T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

BLOK ZİNCİR TEKNOLOJİSİNİ KULLANARAK YENİ BİR KRİPTO PARA OLUŞTURMA ESASLARI

Sezen GEÇEL ŞENER

DANIŞMAN
Dr. Özgür Can TURNA

Bilgisayar Mühendisliği Anabilim Dalı Adı

Tezli Yüksek Lisans

İSTANBUL-2022

Bu çalışma 23.05.2022 tarihinde aşağıdaki jüri tarafından ……., ……..nda ……. olarak kabul edilmiştir.

Tez Jürisi

Dr.Öğr.Üyesi Özgür Can TURNA
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Doç.Dr. Muhammed Ali AYDIN
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Doç.Dr. Şerif BAHTİYAR
İstanbul Teknik Üniversitesi
Bilgisayar ve Bilişim Fakültesi

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi-Cerrahpaşa'nın abonesi olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

ÖNSÖZ

Bu çalışmanın gerçekleştirilmesindeki destek ve katkılarından dolayı öncelikle danışman hocam Dr. Özgür Can TURNA'ya sonsuz teşekkürlerimi sunarım. Yüksek lisans eğitim sürecinin her aşamasında bilgi birikimlerini paylaşan ve emeklerini esirgemeyen tüm hocalarımıza da ayrıca teşekkür ederim.

23.05.2022 Sezen GEÇEL ŞENER

İÇİNDEKİLER

Sayfa No

ÖNSÖZ ..... **iv**
İÇİNDEKİLER ..... **v**
ŞEKİL LİSTESİ ..... **viii**
TABLO LİSTESİ ..... **ix**
SİMGE VE KISALTMA LİSTESİ ..... **x**
ÖZET ..... **xi**
SUMMARY ..... **xii**
1. GENEL KISIMLAR **1**
1.1. BLOK ZİNCİR 2
1.1.1. Tarihsel Gelişim 3
1.1.2. Blok Zincir Avantajları 4
1.1.3. Blok Zincir Zorlukları 5
1.1.4. Blok Zincir Uygulama Alanları 7
1.1.4.1. Finans ve Bankacılık *8*
1.1.4.2. İthalat ve İhracat *8*
1.1.4.3. Petrol ve Enerji *8*
1.1.4.4. Elektronik Mağazacılığı *9*
1.1.4.5. Elektronik Ortam ve Web Siteleri *9*
1.1.4.6. Emlak Ticareti *9*
1.1.4.7. Mülkiyet Takibi *9*
1.1.4.8. Gıda Tedarik Zinciri *9*
1.1.4.9. Sağlık *9*
1.1.5. Blok Zincir Türleri 10
1.1.5.1. İzinsiz (Permissionless) Blok Zincir *10*
1.1.5.2. İzin Verilen (Permissioned) Blok Zincir *10*
1.1.6. Blok Zincir Mimarisi 11
1.1.6.1. Kriptografik Hash Fonksiyonları *11*
1.1.6.2. Dijital İmza *13*
1.1.6.3. Simetrik Olmayan Açık Anahtar Kriptografi *14*

1.1.6.4. İşlemler *15*
1.1.6.5. Defter (Ledger) *17*
1.1.6.6. Bloklar *18*
1.1.7. Blok Zincir İşleyişi 20
1.1.7.1. Yumuşak Çatallanma (Soft Forking) *22*
1.1.7.2. Sert Çatallanma (Hard Forking) *23*
1.1.8. Mutabakat 23
1.1.8.1. İş İspatı (PoW-Proof of Work) Mutabakatı *25*
1.1.8.2. Hisse İspatı (PoS-Proof of Stake) Mutabakatı *27*
1.1.8.3. Delege Hisse İspatı (DPos - Delegated Proof of StakeDPos) Mutabakatı *28*
1.1.8.4. Pratik Bizans Hata Toleransı Mutabakatı *28*
1.1.8.5. Tendermint *28*
1.1.8.6. Round Robin Mutabakatı *29*
1.1.8.7. Otorite/Kimlik İspatı (Proof of Authority/Identity) Mutabakatı *29*
1.1.8.8. Zaman İspatı (Proof of Elapsed Time) Mutabakatı *29*
1.1.9. Akıllı Sözleşmeler (Smart Contracts) 30
1.2. KRİPTO PARA BİRİMLERİ 31
1.2.1. Kripto Para Birimlerinin Avantajları 33
1.2.2. Kripto Para Birimlerini Tehdit Eden Sorunlar 34
1.2.3. Kripto Para Birimlerinin Güvenlik Sorunları 34
1.3. BITCOIN 36
1.3.1. Bitcoin Ağ Yapısı ve İşleyişi 37
1.3.2. Cüzdanlar 39
1.3.3. İşlem ve Script 40
1.3.3.1. P2PKH (Pay-to-Pubkey-Hash) *41*
1.3.3.2. P2SH (Pay-to-Script-Hash) *42*
1.3.4. Bitcoin ağında karşılaşılan saldırılar 42
1.3.4.1. Finney Saldırısı *43*
1.3.4.2. Brute Force (Kaba Kuvvet) Saldırısı *43*
1.3.4.3. Sybill Saldırısı *43*
1.3.4.4. Bribery (Rüşvet) Saldırısı *43*
1.3.4.5. Feather (Köpük) Saldırısı *43*
1.3.4.6. Eclipse/Netsplit Saldırısı *43*

1.4. ETHEREUM 44
1.4.1. İşlemler 46
1.4.2. EVM (Ethereum Virtual Machine) 50
1.4.3. Akıllı Sözleşmeler (Smart Contracts) 51
1.5. RIPPLE 52
1.6. KARŞILAŞTIRMA 56
2. MALZEME VE YÖNTEM 59
2.1 MODEL İŞLEYİŞİ 60
2.2. MODEL BİLEŞENLERİ 62
2.2.1. Wallet (Cüzdan) 63
2.2.2. Message (Mesaj) 65
2.2.3. Network 67
2.2.4. Transaction (İşlem) 68
2.2.5. Mempool 70
2.2.6. Blocks 73
2.2.7. Blockchain 74
2.2.8. Merkletree 75
2.2.9. UTXO (Harcanmamış İşlem Çıktıları) 76
2.2.10. Peers (Eşler) 77
2.2.11. SQLdb - Veritabanı 78
2.2.12. App & Forms 78
3. BULGULAR 79
4. TARTIŞMA VE SONUÇ 83
KAYNAKLAR 85
EKLER 88
ÖZGEÇMİŞ 89

ŞEKİL LİSTESİ

Sayfa No

Şekil 1: Blok Zincir Uygulama Alanları……..8

Şekil 2: Açık Anahtar Kriptografi……..14

Şekil 3: İşlem Zinciri ……..…15

Şekil 4: Blok Zincir İşleyişi ..…….21

Şekil 5: P2PKH ve P2SH Örnek Şablonları ……42

Şekil 6: Ethereum Sanal Makinası (EVM) ..……..50

Şekil 7: Solidity Modify Örneği ..……..52

Şekil 8: Örnek Ripple Grafı ..…….55

Şekil 9: Model İşleyişi ……..60

Şekil 10: Bileşenler Arası İlişki Şeması ……62

Şekil 11: Özel ve Açık Anahtar Oluşturma Ekranı……...…65

Şekil 12: İşlem Örneği…….71

Şekil 13: İşlem Doğrulama Fonksiyonu ……..……72

Şekil 14: Yeni Blok Gösterimi……..……..73

Şekil 15: Merkle Ağacı ……..……..75

Şekil 16: Blok işlemleri Listeleme..……76

TABLO LİSTESİ

Sayfa No

Tablo 1: Blok Zincir Türleri ve Özellikleri……..10

Tablo 2: SHA-256 Örnek Girdi ve Çıktı Değerleri..……12

Tablo 3: Bitcoin Örnek İşlem Bazlı Defteri…….16

Tablo 4: Bitcoin Örnek İşlemleri……..17

Tablo 5: Blok Başlık Alanları ve Açıklamaları …..…….18

Tablo 6: Blok Zincir İşleyiş Adımları ..……20

Tablo 7: Karşılaştırmalı Mutabakat Yöntemleri ..…….25

Tablo 8. P2PKH Yürütme Adımları……..41

Tablo 9: Ethereum Blok Başlık Alanları ve Tanımları……..48

Tablo 10: Kripto Para Birimleri Karşılaştırma Tablosu ..…….57

Tablo 11: Bitcoin Blok Zincir Platformu Seçim Kriterleri ……..79

Tablo 12: Model Konfigürasyonu ……80

Tablo 13: Bileşen Özellikleri……81

SİMGE VE KISALTMA LİSTESİ

Simgeler **Açıklama**

ROTR : Sağa kaydırma fonksiyonu

Kısaltmalar **Açıklama**

BTC : Bitcoin
ETH : Ethereum
XRP : Ripple
MD : Message Digest
SHA : Secure Hash Algorithm
EVM : Ethereum Virtual Machine

ÖZET

YÜKSEK LİSANS TEZİ

BLOK ZİNCİR TEKNOLOJİSİNİ KULLANARAK YENİ BİR KRİPTO PARA OLUŞTURMA ESASLARI

Sezen GEÇEL ŞENER

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı Adı

Danışman : Dr. Özgür Can TURNA

Finans sektörünün altyapısını ve işleyişini değiştirme gücüne sahip olduğu düşünülen kripto paralar, önümüzdeki yıllar içerisinde ulusal ve uluslararası para politikalarını derinden etkileyecek ve şekillendirecektir. Geleceğin para piyasasına hakim olmak isteyen hükümetler ve şirketler, kripto para teknolojisi üzerine çalışmalarını yoğunlaştırarak finans dünyasına yön verecek önemli bir finansal araç olan kripto para sistemini hayata geçirmek zorunda kalacaklardır. Bu çalışmanın amacı, tüm hakları ülkemize ait olacak, tasarımcılarımızın ve mühendislerimizin emekleri ile yoğurulan, milli teknolojilerle üretilen ve dünyaya adımızı duyuracak bir kripto paranın ilk temellerini atabilmektir.

Yeni bir kripto para tasarımını oluşturmak bu çalışmanın kilit noktasını oluşturmuş, sistemin temel yapı taşları tasarlanmış, blok zincir ve Phyton yazılım platformu kullanılarak mimari hayata geçirilmiştir.

23.05.2022, 101 sayfa.

Anahtar kelimeler: blok zincir, kripto para, kriptografi, akıllı sözleşmeler, Phyton, Bitcoin, Ethereum, Ripple.

SUMMARY

M.Sc. THESIS

FUNDAMENTALS OF CREATING A NEW CRYPTOCURRENCY USING BLOCKCHAIN TECHNOLOGY

Sezen GECEL SENER

Istanbul University-Cerrahpasa

Institute of Graduate Studies

Computer Engineering

Supervisor : Dr. Ozgur Can TURNA

Cryptocurrencies, which are thought to have the power to change the infrastructure and functioning of the financial sector, will deeply affect and shape national and international monetary policies in the coming years. Governments and companies that want to dominate the money market of the future will have to concentrate their studies on crypto money technology and implement the crypto money system, which is an important financial tool that will guide the financial world. The aim of this study is to lay the first foundations of a crypto currency, all rights of which will belong to our country, which is created by the efforts of our designers and engineers, produced with national technologies and which will make our name known to the world.

Creating a new crypto currency design was the key point of this study, the basic building blocks of the system were designed, the architecture was implemented by using the blockchain and Phyton software platform.

23.05.2022, 101 pages.

Keywords: blockchain, crypto currency, cryptography, smart contracts, Phyton, Bitcoin, Ethereum, Ripple.

1. GENEL KISIMLAR

Son yılların İnternetten sonra en önemli buluşlarından biri tartışmasız blok zincir teknolojisidir. Kripto paralar, ürün tedarik zincir yönetimi, gıda güvenliği, akıllı cihazlar, Nesnelerin İnterneti gibi uygulamalarda çok çeşitli kullanım örnekleri bulunan blok zincir teknolojisinin gelecekte önemli bir potansiyele sahip olduğu açıkça görülmektedir.

Blok zincir aslında İnternet gibi temel bir teknolojidir, kripto para gibi uygulamalar ise, e-posta veya video akışı gibi İnternet tabanlı uygulamalara eşdeğerdir. Kavramsal olarak blok zincir, katılan üyeler arasında paylaşılan işlemlerin kayıtlarını içeren dağıtılmış bir veritabanıdır. Her işlem, katılımcıların çoğunluğunun mutabakatıyla onaylanır ve hileli işlemlerin toplu onayı geçememesi sağlanır. Bir kayıt oluşturulduktan ve blok zincir tarafından kabul edildikten sonra asla değiştirilemez veya yok edilemez.

Merkezi olmayan dağıtılmış bir veritabanı teknolojisi olan blok zincir, dünya genelinde geniş bir ilgi görmeye başlayan oldukça yeni bir teknolojidir. Tüm dünya genelinde finans, emlak, sağlık hizmetleri ve diğer sektörlerdeki şirketlerin blok zincir hakkında bilgi edinmeye ve bu teknolojiyi kullanmaya çok hevesli oldukları görülmektedir.

Blok zincir, birçok kripto para biriminin temel teknolojisidir. Aslında, blok zincir kripto paraların yaratılacağı ve gerçekleştirileceği bir platform olarak tasarlanmıştır. Kripto para birimleri arasında en bilinen para birimi şüphesiz Bitcoin'dir. Kripto para dünyasında en geniş işlem hacmine sahip olan Bitcoin aynı zamanda ilk kripto para birimi olarak bilinir. Piyasaya ilk çıkışı 2008 yılı sonlarında olmuştur. Gerçek kimliği hala gizemini koruyan Satoshi Nakamoto tarafından "Bitcoin: Kişiden kişiye elektronik nakit sistemi" adlı makale ile teorik altyapısı ana hatlarıyla tanımlanan Bitcoin, 2009 yılında kullanılmaya başlanmıştır.

Kripto para birimleri merkezi bir otoritenin yer almadığı merkeziyetsiz sistemlerdir. Bir başka deyişle, kripto para sistemi ağ üzerinde birden fazla cihazın birbiriyle iletişime geçebildiği dağıtık bir sistemdir. Bu sistemde gerçekleştirilen işlemlerin kontrolünü sağlamak, para basımı olarak düşünülebilecek yeni paraları sisteme dahil etmek ve sahtekarlığı önlemek için kriptografi kullanılır. Sistem içerisinde gerçekleştirilen ödeme işlemlerinin onaylanması temel prensiptir. Ağı oluşturan düğümler tarafından onaylanan işlemler belli kurallar çerçevesinde bir araya gelerek blokları oluşturur ve blok zincir yapısına kaydedilirler. Blok zincir tüm ödeme işlemlerinin kaydedildiği bir çeşit muhasebe sistemi olarak düşünülebilir. Blok zincir yapısını sıradan bir muhasebe sisteminden ayıran özelliği herkese açık ve erişilebilir olmasıdır.

Bitcoin ve alternatif kripto para birimlerinin finansal piyasalar üzerinde etkisi her geçen gün artarak devam etmektedir. Geleneksel ve ulusal para birimlerine nazaran üstün tarafları kripto paralara olan ilgiyi arttıran en önemli unsurdur. Kesintisiz 7 gün 24 saat işlem yapma imkanı, düşük komisyon ücretleri ve hızlı işlem yapabilme üstünlük sağlayan bu özelliklerin başında gelir. Kripto paraların sunduğu bu avantajlar, para piyasası işleyiş kurallarının değişmesine de sebep olmaktadır.

Bu çalışmada, kripto para yapısının iyi anlaşılabilmesi için öncelikle blok zincir teknolojisi, tarihsel gelişimi, çalışma esasları, potansiyel uygulama alanları incelenmiştir. Yüksek güvenlik sağlamak için kullanılan kriptografi, hash fonksiyonlar ve özellikle SHA-256

fonksiyonu detayları ile açıklanmıştır. Merkezi bir otoriteye ihtiyaç duyulmadan dağıtılmış bağımsız bir ağda tüm katılımcıların fikir birliğine varmasını sağlayan mutabakat mekanizmaları ve protokolleri ile ilgili literatür çalışması yapılmıştır.

Kripto para oluşturma prensipleri, kripto paralar ile gerçekleştirilen işlemleri kaydetme yöntemleri, kripto paraların çalışması için gerekli olan cüzdan, mempool ve akıllı kontrat teorisi literatürde incelenerek çalışmaya eklenmiştir. Günümüz ekonomisinde bu alanda lider konumundaki üç kripto para birimi Bitcoin, Ethereum ve Ripple, sistem özellikleri ve işleyişleri detaylandırılarak karşılaştırılmıştır.

Yeni bir kripto para tasarımı Phyton yazılım platformu kullanılarak oluşturulmuştur. Tasarım ve yazılım detayları, modül ekran görüntüleri, yazılım parçaları, geliştirme sürecinde karşılaşılan zorluklar ve çözüm yolları çalışma kapsamında paylaşılmıştır.

1.1. BLOK ZİNCİR

Blok zincir, merkezi bir otoritenin olmadığı, şifrelenmiş dijital verinin sistem katılımcıları arasında transfer edildiği ve bu transfer işlemlerinin o sistem içinde paylaşılan herkese açık bir deftere kaydedildiği dağıtılmış bir sistemdir. Dağıtılmış bir sistemdir çünkü işlemlerin kaydedildiği işlem defteri kopyası sistem katılımcılarının bilgisayarlarında mevcuttur. Birçok kripto para biriminin ilki olan Bitcoin ağının piyasaya sürülmesiyle 2009 yılında yaygın olarak tanınmaya başlamıştır. Blok zincir teknolojisi sayesinde, Bitcoin sonrasında Ethereum gibi daha birçok kripto para birimi sisteminin geliştirilmesi sağlanmıştır. Bu nedenle, blok zincir teknolojisi genellikle Bitcoin'e veya kripto para birimi çözümlerine bağlı olarak görülür. Ancak, teknolojinin aslında yaklaşık otuz senelik bir geçmişi vardır ve çok çeşitli uygulamalar için blok zincir uygulamaları mevcuttur ve çeşitli sektörler için de araştırmalar yapılmaktadır.

Blok zincir özünde, kriptografi ile şifrelenmiş işlemlerin blok adı verilen gruplar halinde kaydedildiği tüm katılımcıların paylaştığı ve doğruladığı bir sayısal defterdir (David, 2018). Bu sayısal defter, sistem içerisinde katılımcılar tarafından gerçekleştirilen tüm işlerin tutulduğu bir veritabanına benzetilebilir. Bloklar, defterin her bir sayfasını temsil ederken blokları temsil eden defter sayfaları ise katılımcılara ait şifrelenmiş işlemlerden oluşur. Her blok, doğrulama ve bir fikir birliği kararından geçtikten sonra, bir öncekine kriptografik hash değeri ile bağlanır. Yeni bloklar eklendikçe eski blokların değiştirilmesi daha zor hale gelir, bu kurcalamaya karşı direnç oluşturur. Yeni bloklar, ağdaki yedek defter kopyalarında da güncellenir. Katılımcı düğümlerin sahip olduğu defter kopyaları arasında oluşacak bir tutarsızlık blok zincir ağı için tanımlanan kurallar çerçevesinde otomatik olarak çözülür.

Blok zincir teknolojisinin getirdiği en önemli fark, banka ve aracı kurumlar gibi üçüncü taraflara ihtiyacı kaldırmasıdır. Yapılan işlemler için çoklu defter yerine yalnızca bir ortak defter olması, teknoloji yatırım gereksiniminin daha düşük olmasını sağlar. Tüm katılımcılar istedikleri zaman ağa erişebilirler. Bunun sonucu olarak, çok daha sıkı bir kontrol uygulanmış olur ve bu sayede verinin manipülasyonu daha kolay engelenebilir. Katılımcıların kendileri sistemin çalışmasını garanti ederler, bunun için özel bir altyapı gerekmez. İşlem akışını

sağlamak için dışarıdan bir müdahale ihtiyacı yoktur, katılımcılar ve ağ geleneksel yapılarda üçüncü tarafların sağladığı ihtiyaç duyulan tüm hizmetleri yerine getirebilir.

1.1.1. Tarihsel Gelişim

Blok zincir teknolojisi temel fikri, 1980'lerin sonunda ve 1990'ların başında ortaya çıktı. David Lee Chaum, 1982 yılında yayınladığı "Blind Signatures for Untraceable Payments" makalesinde, dijital nakit ve kör imzalar kavramlarını açıklamıştır. Bu araştırmada, kullanıcıların diğer taraflarca izlenemeyecek şekilde dijital para harcamasına ve transfer etmesine olanak tanıyan bir yöntem önerilmiştir.

1989'da Lamport, Paxos fikir birliği protokolünü geliştirdi ve 1998 yılında yayınlanan makalede, bir bilgisayar ağında bilgisayarların ve ağın kendisinin güvenilmez olabileceği, bir sonuç üzerinde anlaşmaya varmak için bir fikir birliği modelinin gerekliliği ve detayları açıklanmıştır. Bu modelin benzeri bir teknik, daha öncesinde Bizans Generalleri Problemi için sunulmuştur. Bizans Generalleri Problemi, dağıtılmış sistemlerde koordinasyon ve entegrasyon problemlerinde çıkan bir mutabakat sorunudur. Bir şehri kuşatan bizanslı generallerden bazıları saldırıyı bazıları geri çekilmeyi savunur. Burada önemli olan nokta herkesin aynı kararı almasıdır. Eğer mutabakata varılamazsa sonuç ortak alınacak saldırı ya da geri çekilme kararından kötü olacaktır. Generaller kararlarını birbirlerine ulaklar aracılığıyla göndermektedir. Ayrıca bazı generaller hain, bazıları özellikle daha kötü olan kararı seçme eğilimindedir. Bu durum ordunun bölünmesine ve en kötü sonucun alınmasına sebebiyet verecektir. İşte bu durumun önüne nasıl geçilebilir sorusu "Bizans Generalleri Problemi" olarak adlandırılmıştır.

1991 yılında, elektronik olarak imzalanan belgelerin bir bilgi zinciri olarak kaydedildiği bir elektronik defter tasarlandı. Yaga ve diğ. (2019) çalışmasında, defteri oluşturan tüm imzalı belgelerin hiçbirinin değiştirilmediğini kolayca gösterebilecek bir sistemin inşa edilmiş olduğunu belirtilmiştir.

1994'te ilk elektronik ödeme işleminin başarılmasına katkıda bulunan Digicash olarak adlandırılan yeni bir sistem ortaya konuldu. Sistem, içeriğini gizlemek için günümüz kripto para birimlerinin kullandığı kriptografik anahtar ve dijital imza benzeri araçlar kullandı. Digicash'in 1998'de iflas etmesine neden olan farklı dezavantajları vardı. En önemli dezavantajlardan biri, harcanan paranın onay olmadan sadece kopyala ve yapıştır ile bir kez daha harcanabildiği çift harcama sorunuydu (Raneem, 2019).

1998 yılında, bilim adamı Nick Szabo, kriptografik bir bulmacayı çözmeye dayalı merkezi olmayan Bitgold dijital para birimini tasarladı. Tasarımda, bir bulmaca çözümü bir sonraki bulmacanın çözümüne bağlanarak oluşturulan zaman damgası eklenmiş ve Proof-of-Work (İş İspatı) kullanan blokların bir zinciri yer alıyordu. Bitgold çözümü sadece teoride kalıp gerçeğe dönüşmemesine rağmen Bitcoin gibi günümüzün kripto para birimleri için gerçek temel olarak kabul edildi. Aynı yıl, Wei Dai tarafından yayınlanan "B-Money, an anonymous distributed cash system" başlıklı makalede dijital paraların temeli anlatıldı (Raneem, 2019).

2000 yılında Konst S. tarafından yayınlanan çalışmada, güvenliği garanti edilen ve kriptografi kurallarını uygulayan başka bir sistem çözümü açıklanmıştır.

Tüm bu çalışmalar ve kavramlar birleştirilerek 2008 yılında elektronik paraya uygulandı ve Satoshi Nakamoto takma adıyla yayınlanan “Bitcoin: A Peer to Peer Electronic Cash System” isimli çalışmaya temel teşkil etti. 2009 yılında ise, Bitcoin kripto para blok zincir ağının kurulmasıyla hayata geçirildi.

Nakamoto’nun makalesinde önerdiği Bitcoin kripto para çözümünden önce de ecash, NetCash gibi elektronik para uygulamaları vardı. Ancak hiçbiri Bitcoin kadar yaygın bir kullanıma erişemedi. Bir blok zincirinin kullanılması Bitcoin'in dağıtılmış bir şekilde uygulanmasını sağladı, banka gibi finans kuruluşları ve aracı kurumlara eşdeğer, herkesin kolayca erişebileceği açık bir finans sistemi kurmanın yolunu açtı ve paranın küreselleşmesini sağladı.

1.1.2. Blok Zincir Avantajları

Sunduğu avantajlı özellikler blok zincir teknolojisinin finans dünyasının başı çektiği bir yolda öne çıkmasına sebep olmuştur.

- Dağıtılmış Yapı: Herkese açık olarak işlem defteri tüm katılımcılarla paylaşılır ve katılımcıları arasındaki her yeni işlemle güncellenir. Bu işlem gerçek zamanlı olarak yürütülür çünkü onu merkezi olarak kontrol eden bir sunucu ve otorite yoktur.

- Düşük Maliyet: Blok zincir ağı, katılımcılar arasında dağıtık bir ağ olduğu için merkezi bir otorite ve altyapıya ihtiyaç duymaz. Bu sebeple, merkezi bir altyapı kurmak için gerekli teknolojik yatırımlara harcama yapılması gerekmez. Katılımcılar kendilerine ait bilgisayarlarla blok zincir ağına katıldıkları için işlem ve bakım ücretleri düşüktür. Madencilik için katılımcılar kullanacakları işlemcileri ve kapasiteleri kendi mali güçlerine göre seçerler (David, 2018).

- Yüksek Güvenlik ve Düşük Dolandırıcılık Riski: İşlemlerin tüm üye katılımcıların erişimine açık şekilde onaylanması, katılımcılar dışında aracıların ya da bir merkezi otoritenin olmayışı blok zincir ağ güvenliğinin yüksek olmasının anahtarıdır. Ancak, blokların kronolojik sırada kaydedildiği zincirde yeni bir blok eklenmesi ancak yeni blok hash şifresinin en azından %51 katılımcı bilgisayarında aynı olmasını gerektirir. Ağın %50’den fazlasını eline geçirebilen tek bir madenci grubu zincir ağı için en önemli tehdit kaynağı olabilir.

- Anonimlik: Kelime anlamı “isimsiz” olan anonim, blok zincir çözümlerinde katılımcıların gerçek isimlerini kullanmadan iletişime geçmeleri anlamındadır. İletişim için kullanıcıların kendi isimlerini kullanmaları gerekmez. Gerçek isimlerin ve kimliklerin kullanılmadığı blok zincirde katılımcılar işlemlerini anonim özellik taşıyan bir ortamda gerçekleştirirler.

- Doğruluk: İşlem bloklarının katılımcılar tarafından doğrulanması ve katılımcıların %51 oranında mutabakata vararak blokları onaylaması, gerçekleşen işlemlerin ve işlemlere ait verilerin değiştirilememesi blok zincir doğruluğunu arttırıcı unsurlardır.

- Hız: Merkezi bir kurum veya aracı bir kurumun varlığında, işlemleri yavaşlatan engellerle karşılaşılır. Finansal alanda, uzun zaman alan banka işlemleri ve resmi tatillerde çalışmayan finans kurumları bu engellere örnek olabilir. Blok zinciri ağı ile kripto para uygulamalarında dünyanın her yerinden herhangi birine anında bir ödeme yapılabilir, çünkü işlem bloklarının işlenmesi kesintisiz olarak günün her saatinde devam eder (David, 2018).

- Şeffaflık: Blok zincir ağına üye olan her katılımcı blok zincir bloklarının bir kopyasına sahip olduğu için bloklara, bloklardaki işlemlere ve işlem verilerine erişebilir ve görüntüleyebilir.

- Mutabakat: İşlemlerin geçerliliği uygulanan mutabakat protokolü kriterlerine göre, tüm katılıcıların mutabakata varması ile gerçekleştirilir.

- Ölçeklenebilirlik: Blok zincir ağının dağıtılmış olması, ağa istenildiğinde katılma ve ağdan ayrılma imkanı sunar. Üye katılımcı sayısını ölçeklendirme, blok zincir ağının kötü niyetli katılımcıların saldırılarına karşı daha dayanıklı hale gelmesine izin verir. Üye katılımcı sayısının artması, saldırganın kullanılan mutabakat protokolünü etkileme yeteneğini azaltır.

- Değişmezlik: Blok zincir teknolojisi hakkındaki çoğu yayın, blok zincir defterlerini değişmez olarak tanımlar. Finansal işlemlerde güvenilir olmalarının bir nedeni kurcalanmaya karşı dayanıklı olmalarıdır. Ancak bu kesinlikle doğru değildir. Tamamen değişmez olarak kabul edilemezler çünkü blok zincirinin değiştirilebileceği durumlar vardır. En son yayınlanan blokların değiştirilme olasılığı mevcuttur. Blok zinciri ağı, birbirine rakip birden fazla zincir içerebilir. Bu durumda, zincirlerden en uzun olanı bir başka deyişle en fazla işlem sayısı içeren zincir gerçek blok zincir olarak atanır. Ancak seçilmeyen zincirdeki işlemler kaybolmaz, farklı bir bloğa kaydedilir ya da bekleyen işlem havuzuna geri döndürülürler. Bu, değişmezlik özelliğini zayıflatan bir durumdur. Çoğu blok zinciri ağ kullanıcısının işlem geçerliliğine yönelik mutabakata varmadan önce beklemesinin nedeni sistemin arka plandaki bu davranışıdır.

1.1.3. Blok Zincir Zorlukları

Finans sektöründe devrim yaratan bir teknoloji olarak gündeme gelen blok zincir, diğer endüstrilere faydalı olacak uygulamaların geliştirilmesine de olanak sağlar. Merkeziyetsiz yapısı, herhangi bir aracıya ihtiyaç duymadan çalışabilme, saklanan verinin kurcalanmaya karşı korunması ve işlem takibi ile veri geçmişine kolay erişim blok zincir yapısının iş ve teknoloji dünyasında öne çıkmasına neden olmuştur. Sunduğu tüm avantajlara rağmen diğer teknolojiler gibi blok zincir teknolojisinin de kendi sınırları vardır ve her iş alanında kullanmak iyi bir fikir olmayabilir.

- Performans ve Ölçeklendirebilirlik: Farklı iş kollarında blok zincir tabanlı çözümlere artan ilgi, özellikle performans ve ölçeklenebilirlik açısından artan talebin karşılanıp karşılanamayacağına dair bir endişe yaratmaktadır. Blok zincir ağının genişletildiğinde, düğümlerin sahip olduğu defter kopya sayısı ve dolayısıyla işlem yoğunluğu artar, bu da yeni bir blok eklenmesinde gecikmelere sebep olur. İş ispatı-Proof of Work (PoW) gibi protokoller, çok sayıda defter kopyasını ölçeklendirebilir, ancak düşük verim ve yüksek gecikmeye sebep olur. Örneğin, iş ispatı protokolünü kullanan Bitcoin, ölçeklendirebilirliği destekler ancak saniyede 6-10 işlem gerçekleşir ve ortalama 10 dakikada bir blok oluşturarak düşük bir verim sağlar. Çift harcama sorununa yol açabilecek çoklu dallanma riski vardır. Bir işlemin tam olarak onaylanması için 6 blok veya bir bloğun yayınlama süresi 10 dakika farz edilirse toplam 60 dakika beklenmesi gerekir. Blok büyüklüğünün sınırlı olması ise, düşük işlem hacmi ve düşük işlem hızı sorununu beraberinde getirir. Ethereum ise iş ispatı protokolünü farklı bir şekilde kullanır ancak Bitcoin'in dezavantajlarını ortadan kaldıramaz. Sistemlerin işlem kapasitesi iki parametreye bağlıdır: blok boyutu ve iki blok arasındaki zaman aralığı. Blok boyutunu artırmak, verimi iyileştirir, ancak sonuçta oluşan daha büyük blokların ağda yayılması daha uzun sürer. Blok aralığını azaltmak gecikmeyi azaltır, ancak sistemin anlaşmazlık içinde olduğu ve blok zincirinin yeniden düzenlenmeye tabi olduğu durumlarda istikrarsızlığa yol açar.

- Mahremiyet: Her ne kadar blok zincir ağında düğümler kendi gerçek kimlikleri dışında takma kimlikler kullansalar da işlem geçmişleri düğümlerin gerçek kimlikleri hakkında ipuçları verebilir. Bir işlemi başlatmak için açık anahtarların diğer düğümler tarafından biliniyor olması, blok zincir işlemlerinin gizlilik açısından savunmasız olabileceğini düşündürebilir. Ayrıca, firewall veya ağ adresleri kullanılarak katılımcı düğüm kimlikleri ile IP adresleri arasında bağlantıların saptanmasına yönelik çalışmalar kripto para mahremiyetini olumsuz şekilde etkileyebilir. Gizlilik ve güvenlik gereksinimlerinin blok zincir uygulamalarının ilk aşamasında tanımlanması mahremiyetin sağlanması açısından önemlidir.

- Birlikte Çalışabilirlik: Birçok iş kolu için farklı blok zincir çözümleri sunulmasına rağmen, bu çözümlerin birbirleriyle entegre olarak ve işbirliği yaparak işlemelerini mümkün hale getirecek standart bir protokol yoktur. Bu birlikte çalışabilirlik eksikliği blok zincir teknolojisinin büyümesi önünde önemli bir engeldir. Blok zincir uygulamaları farklı programlama platformlarında kodlanabilirler, ancak geliştirilen tüm ağlar birbirleriyle etkileşime girmeden çalışırlar. Blok zincir uygulamaları, farklı platformlarda çeşitli programlama dilleri ile geliştirilmiştir, farklı mutabakat mekanizmaları ve gizlilik özellikleri kullanırlar, ancak bu uygulamaları birbirleriyle ve mevcut başka sistemlerle entegre etmek için bir standardizasyon mevcut değildir.

- Enerji Tüketimi: Özellikle iş ispatı mutabakat algoritması, madenci bilgisayarların yüksek miktarda elektrik enerjisi harcamasına yol açar. Madenciler için uygulanan teşvik mekanizması, dünyanın her yerinden insanları para kazanmak için cihazları çalıştırarak kripto para madenciliği yapmaya yöneltir. Örneğin, Bitcoin ağının toplam enerji tüketim oranı, Arjantin gibi bazı ülkelerin enerji tüketiminden daha yüksektir. Büyük miktarda enerji

tüketiminin yanısıra aşırı karbon salınımına yaptığı katkı da blok zincir teknolojisinin sürdürülebilirliği konusunda bir uyarı niteliğindedir.

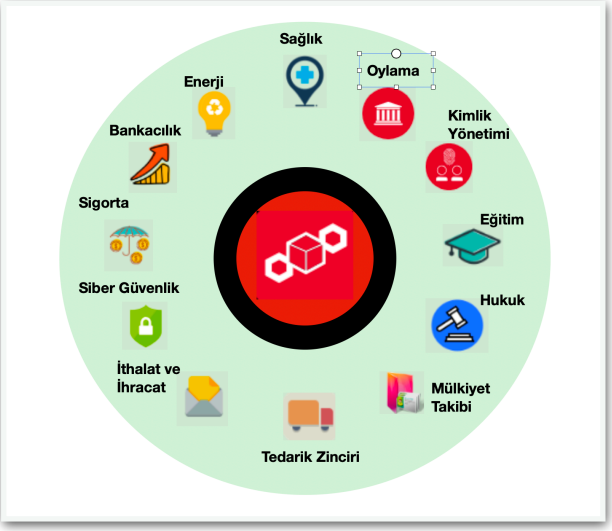
- Güvenlik ve Haksız Madencilik: Blok zincir çözümlerinin güvenliği tehdit eden zayıflıkları vardır. En çok bilenen sorunların başında %51 saldırıları gelir. %51 saldırısında, kötü niyetli katılımcılar, bir blok zincirin yarıdan fazla çoğunluğunu ele geçirirler, işlemlere müdahale ederek çift harcamalar yapar ve diğer madencilerin blokları onaylamasını önleyebilirler. Blok zincir ağının kontrolünü ele geçiren saldırganlar, madencilik ödüllerini arttırmak için Bencil Madencilik (Selfish Mining) adı verilen haksız bir ödül dağılımına sebep olabilir. Bencil bir madenci veya madenci grubu doğruladıkları blokları ağdaki diğer düğümlere yayınlamazlar fakat liderliklerini devam ettirmek için sonraki bloklar için madencilik çalışmalarına devam ederler. Bu nedenle, bencil madenci zinciri uzarken diğer madenciler kaynaklarını işe yaramaz bir dalda boşa harcar. Bencil madencilerin kazandıkları teşvikten pay almak isteyen diğer madencilerden bazılarının bencil madenci zincirine katılması bu zinciri daha da büyütür ve sistemin %51'lik çoğunluğunun ele geçirilmesine yol açabilir. Kendini ispat etmiş blok zincir platformları, saldırılara karşı daha dayanıklıdırlar ve çok az ciddi hata ile karşılaşırlar. Bununla birlikte, bu platformalarda geliştirilen akıllı sözleşmeler gibi uygulamalar hala ciddi sonuçları olabilecek hatalara karşı hassastır. Bu güvenlik tehditleri giderilene kadar potansiyel kullanıcılar dikkatli olmaya devam edecek ve toplu olarak benimseme gecikecektir.

- Hukuksal Düzenleme Sorunları: Özellikle finansal alanda geliştirilen blok zincir çözümlerinin sunduğu merkeziyetsiz ödeme sistemleri, ülkelerin para politikasından sorumlu merkez bankaları için endişe uyandıran bir rakip niteliğindedir. Buna karşın, bu ödeme sistemlerine karşı artan ilgi, dünya genelinde daha geniş bir kullanıma sahip olmalarına yol açmaktadır. Fakat, para aklama gibi bazı yasa dışı eylemlerle ilişkilendirilmeleri blok zincir teknolojisinin adını da karalamaktadır. Bu sebeple, birçok ülke blok zincir temelli ödeme sistemelerine kuşku ile yaklaşmakta, hatta bazı ülkeler kullanımlarını yasaklamaktadır. Bunun önüne geçmek için, blok zincir işleyişinin tüm detaylarına hakim olduktan sonra blok zincir oluşturma maliyetleri ve faydaları, teşvik ayarları, gelişen standartlar, ölçeklenebilirlik, yönetim, yasal riskler, güvenlik ve düzenleyici müdahaleler ile ilgili konuları içeren kanunlar ve düzenlemeler yapılmalıdır. Hukuksal alandaki eksikliklerin giderilmesi, bu teknolojinin ilerlemesinde ve hızlı şekilde gelişmesinde olumlu bir etki yapacaktır.

1.1.4. Blok Zincir Uygulama Alanları

Blok zincir teknolojisi her ne kadar finansal alanda dijital ödeme sistemleri ile hayata geçmiş olsa da farklı iş kollarında blok zincir uygulamaları görülmeye başlamıştır. Gelecek yıllarda şüphesiz daha başka alanlardaki uygulamaları da söz konusu olacaktır.

Belli başlı blok zincir uygulama alanları Şekil 1'de gösterilmiştir.



Şekil 1: Blok Zincir Uygulama Alanları

1.1.4.1. Finans ve Bankacılık

Günlerce sürebilen ülkelerarası para transferlerinin dakikalar içinde gerçekleşebilmesi, blok zincir teknolojisini kullanan kripto para birimlerinin çoğalmasına ve yaygınlaşmasına yol açtı. Blok zincir kullanımının getirdiği düşük maliyet ve komisyonlar, kripto para birimlerine diğer varlıklar ve ödeme sistemleri karşısında ayrıcalık kazandırdı. Kripto varlıklar finans dünyasında değer kazanarak diğerleriyle daha iyi rekabet etme avantajına kavuştular.

1.1.4.2. İthalat ve İhracat

Evrak işleri, banka işlemleri ve uzun mesafeli iletişimlerin yoğun olduğu dış ticaret sektöründe, blok zinciri çözümü, merkezi olmayan bir defter üzerinden gerçek zamanlı bilgi alışverişi sağlayarak ihracatçılar ve ithalatçılar arasındaki bağlantıyı kuvvetlendirebilir. Faturalar, rezervasyon, teslimat, siparişler gibi tüm temel belgelerin şifrelendirilerek güvenli bir şekilde teminat altına alınmasını sağlar.

1.1.4.3. Petrol ve Enerji

Bir kısım kripto para uygulaması petrol ve enerji üreticilerine özel blok zincir hizmetleri sunmaya çalışırlar. Bazı para birimleri, madencilik için elektrik enerjisi yerine güneş enerjisini kullanarak kazanç sağlamayı hedeflerler. Petrol ve enerji endüstrisindeki karmaşık tedarik zincirinde, katılımcıların kendi kayıtlarını tutmaları, işlemlerini ve sistemlerini güncellemeleri ve ağdaki diğer katılımcılarla mutabakatları blok zincir sayesinde hızlı, eksiksiz ve düşük bir maliyet ile mümkün olur.

1.1.4.4. Elektronik Mağazacılığı

Günümüzde elektronik alışveriş firmalarının sunduğu satıcı ve tüketici arasındaki aracılık hizmetini üstlenmeye aday yeni blok zincir elektronik mağaza uygulamaları sayesinde tam anlamıyla şifrelenmiş müşteri verisi ve alışveriş işlemleri gerçekleşebilecektir. Kripto para birimleri kullanılarak yapılacak mağaza içi işlemler herhangi bir üçüncü taraf tarafından izlenemeyecektir.

1.1.4.5. Elektronik Ortam ve Web Siteleri

İnternet reklamcılığı, elektronik ortamlar ve web siteleri için önemli bir gelir kaynağıdır. Blok zincir ile birlikte kullanıcı bilgisayarlarında reklamlar yerine kripto para madenciliği yapmak ve bundan gelir elde etmek hedeflenecektir.

1.1.4.6. Emlak Ticareti

Gayrimenkul sahipliğinin kişiden kişiye devrinin hızlanması, akıllı kontratların yapılandırılması ve finansal işlemlerin kronolojik sıralanmasını mümkün hale getiren blok zincir, emlak alım-satımında da öne çıkan bir teknoloji olmaya adaydır.

1.1.4.7. Mülkiyet Takibi

Kayıtlı olmayan mülkiyetlerin kayıt altına alınması, varlık değerlerinin korunması ve değer kaybının önlenmesi için blok zincir kullanımı önemlidir. Mülkiyet haklarını kaydetme süreci oldukça külfetli ve zaman alıcıdır. Mülkiyet hakkını gösteren belgelerin tapu dairesi gibi kamuya ait kayıt ofislerinde veri tabanına manuel olarak girilmesi gerekir. Ayrıca, anlaşmazlık durumu söz konusu olursa, mülkle ilgili talep ve itirazlara yönelik kamu kayıt veritabanı ile mutabık kalınması şarttır.

1.1.4.8. Gıda Tedarik Zinciri

Üreticiden tüketiciye ulaşıncaya kadar gıdaların üretim, paketleme, saklama, dağıtım ve taşıma adımlarından oluşan tedarik sürecinin kayıt altına alınması ve tüketiciye satışının akıllı kontrat ile gerçekleşmesini sağlamak için blok zincir yapısı kullanılabilir. Blok zincirinde kaydedilen her gıda tedarik adımı, değiştirilemez özelliği sayesinde gıda güvenliği açısından önem taşır.

1.1.4.9. Sağlık

Sağlık hizmetleri sistemleri arasında veriler aktarılırken hasta kayıtlarının büyük bir kısmında uyumsuzluk olur. Bu nedenle, blok zincir uygulanarak hastaların kayıtları izlenebilir ve blok zincir değişmez olduğu için veriler herhangi bir manipülasyon veya bozulma olmadan sistemler arası kolayca gönderilebilir.

1.1.5. Blok Zincir Türleri

Blok zincir ağları, ağa katılan düğümlerin kontrol şekline göre iki farklı kategoride değerlendirilir.

1.1.5.1. İzinsiz (Permissionless) Blok Zincir

İzinsiz bir blok zincir açık (public) blok zinciri olarak da bilinir çünkü herhangi bir yetkilendirme olmaksızın ücretsiz katılmaya ve ayrılmaya izin verir. Ağ içindeki herhangi bir katılımcı, blok yayınlayabilir, blok zincir defterini okuyup yazabilir. İzinsiz blok zinciri ağları herkesin katılımına açık olduğundan, kötü niyetli kullanıcılar, ağ işleyişini bozacak şekilde bloklar yayınlamaya çalışabilir. Bunu önlemek için, izinsiz blok zinciri ağları genellikle bir "mutabakat" protokolü kullanır. Öne gelen mutabakat protokolü örnekleri arasında iş kanıtı proof of work ve hisse kanıtı-proof of stake yöntemleri bulunur. Protokole uygun blokları yayınlayan katılımcılar ödüllendirilerek dürüst ve kötü niyetli olmayan katılımcıların davranışları teşvik edilir. İzinsiz blok zinciri uygulamaları genellikle açık kaynaklı yazılımlardır ve bunlara erişmek isteyen herkes tarafından ücretsiz olarak kullanılabilir.

Tablo 1: Blok Zincir Türleri ve Özellikleri

ÖZELLİK	İZİNSİZ BLOK ZİNCİR	İZİN VERİLEN BLOK ZİNCİR
Yönetim	Herkese açık	Yetkili kullanıcılar
Katılım	Ücretsiz katılım ve ayrılma	Yetki verme
Kimlik	Takma isim ve gizli	Açığa
Yazılım Kodu	Açık	Açık ya da açık değil
Ağ büyüklüğü	Binlerce düğüm ve fazlası	Onlar ve yüzlerce düğüm
Bağlantı	Düşük	Yüksek, genellikle tam bağlantı
Eş Zamanlama	Eş zamanlı değil	Eş zamanlı
İşlem Kapasitesi	Düşük (Saniyede onlar bazında)	Yüksek (Saniyede binler bazında)
Uygulamalar	Kriptopara, akıllı kontrat, dağıtılmış uygulamalar	Bankalar arası takas, tedarik zinciri

1.1.5.2. İzin Verilen (Permissioned) Blok Zincir

İzin verilen bir blok zinciri, katılımcıların ağa dahil olmadan önce yetkilendirilmesini ve kendi açık kimlikleriyle işlemlerini yapmalarını gerektirir. Ağın yönetimi sadece yetki sahibi kullanıcılar tarafından yapılır. İşlem yapma, defter okuma hakkı yetkilendirilmiş bir grup katılımcıya verilebileceği gibi tüm katılımcılara da bu haklar verilebilir. Kimlik zorunluluğu olması, kötü niyetli katılımcıların açığa çıkmasına ve yetkilerinin iptal edilmesine yol açar. Mutabakat daha hızlı gerçekleşir ve mutabakat maliyeti daha azdır. Bu sayede etkili bir ağ

yönetimi sağlanmış olur. İzinsiz blok zinciri ile karşılaştırıldığında, kimlik açığa çıkaran işleyiş ve daha etkili bir ağ yönetimi mevcuttur. Xiao ve diğ. (2020) çalışmasında, izin verilen blok zincir ağlarının kontrolün öne çıktığı kuruluşlar veya birbirine tam olarak güvenmeyen ancak birlikte çalışmak isteyen kuruluşlar için ideal olduğunu ifade edilmiştir.

1.1.6. Blok Zincir Mimarisi

Bir blok zincir ağının temelini oluşturan unsurların detaylandırılması ağ yapısının anlaşılması adına önemlidir. Hash fonksiyonları, dijital imzalar, asimetrik anahtar kriptografi, defterler, bloklar, işlemler bir blok zinciri ağının başlıca bileşenleridir.

1.1.6.1. Kriptografik Hash Fonksiyonları

Hash fonksiyonları, bir verinin bütünlüğünün belirlenmesini sağlar. Fonksiyon, bir metin, bir dosya içeriği hatta bir görüntüyü girdi olarak alır. Girdi olarak alınan verinin içeriğinde küçük bir değişiklik olsa bile, çok farklı bir hash değerinin elde edilmesine sebep olur. Bir kriptografik hash fonksiyonu bir dosya veya şifre üzerinde checksum adı verilen bir değer üretmek için çalışan bir algoritmadır. Bu özellik, dijital imzaların ve işlemlerin doğrulanmasında, rastgele sayıların veya bitlerin oluşturulmasında ve blok zincirinde blok verilerinin ve başlıklarının güvenliğinin sağlanmasında kullanışlıdır.

Narayanan ve diğ. (2016) çalışmalarında kriptografik hash fonksiyonun özelliklerini şöyle sıralarlar:

- Girdi, herhangi bir boyuttaki bir dizgi olabilir.
- Sabit boyutta bir dizgi üretir.
- Belirli bir girdi dizgisi için, fonksiyon çıktısı makul bir süre içinde elde edilir.

Bir kriptografik hash özet fonksiyonunun güvenilir olması için;

- İki farklı girdi dizgisi için aynı çıktıyı üretmemesi gerekir. (Collision-resistent) (256 bitlik çıktı üreten bir hash fonksiyonu için bir çarpışma bulma olasılığı hala son derece küçük)
- Verilen bir çıktı için girdi dizgisinin geriye dönük hesaplanamaması
- Kullanıcı herhangi bir beklenen çıktı değerine dayalı girdileri seçememelidir, tüm girdi değerleri aralığı istenen çıktıyı döndürmek için eşit bir şansa sahip olmalıdır. Aksi takdirde, kullanıcı belirli aralıklar içindeki girdi değerlerine karşı ayrımcılık yapabilir, böylece arama alanını daraltabilir ve geçerli bir çıktı bulma şansını artırabilir.

Hash fonksiyonu çok güçlü bir fonksiyondur çünkü tek yönlüdür. Bir başka deyişle, hash fonksiyonun çıktı değeri bilindiğinde bu çıktıyı oluşturan doğru girdi değerini hesaplamak mümkün değildir. Bunun dışında, bir çıktı değerini üreten ancak ve ancak bir girdi değeri vardır, yani aynı çıktı değerini oluşturan ikinci bir girdi değeri yoktur. Hash fonksiyonlarında öne çıkan bir başka özellik çarpışma direnci olarak adlandırılır. Çarpışma direnci aynı çıktı değerini üreten birbirinden farklı iki ayrı girdi değeri bulunamaması özelliğidir. Aslında, çarpışmalar vardır, çünkü hash fonksiyonun girdi uzunluğu ve alabileceği değerleri sonsuzdur,

fakat çıktı uzunluğu ve alabileceği değerleri sonludur. Ancak, bir çarpışma durumunun oluşma olasılığı önümüzdeki iki saniye içinde dünyanın dev bir meteor tarafından yok edilme olasılığından bile daha düşüktür.

Günümüze kadar pek çok farklı hash fonksiyonu kullanılmıştır. İlk defa 1990'lı yıllarda MD-4 ve MD-5 algoritmaları kullanılmaya başlanmıştır. Bu algoritmaların güvenlik zafiyetlerinin ortaya çıkması sonrasında, MD-4 ve MD-5 tabanlı geliştirilen SHA-0 ve SHA-1 algoritmaları da kırılarak güvenilirliklerini kaybetmişlerdir. Günümüzde SHA-2 ve Ekim 2012 yılında NIST yarışmasını kazanan Keccak algoritmaları en çok adı geçen algoritmalardır.

Blok zincir uygulamalarında kullanılan en popüler hash fonksiyonu 256 bit uzunluğunda özet bir çıktı üreten Secure Hash Algorithm, SHA-256 algoritmasıdır. Hesaplama hızı yüksek olduğu için donanımların çoğu bu algoritmayı destekler. SHA-256 algoritması her ne kadar 256 bit=32 byte (1 byte=8bit) uzunluğunda olsa da genellikle 16'lık sayı düzeninde (hexadecimal- 4 bir 1 karakter olacak şekilde) 64 karakter uzunluğunda gösterilir. Yaga ve diğ. (2019) çalışmasında 256 bit uzunluğundaki özet çıktı için, $2^{256} \approx 10^{77}$ farklı çıktı değeri üretildiğini belirtmiştir. Narayanan ve diğ. (2016) çalışmasındaki örnek SHA-256 girdi ve çıktı değerleri Tablo 2'de yer almaktadır.

Tablo 2: SHA-256 örnek girdi ve çıktı değerleri

GİRDİ	SHA-256 ÖZET ÇIKTI
1	0x6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b
2	0xd4735e3a265e16eee03f59718b9b5d03019c07d8b6c51f90da3a666eec13ab35
Hello World	0xdffd6021bb2bd5b0af676290809ec3a53191dd81c7f70a4b28688a362182986f

Hash fonksiyonu ön işlem ve hesaplama olmak üzere iki aşamadan oluşur. Ön işlem aşamasında, girdi mesajın doldurulması, doldurulan mesajın bloklara ayrılması ve hesaplama aşamasında kullanılacak başlangıç değerlerinin belirlenmesi adımları yer alır. Doldurma işlemi, girdi olarak alınan mesajın bit uzunluğunun kullanılan algoritmaya bağlı olarak 512 ya da 1024 ile orantılı olmasını sağlamak için mesaj sonuna gerekli sayıda bit eklenmesi işlemidir.

SHA-256 algoritmasında, bit uzunluğu "l" olan bir girdi mesajın sonuna sırasıyla;

- 1 adet bit "1"
- (512 - 64 - l -1) uzunluğunda bit "0"
- Girdi mesaj uzunluğunun binary gösterimini içeren 64 bit uzunluğunda bir blok eklenir.

İlk adımda elde edilen doldurulmuş mesaj, bir sonraki adımda 512 bit uzunluğunda bloklara ayrılır. SHA-256 algoritmasında word uzunluğu 32 bit olarak belirlenmiştir. Bu bağlamda, 512 bit uzunluğundaki her bir blok 16 adet word oluşturmuş olur. Ön işlem aşamasının son adımında ise, SHA-256 algoritması sonucu elde edilecek 256 bit uzunluğunda çıktıyı

oluşturan 8 adet 32 bit uzunluğundaki word için başlangıç hash değeri ataması yapılır. Bu değerler, ilk 8 asal sayının (2..19) kareköklerinin ondalık kısımlarının 32 bit uzunluğunda heksadesimal karşılıklarından oluşturulur.

Hash hesaplama aşamasında, 512 bit uzunluğundaki her blok arka arkaya işleme alınır. Orijinali 512 bit olan mesaj bloğu 16 adet 32 bitlik word dizgisini oluşturur. W_0'dan W_{15}'e elde edilen 16 adet word dizgisi 64 word dizgisine (W_0 - W_{63}) genişletilir. Bu genişletilme aşağıdaki tekrarlama formülüne göre yapılır: (Gilbert ve Handschuh, 2004)

$$W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} \quad 16 \leq t \leq 63$$

σ_0 ve σ_1 fonksiyonları 32 bit word dizgilerini girdi olarak alır ve 32 bit word dizgilerini üretir.

$$\sigma_0(x) = \text{ROTR}^{7}(x) \oplus \text{ROTR}^{18}(x) \oplus \text{SHR}^{3}(x)$$
$$\sigma_1(x) = \text{ROTR}^{17}(x) \oplus \text{ROTR}^{19}(x) \oplus \text{SHR}^{10}(x)$$

$\text{SHR}^n(x)$ fonksiyonu, x dizgisinin bitlerini n basamak sağa doğru kaydırmak için kullanılır. Kaydırma sonucunda dizgi sonunda yer alan bitler kaybedilir ve onların yerine dizginin başlangıcının sıfırlarla değiştirilmiş olur. $\text{ROTR}^n(x)$ fonksiyonu ise, sağa doğru kaydırılan bitler dizgi sonuna ulaştıklarında kaybedilmez ve dizginin başlangıcına döndürülür (Quynh, 2015).

Bir önceki adımda oluşturulan 32 bit uzunluğunda 64 adet word dizgisine sırasıyla bir sıkılaştırma işlemi uygulanır. Bu işlemde, 8 ara 32 bitlik word ara değişken (a,b,c,d,e,f,g) kullanılır. Başlangıçta bu ara değişkenlere ön işlem aşamasında belirlenen başlangıç hash değerleri atanır. Sıkılaştırma işleminde, her word dizgisi için ara değişkenler yeniden hesaplanır ve hesaplanan ara değişken değerleri hash başlangıç değerlerine eklenir. Tüm dizgiler için 64 kez tekrarlanan hesaplamalar sonrasında hash elde edilmiş olur. Hesaplamaları yaparken kullanılan 4 ana fonksiyon aşağıda yer almaktadır. (Gilbert ve Handschuh, 2004)

$$\textbf{\textit{Ch}}(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z)$$
$$\textbf{\textit{Maj}}(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$
$$\Sigma_0(x) = \text{ROTR}^{2}(x) \oplus \text{ROTR}^{13}(x) \oplus \text{ROTR}^{22}(x)$$
$$\Sigma_1(x) = \text{ROTR}^{6}(x) \oplus \text{ROTR}^{11}(x) \oplus \text{ROTR}^{25}(x)$$

1.1.6.2. Dijital İmza

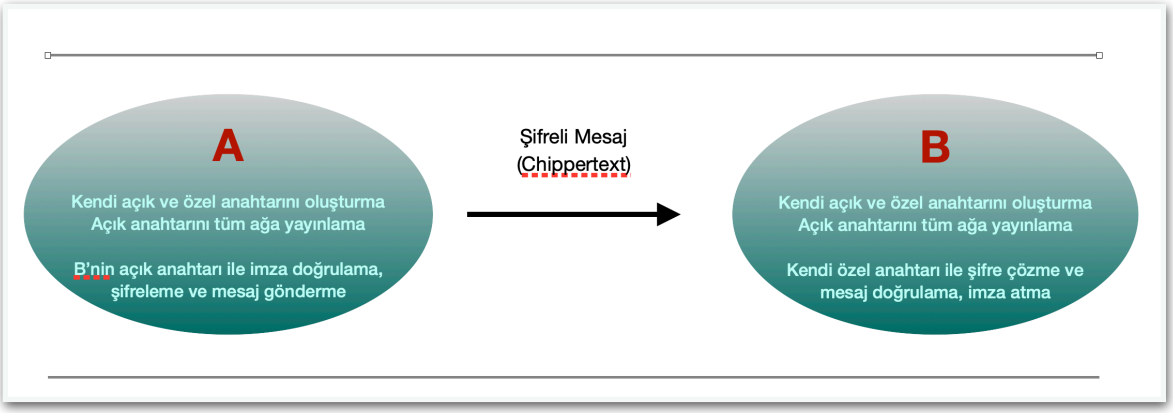
Dijital imzalar genellikle siber güvenlik dünyasında kimliği kanıtlamak için kullanılır. Asimetrik kriptografiye dayalı bir dijital imza, blok zinciri ağı gibi güvenilmez bir ortamda faydalıdır. Blok zincir ağındaki her katılımcı bir özel anahtar ve açık anahtar çiftine sahiptir. Özel anahtar, işlemleri imzalamak için kullanılır. Dijital imzalı işlemler ağdaki tüm katılımcılara dağıtılır ve daha sonra ağdaki herkes tarafından görülebilen açık anahtarlarla erişilir. Özel anahtar işlemi imzalamak veya şifrelemek için kullanılırken, açık anahtar ağdaki tüm katılımcılara dağıtılır ve herkes tarafından görülebilir, bu da bir sonraki işlemin şifresinin çözülmesine yardımcı olur.

1.1.6.3. Simetrik Olmayan Açık Anahtar Kriptografi

İkinci Dünya Savaşı'nda güvenli bir iletişim kurabilmek için kullanılan şifreli mesajlaşma, savaş sonrasında kriptografinin temelini oluşturmuştur. Zor ve karmaşık matematiksel denklemlerin çözümü esasına dayanan kriptografi, bilgisayarların oluşturduğu bir ağ kapsamında gerçekleştirilen elektronik ödeme sistemlerinin bir başka deyişle kripto para birimlerinin ortaya çıkmasına vesile olmuştur.

Simetrik olmayan kriptografi birbirilerini tanımayan kişilerin güvenli bir şekilde haberleşmesini sağlar. Gönderilen mesajı şifrelemek için kullanılan açık anahtar (public key) ve mesaj şifresini çözmek için kullanılan anahtar (private key) birbirinden farklıdır. Aslında açık ve özel anahtarlar arasında bir ilişki olsa da şifre çözmek için kullanılan özel anahtarları açık anahtardan türetmek mümkün değildir.

Ağ içerisindeki tüm katılımcılar kendi açık ve özel anahtarlarına sahiptir. Her katılımcı açık anahtarını tüm ağ ile paylaşır. Açık anahtarların yayımlanmasında sorun yoktur, çünkü bu sayede diğer ağ katılımcıları istedikleri katılımcıya şifreli mesaj atabilir. Ancak katılımcılar özel anahtarlarını gizli tutmak zorundadır. İletişime geçmek isteyen katılımcı karşı taraftaki katılımcının açık anahtarı ile mesajı şifreler. Şifrelenmiş mesajı alan taraf ise kendi özel anahtarı ile mesajın şifresini çözer. Diğer bir deyişle, Şekil 2'de gösterildiği gibi açık anahtar kullanarak şifrelenen mesajın özel anahtar kullanarak şifresi çözülür.



Şekil 2: Açık Anahtar Kriptografi

Simetrik olmayan kriptografide sık kullanılan algoritmalardan biri olan RSA (Rivest-Shamir-Adleman) algoritmasında, açık ve özel anahtarların rolleri yer değiştirebilir. Bir başka ifadeyle, gönderilen mesaj gönderen kişinin özel anahtar ile şifrelenirken şifre çözme işlemi için gönderen kişinin açık anahtarı kullanılabilir, böylece şifreleme özel anahtarlar, şifre çözme açık anahtarla yapılabilir. Bu sayede, elektronik imza elde edilir. Elektronik imza sadece gizli anahtar ile atılabilir, herkes imzanın doğruluğunu kontrol edebilir. Tüm mesaj içeriğini imzalamak maliyetli bir işlemdir, bu sebeple mesajın sadece hash kısmı imzalanır.

Elektronik imzalar kriptografinin temelini oluşturur. Her kişi kendi imzasını yapar ancak imzayı gören herkes onun geçerliliğini doğrulayabilir. Ayrıca, imza belirli bir belgeyle

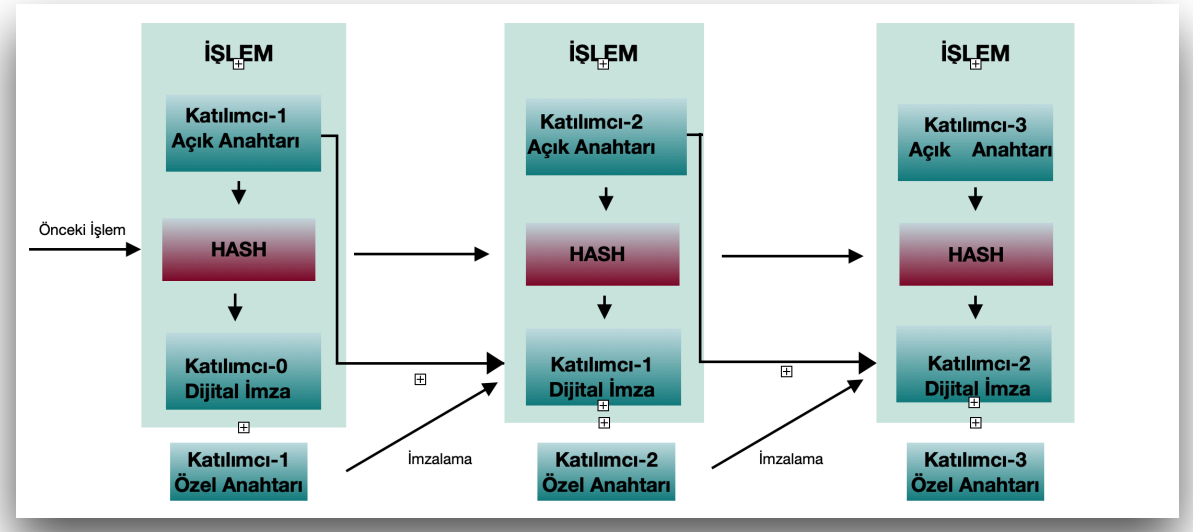
ilişkilendirildiğinde sadece o belgenin kabulü veya onaylanması için kullanılır, farklı bir belge için kullanılamaz.

Mesaj imzalama sig = sign(secretkey, message) özel anahtar kullanılır
Mesaj doğrulamı IsValid = verify(publickey, message,sig) açık anahtar kullanılır

Simetrik olmayan şifreleme işlemi simetrik şifrelemeye göre daha yavaştır. Asimetrik anahtar şifreleme kullanarak bir şeyi şifrelediğinde çoğu zaman veriler simetrik anahtar şifreleme ile, simetrik anahtar ise asimetrik anahtar şifreleme yöntemi ile şifrelenir. Bu sayede asimetrik anahtar şifrelemesinin büyük ölçüde hızlandırıldığı Yaga ve diğ. (2019) çalışmasında ifade edilmiştir.

1.1.6.4. İşlemler

Blok zincir ağını oluşturan katılımcılar arasında karşılıklı yapılan elektronik para, eşya, belge gibi varlık aktarımını temsil eden işlemler, varlık aktarım aktivite ve etkileşimlerine ilişkin verileri ve sonuç kaydını tutar. İşlemler, Şekil 3'de görüldüğü gibi genellikle asimetrik anahtar şifreleme uygulamasına uygun gönderenin özel anahtarı ile dijital olarak imzalanır, ilgili açık anahtar kullanılarak doğrulanır ve bir işlem zinciri oluşturulur. Ön planda görülen elektronik varlık aktarımı olsa da işlemler asıl olarak veri aktarımı için kullanılırlar. Akıllı kontrat sistemleri söz konusu olduğunda, işlemler veri göndermek, bu verileri işlemek ve bazı sonuçları blok zincirinde depolamak için kullanılabilir. Örneğin, blok zinciri teknolojisi tabanlı bir tedarik zinciri sistemi içindeki bir gönderinin konumu gibi elektronik bir varlığın bir özelliğini değiştirmek için bir işlem kullanılabilir.



Şekil 3: İşlem Zinciri

Blok zincir teknolojisini öne çıkartan ilk uygulama olan Bitcoin kripto para birimi ağında, bir işlem üç kısımdan oluşur.

Meta Veriler: İşlemin büyüklüğü, girdilerin ve çıktıların sayısı, bir işlem yayınlanmadan önce beklenmesi gereken süreyi temsil eden lock_time parametresi gibi bazı temel bilgilerin tutulduğu kısımdır.

Girdiler: İşleme girdi olacak önceki işlemin hash özet değeri ve kullanılacak çıktının önceki işlemdeki sıra numarası mevcut işlemin girdileridir. Girdiler bir dizi oluşturur.

Çıktılar: Çıktılar, iki sayısal alanı olan bir dizidir. Bir işleme ait iki çıktı değerinin toplamı, tüm girdi değerlerinin toplamından küçük veya eşit olmalıdır. Çıktı değerlerinin toplamı, girdi değerlerinin toplamından daha küçük olması durumunda girdi toplamı ve çıktı toplamı arasındaki fark bu işlemi yayınlayan madenciye bir işlem ücreti olarak atanır.

Tablo 3: Bitcoin Örnek İşlem Bazlı Defteri

İŞLEM SIRASI	GİRDİ (INPUT)		ÇIKTI (OUTPUT)	
	Önceki İşlem Sırası	Önceki İşlem Çıktı Sırası	0. Çıktı	1. Çıktı
1	0		25 [A]	-
2	1	0	17 [B]	8 [A]
3	2	0	8 [C]	9 [B]
4	2	1	6 [D]	2 [A]
5	3	1	1 [C]	8 [B]

Bir para transfer işlemi bir kripto para birimi işlem defterine kaydedilirken işlem girdi (input) ve işlem çıktı (output) dizilerinin belirtilmesi gerekir. İşlem girdileri harcanan madeni paralar (coin), çıktıları ise oluşturulan madeni paralar (coin) olarak düşünülebilir. Çıktı dizisi 0 ile başlayacak şekilde indekslenir.

Basitleştirilmiş örnek bir para transfer hareketlerinin, işlem bazlı bir deftere işlenmesi Tablo 3'te gösterilmiştir. İlk işlemin girdisi bulunmuyor çünkü bu işlem yeni madeni paralar yaratıyor ve A'ya giden 25 coin çıktısı var. Ayrıca, bu yeni madeni paraların yaratıldığı bir işlem olduğu için imza gerekmiyor. A, bu paralardan bazılarını B'ye göndermek istediğinde, yeni bir işlem yaratır.

Tablo 4. Bitcoin Örnek İşlemleri

TRANSFER İŞLEMLERİ
Madencilerin çıkardığı 25 coin A'ya transfer
A'dan B'ye 17 coin transfer ve A'nın imzası
B'den C'ye 8 coin transfer ve B'nin imzası
A'dan D'ye 6 coin transfer ve A'nın imzası
B'den C'ye 1 coin transfer ve B'nin imzası

Yeni işlem girdi olarak bu madeni paraların geldiği önceki işleme açıkça atıfta bulunmalıdır. Tablo 4'teki örnekte yer alan ikinci işlem yaratılırken, A'ya 25 coin atayan birinci işlemin sıfırıncı çıktısı işlem girdisi olarak işaretleniyor. İşlem çıktısı, ilk indeksi 0 olan iki elemanlı bir dizidir. Dizinin 0. elemanı paranın gönderildiği B kişisinin yeni bakiyesini, 1. elemanı ise parayı gönderen A kişisinin kalan bakiyesini gösterecek şekilde işaretlenir.

İşlemlerin birçok girdisi ve birçok çıktısı olabilir. Örneğin, C kişisine, birisinde 8 diğerinde 1 madeni para olmak üzere iki farklı işlemde para aktarılıyor. Sonrasında, eğer C 8 madeni paranın hepsini kullanarak bir işlem yapmak isterse, iki girdiyi kullanarak bir çıktının yer aldığı bir işlem oluşturur. Bu, iki işlemi konsolide etmesine olanak tanır.

Benzer şekilde, ortak ödemelerin yapılması da kolaydır. B ve C kişilerinin ikisi de D kişisine ödeme yapmak isterse, iki girdi ve bir çıktıyla, ancak iki farklı kişiye ait iki girdiyle bir işlem oluşturabilirler. Ve Tablo 3'de yer alan örnekten farklı olarak, burada talep edilen önceki işlemlerin iki çıktısının farklı adreslerden olması nedeniyle, işlemin hem B hem de C'nin imzasına ihtiyaç duymasıdır.

Sonuç olarak, madeni para (coin) sahibi olan katılımcı madeni parayı bir başka katılımcıya gönderirken dijital imza atar. Bu dijital imza, bir önceki işlemin hash değeri ve paranın gönderileceği katılımcının açık anahtarından oluştur. Parayı alan katılımcı kendi açık anahtarını kullanarak dijital imzayı doğrular. Bu doğrultuda, madeni para dijital imzaların bir zinciridir (Satoshi, 2008).

Bazı blok zincir uygulamalarında, katılımcı açık anahtarı yerine açık anahtarın kriptografik bir hash fonksiyondan türetilen daha kısa alfanumerik adres değeri kullanılır. Bir transfer işleminde, alıcı ve gönderici için türetilmiş adres bilgileri geçerli olur.

1.1.6.5. Defter (Ledger)

Muhasebe işlemlerinde tarih boyunca kullanılan kağıt defterler yerine modern zamanlarda muhasebe defterleri elektronik ortamlarda büyük veritabanlarında saklanmıştır. Genellikle bu elektronik defterlere sahip olan ve onları işleyen merkezi bir üçüncü tarafın varlığı sisteme güven duyulması ve sistem güvenliğinin sağlanması noktasında bazı sorunları gündeme

getirmiştir. Farklı sistem arayışları neticesinde defterlerin hem fiziksel hem de mülkiyet olarak çok daha geniş bir bilgisayar ve kullanıcı grubuna dağıtılması ilgi çekici hale gelmiştir. Blok zincir teknolojisi, hem dağıtılmış mülkiyeti hem de dağıtılmış bir fiziksel mimariyi kullanarak böyle bir yaklaşımı mümkün kılar.

İşlemlerin güvenli bir şekilde kaydedildiği tüm ağ katılımcılarına açık bir defter blok zincir teknolojisinin en önemli bileşenlerinden biridir. Defter, temelde şifrelenmiş işlem grupları olan bloklardan oluşur. Bu bloklar deftere kronolojik bir sırayla kaydedilirler. Blok zincirine yeni bir blok ekleyebilmek için, %51 oranında katılımcının fikir birliğine varması gerekir. Bir diğer deyişle, şifrelenmiş hash özet değerinin katılımcıların en az %51'i için aynı olması gerekir. Ağ katılımcıları fikir birliğine vardıkları işlemlerin bir listesini içeren bir dizi bloktan oluşan defterin bir kopyasını kendi bilgisayarlarında bulundururlar.

Asıl olarak, defter (ledger) işlemlerin veritabanı anlamındadır ve ağdaki tüm katılımcılar tarafından paylaşılır. Defter blok zincir ağındaki her bir katılımcı arasında paylaşıldığından, her katılımıcının erişebileceği anlamına gelir. Ledger isimli defterde yer alan işlemleri değiştirmek zordur fakat yüzde yüz imkansız olduğunu söylenemez (Satoshi, 2008).

1.1.6.6. Bloklar

İşlemler gruplar halinde biraya gelerek blokları oluştururlar. Aslında bu bir optimizasyondur. Madenciler her bir işlem üzerinde ayrı ayrı mutabakata varsaydı, yeni işlemlerin blok zincir tarafından kabul edilme oranı çok daha düşük olurdu. Ayrıca, işlemlerin oluşturacağı bir zincir yerine çok sayıda işlemin meydana getirdiği blokların oluşturacağı bir zincir daha kısa olacağı için blok zinciri yapısını çok daha verimli hale getirecektir.

Blok zincir ağ üzerinde gerçekleşen işlem verilerini bloklar halinde saklayan bir yapıdır. Her blok kendisinden önce gelen bloğun hash değerini saklar. Bu sayede, bloklar kendilerinden önce gelen blokları gösterecek şekilde birbirlerine bağlanmış olurlar. Blok zincirin ilk bloğuna genesis blok adı verilir ve öncesinde bağlı olduğu herhangi bir blok bulunmaz. Bir blok, blok başlığı ve blok gövdesi olmak üzere iki kısımdan oluşur.

Tablo 5: Blok Başlık Alanları ve Açıklamaları

BLOK BAŞLIK ALANLARI	AÇIKLAMA
Blok Versiyon	Blok kabul ve red kuralları
Önceki Blok Hash	Önceki bloğa ait 256 bit hash değeri
Blok Hash	Merkle Ağacı kök hash değeri
Timestamp	Zaman Damgası
nBits	32 bitlik sıkıştırılmış hedef blok hash değeri
Nonce	4 byte uzunluğunda

Blok hash değeri, blokta yer alan işlemlere ait hash değerlerin oluşturduğu bir merkle ağacının kök hash değeri alınarak elde edilir. İşlem hash'leri ikili gruplar halinde birbirine bağlanarak elde edilen 512 bit (256x2) uzunluğundaki değer için hash hesaplaması yapılır. Eğer bloktaki işlemlerin sayısı tek sayı ise, son işlem kendisi ile birleştirilerek 512 bit olarak hash fonksiyonuna gönderilir. Böylece ağacın ikinci seviyesi oluşturulmuş olur. Oluşan yeni seviyede ikiden fazla hash değeri bulunuyorsa üçüncü seviyeyi oluşturmak için hash'ler eşleştirilir ve yeni hash değerleri hesaplanır. Aynı işlem, sadece iki hash değeri birleştirilerek Merkle ağaç kökü olan tek bir hash değeri oluşturuluncaya kadar devam eder.

Aslında blok zincir, hash tabanlı iki farklı veri yapısının akıllı bir kombinasyonudur. İlki, her bloğun zincirdeki önceki bloğa bağlanmasını sağlayan hash zinciridir. İkinci veri yapısı, bir bloğa dahil olan tüm işlemlerin hash değerlerinin oluşturduğu bir Merkle ağacıdır. Bu ağaç yapısı sayesinde, bloktaki tüm işlemlerin bir özeti kolay bir şekilde elde edilir. Merkle ağacında işlem bilgileri yer almaz sadece hash değerleri bulunur. Bir işlemin herhangi bir blok içerisinde olup olmadığını anlamak için tüm bloğun baştan sona incelenmesi yerine, yalnızca ağaçtaki bazı katmanlardaki haşh değerlerini kullanmak yeterli olacaktır. Bu şekilde, blokların hızlı doğrulanabilmesi sağlanmış olur.

Tablo 5'te yer alan blok başlık alanlarıdan biri olan nBits alanı, 32 bit uzunluğunda bir alan sağlar. Bu sebeple, 256 bit uzunluğundaki hedef blok hash değeri sıkıştırılarak saklanır. İlgili bloğa ait olacak madenciler tarafından hesaplanan hash değeri, nBits alanında tanımlanan hedef blok hash değerinden küçük veya eşit olmalıdır. Madencilik zorluk seviyesini belirlemek için kullanılan hedef hash değeri yükseltilerek ya da düşürülerek madencilik zorluk seviyesi ayarlanır.

32 bite sıkıştırılmış nBits değerinden 256 bitlik hedef hash değerinin elde edilmesi için şu yöntem uygulanır. 32 bitlik nBits değeri, heksadesimal olarak 8 basamaklı $0xh_0h_1h_2h_3h_4h_5h_6h_7$ sayısı şeklinde ifade edildiğinde, $0xh_2h_3h_4h_5h_6h_7$ değerleri katsayı, $0xh_0h_1$ değerleri ise üs olarak alınır.

$$\text{Hedef hash}=\text{Katsayı}*256^{(\text{üs}-3)}=0xh_2h_3h_4h_5h_6h_7*256^{(0xh_0h_1-3)}=0xh_2h_3h_4h_5h_6h_7*2^{8*(0xh_0h_1-3)}$$

formülasyonu uygulanarak 256 bit uzunluğundaki hedef blok hash değeri elde edilir.

Blok başlığındaki yer alan Nonce değeri, aynı veri için farklı hash özet değerleri oluşturmak için rastgele oluşturulur ve verinin başına eklenir. Her hesaplama öncesi yeniden oluşturulan Nonce değeri madencilik yapan blok zincir uygulamalarında karmaşık hash fonksiyonu değerlerini çözebilmek için kullanılır. Genellikle "0" değeri ile başlayıp her hash hesaplamasında arttırılır.

Bloklar kendilerinden önce gelen bloğa ait hash özet değerini kendi blok başlıklarında saklayarak bir önceki bloğa bağlanmış olurlar. Bu şekilde tüm bloklar birbirine bağlanarak bir blok zinciri oluşturulur. Daha önce yayınlanan bir blok içeriğinde herhangi bir değişiklik yapılırsa, bloğun farklı bir hash değeri olacaktır. Bu da, önceki bloğun hash değerini

içerdikleri için sonraki tüm blokların da farklı hash özetlerine sahip olmasına neden olur. Bu şekilde, değiştirilmiş blokların kolayca tespit edilmesi ve reddedilmesi mümkün hale gelir.

Blok gövdesi ise işlem sayacı ve blokta kayıtlı olan işlemlerden oluşur. Blokta yer alacak maksimum işlem sayısı, blok ve işlem büyüklüklerine bağlıdır.

1.1.7. Blok Zincir İşleyişi

Blok zincir ağındaki katılımcı düğümler fiziksel bir cihazdan ziyade mantıksal bir varlığı, bir başka deyişle bir blok zinciri kullanıcısının kimliğini ifade eder. Örneğin, birden çok düğüm aynı fiziksel makinede barındırılabilir. Katılımcı düğümler genel olarak üç farklı grupta toplanırlar: tam katılımcı düğüm (full node), yayınlama yapan düğüm (publishing node) ve hafif katılımcı düğümler (lightweight node).

Tam katılımcı düğümler, blok zincirinin tüm geçmişine sahiptir, ancak ağa yeni bloklar yayınlayamazlar. Yayınlama yapan düğümler de aslında bir çeşit tam katılımcı düğümdür, ancak bu düğümler yeni blok yayınlarlar. Hafif düğümler ise yeni blok öneremezler, blok zincir geçmişini kendi bilgisayarlarına indiremezler, zincir geçmişi için tam düğümlere güvenirler. Kendi işlemlerini tam katılımcı düğümlere iletmek zorundadırlar.

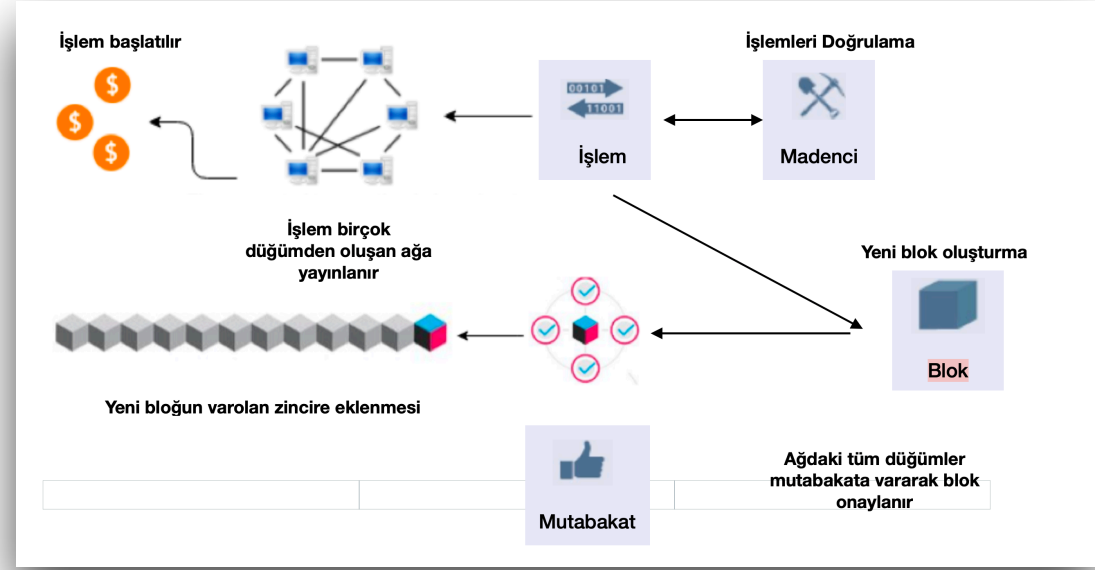
Tablo 6. Blok Zincir İşleyiş Adımları

BİLEŞEN	AMAÇ
Blok Önerisi	Tüm geçerli işlemleri toplamak ve yeni bloklar oluşturmak
Blok Doğrulama	Ağda yayılmadan önce blokların bağımsız kontrolünü yapmak
Blok Yayınlamak	Ağ genelinde yeni blokları ve blok işlemlerini yaymak.
Blok Sonlandırma	Doğrulanmış blokların kabulü konusunda anlaşmaya varmak
Ücretler ve Ödül	Dürüst katılım desteklemek için madeni para yaratmak

Blok zincir ağında işlemler ve işlem blokları gossip yönlendirme protokolü kullanılarak ağdaki tüm katılımcı düğümlere aktarılır. Düğümler aldıkları ve doğruladıkları yeni bir bloğu komşu düğümlere tanıtır ve zincirlerinde bu bloğa sahip olmayan düğümlere sektirerek iletirler. Ağdaki her düğüm bu bloğa sahip olana kadar sektirme işlemi devam eder.

Blok zincir işleyişi Şekil 4'te resmedilmiştir. Blok zincir ağındaki bir katılımcı düğüm açık anahtar şifreleme ile dijital imzasını kullanarak bir işlem başlatır ve blok zincir ağına sunar. Blok zincir ağında katılımcı düğümler arasında güveni oluşturmak için açık anahtar şifreleme yöntemi kullanılır. Blok zincirde işlem, Monrat ve diğ. (2019) tarafından elektronik varlıkların ağdaki katılımcılar arasında aktarımına ait verileri saklayan bir yapı olarak tanımlanmıştır. İşlemi başlatmak için masaüstü ve akıllı telefon uygulamaları, dijital cüzdanlar, web servisi gibi yazılımlar kullanılır. İlgili yazılım işlemi blok zincir ağı içindeki hem yayınlama yapan hem de yapmayan tam düğüm veya düğümlere gönderir. İşlem

onaylanmamış bir işlem havuzunda saklanır ve Gossip yönlendirme protokolü ile ağda yayılır, ancak bu tek başına işlemi blok zincirine eklemek için yeterli değildir. Birçok blok zinciri uygulaması için, bekleyen bir işlem düğümlere dağıtıldıktan sonra, bir yayınlama düğümü tarafından blok zincirine eklenene kadar bir kuyrukta beklemelidir.



Şekil 4. Blok Zincir İşleyişi

İşlemlerin önceden belirlenmiş bazı kriterlere göre seçilmesi ve doğrulanması gerekir. Düğümler varlık aktarımını yapmak isteyen ve bir işlem başlatan katılımcının kimliğini, yeterli bakiyesinin olup olmadığını veya çift harcama yaparak sistemi kandırmaya çalışıp çalışmadığını kontrol ederek işlemleri doğrulamak için çalışır. Blok zincir işlemi başlatan katılımcının kimliğini doğrulamak için açık ve özel anahtarları kullanan dijital imzaları kullanır. Bakiye kontrolü geçmişteki her başarılı işlem hakkında bilgi tutan deftere bakılarak gerçekleştirilir. Çift harcama, iki veya daha fazla farklı işlem için aynı girdi miktarını kullanmak anlamına gelir.

İşlem madenciler tarafından doğrulanıp onaylandıktan sonra bir bloğa dahil edilir. Blokların doğrulanması için madencilik yaparak hesaplama güçlerini kullanan katılımcı düğümlere madenci denir. Madenci düğümlerin ağda belirlenmiş zorluk seviyesine uygun bir hesaplama bulmacasını çözmesi ve bir bloğu yayınlamak için yeterli miktarda hesaplama kaynağı harcaması gerekir. İlk önce bulmacayı çözebilen madenci kazanır ve yeni bir blok oluşturma fırsatı elde eder. Madenci düğüm başarılı şekilde yeni bir blok oluşturduktan sonra bu düğüme küçük bir teşvik verilir. Teşviğin amacı blok zincir ağına dürüst katılımı destelemektir. Sonrasında, merkezi olmayan blok zincir ağındaki tüm katılımcılar kullanılan mutabakat mekanizması kurallarına göre fikir birliğine vararak yeni bloğu doğrular. Doğrulanan blok mevcut zincire ve her bir katılımcının kendi bilgisayarlarında kayıtlı yalnızca blok eklenmesine müsade eden açık defterin kopyasına eklenir. Bu noktada, blokta yer alan işlemler de ikinci defa onaylanmış olur. Bloğa dahil edilmemiş bazı bekleyen geçerli işlemler olabilir, ancak bu bir sorun değildir. Bir işlem bir bloğa girmediyse bekleyebilir ve bir sonraki

bloğa geçebilir. Monrat ve diğ. (2019) çalışmasında belirtildiği üzere zincire her yeni blok eklendiğinde işlem her seferinde yeniden teyit edilmiş olur.

Her yeni blok kendisinden önce gelen bloğa ait blok başlık hash değerini kendi blok başlığında saklar. Bu şekilde, her blok önceki bloğa hash değeri üzerinden bağlanarak blok zinciri oluşturur. Daha önce yayınlanan ve zincire eklenen bir blok değiştirilirse, farklı bir hash değere sahip olur ve bu bloktan sonra gelen tüm blokların da hash değerlerinin farklılaşmasına neden olur. Bu durum, değiştirilmiş blokların kolayca tespit edilerek reddedilmesini mümkün kılar. Yaga ve diğ. (2019) çalışmasında, bazı blok zinciri uygulamalarında belirli kontrol noktaları oluşturularak blok zincir içindeki eski blokların kilitlendiğini ve bu şekilde geçmiş blokların değiştirilmesinin engellendiğini ifade etmiştir.

Bazı blok zincir ağlarında genellikle coğrafi konum farklılıklarından dolayı iki veya daha fazla madenci düğüm bulmaca çözüm hash değerini aynı anda bulabilir. Böylece, birbirinden farklı birden fazla blok oluşur ve ağa duyurulur. Sonuç olarak, blok zincir ağında farklı zincirler ortaya çıkar. Aynı anda birden fazla blok zincire eklenemeyeceği için yalnızca bir blok zincire eklenerek yola devam eder. Zincire eklenemeyen bloğa "Yetim Blok" (Orphan Block) adı verilir. Örneğin, Bitcoin kripto para uygulamasında yetim blok olasılığı %1,69 ile %1,74 arasındadır. Bu bloklardan hangisinin zincire ekleneceği ilk aşamada belli değildir. Çoğu blok zinciri ağı bir sonraki blok yayınlanana kadar bekler ve hangi zincir daha çabuk uzarsa o zinciri kabul eder ve kullanır. Bir işlem içinde yer aldığı bloktan sonra birkaç ek blok oluşturulana kadar genellikle onaylanmış olarak kabul edilmez. Yayınlanmış bir bloğun üzerine ne kadar çok blok inşa edilirse, ilgili bloğun ve içerdiği işlemlerin değiştirilme olasılığı o kadar azalacaktır.

Aynı blok zincir ağında bu şekilde farklı zincirlerin oluşması durumuna çatallanma (forking) adı verilir. Blok zincirler, özellikle çatallara sahip olma konusunda oldukça hassastır. Farklı madenci düğümler tarafından aynı anda farklı bloklar keşfedildiğinde meydana gelen geçici çatallanmaların yanısıra, farklı defter kopyasına sahip olan düğümlerin oluşturduğu geçici çatallanmalar da mevcuttur. Düğümler bazen blok zincirin son bloğu konusunda farklı düşünebilirler, bazı düğümlerin sahip oldukları defter kopyasında blok zincirin son bloğu olarak farklı bir blok kaydedilmiş olabilir. Çalışmalar (Carlos, 2018) bu farklılaşmanın oluşturduğu geçici çatallanmanın bir saldırganın blok zincir defterini bozmaya çalışmasından ya da sadece ağda meydana gelen gecikmelerden kaynaklandığını ortaya koymuştur.

Geçici çatallanmalar defterdeki bir farklılaşmadır ve kalıcı değildir. Herhangi bir yazılım değişikliğinden kaynaklanmaz. Ancak, gerçek çatallanmalar bir blok zinciri ağının protokolünde ve veri yapılarında yapılan değişiklikler sonucu farklı iki zincirin oluşmasıdır. Çatalanmaya kadar zincirler aynı blok geçmişine sahiptir fakat çatallanma sonrasında yeni çalışma kuralları uyguladıkları için farklılaşırlar. Çatallanmalar iki kategoriye ayrılır.

1.1.7.1. Yumuşak Çatallanma (Soft Forking)

Blok zincirin kullanımını geliştirmek ve iyileştirmek amacıyla yapılan teknik değişikliklerdir. Düğümler yapılan iyileştirme ve geliştirmeleri kullanabilmek için yazılımlarını güncellemek

zorundadırlar. Eski yazılımlar ile blok zinciri kullanmaya devam etmek isteyen düğümler yeni özellikleri kullanamazlar. 2014 yılında Bitcoin kripto para biriminde yapılan BIP65 yumuşak çatalı ile üçüncü tarafa yatırılan fonların emanetini (escrow) ve paranın gelecekte belirlenmiş bir zamanda harcanmasını desteklemek için NOP2 operasyon kodunun CHECKLOCKTIMEVERIFY parametresi olarak tanımlanması sağlanmıştır. Bir blok zincir blok boyutunu 1.0 MB'den 0.5 MB'ye azaltmak yumuşak çatalın bir başka örneği olabilir. Güncellenmiş yazılımı kullanan düğümler, blok boyutlarını yeni değere (0.5MB) göre ayarlayarak işlemlerine devam ederler. Yazılımlarını güncellemeden kullanan düğümler yeni yazılım tarafından oluşturulan 0.5MB'den küçük blokları kabul edeceklerdir. Ancak, yazılımını güncellemeyen bir düğüm boyutu 0,5 MB'den büyük bir blok oluşturursa, güncellenmiş düğümler bunları geçersiz sayarak red edecektir.

1.1.7.2. Sert Çatallanma (Hard Forking)

Sert çatallanmalar blok zincirin kurallarını değiştiren protokol güncellemeleridir. Düğümler yeni blok zinciri kullanabilmek için yeni protokol kurallarının geçerli olduğu yazılımları kullanmak zorundadırlar, aksi halde yeni zincire bağlanamazlar ve işlemlerine devam edemezler. Eski protokol kurallarıyla çalışmaya devam edilmesi yeni bir blok zincir ve hatta yeni bir kripto paranın ortaya çıkmasına neden olabilir. Çoğu sert çatallanma planlı olmakla birlikte, yazılım hataları sonucu ortaya çıkabilen sert çatallanmalar da olabilir. Bitcoin Cash, sert çatallanmaya güzel bir örnek teşkil eder. Bitcoin blok zincir uygulamasında yazılım geliştiricilerin fikir ayrılığı sonucu meydana gelen çatallanma yeni bir kripto para olan Bitcoin Cash blok zincirini oluşturmuştur. Akıllı kontratlar üzerinden yapılan bir saldırı sonrası Ethereum blok zincirinde gerçekleşen çatallanma sonucunda oluşan yeni Ethereum blok zinciri ve Ethereum Classic olarak adlandırılmaya başlanan eski blok zincir sert çatallanmanın öne çıkan bir diğer örneğidir.

1.1.8. Mutabakat

Her katılımcı düğüm, blok zincir ağını dinler ve duyduğu ancak henüz blok zincirine dahil edilmemiş işlemleri bir bekleyen işlem havuzuna kaydeder. Bu işlemler için henüz bir mutabakat veya fikir birliği gerçekleşmemiştir ve pratikte her düğüm bekleyen işlem havuzunun farklı bir sürümüne sahip olabilir. Ayrıca, katılımcı düğümler arasındaki ağın mükemmel olmaması nedeniyle bazı düğümler diğer düğümlerin duymadığı bir işlemi duyabilir ve bekleyen işlem havuzlarına ekleyebilirler.

Bekleyen işlemlerin yer aldığı blokların zincire eklenmesinde hangi katılımcı düğümün bir sonraki bloğu yayınlayacağı önemli bir karar aşamasıdır. Bu aşamada mutabakat protokolleri uygulanır. Özellikle kripto para uygulamalarında, genellikle kripto para ödülü ya da işlem ücreti teşviklerini kazanmak için bir sonraki bloğu yayınlamak isteyen birçok yayınlama düğümü arasında tam anlamıyla bir yarış vardır. Blok yayınlama hakkını kazanan blok fikir birliğine varılarak belirlenir. Ayrıca, merkezi bir otorite olmaması nedeniyle, yayınlanan blokların yer aldığı defterlerin tüm katılımcı düğümlerde aynı ve tutarlı olmasını sağlamak için de mutabakat protokollerine ihtiyaç vardır.

Blok zinciri genel olarak birbirine tam olarak güvenmeyen katılımcı düğümlerin Bizans tarzında davranabileceğini varsayar (Carlos, 2018). Aslında, güvenilmez düğümler arasında mutabakata nasıl varılacağı, Bizans Generalleri (BG) Probleminin bir dönüşümüdür. Bir şehri kuşatan Bizans ordusuna komuta eden bir grup generalin sadece bir kısmının şehre saldırması halinde saldırı başarısız olacaktır. Generallerin saldırı yapılıp yapılamayacağı konusunda bir anlaşmaya varmak için iletişim kurması gerekir. Ancak generaller arasında hainler olabilir ve farklı generallere farklı kararlar gönderebilir. Bu güvensiz ortamda bir mutabakata nasıl varılacağı önemli bir sorundur. Blok zincir ağına yeni katılımcıların eklenmesi ile ağın genişlemesi durumunda mutabakata varmak blok zincirini zorlayacaktır.

Öncelikle, mutabakata ya da fikir birliği varmak genel olarak çözülmesi zor bir sorundur çünkü katılımcı düğümler çökebilir veya tamamen kötü niyetli olabilir veya ağın kendisi oldukça sorunlu olabilir. Blok zincir katılımcılar arası bir sistemdir ve tüm katılımcı düğüm çiftleri birbirine bağlı değildir. Narayanan ve diğ. (2016) tarafından yapılan araştırmada, zayıf İnternet bağlantısı nedeniyle ağda hatalar ve gecikmeler olabileceği, bu nedenle tüm düğümlerin katılması gereken bir mutabakat protokolü çalıştırmanın gerçekten zor olduğu ifade edilmiştir.

Bir blok zinciri mutabakat protokolünün temel amacı, katılan tüm düğümlerin ortak bir ağ işlem geçmişi üzerinde anlaşmasını sağlamaktır. Xiao ve diğ. (2020) çalışmasında, mutabakat protokolünün belli gereksinimleri mevcut olduğu belirtilmiştir.

- Sonlandırma: Yeni bir işlem, bir blok içinde yer almak koşuluyla blok zincir tarafından kabul edilmelidir, aksi durumda sistem tarafından reddedilir.
- Anlaşma: Her yeni işlem ve bu işlemin yer aldığı blok, blok zincir ağındaki güvenilir düğümlerin tamamı tarafından kabul edilmeli veya iptal edilmelidir. Kabul edilen bir bloğa her katılımcı düğüm tarafından aynı sıra numarası atanmalıdır.
- Geçerlilik: Her katılımcı düğüm aynı geçerli işlemi ve bloğu alırsa, bu işlem ve blok blok zincire kabul edilmelidir.
- Bütünlük: Her katılımcı düğümün sahip olduğu defterde, kabul edilen tüm işlemler çift harcamayı ve kurcalamayı önlemek içim birbiriyle tutarlı olmalıdır. Kabul edilen tüm bloklar doğru şekilde oluşturulmalı ve blok zincire kronolojik sıraya göre kaydedilmelidir.

Blok mutabakat işlemi, hepsi aynı kuralları izleyen binlerce bağımsız düğümün eşzamanlı olmayan etkileşimlerinin sonucu ortaya çıkan merkeziyetsiz bir mekanizmadır (Carlos, 2018). Wang ve diğ. (2018) tarafından yapılan çalışmada incelenen mutabakat yöntemleri Tablo 7'de karşılaştırılmış ve sonrasında her biri detayları ile açıklanmıştır.

Tablo 7: Karşılaştırmalı Mutabakat Yöntemleri

1.1.8.1. İş İspatı (PoW-Proof of Work) Mutabakatı

Özellik	MUTABAKAT YÖNTEMLERİ					
	İş İspatı	Hisse İspatı	Delege Hisse İspatı	Bizans Hata Toleransı	Tendermint	Zaman İspatı
Ağ Türü	İzinsiz	İzinli/ İzinsiz	İzinsiz	İzinli	İzinli	İzinli/İzinsiz
Enerji Tasarrufu	Yok	Kısmen	Kısmen	Var	Var	Var
Saldırı Toleransı	≤ %25 (işlem gücü)	< %51 (hisse payı)	< %51 (doğrulayıcı)	≤ %33.3 (oylama gücü)	< %33.3	-
Enerji tüketimi	Yüksek	Düşük	Çok düşük	Çok düşük	Çok düşük	Çok düşük
Ölçeklenebilirlik	Yüksek	Yüksek	Yüksek	Düşük	Yüksek	Yüksek
İşlem ücreti	Düşük	Yüksek	Orta	Yüksek	Yüksek	Orta
Performans (İşlem sayısı/ saniye)	< 20	< 20	< 500	< 1000	< 10000	
Mutabakat Süresi	Yüksek	Yüksek	Orta	Düşük	Düşük	Düşük
Blok Oluşuma Hızı	Yavaş	Hızlı	Değişken	Hızlı	Hızlı	
Örnek	Bitcoin, Etherium	Peercoin, Nextcoin	Bitshares	Hyperledger Fabric	Tendermint	Coindesk, Hyperledger

İlk defa Nakamoto makalesinde tanımlanan mutabakat protokolüdür. Bitcoin, Ethereum ve Litecoin gibi birçok kripto para birimi sistemi bu mutabakat algoritmasını kullanır. İş ispatı yeni bir bloğun blok zincirine eklenmesi için zor bir kriptografik hash fonksiyonun çözülmesi prensibine dayanır ve önemli bir hesaplama çabası harcandığının kanıtı anlamına gelir. Bulmacayı çözen katılımcı, blok zincirine yeni işlemleri ve bloğu kaydetme hakkını kazanmış olur. Dürüst katılımcıları teşvik etmek için bir ödül sistemi uygulanır ve bu sayede blok zincir ağ güvenliğinin sağlanması hedeflenir (Carlos, 2018).

Genel olarak kullanılan bulmaca yöntemi, bir blok başlığı için hash özet değerinin hesaplanmasıdır. İş ispatı mutabakatına göre, blok başlığı için hesaplanan hash özet değeri hedef hash değerini geçmemelidir. Bu kritere uygun hash değerinin üretilmesi için blok başlığında bulunan nonce değeri rastgele şekilde değiştirilir ve blok başlığı pek çok defa hash fonksiyonuna girdi olarak aktarılır. Hedefi karşılayan hash'i üreten bir nonce değeri başarıyla bulduktan sonra, blok ağa yayınlanır ve blok zincirine eklenir. Ağdaki diğer düğümlerin bakış açısından, eğer birisi doğru nonce ile bir bloğu birleştirerek mevcut hedefin altında bir hash değeri elde etmeyi başarırsa, bu yoğun bir hesaplama işinin yapıldığının kanıtıdır.

Bir yayınlama düğümü bu hesaplama işini gerçekleştirdiğinde, hesaplama yaptığı bloğu nonce değeri ile birlikte blok zinciri ağındaki tam düğümlere gönderir. Bloğu alan tam düğümler, yeni bloğu doğrulamasını yapmak için bir kez hash fonksiyonunu çağırır ve nonce ile kontrolünü yaparak bloğun bulmacayı çözüp çözmediğini doğrular. Blok doğrulandığı durumda, katılımcı düğüm bloğu kendi blok zinciri kopyasına ekler ve komşu düğümlerine gönderir. Bu şekilde yeni blok ağ boyunca iletilerek katılımcı düğümlerin tamamına dağıtılır.

Hedef hash değeri arttırılıp azaltılarak blok yayınlama zorluğu ve sıklığı değiştirilebilir. Örneğin, iş ispatı mutabakatını kullanan Bitcoin uygulamasında her on dakikada bir blok yayınlanacak şekilde bir ayarlama yapılmıştır. Ayarlama bulmacanın zorluk seviyesine göre yapılır ve Hash değerinin kaç adet sıfır ile başlaması gerektiği iş ispatı algoritmasının arkasındaki bulmacanın zorluk seviyesini gösterir. Baştaki sıfırların sayısını arttırmak bulmacanın zorluğunu artırırken sıfırların sayısını azaltmak zorluk seviyesini düşürür. Bitcoin para sisteminde, zorluk seviyesi 2016 blokta bir ayarlanır.

Hesaplamanın yoğun olduğu iş ispatı algoritması Sybil saldırıları azaltmak için tasarlanmıştır. Sybil saldırılarında, çok sayıda sahte kimlik yaratan saldırganlar ağda büyük oranda çoğunluk sağlayarak ağın işleyişini altüst etmek amacı güderler. Örneğin, Bitcoin gibi izinsiz katılım ve gerçek olmayan kimlik kullanımı sunan blok zincir ağlarında Sybil saldırganları çok az bir çabayla sahte kimlikler veya hesaplar edinebilir. İş ispatı algoritmasını uygulamak yüksek bir hesaplama gücü gerektirir. Kullanılan donanım gücünü arttırmak hesaplama gücünü arttırır ancak ciddi bir maliyete yol açar. Ayrıca, blokların başarılı bir şekilde yayınlanmasını ödüllendirmek için bir teşvik sistemi kullanılır. Xiao ve diğ. (2020) çalışmasında, yüksek yatırım maliyetinin caydırıcılığının yanısıra katılımcılara sunulan blok ödüller ve ağa enjekte edilen yeni madeni paralar sayesine blok zincir ağında dürüst bir katılımın teşvik edildiğini belirtmişlerdir.

İş ispatı algoritması, düşük işlem kapasitesinden ötürü eleştirilmektedir. Ardışık blok yayınlamaları arasında atanan ortalama on dakikalık zaman aralığının azaltılması ve bir blok uzunluğunun arttırılması işlem kapasitesini arttırabilir, ancak blok zincir yapısının bozulmasına, blok iletim gecikmelerine, blok yetersiz yayılımına ve güvenlik sorunlarına sebep olabilir. Özellikle artan kripto para popülerliği hash algoritmalarının zorluk dereceklerinin yükseltilmesine ve dolayısıyla daha büyük bir enerji kullanımına yol açmaktadır. Bu iş ispatı mutabakatının bir diğer sakıncasıdır ve enerjinin verimsiz bir şekilde kullanılması protokolün önüne çıkan önemli bir engeldir. Ayrıca, iş ispatının kullanıldığı sistemlerde iyi bir gelir elde etmek maliyetlidir, büyük bir donanım ve sermaye gücü gerektirir. Xiao ve diğ. (2020) çalışmalarında, genellikle büyük kuruluşlar tarafından desteklenen is ispatı sistemlerinde küçük bireysel katılımcıların oyun dışı kaldığını ya da alternatif olarak daha istikrarlı bir gelir için daha büyük madenci havuzlarına katıldığını belirtmişlerdir. Ağı ele geçirmek için hesaplama gücünün çoğunu ele geçirme zorunluluğu, sağladığı güvenlik sebebiyle iş ispatı protokolünün en büyük avantajlarından biri olarak görülse de, bu durum tekelleşmeye yol açabilir ve protokolün en büyük dezavantajlarından biri haline gelebilir. Çünkü en büyük paya sahip işlemciler, ağda en yüksek oylama gücüne sahip olabilmektedir. Örneğin, Bitcoin ağının en önemli özelliği olarak karşımıza çıkan

merkeziyetsizlik ilkesi, güçlü olanın rahatlıkla otorite kazanabilmesi riski nedeniyle ağ ilkeleriyle çelişki yaratmaktadır.

1.1.8.2. Hisse İspatı (PoS-Proof of Stake) Mutabakatı

Hisse ispatı mutabakat yönteminde matematik bulmacası çözmek için çok büyük miktarda bilgisayar kaynağının israf edilmesine gerek yoktur. Hisse genellikle kripto para uygulamalarında blok zincir ağı kullanıcısının sisteme yatırdığı ve bloke edilerek harcanmasına izin verilmeyen bir kripto para miktarıdır. Bir başka deyişle, katılımcının ağ içerisindeki teminatıdır denilebilir. Blok oluşturma sürecine katılmak için sadece sistemde yeterli hisseye sahip olmak gerekir. Bir bloğu doğrulama fırsatı elde etme şansı tamamen katılımcı düğümün hissesine bağlıdır. Blok onaylandığında, bloğu doğrulayan hisse sahibi katılımcı blokta yer alan işlemlerden işlem ücreti tahsil eder.

Hisse ispatı algoritması iş ispatı mutabakatına göre düşük bir enerji kullanımına ihtiyaç duyar ve kısa sürede gerçekleşen işlemler sayesinde yüksek verim sunar. Bu yöntem, katılımcı düğümler arasındaki rekabetin azaltılmasına olanak tanıyabilir. Ayrıca, piyasa değeri yüksek bir kripto para biriminde hisselerin yarısından fazlasına sahip olmak hem zor hem de pahalı bir işlemdir. Böylesine büyük bir yatırım yapan katılımcının ağa saldırması akıllıca bir durum değildir. Ancak, çok hisse sahibi zengin katılımcı düğümler ağda daha baskın hale gelebilir, bu da adil olmayan dağıtım veya merkezileştirmeye yol açabilir. Blok doğrulama maliyeti ve çabası iş ispatına kıyasla çok daha düşük olduğu için hisse ispatı mutabakatının kötü niyetli saldırılara daha yatkın olduğu Monrat ve diğ. (2019) çalışmasında ifade edilmiştir.

Bir sonraki bloğu oluşturma şansı elde edecek katılımcı düğümün tespit edilmesinde, hisse sahibi katılımcının seçilme yöntemi uygulamalar arasında farklılıklar göstermektedir. Daha eski hisselerin blok doğrulama için daha fazla önceliğe sahip olduğu hisse sahiplik süresine dayalı bir hisse ispatı sistemi (Peercoin), rastgelelik kullanan ve hissenin büyüklüğü ile birlikte en düşük hash değerini arayan bir hisse ispatı sistemi (Blackcoin) bu uygulamalardan sadece ikisidir. Ayrıca, blok doğrulamak için hisse oranlarından başka düğümlerin yaptıkları işlem sayısı (proof of importance), sahip oldukları sabit disk alanı (proof of capacity) gibi özellikleri blok doğrulayacı katılımcıyı seçme kriteri olarak dikkate alan hisse ispat yöntemleri de bulunmaktadır.

Hisse ispatı mutabakatında, blokların yayınlanması için gerekli olan maliyet blok zincirinde herhangi bir noktada ister yanlışlıkla isterse kötü niyetli bir şekilde birden fazla rakip blok zinciri mevcut olursa, yüksek hisseye sahip bir katılımcı bu tür rakip zincirlerin her birinde hareket edebilir ve bunu ödül kazanma şansını artırmanın bir yolu olarak kullanabilir. Büyümesi gereken blok zincirle birlikte başka blok zincirlerin de uzamaya devam etmesi durumu söz konusu olabilir (Nothing at Stake problemi).

Hisse ispatı algoritması dört ayrı kategoride incelenebilir. Zincir hisse ispatı, komite hisse ispatı, Bizans Hata Toleransı hisse ispatı ve delege hisse ispatı algoritmalarıdır. Zincir hisse ispatı daha az hash hesaplama gücüne ihtiyaç gösterir. Komite hisse ispatında hisse sahiplerinin her blok için oluşturduğu gruplar blok yaratma ve yayınlama hakkına sahip olur.

Bu iki algoritma iş ispatı algoritmasının ana mutabakat çerçevesini oluşturan hesaplama olasılığı esasına dayanır. Bizans Hata Toleransı ise diğer hisse ispatı algoritmalarına göre hızlı ve kararlı bir yapı sunar. Tendermint protokolü bu algoritmalar arasında en popüler olanlarından biridir.

1.1.8.3. Delege Hisse İspatı (DPos - Delegated Proof of StakeDPos) Mutabakatı

Hisse İspatı protokolünün bir türevi olan Delege Hisse İspatı yönteminde ağ düğümleri oy kullanarak delege olarak adlandırılan blok yayınlama hakkına sahip olacak katılımcı düğümü seçerler. Katılımcı düğümlerin oy kullanma gücü hisse sayılarına bağlıdır, hisselerinin büyüklüğü ölçüsünde oyların ağırlığı da artar. En çok oyu alan düğümler yayınlama düğümleri haline gelir ve blokları doğrulayıp yayınlayabilir. Blok zincir ağı katılımcıları için yayınlama düğüm seçimi oylamaları süreklidir. Bu nedenle, hem yayınlama düğümü olmak hem de bu statüyü sürdürmek önemlidir ve rekabete açıktır. Yayın düğümü statüsünü kaybetmek istemeyen düğümler kötü niyetli davranmamaları için ekonomik bir şekilde teşvik edilir. Blok doğrulamanın az sayıda delege tarafından yapılması, daha hızlı blok oluşturma ve işlemleri hızlı bir şekilde onaylamayı mümkün kılar. Ancak, bu hisse ispatı mekanizması merkezileşme eğilimi gösterebilir. Yüksek riskli katılımcılar kendileri oylayabilir ve onaylayıcı olmak için başkalarını oy kullanmaları için manipüle edebilirler.

1.1.8.4. Pratik Bizans Hata Toleransı Mutabakatı

Blok zincir ağını oluşturan düğümler arasında kötü niyetli düğümler olduğu varsayılır. Bu sebeple, mutabakat algoritması yalnızca ağdaki dürüst düğümlerin katılacağı şekilde tasarlanır. Ardışık bir düzene sahip olan düğümlerin biri diğer düğümlere liderlik yapar, diğer düğümler ise yedek düğüm olarak sistemde varolur. Lider düğüme bir talep geldiğinde, lider bu talebi yedek düğümlere yayınlar. Lider ve yedek düğümler talebi işleme alıp sorgularlar. Talep sahibi düğüm lider düğüm ve yedek düğümleri sorgulamaları sonucu hakkında bilgilendirilir. Talep sahibi, lider ve yedek düğümlerden gelen sonuçların aynı olmasını bekler. Katılımcıların çoğunluğu blok için onay verirse, sistemdeki herkes bu bloğun doğruluğunu kabul eder. Her turda yeni bir blok belirlenir ve bazı kurallara göre yeni bir lider düğüm seçilir.

Tüm dürüst düğümlerin amacı, çoğunluğun görüşüne dayanarak sistemin durumu üzerinde anlaşmaktır**.** Mutabakat algoritmasının düzgün çalışması için dürüst düğümlerin anlık sayısının tüm ağ düğümlerinin üçte birinden (1/3) büyük olması gerekir. Sistemdeki düğüm sayısı arttıkça tüm düğümlerin üçte birinin kötü niyetli olma olasılığı azalacaktır.

1.1.8.5. Tendermint

Temeli Bizans Hata Toleransı mutabakat algoritmasına dayanan Tendermint yönteminde, her turda onaylanacak blok için bir doğrulayıcı ve yayınlayıcı düğüm seçimi yapılır. Bu nedenle, seçim için tüm düğümlerin bilinmesi gerekir. Blok yayınlayıcı düğümü seçim işlemi üç adıma bölünebilir: ön oy (prevote), ön onaylama (precommit) ve onaylama (commit). İlk adımda, bloğu yayınlamak isteyen düğüm, önerilen blok için ağ boyunca seçim öncesi mesajı yayınlar

ve eğer iletisi için ⅔‘ten fazla ön oy alırsa, bu blok için bir ön onay mesajı yayınlar. Düğüm ⅔‘ten fazla ön onay kabul mesajı alırsa onaylama adımına girer. Düğüm, bloğu doğrular ve son adımda bu blok için bir kayıt yayınlar. Düğüm, işlemlerin ⅔'ünü alır ise bloğu kabul eder. Süreç PBFT’ye (Pratik Bizans Hata Toleransı Mutabakatı) oldukça benzer, ancak Tendermint düğümlerinin doğrulayıcı olmak için coinlerini kilitlemesi gerekir. Doğrulayıcı dürüst olmadığı tespit edildiğinde cezalandırılır.

Tendermint mutabakatında, “kazanan her şeyi alır” kuralı yerine eşit paylaşım tarzı bir teşvik mekanizması mevcuttur. Her blok için verilen ödül, blok önerisi veren düğüm, onay oylarını veren ve onay oyunu alan düğümler arasında dağıtılır. Bununla birlikte, onay oyları bir sonraki döngüden önce zamanında teslim edilmezse ağın güvenilirliği bozulabilir.

1.1.8.6. Round Robin Mutabakatı

Katılımcı düğümler sırayla blokları oluşturur ve yayınlar. Sırası gelen katılımcı sistem tarafından belirlenmiş süre içinde yayın yapamaz ise, blok yayın sırası bir sonraki düğüme geçer. Bu şekilde, blok yayını devam eder, ağ işleyişinde bir duraklama olmaz. Tek düğüm merkeziyetçiliğinden uzak, kriptografik hash algoritmalarının kullanılmadığı, enerji ihtiyacının düşük olduğu bir yöntemdir, düğümlerin birbirine güven duyması önemlidir. Yaga ve diğ. (2019), bu yöntemin kripto para birimleri tarafından kullanılan izinsiz blok zinciri ağları için çok uygun bir yöntem olmadığını ifade etmiştir. Bunun nedeni kötü niyetli düğümlerin yeni bloklar yayınlama olasılıklarını artırmak için sürekli olarak ek düğümler ekleyebilmeleridir.

1.1.8.7. Otorite/Kimlik İspatı (Proof of Authority/Identity) Mutabakatı

Gerçek kimliklerin kullanıldığı bu mutabakat yönteminde, katılımcılar ağ içerisindeki davranışlarına göre kazandıkları itibar derecelerine göre blok yayınlama hakkı kazanırlar. Yayınlama düğümü yeni bloklar yayınlamak için kimliğini ve itibarını riske atmış olur. Yayınlama düğümleri, ağ katılımcılarının kabul ettiği şekilde davranarak itibar kazanırlar ya da kabul edilmediği bir şekilde davranarak itibar kaybedebilir. İtibar derecesi ne kadar yüksekse, bir blok yayınlama olasılığı o kadar artar. Gerçek kimlik bilgileri ifşa edildiği için, ağ düğümleri arasında yüksek düzeyde bir güvene ihtiyaç duyulur. Bu sebeple, izinli blok zinciri ağları için uygun bir mutabakattır.

1.1.8.8. Zaman İspatı (Proof of Elapsed Time) Mutabakatı

Ağdaki her katılımcı düğümün yeni bir blok yayınlamak için rastgele seçilen bir süre boyunca beklemesi gerekir. Bekleme süresini ilk tamamlayan katılımcı düğüm yeni bloğu yayınlar. Blok zinciri ağındaki her düğüm kendilerine verilen rastgele bir bekleme süresi boyunca uyku moduna geçer. En kısa bekleme süresine sahip olan önce uyanır ve blok zincirine yeni bir blok yükleyerek gerekli bilgileri tüm ağa yayınlar. Aynı işlem daha sonra bir sonraki bloğun keşfi için tekrarlanır.

Düğümlere atanan zaman rastgele bir yöntemle belirlenmelidir. Bekleme süresi rastgele seçilmezse, dürüst olmayan bir düğüm sisteme hakim olmak için minimum süreyi bekleyebilir. Ayrıca, düğümlerin kendilerine verilen zamanı beklediğinden ve erken başlamadığından emin olmak gerekir.

1.1.9. Akıllı Sözleşmeler (Smart Contracts)

Akıllı sözleşmeler, blok zincirde katılımcılar arasında yapılan kira sözleşmesi, sigorta, kredi kullanımı gibi sözleşmeleri dijital ortamda uygulayan komut dosyalarıdır. Bir başka deyişle, akıllı sözleşmeler, blok zincir üzerinde belirli bir iş mantığına hizmet eden ve belirli koşullar gerçekleştiğinde otomatik olarak tetiklenen bilgisayar programlarıdır. İlişkisel veritabanı yönetim sistemlerindeki stored prosedürlerin benzeri olarak düşünülebilir.

Blok zincir ağında her akıllı sözleşmenin kendine ait özel bir adresi vardır. Akıllı bir sözleşme, bu adrese gönderilen bir mesaj ya da işlem tarafından tetiklenir. Tetiklenen akıllı sözleşme programı, ilgili işlem veya mesajda yer alan verilere göre ağdaki her katılımcı düğümde önceden belirlenmiş şekilde bağımsız ve otomatik olarak çalıştırılır. Bu noktadaki en önemli husus, akıllı sözleşmeyi çalıştıran tüm düğümlerin aynı sonuçları elde etmesi zorunluluğudur. Bu durum akıllı sözleşmenin kararlı yapısının bir göstergesidir. Eğer sözleşme kararlı olmaz ise, her düğümde çalıştırıldığında farklı sonuçlar elde edilir, böylece ağ fikir birliğine varamaz. Ek olarak, akıllı sözleşmeyi çalıştıran tüm düğümler çalışmadan sonra elde edilen yeni durum üzerinde de anlaşmalıdır. Akıllı sözleşmeler fikir birliğine varmak için parametre olarak aktarılan veriler dışındaki verileri kullanamaz. Akıllı sözleşme programının çalışması sonucunda elde edilen veriler blok zincire kaydedilir. Christidis ve Devetsikiotis (2016) tarafından akıllı sözleşmeler, her programlama dilinde kullanılan “if-then” ifadeleri gibi önceden belirlenen bazı koşulların karşılanması durumunda belirli eylemlerin uygulanmasından ibaret olan bir kod ve veri koleksiyonu olarak tanımlanmıştır.

Bir akıllı sözleşme, blok zincir üzerinde uygulandığı için sözleşme kapsamında yapılan işlemler ve program kodu katılımcı düğümlere dağıtılır ve yalnızca eklemenin yapıldığı değiştirilmesi çok zor olan defterlerde kayıt altına alınır. Ayrıca, sözleşme blok zincirde doğrulandığı için kurcalamalara karşı korumalıdır. Bu sayede, merkezi bir kontrol ve üçüncü taraf koordinasyonun olmadığı, güven eksikliği ile etkileşimde bulunan bireysel düğümler arasında etkili bir güvenlik ve dokunulmazlık sağlanmış olur. Akıllı bir program kodu olarak bir sözleşme, kendi kripto para birimlerine veya diğer dijital varlıklara sahip olabilir ve önceden tanımlanmış koşullar tetiklendiğinde bu varlıkların aktarımını yapabilir. Christidis ve Devetsikiotis (2016) çalışmalarında, bu sözleşmelerin blok zincirde genel amaçlı hesaplamalar yapma imkanı verse de ağı oluşturan katılımcılar arasında gerçekleşen aktarım ve etkileşimleri yönetmek için kullanıldıkları zaman daha etkin bir sonuç verdiğini ifade etmişlerdir.

Birçok blok zincir uygulamasında, yayınlama yapan düğümler yeni bir blok yayınlarken diğer düğümlerle eşzamanlı olarak akıllı sözleşme kodunu çalıştırır. Akıllı sözleşme kodunu çalıştırmayan, bunun yerine sözleşme kodunu çalıştıran düğümlerin sonuçlarını doğrulayan yayınlama düğümlerinin bulunduğu bazı blok zincir uygulamaları da bulunmaktadır. Her iki

uygulamada da akıllı sözleşme program kodunu çalıştıran katılımcı düğümün kod çalıştırma maliyetinin ödenmesi gerekir. Ödeme, akıllı sözleşme programına kodu çalıştırmak için talepte bulunan katılımcı tarafından yapılır. Kodun çalışma süresi için bir üst limit vardır ve limit aşılırsa çalışma durur, işlem iptal edilir. Bu tasarım sayesinde, kötü niyetli katılımcıların tüm kaynakları tüketmesinin önüne geçilerek "Hizmet Reddi Saldırısı" (Denial of Service) gerçekleştirmesi engellenmiş olur.

Akıllı sözleşmelerin blok zincir teknolojisi örnek uygulamaları arasında Ethereum'un akıllı sözleşmeleri ve Hyperledger Fabric'in zincir kodu önde gelir. Aslında, Etherum gelişmiş ve özelleştirilmiş akıllı sözleşmeleri destekleyen ilk açık blok zincir platformudur. Akıllı sözleşme programının çalıştırılması, Ethereum sanal makinesi (EVM-Ethereum Virtual Machine) adı verilen Turing sanal makinesinin yardımıyla gerçekleşir. Ethereum ağındaki her düğüm bir EVM uygulamasınındaki aynı komutları çalıştırır. Solidity3 ve Serpent 4 gibi birçok üst düzey programlama dilleri Ethereum akıllı sözleşmeleri yazmak için kullanılabilir ve sözleşme kodu EVM byte koduna göre derlenir ve çalıştırmak üzere tüm blok zincir ağına dağıtılır. Ethereum günümüzde akıllı sözleşmeler için en popüler geliştirme platformudur ve çeşitli merkeziyetsiz uygulamaları (DApps-DecentralizedApps) tasarlamak için kullanılır.

Akıllı sözleşmeler son yıllarda büyük ilerlemeler kaydetmiş olsa da hala birçok zorluklarla karşı karşıyadır. Güvenlik sorununa ek olarak, performans, mahremiyet ve yasal sorunlar bu zorluklar arasında yer alır. Ethereum blok zincirinin karşılaştığı en bilinen saldırı Haziran 2016'da gerçekleşmiştir. Ethereum blok zinciri tarafından güvence altına alınan, ticari ve kar amacı olmayan girişimleri organize etmek amacıyla Nisan 2016 tarihinde başlatılan Decentralized Autonomous Organization (DAO), merkezi bir hükümetten ya da geleneksel bir yönetim yapısından etkilenmeyen bir bilgisayar programı olarak kodlanmış kurallarla temsil edilen bir fonlama organizasyonudur. DAO program kuralları ve gerçekleşen para fonlama işlemleri bir blok zinciri üzerinde tutulur. Kesin yasal statüsü belirsiz olan bu organizasyonun yeni merkezi olmayan bir iş modeli sağlama hedefi vardı. Yazılım kodları açık olan DAO, "Tekrarlayan Çağırma" (Recursive Call) adı verilen ciddi bir program hatasından yararlanılarak saldırıya uğramıştır. Saldırgan, yaklaşık 50 milyon dolar değerindeki Ether para birimini DAO benzeri yeni bir blok zincirine aktarmıştır. Aktarılan fonları geri almak için Ethereum bir çatal uygulayarak DAO'daki orijinal fonları bir kurtarma adresine taşımış ve Ethereum'a değiştirebilme olanağı sunmuştur. Bununla birlikte, fonlarını kurtarma adresine taşımayı kabul etmeyenler eski blok zinciri kullanmaya devam etmişlerdir. Günümüzde Ethereum Classic olarak bilinen zincirin aslında yeni zinciri reddeden katılımcıların kullandığı orijinal Ethereum blok zinciri olduğu Wang ve diğ. (2019) tarafından yapılan çalışmada belirtilmiştir.

1.2. KRİPTO PARA BİRİMLERİ

Son yıllarda İnternet üzerinden yapılan alışverişler, Visa ve Mastercard gibi ödeme kartı ağlarına ek olarak, Paypal, Google Checkout ve WebMoney gibi e-cüzdanlar, eBillMe gibi otomatik ödeme sistemleri ve Moneygram gibi para transfer sistemlerini ortaya çıkarmıştır. Bu ödeme sistemlerinin tümü dolar ve türk lirası gibi mevcut para birimleri cinsinden işlem yapar. İşlemlerde ödeme yapan ve alacaklı kişinin kimliği açıkça belllidir. Merkezi ya da yarı

merkezi yönetilme özelliğine sahiptirler. Meiklejohn ve diğ. (2016) çalışmalarında, nakit ve mevcut ödeme yöntemlerinin bazı özellikleri birleştirilerek ödeyeni veya alacaklıyı açıkça tanımlamayan, kriptografik olarak imzalanmış bir açık anahtardan diğerine bir fon verisi transferinin yapıldığı bağımsız bir para sisteminin ortaya çıkışını tüm ödeme sistemleri arasında en ilgi çekici olanı olarak vurgulamışlardır.

Kripto para dünyasında sıklıkla adı geçen "coin" ve "token" kavramları aslında farklı yapıları ifade eder. Coin kendi blok zincirine sahiptir, token ise var olan bir coin blok zincirini çatallayarak kullanır. Yeni bir blok zincir yaratmak zordur, derin bilgi birikimi, teknik tecrübe, ve özverili çalışmalar gerektirir. Token oluşturmak ise, mevcut blok zincir sisteminden yararlanıldığı için daha kolaydır ve kısa bir sürede gerçekleştirilebilir.

Merkeziyetsiz bir işleyiş ve depolama sunan, geriye dönük işlemlerin değiştirilmesine izin vermeyen ve kurcalamalara karşı dayanıklı bir platform olarak tanımlanan blok zincir teknolojisi, 2008 yılında merkezi bir otorite yerine kriptografik mekanizmalarla korunan kripto para birimlerinin temelini oluşturmuştur. Kripto para birimleri, eşler arası oluşturma (peer-to-peer networking), kriptografi (hash fonksiyonları, dijital imzalar) ve ekonomik alanda oyun teorisi gibi çeşitli bilimlerdeki başarıların bir kombinasyonu sonucu ortaya çıkmıştır (Marius, 2018).

Kripto para birimleri, bir ödeme aracı olan paranın küreselleşmesini sağlamak için geliştirilmiştir. Ancak gerçek şudur ki, insanlar kripto paraları onlarla ödeme yapmak için değil bir yatırım aracı olarak kullanmaktadır. Asıl amacı satın alınan mal ve hizmet karşılığında para transferi yapmak olan kripto paralara bireyler ve kuruluşlar yatırım yaparlar ve fiyat dalgalanmalarından faydalanarak spekülatif hareketlerle para ticareti gerçekleştirirler.

Kripto para ağları, eşler arası temelinde çalışan bir ağ modellemesidir. Eşler arası ağda, aynı ağda yer alan bilgisayarlar arasında direk bir bağlantı vardır. Bir düğümden diğerine veri iletiminde, bu iletişimin yönetmek için herhangi bir sunucuya gerek kalmaz. Bu sebeple, kripto para uygulamaları merkezi bir sunucuya bağlı kalmayı gerektirmez. Katılımcılar arası İnternet tabanlı kurulan esnek bir ağ üzerinde çalışmayı öngörür. Ayrıca, yazılımlar açık kaynaklıdır, ücretsiz şekilde indirilerek kullanılabilir, hatta katılımcılar tarafından güncellenerek iyileştirilebilir.

Kripto para ile ödeme yapmak için gerçek kimliklerin kullanılması gerekmez. Katılımcıları tanımlayan gerçek kimlikler yerine kriptografik kimlikler vardır. Ancak dikkat edilmezse, tüm katılımcılara açık işlem defterlerinde akıllı algoritmalar sayesinde birbirlerine bağlanan işlemler gerçek kimliğin ortaya çıkmasına sebep olabilir. Her katılımcı istediği zaman sisteme girip çıkmakta özgürdür. Bir kişi, madencilik sürecine katılarak ve hesabına aktarılan kripto parayı kabul ederek kripto para sahibi olabilir, kripto para biriminin fiyat dalgalanmalarına yatırım yapabilir.

Madencilik, yeni kripto para biriminin oluşturulduğu süreçtir, ancak tüm kripto para birimlerinde madencilik yapılmaz. Aslında madencilik, yeni işlemleri içeren yeni bir blok oluşturulduktan sonra doğrulanması ve blok zincirine eklenmesi sürecidir. Temelinde,

karşılığında madencinin ödül olarak madeni para kazandığı karmaşık matematik algoritmalarını çözmek yatar. Madencinin bilgisayarı yeni bir bloğu doğruladığında, bir miktar ödül para kazanır. Örneğin, bitcoin kripto parasında yeni bir bloğun doğrulanması madenci katılımcıya 12,5 BTC kazandırmaktadır. 2016 yılından önce bu ödül tutarı 25 BTC idi. Bitcoin kripto parasında ödül tutarı dört yılda bir %50 oranında azaltılır.

Madencilik yapmak için ilk zamanlarda donanım olarak kullanılan CPU (Central Processing Unit) üniteleri güncel madencilik bulmaca zorluk derecelerini çözmek için çok yavaş kalmışlardır. Bu yüzden, madenciler CPU yerine paralel bağlanarak çözüm zamanını kısaltan GPU (Graphics Processing Unit), enerji verimliliğini arttıran ve kullanımı kolay Field Programmable Gate Arrays (FPGA) ve enerji tüketimini azaltarak hash gücünü arttıran ASICs (Application Specific Integration Circuits) donanımlarını kullanırlar. Bunların dışında, madenci bilgisayarlarında web tabanlı yazılımların da yüklü olması gerekir. Birçok madencinin işbirliği yaparak madencilik ödülünü paylaştığı madencilik havuzlarının yanısıra, madencilik fabrikaları, bulut madenciliği (cloud mining), uzaktan depolama hizmeti (mining though remote hosting service) ile madencilik gibi yöntemleri de bulunmaktadır.

Kripto para sahibi olmak için öncelikle paranın saklanabileceği bir cüzdana (wallet) ihtiyaç vardır. Kripto para cüzdanları, kripto para harcanırken işlemleri imzalamak için gerekli olan katılımcının özel anahtarına ek olarak kripto madeni paralarını içeren bir hesaptır. Kripto para cüzdanlarının değişik biçimleri mevcuttur. İlk bloktan son bloğa tüm blok zinciri bulundurmaya izin veren masaüstü bilgisayarlarına yüklenmiş cüzdan yazılımları, blok zincirin sadece bir kısmına izin veren akıllı telefon uygulamaları mobil cüzdanlar, İnternete bağlı bir sunucuda saklanan özel anahtarların her yerden erişilebildiği görece daha riskli web cüzdanları, anahtarların İnternete bağlı olmayan donanımsal cihazlarda saklandığı soğuk (hardware) cüzdanlar, anahtarların karekod formunda saklanabildiği kayıt (paper) cüzdanlar kullanılmakta olan cüzdan biçimleridir.

1.2.1. Kripto Para Birimlerinin Avantajları

Efanov ve Roschin (2018) kripto paraların öne çıkan avantajlı özelliklerini şu şekilde sıralar:

- İnternet üzerinde gerçekleşen işlemler için büyük ölçüde düşük işlem ücreti imkanı sunar.
- Kredi kartlarından daha fazla anonimlik sağlar. Hesaplar, gerçek kimlik bilgileri ile değil takma isimler üzerinden oluşturulur ve protokol, her işlem için yeni hesap numaralarının kullanılmasını teşvik etmek üzere tasarlanmıştır.
- Merkezi olmayan tasarım enflasyona karşı koruma sağlar. Geleneksel para birimleri, para arzını düzenlemek için bir merkez bankasına güvenir ve gerektiğinde para basma yetkilerini kullanarak yeni parayı dolaşıma sokar. Bunun aksine, kripto paralar düzenli aralıklarla büyümesine izin verilen nispeten sabit bir para arzını garantilemeye çalışır ve bunun için kriptografi kullanır.

1.2.2. Kripto Para Birimlerini Tehdit Eden Sorunlar

Kripto para birimlerinin genel anlamda üç ana sorunu vardır:

1. Ölçeklenebilirlik: Kripto para ağları ölçeklenebilirlik sorunları ile karşı karşıyadır. Özellikle ağ bant genişliği, ağ boyutu ve depolama gereksinimleri zorluklar ortaya çıkarır. Blok onaylanması için ihtiyaç duyulan on dakikalık süre küçük meblağlarda yapılan işlemlerde uzun ödeme süreleri anlamına gelir. Örneğin, bir ekmek almak için on dakika gibi bir süre beklemek çok tercih edilen bir durum değildir. Bu, sistemin yavaş işlediği sonucunu ortaya koyar. Bu durumu iyileştirmek için daha iyi altyapılar gerekir.
2. Esneklik: Varolan geleneksel para birimleri ve ödeme sistemlerinin birlikte uyumlu çalışması için kuralların tanımlanması önemlidir, ilgili düzenlemelerin yapılması sistemin esnekliğine katkı sağlayacaktır.
3. Yönetilebilirlik: Kripto paraların gelecek kullanımları, uzun vadede nasıl etkileneceğinin ve değişebilirliğinin belirlenmesi önemlidir.

Ana sorunların yanısıra, pump and dump sorunu, %51 saldırısı ve madencilik havuzları, güvenilir olmayan ICO (Initial Coin Offerings) fon toplama projeleri, Arjantin enerji tüketimine eşdeğer enerji ihtiyaçları ve çevreye etkileri diğer sorunlar arasında yer alır.

1.2.3. Kripto Para Birimlerinin Güvenlik Sorunları

Blok zincir teknolojisinin güvenlik mekanizması, açık defter yapısı ve karşılıklı güveni sağlayan dağıtılmış fikir birliği esaslarına dayanmaktadır. Ancak sunduğu bu özellikleri, güvenliğine yönelik sorunları olmadığı anlamına gelmemelidir. Li ve diğ. (2020) blok zincir teknolojisinde karşılaşılan güvenlik sorunlarını şu şekilde gruplandırmıştır.

- %51 saldırısı: Karşılıklı güven temeline dayanan dağıtılmış fikir birliği mekanizması, bütün blok zincirin %51'lik kısmının kontrolünün tek bir saldırgan katılımcı ya da grubun eline geçmesi durumunda kötü niyetli kullanıma açık hale gelebilir. Tek bir madencinin Proof-of-Work temelinde çalışan bir blok zincirinde hashing gücünün %50'den fazlasını ya da Proof-of-Stake temelinde çalışan bir blok zincirinde tüm madeni paraların %50'den fazlasını ele geçirmesi bu tür saldırılara sebep olabilir. Saldırgan, çift harcama saldırısı başlatabilir, işlem sıralamasını değiştirebilir, ters işlem, işlem iptali yapabilir, işlem onayı ve madencilik faaliyetlerini engelleyebilir.

- Pump and Dump: Bir grup katılımcı büyük meblağlarda kripto para birimi satın alır veya kripto para biriminin satışını teşvik etmek için reklam ve medya vasıtasıyla pazarlamasını yaparak kripto paranın değerini arttırır. Bu manipülasyonun ilk dalgasıdır. Para birimi istenilen değere ulaştığında ise, kar elde etmek için önceden satın alınan para toplu şekilde satılır. Büyük meblağlarda yapılan satışlar paranın değerinin düşmesine neden olur. Bu ikinci dalgada pek çok yatırımcı para kaybeder.

- Özel Anahtar Güvenliği: Blok zincir sisteminde merkezi bir üçüncü taraf yer almadığı için katılımcının kimliği yerine geçen özel anahtar katılımcının kendisi tarafından oluşturulur. Bu anahtarın kaybedilmesi katılımcı hesabına erişimi imkansız hale getirecektir. Özel anahtar iyi muhafaza edilmelidir, iyi niyetli olmayan kişilerin eline geçmesi durumunda katılımcı hesabının başkaları tarafından kurcalanması riski doğacaktır.

- Yasadışı Faaliyet Zaafiyeti: Çoklu veya gerçek kimlikle ilgisi olmayan bir kripto para kimliğine sahip olabilme imkanı yasadışı faaliyetlere davetiye çıkarabilir. Kara para aklama, e-postalara eklenen programlar aracılığı ile toplu para gasbı, yeraltı dünyası para birimi olarak kullanılması başlıca suç faaliyetleri arasındadır.

- Çift Harcama: Bir kripto paranın aynı anda farklı işlemler için harcanması çift harcama sorunu olarak tanımlanır. Bitcoin çift harcama sorununu çözmek için tasarlanmış ilk ödeme sistemidir. Buna rağmen bu sorun hala bir güvenlik açığı olarak karşımıza çıkmaktadır. Özellikle PoW tabanlı blok zincirlerinde, saldırganın iki işlemin başlatılması ve onaylanması arasındaki zamandan faydalanarak çift harcama gerçekleştirmesi daha olasıdır. Blok zincir çatalları (fork) çift harcamayı kolaylaştırdıkları için istenmeyen bir durumdur.

- İşlem Mahremiyeti: Katılımcıların blok zincirindeki işlemleri izlenebilir olduğundan, blok zinciri sistemleri katılımcıların mahremiyetini korumak için önlemler alır. Bitcoin ve Zcash kripto para uygulamaları, kripto para alma işlemini saklamak için tek seferlik hesaplar kullanır. Ayrıca, kullanıcının her işleme özel bir anahtar ataması gerekir. Bu şekilde saldırgan, farklı kripto para işlemlerinin aynı katılımcıya ait olup olmadığını anlayamaz. Monero kripto para uygulamasında ise, katılımcılar bir işlem başlattıklarında, önceki işlemlerin çıktısı hangi madeni paraların harcandığını saklayabilmek adına “mixins" olarak adlandırılan madeni paraları ekleyebilirler, böylece saldırgan işlem tarafından harcanan gerçek madeni paraların bağlantısını çıkaramaz.

- Akıllı Kontrat Sorunları: Blok zincir teknolojisinin geliştirilmiş sürümünde kullanılmaya başlanan akıllı kontratlar işleyişte sorunlara yol açabilmektedir. Gizli bilgilerin sızdırılması ve kriptografi anahtarların çalınması akıllı kontratların kullanılması ile daha kolaylaşabilir. Akıllı kontrat yazılımlarından kaynaklanan hatalar da güvenlik açığına sebep olabilmektedir. Yazılımda kullanılmayan ya da kullanılsa bile her seferinde aynı sonucu üreten kod parçaları ya da çalıştırılması yüksek maliyete sebep olan döngüler sistem işleyişine uygun düşmemektedir. Sistem içerisinde fazla kaynak tüketimine sebep olan işlemler için maliyetinin altında ücretlendirme yapılması kötü niyetli bir katılımcının kaynakların tükenmesi için bu işlemi pek çok kez çağırarak Denial of Service saldırısı başlatması ihtimali de bulunmaktadır. Denial of Service (DoS) saldırısında saldırgan, ağdaki düğümlerin ağın sunduğu hizmetlerden yararlanmasına ve veriye ulaşmasına engel olmaya çalışır. Hedeflenen sistemdeki belleği tıkayarak sistemin çökmesine, yeniden başlamasına ve kullanıcılara hizmet verememesine sebep olur.

1.3. BITCOIN

Blok zincir tabanlı ilk kripto para birimi Bitcoin'dir. 2009'da devreye alınan Bitcoin, nakit ve kredi ile yapılan ödeme türlerinin bazı özelliklerini birleştiren ve bilgisayarlar arasında doğrudan bağlantının olduğu (peer-to-peer) geniş kapsamlı hatta küresel bir ağ üzerinde işleyen bağımsız bir para sistemidir. Nakit ve kredi tabanlı ödeme yöntemlerinin ikisi de açıkça üstün değildir. Nakit tabanlı bir sistemde, öncelikle bir nakit tahsisi yapılması gerekir, bu olmadan hiçbir işlem gerçekleşemez. Kredi temelli bir sistemin böyle bir nakit tahsisine ihtiyacı yoktur, ancak borcu olan kişinin borcu kapatmak için ödeme yapmaması riski vardır.

Nakit ödeme yönteminde olduğu gibi Bitcoin işlemlerinde de gönderen ve gönderilen katılımcılar açıkça bilinmez. Kripto para işlemi, bir katılımcı açık anahtarından diğerine kriptografik olarak imzalanmış bir elektronik fon verisi transferidir. Ayrıca nakit gibi Bitcoin işlemleri de geri alınamaz, özellikle kredi kartlarında olduğu gibi ödemelerin geri iadesi söz konusu değildir. Bununla birlikte, nakit paradan farklı olarak, Bitcoin kripto para küresel ağında katılımcıların tüm işlemleri doğrulaması ve onaylaması gerekir. Bu, bir tür merkezi olmayan ve katılımcılara dağıtılarak işlenen muhasebe yapılmasını sağlar, bu sayede tüm katılımcılar sistemin tüm işlem geçmişine erişerek kendi kopyalarını saklayabilirler. Ayrıca, bir katılımcının kredi kartından veya banka hesabından cüzdanını (wallet) fonlaması katılımcının kimliğinin ortaya çıkmasına sebep olabilir (David, 2018). İşlemler sırasında gerçek kimlikler kullanılmasa da tüm işlemlerin tamamen şeffaf olmasının Bitcoin kripto para biriminin tam manasıyla anonim olmadığı sonucunu ortaya koyması Meiklejohn ve diğ. (2016) tarafından yapılan araştırmada ifade edilmiştir.

Banka tamamen ortadan kaldırıldığı için dağıtılmış bir kripto para sisteminde, paranın iki kez harcanma olasılığı vardır. Bir kişi, aynı parayı farklı alıcılara aktararak iki işlemi paralel olarak düzenleyebilir (race attack) ya da henüz onaylanmamış bir işlemi kopyalayıp yeniden yayınlayabilir. Merkezi bir senaryoda, banka bu girişimi algılayabilir ve önleyebilir. Ancak dağıtılmış bir ortamda özellikle kötü niyetli katılımcıların varlığında benzer bir girişimin önlenmesi zordur. Bu zorluk, 2008 sonunda Satoshi Nakamoto tarafından yayınlanan "Bitcoin: electronic cashless peer-to-peer" çalışmadaki Bitcoin tasarımıyla aşılmıştır. Bitcoin, işlemlerin girdisinde yalnızca daha önce harcanmamış işlem çıktıları kullanılır, blok zincirde zaman damgalanmış işlemler kronolojik olarak sıralıdır. Hangi işlemin önce geldiğini belirlemek için işlemler bloklar halinde gruplanır. Aslında bloklar, içerdikleri işlemlerin zaman damgasıdır ve işlem geçerliliklerinin onaylanmış bir kanıtıdır. Bu şekilde oluşturulmuş halka açık bir blok zincir çift harcamaya karşı koruma sağlar.

Nakamoto'nun çalışmasında, Bitcoin para biriminin yaratılmasının başlıca iki ana amacı olduğu vurgulanmaktadır. Bunlardan ilki, normal insanların da para basabileceği gerçeğidir. Diğer para sistemlerinde kullanılan seri numaralarının rolünü işlem hash özetleri ve önceki işlemlere yapılan referanslar üstlenir. Bu sayede, seri numaraları veren bir banka ihtiyacı ortadan kalkmış olur. Diğer amacı ise, kripto paraların gerçek para birimlerinin aksine sınırlı miktarda yaratılması ve sınırsız para üretenlere karşı bir çözüm yolu sunuyor olmasıdır.

1.3.1. Bitcoin Ağ Yapısı ve İşleyişi

Bitcoin, temel iletişim yapısı olarak TCP bağlantılarını kullanarak dinamik şekilde oluşturulan, yapılandırılmamış bir eşler arası ağ kullanır. Belirli bir düğüm organizasyonunun bulunmadığı ağlarda katılımcılar rastgele iletişime geçerler. Düğümler diledikleri gibi ağa katılır ve ağdan ayrılırlar. Tschorsch ve Scheuermann (2016) çalışmasında ifade edildiği üzere Bitcoin eşler arası ağında amaç, blok zinciri üzerinde mutabakata varmak için bilgileri olabildiğince hızlı dağıtmaktır.

Tapsell ve diğ. (2018) araştırmasında belirtildiği üzere Bitcoin düğümleri IP adresleriyle tanımlanır ve bitcoin mesajlarının düğümler arasında iletilmesi için güvenilir bir kanal sağlayan TCP üzerinden çalışır. Bitcoin ağındaki her düğüm, en az sekiz bağlantıyı aktif olarak korumaya çalışır, bu sayının yetersiz kalması durumunda ek bağlantılar kurulur. Maksimum bağlantı sayısı 125'e kadar çıkabilir. Bir düğüm gelen bağlantılar için 8333 numaralı portu dinler. Bir düğüme bağlanmak için Bitcoin protokolü versiyon numarasını, blok sayısını ve bir zaman damgasını içeren "version" mesajı gönderir. Mesajı alan düğüm, talepte bulunan düğümün versiyonundan bağlantı kabul ediyorsa, bağlantıya hazır olduğunu gösteren bir "verack" mesajı ve kendi version mesajını geri gönderir. Bitcoin 0.7 sürümünden beri IPv6 desteklenmektedir.

Düğümler arasında en son mesaj alışverişinin üzerinden 30 dakika geçtiyse, düğümler bağlantıyı canlı tutmak için bir "heartbeat" mesajı gönderir. Karşı taraftan 90 dakika süresince gelen herhangi bir mesaj olmazsa, karşı tarafın ağı terkettiği varsayılır. Düğümler her 24 saatte bir kendi IP adreslerini içeren bir "addr" mesajı yayınlar. Mesajı alan düğümler, aldıkları addr mesajlarınındaki bilgileri saklar ve kendisi ile bağlantısı olan düğümlere iletir. Bu akış sayesinde, Bitcoin ağındaki düğümler, güncel aktif düğümlerin IP adreslerini bir zaman damgası ile beraber veritabanında tutar. Ağa yeni bağlanan bir düğüm kısa sürede düğümlerin aktif düğüm veritabanına eklenir. Bir düğümden mesaj gelmemesi ilgili düğümün artık ağdan ayrıldığının bir işareti olarak yorumlanır. addr mesajları, ağdaki düğümleri keşfetmenin en yaygın yoludur. Bir düğüm başka bir düğümle bağlantı kurduğunda, bir "getaddr" mesajı göndererek komşu düğümün farkında olduğu ağ düğümlerinin bir listesini sorgular. Mesajı alan komşu düğüm, farkında olduğu aktif düğümler listesinden rastgele seçilen 1000 kadar düğümü içeren listeyi içeren bir addr mesajı ile cevap verir.

Bitcoin ağında, mutabakat alınması için yeni blok ve işlemlerin ağdaki tüm katılımcı düğümlere dağıtılması gerekir. Bir katılımcı düğüm, yeni geçerli işlemler oluşturduğunda, komşu düğümlerine "inv" (inventory) mesajı göndererek yeni işlemler olduğunu haber verir. Bu inv mesajı işlemlerin tüm detaylarını içermez, yalnızca işlem hash değerini içerir. Bu mesajı alan komşu düğümler, mesajdaki hash değerine sahip işlemler veritabanlarında mevcut değil ise, getdata mesajı ile geri dönüş yaparak ilgili işlem detay verilerini almak ister. İletişimi başlatan düğüm, getdata mesajına cevap olarak "tx" mesajı ile işlem detay kayıtlarını komşu düğümlere gönderir. İşlem kaydını alan düğümler işlemi doğruladıktan sonra, yeni bir inv mesajıyla başlayarak kendi komşularına işlem bilgilerini sunmak için yeni bir iletişim başlatırlar.

Bir düğüm, komşu düğümlerine farklı işlemleri içeren inv mesajları gönderir ve kendi veritabanında komşu düğüm bazında bu işlemlerin bir listesini tutar. inv mesajında yer alan sınırlı sayıda işlemlerin hangi işlemler olacağı rastgele şekilde veya işlem ücretleri dikkate alınarak seçilir. Bir düğümün, onaylanmamış bir işlemi, iletişimi başlatan düğüm yerine düğümün komşularından alma olasılığı da vardır.

İşleminin yaratıcısı işlemin tüm düğümlere dağıtımından sorumludur. Bu nedenle, işlem blok zincirine girmediyse, bir sonraki blokta dikkate alınmasını sağlamak için yeniden yayınlaması gerekebilir. Saldırganların yine de ağa özel bilgileri elde edebilme durumu söz konusudur.

Blok yaratılmasında difficulty (zorluk) derecesi ve nonce değeri önemlidir. Nonce değeri, "blok numarasına hangi sayı eklenirse (difficulty=3) 3 tane sıfır ile başlayan bir hash fonksiyonu çıktısı elde edilir?" sorusunun cevabıdır. Cevabı bulan katılımcı, bloğu bulduğu nonce değeri (sertifikasyon kodu) ile sertifikalandırmış olur ve bu bilgiyi tüm katılımcılara haber verir. Bu sertifikasyon kodunun bulunması adımına MINING (madencilik) adı verilir.

Doğrulanmış blokların yayılması işlemlerin yayılmasına benzer. Mutabakat algoritmasını başarıyla çözen bir madenci düğüm, önce tüm bağlı olduğu düğümlere bir inv mesajı yayınlar. Mesaj içeriğinde tüm blok gönderilmez, yalnızca istenirse aktarılır. Yeni bir blok alan düğümler, bloğu aynı şekilde komşularına ileterek bloğun tüm ağa yayılmasını sağlarlar. Yeni bir bloğun komşuluklar aracılığıyla düğümden düğüme iletilerek ağa yayılması bir gecikmeye neden olur. Yayılma gecikmesini azaltmak amacıyla bloklar için trickling algoritması kullanılır.

Bir düğüm ağa bağlıyken ağda yayınlanan blokları alarak blok zincir kopyasına kaydeder ve veritabanını güncel tutar. Ancak, bir düğüm çevrimdışı iken, yayınlanan yeni blokların farkında olamayacaktır, bu nedenle ağa yeniden bağlandığında kendi blok zincirindeki eksik blokları elde etmesi gerekecektir. Bunun için, bağlı olduğu bir düğüme, veritabanında kayıtlı son bloğun blok başlığını içeren bir "getheaders" mesajı gönderir. Mesajı alan diğer düğüm, mesajdaki blok başlığı ile kendi blok zincir kopyasındaki son blokları karşılaştırır ve eksik olan blok başlıklarını "headers" mesajı ile cevap verir, headers mesajında en fazla 2000 blok başlığı yer alabilir. Eğer daha fazla blok başlığı gönderilmek istenirse, getheaders mesajının tekrar gönderilmesi gerekir. Bloklara ait detay bilgilerin alınması için "getdata" mesajı gönderilir.

Bir madenci düğüm, madencilik yaptığı bloğa dahil etmek için onaylanmamış işlemleri toplar ve saklar. Ağdaki onaylanmamış işlemlerin toplu havuzuna mempool adı verilir ve madenciler, bir bloğu keşfettikten sonra en fazla işlem ücretini kazanmak için, onaylanmamış işlemlerin olabildiğince büyük bir kısmını toplamaya çalışırlar. Çevrimdışı olduktan sonra ağa yeniden bağlanan düğümler, bağlı oldukları düğümlerden onaylanmamış işlemleri toplamak için "mempool" mesajı gönderirler. Bu mesajı alan düğünler veritabanlarındaki onaylanmamış işlemleri inv mesajı ile cevap dönerler.

Bitcoin kripto para ağının işleyiş adımları şu şekilde sıralanmıştır: (Satoshi, 2008)

1. Yeni işlemler tüm katılımcılara yayınlanır.
2. Her katılımcı onaylanmamış işlemlerden bir blok oluşturur.
3. Her katılımcı kendi bloğu için ağ zorluk derecesine uygun bir proof-of-work (hash) bulmak için çalışır.
4. Bir katılımcı bir proof-of-work (hash) bulunduğunda, bloğu tüm katılımcılara yayınlar.
5. Katılımcılar, blokta yer alan tüm işlemler geçerli ise ve bu işlemlerin çıktıları henüz harcanmadıysa, bloğu kabul ederler.
6. Katılımcılar bloğu kabul ettiklerini belirtirler. Kabul ettikleri bloğun hash değeri önceki hash değeri olacak şekilde, zincirde yer alacak bir sonraki bloğu yaratmak için çalışmaya başlarlar.

1.3.2. Cüzdanlar

Bitcoin ağına katılmak isteyen bir kullanıcı, protokolü yürütmek için gerekli dosyaları ve güncel ağdaki tüm işlemleri içeren ve Bitcoin Data Directory olarak adlandırılan bir veritabanını kendi bilgisayarına yüklemelidir. Yüklemeyi yapmak için bitcoin.org/en/download/ adresine erişerek Bitcoin Core Software indirilir. Tüm blok zinciri işleyen Core yazılımı lokal bilgisayara kurulur. Kurulum tamamlandıktan sonra, kullanıcı yazılımı çalıştırarak güncel blok zinciri kendi bilgisayarına indirir ve Bitcoin cüzdanını yaratır. Kullanıcının özel bir anahtar ataması yaparak oluşturulan cüzdanı şifrelemesi gerekir. Ardından kullanıcı ilk Bitcoin işlemlerini yapabilir. Bir diğer yöntemde, kullanıcı bir tarayıcı aracılığıyla Bitcoin veritabanını içeren ve Bitcoin protokolünü çalıştıran bir web sunucusuna erişerek Bitcoin ağına katılabilir. Pazmino ve Rodrigues (2015) çalışmalarında, kullanıcıların İnternet üzerindeki https://blockchain.info/wallet adresindeki yönlendirmeleri takip ederek cüzdan oluşturma ve cüzdana ait özel anahtar bilgisini girerek Bitcoin işlemlerini gerçekleştirme adımlarını açıklamışlardır.

Bitcoin ağına katılmak için, Armory (bitcoinarmory.com) veya Electrum (electrum.org) gibi ek özellikler sunan birçok alternatif uygulama da mevcuttur. Bu uygulamalarda, cüzdanlar merkezi olarak yönetilir. Blockchain.info dışında cüzdanların şifrelenmiş şekilde kaydedildiği tarayıcılar üzerinde işleyen Coinbase (coinbase.com) gibi çevrimiçi cüzdanlar, ağa katılmanın başka bir popüler yoludur. Tüm yazılımlar ve çevrimiçi cüzdanlar doğası gereği güvenlik sorunlarına eğilimlidir, çünkü hedeflenen bir makineye erişim sağlayan bir saldırgan, kullanıcının cüzdanına da erişim sağlayabilir. Bitcoin kullanabilmek için ihtiyaç duyulan ilk şey cüzdandır, bu sebeple cüzdanı güvence altına almak adına koruyucu araçlar kullanmak çok önemlidir. İşlemleri güvence altına almak için Bitcoin protokolü eliptik eğri kriptografisini yoğun bir şekilde kullanır. Özellikle dijital imzalama için Eliptik Eğri Dijital İmza Algoritması (ECDSA) kullanılır.

Madeni para (coin) almadan önce kullanıcının bir genel/özel anahtar çiftinden oluşan sanal bir cüzdan sahibi olması gerekir. Kullanıcı açık anahtarı, önce SHA-256 ve ardından RIPEMD-160 hashing algoritması uygulanarak, başına bir versiyon numarası ve hata tespiti için bir checksum (sağlama) eklenerek kullanıcının Bitcoin adresi türetilir. Belirsiz

karakterleri ortadan kaldırmak için adresler base58 olarak kodlanmıştır. Aslında, Bitcoin adresi açık anahtarın farklılaştırılmış bir gösterimidir ve açık anahtardan daha kısadır, kullanımı daha kolaydır. Bitcoin adreslerinin amacı, açık anahtarları kısaltmak ve gizlemektir. Bitcoin adresleri ödemeleri almak için kesinlikle gerekli değildir, ancak güvenli ve uygun bir ödeme sağlarlar. Tschorsch ve Scheuermann (2016) araştırmalarında, güvenlik ve gizliliğe zarar verdiği için anahtarın ve adresin yeniden kullanımından kaçınılarak işlemler için yenilenmiş anahtar ve adres kullanılmasının gerekliliğinden bahsederler.

1.3.3. İşlem ve Script

Bir Bitcoin işlemi, işlem hash değeri TXID (Transaction ID), girdiler (inputs) ve çıktılar (outputs) alanlarından oluşur. İşlem girdileri daha önceki işlemlerin harcanmamış çıktılarından oluşur. Bir işlemin her çıktısı, çift harcamayı önlemek için tüm blok zincirinde yalnızca bir kez girdi olarak kullanılabilir. Bir işlemin her bir çıktısı, sonraki bir işlem tarafından referans edilmemişse Harcanmamış İşlem Çıktısı-Unspent Transaction Output (UTXO) veya Harcanmış İşlem Çıktısı-Spent Transaction Output (STXO) olarak tanımlanır. İşlem girdileri her zaman referans verilen çıktıdaki tüm madeni paraları (coin) kullanır. İşlemi başlatan kullanıcı, tüm işlem çıktılarını başkalarının adreslerine aktarmak zorunda değildir, işlemin bir çıktısını da, para üstü (change) olarak kendi adresine atamak için kullanabilir. Girdilerin toplamı çıktıların toplam değerinden fazla ise aradaki fark madencilere işlem ücreti olarak atanır. İşlem girdilerinin önceki işlem çıktılarına bağlanması işlem geçmişinin genesis bloğa ve madeni para yaratılan bir işleme kadar izlenmesine olanak verir. Genesis bloğu, blok zincirin ilk bloğudur ve sisteme 50 BTC'lik ilk tedariği sağlar. Tschorsch ve Scheuermann (2016) çalışmasında madeni para bazlı işlem, bir bloğu onayladığı için madenciyi ödüllendiren ve böylece sisteme yeni madeni paralar getiren işlem olarak tanımlanır.

Bitcoin yazılımında, işlemlerin programlanması yığın tabanlı (stack-based) ve sonsuz döngülere izin vermeyen Turing-incomplete bir programlama dili olan Bitcoin Script ile yapılmıştır. Bitcoin Script, Bitcoin yazılımı ile etkileşimde bulunarak UTXO-harcanmamış işlem çıktılarının harcanma kurallarının tanımlandığı bir komut dosyasıdır. Bitcoin Core olarak bilinen Bitcoin yazılımının tamamı Bitcoin script ile yazılmamıştır. Orjinal Bitcoin Core, C++ progamlama dili kullanılarak geliştirilmiş, sonrasında eklenen yeni modüller Phyton, Java ve Go yazılım dilleri ile kodlanmıştır. Yazılım hangi dilde ise Bitcoin script de o programlama dili kullanılarak yazılmıştır.

Bitcoin yazılımında, alıcının parayı alabilmesi için paranın gönderildiği adresin gerçekten kendisine ait olduğunu kanıtlamasını gerektirir. Bu amaç doğrultusunda, bir Bitcoin işlemindeki her çıktı (output) bir komut dosyası olan Bitcoin script ile tanımlanır. Script, sistem içi tanımlı sabitler ve gerçekleştirilmesi istenen komut adımlarından oluşur ve scriptPubKey olarak adlandırılır. scriptPubKey isimli komut dosyası, scriptSig adında bir dizi “parametre” bekler. Bir çıktıya bağlanan bir girdi, ilgili scriptPubKey komut dosyası için scriptSig parametre değerini sağlamalıdır. Parametre olarak sağlanan scriptSig için, script “TRUE” sonucunu dönüyorsa, ilgili girdi-çıktı arasında bağlantının geçerli olduğu kabul edilir.

Bir Bitcoin işlemini oluşturan alanlar şunlardır:

prevTx: Önceki işlemi tanımlayan hash
index: Önceki işlemdeki ilgili çıktının indeksi
value: Ödeme meblağı
scriptKeyPub: Madeni paraları harcamak için yetkilendirme scripti
scriptSig: Bir çıktıya bağlanan bir girdinin sağlaması gereken script

Ödemenin yapılacağı alıcı adresini oluşturmak için scriptPubKey komut dosyasına gönderilen parametrenin türüne bağlı olarak farklı işlem türleri bulunur. Güncel Bitcoin Core, bu farklı işlem türlerinden ikisini kullanır:

1.3.3.1. P2PKH (Pay-to-Pubkey-Hash)

Ödeme işlemi, alıcının açık anahtarına değil açık anahtarın hash değerine gönderilecek şekilde işlenir. Yaygın olarak kullanılan en önemli script Tablo 8'de paylaşılan P2PKH scriptidir. P2PKH kullanan bir işlem, çıktı adreslerinden gelen madeni paraları tek bir hedef adrese aktarır. Temel fikir, bir işlem girdisinde yer alan madeni paraların geldikleri çıktılardaki madeni paraların açık anahtarının hash değeri ile imzalanıp imzalanmadığını kontrol etmek ve bu kontrolü gerçekleyen bir çıktı script koduna sahip olmaktır. Bitcoin scriptlerindeki komutlar soldan sağa doğru işlenir. P2PKH kodunda, scriptSig parametresine, bir açık anahtar (pubKey) için hesaplanan hash değeri (pubKeyHash) Bitcoin adresi olarak atanır ve ilgili özel anahtara sahip olduğunu kanıtlayan bir imza (sig) gerektirir.

Tablo 8: P2PKH Yürütme Adımları

Stack (Yığın)	Script
Boş	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
<sig> <pubKey>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
<sig> <pubKey> <pubKey>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
<sig> <pubKey> <pubHashA>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
<sig> <pubKey> <pubHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG
<sig> <pubKey>	OP_CHECKSIG
TRUE	Boş

Script çalıştırıldığında, script komutları soldan başlayarak adım adım yığına atılarak işlenir. Burada, pubKey alıcının açık anahtarını, pubKeyHash alıcının açık anahtarının hash değerini (alıcının adresi) ve sig alıcının imzasını belirtir. İlk adımda, girdiden gelen scriptSig

parametreleri ve çıktıdaki scriptPubKey script kodu birleştirilir, sonuç sig ve pubKey değerleri yığına atılır. Scriptteki bir sonraki adım olan OP_DUP sabiti, yığındaki en son girişi tekrarlayarak, pubKey alanını bir kez daha yığına taşır. Ardından gelen OP_HASH160 sabiti ise, yığındaki son girişi SHA-256 ve RIPEMD-160 hash fonksiyonları ile peş peşe iki kez hashlenmesini sağlar. Sonuç hash değeri, bu scriptte pubHashA değerini yığına atar. Bir sonraki adımda pubKeyHash yığına itilir. OP_EQUALVERIFY sabiti, en üstteki iki yığın girdisinin eşitliğini doğrular ve farklılarsa bir hata verir. Bu adım, doğru açık anahtarın sağlanıp sağlanmadığının ilk kontrolünü yapar. Kalan son işlem, OP_CHECKSIG, alıcının imzasını açık anahtar ile kontrol eder ve eşleşirlerse TRUE gönderir. Bu son kontrol de geçilirse, script ödeme yapan tarafın harcama yapmasına izin verir.

1.3.3.2. P2SH (Pay-to-Script-Hash)

Ödeme işleminde alıcı tarafın kendi adresini dinamik şekilde oluşturan bir script yazmasına izin veren özel bir işlem türüdür. P2SH, alıcının bir paraya çevirme scripti (redeemScript) belirlemesini sağlar. Paraya çevirme scripti için hesaplanan hash değeri, Bitcoin adresi benzeri bir forma dönüştürülür ve işlemi başlatan tarafa gönderilir. Alıcının Bitcoin adresi yerine geçecek şekilde bu hash bir çıktı scriptine dahil edilir. Prensipte, paraya çevirme scripti alıcı tarafın belirleyeceği herhangi bir kontrol kriterini içeren herhangi bir script olabilir. pay-to-X olarak ifade edilirse, X alıcı tarafta ödeme yapılmadan önce kontrol edilmesi istenen özelliği gösterir. Tschorsch ve Scheuermann (2016) çalışmasında belirtildiği üzere P2SH, gelecekteki geliştirmeleri destekler ve yeni standart işlem türlerinin tanıtılmasını ve uygulanmasını kolaylaştırır.



Şekil 5: P2PKH ve P2SH Örnek Şablonları

1.3.4. Bitcoin ağında karşılaşılan saldırılar

Bitcoin özel anahtarları, Bitcoin ağındaki cüzdanları güvende tutmanın önemli bir parçasıdır. Bitcoin özel anahtarı, temelde belirli bir açık anahtara ve cüzdan adresine ait olan cüzdan kilidini açmak ve harcamak için kullanılan çok güvenli bir şifredir. Bu, Bitcoin özel anahtarlarının her zaman gizli tutulması gerektiği anlamına gelir. Bir kişinin özel anahtara erişimi varsa, cüzdandaki paraları kolayca çalabilir. Blok zincirini koruyan bir denetim kurulu veya kurum olmadığından bir katılımcının bitcoin cüzdanı ve işlemlerinin kötü niyetli tarafların saldırısına uğraması durumunda parasını geri alacağının garantisi yoktur. Bitcoin ağında karşılaşılan belli başlı saldırı türleri listelenmiştir.

1.3.4.1. Finney Saldırısı

Bitcoin ağında çift harcama hatasının gerçekleşmesi ihtimali az da olsa bulunur. Bir malın satın alınması durumunda, alıcı taraf satıcı tarafa bir para gönderme işlemi başlatır. Bir grup madenci işlemi onaylar ve satıcı taraf parayı gönderen alıcı tarafa işlemin onaylandığını düşünerek malı gönderir. Alıcı taraf kötü niyetli ise, kendi adresine veya işbirliği yaptığı bir başka adrese bir işlem daha tanımlar. Bu işlemin bir başka grup madenci tarafından onaylanması durumunda, zincirde bir çatallanma oluşur. Sahte işlemin yer aldığı çatal zincirin daha uzun olması durumunda, satıcıya para gönderen işlem geçersiz duruma düşer ve satıcı malını kaybetmiş olur.

1.3.4.2. Brute Force (Kaba Kuvvet) Saldırısı

Bazı düğümler ağda daha güçlüdürler ve diğer düğümler üzerinde etkili olabilirler. Bu düğümler hep beraber çift harcama içeren bir zincir için özel madencilik yaparak yeni bir çatalı genişletmek için çaba harcarlar.

1.3.4.3. Sybill Saldırısı

Bir düğümün, gerçekte varolmayan sahte kimlik oluşturarak yaptığı saldırıdır. Saldırganın amacı, bu sahte kimlikleri kullanarak büyük bir etki elde etmektir. Çift harcama ve Denial of Service saldırıları olasılığını arttırır. Onaylanmamış işlemlerin düğümler arasında iletilmesinde, Sybil saldırganlarının işlem kaynak IP adresini bulmasını zorlaştırmak ve gecikmeleri azaltmak amacıyla, trickling olarak bilinen bir algoritma kullanır.

1.3.4.4. Bribery (Rüşvet) Saldırısı

Kötü niyetli düğümler, madenci düğümlere rüşvet vererek kendi lehlerine madencilik yapmalarını isterler. Bloğu durdurma ve çift harcama olasılığı arttırır.

1.3.4.5. Feather (Köpük) Saldırısı

Kötü niyetli bir madenci düğüm, belli bir katılımcının işlemini içeren bir bloğu görürse, çatallanma girişiminde bulunacağını bildirir. Fakat sonra bu niyetinden vazgeçtiğini ilan eder. Doğrulama sayısını arttırmak bu tip saldırganlar için caydırıcı olabilir, ancak yeterli değildir.

1.3.4.6. Eclipse/Netsplit Saldırısı

Saldırgan, ağı kontrol ederek düğümler arasında iletilen mesajları kontrol etmeyi hedefler. Kurban düğüm düşman düğümler tarafından kuşatılır ve ağın kalan kısmından izole edilir. Bu durum ağın işleyişini tutarsız hale getirir.

1.3.4.7. Goldfinger Saldırısı

Madenci bir düğümün ağdaki hesaplama gücünün %50'sinden fazlasına sahip olması durumunda gerçekleşir. Katılımcıların ağdan kaçınmasına sebep olur. Mutabakat sistemini tehdit eder.

1.3.4.8. Selfish Mining (Bencil Madencilik) Saldırısı

Madenci düğüm, çıkardığı bloğu gizleyerek ağdaki düğümlerle paylaşmazlar, diğer düğümlere yanlış bilgiler verebilir. Çatallara sebep olarak diğer madenci düğümlerin blok çıkarmak için kaynaklarını boşa harcamalarına yol açar.

1.3.4.9. Block Withholding (Blok Tutma) Saldırısı

Bencil madencilik saldırısına benzer şekilde kötü niyetli düğümler blokları asla paylaşmazlar. Hedefte meşru madenci düğümler yer alır. Amaç, ödül paraların alınmasına engel olmaktır.

1.4. ETHEREUM

Ethereum, düğümlerin diledikleri gibi ağa katılabildiği ve ağdan ayrılabildiği merkeziyetsiz bir ağda çalışan bir izinsiz (permissionless) blok zinciridir. Ethereum blok zinciri, Bitcoin scriptlerinin kısıtlarına karşın karmaşık durumları tanımlayabilen akıllı sözleşmeler olarak da bilinen kodların yürütülmesini sağlayan bir Turing-tam dilini içerir. Bitcoin ağındaki dağıtılmış bir defterden farklı olarak Ethereum, dağıtılmış bir durum makinesidir. Ethereum'un durumu, yalnızca tüm hesapları ve bakiyeleri değil, aynı zamanda önceden tanımlanmış bir dizi kurala göre bloktan bloğa değişebilen ve rastgele makine kodunu çalıştırabilen bir makine durumunu tutan büyük bir veri yapısıdır. Durumu bloktan bloğa değiştirmenin özel kuralları Ethereum Sanal Makinesi-Ethereum Virtual Machine (EVM) tarafından tanımlanır. EVM, sanal bir bilgisayarı açık ve dağıtılmış bir şekilde yürütmek için kullanılan yığın tabanlı (stack-based) bir protokol ve mimaridir. Geçici değerler son giren ilk çıkar (LIFO) yığın yöntemi ile kaydedilir. Yığının her öğesi, Keccak-256 hash algoritması ve eliptik eğri hesaplamalarını kolaylaştırmak için 256 bit olarak seçilmiştir ve yığının maksimum eleman derinliği 1024'tür (Gavin, 2014). EVM'deki programlara Ethereum akıllı sözleşmeleri (smart contracts) denir. Ethereum akıllı sözleşmeleri açık kodlardır ve güncellenebilir.

Ethereum'da bir sözleşme, blok zincirinde yaşayan bir programdır. Bir katılımcı, program kodunu özel bir işlemle yükleyerek küçük bir ücret karşılığında bir Ethereum sözleşmesini yayınlar. Bu noktada kontrat herkes tarafından kullanılabilir hale gelir. Ayrıca geliştirici kodu yazarken özel bir koşul belirtmediği sürece kontrat silinemez. Akıllı sözleşmeler bytecode ile yazılır ve EVM sanal makinesi tarafından yürütülür. Yüklendikten sonra, sözleşme blok zincirinde geçerli olur. Akıllı sözleşmenin kendi bakiyesi vardır ve para gönderip alabilir. Diğer kullanıcılar akıllı sözleşme kodunun sunduğu API aracılığıyla, sözleşmede yer alan prosedür ve fonksiyonlara çağrılar yapabilir.

Adını 2014 yılında duyuran Ethereum, küresel bir blok zincir tabanlı uygulama platformu oluşturmayı hedefler. İlk defa 2015 yılında para piyasasında kullanılmaya başlanmıştır. Kullandığı Turing-tam programlama sayesinde, kullanıcıların isteklerine yanıt verebilen ve blok zincirinde kalıcı olarak depolanan kod parçaları olan akıllı sözleşmelerdeki tüm pratik hesaplamaları teorik olarak mümkün kılar (Sergei, 2017). Ethereum sistemindeki katılımcılar ve sözleşmeler, Ether olarak bilinen dağıtılmış bir para biriminde işlem yapar. Ayrıca, işlem hesaplamalarını beslemek için kullanılan ve genellikle bir Ether'in yaklaşık 1/100.000'i kadar

olan gaz olarak bilinen bir mekanizma da vardır. Her işlem yürütülürken gaz birimi kullanır ve yeterli gaz verilmezse, işlemin yürütülmesi sona erer. Gaz mekanizması, uzun süre çalışan sözleşmeleri sınırlandırmanın bir yoludur ve Bitcoin sistemindeki işlem ücretinin Ethereum'daki karşılığı olarak düşünülebilir. Ethereum'da bulunan talimatların her birinin bir gaz maliyeti sabittir. Narayanan ve diğ. (2016) yaptıkları çalışmada, gaz maliyetlerini değiştirmenin sert bir çatala yol açabileceğini belirtmişlerdir.

İşlem tabanlı bir yapı kullanan Bitcoin para biriminden farklı olarak Ethereum hesap tabanlı bir model kullanır. Aslında Ethereum'da iki tür hesap vardır: özel (private) anahtarla kontrol edilen kullanıcı hesapları (externally- owned account) ve akıllı bir sözleşmeyle kontrol edilen sözleşme hesapları (smart contracts). İki hesap türünün de bir Ether bakiyesi vardır, başka bir hesaba Ether gönderebilir, bir akıllı sözleşmenin katılımcılara açık fonksiyon ve prosedürünü çağırabilir. Hesaplar, 160 bit uzunluğunda heksadesimal olarak ifade edilen adreslerdir ve kullanıcı açık anahtarından veya sözleşmeyi oluşturan kişinin adresinden ve adından türetilirler. Bir hesap oluşturmak istendiğinde, rastgele 64 heksadesimal karakterden oluşan bir özel anahtar üretilir. Açık anahtar, ECDSA kullanılarak özel anahtardan üretilir. Açık anahtarın Keccak-256 karmasının son 20 byte (=160 bit) değeri alınarak hesap için açık bir adres oluşturulur. Mesajları ve işlemleri imzalamak için ihtiyaç duyulan özel anahtarlar, açık anahtardan türetilebilir. Ancak, mesaj ve işlemleri doğrulamak için kullanılan açık anahtardan özel bir anahtar türetilemez. Bu sebeple, özel bir anahtarı güvende tutmak hayati önem taşır.

Ethereum, program kodlarını çalıştıran işlem tabanlı bir durum makinesi (state machine) olarak tanımlanır. Başlangıç bloğu genesis bloktur ve genesis bloktan başlayarak gerçekleştirilen her işlem ile sistem durumu güncellenir (Gavin, 2014). Ethereum sisteminde, her adresin oluşturduğu hesap (account) ile bu hesaba ait verilerin oluşturduğu hesap durumu (account state) arasında bir eşleştirme ve bağlantı vardır. Tüm kayıtlı hesap verileri ve bağlantıların tamamı, Ethereum ağının "world state" olarak adlandırılan genel durumunu oluştur ve hesap bakiyelerini, sözleşme verilerini ve gaz tahminlerini içerir. Bir hesabı 160 bit uzunluğunda bir adres tanımlar ve bu adres ile hesaba ait bilgilerin yer aldığı hesap durumları Merkle/Patricia ağaç yapısı kullanılarak eşleştirilir ve bu yapıya trie adı verilir. Arka planda eşleştirme bir durum veritabanı (state database) kullanılarak gerçekleştirilir. Bu durum yapısı sayesinde, hesaplara ait bakiyeler ve akıllı sözleşmelere ait veriler blok zincir içinde değil Merkle/Patricia isimli özel ağaç yapısı içerisinde farklı bir veritabanında saklanmış olur. Blok zincir içinde saklanan Merkel ağacı kök (root) hash değerleri ile veritabanına ve dolayısıyla ilgili tüm verilere ulaşmak hash değerlerini tüm sistem durumu için güvenli bir kimlik olarak ortaya çıkarır. Ayrıca, tüm kök hash değerleri blok zincirinde saklandığı için, önceki sistem durumlara geri dönülebilir. ETH blok başlığı durum, işlemler ve makbuz ağaçlarının kök karmasını içerdiğinden, herhangi bir düğüm, potansiyel olarak sınırsız olabilen tüm durumu depolamaya gerek kalmadan Ethereum durumunun küçük bir bölümünü doğrulayabilir (Gavin, 2014).

Bir hesap durumu aşağıdaki alanlardan oluşur: (Gavin, 2014)

- Nonce: Özel anahtarla kontrol edilen hesaplar için hesap tarafından gönderilen işlemlerin sayısı veya sözleşme hesapları için hesap tarafından oluşturulan sözleşme sayısı

- Bakiye: Hesaba/adrese ait wei sayısı. Wei, Ether'in bir alt birimidir ve Bir Ether $1x10^{18}$ wei vardır.
- storageRoot (Depolama Ağacı Kök Değeri): Her Ethereum hesabının kendi depolama alanı vardır. Bu depolama alanı bir Merkle/Patricia ağacıdır ve bu ağacın kök düğümünün 256 bitlik hash değeri, storageRoot değeri olarak saklanır. Bir blok onaylandığında işlem gören bir hesabın bakiyesinde bir değişiklik olmuşsa, blok içerisindeki kök (root) hash değerinden ağacın ilgili dalı bulunur ve ilgili hesap bakiyesi güncellenir.
- codeHash: Hesap bir akıllı sözleşme ise sözleşmenin bytecode kodunun hash değeri

1.4.1. İşlemler

Ethereum kripto para ağında, bir adresten diğerine Ether para birimi göndermek veya bir adresteki akıllı sözleşmenin bir fonksiyonunu çağırmak işlem olarak adlandırılır. Bir işlem, Ethereum sanal makinesi (EVM) tarafından otomatik yürütülecek bir dizi talimatı içeren dijital olarak imzalanmış bir veri yapısıdır. Bir hesap, Ethereum ağının durumunu güncellemek için EVM'de yürütülecek bir işlem isteğinde bulunur. Kriptografik hash değeri oluşturulan işlem ağın tamamına iletilir ve onaylanmamış işlemlerin bulunduğu mempool havuzuna dahil edilir. Bir madenci düğüm, işlemi mempool'dan seçerek yürütür, bir bloğa dahil ederek geçerliliğini doğrular, ilgili hesaplamaları yapar ve ortaya çıkan durum değişikliğini ağın geri kalanına yayar. Bir işlemin geçerli olması için gaz ücretinin ödenmesi ve işlemin madenci düğüm tarafından çıkarılması gerekir. Madenci düğümlerin her zaman yüksek gaz ücretli işlemlere öncelik verdikleri unutulmamalıdır. İşlemin yer aldığı bloktan sonra oluşan blokların sayısı arttıkça işlemin geçerliliği de artmış olur.

Ethereum'da iki tür işlem vardır. Yeni bir sözleşmenin yaratılması, mevcut bir sözleşme fonksiyonunun yürütülmesi ile ilgili işlemler ve Ether para birimi transferi işlemleri. Her iki işlem türü aşağıdaki ortak alanları içerir: (Sergei, 2017)

- Nonce*:* Gönderici tarafından gönderilen işlemlerin sayısı
- gasPrice: İşlemi yürütmek için ortaya çıkan hesaplama maliyetlerini karşılamak için kullanılacak gazın bir birimi için ödenecek Wei miktarı
- gasLimit: Sözleşmeyi yürütmek için harcanacak maksimum gaz miktarı. Bu gaz miktarı yürütme başlamadan önce ödenir, sonrasında değiştirilemez.
- to: Mevcut sözleşmenin bir fonksiyonun çağıran ya da Ether transferi yapan bir işlemde, 160 bit uzunluğunda çağıran kullanıcının ya da alıcının adresi. Eğer yeni bir sözleşme yaratan bir işlemse, herhangi bir değer yer almaz.
- value: Aktarılacak tutar - wei para birimi cinsinden
- v, r, s: İşleme ait dijital imza verileri. Bu veriler kullanılarak işlemi başlatan hesap elde edilir.

Ortak alanlar dışında, yeni bir sözleşmenin yaratıldığı işlem türünde ek bir init alanı bulunur. Bu alan yeni sözleşmeyi yaratmak için çalıştırılacak EVM kodunu içerir. EVM kodu yalnızca bir kez çalıştırılır ve ardından hemen silinir. Çalıştırma sonucunda yeni sözleşmenin program kodu oluşur. Oluşan sözleşme kodu sözleşme her çağrıldığında çalıştırılacak olan koddur.

Mevcut bir sözleşme fonksiyonun çağrıldığı diğer işlem türünde ise, çağrılan fonksiyonun girdi parametre değerlerini içeren data alanı mevcuttur (Gavin, 2014).

İşlemi başlatan katılımcı, yapılacak hesaplamanın tüketmesi beklenen maksimum gaz miktarını (gasLimit) ve birim gaz başına ödemek istediği fiyatı (gasPrice) belirtir. İşlem ücreti, gasLimit ile gasPrice değerleri çarpımına eşittir. İşlemi başlatan katılımcı, işlemi gerçekleştirmek için gerekli gazı sağlamazsa işlem geçersiz sayılır. İşlem yürütme başarılı olursa ve yürütme sonrasında kalan Ether olursa, iade edilir. Yürütme sırasında bir hata meydana gelirse işlemin sistem durumu üzerinde bir etkisi olmaz, ancak işlem için ayrılan tüm gaz tüketilmiş olur (Sergei, 2017). Gaza harcanan tüm para genellikle madencinin adresine gönderilir, madenciler ödül olarak gaz ücretini alırlar. Gaz sadece işlemleri yürütmek için değil, depolama ödemelerini yapmak için de kullanılır. Artan depolama, tüm düğümlerde Ethereum durum veritabanının boyutunu artırdığından, depolanan veri miktarını küçük tutmak için bir teşvik vardır. Bu nedenle, bir işlem depolanan verilerde bir silme adımına sahipse, o işlemin yürütülmesi için gerekli gaz ücreti iade edilir.

Bir işlemin yürütülmesi için başlangıç gereksinimleri Krupp ve Rossow (2018) tarafından yapılan çalışmada aşağıdaki şekilde listelenmiştir:

- İşlemler, RLP (Recursive Length Prefix) isimli, iç içe binary veri dizilerini (nested arrays) kodlamak için kullanılan bir veri formatıdır. Ethereum, işlemleri seri hale getirmek için bu formatı kullanır.
- Geçerli bir katılımcı dijital imzası
- Katılımcı hesaptan gönderilen işlemlerin sayısını tanımlayan nonce değerinin geçerli olması
- İşlemin gaz limiti, işlem kapsamında yapılacak işlem yürütme, veri işleme ve sözleşme yaratma kriterleri dikkate alınarak doğru hesaplanmalıdır.
- Gönderenin hesap bakiyesinde, ödemesi gereken gaz maliyetlerini karşılamaya yetecek kadar Ether bulunmalıdır.

İşlem verilerinde yapılacak aramaları kolaylaştırmak için işlem verileri üzerinde indeks oluşturabilmek ve işlem veri güvenliğini, doğruluğunu sağlamak amacıyla her işlemin yürütülmesine ilişkin bazı bilgileri içeren işlem makbuzu (transaction receipt) oluşturulur. Her işlem makbuzu, endekslenerek Merkle/Patricia ağacından oluşan trie yapısına yerleştirilir ve ağacın tepesindeki kök (root) değeri blok başlığına kaydedilir. İşlem makbuzu, bir işlemin sonuçlarını gösterir. İşlem sonrası oluşan durum (poststate), işlem gerçekleştirilirken kullanılan gaz miktarı (cumulativegasused), işlemin yürütülmesi sırasında oluşturulan log (logs) ve bu log bilgilerinden oluşan bloom filtresi (bloom) alanlarından oluşur. Bloom filtresi, veritabanına daha az sorgu yaparak işlem log kayıtlarına kolay erişmek için kullanılır. Filtre bir çeşit şablon olarak düşünülebilir. Bir hash verisinde belirli sıradaki bitlerin her koşulda “1” olarak atanması, işlem log kayıtlarını diğer veriler arasında kolaylıkla seçilmesini sağlar (Gavin, 2014).

Ethereum kripto para sisteminde, birden fazla madenci düğüm yakın zamanlarda blok yayınlayabilir. Bu durumda, ağı oluşturan tüm katılımcı düğümler aynı anda kendi defterlerini güncelleyemezler ve Bitcoin sisteminde oluşan yetim (orphan) bloklara benzer şekilde, bir

ana (parent) bloktan birden fazla çocuk (child) blok oluşur. Ancak, oluşan bu bloklardan yalnızca bir tanesi doğrulanır, diğer blok Ethereum sisteminde amca (uncle) bloğu olarak adlandırılır ve blok başlıklarında amca blokların hash değerleri saklanır. Madenciler kazandıkları ödülün belirli oranda bir kısmını, geçerli blok zincirine eklenen bloğun blok başlığında kayıtlı amca blok madencileri ile paylaşırlar. Amca bloklar nedeniyle, yetim bloklara harcanan enerji çift harcama için gereken iş miktarını artırarak güvenliğe katkıda bulunur (Sergei, 2017).

Bir bloğun büyüklüğünün belirleyicisi gaz limitidir. Blokta yer alan işlemlerin toplam gaz miktarının blok için belirlenen gaz limitini geçmemesi gerekir. Herhangi bir madenci, gaz limitini, ana blok gaz limitinin yaklaşık %0,1'ine kadar arttırarak ya da azaltarak değiştirebilir. Bundan dolayı, Ethereum ilk bloğundaki gaz limiti 5.000 iken Nisan 2021 itibariyle gaz limiti 15.000.000 civarındadır (Preethi, 2017).

Ortalama 12 saniyede yeni bir blok oluşturulur ve zincire eklenir. Etherscan, Ehterchain, Ethplorer, OKLink portalları hızla oluşan bu verileri takip etme imkanı sunar. Kullanıcılar, bloklar, işlemler, hesaplar, madencilik ve blok zincir ağı hakkında sürekli güncel bilgi sahibi olabilirler. Bir Ethereum bloğunda, blok başlığı, blok başlığı formatında amca blokların başlıkları ve bloğun içerdiği işlemler yer alır (Gavin 2014).

Tablo 9: Ethereum Blok Başlık Alanları ve Tanımları

ETHEREUM BLOK BAŞLIĞI	
ALAN ADI	TANIMI
ParentHash	Geçerli bloktan önce gelen bloğun Keccak 256-bit hash değeri
UncleHash	Tüm amcaların birleşik Keccak 256-bit hash değeri
Beneficiary	Bloğun madenciliğinden toplanan tüm ücretlerin aktarılacağı 160 bitlik adres
StateRoot	Sistemin tüm durumunu depolayan Merkle trie yapısının kök Keccak 256-bit hash değeri
TransactionsRoot	Bloğun işlem listesi kısmındaki her işlemle doldurulmuş trie yapısının kök düğümü Keccak 256-bit hash değeri
ReceiptsRoot	İşlem makbuzları trie yapısının kök düğümeü Keccak 256-bit hash değeri
logsBloom	Bloğu oluşturan işlem log verilerine kolay erişim şablonu
Difficulty	Blok zorluk seviyesi. Zorluk seviyesi önceki Bloğun zorluk seviyesinden ve zaman damgasından hesaplanabilir.
Number	Ata bloklarının sayısı. Genesis bloğa sıfır değerinin atanır.
GasLimit	Bloktaki tüm işlemlerde harcanan toplam gaz miktarı blok gaz limitinden az olmalıdır.
GasUsed	Bloktaki işlemleri gerçekleştirmek için harcanan toplam gaz miktarı
Timestamp	Blok zaman damgası
Extra	Blokla ilgili verilerin tutulduğu en fazla 32 byte uzunluğunda elemanları olan bir dizi
MixHash	Nonce ile birleştirildiğinde, bu blokta yeterli miktarda hesaplamanın gerçekleştirildiğini kanıtlayan 256 bitlik bir hash;
Nonce	MixDigest ile birleştirildiğinde, bu blokta yeterli miktarda hesaplamanın gerçekleştirildiğini kanıtlayan 256 bitlik bir hash;

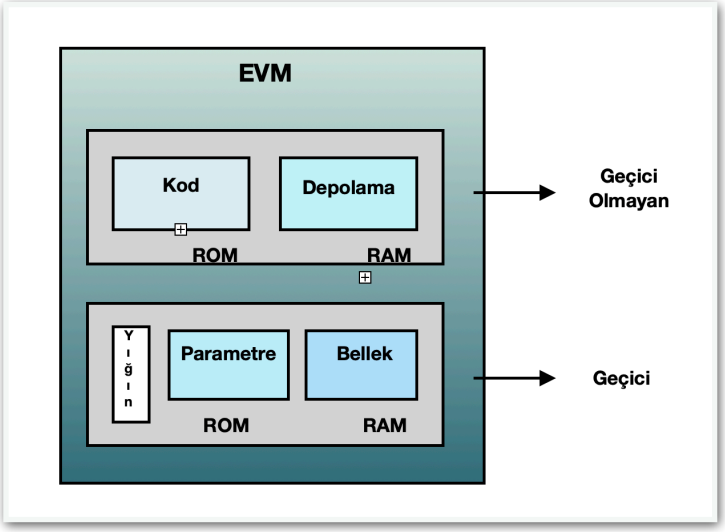
Tablo 9'da listlenen blok başlığında yer alan MixHash ve Nonce alanları iş ispatı (PoW) mutabakat algoritmasının çalıştırılması sonucunda hesaplanır. Ethereum, Bitcoin gibi şu anda iş ispatı mutabakatını kullanır, ancak sonraki sürümlerde mutabakat protokolünü hisse kanıtı (PoS) protokolüne yükseltmeyi planlamaktadır. Ethash, Ethereum için tasarlanan iş ispatı (PoW) algoritmasının özel adıdır. Bitcoin madenciliğinde kullanılan ASIC cihazlara karşı, Ethereum CPU ve GPU üzerinde çalışan Ethash algoritmasını kullanır. Algoritma, epoch adı verilen periyotlarda Directed Acyclic Graph yapısı üzerinde rastgele şekilde bir DAG veri seti oluşturur. Güncel konfigürasyonda her periyod güncel konfigürasyonda 30000 bloğa denk gelir. Büyüyen blok zincir ile birlikte rastgele şekilde oluşturulan bu veri seti de genişler. Her epoch periyodu için seed adı verilen bir başlangıç hash değeri vardır ve seed değeri, o noktaya kadar oluşan blok başlıkları taranarak hesaplanır. İlk epoch periyodu için seed değeri, 32 byte uzunluğunda "0" serisine karşılık gelen hash değeridir. Devam eden epoch periyodlarda, bir önceki periyodun seed hash değeri yeni seed değeri olarak atanır. Hesaplanan seed başlangıç değeri kullanılarak, güncel konfigürasyonda 16MB olarak belirlenen cache (önbellek) büyüklük parametresine uygun rastgele bir önbellek (cache) oluşturulur ve tüm normal Ethereum katılımcı düğümleri bu önbelleği saklar. Önbellek verilerinden rastgele seçilen veriler ile 1GB büyüklüğündeki DAG veri seti oluşturulur. DAG veri kümesi, Ethereum madenci ve tam katılımcı düğümleri tarafından saklanır. Madencilik, DAG veri kümesinden rastgele veri dilimleri almayı ve bunları bir araya getirerek hash hesaplamayı amaçlar. Normal düğümler, önbelleği kullanarak veri kümesinin sadece ihtiyaç duydukları belirli bir bloğunu yeniden oluşturur ve bu şekilde işlem/blok doğrulayabilirler. Önbellek kullanımı daha düşük bir belleğe ihtiyaç duyacaktır. Bu sebeple, daha düşük bir donanıma sahip normal statüdeki katılımcı düğümler için bütün veri kümesini saklamalarına gerek kalmadan sadece önbellek saklamaları daha uygundur (Gavin ,2014).

Bir düğüm zincirine bir blok eklediğinde, oradaki işlemleri sırasına göre yürütür, böylece Ethereum hesaplarının ETH bakiyelerini ve diğer depolama değerlerini değiştirir. Toplu olarak durum olarak bilinen bu bakiyeler ve değerler, düğümün bilgisayarında bir Merkle ağacında blok zincirinden ayrı olarak tutulur.

Bir blok zinciri protokolünün ya da bir akıllı sözleşme kodunun geliştirilmesi aşamasında, programcılar ve geliştiricilerin blok zincir ağı ve akıllı sözleşme programı işlevlerini test ettikleri testnet adı verilen farklı bir test ağ yapısı mevcuttur. Test ağında yapılan testler sayesinde hatalar düzeltilir, oluşturulan sistemin güvenli olduğu ve ana blok zincir ağının kullanıma hazır olduğu hususunda emin olunduktan sonra, blok zincir işlemlerinin mainnet olarak adlandırılan halka açık ana ağ üzerinde başlatılması mümkün olur. Genellikle bir blok zincir ana ağı başlatılmadan önce, proje ekibi Initial Coin Offering (ICO), Initial Exchange Offering (IEO) gibi yöntemleri kullanarak projeye ait Ether para arzını oluşturur ve ağa dahil olacak katılımcıların artması için çabalar. Ayrıca çoğu test ağı, işlemleri doğrulamak ve yeni bloklar oluşturmak için az sayıda düğümün seçildiği bir yetki ispatı (Proof of Authority*)* mutabakat mekanizması kullanır. Bu yetki ispatı mutabakat mekanizmasını kullanan test ağları arasında Görli, Kovan ve Rinkeby sayılabilir (Preethi, 2017).

1.4.2. EVM (Ethereum Virtual Machine)

EVM olarak bilinen Ethereum sanal makinası, bir tam Turing makinasıdır, ancak gaz miktarı ile sınırlandırılan hesaplama kapasitesi onu bir tam Turing makinesinden ayırt eden en önemli özelliğidir. Şekil 6'da gösterilen EVM sanal makinası, yığın temelli bir sanal makinadır ve Last In, First Out (LIFO) temelinde son giren girdinin ilk olarak çıkartıldığı bir yığın mantığında çalışır. Yığındaki her eleman 256 bit uzunluğundadır ve yığın en fazla 1024 eleman saklayabilir. EVM, word olarak adreslenen ve byte dizileri olarak kaydedilmiş elemanların saklandığı geçici bir belleğe sahiptir. Bellek dışında, geçici olmayan ve sistem durumunun bir parçası olan bir depolama ünitesi (RAM) mevcuttur. EVM makinasında program kodları ise, bellek ya da depolama kısmında değil farklı sanal bir ROM üzerinde saklanır ve sadece özel bazı komutlarla erişilebilir. EVM sanal makinası, EVM byte kodu isimli kendine özgü bir dile sahiptir.



Şekil 6: Ethereum Sanal Makinası (EVM)

Bir blokta yer alan işlemler EVM tarafından birbiri ardına sırasıyla yürütülür. Herhangi bir hesaplama öncesinde bellek ve yığın boştur, program sayacı sıfır olarak atanır. Hesaplama öncesinde, işlemci Ethereum sistem durumu, gaz değeri, program koduna sahip hesap, program kodunu çalıştıran hesap, işlemi başlatan adres, gaz ücret, girdi değerleri, program kodu, blok başlığı, program kod uzunluğu verilerini bilmelidir. Her işlem sonrasında Ethereum sistem durumu ve makina durumu güncellenir. Makina durumu, güncel gaz miktarı, program sayacı, bellek ve yığın elemanlarının güncel değerlerini içerir. Ayrıca her düğüm bilgisayarında, blok zincirinden ayrı olarak bir Merkle/Patricia ağacında tutulan Ethereum hesaplarının bakiyeleri ve diğer depolama değerleri değiştirilir. Program kodunu çalıştırıldıktan sonra, program sayacı arttırılırken gaz miktarı azaltılır. Program kodu, kontrollü (işlem sonu) ya da kontrolsüz şekilde hata (yetersiz gaz miktarı, yığın eleman sayının 1024 geçmesi gibi) alarak kesilmediği sürece sonraki işlemler ya da program kodları çalıştırılmaya devam eder.

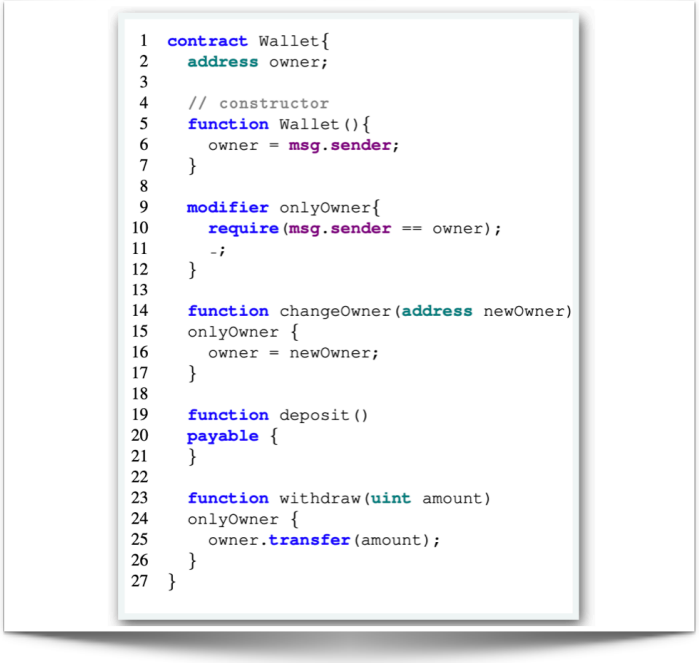
1.4.3. Akıllı Sözleşmeler (Smart Contracts)

Akıllı sözleşme, program koduna sahip özel bir Ethereum hesabıdır. Diğer Ethereum hesapları gibi akıllı sözleşme hesapları da Ether bakiyesi saklayabilirler, ek olarak kendilerine özel bir kayıt depo alanları mevcuttur. 256 bit anahtar değeri ile erişilen 256 bit uzunluğundaki veri dizisinin kaydedildiği bu kayıt depolama alanı, diğer akıllı sözleşmeler tarafından okuma ya da güncellemeye açık değildir. Sadece kriptografik olarak dışarıdan yapılacak değişikliklere karşı korumalıdır ancak gizli bir alan değildir. Tüm Ethereum işlemleri blok zincir üzerinde kaydedildiği için işlemler incelenerek bu özel depolama alanının içerdiği veriler tekrar oluşturularak elde edilebilir.

Ethereum sisteminde çalıştırılacak bir akıllı sözleşme, genellikle Javascript benzeri Solidity gibi yüksek seviyeli bir dilde kodlanır. Akıllı sözleşmeler, EVM'nin anlayabileceği EVM byte koduna dönüştürülecek şekilde derlenmelidir (Preethi, 2017). Derlenmiş akıllı sözleşme bayt kodu XOR, AND, ADD, SUB gibi standart yığın işlemlerini gerçekleştiren bir dizi EVM işlem kodu olarak yürütülür. EVM ayrıca, ADDRESS, BALANCE, KECCAK256, BLOCKHASH gibi bir dizi blok zincirine özgü yığın işlemini de uygulayabilir. Krupp ve Rossow (2018) çalışmasında ifade edildiği üzere EVM, güncel sözleşme işlem alanlarına erişime, sözleşme özel kayıt alanının güncellenmesine, güncel blok zincir durumunun sorgulanmasına izin verir.

Para yatırma (deposit), para çekme (withdraw) ve cüzdan sahibinin değiştirilmesi (changeOwner) fonksiyonlarına izin veren Şekil 7'de yer alan akıllı özleşme örneğinde bir cüzdan (wallet) modellenmektedir. Solidity dili kullanılarak yazılan bu akıllı sözleşmede, sözleşme (contract) ile aynı ismi taşıyan fonksiyon (function) constructor olarak adlandırılır. Krupp ve Rossow (2018) çalışmasından alınan örnek sözleşmede, wallet constructor fonksiyonudur. Constructor kodu sözleşme oluşturulurken sadece bir kez çalıştırılır ve bir daha sözleşmenin bir parçası olmaz.

Ayrıca Solidity dilinde modifier kullanımı mevcuttur. Modifier, mantıksal ve güvenlik kontrollerini yapmak için kullanılan özel bir fonksiyondur. "_" işareti ile belirlenen modifier fonksiyonu diğer fonksiyonlar tarafından çağrılırlar. Şekil 7'deki örnek cüzdan modelinde, onlyOwner modifier fonksiyonu, para gönderen hesabın kayıtlı cüzdan sahibi hesap olup olmadığı kontrolünü yapmaktadır. Yalnızca kontrol sonucunun olumlu olması durumunda, cüzdan sahibi para çekme ve cüzdan sahipliğinin değiştirilmesi işlemlerini geçekleştirebilir.



```
contract Wallet{
  address owner;

  // constructor
  function Wallet(){
    owner = msg.sender;
  }

  modifier onlyOwner{
    require(msg.sender == owner);
    _;
  }

  function changeOwner(address newOwner)
  onlyOwner {
    owner = newOwner;
  }

  function deposit()
  payable {
  }

  function withdraw(uint amount)
  onlyOwner {
    owner.transfer(amount);
  }
}
```

Şekil 7: Solidity Modify Örneği

Fonksiyon ve modifier fonksiyonlar sadece Solidity dilinde mevcuttur. Derleme sonrasında EVM byte koduna dönüştürülen akıllı sözleşme, yalnızca tek bir byte kod dizgisinden ibarettir. Akıllı sözleşme bir çeşit işlem olduğu için gönderici (sender), alıcı (to), değer (value) ve veri (data) alanlarından oluşur. EVM byte koduna dönüştürülen akıllı sözleşme fonksiyonlarının dört byte uzunluğundaki KECCAK-256 hash özet değerleri işlemin veri (data) alanında yer alır. Derlenmiş sözleşme byte kodu, genellikle dallandırılmış bir fonksiyon serisi ile başlar. Her bir dalda yer alan fonksiyon, sözleşmedeki fonksiyonların dört byte uzunluğundaki KECCAK-256 hash değeri ile karşılaştırılır.

Merkezi bir sunucu yerine merkezi olmayan bir ağda çalışan Dapp (Distributed Application) uygulamaları, program mantığı için akıllı sözleşmeleri ve veri depolama için Ethereum blok zincirini kullanır. Eğer varolan akıllı sözleşmeleri kullanan bir Dapp geliştiriliyorsa, kullanılan test ağında, öncesinde varolan akıllı sözleşmenin dağıtılmış kopyaları bulunur.

1.5. RIPPLE

Bir diğer alternatif madeni para XRP kodu ile piyasaya sürülen Ripple kripto para ağıdır. 2012 yılında Ripple isimli bir şirket tarafından "güvenli, anında ve neredeyse ücretsiz küresel finansal işlemler" sağlamak için geliştirilmiş ve piyasaya sürülmüştür. Bitcoin'e benzer ilkeler üzerine inşa edildiği için kripto para birimi olarak tanımlanır. Ayrıca Bitcoin'den farklı olarak, uygulamaya ait kaynak kodlar Ripple şirketine aittir, bu da herhangi bir yabancı tarafından müdahale edilemeyeceği anlamına gelir. Sistem ve kaynak kodlara sadece şirketin kendisi, İnternet servis sağlayıcıları ve Massachusetts Teknoloji Enstitüsü erişim sağlar. Ripple çok popüler bir ağdır, dünya çapında birçok banka bunu kendi yerleşim altyapıları için temel olarak kullanır ve XRP para birimi, son birkaç yıldır piyasa değerine göre sürekli olarak ilk 5 kripto para biriminde yer almaktadır. Sistem, Bitcoin'in merkezi borsalara olan bağımlılığını

ortadan kaldırmak, Bitcoin'den daha az elektrik kullanmak ve işlemleri Bitcoin'den çok daha hızlı gerçekleştirmek için tasarlanmıştır (Shailak, 2018).

Bitcoin'den farklı olarak, Ripple ağı üzerinde farklı döviz türlerinin karşılıklı değiştirilmesi de mümkündür. Ripple, her ne kadar açık kaynak kodu ile piyasaya sunulmuş olsa da güncel kod güncelleme ve canlıya geçişleri sadece Ripple Labs tarafından yönetilmektedir. Ripple ağı ilk kez 100 milyon XRP gibi sınırlı bir para arzı ile hayata geçirilmiştir. Bu arzın %20'lik kısmı Ripple kurucuları, %25'lik kısmı Ripple Labs ve %55'lik kısmı da ağın geliştirilmesi için paylaştırılmıştır. Armknecht ve diğ. (2015) yaptığı çalışmada ilk kullanımından bugüne günlük ortalama 170 hesap yaratıldığı belirtilmiştir.

Ripple kripto para ağında yer aşan katılımcı düğümler üç ana kategoride değerlendirilir.

- Kullanıcılar (users): Ödeme gönderen ve ödeme alan düğümler
- Pazar yapıcılar (market makers): Para alış verişini ve geçişini mümkün kılan düğümler
- Doğrulayıcı sunucular (validating servers): Sistemde gerçekleşen tüm işlemlerin kontrol ve doğrulanmasını sağalamak için kullanılan mutabakat protokolünü işleten düğümler

Ripple ağındaki düğümler gerçek isimleri yerine takma isim kullanırlar. Diğer kripto paralarda olduğu gibi, Ripple cüzdanı açık ve özel anahtar çiftinden oluşur ve düğümler bir açık/özel anahtar çifti ile donatılır. Bir katılımcı düğüm diğer düğümlere XRP Ripple para biriminde veya başka bir para birimininde para gönderebilir. Ödeme yapan düğümün özel anahtarı ile kriptografik olarak imzalanan işlemlerde, işlemi doğrulamak için ilgili düğümün açık anahtarı da yer alır. XRP olmayan para birimlerinde yapılan ödemeler için, Ripple dağıtılmış bir kredi ağı sistemi uygular. Eğer özel anahtar kötü niyetli biri tarafından ele geçirilecek olursa, ele geçiren kişi ilgili cüzdanın sahip olduğu tüm bakiyelere ve genişletilmiş kredi limitlerine koşulsuz olarak erişebilir. Kötü niyetli taraf, gerçek cüzdan sahibi adına kredi limitleri dahilinde kredi kullanabilir.

Bir Ripple cüzdanı aktif durumda kalabilmek için bir XRP bakiyesine sahip olmalıdır. Ayrıca, cüzdan sahibi bir işlem başlatabilmek için XRP para cinsinden bir işlem ücretini ödemek zorundadır.

Armknecht ve diğ. (2015) yaptığı çalışmada, Ripple kripto para sisteminin iki tür işleme izin verdiğini ifade ederler:

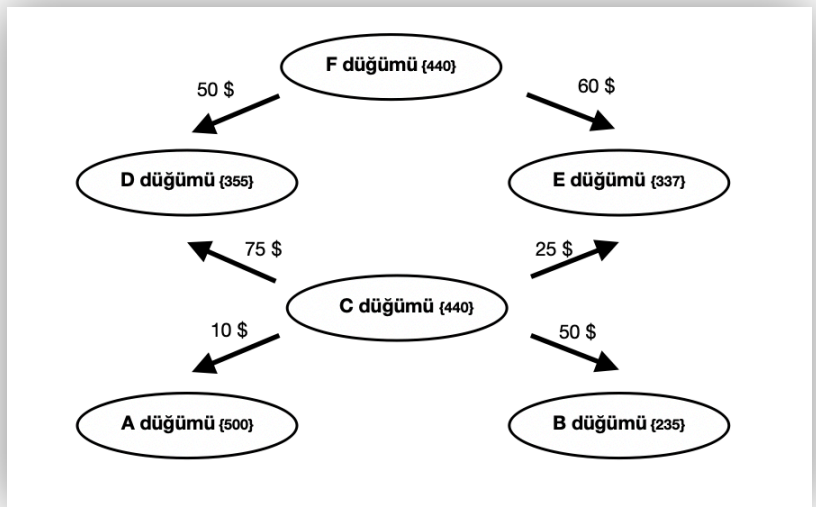
1. Direk XRP ödemeleri: Ripple para birimi XRP cinsinden yapılan bu ödemelerde, ödeme yapacak katılımcı düğüm cüzdanında gönderilmek istenen meblağ ve işlem ücretinin olması yeter koşuldur.
2. XRP olmayan ödemeler: Düğümler arasında kurulan bir kredilendirme temeline dayanan ödemelerdir. Bir A düğümü diğer bir B düğüme ancak B düğümü kendisine güveniyor ve yeteri kadar kredi veriyorsa ödeme yapabilir ve A düğümünün başlattığı IOU (I Owe You) işlemi başarılı ile gerçekleştirilmiş olur.

IOU ödeme işlemleri üç ayrı para biriminde gerçekleşebilir. Fiat currencies olarak adlandırılan USD, EUR gibi bir hükümetin kararı ile basılan paralar, kripto paralar, kullanıcılar tarafından tanımlanan paralar bu üç grubu oluşturur. Düğümler arasında bu üç ayrı birimde gerçekleşen ödemeler için aynı yöntem uygulanır. Ayrıca, pazar yapıcılar (market makers) olarak adlandırılan bir grup katılımcı düğüm vardır ki bunlar bir katılımcı düğümden aldıkları herhangi bir para birimi ödemeyi bir diğer para birimine dönüştürürler ve alıcı diğer düğüme aktarırlar ve para dönüştürme karşılığında küçük bir ücret alırlar. Sanchez ve diğ. (2016) çalışmasında, Ripple ağının farklı döviz cinsleri arasında takas işlemini gerçekleştiren bir köprü görevini gördüğünü anlatırlar.

Yeni oluşturulan ve henüz güvenilir bir kredi bağlantısı kurmamış bir Ripple cüzdanının diğer katılımcı cüzdanları ile para alışverişine girebilmesi için öncelikle bir miktar IOU alması gerekir. Sanchez ve diğ. (2016) bu işlevi gerçekleştirmek için Ripple ağındaki pek çok cüzdanın kredi bağlantısı kurma konusunda güvenebileceği ve tüm ağ tarafından tanınan ağ itibar derecesi yüksek olan gateway cüzdanları kullanıldığını ifade ederler. Gateway, kullanıcıların Ripple likidite havuzuna para yatırmasını ve buradan para çekmesini sağlayan herhangi bir kişi veya kuruluştur. Pratikte bankalara benzeyen gateway cüzdanları Ripple protokolü olarak bilinen bir küresel defteri paylaşırlar (Shailak, 2018). Gateway cüzdanlarının ağ kapsamındaki cüzdanların çoğunluğu ile kredi bağlantısı bulunur. Bu sayede, yeni oluşturulan bir Ripple cüzdanı ağdaki diğer cüzdanlarla gateway cüzdanı aracılığıyla kolaylıkla etkileşime geçebilir.

Ripple ağı, ağırlıklandırılmış yönlü bir G=(V,E) grafıdır. Graftaki köşeler (V) düğümlere ait cüzdanları sembolize ederken yönlü kenarlar (E) düğümler arası IOU kredi bağlantılarını gösterir. Ağırlandırılmış kenarlardaki IOU değeri, negatif değerler alamaz ve önceden tanımlanmış üst sınır değerini de aşamaz. IOU işlemlerinde ağdaki **u** düğümü **v** düğümüne $\boldsymbol{\beta}$ kadar IOU göndermek isterse, tüm kenarlar yönsüz düşünülerek **u** ile **v** arasında bir veya birden fazla düğümün yer aldığı bir yol oluşturulur. (**u**-u_1-u_2-…-u_n-**v**) İşlemi gerçekleştirmek için kenarların yönleri dikkate alınarak kenar IOU ağırlıkları güncellenir. u → v ile aynı yönde olan kenar IOU ağırlıkları $\boldsymbol{\beta}$ kadar arttırılırken ters yönde olan kenar IOU ağırlıkları $\boldsymbol{\beta}$ kadar azaltılır.

Şekil 8'de yer alan örnek Ripple grafında, düğümlere ait XRP bakiyeleri {} parantezleri ile gösterilmiştir. D → E düğümü arasında D'den E'ye doğru 50$ değerinde bir para transfer işlemi gerçekleştirilmek istenirse, iki düğüm arasındaki yol D ← F → E şeklinde kurulabilir. D ← F kenarı en azından 50$ kredi sahibidir ve F → E kenarının bir üst kredi sınırı yoktur. İşlem sonrası F → E kredi ağırlığı 110$ değerine arttırılırken D ← F kenarı iptal edilir.



Şekil 8: Örnek Ripple Grafı

Bir IOU işlemi için tek bir yol yerine kredi ağırlıkları toplamı gönderilecek para miktarına eşit ya da büyük olan birden çok yol da kullanılabilir. Örneğin, D → E düğümü arasında D'den E'ye doğru 70$ değerinde bir para transfer işlemi gerçekleştirilmek istenirse, transferin 50$ değerindeki kısmı yukarıda açıklanan D ← F → E yolu üzerinden 20$ eğerindeki kısmı D ← C → E yolu üzerinden gerçekleştirilebilir. İşlem kaydında yer alan path alanında kullanılan yolların listesi her yol için kullanılan kredi değeri ile birlikte kaydedilir. Aynı işlem kaydında yer alan amount alanında ise transfer edilen toplam para miktarı kaydedilir.

Ripple kripto para sisteminde ödeme işlemlerinden farklı beş işlem türü daha mevcuttur. Bir cüzdan hesabına ilişkin seçeneklerin tanımlanmasını sağlayan AccountSet işlemi, bir düğümün anahtarlarını belirleyen ve değiştirmeye yarayan SetRegularKey işlemi, döviz kuru değiştirme isteğinin belirtildiği ve işlem defterine kaydedildiği OfferCreate işlemi, işlem defterine kaydedilen döviz değiştirme isteğinin silinmesi için kullanılan OfferCancel işlemi ve iki cüzdan arasında güven bağlantısının kurulması ve değiştirilmesi için kullanılan TrustSet işlemi diğer işlem türlerini oluşturur.

Tüm altı işlem türü kayıtlarında temelde ortak kayıt alanları vardır. Ripple ağında yeni bir cüzdan hesabı oluşturmak için bir düğüm, önceden belirlenmiş asgari bir XRP değerinden büyük bir XRP tutarını varolmayan bir hesap numarasına gönderen bir ödeme işlemi yapmak zorundadır. Bu işlem gerçekleştirildikten sonra, Ripple defterine yeni bir AccountRoot eklenerek yeni bir hesap oluşturulmuş olur.

Ripple, aynı ad ile anılan kendine özgü Ripple mutabakatı protokolünü işletir. Ripple mutabakat protokolünde, ağ üzerinde yayınlanan işlemler, doğrulayıcı sunuculara (validating servers) ulaştığında öncelikle işlemi talep eden katılımcı düğümün açık/özel anahtarları kontrol edilerek bir doğrulama yapılır. Geçerli bir özel anahtar ile imzalanmış işlemler, CS (Candidate Set) olarak adlandırılan aday işlem havuzuna kaydedilir. Havuza alınan işlem için XRP işlemi olması durumunda yeterli kredi, IOU işlemi olası durumunda gönderici/alıcı arasındaki güven bağlantısı kontrolü gibi daha detaylı kontroller yapılır. Doğrulanan tüm

işlemlerin oluşturduğu bir hash ağacı ve bu ağacın kök hash değeri imzalanarak oluşturulan bir veri paketi, ağ üzerinde önerilen işlemler olarak yayınlanır.

Armknecht ve diğ. (2015) çalışmalarında bir doğrulayıcı sunucu (validating server) bir işlem öneri veri paketi aldığı zaman, öncelikle bu öneri paketini gönderen sunucunun kendi lokal diskinde kayıtlı Unique Node List (UNL) düğüm listesinde olup olmadığını kontrol ettiğini ifade etmişlerdir. UNL, sunucu için önemli bir veri kaynağıdır. Bir işlemin deftere kaydedilip edilmeyeceğine karar verirken sunucu düğümeleri UNL düğüm listesinden sorgular. UNL kontrolü sonrasında, gelen işlem öneri paketinde yer alan işlemlerin doğruluğu kontrol edilir. Sunucu lokal diski üzerinde kayıtlı bir işlem listesi (TL-Transaction List) ve her işlem için bir oylama listesi (VL-Voting List) bulunur. Doğrulanan işlem, TL işlem listesine kaydedilirken ilgili işlem için VL oylama listesinde bir güncelleme yapılır. UNL düğüm listesindeki düğümlerin %80 çoğunluğu tarafından doğrulanan işlem, CS aday işlem havuzundan çıkarılır ve defterde yer alan kayıtlı işlemlerle karşılaştırılarak çift harcama için kontrol edilir. Kontrolü geçen işlem, deftere işlenir ve gönderici/alıcı cüzdan bakiyeleri güncellenir. Her doğrulayıcı sunucu, kendi defter kopyalarını imzalayarak Ripple ağı üzerinde yayınlar. Bir doğrulayıcı sunucu, kendi UNL listesinde kayıtlı doğrulayıcı sunucu düğümlerin %80 çoğunluğu tarafından imzalanan defter kopyasını onaylanmış yeni güncel defter olarak kabul eder ve defteri kapatır. Defter kapatıldıktan sonra, her oluşan yeni defter için mutabakat protokol adımları tekrarlanır.

Ripple ağında, defterler birkaç saniyede bir oluşturulur. Bir Ripple defteri, işlem listesi, hesaplara ait ayarlar, toplam bakiye, güven bağlantıları bilgileri, bir zaman damgası (timestamp), bir defter numarası ve defterin doğrulanmış olup olmadığını gösteren bir statü belirteci alanlarından oluşur. En son doğrulanan defter, son kapatılmış defter olarak adlandırılır. Diğer yandan, eğer defter henüz onaylanmamış ise bu defter açık olarak işaretlenir.

Ripple kripto para sisteminde madencilik işlemi yapılmaz. Başlangıçta 100 milyar XRP yaratılmıştır. Bu meblağın, 20 milyarı Ripple geliştiricisi Ripple Labs kurucuları tarafından muhafaza edilmiştir. Yaklaşık %80'lik kısım ise piyasa yaratma faaliyetini teşvik etmeye, Ripple likiditesini artırmaya ve Ripple piyasalarını güçlendirmeye yönelik ayrılmıştır. Toplam Ripple'ın %0,2'lik kısmı hayır kurumlarına verilmiştir. Sistem, çeşitli projeler için aylık bazda 1 milyara kadar Ripple piyasaya sunuyor. Diğer bir özellik ise, sistemde işlem yapabilmek için her Ripple hesabının 20 XRP gibi küçük bir meblağa sahip olması gereklidir ve kullanıcı çok sayıda işlem yapması durumunda artan bir işlem ücreti öder.

1.6. KARŞILAŞTIRMA

Temel sistem özellikleri ve işleyiş mekanizmaları ana hatlarıyla açıklanan Bitcoin, Ethereum ve Ripple para birimleri Tablo 10'da karşılaştırmalı olarak incelenmiştir. Kripto para dünyasının en önde gelen bu üç para birimi benzer temeller üzerinde tasarlanmış olsalar da bu para sistemlerini birbirinden ayıran yönleri de mevcuttur.

Tablo 10: Kripto Para Birimleri Karşılaştırma Tablosu

ÖZELLİK	BITCOIN	ETHEREUM	RIPPLE
Tür	Blokzincir, kripto para birimi	Blokzincir, Dağıtılmış hesaplama	Gerçek zamanlı para transferi havale
Sembol	BTC, XBT	ETH	XRP
Alt Birimler	10^{-3} millibitcoin 10^{-6} bit 10^{-8} satoshi	10^{-9} Gwei 10^{-18} Wei	
Yaratıcı	Satoshi Nakamoto	Vitalik Buterin, Mihai Alisie, Anthony Di Iorio, Charles Hoskinson	Arthur Britto, Davis Schwartz, Ryan Fugger
Çıkış Tarihi	9.Ocak.2009	30.Temmuz.2015	2012
Mutabakat	İş İspatı (PoW)	İş İspatı (PoW)	FBA (Federated Byzantine Aggrement)
Şifreleme Fonksiyonu	SHA-256	ETHASH, KECCAK	-
Programlama Dili	C++	Go, C++, Rust	C++
İşletim Sistemi	Windows, MacOS X, Linux	Linux, Windows, macOS, POSIX, Raspbian	GNU/Linux (RHEL, CentOS, Ubuntu), Windows, OS X
Blok Numaralandırma	12,5 BTC	3 ETH	
Blok Oluşturma Periyodu	10 dakika	14-15 saniye	
Dolaşımdaki Birim	16.858.762	97.762.514	39.009.215.838
Tür	Blokzincir, kripto para birimi	Blokzincir, Dağıtılmış hesaplama	Gerçek zamanlı para transferi havale
Maksimum Birim	21 Milyon		100 Milyar
Tartışma Platformu	bitcointalk.org Yazılım üzerinde yapılan değişikliklerin paylaşılıp tartışılarak oylanabileceği platform	reddit.com/r/ethereum ethereum.org Platformları	xrpchat.com Bitcoin ile kıyaslandığında çok daha az şeffaf bir politika izlenen paylaşım ve tartışma platformu
Client Yazılım Konfigürasyonu	Cep telefonları gibi enerji kısıtlı cihazlarda az miktar bir ödeme ile çalışabilme olanağı	Cep telefonları gibi enerji kısıtlı cihazlarda çalıma imkanı	Küçük cihazlar için uygun güvenli bir sürümü bulunmamaktadır.

ÖZELLİK	BITCOIN	ETHEREUM	RIPPLE
Güvenlik Mekanizması	İşlem güvenliği PoW ile sağlanır. Yalnızca çok az sayıda düğüm tüm işlemlerin güvenliğini kontrol edebilir.	Serenity sürümü ile birlikte Proof of Stake kullanılması %51 saldırılarını engeller.	İşlem güvenliği doğrulayıcı sunucular ile sağlanır. Ripple Labs tüm işlemlerin güvenliğini kontrol edebilir.
Sistem Güncellemeleri	%51 madencilik gücü gerektiren soft ve hard çatallar ile gerçekleşir.		2 haftalık süre içinde %80 düzeyinde mutabakat arayışı ile gerçekleşir.
Mahremiyet ve Anonimlik	İşlemler farklı işlem çıktılarını girdi olarak kabul eder. Kimlikler korunur fakat işlemler sistemden açıkça paylaşıldığı için katılımcıların işlem geçmişi sızabilir.	Zero Knowledge Proof teknolojisi ile belirli bilgilerin matematiksel olarak gizlenmesini sağlar.	Ödemeler tek bir hesabı girdi olarak alırlar. Kimlikler korunmasına karşın işlem sırasında katılımcı işlem geçmişi sızabilir.
Sahiplik	Değişilikler konusunda anlaşma yapan bir grup Bitcoin Foundation	Ethereum Vakfı	Ripple Labs isimli özel bir firma

2. MALZEME VE YÖNTEM

Varolan bir blok zincir kullanılarak çok kısa sürede yeni bir kripto para sistemi ortaya çıkarılabilir. Zor olan tüm tasarım ve yazılım süreçleri baştan yapılmış yeni bir blok zinciri yaratmaktır. Bu çalışmada sıfırdan yeni bir blok zincir platformu oluşturmak için aşılması gereken tüm süreçler incelenmiş ve yeni bir model olarak hayata geçirilmiştir. İlk bloktan itibaren bir blok zincir defterini hazırlayan her aşamanın titizlikle ele alındığı detaylı bir çalışma yapılmıştır. Blok zincir ağı haberleşme protokolüne ait teknik detayların araştırılması, ağ üzerinde iletilecek veri paketlerini oluşturacak ağa özel mesajların tasarımı ve zincir ile etkileşimleri, kripto para sistemi temel yapılarının oluşturulması, para aktarım işleyişini sağlayacak modül yapıları, modüller arası etkileşimlerin belirlenmesi, kriptografi esaslarının gerçeklenmesi, sistem ön yüzlerinin hazırlanması yapılan çalışmanın ana başlıkları arasında yer almaktadır.

Hedeflenen model, Bitcoin eşleniği olacak sadeleştirilmiş yeni bir blok zincir platformu yaratmak ve bu zincir platformu üzerinde bir kripto parayı yürütebilmektir. Bu amaç doğrultusunda, Github kaynak depolama platformundaki Bitcoin modüllerine ait yazılım parçaları gereksinimlere uygun olarak incelenmiştir. Kripto para sistem bileşenleri hakkında ciddi boyutta araştırmalar yapılmış ve modele uyarlanması sağlanmıştır. Sistem genelinde yeniden kullanılabilecek işlem, blok ve mesaj gibi bileşenler dikkatlice tanımlanmış ve tüm bileşenler entegre edilerek hedef model ortaya çıkarılmıştır.

Model, Python ve Flask üzerinde uygulanan bir kripto para hizmet sağlayıcısı ve ödeme sistemidir. Hızlı, kolay ve sade bir kodlama, hızlı aksiyon alabilmek ve tekrarlanan kod kullanımı için web programlamada öne çıkan Python dili seçilmiştir. Veritabanı olarak piyasa lideri Bitcoin kripto para sistemi tarafından kullanılan ve öne çıkan özelliklerinden dolayı SQLite3 veritabanı tercih edilmiştir. Özel anahtarların tutulduğu dosyalara yüksek koruma sağlayabilme garantisi, taşınabilirlik konusunda optimum çözümler sunabilmesi ve eski sürümlere entegre olabilecek uygun değişiklikler yapması SQLite3 veritabanının başlıca olumlu özellikleridir.

Model kripto para zincirinde inşa edilen ağı kullanan katılımcılar arasındaki karşılıklı elektronik para aktarımı TCP/IP protokolü ve socket programlama sayesinde gerçekleştirilir. Ağ düğümlerinin İnternet ortamında 10000 nolu port üzerinden iletişim kurmalarını sağlayacak şekilde bir konfigürasyon yapılmıştır. Düğümlerin haberleşmesi ve kendi bilgilerini güncellemeleri ağa özel tasarlanan mesajların iletilmesi ile gerçekleştirilmektedir.

Model önyüz geliştirmeleri Flask framework, form yapıları, templates (html) kullanımı ile hayata geçirilmiştir. Katılımcıların kullanımına açık ön yüz sayfaları aşağıda listelenmiştir.

- Register - Cüzdan, adres ve anahtar oluşturma sayfası
    - New Keyset - Yeni bir anahtar çifti oluşturmak

- Login Sayfası - Welcome to Crypto World
    - Send Crypto - Para gönderme, işlem oluşturma, mempool ekleme, “tx” mesajı

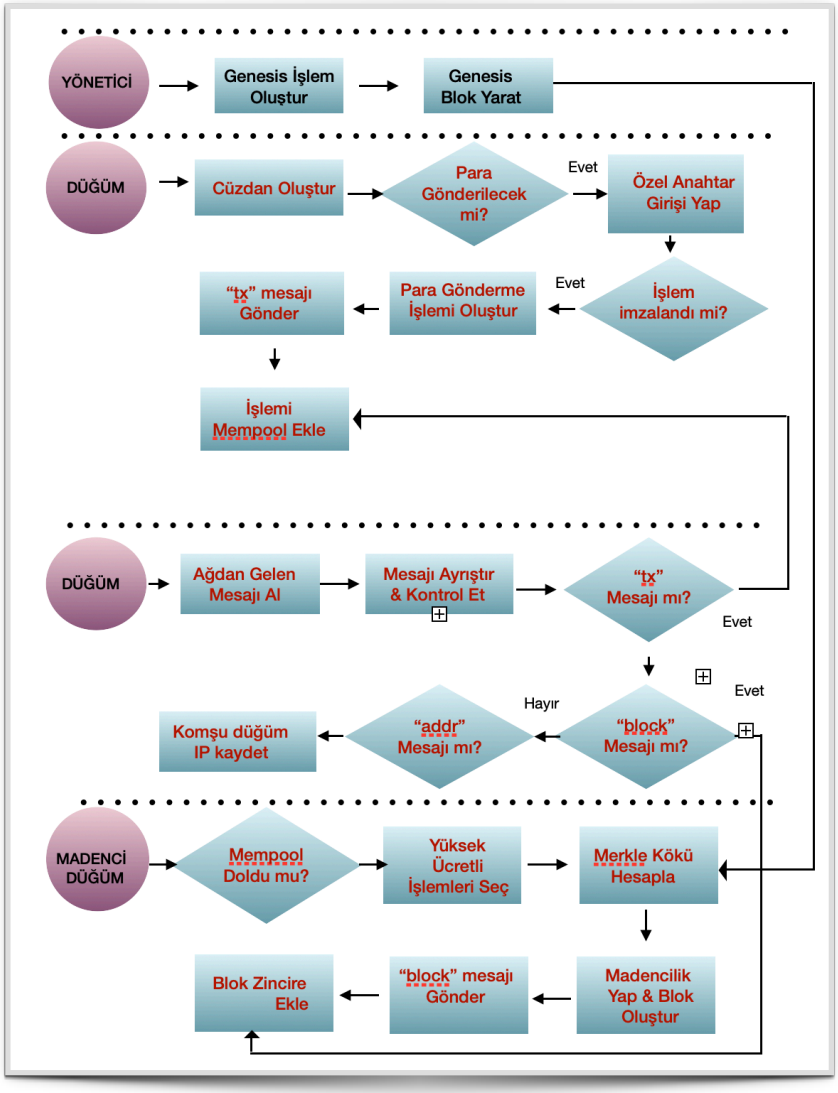
- Mining - İşlem doğrulama, ücret hesaplama, blok oluşturma, “block” mesajı
- Wallets - Cüzdan işlem detayları görüntüleme
- Mempool Explorer - Bloğa dahil edilmeyi bekleyen işlemleri görüntüleme
- Block Explorer - Zincirdeki blokları listeleme
- Genesis Operations - İlk blok oluşturma sayfası

Model zincirde blok yaratma süresinin kısaltılması hedeflenerek işlemlerin daha hızlı gerçekleşmesi ve onaylanması sağlanacaktır. Madencilik aşamasında blokları meydana getiren işlemleri seçme kriteri ve bloklarda yer alacak işlem sayısı yeni blok oluşumu süresini belirleyen başlıca unsurlardır. Önceki işlemlerden kalan harcanmamış çıktıların işlem girdi sayılarını azaltacak şekilde seçilmesini sağlamak işlemlerin imzalanma süresini kısaltacak ve model sistemimize hız kazandıracaktır. Optimum sayıda girdi kullanmanın işlem ücretleri üzerinde de olumlu etkileri olacaktır.

Model sistemimiz geleneksel kripto para birimlerinden daha düşük enerji ve güç tüketimine sahip olacak ve dolayısı ile çok daha düşük bir karbon ayak izi bırakacaktır. İş ispatı mutabakat algoritmasının işleyeceği sistemde belirli bir kapasiteye sahip madenci düğümlere sırasıyla verilecek yetki ile nonce hesaplaması yapılacaktır. Model sistemde zorluk derecesi küçültülerek bir nonce hesaplama yapılması enerjinin verimli kullanılmasını sağlayacaktır.

2.1 MODEL İŞLEYİŞİ

Her katılımcı düğümün küçük bir otorite olduğu blok zincir modelinde düğümlerin güncel kalabilmesi ağ üzerinde iletilen mesajlar sayesinde gerçekleşir. Ağ katılımcıları kendi cihazlarında sakladıkları veritabanına kaydettikleri veya kendilerine iletilen para aktarımına ilişkin verileri belli türlerde paketlenmiş mesajları kullanarak diğer katılımcılara iletirler. Her düğüm öncelikle ağ tarafından kendine iletilen mesajı türüne göre ayrıştırır, mesajın içerdiği veriler için önceden tanımlanmış kontrolleri sağlar ve kendi cihazındaki blok zincir veritabanı ilgili dosyasına mesajda yer alan verileri işler. Yazılım tüm mesaj akışını yönlendirir.



Şekil 9: Model İşleyişi

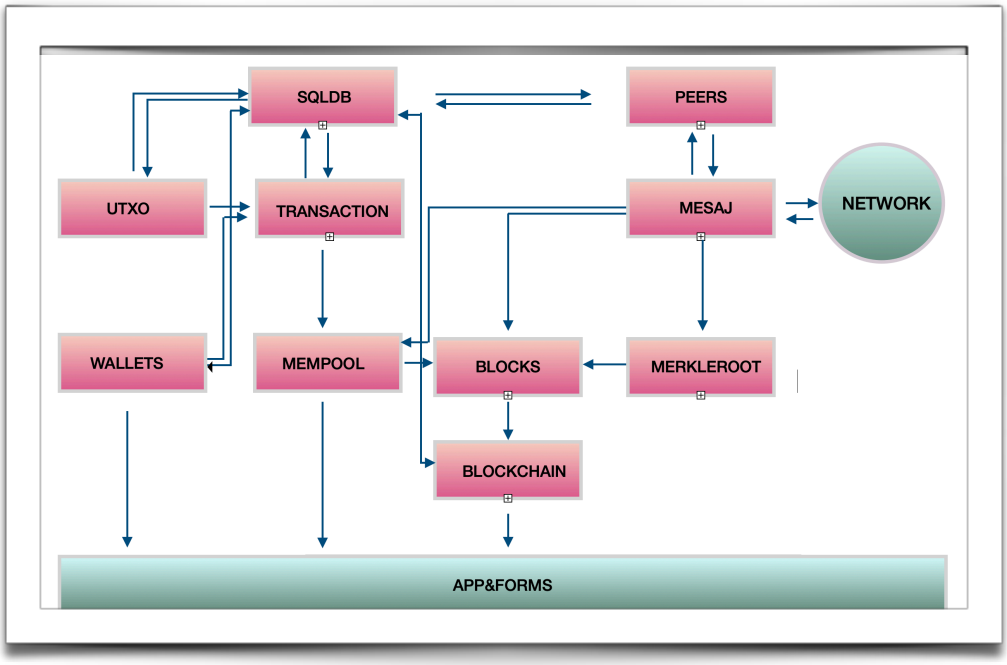
Zincirin ilk halkası Genesis blok olarak tanımlanmıştır. Bu özel blok, zincire sonradan eklenecek diğer bloklardan farklı kriterler dikkate alınarak yaratıldığından blok zinciri başlatma yetkisi olan yönetici düğüm tarafından kullanılacak bir menü adımından tanımlanır. Ağa katılmayı talep eden düğümlerin modele bağlanmak ve giriş yapmak için öncelikli olarak kripto varlıklarını saklayacakları cüzdanları ve gizliliği sağlayacak kriptografik anahtarları oluşturması zorunludur. Cüzdan oluşturma, katılımcının kripto adres oluşturma adımını da kapsayan bir süreçtir. Elektronik varlık transferleri katılımcı cüzdanlarına atanan kripto adresler aracılığıyla yapılır. Bitcoin ve diğer kripto uygulamalarına benzer şekilde modelde, kripto varlıkların transfer işlemleri doğrulanarak bloklara eklenmeden önce saklandığı mempool yapısı mevcuttur. Mempool'da onaylanmayı bekleyen işlemlere ait veriler ağdaki diğer düğümlere İnternet üzerinde TCP/IP iletişim protokolü çalışma prensiplerine uygun bir mesajlaşma ile aktarılır. Mempool kuyruğunun belli bir işlem sayısına erişmesi sonucunda madencilik süreci başlatılır. Madencilik, katılımcının isteği ile başlatılır. İşlem ücretleri hesaplanır, blok için en yüksek ücretli işlemler seçilir, işlemler doğrulanır, blok oluşturulur,

zincire eklenir ve ağa blok mesajı iletilir. Tüm madencilik aşamaları otomatik şekilde yürütülür. İşleyiş ana hatlarıyla Şekil 9'da resmedilmiştir.

2.2. MODEL BİLEŞENLERİ

Model, ilk kripto para olma özelliği taşıyan ve yaygın şekilde kullanılan Bitcoin blok zincir platformu yazılım parçaları incelenerek tasarlanmıştır. Uygulama altyapısını oluşturan bileşenler tespit edilerek modelin işleyiş şekli düzenlenmiş ve bileşenler arası karşılılıklı etkileşimler belirlenmiştir. Bitcoin platformuna eşlenik bir kripto zincirini yaratmak için gereken tüm bileşenler sırasıyla geliştirilmiş ve uyum içinde çalışmaları için gerekli adaptasyonlar yapılmıştır. Kripto para zincirinin yürütülmesini sağlayan ana bileşenler aşağıda listelenmiştir.

- Wallet (Cüzdan) - Kriptografik anahtarlar oluşturma, imzalama ve doğrulama
- Mesaj - Ağa özel mesaj oluşturma ve ayrıştırma
- Network - Bağlantı kurma, mesaj gönderme ve alma
- Transaction - İşlem oluşturma
- Mempool - Bekleme havuzu
- Blocks - Blok oluşturma
- Merkletree - Merkle kökü oluşturma ve ispatı
- Blockchain - Blok zincire ekleme
- UTXO - İşlem girdileri
- Peers - Komşu düğüm tanımları
- SQLdb - Veritabanı işlemleri
- Forms - Önyüz alt bileşenleri



Şekil 10: Bileşenler Arası İlişki Şeması

Şekil 10'da paylaşılan sistem bileşenleri arasındaki ilişki-etkileşim diyagramı sistem genel davranışı hakkında bilgi vermektedir. Şemada gösterilen her bileşenin oluşturulma prensipleri, işlev ve yapısal özellikleri, birbirleriyle olan etkileşimleri geniş kapsamda açıklanmıştır.

2.2.1. Wallet (Cüzdan)

Kriptografi üzerine kurulmuş olan para modelinde, cüzdanlar kripto dünyasına açılan ana kapıdır. Banka müşterilerinin sahip olduğu hesap cüzdanlarına benzer kripto cüzdanlar, kriptografik anahtarların saklandığı veri alanlarıdır. Kripto dünyasının temel taşı olan açık ve özel anahtarlar dışarıya karşı koruma sağlar. Bundan dolayı anahtarların oluşturulması ve saklanması büyük bir önem taşır.

Modelde, özel ve açık anahtarlar Eliptik Eğri kriptografisi kullanılarak oluşturulmaktadır. Gün geçtikçe küçülen cihazlara ve artan bilgi güvenliği ihtiyacına daha üst düzey bir cevap veren bu yöntemi gerçekleştirmek için Python yazılım platformunun sunduğu kullanımı nispeten kolay ECDSA algoritma uygulaması kullanılmıştır. Algoritmanın yüklenmesi için <pip install cryptography> ve <pip install ecdsa> komutlarından yararlanılmıştır. Python kriptografi kütüphanesinde bulunan ve alt seviyede bir kullanım gerektiren hazmat katmanı fonksiyon seti, anahtarları üretmek, veritabanına kaydetmek, veritabanından okumak, işlem verisini imzalamak ve imzaları doğrulamak için kullanılmıştır. Adı geçen işlem adımlarını yürütebilmek için şu kütüphanelerden faydalanılmıştır.

- binascii
- random
- cryptograhy.hazmat.backends kütüphanesi default_backend algoritması
- cryptograhy.hazmat.primitives kütüphanesi hashes, serialization algoritmaları
- cryptograhy.hazmat.primitives.asymmetric kütüphanesi ec algoritması
- cryptograhy.exceptions kütüphanesi InvalidSignature algoritması

Modelde katılımcı açık (public) anahtarı ve adresi, özel (private) anahtardan türetilecek şekilde tasarlanmıştır. İlk olarak, 10^{75} ve 10^{76} aralığında sistem tarafından rastgele belirlenen bir sayı ve ilgili eliptik eğri fonksiyonu ile özel anahtar türetilir. Özel anahtarın bilinmesi durumunda, katılımcı açık anahtarı ve adresi oluşturulabildiğinden modelde veritabanında sadece özel anahtarlar saklanır. Eliptik eğri olarak daha hızlı sonuçlar ürettiği ve rastgele seçimleri desteklediği için SECP256R1 seçilmiştir.

Python kriptografi kütüphanesinde anahtarlar bir çeşit objedir ve bu objeler ancak pem uzantılı özel bir dosyada saklanmak zorundadır. Aksi takdirde, anahtar değerlerine erişmek mümkün olmamaktadır. PEM formatındaki bir özel anahtar örneği paylaşılmıştır.

-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIBJ+V61HWT/j9RFqy3e6Dzg5tJTYNWFO0SWp9rLT3HZ2oAoGCCqGSM49
AwEHoUQDQgAE6wUsgrW40kKB91FjfptNcHh+Tt9S/A7Pfq+cWynS+u51RYwLGDyM
KcMKd4+OwPG8AJB2G2lkax3f+8Au4gqKXQ==
-----END EC PRIVATE KEY——

Bir özel anahtar objesini pem dosyasına kaydetmek için özel anahtarın aşağıdaki komutlar kullanılarak pem formatına dönüştürülmesi gerekir. PEM formatındaki özel anahtar aslında belli ayıraçlar (başlık ve kuyruk) ile belirlenerek kodlanmış bir metindir.

```
pem = pk.private_bytes(encoding=serialization.Encoding.PEM,
format=serialization.PrivateFormat.TraditionalOpenSSL,
encryption_algorithm=serialization.NoEncryption())
```

Bir başlık ve bir alt bilgi (footer) arasına sıkıştırılan özel anahtarı pem dosyasına kaydetmek için dosya append binary (open(pemfile,'ab')) modunda açılırken anahtarı geri yüklemek için dosya read binary (open(pemfile, 'rb') modunda açılmalıdır. Özel anahtarın tekrar yüklenebilmesi pem format kodunun çözümlenmesini gerektirir. Başlık ve alt bilgi kısımlarından ayrılan metin için uygulanan aşağıdaki komutlar sırasıyla özel anahtar obje değerine ve özel anahtar heksadesimal değerine erişimi sağlanır.

```
private_key_obj = serialization.load_pem_private_key(pk, None, default_backend())
private_key_hex = private_key_obj.private_numbers().private_value.to_bytes(32, 'big').hex()
```

Eliptik Eğri algoritması ECDSA, oluşturulan özel anahtar objesi ile ilişkili açık anahtarı da yaratma imkanını verir. Modelde, açık anahtarlar daha küçük bir alan işgal etmeleri sebebiyle sıkıştırılmış (compressed) formunda kullanılmıştır.

ec.EllipticCurvePublicKey.from_encoded_point(curve, pubkey) - byte değerinden açık anahtar objesi oluşturma
pubkey.public_bytes(serialization.Encoding.X962,
serialization.PublicFormat.CompressedPoint).hex() - açık anahtar objesi karşılık gelen string değeri oluşturma

Özel anahtar objesi kullanılarak elde edilen açık anahtar özel anahtarın sahibi katılımcı için bir kripto adresinin oluşturulması için yeterlidir. Bir açık anahtar, eğri üzerindeki bir noktanın (x,y) koordinatlarıdır. Sıkıştırılmamış (uncompressed) açık anahtarlar 0x04 ile başlayan 65 byte uzunluğundadır. Sıkıştırılmamış açık anahtarlarda, sadece x koordinatı kodlandığı için açık anahtar değerinin tek ya da çift olması y koordinatını tayin eder ve anahtar 33 byte uzunluğundadır. Kripto adresinin oluşturma adımları şu şekilde özetlenebilir.

1. Sıkıştılmamış bir açık anahtar kullanılıyorsa herhangi bir ekleme yapmadan 3. adıma devam edilir.
2. Sıkıştırılmış anahtar için anahtar değerinin çift ve tek sayı olmasına göre anahtarın başına 0x02 ya da 0x03 byte değeri eklenir.
3. Heksadesimal açık anahtar byte dönüşüme yapılır ve başına 0x00 eklenir.
4. Elde edilen değer için SHA-256 hash algoritması işletilir.
5. Çıkan hash sonucu üzerinde RIPEMD-160 hash algoritması işletilir.
6. Sonuç değerin başına versiyon 0x00 byte eklenerek elde edilen genişletilmiş RIP değeri tekrar SHA-256 uygulanır.

7. Bir kez daha SHA-256 ile hash edilir ve heksadecimal değerin ilk 8 karakteri checksum olarak atanır.
8. Genişletilmiş RIP değerine checksum eklenerek base58 string formatına çevrilir.

Adres oluşturma fonksiyonunda Python hashlib kütüphanesi SHA-256, RIPEMD-160 algoritmaları ve base58 kütüphanesi base58 string formatına dönüşüm algoritması kullanılmıştır.

Katılımcı cüzdanları SQLite3 veritabanında saklanır ve yalnızca özel anahtarları muhafaza ederler. Açık anahtar ve kripto adres özel anahtardan türetildiği için veritabanına kaydetmeye gerek duyulmamıştır. Yeni bir cüzdan oluşturmak isteyen katılımcı, anahtar çiftini model önyüzü ana ekrandan <Register> tuşu ile geçilen ekranda <New Keyset> ile türetir ve kaydeder. Kayıt sonrası artık yeni bir cüzdan yaratılmıştır ve özel anahtarlar koruma altına alınmış olur. Model ön yüzünde cüzdan anahtarları oluşturma ekran örneği Şekil 11'de paylaşılmıştır.



Şekil 11: Özel ve Açık Anahtar Oluşturma Ekranı

Kriptografi işleyişi çerçevesinde, verilerin imzalanması ve doğrulanması için cüzdana ait özel ve açık anahtarlara ihtiyaç duyulur. İmzalama ve doğrulama, wallet (cüzdan) modülünden çağrılan iki farklı fonksiyon ile gerçekleştirilir. Veri imzalama ve doğrulama fonksiyonları cüzdan modülünde yer alır. İmzalama ve doğrulama algoritması eliptik curve ECDSA (hashes.SHA-256) olarak belirlenir ve özel anahtar objesi sign özelliği ile istenilen veriler imzalanır. Doğrulama için açık anahtar objesi verify özelliği kullanılır.

2.2.2. Message (Mesaj)

Mesajlaşma, dağıtılmış sistem alt yapısının vazgeçilmez bir unsurudur. Düğüm cihazlar güncel kalabilmek için birbirleriyle veri alış verişi yapmak zorundadır. Veri iletimi, sisteme özel tanımlanan mesajlar üzerinden gerçekleştilir. Çalışmaları yapılan modelde, geliştirilen mesaj türleri version, verack, addr, getaddr, inv, tx, block mesajlarıdır. Mesajları paketlenmiş halde gönderime hazır hale getiren rutinler mevcuttur.

Tüm mesajlar başlık (header) ve gövde (body) kısımlarından oluşur. Başlık kısmı, her mesaj türü için standart bir yapıdadır. Mesaj başlığı, magic değeri, checksum, mesaj türü, payload

uzunluğu alanlarından oluşur. Magic değer, sistem test ve production ortamları için tanımlanan sabit magic_value = 0xd9b4bef9 değeridir. Payload değeri, checksum hesaplanırken kullanılır ve payload uzunluk değeri mesaj başlığına eklenir. Çoğu mesaj türü için payload değeri mesaj gövde kısmıdır, ancak addr mesajı için mesajda yer alan toplam düğüm IP sayısıdır. Checksum, payload verisinin byte cinsinden iki kez hash edilmesi sonucu oluşan değerin ilk dört karakterinden meydana gelir.

Düğüm çalıştırdığı yazılım versiyonunu iletişim kurduğu karşı tarafa versiyon mesajı ile iletir. Versiyon mesajında dikkat edilmesi gereken husus, mesajı ileten ve karşı taraf IP adreslerinin 32-bit paketlenmiş formata çevrilmesi zorunluluğudur. Model kapsamında, create_network_address() fonksiyonu ile paketlenen IP adresleri, gelen mesajlardaki paketlenmiş adres, split_network_address() fonksiyonu ile IP adres formatına dönüştürülür.

```
def create_message_version(peer_ip):
   body = struct.pack('<LQQ26s26sQL',
                      version, service, timestamp, peer_address,
                      local_address, nonce, height)

def create_network_address(ip, port):
   return struct.pack('>8s16sh', b'\x01', (bytearray.fromhex("00000000000000000000ffff") +
                                           socket.inet_aton(ip)), port)

def split_message_version(msgbody):
   (version, service, timestamp, local_address, peer_address, nonce, height) =
               struct.unpack('<LQQ26s26sQL', msgbody)

def split_network_address(network_address):
   (service, addr, port) = struct.unpack('>8s16sh', network_address)
   aton_value = addr[12:16]
   ip = socket.inet_ntoa(aton_value)
   return ip, port
```

“getaddr” mesajı, düğümlerin komşu düğüm IP adreslerini elde etmek için gönderdikleri mesaj türüdür. Mesaj yalnızca başlık kısmından oluşur. getaddr mesajına cevap olan mesaj ise “addr” mesajıdır. Düğüm, peers veritabanında kaydettiği aktif komşu düğümlerin IP adres ve port numaralarını network ağ formatına dönüştürerek bu mesajda gönderir.

Diğer mesaj türü olan “inv” mesaj, mempool’da bekleyen henüz doğrulanmamış işlem hash bilgilerinin gönderildiği mesajdır. Bu mesajı alan düğümler, sonrasında getdata mesajını göndererek işlem ve blok detay bilgilerini içeren tx ve block mesajlarını beklerler. Modelde, yeni bir işlem yaratıldığında tx mesajı ve yeni bir blok yaratıldığında ise block mesajı gönderilir. İşlem tx mesajı gövdesi, ham işlem olarak da adlandırılan paketlenmiş işlem verisinden oluşur ve bu dönüşüm transaction modülünde yapılır. Blok verilerinin paketlenmiş hale getirilmesi de blockchain modülünde bulunan form_message_block() fonksiyonunda yerine getirilir. Mesaj başlıklarının eklenmesi ve bu şekilde mesajların son halini alması ise message modülünün görevidir.

Düğüm, ağdan gelen “tx” mesajı pack edilmiş formattadır ve işlemi mempool havuzuna ekleyebilmek için mesajı önce unpack etmek ve alanlarına ayrıştırmak gerekir. Mesaj gövde kısmı split_message_tx() fonksiyonu ile “tx” mesaj paket formatına uygun olacak şekilde yazılım tarafından ayrıştırılır. Özellikle işleme ait girdi ve çıktıların düzgün şekilde ayırt edilmesi önemlidir. Çoklu girdi ve çıktılar için while döngü yapısı kullanılır. Elde edilen girdiler txinput sınıfı formatında bir diziye kaydedilirken çıktılar txoutput sınıfı formatlı bir diziye eklenir. Mesaj alanlarına ayrıştırıldıktan sonra tüm mesaj alanları ile bir işlem sınıfı nesnesi oluşturulur ve nesne mempool veritabanına yazılır. Gelen tx mesajı ile sadece mempool veritabanına işlem kaydedilebilir. Ağdan alınan “tx” mesajları ile gelen işlemler mempool bekleme havuzuna eklenirken herhangi bir kontrol yapılmaz. Tüm işlem doğrulama ve kontrolleri madencilik aşamasında yürütülür ve uygunluk kontrolünden geçen işlemler yeni bloklara dahil edilir.

Bir “block" mesajı ise, bloğa ait versiyon, bağlı olduğu önceki blok hash değeri, merkle kök değeri, blok oluşturma zamanı (Unix), zorluk derecesi, madencilik nonce değeri, o blokta yer alan işlemlerin toplam sayısı ve ham işlem formatındaki işlemlerden oluşur. Mesaj verilerinin her biri kendi veri türlerine göre paketlenir. Düğüm, bir block mesajı aldığında, gelen mesaj paketini split_message_block() fonksiyonunda çözümler. Bloğun içerdiği işlemlerin her biri ham formatta paketlenmiş olduğu için her işlem için txid kimlik bilgisi kolaylıkla oluşturulur. İşlem belirleyicisi olarak tanımlanabilecek txid, ham formattaki işlemin SHA-256 fonksiyonu sonucunda oluşan 64 byte uzunluğunda bir heksadesimal değerdir. Aynı zamanda split_message_tx() fonksiyonu ile alanlarına ayrıştırılır. Ağdan gelen bloğun doğruluğunu ispat etmek önemlidir, bu sebeple blokta yer alan tüm işlemlerin hesaplana txid kimlik verileri kullanılarak merkle kök değeri hesaplanır. Elde edilen merkle kök değeri, mesajın ilk kısmında yer alan merkle kök değeri ile eşit ise, blok blokların yer aldığı blockchain veri tabanına kaydedilir, aksi takdirde rededilir.

2.2.3. Network

TCP/IP protokol esaslarının gerçeklendiği ve socket programlamanın yapıldığı ağın temelini oluşturan ana modül network olarak adlandırılmıştır. OpenNetworkConnection() ve SocketHandler() rutinleri iki farklı thread olarak modelin paralel şekilde çalışmasını yürütür. OpenNetworkConnection, peers veritabanında kayıtlı adreslere bağlantının kurulduğu fonksiyondur. Herhangi bir kesinti sırasında bağlı olunan adreslerin kaydedildiği anchors veritabanının uygulanması planlanmıştır. Eğer anchors veri içermiyorda, öncelikle peers veritabanı adreslerine yönelinir. Ağa ilk kez katılan yeni katılımcı düğümler için komşu düğüm adresleri bulunmadığından, sistem konfigürasyon dosyasında önceden tanımlanmış seed adı verilen belirli adresler için bağlantılar oluşturulur. Tüm komşu düğüm bağlantıları socket üzerinden kurulur ve cnode sınıfı nesnesine kaydedilir. Düğümlerle oluşturulan bağlantılar, düğüm ve bağlantı bilgilerinin tanımlı olduğu düğümler (nodes) dizisinde saklanır ve uygulama çalıştığı sürece bu dizi üzerinden işlem yapılır.

```
node=cnode(id, sock, addrConnect, addrbind,
                conntype, recvmsg, sendmsg, sendbytes,
                recvbytes, lastsend, lastrecv, disconnect)
```

Modelde bağlantı kurulacak port 10000 olarak seçilmiştir. Port bilgisi konfigürasyon dosyasında tanımlanır. Oluşturulan her bağlantı için bir düğüm sınıfı (cnode) nesnesi yaratılır ve düğüm bağlantı dizisine eklenir. Ekleme sırasında, initializenode() fonksiyonu ile düğüm bağlantıları için belirlenen kuyruklar tanımlanır. Kuyruk (queue) yapısı, gönderilen ve alınan mesajlar için kullanılır. Düğüm dizisindeki her bağlantı için sendmsg ve recvmsg kuyrukları bulunur. Oluşturulan mesajlar, pushmessage() fonksiyonu ile sendmsg kuyruğuna yazılır. Bağlantılar üzerinden gönderilen ve alınan mesajlar için toplam byte sayısı ve son mesaj alınma ve gönderilme zamanı verileri saklanır.

Düğümün dinleme işlemini BindListenPort() fonksiyonu başlatır. AcceptConnection() ile kabul edilen bağlantılar için bağlantı dizisine ekleme yapılır. Bağlantı dizisindeki her düğüm için gönderilecek mesaj varsa socketsenddata() fonksiyonu ile bağlantıda kayıtlı socket üzerinden mesaj gönderimi gerçekleştirilir. Kabul edilen bağlantılardan paketlenmiş mesaj gelmesi durumunda, process_message() fonksiyonunda gelen mesajın başlık ve gövde bölümleri birbirinden ayrılarak işleme alınır. Öncelikle, başlıkta yer alan mesaj türü, magic değeri ayrıştırılarak konfigürasyondaki tanımlı değerlerle kontrol edilir. Ayrıca, ilgili mesaj verileri ile checksum hesaplanır ve mesaj başlığında yer alan checksum değeri ile karşılaştırılır. Checksum değerlerinin farklı olması durumunda, mesaj geçersizdir. Gelen mesaj başlıkları veritabanında saklanır.

2.2.4. Transaction (İşlem)

Kripto para modelinin temel taşı olan işlem yapısını düzenleyen modüldür. İşlem (transaction) sınıfını tanımlar, işlenmemiş (raw) işlem formatındaki işlem verisini oluşturur, imzalar ve mesaj olarak gönderilmesini tetikler.

İşlem, bir miktar kripto para meblağının bir cüzdan adresinden diğer bir cüzdan adresine aktarımıdır. Bir işlem, bir kişinin yalnızca cebindeki nakit banknot ve madeni paraları kullanarak bir başka kişiye belli bir miktar para ödemesi ve para üstü gerekiyorsa yine nakit şekilde geri ödeme alması olarak özetlenebilir. Bu işleyiş kripto paraya uygulandığında, işlemi oluşturacak girdi para ve çıktı para değerlerinin tanımlanması önceliklidir.

İşlem girdi değerleri, gönderen cüzdan adresine ait harcanmamış (utxo) kayıtlarından müsaitlik durumuna seçilen kayıtların bilgilerini içerir. Çıktı bilgileri ise, ödemenin ve varsa geri ödemenin yapılacağı adres ve tutar bilgilerinin yapılacağı verileri içerir. Model yazılımında, işlem sınıfının yanısıra, girdi ve çıktı verileri için de iki farklı sınıf tanımı bulunmaktadır. Bir işlem sınıfı aşağıdaki alanlardan oluşmaktadır:

version - Model versiyonu
status - Bekleyen işlemler için ‘P’ (Pending) ataması
no_inputs - İşlem girdilerinin sayısı
txin - İşlem girdileri (string olarak)
no_outputs - İşlem çıktı sayısı
txout - İşlem çıktıları (string olarak)
txtime - İşlem oluşma zamanı

İşlem girdi sınıf yapısı, işlem tutarının karşılanacağı önceki işlemlerin transaction id (intxid), o işlemdeki çıktının sıra numarası (invout), girdi değeri için oluşturulan imza ve cüzdan adres sahibini açık anahtar bilgilerinin birleştirilmesi ile oluşturulan (sigpubkey) ve sigpubkey alanının uzunluğu olan (lenscript) alanlarından oluşan bir yapıdadır. İşlem çıktı sınıf yapısı, varlık aktarımı yapılan adresi ve kalan varlık varsa döneceği adresi (outaddress), adres uzunluğunu (outaddresslength) ve adreslere aktarılacak tutarları (outamount) içerir.

```
class txinput():
        def __init__(self, intxid, invout, lenscript, sigpubkey)
class txoutput():
        def __init__(self, outaddresslength, outaddress, outamount)
```

Bir işlemi tanımlayan ve diğer tüm işlemlerden ayıran veri o işleme ait transaction id (txid) kimlik bilgisidir. İşlem txid alanı, bytearray formunda oluşturulan işlenmemiş (raw) işlem verisinin heksadecimal karşılığının SHA-256 hash algoritması kullanılarak iki kez işlenmesi sonrası elde edilen 64 byte uzunluğunda bir değerdir.

Örnek txid: “c9e9e7069e2c48693614f09dab699faf7a86f5b8823600a85e9ffc351f52acf8”

Model uygulamasında, txid bilgisi her ne kadar transaction sınıf tanımında yer alsa da txid veritabanına kaydedilmez. Amaç, işlemlere ait ayırt edici bu benzersiz bilginin erişimine engel olmak ve işlem verilerini korumaktır. Yazılım içinde ihtiyaç duyulan yerlerde txid, formtxid() fonksiyonu çağrılarak hesaplanır.

İşlem (transaction) sınıfı tanımında yer alan txin alanı txinput sınıfı verinin, txout alanı ise txoutput sınıfı verinin dizi olarak saklandığı alandır. İşleme ait tüm girdi değerleri txinput sınıfı için tanımlanan addinput() fonksiyonu ile bir diziye eklenir ve işlem sınıfı txin alanına aktarılır. Aynı şekilde, işlem çıktı değerleri addoutput() fonksiyonu ile disiye eklenir ve işlem sınıfı txout alanına taşınır. SQLite3 veritabanı tablolarına dizi kaydedilmesine imkan vermediği için, txin ve txout alanlarının veritabanına aktarımında, dizi elemanları string haline dönüştürülerek saklanır. Veri tabanı okumalarında ise, string veriden dizi dönüşümü için algoritmalar mevcuttur.

- Tek girdiye sahip işlem “txin” sınıfı string değeri

```
txid:3a5098db57bd6bbc07a32ac97b7870a4e9fd547d657900df91527e56cb82654f
vout:0
lscr:208
sigp:30450220507ac970fa86de16b2ea062beee28a1ff9c23145943b7b3ffed67b20646a
2c330221008e2cbae74d46d8fca6eb3c46632e5feb32f42ed8f2e998fe37e4fe56707f081
203614bb14a42327054666767dc65aedbc0b9ba64a99521a564ff6c454220848b66
```

- Tek çıktıya sahip işlem “txout” sınıfı string değeri

```
alen:34
addr:1L3B3QfBtj44vVELraggjpSiBhDAT9v9Jj
amnt:1.0
```

alen:34
addr:18gZhUfW5gtxuiLF824EkXn1bSJv6sY7z3 amnt:8.0

Örnek işlem, tek bir girdi alarak oluşmuştur. Önceden gerçekleştirilen txid="3a5098db57bd6bbc07a32ac97b7870a4e9fd547d657900df91527e56cb82654f" hash değerine sahip işlemin vout = 0 yani ilk çıktısı (çıktı sıra numarası 0 ile başlar) yeni işlemin girdisi olmuştur. Bu girdilerin işlemin doğrulanması sırasında harcanabilir olarak işaretlenmesi için kilitlerini açacak olan komut (script) ise sigp kısaltması ile veride yer alan sigpubkey alanında saklanır. Girdilerin imzalanması işlemleri doğrulamak ve bloklara dahil etmek için önemli bir aşamadır. İşlem girdilerinin için imzalama öncesinde imzaya uygun hale dönüştürülmesi sağlanmalıdır. Bu uygunluk prepareinputforsign() fonksiyonu ile gerçekleştirilir.

- Girdi sigpubkey ve lenscript alanlarına sırasıyla aynı işlemin txout alanındaki ilk çıktı adres değeri (txout.outaddress[0]) ve uzunluk(txout.outaddresslength[0]) ataması yapılır.
- Değişiklik yapılan girdiden sonrasında txin dizininde başka girdiler mevcutsa, o girdi sigpubkey ve lenscript değerileri boşluk ve "0" değerlerine eşitlenir.
- Transaction (işlem) kaydı için işlenmemiş forma dönüştürülür (rawtx).
- rawtx heksadesimal karşılık gelen değere iki kez SHA-256 hash algoritması uygulanır.

İşlemdeki her girdi değeri için prepareinputforsign() fonksiyonu çağrılarak girdi değerleri imzaya hazırlanır. Fonksiyon sonucu dönen değer gönderen tarafın özel anahtarı ile imzalanır. İmza sonucu oluşan imza (signature), gönderen tarafın açık anahtarı (pubkey) ile birleştirilerek sigpubkey ve lenscript alanları elde edilir. Girdi imzaları tamamlandıktan sonra, createrawtransaction() fonksiyonu çağrılarak işlenmemiş ham işlem (rawtx) oluşturulur. Ham işlem, işlem versiyonu, no_inputs, txin, no_outputs, txout, txtime verilerinin struct kütüphanesi pack komutu ile belirlenen formatlara göre paketlenmesi sonucunda oluşur. Oluşturulan işlemi ağ üzerinde "tx" mesajı ile gönderebilmek için pack edilmesi önemlidir.

- versiyon ve girdi sayısı için pack
  rawtx = struct.pack('<2sB', byteversion, self.no_inputs)
- her bir girdi (txin) için pack yap ve ham işleme ekle
  rawtx += struct.pack("<32sLH", byteintxid, self.txin.invout[i], self.txin.lenscript[i])
- her bir çıktı (txout) için pack yap ve ham işleme ekle
  rawtx+=struct.pack('<H36sf', self.txout.outaddresslength[i], self.txout.outaddress[i].encode('utf-8'), self.txout.outamount[i])

2.2.5. Mempool

Kripto para gönderme işlemi doğrulanarak bir bloğa dahil edilmeden önce mempool veritabanına kaydedilir. Modelde kripto para gönderme, ana ekrandan yönlenen ve ana ekran <Send Crypto> menü bağlantısı ile gerçekleştirilir. Arka planda atılması gereken ilk adım, txin ve txout dizilerine atanacak girdi ve çıktıların belirlenmesidir. Yazılımda, utxo modülünden çağrılan getavailablecoins() fonksiyonunun döndürdüğü utxo kayıtları (coins) txin dizisine eklenirken, aynı fonksiyondan dönen para üstü (remainder) kalan bakiyeye göre

txout dizisi oluşturulur. Seçilen utxo kayıtları, işlemler doğrulanana kadar diğer işlemler tarafından harcanmaya karşı bloke edilmek zorundadır. Bu veriler transaction (işlem) sınıfı nesnesi oluşturmak için yeterlidir.

Oluşturulan işlem sınıfı nesnesi için sırasıyla bytearray olarak paketlenmiş ham işlem (rawtx) yaratılır (createrawtransaction) ve bu ham işlem imzalanır (signrawtransaction). İmzalanmış işlem diğer düğümlere gönderilmek üzere hazır hale gelmiş demektir. İlgili fonksiyon (sendrawtransaction) çağrılarak yeni işlem diğer düğümlere gönderilir ve işlem için kimlik numarası niteliğindeki txid verisi oluşturulur. Son adım olarak işlem, mempool veritabanına kaydedilir (storeintomempool). Kayıt, işlem txid verisini içermez, gereken yerlerde işlem txid değeri oluşturulcak şekilde bir altyapı inşa edilmiştir. Bir işlem örneği Şekil 12'de paylaşılmıştır.

```
rawtx = rawtransaction(tx.createrawtransaction())
signedrawtx = tx.signrawtransaction(from_wallet)
txid = sendrawtransaction(signedrawtx, sendflag)
tx.storeintomempool()
```



Şekil 12: İşlem Örneği

Normal bir para gönderme işlemi dışında, modelde gerçeklenen iki ayrı tür işlem de bulunmaktadır: Genesis işlem ve Coinbase işlem. Genesis işlem, kripto para modelinin ilk işlemi olma özelliğini taşır. Ana menüden <Genesis Operations> bağlantısından çalıştırılır. Genesis işlem, diğer para gönderme işlemlerinden farklı girdi ve çıktı değerlerine sahip olan

bir işlemdir. İlk işlem olan genesis işlem için öncesinde herhangi bir işlem olmamasından dolayı, girdi işlem txid değeri “0”x64, girdi vout değeri 999 olarak kodlanır. Genesis işlem için ham işlem türetme ve imzalama rutinleri normal bir işlemden farksız şekilde yürütülür. İlk işlem olan genesis işlem gerçekleştirildiğinde otomatik olarak tek bir işlem içerecek olan genesis blok da oluşturur.

Diğer bir işlem türü ise coinbase işlemdir. Coinbase işlem, madecilik sırasında bir blok oluşturulurken o bloğa dahil edilen tüm işlemler için hesaplanan ücret toplamı kadar bir ödül para işlemi yaratılır. Yaratılan bu ödül para işleminin adı coinbase işlemdir. Modelde işlem ücretleri, girdilerin sayısına bağlı olarak belirlenir. Coinbase ödül işleminde, genesis işlem benzeri girdi txid ve vout atamaları yapılır. Bu atamalar, işlemin diğer işlemlerden ayrılmasında etkilidir. Blok oluştuğunda yaratılan ve yeni bloğa dahil edilen coinbase işlemle birlikte ödül para, madenciliği yapan katılımcı cüzdan adresine utxo olarak kaydedilir. Coinbase işlemler, oluşturdukları ödül paralarla sisteme bir para akışı sağlarlar.

validate_transaction fonksiyonu:
işlem sigpubkey ve lenscript dizi değerilerini yeni dizilere taşı
her işlem girdisi için tekrar et:
eğer girdi vout sırası=999 ve sigpubkey=0x64 ise coinbase & atla
değil ise devam et
imza uzunluğunu bul
sigpubkey alanı imza uzunluğu kadar karakter al
imzayı oluştur
kalan değerleri açık anahtar olarak al
ilgili girdi sigpubkey ve lenscript alanlarını sıfırla
tüm işlem için **prepareinputforsign** fonksiyonu çağır
imzalanacak veriyi oluştur
wallet **verify_signature** fonksiyonunu imza, açık anahtar ile çağır
verify_signature boolean sonucunu dön

Şekil 13: İşlem Doğrulama Fonksiyonu

Blok madenciliğine dahil edilecek işlemler, modelin mempool modülünde tespit edilir. Genesis blok ve sonrasında gelen ilk bloklar tek işlemden meydana gelir. Bu bir zorunluluktur. Kripto para ağı temelinde bir nakit para değiş tokuşu gibidir. Genesis işlem ve bloğun ilk nakit kayıt parayı yarattığı düşünülebilir. Tek bir kağıt paranın olduğu bir para sisteminde, aynı anda birden fazla işlem yapılması ihtimali yoktur. Bundan dolayı, genesis sonrası gelen blokların da işlem sayılarının bir olması gerekir. Örnek bloklar oluşturulurken genesis sonrası ilk iki blok için işlem sayısı bir olarak atanırken, sonrasındaki blok için işlem sayısı üç olarak belirlenmiştir.

Mempool’da toplanan işlemler config konfigürasyon dosyasında belirlenen işlem sayısına ulaştığında, işlemler madencilik (mining) uygulamasına alınır. Madencilik işlem listesi generatetxlist() fonksiyonu ile belirlenir. Bekleyen işlemler arasından en fazla ücret bedeline sahip olanlar seçilir. Bunun için öncelikli olarak işlemler, ücret bedellerine göre sıralanır ve

doğrulama işlemini geçen konfigürasyon dosyasında belirlenen sayıda işlem madencilik işlemine tabi tutulur.

Algoritma adımları Şekil 13'te paylaşılan işlem doğrulama fonksiyonu, işleme tanımlanan ve imzalanan tüm girdi değerlerinin gerçekten gönderen katılımcı cüzdanı tarafından imzalanıp imzalanmadığını kontrol eder. Doğrulanacak işlem girdilerinin her biri için, imzalama öncesi işlem verisi prepareinputforsign() fonksiyonu ile tekrar oluşturulur. Kriptografi işleyiş mantısı çerçevesinde doğrulama için gerekli olan açık anahtar (pubkey), girdi sigpubkey alanının son 66 karakterinde saklanmıştır. İhtiyaç duyulan bir başka veri olan imza (signature) ise, açık anahtar (pubkey) alanından arta kalan sigpubkey alanının ilk karakterlerinde gizlenmiştir. İlgili girdi için hazırlanan imza, açık anahtar ve işlem verisi ile wallet modülü verify_signature() fonksiyonu çağrılır. Fonksiyon, imzanın gerçekten bu işlem verisine ait olduğu kontrolünü yapar ve işlemin doğrulanmasını gerçekleştirir.

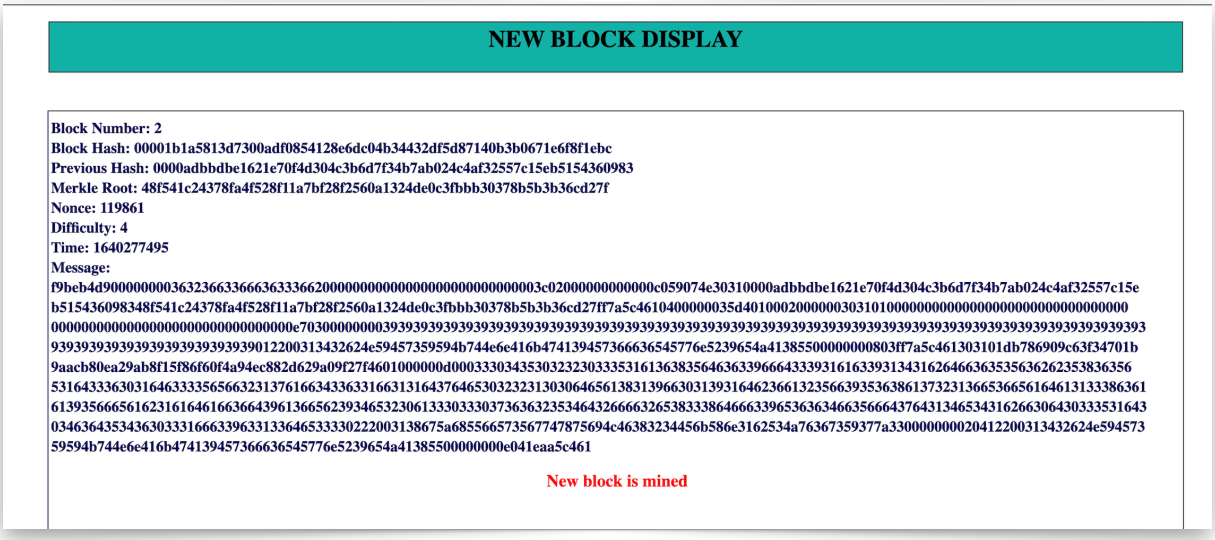
Doğrulanan işlemler, bir kez daha bakiye kontrolünden geçer. Kontrolü sorunsuz olan işlem girdileri için utxo kayıtları silinmiş olarak işaretlenir. Silinme spendable alanın "X" olarak işaretlenmesinden ibarettir. Fiziksel bir silme yapılmaz. İşlem çıktıları için uxto kayıtları oluşturulur ve veritabanına işlenir. Bloğa alınacak işlemler, mempool'dan çıkarılmalıdır. Bu amaçla, mempool verisi status alanı "9" olarak işaretlenir. Bu değer konfigürasyon dosyasında kayıtlıdır. Status alanı "9" olan işlemler mempool'dan silinmiş anlamına gelir. Mempool veritabanına silinmiş olarak kaydedilen işlemler aynı zamanda blockedtxs olarak adlandırılan bir başka veritabanına kaydedilir.

Model yazılımında calculatetxamount() ve calculatetxfee() fonksiyonları işlem tutar ve ücretlerini hesaplar. İşlem tutarı ve ücreti, girdi ve çıktı değerleri kullanılarak hesaplanır. İşlem tutarı, gönderen katılımcı adresi dışında aktarım yapılan adreslere aktarılan çıktı tutarları toplanarak hesaplanır. İşlem ücreti işlem girdi tutar toplamından çıktı tutar toplamı çıkartılarak elde edilir. Yeni bir blok yaratılırken ilgili bloktaki tüm işlemler için ayrı ayrı işlem ücreti hesaplanır ve toplam işlem ücreti bloğu madencilik işleminde çıkarmayı başaran düğüme ödül olarak aktarılır. Yazılım, ödül para aktarımını özel bir işlem türü olan coinbase ödül işlemi ile otomatik olarak gerçekleştirir. Coinbase ödül işlem tutarı blokta yer alan tüm işlemlerin ücret toplamına eş değerdir ve coinbase işlem yeni oluşturulan bloğa dahil edilir.

2.2.6. Blocks

Model yazılımında, madencilik otomatik olarak değil, ana menüden <Mining> bağlantısından gerçekleştirilir. Çalışma kapsamında, manuel olarak gerçekleştirilen madencilik, istendiği takdirde otomatik şekilde yapılacak şekilde düzenlenebilir.

Modülde, blok oluşturma adımlarını içeren fonksiyonlar yer alır. Genesis ve normal blokların yaratılması için farklı iki fonksiyon createblock() ve creategenesisblock() isimli fonksiyonlar bulunur. Her ne kadar benzer rutinleri gerçekleseler de genesis blok özel bir blok olduğu için blok yaratılırken bazı varsayılan atamalar yapılır. Blok yüksekliği sıfır, önceki blok hash değeri (prevhash) "0"x64 olarak kodlanır. Ayrıca, blokta yer alacak işlemler için merkle kök (root) değeri hesaplaması yapılır.



Şekil 14: Yeni Blok Gösterimi

Blockchain modulündeki addblocktochain() fonksiyonu çağrılarak yeni blok diğer verileri de tamamlanarak blok zincire eklenir. Blok zincire eklenen blok, struct kütüphanesi pack komutları ile paketlenir ve mesaj olarak diğer ağ düğümlerine gönderilir. Ekran gösteriminde, mesajın heksadesimal karşılığı gösterilir. Block mesajı, versiyon, önceki blok hash değeri, merkle kökü hash değeri, timestamp, zorluk seviyesi, nonce değeri, bloktaki toplam işlem sayısı paketlenerek oluşturulur. Tüm blok işlemleri de ham formatta mesaja eklenir. Oluşturulan yeni bir blok model ön yüz ekranında Şekil 14'deki gibi görüntülenir.

2.2.7. Blockchain

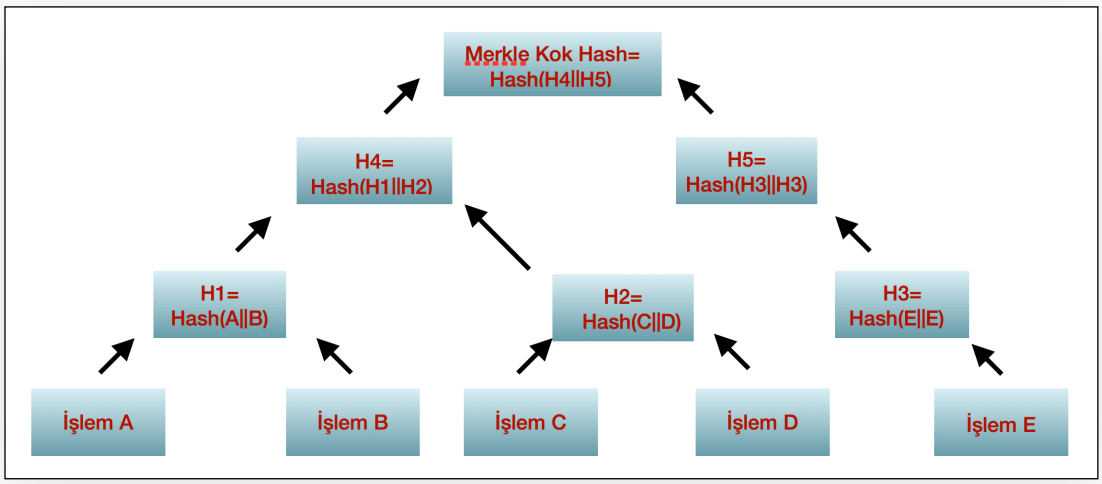
Block ve blockchain sınıflarını tanımlayan ana modüllerden biridir. Blok sınıfı, model sürümü versiyon, blok yüksekliği height, blok hash değeri, bir önceki bloğun hash değeri prevhash, blok işlemlerinin toplam tutarı blockamount, merkle kök değeri merkle, blok hash değerini bulmak için kullanılan nonce değeri, olması gereken hash zorluk seviyesi difficulty, blok oluşma zamanı timestamp alanlarını içerir.

Blok hash değeri için SHA-256 hash algoritması, blok yüksekliği, blok zincirdeki son blok hash değeri, merkle kök değeri ve nonce değerleri kullanılarak hesaplanır. Elde edilen hash değerinin zorluk seviyesine uygun olması koşulu vardır. Zorluk seviyesi, hash değerinin kaç adet "0" ile başlayacağını tayin eder ve konfigürasyon dosyasından ayarlanır. Model çalışmasında zorluk seviyesi "4" olarak belirlenmiştir. Bu koşul, hash değerinin dört adet "0" ile başlaması gerekliliğini belirler. Koşulu sağlayan SHA-256 hash sonucu dönünceye kadar algoritma çalışmaya devam eder ve kaçıncı denemede istenen sonuca ulaşılıyorsa, bu değer nonce değeri olarak kodlanır.

Blok zinciri oluşturan bloklar, ana menüden <Block Explorer> bağlantısı ile görüntülenir. Bloklara ait blok yüksekliği, blok hash değeri, önceki blok hash değeri (prevhash), blok toplam tutarı, nonce değeri ve blok oluşturulma zaman bilgileri listelenir. İstendiği takdirde seçilen bir blok için ilgili bloğu oluşturan işlemler de görüntülebilir.

2.2.8. Merkletree

Blokların altyapısını oluşturan modüllerden biri de merkletree modülüdür. Modelde, blok işlem txid hash değerleri üzerinden merkle kökü hesaplaması yapılır ve blok üzerine kaydedilir. Modelde computemerkleroot() fonksiyonunda merkle kökü hesaplanır. Kaydedilen merkle bilgisi bloğa ait işlemlerin belirlenmesinde de faydalıdır.



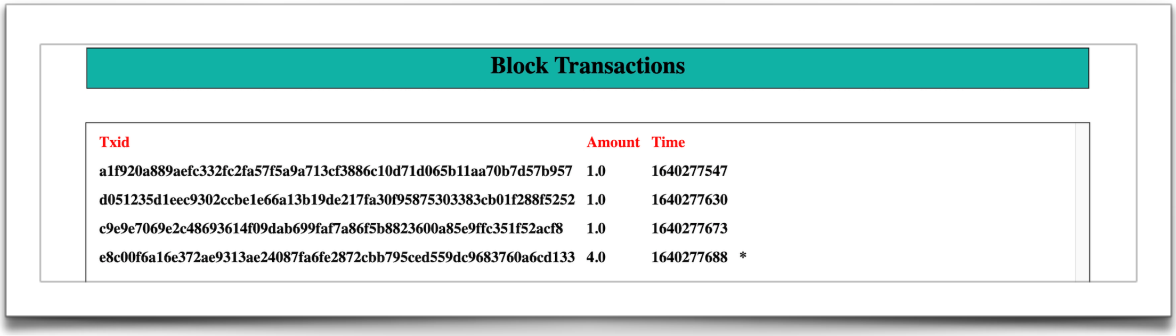
Şekil 15: Merkle Ağacı

Örnek bir ağacın Şekil 15'te görüntülendiği Merkletree binary (ikili) ağacını oluşturan işlem txid değerleri, ağacın yaprakları (leaves) olarak tanımlanır. Ağaç yapısına öncelikle yapraklar eklenir. İkili ağaçta, yaprakların sayısının çift sayı olması önemlidir. Eğer yaprak sayısı tek sayı ise, son yaprak bir kez daha ağaç yapraklarına eklenerek toplam yaprak sayısının çift sayı olması garantilenir. Ağacın en alt katmanında yer alan yapraklar, ikili gruplar halinde hash algoritmasına gönderilir ve elde edilen hash değerleri ile ağacın bir üst katmanı oluşturulur. Yaprak katmanının üstündeki bu katmanlara dallar (branches) adı verilir. Örneğin, 2^4 yaprak sayısına sahip olan bir ağaçta dallar 2^3 - 2^2 - 2^1 sayıda eleman içerir. Tek bir hash değeri elde edilinceye kadar dalların ikili gruplar halinde hash hesaplanmasına devam edilir. Ağacın tepe noktası merkle kök olarak kaydedilir.

Modelde ağaç en alt yapraklarını bloğa dahil olan işlemlerin 64 byte uzunluğundaki txid kimlik verileri oluşturur. Merkle kökü hesaplanacak tüm işlem txid verileri bir diziye atanır. İşlem sayısı tek sayı ise son işlem diziye eklenir. Böylece ağacın en alttaki dalı oluşturulmuş olur. Tek elemanlı bir dal kalıncaya kadar ikili grup şeklinde işlemlere hash hesaplanır.

Bir işlemin herhangi bir bloğa ait olup olmadığını anlamak için merkle ispatı (merkle proof) algoritması işletilir. Merkle ispatı, herhangi bir işlem txid hash değerinin, yalnızca karşılık gelen hash değerleri ile hash edilmesi ve bu şekilde ağaca tırmanarak aynı kök değerine erişilip erişilemediğinin kontrol edilmesidir. İki farklı metinin aynı hash değerine sahip olamayacağı mantığından yola çıkılarak geliştirilen bu algoritma, modelde getmerkleproof() ve merkleinclusionproof() rutinleri tarafından gerçekleştirilir.

Merkle ispatının gerçekleştirmek için öncelikle hash edilmesi gereken ağaç dallarının tespit edilmesi gerekir. İlk adımda, işlem hash değerinin sıra numarasını bulmak önem taşır. Bu sıra numarasına göre, hash değerin üst dalın sağ ya da sol çocuğu (right/left child) olduğuna karar verilir. Hash hesaplaması yapılan değerlerin sıralaması önemli olduğu için sağ ve sol çocuğun doğru tespit edilmesi önemlidir. Bir döngü içerisinde, ağacın her katmanında karşılık gelen hash değerini bulmak için matematiksel bir mantık uygulanır. Ağaç üzerinde ispat için kullanılacak dallar (txid hash) belirlendikten sonra yapraktan tepeye kadar sağ ve sol çocuk kavramına dikkat edilerek hash değerler oluşturulur. Tepe hash değeri ile blok merkle kök değerinin aynı olması, işlemin ilgili blokta olduğunun göstergesidir.



Şekil 16: Blok İşlemleri Listeleme

<Block Explorer> menü bağlantısından görüntülenen bloklara ait işlemlere erişilebilir. İlgili işlemlerin tespit edilmesinde merkle ispatı kullanılır. Bloğu oluşturan işlemler arasında bulunana coinbase işlem "*" ile işaretlenir. Modelde, işlemi oluşturan her girdi için 1 birim para ücret olarak hesaplanır. Gönderen tarafın cüzdan bakiyesinin hesaplanan ücret tutarını karşılar olması zorunludur. İşlem listesi Şekil 16'da örnek bir blok için paylaşılmıştır.

2.2.9. UTXO (Harcanmamış İşlem Çıktıları)

İşlem altyapısını destekleyen modüldür. UTXO (Unspent Transaction Output), işlemleri birbirine bağlayan yapıdır. Bu yapı sayesinde, sadece bloklar değil örtülü şekilde işlemler de ard arda bağlanmış bir işlem zinciri oluştururlar. Bir aktarım işleminin yapılabilmesi için gönderen adresin önceki işlemlerinden kalan harcanmamış işlem çıktılarının toplam değerinin aktarım tutarından büyük ya da eşit olması gerekir. Para gönderme işlemi başlatıldığında gönderen adrese ait bakiyenin hesaplanması yürütülecek ilk adımdır. Hesap bakiyesi harcanmamış işlem çıktı verileri kullanılarak elde edilir.

```
class utxo():
        def __init__(self, txid, vout, address, amount, spendable)
def storeintoutxo()
def getbalance(address)
def getblockedbalance(address)
def getavailablecoins(address, amount)
def gettxinputamount(intxid, invout, inaddress)
def removefromutxo(txin)
```

def updateutxoblocked(txin)
def getfromutxo(address)

Modül utxo, işlem (transaction) çıktılarını kaydeder (storeintoutxo), harcanan işlem girdilerini siler (removefromutxo), işlem doğrulanıncaya kadar geçen sürede utxo kayıtlarının bloke koyar (updateutxoblocked), cüzdan adresinin bakiyesini hesaplar (getbalance), bloke konulmuş kayıtların bakiye toplamlarını bulur (getblockedbalance), para gönderme işleminde girdi olarak kullanılacak utxo kayıtlarını tespit eder (getavailablecoins), bir adrese ait tüm uxto kayıtlarını sonuç olarak bildirir (getfromutxo), toplam girdilerin hesaplanılmasında kullanılmak üzere tek bir işlem girdisi için utxo coin tutarını hesaplar (gettxinputamount).

Para gönderme işlem tutarı için getavailablecoins() fonksiyonu ile para gönderen cüzdanın müsaitlik durumu belirlenir ve işlem girdisi olacak utxo kayıtlarına ait bilgiler fonksiyonun sonucu olarak döndürülür. Müsaitlik hesaplamasında işlem ücretleri de dikkate alınır. Modelde işlem ücreti, doğrudan o işlem için işlenecek girdi utxo kayıdı sayısı ile orantılı olarak hesaplanır. Her bir utxo kaydı için config isimli konfigürasyon dosyasında belirlenen kayıt başı ücreti (config.fee) gönderen cüzdan hesabından alınmalıdır. Cüzdan bakiyesinin ücret için de müsait olması şarttır. Bu kural müsaitlik hesaplamasına dahil edilir.

Harcanmamış işlem çıktıları olan utxo kayıtları işlem girdilerini oluşturmak için gereklidir. Mempool havuzuna kaydedilen onaylanmamış bir işlemin girdileri olan utxo kayıtları bloke edilir. İşlemler doğrulanarak madencilik sonrası yeni bir bloğa dahil edildiğinde işlem girdileri olan utxo kayıtları silinir ve işlem çıktıları için yeni utxo kayıtları yaratılır.

2.2.10. Peers (Eşler)

Düğüm, iletişim halinde olduğu diğer cihaz adreslerini kaydeder. Komşu düğümlerle yaptığı mesaj alışverişinde gönderdiği ve aldığı mesaj uzunluk toplamları, son mesaj gönderilme zamanı ve son mesaj alma zamanları bu veritabanında saklanır. Model uygulaması çalıştırıldığında, daha önce peers veritabanına kaydedilmiş olan komşu düğüm adresleri bağlantıların kurulmasında kullanılır.

```
class peer:
def __init__ (self, id, addr, lastsend, lastrecv, bytessent, bytesrecv):
        self.id = id
        self.addr = add
        self.lastsend = lastsend
        self.lastrecv = lastrecv
        self.bytessent = bytessent
        self.bytesrecv = bytesrecv
def storeupdatepeer(self)
def getactivepeers()
def getaddresses()
```

2.2.11. SQLdb - Veritabanı

Model veritabanı yönetiminin gerçekleştiği bileşendir. Her tablo bir dbtable sınıfı nesnesi olarak tanımlanmıştır. SQLite3 veritabanında tablo tanımlamaları, tablo okuma, yazma ve güncelleme işlemleri dbtable sınıfının alt fonksiyonlarıdır. Tablo işlemleri yapmak için öncelikle istenen tablo için dbtable sınıf nesnesi oluşturulur ve Python *args ve **kwargs parametreleri kullanılarak sql sorgu seçim kriterleri belirlenir. Örneğin, istenen bir adresin utxo kayıtlarını seçmek için aşağıdaki şekilde bir kullanım gerekir. SQLite3 veritabanı bağlantı ve query sorguları dbtable alt fonksiyonlarında gerçekleştirilir.

```
utxo = dbtable(dbpath, tblname, "txid", "vout", "address", "amount", "spendable")
coins = utxo.getdbtableitems(address = straddress)
```

Sistem veritabanı erişim ve tüm tablo işlemleri için genelleştirilen fonksiyonlar sayesinde kullanılmak istenen tablo için dinamik bir yapı sunulmuştur.

```
create_db(self)
getalldbtable(self)
getdbtableitems(self, **kwargs)
getonedbitem(self, **kwargs)
getlikedbitem(self, **kwargs)
insertdbitem(self, *args)
updatedbitem(self, **kwargs)
newdbtable(self)
newdbitem(self, **kwargs)
executesql(self, sql)
getlastinserteditem(self)
```

2.2.12. App & Forms

Model ön yüzü web uygulamaları için hazır yapılar sunan Python Flask uygulama geliştirme platformu kullanılarak yapılmıştır. Ön yüz rutinlerini içeren ana modül @app ve tüm ön yüz formları içeren forms modülleri flask platformunun sunduğu kütüphaneler ve servisler ile geliştirilmiştir. Form için GET, POST metodları, render_template() ve redirect(url_for()) kullanılan belli başlı fonksiyonlardır.

Aynı zamanda @app modülünde thread tanımları yapılır. Ön yüz ekranlarının çalıştırıldığı threadinterface, diğer düğümlerle ağ bağlantılarını başlatan threadnetwork ve socket üzerinden mesaj alış verişini gerçekleştiren threadmessaging bu thread'lerdir.

```
FlaskThread = threading.Thread(target=threadinterface)
NetworkThread = threading.Thread(target=threadnetwork)
SocketThread = threading.Thread(target=threadmessaging)
```

3. BULGULAR

Finans alanında devrim yaratacak bir altyapı ortaya koyan blok zincir teknolojisi, merkezi otoriteye duyulan güveni katılımcılarına dağıtan bir elektronik varlık transfer sistemidir. Sistemin ilk kullanımı Bitcoin blok zincir platformununun ortaya çıkışı ile gerçekleşmiştir. Kripto dünyasına giriş üstünlüğünü yakalayan Bitcoin blok zincir platformu sunduğu yüksek güvenlik sayesinde hala yaygın şekilde kullanılmaktadır. Rakiplerine göre hep bir adım önde olan Bitcoin para birimi bu çalışmada örnek alınan model olmuştur.

Her isteyenin katılabileceği izinsiz bir blok zincir üzerinde çalışacak yeni bir kripto blok zincir platformu tasarlamak amacıyla yapılan çalışmada, Github açık kaynak deposundaki Bitcoin kripto sistemini oluşturan bileşenler incelenerek sıfırdan modüler bir yapı tasarlanmış ve tüm geliştirme aşamaları adım adım gerçekleştirilmiştir. Bitcoin blok zincirinin örnek model olarak seçilmesinde Tablo 11'deki kriterler dikkate alınmıştır.

Tablo 11: Bitcoin Blok Zincir Platformu Seçim Kriterleri

Özellik	Bitcoin
Destekleyen Topluluk	bitcointalk.org, bitcoin.StackExchange.com
Açıklık	İzinsiz açık blok zincir
Ciro	Scripting (P2PKH, P2SH)
Github Çatal Sayısı	32+ K
Gerçeklenebilirlik	Açık kaynak kod desteği
Mutabakat	İş İspatı (PoW)
Piyasa Payı	~%40

Blok yaratma süresinin kısaltılması hedeflenen modelde işlemlerin daha hızlı ve düşük enerji kullanılarak gerçekleşmesi ve onaylanması sağlanmıştır. Zinciri oluşturan harcanmamış işlem çıktıların optimize edilerek seçilmesi işlem imzalama ve doğrulama süresini kısaltarak model sistemin daha verimli çalışmasına imkan vermiştir. Madenci düğümlere sırasıyla verilecek yetki sayesinde nonce hesaplaması yapılmış ve model sistemde zorluk derecesi küçültülerek enerji tüketiminin azaltılması sağlanmıştır.

Model, Python yazılım platformunda bir kripto para hizmet sağlayıcısı ve ödeme sistemi olarak geliştirilmiştir. Hızlı, kolay ve sade bir kodlama sağlayan Python, hızlı aksiyon alabilmeyi kolaylaştırır ve tekrarlanan kod kullanımı sayesinde web programlamada öne çıkar. Bu özellikleri çalışmada yazılım platformu olarak seçilmesinde etkili olmuştur. Veritabanı olarak piyasa lideri Bitcoin kripto para sistemi tarafından kullanılan SQLite3 tercih edilmiştir. Özel anahtarların tutulduğu dosyalara yüksek koruma sağlayabilme garantisi, taşınabilirlik konusunda optimum çözümler sunabilmesi ve eski sürümlere entegre olabilecek uygun değişiklikler yapması SQLite3 veritabanının kullanımına karar verilmesinde etkili olmuştur. Halen Bitcoin cüzdanlarında tercih edilen SQLite3 veritabanının ilerleyen

zamanlarda diğer Bitcoin modüllerinde de kullanılma olasılığı kararı destekleyen bir diğer unsur olmuştur.

Tablo 12: Model Konfigürasyonu

Özellik	Model Değeri
İletişim Port Numarası	10000
Buffer büyüklüğü	1024
Blok yüksekliği	3-10
İşlem ücreti	1 kripto birimi / Girdi
Zorluk Derecesi	4
Magic	0 x d9b4bef9
Desteklenen mesajlar	addr, getaddr, version, tx, block, getdata
Hash Algoritmaları	SHA-256, RIPEMD-160
Eliptik Eğri	SECP256R1
Anahtar ve Adres uzunluğu	64 byte ve 32 byte
Anahtar saklama ve string format	PEM, Base 58

Elektronik para aktarımı, sistemin işlem oluşturabilme kapasitesi ile ilişkilidir. Aktarımın yapılabilmesi için cüzdana ait harcanmamış paraların tespit edilerek girdi verileri olarak işlenmesi ve aktarım sonucu oluşacak çıktı verileriyle birlikte işlem yapısı altında kaydedilmesi gerekir. Sistem işleyişinin bu doğrultuda gerçeklenmesi için aktarım işlemini oluşturan girdi ve çıktı yapılarının tasarımı, girdilerin beslendiği harcanmamış işlem çıktı (utxo) yapısı, bu yapıların üstüne konumlandırılan işlem yapısı ve fonksiyonları oluşturulmuştur. İşlem girdileri para aktarımı tamamlandığında bulundukları cüzdan hesabından bir diğer hesaba aktarılacakları için girdilerin aktarım öncesinde onaylanması zorunludur. Bundan dolayı model işleyişinde işlem girdi değerleri kriptografik algoritmalar kullanılarak imzalanır ve işlem bir bloğa dahil edilmeden önce imzalar doğrulanır. Geliştirilen sistem altyapısında her girdisi imzalanan işlem mempool bekleme havuzuna kaydedilir. Madencilik süreci başlatıldığında havuzda bekleyen işlemler arasından işlem ücreti nispeten daha yüksek olanlar seçilir, seçilen işlemler bakiye ve imza kontrollerinden geçirilir. Bloğa dahil edilen tüm işlemlerin işlem ücretleri toplamı madenci düğümün hesabına ödül para olarak aktarılır. Yeni bir blok yaratılırken blokta yer alan işlem hash değerlerinden merkle kökü ve zorluk seviyesine göre nonce değeri hesaplanır ve yeni blok zincirdeki son bloğun hash değerine bağlanarak blok zincirine eklenir.

Model blok zincirini oluşturan blok ve işlemlerin kurulan ağda iletiminin sağlanması için temel bir TCP/IP rutini oluşturulmuştur. Yeni işlem ve blokların ağ üzerinden katılımcı düğümlere iletimini sağlayan mesaj yapısı mevcuttur. Tanımlanmış mesaj biçimlerini oluşturan ve yeni bir mesaj alındığında mesaj alanlarına ayrıştırarak ilgili veritabanına işleyen bir altyapı kurulmuştur. Veritabanına ilişkin tüm güncelleme ve sorgulama işlemleri

tanımlanan bir sınıf yapısı üzerinden gerçekleştirilir ve veritabanı kullanımı diğer modüllerden kolaylıkla erişilebilen sınıfa ait fonksiyonlar üzerinden gerçekleştirilir.

Kullanıcı ile etkileşim Flask paketi kullanılarak geliştirilen ön yüz ekranlarından sağlanır. Para gönderme (Send Crypto), madencilik (Mining), cüzdan görüntüleme (Wallets), Mempool havuzu görüntüleme (Mempool Explorer), blok görüntüleme (Block Explorer) ve Genesis blok oluşturma (Genesis Operations) menü ekranlarıdır. Genesis blok özel bir bloktur ve yalnızca bir defaya mahsus blok zincir ilk kez oluşturulurken çalıştırılır. Sonrasında Genesis blok erişime kapatılır. Katılımcılar başka bir hesaptan hesaplarına aktarılan para veya madencilik ödül para kazanımları sayesinde cüzdan adreslerindeki bakiyelerini arttırabilirler. Ön yüz para gönderme bağlantısı kripto varlığın aktarım adımlarını gerçekleştirir. Mempool bekleme havuzu yeterli seviyeye ulaştığında yeni blok çıkarma aşamaları madencilik madencilik bağlantısı arka planında yürütülür. Havuzda çıkartılmayı bekleyen işlemler ve çıkartılan bloklar kullanıcılar tarafından görüntülenebilir. Modüler uygulama ile tüm akış yürütülür. Modeli oluşturan modüller Tablo 13'te detaylıca incelenmiştir.

Tablo 13: Bileşen Özellikleri

Modül	Güvenlik	Performans	Kullanılabilirlik	Ölçeklenebilirlik
Walllet (Cüzdan)	256 bit ECDSA ve hash	RSA'ye göre daha düşük	Ön yüz ile cüzdan oluşturma ve takip	Diğer şifrelemelere göre yüksek
Message (Mesaj)	pack/unpack ile mesaj paketleme	Yüksek hız	Para gönderme ve madencilik	İletim gecikebilir
Network	Güvenli bağlantı	TCP/IP ile yüksek hız	Bağlantı kurma, mesaj gönderme ve alma	Düğüm artışında risk yok
Transaction (İşlem)	SHA-256 hash, 256 bit özel anahtar ile imza	İmzalama yavaş	Ön yüz ile işlem oluşturma ve takip	İşlem ücreti artabilir
Mempool	İşlem 256 bit hash zinciri	Yüksek hız	Mempool izleme	Bloklara dahil olmak zorlaşır, bekleyen işlem sayısı artar
Blocks	256 bit açık anahtar ile imza doğrulama	İmza doğrulama daha hızlı	Madencilik	Blok büyüklüğü artışına limit
Blockchain	Merkle Tree, İş İspatı (PoW)	Zorluk seviyesine bağlı	Zincir ekleme ve izleme	Güncel kalabilme zorlaşır
Merkletree	32 byte hash dalları	Blok büyüklüğüne bağlı	Blok işlem kontrolü	Blok büyüklüğü limiti gerekir
UTXO	256 bit hash ve açık anahtar	Hızlı girdi seçimi, hızlı işlem	İşlem girdi oluşturma	Girdi artışı işlem hızını olumsuz etkiler
Peers	Network adres formatı	Hızlı	Mesaj gönderme	Belli sayıda düğüme mesaj iletimi
SQLdb	SQLite3 ile veri güvenliği ve bütünlüğü	Hızlı ve sağlam	Model kullanımı	DB işlem artışı, daha fazla bellek gerekebilir

Modül	Güvenlik	Performans	Kullanılabilirlik	Ölçeklenebilirlik
App&Forms	Güvenli	Thread ile işlem gücü ve hızı arttırımı	Model önyüzü	Risk yok

4. TARTIŞMA VE SONUÇ

Çalışma kapsamında blok zincir işleyiş ve prensipleri, mutabakat yöntemleri ve akıllı sözleşmeler hakkında kapsamlı bir literatür araştırması gerçekleştirilmiştir. Blok zincir uygulamalarında güvenliği sağlayan yapı taşları, kriptografik şifreleme ve hash algoritma yöntemlerinin tüm aşamaları detaylıca incelenmiştir. Kripto para teknolojisinin öncüsü Bitcoin para sistemi ile kripto dünyasının yaygın kullanılan diğer para birimleri Ethereum ve Ripple sistemlerinin öne çıkan özellikleri ayrıntılı şekilde araştırılmış, ortak ve farklı yönleri karşılaştırmalı olarak açıklanmıştır.

Araştırmalar ışığında yeni bir kripto para sistemini hayata geçirme süreci tasarlanmış, her sistem parçasının tamamen bu çalışma kapsamında üretilmesi için gerekli unsurlar ve aşamalar tespit edilmiş, tüm unsurların birbirleriyle etkileşimleri belirlenerek sistemin bir bütün olarak çalışabilmesini sağlayacak şekilde planlanan uygulama adımları yerine getirilmiştir. Sistem tasarımı ve yazılım geliştirme aşamalarında kaynak kodların depolandığı GitHub kaynak yönetim platformundan yararlanılmış ve ağırlıklı olarak Bitcoin kaynak kodları incelenmiştir. Özellikle ağ ve mesajlaşma yapılarını oluşturma, veritabanı kullanımı, kriptografik işlem adımları, varlık transferine ilişkin kavramların modele uyarlanması, kullanıcı etkileşiminin sağlanmasına yönelik ayrıntılı ve yoğun çalışmalar yapılmıştır.

Kripto para modeli işleyişini şekillendiren başlıca unsurlardan varlık transfer işlemleri ve madencilik bloklarına ilişkin tüm temel ve destekleyici yapıların seçilen Python yazılım platformunda uygun şekilde geliştirilmesi sağlanmıştır. Platformun şifreleme için sunduğu kriptografi paketi sayesinde cüzdan ve anahtar oluşturma, veri imzalama ve imza doğrulama adımları gerçeklenmiştir. Blok zincir ağında bilgi akışını sağlayan mesajların iletimi ve düğümler arası karşılıklı iletişim TCP/IP protokolünün socket programlama paketi kullanılarak uygulanması sayesinde mümkün olmuştur. Mesaj yapı ve biçimleri Bitcoin sisteminde kullanılan mesaj tiplerinden faydalanılarak tasarlanırken model ekran yazılımları Flask paketi ve html şablonları ile düzenlenmiştir.

Model ön yüzü ile arka planda çalışan ve ağ iletişimini sağlayan mesajlaşmanın aynı anda paralel şekilde yürütülmesi için thread parçaları kullanılmıştır. Modelde tanımlı ilk thread ön yüz Flask uygulaması, diğer thread parçaları ise ağ bağlantılarının oluşturulmasını sağlayan rutin ile mesajların işleme konulmasını sağlayan rutinlerdir.

Kripto ödeme sistemleri şüphesiz bugünün finans dünyasını farklı bir yere taşıyacak ve geleceğin finans dünyasını şekillendirecektir. Bir sunucunun sağladığı merkezi yönetimi ve bir otoritenin sunduğu güveni İnternet ortamında katılımcıları arasında dağıtmaya olanak sağlayan kripto paralar bu çalışmanın esas konusu olmuştur. Modeli meydana getiren modüller araştırmalardan edinilen bilgiler doğrultusunda baştan geliştirilmiş olup sadece bu çalışmaya özeldir. İşlem imzalama ve doğrulama sürelerinin kısaltılarak daha verimli bir işleyiş sunulan model sistemde hız ve düşük enerji kullanımı ön planda yer almaktadır. Madenci düğümlere sırasıyla verilecek yetki ile yapılan nonce hesaplaması ve zorluk derecesi küçültülerek enerji tüketimi azaltılmıştır.

Bir kripto para sistemi temelinin inşa edildiği modelde yapılan çalışmaların gelecekte yapılacak kripto para çalışmalarına bir kaynak olması amaçlanmıştır. İleri boyutlara taşınan bir kripto para sisteminin geniş bir katılımcı topluluğu tarafından kullanılacak düzeye getirilerek ölçeklenebilirliğinin araştırılması sonraki çalışmaların hedefleri arasında yer alabilir. Gelecekte finans piyasalarında adı geçen bir kripto varlığın yaşam döngüsünde, yürütülen çalışmanın izlerini bulabilmek temelin oldukça sağlam atıldığının en iyi kanıtı olacaktır.

Kripto varlık aktarımına ilişkin bir sistemin tasarlanmasında şüphesiz güvenlik ve gizlilik öne çıkar. Bir kripto sisteminin herkesin kullanımına açık İnternet ağı üzerinde güvenilirliği veri güvenliği ve mahremiyeti ile ölçülür. Model çalışmasında sistem bünyesinde gerçekleşen aktarım hareketlerine ilişkin verilerin korunması ve mahremiyeti için SHA-256 ve RIPEMD-160 hash algoritmaları kullanılmıştır. Hash sonuç değerlerinin çarpışma olasılıklarını çok daha azaltabilmek için 64 byte uzunluğunda sonuçlar üreten SHA512 gibi algoritmalar kullanılarak oluşturulacak kripto uygulamalarının sağlayacağı faydaları inceleyen bir çalışma ilerleyen zamanlarda yapılabilir. Benzer şekilde gelecek dönemlerde keşfedilecek gelişmiş farklı hash algoritmalarının kripto varlık aktarım sistemleri üzerindeki etkilerini araştıracak çalışmalar faydalı olacaktır.

Mutabakat mekanizması, ağa dağıtılmış güvenin tesisini sağlayarak katılımcıların koordine olmasını mümkün kılar. Çalışma modelinde kripto para aktarımları üzerinde fikir birliğini temin eden mutabakat algoritması olarak iş ispatı yöntemi seçilmiştir. İş ispatı dışında farklı mutabakat algoritmalarının yürütüleceği yeni bir model türetmek gelecek çalışmaların hedefleri arasında yer alabilir. Özellikle iş ispatı yönteminin enerji ihtiyacı ve hızı ile yarışacak yöntemler seçilerek performans karşılaştırmalı sonuçlar sunacak modeller geliştirilmesi ileriki dönem araştırmalarının konusu olmalıdır.

Kripto para modelinde cüzdan şifreleme, işlem imzalama, imza doğrulama saldırganlara karşı koruma sağlar. Modelde koruma adımları 256 bit eliptik eğri şifreleme paketi kullanılarak gerçeklenmiştir. Her geçen gün gelişen teknolojik ürünler ve insanoğlunun gizliliği keşfedip ortaya çıkarma tutkusu mahremiyet üzerine kurulmuş kripto sistemler için ciddi bir tehlike yaratmaktadır. Kolay para kazanma hayalinin çekiciliği de eklendiğinde bir kripto paranın gelişmiş bir korumaya ihtiyacı vardır. İlerleyen dönemlerde tüm tehditlere karşı daha yüksek bit uzunluğunda şifreleme yapacak modeller geliştirilerek sistem işleyişine etkilerini konu alan çalışmalar yapılması mutlaka faydalı olacaktır.

KAYNAKLAR

- Armknecht F., Karame G.O., Mandal A., Youssef F., Zenner E., 2015, Ripple: Overview and Outlook, *8th International Conference, TRUST 2015*, 24-26 August 2015 Heraklion, Greece, 163-180

- Biczok D., 2018, *The future of Bitcoin and the Blockchain Technology*, Master's Thesis, Universite Du Luxembourg

- Christidis K., Devetsikiotis M., 2016, Blockchains and Smart Contracts for the Internet of Things, *IEEE Access*, 4, 2292-2303

- Dang Q.H., 2015, Secure Hash Standard (SHS), *Federal Information Processing Standards Publication*, U.S. Department of Commerce, FIPS PUB 180-4

- Efanov D., Roschin P., 2018, The All-Pervasiveness of the Blockchain Technology, *Elsevier Procedia Computer Science,* 123, 116-121

- Faria C.S.F., 2019, *BlockSim: Blockchain Simulator*, Tecnico Lisboa

- Gilbert H., Handschuh H., 2004, Security Analysis of SHA-256 and Sisters, *10th International Workshop on Selected Areas in Cryptography SAC*, 14-15 August 2003 Berlin, Heidelberg, Springer Verlag, ISBN: 3-540-21370-8, 175-193

- Gust M., 2018, Cryptocurrencies: Technical and Functional Aspects, *Contemporary Economy Journal*, 3(3), 198-212

- Jani S., 2018, An Overview of Ripple Technology & its Comparison with Bitcoin Technology, Distributed Ledger Technology

- https://en.bitcoin.it/wiki/

- https://developer.bitcoin.org/

- https://river.com/learn/bitcoins-utxo-model/

- https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch05.html

- https://stackoverflow.com/

- https://en.wikipedia.org/wiki/Ethereum

- Kasireddy P., 2017, *How does Ethereum work, anyway?*, https://www.preethikasireddy.com/post/how-does-ethereum-work-anyway, [Ziyaret Tarihi: 7.Haziran.2021]

- Krupp J., Rossow C., 2018, TEETHER: Gnawing at Ethereum to Automatically Exploit Smart Contracts, *27th Usenix Security Symposium*, 15-17 August 2018, Baltimore, MD, USA

- Li X., Jiang P., Chen T., Luo X., Wen Q., 2020, A survey on the security of blockchain systems, *Elsevier Future Generation Computer Systems,* 107, 841-853

- Meiklejohn S., Pomarole M., Jordan G., Levchenko K., McCoy D., Voelker G.M., 2016, A Fistful of Bitcoins: Characterizing Payments Among Men with No Names, *Communications of the ACM*, 59(4), 86-93

- Monrat A.A., Schelen O., Andersson K., 2019, A Survey of Blockchain From the Perspectives of Applications, Challenges and Opportunities, *IEEE Access*, 7, 117134-117151

- Nakamoto S., 2008, A Peer-to-peer Electronic Cash System, *metzdowd.com's Cryptography Mailing List*

- Narayanan A., Bonneau J., Felten E., 2016, *Introduction to Cryptography & Cryptocurrencies*, *Introduction to Cryptography & Cryptocurrencies,* Bitcoin and Cryptocurrency Technologies, In: Miller A., Goldfeder S., Chapter 1-2, Princeton University Press, ISBN: 9780691171692, 23-50, 75-100

- Pazmino J.E., Rodrigues C.K.S., 2015, Simply Dividing a Bitcoin Network Node may Reduce Transaction Verification Time, *The SIJ Transactions on Computer Networks & Communication Engineering (CNCE)*, 3(2), 17-21

- Sanchez P.M., Zafar M.B., Kate A., 2016, Listening toWhispers of Ripple: Linking Wallets and Deanonymizing Transactions in the Ripple Network, *Proceedings on Privacy Enhancing Technologies*, 2016(4), 436-453

- Seirawan R., 2019, *Veri Transferinde Blok Zinciri Uygulaması*, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi

- Tapsell J., Akram R.N., Markantonakis K., 2018, An evaluation of the security of the Bitcoin Peer-to- Peer Network, *IEEE/ACM International Conference on Cyber, Physical and Social Computing (CPSCom), Greeen Computing and Communications,* Halifax, NS, Canada, 1057-1062

- Tikhomirov S., 2017, Ethereum: state of knowledge and research perspectives, *The 10th International Symposium on Foundations and Practice of Security*, 23-25 October 2017, Nancy, France, 206-221

- Tschorsch F., Scheuermann B., 2016, Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies, *IEEE Communications Surveys and Tutorials*, 18(3), 2084 - 2123

- Wang H., Xie S., Dai H.N., Zheng Z., 2018, Blockchain challenges and opportunities: a survey, *International Journal of Web and Grid Services*, 14(4), 352-375

- Wang S., Ouyang L., Yuan Y., Ni X., Han X., Wang F.Y., 2019, Blockchain-Enabled Smart Contracts: Architecture, Applications and Future Trends, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11), 2266- 2277

- Wood G., 2014, Ethereum: A Secure Decentralized Generalised Transaction Legder EIP-250 Revision, *Ethereum Project Yellow Paper*, 151, 1-32

- Xiao Y., Zhang N., Lou W., Hou Y.T., 2020, A survey of Distributed Consensus Protocols for Blockchain Networks, *IEEE Communications Surveys &Tutorials*, 22(2), 1432-1465

- Yaga D., Mell P., Roby N., Scarfone K., 2019, NISTIR 8202 Blockchain Technology Overview, *Computer Science, Cryptography and Security*, U.S. Department of Commerce, NISTR 8202

EKLER

Modeli oluşturan modül yazılımları

ÖZGEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı	
Doğum Yeri	
Doğum Tarihi	
Uyruğu	❒ T.C. ❒ Diğer:
Telefon	
E-Posta Adresi	
Web Adresi	

Eğitim Bilgileri	
Lisans	
Üniversite	
Fakülte	
Bölümü	
Mezuniyet Yılı	

Yüksek Lisans	
Üniversite	
Enstitü Adı	
Anabilim Dalı	
Programı	

Makale ve Bildiriler