CARİ MUTABAKAT VE ÖDEME İŞLEMLERİ İÇİN ETHEREUM TABANLI BLOKZİNCİR TEKNOLOJİSİNİN KULLANIMININ ÖNERİLMESİ

Muhammet Ali EFENDİOĞLU

Kütahya Dumlupınar Üniversitesi

Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliği Uyarınca

Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında

YÜKSEK LİSANS TEZİ

Olarak Hazırlanmıştır.

Danışman: Dr. Öğr. Üyesi Muammer Akçay

Eylül - 2020

KABUL VE ONAY SAYFASI

Muhammet Ali EFENDİOĞLU tarafından hazırlanan "CARİ MUTABAKAT VE ÖDEME İŞLEMLERİ İÇİN ETHEREUM TABANLI BLOKZİNCİR TEKNOLOJİSİNİN KULLANIMININ ÖNERİLMESİ" adlı tez çalışması, aşağıda belirtilen jüri tarafından Kütahya Dumlupınar Üniversitesi Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek OY BİRLİĞİ ile Kütahya Dumlupınar Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

15/09/2020

Prof. Dr. Şahmurat ARIK
Enstitü Müdürü, Lisansüstü Eğitim Enstitüsü ……

Doç. Dr. Doğan AYDIN
Anabilim Dalı Başkanı, Bilgisayar Mühendisliği Anabilim Dalı ……

Dr. Öğr. Üyesi Muammer AKÇAY
Danışman, Bilgisayar Mühendisliği Anabilim Dalı ……

Sınav Komitesi Üyeleri

Dr. Öğr. Üyesi Muammer AKÇAY
Bilgisayar Mühendisliği Bölümü, Kütahya Dumlupınar Üniversitesi ……

Prof. Dr. Eyyüp GÜLBANDILAR
Bilgisayar Mühendisliği Bölümü, Eskişehir Osmangazi Üniversitesi ……

Dr. Öğr. Üyesi Durmuş ÖZDEMİR
Bilgisayar Mühendisliği Bölümü, Kütahya Dumlupınar Üniversitesi ……

ETİK İLKE VE KURALLARA UYGUNLUK BEYANI

Bu tezin hazırlanmasında Akademik kurallara riayet ettiğimizi, özgün bir çalışma olduğunu ve yapılan tez çalışmasının bilimsel etik ilke ve kurallara uygun olduğunu, çalışma kapsamında teze ait olmayan veriler için kaynak gösterildiğini ve kaynaklar dizininde belirtildiğini, Yüksek Öğretim Kurulu tarafından kullanılmak üzere önerilen ve Kütahya Dumlupınar Üniversitesi tarafından kullanılan İntihal Programı ile tarandığını ve benzerlik oranının %3 çıktığını beyan ederiz. Aykırı bir durum ortaya çıktığı takdirde tüm hukuki sonuçlara razı olduğumuzu taahhüt ederiz.

Dr. Öğr. Üyesi Muammer AKÇAY Muhammet Ali EFENDİOĞLU

CARİ MUTABAKAT VE ÖDEME İŞLEMLERİ İÇİN ETHEREUM TABANLI BLOKZİNCİR TEKNOLOJİSİNİN KULLANIMININ ÖNERİLMESİ

Muhammet Ali EFENDİOĞLU

Bilgisayar Mühendisliği, Yüksek Lisans Tezi, 2020

Tez Danışmanı: Dr. Öğr. Üyesi Muammer Akçay

ÖZET

Bu tez çalışmasında, işletmelerin dönemsel olarak birbirleri arasında yaptığı cari mutabakatlarda ve sonrasında ortaya çıkan borç veya alacak ödemelerinin tahsilinde yaşanan sorunlara çözüm aranmıştır. Günümüzde cari mutabakatların büyük çoğunluğu E-posta, fax ve telefon ile yapılmaktadır. Bu yöntemler raporlama, kargo ve arşivleme maliyetleri, takip ve geri dönüş gecikmeleri ve ek personel ihtiyacı gibi problemleri beraberinde getirmektedir. Diğer bir mutabakat yöntemi ise E-mutabakat yazılımlarıdır. Yıllık program maliyetlerinin yanı sıra mutabakat verilerini firmaların kendi bünyesinde veya ücret karşılığı üçüncü taraflardan hizmet alarak merkezi sunucularda saklanması verinin gizliliğini, izlenebilirliğini ortadan kaldırarak veriye olan güvenilirliği azaltmaktadır. Diğer bir problem ise mutabakat sonrası bankalar aracılığı ile gerçekleşen para transfer süreçlerinde yaşanan yüksek komisyon ücret kesintileri, banka prosedürleri ve işlemlerin beklenen hızda gerçekleşmemesidir. Bu sorunların çözümü için son yıllarda bilişim alanında ortaya çıkan en önemli yeniliklerden biri olan blokzincir teknolojisi tercih edilmiştir. Çalışma kapsamında Akıllı sözleşme kullanan Ethereum tabanlı DApp uygulaması geliştirilmiştir. Veriler bloklar üzerinde tutularak gerçek zamanlı ve geriye dönük izlenebilirlik sağlanmıştır. sBu şekilde, günümüz mutabakat yöntemlerinden olan E-Mutabakat uygulamalarındaki tek merkeze dayalı güven problemini çözüme kavuşturduğu görülmüştür. Diğer taraftan geleneksel yöntemlerden telefon, fax veya E-Posta yolu ile yapılan mutabakatlardaki yaşanan veri erişim ve raporlama problemlerini, zaman kayıplarını, kargo, arşivleme masraflarını ve ek personel ihtiyacını ortadan kaldırmıştır. Ayrıca mutabakat sonrası işletmeler arası gerçekleşen borç veya alacak ödemelerinin kripto para birimi olan Ether ile uygulama üzerinden gerçekleşmesi sağlanmıştır. Üçüncü taraflara gerek duymadan, gün veya saat kısıtlaması olmadan, düşük ücret kesintisi ile para transferi gerçekleştirilmiştir. Böylelikle para transferlerinde bankaların aracı olarak kullanılması ile yaşanan problemler çözülmüştür.

Anahtar Kelimeler: Blokzincir, Cari Mutabakat, Kripto Para, Ödeme Sistemi, DApp, Ethereum, Dağıtık Sistemler

A SUGGESTION OF THE USING ETHEREUM-BASED BLOCKCHAIN TECHNOLOGY FOR ACCOUNT RECONCILIATION AND PAYMENT TRANSACTIONS

Muhammet Ali EFENDİOĞLU

Computer Engineering, M.S. Thesis, 2020s

Thesis Supervisor: Asst. Prof. Dr. Muammer Akçay

SUMMARY

In this thesis, a solution has occurred sought for the problems encountered in the current account reconciliation between the companies with commercial relations and the collection of debt or credit payments arising after the reconciliation. Today, most of the current reconciliations are made by e-mail, fax and telephone. These methods bring problems such as reporting, shipping and archiving costs, tracking and return delays, access to data and the need for additional personnel. Another method of reconciliation is e-reconciliation software. In addition to the annual program costs, storing the reconciliation data on central servers by receiving services from the companies themselves or third parties for a fee reduces the confidentiality and traceability of the data and reduces the reliability of the data. Another problem is the high commission fees charged in money transfers with banks and bank procedures and slow transactions. Blockchain technology has been preferred for the solution of these problems. Ethereum based decentralized application (DApp) was developed within the study. In this way, it was seen that he solved the single-center based trust problem in E-Reconciliation practices, which is one of today's reconciliation methods. On the other hand, it eliminated the data access and reporting problems, loss of time, cargo, archiving costs, and the need for additional personnel, which are experienced in the reconciliations made by telephone, fax, or E-mail. Also, after the reconciliation, it was ensured that the debt or credit payments between businesses were made via the application with the cryptocurrency Ether. Money transfers were carried without the need for third parties, no matter the day or the hour, with a low fee deduction. Thus, the problems encountered in money transfers made through banks were solved.

Keywords: Blockchain, Account Reconciliation, Cryptocurrency, Payment System, DApp, Ethereum, Distributed Systems

TEŞEKKÜR

Bu çalışmada bana yardımcı olan başta danışman hocam Dr. Öğr. Üyesi Muammer Akçay'a, desteklerini hep yanımda hissettiğim babam Mustafa Efendioğlu, eşim Tuğçe Efendioğlu ve oğlum Ömer Taha Efendioğlu'na ve emeği geçen herkese teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

ŞEKİLLER DİZİNİ (devam)

Şekil **Sayfa**

ÇİZELGELER DİZİNİ

Çizelge **Sayfa**

SİMGİLER VE KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
DApp	Decentralized Applications (Dağıtık Uygulamalar)
PoW	Proof of Work (İş Kanıtı)
PoS	Proof of Stake (Varlık Kanıtı)
PBFT	Practical Byzantine Fault Tolerance (Bizans Hata Toleransı)
ETH	Ethereum
BTC	Bitcoin
EVM	Ethereum Virtual Machine (Ethereum Sanal Makinası)
GHOST	Greedy Heaviest Observed Subtree
ERP	Enterprise Resource Planning (Kurumsal Kaynak Planlama)
SHA	Secure Hash Algortihm (Güvenli Hash Algoritması)
API	Application Programming Interface (Uygulama Programlama Arayüzü)
TL	Türk Lirası
Http	Hyper text transfer protocol (Hiper Metin Transfer Protokolü)
JSON	JavaScript Object Notation (JavaScript Nesnesi Gösterimi)
RPC	Remote Procedure Call (Uzak yordam çağrısı)
IPC	Interprocess communication (Mesajlaşma sistemi)

1. GİRİŞ

Firmaların ticari ilişkilerine sağlıklı bir şekilde devam edebilmesini sağlayan en önemli faktörlerin başında borç ve alacak takibi gelmektedir. Küreselleşmeyle beraber büyüyen ihracat ve ithalat, firmaların cari hesaplarındaki hareket sayısını arttırmıştır. Bu artış ile beraber firmaların ticari etkileşimleri sonrası dönemsel olarak birbirleri arasında yaptığı borç ve alacak mutabakatları (cari mutabakatlar) daha da karmaşık hale gelmiştir. Birçok firma mutabakatların yapılması ve ortaya çıkan verinin saklanması ile alakalı kendi yöntemlerini geliştirmiştir. Bunların başında tarafların mutabakat verilerini telefon veya mail yolu ile birbirlerine iletmesi ve görüşmeler sonrası ortaya çıkan sonuçları kendi fiziksel veya elektronik ortamında tutmasıdır. Bu süreç gizlilik içeren bilgilerin kötü niyetli üçüncü tarafların eline geçme ihtimalini arttırmakla birlikte verilerin kaybolması ve değişimi gibi riskleri beraberinde getirmektedir. Mutabakat işlemi için kullanılan eski usul yöntemler maddi hataların yanı sıra kâğıt, kargolama ve arşivleme masraflarını arttırmıştır (Schultz, 2020). Günümüzde bu soruna çözüm üretmek amacı ile e-mutabakat yazılım programları piyasaya çıkarılmıştır. Bu programlar sayesinde belirtilen hata ve masrafların bir kısmının önüne geçilse bile halen problemler yaşanmaktadır (Godambe ve Samudrala, 2017). Bunların başında firmalar arasındaki güven problemi gelmektedir. Finansal verilerin güvenilirliği, şeffaflığı, izlenebilirliği, gizliliği ve değiştirilemez olması taraflar için büyük öneme sahiptir. Kullanılan programlarda oluşan verileri firmaların kendi bünyesinde barındırması veya ücret karşılığında üçüncü taraflardan veri barındırma ve güvenlik hizmeti satın alması verilerin gizliliğini, şeffaflığını ve izlenebilirliğini ortadan kaldırmaktadır. Bu durum veriye olan güveni azaltmaktadır. Geçmişe yönelik veri bütünlüğünün korunması ve kötü amaçlı insanların eline geçme ihtimali taraflarda tek merkeze dayalı güven problemi oluşmasına sebebiyet vermektedir. Bu durum gayri merkezi ve yüksek izlenebilirlik sağlayan bir sisteme olan gereksinimi ortaya koymuştur. Diğer bir problem ise mutabakat sonrası gerçekleşen para transfer süreçleridir. Bu süreçte üçüncü taraf olarak bankaların araya girmesi ile işlemlerin beklenen hızda gerçekleşmemesi, yüksek komisyon ücretleri ve banka prosedürleri nedeniyle tarafları memnun etmemektedir (Tapscott ve Tapscott, 2017). Yaşanan bu problemlerin çözümleri için var olan teknolojiler yetersiz kalmış, yeni bir teknolojik altyapıya ihtiyaç duyulmuştur. Son yıllarda adından sıkça bahsedilen blokzincir teknolojisi bu ihtiyaçları çözebilecek teknik altyapıya sahiptir.

Blokzincir teknolojisi sahip olduğu özelliklerle finansal mutabakat ve çevrimiçi ödeme alanında birçok yenilik ve köklü değişiklikler getirecektir (Guo ve Liang, 2016). Herhangi bir merkezi yapıya bağlı olmaması ve dağıtık mimari yapısı ile verilerin tek bir merkezde

toplanmasının önüne geçmektedir. Verinin doğruluğu ve saklanması tek bir bilgisayar tarafından değil ağda bulunan tüm bilgisayarlar tarafından yapılmaktadır. Her bir bilgisayar verilerin doğruluğunu kontrol ederek ortak ağda paylaşmaktadır. Verinin zincire eklenebilmesi için ağdaki bilgisayarlarının çoğunun bu işlemi onaylaması gerekmektedir. Bu sayede verilerin bozulmasının ve müdahale edilmesinin önüne geçilmektedir. Blok mimarisi ile geçmişten günümüze bilgisayar tarafından onaylanmış tüm işlemler kriptografi ile şifrelenerek birbiri ile bağlantılı blok zincirleri oluşturması ile verinin izlenebilirliği ve değiştirilemez olması sağlanmaktadır. Diğer bir önemli özellik ise akıllı sözleşmelerdir. Blokzincir üzerinde tutulan ve taraflar arasında anlaşma sağlandıktan sonra gerçekleşecek aksiyonların tanımlandığı programlardır. Bu sayede üçüncü taraflara gerek kalmadan anlaşma koşullarının hayata geçirilmesi sağlanarak yüksek komisyon ücretlerinin önüne geçilmesi ve gerçek zamanlı izlenebilirlik sağlamaktadır. Veri transferi ve doğrulanması için asimetrik şifreleme, özetleme algoritmaları ve dijital imza kullanılmaktadır. Bu sayede veri bütünlüğü ve taraflar arasındaki güven zorlu matematiksel işlemlerle güvence altına alınmaktadır (Yaga vd., 2018).

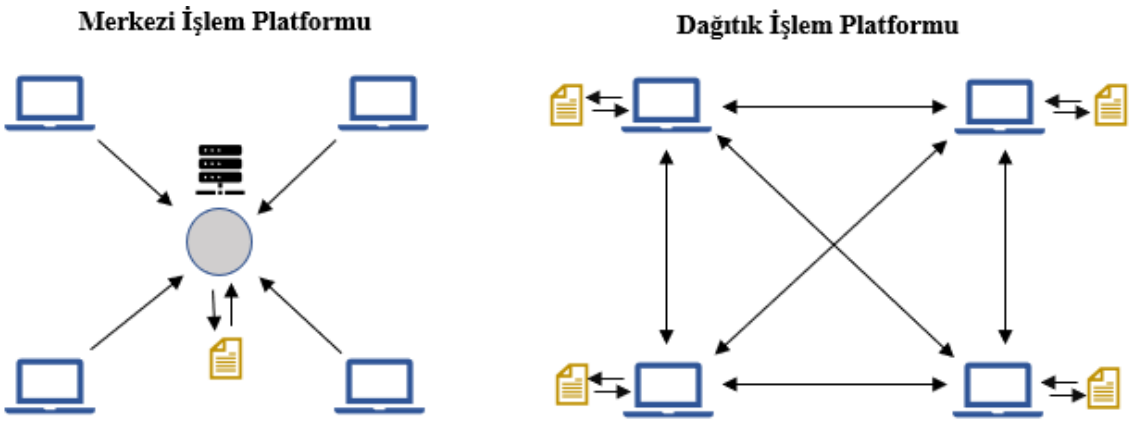
Blokzincir teknolojisinin her iş modeli için kullanılması uygun değildir. Bu yüzden iş modeli gereksinimlerine uygun olup olmayacağı hakkında çalışmalar yapılmalıdır. Bu amaçla akış şemaları oluşturulmaya başlanılmıştır. Kullanıcılara akış üzerinde sorular sorularak verilen cevaplara göre blokzincir teknolojisi kullanmak gerekli mi sorusuna yanıt vermektedir (Wüst ve Gervais, 2018). Blokzinciri kullanımının sektörel bazda uygunluğu ile alakalı en önemli belirleyici ölçüt uygulamalı bilimsel çalışmalar olacaktır. Bu anlamda yapılacak çalışmalar ile blokzincir teknolojisinin daha verimli kullanılması sağlanacaktır.

Bu çalışmanın ikinci bölümünde blokzincir teknolojisinin tarihçesi verilmiştir, temel bileşenleri açıklanmıştır. Üçüncü bölümde blokzincir ortamında çalışabilen, uygulamalar geliştirmeye olanak sağlayan dağıtık uygulamalar (DApp) hakkında bilgiler verilmiş, tez kapsamında geliştirilen uygulamada tercih edilen, dağıtık uygulama platformu olan Ethereum ve akıllı sözleşmeler ele alınmıştır. Dördüncü bölümde tez kapsamında ele alınan mevcut cari mutabakat yöntemleri ve sonrasında gerçekleşen ödemeler ile alakalı sorunlar detaylı olarak değerlendirilmiştir. Beşinci bölümde bir önceki bölümde bahsedilen problemlere çözüm olarak Ethereum tabanlı merkezi olamayan ve akıllı sözleşme kullanan mutabakat ve ödeme işlemleri yapılabilen DApp uygulaması geliştirilmiştir. Son bölümde ise elde edilen sonuçlara ve önerilere yer verilmiş, gelecek çalışmalar açıklanmıştır.

2. BLOKZİNCİR TEKNOLOJİSİ

Blokzincir teknolojisi ilk olarak 2008 yılında Satoshi Nakamoto tarafından yayınlanan "Bitcoin: A peer-to-peer electronic cash system" adlı makale ile ortaya konulmuş bir teknolojidir. Bitcoin blokzincir teknolojisi ile geliştirilen ilk uygulamadır. Herhangi bir merkezi otoriteye bağlı olmayan, üçüncü tarafların olmadığı, kriptoloji biliminin güvenliğini sağladığı ödeme sistemi olarak tanımlanabilir. Bu uygulama ile birlikte blokzincir teknolojisinin farklı birçok alanda kullanılabileceği, sosyal ve ekonomik alanlarda yeni temeller oluşturma potansiyeline sahip olduğu görülmüştür (Iansiti ve Lakhani, 2017).

Blokzinciri dağıtık bir kayıt defteridir. İşlemlerin belirlenmiş bir uzlaşma (konsensüs) algoritması sonucu dağıtık veri tabanına kaydedilmesidir. Ağdaki tüm olaylar düğüm (node) diye tanımlanan bilgisayarlar tarafından doğrulanmakta ve bloklar oluşturulmaktadır. Merkezi sistemlerin aksine blokzincirin her üye düğümü ağda bulunan tüm verilerin kopyasını elinde bulundurmaktadır (Şekil 2.1). Düğümler ağda gerçekleşen işlemleri izleyerek ağ protokolüne uyup uymadığını kontrol ederler. Belirli bir işlem sayısına ulaştığında, işlemleri bir araya getirerek bir blok oluşturup ağda paylaşırlar. Blokzincirde gerçekleşen işlemlerin kayıt defterine eklenebilmesi için doğrulanmış bir bloğun oluşması gerekmektedir. Ağda paylaşılan blok diğer düğümler tarafından içerisindeki işlemler ile birlikte doğrulanmaktadır. Doğrulama işlemi başarılı olursa blok, düğümler tarafından kendi kayıt defterlerine eklenmektedir. Gerçekleşen bu işlemlere uzlaşma adı verilmektedir. Günümüzde gelişen teknolojiye bağlı olarak uygulama gereksinimleri artmıştır. Bu gereksinimler karşısında birçok yeni uzlaşma modelleri tasarlanmış ve yeni modeller ortaya çıkmaya devam etmektedir. Uzlaşma modelleri devam eden bölümlerde detaylı olarak ele alınmıştır.



Şekil 2.1. Merkezi ve dağıtık işlem platformları.

Blokzincir üzerinde işlemlerin tutulduğu bloklar arasında kriptografik bir ilişki vardır. Her bloğun bir hash değeri vardır. Bloklar, kendinden bir önceki bloğun hash (özet) değerini kendi içinde barındırır. Her bir yeni blok sisteme dâhil edildiğinde bu işlem gerçekleştirilir. Böylece sistem üzerindeki bir veriyi değiştirmek isteyen kişi ilgili bloktaki hash değerini bozacağından dolayı kendinden sonra gelen her blokta özet değerlerini tekrardan hesaplamak zorundadır. Bu işlemin yanı sıra ağda bulunan tüm düğümlerin kayıt defterlerinde de bu değişikliğin yapılması gerekmektedir. Pratik olarak mümkün olmayan bu durumdan dolayı blokzincir değiştirilemez olarak kabul edilmektedir.

Blokzincir altı temel unsurdan oluşmaktadır (Lin ve Liao, 2017). Bunlar;

Dağıtık: Veriler ağ üzerinde bulunan düğümler tarafından tutulmaktadır.

Şeffaf: Düğümler ağ üzerinde gerçekleşen tüm işlemleri eş zamanlı izleyebilirler.

Açık Kaynaklı: Çoğu blokzincir sistemi herkese açıktır. Kullanıcılar bir uygulama geliştirmek için blokzincir teknolojilerini kullanabilirler.

Değiştirilemez: Dağıtık kayıt defteri yapısı ve blok sistemi sayesinde veriler değiştirilemez olarak kabul edilmektedir.

Anonimlik: Kullanıcıların sadece açık anahtarları ile işlem yapıldığından veri aktarımı anonim olabilmektedir.

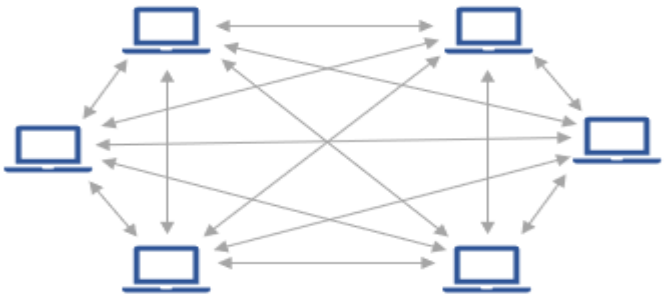
Özerklik: Herhangi bir merkez veya kuruma bağlı olmadan bir işlemi gerçekleştirme ve doğrulama işlemi güven içerisinde yapılmaktadır.

2.1. Blokzincir Ağları

Blokzincir teknolojileri kullanım durumlarına göre genel ağlar, özel ağlar ve şirketler birliği (konsorsiyum) ağları olarak üç türe ayrılabilir (Lin ve Liao, 2017).

Açık blokzincirler

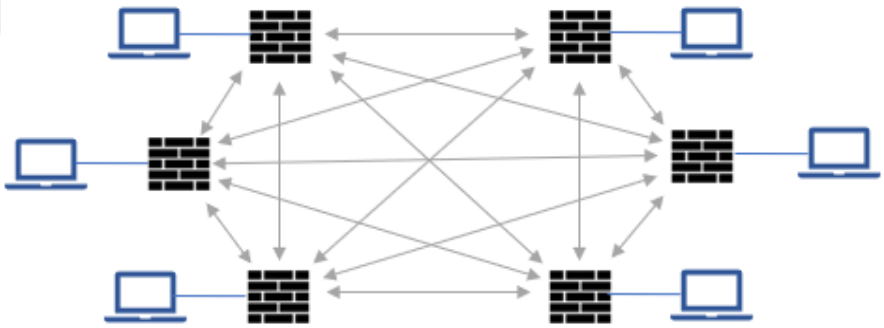
Veri okuma, doğrulama ve blok oluşturma işlemlerinin herkes tarafından yapılabilen ağlardır. Bitcoin ve Ethereum halka açık blok zincir ağına örnek olarak verilebilir. Şekil 2.2’ de açık blokzincir ağı mimarisi gösterilmiştir.



Şekil 2.2. Açık blokzincir ağı (Lin ve Liao, 2017).

Özel blokzincirler

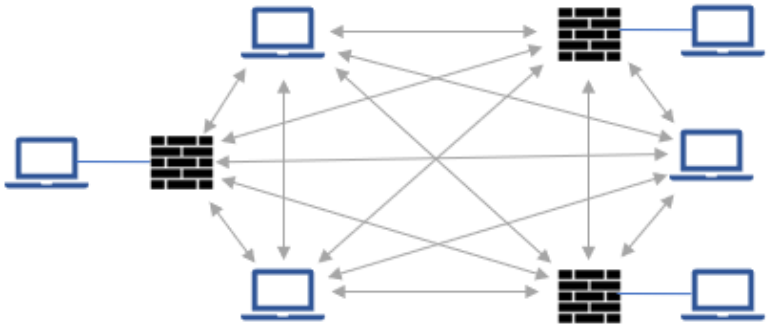
Bazı durumlarda kurum ve kuruluşlar tarafından veri okunması, doğrulanması ve blok oluşturulmasının herkes tarafından yapılması istenmeyebilir. Bu tarz sistemlerde sadece istenilen düğümlere izin verilerek herkesin ağa katılmasının önüne geçilmektedir. Şekil 2.3' de özel blokzincir ağı mimarisi gösterilmiştir.



Şekil 2.3. Özel blokzincir ağı (Lin ve Liao, 2017).

Konsorsiyum blokzincirler

Herkese açık ve özel blokzincirlerin beraber kullanıldığı ağlardır. Özel blokzincirlerden farklı olarak birden fazla sistemi kontrol eden kuruluşun olmasıdır. Aynı sektörde faaliyet gösteren firmaların tek bir ağda işlemlerini yönetebilmelerini sağlayacak altyapıya sahiptir. Hyperledger ve R3CEV platformları konsorsiyum blokzincir ağına örnek olarak verilebilir (Lin ve Liao, 2017). Şekil 2.4' de konsorsiyum blokzincir ağı mimarisi gösterilmiştir. Çizelge 2.1' de blokzincir ağlarının detaylı olarak karşılaştırılması verilmiştir.



Şekil 2.4. Konsorsiyum blokzincir ağı (Lin ve Liao, 2017).

Çizelge 2.1. Blokzincir ağlarının karşılaştırılması (Zheng vd., 2017).

Özellik	**Açık Blokzincirler**	**Konsorsiyum Blokzincirler**	**Özel Blokzincirler**
Uzlaşmanın Sağlanması	Tüm düğümler tarafından	Çoklu kuruluş tarafından seçilmiş düğümler tarafından	Tek kuruluş tarafından seçilmiş düğümler tarafından
Okuma İzni	Herkese açık	Herkese açık veya kısıtlı	Herkese açık veya kısıtlı
Veride Değişmezlik	Neredeyse imkânsız	Değiştirilebilir	Değiştirilebilir
Verimlilik	Düşük	Yüksek	Yüksek
Merkezileştirme	Hayır	Kısmi	Evet
Uzlaşma Süreci	İzinsiz	İzinli	İzinli

Konsorsiyum ve özel blokzincir ağlarında düğümler belirli bir kullanıcı tarafından seçilmesi, okuma izinlerinin verilmesi ve uzlaşma sürecinin izinli olması nedeniyle kısmi merkezileşme olduğu söylenebilir. Ayrıca düğüm sayısının az olması verinin değiştirilme ihtimalini arttırmaktadır.

2.2. Blokzincirin Temel Bileşenleri

Bu bölümde blokzincir teknolojisinin ekosistemini oluşturan temel bileşenler başlıklar halinde açıklanmıştır. Çalışma prensibi hakkında bilgiler verilmiştir.

2.2.1. Dağıtık kayıt defteri teknolojisi (Distributed ledger technology)

Blokzincir dağıtık kayıt defteri teknolojisi ile özleştirilebilir. Merkezi bir otorite olmaksızın işlemlerin sisteme katılan kullanıcılar tarafından doğruluğu üzerinde mutabakat sağlayarak sisteme dâhil edilmesidir. Sistem kullanıcıları ağda bulunan veri kümesinin kopyasını

elinde bulundurur. Bu sayede herhangi bir düğüm silinse bile diğer kullanıcılarda bulunan defter sayesinde bilgiler kaybolmaz.

Sistemde bulunan düğümler tüm verilere eş zamanlı ulaşabilir. Sistem üzerinde oluşmuş işlemler herhangi bir düğüm tarafından doğrulanıp bir blok yapısı haline getirilerek ağda yayınlanır. Bloğu elde eden her düğüm doğrulama işlemi sonrası bloğu kendi defterine ekler. Bu sayede ağ genelinde senkronize çalışan bir yapı elde edilir. Her düğüm defterin bir kopyasını elinde bulundurmasından dolayı herhangi bir düğüme yapılacak saldırı, tüm ağda aynı etkiyi göstermez. Bu yüzden defterin zarar görmesi veya ortadan kaldırılmasının önüne geçilir. Dağıtık kayıt defterine kötü niyetli bir düğüm yanlış içerikli veri eklemek istediğinde ağdaki diğer düğümler tarafından bu işlem kabul görmez ve deftere eklenmez. Veri bütünlüğünü sağlamak için özetleme fonksiyonları ve dijital imza kullanılır.

2.2.2. Bloklar

Ağda gerçekleşen işlem veya işlemlerin tutulduğu yapılardır. Ağ üzerinde gerçekleşen işlem bloğa eklenmeden önce kuyruğa eklenir. Düğümler kuyruktaki işlemleri belirlenen mutabakat çerçevesinde doğrulayıp bir blok oluştururlar. Bloğu ilk oluşturan düğüm ağda paylaşır. Diğer düğümler ilgili bloğun doğruluğunu kontrol ederek kendi defterlerine dâhil ederler. Eğer düğümler tarafından aynı anda birden fazla blok ağda paylaşılırsa blokzincirde çatallanma oluşabilir. Bu süreçte düğümler çalışmaya devam etmektedir. Çatallaşmış zincirin birine yeni bir blok eklendiğinde diğer düğümler uzun olan çatala geçerek çalışmaya devam ederler.

Blok; başlık ve gövde kısımlarından oluşmaktadır. Başlık kısmında sürüm numarası, blok hash değeri, bir önceki bloğun hash değeri, oluşturulduğu zaman, Merkle ağaç hash değeri, blok büyüklüğü, işlem sayısı, sayaç, zorluk değerinden oluşur. Gövde kısmı ise onaylanmış işlemlerin listesini barındırmaktadır. Çizelge 2.2’ de blok içerisindeki terimlerin açıklamaları verilmiştir.

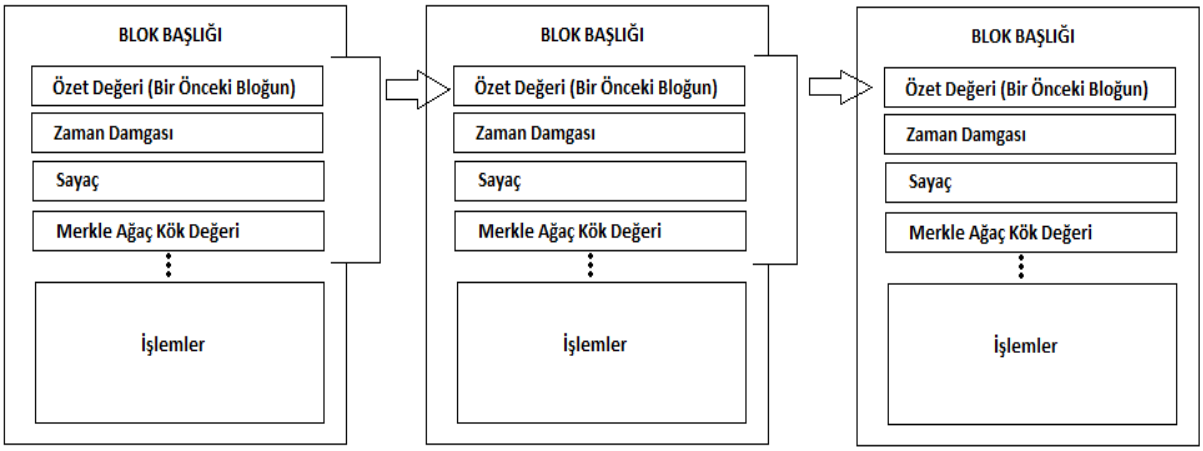
Çizelge 2.2. Blok içeriğini oluşturan terimler (Özcan, 2019).

Sürüm numarası	Blokzincirin sürüm numarasıdır.
İşlem sayısı	Blok içerisindeki işlemlerin toplam sayısını gösterir.
Blok büyüklüğü	Bloğun büyüklüğünü ifade eden terimdir.
Blok ödülü	Bitcoin blok yapısında bulunan ve madencinin bloğu zincire eklemesi karşılığı kazanacağı Bitcoin değerini gösteren terimdir.

Çizelge 2.2. (devam) Blok içeriğini oluşturan terimler.

Önceki bloğun özet değeri	Bir önceki bloğun özet değeridir. Bloklar arası bağlantı bu değer sayesinde sağlanır. Bu şekilde zincir yapısı oluşturulur.
Zorluk	Bloğun ortaya çıkarılma süresini değiştirmeye yarayan parametredir.
Sayaç	Madencilik yapılan blokzincir ağında özet fonksiyonunun değişmesi için kullanılan değer.
İşlemler	Blok içerisindeki onaylanmış işlemler listesidir.
Zaman damgası	Bloğun oluşturulma zamanıdır.
Merkle ağacı kök değeri	Blok içerisindeki işlemler Merkle denilen bir veri yapısında tutularak işlemler belirli bir algoritmaya tabi tutularak bir hash değeri üretilir.
Zorluk	Bloğun ortaya çıkarılma süresini değiştirmeye yarayan parametredir.

Bloklar birbirleri arasında özet (hash) değeri ile zincirlenir. Her bir blok kendinden önceki bloğun özet (hash) değerini kendi içerisinde barındırır (Şekil 2.5). Bu şekilde ilk bloktan son bloğa kadar bloklar arası bir bağ oluşur. Herhangi bir blokta işlem veya blok parametrelerinde yapılacak bir değişiklik bloğun özet değerini değiştireceği için zincirde kopukluk meydana gelir. Böylece değiştirilmiş bloklar kolayca tespit edilerek diğer düğümler tarafından reddedilir (Yaga vd., 2018).



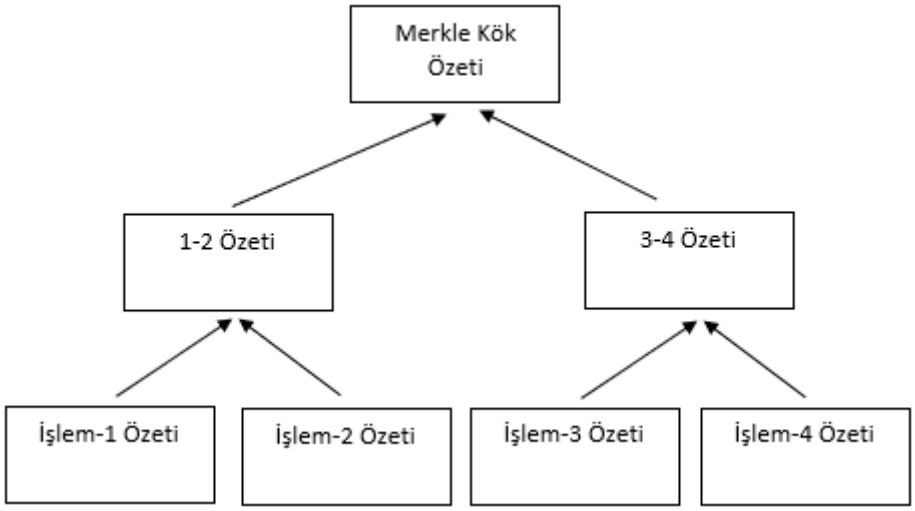
Şekil 2.5. Blokzincir yapısı.

Blokzincir ağında ilk oluşan bloğa Genesis bloğu adı verilir. Aynı zamanda sıfır bloğu veya başlangıç bloğu olarak da adlandırılmaktadır. Her blokzincir ağının yayınlamış bir başlangıç bloğu vardır. Bu bloktaki durumu kabul eden kullanıcılar sisteme katılır. Ağ üzerinde yayınlanmış

mutabakat modeline uygun bloklar başlangıç bloğuna eklenerek zincir devam etmektedir. Blok üzerindeki işlemlerin belirli bir algoritma sonucu hash değeri hesaplanmaktadır. Bu algoritmada uygulanan yönteme Merkle ağaç yapısı denir.

Merkle ağaç yapısı

Blok içerisindeki her bir işlemin özet (hash) değerlerinin alınıp, tek bir özet değeri elde edene kadar belirli bir işleme tabi tutulmasıdır (Şekil 2.6). İşlemlerin tek tek hash değeri alınır ve bu değerler ağacın en alt sırasını oluşturur. Sonrasında eşlenik olarak kendi kardeşi olan hash değeri ile birleştirilip yeni bir hash değeri elde edilir. Bu değer bir üst düğüme eklenir. Tek bir hash değeri elde edilene kadar bu işleme devam edilir. Elde edilen değer Merkle ağaç kökü özeti (Merkle hash root) olarak blok başlığında saklanır. Bu algoritma sayesinde bloklarda işlem bütünlüğünün sağlanabilmesi ve işlemlerin hızlı doğrulanabilmesi sağlanmıştır.



Şekil 2.6. Merkle ağaç yapısı.

2.2.4. Uzlaşma protokolleri

Uzlaşma protokolleri blokzincirin düzgün çalışması için anahtar görevindedir. Blokzincire eklenecek blokların doğruluğunu ve geçerlilik koşullarını belirleyen bir mekanizmadır. Uzlaşma mekanizması tüm madencilere aynı ağırlığı vermekten ve çoğunluğuna göre karar vermekten oluşur (Fernández-Caramés, Fraga-Lamas, 2018). Merkezi sistemlerde verinin tek bir merkezde toplanması güvenlik zafiyeti oluşturması sebebiyle ortaya çıkmıştır. En popüler uzlaşma algoritmaları PoW, PoS ve PBFT algoritmalarıdır (Usta ve Doğantekin, 2018).

İş kanıtı algoritması (PoW)

Bu algoritmada ağda bulunan düğümler zorlu bir matematiksel problemi en kısa sürede diğer düğümlerden önce çözmeye çalışmaktadır. Genel olarak bir bloğun zincire eklenebilmesi için blok hash değerinin ilk 6 hanesi 0 olmalıdır. Bu değeri bulmak için düğümler, blok yapısı içinde bulunan sayaç (nonce) değerini değiştirerek SHA-256 özetleme algoritması ile bloğun özet değerini hesaplar. İlk 6 hanesi 0 olan hash değeri bulunduğunda ilgili düğüm tarafından ağda yayınlanır. Bu işlem sonrası diğer düğümler ilgili hesaplama işlemini bırakırlar. Bunun yerine yayınlanan bloğun geçerliliğini kontrol ederler. Diğer düğümler tarafından bloğun hash değerinin doğrulaması yapıldıktan sonra blokzincirlerine eklenmektedir. Blok, blokzincire eklendikten sonra ilgili madenciye ödül olarak sanal para verilmektedir. Bu işlemlere madencilik adı verilmektedir. Blokzincire yeni bir blok ekleme süresi ortalama 10 dakikadır (Nguyen ve Kim, 2018).

Madenciler blok özet değerini üretmek için yüksek miktarda işlem gücü harcamaktadır. Bu sebeple madenciler, matematiksel problemi ilk çözebilmek için donanıma fazla yatırım yapması gerekmektedir. Bu durum madenciler arasında etkin bir rekabet ortamının oluşmasını engellemektedir. Bunun yanı sıra kullanılan donanıma bağlı olarak yüksek miktarda enerji gerekmektedir. Diğer bir problem ise işlemlerin ortalama 10 dakika gibi bir sürede onaylanmasıdır. Bir kişi herhangi bir ödeme veya farklı bir işlem yaptığında bu işlemin düğümler tarafından onaylanıp bir blok altında doğrulanması için ortalama 10 dakika beklemesi gerekmektedir. Bu sebeplerden dolayı iş kanıtı algoritması ile alakalı birçok iyileştirme modelleri ortaya çıkarılmış ve bu durum yeni algoritmaların tasarlanmasına sebep olmuştur (Nguyen ve Kim, 2018). Sanal para dünyasının en popüleri Bitcoin tarafından kullanılmaktadır.

Varlık kanıtı algoritması (PoS)

PoW algoritmasında yüksek işlem gücü ve yüksek miktarda enerji tüketimi gereksiniminden dolayı ortaya çıkmış ve kripto para dünyasında ikinci büyük para birimi olarak değerlendirilen Ethereum tarafından kullanılması planlanan algoritmadır. Bu yöntemde bloğu doğrulayacak kişi, sistem içerisinde bulunan katılımcıların elinde bulundurduğu hisse oranı miktarınca seçime tabi tutularak seçilmektedir. Katılımcılar sisteme dâhil olarak belirli bir miktar üstü kripto para ile hisse satın alırlar. Aldıkları hisse oranı ne kadar yüksek olursa işlemleri doğrulama, blok oluşturma olasılıkları o kadar artar (Usta ve Doğantekin, 2018). PoW algoritmasında bulunan zorlu matematiksel problem çözme durumu varken PoS algoritmasında katılımcı elindeki hisse miktarınca ödül kazanmaktadır. Bu duruma Minting (Para basma)

denilmektedir. Bu yöntem sayesinde yüksek enerji tüketiminin önüne geçilmiş ve işlemlerin kısa sürede onaylanması sağlanmıştır.

PoS yönteminde büyük hisse sahibi kişilerin küçük hisse sahibi kişiler üzerinde baskın olmasını engellemek için bazı yöntemler geliştirilmiştir. Bunlardan bir tanesi coin yaşı (Age of Coin) olarak adlandırılan yöntemdir. Bu yöntemde kullanıcı coinlerinin elde tutma süresi temel alınarak seçim yapılmasıdır. Coin yaşı ne kadar yüksek ise kullanıcının blok oluşturma ihtimali o kadar yüksektir. Blok işlemi oluşturduktan sonra kullanıcının coin yaşı sıfırlanır ve tekrar artmaya başlar. Bu şekilde sistemin domine edilmesi ve %51 saldırılarının önüne geçilmesi için etkili bir yöntem sunmaktadır.

PoS sisteminde herhangi bir kötü niyetli saldırı veya işlem tespit edildiğinde ilgili kullanıcının oylama işlemi için ayırdığı hisseye sistem tarafından el konulur. Bunun yanı sıra kullanıcı gelecekte blok oluşturma yetkisini kaybetmektedir. Kullanıcının ayırdığı hisse ödül oranından çok daha yüksek olduğu için bu şekilde bir saldırı yapmak karlı bir yöntem değildir.

Bizans hata toleransı (PBFT)

Bizans generallerinin bilgi doğrulamak için kullandığı bir yöntemden adını alan PBFT, ağ üzerinde bulunan düğümlerin çoğunluğunun onayını alarak sistemin doğruluğu üzerinde anlaşmaya varma prensibine dayanmaktadır. Düğümler ana düğüm ve yedek düğümlerden oluşmaktadır. Ağ üzerindeki tüm düğümler birbirleri ile iletişim halindedir. Her düğüm ağdaki diğer düğümlerin açık anahtar bilgisine sahiptir. Kullanıcı tarafından ağa gönderilen istek lider düğüme iletilir. Lider düğüm isteği yedek düğümler arasında yayınlanır. Yedek düğümler isteği doğrular ve istemciye bir yanıt gönderir. Çoğunluğun verdiği cevaba göre işlem geçerli veya geçersiz olur.

PBFT yaklaşımında ağa dâhil olan tüm düğümler merkezi bir sistemin onayından geçmektedir. Bu durum blokzincirin temeli olan merkeziyetsizlik kavramı ile çelişmektedir. Bu sebepten dolayı PBFT mutabakat algoritması, özel blokzincir ağları için uygundur.

PBFT protokolü yüksek işlem hacmini işleyebilir ve ağ genelinde iyi derecede ölçeklenebilir. Asenkron çalışmasından dolayı performans ve hızlı yürütme süresi sağlamaktadır. Linux Foundation tarafından 2015 yılında duyurulan Hyperledger modeli tarafından kullanılmaktadır. Bu teknoloji Intel, IBM ve Cisco gibi kuruluşlar tarafından desteklenmektedir (Wahab ve Memood, 2018).

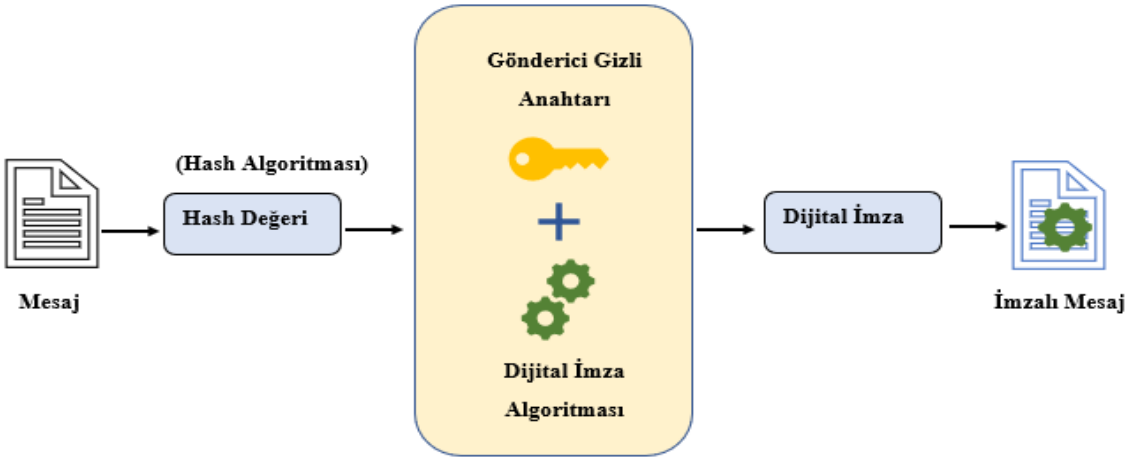
2.2.6. Dijital imza

Dijital imza, değişmez veri kopyaları sağlayabilen, bir mesajın değiştirilmeden belirli bir kişiden geldiğini kanıtlamaya yarayan bir sistemdir. Sistemler arası veri bütünlüğü ve kimlik doğrulama yöntemidir. Blokzincir platformunda akıllı sözleşmeler ve kullanıcılar arası yapılan veri transferleri dijital imza ile imzalanmaktadır. Blokzincir ağında bulunan düğümlerin görevlerinden biri de dijital imza doğrulaması yapmaktır.

Dijital imzalar asimetrik şifreleme kullanır. Verileri şifrelemek ve şifreyi çözmek için açık ve özel anahtar kullanılır. Açık anahtarlar herkes ile paylaşılabilir. Gizli anahtar ise kullanıcı tarafından saklanmalıdır. Gizli anahtarlar veriyi şifrelemek için kullanılır. Açık anahtar ise şifrelenmiş veriyi çözmek için kullanılmaktadır. Bir mesajın imzalanma aşamaları aşağıdaki gibidir.

Mesajın Özel Anahtar ile İmzalanması

Gönderilecek mesajın, mesaj özeti algoritması kullanılarak hash değeri bulunur. Bu değer veriler için benzersizdir. Elde edilen hash değeri göndericinin özel anahtarı ile dijital imza algoritmasına girdi olarak verilir. Bu işlem sonrası dijital imza elde edilir. Mesaj ile birlikte alıcıya gönderilir. Şekil 2.7' de bir mesajın dijital imza ile imzalanma akışı gösterilmiştir.

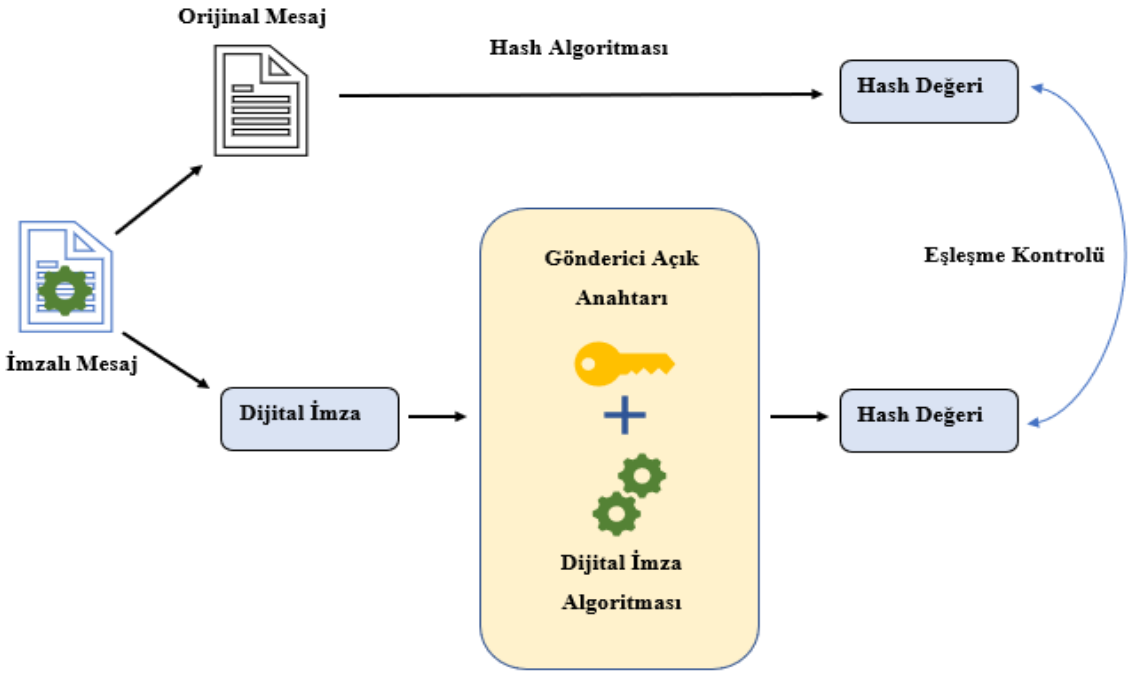


Şekil 2.7. Dijital imza ile mesajın imzalanması.

Mesajın Açık Anahtar ile Doğrulanması

Alıcı dijital imzalanmış mesajın orijinal mesajını, göndericinin kullandığı özetleme algoritmasını kullanılarak özet değerini hesaplar. Bunun yanı sıra mesaj ile birlikte gönderilen

dijital imzayı, göndericinin açık anahtarı ile dijital imza algoritması kullanarak bir özet değeri elde eder. Elde edilen iki hash değeri eşleşirse, alıcı dijital imzanın geçerli olduğundan emin olur. Şekil 2.8' de bir mesajın dijital imza ile doğrulanması akışı gösterilmiştir.



Şekil 2.8. Dijital imza ile mesajın doğrulanması.

2.2.1. Kriptoloji

Blokzincir teknolojisi yapılan işlemler, cüzdanlar, mutabakatlar, güvenlik ve gizlilik protokolleri için kriptografiyi birçok şekilde kullanmaktadır. Blokzincirin güvenliği kriptoloji tarafından sağlanmaktadır. Olası bir saldırı veya kötü niyetli kişilerce ağı dinleme durumlarının engellenebilmesi için büyük öneme sahiptir. İşlemlerin herhangi bir algoritma kullanılarak okunamayacak biçime dönüştürülmesine şifreleme (encryption), okunamayacak verinin orijinal haline dönüştürülmesine şifre çözme (decryption) denir. Şifrelemede kullanılan anahtar ve algoritmanın çeşitlerine göre iki ana gruba ayrılmaktadır. Blokzincir yapılarında asimetrik şifreleme ve mesaj özeti algoritmalarından yararlanılmaktadır.

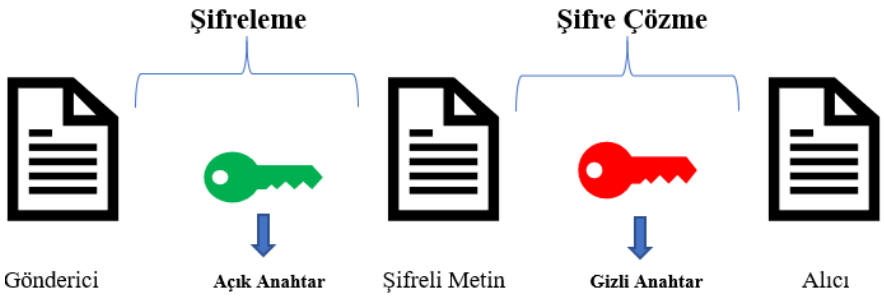
Simetrik şifreleme

Şifreleme ve şifrenin çözülmesi için tek anahtar kullanılmaktadır. Alıcıya gönderilecek veri gizli anahtar ile şifrelendikten sonra alıcıya gönderilir. Alıcı şifreli veriyi aynı gizli anahtarı kullanarak çözer (Beşkirli vd., 2019). Simetrik şifreleme algoritmalarında şifreleme ve şifre

çözme işlemleri hızlı gerçekleşir. Bunun yanı sıra bazı dezavantajları mevcuttur. Gizli anahtarın alıcıya güvenli bir yol ile ulaştırmak gerekmektedir. Ağı dinleyen ve gizli anahtara sahip olan kötü niyetli bir kişi tarafından verilerin ele geçirilme ve değiştirilebilme riski vardır (Usta ve Doğantekin, 2018).

Asimetrik şifreleme

Şifreleme ve şifre çözme işlemleri birbirinden farklı anahtarlar ile yapılmaktadır. Bu anahtarlara açık (public) ve gizli (private) anahtar denmektedir. Açık anahtar ve gizli anahtar arasındaki ilişki karmaşık matematiksel problemlere dayanmaktadır. Bir anahtardan diğerine ulaşmak mümkün değildir. Genel anahtar herkese paylaşılabilirken gizli anahtar sadece sahibinde bulunur. Alıcının açık anahtar ile imzalanan veri yalnızca alıcının özel anahtarı ile çözülebilmektedir (Beşkirli vd., 2019). Bu sebeple gönderici veriyi alıcının açık anahtarı ile şifrelemektedir. Bu şekilde verinin sadece gizli anahtara sahip alıcı tarafından okunulduğundan emin olunur. Aynı şekilde, göndericinin gizli anahtarı ile imzalanmış bir veri sadece göndericin açık anahtarı ile çözülebilmektedir. Gönderici veriyi gizli anahtarı ile imzalar ve gönderir. Alıcı, göndericinin açık anahtarını kullanarak şifreyi çözer. Bu şekilde sadece özel anahtara sahip gönderici tarafından verinin gönderildiğinden emin olunur (Usta ve Doğantekin, 2018). Şekil 2.9 de asimetrik şifreleme yapısı gösterilmiştir.



Şekil 2.9. Asimetrik şifreleme (Usta ve Doğantekin, 2018).

Asimetrik algoritmalar çalışma yapısı ile kimlik doğrulama sistemi (User Authentication) olarak kullanılmaktadır. Bu sayede mesajın ilgili kişi tarafından gönderilip gönderilmediği anlaşılır. Blokzincir yapısında işlemlerin transferi ve doğrulanması aşamasında asimetrik algoritmalardan faydalanılmaktadır.

Mesaj özeti (Hash) algoritmaları

Herhangi bir uzunluktaki girdiden sabit uzunlukta bir çıktı üretmek amacı ile kullanılan, tek yönlü çalışan algoritmalardır. Özetleme algoritmalarının genel kullanım amacı herhangi bir veride değişiklik olup olmadığının tespitinin yapılmasıdır. Bu algoritmalarda girdi verisi değişmedikçe her zaman aynı çıktıyı vermektedir. Girdi verisinde herhangi bir noktalama işareti veya karakter boyutunda yapılacak küçük bir değişiklik özet değerinin tamamen değişmesini sağlamaktadır (Usta ve Doğantekin, 2018).

Örnek olarak, "Bilgisayar" ve "bilgisayar" kelimelerinin SHA-256 özetleme algoritması kullanılarak elde edilen değerler çizelge 2.3'te verilmiştir.

Çizelge 2.3. Sha-256 algoritması sonuçları.

Girdi	**Çıktı**
Bilgisayar	06e44c77accf6db4b944092b65ae8e7a173287ff569d95e72ffda7f30e7b5186
bilgisayar	0734326bf69f65d0071300c274c0f0ddc1a343b5161519597d2bf2854b8455a1

Hash algoritmaları blokzincir sürecinin önemli parçalarından biridir. Madencilik süreci, blokların oluşturulması ve bağlanmasında önemli rol oynamaktadır. Blokzincirdeki her bloğun bir özet değeri vardır. Her blok kendinden bir önceki bloğun özet değerini kendi içinde barındırır. Hash algoritmasının deterministtik yapısı sayesinde bloklarda yapılacak herhangi bir değişiklik düğümler tarafından algılanır. Bu şekilde blokzincir ağının güvenliği sağlanmış olur. Blokzincir altyapısı kullanan popüler kripto para birimi Bitcoin SHA-256, Ethereum ise keccak256 özetleme algoritmalarını kullanmaktadır.

2.2.3. İşlemler (Transactions)

Blokzincir ağında kullanıcıların etkileşimi sonrası ortaya çıkan kayıtlara işlem denmektedir. İşlemler bloklar içerisinde tutulmaktadır. Bir blokta birden fazla işlem olabilir. İşlemler madenciler tarafından blok içerisine eklenene kadar onaylanmamış işlemlerin bulunduğu bir havuzda bekletilir. Madenciler tarafından havuzdaki işlemler doğrulanarak blok oluşturulur. Oluşturulan blok madenci tarafından ağda yayınlanarak diğer düğümlerin de senkron çalışmasını sağlar.

İşlemler genel olarak teknik detaylar, girdi ve çıktı alanlarından oluşur. Şekil 2.10 'da Bitcoin kripto para biriminin işlem yapısı gösterilmiştir.

Şekil 2.10. Bitcoin işlem (Transaction) mimarisi.

Teknik detayların bulunduğu bölümde 4 temel unsur bulunmaktadır.

Versiyon numarası: Ağ içerisindeki işlemlerin onaylanması için oluşturulan protokollere verilen numaradır.

İşlem kilit süresi: İşlemin blok zincirine eklenebileceği en erken süreyi veya en erken bloğu belirten kısımdır.

Girdi sayısı: İşlem içerisindeki girdi sayısıdır.

Çıktı sayısı: İşlem içerisindeki çıktı sayıdır.

Girdi verilerinin bulunduğu bölümde 4 temel unsur bulunmaktadır.

Girdi uzunluğu: Girdi verilerinin boyut değeridir.

Sıra numarası: Bir işlemdeki kilit süresini devre dışı bırakmak için kullanılan değerdir.

Önceki işlem özet & index değeri: Önceki işlemden bir çıktı belirterek jetonların nereden geldiğini tanımlar.

Girdi verisi: İşlemin kime ait olduğunun tanımlandığı, Bitcoin transferi için özel anahtar ile işlemin imzalandığı yerdir.

Çıktı detaylarının bulunduğu bölüm ise 3 ana unsurdan oluşmaktadır.

Çıktı uzunluğu: Çıktı verilerinin boyut değeridir.

Miktar: Gönderilecek BTC miktarıdır.

Çıktı Verisi: Bitcoin'nin gönderileceği adreslerin tanımı ve göndericinin açık anahtarı ile hesaplanmış değerlerin olduğu bölümdür.

2.2.6. Hesap adresleri ve cüzdan

Blokzincir ağına dâhil olan her kullanıcıya özel bir hesap adresi verilmektedir. Kullanıcılar açık ve gizli olmak üzere iki farklı anahtara sahiptirler. Kullanıcının sahip olduğu açık anahtara ek alanlar eklenerek özet (hash) değeri alınmasıyla hesap adresi oluşturulur. Ağ üzerinde yapılan işlemlerin takibi bu adresler ile yapılmaktadır.

Kullanıcının ağ içerisindeki sahip olduğu tüm değerler gizli anahtarı ile imzalanmıştır. Gizli anahtarın kullanıcı tarafından saklanması büyük öneme sahiptir. Gizli anahtarın kötü niyetli bir kişi tarafından ele geçirilmesi ile kullanıcının sahip olduğu tüm değerleri kaybetmesine sebebiyet verebilir. Bu sebeple gizli anahtarların saklanması için uygulamalar geliştirilmiştir. Bu uygulamalara cüzdan adı verilmektedir. Cüzdanlar soğuk ve sıcak olmak üzere ikiye ayrılmaktadır. Soğuk cüzdan internet erişiminin olmadığı, donanımsal cihazlarda saklanan uygulamalardır. Bu cüzdanlar sd kart, flash bellek veya taşınabilir disk olabileceği gibi bir kâğıt parçasında olabilirler. Sıcak cüzdanlar ise internet erişiminin olduğu cüzdanlardır.

Sıcak cüzdanların internet erişiminin olmasından dolayı soğuk cüzdanlara göre daha az güvenliklidir. Herhangi bir siber saldırıya maruz kalma şansı daha yüksektir. Bu sebeple büyük miktar kripto paraya sahip hesapların sıcak cüzdan kullanılması önerilmemektedir. Soğuk cüzdanları kullanmak sıcak cüzdanlara göre daha zahmetlidir. Anlık işlemlerde birçok süreci uygulamak ve tekrar etmek gerekmektedir. Bu sebeple soğuk cüzdanlar birikim yapmak için idealdir.

3. DAĞITIK UYGULAMALAR (DApp)

Dağıtık ağ ortamında çalışan ve akıllı sözleşmeler kullanan uygulamalar “Distributed Applications” veya “DApp” olarak adlandırılmaktadır. Günümüzde yaygın olarak merkezi uygulamalar kullanılmaktadır. Uygulama kararları merkezi bir mekanizmadan alınır ve kullanıcılara iletilir. Kullanıcıların uygulama üzerinde bir işlem yapabilmesi merkezi sisteme bağlıdır. Merkezi sistemde gelişecek olumsuz durumlar tüm kullanıcıları etkilemektedir. Dağıtık uygulamaların farkı herhangi bir merkezi sisteme bağlı kalmadan otonom bilgisayarlardan oluşan ağ üzerinde çalışmasıdır. Bilgisayarlar ortak bir hedefe ulaşmak yerine birbirleri ile iletişime geçmektedir. Uygulama verileri her bilgisayarda kopyası bulunacak şekilde dağıtık tutulmaktadır. Bu şekilde uygulama üzerinde yaşanacak veri kaynaklı problemlerin önüne geçilmektedir. Bir uygulamanın dağıtık olarak değerlendirilebilmesi için aşağıda belirtilen kriterlere uyması gerekmektedir (Buterin, 2014).

- **Açık Kaynaklı:** Uygulama açık kaynaklı olmalıdır. Tüm kullanıcılar uygulamanın kaynak koduna ulaşabilmelidir. Değişikliklerin kullanıcılar tarafından fikir birliği ile belirlenmesi gerekmektedir.
- **Dağıtık olan:** Uygulama verileri şifrelenmiş şekilde blokzincir ağı üzerinde tutulmalıdır.
- **Kripto para kullanan:** Uygulama içerisinde düğümlerin ve madencilerin ödüllendirilmesi için şifrelenmiş kripto para birimi kullanmalıdır.
- **Protokol:** Uygulama ağında verinin ve kripto paranın yönetimi belirli bir mutabakat algoritması dâhilinde olmalıdır.

Dağıtık uygulamaların açık kaynaklı olması, uygulama kodlarının herkes tarafından görülebileceği ve katkıda bulunabileceği bir duruma getirir. Bu durum uygulamanın kapasitesini ve geliştirme sürecini hızlandırır.

Günümüzde en popüler dağıtık uygulama geliştirme ağı Ethereum olarak bilinmektedir. Blokzincir teknolojisinin kripto para birimlerinin altyapısı olması dışında birçok alanda kullanılabileceği potansiyelini ortaya çıkaran platform olarak kabul edilmektedir. Ağda bulunan kullanıcı sayısı, güncel uygulama sayıları, dijital cüzdan uyum yeteneği ve kendi altyapısı olan Ethereum Sanal Makinesi (EVM) üzerine kurulmuş olması diğer DApp uygulama platformlarından daha fazla tercih edilmesine sebep olmuştur. Bu sebeple tez kapsamında geliştirilen uygulama için Ethereum platformu tercih edilmiştir. Ethereum ağının yanı sıra diğer popüler olan DApp platformları EOS, TRON, NEO olarak sıralanabilir (DappRadar, 2018).

Bu bölümde dağıtık uygulamaların özelliklerinin yanı sıra çalışma kapsamında geliştirilen uygulamada tercih edilen, dağıtık uygulama geliştirme platformu Ethereum hakkında teknik bilgilere yer verilmiştir.

3.1. Ethereum Platformu

Ethereum dağıtık mimariye sahip ve üzerinde akıllı sözleşmeler kullanılabilen blokzincir ağıdır. Bitcoin platformunun gerçek dünya ihtiyaçlarına cevap veremediğini savunan Vitalik Buterin tarafından 2013 yılında kullanıma sunulmuştur. Genel olarak bir kripto para gibi görülse de blokzincir teknolojisine köklü değişiklikler getirmiştir. Ethereum 'un getirdiği en büyük yenilik, üzerinde kullanılan akıllı sözleşmeler ile dağıtık uygulamaların geliştirilmesine imkân sağlamasıdır. Bu sayede kullanıcıların kendi uygulamalarını geliştirebilmesi ve Ethereum ağında çalıştırabilmesi sağlanmıştır. Blokzincirin temelini oluşturan merkeziyetsizlik ve açık kaynak özelliklerinin tümüne sahiptir. Platform üzerinde gerçekleşen işlemler dağıtık ve anonim bir şekilde birden fazla bilgisayar üzerinde saklanmaktadır. Bu sayede işlem güvenliği üst düzeydedir. Verilerin ele geçirilmesi için ağın %51 oranındaki bilgisayarı ele geçirmek gerekmektedir. Bu durum bilgisayarların farklı lokasyon ve ağda bulunmasından dolayı mümkün görülmemektedir.

Ethereum, Ether (ETH) adında kendine ait kripto para kullanmaktadır. Kendi içerisindeki uygulamaların çalıştırılması ve uygulama içi yapılan işlemlerin ücretlendirilmesinde kullanılmaktadır. Bunun yanı sıra platform üzerinde kötü niyetli ve hatalı işlemlerin önüne geçmek amaçlıda kullanılmaktadır. Ethereum kurucusu Vitalik bir konuşmasında Ethereum' un sahip olduğu teknoloji ile dünyanın internet sistemindeki kaynağı olacağını söyleyerek dünyadaki petrole benzetmiştir. Bu platformun ihtiyaç duyduğu enerjisinin ise Ether olacağını söylemiştir.

Ethereum, Bitcoin 'de olduğu gibi PoW mutabakat algoritmasını kullanmaktadır. Bitcoin protokolünde bir blok ortalama 10 dakikada oluşurken Ethereum da bu süre ortalama 15 saniyedir. Bu süreler, herhangi bir ödeme işlemi yapıldığında ödemenin karşı tarafa ulaşma süresi olarak değerlendirilebilir. İki kripto paranın aynı mutabakat algoritmasını kullanması ve farklı blok onaylama sürelerinin olmasının nedeni, Ethereum platformunun kullandığı GHOST protokolüdür. En ideal ve uzun zinciri bulmak için düğümler üzerindeki blokları ebeveyn (parent) ve amca (uncle) blok olarak adlandırılıp, bu bloklar üzerinde yedi adımdan oluşan kuralları uygulayarak işlem hızını arttırmak amaçlanmaktadır. GHOST protokolüne karşı bazı kesimler tarafından güvenlik eleştirileri vardır. Bu konu ile alakalı farklı çözüm algoritmaları geliştirilse de yakın zamanda Ethereum protokolü, PoW mutabakat algoritmasını kullanımından kaldırarak PoS mutabakat algoritmasına geçeceğini duyurmuştur (Dannen, 2017).

Ethereum ’un tasarımında aşağıdaki ilkeler amaçlanmıştır (Buterin, 2014).

- **Basitlilik:** Ethereum geliştirmesi yapan yazılımcılar tüm içerikleri takip edebilmeli ve uygulayabilmelidir.
- **Evrensellik:** Yazılımcıların kendi finansal uygulamalarını, kripto paralarını ve birçok uygulamayı geliştirebileceği script programlama dili içermektedir.
- **Modüler Olmak:** Ethereum bölümleri olduğunca birbirinden bağımsız modüler yapıda olmalıdır. Bu şekilde olası bir protokol değişikliğinde uygulamanın çalışması devam etmelidir.
- **Çeviklik:** Ethereum mimarisi değiştirilebilir yapıda olmalıdır. Temel yapı taşları olan EVM ve protokollerde güvenlik değişiklikleri yapılabilmelidir.
- **Ayrımcılık yapmama ve Sansürsüz Olmak:** Platform tüm kullanım kategorilerine açık olmalıdır. Uygulama temelli engelleme olmamalıdır. Gerekli ödemeler sonrası her yazılımcı Ethereum üzerinde script çalıştırabilir.

Ethereum ağ yapısında yapılanlar işlemler ve akıllı sözleşmeler düğümler üzerinde bulunan EVM adı verilen sanal makinalarda çalıştırılmaktadır. İşlemlerin çalışabilmesi için Ethereum ağı tarafından ne kadar işlem yapılması gerektiğini ölçmek için gas denilen birim kullanılmaktadır. Bu değer ile işlemin gerçekleşmesi için gereken ücret hesaplanmaktadır. EVM ve gas hesaplama işlemlerinin yanı sıra Ethereum’ un teknik altyapısı ve akıllı sözleşmeler bu bölümde detaylı olarak ele alınmıştır.

3.1.1. Ethereum sanal makinesi (Ethereum virtual machine, EVM)

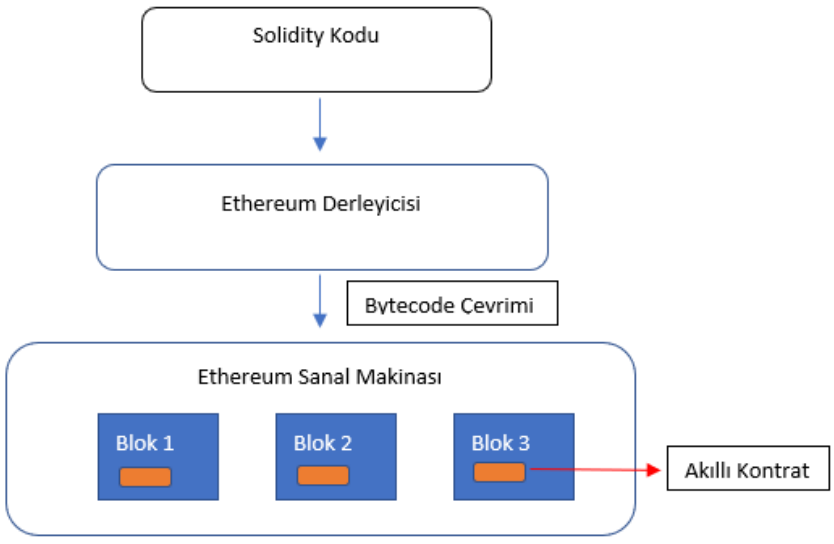
EVM, Ethereum platformda işlemlerin tutarlı ve hızlı çalışmasını, kullanıcı dostu olmasını sağlayan en önemli yapı taşıdır. Her bir düğüm içinde bulunan, yazılımcılar tarafından yazılan akıllı sözleşmelerin bayt koduna çevrilmesinden sorumlu olan güvenli sanal makinalardır. Sözleşmeler genellikle Solidity gibi üst düzey dillerde yazılmaktadır (Dannen, 2017). Yazılan sözleşmeler EVM aracılığı ile bayt koduna derlenmektedir. EVM C++, Java, JavaScript, Python ve Ruby gibi programlama dillerinde başarılı bir şekilde çalışabilmektedir.

EVM yürütme kodu ile yürütme makinası arasında bir soyutlama seviyesi oluşturur. Bu sayede geliştirilen yazılımların taşınabilirliği ve uygulamaların birbirinden bağımsız çalışabilmesini sağlamaktadır.

Ethereum ağında yapılan her işlem için EVM aşağıdaki adımları gerçekleştirmektedir (Dannen, 2017).

1. İşlemin doğru olup olmadığını kontrol edilir. İşlemin doğru sayıda değere sahip olup olmadığı, imzanın geçerliliği, işlemdeki nonce değeri ile hesaptaki nonce değerinin eşleşme durumları kontrol edilmektedir. Herhangi bir eksiklik durumunda hata oluşur.
2. İşlem içerisindeki kullanılan metotların maliyeti hesaplanarak işlem ücreti çıkarılır. Kullanıcının hesap bakiyesinden ilgili ücret düşürülür. Hesapta yeterli Ether yoksa hata oluşur.
3. Gas ödemesi başlatırlar.
4. İşlem ücreti alıcı hesaba aktarılır. Alıcı hesabı henüz mevcut değil ise oluşturulacaktır. Alıcı adresi bir sözleşme adresi ise sözleşmenin kodu çalıştırılır. Gas ödemesi bitene kadar kod yürütülmesi devam eder.
5. Gönderici hesapta işlemi tamamlamak için yeterli miktarda Ether yoksa veya gas biterse, tüm değişiklikler geri alınır.
6. İşlem farklı bir sebep ile hata verirse, gas göndericiye iade edilir. Kullanılan gas miktarları madenciye gönderilir.

EVM mimarisi Şekil 3.1 ‘de verilmiştir.



Şekil 3.1. Evm çalışma mimarisi (Paul, 2019).

3.1.2. Ethereum işlemler ve mesajlar

Ethereum tarafından iletilen veya blokzincire kaydedilen, hesaplar tarafından kriptografik olarak imzalanmış mesajlara işlem denir. Ethereum' da sözleşmelerin yürütebilmesi için herhangi bir işlem tarafından tetiklenmesi gerekmektedir. Bir işlem aşağıdaki özellikleri içermektedir (Buterin, 2014).

- Bir alıcı adresi; sözleşmenin gelecekte erişebileceği sözleşme adresi döndürülür.
- Göndereni tanımlayan imza
- Gönderilen miktar
- İsteğe bağlı kullanılabilecek veri alanı
- İşlem başına harcanan maksimum gas miktarı değeri
- Gönderenin ödemek istediği gaz fiyatı

Mesajlar sözleşmeler arası gönderilen veri yığınlarıdır. Sadece EVM' de bulunun sanal nesnelerdir. Sözleşmeler birbirleri ile iletişime mesajlar aracılığı ile geçebilir. Ethereum ağında bulunan madenciye ödeme yapıldığında, madencinin hesabındaki artış mesaj yolu ile gerçekleştirilir. Gerçekleşen bu olay işlem değildir. EVM tarafından bir sözleşme yürütüldüğünde bir mesaj gönderilir. Bu mesaj ile birlikte CALL ve DELEGATECALL kodları yürütülür. Bir mesaj aşağıdaki özellikleri içermektedir (Buterin, 2014).

- Mesajı gönderen adres
- Mesajın alıcı adresi
- Değer alanı (Gönderilen Ether miktarı olabilir)
- İsteğe bağlı kullanılabilecek değer alanı
- Mesaj başına kullanılabilecek maksimum gaz miktarı değeri

3.1.3. Ethereum blok mimarisi

Ethereum platformunda yapılan işlemler ve akıllı sözleşmeler bloklar üzerinde tutulmaktadır. Ethereum blok yapısında Bitcoin blok yapısından farklı olarak üç ayrı Merkle/Patricia ağacı yapısı bulunmaktadır. Bloklar genellikle aşağıdaki alanlardan oluşmaktadır (Antonopoulos ve Wood, 2018);

- **Önceki Blok Özet Değeri:** Bir önceki bloğun özet değeridir. Bu şekilde bloklar arası zincir oluşturulur.
- **Madenci:** Bloğu oluşturan madenci hesabıdır.

- **Durum (State) Ağacı Kök Değeri:** Hesap adresleri, hesap parametreleri ve akıllı sözleşme verilerinin tutulduğu yapıdır. Ağaç yapısı içerisinde Nonce, balance, storageRoot ve CodeHash bilgileri tutulmaktadır. Burada storageRoot farklı bir Merkle/Patricia ağacını referans tutmaktadır. Bu referans ile ilgili bloktaki akıllı sözleşmelerin tüm verilerine ulaşılabilmektedir.
- **İşlem (Transaction) Ağacı Kök Değeri:** Blok içerisindeki işlemlerin kök düğümlerinin Keccak 256-bit algoritması ile özetlenmiş değeridir.
- **Makbuz (Receipts) Ağacı Kök Değeri**: Ethereum platformunda bir işlem yürütüldüğünde, ilgili işlem hakkında bilgi içeren bir makbuz oluşturulur. Bu alanda işlem makbuzlarının kök düğümlerinin özet değeri mevcuttur.
- **Zaman Damgası:** Bloğu başlatma zamanıdır.
- **Zorluk:** Bloğun oluşturma süresini değiştirmek için kullanılan sayısal parametredir.
- **Nonce Değeri:** Bloğun düzgün bir şekilde oluştuğunu doğrulmak için kullanılan değer.
- **Gas Limiti:** Blok içinde kullanılacak maksimum gas limitidir.
- **Gas Kullanımı:** Bloktaki toplam kullanılan gas miktarıdır.
- **Ekstra Veri:** Blok ile alakalı ekstra tutulacak bilgileri içeren alandır.
- **Numara:** Mevcut blok sayısını gösterir.

3.1.4. Akıllı sözleşmeler (Smart contract)

Akıllı sözleşmeler, üçüncü taraflara gerek kalmadan kişiler arası işlemlerin güvenilir şekilde yapılmasını sağlayan programlanabilir bilgisayar protokolleridir. İlk olarak 1994 yılında Nick Szabo tarafından sözleşme şartlarını yürüten bilgisayar protokolü olarak tanıtılmıştır (Christidis ve Devetsikiotis, 2016). Günümüzde Ethereum platformunda yaygın olarak kullanılmakla birlikte birçok protokol ve akışın programlanabilmesine olanak sağlamıştır.

Akıllı sözleşmeler üst düzey yazılım dilleri kullanılarak yazılmaktadır. Günümüzde yaygın olarak Ethereum' un kendi programlama dili olan Solidity kullanılmaktadır. Programcı tarafından yazılan sözleşme yayınlanmadan önce derlenerek bytecode a çevrilir ve Ethereum blok zincirine işlem olarak gönderilir. Her sözleşme gönderici hesabı ve nonce değeri kullanılarak üretilen bir Ethereum adresi ile tanımlanır. Tanımlanan adres ile bloğa eklenir. Bir sözleşmenin Ethereum adresi herhangi bir işlemde alıcı olarak kullanılabilir. Sözleşmeye para gönderebilir veya sözleşme işlevlerinden biri çağrılabilir. Sözleşme ağda bir kez yayınlandıktan sonra, yayınlanan sözleşmenin içeriği değiştirilememektedir. Sözleşme içeriğinde herhangi bir

güncelleme yapılırsa, sözleşmenin son güncel hali ağda paylaşılarak kullanıma alınır. Eğer akıllı sözleşmede kendi kendini yok etme fonksiyonu tanımlı değil ise sözleşmelerin eski versiyonları blok zincirde kalacaktır (Antonopoulos ve Wood, 2018).

Akıllı sözleşmeler sadece bir işlem tarafından tetiklendiğinde çalışmaktadır. Sözleşmeler yazılımsal olarak birbirlerini çağırabilirler. Bu durumun oluşması için ilk sözleşmeyi mutlaka bir kullanıcı tetiklemelidir. Sözleşmeler kendi kendilerine çalışamazlar. Herhangi bir programcı tarafından arka planda yürütülemezler. Herhangi bir kullanıcı tarafından veya başka bir sözleşme tarafından tetiklenene kadar sözleşme durgun bir şekilde bekler (Antonopoulos ve Wood, 2018).

Akıllı sözleşmeler deterministtik olarak çalışmaktadır. Bir sözleşmenin yürütülmesi, sözleşmeyi yürüten tüm kullanıcılar için aynı sonucu vermektedir. Yürütme işlemi başarılı bir şekilde sonuçlanırsa bloklara eklenir. Bloğa eklenmiş her işlem yürütmede herhangi bir hata alınmadığının göstergesidir. Herhangi bir işlemde hata ile karşılaşılırsa, işlemin etkilediği tüm durumlar geri alınır. Ancak yürütmek için harcanan gas maliyeti istemci hesabından düşülür. Akıllı sözleşmeleri ücretsiz değildir. İstemcinin sözleşmenin herhangi bir metodunu kullanabilmesi için düşük de olsa bir gas ücreti ödemesi gerekmektedir.

Akıllı sözleşmeler manuel işlemlerin otomatik olarak yapılmasını sağlamak için elverişli bir yapıdır. Manuel yapılan işlemlerde ortaya çıkan hatalara karşı çok daha dayanıklıdır. Güven amaçlı üçüncü taraflara duyulan ihtiyacı ortadan kaldırdığından dolayı daha düşük maliyetlidir. Akıllı sözleşmelerin hayatımıza getirdiği birçok kolaylığın yanı sıra dezavantaj oluşturabilecek durumlarının olduğunu hatırlamak gerekir. Sözleşme içerisinde kullanılan her fonksiyonel metodun bir ücret karşılığı olduğu için yazılımsal olarak geri döndürülemez hatalar içerebilir. Bu sebeple akıllı sözleşmeler programlanmadan önce çok iyi analiz edilmelidir. Blok zincire yazılan sözleşmeler değiştirilemez olması nedeniyle tüm olası senaryolar önceden kurgulanmalıdır.

Akıllı sözleşmeler için kullanılan yüksek düzey bazı programlama dilleri aşağıda belirtilmiştir (Antonopoulos ve Wood, 2018);

LLL: Lisp programlama diline benzer söz dizimine sahip programlama dilidir. Ethereum akıllı sözleşmeleri için kullanılan ilk üst düzey programlama dilidir. Günümüzde nadiren kullanılmaktadır.

Serpent: Phyton diline benzer söz dizimine sahip yordamsal bir programlama dilidir.

Solidity: Javascript, C++ veya Java ya benzer bir söz dizimine sahip yordamsal programlama dilidir. Ethereum akıllı sözleşmeleri için kullanılan en popüler programlama dilidir.

Tez kapsamında yazılan uygulamanın akıllı sözleşme tarafı Solidity dili kullanılarak kodlanmıştır.

Vyper: Phyton diline benzer söz dizimine sahiptir. Serpent dilinden daha kolay kodlanabilir olması amacı ile geliştirilmiştir. Fakat yerine geçememiştir.

Bamboo: Erlang programlama dilinden etkilenmiş, yeni geliştirilmiş programlama dilidir. Sözleşmelerde yaşanan yan etkileri azaltmak amaçlıdır. Henüz yaygın olarak benimsenmemiştir.

Akıllı sözleşmeler kodlanırken güvenlik en önemli hususlardan biridir. Sözleşmede oluşabilecek hatalar maliyetlidir. Herhangi bir işlemin yarıda kesilmesi işlemi çağıran kullanıcıdan işlem ücreti kesintisine sebep olmaktadır. Oluşan kayıpların geri getirilmesi imkânsızdır. Bu nedenle akıllı sözleşme geliştirilirken yazım kurallarına dikkat edilmeli, tasarım desenleri kullanılmalıdır. Akıllı sözleşme yazılırken dikkat edilmesi gereken hususlar aşağıdaki gibidir (Antonopoulos ve Wood, 2018);

Basitlik: Karmaşık bir kod güvenliğin zayıflamasına neden olabilmektedir. Yazılacak kodun basit ve anlaşılabilir olması hata yapma şansını ve öngörülemeyen bir etki oluşmasını engelleyebilir.

Kod tekrarının önlenmesi: Akıllı sözleşmelerde işlem ücreti yürütülen satır başına alınmaktadır. Bu nedenle tekrar eden kod bloklarından kaçınılmalıdır. Tekrar eden blokların fonksiyon veya kütüphane olarak kullanılabilirliği üzerine çalışma yapılmalıdır.

Kod kalitesi: Hatalar parasal kayıplara yol açacağından akıllı sözleşme geliştirmesi genel amaçlı programlamadan farklıdır. Yazılan kodun defalarca üzerinden geçilmelidir. Yazılım geliştirme teknikleri ve titiz mühendislik uygulanmalıdır.

Okunabilirlik/Denetlenebilirlik: Akıllı sözleşmeler herkese açıktır. Bayt koduna çevrilen sözleşmeler tersine işlem yapılarak okunabilir hale gelebilir. Bu nedenle açık kaynak metodolojileri kullanarak geliştirme yapılmalıdır. Ethereum topluluğunun stil ve adlandırma kuralları izlenerek kodlama yapılmalıdır.

Test işlemleri: Yazılan her kod test edilmelidir. Akıllı sözleşmeler herkese açık olduğundan her kullanıcı istediği bir sözleşmeyi girdi ile yürütebilir. Bu nedenle program girdilerinin düzgün bir şekilde sınırlandırıldığının, beklenen aralıkların belirlenip belirlenmediğinin test edilmesi gerekmektedir.

3.1.5. Gas ve Gaslimit değerleri

Ethereum ağındaki işlemlerin ve akıllı sözleşmelerin çalışması için gerekli olan değeri ölçen birime gas denir. Ethereum sanal makinesi tarafından yürütülen her işlem için, işlem sahibi tarafından ödeme yapılması gerekmektedir. İşlem başına ödenecek ücret akıllı sözleşme üzerinde kullanılan operasyon kodlarının toplam gas değeri ile gas fiyatının çarpılması sonucu elde edilir. Sonuç olarak bir işlem sonucunda ödenecek ücretin hesaplanmasında aşağıdaki formül kullanılır (Özcan, 2019).

Toplam Ödenecek Ücret= Toplam Gas * Gas Fiyatı (1)

Ethereum ağındaki işlemler Ether olarak ücretlendirilir. Ether' in ana birimi Wei olarak adlandırılır. Çizelge 3.1' de Ether alt birimlerinin değerleri verilmiştir. Bu değerler ile birlikte işlemler için gerekli Ether ücretleri birbirleri arasında dönüşüm yapılarak hesaplanmaktadır. Gaz fiyatının ölçü birimi Gwei olarak belirlenmiştir. Ortalama gaz fiyatı 20 Gwei (0.00000002 ETH) civarındadır. Bu fiyat yüksek ağ trafiği zamanlarında, yeni blok oluşturma rekabeti nedeniyle artabilir (Rosic, 2018).

Çizelge 3.1. Ether alt birimleri ve değerleri (Özcan, 2019).

Birim	**Wei Değeri**	**Wei**
wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000

Akıllı sözleşmeler içerisinde kullanılan ve EVM içerisinde yürütülen kodlar mevcuttur. Her kodun bir Gas değeri vardır. Yürütülen bir akıllı sözleşme içerisinde kullanılmış olan operasyonel kodların gas değerleri toplanarak, işlem için gerekli olan toplam gas değeri bulunur. Bu değer Denklem 1' de bulunan toplam gas değerini ifade etmektedir. Çizelge 3.2' de EVM içerisinde kullanılan Opcode' ların açıklamaları ve gas değerleri verilmiştir.

Ethereum' da bir işlemin yapılabilmesi için, işlem sahibinin bir gas limiti belirtmesi gerekmektedir. Gaz limiti, bir kullanıcının o işlemi yapabilmesi için ödemek istediği en yüksek gas miktarıdır. Bu sayede işlem yapan kullanıcı, sözleşme ile alakalı bir şeyler ters gittiğinde planlanan gas ödemesinden daha fazla ödeme yapmak durumunda kalmasının önüne geçilmiş olur. Sonuç olarak belirtilmiş gas miktarından sonra işlemin durdurulacağı garanti edilmiştir. Gaz

limiti manuel belirleneceği gibi günümüzde mevcut olan çoğu cüzdan bu işlemi otomatik yapmaktadır. Gas limiti belirtmede iki farklı senaryo mevcuttur. Birinci senaryo gas limitinin çok düşük belirtilmesidir. Bu durumda işlem başarısız olmakla birlikte madencilerin hesaplama maliyetini işlem sahibinin hesabından düşmektedir. İkinci senaryo ise gas limitinin çok yüksek belirtilmesidir. Bu durumda işlem başarılı bir şekilde gerçekleştirilmekle beraber kullanılmayan gas göndericiye geri gönderilmektedir. İşlemler için gas limitinin çok yüksek belirlenmesinin bir avantajı yoktur. İşlemler için gas limiti olduğu gibi bloklar içinde bir gas limiti mevcuttur. Bu limit blok yüksekliğini belirlemektedir. Bu yüzden işlemler için çok yüksek gas limitleri belirtmek mantıklı değildir (Ergin, 2018).

Çizelge 3.2. Ethereum Sanal Makinesi opcode gas maliyetleri (Djrtwo, 2017).

Hx Değeri	**Opcode**	**Gas Değeri**	**Açıklama**
0x00	STOP	0	Yürütme işlemini durdurur
0x01	ADD	3	Toplama işlemi
0x02	MUL	5	Çarpma işlemi
0x03	SUB	3	Çıkarma işlemi
0x04	DIV	5	Tamsayı bölme işlemi
0x05	SDIV	5	İşaretli tamsayı bölme işlemi
0x06	MOD	5	Mod işlemi
0x07	SMOD	5	İmzalı mod işlemi
0x08	ADDMOD	8	Mod toplama işlemi
0x09	MULMOD	8	Mod çarpma işlemi
0x0a	EXP	FORMULA	Üstel işlemler
0x0b	SIGNEXTEND	5	y = SIGNEXTEND(x, b) kullanımı ile işaretli x' i (b + 1) * işlemine göre 8 bit' ten 256 bite uzatır.
0x10	LT	3	Karşılaştırmada kullanılan daha az ifadesi
0x11	GT	3	Karşılaştırmada kullanılan daha büyük ifadesi
0x12	SLT	3	Karşılaştırmada kullanılan daha az ifadesi (imzalı)
0x13	SGT	3	Karşılaştırmada kullanılan daha az ifadesi (imzalı)
0x14	EQ	3	Eşitlik karşılaştırması
0x15	ISZERO	3	NOT operatörü
0x16	AND	3	AND operatörü
0x17	OR	3	OR operatörü
0x18	XOR	3	XOR operatörü
0x19	NOT	3	NOT operatörü
0x1a	BYTE	3	Kelimeden tek bayt çeker
0x20	SHA3	FORMULA	Keccak-256 hash değerini hesaplar
0x30	ADDRESS	2	İşlem sırasında kullanılan hesabın adresini verir
0x31	BALANCE	400	İstenilen hesabın bakiyesini verir
0x32	ORIGIN	2	Yürütülen işlemin kaynak adresini verir
0x33	CALLER	2	İşlemi çağıran kişinin adresini verir
0x34	CALLVALUE	2	İşlemin yürütülmesi için yatırılan değeri verir

Çizelge 3.2. (devam) Ethereum Sanal Makinesi opcode gas maliyetleri.

0x35	CALLDATALOAD	3	Ortamın girdi verilerini almak için kullanılır
0x36	CALLDATASIZE	2	Girdi verilerinin boyutunu verir
0x37	CALLDATACOPY	FORMULA	Girdi verilerini belleğe kopyalar
0x38	CODESIZE	2	Çalışan kodun boyutunu verir
0x39	CODECOPY	FORMULA	Çalışan kodu belleğe kopyalar
0x3a	GASPRICE	2	Ortamdaki gasın fiyatını verir
0x3b	EXTCODESIZE	700	Hesap kodunun boyutunu verir
0x3c	EXTCODECOPY	FORMULA	Hesap kodunu belleğe kopyalar
0x40	BLOCKHASH	20	En son tamamlanmış bir bloğun hash değerini verir
0x41	COINBASE	2	Bloğu oluşturan kişinin adresini verir
0x42	TIMESTAMP	2	Bloğun zaman damgasını verir
0x43	NUMBER	2	Bloğun numarasını verir
0x44	DIFFICULTY	2	Bloğun zorluk değerini verir
0x45	GASLIMIT	2	Bloğun gaz limit değerini verir
0x50	POP	2	Bellekten kelime kaldırır
0x51	MLOAD	3	Bellekten kelime yükler
0x52	MSTORE	3	Kelimeyi belleğe kaydeder
0x53	MSTORE8	3	Belleğe byte kaydeder
0x54	SLOAD	200	Depolama alanından kelime yükler
0x55	SSTORE	FORMULA	Depolama alanına kelime yükler
0x56	JUMP	8	Program sayacını değiştirir
0x57	JUMPI	10	Sayacı koşula bağlı olarak değiştirir
0x58	PC	2	Sayacın değerini verir
0x59	MSIZE	2	Aktif belleğin bayt cinsinden değerini verir
0x5a	GAS	2	Mevcut gas miktarını verir
0x5b	JUMPDEST	1	Atamalar için geçerli bir hedef işaretler
0x60 -- 0x7f	PUSH*	3	1-32 bayt arası veriyi yığına yerleştirir.
0x80 -- 0x8f	DUP*	3	1-16 arası elemanı yineler
0x90 -- 0x9f	SWAP*	3	1-16 arası elemanların yerlerini değiştirmek için kullanılır
0xa0	LOG0	FORMULA	Konu içermeyen günlük log kaydı ekler
0xa1	LOG1	FORMULA	Bir konu içeren günlük log kaydı ekler
0xa2	LOG2	FORMULA	İki konu içeren günlük log kaydı ekler
0xa3	LOG3	FORMULA	Üç konu içeren günlük log kaydı ekler
0xa4	LOG4	FORMULA	Dört konu içeren günlük log kaydı ekler
0xf0	CREATE	32000	İlgili kod ile yeni bir hesap oluşturur
0xf1	CALL	FORMULA	Herhangi bir hesaba mesaj çağrısı oluşturur
0xf2	CALLCODE	FORMULA	Alternatif hesap kodu ile mesaj çağırısı oluşturulur
0xf3	RETURN	0	İşlem sonucu oluşan veriyi döndürür ve işlemi durdurur
0xf4	DELEGATECALL	FORMULA	Alternatif bir hesap kodu ile mesaj çağrısı yapar, ancak mevcut değerleri korur
0xfe	INVALID	NA	Belirlenmiş geçersiz talimat
0xff	SELFDESTRUCT	FORMULA	Hesabın kendini durdurması ve daha sonra kendini yok etmesi için kullanılır

3.1.6. Ethereum geliştirme ağı ortamları

Ethereum platformunda geliştirilen bir DApp uygulamasının veya akıllı sözleşmenin çalıştırılabilmesi için Ethereum ağına ihtiyaç vardır. Ethereum ağları; Mainnet ve Testnet olarak iki başlık altında toplanabilir.

Mainnet, kullanıcıların yaptıkları işlemlerin tutulduğu, canlı verilerin işlendiği gerçek ortamdır. Ağa paylaşılan veriler düğümler tarafından bloklarda saklanmaktadır. Gerçek ortamda yapılan her işlem için ücret talep edilmektedir. Yapılan hatalı bir işlemin geri dönüşü yoktur. Bu sebeple herhangi bir akıllı sözleşme veya DApp gerçek ağda yayınlanmadan önce test ağlarında çalıştırılmalıdır. Günümüzde farklı diller ile yazılmış birçok gerçek ağ ortamı mevcuttur. En yaygın olarak kullanılan beş ağ aşağıdaki gibidir (Ethereum Mainnet Statistics, 2020).

Geth: Go programlama dili ile yazılmıştır.

Parity: Rust programlama dili ile yazılmıştır.

Nethermind: .NET Core programlama ile yazılmıştır.

Besu: Java programlama dili ile yazılmıştır.

Openethereum: Rust programlama dili ile yazılmıştır.

Testnet, kullanıcıların gerçek ortama çıkmadan önce test işlemlerini yaptıkları ağlardır. Bu ağlar üzerinde yapılan işlemlerde gerçek para kullanılmaz. Programcılar geliştirdikleri DApp veya akıllı sözleşmeleri test ağları üzerinde çalıştırarak, canlı ortamda yaşanacak birçok sorunun önüne geçme avantajını elde etmektedir. Test ağları genel (Public) ve yerel (Local) olarak ikiye ayrılabilir. Genel ağlar, gerçek kullanıcıların olduğu ve çevrimiçi olarak takip edilebilen ortamlardır. Gerçekleştirilen işlemler diğer kullanıcıların işlemleri ile birlikte aynı blok zincire eklenerek işlem görür. En yaygın olarak kullanılan public test ağları aşağıdaki gibidir.

Ropsten Test Ağı: 2016 yılında kullanıma alınan, Ethereum gerçek ortam ağına en yakın ağdır. PoW mutabakat algoritması kullanılmaktadır. Geth ve Parity ağları ile çalışabilir. Blok oluşma süresi yaklaşık 30 saniyedir. Ağ kimlik idsi 3 olarak tanımlanmıştır. Spam saldırılarına karşı az dirençlidir. Ether üretilebilir veya kaynaklardan talep edilebilir.

Kovan Test Ağı: 2017 yılında kullanıma alınan, PoA mutabakat algoritmasını kullanan test ağıdır. Ağ kimlik idsi 42 olarak tanımlanmıştır. Ortalama blok oluşma süresi 4 saniyedir. Spam saldırılarına karşı dirençlidir. Geth desteklenmez. Ether üretilemez ve sadece kaynaklardan talep edilerek elde edilebilir.

Rinkeby Test Ağı: 2017 yılında kullanıma alınan, PoA mutabakat algoritmasını kullanan test ağıdır. Ağ kimliği 4 olarak tanımlanmıştır. Ortalama blok oluşturma süresi 15 saniyedir. Sadece Geth tarafından desteklenmektedir. Spam saldırılarına karşı dirençlidir. Ether üretilemez ve sadece kaynaklardan talep edilebilir.

Görli Test Ağı: 2018 yılında kullanıma alınmıştır. PoA mutabakat algoritmasını kullanmaktadır. Ağ kimlik idsi 5 olarak tanımlanmıştır. Ortalama blok oluşma süresi 15 saniyedir.

Genel test ortamlarının yanı sıra dış dünyaya kapalı, programcının kendi bilgisayarı üzerinde çalıştırabileceği yerel Ethereum blokzinciri oluşturmayı sağlayan simülasyonlar vardır. Bunlar arasında en popüler olan Ganache isimli uygulamadır (Ganache, 2017). Tez kapsamında kodlanan uygulama Ganache ortamı üzerinde geliştirilmiştir. Ganache ortamı ilerleyen bölümlerde detaylı olarak ele alınmıştır.

4. MEVCUT CARİ MUTABAKAT SÜREÇLERİ VE SORUNLARI

Bu bölümde tez kapsamında ele alınan cari mutabakat yöntemleri ve sonrasında gerçekleştirilen ödeme işleminde yaşanan problemler detaylandırılmıştır.

4.1. Cari Mutabakat Tanımı ve İşletmeler İçin Önemi

İşletmelerin, ticari ilişkide oldukları diğer işletmelerle alacak veya borç bakiyelerinin karşılaştırılmasına cari mutabakat adı verilmektedir. İşletmeler için oldukça önemli olan cari mutabakat, ticari ilişkilerin sağlıklı bir şekilde devam etmesini garanti altına almaktadır. Taraflar muhasebe hesaplarının detaylı analizi sonrası ortaya çıkan cari hesap bakiyelerini ve hesap ekstrelerini birbirleri ile paylaşarak hatalı olan kayıtların düzeltilmesini sağlamaktadır. Bu sayede cari hesaplardaki tutarsızlıkların önüne geçilmiş olunur (Schultz, 2019).

Cari mutabakatlar düzenli aralıklarla yapılmaktadır. Bazı işletmeler her ayın sonunda mutabakat işlemi yaparken, bazı işletmeler ise dönemsel olarak yapmaktadır. Bu durum 6102 sayılı Türk Ticaret Kanunu' nun 94. Maddesi ile zorunlu hale getirilmiştir. İlgili yasada taraflar arasında belirlenmiş hesap devreleri sonunda kapatılacak şekilde alacak veya borç kalemlerinin belirlenmesi, eğer bir tarih belirlenmemiş ise her takvim yılının son günü olacak şekilde hesabın kapatılması belirtilmiştir. Kanun gereği karşı taraf mutabakatı 1 ay içerisinde cevaplamaması durumunda bakiyeyi kabul etmiş sayılır.

Mutabakat mektubu, dönem sonları hesapların analizi sonucu ortaya çıkmış borç veya alacak bilgisinin yer aldığı bilgilendirme formudur. Belli bir tarih aralığındaki cari hesap bakiye bilgilerinin olduğu mutabakat mektubu, iki işletme arasındaki mutabık olma durumu için hazırlanmaktadır. Genellikle muhasebe birimi tarafından hazırlanan mektup, müşteri veya tedarikçiye iletilerek cevaplanması istenir. Müşteri veya tedarikçi mutabakat mektubu üzerinde belirtilmiş cari hesap bakiyesini, kendi cari hesap analizi sonucu ortaya çıkmış tutar ile karşılaştırmasını yapar. Eğer tutar üzerinde mutabık ise formu onayladığını bildirir. Mutabık olmama durumunda ise taraflar karşılıklı olarak cari hesap bakiye ve hesap ekstrelerini paylaşırlar. Hatalı veya eksik kayıtların tespitini yaparak tek bir tutar üzerinde mutabık olmaya çalışırlar. Bu durum işletmelerin cari hareketlerini tek tek kontrol etmelerini ve tutarsız kayıtların düzeltilmesini sağlamış olur (Schultz, 2019).

Günümüzde iki farklı şekilde mutabakat işlemleri yapılmaktadır. Birinci yöntem, muhasebe birimi tarafından muhasebe hesaplarının detaylı analizi sonrası ortaya çıkan borç veya alacak bakiye bilgisinin yer aldığı mutabakat mektubu hazırlanarak, karşı firmaya telefon, fax veya e-posta yolu ile iletilmesidir. Mutabakat üzerinde mutabık olunup olunmaması ile alakalı

süreç bu iletişim yolları ile tamamlanmaktadır. İkinci yöntem ise hazırlanan mutabakat mektubunun web üzerinde yayınlanmış bir yazılımsal platform aracılığı ile karşı firmayla paylaşılmasıdır (Schultz, 2020). Bu yönteme e-mutabakat adı verilmektedir. Günümüzde yazılım firmalarının geliştirdiği birçok e-mutabakat platformu mevcuttur. Çizelge 4.1' de iki yöntem arasındaki karşılaştırmaya yer verilmiştir.

Çizelge 4.1. Cari Mutabakat yöntemlerinin karşılaştırılması (Schultz, 2020).

Özellik	**Telefon, Fax veya E-Posta yolu ile Mutabakat**	**E-Mutabakat**
Maliyet	Kâğıt, arşivleme ve kargolama masrafları mevcuttur	Tüm işlemler çevrimiçi olarak tek bir platformdan gerçekleştirilir
Zaman	Hesapların analizi ve mutabakatın işletmeye ulaştırma süreçleri uzundur	ERP programları ile uyumlu çalışabilmesi sayesinde hesapların çok kısa bir sürede analizi gerçekleştirilip işletmeye ulaştırılabilir.
Erişim	Geçmişe yönelik verilere arşiv üzerinden erişilmesi zaman kaybına neden olur	Elektronik ortamda saklanan verilere farklı yerlerden ve zamanlarda erişim sağlanabilir
Raporlama	Anlık rapor alma durumu yoktur	Anlık veya geçmiş yıllara ait raporlar hızlı bir şekilde alınabilir
Yasal Geçerlilik	Islak imza kullanımı dolayısı ile yasal geçerlidir	Kayıtlı Elektronik Posta (KEP) ile uyumlu çalışabilmesi sayesinde yasal geçerlilik sağlanabilir.

E-mutabakat ile işletmelerin karşılıklı olarak yaptıkları mutabakatların takibi ve yönetilmesi çok daha kolay olmuştur. Mutabakat mektuplarının kâğıt üzerinde hazırlanması ve bu formların arşivleme, kargo ve kâğıt gibi masraflarını ortadan kaldırmıştır. İşletmeler arasındaki geri dönüşleri hızlandırmıştır. Mükerrer kayıt hatalarının yanı sıra kurumsal görev değişikliklerinde ortaya çıkabilecek hafıza kayıplarını ortadan kaldırmıştır. ERP programları ile uyumlu çalışabilmesi sayesinde cari hesap hareketlerinin otomatik hesaplanması sağlanarak, insan kaynaklı hataların önüne geçilmiştir.

4.2. Cari Mutabakat İşlemlerinde Yaşanan Problemler

Günümüzde birçok firma tarafından cari mutabakatlar telefon, fax veya e-posta yöntemi ile yapılmaktadır. Bu yöntem birçok problemi beraberinde getirmiştir. Bu problemler aşağıdaki gibidir.

Maliyet problemi

Cari mutabakat mektuplarının kâğıt üzerinde hazırlanması ve müşteriye iletilmesi için kargo yönteminin kullanılması işletmeler için maliyet ortaya çıkarmaktadır (Tucker ve Cooper, 2009). Özellikle tedarikçi veya müşterisi fazla olan işletmeler her ay yaptıkları mutabakatlar sonrası ciddi bir mali harcama yapmaktadırlar.

Zaman problemi

Cari mutabakat mektupları hazırlanmadan önce muhasebe hesaplarının detaylı analizi yapılmaktadır. Hesapların müşteri bazında incelenmesi, çalışmayı yapan personele önemli ölçüde zaman kaybettirmektedir (Tucker ve Cooper, 2009). Çalışma tamamlandıktan sonra mutabakat mektubunun müşteriye kargo yolu ile gönderilmesi, kargo hareketlerinde yaşanabilecek olumsuzluklardan dolayı uzayabilmektedir.

Erişim problemi

Cari mutabakat mektuplarının kâğıt veya e-posta olarak arşivde tutulması, eski kayıtlara ulaşılmak istenildiğinde ciddi bir zaman kaybı yaşanmaktadır (Schultz, 2020). Ayrıca verilerin fiziksel olarak kaybolma veya e-posta ortamından silinme ihtimalleri vardır. İşletmelerin arşivleme ile alakalı ayrıca bir çalışma yapması gerekmektedir.

Raporlama problemi

Mutabakatlar ile alakalı anlık rapor alma durumu yoktur. Herhangi bir işletme ile alakalı detay raporu alınmak istendiğinde arşiv üzerinde detaylı çalışma yapmak gerekmektedir (Schultz, 2020). Bazı firmalar bu işlem için personel istihdamı yapmaktadır.

Takip ve geri dönüş problemi

Kullanılan bu yöntem ile gönderilen mutabakatların geri dönüşleri uzun sürmektedir. Ayrıca işletmeler mutabakat mektuplarının müşteriye ulaşma durumunu kontrol etmek zorunda kalmaktadır. İşletmeler sadece mutabık olmama durumu ile ilgilenmek yerine tüm olası durumlar ile ilgilenmek zorunda kalmaktadırlar.

Personel problemi

İşletmeler yukarıdaki problemlerin birçoğunun çözümü için personel istihdamı yapmaktadır. Bu durum işletmelere personel fazlalığı ve ekonomik yük getirmektedir (Tucker ve Cooper, 2009).

Mutabakat işlemleri için diğer bir yöntem olan E-mutabakat, yukarıdaki belirtilen problemlerden birçoğunu çözüme kavuşturmuş olmasına rağmen bazı problemleri beraberinde getirmiştir (Godambe ve Samudrala, 2017). Bu problemler aşağıdaki gibi başlıklandırılabilir.

Maliyet problemi

E-mutabakat programları, yıllık ödeme veya mutabakat adedi şeklinde satışı yapılmaktadır. Bu durum dönemsel mutabakat sayıları fazla olan işletmelerin maliyetini arttırmaktadır. Ayrıca ERP programları ile uyumlu çalışma veya bulut çözümleri gibi yazılım tabanlı talepler program ücretlerini yükseltmektedir.

Tek merkeze dayalı güven problemi

Finansal verilerin güvenilirliği, gizliliği ve şeffaf olması işletmeler için büyük öneme sahiptir. Verilerin değiştirilemez olması ve izlenebilirliği tarafların birbirine olan güven probleminin ortadan kalkmasını sağlayacaktır (Chang vd., 2020). Günümüzde kullanılan e-mutabakat programlarında oluşan verileri firmalar kendi bünyelerindeki sunucularda veya ücret karşılığında üçüncü taraflardan veri barındırma hizmeti alarak saklamaktadır. Bu durum verinin izlenebilirliğini ortadan kaldırmaktadır. Taraflar verilerin tutarlığını, veri barındırma hizmeti veren tarafın sorumluluğuna bırakmıştır. Herhangi bir güvenlik açığı ile verilerin ele geçirilmesi veya kötü amaçlı insanlarca değiştirilme ihtimalleri tek merkeze dayalı güven problemini ortaya çıkartmaktadır (Yaga vd., 2018).

Ödeme İşlemlerinde Yaşanan Problemler

Cari mutabakatlar sonrası işletmeler birbirleri arasındaki borç veya alacak durumunu bir tutar üzerinde netleştirirler. Belirlenen bu tutarın işletmeler arasındaki anlaşmaya göre tahsilatı gerçekleşir. Para transferleri banka yolu ile gerçekleştirilir. Mevcut bankacılık sisteminde para transfer süreçleri uzayabilmektedir. Örneğin; EFT para transferlerinde haftanın belirli günleri ve belirli saatleri arasında işlem yapılabilmektedir. Para transferlerinde yüksek komisyon ücretlerinin yanı sıra doğrulama ve kayıt süreçleri gibi prosedürler uygulanmaktadır. Bu durum işlemlerin uzamasını ve işletmeler için maliyeti arttırmaktadır (Guo ve Liang, 2016).

5. CARİ MUTABAKAT VE ÖDEME İŞLEMLERİ ETHEREUM TABANLI UYGULAMA ÖRNEĞİ

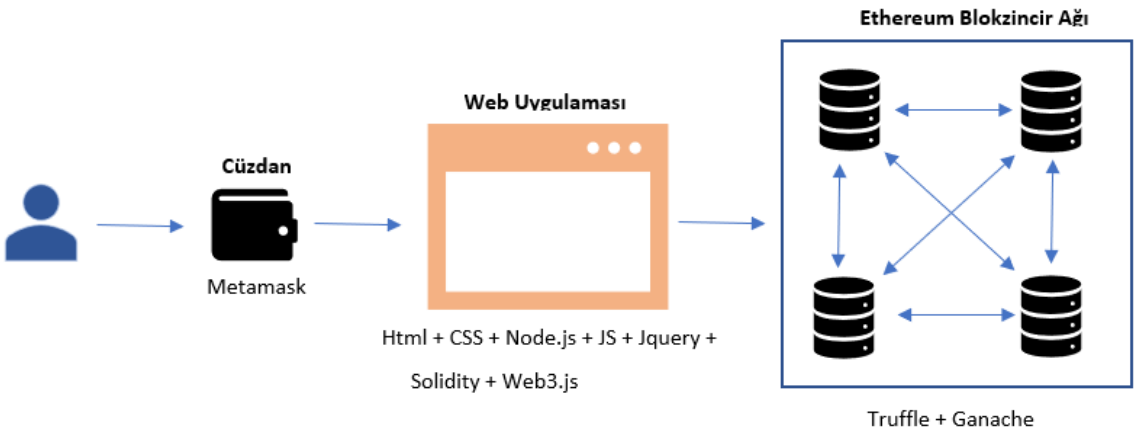
Blokzincir teknolojisinin önceki bölümlerde detaylandırılan mevcut cari mutabakat ve ödeme işlemlerindeki problemleri çözme üzerine etkisini görmek amacı ile dağıtık, Ethereum tabanlı DApp uygulaması geliştirilmiştir. Bu bölümde geliştirilen uygulamanın mimarisi, uygulama geliştirmede kullanılan araçlar, uygulama kapsamında geliştirilen akıllı sözleşme ve uygulamanın kullanımı ile alakalı detaylı bilgiler verilmiştir.

5.1. Uygulama Mimarisi

Tez kapsamında dağıtık, Ethereum tabanlı DApp uygulaması geliştirilmiştir. Uygulama kapsamında herhangi bir merkezi sunucu taraflı servis kullanılmamıştır. Son kullanıcının çevrimiçi olarak mutabakat işlemlerini gerçekleştirebilmesini sağlayan web uygulaması olarak geliştirilmiştir. Son kullanıcının uygulama arayüzünde oturum açabilmesi için tarayıcı eklentisi olarak çalışan Metamask programı tercih edilmiştir (Metamask, 2020). Metamask programı sayesinde kullanıcılar uygulamaya açık anahtarları ile bağlanabilmekte, kendisine ait mutabakat verilerini görüntüleyebilmektedir. Ayrıca mutabakat mektubu onayı veya ret işlemlerini cüzdan aracılığı ile yapabilmektedir. Uygulamanın Backend tarafında NodeJS teknolojisi kullanılırken, Frontend tarafında HTML5, Jquery ve Javascript teknolojileri kullanılmıştır. Web uygulamasının blokzincir ile veri transferi gerçekleştirebilmesi için Web3.js javascript kütüphanesinden yararlanılmıştır.

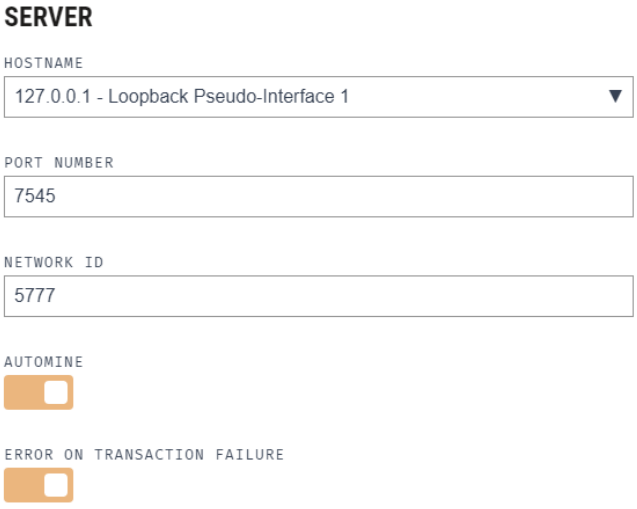
Blokzincir üzerinde kayıt işlemleri, onaylama işlemleri ve veri yönetimi akıllı sözleşme ile yönetilmektedir. Uygulama kapsamındaki akıllı sözleşme, Solidity programlama dili kullanılarak kodlanmıştır. Sözleşmeler proje içerisinde **/public/contracts** klasöründe bulunmaktadır. Uygulama katmanında kodlanan akıllı sözleşmeler Truffle kütüphanesi yardımı ile blokzincire yazılmıştır.

Mutabakat verileri ve akıllı sözleşmeler yerel blokzincir ağı üzerinde tutulmaktadır. Yerel blokzincir ağı oluşturmak için Ganache programından faydalanılmıştır. Ganache programı üzerinden blok yapıları ve uygulama üzerinde gerçekleşen işlemler incelenebilmektedir. Şekil 5.1’ de uygulamanın görsel mimarisi gösterilmiştir.



Şekil 5.1. Uygulamanın görsel mimarisi.

Ganache yerel blokzincir ağı kurulduktan sonra ağ ve port ayarları yapılmaktadır. Ganache programı üzerinden yapılan bu ayarlar ile Ganache programının çalışacağı ağ ve port belirtilmiş olur. Geliştirilen web uygulamasının Ganache üzerinde kurulu Ethereum blokzincir ağı ile bağlantı ayarları ise **truffle-config.js** dosyası altında tanımlanmıştır. Burada tanımlanan bağlantı bilgilerinin Ganache üzerindeki bağlantı bilgileri ile aynı olması gerekmektedir. Ganache çalışma ortamı ayarları Şekil 5.2' de, web uygulamasının Ganache blokzincir ağı bağlantı ayarları ise Şekil 5.3' de gösterilmiştir.



Şekil 5.2. Ganache çalışma ortamı ayarları

```
module.exports = {
    // See <http://truffleframework.com/docs/advanced/configuration>
    // to customize your Truffle configuration!
    networks: {
        ganache: {
            host: "localhost",
            port: 7545,
            network_id: "*", // Match any network id
            gas: 4700000
        },
```

Şekil 5.3. Web uygulamasının Ganache blokzincir ağı bağlantı ayarları

Web uygulaması proje konsol ekranından **npm run start:dev** komutu ile çalıştırıldıktan sonra **http://localhost:3000** adresinden ulaşılabilmektedir. Uygulama içerisinde kullanıcıların görüntüleyebilmesi için ETH' in TL karşısındaki kur bilgisi anlık olarak alınmaktadır. Kur bilgisi için BtcTurk API kullanılmıştır (BtcTurk, 2013).

5.2. Geliştirme Araçları

Tez kapsamında geliştirilen DApp uygulamasında kullanılan teknoloji ve platformların detayları aşağıdaki gibidir:

Truffle Suit

DApp uygulamalarının geliştirilebilmesi için EVM blokzincir ağının kurulmasının yanı sıra akıllı sözleşmelerin geliştirilmesi, derlenmesi ve blokzincire yüklenmesini sağlayan platformdur. Üç ana yapıdan oluşmaktadır. Geliştirme ortamı oluşturmak için Truffle, yerel blokzincir ağı oluşturmak için Ganache ve DApp uygulama geliştirmek için kitaplık koleksiyonları sağlayan Drizzle platformlarından oluşmaktadır. Tez kapsamında geliştirilen uygulamada Truffle kütüphanesi ve Ganache platformu kullanılmıştır.

Truffle, geliştiriciler için kendi internet sitesi üzerinden akıllı sözleşme içerikli hazır DApp projeleri yayınlamaktadır. Geliştiriciler, bu projeleri indirerek uygulama mimarisini inceleyebilir veya proje üzerinden kendi uygulamalarını geliştirebilmektedir. Proje komut satırı üzerinden **truffle unbox <proje-adı>** komutunu çalıştırarak proje dosyaları indirilebilir. Sıfırdan bir proje oluşturulmak istenirse **truffle init** komutu çalıştırılmalıdır.

Ganache

Kullanıcıların yerel ortamda kişisel blokzinciri oluşturmasına olanak sağlayan programdır. Kullanıcıların dağıtım maliyeti ve işlem gecikmesi olmadan yerel olarak yazdıkları akıllı sözleşmelerin hatalarını ayıklamaya ve test etmelerine olanak sağlamaktadır. İçeriğinde 10 adet finanse edilmiş ve kilidi açılmış test hesabı mevcuttur. Kullanıcılar bu hesapları kullanarak ETH transferi, akıllı sözleşmelerdeki gaz maliyetleri gibi durumları test edebilirler. Blokzincir ağı üzerinde gerçekleşen işlemler ve oluşan bloklar program üzerinden izlenebilmektedir. Masaüstü uygulaması ve komut satırı sürümü olarak iki sürümü mevcuttur (Ganache, 2017).

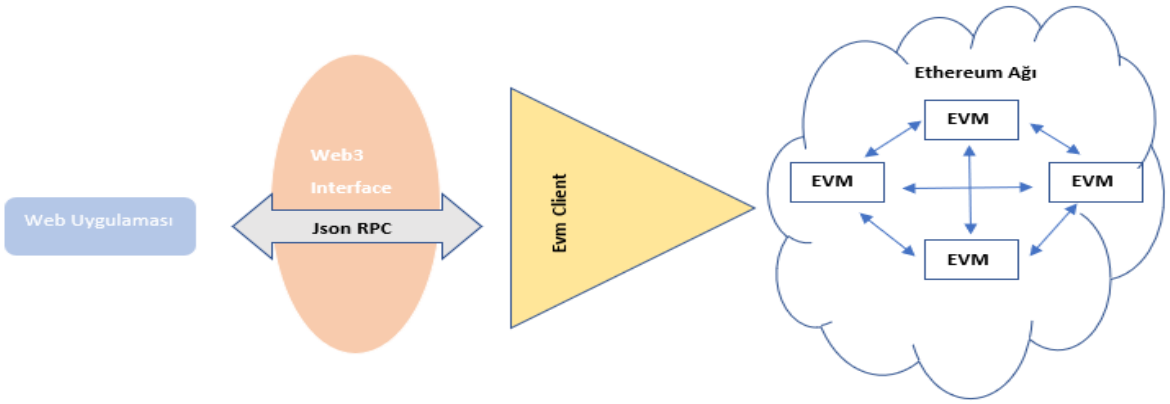
Metamask

Tarayıcı eklentisi olarak çalışan kripto para birimi cüzdanıdır. Chrome, Firefox ve Brave tarayıcılarında kullanılabilmektedir. Hesaplar arası ETH transferi yapılabilmektedir. Ayrıca DApp uygulaması üzerinde yapılacak herhangi bir işlemi imzalamak içinde kullanılabilmektedir. Ethereum Mainnet ve test netlerinin yanı sıra Ganache üzerinde oluşturulmuş yerel blokzincire bağlanılabilir. Yerel blokzincir üzerindeki test hesapları Metamask üzerinde kullanılabilmektedir. Bu sayede yerel ortamda geliştirilmiş DApp uygulamasının test işlemleri, yerel hesaplar ile tarayıcı üzerinden gerçekleştirilebilir.

Metamask, açık kaynaklı, güvenli ve esnek bir programdır. Kullanıcılar program arayüzünden Ether satın alabilmektedir. Kullanıcıların hesap anahtarları herhangi bir uzak sunucuda değil tarayıcı üzerinde saklanmaktadır (Metamask, 2016).

Web3.js

Web projelerinde Ethereum blokzinciri ile iletişime geçmeyi sağlayan Javascript kütüphanesidir. Bir hesaptan diğerine Ether gönderme, akıllı sözleşmelerden veri okuma ve yazma, akıllı sözleşmeler oluşturma gibi eylemlerin gerçekleştirilmesini sağlar. Http veya IPC bağlantısı ve JSON RPC protokolü kullanarak yerel veya uzak bir Ethereum düğümü ile etkileşime girmektedir (Web3, 2016). Şekil 5.4’ de Web3 kütüphanesinin çalışma diyagramı gösterilmiştir.



Şekil 5.4. Web3 kütüphanesi çalışma diyagramı.

Solidity

EVM üzerinde çalışan akıllı sözleşmeleri kodlamak için kullanılan nesneye dayalı programlama dilidir. Ethereum platformunda kullanılan en popüler dil olarak bilinmektedir. Yazılan akıllı sözleşmeler EVM üzerinde bytecode olarak derlenmektedir (Dannen, 2017).

5.3. Akıllı Sözleşme

Uygulama kapsamında geliştirilen akıllı sözleşme için Solidity programlama dili kullanılmıştır. Blokzincir üzerinde veri kaydetme ve okuma işlemleri akıllı sözleşme aracılığı ile yapılmaktadır. Akıllı sözleşmeler proje dizininde **/public/contracts** klasörü altında toplanmıştır. Klasör altında üç adet akıllı sözleşme mevcuttur. İlgili sözleşmelerin detayları Çizelge 5.1' de verilmiştir.

Çizelge 5.1. Uygulama kapsamında geliştirilen akıllı sözleşmeler.

Akıllı Sözleşme Adı	**Açıklama**
Migrations	Akıllı sözleşmelerin blokzincire geçişlerini gözlemlemek için yazılmıştır. Gelecekte içeriği değişmeyen sözleşmelerin blokzincire iki kez taşınmasını engeller.
Ownable	Akıllı sözleşme üzerinde yetkilendirme kontrolü sağlamaktadır. Sözleşmenin sadece sözleşme sahibi tarafından kontrol edilmesini sağlar.
Reconciliation	Müşteri ve mutabakatların oluşturulması ve çağrılması, para transfer süreçlerinin yönetildiği sözleşmedir.

Akıllı sözleşmelerin blokzincire yazılması için Truffle kütüphanesi kullanılmıştır. Proje komut satırı üzerinde **truffle migrate --compile-all --reset --network Ganache** komutu ile sözleşmeler Ganache üzerinde oluşturulmuş blokzincire yazılmaktadır. Blokzincirin mimarisi gereği, blokzincire gönderilmiş sözleşmeler üzerinde güncelleme işlemi yapılamamaktadır. Sözleşme üzerinde herhangi bir değişiklik yapma gereği duyulursa, sözleşmenin son hali truffle komutu ile blokzincire gönderilir. Blokzincire gönderilen son sözleşme aktif kullanıma alınır. Eski sözleşmeler blokzincir üzerinde kalmaya devam etmektedir. Uygulama kapsamında geliştirilen, mutabakat süreçlerinin yönetildiği "Reconciliation" akıllı sözleşmesinin fonksiyon imzaları ve parametreleri Şekil 5.5' de gösterilmiştir.

```
1   pragma solidity >0.4.99 <0.6.0;
2
3   import "./Ownable.sol";
4
5   contract Reconciliation is Ownable {
6       struct Mutabakat {
7           uint id;
8           address payable seller;
9           address buyer;
10          string name;
11          string aciklama;
12          uint256 tutar;
13          uint yil;
14          uint donem;
15          string durum;
16          string dosya;
17      }
18       struct Musteri {
19          uint ms_id;
20          string ms_name;
21          string ms_wallet;
22
23      }
24
25      mapping(uint => Mutabakat) public mutabakats;
26      mapping(uint => Musteri) public musteris;
27      uint mutabakatCounter;
28      uint MusteriCounter;
29
30      event LogMusteriEkle {};
35
36      event LogMutabakatOnay {};
46
47      function kill() public onlyOwner {}
50      function MusteriEkle(string memory _ms_name, string memory _ms_wallet) public {}
61      function MutabakatOlustur(string memory _name, string memory _aciklama, uint256 _tutar,uint _yil,uint _donem, string memory _
77      function MutabakatOnay(uint _id,uint kabultipi,string memory dosyaadi, string memory ret_aciklamasi) public payable {}
105     function getNumberOfmutabakats() public view returns (uint) {}
108     function MusterileriGetir() public view returns (uint[]memory) {}
129     function MutabakatlariGetir() public view returns (uint[]memory) {}
150 }
151
```

Şekil 5.5. Akıllı sözleşme kodu

Akıllı sözleşme 2 adet yapı (struct) ve 7 adet fonksiyondan oluşmaktadır. Solidity versiyonu **solidity>0.4.99<0.6.0** şekilde ayarlanmıştır. Bu gösterim şekli ile iki versiyon arasındaki çıkacak tüm güncel versiyonlarda çalışacağı belirtilmiştir. Sözleşme fonksiyon parametrelerinin alabileceği değer aralıkları Solidity tarafından belirlenmiştir (Solidity Types, 2016). Geliştirilen akıllı sözleşme içerisindeki fonksiyonların işlevleri Çizelge 5.2' de açıklanmıştır.

Çizelge 5.2. Akıllı sözleşme içerisindeki fonksiyonların işlevleri.

Fonksiyon Adı	**Açıklama**
kill()	Akıllı sözleşmenin blokzincir üzerinden kaldırılmasını sağlar. Sadece sözleşme sahibi tarafından yapılabilir.
MusteriEkle (string memory _ms_name, string memory _ms_wallet)	Müşterileri blokzincire ekler. Müşterinin adı ve cüzdan adresi olarak iki farklı parametreye sahiptir.
MutabakatOlustur (string memory _name, string memory _aciklama, uint256 _tutar, uint _yil, uint _donem, string memory _durum, bool _alacak)	Müşteri ile yapılan mutabakatın blokzincire yazılmasını sağlar. Mutabakatın adı, açıklaması, mutabakat tutarı, yıl, dönem, mutabakatın durumu ve alacak durumu parametrelerine sahiptir.
MutabakatOnay (uint _id, uint kabultipi, string memory dosyaadi, string memory ret_aciklamasi)	Mutabakatın onaylanıp blokzincire yazılmasını sağlar. Mutabakat id, ret veya onay durumu için kabul tipi, ret durumu için dosya adı ve ret açıklaması parametrelerine sahiptir.
getNumberOfmutabakats()	Mutabakatların sayısını döndürür.
MusterileriGetir()	Tüm müşterileri döndürür.
MutabakatlariGetir()	Tüm mutabakatları döndürür.

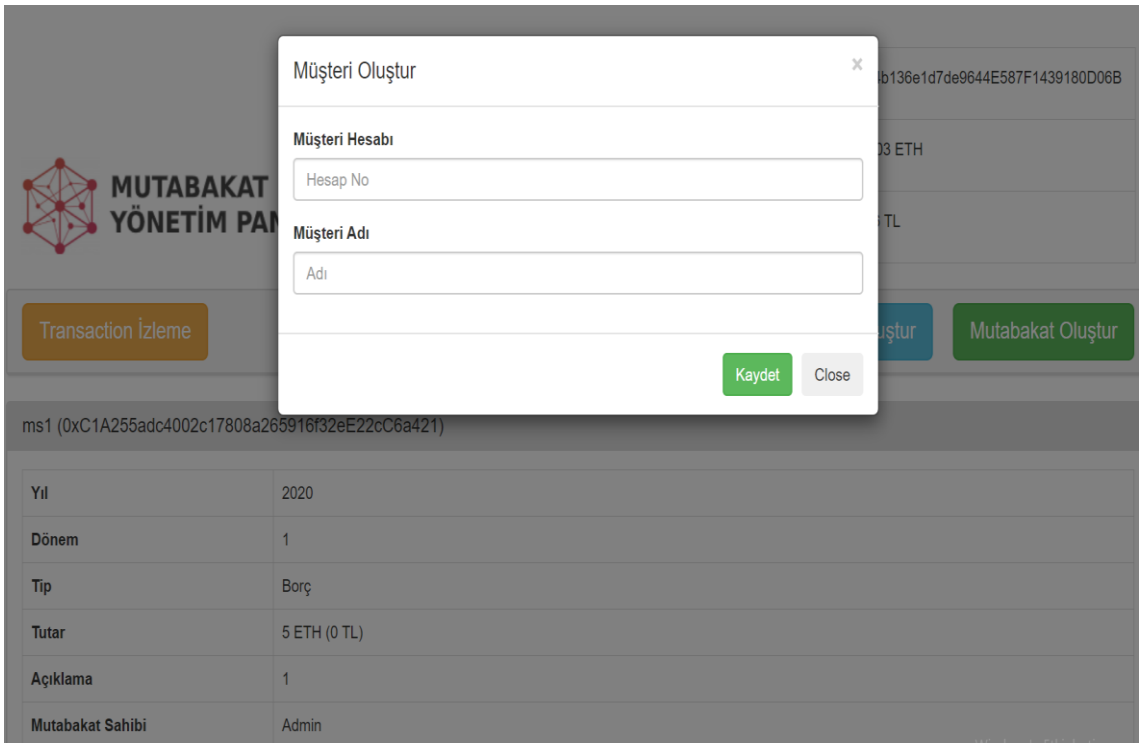
Akıllı sözleşme uygulama ile blokzincir arasında köprü görevi görmektedir. Blokzincir üzerinde müşterilerin eklenmesi, mutabakatların oluşturulması, onaylama işlemleri, müşterilerin ve mutabakatların çağrılması burada yapılmaktadır.

5.4. Uygulamanın Kullanımı

Uygulama içerisinde **Yönetici** ve **Müşteri** adında iki farklı kullanıcı tipi mevcuttur. Uygulama, yönetici tarafından müşterilerin sisteme eklenmesi, yönetici tarafından müşteriler üzerinde mutabakatların oluşturulması ve müşteriler tarafından mutabakatların cevaplanması adımlarından oluşmaktadır.

Müşteri ekleme

Mutabakat yapılacak müşteriler admin kullanıcısı tarafından eklenebilmektedir. Yönetici kullanıcısı sisteme giriş yaptıktan sonra anasayfada bulunan ve sadece yönetici kullanıcısı tarafından görüntülenen **Müşteri Oluştur** butonu ile yapılabilmektedir. Açılan arayüzde müşterinin açık anahtarı ve müşterinin adı alanları doldurularak kayıt işlemi yapılmaktadır. Eklenen müşteriler blokzincir üzerinde tutulmaktadır. Şekil 5.6' de müşteri ekleme arayüzü gösterilmiştir.



Şekil 5.6. Müşteri ekleme arayüzü.

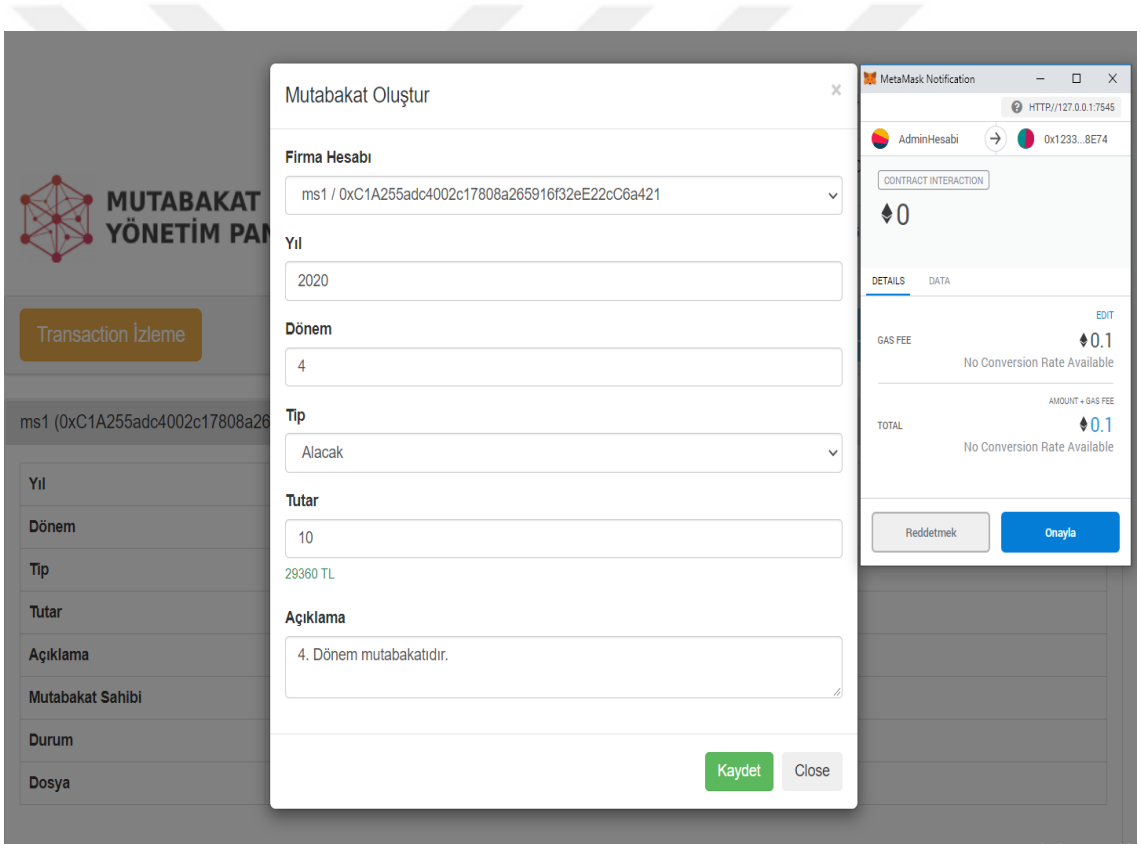
Mutabakat oluşturma

Mutabakat oluşturma işlemi sadece yönetici kullanıcısı tarafından yapılabilmektedir. Mutabakatlar müşteri bazında oluşturulmaktadır. İlk defa oluşturulan mutabakatın durumu **Beklemede** olarak güncellenir. Ana sayfada bulunan **Mutabakat Oluştur** butonu ile açılan arayüzde aşağıdaki adımlar takip edilerek müşteri bazında mutabakat oluşturulabilir.

- Mutabakat yapılacak müşterinin seçilmesi.
- Mutabakat yapılacak yılın doldurulması.
- Mutabakat yapılacak dönemin doldurulması.

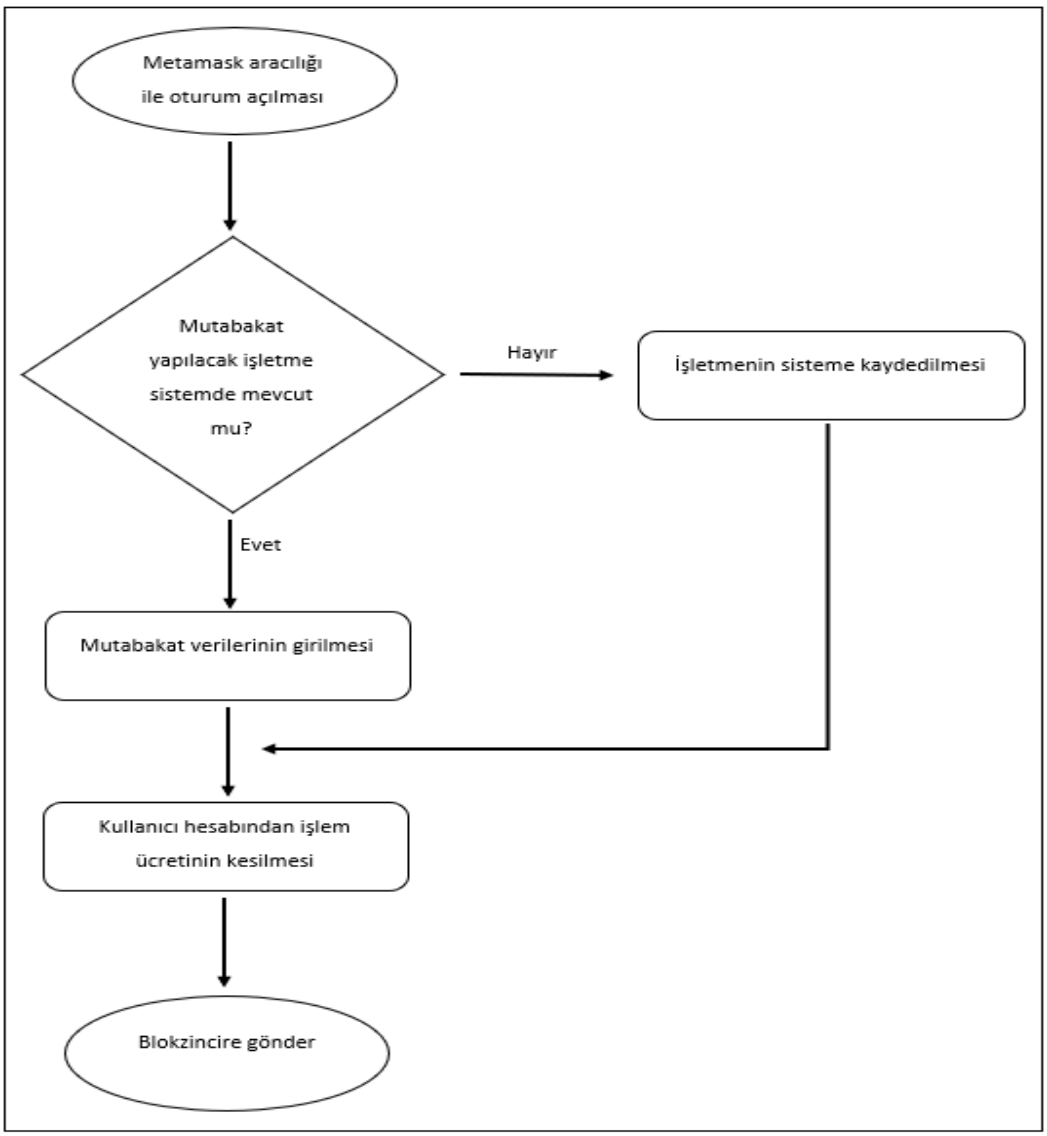
- Mutabakat ücretinin doldurulması (Ether kripto para birimi olarak doldurulması).
- Mutabakat ile alakalı açıklamanın doldurulması.

Yukarıdaki adımlar tamamlandıktan sonra Kaydet butonuna tıklanmaktadır. Kaydet butonu tıklandıktan sonra Metamask arayüzü açılmaktadır. Bu arayüzde ilgili işlemin gas fiyatı ve verinin hexadecimal şekilde gösterimi mevcuttur. Mutabakat oluşturma işleminin tamamlanabilmesi için Metamask arayüzünde onaylama işlemi yapılmalıdır. Onaylama sonrası kullanıcı hesabından Metamask arayüzünde **Gas Fee** alanında belirtilen işlem ücreti kadar ether düşecektir. Şekil 5.7' de mutabakat oluşturma arayüzü ve sonrasında açılan Metamask arayüzü gösterilmiştir.



Şekil 5.7. Mutabakat oluşturma arayüzü ve kayıt işlemi sonrası açılan Metamask arayüzü.

Müşterilerin blokzincire eklenmesi ve eklenen müşteriler üzerinde mutabakatların oluşturulması süreçleri ile alakalı akış şeması Şekil 5.8' de verilmiştir.



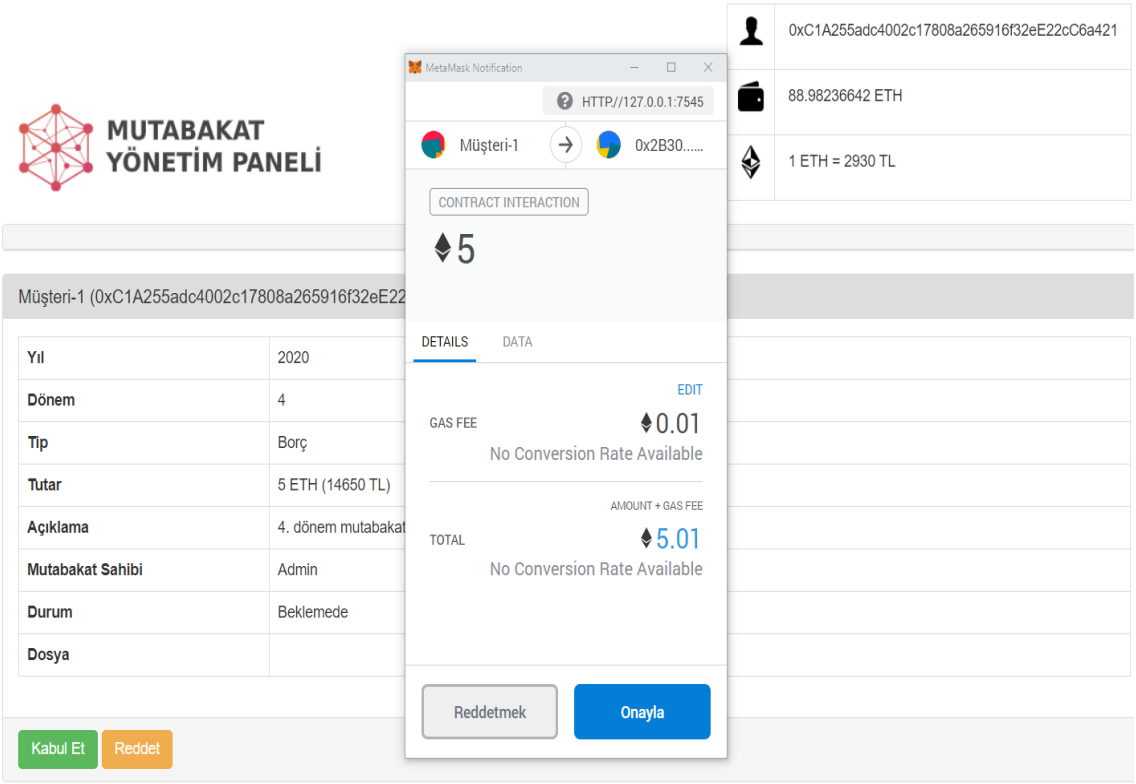
Şekil 5.8. Müşteri ekleme ve mutabakat oluşturma süreci akış şeması.

Mutabakat cevaplama

Müşteriler arayüze Metamask yardımı ile giriş yaptıktan sonra ana sayfada sadece kendilerine tanımlanmış mutabakatları görüntüleyebilmektedir. Mutabakat tipi alacak veya borç olma durumuna göre iki farklı cevaplama senaryosu vardır.

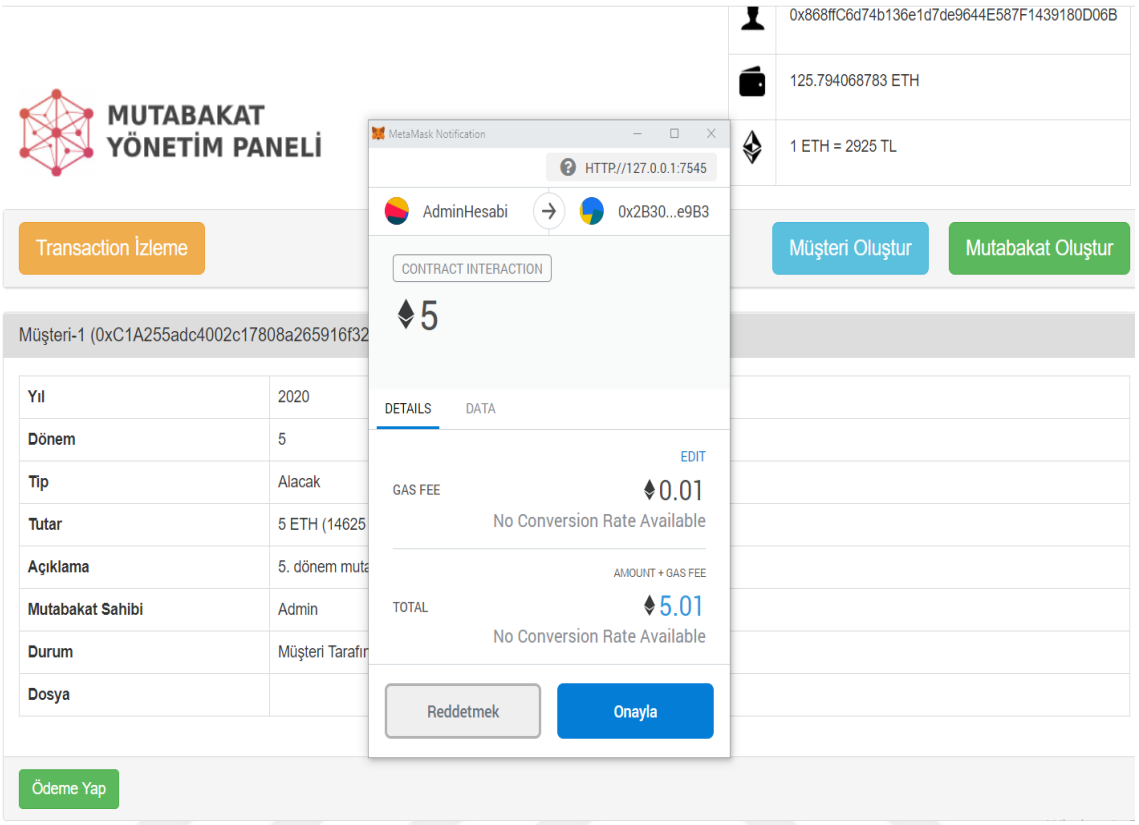
Mutabakat tipi **Borç** olan ve durumu **Beklemede** olan mutabakatlarda **Kabul Et** ve **Reddet** butonları aktif olarak görünmektedir. Müşteri borçlu olduğu tutar üzerinde mutabık ise **Kabul Et** butonunu tıklamalıdır. Bu işlem sonrası Şekil 5.9' de gösterilen Metamask arayüzü açılacaktır. Arayüzde gösterilen **Gas Fee** miktarı yapılan işlemin ücretidir. **Total** olarak gösterilen miktar ise işlemin ücreti ve mutabakat ücretinin toplamıdır. Müşteri ilgili işlemi onayladığı

takdirde total alanında yazan ether miktarı kadar hesabından düşecektir. Ayrıca mutabakat durumu **Kabul Edildi** olarak güncellenmektedir.



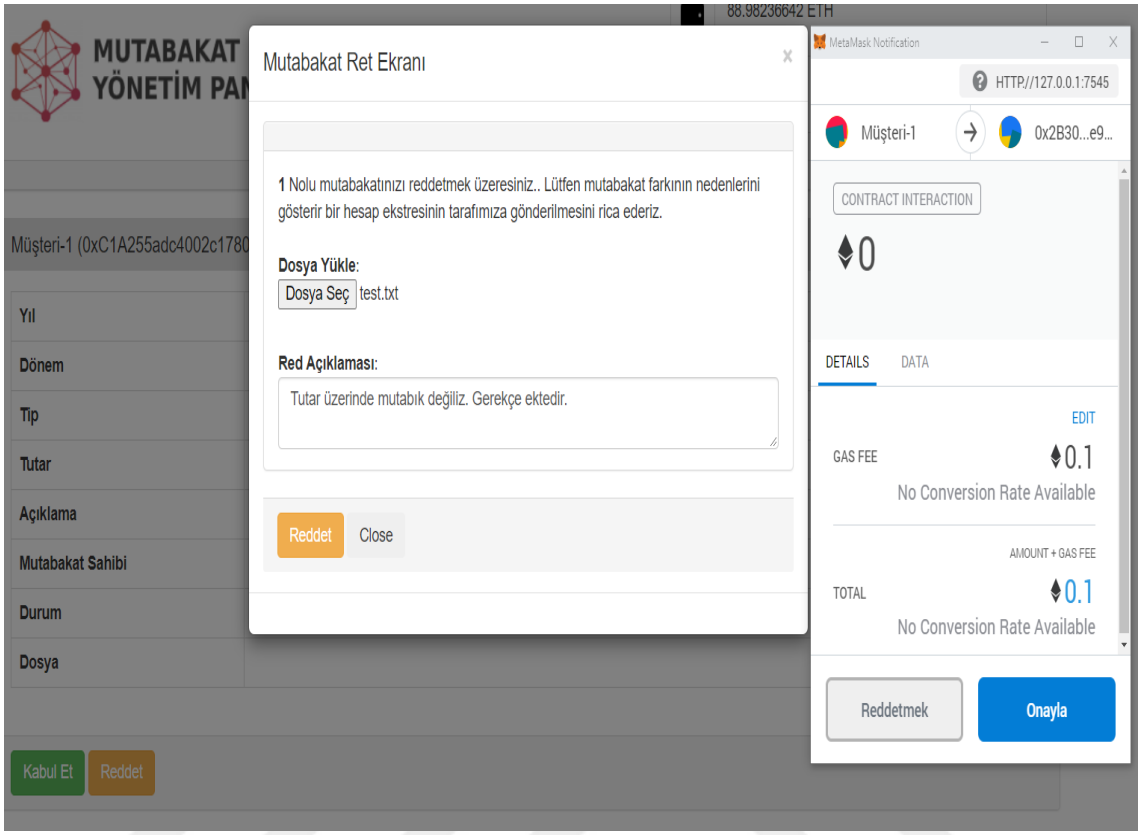
Şekil 5.9. Mutabakat kabul ekranı ve sonrası açılan Metamask arayüzü.

Mutabakat tipi **Alacak** olan ve durumu **Beklemede** olan mutabakatlarda aynı şekilde **Kabul Et** ve **Reddet** butonları aktif olarak görünmektedir. Müşteri, mutabakat sahibinden alacağı tutar üzerinde mutabık ise **Kabul Et** butonunu tıklamalıdır. Bu işlem sonrası Şekil 5.9' da gösterilen Metamask arayüzü açılacaktır. Müşteri ilgili işlemi onayladığı takdirde durumu alacaklı olduğu için hesabından sadece işlem ücreti kadar ether düşecektir. Onay sonrası mutabakat durumu "**Müşteri Tarafından Kabul Edildi. Admin Para Transferi Bekleniyor.**" şeklinde güncellenmektedir. Mutabakat sahibi ekranında durumu bu şekilde olan mutabakatlarda Şekil 5.10' de gösterildiği gibi "**Ödeme Yap**" butonu aktif olmaktadır. Ödeme yap butonu tıklandıktan sonra müşterinin alacağı tutar mutabakat sahibi hesabından müşteriye aktarılır.



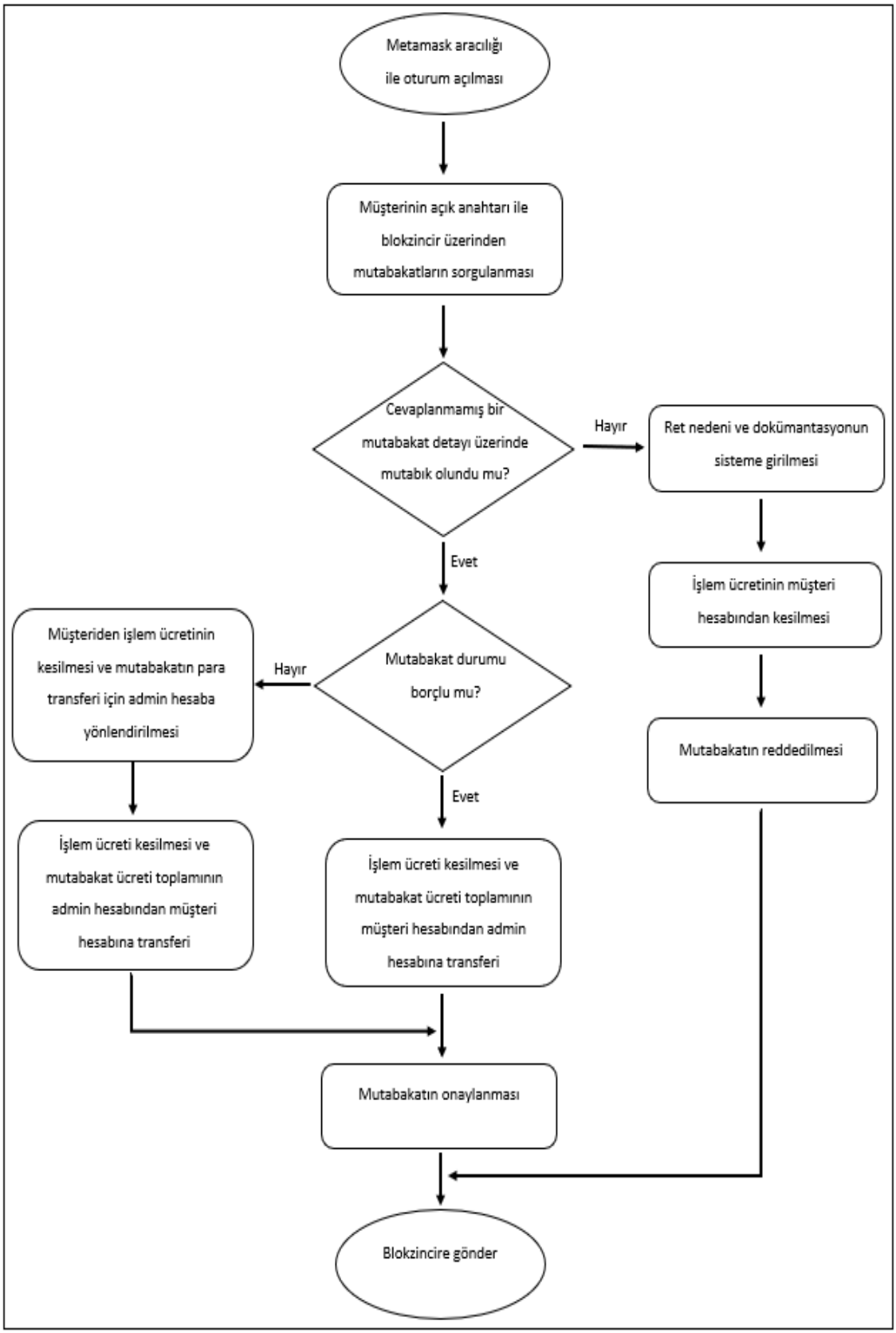
Şekil 5.10. Mutabakat sahibi tarafından ödeme yapılacak mutabakatların görüntülenmesi ve sonrasında açılan Metamask arayüzü.

Müşteri mutabakat üzerinde mutabık değil ise **Reddet** butonunu tıklamalıdır. Açılacak arayüzde ret açıklaması ve hesap ekstrelerini yükleyebileceği alanlar açılmaktadır. İlgili alanlar doldurulduktan sonra **Reddet** butonu tıklanmalıdır. Bu işlem sonrası Şekil 5.11' de gösterilen Metamask arayüzü açılacaktır. Müşteri ilgili işlemi onayladığı takdirde hesabından sadece işlem ücreti kadar ether düşecektir. Ayrıca mutabakat durumu **Reddedildi** olarak güncellenmektedir.



Şekil 5.11. Mutabakat ret ekranı ve sonrasında açılan Metamask arayüzü.

Yukarıda açıklanan mutabakatların cevaplanma süreçleri, akış şeması olarak Şekil 5.10' da gösterilmiştir.



Şekil 5.12. Mutabakat cevaplanma süreci akış şeması.

5.5. Olası Senaryoların Gerçekleştirilmesi

Bu bölümde Şekil 5.12' da gösterilen mutabakat cevaplanma süreci akış şemasındaki olası senaryoların, geliştirilen program üzerinde gerçekleştirilmesi sonucu ortaya çıkan sonuçlara yer verilmiştir. Olası senaryolar sisteme eklenen Müşteri-1 ve Admin kullanıcısı ile mutabakat tutarı 10 ETH üzerinden test edilmiştir. Olası senaryolar başlıklar halinde verilmiştir. Test ve admin kullanıcılarının bilgileri Çizelge 5.3' de gösterilmiştir.

Çizelge 5.3. Test kullanıcı bilgileri.

Kullanıcı	**Cüzdan Adresi**
Admin	0xC1A255adc4002c17808a265916f32eE22cC6a421
Müşteri-1	0x868ffC6d74b136e1d7de9644E587F1439180D06B

Durumu borçlu olan mutabakatın müşteri tarafından onaylanması

Mutabakat öncesi yönetici ve müşteri hesabında 100 Ether bulunmaktadır. Borçlu olan müşterinin mutabakat öncesi ve mutabakat onayı sonrası mutabakat durumu, hesaplardaki bulunan Ether miktarları ve işlem ücretleri aşağıdaki gibidir. Onay sonrası müşteri hesabından işlem ücreti ve mutabakat tutarının düştüğü görülmüştür. Yönetici hesabında ise mutabakat tutarı kadar bir artış gözlemlenmiştir. Mutabakat durumu "**Kabul Edildi**" olarak güncellenmiştir.

Çizelge 5.4. Durumu borçlu olan mutabakatın onaylanma sonuçları.

	Kullanıcı Tipi	**Mutabakatı Durumu**	**Ether Miktarı**	**İşlem Ücreti**
Mutabakat öncesi	Yönetici	Beklemede	100.00 ETH	-
	Müşteri	Beklemede	100.00 ETH	
Mutabakat onayı sonrası	Yönetici	Kabul Edildi	110.00 ETH	0.001445 ETH
	Müşteri	Kabul Edildi	89.9985 ETH	

Durumu alacaklı olan mutabakatın müşteri tarafından onaylanması

Durumu alacaklı olan mutabakatların bir önceki durumdan farklı olarak müşteri onayı sonrası yönetici hesabına tekrar onaya gelmesidir. Bu durum bir önceki bölümde detaylı olarak anlatılmıştır. Senaryo gerçekleştirilmeden önce yönetici ve müşteri hesaplarında 100 ETH

mevcuttur. İlk olarak müşteri onayı sonrası elde edilen sonuçlar Çizelge 5.5' te gösterilmiştir. Müşteri onayı sonrası müşteri hesabından sadece işlem ücreti kesildiği görülmüştür. Mutabakat durumu ise "**Müşteri Tarafından Kabul Edildi. Admin Para Transferi Bekleniyor.**" şeklinde güncellenmiştir.

Çizelge 5.5. Durumu alacaklı olan mutabakatın müşteri tarafından onaylanma sonuçları.

	Kullanıcı Tipi	**Mutabakatı Durumu**	**Ether Miktarı**	**İşlem Ücreti**
Mutabakat öncesi	Yönetici	Beklemede	100.00 ETH	-
	Müşteri	Beklemede	100.00 ETH	
Mutabakat onayı sonrası	Yönetici	Müşteri Tarafından Kabul Edildi. Admin Para Transferi Bekleniyor.	100.00 ETH	0.002418 ETH
	Müşteri	Müşteri Tarafından Kabul Edildi. Admin Para Transferi Bekleniyor.	99.9975 ETH	

Müşteri onayı sonrası mutabakat yöneticinin onayına düşmüştür. Yönetici onayı sonrası yönetici hesabından işlem ücreti ve mutabakat tutarının düştüğü, müşteri hesabında ise mutabakat tutarı kadar artış gözlemlenmiştir. Mutabakat durumu ise "**Kabul Edildi**" olarak güncellenmiştir. Yönetici onayı sonrası elde edilen sonuçlar Çizelge 5.6' da gösterilmiştir.

Çizelge 5.6. Durumu alacaklı olan mutabakatın yönetici tarafından onaylanma sonuçları.

	Kullanıcı Tipi	**Mutabakatı Durumu**	**Ether Miktarı**	**İşlem Ücreti**
Mutabakat öncesi	Yönetici	Müşteri Tarafından Kabul Edildi. Admin Para Transferi Bekleniyor.	100.00 ETH	-
	Müşteri	Müşteri Tarafından Kabul Edildi. Admin Para Transferi Bekleniyor.	99.9975 ETH	
Mutabakat onayı sonrası	Yönetici	Kabul Edildi	99.9993 ETH	0.000676 ETH
	Müşteri	Kabul Edildi	109.9975 ETH	

Mutabakatın reddedilmesi

Cevaplanmamış mutabakatın müşteri tarafından reddedilmesi sonrası müşteri ve yönetici hesaplarında gerçekleşen durumlar Çizelge 5.7' de gösterilmiştir. Mutabakat reddedilmeden önce müşteri ve yönetici hesaplarında 100 ETH bulunmaktadır. Ret işlemi sonrası mutabakat durumu "**Reddedildi**" olarak güncellenmiş, müşteri hesabından sadece işlem ücreti kesintisi olmuştur.

Çizelge 5.7. Cevaplanmamış mutabakatın müşteri tarafından reddedilme sonuçları.

	Kullanıcı Tipi	**Mutabakatı Durumu**	**Ether Miktarı**	**İşlem Ücreti**
Mutabakat öncesi	Yönetici	Beklemede	100.00 ETH	-
	Müşteri	Beklemede	100.00 ETH	
Mutabakat onayı sonrası	Yönetici	Reddedildi	100.00 ETH	0.002464 ETH
	Müşteri	Reddedildi	99.9975 ETH	

5.6. Önerilen Sistemin Değerlendirilmesi

Yapılan uygulama testleri sonucu sistem, işlem ücreti kesintileri, verilerin blokzincire kaydedilmesi ve güvenlik analizi başlıkları altında değerlendirilmiştir.

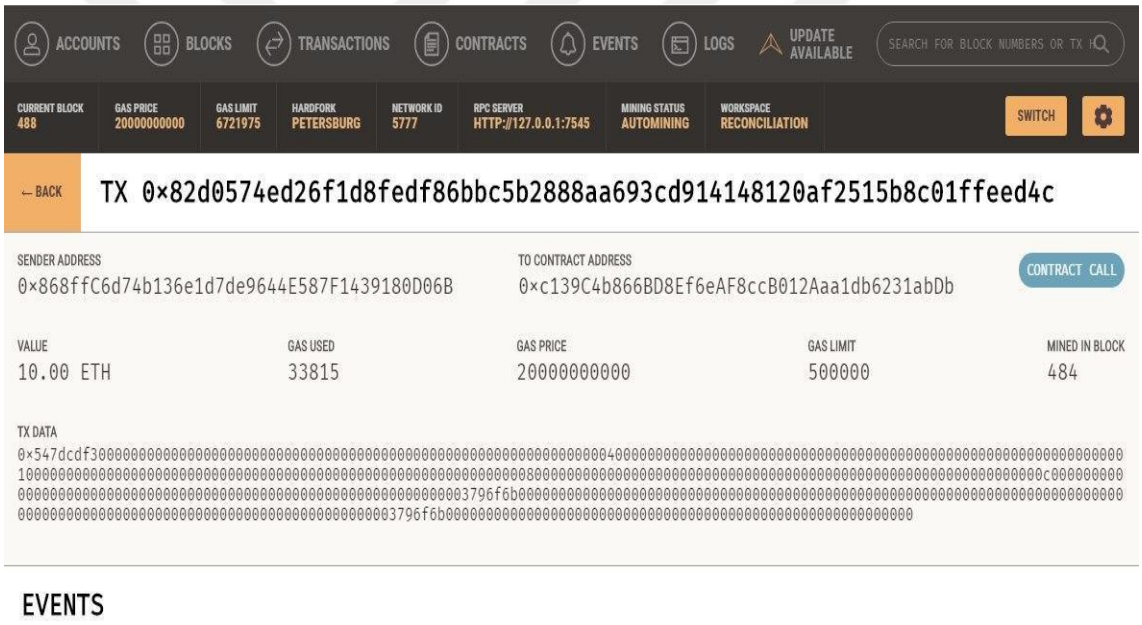
İşlem ücreti kesintileri

Olası senaryoların gerçekleştirilmesi aşamasında müşterilerden kesilen işlem ücretlerinin, test müşteri sayısı arttırılarak, müşteri bazında değişimi gözlemlenmiştir. Ganache üzerinde bulunan 10 test kullanıcısı sisteme müşteri olarak eklenmiştir. Tüm müşterilere dönem, yıl ve diğer parametreleri aynı olacak şekilde, 10 ETH üzerinden mutabakatlar yapılmıştır. Yapılan testler sonucu tüm müşterilerin hesaplarından mutabakat tutarı dışında 0.002418 ETH işlem ücreti kesintisi gerçekleştiği görülmüştür. Müşterilerin mutabakat tutarı, yıl, dönem ve diğer parametreleri farklı olacak şekilde mutabakatlar yapılmıştır. Mutabakatların onaylanması sonrası tüm müşterilerde 0.002418 ETH işlem ücreti kesintisi gerçekleştiği ve ücretin değişmediği görülmüştür. Ayrıca mutabakat durumu alacaklı olan ve yönetici hesabından müşteri hesabına gerçekleşen para transferlerinde gerçekleşen 0.000676 ETH işlem ücreti kesintisi, mutabakat tutarı ve parametrelerindeki değişiklik sonrası işlem ücretinin aynı kaldığı görülmüştür. Mutabakat durumu alacak ve borçlu olmasına göre mutabakat parametre ve tutarlarının farklı olması işlem ücretini etkilemediği görülmüştür. Durumu borç olan mutabakatların, durumu alacak olan mutabakatlara göre işlem ücreti kesintilerinin farklı olmasının nedeni akıllı sözleşme

içerisinde bulunan onay fonksiyonundaki, mutabakat durumuna göre çalışan farklı kod bloklarıdır. Akıllı sözleşmeler, yürüttüğü kod bloğunda bulunan operasyonel kodlar değerince gas ücreti hesaplamakta ve işlem ücreti kesintisi yapmaktadır.

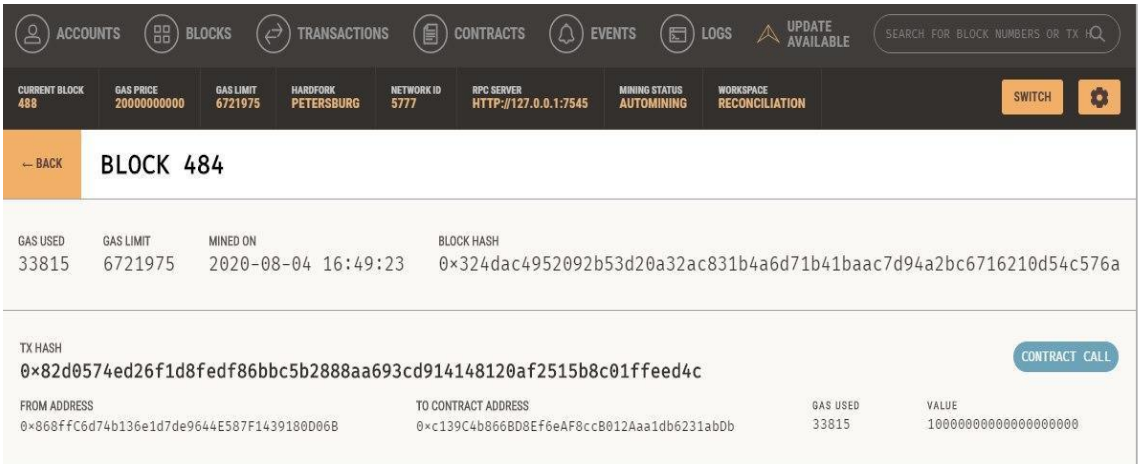
Verilerin blokzincire kaydedilmesi

Müşterilerin eklenmesi, mutabakat onay süreçleri ve sonrasında gerçekleşen para transfer işlem verileri yerel blokzincir ağına kaydedilmektedir. Yapılan işlemler sonrası Ganache programı üzerinden oluşan işlem ve bloklar takip edilmiştir. Yönetici ve müşteri arasında 10 ETH üzerinden yapılan mutabakatın başarılı bir şekilde sonuçlanması sonrası işlem 0x82d0574ed26f1d8fedf86bbc5b2888aa693cd914148120af2515b8c01ffeed4c özet değeri ile blokzincire yazılmıştır. İşlemin Ganache programı üzerinde görüntüsü Şekil 5.13' de verilmiştir.



Şekil 5.13. Mutabakat sonrası oluşan işlemin Ganache üzerinde gösterimi.

Blokzincir üzerinde işlemler bir araya gelerek blokları oluşturmaktadır. Yapılan işlem 484 numaralı blok içerisine kaydedilmiştir. Bloğun Ganache programı üzerindeki görüntüsü Şekil 5.14' de verilmiştir.



Şekil 5.14. Mutabakat sonrası oluşan bloğun Ganache üzerinde gösterimi

Blokzincire kaydedilen işlemlerin şifreli olarak tutulduğu görülmüştür. Blok yapıları incelendiğinde, blokların birbirleri arasında kriptolojik ilişki olduğu tespit edilmiştir. Blokların bir önceki bloğun özet değerini **parentHash** değeri altında tuttuğu görülmüştür. Uygulama üzerinde yapılan son işlem sonrası oluşan bloğun yapısı ve parentHash değeri Şekil 5.15' de gösterilmiştir.


```
<     "id": 925,
<     "jsonrpc": "2.0",
<     "result": {
<       "number": "0x25",
<       "hash": "0x776fe1074c1588753cb7a4730114edd6c86175ac3bd6d5a80e3c234a46afb60e",
<       "parentHash": "0xd1e8a21266de4901d20393810b2c555e087696d22d14ceaedf9bf9ddacfe9fe2",
<       "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
<       "nonce": "0x0000000000000000",
<       "sha3Uncles": "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
<       "logsBloom": "0x0000000000000000000000000000000000000000000000000000000020000000000000000000000000
<       "transactionsRoot": "0x0b08542e674679498e8f1c8498f4d824da2d9019e6fd5e95981a93a8942df091",
<       "stateRoot": "0xd66e432e7842aa9e61e96432c78caa5c75913dfda4406f8808a736440b27fb55",
<       "receiptsRoot": "0x47974afb6bd1d487ffcb426da13435bc8a34d5251ba0235aed82a397e40b258a",
<       "miner": "0x0000000000000000000000000000000000000000",
<       "difficulty": "0x0",
<       "totalDifficulty": "0x0",
<       "extraData": "0x",
<       "size": "0x3e8",
<       "gasLimit": "0x6691b7",
<       "gasUsed": "0x8417",
<       "timestamp": "0x5f50b964",
<       "transactions": [
<         "0x61e300162deb65618fcd4c0505b2d2b894cc39d65b75ee63834457fca256acd1"
<       ],
<       "uncles": []
<     }
```


Şekil 5.15. Mutabakat sonrası oluşan bloğun yapısı ve parentHash değeri.

Güvenlik analizi

Geliştirilen uygulama kapsamında veriler lokal blokzincir ağında tutulmaktadır. Veriler şifreli olarak, birbirleri ile kriptolojik ilişkisi olan bloklarda saklanmıştır. Blokların ve içerisindeki verilerin değiştirilemez olması sağlanmıştır. Herhangi bir bloktaki işlem değiştirildiğinde ilgili bloğun özet değeri değişir ve ilişkili kripto zinciri kırılır. Bu durum diğer düğümler tarafından dikkate alınmamaktadır. Kötü niyetli bir kişinin veriyi değiştirebilmesi için uygulamanın çalışacağı Ethereum veya konsorsiyum ağındaki düğümlerini %51 oranında ele geçirmesi gerekmektedir. Ana Ethereum ağında 9038 düğüm bulunmaktadır (Ethernodes, 2020). Farklı ülke ve ağlarda bulunan bu düğümlerin yarıdan fazlasının ele geçirilmesi mümkün görülmemektedir.

Verilerin düğümler tarafından saklanması birçok avantajı beraberinde getirmiştir. Günümüz merkezi sistemler göz önüne alındığında veriler sunucularda veya üçüncü taraflardan ücret karşılığı veri barındırma hizmeti alınarak saklanmaktadır. Ayrıca şirketler tarafından veri yedekleme çalışmaları yapılmaktadır. Günümüz sistemlerinde veri yedekleri belirli zaman aralıklarında alınmaktadır. Olası bir kötü senaryoda sistemin yedek zamanına geri alınması farklı kullanıcılar tarafından yapılan işlemlerin de kaybolmasına neden olmaktadır. Veri kaybına yol açabilecek bu durum blokzincir düğüm mimarisi ile ortadan kalkmıştır. Ağdaki tüm düğümler aynı veri defterlerini tutmaktadır. Ağdaki düğümlerin herhangi birinin çalışamaz duruma gelmesi veya defterin silinmesi durumunda, güncel defterin ağdaki diğer düğümlerde bulunması nedeniyle veri bütünlüğü bozulmamaktadır. Bu sebeple şirketlerin herhangi bir yedekleme çalışması yapması gerekmemektedir.

Müşterilerin sisteme girişleri elektronik cüzdan ile sağlanmıştır. Tarayıcı eklentisi olarak çalışan Metamask programı sayesinde müşteriler, sadece özel anahtarları ile ilgili programda hesaplarını açabilmektedir. Metamask programında ilgili hesap aktif seçili ise, mutabakat uygulamasına herhangi bir kullanıcı adı ve şifre girişi olmadan otomatik bağlanabilmektedir. Bu sayede müşterilerin uygulamaya girişi için kullanıcı adı ve şifre gibi kötü niyetli kullanıcılar tarafından ele geçirilmesi veya kullanıcı tarafından unutulabilmesi daha kolay olan bilgiler kullanılmadan, güvenilirliği daha yüksek bir yol izlenmiştir.

6. SONUÇ VE ÖNERİLER

6.1. Sonuçlar

Bu tez çalışmasında ilk olarak işletmeler arası yapılan cari mutabakatların ve sonrasında gerçekleşen ödeme işlemelerinde yaşanan eksiklik ve problemler detaylı olarak incelenmiştir. Yapılan araştırmalar sonucunda blokzincir teknolojisinin hali hazırda kullanılan yöntem ve uygulamalarda yaşanan problemleri ve eksiklikleri gidermeye yönelik fayda sağlayabileceği görülmüştür. Bu kapsamda blokzincir teknolojisi ile alakalı kapsamlı bir araştırma yapılmıştır. Teknolojinin teknik bileşenleri ve özellikleri incelenmiş, çalışma mantığı irdelenmiştir.

Çalışma kapsamında blokzincir tabanlı dağıtık uygulama geliştirmeye olanak sağlayan DApp hakkında literatür araştırması yapılmıştır. Araştırmalar sonucunda uygulama geliştirme platformu olarak Ethereum tercih edilmiştir. Ethereum çalışma mantığı ve teknik detayları araştırılmış, uygulama geliştirebilmek için gereken araçlar hakkında bilgiler verilmiştir.

Tez kapsamında blokzincir teknolojisinin cari mutabakat ve ödeme işlemlerindeki etkisini incelemek üzere Ethereum tabanlı, local DApp uygulaması geliştirilmiştir. Geliştirilen uygulama ile müşteri veya tedarikçilerin cüzdan bilgisi ile sisteme başarılı bir şekilde kaydı gerçekleştirilmiştir. Kaydı gerçekleşen işletmeler ile mutabakat işlemlerindeki tüm olası senaryolar başarılı bir şekilde gerçekleştirilmiştir. Gerçekleştirilen olası senaryoların test sonuçları bölüm 5.5’ de verilmiştir.

Bu süreçte ortaya çıkan veriler blokzincir üzerinde bloklarda saklanmıştır. Blok mimarisi ile geriye dönük takibi edilebilir, şeffaf ve değiştirilemez şekilde verilerin saklanması sağlanmıştır. Bu şekilde, günümüz mutabakat yöntemlerinden olan E-Mutabakat uygulamalarındaki tek merkeze dayalı güven problemini çözüme kavuşturduğu görülmüştür. Ağa dâhil olan her kullanıcı zinciri okuyabileceğinden, tüm süreçler denetlenebilir olmaktadır. Ayrıca yıllık program ödemeleri ve veri yedekleme gibi güvenlik için oluşabilecek ek maliyetleri ortadan kaldırdığı görülmüştür. Diğer taraftan geleneksel yöntemlerden telefon, fax veya E-Posta yolu ile yapılan mutabakatlardaki yaşanan veri erişim ve raporlama problemlerini, zaman kayıplarını, kargo, arşivleme masraflarını ve ek personel ihtiyacını ortadan kaldırmıştır.

Uygulama kapsamında müşteri veya tedarikçilerin sisteme kişisel elektronik cüzdan hesapları (Metamask) ile giriş yapmaları sağlanmıştır. Mutabakat onayı sonrası mutabakat ücretinin işletme hesapları arası başarılı bir şekilde transfer edildiği gözlemlenmiştir. Transfer işlemlerinin herhangi gün veya saat fark etmeksizin gerçekleştirilebildiği, düşük ücret kesintisi ile üçüncü şahıslara gerek kalmadan yapılabildiği görülmüştür. Bu sayede transfer işlemleri için

bankaların aracı olarak kullanılmasıyla ortaya çıkan yüksek komisyon ücretinin yanı sıra doğrulama ve kayıt prosedürleri gibi işlemlerin uzamasına sebep olan problemlerin ortadan kalktığı görülmüştür.

Blokzincir teknolojisinin temellerinden biri de bloklardaki verilerin değiştirilemez olmasıdır. Bu nedenle olası mutabakat senaryolarının gerçekleştirilmesi aşamasında, hatalı veri girişi yapılan mutabakatlar üzerinde güncelleme işlemi yapılamamıştır. Buna çözüm olarak hatalı mutabakatlar düzeltilip, açıklama alanına değişiklik olduğunu belirten uyarı yazılmış ve sisteme yeni bir mutabakat olarak tekrar eklenmiştir. Ayrıca hatalı olarak mutabakatın onaylaması veya reddedilmesi sonrası gerçekleşen işlem ücreti ve mutabakat tutarı kesintilerinin geri iadesi veya işlemin iptali yapılamamaktadır.

Bu tez çalışmasında, işletmeler arası yapılan cari mutabakat işlemlerinin günümüz yöntemleri dışında, blokzincir üzerinde yapılabilmesini sağlayan ve günümüz yöntemlerinde yaşanan problemleri ortadan kaldıran yeni bir sistem geliştirilmiştir. Bunun yanı sıra cari mutabakat sonrası gerçekleştirilen para transferlerinin günümüz yöntemleri dışında, herhangi bir bankaya gerek duymadan kripto para ile uygulama üzerinden yapılabilmesini sağlayan bir yaklaşım getirilmiştir. Ayrıca bu çalışmanın, gelişimi devam eden blokzincir teknolojisinin sektörel bazda uygunluğu ile alakalı geliştirilecek uygulamalara örnek olması hedeflenmiştir.

6.2. Öneriler

Uygulama içeriğinde ödeme ve işlem ücretleri Ethereum para birimi olan Ether ile yapılmaktadır. Ethereum tabanlı dağıtık uygulamaların çalışma prensipleri göz önüne alındığında uygulamayı kullanacak işletmelerin elektronik cüzdan ve kripto para ile alakalı alt yapı çalışması yapmaları gerekmektedir. Ayrıca uygulama içeriğinde yapılan her işlem sonrası gerçekleşen kripto para kesintisinin olmaması için PoW mutabakat algoritması kullanmayan bir uzlaşma yapısı geliştirilebilir. Uygulama platformu olarak açık blokzincir ağları kullanmak yerine konsorsiyum ağı kurulabilir. Kurulacak yapı ve konsorsiyum ağı ile birlikte uygulamayı kullanacak işletmelerin tek bir ağ üzerinde toplanması ve her bir işletmenin birer düğüm olarak kullanılması ile işlemlerin onaylanması ve saklanması sağlanabilir. Bu şekilde uygulamada karşılaşılan kripto para transferi dışındaki işlemlerin herhangi bir işlem ücreti kesintisi olmadan ücretsiz gerçekleşmesi sağlanabilir. Ayrıca konsorsiyum ağı ile birlikte yüksek güvenlikli, kolay denetlenebilir ve erişilebilir bir yapı kurulabilir.

KAYNAKLAR DİZİNİ

Andoni, M., Robu, V., Flynn, D., Abram, S., Geach, D., Jenkins, D., McCallum, P., Peacock, A. (2019), Blockchain technology in the energy sector: A systematic review of challenges and opportunities, Renew. *Sustain Energy Reviews*, cilt 100, s 143–174.

Antonopoulos, A., Wood, G. (2018). *Mastering Ethereum: Building Smart Contracts and Dapps*. O'Reilly Media, Inc.

Beşkirli, A., Özdemir, D., Beşkirli, M. (2019). Şifreleme Yöntemleri ve RSA Algoritması Üzerine Bir İnceleme. *Avrupa Bilim ve Teknoloji Dergisi*, 284 - 291. doi.org/10.31590/ejosat.638090.

Blockgeeks, Become A Bitcoin Developer: Basic 101. Erişim Adresi: https://blockgeeks.com/guides/bitcoin-developer/.

BtcTurk. (2013). Erişim: 11.06.2020. Erişim Adresi: https://docs.btcturk.com/#introduction.

Buterin V. (2014), A Next-Generation Smart Contract and Decentralized Application Platform. Erişim Adresi: https://github.com/ethereum/wiki/wiki/White-Paper.

Chang, V., Baudier, P., Zhang, H., Xu, Q., Zhang, J., Arami, M. (2020). How Blockchain can impact financial services – The overview, challenges and recommendations from expert interviewees. *Technological Forecasting and Social Change*, cilt 158, 120166. Erişim Adresi: https://doi.org/10.1016/j.techfore.2020.120166.

Chen, H., Pendleton, M., Njilla, L., Xu, S. (2019), A survey on ethereum systems security: Vulnerabilities attacks and defenses. Erişim Adresi: https://arxiv.org/abs/1908.04507.

Christidis, K., Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things, *IEEE Access*, cilt 4, s 2292–2303.

Cong, L. W., He, Z. (2019). Blockchain disruption and smart contracts, *Review of Financial Studies*, 32:1754–97.

Dai, H. N., Zheng, Z., Zhang Y. (2019), Blockchain for internet of things: A survey, *IEEE Internet of Things Journal*, s 1–19.

DappRadar. (2018). Erişim: 10.06.2020. Erişim Adresi: https://dappradar.com/rankings.

Dannen C., 2017, *Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners*. Berkely, CA, USA: Apress.

Djrtwo. (2017). Evm Opcode Gas Costs. Erişim: 12.06.2020. Erişim Adresi: https://github.com/djrtwo/evm-opcode-gas-costs.

EOSIO. (2017). Erişim: 10.06.2020. Erişim Adresi: https://eos.io/.

Ergin, Ö. (2018, 19 Eylül). Ethereum Gas Nedir? Adım Adım Ethereum Gas Rehberi [Blog Yazısı]. Erişim Adresi: https://www.coinkolik.com/ethereum-gas-nedir-adim-adim-ethereum-gas-rehberi/.

ETHEREUM. (2013). Erişim: 10.06.2020. Erişim Adresi: https://ethereum.org/en/.

KAYNAKLAR DİZİNİ (devam)

Ethereum Mainnet Statistics. Erişim: 22.04.2020. Erişim Adresi: https://ethernodes.org/.

Ethereum Remix. Erişim: 22.04.2020. Erişim Adresi: https://remix.ethereum.org/.

Ethernodes. (2020). Erişim:03.09.2020. Erişim Adresi: https://ethernodes.org/.

Fernández-Caramés, T. M., Fraga-Lamas, P. (2018). A review on the use of blockchain for the Internet of Things, *IEEE Access*, cilt 6, s 32979-33001.

Ganache. (2017). Erişim: 10.06.2020. Erişim Adresi: https://www.trufflesuite.com/ganache.

Gerdan, G. (2019). Blokzincir Teknolojisiyle Gıda Güvenliği ve Yumurta Sektörü için Örnek Bir Uygulama, Yüksek Lisans Tezi, Marmara Üniversitesi, İstanbul.

Godambe, S., Samudrala, R. (2017). Simplifying Inter-entity Reconciliation using Blockchain Technology. Tata Consultancy Services. Corpus ID: 203689091.

Guo, Y., Liang, C. (2016). Blockchain application and outlook in the banking industry. *Financial Innovation*. 2 (1), 24.

Iansiti, M., Lakhani, K. R. (2017). The Truth About The Truth About Blockchain (Cilt 95). *Harvard Business Review*.

İşler, B., Takaoğlu, M., Küçükali, U. F. (2019). Blokzinciri ve kripto Paraların İnsanlığa Etkileri, *Yeni Medya Elektronik Dergi*, 3 (2), 71-83.

Johnson, M., Jones, M., Shervey, M., Dudley, J. T., Zimmerman, N. (2019). Building a secure biomedical data sharing decentralized app (DApp): Tutorial, *J. Med. Internet Res.*, cilt 21, no 10.

Karahan, Ç. ve Tüfekci, A. (2019). Blokzincir Teknolojisi ve Kamu Kurumlarınca Verilen Hizmetlerde Blokzincirin Kullanım Durumu, *Verimlilik Dergisi*, (4):157-193.

Karayılan, H. (2019). Blokzincir ve Finansal Teknoloji Çözümleri İçin Uygulamaları, Yüksek Lisans Tezi, İstanbul Üniversitesi, İstanbul.

Keleş, B., Ova, G. (2020). Gıda Tedarik Zinciri Yönteminde Bilgi Teknolojileri Kullanımı, *Adnan Menderes Üniversitesi Ziraat Fakültesi Dergisi*, 17 (1), 137-143.

Kırbaş, İ. (2018). Blokzinciri teknolojisi ve yakın gelecekteki uygulama alanları. *Mehmet Akif Ersoy Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 9 (1), 75-82.

Lallai, G., Pinna, A., Marchesi, M., Tonelli, R. (2020), Software Engineering for DApp Smart Contracts managing workers Contracts. In Proceedings of the Distributed Ledger Technology (DLT'20).

Lin, I.-C., Liao, T.-C. (2017), A survey of blockchain security issues and challenges, *IJ Network Security*, 19 (5), s 653-659.

Metamask. (2016). Erişim: 10.06.2020. Erişim Adresi: https://metamask.io/.

KAYNAKLAR DİZİNİ (devam)

Murat, M. (2018). Blockchain ile Güvenli Elektronik Sağlık Sistemi, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi, İstanbul.

Nakamoto, S. (2008), Bitcoin: A Peer-to-Peer Electronic Cash System. Erişim Adresi: https://bitcoin.org/bitcoin.pdf.

NEO. (2014). Erişim: 10.06.2020. Erişim Adresi: https://neo.org/.

Nguyen, G. T., Kim, K. (2018). A survey about consensus algorithms used in Blockchain, *Journal of Information Processing Systems*, s 101-128.

Ranganthan, V. P., Dantu, R., Paul, A., Mears, P., Morozov, K. (2018), A decentralized marketplace application on the ethereum blockchain, 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), s 90-97.

Rinkeby: Ethereum Testnet. Erişim:15.04.2020. Erişim Adresi: https://www.rinkeby.io/#stats

Özcan, D. (2019). *Blokzincir Mimarisi ve Merkezi Olmayan Uygulamalar* (Birinci Baskı). Türkiye: Pusula Yayıncılık, 9-10.

Paul. (2019, 22 Mayıs). Ethereum Tutorial – A Deeper Look Into Ethereum [Blog Yazısı]. Erişim Adresi: https://www.edureka.co/blog/ethereum-tutorial-with-smart-contracts/.

Putri, M. C. I., Sukarno, P., Wardana, A. A. (2020), Two factor authentication framework based on ethereum blockchain with DApp as token generation system instead of thirdparty on web application, Register: *Jurnal Ilmiah Teknologi Sistem Informasi*, cilt 6, no. 2, s 74-85.

Rosic, A. (2018). What is Ethereum Gas? Step-By-Step Guide. Erişim: 18.04.2020. Erişim Adresi: https://blockgeeks.com/guides/ethereum-gas/.

Schultz, M. (2019, 27 Eylül). What Is Reconciliation In Accounting? [Blog Yazısı]. Erişim Adresi: https://www.blackline.com/blog/account-reconciliations/what-are-account-reconciliations/.

Schultz, M. (2020, 27 Eylül). Measuring The Real Cost Of Manual Accounting [Blog Yazısı]. Erişim Adresi: https://www.blackline.com/blog/finance-automation/measuring-real-cost-manual-accounting/.

Solidity Types. (2016). Erişim: 20.04.2020. Erişim Adresi: https://solidity.readthedocs.io/en/v0.5.13/types.html.

Tapscott, D., A. Tapscott. (2017). "How Blockchain Will Change Organizations." *MIT Sloan Management Review* 58 (2): 10–13.

Tanrıverdi, M., Uysal, M. ve Üstündağ, M. T. (2019). Blokzincir Teknolojisi Nedir? Ne Değildir? Alanyazın İncelemesi. *Bilişim Teknolojileri Dergisi*, 12(3), 203-217.

KAYNAKLAR DİZİNİ (devam)

The Problem with Manual Account Reconciliation Processes. (2017). Erişim Adresi: https://www.datavail.com/blog/the-problem-with-manual-account-reconciliation-processes/.

TRON. (2017). Erişim: 10.06.2020. Erişim Adresi: https://tron.network/.

Tucker, T., Cooper, S. (2009). Using Spreadsheets for Reconciliations – An Obsolete, Risky Practice, *Financial Executive*. Erişim Adresi: https://pages.blackline.com/rs/blacklinesystems/images/Financial-an-Obsolete-Risky-Practice.pdf.

TÜBİTAK. (2019). TÜBİTAK Web Sitesi: https://blokzincir.bilgem.tubitak.gov.tr/blok-zincir.html.

Türkiye Cumhuriyeti Cumhurbaşkanlığı ikinci 100 günlük çalışma planı. (2018) . Erişim Adresi: https://tccb.gov.tr/assets/dosya/2018-12-13-ikinci100gun.pdf.

Understanding Account Reconciliation. (2019). Erişim Adresi: https://www.venasolutions.com/blog/financial-close/understanding-account-reconciliation.

Usta, A., Doğantekin, S. (2018). *Blockchain 101 V2*. İstanbul: Kapital Medya Hizmetleri A.Ş.

Ünsal, E. ve Kocaoğlu, Ö. (2018). Blok Zinciri Teknolojisi: Kullanım Alanları, Açık Noktaları ve Gelecek Beklentileri, *Avrupa Bilim ve Teknoloji Dergisi*, 13, 54-64. Erişim Adresi: https://doi.org/10.31590/ejosat.423676.

Wahab, A., Memood, W. (2018). Survey of Consensus Protocols. Erişim Adresi: https://arxiv.org/abs/1810.03357.

Web3. (2016). Erişim: 10.06.2020. Erişim Adresi: https://web3js.readthedocs.io/en/v1.2.11/

Wüst, K., Gervais, A. (2018, June). Do you need a Blockchain? In 2018 Crypto Valley Conference on Blockchain Technology (CVCBT) (s 45-54). IEEE.

Yaga, D., Mell, P., Roby, N., Scarfone, K. (2018). Blockchain technology overview. *NISTIR*, 8202. (s 8).

Yano, M., Dai, C., Masuda, K., Kishimoto, Y. (2020). *Blockchain and Crypto Currency.* Springer Open. Erişim Adresi: https://link.springer.com/book/10.1007%2F978-981-15-3376-1.

Zhang, R., Xue, R., Liu, L. (2019). Security and privacy on blockchain. *ACM Comput. Surv.*

Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends, Proceedings of the 2017 IEEE BigData Congress, Honolulu, Hawaii, USA, s 557–564.

Zhou, Q., Huang, H., Zheng, Z., Bian, J. (2020), Solutions to scalability of blockchain: A survey, *IEEE Access*, cilt 8, s 16440-16455.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : Muhammet Ali Efendioğlu

Doğum tarihi ve yeri : 1992 – Bursa

E – mail : muhammetali.efendioglu@gmail.com

Eğitim

Derece	Eğitim	Mezuniyet Tarihi
Yüksek Lisans	Kütahya Dumlupınar Üniversitesi Bilgisayar Mühendisliği	2020
Lisans	Kütahya Dumlupınar Üniversitesi Bilgisayar Mühendisliği	2015
Lise	Bursa Anadolu Kız Lisesi	2010

İş Deneyimi

Yıl	Yer	Görev
2015-2016	Kodyazan Yazılım	Kurucu Ortak
2016-2018	Belsis Yazılım	Yazılım Uzmanı
2018-	Ermetal Şirketler Grubu	Yazılım Sorumlusu

Yabancı Dil

İngilizce