üzerine geliştirilmiş "GİO Kullanıcı Uygulama Geliştirme Ara Yüzü" fonksiyonlarını kullanılır. Bu fonksiyonlar kullanılarak istenilen haberleşme Android çekirdeğinde kurulu GİO Sürücüsü ile Güvenilir Monitör' e aktarılır. Böylelikle, haberleşmeye ait bilgi güvenilir dünyada GİO çekirdeğine ulaşır. GİO çekirdeği, ilgili güvenilir uygulamaya "GİO İç Uygulama Geliştirme Ara Yüzü" ile bu bilgiyi ulaştırır. Bu bilgi aslında bir komut gönderme işlemidir. Komut içerisinde girdi-çıktı verileri ve güvenilir uygulamaya ait hangi fonksiyonda işlem yaptırılacağı verileri bulunur. İlgili komut, güvenilir uygulamanın ilgili fonsiyonunda işleme sokulur. İşlemin gereksinimlerine göre "GİO İç Uygulama Geliştirme Ara Yüz" fonksiyonları kullanılır. Bu fonksiyonlar GİO çekirdeğinin uygulama geliştiriciye açtığı bir ara yüz olması vesilesiyle, çekirdekte işlemler gerçekleştirilir. Yine, güvenilir uygulamanın yaptığı iş veya işlemler sonuçlanınca istenilen çıktı verileri çekirdek üzerinden Güvenilir Monitör'e aktarılır. Burada bahsedilen güvenilir ve kullanıcı taraf uygulamalarının ait olduğu işletim ortamlarına nasıl kurulduğu ve haberleşmenin nasıl gerçekleştirildiği detayları bölüm 3'te anlatılmaktadır.

2.3 Donanımlar

Güvenilir İşletim Ortamının, Zengin İşletim Ortamı ile beraber bir donanım üzerinde çalışabilmesi için donanımın ana işlemcisinin buna uygun olması gerekmektedir. ARM ve INTEL gibi ana işlemci üreticileri bahse konu özellikte işlemcilerini geliştirmiştir. ARM firması "ARM TrustZone" adını verdiği donanım mimarisine

sahip işlemciler ürettiği gibi, INTEL firması da donanım mimarisine "Trusted Execution Technology (TXT)" adını verdiği işlemciler üretmeye başlamıştır [13, 14]. Fakat, üretilen bu yeni mimarili işlemciler arasında ARM TrustZone mimarili işlemciler daha yaygın ve uygun fiyatlı donanımlarda bulunabilmektedir.

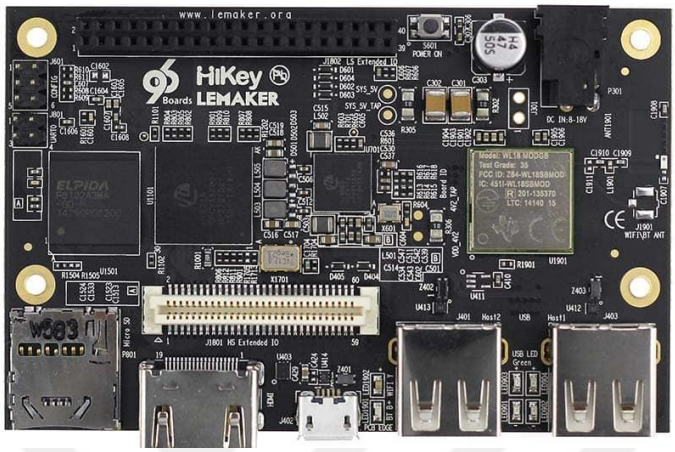


Şekil 2.2 : ARM TrustZone'un mimari yaklaşımı.

ARM firması, cihazların güvenlik sorunlarına bir nevi önlem niteliğinde ARM TrustZone teknolojisini geliştirmiştir. ARM TrustZone'nun mimari yaklaşımında donanımsal olarak birbirinden ayrılmış güvenli ve güvenli olmayan dünyalar vardır. Literatürde bu kavramlar, güvenli olmayan dünya için Zengin İşletim Ortamı veya normal dünya, güvenli dünya için Güvenilir İşletim Ortamı veya güvenilir dünya olarak anlandırılır. Mimari yaklaşımın donanım seviyesinden ayrılmasında ötürü, güvenli olmayan dünya güvenli dünyadaki kaynaklara (gizli anahtarlar, çalışmamız bünyesinde oluşturduğum mobil kimlik sertifikası) doğrudan erişemez. Bundan dolayı, üçüncü şahıs kişilerin internet üzerinden saldırıları veya korsan yazılımlarla bilgi hırsızlığının önüne geçilmiş olur. Ek olarak, çalışmamızda kullandığımız Android çekirdeği işlemleri bile OP-TEE Güvenilir İşletim Ortamı'nın güvenlik özelliklerine, kaynaklarına erişemez. Sonuç olarak, bu iki dünya arasında ki geçiş "güvenli monitör" adı verilen yazılım bileşeni aracılığıyla gerçekleştirilir [13].

Mobil telefon üreticileri, Güvenilir İşletim Ortamı teknolojilerine benzer ürünlerini son kullanıcıya sunmaya başlamışlardır. Samsung, Huawei gibi markalar, akıllı

telefonlarında kendi Güvenilir İşletim Ortam'larını geliştirmiştir. Sunulan bu hizmetler uygulama geliştiricilere kapalı, cihaz üzerinde son kullanıcıya sunduğu servislerde kısıtlı ve sahip oldukları işlemciler uygun nitelikte değildir. Bu nedenle, çalışmamızın amaçlarına uygun araştırma ve geliştirme yapabileceğimiz bir mobil telefon şu an için piyasada bulunmamaktadır.



Resim 2.1 : HiKey (LeMaker versiyonu) geliştirme kartı ön yüzü.

Araştırmalarımız sonucu elde edilen bilgiler ışığında, seçtiğimiz donanım HiKey geliştirme kartı olmuştur. LeMaker adlı versiyonunu kullandığımız HiKey geliştirme kartı 2GB'lık hafıza kapasiteli ve Trustzone mimarisine sahip ARM işlemciye sahiptir [15]. HiKey geliştirme kartı diğer geliştirme kartlarına göre Android ve OP-TEE'yi beraber daha stabil bir biçimde çalıştırabildiği görülmüştür. Ayrıca her iki işletim sisteminin HiKey geliştirme kartına destekleri bulunduğundan çalışmamızda rahatça kullanabildiğimiz en iyi donanım olmuştur.

Geliştirdiğimiz mobil kimlik uygulamasının arayüz çalışmaları için WaveShare marka 7inç dokunmatik ekran kullanılmıştır.

3. GELİŞTİRİLEN MOBİL KİMLİK UYGULAMASI

Uygulamamızın geliştirilmesine başlamadan önce, Android İşletim Sistemi ve Güvenilir İşletim Ortamı teknolojisi OP-TEE'nin HiKey geliştirme kartı üzerinde kurulumu gerçekleştirilmiştir. Burada OP-TEE geliştiricilerinin sunduğu kurulum aşamaları takip edilmiştir. Kurulumun başarıyla sonuçlandığının garantisi, kurulum içerisinde gelen ve OP-TEE geliştiricileri tarafından yazılmış, Güvenilir İşletim Ortamının tüm gerekli işlevlerini test eden "xtest" programı ile alınmıştır. Bununla beraber WaveShare marka 7inch dokunmatik ekran, geliştirme kartımıza bağlanıp Android İşletim Sisteminin de başarıyla kurulduğu görülmüştür.

Güvenilir İşletim Ortamı teknolojisi kullanarak geliştirilen mobil uygulamaların güvenilir dünyada çalışacak kısmına "Güvenilir Uygulama (Trusted App)", normal dünyada çalışacak olanına ise "Kullanıcı Uygulaması (Client App)" denmektedir. Çalışmamızda gerçekleştireceğimiz güvenilir uygulama C programlama dili kullanılarak geliştirilmiştir. Güvenilir Uygulama, güvenilir işletim ortamı fonksiyonlarını kullanarak kullanıcı tarafı uygulamadan gelecek komutları gerçekleştirecek ve sonuçları geri döndürecektir. Kullanıcı uygulamamız Java Programa Dili ile yazılmış bir Android Mobil Uygulamasıdır. Bu uygulama mobil kimlik uygulamamızın kullanıcı ara yüzlerini içermektedir. Aynı zamanda güvenilir uygulama ile güvenli yoldan haberleşmeye geçecek, Android Native kütüphanesini kullanan C programlama dili ile geliştirilmiş bir program da mobil uygulamamız içerisinde mevcuttur.

Açık anahtar altyapısı kullanan akıllı kartlarımız (kimlik kartı), bir özel anahtara ve bu özel anahtar ile gerçekleştirilecek imzalama fonksiyonuna sahiptir. Böylece, akıllı kartımızda imzalanan bir veri, akıllı kart özel anahtarına karşılık gelen açık anahtarı güvenli bir yoldan temin etmiş herhangi bir uzak sistemde başarıyla doğrulanacaktır. Akıllı kartların bu özelliği bize bir mobil kimlik oluşturulabilme fırsatı vermektedir. Bunun için ilk olarak mobil cihazımızda bir RSA anahtar çifti üretmemiz ve anahtar çiftinin bütünlüğünün ve gizliliğinin sağlanacağı bir sistemde saklanması gerekmektedir. Bir Güvenilir İşletim Ortamı içerisinde RSA anahtar çifti

üretebildiğimiz gibi, sahip olduğumuz anahtarları güvenli bir şekilde cihaz üzerinde saklayabiliriz. Uygulamamızdan örnek vermek gerekirse, üretilen RSA anahtar çiftinin açık anahtarı akıllı kartımız üzerinde imzalanarak elde edilen yeni imzalı açık anahtar tekrar güvenilir tarafa alınarak bütünlüğü ve gizliliği sağlanarak cihazda saklanabilecektir [7].

Cihazımız ve akıllı kartımız ile üretilen imzalı açık anahtar, mobil kimlik uygulamamız için ilk aşamayı oluşturmaktadır. Bu işlemden sonra artık akıllı kartımıza ihtiyacımız yoktur. İmzalı açık anahtarımız, bir uzak sisteme gönderildiğinde, sistemde başarılı olarak kimliğimiz doğrulanabilecektir. Doğrulama işlemi sonucunda, uzak sistem Güvenilir İşletim Ortamında oluşturduğumuz RSA anahtar çiftinin açık anahtarını elde edecek ve sistem "Artık bu akıllı kart kullanıcısından gelecek doğrulama taleplerini bu RSA açık anahtarı ile yap." mesajını alacaktır. Böylelikle, cihazımızın güvenilir tarafında oluşturduğumuz ve güvenle sakladığımız özel anahtarımız ile karşı sistemin yeni sahip olduğu açık anahtarımızla birlikte karşılıklı bir imzalama-doğrulama sürecinin işletilebileceği bir mobil kimliğe sahip olmuş olacağız.

3.1 Genel Bakış

Mobil Kimlik uygulamamız, dört ana bileşenden oluşmaktadır. Bunlar çalışmamızda mobil cihaz olarak kullandığımız HiKey geliştirme ortamına kurulu olan Android İşletim Sistemi ve Güvenilir İşletim Sistemi OP-TEE, akıllı kimlik kartı üzerinden sayısal imzalama ile mobil kimlik oluşturduğumuz masaüstü uygulaması ve mobil kimlik ile uzak sunucu üzerinde kimlik doğrulamasını örneklendirdiğimiz uzak sunucudur. Android İşletim Sistemi üzerinde geliştirilen Android uygulama, Güvenilir İşletim Ortamı teknolojisi yaklaşımına göre "Kullanıcı Tarafı Uygulama" adını almaktadır. Bu uygulama güvenilir dünya ile haberleşmeyi sağlayan ve bir nevi normal dünyaya açılan kapı niteliğinde native uygulama içermektedir. Ayrıca, son kullanıcının mobil kimlik sürecini kontrol edeceği ekranlar ve ekranlara ait kontrol mekanizmalarını da içerir. Güvenilir İşletim Ortamı terminolojisinde "Güvenilir Uygulama" adı verilen güvenilir dünyaya ait uygulama ise, Mobil Kimlik uygulamamız için gerekli sayısal imzalama, RSA anahtar çiftinin oluşturulması ve güvenilir depoda saklanması, açık anahtar üzerinden oluşturulacak mobil kimlik verilerinin güvenilir depoda saklanabilmesi sağlamak gibi görevlere sahiptir.

Mobil Kimlik Uygulamamız temelde, akıllı kimlik kartlarımızı mobil cihazlarımıza taşıyabilmemizi amaçlamaktadır. Ayrıca, Güvenilir İşletim Ortamı'nın sunduğu güvenlik mekanizmaları ve fonksiyonları ile akıllı kimlik kartının, kart okuyucu vasıtasıyla her seferinde okutulmasının önüne geçerek son kullanıcı için kullanılabilirliğin ve gizliliğin artırılabilmesi hedeflenmiştir. Fakat, mobil cihaz üzerinde üretilen RSA anahtar çiftinin açık anahtarı, oluşturulacak mobil kimlik için bir kereye mahsus standart kart okuyucusu yardımıyla akıllı kimlik kartımız üzerinde imzalanmalıdır. Bunun için çalışmamızda, bu imzalama işlemini gerçekleştirecek bir masaüstü uygulama geliştirilmiştir. Java programlama dili Swing kütüphanesi ile geliştirilen bu uygulama JVM üzerinde çalışacağından, üzerinde çalışacak olan bilgisayarın işletim sisteminden bağımsız olarak çalışacaktır. Masaüstü uygulamamız, bilgisayara bağlı olduğu gördüğü ve tanıdığı mobil cihaz üzerinden açık anahtarı çekecek ve yine bilgisayara bağlı olduğunu gördüğü standart kart okuyucusuna takılı akıllı kimlik kart üzerinde imzalama işlemini gerçekleştirecektir.

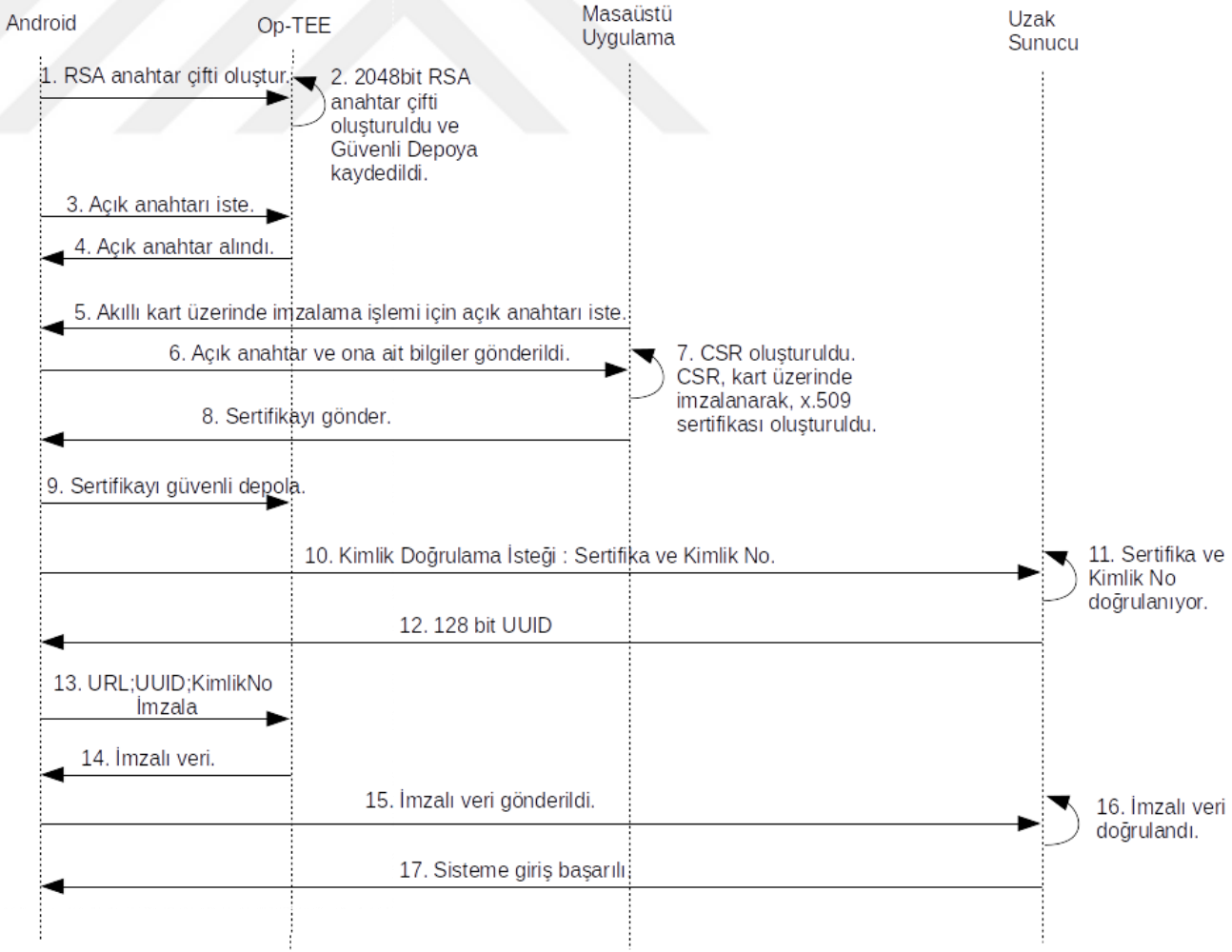
Son olarak, çalışmamız için bir uzak sunucu geliştirilmiştir. Uzak sunucu, mobil kimliğimiz ile kimlik doğrulama yapacağımız örnek bir web servisidir. Detayları altbölüm 3.5'de anlatılacak olan uzak sunucuda, nasıl bir kimlik doğrulama yapılmasının gerekliliği konusunda bilgilere verilerek örneklendirmiştir. Bu aşamada, son kullanıcı gizliliğini ağ üzerinde artıracak bir aşama olan "Cihaz Doğrulama"ya yer verilmiştir. Tüm bu özet bilgiler ışığında Mobil Kimlik Uygulamamıza ait genel iş akışı Şekil 3.1'de gösterilmiştir. Şekle göre adımlar aşağıdaki gibidir:

1-) Mobil Kimlik Uygulamamızı ilk olarak açan son kullanıcı mobil cihaza ait herhangi bir RSA anahtar bilgisine sahip olmadığını görerek RSA anahtar üretme işlemine geçecektir. Android uygulamamız içerisinde bulunan native programa ait fonksiyon çağırılacaktır. Native program son kullanıcıdan aldığı veriler ile OP-TEE içerisinde bulunan güvenilir uygulamaya "RSA anahtar çifti oluştur ve bunu güvenilir depoda sakla" komutunu gönderecektir.

2-) Alınan komut, güvenilir uygulamada ilgili fonksiyona düşer ve fonksiyon 2048 bit RSA anahtarı oluşturur. Oluşturulan RSA anahtar çifti Güvenilir İşletim Ortamı'nın standartlarca belirlenen ve olması gerekli güvenilir depoya kaydedilerek saklanır.

3-) RSA anahtar çifti oluşturulması ile, Android uygulamamız akıllı kart üzerinde imzalama işlemi için RSA anahtar çiftinin açık anahtarını güvenilir uygulamadan ister.

Şekil 3.1 : Geliştirilen mobil kimlik uygulaması iş akışı.



4-) İstenen açık anahtar Android uygulamamız içerisinde veritabanında saklanır. Ayrıca, bu aşamada son kullanıcıya cihaza ait bir RSA anahtar çiftini olduğu, hangi kimlik numarasına ait olduğu, oluşturulma tarihi ve geçerlilik süresi gibi bilgiler gösterilir.

5-) Bu aşamada, son kullanıcı masaüstü uygulamamızın kurulu olduğu bilgisayarına akıllı kimlik kartını standart kart okuyucusuyla ve mobil cihazını USB kablo vasıtasıyla bağlar. Masaüstü uygulamasının açılmasıyla beraber son kullanıcı, akıllı kimlik kartına ait bilgileri ve bilgisayara bağlı olan mobil cihazları görür. İlgili mobil cihazın seçilmesi ile "RSA açık anahtarını getir" komutunu verir.

6-) USB kablo aracılığı ile masaüstü uygulama ve Android uygulamamız haberleşmeye başlar. Android uygulama, cihazda bulunan açık anahtarı ve ona ait bilgileri masaüstü uygulamasına gönderir.

7-) Bu aşamada, açık anahtar ile beraber gelen bilgiler kontrol edilir. Örneğin, gelen bilgiler içerisinde kimlik numarası da bulunmaktadır. "Kimlik numarası, akıllı kimlik kartındaki kimlik numarası ile eşleşiyor mu?" gibi kontroller yapılır. Her bilginin kurala göre uygun olduğu görüldükten sonra mobil cihazdan gelen açık anahtar ve bilgiler ile sertifika imzalama isteği (Certificate Signing Request) oluşturulur. Oluşturulan CSR, akıllı kart üzerinde imzalanarak mobil kimliğimizi oluşturacak "X.509" sertifikası elde edilir.

8-) Mobil kimliği oluşturacak sertifikanın üretilmesi sonucunda, USB kablo aracılığıyla bu sertifika Android uygulamaya gönderilir.

9-) Mobil kimliğimiz artık akıllı kimliğimizin bağlı bulunduğu sistemlere giriş için bir anahtar görevi göreceğinden bunun güvende saklanması gerekecektir. Burada devreye yine Güvenilir İşletim Ortamı'nın güvenilir depolama özelliği devreye girer. Android uygulamamızın içinde bulunan native programın ilgili fonksiyonu çağrılarak sertifika güvenilir uygulamaya gönderilir. Bunun sonucunda, OP-TEE'de bulunan güvenilir uygulamamız bu sertifikayı güvenilir depoda saklayacaktır.

10-) Mobil kimliğimiz artık bağlı olduğu sistemlerde kimlik doğrulama işlemlerine hazırdır. Çalışmamız için oluşturduğumuz örnek web servisi sunucusuna, Android uygulamamızda geliştirdiğimiz giriş ekranı kullanılarak bir kimlik doğrulama isteği gönderilir. Bu aşamada ekran vasıtasıyla girilen kimlik numarasının mobil kimliğe ait olup olmadığı da kontrol edilir.

11-) Gelen istek ile uzak sunucuda, kimlik bilgilerinin kontrolüne takiben mobil kimliğimizi oluşturan sertifika doğrulanır. Örneğimizde akıllı kimlik kartımızın bağlı olduğu bir uzak sunucu modellendiğinden kimlik karta ait açık anahtar bilgisinin önceden uzak sunucu ile paylaşıldığını varsaymaktayız. Sonuç olarak, uzak sunucunun sahip olduğu kimlik kartına ait açık anahtar ile mobil kimliğimiz doğrulanır.

12-) Doğrulama işleminin başarılı olması durumunda uzak sunucuda evrensel olarak benzersiz tanımlayıcı bir numara (UUID) oluşturulur ve bu Android uygulamanın kimlik doğrulama isteğine cevap olarak gönderilir.

13-) Android uygulama gelen UUID ile "URL;UUID;KimlikNo" deseninde bir karakter dizisi oluşturur. Native program aracılığıyla bu karakter dizisini Güvenilir İşletim Ortamı'nda üretilen RSA anahtar çiftinin gizli anahtarı ile imzalamaya gönderilir.

14-) OP-TEE ortamında imzalanan veri, uzak sunucuya gönderilmek üzere güvenilir uygulamadan istenilir.

15-) İmzalı veri uzak sunucuya doğrulanmak üzere gönderilir.

16-) Uzak sunucu, 11nci adımda elde edilen mobil kimliğimiz ile 15nci adımda Mobil Kimlik Uygulamamızdan gelen imzalı veriyi doğrulama işlemini gerçekleştirir. Bu doğrulama işlemi ve adım 12, 13, 14 ve 15'i içeren işlemler, detayları altbölüm 3.5'te anlatılacak olan ortadaki adam saldırılarını engelemek içindir. Cihaza ait imzalı verinin uzak sunucuda doğrulanmasını içeren bu adıma çalışmamızda "Cihaz İmza Doğrulaması" adı verilmiştir.

17-) Başarılı bir doğrulama sonucuda mobil uygulamaya, sistemde başarılı bir oturum açtığı bilgisi gönderilir.

3.2 Güvenilir Uygulama

Güvenilir uygulama, OP-TEE çekirdeği üzerinde Android İşletim Sistemi'nden izole olarak çalışmaktadır. Bu uygulamada Android tarafından gelecek girdiler, Güvenli Dünya'da işlem görür ve çıktılar tekrar Zengin İşletim Ortamı'na gönderilir. Mobil Kimlik fikrimizde kullandığımız RSA anahtar çifti oluşturma, ilgili RSA anahtar çiftlerinin ve oluşturulan imzalı açık anahtarın güvenilir bir şekilde mobil cihazımızda

saklanması işlemi “Güvenilir İşletim Ortamı İç Uygulama Geliştirme Ara Yüzü” fonksiyonları kullanılarak güvenilir tarafta gerçekleştirilmiştir.

Çalışmamızda, OP-TEE Güvenilir İşletim Ortamı’nın Android İşletim Sistemi’nin “Oreo” versiyonu ile beraber derlenerek oluşturulan imajlar, HiKey geliştirme kartına yazılarak test ortamımız oluşturulur [16]. Bundan dolayı, Android çalışan bir sisteme OP-TEE’i ekleyemezsiniz. OP-TEE’nin Android İşletim Sistemi ile kurulduğu bir sistemde, güvenilir uygulamalar “/system/lib/optee_armtz” dizini altında bulunur. Çalışmamızın güvenilir uygulamasının sisteme eklenmesi, Android’in OP-TEE ile derlenme aşamasında gerçekleştirilmiştir. Cihaz üzerinde çalışan bir sisteme, güvenilir uygulamanın ayrıca derlenip çalıştırılabilir versiyonunun gerekli dizine eklenmesi, kullanıcı taraf uygulamanın güvenilir uygulamaya ulaşamaması gibi sıkıntılar yaşatmaktadır. Bunun için aşağıdaki iki aşamanın gerçekleştirilmesi sonucunda derlediğimiz Android ve OP-TEE sisteminde güvenilir ve kullanıcı taraf uygulamanın daha stabil çalıştığı görülmüştür.

1-) Güvenilir uygulamanın kaynak kodlarının eklenmesi derlenecek dosyalar arasına eklenmesi: Kaynak [16]’da, derleme ortamını oluşturan Android ve OP-TEE’ye ait kaynak kod depolarının yanında her iki sistemin HiKey geliştirme kartı üzerinde çalıştırılabilmesi için gerekli tüm kaynak kodları görebilirsiniz. Tüm bu kaynak kodlar, onları derleyecek komut dosyaları (Make dosyaları) ile belirli dizinler şeklinde bilgisayara indirilir ve derleme başlatılır. Bu belirli dizinler içinde güvenilir uygulamamızın kaynak kodları ve derleme komutları “~/optee_android_manifest/external/optee_examples/” dizininde bulunan güvenilir uygulamalar yanına eklenir. Böylece, OP-TEE Güvenilir İşletim Ortamı’nın kaynak kodları derlenirken, güvenilir uygulamamızda derlenecektir.

2-) “optee_packages.mk” derleme dosyasına güvenilir uygulamamızın evrensel olarak benzersiz tanımlayıcı numarasının (UUID) eklenmesi: Bir güvenilir uygulama, yüklendiği sistemdeki diğer güvenilir uygulamalardan evrensel olarak eşsiz tanımlayıcı bir sayı ile ayrılır. Kullanıcı tarafı uygulama 128 bitlik bu sayı ile bağlı olduğu güvenilir uygulama üzerinde oturum açar. Detayları altbölüm 3.3’de verilen UUID, birinci madde sonunda elde edilen çalıştırılabilir dosyanın adı olarak verilir. Çalışmamızda bu güvenilir uygulamamızın sistemdeki adı “c4d22850-20f1-4b12-823aeb7aefba62.ta” dır. Bu çalıştırılabilir dosyanın Android ve OP-TEE’nin beraberce derlenmesi sonucunda elde edilen imaj dosyalarına eklenmesi gerekmektedir. Bunun

için, derleme komutları dosyaları arasında ve "~/optee_android_manifest/device/linaro/hikey" dizini altında bulunan "optee_packages.mk" dosyasına güvenilir uygulamamız aşağıdaki şekilde eklenir:

PRODUCT_PACKAGES += c4d22850-20f1-4b12-823aeb7aefbae462.ta

Yukarıdaki komutun amacı, Android ve OP-TEE sistemlerinin derlenmesi sonucunda çıkan gerekli imaj içerisine "c4d22850-20f1-4b12-823aeb7aefbae462.ta" adlı güvenilir uygulamayı eklemektir. Eğer bu komut eklenmez ise, ilgili güvenilir uygulama derlenecek fakat, gerekli imaj dosyasına eklenmeyecektir.

3.2.1 Güvenilir uygulama ara yüz fonksiyonları

Her güvenilir uygulama, "Güvenilir Uygulama Ara Yüzü" adı verilen bir dizi fonksiyon sunmalıdır [17]. Bu fonksiyonlar, güvenilir uygulamanın örneğini oluşturmak, kullanıcının bağlı olduğu örneği bildirmek, kullanıcı bir komut çağırdığında örneği bildirmek vb. için Güvenilir İşletim Ortamı tarafından çağrılan giriş noktalarıdır. Bu giriş noktaları, geliştirici tarafından fonksiyon olarak geliştirilerek güvenilir uygulamaya eklenir. Bu güvenilir uygulama ara yüzü fonksiyonları sayesinde Güvenilir İşletim Ortamı'na bağlı olacaktır. Bunlar:

-*TA_CreateEntryPoint*: Güvenilir uygulamanın yapılandırıcı fonksiyonudur. Fonksiyon başarısız olursa, güvenilir uygulamanın bir örneği oluşturulamaz ve güvenilir uygulama durur. Oluşturulan örneğin yaşam döngüsü boyunca sadece bir kez çağılır.

-*TA_DestroyEntryPoint*: Güvenilir uygulamanın imha edici fonksiyonudur. Güvenilir İşletim Ortamı, bu fonksiyonu güvenilir uygulama örneği sonlandırılmadan önce çağırır. Böylelikle, Güvenilir İşletim Ortamı bu fonksiyonu çağırdığında hiçbir oturumun açık olmadığını garanti eder. *TA_DestroyEntryPoint* fonksiyonu çalışma süresi bitiminde, Güvenilir İşletim Ortamı güvenilir uygulama örneğinin kullandığı tüm kaynakları geri toplar.

-*TA_OpenSessionEntryPoint*: Bir kullanıcı tarafı uygulama yeni bir oturum açmak için güvenilir uygulamaya bağlanmaya çalıştığı zaman bu fonksiyon çağırılır. Açılan oturum sonraki tüm güvenilir uygulama çağrılarında kullanılır.

-*TA_CloseSessionEntryPoint*: Bu fonksiyon, kullanıcı tarafı uygulama oturumunu kapattığında ve güvenilir uygulama ile olan bağlantısını kestiğinde çağrılır. Böylelikle,

Güvenilir İşletim Ortamı, oturumun kapatılmasında güvenilir uygulama aktif bir komutun olmadığını garanti eder.

-*TA_InvokeCommandEntryPoint*: Bir kullanıcı tarafı uygulamanın ilişkili olduğu güvenilir uygulamaya çağrılarını komutlar vasıtasıyla yollamaktadır. Yani kullanıcı tarafı uygulama, güvenilen uygulama komutunu çağırdığında bu fonksiyon çağrılarak işlem görür.

Bir güvenilir uygulamanın yaşam döngüsünü kontrol eden bu "Güvenilir Uygulama Ara Yüzü" fonksiyonları, çalışmamızdaki güvenilir uygulamaya da eklenmiştir. Fakat, gereksinimlerimiz için herhangi bir ihtiyaç olmadığından, *TA_InvokeCommandEntryPoint* fonksiyonu haricindeki fonksiyonlar içinde bir iş veya işlemi gerçekleştirmek için geliştirme yapılmamıştır. *TA_InvokeCommandEntryPoint* fonksiyonu içinde gerçekleştirilen işlemler altbölüm 3.2.3'de anlatılmıştır.

3.2.2 Güvenilir uygulamada geliştirilen fonksiyonlar

Gereksinimlerimize göre, güvenilir uygulamada kullandığımız fonksiyonlar aşağıdaki gibidir:

1-) generate_and_save_rsa_key (): Bu fonksiyon ile mobil cihazımızda bir RSA anahtar çifti oluşturup, anahtarı güvenilir işletim ortamının güvenilir depolama özelliği kullanarak saklayabiliriz. Oluşturulan RSA çifti mobil kimlik uygulamamızda 2048 bit'dir. Bu fonksiyonda yapılmak istenen iş "Güvenilir İşletim Ortamı İç Uygulama Geliştirme Ara Yüzü" nün aşağıdaki fonksiyonlar ile geliştirilmiştir. Bunlar:

- *TEE_AllocateTransientObject (TEE_TYPE_RSA_KEYPAIR,*
  RSA_KEY_SIZE,
  &transientKey):

  İlk olarak, *TEE_AllocateTransientObject* adlı fonksiyon ile oluşturulmak istenen RSA anahtar çifti için geçici nesne tipi Güvenilir İşletim Ortamı'nda oluşturulur. Burada ilk parametre *TEE_TYPE_RSA_KEYPAIR,* "Güvenilir İşletim Ortamı İç Uygulama Geliştirme Ara Yüzü" nün güvenilir uygulama geliştiricisine sunduğu bir C programlama dili tanımlayıcı makrosudur. Böylece ilgili fonksiyona oluşturulacak geçici nesnenin bir RSA anahtar çifti tutacağı bilgisi verilir. İkinci parametrede ise, oluşturulacak nesnenin boyutu

verilmektedir. Bu boyut muhtemel nesne tiplerine göre değişiklik gösterdiği gibi [18], uygun olmayan boyutlarda ilgili fonksiyon hata gösterecektir. Çalışmamızda kullandığımız RSA anahtar çiftinin boyutu 2048 bit olduğu için *RSA_KEY_SIZE* makrosu 2048 sayısını göstermektedir. Son parametre ise bir çıktı değişkenidir. Güvenilir İşletim Ortamı'nda nesneler *TEE_ObjectHandle* türünde nesne tutucu değişkenlerde tutulur. İlgili fonksiyon, tutacağımız nesne tipi için sistemde gerekli alanı ayırdıktan sonra *TEE_ObjectHandle* göstericisinde bu alanın adresini bize verir.

- *TEE_GenerateKey (transientKey,*
  RSA_KEY_SIZE,
  NULL, 0):

  TEE_AllocateTransientObject fonksiyonu ile oluşturduğumuz geçici alan, bir RSA anahtar çiftini tutmak istediğimiz boş bir alandır. Çalışmamızda kullandığımız RSA anahtar çifti *TEE_GenerateKey* fonksiyonu ile üretilmektedir. Bu fonksiyonun ilk parametresi oluşturduğumuz geçici nesnenin adresidir. İlgili fonksiyon bu geçici nesnenin tipine ve ikinci parametrede verilen anahtar boyutuna göre bu nesnenin içini doldurur. Üçüncü parametre anahtar oluşturmada kullanacak nitelikler ve son parametre ise bu parametrenin boyutudur. Örneğimizde herhangi bir nitelik kullanılmamız gerekmemektedir.

- *TEE_CreatePersistentObject (TEE_STORAGE_PRIVATE,*
  &keyId, sizeof(keyId);
  flags, transientKey,
  NULL, 0, &persistentKey):

  Yeni RSA anahtar çiftini tutan geçici nesneyi, çalışmamızın bu aşamasından sonraki işlemlerinde kullanılmak üzere Güvenilir İşletim Ortamı'na kaydedilmesi gerekmektedir. Böylelikle oluşturduğumuz RSA anahtarı Güvenilir İşletim Ortamı'nın güvenilir depolama özelliği ile Normal Dünya'dan gelebilecek tehlikelere karşı korunacaktır. *TEE_STORAGE_PRIVATE,* RSA anahtaramızın kaydedileceği güvenilir depolama alanının numarasıdır. "keyId" adı verilen değişken, RSA anahtarımızın güvenilir depolama alanındaki benzersiz numarasıdır. Bu numara örneğimizde 100'dür. "flags" adlı parametresinde ise kaydedilecek

RSA anahtar çiftimizin erişim hakları ve paylaşım izinleri verilir. Mobil Kimlik Uygulamamız, son kullanıcıya güvenilir depoda tutulan RSA anahtar çifti veya save_certificate () fonksiyonunda da görüldüğü gibi mobil kimliği üzerinde istediğini yapma esnekliği sunmaktadır. Yani son kullanıcı güvenilir depoda tutulan RSA anahtar çifti veya mobil kimliğini görüntüleyebilir, güncelleyebilir veya silebilir. Bundan dolayı, güvenilir depoda tutulan verilerin erişim ve paylaşım izinleri aşağıdaki gibi verilmiştir.

int32_t flags = TEE_DATA_FLAG_ACCESS_READ |
TEE_DATA_FLAG_ACCESS_WRITE |
TEE_DATA_FLAG_ACCESS_WRITE_META |
TEE_DATA_FLAG_SHARE_READ |
TEE_DATA_FLAG_SHARE_WRITE;

Son olarak, "transientKey" ile adlandırılmış geçici nesnemiz, güvenilir depolama alanına kaydedilerek, "persistentKey" adını verdiğimiz *TEE_ObjectHandle* gösterici türünde geri döner.

2-) get_public_key_exponent_modulus (): İlgili fonksiyon, güvenilir tarafta oluşturulan RSA anahtar çiftinin açık anahtarını Android tarafına gönderir. Bu fonksiyonumuz ile "generate_and_save_rsa_key" fonksiyonunda oluşturup kaydettiğimiz RSA anahtar çiftimizin açık anahtarının üs (exponent) ve modül (modulus) değerleri okunarak, çalışmamızın devamında ki iş akışının sağlanabilmesi için kullanıcı tarafı uygulamaya gönderir. Bu işlemler şu şekilde gerçekleştirilmiştir:

- *TEE_OpenPersistentObject (TEE_STORAGE_PRIVATE,*
&rsa_keypair_id,
sizeof(rsa_keypair_id),
flags, &rsa_keypair):

  TEE_OpenPersistentObject fonksiyonu ile güvenilir depolama alanına kaydettiğimiz RSA açık anahtarı, *TEE_STORAGE_PRIVATE* numaralı depolama alanından, "rsa_keypair_id" değişkeni ile verdiğimiz 100 benzersiz nesne numarası ile çekilip "rsa_keypair" adı verilen *TEE_ObjectHandle* nesne tutucusunda bize RSA anahtarımız geri döner.

- *TEE_GetObjectBufferAttribute (rsa_keypair,*
TEE_ATTR_RSA_PUBLIC_EXPONENT, buffer, &buffer_len):

Artık, “rsa_keypair” adlı nesne tutucusu değişken ile RSA anahtarımız elimizdedir. *TEE_GetObjectBufferAttribute* fonksiyonunun ikinci parametresine girilen ve Güvenilir İşletim Sistemi’nin sunduğu *TEE_ATTR_RSA_PUBLIC_EXPONENT* ile *TEE_ATTR_RSA_MODULUS* nitelik numaraları ile anahtarımızın bulunduğu “rsa_keypair” adlı nesne tutucusundan açık anahtarımıza ait üs (exponent) ve modül (modulus) verilerini çekmekteyiz. Elde edilen bu iki veri, açık anahtar oluşturulmak üzere belirli bir kalıp içinde kullanıcı tarafı uygulamaya gönderilir.

3-) save_certificate (): Güvenilir İşletim Ortamı’nda oluşturduğumuz ve sakladığımız RSA anahtar çiftimizin açık anahtarı, akıllı kartımız üzerinde imzalanarak, mobil kimlik görevini üstlenecek yeni bir sertifika oluşturulur. Oluşturulan sertifika kimliğimiz yerine geçeceğinden, üçüncü taraf kişilerden korunması gerektiği gibi güvenilir bir şekilde cihaz üzerinde saklanması gerekmektedir. Güvenilir İşletim Ortamı’nın sunduğu güvenlik önlemleri ile bu sertifika güvenilir depolama alanında tutulur. Kullanılan fonksiyonlar şu şekildedir:

- *TEE_CreatePersistentObject (TEE_STORAGE_PRIVATE,*
  &cert_id, sizeof(cert_id),
  flags, TEE_HANDLE_NULL,
  NULL, 0, &persistentKey):

  Yeni oluşturulan sertifikamız için Güvenilir İşletim Ortamı’nın *TEE_STORAGE_PRIVATE* numaralı güvenilir deposunda, *TEE_CreatePersistentObject* fonksiyonu ile yeni bir alan oluşturulur. Bu alanın güvenilir depodaki numarası “cert_id” değişken ile tanımlanmış olup, örneğimiz için bu numara 101 olarak tanımlanmıştır. Bu fonksiyonun başarılı olması ile birlikte, bu alanın adresi *TEE_ObjectHandle* gösterici türünden “persistentKey” adını verdiğimiz değişkene fonksiyon tarafından yazılır.

- *TEE_WriteObjectData (persistentKey,*
  buffer, buffer_len):

  “persistentKey” değişkeninde adresini tuttuğumuz güvenilir depo alanına, *TEE_WriteObjectData* fonksiyonu ile zengin işletim sisteminden gelen sertifika nesnesini yazmaktayız. Fonksiyona verilen “buffer” adlı değişken

güvenilir depolama alanına yazılacak veridir. Örneğimizde bu değişken masaüstü uygulaması tarafından üretilen sertifikadır.

4-) get_certificate (): Akıllı kartımızın bağlı olduğu uzak sistemde tekrardan bir kimlik doğrulama gerektiği durumlarda kullanıcı tarafının, güvenilir tarafta depolanmış imzalı açık anahtarı çekmesine yarar.

- *TEE_OpenPersistentObject (TEE_STORAGE_PRIVATE,*
  &cert_id, sizeof(cert_id),
  flags, &cert_handle):

  TEE_STORAGE_PRIVATE depo numaralı alandan, "cert_id" değişkeni ile 101 numara verilen sertifika verisini çekmek için ilk olarak *TEE_OpenPersistentObject* fonksiyonu kullanılır. Sertifika verisinin adresi, bize "cert_handle" adını verdiğimiz *TEE_ObjectHandle* gösterici türünden değişkene yazılır.

- *TEE_GetObjectInfo1 (cert_handle,*
  &object_info):

  TEE_GetObjectInfo1 fonksiyonu ile sertifika verimize ait bilgileri çekmekteyiz. *TEE_ObjectInfo* türünden olan "object_info" adlı değişkene "cert_handle" değişkeninin boyutu, maksimum boyutu, nesnetipi gibi verileri elde edilir. Bu fonksiyon ile verinin boyutu kontrol edilerek bir sonraki fonksiyon ile "cert_handle" değişkeninden asıl sertifika verisi çekilir.

- *TEE_ReadObjectData (cert_handle, buffer,*
  object_info.dataSize, &read_bytes):

  TEE_ObjectHandle bir nesne tutucusudur. Veri ve veri ait bilgileri içeren bir değişken tipidir. Zengin işletim ortamına cevap olarak döndürülecek veri sadece sertifika verisi olmalıdır. Bu fonksiyonda, girdi olarak verilen "cert_handle" adlı nesne tutucusundan, bir önceki fonksiyon ile elde ettiğimiz veri boyutu kadar veriyi çeker. Sonuç "buffer" adlı byte dizisine yazılır. Okunan byte boyutu veriside ayrıca çıktı değişkeni olarak fonksiyon tarafından verilir. Elde edilen "buffer" boyutu, verinin boyutu kadar olup olmadığı böylelikle kontrol edebilmekteyiz.

5-) sign_message (): Mobil cihaz üzerinde Güvenilir İşletim Ortamı'nda RSA anahtar çifti oluşturmuştuk. Bu fonksiyon, oluşturulan bu RSA anahtar çiftinin gizli anahtarı

ile bir imzalama işlemi gerçekleştirilmektedir. Bu imzalama işleminde kullanılan Güvenilir İşletim Ortamı'nın İç Uygulama Arayüzü fonksiyonları aşağıdaki gibidir:

- *TEE_AllocateOperation (&operation,*
  TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256,
  TEE_MODE_SIGN, RSA_KEY_SIZE):

  TEE_AllocateOperation fonksiyonu, verilen mod ve algoritma ile yeni bir kriptografik işlem için *TEE_OperationHandle* türündeki C programlama dili yapısını oluşturur. Operasyon nesne tutucusunun yapabileceği işlemler için uygun maksimum anahtar boyutu da bu fonksiyonun son parametresine girilir. Çalışmamızda, kullandığımız RSA anahtarının boyutunun 2048 bit ve bu boyutu *RSA_KEY_SIZE* tanımlayıcı makrosu ile güvenilir uygulamamıza eklediğimizi söylemiştik. Buna göre, *TEE_AllocateOperation* fonksiyonuna girilecek uygun maksimum anahtar boyutunun RSA anahtarımızın boyutu kadar olması yeterlidir. Güvenilir İşletim Ortamı İç Uygulama Geliştirme Ara Yüzü geliştiricilere yapılacak şifreleme işlemleri için modlar ve algoritmalar sunar. *TEE_AllocateOperation* fonksiyonu ile oluşturulacak operasyon nesnesi imzalama işleminde kullanılacağından, operasyon modu *TEE_MODE_SIGN* olmalıdır. İkinci olarak, imzalama işleminde kullanılacak algoritma olarak *TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256* makrosu ile tanımlanan algoritma seçilmiştir. Bu algoritma seçmemizin nedeni çalışmamızda kullanılan akıllı kimlik kartının da imzalama işlemini bu algoritmada yapmasıdır. İlgili algoritmada, imzalanacak verinin SHA256 algoritmasıyla alınan özeti (digest) ile imzalama işlemi yapılır.

- *TEE_OpenPersistentObject (TEE_STORAGE_PRIVATE,*
  &keyId, sizeof(keyId),
  flags, &key_handle):

  Diğer fonksiyonlarımızda belirtildiği gibi güvenilir depo da depolanan verilerimizi tekrar elde etmek için kullandığımız fonksiyondur. Bu fonksiyon ile cihaz içinde ürettiğimiz RSA anahtarımızı *TEE_ObjectHandle* türünden "key_handle" adlı değişkende tutmaktayız.

- *TEE_SetOperationKey (operation, key_handle)*:
  Yeni bir kriptografik işlem için *TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256* fonksiyonu ile oluşturulan

"operation" adlı operasyon tutucusuna, imzalama işlemini yapacak anahtar verilir.

- *TEE_AsymmetricSignDigest (operation,*
  *(TEE_Attribute *) NULL, 0,*
  message, message_len,
  signed_message, &signed_message_len):

  İmzalama işlemini gerçekleştiren fonksiyondur. "operation" adlı değişkende belirlenen algoritma, mod ve anahtar ile gerekli işlemleri yapar. İlk maddede belirtildiği gibi SHA256 algoritması ile özeti alınan mesaj ("message" adlı değişken), bu fonksiyona girdi olarak verilir. "signed_message" çıktı değişkeni ile imzalanmış veri alınarak, Zengin İşletim Ortamı'na iletilir.

6-) digest_message (): Açık bir mesajın SHA256 algoritması ile özetini almaktadır. Alınan özet, imzalama işlemi için Zengin İşletim Ortamı'ndan "sign_message" adlı fonksiyona gönderilecektir.

- *TEE_AllocateOperation (&operation,*
  TEE_ALG_SHA256,
  TEE_MODE_DIGEST, 0):

  Bir kriptografik işlem gerçekleştirileceği için *TEE_AllocateOperation* ile bir operasyon tutucusu oluşturulur. *TEE_OperationHandle* türünden oluşturulan "operation" adlı değişken için mod ve algoritma olarak *TEE_MODE_DIGEST, TEE_ALG_SHA256* ayarlanır. Özet alma işlemlerinde bir anahtar kullanılmadığı için maksimum anahtar boyutu "0" olarak girilir.

- *TEE_DigestDoFinal (operation,*
  message, message_len,
  digest, &digest_len):

  Bu fonksiyonda verilen "operation" değişkeni ile bir mesajın özeti alınır. Özet bilgisi "digest" değişkeninde tutularak bu bilgi kullanıcı tarafı uygulamaya gönderilir.

7-) delete_rsa_key (): Mobil Kimlik Uygulamamıza ait RSA anahtar çiftine bir geçerlilik süresi verilmiştir. Bu geçerlilik süresi bitiminde veya cihazımızda oluşturduğumuz anahtar çiftinin yenilenmesi sürecinde anahtarların silinmesi gerekmektedir.

- *TEE_OpenPersistentObject (TEE_STORAGE_PRIVATE,*
  &keyId, sizeof(keyId),
  flags, &key_handle):

  Cihaz üzerinde oluşturulan RSA anahtar çiftinin silinme işlemi için güvenilir depodan çekilerek bir nesne tutucusu oluşturulur.

- *TEE_CloseAndDeletePersistentObject (key_handle)*:

  Güvenilir İşletim Ortamı'ndan RSA anahtarımızı silme işlemi gerçekleştirilir.

8-) delete_certificate (): Mobil Kimlik Uygulamamız için cihazda üretilen RSA anahtar çiftimizin açık anahtarının, akıllı kart üzerinde imzalanarak elde edilen sertifikayı silme işlemini gerçekleştirir. RSA anahtar çiftine verilen geçerlilik süresi otomatik olarak sertifika oluşturma sürecinde sertifikaya işlenir. Geçerlilik süresi bitiminde veya kullanıcı tarafından yeni oluşturulmak istenen sertifika durumunda eskisi silinmelidir. Ayrıca, sertifika ve cihaza ait RSA anahtar çiftine sahip bir cihazda, sadece RSA anahtar çiftinin silinmesi istendiğinde sertifika ile eşleşen bir RSA anahtar çifti olmayacağı için bu fonksiyon çağrılarak sertifika da silinecektir. Bu fonksiyonun kullandığı İç Uygulama Ara Yüzü fonksiyonları delete_rsa_key fonksiyonu ile aynıdır. Sadece, *TEE_OpenPersistenceObject* fonksiyonu için kullanılacak, sertifikanın güvenilir depodaki numarası (101) girilir.

3.2.3 Güvenilir uygulama komut alımı

Kullanıcı tarafı uygulama, güvenilir tarafta yaptıracağı iş veya işlemler için, güvenilir uygulamaya komutlar gönderir. Bu komutlar, hem kullanıcı taraf uygulamada hem de güvenilir uygulama tarafından bilinir. Kullanıcı tarafı uygulamanın altbölüm 3.2.2'de geliştirilen fonksiyonlara ulaşabilmesi için her bir fonksiyona ait komutlar her iki taraf uygulama içinde tanımlanmıştır. Mobil Kimlik Uygulamamız için hazırlanan komutlar şu şekildedir:

```
#define TA_GENERATE_RSA_KEY_CMD 0
#define TA_DELETE_RSA_KEY_CMD 1
#define TA_GET_PUBLICKEY_EXP_MOD_CMD 2
#define TA_DELETE_CERTIFICATE_CMD 3
#define TA_SAVE_CERTIFICATE_CMD 4
#define TA_GET_CERTIFICATE_CMD 5
#define TA_SIGN_CMD 6
#define TA_DIGEST_CMD 7
```

Görüldüğü üzere komutlar, her iki taraf uygulamanın başlık (header) dosyalarına tanımlanmış ve bir sayıyı gösteren C programa dili makrolarıdır. Her bir komutu işaret eden bu sayılar, geliştirilen uygulamalar için eşsiz ve her iki taraf uygulama için aynı olmalıdır. Aksi taktirde çalışmalarda mantık hatalarına yol açacaktır.

Kullanıcı tarafı uygulama, güvenilir uygulamaya yaptırmak istediği işin komutunu yani numarasını yollar. Gelen komut güvenilir uygulamada *TA_InvokeCommandEntryPoint* ara yüz fonksiyonuna düşecektir. Çalışmamızda bu fonksiyon şu şekilde geliştirilmiştir:

```
TEE_Result TA_InvokeCommandEntryPoint (
        void *sess_ctx, uint32_t cmd_id,
        uint32_t param_types, TEE_Param params[4])
{
  /* Unused parameters */
 (void)&sess_ctx;
  switch (cmd_id) {
    case TA_GENERATE_RSA_KEY_CMD:
      return generate_and_save_rsa_key(param_types, params);
    case TA_DELETE_RSA_KEY_CMD:
      return delete_rsa_key(param_types, params);
    case TA_DELETE_CERTIFICATE_CMD:
      return delete_certificate(param_types, params);
    case TA_GET_PUBLICKEY_EXP_MOD_CMD:
      return get_public_key_exponent_modulus(param_types, params);
    case TA_SAVE_CERTIFICATE_CMD:
      return save_certificate(param_types, params);
    case TA_GET_CERTIFICATE_CMD:
      return get_certificate(param_types, params);
    case TA_SIGN_CMD:
      return sign_message(param_types, params);
    case TA_DIGEST_CMD:
      return digest_message(param_types, params);
    default:
      return TEE_ERROR_BAD_PARAMETERS;
}
```

Güvenilir İşletim Ortamı tarafından kullanıcı taraf uygulamadan gelen istek *TA_InvokeCommandEntryPoint* ara yüz fonksiyonuna düşürülür. "sess_ctx" değişkenine oturum ile ilgili bilgiler, "cmd_id" komut olarak kullandığımı numara, "param_types" ve "params" değişkenlerine, güvenilir uygulamada geliştirdiğimiz fonksiyonların kullanacağı girdi-çıktı değişkenleri tanımlanır. "cmd_id" değişkenine tanımlanan komut numarası switch-case yapısında kullanılarak, akış ilgili fonksiyona yönlendirilir.

3.3 Kullanıcı Tarafı Uygulama

Kullanıcı uygulaması, "Güvenilir İşletim Ortamı Kullanıcı Uygulama Geliştirme Ara Yüzü" fonksiyonlarını kullanarak güvenli yol üzerinden güvenli uygulamaya girdileri yollayacak ve gelen çıktıları alacak bir Android Native uygulaması içerir. Kullanıcı tarafı uygulamasının görevlerinden birisi güvenilir uygulamanın bir grafiksel kullanıcı ara yüzü olmasıdır. Son kullanıcı tarafından alınan komutları güvenilir uygulamada gerekli olan fonksiyonlara yönlendirmektedir. Yani mobil kimlik uygulamamızda kimliği oluşturan anahtarların yönetimini sağlamaktadır.

İkinci olarak ise, kullanıcı tarafı uygulama güvenilir tarafta oluşturulan açık anahtarı, akıllı kart üzerinde imzalanmasını sağlamak amacıyla detayları altbölüm 3.4'de anlatılacak olan masaüstü uygulamasına USB kablo üzerinden göndermektir. Burada uygulama NFC teknolojisini veya mobil cihaza takılacak bir akıllı kart okuyucusu ile de ilgili veriyi paylaşabilir. Fakat, örneğimizde kullandığımız kişisel kimlik doğrulama kartlarımızın NFC özelliği olmadığı gibi, akıllı kart üzerinde işlem yapmaya yarayacak Android uyumlu bir kütüphanesi bulunmamaktadır. Akıllı kart üzerinde işlem yapılabilecek kütüphanenin Android uyumlu hale getirilmesi bu çalışmanın kapsamı dışındadır.

Zengin Uygulama adı da verilen kullanıcı tarafı uygulama, güvenli yol üzerinden güvenilir uygulama ile yapacağı mesajlaşman için kuracağı bağlantıyı aşağıdaki "Güvenilir İşletim Ortamı Kullanıcı Uygulama Geliştirme Ara Yüzü" fonksiyonları ile yapmaktadır. Altbölüm 3.2.2'de bahsedilen tüm güvenilir uygulamada ki geliştirdiğimiz fonksiyonlara karşılık gelecek, kullanıcı tarafı uygulamada bir fonksiyon vardır. Güvenilir uygulamada geliştirdiğimiz bu uygulamalara erişmek için kullanıcı taraf uygulamada kullanılması gereken bazı ara yüz fonksiyonları vardır. Bu

ara yüz fonksiyonları “Güvenilir İşletim Ortamı Kullanıcı Ara Yüzü”ne ait olup, fonksiyonlar adları ve açıklamaları aşağıdaki gibidir [18]:

- *TEEC_InitializeContext (const char* name, TEEC_Context* context)*:

  Bu fonksiyon, kullanıcı tarafı uygulama ile Güvenilir İşletim Ortamı arasında bir bağlantı oluşturan yeni bir Güvenilir İşletim Ortamı içeriği başlatır. “name” adlı değişken bağlanılacak Güvenilir İşletim Ortamı’nı tanımlayan bir karakter dizisidir. Geliştirici tarafından bu değişkene “NULL” değeri girilebilir. Böylece, fonksiyon varsayılan Güvenilir İşletim Ortamı üzerinde yeni bir içerik oluşturur. Çalışmamız bünyesinde bu tarz bir yapıya sahip olmadığımız ve gerekli olmadığı için “NULL” değeri kullanılmıştır. İkinci parametre olarak ise, Güvenilir İşletim Ortamı içeriğini ifade eden *TEEC_Context* türünde değişkeninin adresidir. *TEEC_Context*, kullanıcı tarafı uygulamasını belirli bir Güvenilir İşletim Ortamı ile bağlayan ana mantıksal konteyner işlevini görür.

- *TEEC_OpenSession (TEEC_Context* context, TEEC_Session* session,*
  const TEEC_UUID destination,*
  uint32_t connectionMethod,
  const void connectionData,*
  TEEC_Operation operation,*
  uint32_t returnOrigin)*:

  TEEC_OpenSession, kullanıcı tarafı uygulama ile belirtilen güvenilir uygulama arasında yeni bir oturum açar. Güvenilir İşletim Ortamı üzerinde açılan oturum, kullanıcı taraf uygulamasını belirli bir güvenilir uygulama ile bağlayan mantıksal bir kapsayıcı anlamına gelir. *TEEC_Session* türünden oluşturulan boş bir değişken, oturum açılması ile bu fonksiyon tarafından doldurulur. Ayrıca, *TEEC_InitializeContext* ile oluşturduğumuz Güvenilir Uygulama içeriği bu fonksiyonun ilk parametresini oluşturur.

  Güvenilir İşletim Ortamı’nda güvenilir uygulamalar birbirinden UUID adlı verilen 128 bitlik evrensel olarak benzersiz tanımlayıcı bir sayı ile ayrılır. Bu sayı güvenilir uygulamamızın derlenmesi aşamasında uygulamaya tanımlanır. Kullanıcı tarafı uygulama bu sayıya denk gelen güvenilir uygulama üzerinde oturum açar. İlgili fonksiyonumuz da *TEEC_UUID* türünde “destination” adlı bir göstericiye UUID’nin adresi verilir. Çalışmamızda güvenilir

uygulamamızın 128 bitlik sayısı, (c4d22850-20f1-4b12-823aeb7aefba62)'dir. Kullanıcı tarafı uygulamamız olan Android Native uygulamamızın native kısmında bu sayı bir C programlama dili makrosu ile başlık dosyasına eklenmiştir.

```
#define TA_MIC_UUID {0xc4d22850, 0x20f1, 0x4b12,
                     {0x82, 0x3a, 0xeb, 0x7a, 0xef, 0xba, 0xe4, 0x62}}
```

Ayrıca, *TEEC_OpenSession* fonksiyonunda değinilmesi gereken hususlardan biriside bağlantı metotlarıdır. Bu metodlar, oturum giriş verileri hakkında bilgi verir. Bu metotlar aşağıdaki gibidir:

TEEC_LOGIN_PUBLIC
TEEC_LOGIN_USER
TEEC_LOGIN_GROUP
TEEC_LOGIN_APPLICATION
TEEC_LOGIN_USER_APPLICATION
TEEC_LOGIN_GROUP_APPLICATION

Yukarıda adlarından da anlaşılacağı üzere, kullanıcı, kullanıcı grupları veya kullanıcı tarafı uygulamanın kendi sağlayacağı kullanıcı, kullanıcı grupları verileri ile giriş izni alınır ve oturum açılır. Her bir bağlantı metodunun kullanıma göre ilgili veriler "connectionData" adlı değişkene verilir. Çalışmamızda donanımsal gereksinimlerin kısıtlılığı ve OP-TEE Güvenilir İşletim Ortamı'nın da güvenilir girdi kaynakları sunmamasından ötürü (Bölüm 5) *TEEC_LOGIN_PUBLIC* metodu kullanılmıştır. Böylece, herhangi bir bağlantı verisine de ihtiyaç duyulmadığı için "connectionData" adlı değişkene NULL değeri verilir.

Güvenilir uygulama ile bir oturum açma esnasında, bir girdi-çıktı kümesi değiş tokuş edilmek istenebilir. Çalışmamızın gereksinimlerinde oturum açma esnasında buna ihtiyaç olmadığı için "operation" adlı değişkene NULL değeri verilmiştir. *TEEC_Operation* türü hakkında detaylı bilgi altbölüm 3.3.1'de verilmiştir. Son olarak, "returnOrigin" adlı gösterici değişken ile geliştiriciye kolaylık sunulmuştur. Oturum açma işleminin başarısız olması durumunda kaynağının nere olduğu göstererek, *TEEC_OpenSession* fonksiyonu bu değişkene bir sayı değeri atar. Bu sayılar kaynak [18]'de tanımlanmıştır. Buna göre de geliştirici hatanın, kullanıcı uygulama geliştirme ara yüzünde,

iletişimde, güvenilir uygulama da veya Güvenilir İşletim Ortamı'nda olup olmadığını anlar.

- *TEEC_InvokeCommand (TEEC_Session* session, uint32_t commandID,*
  TEEC_Operation operation,*
  uint32_t returnOrigin)*:

  Açılan oturum üzerinde, güvenilir uygulamaya yaptıracağı iş ve işlemler için gerekli komutu ve girdi-çıktı verilerini gönderir. Çalışmamız gereksinimlerine göre geliştirdiğimiz güvenilir uygulamada ki bir fonksiyona erişmemiz için gerekli komut bu fonksiyona verilerek güvenilir tarafa yollanır. Ayrıca, güvenilir tarafta yaptıracağımız işlemler için kullanıcı taraf uygulamadan alınan verilerin de güvenilir uygulamaya gönderilmesi gerekmektedir. Bunun için *TEEC_Operation* türünde oluşturduğumuz bir girdi-çıktı kümesinin adresi bu fonksiyona verilir. Bu konuda detaylı bilgi altbölüm 3.3.1'de verilmiştir.

- *TEEC_CloseSession (TEEC_Session* session)*:

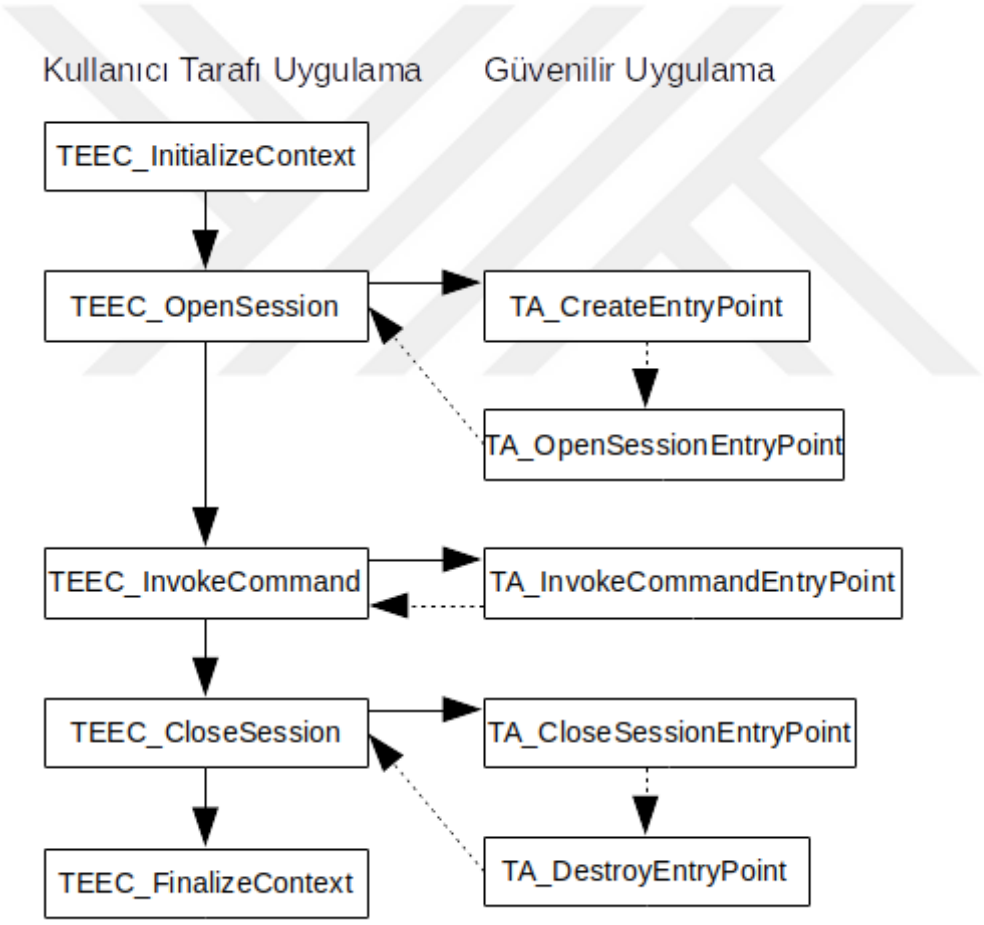
  Güvenilir uygulama üzerinde açılan oturumu kapatır. Bunun için *TEEC_OpenSession* fonksiyonu ile elde edilen *TEEC_Session* türündeki nesnenin adresi fonksiyona verilerek yapılır.

- *TEEC_FinalizeContext (TEEC_Context* context)*:

  Güvenilir İşletim Ortamı içeriğini sonuçlandırır. *TEEC_InitializeContext* fonksiyonu ile elde ettiğimiz *TEEC_Context* türündeki nesnenin adresi verilerek bu işlem gerçekleştirilir.

Şekil 3.2'de kullanıcı taraf uygulamanın güvenilir uygulama ile iletişime geçmek için gerekli fonksiyonların akışı görülmektedir. Ayrıca, burada güvenilir uygulamanın hayat döngüsünün kullanıcı taraf uygulama tarafından kontrol edildiğini de görebiliriz. Bir kullanıcı tarafı uygulamanın, güvenilir uygulamaya yaptıracağı iş ve işlemler, kullanıcı taraf uygulamada *TEEC_InitializeContext* fonksiyonu ile Güvenilir İşletim Ortamı içeriğinin oluşturulması ile başlar. Kullanıcı tarafta *TEEC_OpenSession* fonksiyonun çağrılması ile güvenilir uygulamada *TA_CreateEntryPoint* adlı yapılandırıcı fonksiyon ile güvenilir uygulamanın örneği oluştulur. Daha sonra, *TA_OpenSessionEntryPoint* giriş noktası ile oluşturulan güvenilir uygulama örneği üzerinde oturum açılır ve oturumun açılması sonucu *TEEC_OpenSession* fonksiyonuna geri döner. Açılan oturum bilgisi ile beraber, kullanıcı taraf uygulama

TEEC_InvokeCommand fonksiyonu ile güvenilir tarafa yapılmak istenilen iş ve işlemler için komut gönderir. Güvenilir tarafta *TA_InvokeCommandEntryPoint* güvenilir uygulama ara yüz fonksiyonuna, kullanıcı tarafından yollanan komut düşer. Yapılan işin güvenilir tarafta sonuçlanması akabinde çıktılar kullanıcı tarafa gönderilir. Bunun akabinde artık kullanıcı taraf uygulama oturumu kapatmak için *TEEC_CloseSession* fonksiyonunu çağırılması ile güvenilir uygulamada, *TA_CloseSessionEntryPoint* fonksiyonu ile oturum kapatılır, *TA_DestroyEntryPoint* fonksiyonu ile beraber güvenilir uygulamanın oluşturulan örneği silinir. Güvenilir tarafta kapatma işlemlerinin başarıyla bitmesi sonucunda da, kullanıcı tarafta *TEEC_FinalizeContext* fonksiyonu ile Güvenilir İşletim Ortamı içeriği de sonlandırılacaktır.



Şekil 3.2 : Kullanıcı taraf uygulamanın güvenilir uygulama ile bağlantısı.

Kullanıcı tarafı uygulamamız, güvenilir uygulamaya bağlantı sunması yanında, kullanıcıya grafiksel bir ara yüzde sunmaktadır. Bu nedenle kullanıcı tarafı uygulamamız iki kısımda incelenebilir.

3.3.1 Android Native kullanılarak güvenilir uygulama ile haberleşen program

OP-TEE Güvenilir İşletim Ortamı'na ait "Güvenilir İşletim Ortamı Kullanıcı Uygulama Geliştirme Ara Yüzü" kütüphanesi C programlama dili ile yazılmış olup, bu yazılım kütüphanesi Android İşletim Sistemi'nin /system/lib64 dizininde "libteec.so" adı ile bulunmaktadır. Bu durum bize, Android uygulamamız içinde C programlama dili kullanmamız gerekliliğini göstermektedir. Güvenilir uygulama ile iletişimimizi sağlayacak bu yazılım kütüphanesine erişim için Android uygulama geliştirme ortamı bize NDK (Native Development Kit) adı verilen teknolojisini sunmaktadır [19]. Android NDK kütüphanesi Java programlama dili yazılan bir koddan C programlama dilinde yazılmış fonksiyonları çağırmamızı sağlayacaktır.

Android uygulamamızda C programlama dili ile geliştirdiğimiz Native kısım, son kullanıcı tarafından yapılan istekleri, ara yüz fonksiyonları ile güvenilir tarafa iletecektir. Gereksinimlerimize göre 2048 bit bir RSA anahtar çifti oluşturup bunu güvenilir depoda saklayan, güvenilir uygulamamızda ki "generate_and_save_rsa_key" fonksiyonu Android uygulamamızdan aşağıdaki gibi çağrılır:

1-) Güvenilir İşletim Ortamı içeriği oluşturulur. *TEEC_Context* türünden "context" adlı değişkende bu içerik tutulur.

```
TEEC_InitializeContext (NULL, &context);
```

2-) Güvenilir uygulamamız üzerinde yeni bir oturum açılır. İlk maddede oluşturulan içerik aşağıdaki fonksiyona, bağlantı methodu, güvenilir uygulamanın UUID'si ve yeni oluşturulacak oturum içeriğini tutacak *TEEC_Session* türünden "session" adlı değişken verilir.

```
TEEC_OpenSession (&context, &session, &uuid,
                  TEEC_LOGIN_PUBLIC, NULL, NULL, &err_origin);
```

3-) Bir güvenilir uygulamaya gönderilecek girdi ve güvenilir uygulamadan alınacak çıktı parametreleri verilir.

```
operation.paramTypes = TEEC_PARAM_TYPES (
    TEEC_VALUE_OUTPUT,
    TEEC_NONE,
    TEEC_NONE,
    TEEC_NONE);
```

Yukarıda görüldüğü üzere, “operation” adlı bir değişken bulunmaktadır. Bu değişken, *TEEC_Operation* adı verilmiş bir C Progralama Dili’ndeki Yapı (Structure) türüne aittir.

```
typedef struct {
        uint32_t started;
        uint32_t paramTypes;
       TEEC_Parameter params[ 4 ];
} TEEC_Operation;
```

TEEC_Operation türü sayesinde güvenilir uygulamaya gönderilecek komutlara göre girdi-çıktı parametreleri belirlenir. Yukarıdaki yapı türü tanımlama ifadesinde görüleceği gibi tanımlanabilecek parametre sayısı en fazla dörttür. “generate_and_save_rsa_key” fonksiyonumuzda, güvenilir tarafta işlemin başarılı bitip bitmediğini gösterecek “int” türünden bir çıktı parametresine ihtiyaç duyulmuştur. Bunun için *TEEC_PARAM_TYPES* adlı C makrosuna ilk parametrenin türü *TEEC_VALUE_OUTPUT* ile tanımlanarak, başka parametrenin olmadığı için diğer parametre tiplerinde *TEEC_NONE* girilir. Makronun geri dönüş değeri “paramTypes” değişkenine yazılır. Parametre tiplerinin tanımlanması sonucunda, oluşturulan parametreye değer atanarak başlatılır.

```
operation.params[0].value.a = 0;
```

İfadede görüldüğü üzere, tipi *TEEC_VALUE_OUTPUT* ile tanımlanan parametreye bir değer atanır. Güvenilir uygulamadan dönecek sonuç ise yine “operation” değişkeninin “params” dizisinin ilk elemanına yazılır. Sonuç olarak, “operation” değişkeni tekrar güvenilir uygulama tarafından oluşturulmaz. İki taraf arasındaki haberleşme esnasında hangi bellek adresinde oluşturulduysa, tüm işlem boyunca Güvenilir İşletim Ortamı tarafından o adreste tutulur. Fonksiyonumuz için kullandığımız bu parametreden “0” döner ise işlem başarısız, “1” dönerse güvenilir uygulamanın başarıyla işlemi tamamladığı anlaşılır.

4-) Açılan oturum bilgisi, ilgili fonksiyonun komut numarası ve tanımlanan parametrelerin bilgisi güvenilir uygulamaya gönderilir.

```
TEEC_InvokeCommand (&session, TA_GENERATE_RSA_KEY_CMD,
                    &operation, &err_origin);
```

5-) Oturum kapatılıp, Güvenilir İşletim Ortamı içeriğide sonlandırılarak güvenilir uygulamanın gönderdiği sonuç Android uygulamamızın Java kısmına gönderilir.

```
TEEC_CloseSession(&session);
TEEC_FinalizeContext(&context);
```

Güvenilir uygulamada geliştirilen fonksiyonlara karşılık gelecek kullanıcı tarafındaki fonksiyonlar aynı şekilde bu beş adımda çağırılır. Farklılıklar sadece *TEEC_InvokeCommand* fonksiyonu içindeki komut numarası ve parametre tipleridir.

Güvenilir uygulamadaki fonksiyonlara göre kullanıcı taraftaki parametre tanımlamaları ve komut gönderimi aşağıdaki gibidir:

1-) get_public_key_exponent_modulus (): Güvenilir depodan RSA anahtar çiftimizin açık anahtarı bileşenleri olan üs ve modül bilgilerini çeken bu güvenilir taraf fonksiyona parametre aşağıdaki tipte gönderilir.

```
operation.paramTypes = TEEC_PARAM_TYPES (
    TEEC_MEMREF_TEMP_OUTPUT,
    TEEC_MEMREF_TEMP_OUTPUT,
    TEEC_VALUE_OUTPUT,
    TEEC_NONE);
```

Güvenilir uygulama işlemin başarılı olması durumunda açık anahtarın üs ve modül verisini kullanıcı tarafa geri gönderecektir. Bunun için, parametreler tanımlanırken, ilgili veri için bellekte bir alan ayrılarak adresleri parametre olarak tanımlanır. Çalışmamızda bu alan *RSA_KEY_SIZE* makrosu ile tanımlanmış 2048 bittir. Bu tip parametreler için tip olarak *TEEC_MEMREF_TEMP_OUTPUT* tanımlanır. Parametre dizisinin ilk parametresi üs, ikinci parametre ise modül verisi içindir. Üçüncü parametre ise yine fonksiyonun başarı durumu için ayrılmış parametredir. “0” ise işlem başarısız, “1” sonucu ise işlem başarılıdır.

```
out_buffer1 = malloc(RSA_KEY_SIZE * sizeof(uint8_t)); // üs
out_buffer2 = malloc(RSA_KEY_SIZE * sizeof(uint8_t)); // modul
operation.params[0].tmpref.buffer =    out_buffer1;
operation.params[0].tmpref.size = RSA_KEY_SIZE;
operation.params[1].tmpref.buffer = out_buffer2;
operation.params[1].tmpref.size = RSA_KEY_SIZE;
```

Komut numarası ise *TA_GET_PUBLICKEY_EXP_MOD_CMD* makrosu ile tanımlanmıştır.

```
TEEC_InvokeCommand (&session,
                    TA_GET_PUBLICKEY_EXP_MOD_CMD,
                    &operation, &err_origin);
```

2-) save_certificate (): Detayları önceki bölümlerde de anlatılan mobil cihaz üzerinde üretilen RSA anahtar çiftinin açık anahtarı, akıllı kart üzerinde imzalanarak bir sertifika oluşturulur. Bu sertifikanın da güvenilir depoda tutulması gerekliliği Mobil Kimlik uygulamamızın başlıca gereksinimlerindendir. Bunun için kullanıcı taraf uygulamamızda bir girdi alanı tanımlanıp bunun adresi "operation" değişkenine tanımlanmalıdır. Bunun için;

```
operation.paramTypes = TEEC_PARAM_TYPES (
        TEEC_MEMREF_TEMP_INPUT,
        TEEC_VALUE_OUTPUT,
        TEEC_NONE,
        TEEC_NONE);
operation.params[0].tmpref.buffer = buffer;
operation.params[0].tmpref.size = buffer_len;
operation.params[1].value.a = 0
```

"buffer" adlı değişken sertifika içeriğini tutan bellek alanının adresini tutar. Bu adres ve boyutu parametreye tanımlanır. İkinci parametre olarak yine işlemin başarı durumunu tutacak çıktı değişkenidir. "0" ise sertifika kaydedilemediğini, "1" ise sertifikanın başarıyla güvenilir depoya kaydedildiği anlamına gelir.

```
TEEC_InvokeCommand (&session,
                    TA_SAVE_CERTIFICATE_CMD,
                    &operation, &err_origin);
```

TA_SAVE_CERTIFICATE_CMD makrosu ile tanımlanmış komut numarası ile güvenilir tarafa gönderilir.

3-) get_certificate (): Güvenilir uygulamada geliştirdiğimiz bu fonksiyon için kullanıcı taraf uygulamada, çıktı verisi olarak sertifikanın yazılacağı bir adres ve boyut bilgisi için parametre tipleri girilir. İlgili çıktı verisi için bellekte ayrılan alanın adresi ve

boyut bilgisi girildikten sonra *TA_GET_CERTIFICATE_CMD* makrosu ile tanımlanan komut numarası ile güvenilir tarafa gönderilir.

```
operation.paramTypes = TEEC_PARAM_TYPES (
        TEEC_MEMREF_TEMP_OUTPUT,
       TEEC_VALUE_OUTPUT,
      TEEC_NONE,
           TEEC_NONE);
out_buffer = malloc(RSA_KEY_SIZE * sizeof(uint8_t));
operation.params[0].tmpref.buffer = out_buffer;
operation.params[0].tmpref.size = RSA_KEY_SIZE;
TEEC_InvokeCommand (&session,
                        TA_GET_CERTIFICATE_CMD,
                       &operation, &err_origin);
```

4-) digest_message () ve sign_message (): Güvenilir uygulama da geliştirdiğimiz bu fonksiyonlar, açık bir mesajın imzalama işlemi için kullanılır. Altbölüm 3.2.1'de anlatıldığı üzere, açık mesajın özeti alınır ve imzalama işlemine tabi tutulur. Kullanıcı taraf uygulamasında ki imzalama işleminde, güvenilir taraf uygulamamızın bu iki fonksiyonu aynı fonksiyon içinden çağırılır.

İlk olarak, açık metnin özeti için güvenilir uygulamaya komut gönderilir. Bunun için gerekli parametre tipleri aşağıdaki gibi tanımlanmıştır.

```
operation.paramTypes = TEEC_PARAM_TYPES (
        TEEC_MEMREF_TEMP_INPUT,
        TEEC_MEMREF_TEMP_OUTPUT,
        TEEC_VALUE_OUTPUT,
        TEEC_NONE);
```

İlk parametre açık mesaj olacağından bu bir bellekte girdi alanı olacaktır. Cevap olarak gelecek mesajın özeti ikinci parametrede çıktı alanı olacaktır. Son olarak, üçüncü parametrede fonksiyonunu başarılı çalışıp çalışmadığı kontrol edilir.

```
operation.params[0].tmpref.buffer = message;
operation.params[0].tmpref.size = message_len;
operation.params[1].tmpref.buffer = digest;
operation.params[1].tmpref.size = digest_len;
operation.params[2].value.a = -1;
```

İlk parametre için açık metnin bulunduğu bellek alanın adresi ve boyutu verilir. İkinci parametreden özet verisini alacağımızdan gerekli alan aşağıdaki gibi oluşturulur, adres ve boyut bilgisi girilir. *TA_DIGEST_CMD* makrosunda tanımlanan komut numarası ile güvenilir uygulamaya gönderilir.

```
TEEC_InvokeCommand (&session,
                    TA_DIGEST_CMD,
                    &operation, &err_origin);
```

İkinci olarak, elde edilen özet bilgisi imzalanmak üzere güvenilir uygulamaya gönderilir. Güvenilir uygulamada özet alma işlemi akabinde imzalama işlemi de gerçekleştirilebilirdi. Geliştirdiğimiz prototip uygulamamızda, açılan tek bir oturum üzerinden birden fazla güvenilir uygulamaya komut gönderilebileceğini göstermek istememizden dolayı kullanıcı taraf fonksiyondan iki kez güvenilir uygulamaya komut gönderilmiştir.

```
operation.paramTypes = TEEC_PARAM_TYPES (
    TEEC_MEMREF_TEMP_INPUT,
    TEEC_MEMREF_TEMP_OUTPUT,
    TEEC_VALUE_OUTPUT,
    TEEC_NONE);
```

İlk parametre, özet bilgisinin bellekteki alanını gösteren adres olacağından tipi *TEEC_MEMREF_TEMP_INPUT* olarak tanımlanmıştır. İkinci parametre ise imzalanmış veriyi geri kullanıcı tarafına verecek olan çıktı alanıdır. Üçüncü parametre ise yine işlemin başarı durumunu bize gösterir.

```
operation.params[0].tmpref.buffer = digest;
operation.params[0].tmpref.size = digest_len;
operation.params[1].tmpref.buffer = signed_message;
operation.params[1].tmpref.size = signed_message_len;
operation.params[2].value.a = -1;
```

İlk girdi parametresi olarak özet verisi ve boyutu girilmiştir. İkinci parametrede, cevap olarak dönecek imza verisinin yazılacağı bellek adresi ve boyutu girilmiştir. Başarılı bir şekilde imzalama işlemi, üçüncü parametrede "1" olarak döner ve imzalı mesaj kullanıcıya verilir. Son olarak, imzalama işlemi için gönderilecek komut numarası *TA_SIGN_CMD* makrosu ile tanımlıdır.

```
TEEC_InvokeCommand (&session,
                    TA_SIGN_CMD,
                    &operation, &err_origin);
```

5-) delete_rsa_key (): Bu fonksiyon için yapılan parametre tanımlaması "generate_and_save_rsa_key" fonksiyonu ile aynıdır. Güvenilir uygulamadan başarı durumu ile ilgili durum bilgisi gelir. "0" ise işlem başarısız, "1" sonucu ise işlem başarılıdır. Komut numarası *TA_DELETE_RSA_KEY_CMD* makrosu ile tanımlanmıştır.

```
operation.paramTypes = TEEC_PARAM_TYPES (
        TEEC_VALUE_OUTPUT,
        TEEC_NONE,
        TEEC_NONE,
        TEEC_NONE);
operation.params[0].value.a = 0;
TEEC_InvokeCommand (&session,
                    TA_DELETE_RSA_KEY_CMD,
                    &operation, &err_origin);
```

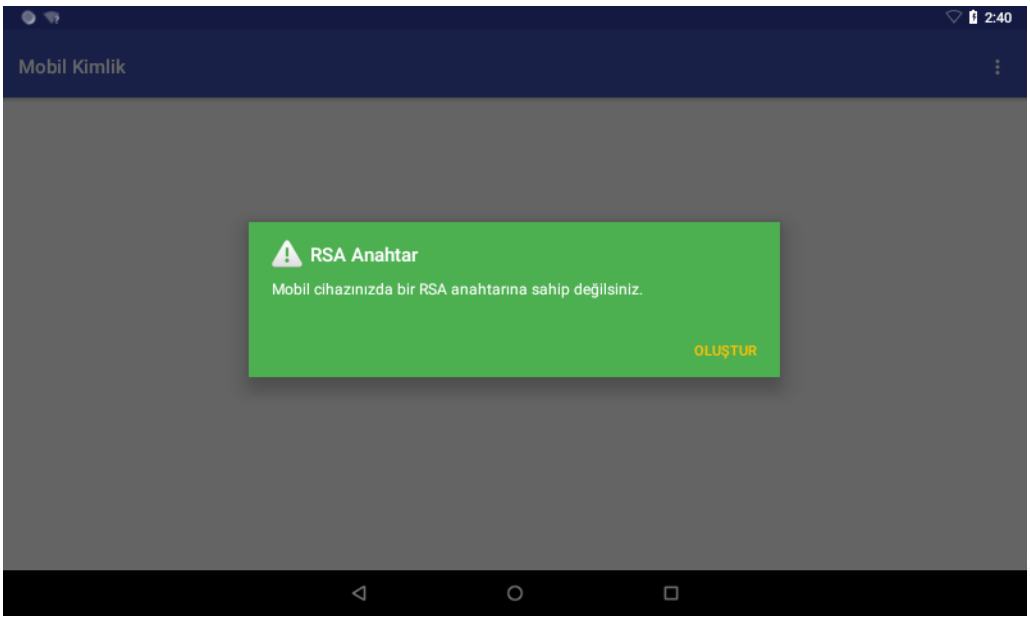
6-) delete_certificate (): Kullanıcı tarafta bu fonksiyon için hazırlanacak parametre, işlemin sonuç bilgisini tutacak çıktı değişkenidir. "0" ise sertifika başarı ile silinememiş, "1" ise başarılı ile güvenilir depodan silinmiştir.

```
operation.paramTypes = TEEC_PARAM_TYPES(
        TEEC_VALUE_OUTPUT,
        TEEC_NONE,
        TEEC_NONE,
        TEEC_NONE);
operation.params[0].value.a = 0;
TEEC_InvokeCommand (&session,
                    TA_DELETE_CERTIFICATE_CMD,
                    &operation, &err_origin);
```

Komut numarası *TA_DELETE_CERTIFICATE_CMD* makrosu ile tanımlanmıştır.

3.3.2 Android grafiksel kullanıcı ara yüzü

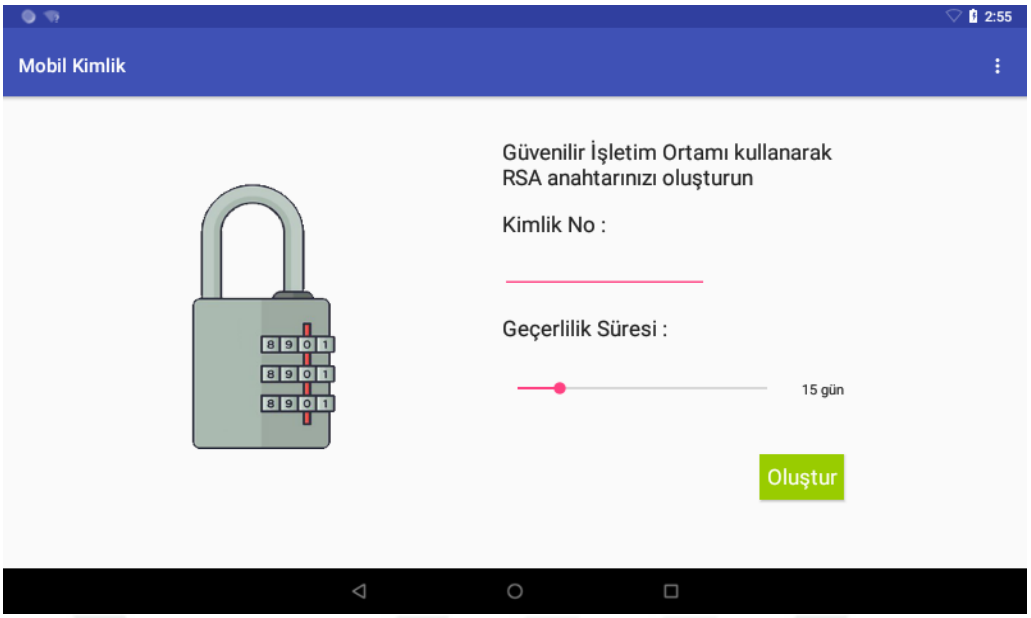
Mobil Kimlik uygulamamızın, son kullanıcı tarafından görülen yüzüdür. Bu kısımda, Mobil Kimlik uygulamamız için yapılması gereken iş ve işlemlerin son kullanıcı tarafından yapılmasını sağlayacak ekranlar geliştirilmiştir. Bu kısımda, son kullanıcı tarafından gelen girdiler kontrol edildiği gibi, güvenilir uygulamaya erişecek native fonksiyonları çağırır. Tüm bunlara ek olarak, son kullanıcıya ait mobil kimliğin, son kullanıcı tarafından idaresini yapabildiği kısım diyebiliriz.



Şekil 3.3 : Mobil cihazda herhangi bir RSA anahtar çiftine sahip olunmaması durumunda kullanıcıya gösterilen mesaj.

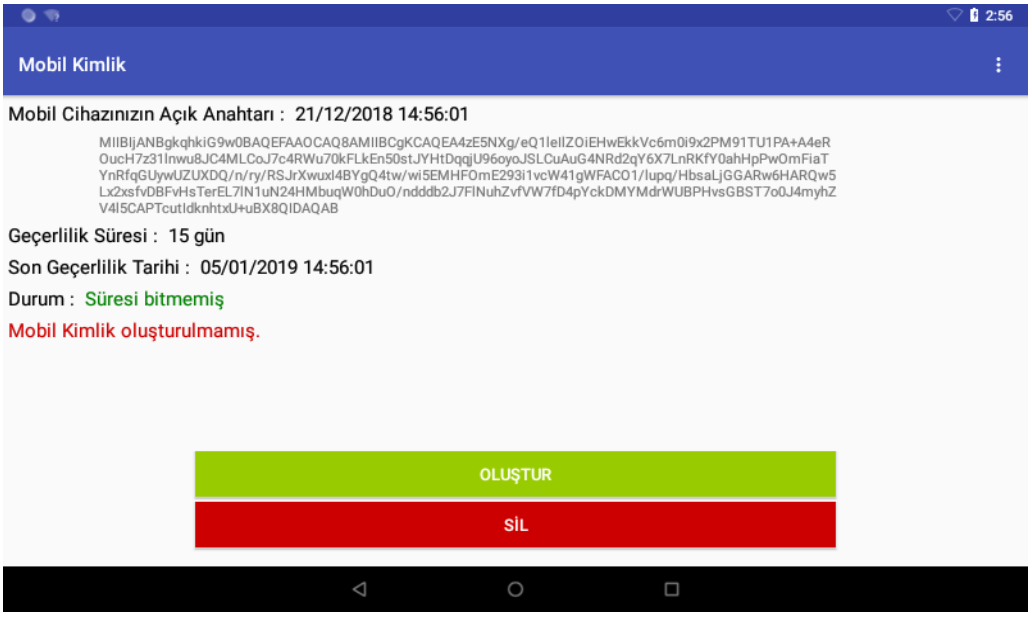
Şekil 3.3'de görüldüğü üzere, Mobil Kimlik Uygulamamızı ilk defa açan bir son kullanıcı, mobil cihazında oluşturulmuş herhangi bir RSA anahtar çifti olmadığı mesajını görür. "Oluştur" mesajı ile şekil 3.4'deki ekrana yönlendirilir.

Son kullanıcıdan şekil 3.4'deki ekran üzerinden kimlik numarası ve oluşturacağı RSA anahtar çifti için bir geçerlilik süresi vermesi istenir. Bu bilgiler detayları altbölüm 3.3'de anlatılacak olan RSA anahtar çiftinin açık anahtarının akıllı kart üzerinde imzalanması aşamasında kullanılır. Yani, oluşturulacak RSA çifti ile kimlik numarasının bir ilişkisi yoktur. Sadece, burada ki geçerlilik süresi hem anahtar çiftinin hem de oluşturulacak Mobil Kimlik sertifikasının geçerlilik süresi olacaktır.



Şekil 3.4 : Güvenilir İşletim Ortamı üzerinde RSA anahtar çiftinin oluşturulması.

Şekil 3.5'te son kullanıcıya mobil cihazına ait açık anahtar gösterilir. Bu açık anahtar verisi, her zaman Güvenilir İşletim Ortamı'nın güvenilir deposunda tutulur. Mobil kimlik oluşturulmamış bir mobil cihazda, uygulamamız açıldığında bu ekran arka tarafta güvenilir depodan açık anahtarı çeker ve kullanıcıya gösterir. Ayrıca, açık anahtarın geçerlilik süresi ve tarihide bu ekranda kullanıcıya gösterilir.



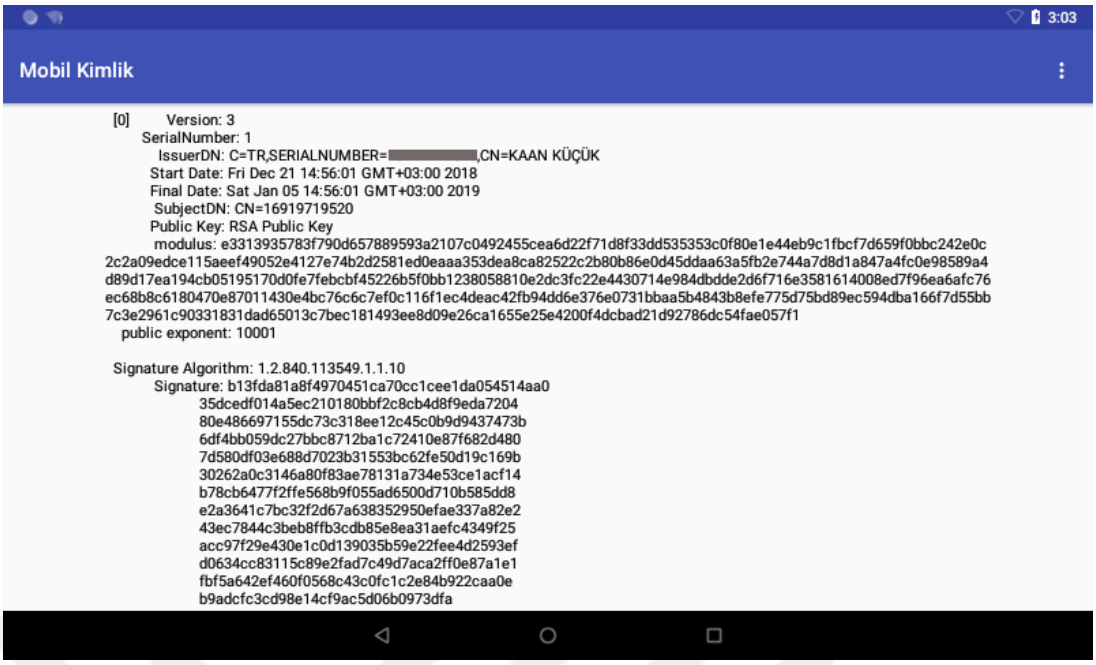
Şekil 3.5 : Sadece RSA anahtarın Güvenilir İşletim Ortamında üretilmesi.

Açık anahtarın geçerlilik süresinin dolmadığı durumlarda, son kullanıcının USB ile mobil cihazını bilgisayara bağlayarak detayları altbölüm 3.4'de anlatılan masaüstü

uygulamasını açması beklenir.Ekranda görülen “Oluştur” butonuna tıklaması sonucunda, son kullanıcıdan masaüstü uygulamasındaki işlemleri gerçekleştirilmesi beklenir. Son kullanıcı geçerlilik süresi dolan bir açık anahtardan mobil kimlik oluşturamayacağından veya RSA anahtar çiftini yenilemek istediği durumlarda, var olan anahtar çiftini “Sil” butonu ile güvenilir depodan silebilecektir.

Şekil 3.6 : Mobil kimliğin Güvenilir İşletim Ortamında üretilmesi.

Masaüstü uygulaması ile akıllı kart üzerinde imzalama yapılarak oluşturulan mobil kimliğimiz ile ilgili detaylar şekil 3.6’da ki gibidir. USB kablo aracılığı ile masaüstü uygulamamızda oluşturduğumuz Mobil Kimlik, mobil cihaza aktarılır aktarılmaz Güvenilir İşletim Ortamı’na aktarılarak güvenilir depoda tutulur. Süresi dolan veya mobil kimliğini yenilemek isteyen son kullanıcı “Sil” butonu ile bu işlemi gerçekleştirebilecektir. Bu durumda, güvenilir depoda bulunan RSA açık anahtar çifti ve mobil kimlik sertifikası silinerek, son kullanıcı uygulamanın ana sayfasında şekil 3.3’yi görecektir. Ayrıca mobil kimliğini oluşturan sertifikanın detaylarını görmek isteyen son kullanıcı “Getir” butonuna tıklayarak görebilecektir. Sertifika detayı şekil 3.7’da ki gibi gösterilecektir.



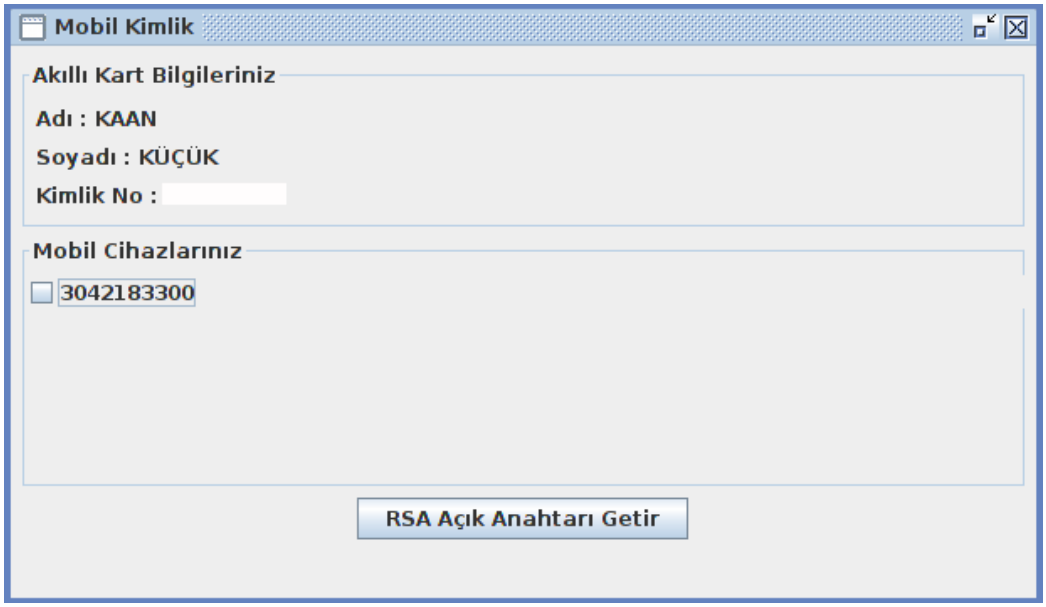
Şekil 3.7 : Mobil kimliğin, kimlik kartı tarafından imzalanması sonrası oluşturulan sertifika.

3.4 Masaüstü Uygulama

Cihazımızın üzerinde oluşturulan RSA anahtar çiftinin kullanılmasından önce açık anahtar akıllı kart üzerinde bir kereye mahsus olarak imzalanması gerekmektedir. Daha sonra mobil kimlik yenilemelerinde veya geçerliliği dolan anahtarlarımız için bu işlemi tekrarlamanız gerekecektir.

Daha önceden de bahsedildiği gibi, akıllı kartımız üzerinde NFC teknolojisi olmadığı ve akıllı kart üzerinde işlem yapmaya yarayacak yazılım kütüphanesinin Android İşletim Sistemi'ne uyumsuz olmasından ötürü bir masaüstü uygulaması geliştirilmiştir. Bu uygulama, akıllı kart okuyucu ve mobil kimlik uygulamamız arasında bir köprü niteliği görmektedir. İlk olarak masaüstü uygulamamız, mobil kimlik uygulamamız ile iletişime geçerek cihazda bulunan açık anahtarı alır. Daha sonra kart okuyucusu yardımıyla, akıllı kart sahibi hakkında açık bilgiler kullanıcıya gösterilir. Son olarak kullanıcıdan alınan akıllı karta ait PIN ve PEN numarasının doğru girilmesi sonrası, açık anahtarımız kart üzerinde imzalanıp tekrar mobil kimlik uygulamamıza gönderilir. Mobil Kimlik Uygulamamızda imzalanmış bu açık anahtarı güvenilir tarafta depolayacaktır. Görüldüğü üzere, geliştirilen uygulama kart üzerinde imzalama işlemini yapacak ve mobil uygulamamızla haberleşecek küçük bir masaüstü uygulamasıdır.

Akıllı kartlar üzerinde bulunan çiplere bilgilerimiz, belirli güvenlik koşulları, standartlar ve dizayn desenleri kullanılarak yazılır. Bununla ilgili detaylar bilinmedikçe kartın okunması zor olacağı gibi, çip üzerindeki verinin okunması için ayrıca bir çalışma yapılması gerekecektir. Çalışmamızda kullandığımız açık anahtar altyapısını kullanan Türkiye Cumhuriyeti Kimlik Kartı üzerinde yapılan okuma, PIN ve PEN doğrulama, imzalama işlemleri için TÜBİTAK'ın geliştirdiği "TCKK Uygulama Geliştirme Ara Yüzü" kütüphanesi kullanılmıştır. Bu kütüphanenin sunduğu fonksiyonlar sayesinde kartın herhangi bir detayını bilmemize gerek kalmadan, kart üzerinde rahatlıkla işlem yapabilmekteyiz.

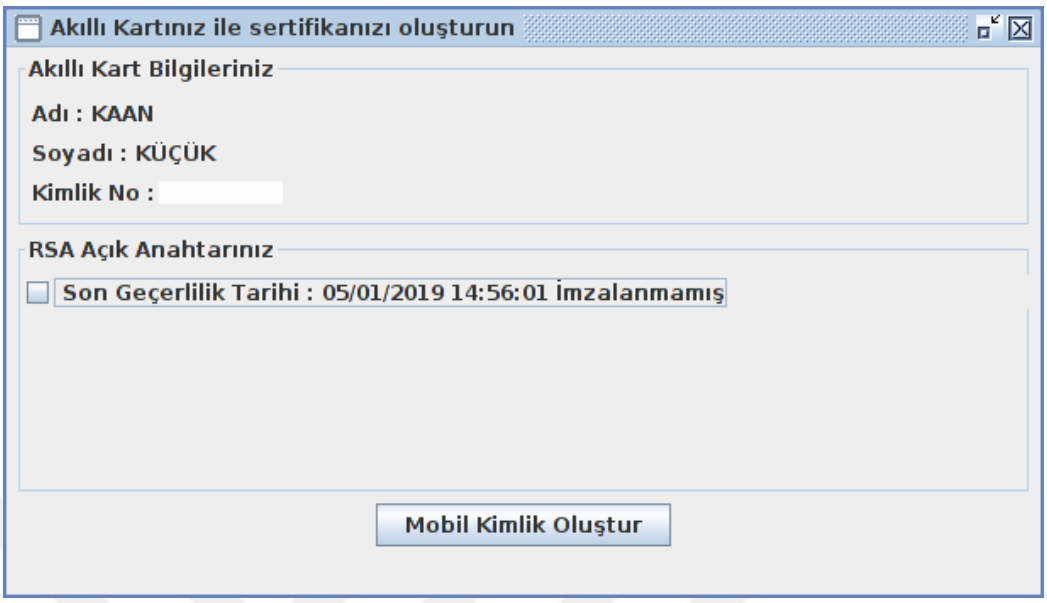


Şekil 3.8 : Masaüstü uygulama başlangıç ekranı.

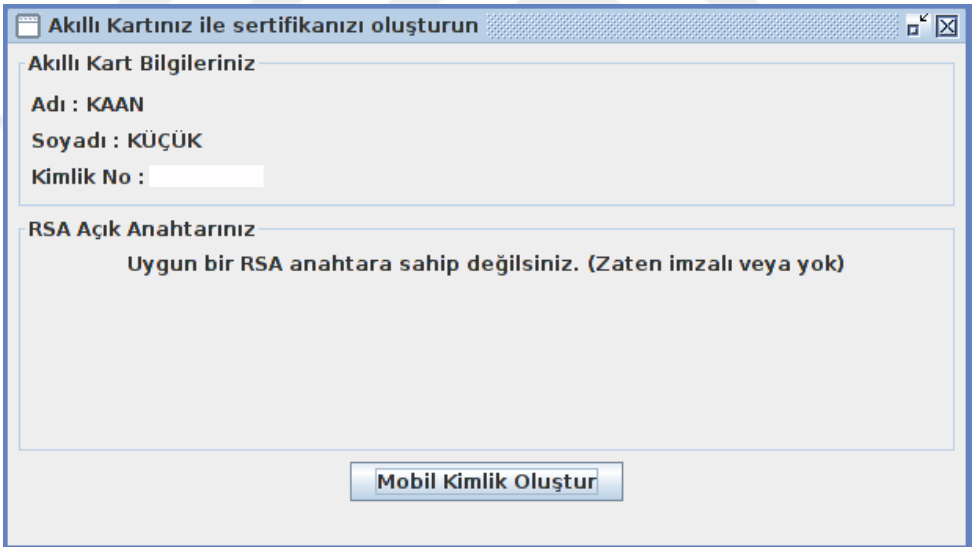
Mobil kimliğimizi ifade edecek sertifikayı oluşturmak için kullanıcıdan mobil cihazını USB ile bilgisayara bağlaması gerektiğini söylemiştik. Ayrıca son kullanıcı bu aşamada standart bir akıllı kart okuyucusu ile kimlik kartını bilgisayara bağlar. Gerekli fiziksel bağlantıların yapılmasından sonra son kullanıcı masaüstü uygulamamızı açabilir. Şekil 3.8'de gösterildiği üzere, masaüstü uygulamamız bilgisayara bağlı bulunan kimlik kartımız üzerinden herhangi bir PIN doğrulama işlemine gerek kalmadan alabildiği açık kişi bilgilerini ve bilgisayara bağlı Android mobil cihazları gösterir.

Mobil Kimlik Uygulamamızın çalıştığı Android mobil cihaz seçilerek "RSA Açık Anahtarı Getir" butonuna tıklanır. Bu aşamada, masaüstü uygulamamız, Mobil Kimlik

Uygulaması ile iletişime geçer. Uygun bir açık anahtar varsa, Şekil 3.9'de ki gibi bir ekran ile kullanıcı bilgilendirilir.



Şekil 3.9 : Masaüstü uygulamasında mobil kimliğin (açık anahtarın) kimlik kartı tarafından imzalanmasını sağlayan ara yüz.

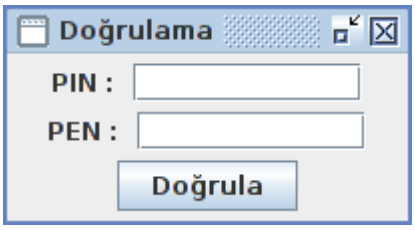


Şekil 3.10 : Mobil Kimliğin hazır olmaması durumunda masaüstü uygulamasında beliren hata mesajı.

Eğer uygun bir açık anahtarın olmaması durumda ise şekil 3.10'da ki gibi bir uyarı kullanıcıya gösterilecektir. Uygulamamızda, akıllı kart üzerinde imzalamaya uygun olmayan bir RSA açık anahtar özellikleri şu şekildedir:

- Açık anahtar zaten imzalanarak bir mobil kimlik vardır.
- Açık anahtarın geçerlilik süresi dolmuştur.

- Mobil cihazda açık anahtar oluşturulurken girilen kimlik numarası, kimlik kartının kimlik numarası ile eşleşmiyordur.
- Son olarak, uygulamada zaten herhangi bir RSA anahtarı yoktur.



Şekil 3.11 : Kimlik kartı üzerinde PIN ve PEN kodu doğrulama ekranı.

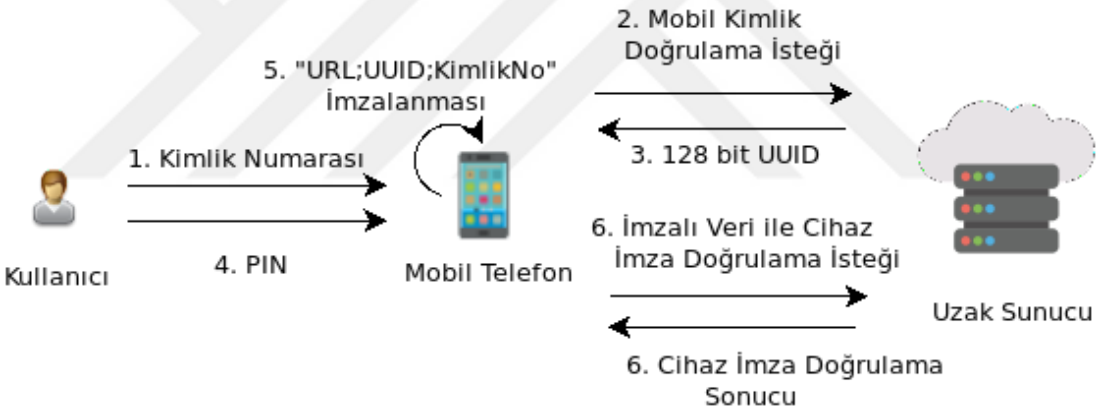
Son olarak kullanıcı, mobil cihazına ait uygun bir RSA açık anahtarının olması durumda masaüstü uygulamasında "Mobil Kimlik Oluştur" butonuna tıklayarak, bu açık anahtarın kart üzerinde imzalanması işlemine geçer. TCKK'nın dağıtımında ilgili kurum tarafından kimlik kartına ait PIN ve PEN numaraları da size verilir. Bu PIN ve PEN numarası ile kimlik kartınız üzerinde doğrulama yapabilir veya çalışmamızda olduğu gibi verileri imzalayabilirsiniz. Eğer kimlik kartlarımızda bir PIN ve PEN numaraları olmasaydı; fiziksel olarak kimlik kartınızı elinde bulunduran her üçüncü şahıs adımıza işlem yapabilecek duruma gelirdi. Bu gibi kötü koşulların önüne geçmek adına bulunan PIN ve PEN numarasının imzalama işlemi için kimlik kartı üzerinde doğrulanması gerekmektedir. Şekil 3.11'de gösterildiği üzere kimlik kartınıza ait PIN ve PEN numaraları girilmesi beklenir. "Doğrula" butonuna tıklanması ile, girilen numaralar kart üzerinde kontrol edilir. Başarılı bir doğrulama sonucunda, masaüstü uygulaması mobil cihaz üzerinden aldığı RSA açık anahtarı, kimlik numarası ve geçerlilik süresi ile "Sertifika Doğrulama İsteği (CSR)" oluşturur. Oluşturulan "Sertifika Doğrulama İsteği" kimlik kartımız üzerinde imzalanarak, mobil kimlik sertifikamız oluşmuş olur. Oluşturulan mobil kimlik USB aracılığı ile mobil cihaza gönderilerek, otomatik olarak güvenilir depoya kaydedilir.

3.5 Uzak Sunucu

Uzak sunucu, mobil cihazımızda oluşturulan ve cihazın Güvenilir İşletim Ortamı'nda saklanan mobil kimliğimiz ile kimlik doğrulamamızın yapılabildiği örnek bir web servisidir. Bu web servis, Java programlama dilinde, JSON format tabanlı, Apache CXF ve Spring Web Uygulama Çatısı teknolojisi kullanılarak geliştirilmiştir. Ayrıca,

geliştirilen web servisi ile kurulacak güvenli bağlantı için SSL/TLS protokolü kullanılmıştır.

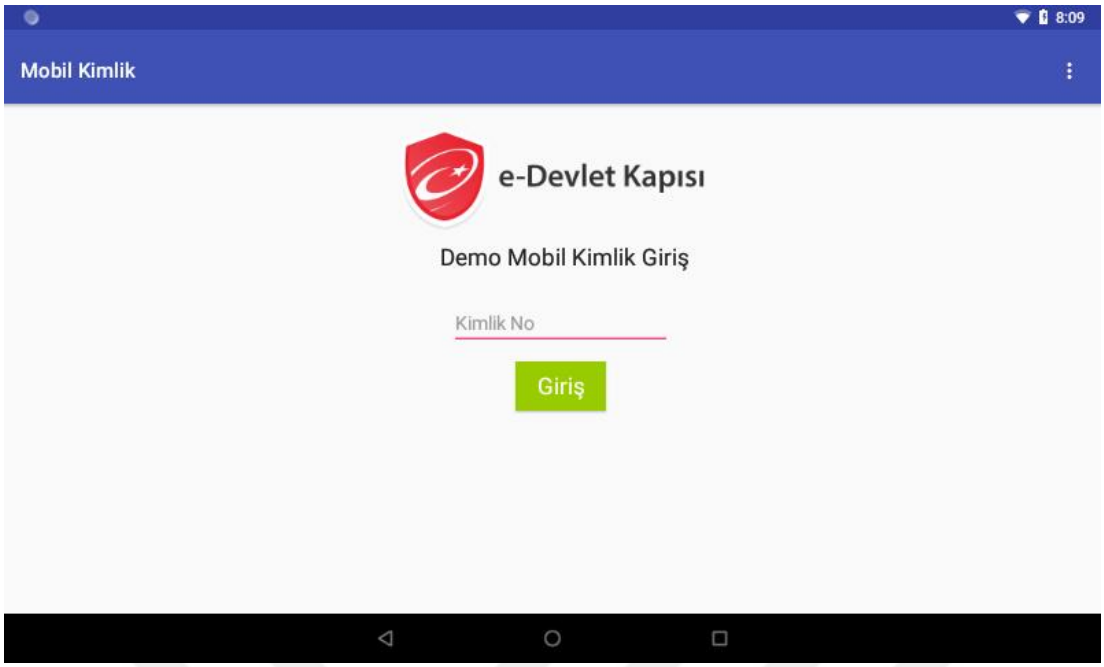
Uzak sunucu adını verdiğimiz web servisi, mobil kimlik çözümümüz üzerinden kimlik doğrulaması yapmak isteyen herhangi bir web uygulaması da olabilir. Mobil Kimlik çözümümüze entegre olmak isteyen bir web uygulamasının mobil kimliği oluşturan sertifikayı doğrulayabilmesi için akıllı kartın açık anahtarına sahip olması gerekmektedir. Akıllı karta ait açık anahtarın nasıl paylaşılacağı bu çalışmanın konusu olmadığı gibi, mobil kimlik çözümünü kullanacak son kullanıcı ile ilgili bir durum değildir. Örneğimizde kullandığımız açık anahtar altyapısına sahip Türkiye Cumhuriyeti Kimlik Kartı, var olan E-Devlet sisteminin bir parçasıdır. Yani kimlik kartımıza ait olan açık anahtar hali hazırda E-Devlet sistemi tarafından bilinmektedir. Bu nedenle uzak sunucuya akıllı kimlik kartının açık anahtarı önceden tanımlanarak mobil kimlik çözümümüz ile kimlik doğrulaması yapan bir E-Devlet sistemi modellenmiştir.



Şekil 3.12 : Mobil kimliğin uzak sunucuda doğrulanması.

E-Devlet sistemini modellediğimiz uzak sunucudaki web servisi güvenli bir mobil kimlik doğrulamasını şekil 3.12'deki gibi yapmaktadır [20]. Şekil 3.12'de verilen protokol adımları aşağıda detaylandırılmıştır.

1-) Mobil Kimlik Uygulamamıza, uzak sunucuda güvenli bir mobil kimlik doğrulama yapmayı örneklendirmek adına gerekli grafiksel kullanıcı ara yüzleri eklenmiştir. Şekil 13'de görüleceği üzere, "Demo Mobil Kimlik Giriş" adını verilen bir E-Devlet giriş sayfası örneklendirilmiştir. Kullanıcı ilgili alana Türkiye Cumhuriyeti Kimlik Numarası'nı girer ve "Giriş" butonuna tıklar.



Şekil 3.13 : Mobil kimlik ile demo E-Devlet giriş sayfası.

2-) Kullanıcının "Giriş" butonuna tıklaması ile Mobil Kimlik Uygulaması, girilen kimlik numarasının mobil cihazımızda ürettiğimiz mobil kimliğe ait olup olmadığını kontrol eder. Verilen bilgi doğru ise uygulama Güvenilir İşletim Ortamı'ndan alır ve uzak sunucu ile güvenli bağlantı üzerinden sertifika doğrulama istediğini gönderir. Sertifika doğrulama isteği için uzak sunucunun servis ara yüzü aşağıdaki gibidir:

URL: https://www.local.com/RemoteSystem/api/verify-certificate
HTTP İsteği Metodu: POST
HTTP İsteği İçeriği:
{
"cert" : "(Mobil Kimlik Sertifikası)",
"cardId" : "(Kimlik Numarası)"
}

Mobil Kimliğimizi oluşturan sertifika güvenilir depoda dosya olarak değil, bir byte dizisi olarak tutulur. Güvenilir depodan alınan sertifika kullanıcı uygulamamızda "String" türüne kodlanır. HTTP isteğinin içeriğinde bu değer, JSON formatında "cert" değişkenine atanır. Diğer değişken "cardId"ye kullanıcının girdiği kimlik numarası girilir. Uzak sunucu aldığı "String" türündeki sertifika bilgisini "X509Certificate" türüne çevirir. X509 sertifika dosyası Java programlama dilinde "X509Certificate" türü ile ifade edilir. Bu noktada uzak sunucu "cardId" değişkeni ile gelen kimlik numarasını sistemde var olup olmadığını, varsa gönderilen sertifika ile uyumlu olup olmadığını kontrol eder. Gerekli kontrollerin yapılmasından sonra uzak sunucuya

önceden tanımlanmış kimlik kartının açık anahtarı ile doğrulanır. Buna ek olarak, her kimlik doğrulama isteği uzak sunucuya geldiğinde, kullanılan açık anahtarda, kök kimlik doğrulama sertifikası ile doğrulanır. Kök kimlik doğrulama sertifikası, altbölüm 3.4'te bahsedilen "TCKK Uygulama Geliştirme Ara Yüzü" ile teslim alınmıştır.

3-) Tüm kontrollerin ve doğrulama işlemlerinin başarılı olması durumunda uzak sunucu cevap olarak aşağıda JSON formatı verilen 128 bit benzersiz sıradan bir numarayı cevap olarak gönderir.

{
"randomNumber":"5a447b08-6b06-4bfc-8774-7849a992bfb7"
}

Mobil kimliğin doğrulanamaması durumunda "randomNumber" değişkenine karşılık gelen 128 bitlik numara boş gelir. Böylelikle, kullanıcıya "Giriş yapılamadı" uyarısı gelecektir.

4-) Uzak sunucudan gelen benzersiz sıradan numaranın mobil cihazda belirlenen bazı değişkenler ile imzalanma işlemi bir PIN kodu ile korunmuştur. Kaynak [20]'e bakıldığında dijital imza oluşturmak için doğru PIN değerinin kullanıcı tarafından girilmesinin, güvenli bir mobil kimlik doğrulama için gerektiği görülür. Böylelikle kimlik doğrulama işlemi, üçüncü bir şahıs tarafından mobil cihazın fiziksel olarak ele geçirilmesi durumunda korunacaktır

Şekil 3.14 : Mobil kimliğin uzak sunucuda doğrulanmasından sonra uzak sunucuda cihaz doğrulaması işlemi için PIN ekranı.

Şekil 3.14'te görülen PIN ekranı, kullanıcı tarafı uygulama içerisinde geliştirdiğimiz bir ekrandır. PIN ekranı Güvenilir Kullanıcı Ara Yüzü teknolojisi kullanılarak geliştirilmesi Mobil Kimlik Uygulamamızı daha da güvenli hale getirecekti. Bu teknolojinin neden kullanılamadığı bölüm 4'te anlatılmıştır.

5-) Doğru PIN değerinin kullanıcı tarafından girilmesi ile uygulamamız üzerinden oluşturulacak dijital imza için URL, benzersiz sıradan numara ve kimlik numarası noktalı virgül ile ayrılmış bir desende mobil cihazımızda bulunan RSA anahtar çiftinin gizli anahtarı ile imzalanır. İmzalanan veri örneğimiz (Örneğimiz de gerçek bir kimlik kartı kullanıldığı için gerçek kimlik numarası yerine "11111111111" olarak yazılmıştır) için şu şekildedir:

https://www.local.com/RemoteServer/api;5a447b08-6b06-4bfc-8774-7849a992bfb7;11111111111

6-) Beşinci madde de imzalanan veri doğrulanmak üzere uzak sunucuya gönderilir. Cihaz imza doğrulama isteği için uzak sunucunun servis ara yüzü aşağıdaki gibidir:

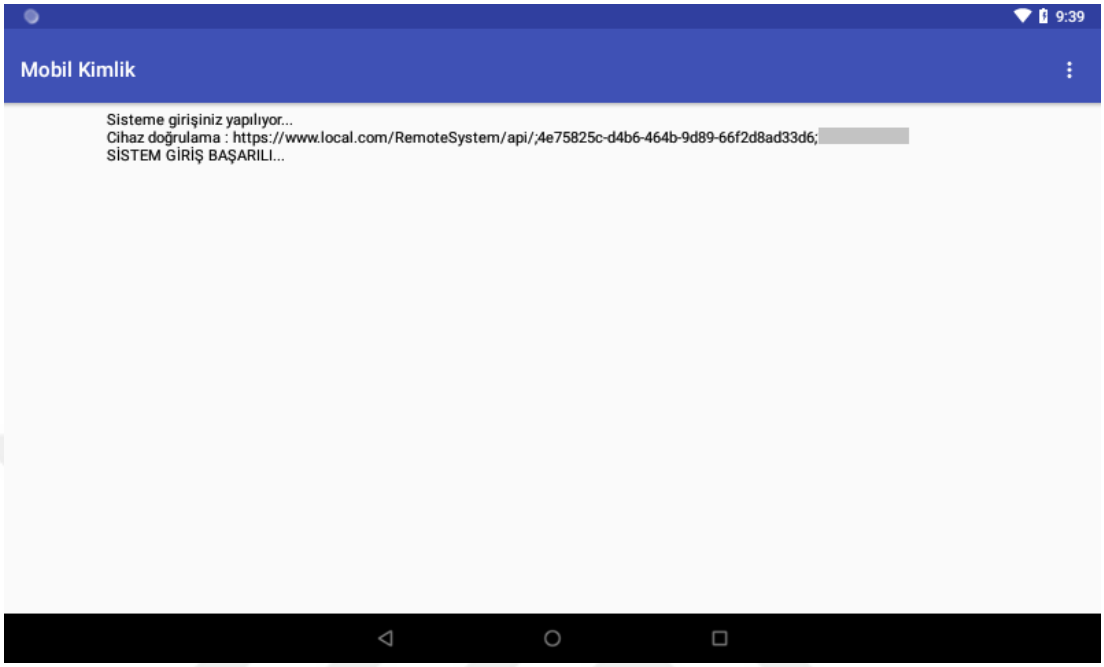
URL: https://www.local.com/RemoteSystem/api/verify-device-signature
HTTP İsteği Metodu: POST
HTTP İsteği İçeriği:
{
"signature" : "(İmza Verisi)",
"cardId" : "(Kimlik Numarası)"
}

Yukarıda ki servis ara yüzüne imzalı veri ve kimlik numarası gönderilir. İlk olarak, kimlik numarasının sistemde kayıtlı olup olmadığı, kayıtlı ise mobil kimlik sertifikasının olup olmadığı kontrol edilir. Başarılı bir kontrolün sonucunda, uzak sunucu doğrulama işlemi için kendi içerisinde "URL;UUID;KimlikNumarası" desenini ilgili veriler ile açık bir şekilde oluşturur. Mobil cihazdan gelen imzalı veri, mobil kimlik sertifikası doğrulama işleminde elde edilen sertifika ve uzak sunucuda oluşturulan açık desen ile doğrulama işlemine tabi tutulur. Başarılı bir cihaz doğrulaması sonucunda aşağıdaki JSON formatı örnek uzak sunucumuzun cevabı olacaktır.

{
"verified" : "1" //Başarısız bir doğrulamada değer 0 olacaktır.
}

Üretilen mobil kimlik ile uzak sunucuda kimlik doğrulamasının başarı durumu kullanıcıya bildirilir. Kimlik doğrulamanın başarısızlıkla sonuçlanması durumunda

kullanıcı ekranına bir bildiri düşer. Eğer işlem başarılı sonuçlanırsa, gerçek kullanımda kullanıcı E-Devlet sistemine giriş yapacaktır. Demo niteliğinde ki bu örneğimizde kullanıcının sisteme başarılı ile girdiğini gösteren şekil 3.15'te ki ekran gösterilir.



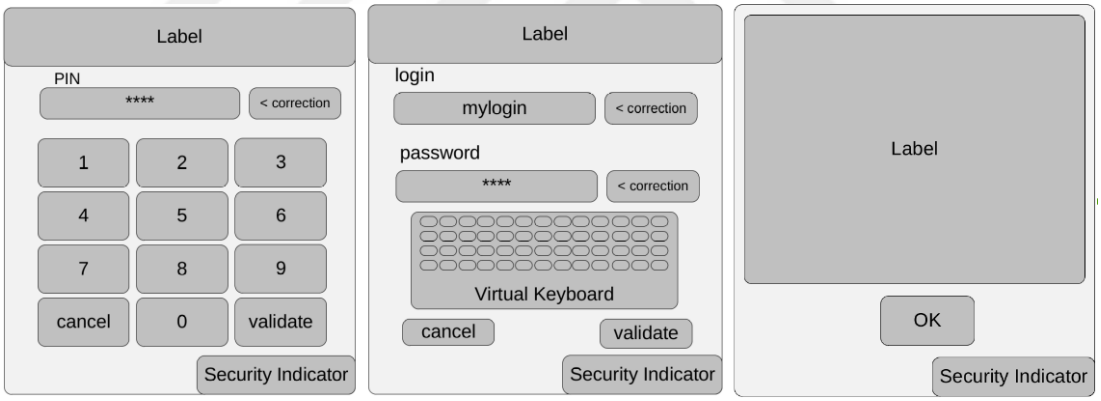
Şekil 3.15 : Demo giriş ekranı için kimlik doğrulamanın başarıyla gerçekleştirildiğini gösteren ekran

Sadece mobil kimliği oluşturulan sertifikanın doğrulanarak kimlik doğrulamasını bitiren bir uzak sunucu ortadaki adam adı verilen saldırıya açık konuma gelecektir. Saldırgan tarafından mobil kimlik çözümümüzü kullanan bir web uygulamasının sahte versiyonunu hazırladığını, bu web sitenin linkini de kurban olarak kullanıcıya yolladığını düşünelim. Saldırganın amacı linke tıklayan kullanıcıya sahte web sitesi üzerinden kimlik doğrulama işlemi yaptırıp mobil kimlik sertifikasını ele geçirmektir. Böylelikle tek bir doğrulama ile oturum açma izni verilen uzak sunucuda saldırgan istediği zaman sisteme giriş yapabilecektir. Bu tarz pasif oltama tekniği kullanan ortadaki adam saldırılarının önüne geçmek için Mobil Kimlik Uygulamamız uzak sunucuda "Cihaz İmza Doğrulaması" adını verdiğimiz ikinci bir doğrulama işlemini yapar. Tabii ki cihaz imza doğrulamasını yapacak olan uzak sunucudur. Uzak sunucu örneğimizde yukarda anlatıldığı gibi imza doğrulama işlemi için servis ara yüzü eklenmiştir. Sahte web uygulaması üzerinden Mobil Kimlik Uygulamamız cihaz imza doğrulaması yapacağında sahte web uygulamasının adresini, uygulamamıza gelen benzersiz sıradan numarayı ve kimlik numarasını "URL;UUID;KimlikNumarası" desenine getirip mobil cihazın gizli anahtarı ile imzalayacaktır ve imzalı veri sahte

web uygulamasına gönderilecektir. Artık saldırganın elinde mobil kimliğimiz ve imzalı veri bulunmaktadır. Pasif olarak sahte web uygulaması üzerinden bile gerçek web uygulamasında oturum açılmak istense bile doğrulama başarısızlıkla sonuçlanacaktır. Çünkü, imzalanan veri sahte web uygulamasının adresini içermekte ve gerçek web uygulamasında cihaz imza doğrulaması yapılacakken oluşturulan desende asıl adres bilgisi yer alacaktır. Bu durumda imza doğrulaması başarılı olamayacak ve oltalama saldırısı başarısızlıkla sonuçlanacaktır [20].

4. SONUÇ VE ÖNERİLER

Çalışmamızda mevcut eksikliklerden birisi Güvenilir Kullanıcı Ara Yüzü teknolojisidir. Güvenilir Kullanıcı Ara Yüzü, Güvenilir İşletim Ortamı teknolojilerinin güvenliğinin bir adım daha öteye geçiren entegre bir platformdur. Bu platformun gereksinim ve standartları GlobalPlatform organizasyonu tarafından belirlenmiştir. Güvenilir Kullanıcı Ara Yüzü sayesinde güvenilir uygulamaları kullanırken girdi değerlerinin güvenliğini daha da artırabiliriz. Güvenilir Kullanıcı Ara Yüzü olan bir Güvenilir İşletim Ortamı teknolojisinde açılan grafiksel kullanıcı ara yüzü Zengin İşletim Sistemi'ni durdurur ve cihaz üzerinde girdileriniz için kullandığınız dokunmatik ekran, tuş takımı gibi araçlar ile güvenilir taraf arasında bir güvenli yol oluşturur. Böylelikle üçüncü şahıs uygulamalar tarafından hassas verileriniz dinlenemez [21].



Şekil 4.1 : Güvenilir Kullanıcı Ara Yüzü Şartnamesi'nde tanımlanan bazı ekran görüntüleri.

Şeki 4.1'de görülen ve Güvenilir Kullanıcı Ara Yüzü Şartnamesi'nden alınan ekran görüntüleri Güvenilir İşletim Ortamı'nın kullanıcı ara yüzünde nasıl olması gerektiğini göstermektedir. Mobil Kimlik Uygulamamız detayları altbölüm 3.3'de verilen *TEEC_LOGIN_PUBLIC* ile güvenilir uygulama üzerinde oturum açma işleminde herhangi bir PIN koruması olmadan oturumu açmaktadır. Kullandığımız donanımlar ve OP-TEE üzerinde daha geliştirilmesi ve ana çekirdeğe entegrasyonu olmadığından "Güvenilir İşletim Ortamı Kullanıcı Ara Yüzü" kullanılamamaktadır. Kullanılamamasına ilişkin detaylı bilgi ilerleyen paragraflarda verilmiştir. Bu

konudaki modelleme, uzak sunucuda kimlik doğrulama aşamasında "Cihaz İmza Doğrulama" öncesinde yapılmıştır. Şekil 3.14'te gösterildiği üzere, Güvenilir İşletim Ortamı Kullanıcı Ara Yüzü'nün PIN giriş ekranı modellendiğini görebilirsiniz. Tabi ki, bu işlemin tüm kaynakları Android İşletim Sistemi kullanılarak geliştirildi. Kullanılan PIN bilgisi uygulamada saklanmaktadır. Asıl güvenilir ara yüzünün kullanılması durumunda, işlem tüm Android İşletim Sistemi işlemlerini durdurur ve güvenilir ara yüz ekranı gösterilir. Güvenilir Kullanıcı Ara Yüzü klavye, dokunmatik ekran gibi girdi mekanizmalarını Zengin İşletim Ortamı'ndan gelecek saldırılara karşı koruduğu gibi Zengin İşletim Sistemi güvenilir ara yüz kaynaklarına ulaşamaz. Bu durumun garantisini veren güvenilir ara yüz sayesinde, tüm girdi-çıktı mekanizmaları Güvenilir İşletim Ortamı tarafından kontrol edilir. Güvenilir Kullanıcı Ara Yüzü'nün bu çalışmaya eklenmesi, güvenilir dünyayı daha da korunaklı hale getirecektir. Ayrıca Güvenilir Kullanıcı Ara Yüzü sayesinde fiziksel olarak mobil cihazın yabancı bir şahıs tarafından ele geçirilmesi durumunda mobil kimliğin yetkisiz kullanımı da engellenecektir.

OP-TEE'de Güvenilir Kullanıcı Ara Yüzü teknolojisi bulunmamasından ötürü prototip olarak bir Güvenilir Kullanıcı Ara Yüzü geliştirilmek istenmiştir. Fakat, HiKey geliştirme kartının ana işlemcisi "TrustZone Address Space Controller (TZASC)" adı verilen teknolojiye sahip değildir. Bu yapıya sahip olmayan bir donanımda, Zengin İşletim Sistemi'nin işlemlerini bekleterek, araya Güvenilir Kullanıcı Ara Yüzü araçlarını alamamaktayız. Bu sebeple, TZASC yapısına sahip olmayan bir donanımda, güvenilir taraf uygulaması kullanılıyorken kullanıcı-cihaz etkileşim araçlarını (tuş takımı, dokunmatik ekran vb.) Zengin İşletim Sistemi'nden gelecek saldırılara karşı koruyamayacağımız anlamına gelmektedir [17]. Öte yandan, HiKey geliştirme kartında, TZASC yapısına benzer bir adres alan denetleyicisi bulunmaktadır. Bu yapı kullanarak da bit Güvenilir Kullanıcı Ara Yüzü geliştirilebilir. Fakat, bu yapının gerekli dokümantasyonu ARM tarafından geliştiricilere açılmamıştır. Bu sebeplerle, Güvenilir Kullanıcı Ara Yüzü konusu gelecek çalışmalar kapsamına alınmıştır.

Mobil Kimlik Uygulamamızda değinilmesi gereken bir diğer konu ise mobil cihazımız üzerinde oluşturduğumuz açık anahtarın USB kablo yardımı ile masaüstü uygulaması ile paylaşılmasıdır. Masaüstü uygulamamızda oluşturulan mobil kimlik USB kablo yardımı ile mobil cihaza aktarılmaktadır. USB portunu dinleyen bir zararlı yazılım bu

haberleşme esnasında mobil kimliğimizi ele geçirebilmesi mümkündür. USB kablo yardımı ile mobil cihaz ve bilgisayar arasındaki kurulan bağlantındaki haberleşmenin korunması için alınması muhtemel önlemler ile masaüstü uygulaması daha korunaklı hale getirilebilir. Öte yandan, mobil kimlik hırsızlığı amacıyla oluşturulan sahte masaüstü uygulamaları ile Mobil Kimlik Uygulamamızın iletişime geçmemesi için muhtemel kontrol mekanizmaları eklenerek mobil kimliğimizin üçüncü şahıslar tarafından ele geçirilmesi engellenebilir.

Çalışmamız mantığıyla oluşturulacak mobil kimlik uygulamasında, NFC teknolojisi kullanılarak tek seferlik bilgisayar kullanımı da ortadan kaldırılabilir. Artık mobil cihazlarımızın bir çoğunda NFC teknolojisi içeren donanımlar mevcuttur. NFC, yakın alana getirilen iki uygun çipin herhangi bir fiziksel bağlantı olmadan, manyetik alan indüksiyonunu kullanarak haberleşmesine verilen teknolojinin adıdır. NFC teknolojisinin uygulamaları özellikle bankacılık sektöründe kredi kartlarımız aracılığıyla hayatımıza girmiş bulunmaktadır. Çalışmamızda akıllı kart olarak kullandığımız açık anahtar tabanlı kimlik kartlarımızda NFC haberleşmeyi sağlayacak herhangi bir modüle sahip değildir. Bu nedenle, mobil kimlik uygulamamızda bu özellik kullanılamamıştır. Fakat, ileri çalışmalarda bu özelliğin kullanılabilmesi ile son kullanıcıya daha kullanışlı bir ürün sunulabileceğini düşünmekteyiz. Herhangi bir kart okuyucusunun muhtemel komplikasyonlarına maruz kalmadan da son kullanıcı akıllı kartını mobil cihazıyla kullanabilecektir.

Masaüstü uygulamasının kullanımdan kaldırma adına, çalışmamızda şu fikir düşünülmüştür: Standart kart okuyucusunun direk mobil cihaza bağlanabilmesi. Çalışmamız kapsamında kullandığımız HiKey geliştirme kartına kurulu Android Nougat İşletim Sistemi kart okuyucusunu tanımıştır. Fakat, TUBITAK E-Kimlik biriminin bize sunduğu TCKK Uygulama Geliştirme Ara Yüzü'nün bazı komplike fonksiyonları Android İşletim Sistemi'nde bulunmayan bazı yazılım kütüphaneleri nedeniyle akıllı kimlik kartımız Android İşletim Sistemi ile okunamamıştır. Akıllı kimlik kartının Android üzerinden herhangi bir kart okuyucu ile okunabilmesi bu çalışmanın kapsamında değildir. Ayrıca, kimlik kartının çipi açık olmayan bazı güvenlik özellikleri ve akıllı karta özel tasarımlar içerdiğinden, farklı bir ara yüz kütüphanesi geliştirmek zor olacaktır. Android İşletim Sistemi'ne uyumlu halde çalışacak akıllı kimlik kartlarının uygulama geliştirme ara yüzleri, akıllı kimlik kartları üzerinden oluşturulacak mobil kimliklerde bilgisayar kullanımını sıfıra indirecektir.

Mobil Kimlik Uygulamamız haricinde, güvenilir uygulama geliştirme hakkında bahsedilmesi gereken bazı konular vardır. Zaafiyetlerin ortaya çıkma olasılığını azaltmak için güvenilir uygulamalar mümkün olduğunca küçük tutulmalıdır. Normal dünyadaki uygulamalar son kullanıcılar için çeşitli özellikler kullanmak adına daha karmaşık işlemler yapabilir. Güvenli dünyada daha karmaşık işlemler yapabiliriz, ancak normal ve güvenli dünya yaklaşımının amacı, güvenilir uygulamaları küçük, basit ve güvenli tutmaktır. Güvenilir İşletim Ortamı teknolojisi kullanılarak geliştirilen bir uygulamada yapılacak iş veya işlemlerden hangisinin güvenli dünyada yapılacak kadar önemli olduğuna karar verilmelidir. Bunun sonucunda da diğer işe veya işlemler kullanıcı taraf uygulamaya bırakılmalıdır. Çalışmamızda görüldüğü üzere, güvenilir uygulama RSA anahtar çiftimizi ve mobil kimliğimizi yöneten basit fonksiyonlardan oluşmaktadır. Tüm kontroller güvenilir işletim ortamına alınmayıp, kullanıcı taraf uygulamada gerçekleştirilerek, güvenilir uygulama basit ve sade tutulmuştur. Ayrıca, geçerlilik süresi, oluşturma tarihi veya kimlik numarası gibi bilgiler güvenilir depoda tutulmamıştır. Güvenilir uygulama geliştirme yaklaşımında güvenlik açısından bu dengenin dikkatli oluşturulması gereklidir.

Günümüzde Güvenilir İşletim Ortamı'nın mobil işletim sistemleri ile kullandığı cihazlarda, güvenilir tarafta işlemler arka planda gerçekleşir. Mobil cihaz üreticileri yazılımlarının alt yapısında güvenli dünya olanaklarını kullanır. Fakat, son kullanıcı bu durumu farketmez. Ayrıca, güvenli dünyada bir uygulama geliştirmek, mobil cihazın üretimi sonrasında uygulama geliştiricilere açık bir durum değildir. Nitekim, güvenilir uygulamalar Android İşletim Sistemi'nde "system" klasörü altında son kullanıcıya açık olmayan bir alanda çalışır. Güvenilir uygulamaya erişmek isteyen bir kullanıcı tarafı Android uygulaması da "system" klasörü altında çalışması gerekmektedir. "system" klasörü altında çalışan uygulamalar genelde mobil cihaz üreticileri tarafından eklenen uygulamaları içerir. Çalışmamızda kullandığımız Android, herhangi bir mobil cihaz üreticisinin Android'i değil, Android Açık Kaynak Projesi'ne ait işletim sistemidir. Dolayısıyla, "system" klasörüne erişebilmekte ve kendi güvenilir uygulamamızı sisteme entegre edebilmekteyiz. Şuan için kendi telefonlarımızda kullandığımız Güvenilir İşletim Ortamı son kullanıcı ortamına indirgenmiş halde değildir. Bunun ile ilgili, güvenilir uygulamanın geliştirici tarafından nasıl eklenceği, güncelleneceği gibi standartlar oluşturulmuş durumda değildir. Belki de, bu teknoloji son kullanıcıya indirgenmeyerek mobil cihaz üreticisi

tarafında eklenebilen uygulamalar olarak kalacak. Fakat, bu konuda oluşturulacak yeni fikir ve standartlaştırmalar daha fazla mobil uygulamanın güvenilir dünyaya entegrasyonunu sağlayacağını düşünmekteyiz.

Bu çalışmamızda, açık anahtar altyapısına sahip yeni çipli kimlik kartlarımızı elektronik doğrulama amacıyla mobil ekosistemine kazandırmak amacıyla Güvenilir İşletim Ortamı kullanarak geliştirdiğimiz bir mobil kimlik uygulaması tanıtılmıştır. Güvenilir İşletim Ortamı Teknolojisi sadece mobil işletim sistemi Android ile kullanılmamaktadır. Birçok uygulamanın güvenliğini artıracak bu teknolojinin günlük cihazlarımızda kullanımının yaygınlaşması ile beraber hassas verilerimizi içeren uygulamaların, alt yapılarını bu teknolojiye uygun hale getirmeleri gerekeceği düşüncesindeyiz.

Güvenilir İşletim Ortamı kullanılarak geliştirdiğimiz mobil kimlik uygulamasının, güvenli ve özgün bir prototip olarak mobil kimlik çözümleri arasında önemli bir konuma sahip olabileceği değerlendirilmektedir. Akıllı kimlik kartlarının mobil ekosisteme entegre edilmesi ile daha güvenli ve hızlı kimlik doğrulamalar yapılabilecektir. SIM kartlar üzerinden kimlik doğrulamada ki üçüncü taraf şahıs veya şirketler olmasına gerek kalmadan, kullanıcının güvenliği daha kolay bir şekilde sağlanacaktır. Bankacılıktan e-devlet sistemlerine kadar bir çok işlemlerimizde kullanabileceğimiz Güvenilir İşletim Ortamı teknolojisi kullanarak geliştirilen Mobil Kimlik Uygulamamızın var olan kimlik doğrulama çözümlerine göre kullanıcı güvenliğini daha da artıracağını değerlendirmekteyiz.

KAYNAKLAR

[1] https://www.idc.com/promo/smartphone-market-share/os Alındığı Tarih: 01/12/2018.

[2] **Sağıroğlu, Ş., Kabasakal, D. ve Alkan, M.,** (2008) *Mobil Elektronik İmza, Altyapısı ve Türkiye,* Gazi Üniv. Müh. Mim. Fak. Der. Cilt 23, No 1, Sf. 49-56.

[3] OP-TEE Güvenilir İşletim Sistemi Github.com Sayfası. Erişim Adresi: https://github.com/OP-TEE/optee_os

[4] **Ekberg, J., Kostiainen, K., Asokan N.,** (2013) Trusted Execution Environments on Mobile Devices, *CS '13 Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, Berlin, Germany, November 04 - 08.

[5] **Ferraiolo, H., Cooper, D., Francomacaro, S., Regenscheid, A., Mohler, J., Gupta, S., Burr, W.,** (2014) *Guidelines for Derived Personal Identity Verification (PIV) Credentials*, National Institute of Standards and Technology Special Publication 800-157, December.

[6] GlobalPlatform Inc. (2015), White Paper, Mobile ID: Realization of Mobile Identity Solutions by GlobalPlatform Technologies, November.

[7] A Secure Techonology Alliance Mobile Council (2017), White Paper, Mobile Identity Authentication, Version 1, March.

[8] Android Open Source Project, Erişim Adresi : https://source.android.com

[9] GlobalPlatfrom Inc., TEE Specification Library, Erişim Adresi : https://globalplatform.org/specs-library/?filter-committee=tee

[10] **Sabt, M., Achemlal, M., Bouabdallah, A.,** (2015), *Trusted Execution Environment: What It Is, and What It Is Not*, IEEE Trustcom/BigDataSE/ISPA, Sf. 57-65, IEEE 978-1-4673-7952-6/15 $31.00.

[11] **Thanh, D.V., Jønvik T., Jørstad, I., Thuan, D.V.,** (2013) *Better user protection with mobile identity*, IEEE 978-1-4799-2845-3/13/$31.00.

[12] **Umar, A., Akram, R.N., Mayes, K., Markantonakis, K.** (2017) Ecosystems of Trusted Execution Environment on Smartphones - A Potentially Bumpy Road, *Third International Conference on Mobile and Secure Services (MobiSecServ)*, Miami, USA, February 11-12.

[13] ARM TrustZone Technology, Erişim Adresi : https://developer.arm.com/technologies/trustzone

[14] Intel Co., Intel Trusted Execution, White Paper, Erişim Adresi : https://www.intel.com/content/www/us/en/architecture-and-technology/trusted-execution-technology/trusted-execution-technology-security-paper.html

[15] HiKey Geliştirme Kartı, Erişim Adresi : https://www.96boards.org/product/hikey/

[16] OP-TEE Project, Android manifest for building OP-TEE in AOSP, Erişim Adresi : https://github.com/linaroswg/optee_android_manifest

[17] GlobalPlatform Inc., (2018) TEE Internal Core API Specification v1.1.2.50 (Target v1.2), June.

[18] GlobalPlatform Inc., (2010) TEE Client API Specification v.1.0, July.

[19] Android Native Development Kit. Erişim Adresi : https://developer.android.com/ndk/

[20] Bıçakcı, K., Ünal, D., Aşçıoğlu, N., Adalier, O., (2014) Mobile Authentication Secure Against Man-In-The-Middle Attacks, *Second IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, IEEE 978-1-4799-4425-5/14 $31.0.

[21] GlobalPlatform Inc., (2013) Trusted User Interface API Specification v1.0, June.

ÖZGEÇMİŞ

Ad-Soyad : Kaan KÜÇÜK

Uyruğu : T.C.

Doğum Tarihi ve Yeri : 26/03/1988 Ankara

E-posta : kaan.kucuk@yahoo.com

ÖĞRENİM DURUMU:

- **Lisans** : 2011, Kadir Has Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği
- **Yükseklisans** : 2019, TOBB Ekonomi ve Teknoloji Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği

MESLEKİ DENEYİM VE ÖDÜLLER:

Yıl	Yer	Görev
2013 - 2014	Strategic Imperatives, Londra / İngiltere	Yazılım Mühendisi
2011 - 2013	Huawei Telekom, İstanbul / Türkiye	Yrd. Yazılım Mühendisi
2010 - 2011	Turkcell Teknoloji, İstanbul / Türkiye	Java Geliştiricisi

YABANCI DİL: İngilizce

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- Bıçakcı, K., **Küçük, K**., 2019. Güvenilir İşletim Ortamı Teknolojisi Kullanılarak Mobil Kimlik Uygulaması Geliştirilmesi, 21. Akademik Bilişim Konferansı, Şubat 13-15, Ordu, Türkiye.