TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BLOKZİNCİR KULLANILARAK KİMLİK DOĞRULAMA SEREMONİSİNİ ORTADAN KALDIRAN BİR GÜVENLİ MESAJLAŞMA UYGULAMASININ GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Enes ALTUNCU

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. Kemal BIÇAKCI

KASIM 2019

Fen Bilimleri Enstitüsü Onayı

.....
Prof.Dr. Osman EROĞUL
Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

.....
Prof.Dr. Oğuz ERGİN
Anabilimdalı Başkanı

TOBB ETÜ, Fen Bilimleri Enstitüsü'nün 171111019 numaralı Yüksek Lisans öğrencisi **Enes ALTUNCU** 'nın ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"BLOKZİNCİR KULLANILARAK KİMLİK DOĞRULAMA SEREMONİSİNİ ORTADAN KALDIRAN BİR GÜVENLİ MESAJLAŞMA UYGULAMASININ GELİŞTİRİLMESİ"** başlıklı tezi **06.11.2019** tarihinde aşağıda imzaları olan jüri tarafından kabul edilmiştir.

Tez Danışmanı:	**Prof.Dr. Kemal BIÇAKCI** TOBB Ekonomi ve Teknoloji Üniversitesi	.....
Jüri Üyeleri:	**Prof.Dr. Ali Aydın SELÇUK (Başkan)** TOBB Ekonomi ve Teknoloji Üniversitesi	.....
	Dr.Öğr.Üyesi Hakan Ezgi KIZILÖZ Türk Hava Kurumu 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.

Enes ALTUNCU

ÖZET

Yüksek Lisans Tezi

BLOKZİNCİR KULLANILARAK KİMLİK DOĞRULAMA SEREMONİSİNİ ORTADAN KALDIRAN BİR GÜVENLİ MESAJLAŞMA UYGULAMASININ GELİŞTİRİLMESİ

Enes ALTUNCU

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

Tez Danışmanı: Prof.Dr. Kemal BIÇAKCI

Tarih: Kasım 2019

Uçtan uca şifreleme, genelde açık anahtar kriptografisine dayanan ve birçok güvenli mesajlaşma uygulamasında kullanılan bir güvenlik özelliğidir. Bu sayede, iletilen mesajlar, tarafların katkısıyla oluşturulan ortak bir anahtar vasıtasıyla şifrelenerek iletilir. Bu özelliğe sahip bir uygulamada, bir kullanıcının açık anahtarı, cihaz veya SIM kart değişikliği ve uygulamanın yeniden kurulması gibi sebeplerle değişebileceği gibi, olası bir saldırı nedeniyle de farklılık gösterebilir. Bu yüzden, muhtemel saldırıları önlemek ve iletişimin güvenli olduğundan emin olmak adına yeni açık anahtarın doğrulanması gerekir. Bu amaçla, birçok güvenli mesajlaşma uygulamasında, açık anahtar değişikliği durumunda "kimlik doğrulama seremonisi" adı verilen ve açık anahtar doğrulamasının kullanıcılar tarafından farklı bir kanal vasıtasıyla karşılaştırılarak yapıldığı bir mekanizma bulunur. Ancak, tüm kullanıcıların kimlik doğrulama seremonisini doğru bir şekilde tamamlamasını beklemek iyi bir fikir olarak görünmemektedir. Bu nedenle, bu tezde, blokzincir tabanlı bir açık anahtar altyapısı (PKI) sistemi olan Blockstack platformunu kullanarak "BlockSignal" adı verilen Signal Android Messenger tabanlı açık kaynaklı bir güvenli mesajlaşma uygulaması geliştirildi. Böylece, Signal uygulamasında da bulunan kimlik doğrulama seremonisinin blokzincir vasıtasıyla ortadan kaldırılarak kullanıcıların güvenlik çemberinin dışına çıkarılması sağlandı. Bunun yanısıra, kimlik doğrulama problemi haricinde Signal uygulaması üzerinde literatürde mevcut tehdit olasılıklarının bir araya getirildiği bir tehdit modeli oluşturulmuş, BlockSignal

uygulamasının güvenlik analizi yapılarak Signal'da karşılaşılabilecek tehditleri bertaraf edilebilecek kabiliyette olduğu gösterilmiştir.

Anahtar Kelimeler: Uçtan uca şifreleme, Güvenli mesajlaşma uygulamaları, Kimlik doğrulama seremonisi.

ABSTRACT

Master of Science

DEVELOPING A SECURE MESSAGING APPLICATION THAT ELIMINATES AUTHENTICATION CEREMONY BY USING BLOCKCHAIN

Enes ALTUNCU

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

Supervisor: Prof.Dr. Kemal BIÇAKCI

Date: November 2019

End-to-end encryption is a security feature that is often based on public key cryptography and is used in many secure messaging applications. Thus, the transmitted messages are transmitted as encrypted with a shared key generated by the contribution of the parties. In an application with this feature, a user's public key may be changed for reasons such as device or SIM card change and reinstallation of the application, or may also be different due to a possible attack. Therefore, the new public key needs to be verified to prevent possible attacks and to ensure that communication is secure. For this purpose, many secure messaging applications have a mechanism called "authentication ceremony" in the case of public key changes, where public key verification is performed by users by comparing the keys from a different channel. However, it does not seem to be a good idea to expect all users to complete the authentication ceremony correctly. Therefore, in this thesis, an open source secure messaging application based on Signal Android Messenger called "BlockSignal" was developed using Blockstack platform, a blockchain based public key infrastructure (PKI) system. Thus, the authentication ceremony in Signal application was eliminated by the blockchain and the users were taken out of the security loop. In addition to this, a threat model was formed on the Signal application, combining the possible threats which exist in the literature apart from the authentication ceremony problem, and the security analysis of the BlockSignal application was shown to be able to show that it can eliminate the threats that may be encountered in Signal.

Keywords: End-to-end encryption, Secure messaging applications, Authentication ceremony.

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Prof.Dr. Kemal BIÇAKCI, kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine, değerli katkılarından ötürü Mehmet Eren ALTUNCU ve Furkan ATAŞ'a ve destekleriyle her zaman yanımda olan aileme ve arkadaşlarıma çok teşekkür ederim.

İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa

KISALTMALAR

MITM : Ortadaki adam saldırıları (Man-in-the-Middle)
PKI : Açık anahtar altyapısı (Public Key Infrastructure)
TTP : Güvenilir üçüncü taraf (Trusted Third Party)
CA : Sertifika otoritesi (Certificate Authority)
CRL : Sertifika iptal listesi (Certificate Revocation List)
PoS : Hisse ispatı (Proof of Stake)
dApp : Merkezi olmayan uygulama (Decentralized Application)
SMS : Kısa mesaj servisi (Short Message Service)
DNS : Alan adı sistemi (Domain Name System)
JWT : JSON web belirteci (JSON Web Token)
SDK : Yazılım geliştirme kiti (Software Development Kit)
PGP : Oldukça iyi mahremiyet (Pretty Good Privacy)
SSL : Güvenli veri alışveriş katmanı (Secure Socket Layer)
DUKPT : İşlem başına elde edilen benzersiz anahtar (Derived Unique Key Per Transaction)
IM : Anlık mesajlaşma (Instant Messaging)
QR : Çabuk tepki (Quick Response)
GCM : Google bulut mesajlaşması (Google Cloud Messaging)
API : Uygulama programlama arayüzü (Application Programming Interface)
AWS : Amazon ağ servisleri (Amazon Web Services)
Redis : Uzak sözlük sunucusu (Remote Dictionary Server)
X3DH : Genişletilmiş üçlü Diffie-Hellman (Extended Triple Diffie-Hellman)
KDC : Anahtar dağıtım merkezi (Key Distribution Center)
BNS : Blockstack adlandırma servisi (Blockstack Naming Service)
LDAP : Basit indeks erişim protokolü (Lightweight Directory Access Protocol)
URI : Tekdüze kaynak belirteci (Uniform Resource Identifier)
SRK : Gizli Kurtarma Anahtarı (Secret Recovery Key)
MRC : Sihirli Kurtarma Kodu (Magic Recovery Code)

SEMBOL LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
g	Diffie-Hellman anahtar değişim üreteci (Generator)
p	Seçilen büyük bir asal sayı
a	Seçilen bir tamsayı
b	Seçilen bir tamsayı

1. GİRİŞ

Anlık mesajlaşma uygulamaları, günümüzde insanlar için vazgeçilmez hale gelmiştir. Bu tür uygulamalar birden fazla kullanıcı arasında bir bağlantı kurmayı amaçladıklarından ve istemci-sunucu mimarisini benimsediklerinden, özellikle "Ortadaki Adam (MITM)" saldırıları gibi iletişim güvenliğini tehdit eden olası saldırılara açık olabilirler. Ayrıca, bir uygulama için bir istemci-sunucu mimarisine sahip olmak, sunucunun kötü niyetli olması durumunda (örneğin, kötü niyetli bir firma çalışanı gibi) güvenliği tehdit edebilecek tek hata noktasına (single point of failure) sahip olduğu anlamına gelir.

MITM gibi saldırılardan ve istemci-sunucu modelinden kaynaklanan olası tehditlerden kaçınmak ve haberleşmenin güvenli olmasını sağlamak için, birçok anlık mesajlaşma uygulaması tarafından uçtan uca şifreleme (end-to-end encryption) tekniği kullanılmaktadır. Bu teknik, kriptografinin siber güvenliğin en güçlü bileşeni olduğu gerçeğinden hareketle açık anahtarlı şifrelemeye dayanır. Uçtan uca şifreleme yönteminde, tüm mesajlar gönderici tarafından alıcının açık anahtarı ile şifrelenir Böylece iletişimi dinleyen üçüncü taraflar gönderici ile alıcı arasında gidip gelen mesajları çözemez. Ancak, bir kullanıcının açık anahtarı bir şekilde değişirse iletişimi güvende tutmak için yeni açık anahtarın doğrulanması gerekir. Açık anahtar değişiminin nedeni, cihaz değişikliği ve uygulamayı silip yeniden yükleme gibi olağan bir durum olabileceği gibi istenmeyen bir durum, yani bir saldırı da olabilir. İlk olasılıkta, iletişim olduğu gibi devam etmeliyken diğer durumda oturum derhal sonlandırılmalıdır. Bu nedenle, açık anahtar doğrulaması yapılarak değişikliğin sebebinin olası bir saldırı olup olmadığı tespit edilmeli ve bu duruma göre aksiyon alınmalıdır. Ancak, sunucunun kötü niyetli olması nedeniyle oluşabilecek muhtemel bir saldırı neticesinde açık anahtar değişikliği gerçekleşebileceğinden, sunucu bu doğrulama işleminin bir parçası olmamalıdır. Sonuç olarak, birçok güvenli anlık mesajlaşma uygulaması, bu konuyu kullanıcılar tarafından gerçekleştirilmesi gereken bir mekanizma olan ve ilk olarak Brainard ve arkadaşları tarafından "kimlik doğrulama seremonisi (authentication ceremony)" olarak adlandırılan yöntem ile çözmeyi amaçlamaktadır.[28] Bu yöntem, aynı zamanda, Carl Ellison tarafından ortaya konulan ve protokol dışındaki her işlemi tanımlayan "güvenlik seremonisi" kavramının özelleşmiş bir türüdür.[33]

Mevcut açık anahtar doğrulaması yöntemleri incelendiğinde, parmak izi (fingerprint) ve emniyet numarası (safety number) metotları ön plana çıkmaktadır. Örneğin, OpenSSH [9], kurulum veya kayıt başına yalnızca bir tarafın açık anahtarından üretilen bir sayı olan ve "İlk Kullanımda Güven (Trust-on-First-Use)" ilkesini benimseyen parmak izi kullanır. Diğer taraftan, Signal ise, konuşma başına her iki tarafın açık anahtarlarından üretilen bir sayı olan emniyet numarasını kullanır. Bu teknikler oldukça yaygın olmasına rağmen, her iki teknik de o anda gerçek bir kullanıcıyla mesajlaşıldığından emin

olunabilmesi için kullanıcılar tarafından iki uzun sayının karşılaştırılmasını gerektirir. QR kodu bu karşılaştırmayı kolaylaştırmak için kullanılsa da, teknoloji aşinalığı yeterli seviyede olmayan kullanıcılar için kimlik doğrulama seremonisini doğru bir şekilde tamamlamak (hatta bu mekanizmanın ne anlama geldiğini anlamak bile [37]) hala zordur . Bu durum, siber güvenlikte insanların güvenlik çemberi dışında bırakılması (human-out-of-the-loop) ilkesiyle çelişki oluşturmaktadır [30]. Başka bir deyişle, doğrulama için bu tür tekniklerin bu haliyle kullanılması, güvenlik zincirine nihayetinde en zayıf halka olacak yeni bir halka eklemekten başka bir şey değildir.

Güvenlik zincirindeki en zayıf halka olan kullanıcıların, birçok çalışmada [26] [52] [41] kimlik doğrulama seremonisini düzgün bir şekilde tamamlama kabiliyetleri açısından zayıf oldukları gösterildiğinden, bu tezde, kimlik doğrulama seremonisi için kullanılabilir güvenlik anlamında daha iyi bir çözüm önerilmiş ve örnek bir prototip olarak Signal Android Messenger üzerinde gerçeklenerek açık kaynaklı olarak paylaşılmıştır[2].

Tezde önerilen ve ilerideki bölümlerde detaylı olarak bahsedilecek çözüm kapsamında, Signal anlık mesajlaşma uygulaması, blockzincir tabanlı ve merkezi olmayan bir internet platformu olan Blockstack ile entegre edilerek kimlik doğrulama seremonisi mekanizması otomatikleştirilmiş, böylece kimlik doğrulama seremonisinin, kullanıcılara kimliklerini nasıl doğrulamaları gerektiğini öğretmek zorunda kalmadan tamamlanabilmesi hedeflenmiştir. Dolayısıyla, eğer bir tarafın açık anahtarı bir şekilde değişirse, açık anahtar doğrulamasının diğer kullanıcı tarafında otomatik olarak gerçekleşmesi ve doğrulamanın gerçekleştirilememesi halinde oturumun sonlandırılarak haberleşme güvenliğinin korunması amaçlanmıştır.

1.1 Literatür Araştırması

Son zamanlarda, yaygın güvenli mesajlaşma uygulamalarında kimlik doğrulama seremonisi için daha iyi bir çözüm elde etmeye yönelik bazı çalışmalar bulunmaktadır. Örneğin, Vaziripour ve ark. tarafından ilk önce Whatsapp, Viber ve Facebook Messenger hakkında bir kullanıcı çalışması yapılmış ve kimlik doğrulama seremonisini bulma ve tamamlama başarısı ve sürelerini ölçülmüştür [55]. Ardından, kimlik doğrulama seremonisini tamamlamak için daha kullanışlı bir yol sağlayan yeni bir Signal sürümü gerçeklenmiştir [54]. Sonuçlar başarı oranları ve tamamlama süreleri açısından çok daha iyi olsa da, kullanıcıların gerektiğinde kimlik doğrulama seremonisini doğru bir şekilde gerçekleştirmelerini garanti edememektedir. Ayrıca, yine aynı araştırma grubu tarafından otomatikleştirilmiş bir kimlik doğrulama seremonisi içeren bir Signal sürümü gerçeklenmiştir. Ancak, bu çözüm, kullanıcıların sosyal medya hesaplarını bir kimlik sağlayıcısı olarak kullandığından yeterince güvenilir görünmemektedir [53].

Kullanıcıların güvenlik bilincini, kullanılabilirlikle artırmayı hedefleyen çalışmaların aksine, açık anahtar doğrulamasını gerçeklemenin daha yenilikçi bir yolu blokzincir tabanlı açık anahtar altyapısı (PKI) kullanmaktır. Son zamanlarda yapılan bazı araştırmalar blokzincir teknolojisinin bölünmüş dünya saldırısı (Split-World), sertifika iptal etme/onaylama (revocation/validation) problemleri ve güvenilir sertifika/anahtar depolama yönetimi problemleri gibi bazı kritik sorunları merkezi bir kontrol sunucusu kullanmadan çözebileceğini göstermiştir [45] [39]. Dahası, blokzincirin sağlayabileceği faydalar arasında, sertifika şeffaflığı, merkezi hata noktalarının ortadan kaldırılması ve güvenilir bir işlem kaydı gibi noktalar da öne çıkmaktadır [24].

Genellikle, blokzincir tabanlı sistemlerin, herhangi bir güvenilir üçüncü tarafa (TTP) ihtiyaç duymadıkları için, istemci-sunucu tabanlı sistemlerden daha güvenli olduğu düşünülse de, yeni nesil PKI sistemleri oluşturulurken sistemin tabanını oluşturan blokzincir bölümünde ölçeklendirilebilir, kararlı, güvenli ve güvenilir bir kriptopara kullanılmasına dikkat edilmelidir. Bu yüzden, bu alandaki çalışmalar, % 51 saldırısı gibi yaygın saldırılara daha dayanıklı olan Bitcoin ve Ethereum'u temel alan blokzincir teknolojilerini kullanmaya odaklanmaktadır. Sonuç olarak, bunlardan birini bir PKI sisteminde taban olarak kullanmak, blokzincir katmanının olası zafiyetleri ile (en azından şimdilik) uğraşmadan daha güvenli ve güvenilir bir sistem elde etmek daha anlamlı olacaktır.

Her şeyden önce, bu çalışmada neden Blockstack platformunun kullanıldığını literatürde yer alan benzer fikirlerle karşılaştırarak açıklamak gerekir. Örneğin, Baldi ve arkadaşları, Bitcoin'den çatallanmış ve özel (private) blokzincirler konuşlandırmaya izin veren MultiChain tabanlı bir çözüm önermiştir. Bu çözüm sadece sertifika iptali (revocation) sorununa odaklanır ve sertifika otoritelerinin (CA) hiyerarşik yapısını korur. Asıl fark, iptal edilen sertifikaların CRL yerine açık muhasebe defterine (public ledger) eklenmesidir[25]. Her ne kadar günümüzde kullanılan konvensiyonel PKI sisteminden daha iyi olsa da, temel sorunları çözen kapsayıcı bir çözüm değildir.

Ethereum, kolayca uygulanabilen, akıllı sözleşmeler ve programlanabilirlik, düşük işlem ücretleri ve daha hızlı işlem süreleri gibi Bitcoin'e göre bazı avantajlara sahip olduğundan, birçok çalışmada baz alınmıştır. Örneğin, Ghazal [47], Ethereum'da akıllı bir sözleşme olarak uygulanan blokzincir tabanlı bir PKI sistemidir. Alan adı kaydı sırasındaki fiyatlandırma problemini, karmaşık bir fiyatlandırma işlevini kullanmak yerine bir isim için en yüksek teklifi verenin bu ismi kaydetme hakkını kazandığı "açık artırmaları" kullanarak kolay bir şekilde çözüyor gibi görünse de, ölçeklenebilirlik ve alan adı doğrulama hızı açısından yeterli görünmemektedir. Dikkate değer başka bir çözüm ise güven ağı modeli (web-of-trust) tasarlayan ve Ghazal gibi akıllı sözleşmeleri kullanan SCPKI [16] sistemidir. Ancak, bu sistem sınırlı uyumluluk ve mahremiyet sağlar.

Bitcoin ve Ethereum tabanlı çalışmaların yanı sıra, mevcut bir kriptopara birimini kullanmayan dikkate değer teorik çalışmalar da bulunmaktadır. Örnek olarak, Fredriksson, hisse ispatı (PoS) protokolünü kullanan ve Bitcoin'den (5.2 MB) daha büyük işlem boyutuna sahip olan Merkle ispatı tabanlı bir çözüm önermiştir. Bu çözümün amacı tamamen yeni bir sistem kurmak yerine mevcut PKI organizasyonuna kolayca uyarlanabilecek bir sistem geliştirmektir. Öte yandan, düşük işlem hacmi ve yüksek blok başlığı boyutu (Bitcoin'de 4.2 MB iken burada 24 MB) gibi sınırlılıkları nedeniyle uygulanabilirlik açısından yeterli görünmemektedir [34].

Netice itibariyle, Blockstack platformunun kimlik doğrulama seremonisi için mobil uygulamalarda kullanılabilecek en uygun blokzincir tabanlı PKI çözümü olduğuna kanaat getirilmiştir.

Bu çalışma haricinde, daha güvenli kimlik doğrulama için Blockstack platformunu kullanan iki projeden daha bahsedilebilir. Bunlardan ilki, Blockstack platformunun barındırdığı tüm merkezi olmayan uygulamaları (dApp) çalıştırmak amacıyla geliştirilen ve merkezi olmayan bir uygulama olan Stealthy'dir. Ayrıca uçtan uca şifreleme için Blockstack kullanan bir anlık mesajlaşma bileşenine sahip olmasına rağmen, kimlik doğrulama seremonisi özelliğine sahip değildir. Dahası, Stealthy, kullanıcı bilgisi olarak sadece Blockstack kullanıcı adını ve oluşturulan açık anahtarı kullanır ki bu durum, Blockstack platformuna tamamen bağımlı olmak anlamına gelecektir[14]. Diğer taraftan, Signal tabanlı uygulamalar SMS doğrulamasını kullanır çünkü telefon numarası, bir kullanıcıya ait olup olmadığı tespit edilebilecek yegane güvenilir bilgidir. İkinci proje ise, güvenli bir tarayıcı tabanlı sohbet platformu olan OpenIntents sohbet platformudur (OI Chat). OI Chat, Signal protokolü gibi Double Ratchet Algoritmasına dayanan bir protokol olan Matrix protokolünü kullanır. Blockstack platformunu yalnızca uygulamaya girişte kimlik doğrulaması (authentication) için kullanır.[8]

2. TEKNİK ARKA PLAN

2.1 Açık Anahtarlı Şifreleme

"Asimetrik şifreleme" olarak da adlandırılan açık anahtarlı şifreleme (public key cryptography), şifreleme ve çözümleme anahtarlarının farklı olduğu ve günümüzde kullanım alanı oldukça yaygınlaşmış bir tür şifreleme metodudur. İlk olarak 1976 yılında Diffie ve Hellman tarafından ortaya atılmıştır[31]. Bu şifreleme yönteminde, her bir kullanıcıda açık (public) ve gizli (private) anahtar olmak üzere iki adet anahtar bulunur. Açık anahtar, gizli anahtar kullanılarak elde edilebilirken açık anahtardan gizli anahtarın elde edilmesi ise ayrık logaritma problemine (discrete logarithm problem) dayanarak pratikte mümkün değildir. Bu yüzden, açık anahtar herkesle paylaşılabilirken gizli anahtar ise yalnızca sahibi tarafından bilinmesi gereken bir anahtardır.

Açık anahtarlı şifrelemede, kullanıcı, verisini kendi açık anahtarıyla şifreler. Bu şifre yalnızca karşılık gelen gizli anahtar ile çözülebildiğinden gizli anahtara sahip olmayan herhangi biri şifreyi çözemez. Diğer taraftan, herkesin erişebildiği açık anahtar vasıtasıyla gizli anahtar kullanılarak oluşturulmuş bir imza doğrulanabilir. Sonuç olarak, açık anahtarlı şifreleme, tasarımı itibariyle oldukça kullanışlı bir yapıya sahiptir.

Açık anahtarlı şifrelemenin kullanışlı yapısı, bu yöntemin günlük hayatta birçok alanda kullanımına olanak sağlamış, dünya ölçeğinde dijitalleşmeyi hızlandırmıştır. İlk etapta, e-posta güvenliği için ortaya atılan PGP protokolü [36] yeterince kullanılabilir olmadığından yaygınlaşmamış olsa da, SSL protokolünün [42] ortaya çıkışıyla birlikte elektronik bankacılık, e-ticaret ve güvenli haberleşme başta olmak üzere birçok alanda açık anahtarlı şifreleme vazgeçilmez hale gelmiştir.

2.2 Uçtan Uca Şifreleme

Merkezi sunuculara bağlı günümüzün internet yapısında SSL protokolünün ortaya çıkışıyla birlikte istemcilerle sunucu arasında gidip gelen verinin şifreli olarak iletimi söz konusu olmuştur. Bu durum, verilerin açık bir şekilde iletilmesine nazaran bir nebze veri güvenliğini artıran bir işlem olarak görünse de, sunucuda saklanan şifreleme anahtarlarının ele geçirilmesi veya sunucuda arka kapı (backdoor) bulunması durumunda iletilen veriyi ele geçirme tehlikesini bertaraf etme yönünde yetersiz kalmaktadır. Bu noktada, şifrelemenin sunucu üzerinden gerçekleştirilmesi yerine, istemcilerin doğrudan aralarında güvenli bir iletişim kurarak şifreli haberleşmeyi sağlamaları fikri ön plana çıkmıştır. Buna göre, gönderici şifrelemek istediği veriyi alıcının açık anahtarıyla şifreleyerek sunucuya iletir. Sunucu ise gelen şifrelenmiş veriyi alıcıya iletir. Bu sayede, sunucu şifreleme işleminin doğrudan bir parçası olmadığından ve alıcının özel

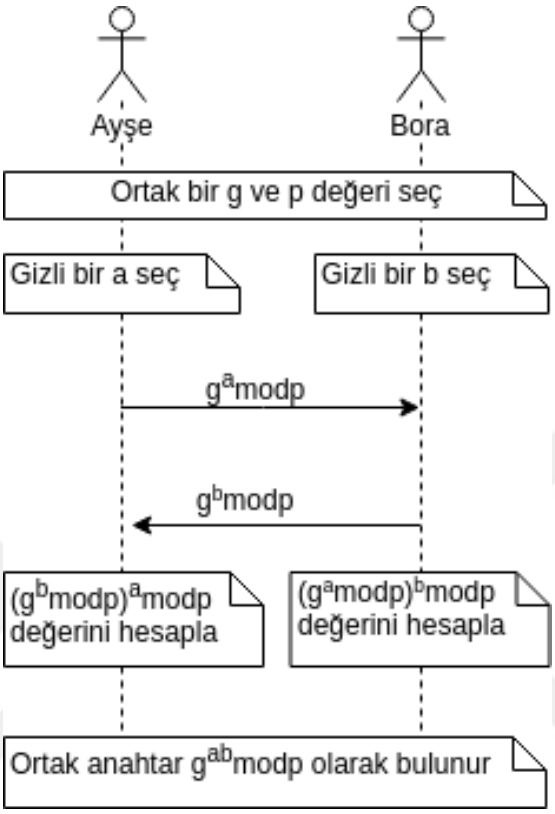
anahtarına sahip olmadığından alıcıya iletilen veriyi çözemez. Böylece, gerek dışarıdan ele geçirilebilecek gerekse arka kapı barındırabilecek bir sunucunun yol açabileceği tehditlerden korunulmuş olur.

Uçtan uca şifreleme metodunda, iki tarafın birbirlerinin şifreleme anahtarlarından nasıl haberdar olacakları önemli bir husus olarak göze çarpmaktadır. Bu noktada, önceden taraflar arasında bir gizli anahtarın paylaşılarak şifrelemede kullanılması (PGP'de olduğu gibi), bu gizli anahtardan elde edilen bir başka anahtarın şifrelemede kullanılması (DUKPT'de olduğu gibi) ve Diffie-Hellman anahtar değişimi olarak adlandırılan iletişim esnasında ortak bir gizli anahtar belirleme gibi yaklaşımlar mevcuttur. Bunlar arasında, daha kabul görülen yaklaşım SSL protokolünde de kullanıldığından güvenli mesajlaşma uygulamalarının önünü açan Diffie ve Hellman'ın 1976 yılında önerdikleri Diffie-Hellman anahtar değişimi yöntemidir [31]. Bu yöntem, iki taraf arasında güvensiz kabul edilen bir hat üzerinde güvenliği ayrık logaritma problemine dayanan güvenli bir ortak gizli anahtar belirlenmesini sağlar.

Diffie-Hellman anahtar değişimi Şekil 2.1'de gösterildiği gibi şu şekilde gerçekleşir. Aralarında güvenli bir iletişim kurmak isteyen Ayşe ve Bora, aralarında bir asal p değeri ve bir de üreteç (generator) adı verilen g değerine karar verir ve bu sayıları deklare ederler. Ardından, Ayşe gizli bir a tamsayısı belirleyerek Bora'ya $g^a mod p$ değerini gönderirken Bora ise Ayşe'ye belirlediği gizli bir b tamsayısı ile elde ettiği $g^b mod p$ değerini gönderir. Burada, a ve b sayıları sırasıyla Ayşe ve Bora'nın gizli anahtarlarıyken p ve g değerleri ise herkese açık değerlerdir. Son aşamada ise, Bora, Ayşe'den aldığı $g^a mod p$ değerinden kendi gizli anahtarı olan b sayısını kullanarak $(g^a mod p)^b mod p$ değerini hesaplarken Ayşe ise kendi gizli anahtarı olan a sayısı ile aynı işlemi yaparak $(g^b mod p)^a mod p$ değerine ulaşır. Bu iki değer, matematiksel olarak aynı değeri ifade eder ve $g^{ab} mod p$ değerine eşittir. Dolayısıyla, bu işlemlerin sonucunda a ve b değerleri başka biri tarafından bilinmediğinden ara değerler açığa çıksa bile oluşan sonuçta elde edilen değere ulaşılamaz ve bu değer, Ayşe ve Bora'nın ortak gizli anahtarı olur.

2.3 Kimlik Doğrulama Seremonisi

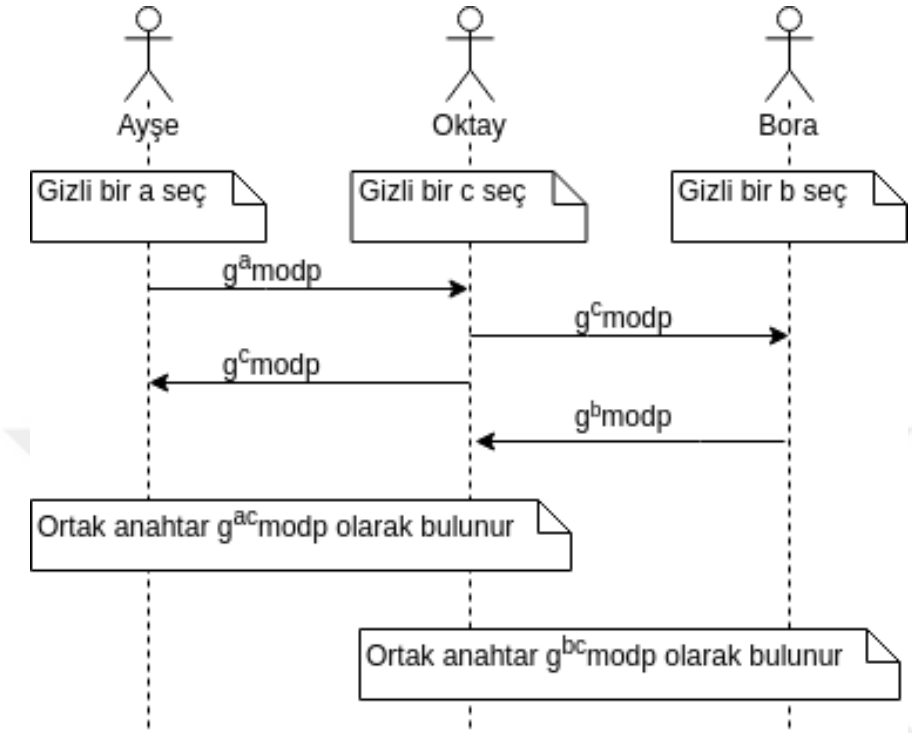
Diffie-Hellman anahtar değişim algoritması oldukça kullanışlı olmasına karşın, iki taraf arasında güvenli bir haberleşmeyi garanti edememektedir. Bu protokolde kimlik doğrulama mekanizması bulunmadığından MITM saldırılarına karşı dayanıklı bir çözüm sunma noktasında yetersiz kaldığı görülmüştür. Örnek olarak bu algoritma yardımıyla yapılan bir anahtar değişimine gerçekleştirilebilecek muhtemel bir saldırı senaryosu Şekil 2.2'de gösterilmiştir. Bu senaryoya göre, araya giren bir saldırgan (Oktay olsun), Ayşe ve Bora'nın birbirlerine gönderdikleri ara değerleri alarak kendi gizli anahtarıyla oluşturduğu $g^c mod p$ değerini gönderir. Bu sayede, Ayşe ve Bora birbirleriyle anahtar değişimi yaptıklarını düşünürken Oktay ile anahtar değişimi yapmış



Şekil 2.1: Diffie-Hellman Anahtar Değişimi

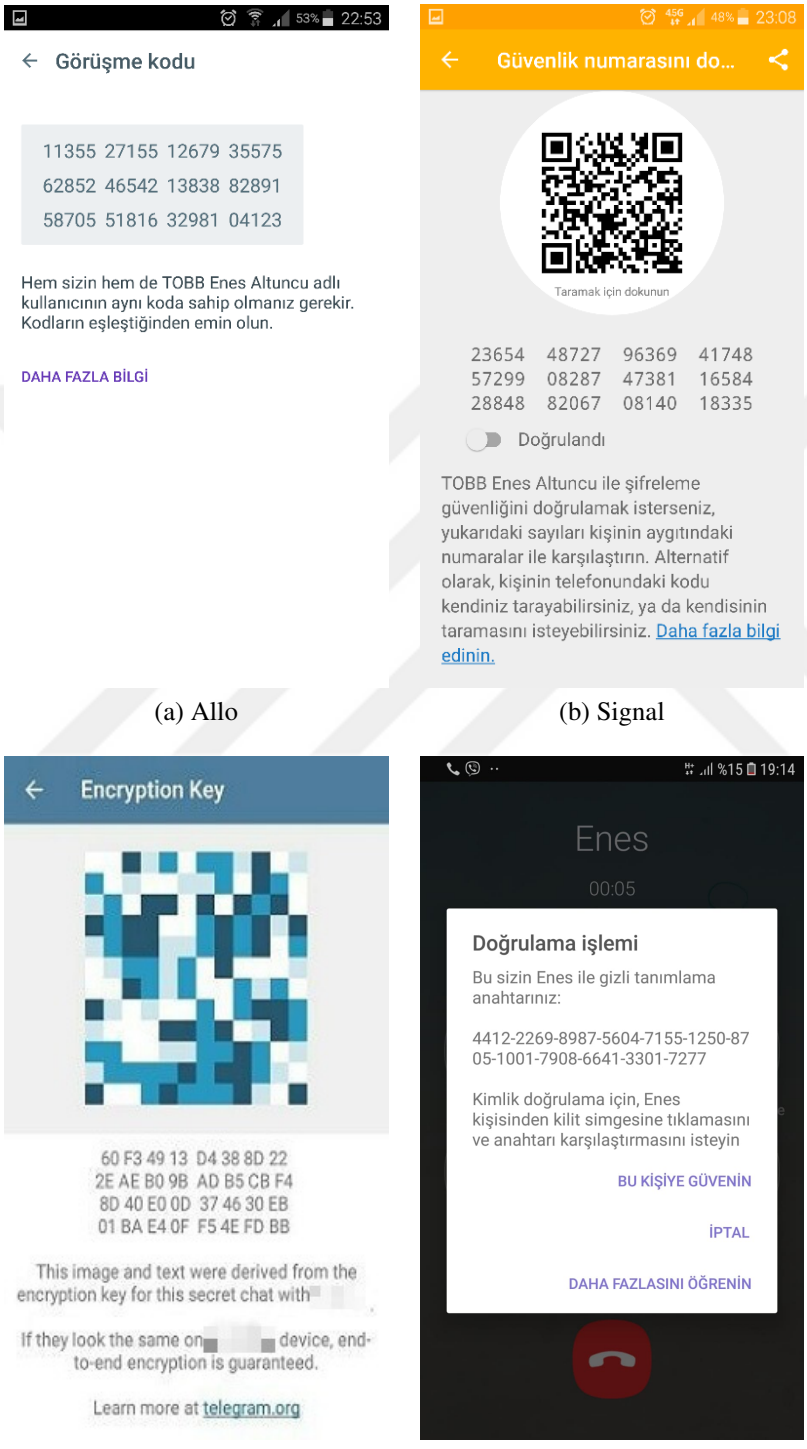
olurlar ve sırasıyla $g^{ac} mod p$ ve $g^{bc} mod p$ gizli anahtarlarını elde ederler. Sonuç olarak, Oktay, Ayşe ve Bora'nın kurmak istedikleri güvenli haberleşme kanalını ele geçirmiş olur.

Uçtan uca şifrelemeden kaynaklanan tehditlerin bertaraf edilebilmesi için kimlik doğrulama mekanizmalarının uygulama tasarımlarında bulundurulması önem kazanmıştır. Bu tür mekanizmaların anlık mesajlaşma uygulamalarında rastlanılan biçimi "kimlik doğrulama seremonisi (authentication ceremony)" adı verilen bir dizi işlemdir. Bu mekanizmanın amacı, istemci-sunucu modelini benimseyen mesajlaşma uygulamalarının sunucu kaynaklı tehditlerden arındırılarak gerektiğinde iki istemcinin, sunucudan bağımsız bir kanal vasıtasıyla açık anahtarlarını, diğer bir deyişle kimliklerini, doğrulayabilmesidir. Kimlik doğrulama seremonisi, yapısı itibariyle kullanıcıyı güvenlik çemberine dahil eden (human-in-the-loop) bir yaklaşım olduğundan kullanılabilirlik-güvenlik ikileminde dengeyi bozmamak adına bu seremoninin tamamlanması inisiyatifi kullanıcıya bırakılmıştır. Şekil 2.3'te görüldüğü üzere, farklı IM uygulamalarında farklı kullanıcı arayüzüne sahip kimlik doğrulama seremonileri gerçeklenmiş olsa da, tümü aynı mantıkla çalışır. Kullanıcı, mesajlaştığı kişinin mesajlaşmak istediği kişi



Şekil 2.2: Diffie-Hellman Anahtar Değişimine yapılan bir saldırı

olup olmadığı hususunda şüpheye düştüğünde veya mesajlaştığı kişinin açık anahtarı herhangi bir sebeple değiştiğinde, uygulama içerisinden mesajlaştığı kişiye ait açık anahtardan türetilmiş ve belirli sayıda karaktere sahip bir dizi karaktere ulaşır. Bu bir dizi karakter, uygulamadan uygulamaya değişen farklı isimlerle adlandırılırlar (Telegram'da şifreleme anahtarı ve Signal'da emniyet numarası vs.). Bu kodun içeriğinde açık anahtar bulunsa da, tasarıma göre telefon numarası gibi kullanıcı bilgileri de harmanlanmış olabilir. Kullanıcı, ulaştığı bu karakter dizisine mesajlaştığı kişinin de ulaşmasını ister ve daha güvenli olduğu varsayılan bir başka kanal vasıtasıyla kendi kodu ile diğer kodu karşılaştırır. Eğer eşleşme sağlanırsa anahtar değişiminin bir saldırı neticesinde olmadığı ortaya çıkmış olur. Kodların kolayca karşılaştırılabilmesi amacıyla QR kod teknolojisi de bahsekonu uygulamalarda yaygın olarak kullanılmaktadır. Buna karşın, gerek kullanıcıların güvenlik farkındalığının yeterli olmaması, gerekse ayrı ve güvenli bir kanaldan gerçekleştirilmesinin kullanılabilir güvenlik açısından zaman alıcı bir işlem olması nedeniyle kimlik doğrulama seremonisinin bu haliyle kullanılabilir güvenlik açısından istenilen sonucu ortaya koymadığı söylenebilir.



(a) Allo (b) Signal

(c) Telegram (d) Viber

Şekil 2.3: Farklı anlık mesajlaşma uygulamalarında kimlik doğrulama seremonisi

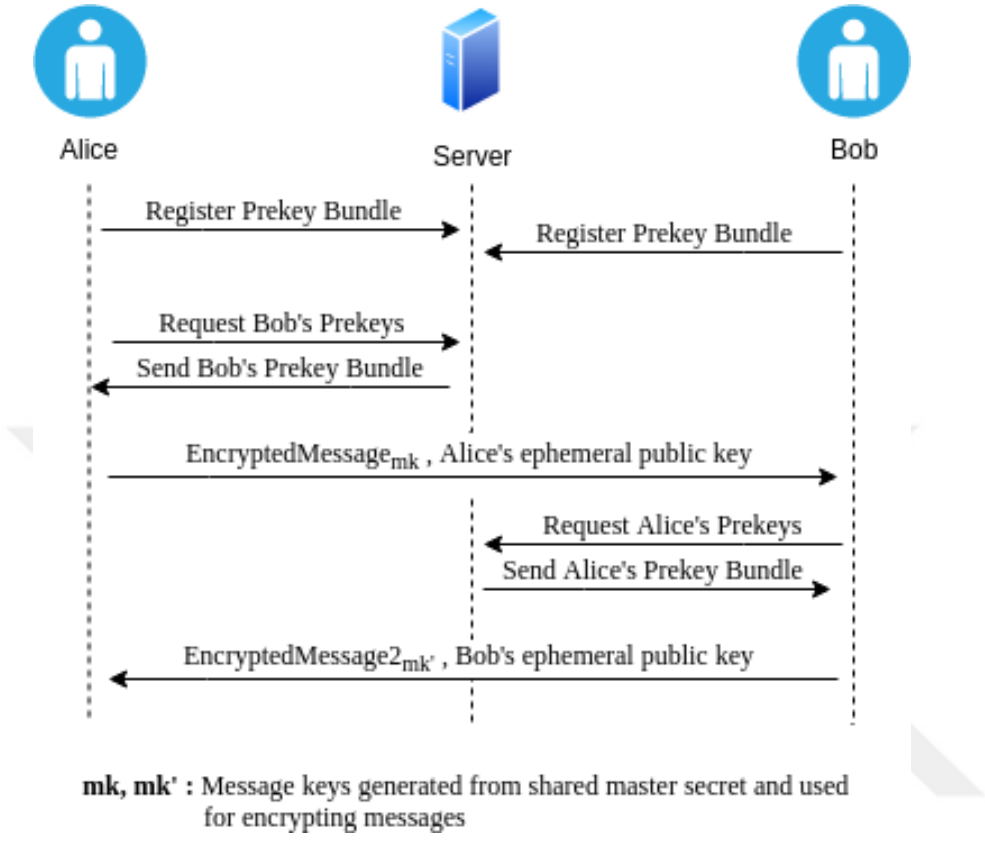
3. SIGNAL ANLIK MESAJLAŞMA UYGULAMASI

3.1 Signal Protokolü

Signal Protokolü [7], anlık mesajlaşma uygulamalarında future ve forward secrecy gibi güvenlik özelliklerini sağlayan ve Open Whisper Systems tarafından geliştirilen açık kaynaklı bir şifreleme protokolüdür. Temel olarak Double Ratchet algoritması [49] ve Extended Triple Diffie-Hellman anahtar anlaşması protokolünü [50] kullanır. Bu protokol, ortak anahtar materyallerinin depolanması ve mesajların kullanıcılar arasında iletiminden sorumlu olan merkezi bir sunucuya bağlıdır. Bir kullanıcının gizli anahtarı, kayıt sırasında mobil cihazında saklanırken, karşılık gelen açık anahtar, merkezi sunucunun kimlik veritabanında saklanır.

İki kullanıcı birbirleriyle güvenli bir şekilde mesajlaşmaya başlamak istediklerinde, anahtar çiftlerini ilk adım olarak oluşturmaları gerekir. Signal Protokolünde, bu anahtar çiftleri uzun vadeli bir kimlik anahtarı çiftine, orta vadeli bir imzalanmış önanahtar (prekey) çifti ve birçok geçici anahtar çiftine karşılık gelir. Bu çiftler istemci tarafında üretilir ve lokal cihazda depolanır. Ardından, kayıt için tüm açık anahtarlar ve kullanıcının kayıt numarası önanahtar buketi (prekey bundle) içine paketlenir ve bu paket bir Anahtar Dağıtım Merkezine (KDC) kaydedildiğinde kayıt aşaması tamamlanır. Kayıttan sonra, A kişisi, B kişisiyle bir oturum başlatmak istediğinde, önanahtar buketini sunucudan istemelidir. Ardından, sunucu B'nin önanahtar buketini A'ya gönderir. A kişisi, kendisinin ve B'nin önanahtar paketlerini, aralarındaki mesajları şifrelemek için kullanılacak ortak anahtarla birlikte, bir kök anahtar ve daha sonra yeni mesaj anahtarlarının üretilmesinde kullanılacak bir zincir anahtar üretmek için kullanır. Bu ortak anahtar, X3DH algoritması kullanılarak üretilir. Son olarak, A kişisi, onaylaması için B'ye ortak anahtarı gönderir, bu da mesajların gönderilmeye başlayabileceği anlamına gelir. Signal Protokolü'nde Alice ve Bob kullanıcıları arasındaki mesajlaşma adımlarına genel bir bakış Şekil 3.1'de görülebilir.

Bir kullanıcının açık anahtarı, kullanıcının oturumu yeniden kurması gerekmediği sürece (örneğin, bir cihaz değişikliği, uygulamayı yeniden yükleme veya bir saldırı durumu olmadığında) aynı kalır. Açık anahtar değiştiğinde, kullanıcı açık anahtarı ve telefon numarası gibi bilgilerle uygulama tarafından oluşturulan 60 karakter uzunluğundaki oturuma özgü emniyet numarası kullanılarak dış bir kanal üzerinden kullanıcı tarafından doğrulanmalıdır ve sunucunun ilgili veritabanı uygun bir şekilde güncellenmelidir. Kimlik doğrulama seremonisi, Şekil 3.2'de gösterildiği gibi emniyet numarasını içeren ilgili QR kodu taranarak veya emniyet numarasının sayısal versiyonu manuel olarak karşılaştırılarak yapılabilir. Sonuç olarak, bu tasarım yeterince güvenli görünse de (açık anahtar şifrelemesi kadar güvenli), güvenliği, yine de, kullanıcıların siber güvenlik farkındalığına bağlıdır çünkü güvenlik zinciri en zayıf halkası kadar



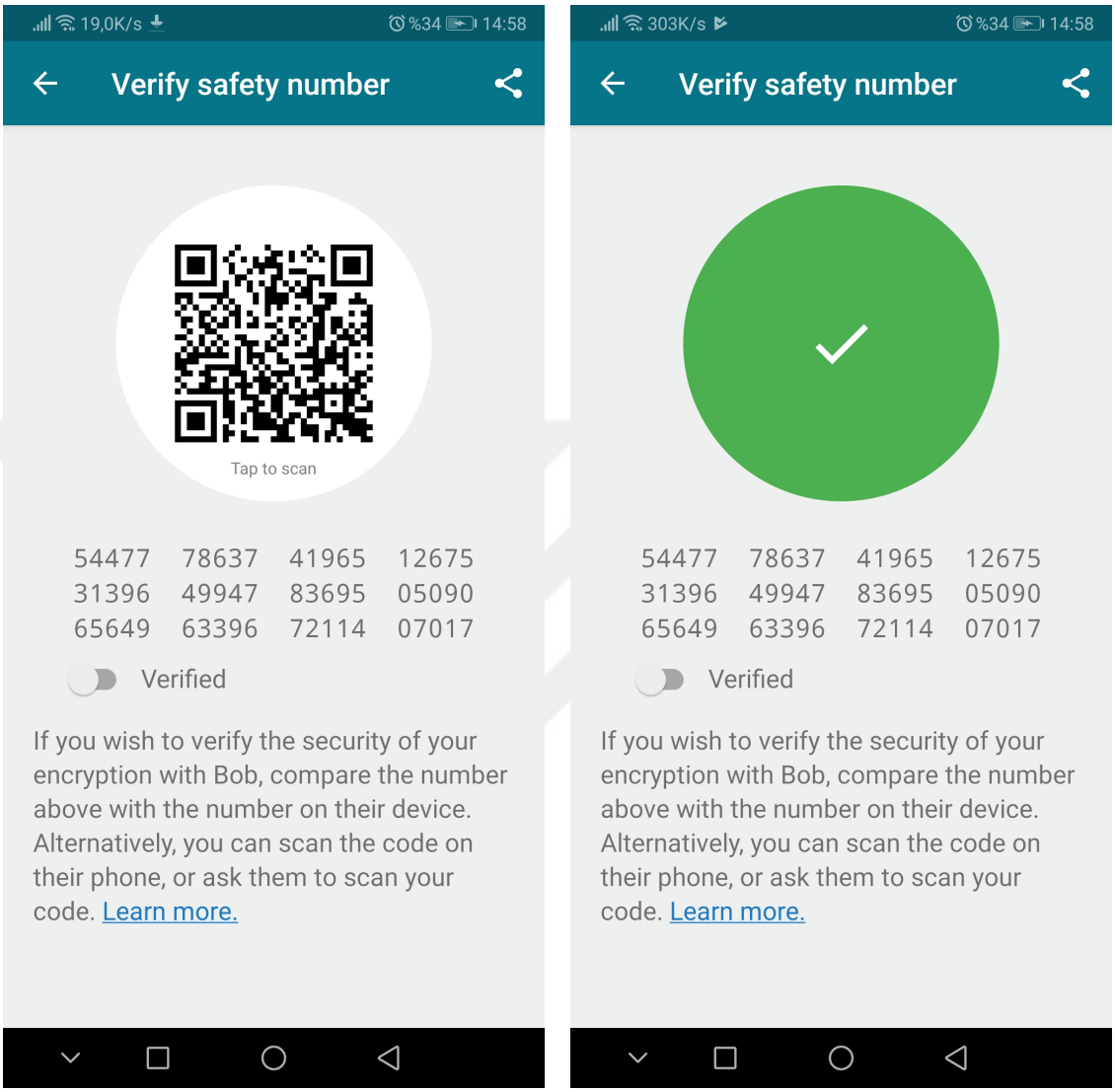
Şekil 3.1: Signal Protokolü ile Mesajlaşma

güçlüdür [32].

3.2 Signal Sunucusu

3.2.1 Temel bileşenleri

Signal, sunucu tarafında mesajların dağıtımından sorumlu Signal Server ve Android cihazlarda bildirim almayı sağlayan GCM bildirimlerinin iletiminden sorumlu Push Server olmak üzere iki sunucu kullanır. Bu bölümde, mesajların dağıtımı ve kullanıcılarla ilgili işlemlerden sorumlu olması nedeniyle Signal Server esas alınmıştır. An itibariyle kullanımda olan Signal uygulamasında, Signal Server içerisinde TextSecure Server ve RedPhone Server olmak üzere iki ana bileşen bulunmasına karşın, Signal üzerinden yapılan şifreli telefon görüşmeleri için kullanılan RedPhone Server'ın kaynak kodları erişime açık olmadığından, bu çalışmada elde edilen BlockSignal uygulamasında tele-



Şekil 3.2: Signal uygulamasında kimlik doğrulama seremonisi

fon görüşmesi yapabilme imkanı bulunmamaktadır. Dolayısıyla, bu bölümde, adı daha sonra "Signal Server" olarak değiştirilen TextSecure Server esas alınarak bu sunucunun yapısı detaylı olarak aktarılacaktır.

Signal uygulamasının işlerliğini sağlayan temel unsur olan Signal Server, uygulamanın sahip olduğu çeşitli fonksiyonların ifası adına üçüncü parti servislerden faydalanır. Bu servisler ve sağladıkları fonksiyonlar aşağıda maddeler halinde anlatılacaktır:

- *Twilio:* Bulut tabanlı bir iletişim platformu olan Twilio, sunduğu API desteğiyle

mobil uygulamaların kullanıcıya SMS gönderebilmesini ve arama yapabilmelerini sağlar. Signal uygulamasında bu platform uygulamaya kayıt esnasında SMS doğrulaması için kullanılır.

- *Amazon AWS S3:* Amazon'un bulut tabanlı depolama ortamı olan S3, mesajlaşma esnasında gönderilen eklentilerin merkezi sunucu kontrolünde saklanabilmesine olanak sağlar.

- *PostgreSQL:* PostgreSQL, ilişkisel veritabanlarını (relational database) esas alan C tabanlı bir veritabanı yönetim sistemidir. Signal sunucusu, kullanıcıların telefon numarası ve açık anahtarı gibi bilgilerini bir PostgreSQL veritabanı yardımıyla saklar ve gerektiğinde bu bilgileri günceller.

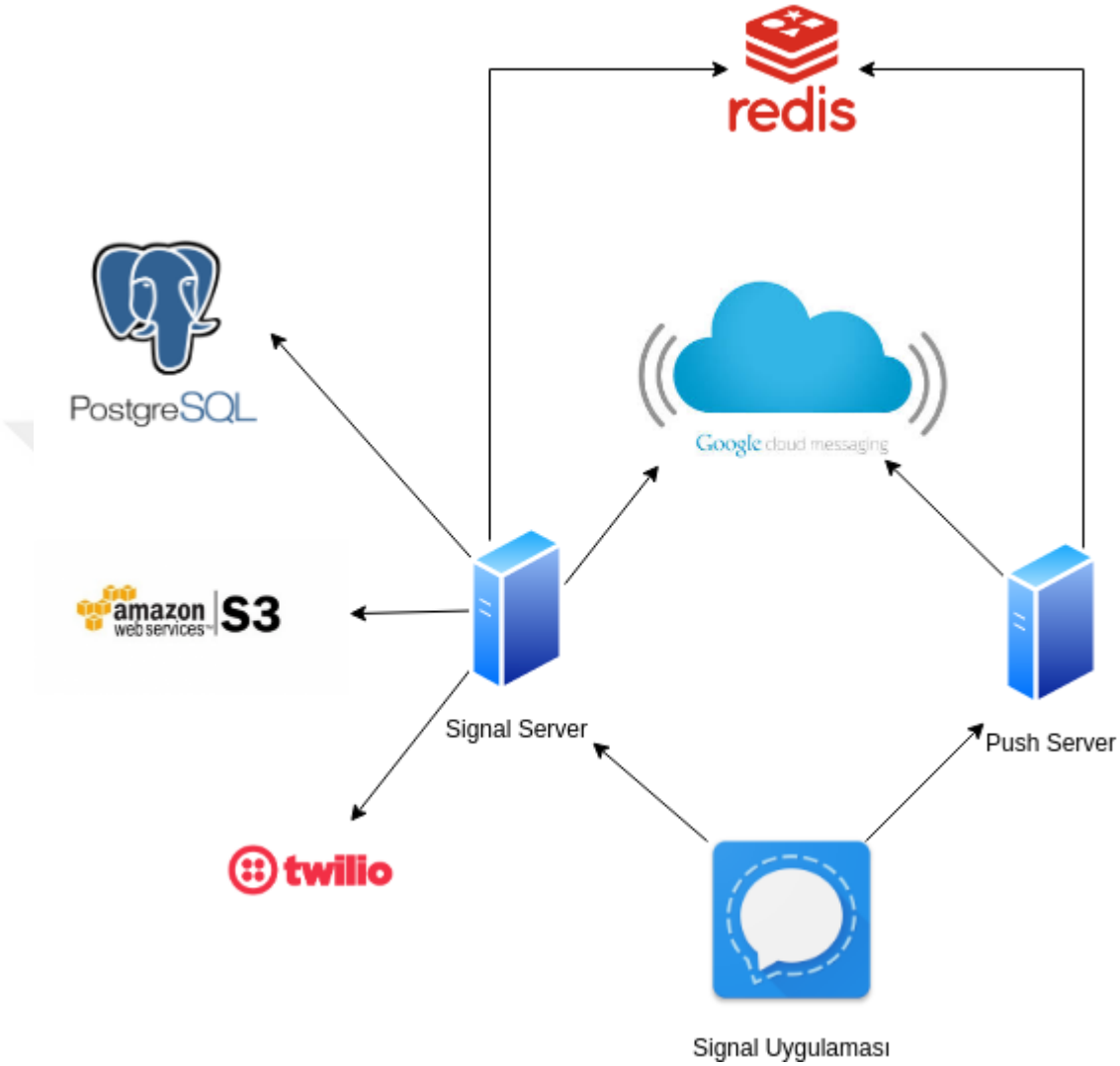
- *Redis:* Açık kaynak kodlu bir veri yapısı sunucusu olan Redis, önbelleğe alma amaçlı olarak Signal Server ve Push Server sunucularında kullanılır.

Signal uygulamasının sunucu yapısı ve sunucuların üçüncü parti yazılımlarla olan ilişkisi Şekil 3.3'de gösterilmiştir. GCM ve Redis, Signal Server ve Push Server arasında bir köprü vazifesi görürken diğer üçüncü parti yazılımların kullanımı Signal Server üzerinden sağlanmaktadır.

3.2.2 Kurulumu

İsteğe bağlı olarak uyarlanan Signal uygulamaları için uygulamayı geliştiren firma olan Open Whisper Systems tarafından Signal sunucularına bağlantı desteği verilmediğinden, bu uygulamaların çalıştırılabilmesi için lokal cihazda bir Signal sunucusu kurularak çalışır halde bulundurulmalıdır. Bunun için, bir önceki alt bölümde belirtilen Signal Server[13] ve Push Server[10] bileşenlerinin açık kaynak kodları indirilmeli ve Maven yardımıyla derlenmelidir. Bununla birlikte, sunucunun çalışacağı cihazda, sunucunun kullandığı Java, PostgreSQL ve Redis yazılımlarının yüklü olduğundan emin olunmalıdır.

İndirilen Signal Server ve Push Server kodları içerisinde birer adet konfigürasyon dosyası bulunur. Bu dosyalar vasıtasıyla, sunucunun kullandığı bir önceki alt bölümde detayları verilen üçüncü parti yazılımlara ait gerekli konfigürasyon bilgileri ve sunucunun çalışacağı port gibi bilgiler yer alır. Dolayısıyla, sunucu kurulumunun ilk aşaması olarak, söz konusu üçüncü parti yazılımlara (GCM, Twilio ve Amazon AWS S3) yeni üyelik kaydı oluşturulmalı ve bu üyeliklere ait bilgiler EK-1 ve EK-2'de verilen şekilde karşılık gelen alanlara yazılmalıdır. Bunun yanısıra, bu çalışmada hedeflenen çıktının bir Android uygulaması olması nedeniyle, Android ve iOS işletim sistemlerine



Şekil 3.3: Signal sunucusunun temel bileşenleri

sahip cihazları destekleyecek şekilde konfigürasyon gerektiren sunucu konfigürasyon dosyalarında iOS ile ilgili alanlar, olası bir vakit kaybını önlemek adına çıkarılmıştır.

Sunucunun uygulama üzerinde iletilen mesajları ve kullanıcı hesap bilgilerini depolayabilmesi için birer adet PostgreSQL veritabanı oluşturulmalıdır. Oluşturulan bu veritabanları, sunucu konfigürasyonlarına uygun olarak konfigüre edilmelidir.Veritabanlarının oluşturulması ve konfigürasyonuna ilişkin terminal komutları Şekil 3.4'te verilmiştir. Son olarak Şekil 3.5'te verilen komutlar kullanılarak sırasıyla Push Server ve Signal Server ayağa kaldırılmalıdır. Ayrıca, sunucunun çalışabilmesi için gerekli olmasa da sunucu güvenliği için olmazsa olmaz olan SSL korumasının sağlanabilmesi için su-

nucu adına bir SSL sertifikası oluşturulmalı ve oluşturulan ".store" uzantılı sertifika dosyası Android uygulama içindeki "res/row" dizinine eklenmelidir. SSL sertifikasının oluşturulması için gereken betik EK-3'te verilmiştir.

```
$ sudo -i -u postgres
$ createdb accountdb
$ createdb messagedb
$ java -jar TextSecureServer-<VERSION>.jar accountdb migrate config/textsecure.yml
$ java -jar TextSecureServer-<VERSION>.jar messagedb migrate config/textsecure.yml
```

Şekil 3.4: Sunucu veritabanlarının oluşturulması

```
$ java -jar Push-Server-<VERSION>-capsule-fat.jar server pushserver.yml
$ java -jar TextSecureServer-<VERSION>.jar server config/textsecure.yml
```

Şekil 3.5: Signal sunucusunun çalıştırılması

4. BLOCKSTACK PLATFORMU

Blockstack, bir veya birden çok sunucuya dayanan mevcut internet yapısı yerine yeni nesil bir internet yapısı olarak geliştirilen kimlik, kimlik doğrulama ve depolama için açık kaynaklı blokzincir tabanlı bir platformdur. Princeton Üniversitesinde doktora öğrencisi olan Muneeb Ali tarafından doktora çalışmaları kapsamında 2014 yılında geliştirilmeye başlanan Blockstack, 2017 yılında aynı isimle şirketleşerek kullanıcıların beğenisine sunulmuştur. Gelinen noktada, 7500'den fazla gönüllü geliştiriciyle masaüstü, web ve mobil bileşenlere sahip bir ekosistem oluşturma yolunda ilerlemektedir.

Blokzincir teknolojisinin benzer şekilde kullanımına yönelik birçok girişim olsa da blokzincirde mevcut olan veri saklama limiti, düşük yazma hızı, sınırlı bant genişliği ve hesap defterinin (ledger) sınırsız olması gibi kısıtlar sebebiyle beklenen seviyede ilerleme kaydedilememiştir. Blockstack ise, sistemi veri ve kontrol düzlemi olmak üzere ikiye ayırarak ve veri düzlemini blokzincirden ayrı tutarak bu kısıtları aşabilmiştir. Böylece, blokzincirde yalnızca asıl veriye ulaştıracak sınırlı bilgiler (özet vb.) saklanarak sistemin ölçeklenebilirliği ve veri saklama kapasitesi artırılmış olur. Ayrıca, bu sayede, bahsedilen diğer kısıtların sistemin işleyişini yavaşlatma düzeyi de minimuma indirgenmiş olur.[19]

4.1 Temel Özellikleri

Merkezi sunuculara bağlı DNS ve PKI vasıtasıyla idare edilen günümüz internet yapısını ademimerkezi hale getirerek daha güvenli bir internet vaat eden Blockstack, bu amaç doğrultusunda mevcuttan farklı bir DNS ve PKI yapısı oluşturmuştur. İlk olarak, bu iki sistemin merkezi sunuculardan arındırılması için blokzincir teknolojisinden faydalanılmıştır. Ayrıca, günümüzde kullanılan PKI sisteminde zaman zaman karşılaşılan CA kaynaklı problemlerin (yanlış/geçersiz sertifika dağıtımı vs.) çözülebilmesi amacıyla DNS ve PKI sistemleri tek bir çatı altında toplanarak "Blockstack adlandırma servisi (BNS)" adı altında mevcut sistemlere alternatif blokzincir tabanlı yeni bir sistem oluşturulmuştur. Bu sayede, halihazırda her alan adının sahip olmadığı SSL sertifikasının (açık anahtarın) varsayılan olarak tüm alan adlarına tahsisi mümkün olabilmektedir. Bir başka deyişle, açık anahtarların alan adlarıyla eşleştirilerek insanlarca okunabilir (human-readable) hale getirilmesi bu şekilde mümkün kılınmıştır.

BNS de DNS gibi bir adlandırma (naming) servisi olsa da DNS ile kıyaslandığında aralarında önemli farklar bulunur. Bunlardan en önemlisi, DNS alan adlarını çözümleyerek IP adreslerini elde ederken BNS ise ademimerkezi bir LDAP sistemi gibi hareket eder ve alan adı formatındaki kullanıcı adlarını çözümleyerek kullanıcı verisine ulaşır. Esasen, BNS, DNS'in görevini yerine getirebilecek olsa da Blockstack'teki kullanım

alanı bu şekilde değildir. İkinci olarak, iki sistem de formatları bakımından aynı olan bölge dosyalarına (zone file) sahip olsa da bu dosyalar anlamsal açıdan birbirlerinden farklıdır. Standart bir BNS bölge dosyasında, DNS bölge dosyasından farklı olarak yalnızca kullanıcının uygulama verilerine işaret eden URI ve TXT formatında kayıtlar bulunur. Ayrıca, DNS'ten farklı olarak her Blockstack kimliği için geçmiş bölge dosyaları da bulunur ve bu dosyalar kullanıcı adı çözümleme şekline etki edebilir. Bir diğer önemli fark ise, iki sistemde de aynı formatta alt alan adları (subdomain) bulunmasına rağmen, semantik olarak birbirlerinden ayrılır. Blockstack platformunda, alt alan adları, durum (state) ve kayıt (transaction) geçmişi blokzincire bağlı olmasına karşın, blokzincirde yer alan bir başka Blockstack kimliğine ait bölge dosya geçmişinde tutulan Blockstack kimliğini ifade eder. DNS alt alan adlarının aksine, BNS alt alan adlarının bağlı bulunduğu alan adından bağımsız bir sahibi bulunur ve bir alt alan adına ait kayıtlar yalnızca sahibi tarafından güncellenebilir. Dolayısıyla, BNS alt alan adları kendi başlarına çözümlenebilir (DNS'te bulunan hiyerarşik yapı burada mevcut değildir).[6]

BNS sistemi, DNS'in aksine ademimerkezi bir yapıya sahip olduğundan isteyen kullanıcıların yeni bir ad uzayı (namespace) oluşturması mümkündür. BNS sisteminde gerçeklenen ücretlendirme algoritması vasıtasıyla uzunluk ve harf-rakam kullanımı gibi çeşitli niteliklere göre ödenecek ücret sistem tarafından belirlenir. Bu ücretlendirme politikası Blockstack geliştiricileri de dahil olmak üzere sistemin tüm kullanıcıları için geçerlidir (Blockstack ".id" ad uzayı için 10000$ değerinde Bitcoin ödemek durumunda kalmıştır).

DNS sisteminde, bir alan adının çözümlenmesi sürecinde kök sunuculardan itibaren hiyerarşik bir yapı takip edilirken BNS sisteminde ise alan adları (kullanıcı adları) blokzincir vasıtasıyla çözümlenerek kullanıcı verilerine ulaşılır. İlk önce, alan adı sanal zincirde (virtualchain) aranarak alan adı-özet çifti elde edilir. Ardından, ulaşılan özet bir sonraki bölümde anlatılacak olan Atlas ağında aranarak ilgili bölge dosyasına ulaşılır. Bu dosyada bulunan URI bilgisi kullanılarak kullanıcı verisi okunarak (şifreliyse çözümlenir) imza veya özeti doğrulanır.[20]

Diğer taraftan, Blockstack, merkezi olmayan yapısıyla, DNS ve PKI gibi geleneksel internet yapısının temel bileşenlerinde bulunan merkezi güven noktalarını ortadan kaldırır. Blockstack üzerinde çalışan ve kısaca "dApp" adı verilen merkezi olmayan uygulamalar geliştirilebilse de, bu platformu mevcut Android, iOS, web ya da masaüstü uygulamalarıyla, platformun uygulama geliştirme kütüphanelerini kullanarak entegre etmek de mümkündür. Böylece, mevcut uygulamalar merkezi olmayan hale getirilebilir ve Blockstack platformunun sunduğu blokzincir teknolojisi başta olmak üzere tüm güvenlik özelliklerinden yararlanılabilir.

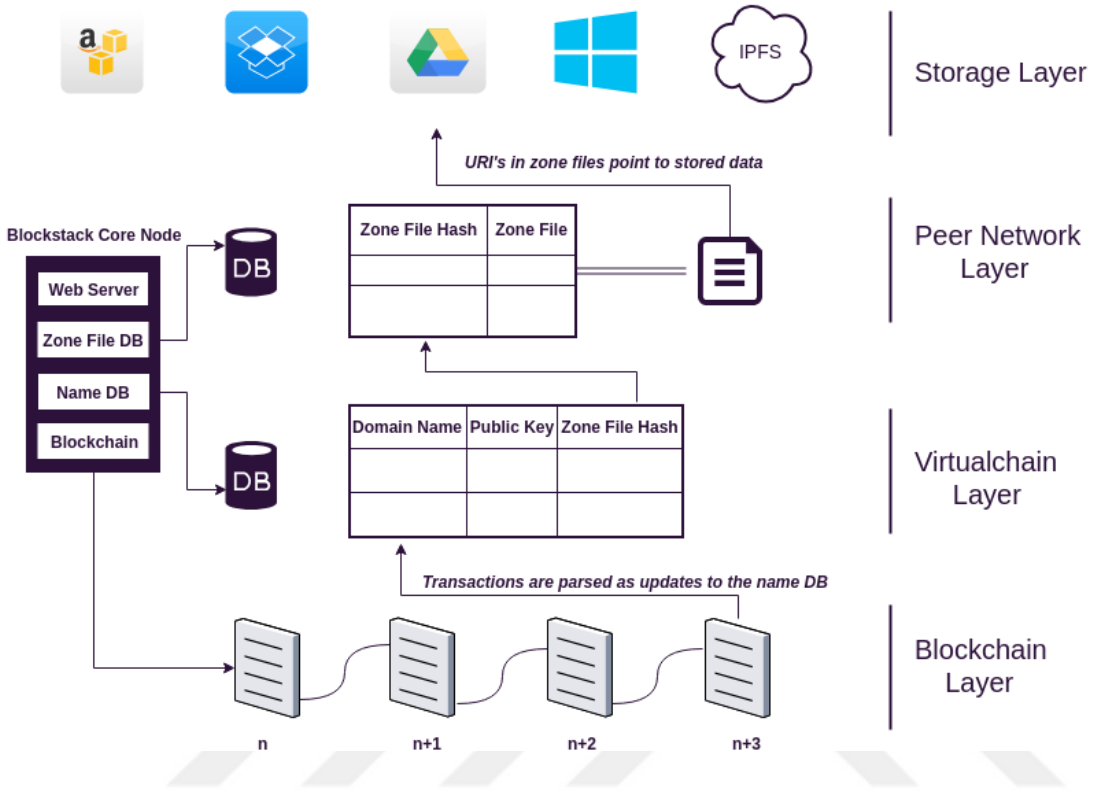
4.2 Sistem Mimarisi

Blockstack, modüler sürdürülebilir çok katmanlı bir mimariye sahiptir. Bu nedenle, bir katmanla ilgili bir güvenlik sorunu oluşması durumunda tüm sistemin işlevselliği etkilenmeyecek, tespit edilen problemi çözmek için yalnızca ilgili katman üzerinde çalışmak yeterli olacaktır. Blockstack, Şekil 4.1'de görülebileceği üzere, sırasıyla blokzincir, eş ağı (peer network) ve depolama katmanı olmak üzere üç ana katmana ve sanal zincir (virtualchain) olarak adlandırılan bir ara katmana sahiptir.

- Blokzincir Katmanı: Aynı zamanda "kontrol düzlemi" olarak da adlandırılan bu katman, operasyonların sırasına göre konsensüs sağlamak ve bu operasyonları depolamaktan sorumludur. Ayrıca, her bir operasyon türünü bir işlem (transaction) içinde kodlayabilmek amacıyla, "sanal zincir" adı verilen bir ara katman, bu katmanın üzerinde bir soyutlama (abstraction) olarak bulunur. Blockstack sistemi ile ilgili tüm işlemler (bir alan adının bir kullanıcı üzerine kaydedilmesi vb.) sanal zincirde tanımlandığından, bu katman, o anda en güvenli kriptopara türünü kullanma esnekliğini sağlar. An itibariyle, bu katmanda mevcut olanlar arasında en güvenli olduğuna kanaat getirilen Bitcoin kullanılmaktadır. Ayrıca, sanal zincir sayesinde farklı projelerde olduğu gibi var olan blokzincirin çatallandırılması gerekmez, blokzincir olduğu gibi kullanılabilir.

- Eş Ağ Katmanı (Atlas): Depolama katmanındaki verilerin sisteme özgü bir depolama birimine ihtiyaç duymaksızın halihazırda mevcut bulunan herhangi bir depolama ortamında saklanabilmesi amacıyla verilerin lokasyonlarının tespiti için kullanılan eşler arası bir ağdır. Bu durum, depolama katmanına esneklik ve mobilite sağlayarak ve sistemin güvenliğini tehdit etmeden mevcut yüksek kapasiteli depolama platformlarından yararlanılmasına yardımcı olur. Eş ağ, yönlendirme bilgilerini saklamak için biçimleri bakımından DNS bölge (zone) dosyalarına özdeş olan bölge dosyalarını içerir. Bölge dosyalarının ilgili özet (hash) değerleri, bütünlük için doğrulama sağlayan blokzincir katmanında saklandığından, kullanıcıların bu katmana güvenmeleri gerekmez.

- Depolama Katmanı (Gaia): Bu katman tüm gerçek verilerin depolandığı ve eş ağ katmanı ile birlikte "veri düzlemini" teşkil eden katmandır. Bu katmandaki depolama ortamı Google Drive, Dropbox ve Amazon S3 gibi yaygın olarak kullanılan bulut sağlayıcıların mevcut depolama ortamlarından oluşur. Bununla birlikte, bu bulut sağlayıcılar, tüm kullanıcı verileri, sistem tarafından oluşturulan ilgili kullanıcının kişisel anahtarıyla şifrelendiği ve/veya imzalandığından ve tüm açık anahtarlar ve veri özetleri blokzincir katmanı vasıtasıyla ulaşılabildiğinden, kullanıcı verilerine erişemez ve verilere müdahale edemezler. Dolayısıyla, kullanıcıların bu katmana da güvenmeleri gerekmez. [18]



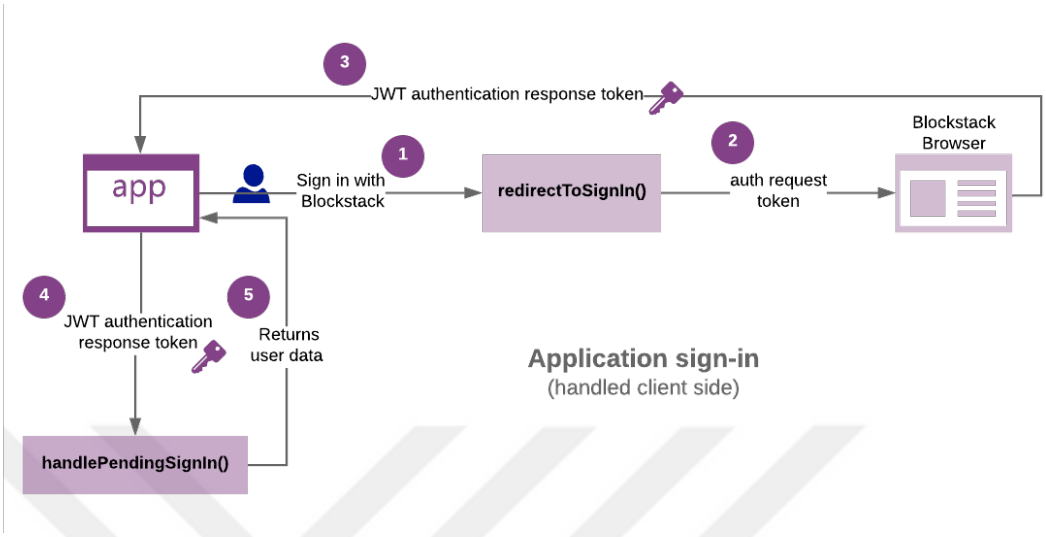
Şekil 4.1: Blockstack platformu katmanları

4.3 Kimlik Doğrulama

Blockstack, kimlik doğrulama için tek oturum açma (single sign on) imkanı sunan ve uzak sunucu veya üçüncü parti uygulamalar kullanmayan Blockstack Auth sistemini geliştirmiştir. Token tabanlı bir kimlik doğrulama sistemi olan Blockstack Auth'da kimlik doğrulama tamamen istemci tarafında gerçekleşir.

Şekil 4.2'de görüldüğü üzere, Blockstack platformunu kullanacak olan uygulama authRequest adı verilen tokenı Blockstack tarayıcısına göndererek kimlik doğrulama talep etmiş olur. Kullanıcı uygulama üzerinden Blockstack hesabına giriş işlemini gerçekleştirdiğinde Blockstack tarayıcı authResponse adı verilen tokenı göndererek uygulamaya cevap verir. Bu tokenlar secp256k1 desteğiyle birlikte RFC 7519 OAuth JWT formatındadır ve URL sorgu dizisi olarak iletilirler.

Blockstack tarayıcısı authRequest tokenını aldığında authResponse tokenını oluşturmak için yalnızca authRequest tokenını imzalamak için kullanılan ephemeral transit key adı verilen kısa ömürlü bir anahtar kullanır. Blockstack kullanacak uygulama, istek oluştururken bu anahtarı saklayarak anahtarın açık bölümünü authRequest içeri-



Şekil 4.2: Blockstack platformunda kimlik doğrulama

sinde bulundurur. Blockstack tarayıcısı ise bu açık bölümü kullanarak uygulama içi gizli anahtarı şifreleyerek authResponse tokenı içine yazar. Şekil 4.3'te authRequest ve authResponse tokenları için birer örneğe yer verilmiştir.

Kimlik doğrulama esnasında, Blockstack tarayıcısı, kullanıcının Blockstack kimliğine ait gizli anahtarı ve uygulamanın alan adını kullanarak uygulama içi gizli anahtarı üretir. Bu anahtarın görevleri, anahtarın tanımlandığı uygulamanın kullanıcıya ait Gaia depolama birimine erişim sağlayabilmesinin sağlanması, uygulama tarafından Gaia'da saklanacak verilerin uçtan uca şifrelenmesinin sağlanması ve uygulamanın kriptografik işlemler için anahtar olarak kullanabilmesidir. Bu anahtarın bir diğer özelliği ise belirli bir kullanıcı adı ve uygulama alan adı verildiğinde her seferinde aynı anahtarın üretilmesidir.[5]

4.3.1 Kimlik kurtarma

Blockstack platformu, kullanıcı adı-parola çiftine ihtiyaç duyan klasik sistemlere kıyasla farklı bir kimlik doğrulama yaklaşımını benimsediğinden kullanıcıların hesaplarını (Blockstack özelinde kimliklerini) kurtarabilmeleri için izlemeleri gereken yol da farklıdır.

Kullanıcı yeni bir Blockstack hesabı açtığında, Bitcoin cüzdanlarında da kullanılan BIP39 standardında "ipucu kodu (mnemonic code)" adı verilen 12 (yeni versiyonlarında 24) kelimeden oluşan rassal bir kelime dizisi Blockstack tarafından üretilir.

```
const requestPayload = {
    jti, // UUID
    iat, // JWT creation time in seconds
    exp, // JWT expiration time in seconds
    iss, // legacy decentralized identifier generated from transit key
    public_keys, // single entry array with public key of transit key
    domain_name, // app origin
    manifest_uri, // url to manifest file - must be hosted on app origin
    redirect_uri, // url to which browser redirects user on auth approval - must be hosted on app origin
    version, // version tuple
    do_not_include_profile, // a boolean flag asking browser to send profile url instead of profile object
    supports_hub_url, // a boolean flag indicating gaia hub support
    scopes // an array of string values indicating scopes requested by the app
  }
```

(a) authRequest

```
const responsePayload = {
    jti, // UUID
    iat, // JWT creation time in seconds
    exp, // JWT expiration time in seconds
    iss, // legacy decentralized identifier (string prefix + identity address) - this uniquely identifies the user
    private_key, // encrypted private key payload
    public_keys, // single entry array with public key
    profile, // profile object or null if passed by profile_url
    username, // blockstack id username (if any)
    core_token, // encrypted core token payload
    email, // email if email scope is requested & email available
    profile_url, // url to signed profile token
    hubUrl, // url pointing to user's gaia hub
    version // version tuple
  }
```

(b) authResponse

Şekil 4.3: Örnek birer authRequest ve authResponse tokenı

Bu kodun Blockstack kullanıcı arayüzündeki ismi "Secret Recovery Key (SRK)" olarak belirlenmiştir. SRK, kullanıcıların anahtar çiftlerinin üretiminde kullanılır. Ayrıca, SRK'dan anahtar üretimi kararlı bir yapıya sahip olduğundan kimlik kurtarma sürecinde eski gizli anahtarın kurtarılmasını da sağlar. Blockstack, SRK'nın üretimi için açık kaynak kodlu ve ipucu kodu üretiminde Bitcoin cüzdanları tarafından yaygın olarak kullanılan "bitcoinjs"[1] kütüphanelerinden faydalanır. Bu kütüphanelerde mevcut olan ipucu kodu üreteci sayesinde İngilizce 2048 adet kelime içerisinden kelimeler kullanılan web tarayıcısının (Chrome, Firefox vb.) rassal sayı üreteci vasıtasıyla rassal olarak seçilir. Kelime listesinde bulunan her bir kelime bir sayıyı ifade eder ve seçilen son kelime, dizinin sağlamasını (checksum) yapmak üzere kullanılır[22]. Üretilen SRK, Blockstack'e özgü bir formata sahip olmadığından ve üreteç kodu açık kaynaklı olarak mevcut olduğundan kullanıcılar bu anahtarı kendileri oluşturarak kullanabilirler. Blockstack hesaplarını oluşturan kullanıcılar, SRK bilgilerini Blockstack web uygulaması üzerinden görüntüleyebilirler.

İpucu kodu (Blockstack'teki ismiyle SRK), Bitcoin cüzdanlarıyla birlikte yaygınlaşmaya başlayan bir hesap kurtarma metodu olsa da bu kodun güvenli olarak kullanılma ve bu kod vasıtasıyla hesap kurtarma biçimi üzerinde henüz bir konsensüs sağlanamamıştır. SRK, gizli tutulması ve güvenli olarak saklanması gereken, aksi halde he-

sabın kaybına sebep olabilecek öneme sahip olmasından ötürü doğrudan bu anahtarın kullanımı güvenlik açısından riskli olarak görünmektedir. Bu probleme Blockstack'in yaklaşımı, doğrudan SRK'nın kullanımı yerine, şifrelenmiş hali olan ve Blockstack tarafından "Magic Recovery Code (MRC)" olarak isimlendirilen bir kodun kullanıcı parolasıyla birlikte kullanılmasının daha doğru olacağı yönündedir. MRC de SRK gibi gizli tutulması gereken bir dizi karakter olsa dahi kullanıcı parolası olmadan kullanılamayacağından Blockstack e-posta sunucusu aracılığıyla kullanıcıya e-posta ile gönderilir.

Yukarıda tanımlanan SRK ve MRC anahtarları vasıtasıyla kullanıcılar Blockstack hesaplarını kurtarmak istediklerinde önlerinde iki seçenek bulunur:

- Kullanıcı, Blockstack kullanıcı arayüzünde belirtilen alana SRK'yı girer. Ardından, kendisine en az 8 karakterden oluşan bir parola seçer. En sonunda, herhangi bir e-posta adresini girerek kimliğini kurtarır.
- Kullanıcı, Blockstack platformuna kaydı esnasında e-posta adresine gönderilmiş olan MRC'yi belirtilen alana girer. Ardından, daha önce seçtiği parolasını girer. Son olarak, herhangi bir e-posta adresini girerek kimlik kurtarma işlemini sonlandırır.

Kullanıcı, bu iki seçenekten birini tercih edebilecek bilgiyi elinde bulundurmuyorsa (örneğin, parolasını unutmuş ve SRK'yı kaybetmiş ise) hiçbir şekilde Blockstack kimliğini kurtaramaz. Blockstack kullanıcıların SRK ve MRC bilgilerini saklamadığından kullanıcıların bu bilgileri güvenli olarak saklamaları oldukça mühimdir. Aksi durumda, kullanıcıların Blockstack platformundan faydalanabilmek için yeni bir kimlik edinmesi gerekir. Diğer taraftan, kullanıcı SRK bilgisini Blockstack dışında kendisi oluşturmuşsa Blockstack kayıt işlemlerini tamamlamadığı için kimlik kurtarma sonrasında Blockstack kullanıcı adını belirlemelidir.

5. TEHDİT MODELİ

Bu bölümde, mevcut Signal Protokolü gerçeklemesinde var olan muhtemel zayıflıklardan bahsedilerek daha güvenli bir uygulama tasarımına zemin hazırlanması hedeflenmektedir. Signal, protokolü yeterince güvenli kılan açık anahtar kriptografisini kullanmasına rağmen [29] [44] [43], daha güvenli bir uygulama elde etmek için ortadan kaldırılması gereken bazı dezavantajlara sahiptir. Tehdit modelimiz için, aşağıdaki senaryolara (T) göre sınıflandırılmış bazı saldırı olasılıkları göz önünde bulunduruldu:

T1) Kötü Amaçlı/Ele Geçirilmiş sunucu. Signal Sunucu bir saldırı nedeniyle ele geçirilirse veya içeriden gelen bir tehdit nedeniyle kötü niyetli hale gelirse protokol MITM saldırılarına karşı zayıf kalır. Bu senaryo için, temelde aşağıdaki gibi iki olası durum vardır:

- *İlk kurulum sırasındaki saldırılar:* İlk kurulum sırasında, kötü niyetli sunucu, saldırganı kamufle etmek adına sıradan bir kullanıcıya saldırganın kimlik anahtarını başka bir kişinin anahtarı yerine verebilir. Bu tehdidi bertaraf etmek için Signal, QR kodları aracılığıyla bir doğrulama mekanizması sunar. Bununla birlikte, Signal bu konuda kullanıcıyı uyarmaz. Doğrulamayı yapmak kullanıcının sorumluluğundadır.

- *İlk kurulumdan sonra saldırılar:* Mesajlaşma sırasında, saldırgan, kötü niyetli veya ele geçirilmiş sunucunun yardımıyla bir MITM saldırısı gerçekleştirebilir. Bu durumda, alıcının açık anahtarı, saldırganın kimlik anahtarı olarak değişecektir. Signal, oturumda böyle bir açık anahtar değişikliği durumunda kullanıcıyı uyarır. Bununla birlikte, açık anahtar değişikliği saldırı durumunu garanti etmediğinden, yalnızca kullanıcıya şüphe uyandırması amaçlanmaktadır ve bu şüphenin üzerine gitmek kullanıcının kararıdır.

T2) SSL sabitlemeyi atlatma. Signal, MITM saldırılarını önlemek için SSL sertifikası sabitlemeyi kullanarak sunucunun açık anahtarını uygulama içine gömer [56] [38]. Böylece bir saldırgan, sunucusunu asıl Signal Sunucu ile değiştiremez. Bununla birlikte, bu önlemin tersine mühendislik gibi farklı yollarla atlatılması mümkündür. [21] [48] [46]

T3) SIM kart klonlama. Yeni güvenlik özellikleri nedeniyle bir SIM kartı klonlamak geçmişte olduğundan çok daha zor olsa da, telefon numaralarını kullanarak kullanıcıları tanımlayan uygulamalar için hala önemli bir problemdir [17] [23] [40]. Örneğin, bir saldırgan, Signal'da kayıtlı bir kullanıcının SIM kartını klonlarsa her telefon numarası yalnızca bir kullanıcı tarafından kullanılabildiğinden, aynı telefon numarasıyla

açılan mevcut oturum sonlandırılır ve saldırgan yeni bir oturum başlatmış olur. Bu durumda, Signal aynı kullanıcının uygulamayı aynı SIM kartla fakat başka bir cihazdan kullandığını varsayarak herhangi bir önleme başvurmaz.

T4) Bilinmeyen anahtar paylaşım saldırıları. Anahtar anlaşma protokolünde gerçekleştirilen bu saldırı türünde, saldırgan, ortak anahtar oluşturacağı kullanıcıya bir başka kişinin anahtarını vererek bu iki kullanıcı arasında istenmeyen bir oturum kurar [35] [27]. Signal, emniyet numaralarını kullanarak bu tür saldırıları önleyebilse dahi bu tür saldırıların gerçekleştirilmesi, daha önce parmakizlerini açık bir ortamda yayınlayan ve o zamandan bu yana Signal sürümünü güncellememiş kullanıcılar için hala mümkündür. Örnek olarak Ayşe'nin Oktay ile iletişim kurmak istediğini varsayalım. Muhtemel bir saldırı senaryosunda Oscar, Bora'nın parmak izini kendi parmakiziymiş gibi Ayşe'ye sunar. Bu durumda, Ayşe kimlik doğrulama seremonisinde bu parmakizini doğruladığında, beklediğinden farklı olarak Bora ile mesajlaşmaya başlar.

T5) Blockstack ile ilgili tehditler. Bu tezde belirtilen tasarımda, Signal'a yeni bir bileşen eklendiğinden, yeni bileşen olan Blockstack'in tasarıma yeni tehditler getirmeyeceğinden emin olunmalıdır. Öncelikle, Blockstack merkezi olmayan bir platform olduğundan ve kullanıcıların hiçbir zaman gizli anahtarlarını uzak sunuculara göndermemesi nedeniyle kullanıcıların verilerini veya kimliklerini kontrol edemez. Ayrıca, bir kullanıcı adı-açık anahtar çifti blokzincire eklendiğinden, saldırganın başarılı bir saldırı yapabilmesi için kullanıcı adı-açık anahtar çifti değiştirebilmesi için blokzincir kayıtlarıyla oynaması gerekir. Böyle bir saldırının gerçekleşmesinin tek yolu, saldırganın çatalı aynı kullanıcı adı için farklı bir açık anahtarın olduğu bir kaydı içeriyorsa ve bu çatalın diğerlerinden daha uzun olmasıyla kazanan çatal olarak seçilmesidir. Bununla birlikte, özellikle Blockstack platformu tarafından kullanılan Bitcoin blokzinciri göz önüne alındığında, böyle bir saldırının gerçekleştirilmesinin pratikte mümkün olmadığı açıktır. İkinci olarak, Blockstack'in depolama katmanının (Gaia) güvenliği, açık anahtarların depolanmasında kullanıldığından göz önünde bulundurulmalıdır. Gaia, bilgileri imzalı ve şifreli olarak depolamayı sağlar. Bu nedenle, ilgili gizli anahtara erişmeden, hiç kimse kullanıcıların verilerini değiştiremez veya silemez. Sonuç olarak, Blockstack'in yeni bir bileşen olarak eklenmesinin yeni güvenlik sorunlarına neden olmayacağı söylenebilir. Bu nedenle, bu yeni bileşenin daha fazla güvenlik önlemi almaya gerek kalmadan kullanılabileceği düşünülmüştür. Ancak, tek sorun, Blockstack'in kullanıcı adı, e-posta adresi ve sosyal medya hesapları gibi kullanıcı tanımı için kullandığı bilgilerin, bir kullanıcıyı doğru şekilde temsil edememesi olarak görünmektedir. Başka bir deyişle, kötü niyetli bir kullanıcı kendisini alıcıyı yanlış yönlendirebilecek farklı bir kişi olarak tanıtabilir.

6. TASARIM VE GERÇEKLEME

6.1 Tasarıma Genel Bakış

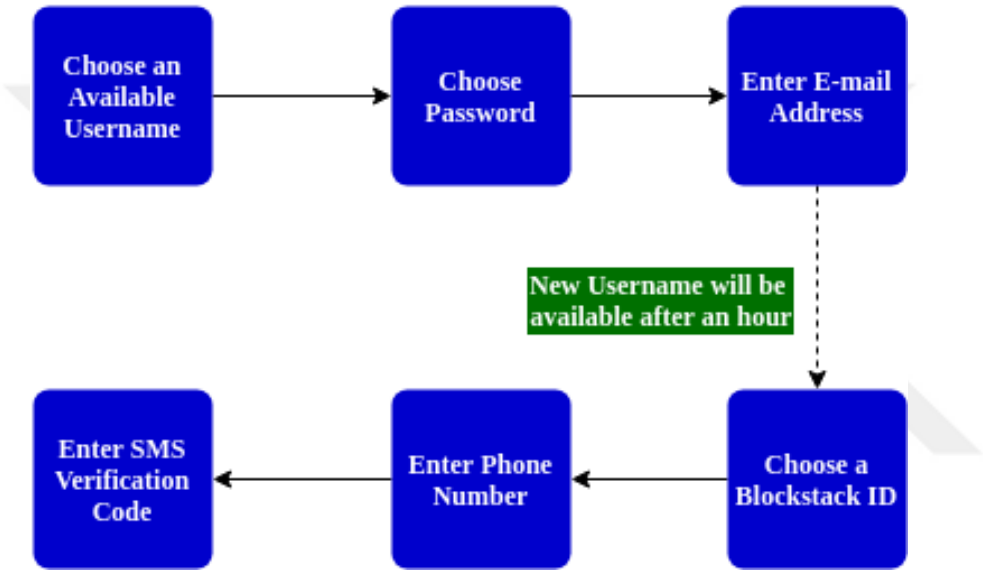
Orijinal Signal uygulaması, uygulamayı kullanarak haberleşen taraflar ve iletişimi sağlamaktan sorumlu olan merkezi Signal sunucusu olmak üzere iki ana bileşenden oluşur. Öte yandan, BlockSignal uygulamasında, tarafların açık anahtarlarının doğrulanmasında kullanılan fazladan bir bileşen olarak Blockstack platformu da bulunmaktadır. Bu tasarımda, Signal'daki iletişim akışı aynı kalırken, Blockstack, otomatikleştirilmiş bir kimlik doğrulama seremonisi mekanizmasını uygulayabilmek için bir kimlik sağlayıcı olarak kullanılır. Orijinal Signal uygulamasını Blockstack ile entegre etmek için Blockstack Android SDK [4] 0.4.3 versiyonu, kütüphanede gerekli değişikliklerle kullanılmıştır. Ayrıca, bu çalışmada, Signal Android [12] uygulamasının 4.20.4 versiyonu baz alınmıştır. Sonraki alt bölümlerde, Signal uygulamasında yapılan değişiklikler ve geliştirmeler detaylı olarak paylaşılacaktır.

6.2 İlk Kayıt

Signal uygulamasının kullanılabilmesi için uygulamaya kayıt olunması ve bu kayıt sonucunda, kullanıcı bilgisinin sunucunun ilgili veritabanına eklenmesi gerekir. Kayıt işlemi sırasında Signal Server, kullanıcıların kimliğini SMS ile doğrulayarak telefon numaralarını kullanıcıları tanımlamada kullanır. Öte yandan, BlockSignal, Blockstack vasıtasıyla kullanıcıları tanımlayabilecek yeni bir bilgi sunmaktadır. Her BlockSignal kullanıcısının uygulamayı kullanabilmesi için bir Blockstack hesabına sahip olması gerektiğinden, yeni tasarım, Blockstack'te oturum açmayı içermeli ve Blockstack kimlik doğrulamasını ile Signal'daki kayıt sürecini tek bir mekanizma olarak birleştirmelidir.

Bu amaçla, BlockSignal kayıt süreci Şekil 6.1'de gösterildiği şekilde tasarlanmıştır. İlk etapta, kullanıcılar uygulama aracılığıyla Blockstack hesaplarına giriş yapmalıdır. Ancak, yeni bir BlockSignal kullanıcısı Blockstack hesabına sahip olmayabilir. Bu durumda, Blockstack Browser'da çalışan oturum açma uygulaması "Yeni ID Oluştur (Create New ID)" seçeneği sunar. Eğer kullanıcı bu seçeneği seçerse, Şekil 6.2'de gösterildiği gibi uygun bir kullanıcı adı, şifre ve e-posta adresini sırasıyla girer. Son adımdan sonra, ".id.blockstack" ile biten alan adı formatındaki kullanıcı adı, bir Bitcoin kaydı içinde blokzincire eklenir. Bitcoin'in işlem süresi göz önüne alındığında, yeni kullanıcı adının kullanılabilir olması için yaklaşık bir saat gereklidir. Kullanıcı adı aktif hale geldiğinde, "Magic Recovery Code" şeklinde isimlendirilen bir anahtarı içeren e-posta, kullanıcının e-posta adresine gönderilir. Bu kod, kullanıcı başka bir cihazdan Blockstack'te ilk kez oturum açmak istediğinde kullanılır. İkinci seçenek olarak, Blockstack, 12 kelimeden oluşan anahtar bir cümle olan "Secret Recovery Key"

adı verilen bir anahtar sunar. Bu anahtar cümleyi, kullanıcılar Blockstack hesaplarına tarayıcıdan giriş yaparak görüntüleyebilirler. Bu anahtarla, kullanıcı Blockstack'e giriş yaparken şifresini değiştirebilir. Bu iki anahtar, Blockstack tarafından tutulmadığından kaybedildiği takdirde Blockstack hesabına ulaşılamaz. Son adımdan sonra, hesap açma işlemi sona erer ve kullanıcının BlockSignal'a kaydolmak için kullanabileceği bir Blockstack hesabı oluşmuş olur. Artık, kullanıcıların yapması gereken tek işlem BlockSignal'ı yeniden açarak mevcut cihazda daha önce giriş yapılan Blockstack hesapları arasından birini seçmektir.



Şekil 6.1: Blockstack hesabı olmayan kullanıcılar için BlockSignal'a ilk kayıt

Blockstack oturum açma uygulaması için, Blockstack tarafından sağlanan oturum açma web uygulaması kullanılmış ve bu uygulama, bir websitesinde çalışır hale getirilmiştir[3]. Blockstack Android SDK'nın yardımıyla, BlockSignal'ı çalıştırdığında kullanıcı oturum açma uygulamasına yönlendirilir. Oturum açma uygulaması, Blockstack Browser'da varsayılan olarak çalıştığından, bu işlem güvenli bir şekilde gerçekleştirilebilir.

Oturum açıldıktan sonra, kullanıcının Signal açık anahtarı, kimlik doğrulamada kullanılabilmesi amacıyla arka planda Signal ve Blockstack arasında bir güven zinciri oluşturmak üzere kullanıcının Blockstack gizli anahtarıyla imzalanarak Gaia hubına yazılır. Hubdaki açık anahtarın konumu varsayılan olarak "app.key" şeklinde belirlenmiştir. Dolayısıyla, aynı kullanıcı BlockSignal'a her kaydolduğunda, açık anahtarı aynı dosyaya ve eski anahtarın üzerine yazılır. Ardından, Signal'da olduğu gibi SMS doğrulama, Twilio adı verilen üçüncü taraf bir bulut iletişim platformu kullanılarak gerçekleşir. Bununla birlikte, Blockstack, e-posta adresi ve sosyal medya hesapları gibi kullanıcıları tanımlamak için yeterince güvenilir olmayan bilgileri kullandığından,

Blockstack kimlik doğrulaması sırasında da telefon numarasının kullanılması gerekli görülmüştür. Bu nedenle, kullanıcı telefon numarasını SMS doğrulaması için ekrana yazdıktan sonra, telefon numarası kullanıcının Gaia hubına yazılır ve ardından, kullanıcının sahip olduğu Gaia hubının URL bilgisi, sunucuya gönderilir. Bu URL'yi kullanarak, Signal Server telefon numarasını hubdan alır. Bu adımlar, sunucunun Blockstack hesabının sahibinin telefonun sahibiyle aynı olduğunu ispatlamasına yardımcı olur çünkü sahibi dışında hiçbir kullanıcının bir Gaia hubına yazma yetkisi yoktur. Diğer taraftan, telefon numarası için de sabit bir dosya konumu tanımlamak adına, kullanıcının Gaia hubında adı "phone.number" olarak belirlenen bir dosya kullanılmaktadır. Ardından, Signal Server Twilio'ya istemciye SMS yoluyla bir doğrulama kodu göndermesi için bir istek gönderir. Kullanıcı, SMS doğrulaması sırasında gönderilen kodu doğru bir şekilde girdiği takdirde kayıt işlemi sona erer. Kayıt işleminin BlockSignal ve Signal'daki işlem basamakları Şekil 6.3'te gösterilmiştir.

Signal kullanıcıları, telefon numaralarını doğrudan Signal Sunucusuna gönderirken, BlockSignal için bunun iyi bir fikir olmadığına kanaat getirilmiştir. Çünkü, bir saldırgan, bir kullanıcının telefon numarasını Gaia hubına yazdıktan sonra kendi numarasıyla değiştirebilir, bu da saldırganın kurbanın Blockstack hesabını BlockSignal'a kaydolmak için kullanmasını sağlar. Bu güvenlik açığını önlemek için, sunucunun işlevselliğini sağlayan Open Whisper Systems tarafından geliştirilen "libsignal-service" kütüphanesinde [11] bazı değişiklikler yapılmıştır [15]. Bu kütüphanede, "SignalServiceAccountManager" adı verilen ve sunucunun bir kullanıcıyı kaydetmesini sağlayan bir sınıf bulunmaktadır. Bu sınıf, ayrıca, yapıcıda karşılık gelen giriş parametresiyle ilklendirilen, kullanıcının e.164 formatında biçimlendirilmiş telefon numarası için bir değişkene sahiptir. Sunucunun telefon numarasını Gaia hubdan çekebilmesi için kullanıcının Gaia hub URL'sini de giriş parametresi olarak gerektiren yeni bir yapıcı tanımlanmıştır. Dahası, telefon numarasını Gaia hubdan alarak ilgili sınıf değişkenini yapıcıda belirlemek için gerekli kodlar yeni yapıcıya eklenmiştir. Böylece, telefon numarasının elde edilmesi işlemi kütüphanedeki tüm bağımlı sınıfları değiştirmek zorunda kalmadan gerçekleştirilebilmiştir.

6.3 Kimlik Doğrulama Seremonisi

Önceki bölümlerde, bu çalışmanın amacının, kullanıcıların güvenlik farkındalığının mesajlaşma güvenliğini etkilemeyeceği bir otomatik kimlik doğrulama mekanizması elde etmek olduğu belirtilmişti. Bu alt bölüm, bu amaca nasıl ulaşılabileceğini göstermektedir.

Signal, insan faktörünü de barındıran bir kimlik doğrulama seremonisi mekanizmasına sahipken; BlockSignal, Blockstack'i kimlik sağlayıcı olarak kullanarak otomatik bir kimlik doğrulama sağlar. Daha net ifade etmek gerekirse, Signal'da olduğu gibi

emniyet numaralarını karşılaştırmak yerine, BlockSignal arka planda alıcının Signal açık anahtarını ve telefon numarasını Blockstack vasıtasıyla doğrular ve doğrulama başarısız olursa oluşturulan oturum sonlandırılır. Şekil 6.4'te, BlockSignal ile Signal'ın kimlik doğrulama mekanizmaları arasındaki farklar açıkça görülebilir. Signal uygulamasında, Signal Server veritabanı, her kullanıcı için doğrulanmış durum bilgilerini depolar. Örneğin, Bora Signal'ı yeniden yükleyip kaydolduğunda, Signal Server Bora'nın doğrulanmış olan durumunu değiştirir. Yani, eğer Ayşe Bora'ya bir mesaj gönderirse, sunucu Bora'nın kimliği için mevcut durumu Ayşe'ye gönderir. Bu durum "doğrulandı" değilse, Ayşe, Bora ile bir kimlik doğrulama seremonisini tamamlaması için Bora'nın emniyet numarasının değiştiğine dair uyarılır. Öte yandan, BlockSignal ise, otomatik kimlik doğrulamayı başlatmak için Signal Server'ın kimlik durum güncellemelerini kullanmasına rağmen, Ayşe'yi Signal'da olduğu gibi bir açık anahtar doğrulaması yapması konusunda uyarmaz. Bunun yerine, Bora'nın kimliğini otomatik olarak doğrulamak için erişime açık olan Bora'nın Gaia hubını kullanır. Blockstack, kullanıcıların Gaia hubında depolanan dosyaların varsayılan olarak imzalı tutulmasını destekler. Bu durum, aslında, Gaia hubda tutulan imzalı her dosyanın aynı zamanda Blockstack tarafından oluşturulan bir imza dosyasına sahip olduğu ve Blockstack API'si vasıtasıyla otomatik olarak doğrulanabildiği anlamına gelir. Doğrulama sırasında Ayşe, Bora'nın Signal açık anahtarını içeren "app.key" ve Bora'nın Blockstack açık anahtarını kullanarak, Bora'nın Signal açık anahtarını içeren "app.key" dosyasının imzasını doğrular. Doğrulama başarısız olursa, iletişimin artık güvende olmadığını belirten bir uyarı diyalogu ekranda belirir. Kullanıcı iletişim kutusundaki Tamam butonuna tıkladığında görüşme sona erer ve uygulama, görüşme listesini gösteren ekrana geri döner. Ayrıca, bu durumda, doğrulama durumu için ilgili veritabanı kaydı "doğrulanmamış" olarak değiştirilir. Diğer tarafta, imzalar doğrulanırsa doğrulama durumu "doğrulandı" olarak işaretlenir. Ancak, bu durumda bir anormallik olmadığı için kullanıcılar bu konuda bilgilendirilmez.

6.4 Tekrar Kayıt Olma

BlockSignal kullanıcıları, farklı sebeplerden ötürü uygulamaya yeniden kaydolmak zorunda kalabilir ve kayıt işleminin nedenine göre adımlar farklılık gösterir. Bu nedenler, iki gruba ayrılabilir. Birinci grup, cihaz değişikliği gibi kullanıcının kimlik anahtar çiftinin tekrar üretilmesini gerektiren durumları içerir. İkinci grup ise, SIM kart değişikliği ve uygulamanın yeniden kurulması gibi gizli anahtarın korunduğu olasılıklarını içerir. Signal'da, her iki grup için de kayıt basamakları aynı olsa da, BlockSignal'da durum böyle değildir.

İlk olasılık grubu için, Blockstack oturumunun açılması kullanıcının cihazında ilk kez gerçekleştirileceği için (daha önce hiç kimse aynı cihazla oturum açmamış olarak ka-

bul edilirse), Blockstack için anahtar çiftleri oluşturulmalı ve cihazda saklanmalıdır. Bu nedenle, kullanıcıdan anahtarların üretimi için kullanılan "Magic Recovery Code" veya "Secret Recovery Key" girmesi istenir. Kullanıcı bunlardan birini girdikten sonra, Şekil 6.5'te gösterildiği gibi sırasıyla e-posta adresi ve parolasını girmesi beklenir. Blockstack kullanıcıların e-posta adreslerini tutmadığından girilen adres ilk kayıt esnasında girilen adresten farklı olabilir. Bu adımlar tamamlandıktan sonra, kullanıcının gizli anahtarı üretilir ve cihaza kaydedilir. İlk olasılık grubu için bahsedilen adımların tümüne restorasyon seremonisi (restoration ceremony) adı verilmiştir.

Birinci gruptan farklı olarak, diğer olasılıklarda, kullanıcıların Blockstack anahtar çifti korunduğundan uygulama çalıştırıldığında daha önce oturum açmış oldukları mevcut Blockstack hesapları görünecektir. Bu nedenle, kullanıcılar listeden bu hesaplardan birini seçebilir ve kaydın bir sonraki adımına geçebilirler. İki grup olasılık arasında Blockstack oturumunun açılması haricinde herhangi bir fark yoktur. Bir sonraki adım olan SMS doğrulaması, iki olasılık grubu için de önceki bölümlerde detayları verildiği üzere aynı şekilde gerçekleşir.

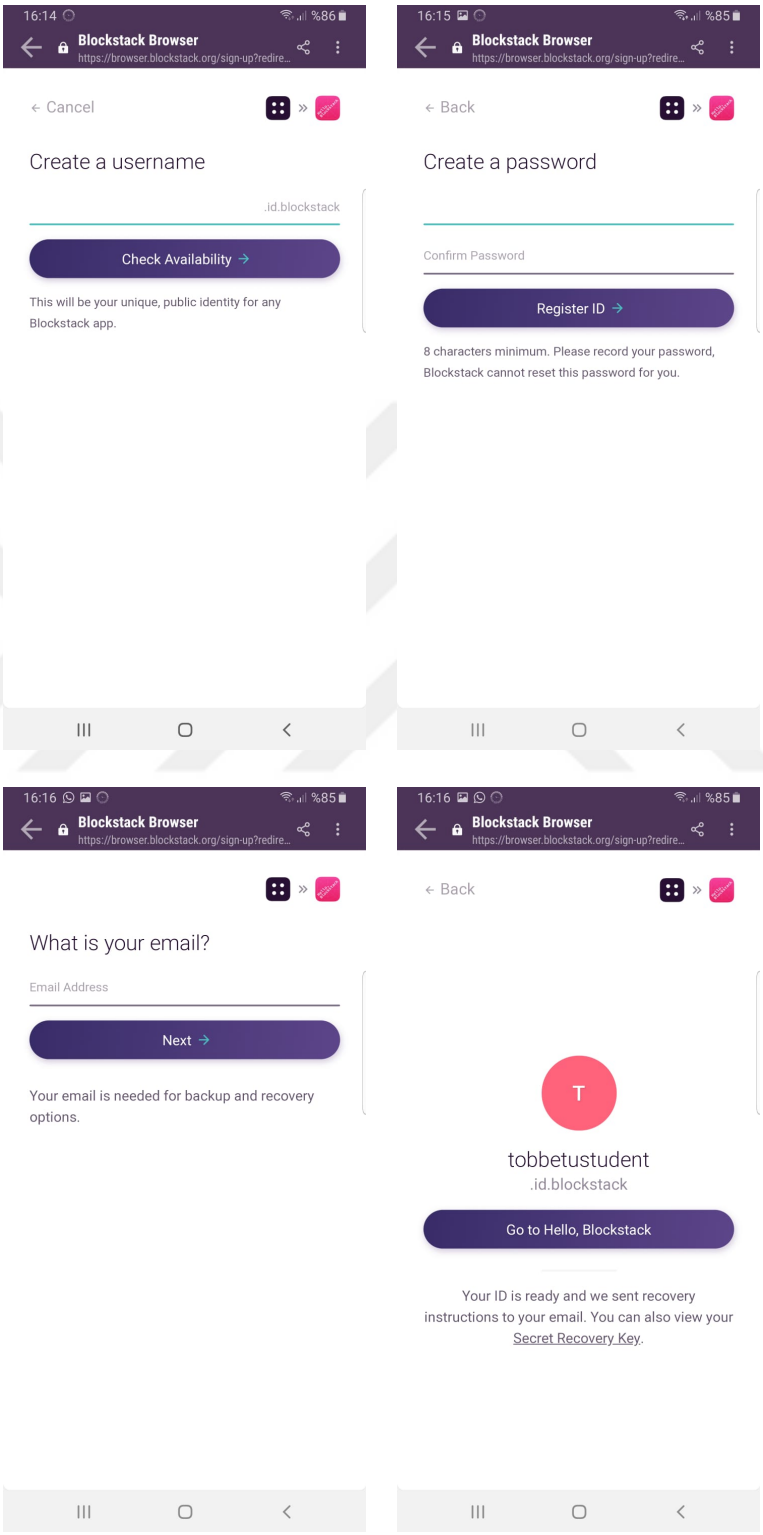
6.5 Kimlik Yönetimi

Blockstack ve Signal olmak üzere iki temel bileşene sahip olan BlockSignal uygulamasında, kullanıcıların iki bileşenden gelen birer kimliği bulunur. Bu kimliklerden kullanıcının telefon numarasını esas alan Signal kimliği kullanıcıyı tanımlarken açık anahtarını esas alan Blockstack kimliği ise kimlik doğrulaması için kullanılır. BlockSignal'ın güvenliği açısından, bu birbirlerinden bağımsız iki kimliğin bir şekilde bir araya getirilerek bir güven ağının oluşturulması oldukça önemlidir. Bu sebeple, BlockSignal'ın tasarımında, bu güven ağının sağlanmasına katkıda bulunabilecek seçimlerin yapılmasına dikkat edilmiştir.

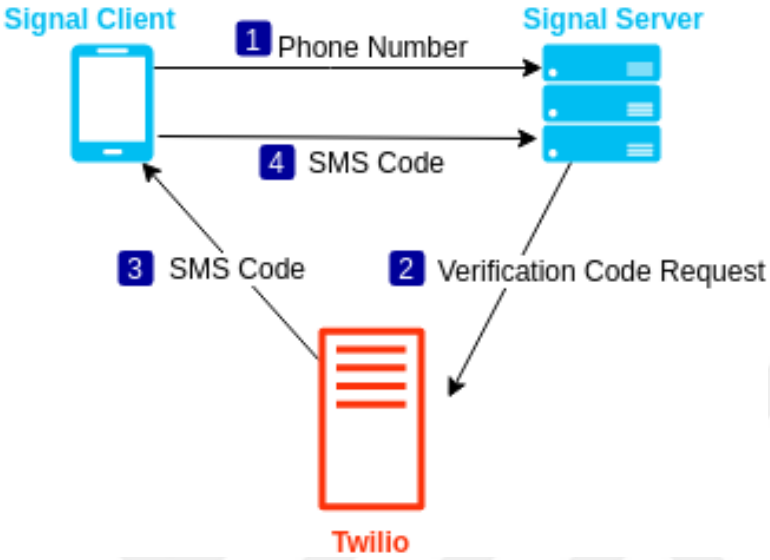
İlk olarak, Blockstack kullanıcı adları, varsayılan olarak bir açık anahtara sahip olduklarından ve bu kullanıcı adları ile erişime açık kullanıcı bilgileri ile Gaia ortamındaki verilerine ulaşılabileceğinden kullanıcıların tanımlanmasında kullanılabilecek akılda kalıcı bir bilgidir. Bu sebeple, iletişimde olduğu bir kullanıcının kimliğini doğrulamak isteyen bir kullanıcının Blockstack açık anahtarı veya Blockstack kimlik numarası gibi uzun ve karmaşık karakter dizileri yerine kullanıcı adına yönelmesi olağandır. Bu olağan tercih BlockSignal tasarımına yansıtılarak Signal uygulamasında kullanıcıların tercihine bırakılan kullanıcı adı bilgisi yerine; varsayılan olarak Blockstack kullanıcı adları, BlockSignal kullanıcı adları olarak kullanılmıştır.

Diğer taraftan, telefon numaralarını kullanıcıların tanımlanmasında kullanan Signal, SMS doğrulaması vasıtasıyla sanal ortam ile fiziksel ortam arasında bir bağ kurarak daha güvenli ve gerçekçi bir kimlik doğrulama sunarken Blockstack'te kimlik doğru-

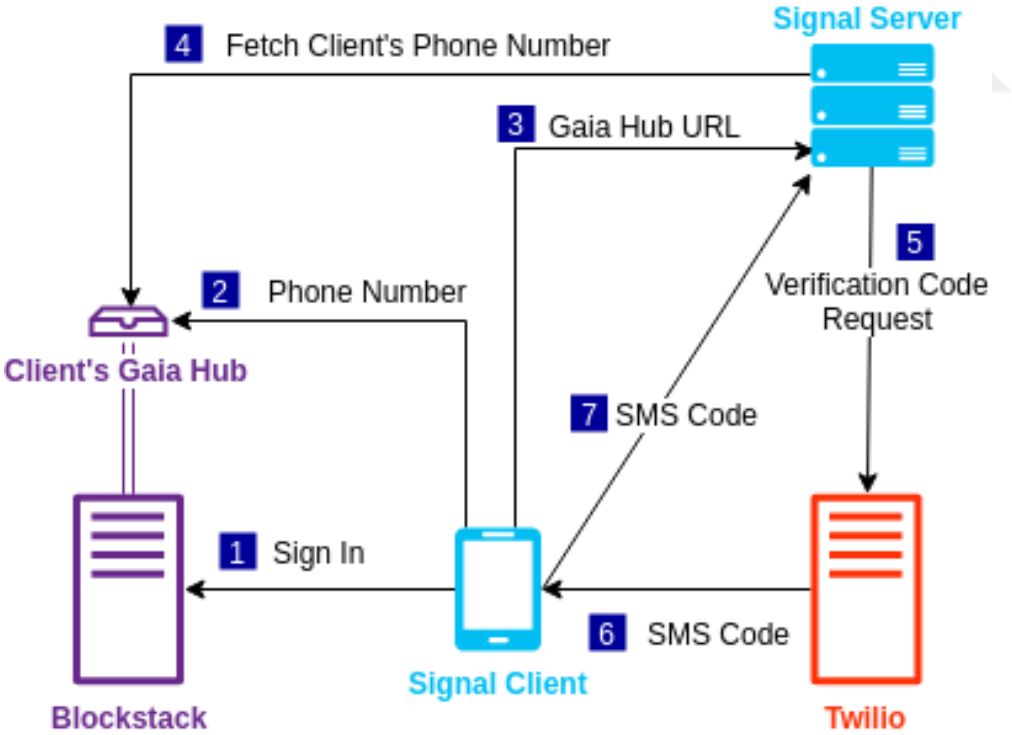
lama tamamen sanal ortamda gerçekleşen bir işlemdir. Bu yüzden, BlockSignal uygulamasının en az Signal kadar güvenli ve gerçekçi bir kimlik yönetimi vaat edebilmesi amacıyla SMS doğrulama esnasında kullanıcı tarafından girilen telefon numarasının doğrudan sunucuya gönderilmesi yerine, kullanıcının Gaia ortamına yazılarak sunucunun numarayı Gaia'dan okuması sağlanmıştır. Böylelikle, doğrulanacak olan telefon numarası ile ilk aşamada kimlik doğrulaması gerçekleşmiş olan Blockstack hesabının aynı kullanıcıya ait olduğu doğrulanabilmiştir. Sonuç olarak, kullanıcıların Blockstack ve Signal hesaplarının eşleştirilmesinde telefon numaraları köprü görevi görmektedir.



Şekil 6.2: BlockSignal üzerinden yeni bir Blockstack hesabı açma

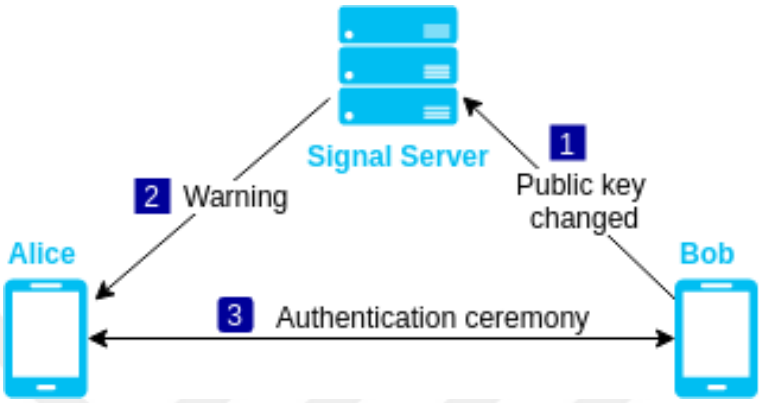


(a) Signal

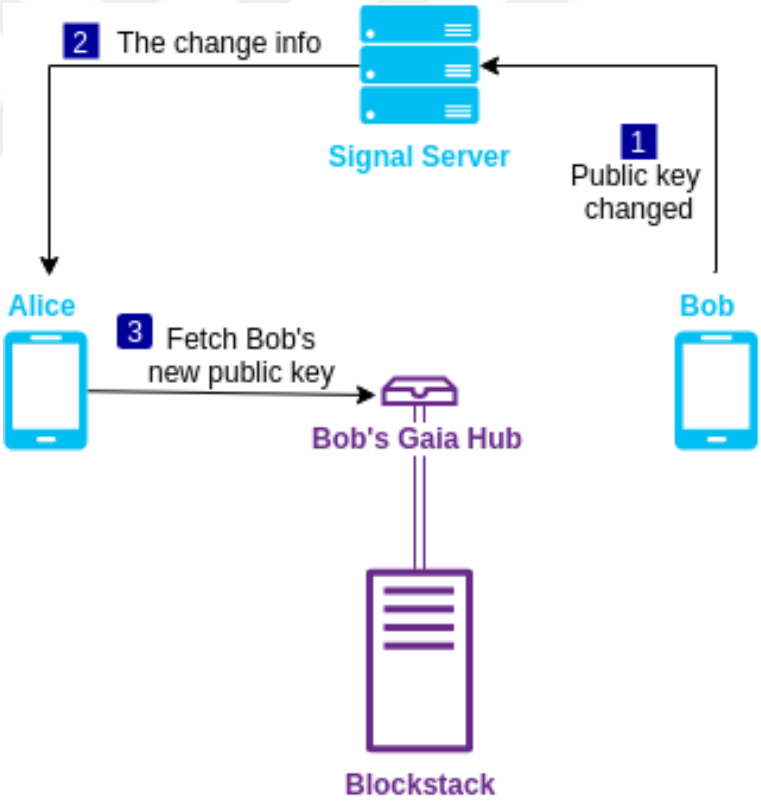


(b) BlockSignal

Şekil 6.3: Signal ve BlockSignal'da kayıt işlemi

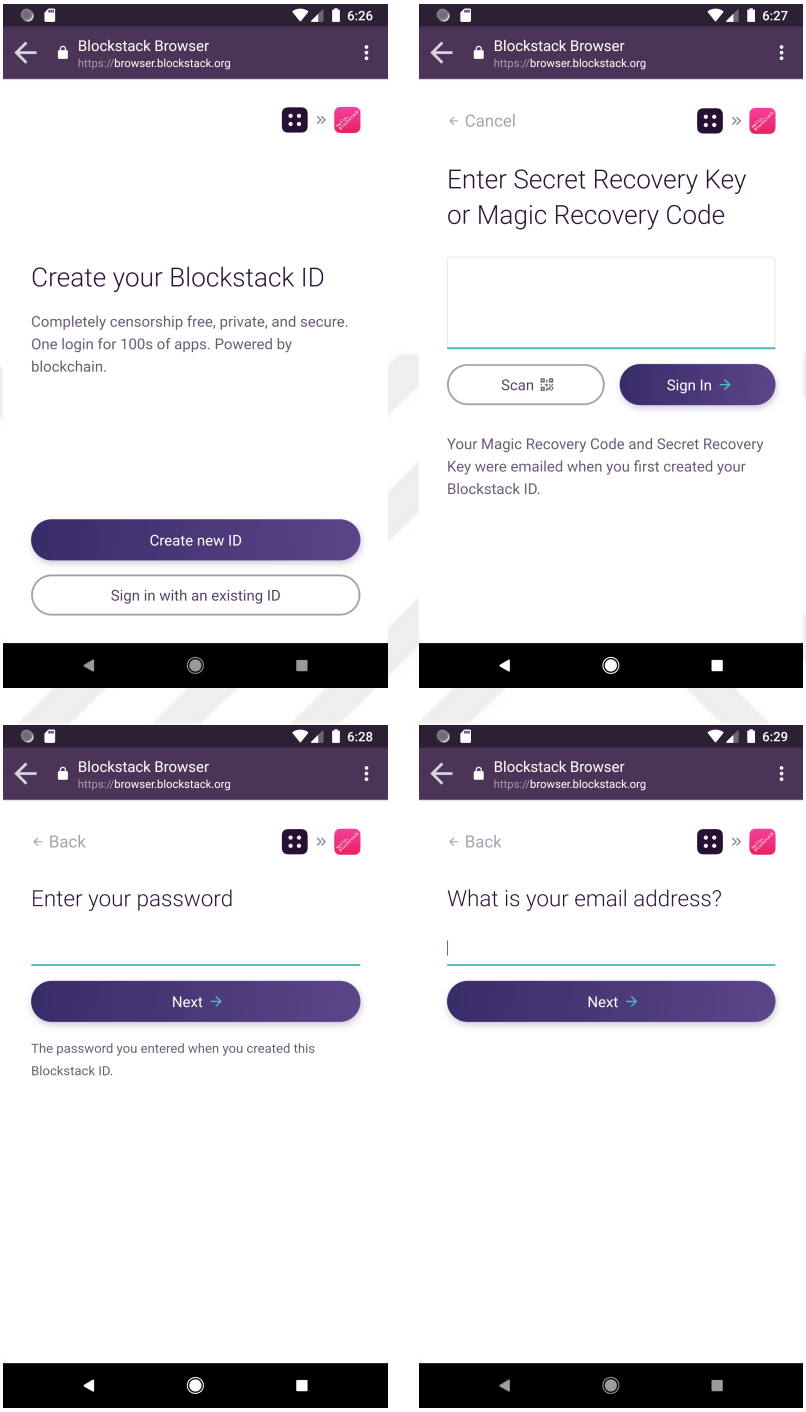


(a) Signal

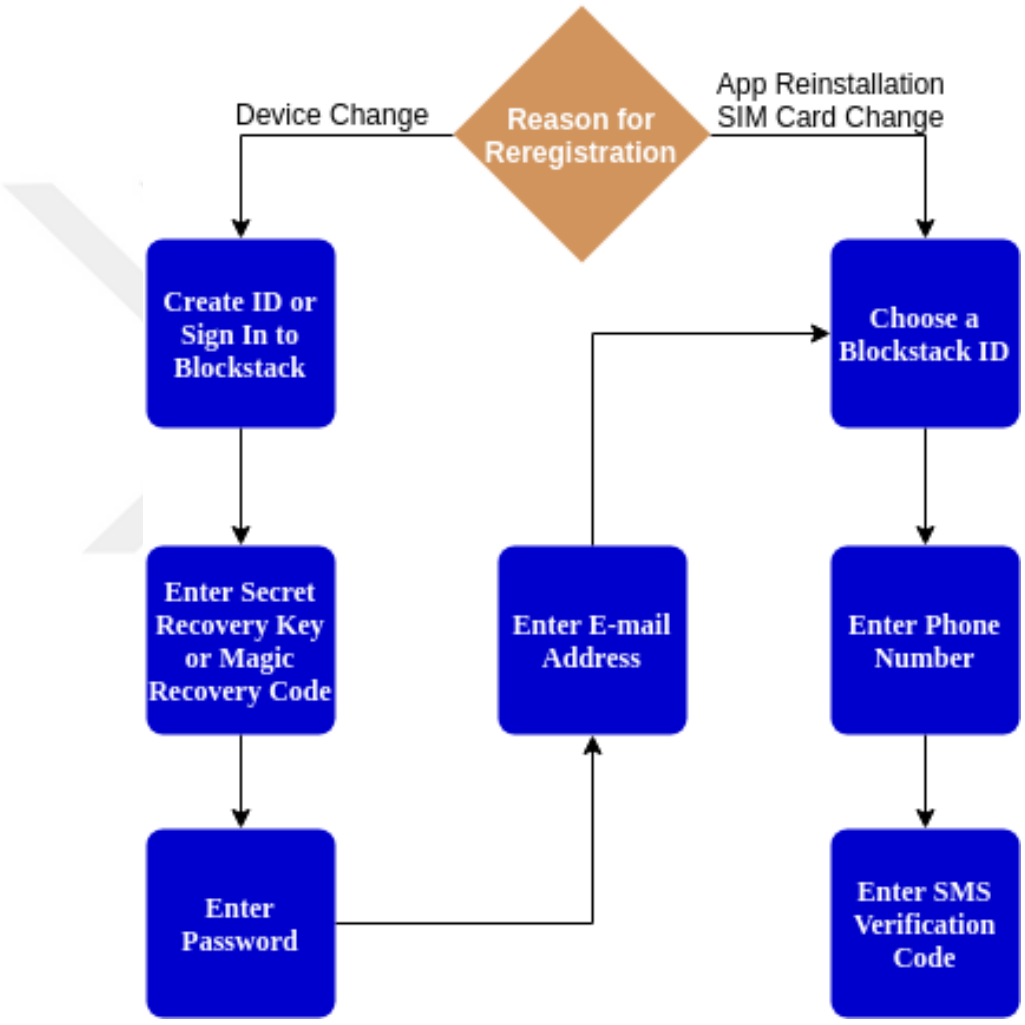


(b) BlockSignal

Şekil 6.4: Signal ve BlockSignal'da kimlik doğrulama mekanizmaları



Şekil 6.5: BlockSignal üzerinden Blockstack hesabının restorasyonu



Şekil 6.6: Mevcut Blockstack hesabıyla BlockSignal'a tekrar kayıt olma

7. GÜVENLİK ANALİZİ

Tehdit modelinde belirtilen tehditlerin ortak noktalarından biri önlenebilmeleri için Signal uygulamasında kullanıcılar tarafından gerekli aksiyonun alınmasının gerekli oluşudur. Diğer taraftan, güvenlik zincirinin tespit edilen tehditleri etkisiz hale getirme noktasında gerçekten sağlam olmasını sağlayabilmek için kullanıcıların güvenlik çemberinden çıkarılması gerekmektedir. Bu nedenle, Google'ın ortaya koyduğu "Kötü olma (Do not be evil)" [51] anlayışı yerine "kötü olamama (Cannot be evil)" yaklaşımına ihtiyaç duyulmaktadır. Sonuç olarak, bu çalışmada, tespit edildiği kadarıyla Signal uygulamasının en problemli kısmı olan kimlik doğrulama seremonisi yerine daha otomatize bir çözüm geliştirmeye odaklanılmıştır.

Daha ayrıntılı olarak, Blockstack vasıtasıyla Bölüm 5'de belirtilen her bir tehdidin aşağıdaki şekilde çözülmesi hedeflenmiştir:

S1) Kötü amaçlı/Ele geçirilmiş sunucu. Bu durumda, Blockstack bir kimlik sağlayıcı olarak kullanılabilir, böylece kullanıcılar, sunucu kötü niyetli veya ele geçirilmiş olsa bile kimliklerini doğrulayabilir. Bu yöntem, sunucuyla ilgili tehditleri ortadan kaldırabilir ve saldırının zamanına göre aşağıdaki şekilde saldırıları engelleyebilir:

- *İlk kurulum esnasındaki saldırılar:* Ayşe ve Bora'nın aralarında bir bağlantı kurmak istediğini ve Oktay'ın bu sohbeti durdurmak istediğini düşünün. Kötü niyetli veya ele geçirilmiş sunucu, Oktay'ın kimlik anahtarını Bora'nın kimlik anahtarı olarak ilan ederse Bora'nın Blockstack'te Blockstack gizli anahtarıyla imzalanmış olarak depolanan Signal açık anahtarı ile Oktay'ın anahtarı eşleşmeyecektir.

- *Oturuma yapılan saldırılar:* Bu tür saldırılar için aynı senaryo yine geçerlidir. Bununla birlikte, bu sefer, Signal doğrulama mekanizmasını tetikler, böylece Blockstack üzerinden alıcının mevcut kullanıcı adı ve açık anahtarıyla doğrulama işlemi gerçekleştirilir. Blockstack, uygulamanın yeniden yüklenmesi ve cihaz değişikliği gibi tehdit oluşturmayan bir açık anahtar değişikliği durumunda yeniden Blockstack hesabına giriş gerektirdiğinden, kullanıcının Gaia hubında depolanan uygulama içi kimlik anahtarı güncellenir ve yeni kullanıcı adı-açık anahtar çifti doğrulanabilir. Öte yandan, bir MITM saldırısı meydana geldiğinde kullanıcının Gaia hubında hala eski açık anahtar saklandığından yeni kullanıcı adı-açık anahtar çifti Blockstack tarafından doğrulanamaz.

S2) SSL sabitlemeyi atlatma. Bu senaryo, sonuçları açısından S1'e benzer. Saldırganın SSL sertifika sabitlemesini atlattığını ve kendi sunucusunu Signal Sunucusunun

yerine koyduğunu varsayalım. Bu sahte sunucu, ele geçirilmiş bir sunucudan farklı değildir. S1'deki gibi, sahte sunucunun dağıttığı saldırganın Signal açık anahtarı, alıcının Gaia hubından alınan anahtarla eşleşmeyecektir.

S3) SIM kart klonlama. Saldırgan, BlockSignal uygulamasına kaydolmak için klonlanmış bir SIM kart kullandığında, öncelikle kendi Blockstack hesabına giriş yapması gerekir. Saldırganın kurbanın Blockstack hesabını da çalmadığını varsayarak, saldırganın kayda devam etmek için başka bir hesapla oturum açması gerekir. Bununla birlikte, yeni tasarım, uygulamada kullanıcı adı olarak Blockstack kullanıcı adını kullandığından, başka bir hesapta oturum açmak, alıcıların kurbanınkinden farklı bir kullanıcı adı göreceği anlamına gelir. Ayrıca, klonlama mevcut bir oturumda gerçekleşirse, saldırganın açık anahtarı mağdurunkiyle eşleşmeyecektir.

S4) Bilinmeyen anahtar paylaşım saldırıları. Signal'da bilinmeyen anahtar paylaşım saldırısı gerçekleştirmek için, taraflardan birinin, saldırganın sağlamış olduğu bir başka kullanıcıya ait açık anahtarını doğrulaması gerekir [35]. Bununla birlikte, BlockSignal, saldırganın bir kullanıcıyı yanlış yönlendirebileceği bir kimlik doğrulama seremonisi mekanizmasına sahip değildir. Dolayısıyla, açık anahtar doğrulaması BlockSignal'da Blockstack kullanılarak otomatik olarak gerçekleştiği için bu tür saldırılar gerçekleştirilemez.

S5) Blockstack ile ilgili tehditler. Signal uygulaması biraz incelendiğinde, bir kullanıcı hakkındaki tek güvenilir bilginin SMS ile kayıt esnasında doğrulandığından telefon numarası olduğu görülebilir. Bu nedenle, Blockstack'in kimlik doğrulama akışı ve kullanıcıları tanımlama biçimi, kullanıcıları doğru bir şekilde tanımlayabilecekler telefon numarasını da içerecek şekilde güncellenerek kullanıcıların Blockstack hesabı ile telefon numaralarının eşleştirilmesi sağlanmıştır.

Geliştirilen BlockSignal uygulaması incelendiğinde, Blockstack'in teorik altyapısının akla ilk gelebilecek birçok güvenlik açığını önlediği görülse de platformun ve kütüphanelerinin gerçeklemesi ve kullanıcı dostu hale getirilmesi sürecinde yeni güvenlik açıklarının ortaya çıkabileceği düşünülebilir. BlockSignal, Blockstack'i yalnızca kimlik doğrulama ve güvenli veri depolama (açık anahtar vs.) amacıyla kullanır. Bölüm 4.3'te anlatıldığı üzere, Blockstack, kimlik doğrulama işleminde uzak sunucu kullanmadığından ve kimlik doğrulamayı tamamen istemci tarafında ve Blockstack tarayıcısı vasıtasıyla gerçekleştirdiğinden oldukça güvenlidir. Ayrıca, cihaz değişikliği durumunda kimlik doğrulama için kullandığı Magic Recovery Code ve Secret Recovery Key bilgilerini ve Magic Recovery Code bilgisinin gönderildiği e-posta adresini saklamadığından bu bilgiler vasıtasıyla bir saldırı gerçekleştirilebilmesi kullanıcı bu bilgileri güvenli bir şekilde sakladığı sürece mümkün görünmemektedir.

Blockstack'in diğer kullanım amacı olan güvenli veri depolama, Bölüm 5'te anlatıldığı üzere, Blockstack depolama birimi olan Gaia ortamına yalnızca kullanıcıların yazma

izninin bulunması ve verilerin imzalı/şifreli olarak saklanması sebebiyle kullanıcının gizli anahtarına sahip olmadan Blockstack de dahil hiçbir otorite tarafından değiştirilemeyeceği ve silinemeyeceğinden ötürü oldukça güvenli görünmektedir.

S6) Hibrit tehditler. Yukarıda bahsedilen tehditler ayrı ayrı var olduğunda alınabilecek önlemlerden bahsedilmiş olsa da bu tehditlerin birlikte gerçekleşebileceği durumlar da göz önüne alınmalıdır. BlockSignal uygulamasında istemci dışında, Signal sunucusu ve Blockstack olmak üzere iki ana bileşen olduğu düşünüldüğünde bu iki bileşeni de etkileyebilecek bir tehdit BlockSignal'da bir zafiyete sebep olabilir. Blockstack, tasarımı itibariyle dışarıdan gelebilecek saldırılara nispeten kapalı olmasına karşın, güvenlik zincirinin en zayıf halkası olan kullanıcılar, bu sistemin de en zayıf halkası olarak görünmektedir. Örneğin, kullanıcıların Blockstack tarayıcısı veya web uygulaması üzerinden görüntüleyebildikleri ve sıkı korunması gereken "Secret Recovery Key" bilgisinin bir saldırgan tarafından elde edilmesi halinde, kullanıcının Blockstack hesabına erişim sağlamış olur. Bu durumda, hem kurban hem de saldırganın kurbanın Blockstack hesabındaki Gaia ortamına yazma hakkı olmuş olur. Dolayısıyla, Bölüm 5'te T1'de anlatılan şekilde Signal sunucusu ele geçirilirse kurbanın Gaia ortamındaki telefon numarası ve açık anahtar bilgisi değiştirilerek (yenisi üzerine yazılarak) bir MITM saldırısı gerçekleşebilir. Bu durumda, kurban, Gaia ortamındaki dosyaları ve özetlerini kontrol ederek veya blokzincire eklenen ve kendi açık anahtarını bulunduran yeni kayıtların olup olmadığını sorgulayarak Blockstack hesabının ele geçirildiğini anlayabilir ve yeni bir hesap açarak BlockSignal kaydını bu hesap üzerinden yapabilir. Ancak, saldırının tespiti için sözü edilen bu işlemlerin sıradan bir kullanıcı için kullanılabilirlik açısından yeterince kolay olmadığı göz önünde bulundurulmalıdır. Ayrıca, bu tür saldırıların önlenebilmesi bakımından, ortalama bir kullanıcının sıklıkla cihaz değişikliği yapmadığı düşünüldüğünde, yeni cihazında Blockstack kimliğine giriş yapabilmek için SRK veya MRC-parola çiftinden en az birini güvenli saklaması gerekliliğinin kullanılabilirlik açısından bir dezavantaj oluşturduğu söylenebilir.

Diğer taraftan, kullanıcılara Blockstack platformuna kayıtları esnasında gönderilen "Magic Recovery Code" bilgisinin Blockstack e-posta sunucusu tarafından gönderildiği düşünüldüğünde, bu kod kullanıcı parolası olmadan kullanılamasa da bu durumun yol açabileceği zafiyetler de değerlendirilmelidir. Blockstack e-posta sunucusu yalnızca MRC bilgisini kullanıcılara göndermek için kullanıldığından ele geçirildiği takdirde saldırganın yapabileceği saldırı gönderilen MRC bilgisini değiştirmek veya bu kodun gönderilmesini engellemek olabilir. Bu durumda, saldırgan kullanıcının MRC bilgisine sahipse kullanıcının Blockstack kimliğine erişim sağlamak için parolayı da çözmesi gerekir. Diğer taraftan, saldırgan MRC bilgisinin gönderilmesini engellerse kullanıcı SRK bilgisiyle giriş yapmaya devam edebilir.

Saldırganın MRC ile birlikte kullanılan parolayı da bir şekilde çözdüğü varsayıldığında, saldırgan, kullanıcının Gaia ortamında sakladığı verileri görebilir ve üzerine

kendi verilerini yazabilir. BlockSignal özelinde, ulaşabileceği veriler kullanıcının Signal açık anahtarı ve telefon numarası ile sınırlıdır. Saldırgan bu bilgilerin üzerine başka veriler yazarak kullanıcının açık anahtarının iletişimde olduğu bir başka kullanıcı tarafından doğrulanmasını engelleyerek kullanıcının iletişimini kısıtlayabilir. Bunun yanı sıra, saldırgan, Blockstack e-posta sunucusu ile birlikte Signal sunucusunu da ele geçirirse kullanıcının Blockstack parolasını da çözdüğü varsayımıyla kullanıcının Gaia ortamında depolanan açık anahtar ve telefon numarası bilgilerini kendi verileriyle değiştirebilir. Bu sayede, kullanıcı bir başkasıyla mesajlaşırken saldırgan Signal sunucusu üzerinden MITM saldırısı gerçekleştirerek diğer kullanıcı tarafından kendi açık anahtarının Blockstack vasıtasıyla doğrulanmasını sağlayabilir.

Blockstack ve Signal bileşenleri kullanılarak yapılabilecek ortak bir saldırının bir başka çıkış noktası olarak BlockSignal'daki kullanıcı adları gösterilebilir. BlockSignal uygulaması, kullanıcıların tanımlanabilmesi için Signal'da olduğu gibi telefon numaralarını kullanmasına karşın, Blockstack kullanıcı adı, kullanıcı adı bilinen bir kullanıcının Blockstack açık anahtarı ve Gaia ortamındaki veriler gibi erişime açık bilgilerine ulaşabilmek için gereklidir. BlockSignal tasarımında, bu kullanıcı adlarının kullanıcılar arasında gönderimi yerine, BlockSignal kullanıcı adı olarak kullanılmasının daha doğru olduğu düşünülmüştür. Bu minvalde, kullanıcı BlockSignal'a kayıt olduğunda kayıt esnasında giriş yaptığı Blockstack hesabına ait kullanıcı adı BlockSignal kullanıcı adı olarak belirlenir. Bu durumda, kullanıcı yeni bir Blockstack hesabı açarak BlockSignal uygulamasına bu hesap üzerinden kayıt olursa BlockSignal kullanıcı adı yeni açtığı Blockstack hesabının kullanıcı adı olur. Bu durum göz önüne alındığında, Signal sunucusunun kendine ait bir Blockstack kimliğini kullanarak kayıtlı bir kullanıcının kullanıcı adını değiştirebileceği ve bu şekilde oturumda bulunan diğer kullanıcıyı aldatarak bir araya girme saldırısı yapabileceği düşünülebilir. Ancak, Signal sunucusu, kullanıcıların yalnızca açık anahtar ve telefon numaralarını depoladığından ve kullanıcıların Blockstack kullanıcı adları lokal olarak cihazlarında saklandığından böyle bir saldırı ihtimali mümkün görünmemektedir. Böyle bir durumda, diğer kullanıcının cihazında saklanan alıcı veritabanında (recipient database) kurbanın asıl Blockstack kullanıcı adı olacağından diğer kullanıcı doğrulamayı yine kurbanın asıl Blockstack kullanıcı adı üzerinden yapacaktır. Dolayısıyla, sunucuya ait Blockstack hesabı vasıtasıyla ulaşılan Signal açık anahtarı ile kurbanın asıl Signal açık anahtarı eşleşmeyecektir. Sonuç olarak, anahtarlar eşleşmediğinden BlockSignal, doğrulamayı yapan kullanıcıyı bilgilendirerek oturumu sonlandırır.

8. SONUÇ VE ÖNERİLER

Günümüzde anlık mesajlaşma uygulamaları akıllı telefonlarda en çok kullanılan uygulama türlerinden biri haline gelmiştir. Bu tür uygulamalarda, uçtan uca şifreleme, kullanıcılar arasında güvenli bir iletişim kurmak için vazgeçilmez durumdadır. Diğer taraftan, en üst düzeyde güvenliğin sağlanabilmesi için, insan faktörünün mümkün olduğunca ortadan kaldırıldığı daha iyi bir çözüme ihtiyaç duyulmaktadır. Bu nedenle, bu tezde, blokzincir tabanlı kimlik, kimlik doğrulama ve depolama platformu Blockstack platformundan yararlanan açık anahtar doğrulamasının otomatikleştirildiği bir çözüm önerilmiş ve Signal Android Messenger anlık mesajlaşma uygulamasını temel alan bir açık kaynak kodlu Android uygulaması geliştirilmiştir. Geliştirilen uygulamada kimlik doğrulama seremonisi yerine restorasyon seremonisi adı verilen bir dizi işlem gelse de bu işlemler kullanıcılar tarafından tek taraflı yapıldıklarından daha güvenli bir tasarıma ulaşıldığı söylenebilir. Ayrıca, restorasyon seremonisi yalnızca cihaz değişikliğinde gerekli olan bir işlem olduğundan kimlik doğrulama seremonisine nazaran çok daha az sayıda gerçekleşecek bir dizi işlem olması hasebiyle daha tercih edilebilir görünmektedir. Sonuç olarak, bu uygulama vasıtasıyla anlık mesajlaşma uygulamalarında kimlik doğrulama seremonisi mekanizmasının kullanıcıları güvenlik çemberinden çıkararak otomatize edilebileceği gösterilmiştir.

Bu çalışmada ulaşılan noktanın, bu konunun gelebileceği son nokta olmadığı düşünülmektedir. Bu nedenle, bu alandaki araştırmacıların üzerinde çalışabilecekleri muhtemel hususlar hakkındaki görüşlerin paylaşılmasının isabetli olacağına kanaat getirilmiştir. Öncelikle, bu çalışmada, kullanıcıların telefon numaraları ve şifrelenmiş ve/veya imzalanmış uygulama içi açık anahtarlar gibi bazı bilgilerini saklamak için Blockstack tarafından sağlanan Gaia depolama merkezi kullanılmıştır. Buna karşın, JWT standardı telefon numarasını "talep (claim)" adı verilen bir alan olarak desteklediğinden daha iyi bir yaklaşım olarak telefon numaraları Blockstack kimlik doğrulama belirteçlerinde (authentication token) depolanabilir. Öte yandan, uygulamanın açık anahtarlarının depolandığı sabit bir lokasyon olmalıdır (An itibariyle, Blockstack tarafından planlanmış ancak henüz tamamlanmamıştır.).

Bu çalışmadaki bir başka olası gelişme de, Blockstack özelliklerini daha fazla kullanarak Signal Messenger uygulamasını tamamen ademi merkezi hale getirmek olabilir. Başka bir deyişle, Signal merkezi sunucusunun kimlik, kimlik doğrulama ve depolama yetenekleri sadece Blockstack kullanılarak elde edilebilir. Bu amaçla, Signal sunucusunun kimlik depolama için kullandığı PostgreSQL veritabanı Blockstack platformunun Gaia depolama ortamına taşınabilir. Ayrıca, doğrudan Amazon S3 kullanmak yerine, kullanıcıların Gaia hubları, mesajlaşma esnasında gönderilen ekleri saklamak için kullanılabilir. Ek olarak, gönderilen mesajlar iletilirken yine Gaia ortamında saklanabilir. Sonuç olarak, Blockstack platformunu daha fazla kullanarak Signal Messenger uygu-

lamasını ve daha güvenli hale getirmek mümkün olabilir.

KAYNAKLAR

[1] Bip39 standard. https://github.com/bitcoinjs/bip39. Alındığı tarih: 2019-11-11.

[2] Blocksignal. https://github.com/altuncu/BlockSignal. Alındığı tarih: 2019-11-11.

[3] Blocksignal signin application. https://blocksignal.netlify.com. Alındığı tarih: 2019-11-11.

[4] Blockstack android sdk. https://github.com/blockstack/blockstack-android. Alındığı tarih: 2019-11-11.

[5] Blockstack authentication. https://blockstack.github.io/blockstack.js/#authentication. Alındığı tarih: 2019-04-10.

[6] Naming system feature comparison. https://docs.blockstack.org/core/naming/comparison.html. Alındığı tarih: 2019-11-08.

[7] Open whisper systems. signal. https://signal.org/. Alındığı tarih: 2019-04-22.

[8] Openintents. https://chat.openintents.org. Alındığı tarih: 2019-11-11.

[9] Openssh. https://www.openssh.com. Alındığı tarih: 2019-11-11.

[10] Push server. https://github.com/signalapp/PushServer. Alındığı tarih: 2019-11-11.

[11] Signal libsignal. https://github.com/signalapp/libsignal-service-java. Alındığı tarih: 2019-11-11.

[12] Signal messenger. https://github.com/signalapp/Signal-Android. Alındığı tarih: 2019-11-11.

[13] Signal sunucu. https://github.com/signalapp/Signal-Server. Alındığı tarih: 2019-11-11.

[14] Stealthy. `https://www.stealthy.im`. Alındığı tarih: 2019-11-11.

[15] Yeni libsignal. `https://github.com/altuncu/libsignal-service-java`. Alındığı tarih: 2019-11-11.

[16] **Al-Bassam, Mustafa**. Scpki: a smart contract-based pki and identity system. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pages 35–40. ACM, 2017.

[17] **Al-Fayoumi, Mustafa A and Shilbayeh, Nidal F**. Cloning sim cards usability reduction in mobile networks. *Journal of network and systems management*, 22(2):259–279, 2014.

[18] **Ali, Muneeb and Nelson, Jude and Shea, Ryan and Freedman, Michael J**. Blockstack: A global naming and storage system secured by blockchains. In *2016 {USENIX} Annual Technical Conference ({USENIX}{ATC} 16)*, pages 181–194, 2016.

[19] **Ali, Muneeb and Nelson, Jude and Shea, Ryan and Freedman, Michael J**. Bootstrapping trust in distributed systems with blockchains. *login: USENIX Mag.*, 41(3), 2016.

[20] **Ali, Muneeb and Shea, Ryan and Nelson, Jude and Freedman, Michael J**. Blockstack technical whitepaper, 2017.

[21] **Andzakovic, D**. Bypassing ssl pinning on android via reverse engineering. *Whitepaper, May*, 15:12, 2014.

[22] **Antonopoulos, Andreas M**. *Mastering Bitcoin: Programming the open blockchain*. " O'Reilly Media, Inc.", 2017.

[23] **Anwar, Nuril and Riadi, Imam and Luthfi, Ahmad**. Forensic sim card cloning using authentication algorithm. *International Journal of Electronics and Information Engineering*, 4(2):71–81, 2016.

[24] **Axon, LM and Goldsmith, Michael**. Pb-pki: A privacy-aware blockchain-based pki. 2016.

[25] **Baldi, Marco and Chiaraluce, Franco and Frontoni, Emanuele and Gottardi, Giuseppe and Sciarroni, Daniele and Spalazzi, Luca**. Certificate validation through public ledgers and blockchains. In *ITASEC*, pages 156–165, 2017.

[26] **Bicakci, Kemal and Altuncu, Enes and Sahkulubey, Muhammet Sakir and Kiziloz, Hakan Ezgi and Uzunay, Yusuf**. How safe is safety number? a user

study on signal's fingerprint and safety number methods for public key verification. In *International Conference on Information Security*, pages 85–98. Springer, 2018.

[27] **Blake-Wilson, Simon and Menezes, Alfred**. Unknown key-share attacks on the station-to-station (sts) protocol. In *International Workshop on Public Key Cryptography*, pages 154–170. Springer, 1999.

[28] **Brainard, John and Juels, Ari and Rivest, Ronald L and Szydlo, Michael and Yung, Moti**. Fourth-factor authentication: somebody you know. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 168–178. ACM, 2006.

[29] **Cohn-Gordon, Katriel and Cremers, Cas and Dowling, Benjamin and Garratt, Luke and Stebila, Douglas**. A formal security analysis of the signal messaging protocol. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 451–466. IEEE, 2017.

[30] **Cranor, Lorrie F**. A framework for reasoning about the human in the loop. 2008.

[31] **Diffie, Whitfield and Hellman, Martin**. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[32] **Ellison, Carl and Schneier, Bruce**. Ten risks of pki: What you're not being told about public key infrastructure. *Comput Secur J*, 16(1):1–7, 2000.

[33] **Ellison, Carl M**. Ceremony design and analysis. *IACR Cryptology ePrint Archive*, 2007:399, 2007.

[34] **Fredriksson, Bastian**. A distributed x. 509 public key infrastructure backed by a blockchain. 2017.

[35] **Frosch, Tilman and Mainka, Christian and Bader, Christoph and Bergsma, Florian and Schwenk, Jörg and Holz, Thorsten**. How secure is textsecure? In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 457–472. IEEE, 2016.

[36] **Garfinkel, Simson**. *PGP: pretty good privacy*. " O'Reilly Media, Inc.", 1995.

[37] **Gerber, Nina and Zimmermann, Verena and Henhapl, Birgit and Emeröz, Sinem and Volkamer, Melanie**. Finally johnny can encrypt: But does this make him feel more secure? In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, page 11. ACM, 2018.

[38] **Graves, Jay**. Ssl pinning for increased app security. `https://possiblemobile.com/2013/03/ssl-pinning-for-increased-app-security`. Alındığı tarih: 2019-07-08.

[39] **Hari, Adiseshu and Lakshman, TV**. The internet blockchain: A distributed, tamper-resistant transaction framework for the internet. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 204–210. ACM, 2016.

[40] **He, Sheng and Paar, Ing Christof**. Sim card security. In *Seminar Work, Ruhr-University of Bochum*, 2007.

[41] **Herzberg, Amir and Leibowitz, Hemi**. Can johnny finally encrypt?: evaluating e2e-encryption in popular im applications. In *ACM Workshop on Socio-Technical Aspects in Security and Trust (STAST)*, 2016.

[42] **Hickman, Kipp and Elgamal, Taher**. The ssl protocol. *Netscape communications corp*, 501, 1995.

[43] **Johansen, Christian and Mujaj, Aulon and Arshad, Hamed and Noll, Josef**. The snowden phone: A comparative survey of secure instant messaging mobile applications (authors' version). *arXiv preprint arXiv:1807.07952*, 2018.

[44] **Kobeissi, Nadim and Bhargavan, Karthikeyan and Blanchet, Bruno**. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 435–450. IEEE, 2017.

[45] **Kubilay, Murat Yasin and Kiraz, Mehmet Sabir and Mantar, Hacı Ali**. Certledger: A new pki model with certificate transparency based on blockchain. *Computers & Security*, 2019.

[46] **Moonsamy, Veelasha and Batten, LM**. Mitigating man-in-the-middle attacks on smartphones-a discussion of ssl pinning and dnssec. In *Proceedings of the 12th Australian Information Security Management Conference*, pages 5–13. Edith Cowan University, 2014.

[47] **Moosavi, Seyedehmahsa and Clark, Jeremy**. Ghazal: toward truly authoritative web certificates using ethereum. In *International Conference on Financial Cryptography and Data Security*, pages 352–366. Springer, 2018.

[48] **Papadopoulos, Elias P and Diamantaris, Michalis and Papadopoulos, Panagiotis and Petsas, Thanasis and Ioannidis, Sotiris and Markatos, Evangelos**

P. The long-standing privacy debate: Mobile websites vs mobile apps. In *Proceedings of the 26th International Conference on World Wide Web*, pages 153–162. International World Wide Web Conferences Steering Committee, 2017.

[49] **Perrin, Trevor and Marlinspike, Moxie**. The double ratchet algorithm. `https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf`, 2016. Alındığı tarih: 2019-04-22.

[50] **Perrin, Trevor and Marlinspike, Moxie**. The x3dh key agreement protocol. `https://signal.org/docs/specifications/x3dh/x3dh.pdf`, 2016. Alındığı tarih: 2019-04-22.

[51] **Schmidt, Eric and Varian, Hal**. Google: ten golden rules. *MSNBC NewsWeek. Retrieved from http://www. msnbc. msn. com/id/10296177/site/newsweek*, 2005.

[52] **Schröder, Svenja and Huber, Markus and Wind, David and Rottermanner, Christoph**. When signal hits the fan: On the usability and security of state-of-the-art secure mobile messaging. In *European Workshop on Usable Security. IEEE*, 2016.

[53] **Vaziripour, Elham and Howard, Devon and Tyler, Jake and O'Neill, Mark and Wu, Justin and Seamons, Kent and Zappala, Daniel**. I don't even have to bother them!: Using social media to automate the authentication ceremony in secure messaging. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 93. ACM, 2019.

[54] **Vaziripour, Elham and Wu, Justin and O'Neill, Mark and Metro, Daniel and Cockrell, Josh and Moffett, Timothy and Whitehead, Jordan and Bonner, Nick and Seamons, Kent and Zappala, Daniel**. Action needed! helping users find and complete the authentication ceremony in signal. In *Fourteenth Symposium on Usable Privacy and Security ({SOUPS} 2018)*, pages 47–62, 2018.

[55] **Vaziripour, Elham and Wu, Justin and O'Neill, Mark and Whitehead, Jordan and Heidbrink, Scott and Seamons, Kent and Zappala, Daniel**. Is that you, alice? a usability study of the authentication ceremony of secure messaging applications. In *Thirteenth Symposium on Usable Privacy and Security ({SOUPS} 2017)*, pages 29–47, 2017.

[56] **Walton, Jeffrey and JohnSteven and Manico, Jim and Wall, Kevin and Iramar, Ricardo**. Certificate and public key pinning. `https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning`. Alındığı tarih: 2019-06-11.

EKLER

EK 1 : Örnek PushServer konfigürasyon dosyası

EK 2 : Örnek TextSecure Server konfigürasyon dosyası

EK 3 : SSL sertifikasının oluşturulması için kullanılabilecek betik

EK 1 : Örnek PushServer konfigürasyon dosyası (pushserver.yml)

<> sembolleri arasında gösterilen alanlar gerekli konfigürasyon bilgileriyle değiştirilmelidir.

redis:
url: redis://localhost:6379/2

authentication:
servers:
-
name: 123
password: 123
gcm:
xmpp: false
apiKey: <USER-API-KEY>
senderId: <SENDER-ID>
redphoneApiKey: <USER-API-KEY>

server:
applicationConnectors:
- type: http
port: 9090
adminConnectors:
- type: http
port: 9091

logging:
level: INFO
appenders:
- type: file
currentLogFilename: /tmp/pushserver.log
archivedLogFilenamePattern: /tmp/pushserver-%d.log.gz
archivedFileCount: 5
- type: console

EK 2 : Örnek TextSecure Server konfigürasyon dosyası (textsecure.yml)

<> sembolleri arasında gösterilen alanlar gerekli konfigürasyon bilgileriyle değiştirilmelidir.

twilio: # Twilio ağ geçidi konfigürasyonu
accountId: <TWILIO-ACCOUNT-ID>
accountToken: <TWILIO-ACCOUNT-TOKEN>
numbers: # Twilio'dan tahsis edilen numaralar
- <TWILIODAN-ALINAN-NUMARA> # İlk numara
messagingServicesId:
localDomain: 127.0.0.1 # Twilio'nun tekrar bağlantı kuracağı servisin alan adı.

push:
queueSize: # Gönderim bekleme kuyruğunun boyutu

turn: # TURN sunucu konfigürasyonu
secret: turn:turn # TURN sunucu gizi
uris:
- stun:127.0.0.1:80
- stun:127.0.0.1:443
- turn:127.0.0.1:443?transport=udp
- turn:127.0.0.1:80?transport=udp

messageCache:
cacheRate: 1
redis:
url: redis://localhost:6379/2

cache: # Önbellek öbeği için Redis sunucu konfigürasyonu
url: "redis://localhost:6379/1"

directory: # Dizin öbeği için Redis sunucu konfigürasyonu
url: "redis://localhost:6379/0"

server:
applicationConnectors:
- type: https
port: 8080
keyStorePath: config/example.keystore

```
keyStorePassword: example
validateCerts: false
adminConnectors:
- type: https
port: 8081
keyStorePath: config/example.keystore
keyStorePassword: example
validateCerts: false

messageStore: # Mesaj depolama için Postgresql veritabanı konfigürasyonu
driverClass: org.postgresql.Driver
user: "postgres"
password: "postgres"
url: "jdbc:postgresql://localhost:5432/messagesdb"

attachments: # AWS S3 konfigürasyonu
accessKey: <AWS-ACCESS-KEY>
accessSecret: <AWS-ACCESS-SECRET>
bucket: blocksignal

profiles: # AWS S3 konfigürasyonu
accessKey: <AWS-ACCESS-KEY>
accessSecret: <AWS-ACCESS-SECRET>
bucket: blocksignal
region: "eu-london-1"

database: # Postgresql veritabanı konfigürasyonu
driverClass: org.postgresql.Driver
user: "postgres"
password: "postgres"
url: "jdbc:postgresql://localhost:5432/accountsdb"

gcm: # GCM konfigürasyonu
senderId: <SENDER-ID>
apiKey: <API-KEY>

logging:
level: INFO
appenders:
- type: file
currentLogFilename: /tmp/signalserver.log
```

```
archivedLogFilenamePattern: /tmp/signalserver-%d.log.gz
archivedFileCount: 5
- type: console
```

EK 3 : SSL sertifikasının oluşturulması için kullanılabilecek betik

```
#/bin/bash
# Bu betik, istemci ve sunucu tarafından kullanılmak üzere kök CA ve sunucu sertifikası oluşturur.
# rootCA.crt sistemdeki kök CA kümesini değiştirebilmek için istemciye kopyalanmalıdır.
# example.keystore sunucunun konfigürasyon dosyasında keyStorePath tarafından gösterilmelidir.
#
#DIŞARIDAN ÇALIŞTIRABİLMEK İÇİN:
#ALTNAME=DNS:signal.foo.org ./gencerts
#
#EĞER SAHİP OLUNAN BİR ALAN ADI VARSA VEYA LAN İÇİNDEYSE IP ADRESİ
#ALTNAME=IP:10.1.4.218 ./gencerts
#
#
# Kök CA sertifikası için gizli anahtarın oluşturulması
openssl genrsa -out rootCA.key 4096


# Kendinden imzalı kök CA sertifikasının oluşturulması
openssl req -x509 -new -nodes -days 3650 -out rootCA.crt -key rootCA.key


# Sunucu sertifika anahtarının oluşturulması
openssl genrsa -out whisper.key 4096


# Sertifika imzalama talebinin oluşturulması
openssl req -new -key whisper.key -out whisper.csr


# Sertifikanın kök CA ile imzalanması


openssl x509 -req -in whisper.csr -CA rootCA.crt -CAkey rootCA.key -CAcreateserial
-days 365 -out whisper.crt -extensions extensions -extfile <(cat «-EOF
[extensions]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
```

```
authorityKeyIdentifier=keyid,issuer
subjectAltName=$ALTNAME
EOF
)


# Anahtar ve sertifikanın Java anahtar aracı (keytool) tarafından tanınan PKCS12 formatına dönüştürülmesi
openssl pkcs12 -export -password pass:example -in whisper.crt -inkey whisper.key -out keystore.p12 -name example -CAfile rootCA.crt


# Anahtar ve sertifikanın, dropwizard tarafından kullanılabilmesi için Java keystore formatına aktarılması
keytool -importkeystore -srcstoretype PKCS12 -srckeystore keystore.p12 -srcstorepass example -destkeystore example.keystore -deststorepass example


#whisper.store Android istemci içine yerleştirilmelidir.
keytool -importcert -v -trustcacerts -file whisper.crt -alias IntermediateCA -keystore whisper.store -provider org.bouncycastle.jce.provider.BouncyCastleProvider -providerpath /data/java/Signal_prj/bcprov-jdk15on-154.jar -storetype BKS -storepass whisper
```

ÖZGEÇMİŞ

Ad-Soyad : Enes ALTUNCU

Uyruğu : Türk

Doğum Tarihi ve Yeri : 11.06.1994 - Colchester/İngiltere

E-posta : altuncuenes@hotmail.com

ÖĞRENİM DURUMU:

- **Lisans** : 2017, Orta Doğu Teknik Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü

YABANCI DİL: İngilizce

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- **Altuncu, E.**, Bıçakcı, K., (2019) BLOCKSIGNAL: Blokzincir Kullanarak Kimlik Doğrulama Seremonisini Ortadan Kaldıran Bir Güvenli Mesajlaşma Uygulaması, 2. Ulusal Blokzincir Çalıştayı, Eylül 25-26, İstanbul, Türkiye.

DİĞER YAYINLAR, SUNUMLAR VE PATENTLER:

- Aksu, M.U., **Altuncu, E.**, Bicakci, K. (2019) A First Look at the Usability of OpenVAS Vulnerability Scanner, Workshop of Usable Security and Privacy (USEC '19)

- Bicakci, K., **Altuncu, E.**, Sahkulubey, M. S., Kiziloz, H. E., Uzunay, Y. (2018). How Safe Is Safety Number? A User Study on SIGNAL's Fingerprint and Safety Number Methods for Public Key Verification. In International Conference on Information Security (pp. 85-98). Springer, Cham.
- **Altuncu, E.**, Bilgehan, B. K., Kartal, Y. S., Kızılgüneş, S., Nikoo, M. S., Oğuztüzün, H. (2017). Blocks and text integration in a language-based editor for a domain-specific language. In Computer Science and Engineering (UBMK), 2017 International Conference on (pp. 1084-1089). IEEE.