FEN BİLİMLERİ ENSTİTÜSÜ

BLOK ZİNCİRİ AĞ GÜVENLİĞİ İLE KONSENSÜS MEKANİZMALARI VE AKILLI SÖZLEŞMELER

Mimar ASLAN

Danışman
Doç. Dr. Mustafa Cem KASAPBAŞI

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
İSTANBUL - 2022

KABUL VE ONAY SAYFASI

Mimar ASLAN tarafından hazırlanan **"Blok Zinciri Ağ Güvenliği ile Konsensüs Mekanizmaları ve Akıllı Sözleşmeler"** adlı tez çalışması 02/02/2022 tarihinde aşağıdaki jüri üyeleri önünde başarı ile savunularak, İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman	**Doç. Dr. Mustafa Cem KASAPBAŞI** İstanbul Ticaret Üniversitesi	.....
Jüri Üyesi	**Doç. Dr. Buket DOĞAN** Marmara Üniversitesi	.....
Jüri Üyesi	**Dr. Öğr. Üyesi Arzu KAKIŞIM** İstanbul Ticaret Üniversitesi	.....

Onay Tarihi: 10.02.2022

İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsünün 10.02.2022 tarih ve 2022/339 numaralı Yönetim Kurulu Kararının 2. maddesi gereğince, ders yüklerini ve tez yükümlülüğünü yerine getirdiği belirlenen "Mimar ASLAN" adlı öğrencinin mezun olmasına oy birliği ile karar verilmiştir.

Prof. Dr. Necip ŞİMŞEK
Enstitü Müdürü

AKADEMİK VE ETİK KURALLARA UYGUNLUK BEYANI

İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversitede veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

10/02/2022

Mimar ASLAN

İÇİNDEKİLER

ÖZET

Yüksek Lisans Tezi

BLOK ZİNCİRİ AĞ GÜVENLİĞİ İLE KONSENSÜS MEKANİZMALARI VE AKILLI SÖZLEŞMELER

Mimar ASLAN

İstanbul Ticaret Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Mustafa Cem KASAPBAŞI

2022, 97 sayfa

Bu çalışmada,

Finans, sağlık, sosyal medya vb. ortamlardaki insanların ihtiyacı olan güven problemine blok zinciri teknolojisi, şifreli algoritmalar ile çözümler sunmaktadır. Güven problemini çözen ve verileri dağıtık olarak kayıt altına alan ve her şeyi şeffaf olarak bizlere sunan blok zinciri bir devrim niteliğindedir. Blok zinciri, akıllı sözleşmeler sayesinde kurumsal projelerde de kullanılmaktadır. Kurumların arasında yeni nesil bir ağ olarak da adlandırılan blok zinciri ile birçok şeyin değişmesi beklenmektedir. İnternetin, mobile cihazların ve sensörlerin yaygınlaşmasıyla birlikte güven problemi her geçen gün daha da önem kazanmaktadır. Farklı amaçlara hizmet eden Ethereum, Cardano, EOS, Cosmos, Hyperledger gibi blok zincir platformları bulunmaktadır. Altyapılarında Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS) ve Directed Acyclic Graph (DAG) gibi farklı fikir birliği mekanizmalarını kullanmaktadırlar. Bu çalışmada blok zinciri platformları, altyapılarında kullandıkları mutabakat mekanizmaları ve blok zinciri ağının güvenliği araştırılmıştır. Ethereum platformu üzerinde çalışan Solidity programlama dili ile akıllı sözleşme örnekleri geliştirilmiştir. Akıllı sözleşme uygulamaları ile blok zincirinin çalışma adımları uygulamalı olarak gösterilmiştir.

Anahtar Kelimeler: Ağ güvenliği, akıllı sözleşmeler, blok zinciri platformları, fikir birliği mekanizmaları.

ABSTRACT

M.Sc. Thesis

CONSENSUS MECHANISMS AND SMART CONTRACTS WITH BLOCKCHAIN NETWORK SECURITY

Mimar ASLAN

İstanbul Commerce University
Graduate School of Applied and Natural Sciences
Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Mustafa Cem KASAPBAŞI

2022, 97 pages

In this study,

Blockchain technology offers solutions with encrypted algorithms to the trust problem that people in finance, health, social media, etc. need. Blockchain, which solves the problem of trust and records data in a distributed manner and presents everything to us transparently, is revolutionary. Blockchain can also be used in corporate projects thanks to smart contracts. A lot of things are expected to change with Blockchain, which is also called a new generation network among institutions. With the widespread use of the Internet, mobile devices and sensors, the problem of trust is gaining more and more importance day by day. There are blockchain platforms such as Ethereum, Cardano, EOS, Cosmos, Hyperledger that serve different purposes. They use different consensus mechanisms such as Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS) and Directed Acyclic Graph (DAG) in their infrastructure. In this study, Blockchain platforms were investigated by using the consensus mechanisms they use in their infrastructure and the security of the blockchain network. Smart contract samples were developed with the Solidity programming language running on the Ethereum platform. The working steps of the blockchain with smart contracts and applications are demonstrated in practice.

Keywords: Blockchain platforms, consensus mechanisms, network security, smart contracts.

TEŞEKKÜR

Bu araştırma için beni yönlendiren, karşılaştığım zorlukları bilgi ve tecrübesi ile aşmamda yardımcı olan değerli Danışman Hocam Doç. Dr. Mustafa Cem KASAPBAŞI'na teşekkür ederim.

Araştırmanın yürütülmesinde maddi ve manevi yardımlarını gördüğüm Selma ERGÜL, Göksel KIBIŞ, Prof. Dr. Necip ŞİMŞEK, Dr. Öğr. Üyesi Muhammet CEYLAN, Doç. Dr. Emine Esra KASAPBAŞI, Doç. Dr. Buket DOĞAN, Dr. Öğr. Üyesi Arzu KAKIŞIM, Muzaffer ALBENİ, Ömer METİN, İslam İNCE, Maruf ÜDÜRGÜCÜ, Sadullah ÜDÜRGÜCÜ, Aydın ERSÖZ, Tanju UÇAN, Alper Utku UÇAN, Uğur ORTAÇ, Ayla KARA ve Milan BABA'ya teşekkür ederim.

Tezimin her aşamasında beni yalnız bırakmayan aileme sonsuz sevgi ve saygılarımı sunarım.

Mimar ASLAN
İSTANBUL, 2022

ŞEKİLLER

ÇİZELGELER

SİMGELER VE KISALTMALAR

API	Application Programming Interface (Uygulama Programlama Ara yüzü)
BTC	Bitcoin
DAG	Directed Acyclic Graph (Yönlendirilmiş Döngüsüz Grafik)
DApp	Decentralized Applications (Merkezi Olmayan Uygulamalar)
DLT	Distributed Ledger Technology (Dağıtık Kayıt Teknolojisi)
DPoS	Delegated Proof of Stake (Temsil Edilen Hisse Kanıtı)
ETC	Ethereum Classic
ETH	Ethereum
EVM	Ethereum Virtual Machine (Ethereum Sanal Makinesi)
IoT	Internet of Things (Nesnelerin İnterneti)
P2P	Peer to Peer (Eşten eşe)
PBFT	Practical Byzantine Fault Tolerance (Pratik Bizans Hata Toleransı)
PoS	Proof of Stake (Hisse Kanıtı)
PoW	Proof of Work (İş Kanıtı)
SHA	Secure Hash Algorithm (Güvenli Hash Algoritması)
TX	Transaction (İşlem)

1. GİRİŞ

Satoshi Nakamoto adındaki bir araştırmacı ya da araştırma grubunun 2008 yılında yayımladığı Bitcoin A Peer-to-Peer Electronic Cash System (Bitcoin Eşler Arası Elektronik Nakit Sistemi) makalesinde finansal işlerde bankalara güvenmekten başka bir yolun olabileceğini insanlara göstermiştir. Makalede Satoshi, finansal sistemle iletişim halinde olmak için aracılara güvenmek zorundayız ama aslında ihtiyacımız olan şey, güvenin yerini kriptolojik algoritmaların aldığı ve insanların aracısız olarak birbirine doğrudan bağlı olduğu bir altyapıdır. Biz bu altyapı gerçekleştirdik ve size anlatmaktayız (Nakamoto, 2008). Kurumlara, kişilere, aracılara hepsine olan güven ihtiyacını ortadan kaldırdıklarını, bunun yerine insanların birbirine doğrudan bağlı olduğu şifreli algoritmalar ile güvenin de otomatik olarak sağlandığı bir altyapıdan söz etmekteydi. Bu da finans sektörü için çok iddialı bir devrimin başlangıcı demekti.

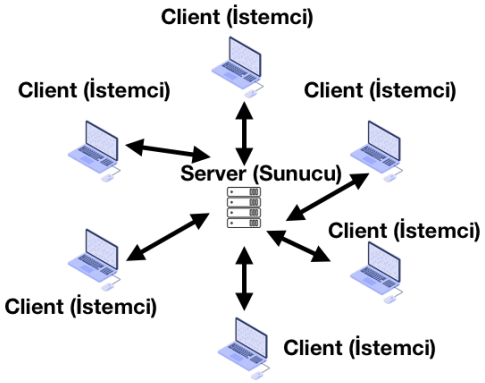
Güvene dayalı iş yapılan kurumların sadece bankalardan ibaret olmadığı 2013 yılına gelindiğinde iyice fark edildi. Belediyeler, noterler, sosyal sigorta, seçim merkezleri, üniversiteler daha böyle birçok kurum güvene ihtiyaç duymaktadır. Bu kurumlarla insanlar arasındaki ilişki güvene dayanmaktadır. Güvene dayalı iş yapmaya biçimine en kısa zamanda bir çare bulunmalıydı. Vitalik Buterin aracılara olan güven sorununa geliştirdiği Ethereum Blockchain (Blok zinciri) platformu üzerinde çalışan, Smart Contracts (Akıllı sözleşmeler) ve Decentralized Applications (DApp, Merkezi olmayan uygulamalar) ile yepyeni çözümler sunmaktaydı.

Blok zinciri, güvene ihtiyaç duyulan bütün kurumları tek bir ağda buluşturarak, kişilerin doğrudan birbirlerine bağlı olmasına imkân vermektedir. Blok zinciri, herkese açık, şeffaf, değiştirilemez dağıtık bir kayıt defteridir. Blok zincirinin çalışmasını bir para transferi üzerinden inceleyelim. Günümüzde iki kişi arasında geçen bir para transferinde arada banka kurumu yer almaktadır. Her iki tarafın da bankada hesapları vardır. Bir A kişisi, bir B kişisine para gönderdiğinde A kişinin hesabında yeterli miktarda para varsa hesabından düşürülmektedir ve gönderilen miktar B kişinin hesabına eklenmektedir. Bankanın veri tabanında

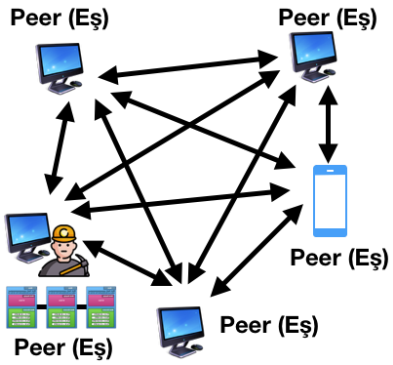
güncellemeler yapılmaktadır. Şimdi aynı transfer işlemini blok zinciri ağı üzerinden takip edelim. A kişisi para gönderme isteğini blok zinciri ağına iletmektedir. Ağa bağlı olan tüm birimlere node (düğüm) denilmektedir. Ağdaki her düğüm bu isteği almaktadır. Düğümler blok zinciri ağında yapılan işlemleri doğrulamak ve yeni bir bloğu oluşturup zincire eklemek için çalışan makinelerdir. Aktif olarak sürekli çalışıp blok zincir ağını ayakta tutan düğümlere miner (madenci) denilmektedir. Bir düğüm, önce A kişisinin hesabında yeterli bakiye olup olmadığını kontrol edip doğrulamaktadır. B kişisine gönderilecek miktarı da oluşturduğu bir kutu içindeki işlem listesine sıralı olarak yazmaktadır. Blok zincir ağın üzerinde yapılan işlemlerin tutulduğu bu kutuya block (blok) denilmektedir. Bir blok hazır edilince zincire ekleme işlemine geçilmektedir. Madenci düğümler bloğun geçerliliğini kanıtlamak için aralarında bir sayı bulma yarışına girişmektedir. Sayıyı ilk kim bulursa diğerlerine de bunu haber vermektedir. Diğer madenci düğümlerden de onay gelince sayıyı bulan madenci yeni bloğu önce kendi makinesindeki zincire eklemektedir. Ardından ağdaki diğer başka bir düğüm de o yeni bloğu alıp kendi zincirine eklemektedir. Bloğu ekleyen madenci düğüm, sistemdeki teşvik ödülü kazanmaktadır. A'dan B'ye yapılan aktarım süreci, yeni blok zincire eklendiği zaman tamamlanmaktadır. Yeni blok eklendiğinde ağın tamamında madenciler zincirlerinde güncelleme yapmaktadır. Madenciler olduğu müddetçe blok zincir ağı ayakta kalmaya devam etmektedir. Ağı ayakta tuttukları için madencilere teşvik ödülü verilmektedir.

Blok zinciri çoğu zaman kripto para Bitcoin (BTC) ile karıştırılmaktadır. Blok zinciri bir teknolojidir ve altyapı sunar. Bitcoin ise sadece bir blok zincir projesidir. Blok zincir altyapısını kullanarak kurumsal projeler geliştirmek de mümkündür. Blok zincirinin karakteristik özelliklerini incelendiğinde kayıt altına alınmış işlemlerin üzerinde daha sonradan bir değişiklik yapılamamaktadır. Gerçekleştirilen işlemler peer-to-peer (P2P, eşten eşe)dir ve birilerinin işlemler için aracılık etmesine de ihtiyaç duyulmamaktadır. Yapılan işlemlerin hepsi ağdaki tüm makinelerde dağıtık olarak kaydedilmektedir. Blok zinciri sayesinde aracıların işlevselliğine gerek kalmamaktadır. Şahıslara ve kurumlara güvenme durumu son yıllarda ortaya çıkan finans ve sosyal medya platformlarında gözlenen skandallarla daha da tartışılır bir hale gelmiştir. Blok zinciri sayesinde

güven problemi geri planda şifreli algoritmalarla sağlamaktadır. Blok zincirinin bir sahibi yoktur. Ağdaki herkes eşit haklara sahiptir ve ağa bağlı olarak çalışan makineler var olduğu sürece de sistem çalışmaya devam etmektedir. Blok zinciri teknolojisi şeffaftır. Yapılan işlemler şifrelenerek kayıt altına alınmakta ve asla kaybolmamaktadır (Yaga vd., 2019). Blok zinciri ağı herkesin görebileceği büyük bir veritabanıdır. Blok zinciri teknolojisini dijital ödeme, finans, sağlık, sigorta, borsa, tedarik zinciri, lojistik, nesnelerin interneti, mülkiyet tescili, devletin resmi belgelerinde, veri yönetiminde ve paylaşım ekonomisinin olduğu her yerde bir noter gibi kullanılmaktadır. Blok zincirinde her şey şeffaftır ve daima denetime açıktır. Ağ üzerinde gerçekleştirilen tüm işlemler kişi ya da kurum güvencesine ihtiyaç duymamaktadır. Algoritmalar, akıllı sözleşmelerle bu güveni garanti altına almaktadır (Seijas vd., 2016). Blok zincirinde veriler kayıt altına alınırken Şekil 1.1'deki gibi geleneksel istemci sunucu mimarisi yerine Şekil 1.2'deki gibi merkezi olmayan dağıtık bir yaklaşım kullanılmaktadır.



Şekil 1.1. Geleneksel istemci ve sunucu mimarisi (Weimeng, 2019)



Şekil 1.2. Eşten eşe çalışan blok zincir mimarisi (Weimeng, 2019)

İşlemlerin geçmişini ağa dağıtılmış olarak paylaşılan bir deftere kaydeder, ağdaki tüm katılımcılar defter hakkında aynı görüşe sahiptir ve deftere yazılan bir veri bir daha asla değiştirilmemektedir (Benhamouda vd., 2019). Blok zinciri, şifrelenen verileri hem şeffaf hem de dağıtık olarak tutmaktadır.

Her düğümde bloklardan oluşan bir blok zinciri tutulmaktadır. Bu zincire dağıtık açık kayıt defteri denir. Bu defter herkese açıktır ve ağdaki tüm düğümlerde bir kopyası dağıtık olarak bulunmaktadır. Ağ üzerinde yapılmak istenen işlemleri doğrulayan, kayıt altına alan, yeni bloklar oluşturup zincir ağına ekleyen birim madenci düğümdür. Ağdaki her düğümün madenci olmasına gerek yoktur. Ağda bazı düğümlerin madenci olarak görev alması zincirin devamı ve tutarlılığı için yeterli olmaktadır. Madenci düğümler çok güçlü donanımlara sahip bilgisayarlardır. Özellikle ekran kartlarının yüksek kapasitelere sahip olması ağdaki işlemlerin doğrulanma aşamasını hızlandırmakta ve madenci düğüme daha çok kazanç sağlamaktadır. Madenci bir düğüm, blok zincir ağında gerçekleştirilen işlemlerin doğruluğunu devamlı olarak kontrol etmektedir. Bir madenci düğümün ağa bağlı, birden fazla makinesi olabilmektedir. Bu makineler, aralıksız olarak çalıştıklarından dolayı zamanla ısınmakta ve çok elektrik enerjisi tüketmektedir. Madenci düğüm tarafından doğrulanan işlemler bir blok haline getirilip blok zincir ağına dahil edilmektedir. Blok zinciri merkezi olmayan, şifreli, üst düzey güvenlikli bir teknolojidir. Ağdaki her bilgisayar eşit kabul edilmektedir ve ağdaki hiçbir düğümün de diğerine karşı da bir üstünlüğü yoktur.

Blok zinciri ile geliştirilen bir operasyonda yapılan işlemlerin hepsi eksiksiz kayıt altına alınmaktadır ve bir daha da değiştirilmemektedir. Bu sayede sürekli büyüyen tutarlı bir defter yapısı mevcuttur. Bu deftere yeni bir kayıt eklenildiği zaman ağdaki tüm makinelere güncelleme bilgisi verilmektedir ve her bilgisayar tuttuğu deftere, gelen bu yeni kaydı ekleyerek verilerini güncellemektedir. Yazılan bilgiler kalıcıdır ve hiç kimse tarafından asla silinemez ve değiştirilemez (Yaga vd., 2019). Eğer üçüncü bir kişi zincirdeki bir veriyi değiştirmeye kalkışırsa hash (özet) fonksiyonları sayesinde yapılan değişiklik anında fark edilerek zincirde değiştirilmiş olan o kısım ve sonrası geçersiz ilan edilmektedir. Ağın tutarlılığının tamamı, gelişmiş şifreli algoritmalarla korunduğundan denetlenme

süreci her zaman otomatik olarak geri planda devam etmektedir. Daha önce doğrulaması yapılıp kaydedilmiş olan işlemlerin kronolojik sıralamasını isteyen herkes ağın kayıtlarını açıp görebilmektedir.

Klasik istemci sunucu mimarisinde kredi kartı ile yapılan alışverişlerde geri planda her şey bankanın ve satışı yapan kurumların sistemleri üzerinde devam etmektedir ve yapılan tüm işlemler sadece ilgili birkaç kurumun ve bankanın veri tabanlarına yazılmaktadır. Blok zincir ağı üzerinde gerçekleştirilen alışverişlerde ise dağıtık olarak yer alan ve o ağa bağlı olan bütün bilgisayarlara yani düğümlere işlemler kaydedilerek yazılmaktadır. Ağdaki her düğümde kayıtların bir kopyası yer almaktadır. Tüm işlem geçmişi blok zincirin üzerinde saklanmaktadır (Yermack, 2019). Şifreli, dağıtık ve şeffaf bir ağ mimarisine sahip olan blok zinciri sayesinde kurumlara, kişilere vb. aracılara olan güven ihtiyacı artık ortadan kalkmaktadır.

Blok zinciri, dijital ödeme sistemleri, para transferleri, finansal hizmetlerde menkul, gayrimenkul tüm piyasalarda kullanılmaktadır. Blok zincir altyapısı ile aradaki aracıları çıkartarak komisyonu ortadan kaldırılmaktadır. Alıcı ve satıcı işlemleri hızlandırılmaktadır. İşlemlerin onaylanması akıllı kontratlar kullanılmaktadır. Tedarik zinciri ve lojistik uygulamalar süreç işlemleri içermektedir. Aşamalı olarak bir ürünün bir yerden bir başka yere taşınması, ürünün özellikleri, alınması, sigortalanması, süreçlerin takibi çok yüksek maliyetlere neden olmaktadır. Blok zinciri sayesinde bu maliyetler ve süreç arasında geçen bekleme süreleri de en aza indirilmektedir. Sağlık alanında sağlık sigortalarının çok hızlı bir şekilde onaylanması ve sağlık hizmeti alan kişinin en mahrem bilgilerinin sadece o kişide kalması da yine blok zinciri ile belirlenebilmektedir. IoT nesnelerin interneti, makineler ve sensörler arasında blok zinciri kullanılmaktadır. Kitle fonlaması, yardım bağışları, şirket hisseleri gibi alanlarda blok zinciri kullanılmaktadır. Akıllı kontratlar ile yapılan yardımları, bağışları, hisselerin durumlarını takip edilebilmek çok kolaylaşmaktadır. Fikri ve mülkiyet haklarını korumak için dijital bir noter gibi blok zincir kullanılmaktadır. Devlet kurumlarında tapu, araç ruhsatı, vergi, evlilik, cüzdan, doğum belgesi, kimlik kartı, diploma kayıtları gibi işlemler için blok

zincirini kullanılmaktadır. Böylece bürokratik işlemler hızlanmaktadır. Uluslararasında pasaport ve vize gibi bilgiler için kullanılmaktadır. Mülteciler için global bir kimlik ve tapu kayıtlarını uluslararası olarak tutulması için blok zinciri kullanılmaktadır. Sosyal ağlarda kişilerin aktifliğe göre platformlardan kazançlar etmesi için kullanılmaktadır. Paylaşımlı ekonomide Uber, Airbnb, Ebay gibi sistemler kullanıldığında kurumlara komisyonlar ödenmektedir. Bu komisyoncuları aradan çıkarıp yerine araya akıllı sözleşmeler girmektedir. Blok zinciri teknolojisi ile yapabileceğimiz projeler, hayal gücünüz ile sınırlı olmaktadır.

2. LİTERATÜR ÖZETİ

Nakamoto (2008), Satoshi Nakamoto isimli biri ya da birileri tarafından Bitcoin'in çıkış amacını ve ne işe yaradığını anlatan Bitcoin a Peer to Peer Electronic Cash System (Bitcoin Eşler Arası Elektronik Nakit Sistemi) isimli bir makale yayımlandı. Satoshi Nakamato'nun kim olduğu bilinmiyor. Kripto para dünyasında bir projeyi anlatan makaleye Whitepaper (Beyaz kâğıt) adı verilmektedir. Bitcoin'i anlatan makalede şu anki bankacılık sistemi ve güvene dayalı bazı sorunlar dile getirilmiştir. En ciddi sorun olarak da aracı kişi ve kurumlara olan güvenme ihtiyacı öne çıkarılmıştır. Bitcoin ile kişilere ve kurumlara güvenme gerekliliğini ortadan kaldırıp eşten eşe güvenli bir sistem getirildiğinden söz edilmiştir. Güvenin yerini kriptolojik algoritmaların ve şifreleme kanıtlarının aldığından söz edilmektedir. 2009 yılında ilk Bitcoin bloğu üretilmiştir. Bir blok zincir ağındaki ilk bloğa genesis adı verilmektedir. 12 Ocak 2009'da ağın 170. bloğunda ilk Bitcoin transferi gerçekleştirildi.

Buterin (2014), Vitalik Buterin adında bir Rus programcı tarafından başka bir makale daha yayımlandı. Ethereum blok zinciri ilk defa Smart Contract (Akıllı Sözleşmeler)den ve Decentralized Applications (DApps, Merkezi Olmayan Uygulamalar) kavramlarından bu makalede yer almaktadır. Makalede özellikle Bitcoin'in 2008 yılında bankacılık sistemlerinde aracıları aradan çıkarıp güven problemini çözmeye çalıştığını ama bunun yeterli olamayacağını çünkü bankacılık sistemleri dışında da insanların güven ihtiyacı yine devam ettiğini ve bu işlemleri otomatikleştirmek ve güven altına almak için bir kontrat tasarladıklarını ve bu sözleşmeleri blok zincirinde taşıyacak bir mimari geliştirildiğinden söz edilmiştir. Akıllı sözleşmeler sayesinde her alanda güvene olan ihtiyaç sorunu da çözümlenmektedir. 2014 yılında Ethereum blok zincir ağı Vitalik Buterin tarafından herkesin kullanımına açıldı.

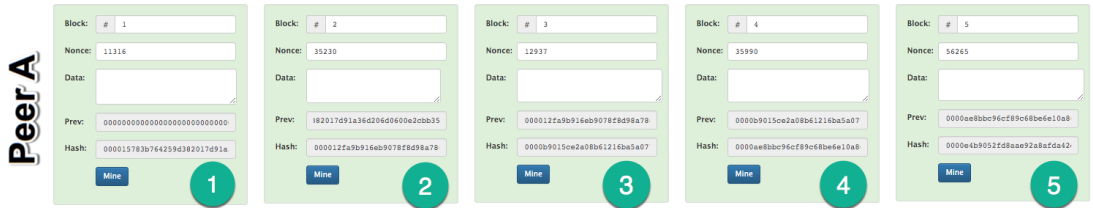
Morin vd. (2021) Blockchain teknolojisi için 2017 yılı çok önemlidir çünkü birçok yeni platform ve kripto para da bu zamanlarda ortaya çıkmıştır. Geliştirilen blok zinciri platformları kendi kripto paralarının bir kısmını yatırımcılara ucuz bir fiyattan pazarlayarak kendilerine maddi gelir toplamışlardır. Buna ICO (Initial

Coin Offering) ilk para teklifi ya da ilk para arzı denilmektedir. Şimdiye kadar binlerce coin (para) ve token (jeton) çıktı ve halka arz edilmiştir.

3. BLOK ZİNCİRİ TEKNOLOJİSİNİN ALTYAPISI

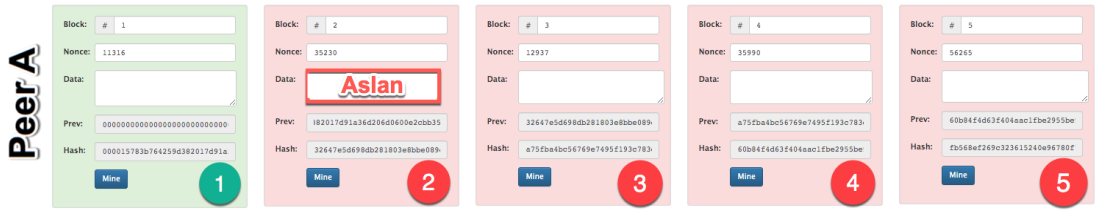
3.1. Blok Zincirinin Karakteristiği

Blok zincirinde veriler Şekil 3.1'deki gibi bloklarda sıralı halde tutulur. Kayıt edilen blokların içeriği ve sırası immutable (değiştirilemez) olarak kayıt altına alınmaktadır.



Şekil 3.1. Blok zinciri blokları (Brownworth, 2021)

Herhangi bir bloğu alıp değiştirmeye çalışıldığında bunun son derece zahmetli bir girişim olduğunu görülecektir çünkü o bloktan sonra gelen diğer bütün blokların da sırasıyla doğrulanıp değiştirilmesi zorunludur. Şekil 3.2'de Peer A (Eş A) adındaki zincirde 5 tane blok bulunmaktadır. Zincirdeki sadece 2. blok değiştirildiğinden o ve ondan sonra gelen tüm bloklardaki verilerin tutarlılığı bozulmaktadır. Ağdaki zincirin en güncel halini içeren düğümlerden biri değiştirilmiş bloğu denetlediği anda yapılan değişimi fark edip bütün ağı bundan haberdar etmektedir.



Şekil 3.2. Zincirde değişiklik yapılan 2. bloktan sonrası (Brownworth, 2021)

Şekil 3.3'te zincirde değişiklik yapılan 2. bloktan sonraki diğer tüm bloklarında geçerliliğini tutarlı hale getirilmesi gerekmektedir. Blok zinciri ağındaki madenciler2. bloktan itibaren zincirin sonuna kadar sırasıyla doğrulama işlemler yapmaktadırlar.

Şekil 3.3. Madenci düğüm ve blok kontrolü (Brownworth, 2021)

Ağa bağlı olan madenci makineler sadece kendi zincirlerini değil, ağdaki tüm zincirleri denetlemektedirler. Zincirde yer alan tüm bloklar baştan sonra geri planda kontrol edilmektedir. Zincire en son eklenen birkaç tane blokta bu değişimi yapmak o kadar masraflı değildir ama binlerce bloğun sıralı olduğu bir zincirde ortalardan bir bloğu seçilirse ve ondan sondan sonrasında da tek tek sırasıyla doğrulamak yaptırmak işte bu çok masraflı bir süreçtir. Madenci bir düğüm kendi makinesindeki veriler üzerinde istediği gibi değişiklikler yapabilmektedir ama bu değişim sadece tek bir zincirde yapmak ne bitmemektedir. Blok zincir ağındaki diğer madenci düğümlere ulaşıp sırasıyla bu değişiklikleri onların üzerinde tutulan zincirlerde de yapmak gerekmektedir. Eğer blok zincir ağının %51'inde başarılı olarak değişiklik yapılırsa o zaman bloktaki değişim bütün ağ tarafından onaylanıp kabul edilmektedir (Yang vd., 2019).

Blok zinciri birden çok düğüm içermektedir. Her düğümde ağa ait zincirlerin bir kopyası tutulmaktadır. Bir düğümdeki zincirde eğer bir değişiklik yapılsa ağdaki diğer düğümler o değişikliği anında tespit etmektedir. Şekil 3.4'te gösterildiği gibi 3 farklı madenci düğümde tutulan aynı blok zincirleri yer almaktadır. Madenci bir makine zincirindeki veride değişiklik yaptığında ağdaki diğer makineler bu değişiklikten asla etkilenmemektedir.

Şekil 3.4. Sırasıyla zincirde doğrulaması yapılan bloklar (Brownworth, 2021)

Değişiklik yapılan bir düğüm, sistem tarafından güvensiz ve dürüst olmayan bir düğüm olarak ilan edilmektedir. Dürüst olmayan bir düğüm, kendini geçerli olan zincire göre güncellemedikçe o ağa yeni bir blok eklemesine asla izin verilmemektedir. Dağıtık mimari sayesinde blok zincirindeki verilerin tutarlılığı ve devamlılığı sağlanmaktadır.

Günlük yaşamda iki taraf kendi aralarından bir iş yaparken eğer birbirlerini tanıyorlarsa aralarındaki güvene göre yola çıkıp adım atabilirler ama birbirlerini hiç tanımıyorlarsa o zaman aracı olarak bir otoriteye ihtiyaç duymaktadırlar. İşte o zaman aralarında sözleşme imzalamak için noter ve avukatlar araya girmektedir. Bazen de güvendikleri bir kuruma giderler o kurumda aracı olup onların iş yapmalarını sağlamaktadır. Taraflardan biri alıcı diğeri de satıcı olsun ve ikisinin de aracılık yapan bu kuruma bir ödeme yatırma şartı bulunsun. Eğer aracı kurum yarın çekip giderse iki tarafta zarar etme ile karşı karşıya kalmaktadır ve arada aracı bulunsa bile yine de ciddi bir güven problemi söz konusu olmaktadır. Blok zinciri trustless (güvene ihtiyaç duymayan) bir mimaride çalışmaktadır. Aynı kişiler blok zinciri ağında iş yapacak olsalardı asla aracıya gerek ihtiyaçları olmayacaktı. Ne kendilerine ne de aracılara güvenmek zorunda kalmayacaklardı. Güvene olan ihtiyaç blok zincirinde altyapıda şifreli algoritmalar tarafından sağlanmaktadır.

Blok zincirinin bir sahibi bulunmaktadır. Ağdaki düğümlerin birbirine üstünlükleri yoktur ve düğümler çalıştığı müddetçe blok zinciri ağı da var olmaya devam etmektedir.

Blok zincir ağı transparent (şeffaf) bir yapıda çalışmaktadır. Ağ üzerinde yapılan işlemleri herkes görülebilmektedir. Ağın herkes tarafından görülebilmesi onun güvensiz olduğu anlamına gelmemektedir. Ağda kimsenin kişisel verileri gösterilmemektedir. Yapılan işlemlere ait hash kodları, işlemleri yapan cüzdanların adresleri, işlemlerin tutulduğu blok bilgileri, bloğu ekleyen madencinin cüzdan bilgisi gibi açıkta duran genel bilgiler gösterilmektedir. Blok zinciri ağında işlem yapan kişiler cüzdan bilgisi temsil edilmektedir.

3.2. Hash Fonksiyonları

Blok zincirinde hash fonksiyonları Şekil 3.5'teki gibi yapılan işlemlerin şifrelemesi için kullanılmaktadır. Şifreleme algoritmaları kritik resmi doğrulama işlemleri için tercih edilmektedir (Harrison vd., 2016). Hash fonksiyonunun girişine veri olarak yazı, video, resim ve başka dosya çeşitlerinin verilmesi mümkündür. Girişe her ne türde ve büyüklükte veri verilirse verilsin, çıkıştaki digest (sonuç) değeri hep aynı uzunlukta ve benzersiz bir değer olmaktadır.

Şekil 3.5. Hash (özet) fonksiyonuna giren değer ve çıktısı (Veness, 2017)

Üretilen her hash kodu değeri bir parmak izi gibi benzersiz olmaktadır. Hash fonksiyonuna aynı giriş değeri verildiği müddetçe hep aynı çıkış değeri elde edilmektedir. Giriş kısmındaki veride küçücük bir değişiklik bile yapılsa çıkıştaki sonuç değeri de anında değişmektedir. Blok zinciri platformlarında bir ağa daha önceden kaydedilmiş işlemlerin sonradan değişip değişmediğinin takibi altyapıda hash fonksiyonları ile yapılmaktadır.

Blok zincir ağının üzerinde gerçekleştirilen her işlem ve her blok için benzersiz bir hash kodu değerleri oluşturulmaktadır. Blok zinciri platformlarının kullandıkları kripto paraların mimari tasarımları birbirinden farklı olduğundan altyapılarındaki SHA (Secure Hash Algorithm, Güvenli Hash Algoritması) türleri de farklı olmaktadır. Blok zinciri platformları geri planda SHA-2, SHA2, SHA3, Keccak gibi hash algoritmalarını kullanmaktadır. Blok zincirinde hash fonksiyonları yapılan işlemleri doğrularken ve yeni bir bloğu zincire eklerken kullanılmaktadır. Bitcoin SHA-256, Ethereum'da Keccak-256'yı kullanmaktadır. Hash fonksiyonları tek yönlü çalışırlar ve 1 girişe sadece 1 çıkış değeri vermektedirler. Çıkış değerinden girişi elde edilememektedir. Şekil 3.6'daki gibi giriş kısmından 2 çeşit saldırı ile çıkış değeri tahmin edilmektedir. Bir girişe sadece benzersiz bir tane çıkış değeri verilmektedir ama Collision Attack (Çakışma Atağı) saldırısında birden fazla girişe aynı çıkış değeri verilmektedir. Çakışma saldırında aynı çıktı hash'ine yol açan iki veya daha fazla mesaj bulmaktır (Alodat vd., 2018). Bu durumda o hash fonksiyonu artık güvenilirliğini kaybetmektedir.

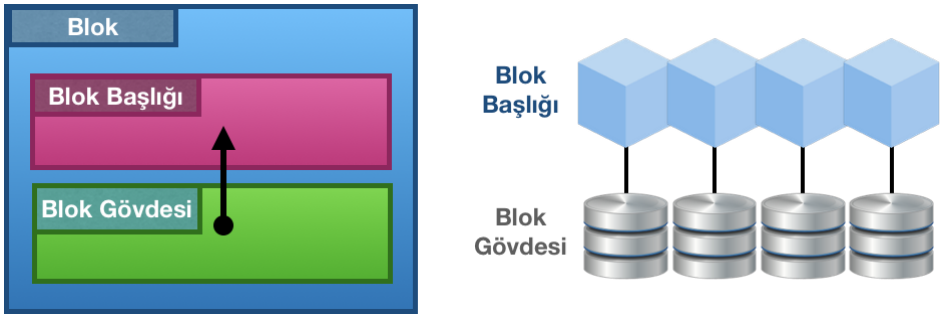
Şekil 3.6. Hash fonksiyonunda çakışma atağı (Alodat vd., 2018)

Şekil 3.7'deki gibi hash fonksiyonun çıkış değeri biliniyorsa girişe rastgele değerler verilerek giriş değeri tahmin edilmektedir. Verilen bir hash'den giriş değerini geri elde etmek Preimage Attack (Ters Görüntü Kümesi Atağı) ile çok zor olmaktadır (Alodat vd., 2018). Girişteki verinin türü video, yazı, resim gibi olabilir ama bu bile bilinmediğinden dolayı ters görüntü kümesi atağı yıllarca sürebilmektedir.

Şekil 3.7. Hash fonksiyonunda ters görüntü kümesi atağı (Alodat vd., 2018)

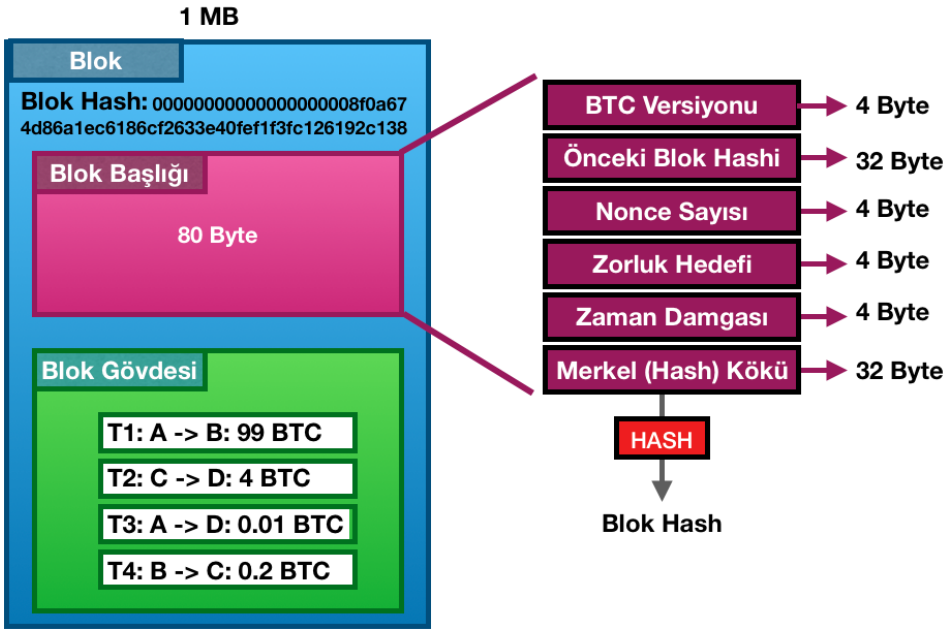
3.3. Block (Blok) Kavramı

Blok zincirinde bir blok, belirli bir zaman diliminde ağa eklenilmek için oluşturulan kapasitesi sınırlı bir kutucuk ile temsil edilmektedir. Blok içerisinde başlık, gövde kısmı ve hash değerleri bulunmaktadır (Doğa, 2020).



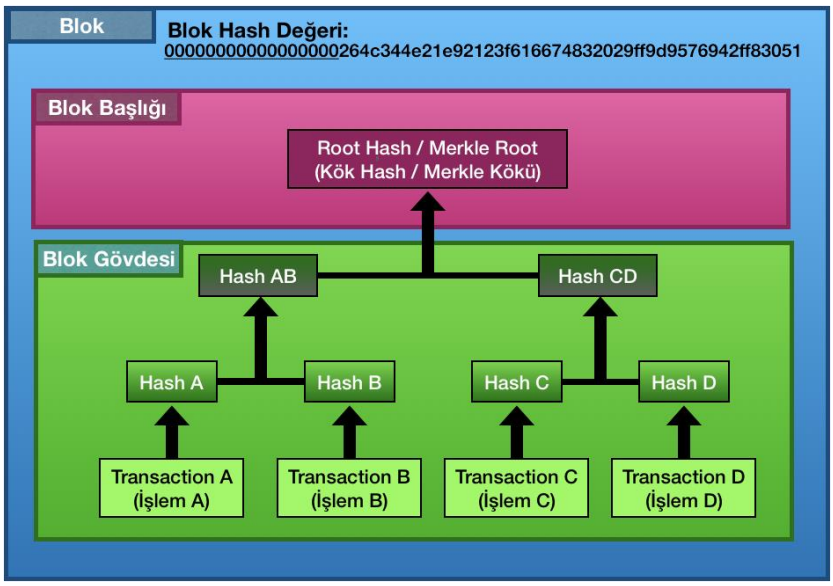
Şekil 3.8. Bir bloğun başlık ve gövde kısımları (Edureka, 2021)

Şekil 3.9'da gösterildiği gibi bloğun başlık kısmında ağın versiyonu, bir önceki bloğa ait hash değeri, nonce sayısı (tek seferlik kullanılan), zorluk hedefi, oluşturulma zamanı, Merkle hash kökü değeri yer almaktadır. Blok gövdesi verilerin tutulduğu alandır ve içerisinde doğrulaması yapılmış transactions (işlemler) bulunmaktadır. Bloklar belirli bir işlem depolama kapasitesine ile sınırlandırılmıştır. Bitcoin bloğunun kapasitesi 1 MB olarak belirlenmiştir. Blok içerisindeki tüm değerler hash fonksiyonuna sokulmaktadır ve en son elde edilen değer bu bloğun hash değeri olmaktadır. Blok zincirinde her blok bir önceki bloğun hash değerini içermektedir ve bu sayede blok içerisindeki tutulan işlemlerin manipüle edilmesini zorlaşmaktadır (Yang vd., 2019).



Şekil 3.9. Blok zincirindeki bir bloğun iç kısımları (Sayeed ve Marco, 2019)

Bloğun gövde kısmında yer alan tüm işlemlerin hash çıktısına Merkle kökü değeri denilmektedir. Ağdaki her madenci düğüm, yapılan işlemlerden bazılarını alıp onaylamakta ve kendine ait bir blok içerisinde bunları biriktirmektedir. Blok başlığı ve gövdesi Merkle kökü ile birbirine bağlanmaktadır. Şekil 3.10'da gösterildiği gibi madenci tarafından blok zincirine eklenmek istenen yeni bloğun hash değerini sırasıyla blok versiyonu, önceki bloğun hash değeri, nonce sayısı, zorluk hedefi, zaman damgası ve Merkle kökünün hash fonksiyonuna girişinin sonucu belirlemektedir. Her bloğun başlık ve gövde kısmının arasında şifreli bir ilişki bağı kurulmaktadır.

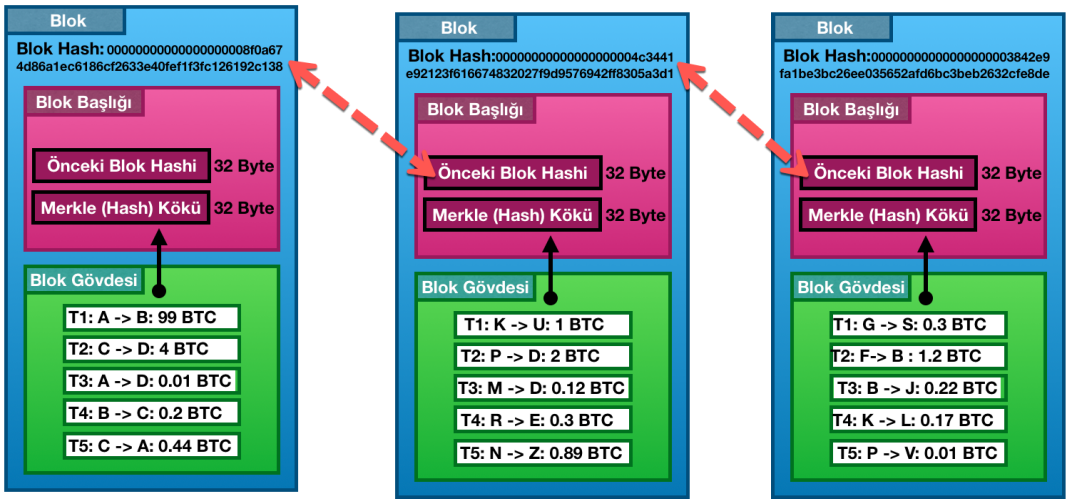


Şekil 3.10. Blok gövdesindeki işlemler ve Merkle kök değeri (Edureka, 2021)

Blok zincir ağına yeni bir blok eklemek için altyapıda bir sayı tahmin etme oyunu düzenlenmektedir. Ağdaki madenci düğümler tarafından bu yeni bloğun üstünde yer alması gereken nonce sayı değerini tahmin etmek için yarış aşamasına geçilmektedir. Nonce sayı değerini ilk bulan madenci yeni bloğu kendi zincirine ekleme hakkını elde etmekte ve bir miktar kripto para ödülünün de sahibi olmaktadır. Ağdaki diğer madenciler bulunan bu nonce sayısını kontrol ederek zincirlerine yeni bloğu eklemektedir. Bir madenci düğümün ağdaki görevi, yapılan işlemlerin geçerliliğinin doğrulanması, onların hash fonksiyonuna sokulması, yeni bloğa ait nonce değerinin bulunması ve zincire yeni bloğun eklenmesidir.

Blok zincirine yeni bir blok eklenirken bir önceki blok dikkate alınmaktadır. Şekil 3.11'deki gibi her blok kendinden bir önceki bloğun hash değerini kendi içerisinde tutmakta ve bu sayede zincir kurulmaktadır. Bir bloktaki işlemlerde bir değişiklik yapıldığında kendisinden sonra gelen bütün bloklardaki hash değerleri değişmektedir ve o bloklar artık geçersiz olmaktadır. Blokları zincire dahil etmek için yeniden madencilerin yapılan işlemleri sırasıyla hash fonksiyonlarına sokması gerekmektedir. Daha sonra Merkle kökü değeri hesaplanıp bloğun kendi hash değeri yeniden oluşturulmalıdır. Zinciri bozan bloktan sonraki bütün bloklarda bu hesaplamanın sırasıyla yapılması gerekmektedir. Yapılan bu

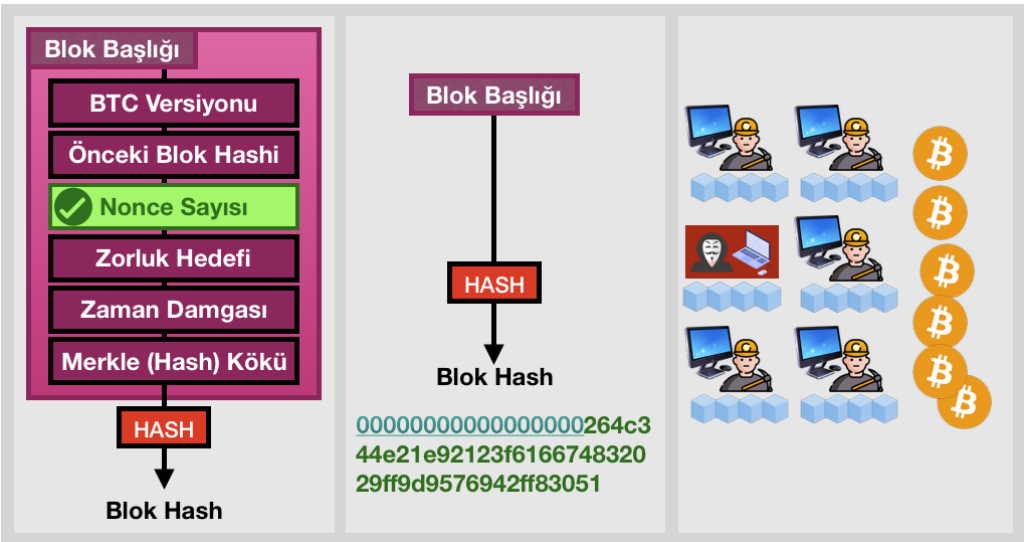
değişikliğin bütün ağda geçerli sayılabilmesi için ağdaki blok zincirlerinin %51'inde de bu doğrulama adımların yapılması gerekmektedir.



Şekil 3.11. Blok hash değerleri ve blokların bağlanması (Sayeed ve Marco, 2019)

Blok gövdesinde sadece ağda gerçekleştirilen veriler bulunmaktadır. Bloklar birbirine blok başlığı kısımlarından bağlanmaktadır.

Şekil 3.12.'de gösterildiği gibi bir madenci düğümün ağda teşvik ödülünü kazanabilmesi için diğer madenciler ile yarışıp eklenecek olan yeni bloğa ait nonce sayısını bulması gerekmektedir.



Şekil 3.12. Nonce değeri ve madenci düğümler (Sayeed ve Marco, 2019)

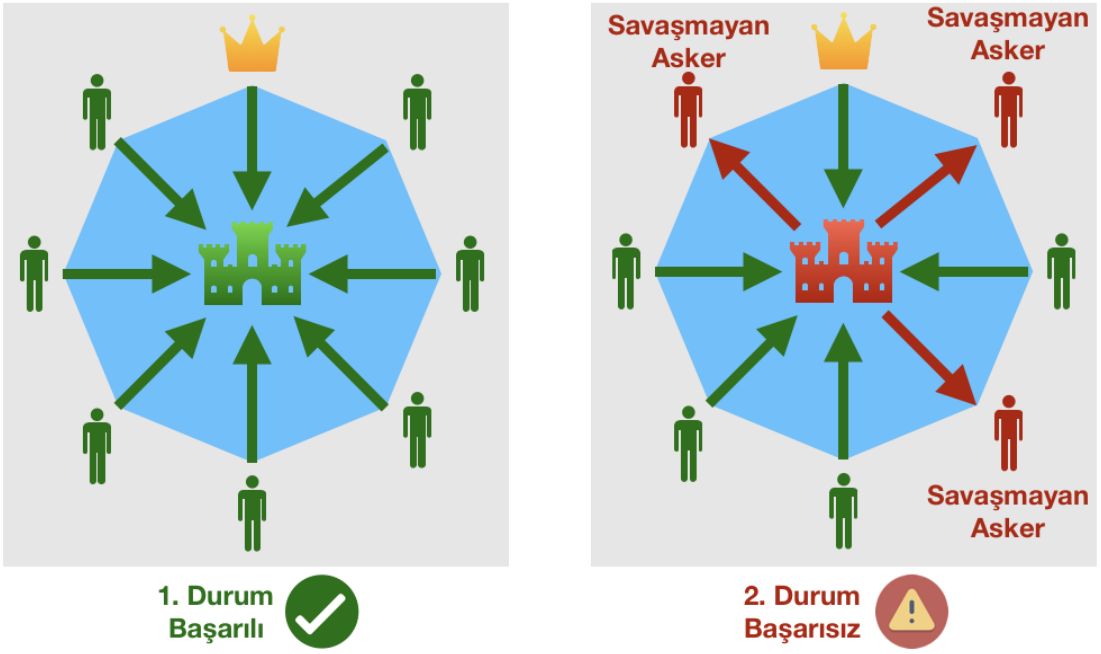
Nonce sayısını bulan madenci düğüm ağın tutarlı olarak ayakta kalmasını sağladığından alt yapıdaki blok zincirinin sistemi tarafından bir miktar kripto para ile ödüllendirilmektedir. Bloğun hash değeri de nonce sayısı bulunduktan sonra oluşturulmaktadır.

3.4. Konsensüs Mekanizmaları

Blok zincir ağındaki her bir düğümde dağıtık bir kayıt defteri tutulmaktadır. Bu defterin daima güncel ve tutarlı olması gerekmektedir (Doğa, 2020). Ağda sürekli olarak yeni işlemler yapılmaktadır. Yeni bloklar oluşturulup güvenli bir şekilde zincire eklenmesi gerekmektedir. Blok zincir ağına herkes özgürce dahil olabilmektedir. Ağda madenci düğüm olmak için hiç kimseden izin alınma şartı aranmamaktadır. Herkes ağa dahil olabildiğinden ve önceki yapılmış tüm işleri makinesine indirebilmektedir. Bazı düğümlerin sahipleri bu veriler üzerinde değişiklik yaparak ağa zarar vermek düşünmektedir. Bazen art niyetli bir kişi kendi oluşturduğu işlemleri ve blokları ağa ekleyip zinciri ele geçirmeyi amaçlamaktadır. Ağa yapılan saldırıları önlemek için sistemde kullanılan mekanizmanın adına consensus (mutabakat) algoritması denilmektedir. Bitcoin'de kullanılan mutabakat mekanizması Proof of Work (PoW) iş kanıtıdır. Diğer blok zinciri platformlarında farklı mutabakat algoritmaları kullanılmaktadır.

3.4.1. Byzantine generals problem (Bizanslı generaller problemi)

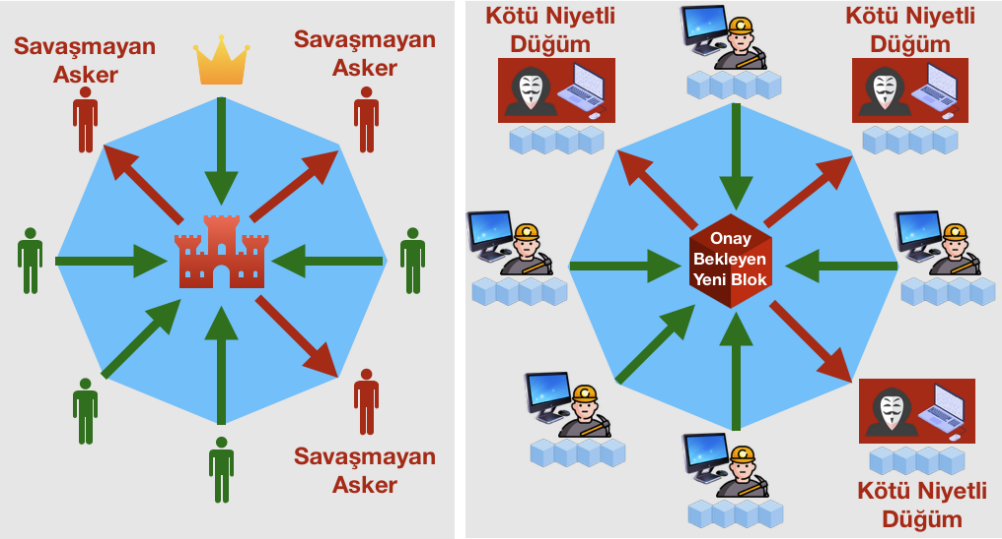
Bizanslı generaller sorunu, bir ordunun birden fazla tümeniyle aynı anda yapılacak bir saldırının zamanlaması konusunda fikir birliğine varması gereken dağıtılmış bir dizi generalin anlaşmasını ifade etmektedir (Kuo vd., 2019). Şekil 3.13'te gösterildiği gibi bir şehri kuşatan bir ordu bulunmakta ve her general ordunun bir kısmını komuta etmektedir. Eğer generaller aynı anda şehre saldırırlarsa şehri alabilmektir. Generaller, haberleşmeyi sadece ulaklar ile sağlamaktadır. Eğer iletişimde bir hata, kopukluk olursa ya da biriler saldırıya katılmayıp kaçarsa o zaman şehri yine alınamamaktadır.



Şekil 3.13. Bizanslı generaller probleminin durumları (Kuo vd., 2019)

Generallerin, saldırı ya da geri çekilme hamlesinde mutlaka bir fikir birliğine varmaları gerekmektedir. Güvene dayalı ve doğru iletişim sorununa literatürde Bizanslı generaller problemi denilmektedir.

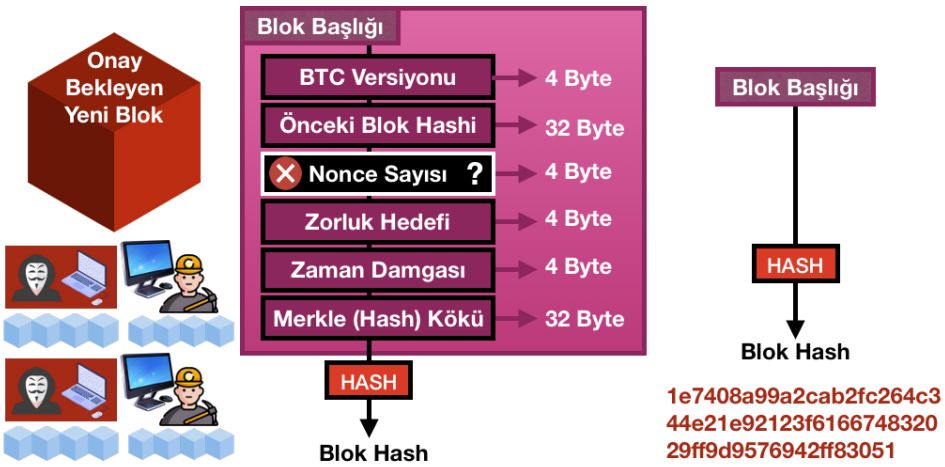
Şekil 3.14'teki Bizanslı generaller probleminde yaşanan iletişim ve güven sıkıntıları blok zinciri ağına güvenli bir şekilde yeni bir blok ekleme sorunu ile birebir benzemektedir. Blok zincirine yeni bir blok eklenilmeyi beklemektedir. Bloğun içinde çok değerli veriler yani mesajlar yer almaktadır. Bu mesajların kötü niyetli kişilerin eline geçmeden ve değiştirilmeden doğru bir şekilde blok zincirine eklenilmesini amaçlanmaktadır. PoW (iş kanıtı) konsensüs mekanizması tam olarak bu probleme çözüm sunmaktadır.



Şekil 3.14. Bizanslı generaller problemi ve iş kanıtı (Kuo vd., 2019)

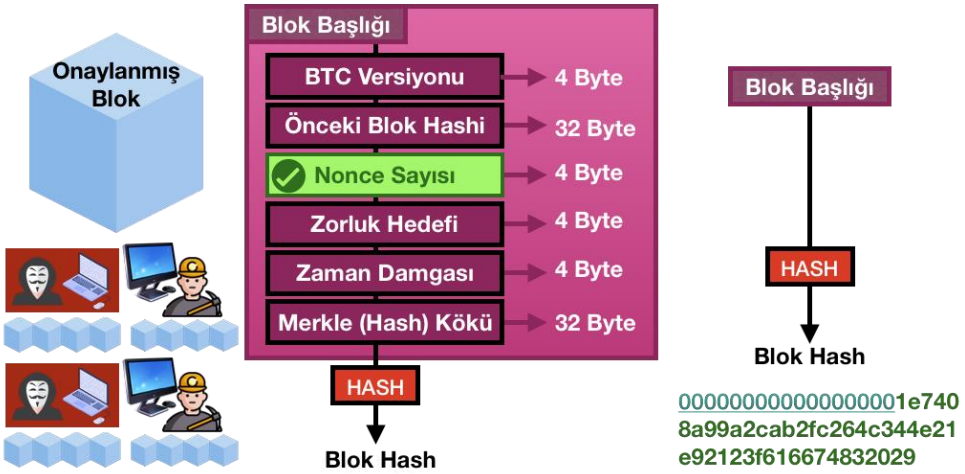
3.4.2. Nonce sayısı, difficulty (zorluk), halving (blok ödülü)

Altın her yede bolca bulanan ve çok kolay çıkarılan bir maden değildir. Bitcoin'de altın gibi değerli olmayı hedeflemektedir. Bitcoin kripto parası 21,000,000 adet ile sınırlandırılmıştır. Kripto paraların kullandıkları doğrulama mekanizmaları tarafından üretilme miktarı ve üretilme hızı daima denetlenmektedir. Blok zincirine yeni bir bloğun eklenilmesi için blok başlığındaki nonce (number only used once) tek seferlik kullanılan sayı değeri geri planda iş kanıtı olarak kullanılmaktadır (Nakamoto, 2008). Nonce hesaplama yaklaşımı kripto paranın üretilme hızının dengelenmesi için kullanılmaktadır. Madenciler tarafından bir nonce sayısı bulunurken sıralı olarak brute-force (kaba kuvvet) araması yapılmaktadır. (Adewumi ve Liwicki, 2020). Şekil 3.15'te blok başlığı bir hash fonksiyonundan geçirildikten sonra blok hash değeri oluşturulmaktadır.



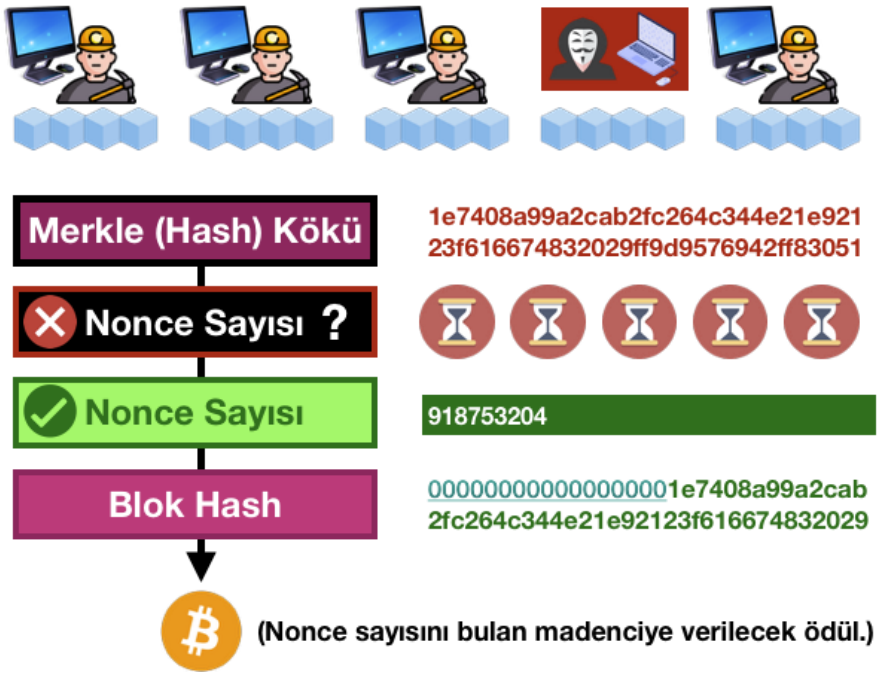
Şekil 3.15. Nonce ve onay bekleyen bloğun hash değeri (Sayeed ve Marco, 2019)

Eğer nonce değeri olmazsa blok başlığındaki hash değeri 64 karakterli sıradan bir hash çıktısı olmaktadır. Şekil 3.16'da gösterildiği gibi blok hash değeri ve nonce sayısı alıp tekrardan hash fonksiyonuna sokunca başında sıfırlar bulunan yeni bir hash çıktısına dönüştürmektedir ve bu yeni bloğun ağa eklenirken ki kullanacağı hash değeridir. Kripto paraların zincirlerine eklenmiş olan her bloğun hash değeri bu nedenle sıfırlarla başlamaktadır. Blok başlığının hash değerinde kullanılan sıfır adeti kripto paranın altyapısındaki mimariye göre değişmektedir. Bitcoin'de sıfır adeti 17 tanedir ama bir başka kripto parada farklı olabilmektedir. Nonce değerini ilk bulan madenci düğüm, buluğu sayı değerini blok zincir ağına bildirmektedir. Diğer madenci düğümler tarafından da bulunan nonce değeri test edilmektedir.



Şekil 3.16. Nonce ve onaylanmış bloğun hash değeri (Sayeed ve Marco, 2019)

Şekil 3.17'de gösterildiği gibi ağdaki diğer madenciler tarafından bulunan nonce sayısının doğruluğuna onay verilmektedir. Nonce sayısını bulan madenci yeni bloğu önce kendi zincirine eklemekte ve incentive (teşvik) ödülünü kazanmaktadır. Daha sonra diğer düğümlerdeki madenciler de bu yeni bloğu alıp kendi kayıt defterlerine eklemektedir.



Şekil 3.17. Nonce sayısı ve teşvik ödülü ilişkisi (Weimeng, 2019)

Bir blok zincirinde yer alan ilk bloğa genesis denilmektedir. Bir blokta yer alan ilk işleme ise coinbase denilmektedir. Genesis bloğunun zincire eklenilmesi için harcanan emek ile sonradan eklenen diğer bloklar için harcanan emek seviyesi aynı değildir. Ağa daha fazla madenci katıldıkça ağdaki Difficulty Target (Zorluk Hedefi) değeri konsensüs mekanizması tarafından ayarlanmaktadır ve blokların sabit sürelerde üretilmesi sağlanmaktadır (Weimeng, 2019). Satoshi Nakamoto, Bitcoin sistemini tasarlarken ağdaki madenci sayısının artacağını ve teşvik ödülüne olan ilginin zamanla yükseleceğini bilmekteydi. Bu sebeple blok üretim durumunu ayarlamak için bazı kurallar belirlemiştir. Bitcoin projesinin çalışmaya başladığı 3 Ocak 2009 tarihinden itibaren zincire eklenen her yeni blok için 50 Bitcoin üretilip madencilere Bitcoin Halving (Blok Ödülü) olarak vermekteydi (Meynkhard, 2019). Çizelge 3.1'de gösterildiği gibi Bitcoin

altyapısında her 210,000 blokta bir bu teşvik ödülünün yarılanma kuralı yer almaktadır.

Çizelge 3.1. Bitcoin blok ödülü ve yarılanma yılları değerleri (Meynkhard, 2019)

Ödül yarılanma yılı	Blok sayısı	Verilen BTC ödülü
2009	210,000	50
2012	420,000	25
2016	630,000	12.5
2020	840,000	6.25
2024	1,050,000	3.125
2028	1,260,000	1.5625
2032	1,470,000	0.78125
2036	1,680,000	0.390625
2040	1,890,000	0.1953125
2044	2,100,000	0.09765625
2048	2,310,000	0.048828125
2052	2,520,000	0.0244140625
2056	2,730,000	0.01220703125
…	…	…
2140	6,930,000	0

Yeni bir bloğun zincire eklenmesi ortalama 10 dakikaya ayarlandığından her 210,000 bloğun üretim süresi yaklaşık 4 yıla karşılık gelmektedir (Usta ve Doğantekin, 2017).

Şekil 3.18'de Bitcoin zincirindeki ilk bloğun içinde yer alan zorluk diğeri gösterilmektedir.

Hash	000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
Confirmations	721,825
Timestamp	2009-01-03 20:15
Height	0
Miner	Unknown
Number of Transactions	1
Difficulty	1.00
Merkle root	4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b
Version	0x1
Bits	486,604,799
Weight	1,140 WU
Size	285 bytes
Nonce	2,083,236,893
Transaction Volume	0.00000000 BTC
Block Reward	50.00000000 BTC
Fee Reward	0.00000000 BTC

Şekil 3.18. Bitcoin Genesis (ilk blok) ve zorluk değeri (Blockchain, 2022)

Bitcoin blok zincirindeki ilk bloğun zorluk değeri 1 olarak belirlenmiştir. Blok zincirine yeni bloklar eklendikçe zorluk derecesinin değeri de artmaktadır (Meynkhard, 2019).

Çizelge 3.2'de ağdaki blokların sayısı ve zorluk derecesi arasındaki ilişki gösterilmektedir. Bitcoin'de mutabakat mekanizması tarafından ağın zorluk derecesi her 2016 blokta bir güncellenmektedir (Usta ve Doğantekin, 2017).

Çizelge 3.2. Bitcoin blokları ve zorluk değerleri (Blockchain, 2022)

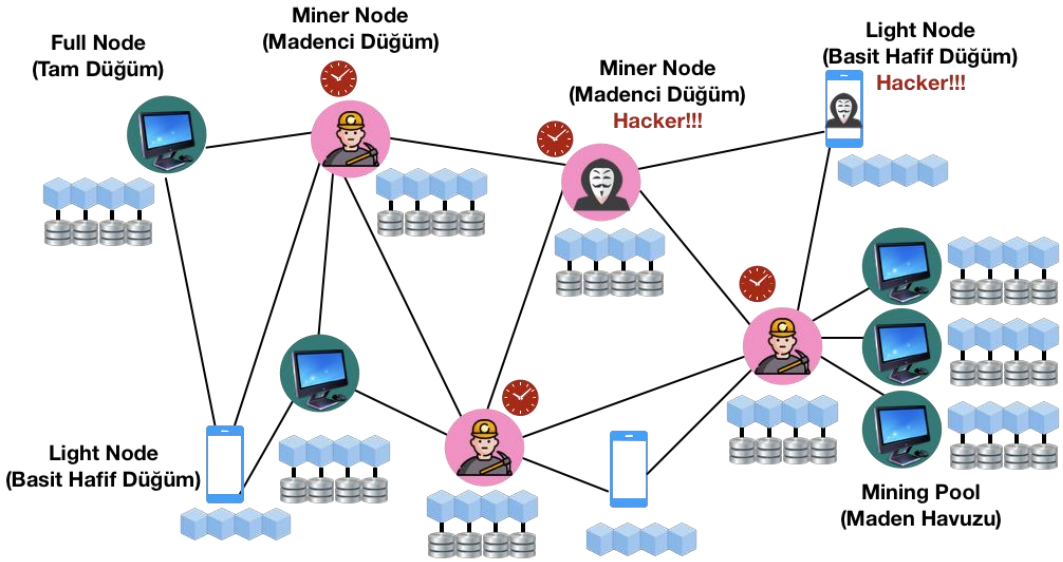
Blok sayısı	Zorluk derecesi
0	1
40,000	1.82
50,000	6.09
60,000	16.62
70,000	181.54
80,000	712.88
90,000	3,091.74
100,000	14,484.16

150,000	1,468,195.43
175,000	1,626,553.48
200,000	2,864,140.51
500,000	1,873,105,475,221.61
700,000	18,415,156,832,118.24
720,000	26,643,185,256,535.47

Zorluk değeri ağdaki tüm madenciler için geçerli olan ortak bir değerdir.

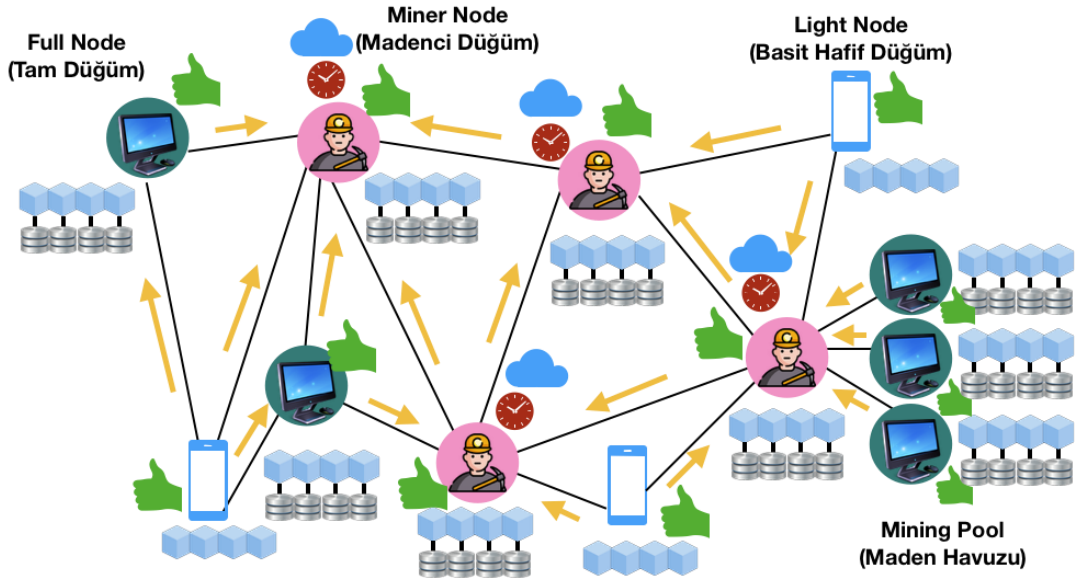
3.4.3. Node (düğüm) çeşitleri

Blok zincir ağı eşten eşe çalışmaktadır. Ağdaki hiçbir düğümün diğer bir başka düğüme üstünlüğü bulunmamaktadır. Şekil 3.19'da gösterildiği gibi her düğüm, ağın içinde hem istemci hem sunucu de olabilmektedir. Düğümler, kendilerinde blok zincir defterinin kayıtlarını barındırmaktadır. Bu defter, bloklardan meydana gelmiştir. Ağdaki ilk bloktan en sonda yer alan düğüme kadar tüm zinciri tutan birimlere Full-Node (Tam Düğüm) denir. Tam düğümlerde başlık ve gövde kısımları birlikte yer almaktadır. Tablet, telefon gibi çok güçlü olmayan cihazlar blok zincirinde blokların sadece başlık kısımlarını tutabilmektedir. Sadece başlık kısımlarını içeren düğüme Light Node (Basit Düğüm) denilmektedir. Basit düğüm ağda tam düğümlere bağlı olarak çalışabilmektedir. Herhangi bir işlem geldiğinde tam düğümlerden sorgulamalar yaparak bu sorgulamaların neticesinde yapılan işleme onay verebilmektedir. Blok zincir ağı ayakta tutan, yapılan işlemleri denetleyip, yeni blokları zincire ekleyen düğümlere Miner Node (Madenci Düğüm) denilmektedir. Madenci düğümler tek başlarına çalışmaktadır. Başka madenciler ile birlikte hareket edip çalışan madencilere Mining Pool (Madenci Havuzu) denilmektedir.



Şekil 3.19. Blok zincir ağındaki düğümler (Myung ve Lee, 2020)

Şekil 3.20'de gösterildiği gibi blok zincir ağına yapılan bir istek eğer bir tam düğüme gelmişse isteğin kimden geldiğini ve ne yapmak istediği kontrol edilmektedir.



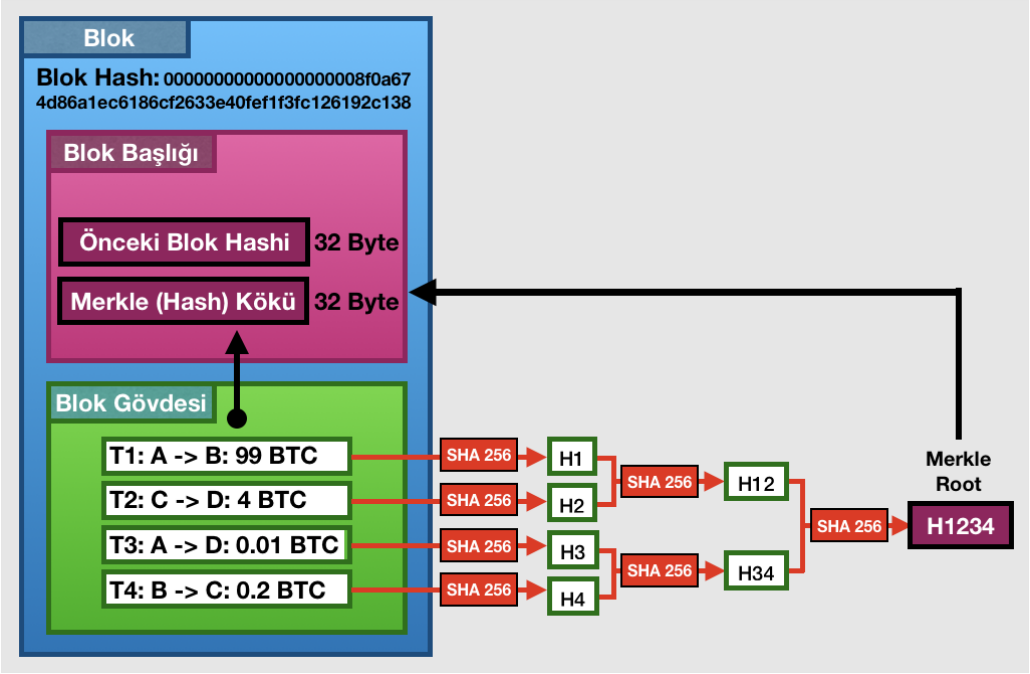
Şekil 3.20. Blok zincir ağında bir işlemin yolculuğu (Myung ve Lee, 2020)

Basit düğümler gelen işlemleri sorgulayabilmektedir. İşlem eğer doğrulanırsa o zaman madencilerin havuzuna bu istek yazılmaktadır. Madenci bir düğüm o

işlemi doğrulayıp yeni bloğunun içerisinde depolamaktadır. Madenci son adımda ise nonce değerini bularak yeni bloğu zincire eklemektedir.

3.4.4. Merkle tree (Merkle ağacı)

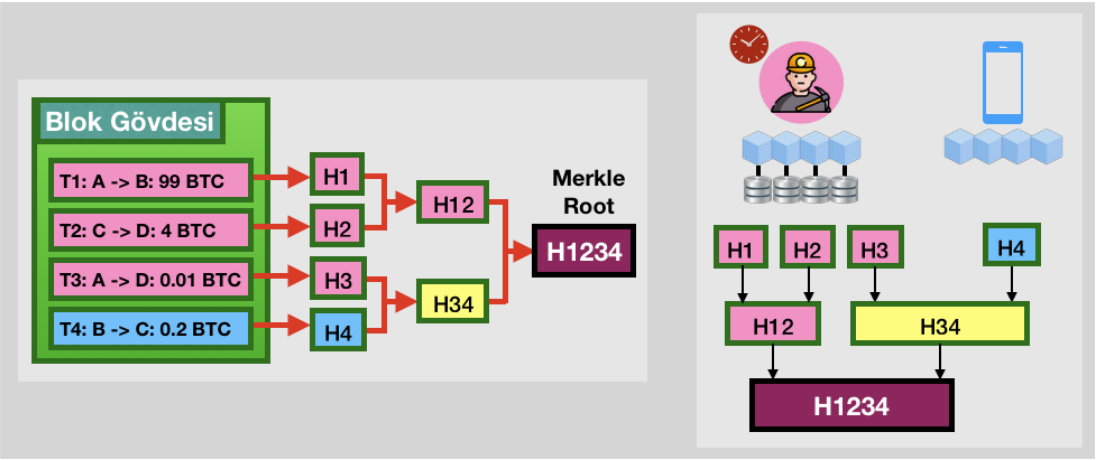
Merkle Tree (Merkle Ağacı), kriptolojik bir veri yapısı olarak çalışmaktadır. Blok gövdesindeki işlemler sırasıyla hash fonksiyonlarına sokulmaktadır. Hash fonksiyonlarından geçirilmiş tüm işlemlerin sonucuna Merkle kökü denilmektedir. Şekil 3.21'de gösterildiği gibi Merkle kökü, blok başlığı kısmında yer almaktadır. Her işlemin tek tek sırasıyla hash kodu alınmaktadır. Sonuçlar H1, H2, H3, H4 gibi belirtilmektedir. Sonra H1 ile H2 kendi arasında hash fonksiyonuna sokulmaktadır.H12 olmaktadır. H3 ile de H4 aynı işleme tabi tutulmakta ve sonuç H34 olmaktadır. En son adımda ise H12 ve H34 hash fonksiyonuna sokulmaktadır ve H1234 çıktısı elde edilmektedir. Tek tek başlayıp ikişer ikişer hash fonksiyonlarına alınma bir ağaca benzediğinden buna Merkle ağacı adı verilmiştir. Merkle ağacının en sonda elde edilen H1234 değerine ise Merkle kökü denilmektedir. Blok gövdesinde yapılacak olan en ufak bir değişiklikte Merkle kökü değeri de değişmektedir.



Şekil 3.21. Merkle Tree (Merkle Ağacı) (Waldman, 2018)

Madenci bir düğüm, blok başlıklarını da gövdelerini de içermektedir. Kendisine gelen işlemleri sıralayıp doğrulanmasını yapmaktadır. Ağda yeni bir bloğun oluşturulması hızlandırmak için ağa telefon ve tablet gibi cihazlarla bağlanmış olan basit bir düğüm de doğrulama işlemine katkı verebilmektedir. Basit bir düğüm bunun yapmak için tam düğümün üzerinden madenci bir düğümün gövde kısmına ulaşmaktadır. Henüz hash değeri elde edilmemiş işlemleri alıp tek tek hash fonksiyonuna sokup gövdeye yazmaktadır. Basit bir düğümün bunu yapması, madenci düğüme zaman kazandırmaktadır. Bir bloğun kapasitesi dolana kadarda madenci düğüme yardım etmeye devam etmektedir. Bu adımların hepsi geri planda konsensüs mekanizmasına ait algoritmalar tarafından otomatik olarak gerçekleştirilmektedir. Bu hash işlemleri yapanlara da ağda yine bir miktar teşvik ödülü verilmektedir.

Şekil 3.22'de madenci gövdesinde doğrulanmayı bekleyen 4 tane işlem yer almaktadır. H1 ve H2'yi madenci düğüm tarafından doğrulamaktadır. Sonucu H12 olarak yazılmaktadır. Bir basit düğümde bu madenciye yardım için bağlanıp sadece H4 işlemlerini doğrulamaktadır ve H3 ile H4'ü de yine hash işlemine sokmaktadır. Elde ettiği H34 sonucunu madenci düğümdeki bloğun gövde kısmına yazmaktadır. Basit düğüm H12 ile H34'ü de doğrulayıp Merkle kökünü değerini de madenci düğüme yazmaktadır.

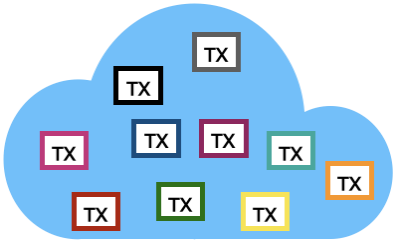


Şekil 3.22. Madenci ve basit düğümün birlikte çalışması (Waldman, 2018)

Madenci de yeni blok ekleme aşamasına böylece geçip nonce sayı değerini bulmaya odaklanmaktadır. Eğer yarışı kazanırsa teşvik ödülünü kazanacaktır. Neticede algoritma basit düğüme de emeklerinin karşılığını bir pay vermektedir.

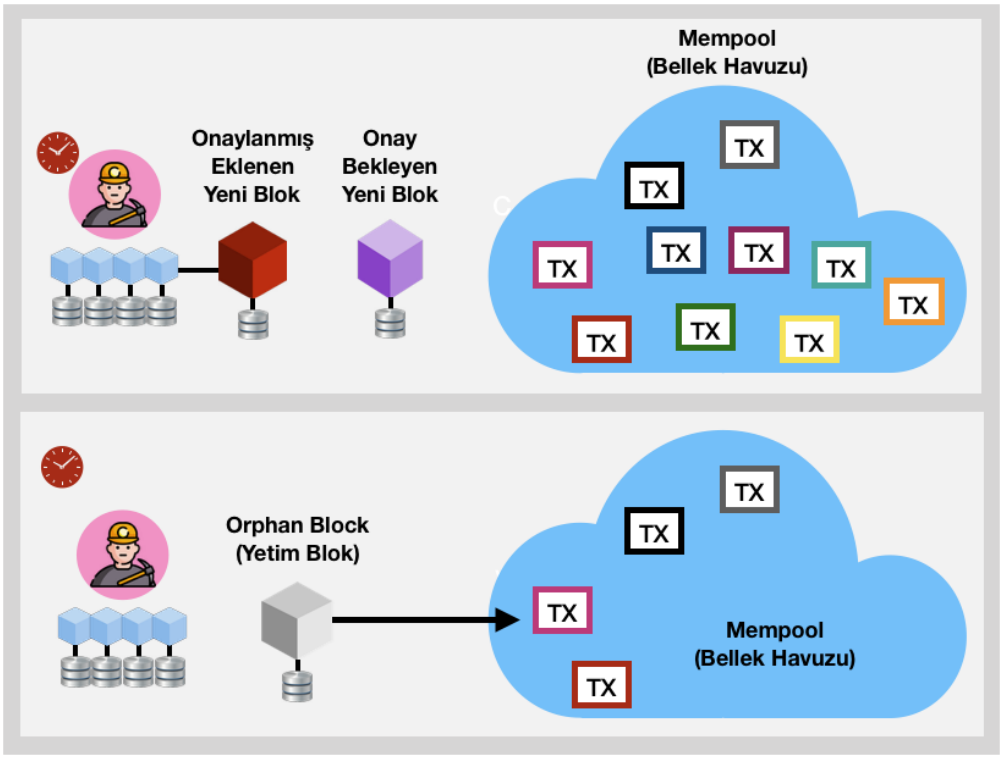
3.4.5. Mempool (bellek havuzu)

Mempool (Bellek havuzu) içerisinde henüz Unconfirmed Tx (onaylanmamış işlemler) tutulmaktadır. Bu işlemler onay aldıkça yeni eklenebilecek aday bloğun içerisine alınmaktadır. Şekil 3.23'te gösterildiği gibi aday blok henüz blok zincirine eklenmediğinden sadece o madenci düğümde yer almaktadır. Bir istek, ağa bildirildikten sonra en yakındaki düğüm onu almaktadır. Aynı işlem, diğer madencilerden bazılarına da gitmektedir. Madenciler tarafından vakit kaybedilmeden doğrulama işlemi başlatılmaktadır. Grup halinde çalışan madenciler gelen işlemleri kendi bellek havuzlarına alıp orada tutmaktadır. Doğrulanan işlemleri bir bloğun içinde biriktirip onu en son adımda da blok zincir ağına eklemeye çalışmaktadırlar. Birden fazla madenci aynı anda farklı işlemleri doğrulayabilmektedir. Bu da demek oluyor ki eklenmek için oluşturulan ama henüz eklenmemiş bütün blokların içlerinde aynı işlemlerden de olabilmektedir. Bazı işlemler aynı olsa bile eklenmek istenen aday blokların hepsi tamamen birbirinden farklı olmaktadır. Madencilerin bellek havuzunun da bir kapasite sınırı bulunmaktadır. İşlemler, madenci tarafından onaylanıp blok zinciri ağına eklenene kadar bellek havuzu içerisinde tutulmaktadır (Imtiaz vd., 2020). Blok zincir ağında yaptığı işlem için yüksek ücret ödeyen kişilerin istekleri doğrulama için her zaman öncelikli olmaktadır. Madencilerdeki blok zincirleri birebir aynı bloklardan oluşmaktadır. Blok havuzlarında zincire eklenmek için oluşturulan aday bloklar ise tamamen birbirlerinden farklı işlemleri içermektedirler.



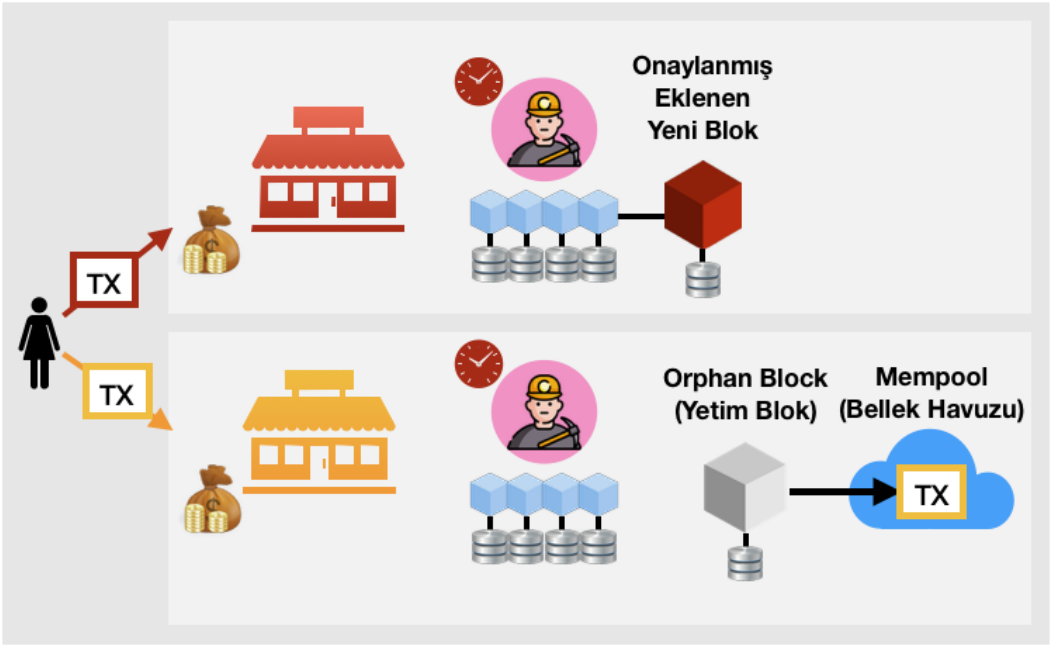
Şekil 3.23. Mempool (Bellek havuzu) (Imtiaz vd., 2020)

Blok zincir ağına yeni bloğu hangi madenci eklerse en uzun zincire sahip olduğundan diğer madenciler de ondaki zincire göre verilerini güncellemektedirler. Uzun olan zincir her zaman üstündür ve o takip edilmektedir. Ağdaki bir başka madenci de bazı işlemleri doğrulayıp yeni bloğu kendi makinesindeki zincire eklemektedir. Bunu daha ağa haber veremeden bir başka madenci ağa yeni bir blok ekleyebilmektedir. Bu durumda geç kalan madenci düğüm, eklediği bloktaki işlemleri bellek havuzuna geri iade edip en uzun zinciri takip etmek zorunda kalmaktadır. Şekil 3.24'de gösterildiği gibi yarışı kaybeden madencilerin kendi düğümlerindeki zincire ekledikleri aday bloğa Orphan (Yetim) blok denilmektedir. İçinde doğrulanmış işlemler olsa bile yine de bellek havuzuna tüm işlemler geri gönderilmektedir. Eğer yeni eklenen blokta iade edilen işlemlerden bazıları yer almakta ise o zaman onlar da bellek havuzundan temizlenmektedir. Eklenilmeyen işlemlerin tekrardan madenciler tarafından alınıp hash fonksiyonlarına sokulup doğrulanması gerekmektedir. Zincire eklenen bir işlem ikinci defa doğrulanmak için bellek havuzundan silinmektedir. Yarışı kaybeden madencilerin hepsi en uzun zincire sahip olan madencinin eklediği en güncel bloğa göre zincirlerini güncellemektedirler.



Şekil 3.24. Orphan (Yetim) blok (Imtiaz vd., 2020)

Bir kişi bir mağazadan ürün satın almaktadır. Bu bir işlem, bellek havuzuna yazılmaktadır. Bu işlem madenciler tarafından doğrulanmayı beklemektedir. İşlem henüz onaylanmadığından bu kişi bir başka mağazadan da yeni bir ürün daha almakta olsun. Şekil 3.25'te gösterildiği gibi şimdi aynı para ile 2 defa harcama işlemi yapıldığını görmekteyiz. Buna double spending problem (çift harcama sorunu) denilmektedir. Blok zincirinde kayıt altına alınan veriler asla değiştirilemediğinden çifte harcama sorunu önlenmektedir (Patil ve Sangeetha, 2021).



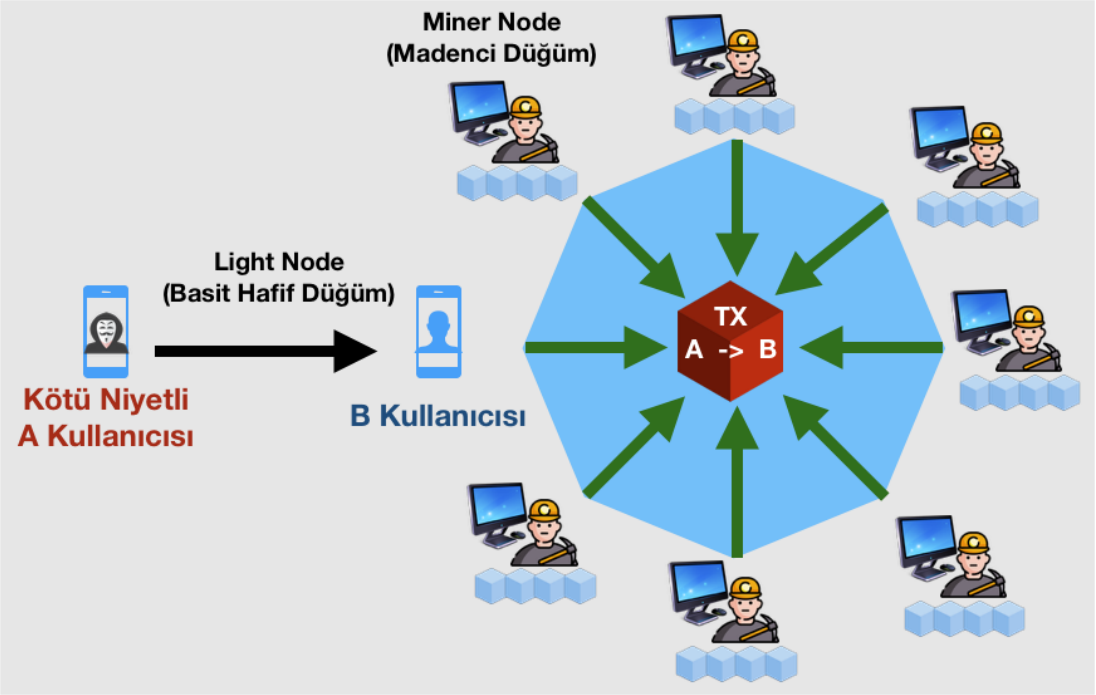
Şekil 3.25. Double spending (Çift harcama) (Patil ve Sangeetha, 2021)

Farklı 2 madenci de bu işlemleri kendi zincirlerinde doğrulayıp eklemektedir. İlk ekleyenin işlemi zincirde kabul edilmektedir. Diğer madencinin aday bloğu artık yetim bir bloktur ve içindeki işlemler de bellek havuzuna geri iade edilmektedir. Bir diğer madencinin bellek havuzunda da yine aynı harcama bu işlem yer alabilmektedir çünkü henüz bellek havuzundan yapılan işlem silinmeden durmaktadır. O madenci de yeni blok ekleme sırasında az önce doğrulama yapmak istediği işlemin zaten blok zincirinde yer aldığını fark etti anda o zaman çift harcama yapıldığına karar vermektedir. O zaman da işlemi bellek

havuzundan silmektedir ve aynı parayı iki defa harcamak istene kişi de sistemde sakıncalı kişi olarak işaretlenmektedir.

3.4.6. Anahtar çifti

A kişi bir işlem yapmak istediğinde blok zincirine bunu bildirmektedir. İşlem öncesi gerçekten isteği yapan A kişi mi diye kontrol edilmektedir. Şekil 3.26'da gösterildiği gibi Ağa bağlanan dışarıdan biri ya da düğümlerden biri art niyetli olarak A kişi gibi hareket edip istekte bulunmaya düşünmüş olabilmektedir. Bu kontrolü yaparken anahtar çiftlerinden yararlanılmaktadır.



Şekil 3.26. Blok zincirinde kimlik kontrolü (Kuo vd., 2019)

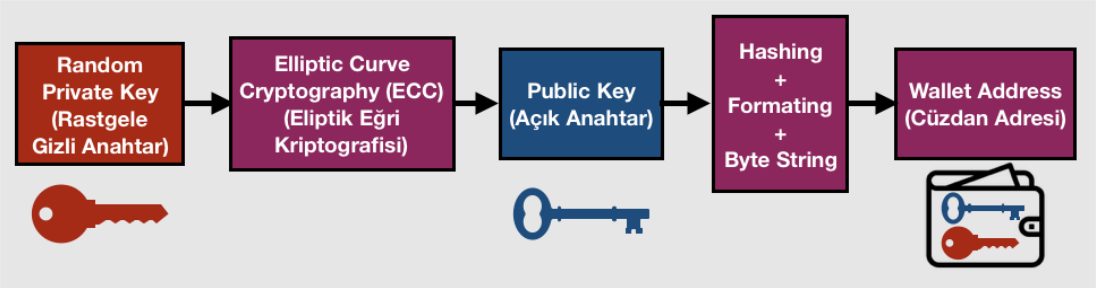
Şekil 3.27'de gösterildiği gibi kimlik doğrulaması için A kişisi, B kişisine çok gizli ve önemli bir mesajı göndermek istemektedir ve mesajın giderken hiçbir şekilde değiştirilmemesi gerekmektedir. Mesajı bir kutuya koyup kapatmaktadır. Kutuda 2 kapağı yer almaktadır. İç kısımdaki camdan bir kapaktır. A kişisi mesajını camdan kapağın altına okunacak şekilde yerleştirmektedir. Sandığı açan B kişisi camın altında duran mesaj okuyabilmektedir ama asla değiştirememektedir. A

kişisi mesajı herkesin yolda giderken okumaması için de kutunun dışında duran kapağı da kapatmaktadır.

Şekil 3.27. Anahtar çifti ve kimlik doğrulaması (Waldman, 2018)

Mesajı sadece alan B kişisinin okuması için de böylece koruma altına almaktadır. A kişi kendi özel anahtarı ile hem cam kapağı hem de dıştaki kapağı kilitlemektedir. Alıcı B kişisi ise kendinde var olan başka bir anahtar ile kutunun sadece dıştaki kapağını açabilmektedir ve camın altındaki mesajı da böylece okuyabilmektedir. Eğer sandığın dışındaki kapak açılmazsa ve camdaki mesaj okunmuyorsa demek ki mesajda problem meydana gelmiştir yani değiştirilmiş demektir.

Şekil 3.28'de gösterildiği gibi bir kişi blok zincir ağındaki hesabına sadece kendisinde bulunan Private Key (Gizli Anahtar) ile erişebilmektedir. Bir kişi blok zincir ağına girip bir başka hesaba bir şey gönderirken gizli ve Public Key (Açık Anahtar) kullanmaktadır. Özel anahtar rastgele üretilmektedir ve Elliptic-Curve Cryptography (ECC, Eliptik Eğri Kriptografisi) adındaki bir şifreleme fonksiyondan geçirilmektedir. Açık anahtar elde edilmektedir. Açık anahtar çok uzun olduğundan o da hash fonksiyonundan geçirilmektedir. Elde edilen sonucu formatlama fonksiyonlarından geçilerek de daha kısa bir hale getirilmektedir. Buna da cüzdan adresi denilmektedir.



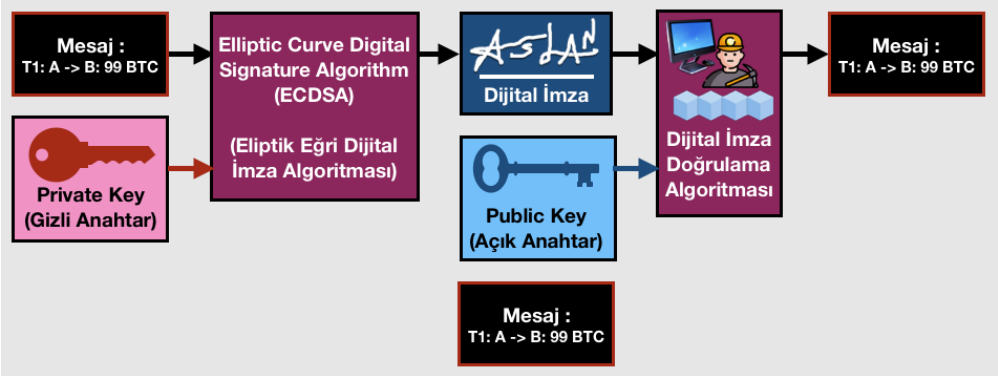
Şekil 3.28. Blok zinciri cüzdan adresi (Waldman, 2018)

Private Key (Gizli Anahtar) örneği, (Dragon, 2021)
493fa332c103acfd19bc025be03604e6f9bd7a089c03643d0868259a4c985067

Public Key (Açık Anahtar) örneği, (Dragon, 2021)
040048c64cf4bd552574064fca7f819c712cca9acc587306efa062ee595a883cd1
e4e2d88d68a7852a7c3119755724c126df8fde4065e2f93b96c7098e64402c2d

BTC cüzdan adresi örneği, (Dragon, 2021)
17gj197eLUAwD71Kjd9ygRPoy96oVB4LnU

Şekil 3.29'da gösterildiği gibi A kişisi, B kişisine 99 BTC göndermek istemektedir. Mesaj, gönderilirken özel anahtar ile kitlenmektedir. Özel anahtar çok önemlidir ve asla hiç kimse ile paylaşılmamalıdır. Blok zinciri için doğrulama kritik olduğundan Elliptic Curve Digital Signature Algorithm (ECDSA, Eliptik Eğri Dijital İmza Algoritması) ile yüksek güçlü bir genel ve özel anahtar çifti oluşturma stratejisi kullanılmaktadır (Waldman, 2018). Kilitleme işlemi için mesaj ve özel anahtar eliptik eğri dijital imza algoritması içerisinden geçirilmektedir. Daha sonra A kişi, bu dijital imzayı, kendine ait açık anahtarı ve ileteceği mesaj ile birlikte bir düğüme doğrulama yapması için göndermektedir. Bunu alan düğüm dijital imzayı ve genel anahtarı bir doğrulama algoritmasına sokmaktadır. Elde edilen sonuç eğer gönderilmek istenen mesaj ile eşleşiyorsa o zaman doğrulama işlemi başarılı olarak yapılmış demektir. Mesajı gönderen kişinin kimliği de böylece ağdaki algoritmalar tarafından bu şekilde doğrulanmaktadır. Mesajı gönderen tarafın özel anahtarı hiç kimsenin eline geçmemelidir.



Şekil 3.29. Mesaj ve gönderen kişinin doğrulanması (Waldman, 2018)

3.4.7. Blok zinciri tipleri

Bitcoin ve Ethereum ağları açıktır ve herkes katılabilmektedir. Ağ tamamen dağıtıktır. Ağın herkese açık olması güvensiz olduğu anlamına gelmemektedir. Ağdaki herkesin işlemleri açıkça görülmektedir. Cüzdanların sahiplerine ait kişisel bilgileri ise bilinmemektedir.

Consortium (Birlik) blok zinciri herkese açık değildir. Kurumsal projeler için uygun olmaktadır. Örneğin, birkaç sigorta şirketi, hastaneler, eczaneler bir araya gelerek kendi aralarında bir blok zincir ağı kurup bunun üzerinden müşterilerine hizmet verebilmektedir. Bu blok zincirine herkes girememektedir çünkü girmek için sistemdeki yetkililerden izin almak gereklidir.

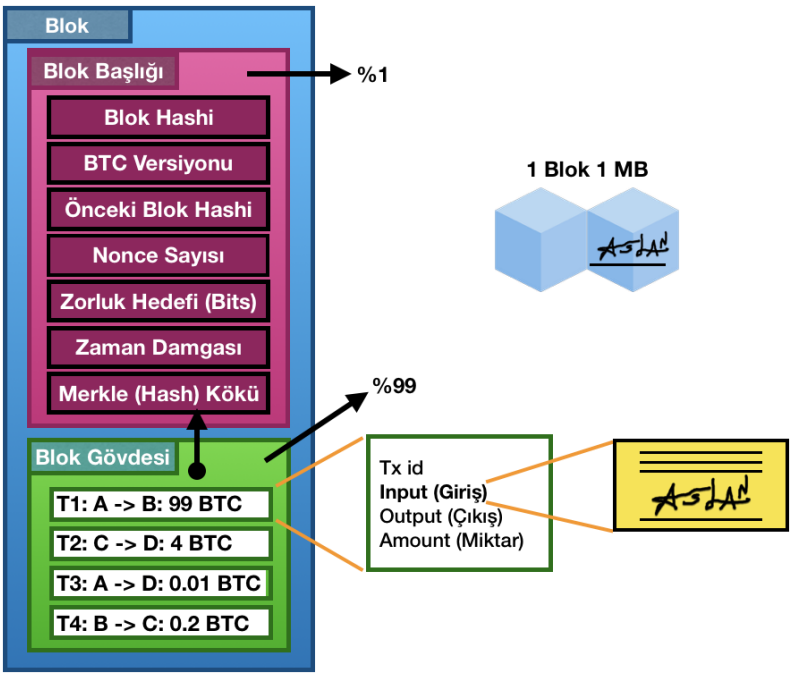
Private (Özel) blok zinciri sadece bir kurumun ve müşterilerinin kullandığı blok zinciri olmaktadır. İstemci sunucu mimarisindeki merkezi yapıya çok benzer gibi görünse bile altyapısı tamamen farklı teknolojilerden oluşmaktadır. Bu zincir herkese açık değildir ve giriş için sistemdeki yetkililerden izin alınması gerekmektedir.

3.4.8. Ölçeklenme çözümleri

Bitcoin blok zincirinde, 1 blok 10 dakikada eklenmektedir. Bloğun boyutu sadece 1 MB ile sınırlandırılmıştır. Saniyede ise sadece 7 işlem yapabilmektedir. Yeni bir blok zincire eklendikten sonra onun onaylanması için de 6 blok bekleme süresi

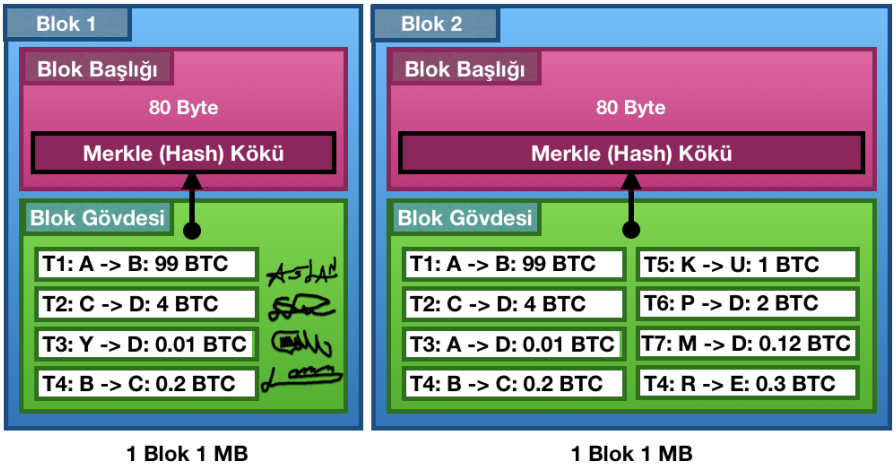
gerekmektedir. Kesin bir doğrulama 600 saniye sürmektedir (Bowden vd., 2018). Kullanıcı sayısı arttıkça bu bekleme süreleri daha da artmaktadır. İnsanlar işlemlerinin onaylanma süresini kısaltmak ve boş yere beklememek için bu sefer çok yüksek işlem ücretleri ödemek zorunda kalmaktadır. Visa saniyede 1700, PayPal 170, Bitcoin Cash 56, Bitcoin ise 7 işlem işleye bilmektedir. Bitcoin'in ölçekleme sorunlarına çözüm için SegWit (Segregated Witness, Ayrılmış Tanık), Bitcoin Cash ve Lightning çözümleri sunulmuştur. SegWit ve Lightning Network (Hafif Ağ) gibi yaklaşımlar, blok zinciri uygulamalarının ölçeklenebilirlik sorunlarını çözmeye çalışmaktadır (Ehmke vd., 2018).

Blockchain'de bir bloktaki gövde kısmında yer alan işlemler bloğun neredeyse %99'unu kaplamaktadır. Şekil 3.30'da gösterildiği gibi bir işlemin içerisinde Tx id (İşlem numarası), Input (Giriş), Output (Çıkış) ve Amount (Tutar) değerleri yer almaktadır. Giriş kısmında dijital imza tutulmaktadır. Mesaj, özel anahtar ile hash fonksiyonuna sokulduktan sonra artık dijital imza olmaktadır. Dijital imza da bir işlemin yaklaşık %65'lik kısmını kaplamaktadır. Bir blok toplamda 1 MB alana sahip bu da demek oluyor ki bloğun yarıdan fazlasını sadece dijital imzalar tarafından işgal edilmektedir.



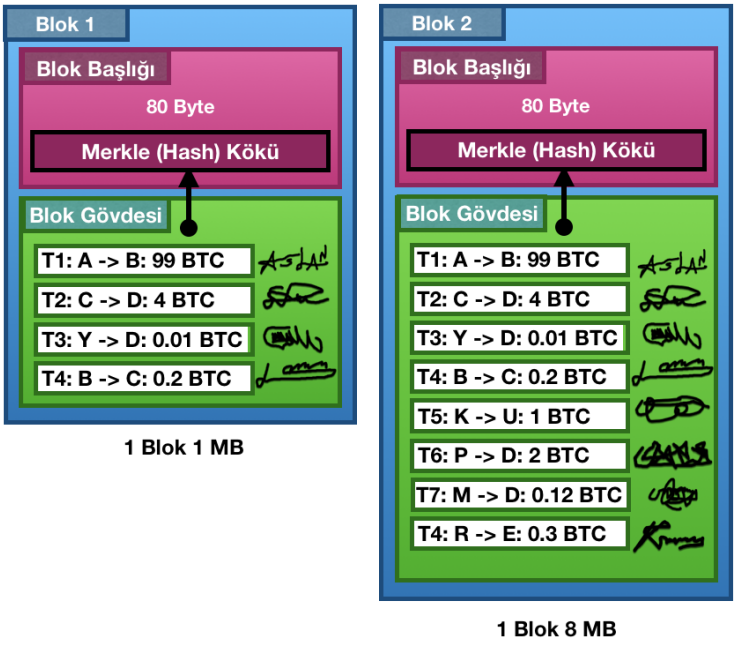
Şekil 3.30. Bloktaki işlemlerin dijital imzası (O3schools, 2021)

Şekil 3.31'de gösterildiği gibi SegWit (Ayrılmış Tanık) yaklaşımı bir bloğun içinde yer alan dijital imzaları bloğun dışına alıp bloğun içine sadece yapılan işlemleri yazarak daha çok işlemi bloğa eklemeyi hedeflemektedir. Bitcoin blok zinciri 2017 Ağustos ayından itibaren güncellenerek SegWit entegre edilmiştir.



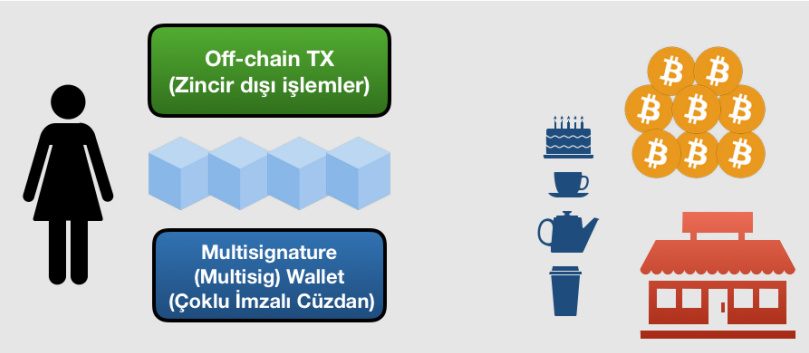
Şekil 3.31. SegWit yaklaşımı ve dijital imzalar (Bitlo, 2018)

Bitcoin Cash'in sevenleri tarafından SegWit çözümünü pek beğenilmemiştir. Onlar da Bitcoin zincirinden tamamen ayrılıp yollarına kendilerine ait yepyeni bir mimari üzerinden devam etmeyi tercih etmişlerdir. Bitcoin Cash, Bitcoin zincirinden türetilen yeni bir zincir olmaktadır. Bitcoin Cash'in kendine ait yeni bir mimarisi olduğundan yeni bir coin'in de sahibi olmaktadır. Şekil 3.32'de görüldüğü gibi Bitcoin Cash blok boyutunu 1 MB yerine 8 MB olarak değiştirmektedir. Bu sayede de saniyede işlenebilecek işlem sayısını yaklaşık 8 kat arttırmayı başarabilmektedir.



Şekil 3.32. Blok gövde boyutu ve Bitcoin Cash (Bitlo, 2018)

Şekil 3.33'te gösterildiği gibi Lightning (Şimşek) ağ çözümü ise her işlemi gerçekleştiği anda blok zincirine eklemeden belirli bir süre offline olarak çalıştıktan sonra yapılan işlemlerin sonucunu Bitcoin zincirine toplu halde bildirmektedir.



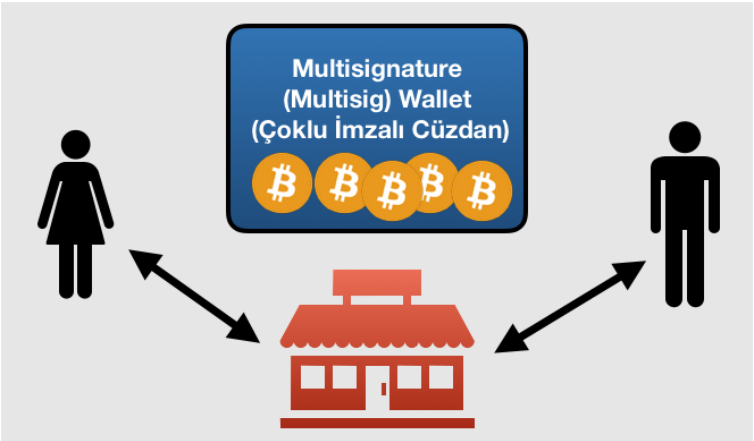
Şekil 3.33. Lightning (Şimşek) ağ çözümü (O3schools, 2021)

Örneğin bir lokantanın müşterileri her gün gelip oradan yemek yemektedir. Şekil 3.34'te gösterildiği gibi yapılan her işlem için Bitcoin zincirini meşgul etmek yerine işlemleri arada tutan Multisignature (Multisig) Wallet (Çoklu İmzalı Cüzdan) birimleri yer almaktadır. Alışveriş yapacak kişi, tutarı bu çoklu cüzdana aktarmaktadır ve hizmetini almaktadır. Yapılan tüm işlemler doğrulandıktan sonra toplu olarak haftanın sadece belirli bir zamanında Bitcoin zincirine yeni bir

blok olarak eklenmektedir. Yeni blok eklendikten sonra çoklu cüzdanda biriken para da mağazanın hesabına yüklenmektedir. Burada taraflar işyeri sahibi ile müşteriler birbirini tanımaktadır.

Şekil 3.34. Çoklu imzalı cüzdan kullanımı (O3schools, 2021)

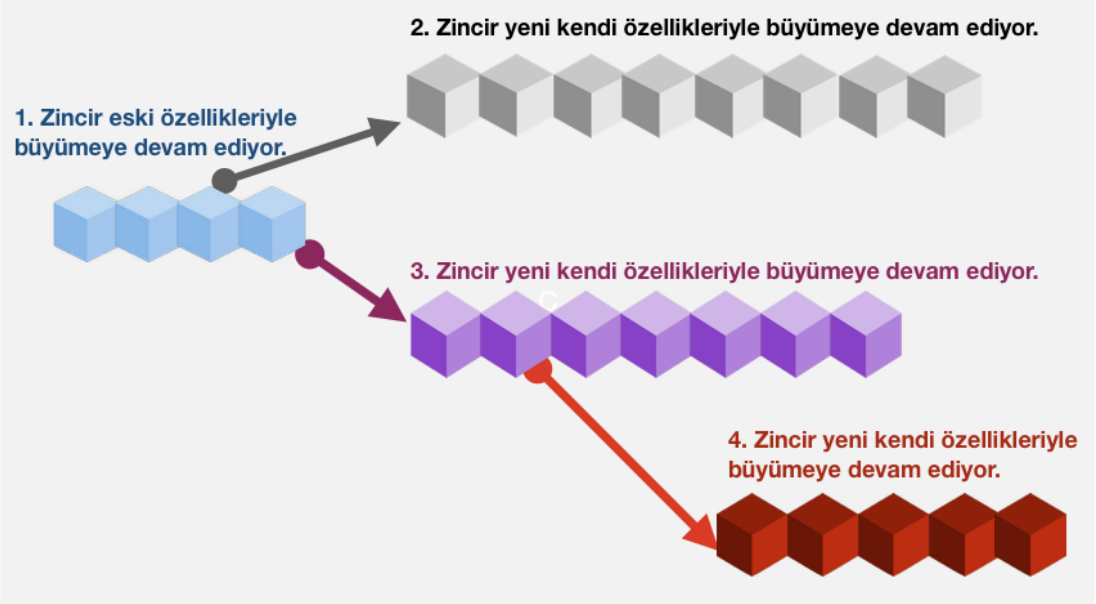
Şekil 3.35'te gösterildiği gibi birbirini hiç tanımayan kişilerin arasında da çoklu imzalı cüzdanı kullanılmaktadır. Blok zinciri aracıları aradan çıkarmayı hedeflemektedir fakat çoklu imzalı cüzdanı, birbirini tanımayan kişiler arasında kullanabilmek için arada çoklu imza hizmeti veren birimlere ihtiyaç duyulmaktadır. Birbirini tanımayan kişiler de para transferlerini ancak onların üzerinden gerçekleştirebilirler. Arada çoklu cüzdan olarak hizmet veren bu yapı da küçük bir komisyon alır. Buna da Lightning ağ çözümü denilmektedir.



Şekil 3.35. Çoklu imzalı cüzdan hizmeti (O3schools, 2021)

Yazılım projeleri güncellendikçe sürümleri olmaktadır. Blok zinciri alt yapısında zincir eğer istenirse ikiye üçe hatta dörde bile ayrılabilmektedir. Bu işleme Fork (Çatal) denilmektedir. Çatal işlemi meydana geldiği anda o zaman yeni bir zincir

ortaya çıkmaktadır. Şekil 3.36'da gösterildiği gibi blok zincir ağına eklenen yeni bir blok geldiğinde düğümler tarafından yeni gelen blok alınıp zincirlere eklenmektedir. Bu sayede zincirler aynı hale getirilmektedir. Düğümlerden bazılarının bulundukları yerde elektriğin kesildiğini varsayalım. O zaman ne olacak? Yeni bloklar gelmeye devam etmektedir. Elektrik kesintisinden dolayı yeni blokları henüz kendine ekleyememiş düğümler elektrik gelince kaldıkları yerden en güncel yere kadar blokları kendilerine otomatik olarak alıp eklemektedir. Zincir çatallanmış ama çatalın kollarından sadece biri takip edilirse o zaman buna Soft Fork (Yumuşak Çatal) denir. Yumuşak çatallanma durumunda geriye uyumlu yazılım güncellemesi yapılmaktadır. Yazılım güncellemesi yapıldıktan sonra kendini henüz güncellememiş olan düğümler de kendini güncellemektedir.

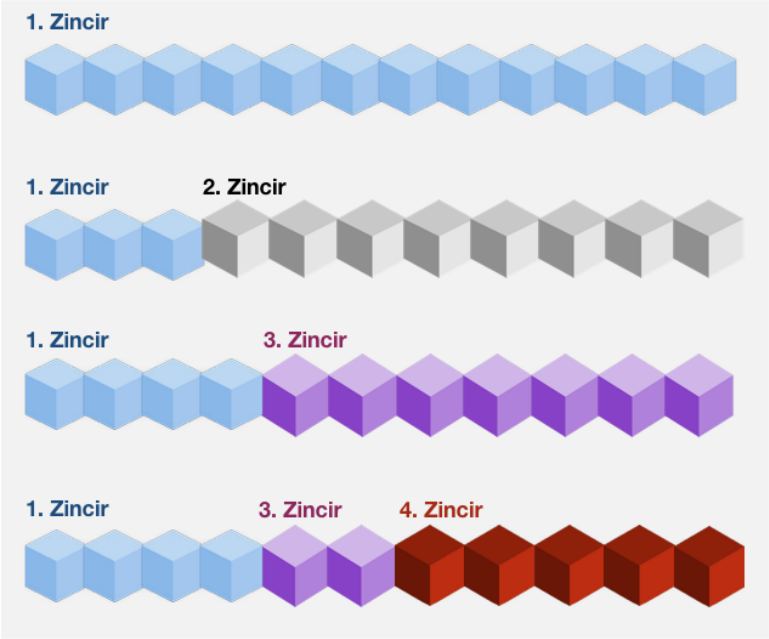


Şekil 3.36. Blok zincirinde yapılan güncellemeler (Usta ve Doğantekin, 2017)

Yumuşak çatallanma zincirdeki geçici bir ayrılmadır ve kalıcı bir durumu olmamaktadır. Yine bir elektrik kesinti senaryosu düşünelim. Zincirin mimarisi yine aynı ama zincire bazı yeni özellikler eklenmiş olsun. Bazen ağa geliştiricileri tarafından yeni özellikler getirilmesi mümkündür. Düğümlerden bazıları kendini güncelleyip yeni özellikler alırken bazıları kendilerini güncellememiş olabilir. Zincirin tam bir noktasından itibaren çatallanma meydana gelirse o zaman ne olacak? Yazılım güncellemesi yaparak yeni özellikleri alan düğümler yeni

özellikler göre blokları eklerken eskide kalanlar düğümler kendi zincirlerine eski özelliklere göre ekleme yapmaya devam etmektedir. Yazılım güncellemesini yapmayanlar yeni yapıya katılamamaktadır ama eski altyapısında çalışmasına devam etmekte ve yeni bloklarını sadece eski sürümde kalan zincire ekleyebilmektedir (Usta ve Doğantekin, 2017). Ağda en çok yeni blok ekleyen hangi grup olursa ya da en uzun zincir takip edilmektedir. Tüm düğümler uzun zincirin özelliklerden eklenmeye devam etmektedir. Yeni özelliklere sahip olan zincir eğer daha uzun olursa blok zincirinde artık diğer düğümler de yeni özellikleri alıp öyle çalışmaya mecbur kalmaktadır. Eski eklenen bloklar ise yetim blok konumuna düşmektedir. Eski özelliklerle çalışmada ısrar eden madencilerin artık ağa yeni blok ekleyememektedir. Kendilerini güncelleyenler ağa blok ekleye bilmektedir. Herkes yeni özelliklere geçince de yumuşak çatallanma zincir birleşimini tamamlayıp güncellenmiş tek bir zincir üzerinden çalışmasına devam etmektedir.

Şekil 3.37'de gösterildiği gibi Hard Fork (Sert Çatal) geriye uyumlu olmayan bir yazılım güncelleme içermektedir. Kendini henüz güncellememiş olan düğümler, kendini güncelleyen blokların yeni zincire eklediği yeni blokları artık görmemektedir çünkü yeni eklenen blokların mimarisi eskiye göre tamamen değiştirilmiştir. Eğer yeni zincirin kendine ait bir mimarisi varsa o zaman sert çatallanma gerçekleşmiştir. Her zincir kendi yolunda çalışmaya devam etmektedir. Sert çatallanma sonrasında ortaya 2 yeni kripto para çıkmaktadır. Belirli bir yere kadar aynı ama sonrasında eklenen yeni bloklar tamimiyle birbirinden mimari olarak farklı olmaktadır. İki farklı zincir ortaya çıktığından çok maliyetli bir güncellemedir.

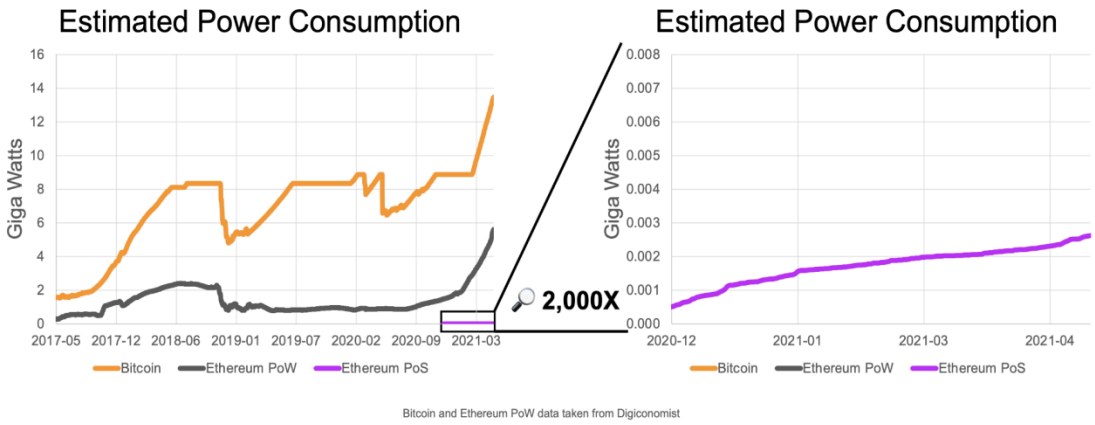


Şekil 3.37. Zincirin yeni zincirlere ayrılması (Usta ve Doğantekin, 2017)

Blok zincirinde yapılan bir işlemi doğrulamak ve aynı işlemin tekrarının önüne geçmek için kontrolü sağlayan bir fikir birliği mekanizmasına ihtiyaç duyulmaktadır. Blok zinciri teknolojisinde birden fazla doğrulama mekanizması mevcuttur. Fikir birliği mekanizmaları literatürde mutabakat, doğrulama ve konsensüs terimleri ile ifade edilmektedir (Jiang vd., 2019). Kullanılan bir mutabakat algoritması sayesinde yapılan bir işlem, kripto para kayıt defterine güvenli olarak kaydedilmektedir. Proof of Work (PoW) mutabakat algoritması Bitcoin ile birlikte öne çıkıp tanınmaktadır. PoW mekanizması ağdaki işlemlerin doğrulanıp yeni blokların eklenmesini sağlamak için çok güçlü donanımlara sahip madenci düğümlere ihtiyaç duymaktadır. Farklı doğrulama mekanizmalarını kullanan çeşitli blok zincir tabanlı geliştirme platformları da bulunmaktadır. Blok zincirinin altyapısında en yaygın olarak Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS) ve Direct Acyclic Graph (DAG) mekanizmaları kullanılmaktadır. Fikir birliği mekanizması, ağdaki işlemlerinin güvenilirliğini ve tutarlılığını garanti altına almaktadır. İlk ve en meşhur blok zinciri projesi olan Bitcoin, konsensüs mekanizması olarak PoW'u kullanmaktadır (Jiang vd., 2019). PoW, Bitcoin'in öncülük ettiği blok zincirindeki en klasik fikir birliği mekanizmasıdır (Cao vd., 2020). Diğer bir kripto para olan Ethereum, konsensüs mekanizması olarak PoW'u kullanır ama en kısa zamanda

PoS mimarisine geçiş yapmayı hedeflemektedir (Košťál vd., 2018). PoS mekanizmasında bir madenci düğümün çok üst düzey donanımlara sahip olması bir önem arz etmemektedir. Madenci düğüm, doğrulama yapmak istediği blok zincir ağının kripto parasına ne kadar çok yatırım yapıp onu cüzdanında bolca bulundurursa yeni bir blok ekleme şansını da o kadar çok arttırmaktadır. Bitcoin ve Ethereum kripto paralarının yaygınlaşmaları ile birlikte blok zinciri ağları üzerinde yapılan işlemlerin sayısı da katlanarak artmaktadır. Bu işlemlerin doğrulanma aşamasındaki bekleme sürelerinin azaltılması için mimarilerde iyileştirilme ihtiyacı doğmaktadır. Bu ihtiyacı karşılamak için DAG tabanlı blok zincir, IOTA projesi geliştirilmiştir. Tangle, DAG altyapısını kullanan ve IoT sistemlerindeki işlemlerin kısa zamanda doğrulanmasını sağlayan bir blok zinciri platformudur (Sagirlar vd., 2018).

PoW mekanizması, çevre dostu değildir ve çalışırken çok fazla enerji tüketmektedir (Košťál vd., 2018). PoS, daha küçük bir karbon ayak izine sahiptir. PoS mekanizmasında yüksek güçte madencilik çiftliklerine ihtiyaç duyulmamaktadır ve harcanan elektrik miktarı, PoW mekanizmasına kıyasla çok daha düşük olmaktadır. Ethereum PoW ile çalışmaktadır ama PoS altyapısına geçmeyi hedeflemektedir (Supreet vd., 2020). Bitcoin PoW, Ethereum PoW ve Ethereum PoS mekanizmalarının güç tüketimi kıyaslaması Şekil 3.38'de verilmiştir. PoS mekanizması, madenci düğümlerin nonce sayısını tahmin ederken ortaya koydukları rekabetin süresini azaltmak ve daha az kaynak tüketimini sağlanmak için o ağa ait kripto para varlığına sahip olma miktarını öne çıkarmaktadır. Ağdaki madencilerin cüzdan hesabında o ağa ait kripto para varlığından kimlerde daha çok varsa yeni bloğu eklemesi için o düğümlere öncelik verilmektedir. Seçilen madenciler, kendi aralarında yarışırlar ve nonce sayısını bulan ilk madenci, yeni bloğu ekleme hakkını elde ederek kripto para ödülünün sahibi olmaktadır. DPoS yapı olarak PoS konsensüs mekanizmasına benzemektedir ama yeni bir bloğu eklemek isteyen madenci düğümler o ağdaki kişiler tarafından özgürce seçilmektedir. DPoS konsensüs mekanizmasında her düğüm bir adaydır. Her düğüm oylama yoluyla birkaç aracı düğümü seçmektedir. Aracı bir düğüm bloğu oluşturup belirlenen programa göre doğrulama için sıraya almaktadır (Xu vd., 2019).

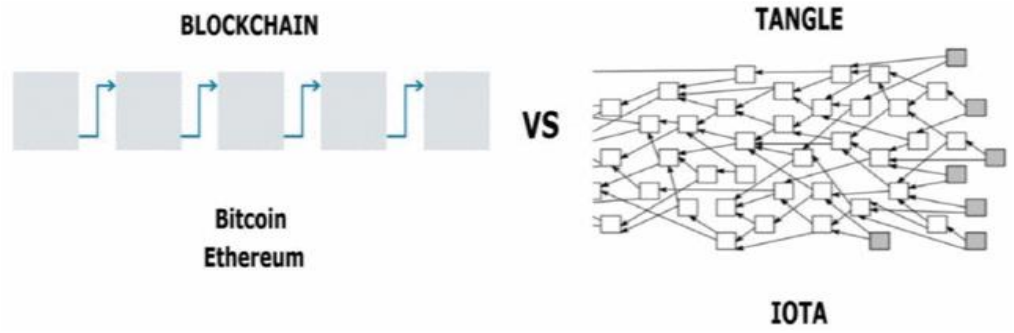


Şekil 3.38. PoW ve PoS enerji tüketimi (Ethereum, 2021)

PoS kullanan ağlarda sadece zengin madenciler ödülleri kazanmaktadır. Bu yaklaşım adil değildir. Bu sorunu gidermek için DPoS mekanizması yeni bir çözüm sunmaktadır. DPoS mutabakat algoritması, PoS mekanizmasına oldukça benzemektedir ama ağdaki kişiler, yeni bloğu eklemek isteyen aday düğümleri bizzat kendileri seçebilmektedir. Hisse sahipleri, sistemi kendileri adına denetleyecek birkaç delege ve şahit için oy kullanma hakkına sahiptirler. Delegeler ve şahitler, işlemlerin doğrulanması ve yeni blokların oluşturulmasında vazife almaktadır (Chen vd., 2021). Yeni bloğu ekleyen düğüm tarafından kazanılan ödül ise kendisine oy verip onu seçen kişilerle pay edilmektedir. PoW, PoS, DPoS mekanizmalarını kullanan blok zincir ağlarında yeni bir blok oluşturmak için madenciler yarışıp rekabet etmektedir. DAG mekanizmasında böyle bir yarış süreci yoktur ve tüm işlemler birbiriyle bağlantılı olarak sürdürülmektedir.

Tangle (IOTA), DAG tabanlı çalışan ilk blok zinciridir. IOTA, mikro ödeme desteği ve işlem ücreti olmaması gibi temel özellikleri ile nesnelerin interneti (IoT) uygulamalarını desteklemek için umut verici bir platform olarak kabul edilmektedir (Guo vd., 2020). DAG, IoT ve mikro işlemler için önerilmektedir. IOTA ağının büyüme ve blokların bağlantı kurma türü Şekil 3.39'da gösterilmektedir. DAG mekanizması çalışma esnasında ağdan rastgele 2 tane onaylanmamış işlemi seçip doğruladıktan sonra ağdaki tepe noktasını güncellenmektedir. Son adımda ise işlemlerin toplam ağırlıkları

hesaplanmaktadır (Wang vd., 2020). Blok zinciri altyapılarında PoW, PoS, DPoS ve DAG fikir birliği algoritmaları çok yaygın olarak kullanılmaktadır. Bu konsensüs mekanizmalarının da kısıtları mevcuttur (Zhao vd., 2019).



Şekil 3.39. Tangle (IOTA) blok zinciri örneği (Bhandary vd., 2020)

DAG mekanizması çalışma esnasında ağdan rastgele 2 tane onaylanmamış işlemi seçip doğruladıktan sonra ağdaki tepe noktasını güncellenmektedir. Son adımda ise işlemlerin toplam ağırlıkları hesaplanmaktadır (Wang vd., 2020). Blok zinciri altyapılarında PoW, PoS, DPoS ve DAG fikir birliği algoritmaları çok yaygın olarak kullanılmaktadır. Bu konsensüs mekanizmalarının da kısıtları mevcuttur (Zhao vd., 2019).

3.5. Smart Contracts (Akıllı Sözleşmeler) Geliştirme Platformları

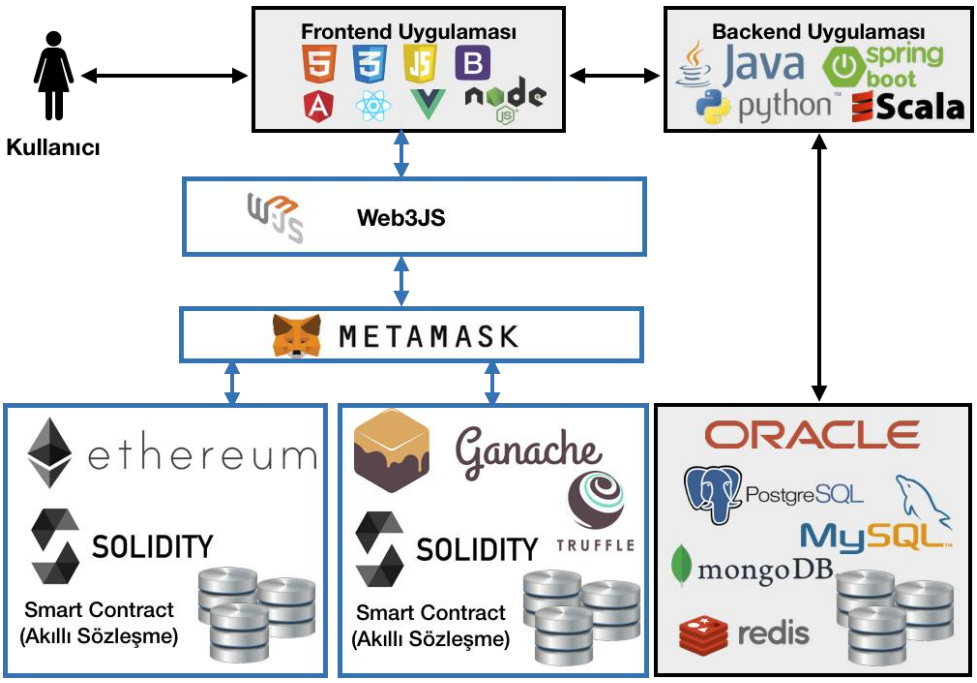
Blok zinciri tabanlı çalışan uygulamaları geliştirmek için birçok yardımcı platform vardır. Bu platformlarda akıllı sözleşmeler kullanılmaktadır. En çok öne çıkan platformlar alt başlıklarda açıklanmaktadır.

3.5.1. Ethereum

Bitcoin, akıllı sözleşmeleri olmayan 1. nesil bir blok zinciri projesidir ve sadece kripto para olarak kullanılmaktadır. Blok zincirinin 2. nesil projesi olan Ethereum bu kalıbı kırarak blok zincirinde akıllı sözleşmelerin kullanımı fırsatını sunmaktadır (Vujičić vd., 2018). Akıllı sözleşmelerin amacı, taraflar arasındaki işlemleri şifreli algoritmalarla şeffaf ve güvenli olarak otomatikleştirmektir (Tonev, 2020). Tarafların arasında hiç kimse ve hiçbir kurum aracı olarak

girmemektedir. Akıllı sözleşmeler, yazılmış talimatları sırasıyla yerine getirerek çalıştırmaktadır. Yapılan tüm işlemler blok zincir ağına şeffaf olarak kaydedilmektedir. Akıllı sözleşmeler sayesinde herkes merkezi olmayan uygulamalarını Decentralized Applications (DApps) oluşturup kolayca kullanabilmektedir (Mishra vd., 2020). DApps uygulamalarının kodu, onu kullanan herkese açık olarak paylaşılmaktadır. Uygulama işlem yaparken üzerinde çalıştığı ağa ait kripto parayı kullanmaktadır. Remix IDE üzerinde akıllı sözleşmeler Solidity programlama dili ile geliştirildikten sonra Ethereum Virtual Machine (EVM) ağına eklenip çalıştırılmaktadır (Aung ve Tantidham, 2019). Ethereum platformu, Ether (ETH) adında kendine ait kripto parasına sahiptir. Akıllı sözleşmeler her çalıştırıldığında bir işlem ücreti ödenmektedir.

Solidity ile geliştirilen akıllı sözleşme kodunun geçerliliği Ganache, Truffle Suite ve Metamask yardımcı araçlarının üzerinden bilgisayarda test edilmektedir. Şekil 3.40'da gösterildiği gibi akıllı sözleşme kodu Web3JS ile iletişime geçmektedir. Oluşturulup ağa eklenen bir akıllı sözleşme kodu daha sonradan kesinlikle değiştirilememektedir. Güncelleme için yeni bir sürümünün yazılıp ağa tekrardan dahil edilmesi gerekmemektedir.



Şekil 3.40. Akıllı sözleşme ve yazılım proje ilişkisi (Moralis, 2022)

Akıllı sözleşme, bütün programlama dilleri ile çalışabilmektedir. Sözleşmedeki her kod ifadesi bir ücrete karşılık gelmektedir. Ağda çalıştırılan akıllı sözleşmenin her çalıştırılmasında bir ücret ödenmektedir. Oluşturulan sözleşme kodlarının olabildiğince sade, masrafsız ve kısa tutulması gerekmektedir. Ethereum ağında çalışması için akıllı sözleşme geliştirmek isteyen kişilerin Solidity programlama diline hâkim olmaları gerekmektedir. Blok zinciri üzerinde çalışan akıllı sözleşme ile 2 örnek uygulama geliştirildi. İlk projede terminal üzerinden JavaScript ile girilen parametreler alınmakta ve blok zincirine eklenmektedir. Ağa kaydedilmiş bilgiler daha sonradan blok zincirinde aratılmakta ve bulup gösterilmektedir. İkinci uygulamada web sayfasından kripto para transferi gerçekleştirilmektedir. Uygulama DApps Web3.0 projesi olarak geliştirilmiştir.

3.5.2. Cosmos

Bitcoin ve Ethereum gibi farklı blok zincir ağları birbiri ile doğrudan iletişim kuramamaktadır. Blok zincir ağları yaygınlaştıkça paralel olarak birlikte çalışabilecek ağlara da ihtiyaç duyulmuştur. Farklı blok zincirlerinin birlikte çalışmasını sağlamak için Cosmos platformu doğmuştur (Dong vd., 2020). Cosmos, altyapı mimarileri bambaşka olan ağlarının hepsini birbiriyle konuşturmayı hedeflemektir. Cosmos sayesinde farklı blok zincir platformları arasında köprüler kurulup daha zengin içeriklere sahip blok zincir uygulamaları geliştirilmesi hedeflenmektedir. Cosmos platformu, işlemleri gerçekleştirirken ATOM adında kendi ait kripto parasını kullanmaktadır.

3.5.3. Cardano

Cardano, 3. nesil bir blok zinciridir. Ölçeklenebilirlik ve birlikte çalışabilirlik getirmeye odaklanmaktadır. Cardano ile akıllı sözleşmeler geliştirilmektedir (Worley ve Skjellum, 2018). Akıllı sözleşmeleri kodlarken Plutus yazılım dilini kullanılmaktadır. Düğümlerin ağ hakkında fikir birliğine ulaşması için altyapısında Ouroboros adında yeni bir algoritma mekanizması kullanmaktadır.

Cardano platformu çalışırken ADA adında kendine ait kripto parasını kullanmaktadır.

3.5.4. EOS

EOS platformu merkezi olmayan bir işletim sistemidir. Blok zinciri üzerinde çalışan merkezi olmayan uygulamaları çalıştırıp desteklemektedir. Saniyede milyonlarca işlem yapma yeteneğine sahip bir platform olmayı ve blok zincirindeki işlem ücretlerini de tamamen ortadan kaldırmayı hedeflemektedir. EOS için akıllı sözleşmeler, C++ yazılım dili ile geliştirilmektedir. EOS, çok hızlı okuma ve yazma işlemleri gerçekleştirmek için depolama alanı olarak disk yerine RAM kullanmaktadır (Dernayka ve Chehab, 2021). DPoS konsensüs mekanizmasını ile çalışmaktadır (Mishra vd., 2020). DPoS ağdaki kullanıcıların kendi temsilcilerini seçmesine izin vermektedir. Seçilen bir temsilci, blok zincir ağına yeni bir blok ekleme hakkına sahip olmaktadır. Ekleme işleminden sonra kazanılan ödül ise kendisini seçen kişilerin hisselerine orantılı olarak paylaştırılmaktadır. EOS platformunun kendine ait kripto parası platformla aynı adı taşıyan EOS'tur.

3.5.5. Hyperledger

Ethereum, Cardano, Cosmos ve EOS platformları kendi kripto para birimlerine ve kendi blok zincir ağlarına sahiptirler. Hyperledger platformunun ise kendine ait bir kripto para birimi yoktur. Hyperledger Linux vakfı tarafından geliştirilmiş açık kaynaklı bir projedir (Benhamouda vd., 2019). Hyperledger, kurumların kendilerine ait hızlı, ölçeklenebilir ve yüksek performanslı blok zinciri ağlarını oluşturmalarını amaçlamaktadır. Hyperledger platformunda akıllı sözleşmeler Chaincode yazılım dili ile geliştirilmektedir (Nguyen vd., 2019).

Bu platformlar blok zinciri projelerinde yaygın kullanılmaktadır. Projedeki ihtiyaçlara göre en uygun platformun kullanılması tercih edilmelidir. Blok zincir uygulama geliştirme platformları Çizelge3.3'te kıyaslamalı olarak verilmektedir.

Çizelge 3.3. Blok zincir uygulama geliştirme platformları

	Ethereum	**Cardano**	**EOS**	**Cosmos**	**Hyper-ledger**
Kripto para birimi	ETH	ADA	EOS	ATOM	-
Sahibi (Firma, vakıf)	Ethereum Vakfı	Cardano vakfı, IOHK, Emurgo	Block One	Interchain Vakfı	Linux Vakfı
Amacı	Dünyanın merkezi-yetsiz blok zincir tabanlı süper bilgisayarı olmak.	Bilimsel olarak destekle-nen akıllı sözleşme platformu olmak.	Endüstriyel ölçekli DApps uygulama-ları için bir platform olmak.	Birlikte çalışabilir-liliği sağlayıp blok zincirleri arasında internet olmak.	İşletme-lerin kendi yerel özel blok zincir ağlarını oluştur-malarını sağlamak.
Konsensüs (mutabakat, doğrulama, fikir birliği) mekaniz-ması	PoW ve PoS	Ouro-boros	DPoS	Tender-mint	Byzantine fault tolerance (Bizans hata toleransı)
Akıllı sözleşme kodlama dili	Solidity	Plutus	C++	Java, JavaScript, Swift	Chaincode
Bir bloğun ağa eklenme süresi	14 saniye	20 saniye	3 saniye	7 saniye	1 saniye

Bu çalışmada blok zincirlerine eklenen her bir blok ile güvenliğin nasıl sağlandığı, arada değişmesinin olasılığı, yani bir saldırı yapılması olasılığı hakkında analitik olarak bir çalışma yapılmıştır. Eklenen her bir bloğun dolaylı olarak güvenliği nasıl etkilediği araştırılmıştır.

3.6. Blok Zinciri Ağ Güvenliği

Bu çalışma kapsamında blok zincir ağının güvenlik analizleri açıklanacaktır. Bu amaçla blok zinciri mimarileri, konsensüs mekanizmaları ve uygulama geliştirme

platformları açıklanmaktadır. Bir saldırganın blok zincir yapısını değiştirmek için yerine getirmesi gerekenler ve ortaya koyacağı efor, teorik modeller ile gösterilerek saldırının gerçekleştirilmesinin zorluk derecesi hesaplanmıştır.

Blok zincirinde ağdaki herkesin kabul ettiği güvenilir zincir, dürüst zincirdir (Nakamoto, 2008). Ağdaki tüm eş düğümler bu doğrulanmış zincirin bir kopyasını kendisine almaktadır. Bu çalışmada blok zincirini değiştirip onu ele geçirmek isteyen kötü niyetli kişinin ağa saldırdığı bir senaryo incelenmektedir. Saldırgan kişi, ağda daha önce hiç yapılmamış bir işlemi yapılmış gibi bir bloğun içine kendi verilerini yerleştirip ağa eklemeye çalışsa bile bunu başarması çok zordur çünkü ağdaki doğrulama mekanizması, blok zinciri üzerinde sonradan yapılan bir değişimin tüm ağ tarafından yeniden onay almasını istemektedir.

Blok zinciri dağıtık olarak tutulduğundan yapılan bir değişiklik sadece saldırganın kendi makinesinde yer alan zincirin üzerinde geçerli olacaktır. Ağdaki diğer düğümlerde çalışan konsensüs mekanizması tarafından denetleme yapılırken bu değişiklik anında fark edilecektir ve o düğüm ağın tamamında dürüst olmayan bir düğüm olarak ilan edilecektir. Blok zincirine saldıran bir kişi daha önce zincire eklenmiş olan bir bloğun içinde yer alan bir işlemi değiştirirse değişiklik yapılan o bloktan itibaren ne kadar blok varsa içindeki işlemlerle birlikte hepsinin yeniden tek tek doğrulanması gerekmektedir. Saldırgan kişi, daha bu doğrulamalarını bitirmeden diğer düğümler onun zincirinde yapılan değişikliği çok kısa zamanda denetleyip fark edecektir ve saldırganın tüm çabası da boşa gidecektir.

Blok zincirine saldırmak isteyenler için en iyi fırsat, zincire yeni bir blok ekleneceği zamandır. Ağın en sonuna yeni bir blok eklenirken dürüst zincir ile saldırgan kişinin makinesindeki zincir birebir aynı uzunluktadır. Madenciler arasında yeni bir bloğun eklenilmesi için bir yarışı yapılmaktadır. Dürüst zincir ile saldırgan arasındaki yarışı Binomial Random Walk (Binom Rastgele Yürüyüş) şeklinde tariflemek mümkündür Başarılı olma durumu dürüst zincirin bir blok uzamasıdır ve +1 puan öne geçirir. Başarısızlık durumu da saldırganın zincirini bir blok uzatmasıdır ve aradaki farkı -1 puan kadar azaltır. (Nakamoto, 2008).

Saldırgan düğüm eğer başarılı olup da kendi zincirine yeni bir blok ekleyebilirse o zaman ağdaki en uzun zincire sahip olmaktadır ama saldırganın ağın tamamını ele geçirmesi için bu çabası da yeterli olmamaktadır. Blok zinciri dağıtık halde tutulduğundan ağın %51'ne de ürettiği bloğunu eklettirmesi gerekmektedir. Ağın %51'ine hâkim olunmadan ağdaki tüm zincirlerin değiştirilmesi imkansızdır. Madenciler arasında blok ekleme yarışını kaybedenin eklemeye çalıştığı bloğun içindeki tüm işlemler madenci havuzuna geri iade edilmektedir. Eklenmeyen bloğa yetim blok denilmektedir.

Blok zincir ağına saldıran kötü niyetli birinin bir gecikme farkını yakalayıp kapatabilme olasılığı Gambler's Ruin (Kumarbazın İflası) problemine benzemektedir (Nakamoto, 2008). Kumarbazın iflası problemi, bir Markov zinciridir ve geçiş olasılıklarının bilindiği varsayıldığında kaybetme ve kazanma olasılıkları tam olarak hesaplanabilmektedir (Akbarzadeh ve Tekin, 2016). Kumarbaz, oyuna borç ile başlamaktadır ve sonsuz bir krediye sahiptir. Rakibiyle başa baş bir seviyeye gelebilmek için istediği kadar deneme yapabildiği bir oyunu oynamaktadır (Feller, 1957). Rakibini yakalayıp onu geçme olasılığı, aynen blok ağına saldıran birinin bir dürüst zincirin uzunluğunu yakaladıktan sonra yeni bir bloğu ekleyerek onu geçme olasılığı gibidir. Kumarbazın beraberliğe ulaşma olasılığını ya da saldırganın dürüst zinciri yakalama olasılığını şöyle hesaplayabiliriz (Nakamoto, 2008).

p = Dürüst bir düğümün bir sonraki bloğu bulma olasılığı,
q = Kötü niyetli bir düğümün bir sonraki bloğu bulma olasılığı,
q_z = Kötü niyetli bir düğümün z blok geriden gelerek dürüst düğümlere yetişip beraberliği yakalama olasılığı,
z = Zincire eklenen blok sayısı,

$$q_z = \{1,\ p \leq q\} \tag{3.1}$$

Formül (3.1)'de kötü niyetli bir düğümün blok bulma yarışını kazanma durumu ele alınmaktadır. Saldırgan düğümün kazanma olasılığı, dürüst düğümlerin bir

sonraki bloğunu bulma olasılığına eşit veya daha yüksekse bloğu manipüle edebileceği anlamına gelmektedir. Saldırgan düğümün yeni bloğu bulma olasılığı daha büyüktür ve ağı yakalayıp yeni blok ekleme şansı vardır.

$$q_z = \{ (p/q)^z , \ p > q \} \qquad (3.2)$$

Formül (3.2)'de dürüst bir düğümün blok bulma yarışını kazanma durumu ele alınmaktadır. Dürüst düğümler tarafından z tane ne kadar çok blok üretilirse kötü niyetli düğümün kazanma olasılığı o kadar azalmaktadır. Saldırganın yakalaması gereken blok sayısı arttıkça bir sonraki bloğu bulma olasılığı da üssel olarak azalmaktadır. Saldırgan kişi eğer şanslıysa yeni bir bloğu bulup eklemektedir ve ağı ele geçirme şansını arttırmaktadır. Aksi halde saldırgan düğüm zincirini uzatamamaktadır ve dürüst düğümlere karşı yarışı kaybederek başarısız olmaktadır.

Yeni bir işlem yapıldığında alıcı taraftaki kişinin veriler üzerinde bir değiştirme yapamayacağından emin olmak için ne kadar zaman beklemesi gerektiği incelenmelidir. Saldırgan kişi bir ödeme işlemi gerçekleştirdikten sonra yaptığı ödemeyi yine kendisine geri döndürmek isterse alıcı taraftaki kişi bunu konsensüs mekanizmaları sayesinde fark etmektedir. Saldırgan kişi, bunun daima çok geç fark edilmesini ümit etmektedir. Alıcı taraf, yeni bir anahtar çifti üretmekte ve bu açık anahtarını imzalamadan önce gönderen tarafa iletmektedir. Karşı tarafa göndereceği anahtarı, kendi özel anahtarı ile imzalamakta ve bu mesaj ancak açık anahtar ile okunmaktadır. Anahtarlar sayesinde gönderme yapan tarafın yaptığı işlemler onay almadan yeni bir bloğun içine konulmamaktadır. Bu sayede doğrulmamış bir işlemin ağa eklenilmesi de engellenmektedir. Saldırgan, bir işlemi karşı tarafa gönderdikten hemen sonra dürüst zincirlere alternatif başka bir zincir oluşturmaya çalışmaktadır. Alıcı taraf ise yeni bir bloğun eklenmesini ve ondan sonra ise z tane bloğun daha ağa dahil edilmesini beklemektedir. Dürüst bir düğümün yeni bir bloğu ağa ekleyebilmesi için diğer madencilerden de doğrulama onaylarının alınması için biraz bekletilmesi gerekmektedir. Kötü niyetli kişinin ağda başarılı olarak ilerleme elde etme potansiyeli Poisson dağılımı ile gösterilmektedir. Poisson dağılımı, belli bir

sabit zaman aralığında ayrık olarak ortaya çıkan olayları incelemektedir. Sabit zaman aralığında gözlenen olayların sayısı random variable (rassal değişken) olmakta ve olasılık dağılımı saptama imkânını sağlamaktadır (Wikipedia, 2002)

λ = Verilen bir sabit aralıkta ortaya çıkan olayların sayısının beklenen değeri,

$$\lambda = z\frac{q}{p} \qquad (3.3)$$

Formül (3.3)'te verilen sabit zaman aralığında, olayların gerçekleşme sayısının beklenen değeri hesaplanmaktadır.

e = Euler sayısı, doğal logaritmanın tabanı (e = 2.71828...),
k = Olasılığı fonksiyon ile verilmekte olan olayın ortaya çıkma sayısı,
k! = k için faktöriyel,
∑ = Blok zincir ağındaki sonsuz dağılım kuyruğunun toplam değeri,

$$\sum_{k=0}^{\infty} \frac{\lambda^k \ e^{-\lambda}}{k!} \{ (q/p)^{(z-k)}, \quad k \leq z \} \qquad (3.4)$$

Formül (3.4)'te olasılığı verilen olayın ortaya çıkma sayısı, zincire eklenen blok sayısından daha küçük ya da küçük eşittir. Dürüst düğüm ağda söz sahibi olmaktadır. Saldırgan z tane blok geriden gelerek zincire yetişip zinciri belirli bir yerden yakalayabilme olasılığı ve oradan ilerleme durumu için Poisson dağılımından yararlanılmaktadır (Nakamoto, 2008).

$$\sum_{k=0}^{\infty} \frac{\lambda^k \ e^{-\lambda}}{k!} \{ 1, \quad k > z \} \qquad (3.5)$$

Formül (3.5)'te olasılığı verilen olayın ortaya çıkma sayısı, zincire eklenen blok sayısından daha büyük olduğu durumda saldırgan düğüm ağı geriden gelip yakalayabilmektedir.

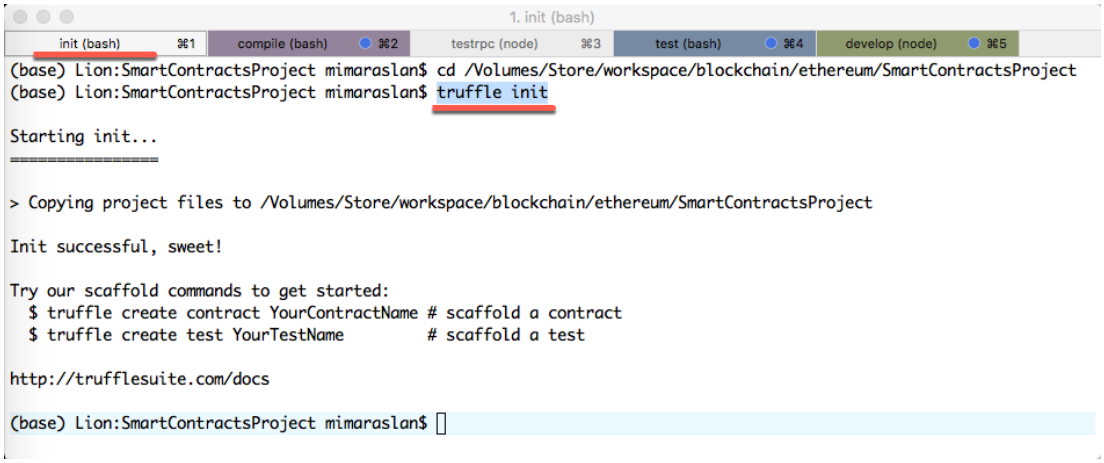
$$1 - \sum_{k=0}^{z} \frac{\lambda^k \ e^{-\lambda}}{k!} \{ 1 - (q/p)^{(z-k)} \} \qquad (3.6)$$

Formül (3.6)'da sonsuz dağılım kuyruğunun toplamını almaktan kaçınmak için formül düzenlenip en son haline getirilmektedir.

4. SMART CONTRACTS (AKILLI SÖZLEŞMELER) UYGULAMALARI

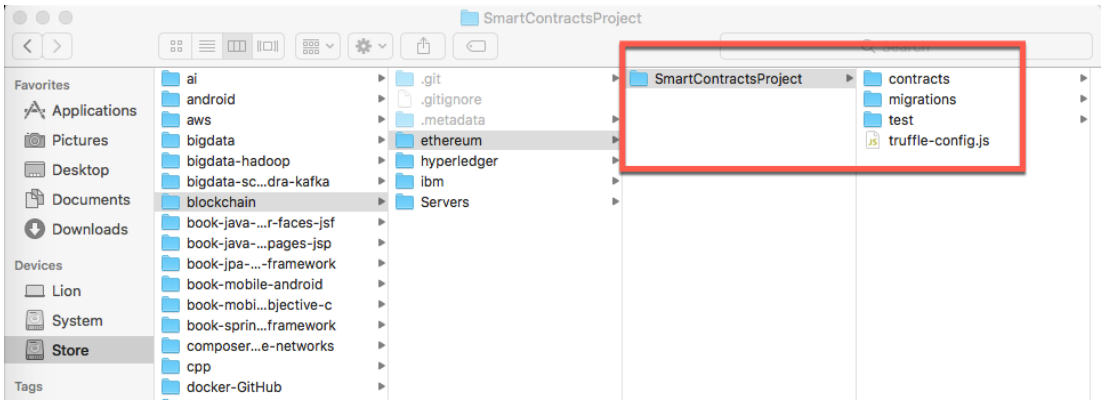
Akıllı sözleşme taraflar arasında önceden belirlenen kurallara göre yazılımcılar tarafından geliştirilmektedir. Ethereum blok zinciri üzerinde çalışan akıllı sözleşmeler Solidity programla dili ile kodlanmaktadır. Geliştirme ortamı için kurulması gereken programlar Metamask, NodeJS, Npm, Truffle, Ganache, Visual Studio Code. Web tarayıcısı üzerinden online kod geliştirme için Remix Ethereum IDE kullanılmaktadır. Bu kurulumların tamamı ücretsiz açık kaynaktır. İşletim sistemi Linux, Windows, Mac OS gibi bir kısıtlama bulunmamaktır. Geliştirme ortamı olarak VSCode kullanacakların Solidity eklentisini eklemeleri kodları yazarken işlerini kolaylaştırmaktadır.

İşletim sistemindeki terminal uygulamasını açıp yeni bir proje oluşturulması ile akıllı sözleşme geliştirmeye başlanmaktadır. Şekil 4.1'de Truffle aracına ait komutlarla örnek bir proje oluşturulmaktadır (Truffle, 2021).



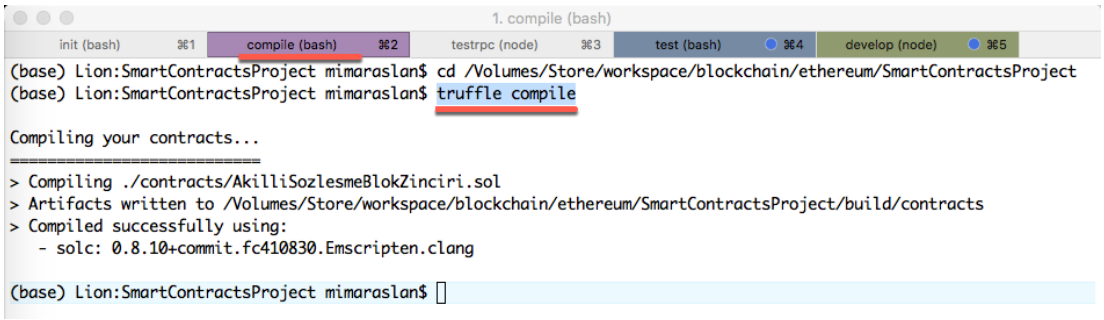
Şekil 4.1. Akıllı sözleşme proje oluşturulması (Truffle, 2021)

Belirtilen adreste Şekil 4.2'de örnek proje başarılı olarak oluşturuldu. Bu projede akıllı sözleşmeler, contracts klasörü içerisinde yer almaktadır.



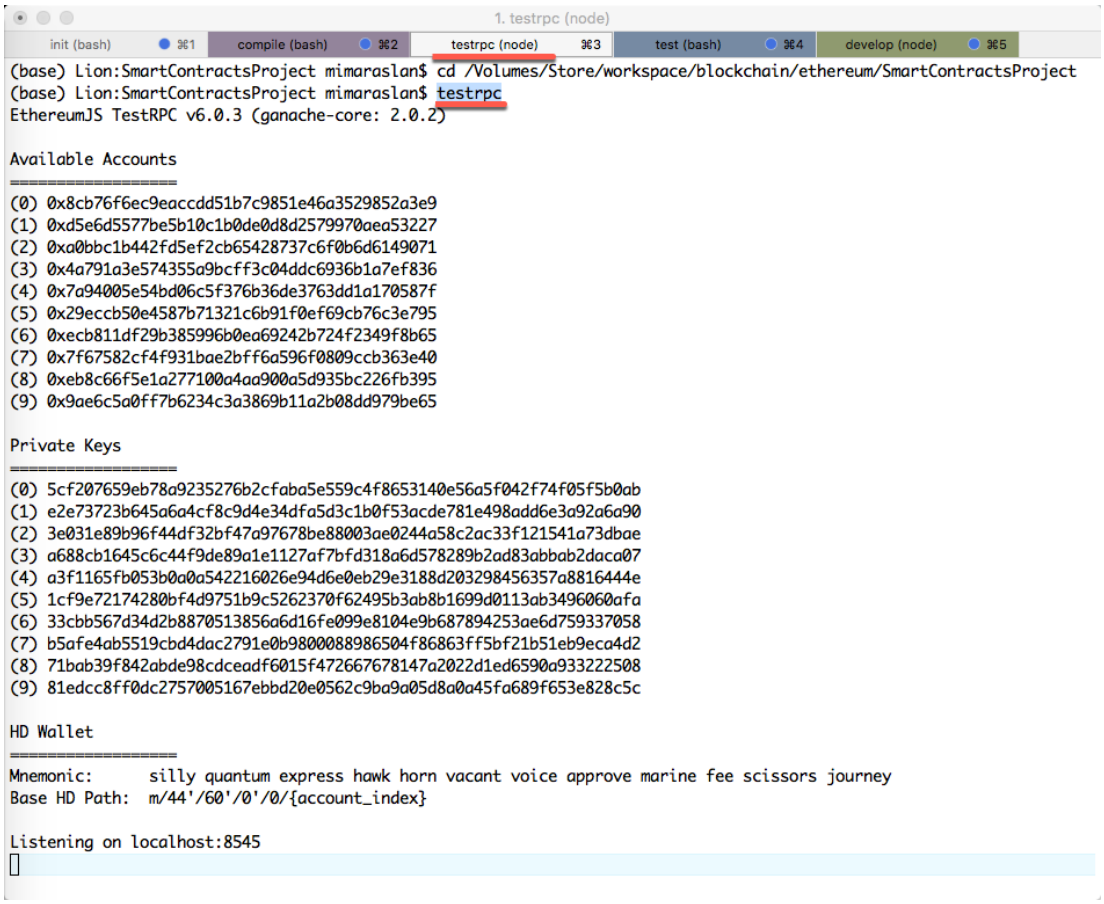
Şekil 4.2. Akıllı sözleşme projesi

Şekil 4.3'te terminal üzerinden projenin bulunduğu adres yoluna geçilmektedir. Truffle aracının komutlarından compile (derleme) ile proje test için hazır hale getirilmektedir.



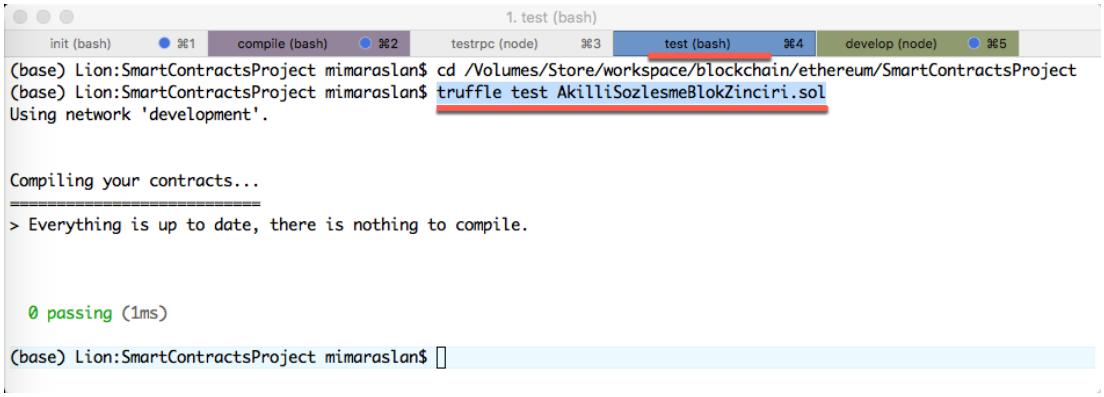
Şekil 4.3. Akıllı sözleşme projesi derlenmekte

Şekil 4.4'te gösterildiği gibi projenin test edilmesi için testrpc komutu kullanılmaktadır. Bu komut sayesinde 10 tane test hesabı oluşturulmaktadır. Test hesaplarına ait cüzdan ve özel anahtar bilgileri kullanıma hazır halde listelenmektedir. Her çalıştırmadan yeni cüzdan ve özel anahtar bilgileri üretilmektedir ama bunun için Truffle kalıcı bir çözüm sunmaktadır.



Şekil 4.4. Akıllı sözleşme için cüzdanlar ve anahtarlar oluşturulmakta

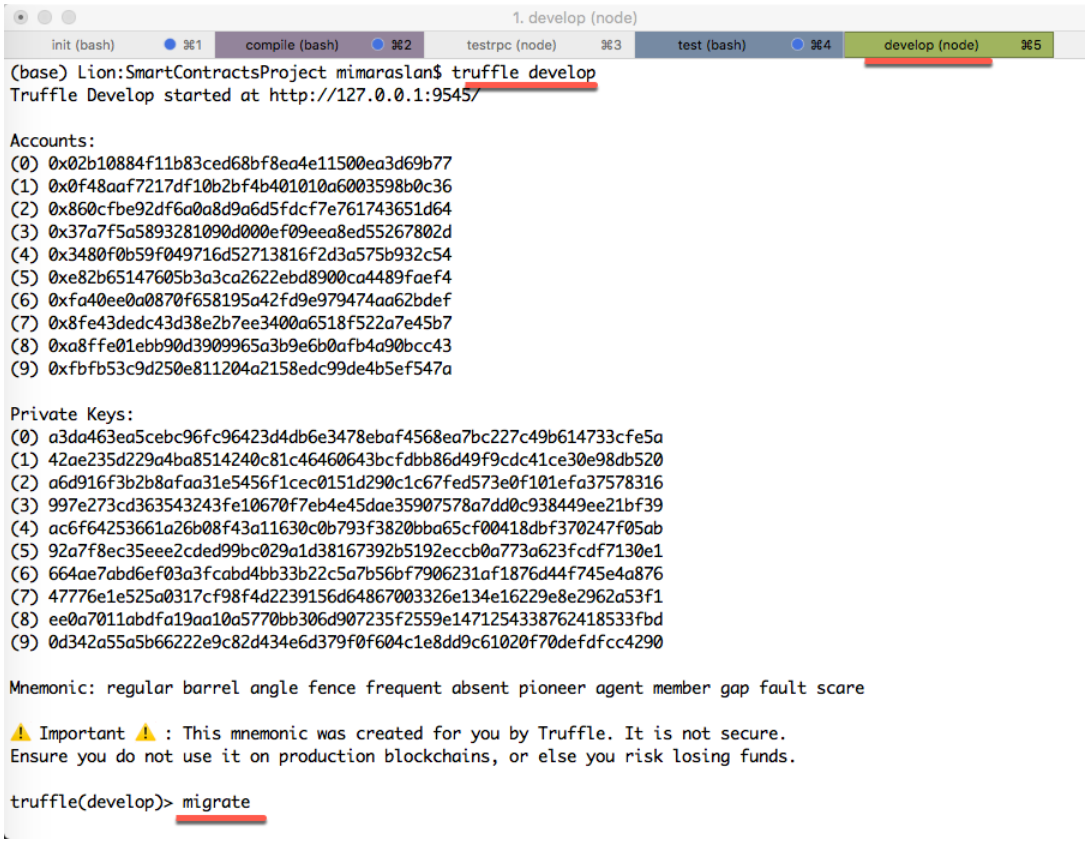
Şekil 4.5'te test edilmek istenen akıllı sözleşme test için Truffle ortamına komut satırından gönderilmektedir. Eğer her şey başarılı olursa terminalde başarılı olarak geçti mesajı gösterilmektedir.



Şekil 4.5. Akıllı sözleşme kodu geçerlilik onayı almakta

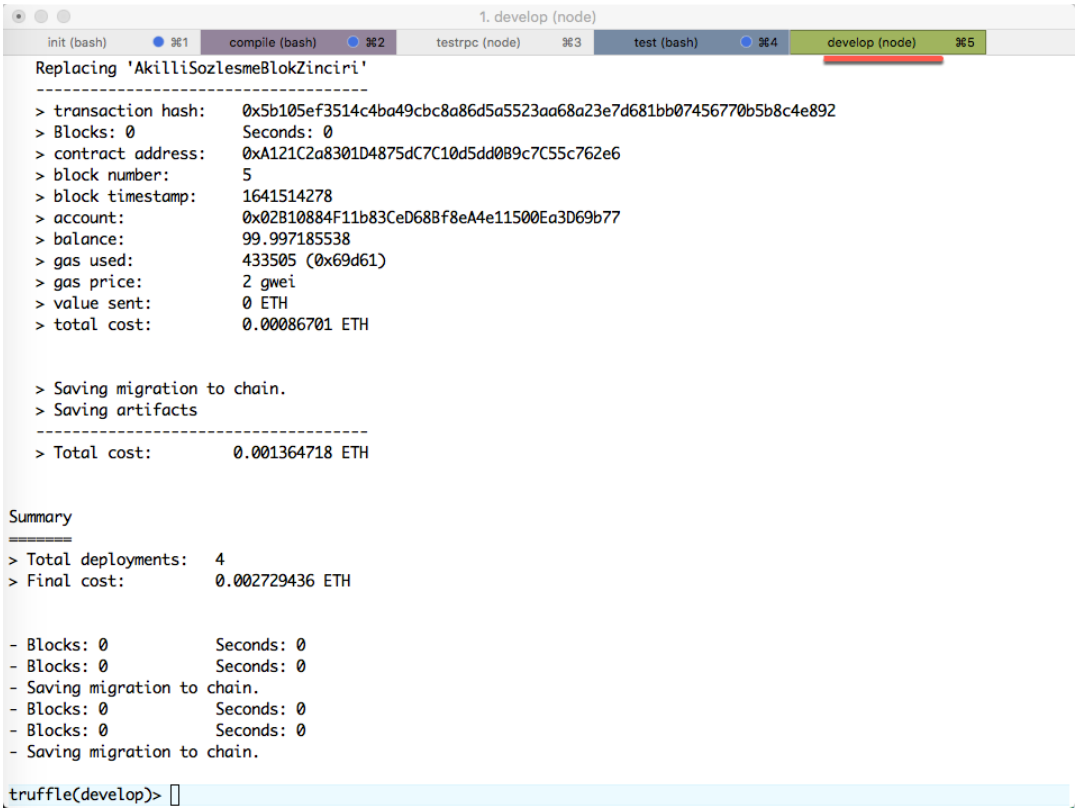
Test denetiminden başarılı olarak geçtikten sonra Şekil 4.6'daki gibi akıllı sözleşme develop (geliştirme) komutu ile gerçek kalıcı Truffle test ortamına

gönderilmektedir. Akıllı sözleşmeyi migrate (göç) komutu ile teste hazır gale getirilmektedir.



Şekil 4.6. Akıllı sözleşme kodu geliştirme ortamına aktarılmakta

Şekil 4.7'deki adımda geliştirme ortamına yüklenilme aşaması biraz zaman alabilmektedir. Bu kısımda sabırla beklemek gerekmektedir. Konsoldaki çıktılar incelediğinde blok zincirine yüklenen akıllı sözleşmelere ve oluşturulan bloklara ait detaylar açıkça gösterilmektedir.



Şekil 4.7. Akıllı sözleşme kodu geliştirme ortamında çalışmakta

Şekil 4.8'de gösterildiği gibi akıllı sözleşme içindeki metotlar JS kodları ile çağırılıp kullanılmaktadır. JS kodlarını HTML içinden de çağırıp kullanmak mümkün olmaktadır. İlk akıllı sözleşme kodu bir selamlama mesajını terminalde göstermektedir. Truffle aracı üzerinden Ethereum için geliştirdiğimiz akıllı sözleşmeleri ücretsiz test etme imkana sahip olunmaktadır. Eğer Doğrudan Ethereum üzerinde testler yapmak da mümkündür ama bu çok pahalı ve masraflı bir tercih olmaktadır. Ethereum blok zincirinde her işlem için ortalama 50 dolar kadar bir komisyon ücreti ödenmektedir. Bu komisyona gas ücreti denmektedir. Burada sadece var olan sabit bir mesaj ekrana bastırılmaktadır. Bunun için bir gas ücreti alınmamaktadır.

Şekil 4.8. Akıllı sözleşme kodundaki fonksiyona JS ile erişilmekte

Gas ücreti yazılan akıllı sözleşmenin boyutuna ve blok zincir ağının yoğunluğuna göre artıp azalabilmektedir. Akılı sözleşmeler az masraflı olması için olabildiğince kısa olmalı ve verimli kodlanmalıdır. Aksi halde her çalıştırıldığından çok masraflı olmaktadır. Şekil 4.9'da akıllı sözleşmedeki bir başka metot çağırılmaktadır. Bu metoda ait 2 parametre gönderilmekte ve bu verilerin blok zinciri üzerinde kaydedilmesi istenilmektedir. Blok zinciri ağı eğer meşgul ediliyorsa o zaman yapılan bu işlemin kayıt altına alınmadan önce doğrulanması gerektiğinden bir gas ücreti ödenme zorunluluğu bulunmaktadır. Yapılan işlemin yer aldığı bloğa ait başlık kısımlarına, cüzdanlara, sözleşme adreslerine ve hash verilerine biraz dikkatle bakıldığında blok içindeki detaylar zihinlerde daha da netleşecektir.

```
1. develop (node)
init (bash) ⌘1 | compile (bash) ⌘2 | testrpc (node) ⌘3 | test (bash) ⌘4 | develop (node) ⌘5
truffle(develop)> iletisim.numaraEkle("Göksel KIBIŞ", 100);
{
  tx: '0x6f6460b16c64a7d5f7a62d4e0f2aa3963afc87baf26a8c2343daaf992c67d8d1',
  receipt: {
    transactionHash: '0x6f6460b16c64a7d5f7a62d4e0f2aa3963afc87baf26a8c2343daaf992c67d8d1',
    transactionIndex: 0,
    blockHash: '0x3a61bd541a55f0f901f31eb188a5a456d01e776186be18bdb17bdc6c30b835aa',
    blockNumber: 7,
    from: '0x02b10884f11b83ced68bf8ea4e11500ea3d69b77',
    to: '0xa121c2a8301d4875dc7c10d5dd0b9c7c55c762e6',
    gasUsed: 44712,
    cumulativeGasUsed: 44712,
    contractAddress: null,
    logs: [],
    status: true,
    logsBloom: '0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000',
    rawLogs: []
  },
  logs: []
}
truffle(develop)> 
```

Şekil 4.9. Akıllı sözleşmedeki fonksiyona parametreler gönderilmekte

Şekil 4.10'da akıllı sözleşmedeki bir metodun her çağırılma işleminde akıllı sözleşme blok zincir ağında çalıştırılmaktadır. Doğrulama işlemleri için de bir gas ücreti ödenmektedir. Akıllı sözleşmeler kullanıldıkça ağa para ödendiğinden bir sözleşme geliştirirken Solidity programlama dilinin inceliklerine hâkim olunması gerekmektedir.

```
truffle(develop)> iletisim.numaraEkle("Muhammet CEYLAN", 200);
{
  tx: '0xc3417688c3b34a1d8688294d09c643f90851affe65d43d69dc4d977562029d77',
  receipt: {
    transactionHash: '0xc3417688c3b34a1d8688294d09c643f90851affe65d43d69dc4d977562029d77',
    transactionIndex: 0,
    blockHash: '0xe52b68234439a5b7f39c060ffe12851f617b7d48d2c3a470d0282e148b426d6b',
    blockNumber: 8,
    from: '0x02b10884f11b83ced68bf8ea4e11500ea3d69b77',
    to: '0xa121c2a8301d4875dc7c10d5dd0b9c7c55c762e6',
    gasUsed: 44724,
    cumulativeGasUsed: 44724,
    contractAddress: null,
    logs: [],
    status: true,
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000',
    rawLogs: []
  },
  logs: []
}
truffle(develop)>
```

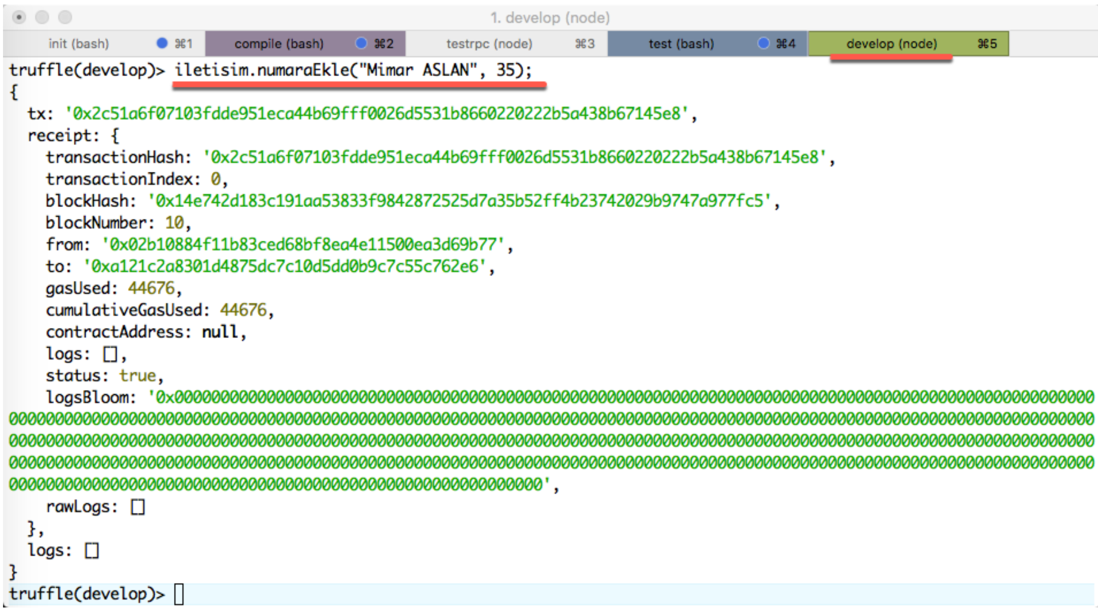
Şekil 4.10. Akıllı sözleşmedeki fonksiyona çağrılmakta

Şekil 4.11’de bir veri daha blok zincirine eklenmektedir. Eklenen her verinin işlem hash değeri farklıdır.

```
truffle(develop)> iletisim.numaraEkle("Mustafa Cem KASAPBAŞI", 61);
{
  tx: '0xccfa1ca5a97454d751433d2857984fe1b9b1b63ac61728b327d0b838bd3d2f79',
  receipt: {
    transactionHash: '0xccfa1ca5a97454d751433d2857984fe1b9b1b63ac61728b327d0b838bd3d2f79',
    transactionIndex: 0,
    blockHash: '0xfbb3c64645372aad948ebb961b94e6b0949c7e16ae275b3f94d5c8ba05ae3336',
    blockNumber: 9,
    from: '0x02b10884f11b83ced68bf8ea4e11500ea3d69b77',
    to: '0xa121c2a8301d4875dc7c10d5dd0b9c7c55c762e6',
    gasUsed: 44808,
    cumulativeGasUsed: 44808,
    contractAddress: null,
    logs: [],
    status: true,
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000',
    rawLogs: []
  },
  logs: []
}
truffle(develop)>
```

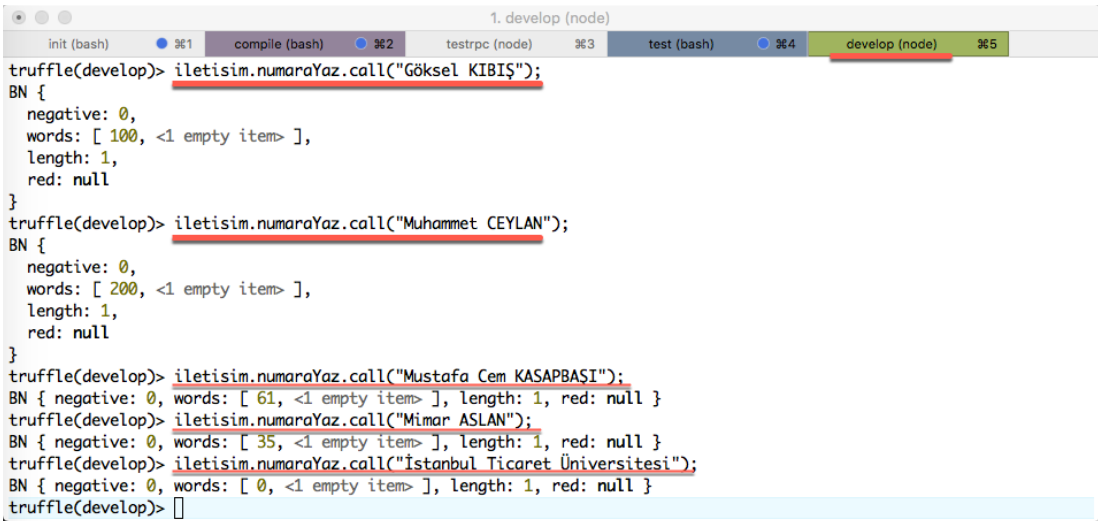
Şekil 4.11. Akıllı sözleşmedeki fonksiyon kullanıldıkça blok dolmakta

Blok zinciri üzerinde yapılan bir işlemin şeffaf olduğu Şekil 4.12’de gösterilmektedir.



Şekil 4.12. Akıllı sözleşme çalıştırıldıkça oluşturulan blok değerleri

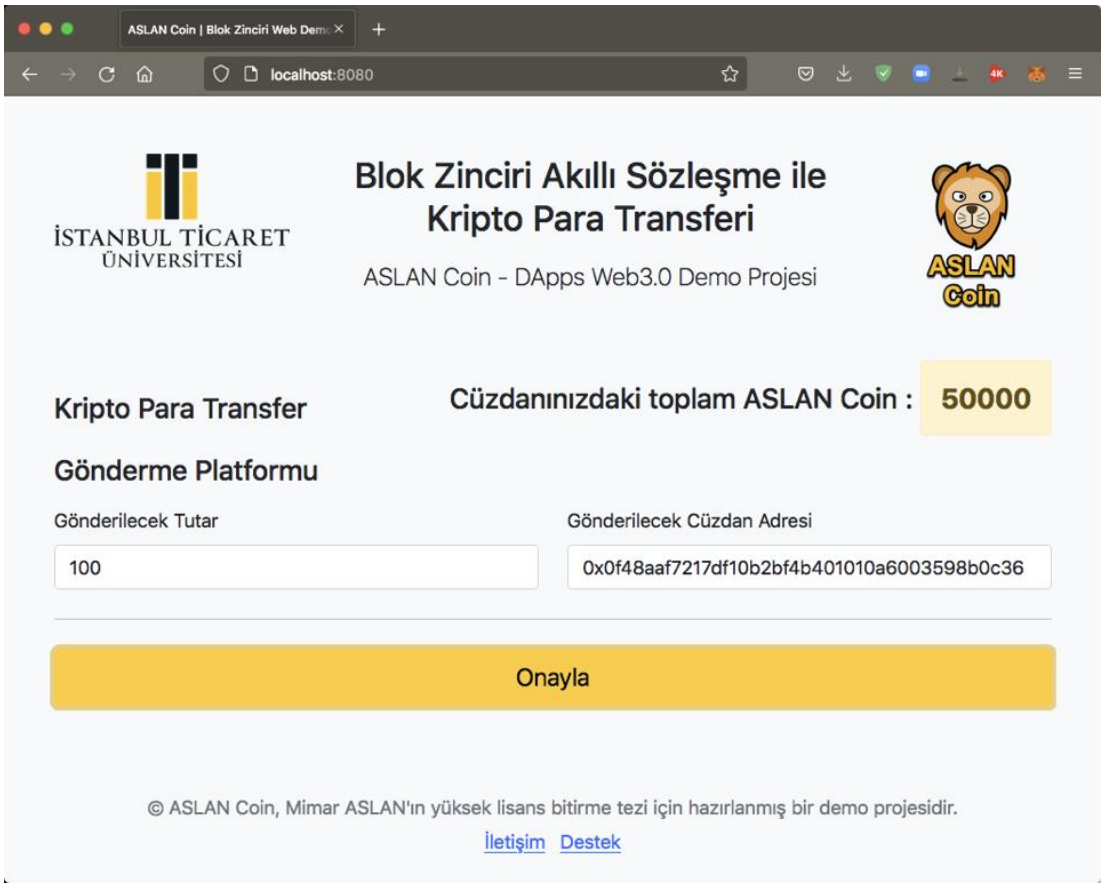
Blok zincirinde kayıtlı veriler Şekil 4.13'teki gibi çağırılmaktadır. Sözleşmeye ait sadece verileri gösteren metot eğer ağa yeni bir veri yazmıyorsa o zaman çağırıldığında bunun için bir gas ücreti alınmamaktadır.



Şekil 4.13. Sözleşmedeki başka bir fonksiyonla sonuçlar gösterilmekte

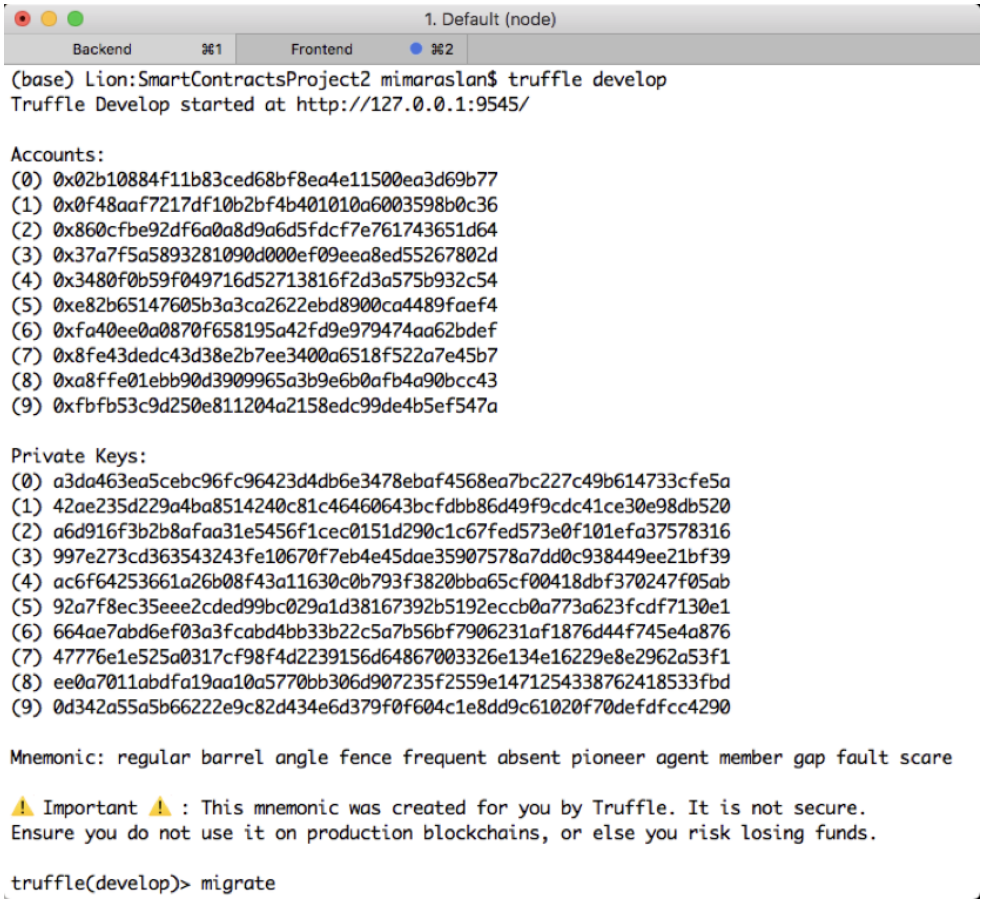
Akıllı sözleşmeleri web sayfaları üzerinden de kullanılmaktadır. Şekil 4.14'te web sayfası üzerinde çalışmakta olan ara yüzden coin transferi gerçekleştirilmektedir. Truffle aracının unbox webpack komutlarıyla bir klasörde web üzerinde çalışacak yeni bir proje oluşturulmaktadır (Truffle, 2021). Aslan Coin adındaki kripto paranın arzı toplamda sadece 50,000 adettir. Kripto paranın adetini akıllı

sözleşmeyi oluşturan yazılımcı, akıllı sözleşme içerisinden istediği gibi değiştirip belirleyebilmektedir. Test hesapları arasında bu kripto parayı akıllı sözleşme üzerinden gönderilmektedir. Web tarayıcısına Metamask isimli bir eklentinin kurulmuş olması gerekmektedir. Metamask ücretsizdir. Test ederken verilen cüzdan hesaplarına terminalden değil de Metamask üzerinden erişmek ve gerçekleştirilen işlemleri oradan takip etmek daha kolay ve profesyonelce bir tercih olmaktadır. Ekler bölümünde projeye ait kodlar yer almaktadır.



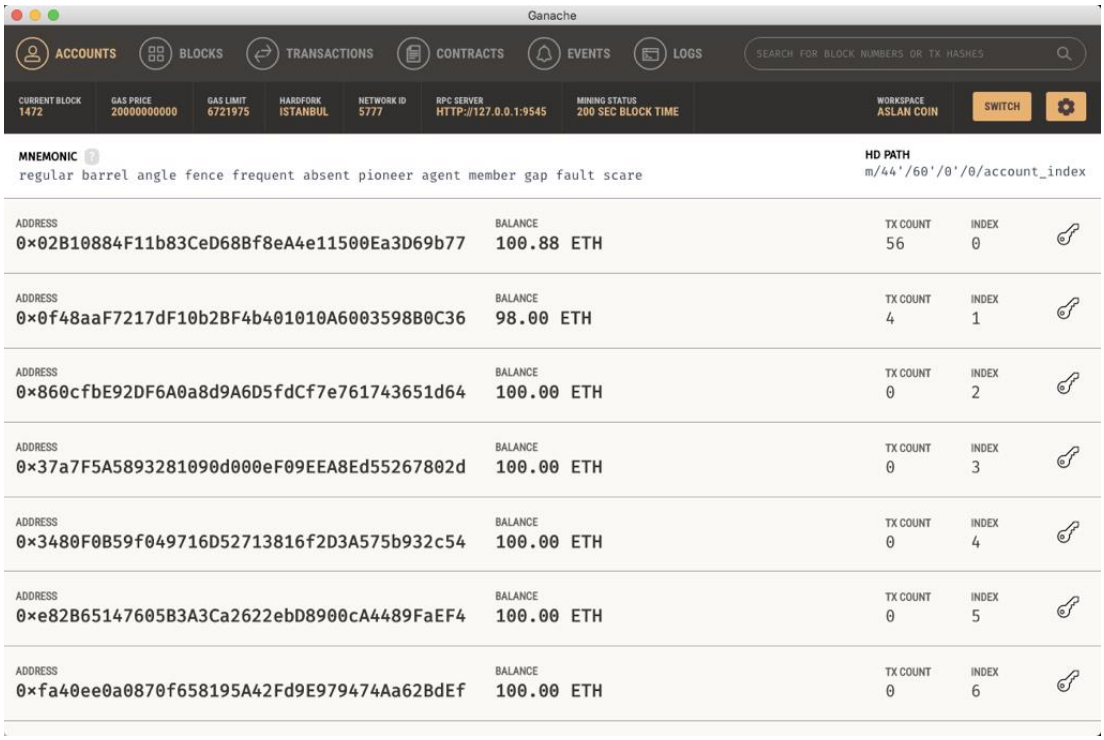
Şekil 4.14. Akıllı sözleşme ile DApps Web3.0 uygulama projesi

Şekil 4.15'te gösterildiği gibi Truffle aracına proje gönderilmektedir. Test için 10 tane cüzdan hesabı ve hesaplara ait özel anahtar değerleri tanımlanmıştır. Her cüzdanda 100 ETH bulunmaktadır. Test ortamında gas ücretleri bu cüzdanlardaki bu ETH ile ödenmektedir. Özel anahtar değerlerinden ilk sıradaki ilk cüzdanı temsil etmektedir ve 50,000 Aslan Coin bu hesapta yer almaktadır. Metamask aracına cüzdanlar bağlanırken bu özel anahtar değerleri kullanılacaktır.



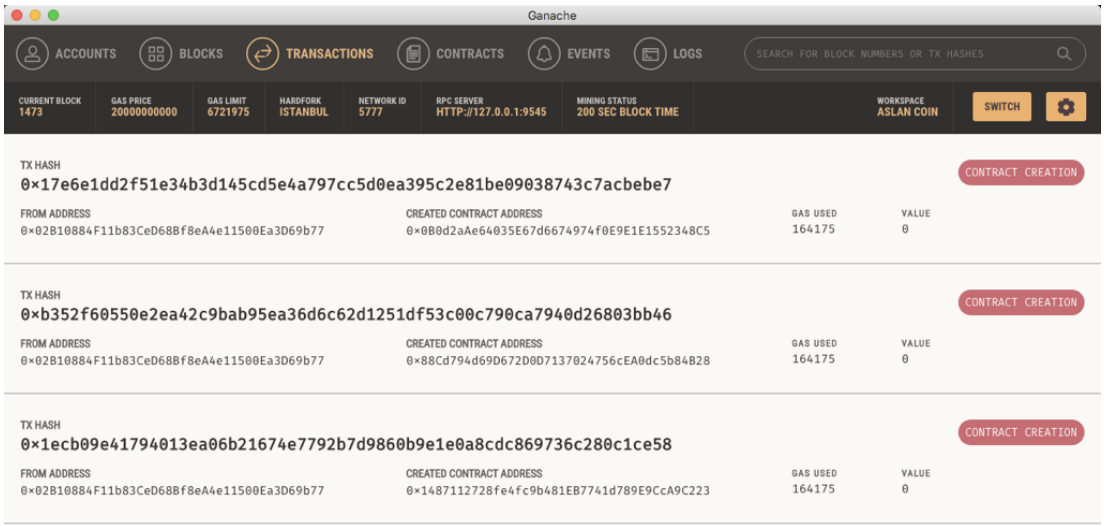
Şekil 4.15. Blok Zinciri web uygulaması çalıştırılmakta (Truffle, 2021)

Çalışma boyunca terminalde çalışan Truffle uygulaması hiç kapatılmamamladır. Ganache aracı çalıştırıldığında Şekil 4.16'daki gibi bir program açılmaktadır. Hesaplardaki 100 ETH, cüzdan bilgileri, blok, sözleşme ve tüm işlem bilgileri Ganache aracı üzerinden takip edilmektedir.



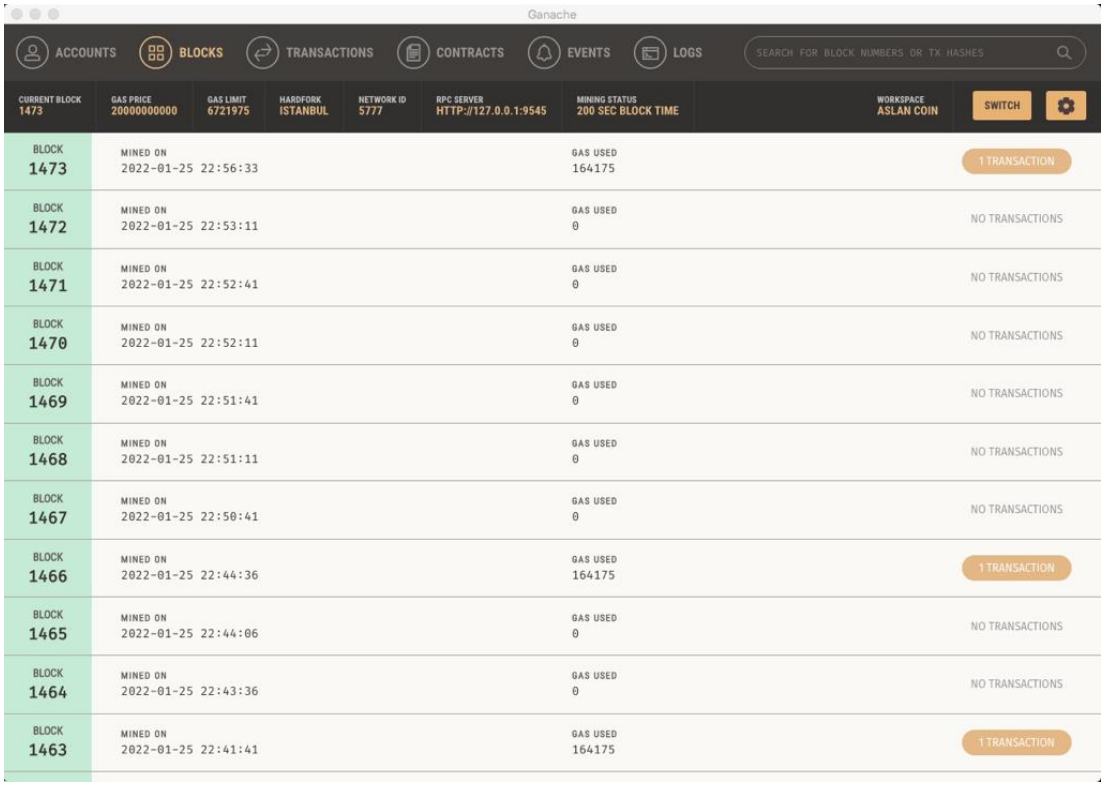
Şekil 4.16. Cüzdanlar oluşturulmakta (Ganache, 2021)

Şekil 4.17'de gösterildiği gibi Ganache aracının işlemler menüsü açıldığında yüklenen sözleşmelere erişilmektedir. Truffle aracı sözleşmeyi Ethereum ağına her yüklediğinde sözleşme için yeni bir hash kodu oluşturulmaktadır. Ağa yüklenen bir sözleşme bir daha asla değiştirilmez. Eğer değişiklik yapılacaksa yeniden yüklenir fakat eski sözleşme asla ağdan silinmemektedir. Blok zincir ağında kayıt altına alınan hiçbir şey asla değiştirilemez ve silinemez.



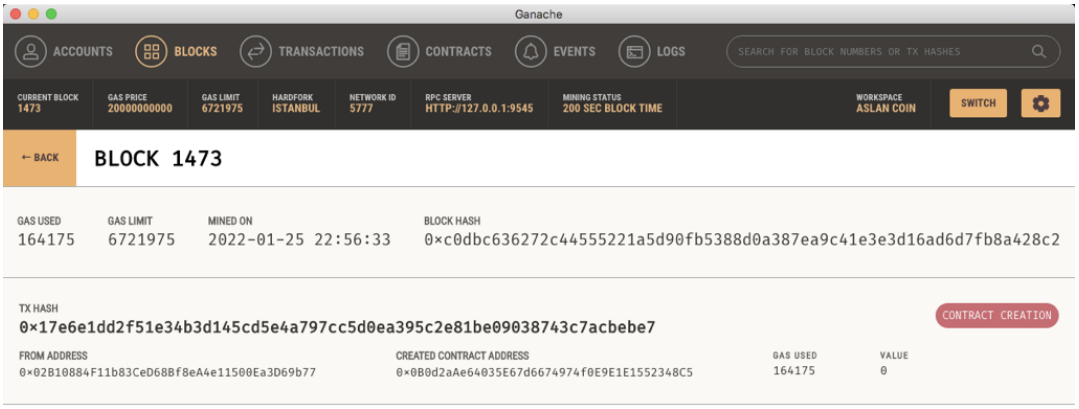
Şekil 4.17. Uygulama her çalıştırıldığında sözleşme ağa yüklenmekte

Şekil 4.18'de blok zincirine ait blokların bilgisi gösterilmektedir. Blok zincirinde hiçbir işlem yapılmasa ağa yine de blok eklenmektedir. Ganache aracının ayarlar menüsünden blok eklenme süresi ayarlanabilmektedir. Bu projede blok eklenme süresi 200 saniye olarak ayarlanmıştır.



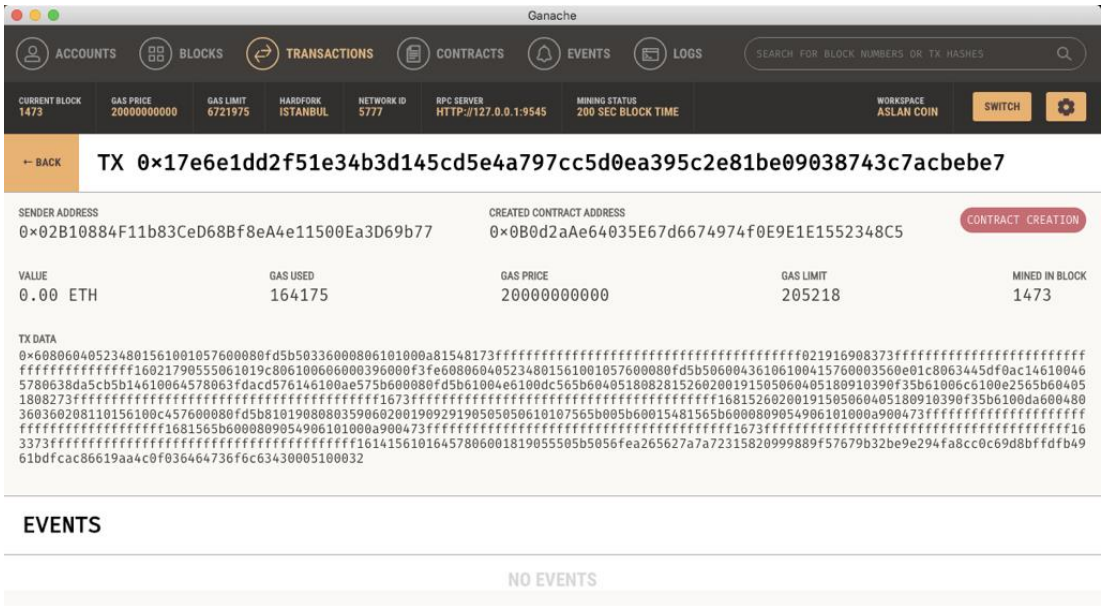
Şekil 4.18. Bloklar ve içlerindeki işlemler

Şekil 4.19'da gösterildiği gibi İşlem yapılan blokların üzerine tıklanırsa içlerindeki işlemlerin listesine erişilmektedir.



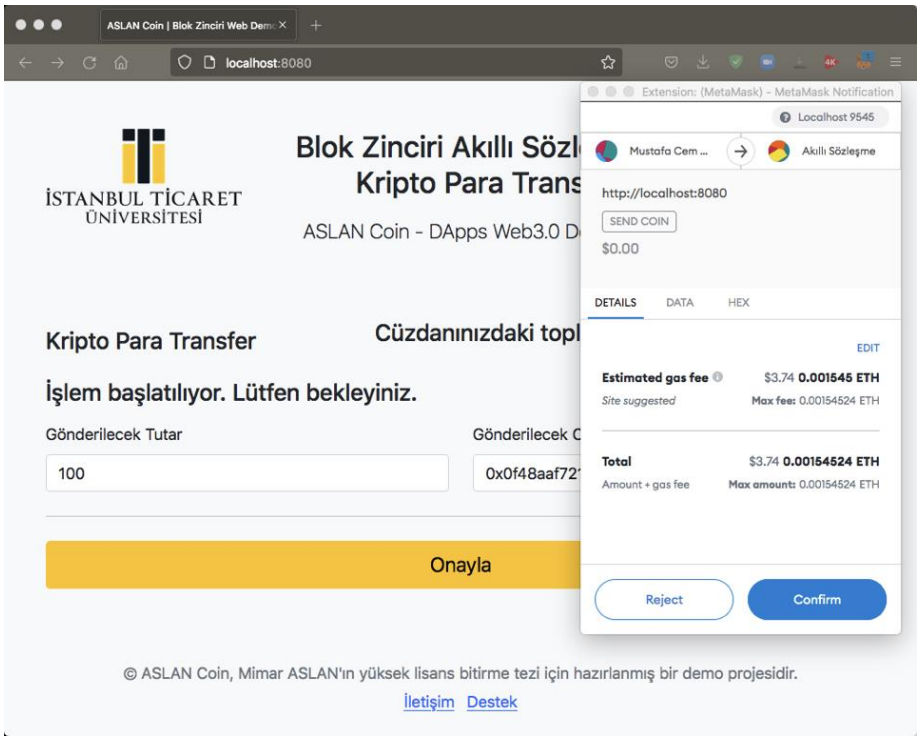
Şekil 4.19. Seçilen blok ve içerisinde ağa eklenen sözleşme

Blok zincirinde Şekil 4.20'deki gibi işlem yapılırken kullanılmış bir akıllı sözleşmenin detayı şeffaf olarak gösterilmektedir.



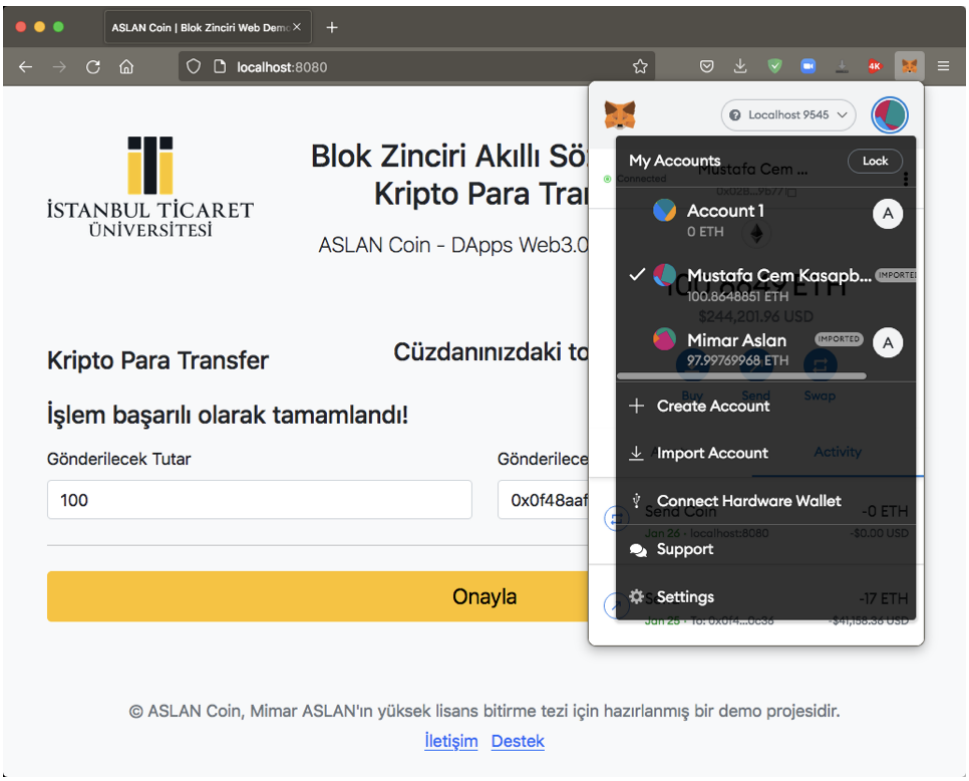
Şekil 4.20. Ağa eklenen bir sözleşmenin ve detayları

Metamask uygulamasının import (içe aktar) kısmından test hesaplarına ait özel anahtarlar değerleri sırasıyla girildiğinde o cüzdanların içleri Metamask programında listelenip gösterilmektedir. Karışmaması için cüzdanlara Metamask takma isimler verilmesini desteklemektedir. Hesabında 50,000 Aslan coin bulunduran hesaba Mustafa Cem Kasapbaşı adı verilmiştir. Sözleşmenin de bir hesabı vardır. Ona da Akıllı Sözleşme adı verilmiştir. Transfer yapılmak istenen cüzdana da Mimar Aslan adı verilmiştir. Böylece cüzdanların yönetimi daha da kolaylaştırılmıştır. Aslan Coin'in gönderilmek istediği miktarı ve cüzdan numarasını belirledikten sonra Şekil 4.21'de gösterildiği gibi web ara yüzünden onay verilmektedir ve Metamask cüzdanı da otomatik olarak açılmaktadır. Metamask'taki confirm (doğrulama) düğmesine basılınca belirtilen miktarda Aslan coin için gas ücreti ödenip alıcı hesaba aktarılmaktadır.



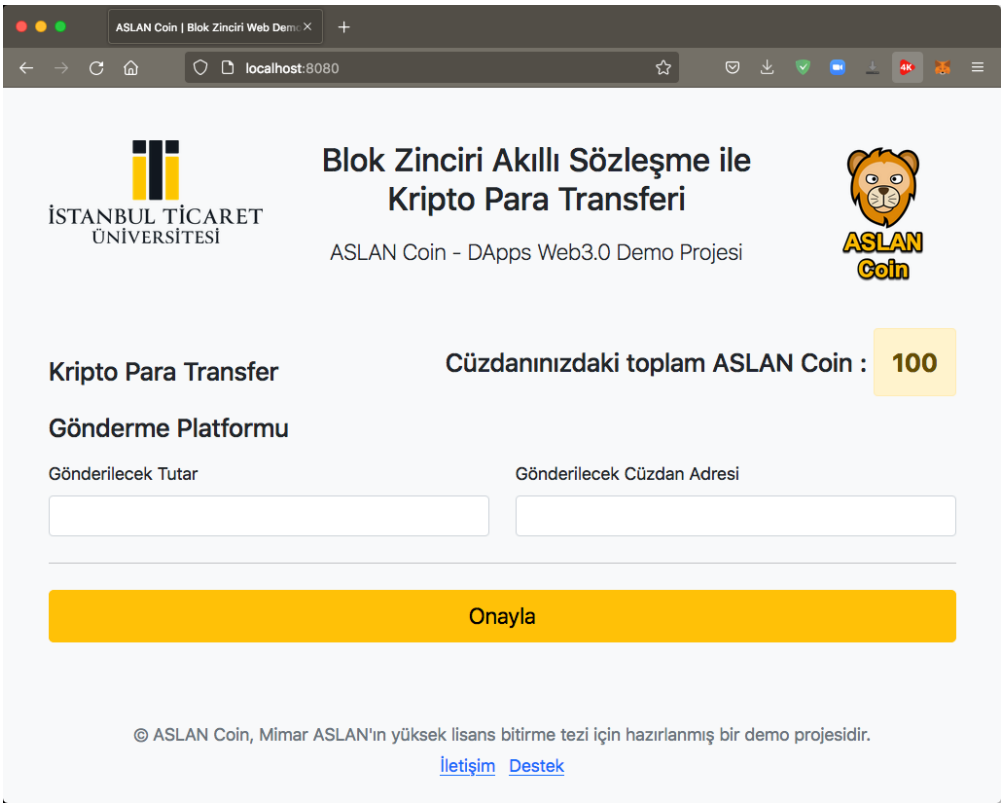
Şekil 4.21. Akıllı sözleşme üzerinden transfer gerçekleştirilmekte

Şekil 4.22'de hesapları listelenmektedir. Aktarılmanın yapıldığı hesabı seçip web ara yüzü yenilendiğinde aktarma işleminin son durumunu gösterilecektir.



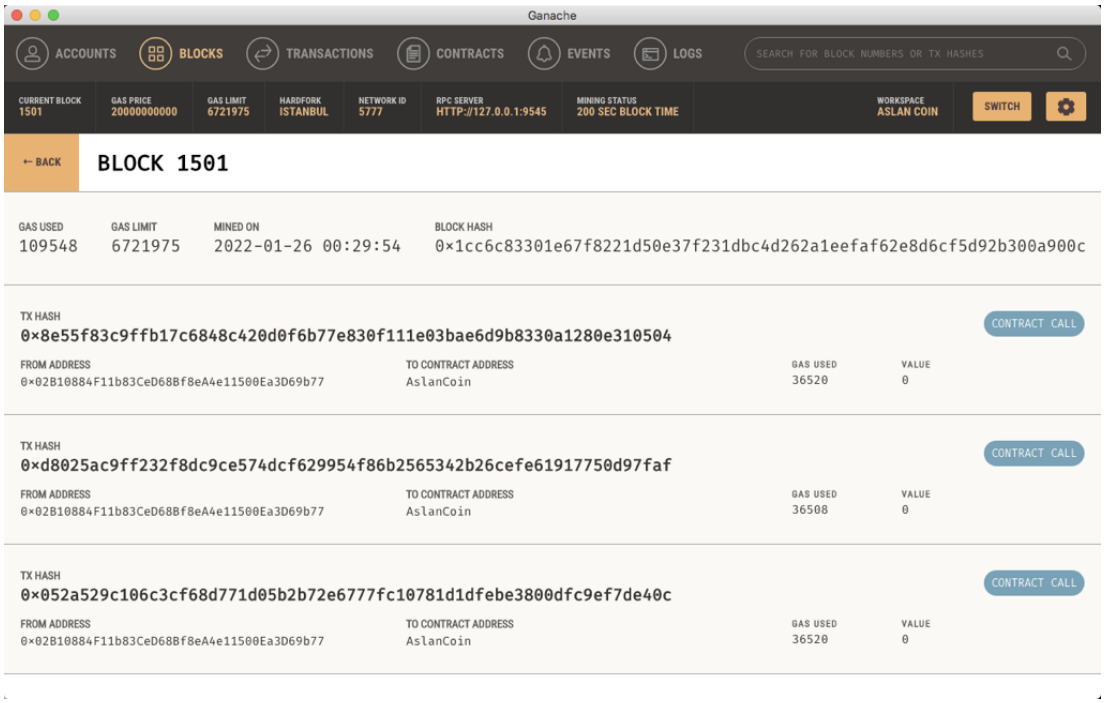
Şekil 4.22. Cüzdan hesapları arasında geçiş yapılmakta

Şekil 4.23'te görüldüğü gibi 100 Aslan Coin kripto parasının aktarılma işlemi başarılı olarak gerçekleştirilmiştir. Akıllı sözleşme başarılı olarak çalışmaktadır. Hesaplar arasında böyle birçok transfer yapılırsa bloklarda yapılan işlemlere dair tüm detaylar blok zincirinde tutulacaktır. Blok zinciri değiştirilemez bir kayıt defteri olarak çalışmaktadır.



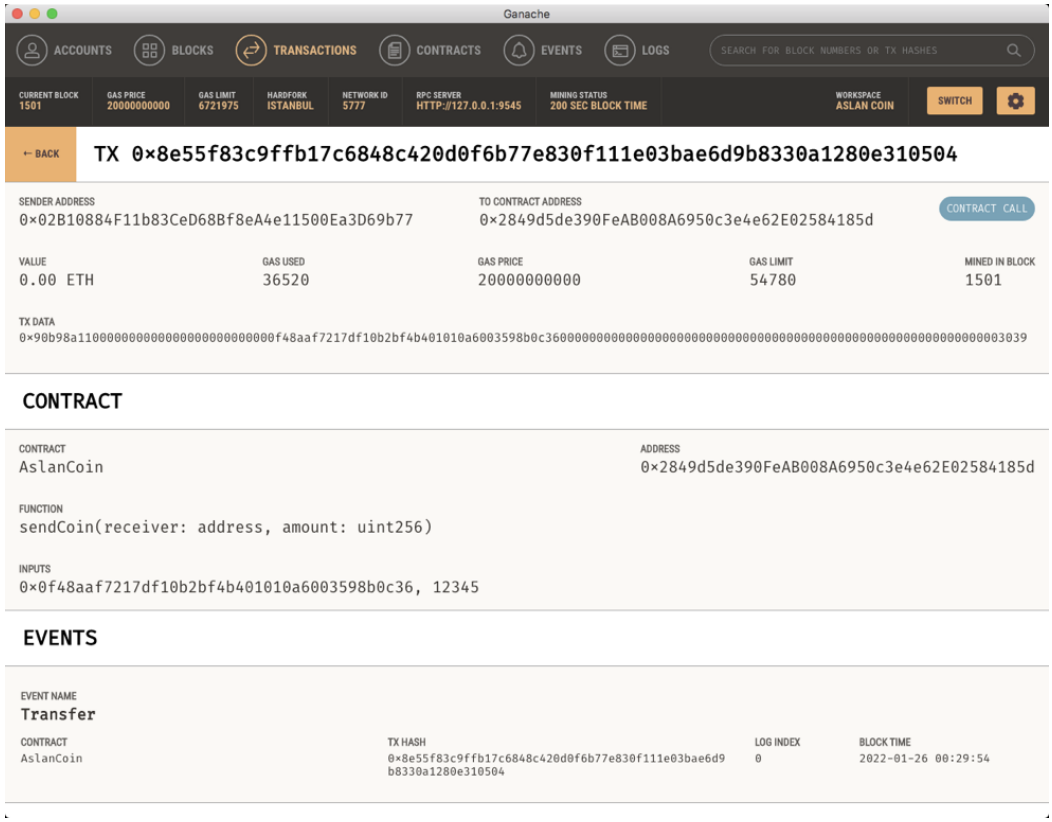
Şekil 4.23. Transfer yapılan cüzdandaki toplam tutar

Şekil 4.24'te gösterildiği gibi işlem yapılan bloklardan birini açtığımızda çağırılan akıllı sözleşmelere ve detaylarına erişilmektedir.



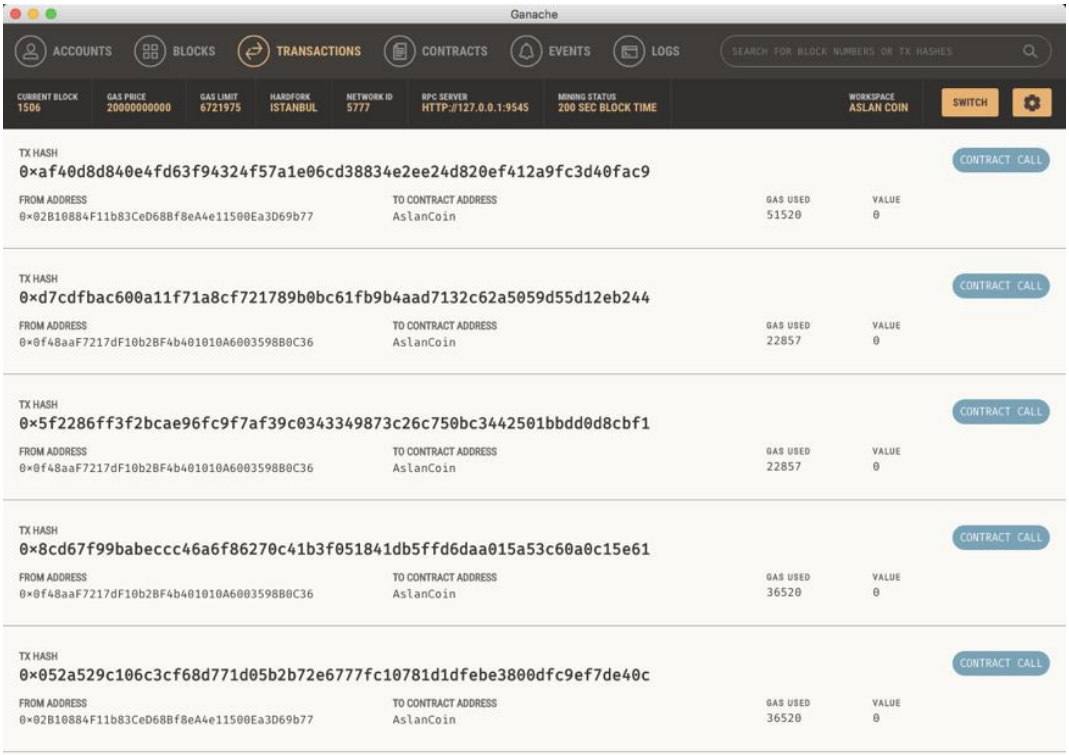
Şekil 4.24. İşlemler akıllı sözleşme üzerinden gerçekleştirilmekte

Şekil 4.25'te seçilen bir işlemin için çalıştırılan akıllı sözleşme ve işlem detayları gösterilmektedir.



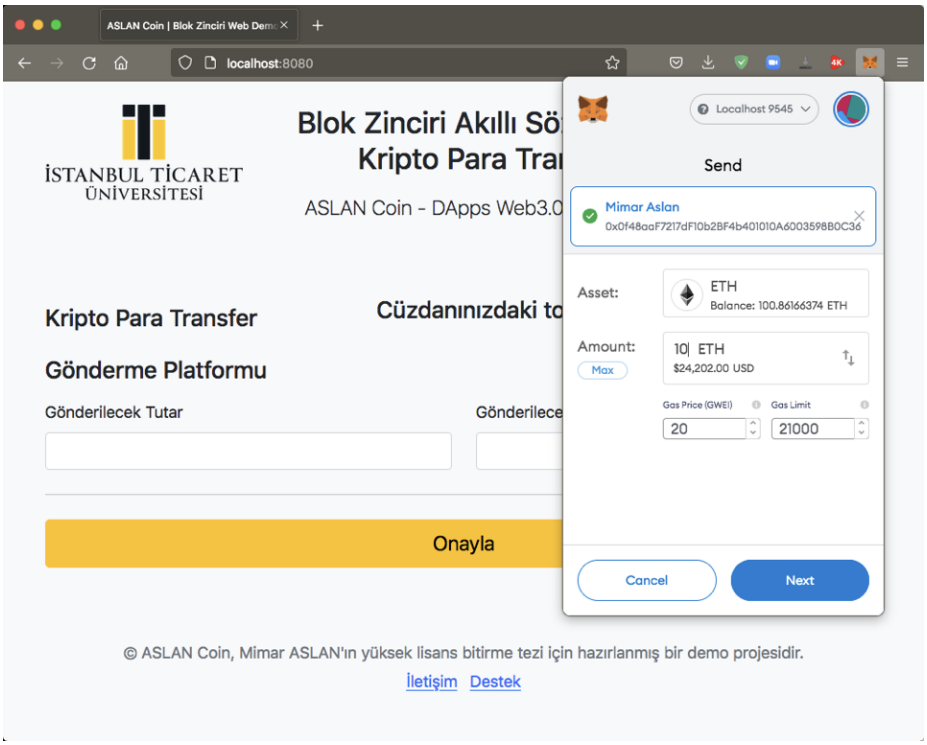
Şekil 4.25. Onaylanan bir işlemin detayı

Cüzdan hesaplarından kim kime hangi akıllı sözleşme üzerinden hangi işlemi gerçekleştirmiş ise hepsinin detaylarına erişmek mümkündür. Şekil 4.26'da işlemler menüsünden gerçekleştirilen tüm işlemlerin listesine erişmek mümkündür. Blok zincirinde yapılan bir işlem bir daha asla geriye alınamamaktadır. Her şey şeffaftır. Görüldüğü gibi kişiye dayalı bir güven söz konusu değildir. Blok zincirinde güveni akıllı sözleşmeler ve doğrulama mekanizmaları gerçekleştirmektedir.



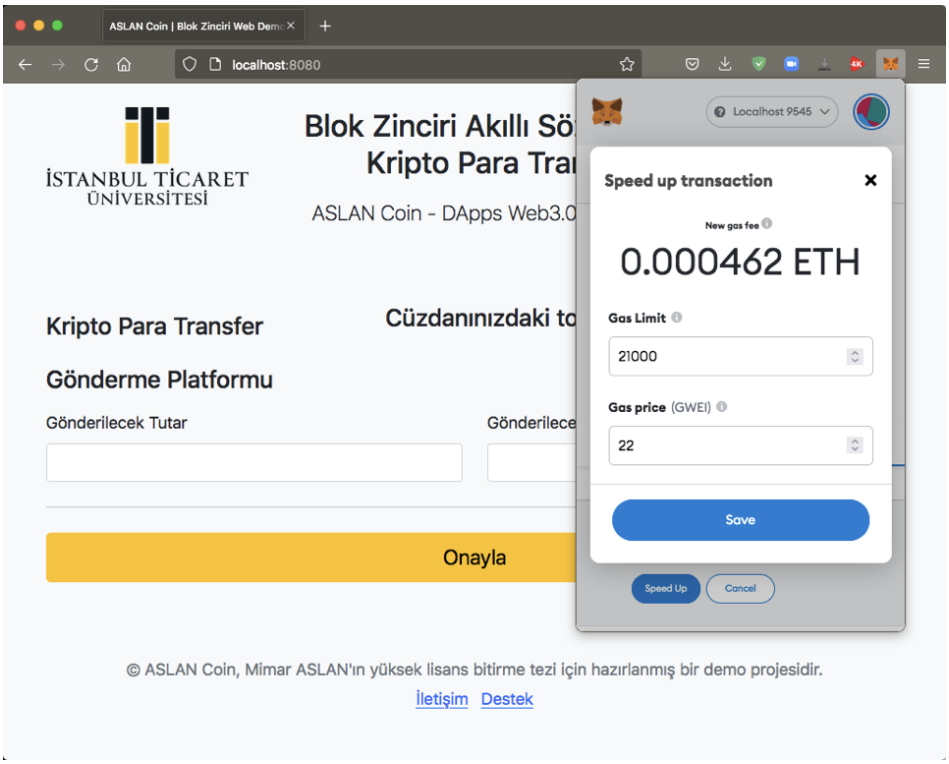
Şekil 4.26. Gerçekleştirilen işlemlerin listesi

Şekil 4.27'de gösterildiği gibi Metamask üzerinden test hesaplarındaki ETH kripto paralarının doğrudan hesaplar arasında gönderilmesi de mümkündür.



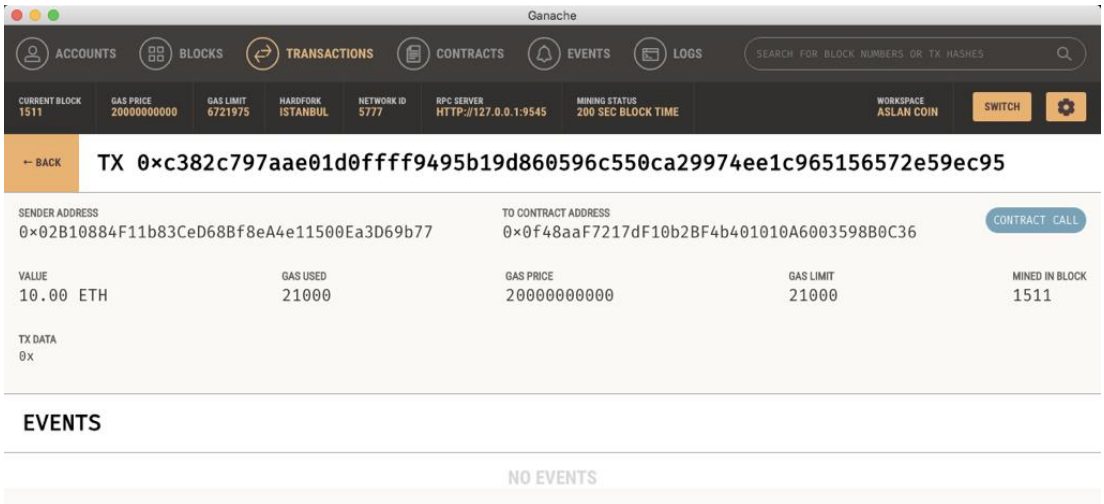
Şekil 4.27. Cüzdanlar arasında ETH gönderilmekte

Yapılan bir işlemin daha çabuk gerçekleşmesi için Şekil 4.28'de gösterildiği gibi daha yüksek bir gas ücreti ödenmesi de mümkündür.



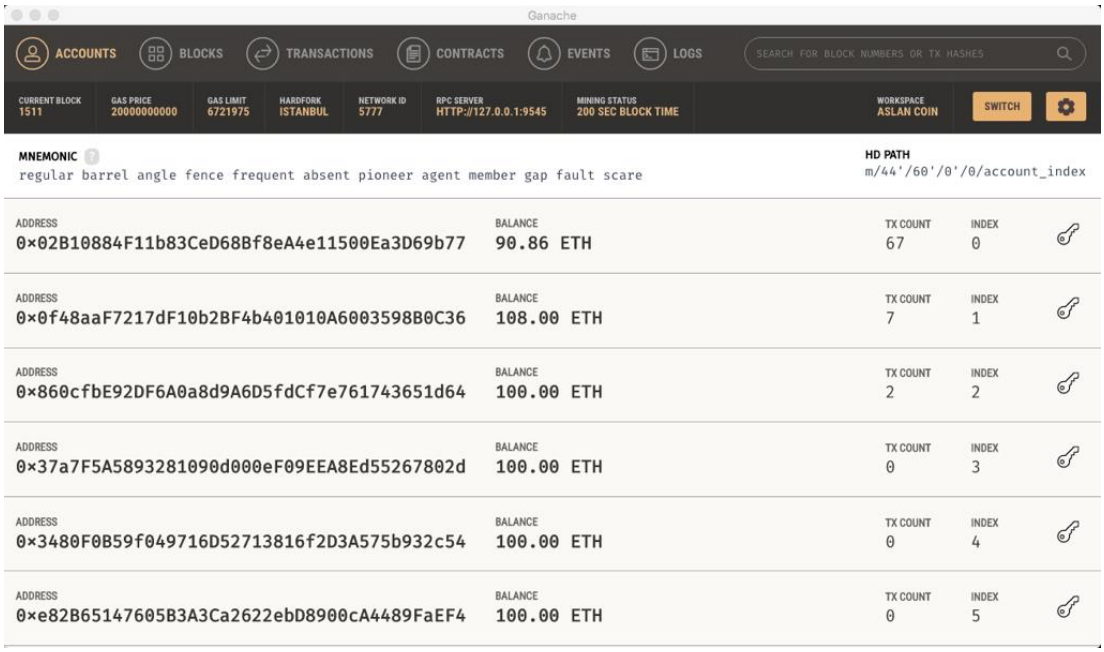
Şekil 4.28. Ağda yapılan işlem için ödenen komisyon

Cüzdanlar arası gerçekleşen bu ETH transferi de Şekil 4.29'da gösterildiği gibi blok zincirinde tutulmaktadır. Cüzdanlara ait her hareket kayıt altına alınmaktadır. Akıllı sözleşme her çağırıldığında araya girmektedir.



Şekil 4.29. Gönderilen ETH işleminin detayı

Cüzdan hesaplarındaki ETH miktarları ve yaptıkları tüm işlemlerin detayı Şekil 4.30'da listelenmektedir. Blok zinciri üzerinde gerçekleştirilen tüm adımlar Ganache uygulaması üzerinden takip edilmektedir.



Şekil 4.30. Cüzdanlar ve hesaplarındaki ETH miktarları

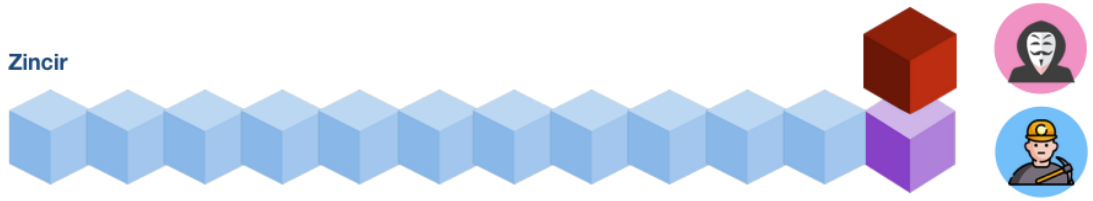
5. ARAŞTIRMA BULGULARI VE TARTIŞMA

Bu çalışmada en yaygın olarak kullanılan blok zinciri uygulama geliştirme platformları, altyapıda kullanılan doğrulama algoritmaları ve blok zincir ağının güvenlik analizi ele alınmıştır. Daha önce yapılan çalışmalarda genelde blok zincirine ait bir ya da birkaç özellik incelenmiştir. Blok zincir ekosisteminin genel altyapısındaki çalışma mekanizmalarını ele alıp blok ağının değiştirilmeye karşı dayanıklılık gücünü deneysel testlerle sayısal olarak gösterme bu çalışmada hedeflenmiştir.

Ethereum, Cosmos, Cardano, EOS, Hyperledger blok zinciri platformlarıdır. Her blok zinciri platformu farklı bir amaca hizmet etmektedir. Geri planda yapılan işlemlerin ağda geçerli olabilmesi için de en az bir tane konsensüs mekanizmasına ihtiyaç duyulmaktadır. En yaygın olarak PoW, PoS, DPoS ve DAG mekanizmaları kullanılmaktadır. PoW PoS, DPoS mimarileri tek ve büyük bir zincir yapısında tasarlanmaktadır. Zincir büyüdükçe yönetimleri zorlaşmakta ve ciddi gecikmeler yaşanmaktadır. DAG mekanizması ağırlıkların birikimine göre çalışır ve mimarisinde yeni işlemler zincire kolayca dahil edilmektedir. Yapılan analizler, blok zincir altyapılarının kurulu olmasını gerektirmektedir. Bu altyapıların analizleri bu sebeple makalede verildiği gibi teorik modeller üzerinde yapılabilmektedir. PoW, PoS, DPoS ve DAG konsensüs mekanizmalarını alıp birkaç bilgisayara kurup her biri için test blok zincir ağı oluşturarak başka yönlerden özel analizler yapılmak istense bile bu sistemler milyarlarca dolarlık dev kripto para platformların altyapı mimarilerindeki ticari projeler olarak yer aldığından kodlarının tamamına erişmek mümkün olamaz. Blok zincirini ele geçirmeye çalışan kötü niyetli saldırgan kişiler elbette olacaktır. Ağa saldırı düzenleyen birinin ağı ele geçirmesi blok zincirinin dağıtık mimari yapısı sayesinde ve ağdaki diğer madenci düğümlerinde çalışan denetleme mekanizma algoritmaları sebebiyle çok zordur.

Blok zincirinde bir önceki bloğun hash değeri bir sonraki bloğa mutlaka kaydedilmektedir. En sondaki bloğun içindeki bilgiler henüz zincire eklenmediğinden başka bir blokla onun karıştırılma imkânı ne yazık ki yoktur. En

sonda yer alan blok daima değiştirilme tehlikesi ile karşı karşıya kalmaktadır. Şekil 5.1'de saldırgana zincire en son bloktan itibaren saldırıp ele geçirilme fırsatı verilmektedir. Hesaplamalar yapılırken simülasyonda ağa eklen her blokta saldırı yapılması için blok sayı değeri z = 1 ve kötü niyetli bir düğümün bir sonraki bloğu bulma olasılık değeri q = 0,1 olarak sabitlenmiştir. Simülasyona ait kodlar ekler bölümünde yer almaktadır.



Şekil 5.1. Zincirdeki en son bloğa yapılan saldırı

Eklenen blokların sayısı arttıkça blok zincir ağını kırılması da sıfıra doğru yaklaşmaktadır. Eklenen 12 bloktan sonra zinciri kırmak neredeyse imkânsız bir hal almaktadır. Çizelge 5.1'de gösterildiği gibi zincire eklenen blok sayısı z ne kadar çok artarsa zinciri ele geçirip değiştirmek de o kadar zorlaşmaktadır.

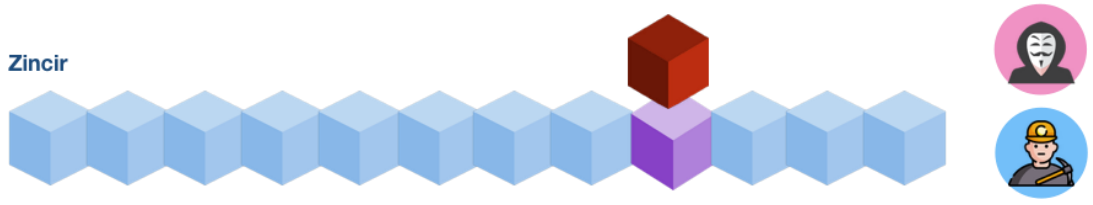
Çizelge 5.1. Saldırgan son bloktan itibaren her blokta saldırı düzenlemekte

z (Zincire eklenen blok sayısı)	P (Saldırganın başarılı olma olasılığı)
0	1
1	0,2045873
2	0,0509779
3	0,0131722
4	0,0034552
5	0,0009137
6	0,0002428
7	0,0000647
8	0,0000173
9	0,0000046
10	0,0000012
11	0,0000003
12	0,0000001

Blok zinciri ağının kırılma olasılığı yeni bloklar eklendikçe katlanarak azalmaktadır. Zincire yeni bloklar eklendikçe zincirdeki verilerin değiştirilme olasılığının sonuçları Şekil 5.2.'de gösterilmektedir.

Şekil 5.2.Saldırganın zincire en son bloktan itibaren saldırma durumu

Şekil 5.3'teki blok zincirinde daha önceden doğrulaması yapılmış bir bloğa saldırı düzenlenmektedir. Hesaplamalar yapılırken simülasyonda ağa eklen her 4 blokta bir saldırı yapılması için blok sayı değeri z = 4 ve kötü niyetli bir düğümün bir sonraki bloğu bulma olasılık değeri q = 0,3 olarak sabitlenmiştir.



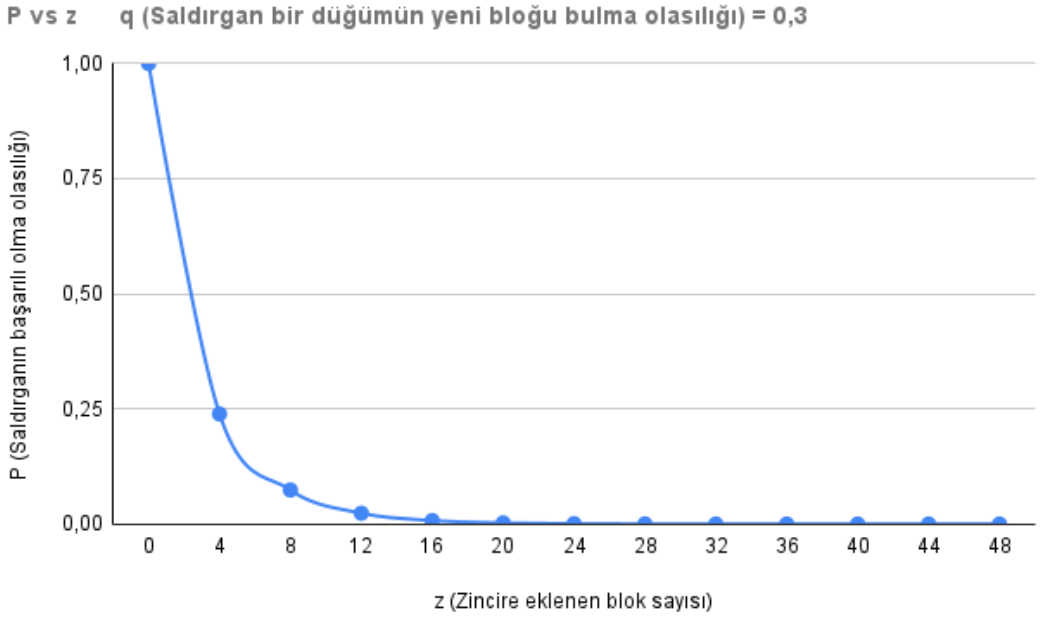
Şekil 5.3. Zincirdeki eski bloklara yapılan saldırı

Çizelge 5.2'deki gösterilen değerlere göre blok zinciri ağı her 4 blokta bir saldırı testine tabi tutulmaktadır. Saldırganın blok zincirindeki önceden kaydedilmiş bloklara saldırıp başarılı olamadığı açıkça görülmektedir. Zincire eklenen blokların sayısı arttıkça kötü niyetli bir düğümün saldırısında başarılı olma olasılığı da azalmaktadır.

Çizelge 5.2. Saldırgan düğüm her 4 blokta bir saldırı düzenlemekte

z (Zincire eklenen blok sayısı)	P (Saldırganın başarılı olma olasılığı)
0	1
4	0,2391269
8	0,0739244
12	0,0235840
16	0,0076219
20	0,0024804
24	0,0008106
28	0,0002657
32	0,0000873
36	0,0000287
40	0,0000095
44	0,0000031
48	0,0000010

Saldırgan bir düğümün zincirde daha önceden kayıtlı olan bir bloğu seçip ona saldırması zincire en sondan saldırmasından daha zor olmaktadır. Şekil 5.4'te olduğu 48 blok eklendikten sonra artık ağın kırılıp ele geçirilme olasılığı neredeyse sıfırdır.



Şekil 5.4. Eklenen her 4. blokta bir gerçekleştirilen saldırı durumu

PoW, PoS ve DPoS madencilerin yarışa ve rekabete dayalı doğrulama yaptıkları mekanizmadır (Cao vd., 2020). Fikir birliği mekanizmalarından en yaygın olarak kullanılan PoW çok enerji, PoS ise daha az enerji tüketmektedir (Košťál vd., 2018). PoW, PoS ve DPoS altyapılarında hash algoritmalarını kullanmaktadır. PoW'da madencilere ihtiyaç varken PoS'ta yüksek güçte madencilik çiftliklerine ihtiyaç yoktur (Ethereum, 2021). Ethereum, PoW ile çalışmaktadır ama PoS altyapısına geçmeyi hedeflemektedir. Bunu başarınca ~%99,95 daha az enerji kullanacaktır. Bu çalışmada, PoS fikir birliği mekanizması ve onu daha ileriye taşıyacak olan DPoS ile de kıyaslaması yapılmaktadır. PoS, blok zincir ağına ait kripto paradan elinde en çok bulunduran sadece zengin madencilere kazanma önceliği tanırken DPoS ağdaki kişilere oy hakkı verip adil bir seçim yaklaşımını önermektedir. PoS'ta madenci düğümler arasında bir seçim yoktur ama DPoS ile blok üreticileri seçimle belirlenmektedir. Bu sayede ağdaki madencilerin kazançları dengelenmektedir. Fikir birliği mekanizmalarından DAG da incelenmiştir ve ağır hash hesaplamaları yerine, yeni bir işlemin onaylanmasını kullanılmaktadır. Yapılan bir önceki işlemin onay verilme referansı dikkate alınmakta ve ağa yeni bloklar eklenerek büyütülmektedir. Fikir birliği mekanizmalarının amacı yapılan işlemleri doğrulamak ve ağa yeni bir blok eklemektir.

Bitcoin blok zincirine yapılan ciddi bir saldırıda 10 bloğa kadar ağın dayanıklılığına garanti verilirken (z=10 P=0,0000012) (Nakamoto, 2008), bu çalışmada Çizelge 5.3'te dayanıklılık seviyesi için blok sayısını 12 bloğa kadar çıkarılması sağlanmaktadır (z=12 P=0,0000001).

Çizelge 5.3. Blok zinciri ağının dayanıklılık kıyası

z (Zincire eklenen blok sayısı)	P (Saldırganın başarılı olma olasılığı)
10	0,0000012
12	0,0000001

Saldırgan bir düğümün yeni ve geçerli bir bloğu bulma olasılığı Çizelge 5.4'te Bitcoin blok zincirinde 5 blokta bir değerlendirilirken (z=5 P=0,1773523 z=10 P=0,0416605 z=15 P=0,0101008) bu çalışmada, saldırgana daha müsamahalı

davranarak her 4 blokta bir yeni ve doğrulanmış bir blok bulma olasılığı fırsatı tanınmakta ve sonrasında veriler kıyaslanmaktadır. (z=4 P=0,2391269 z=8 P=0,0739244 z=12 P=0,0235840 z=16 P=0,0076219).

Çizelge 5.4. Blok zincirinde her 5 blokta bir yapılan saldırı kıyası

z (Zincire eklenen blok sayısı)	P (Saldırganın başarılı olma olasılığı)
5	0,1773523
10	0,0416605
15	0,0101008

Çizelge 5.5'te yapılan saldırı girişimleri açıkça göstermektedir ki blok zincir ağını kırmak eklenen birkaç bloktan sonra çok zordur. Ağa yeni bir blok ekleyerek zincirin ele geçirilme teşebbüslerinin ne kadar çetin ve zahmetli bir süreç olduğunu yapılan testlerin neticesinde açıkça görülmektedir. Ağa eklenen yeni bloklardan sonra zincire belirli bir noktadan saldırıp kırma olasılığının da çok düşük olduğunu elde edilen sonuçlar göstermektedir. Ağa saldırı yapıldığında geçerli olan dürüst bir zincirin yerini alabilmek için ondan daha uzun bir zincire ulaşılması gerekmektedir. Dürüst zincirin uzunluğunu geçme olasılıklarının yer aldığı bölgelerin çok sınırlı olduğunu yapılan saldırı denemeleri neticesinde görülmektedir.

Çizelge 5.5. Blok zincirinde her 4 blokta bir yapılan saldırı kıyası

z (Zincire eklenen blok sayısı)	P (Saldırganın başarılı olma olasılığı)
4	0,2391269
8	0,0739244
12	0,0235840
16	0,0076219

6. SONUÇ VE ÖNERİLER

Bu çalışmada önce Ethereum, Cosmos, Cardano, EOS, Hyperledger blok zincir platformları ve hizmet amaçları kıyaslanmıştır. Blok zincir ağının büyümesi ile saldırgan kişinin düğümleri ele geçirilmesinin zorluğunun üstel olarak arttığı sayısal olarak gösterilmiştir. Aynı zamanda bu çalışmada blok zincir tabanlı platformların altyapılarında kullandıkları konsensüs mekanizmalarının çalıştıkları ağın verimliliğini doğrudan etkilediği de gösterilmiştir. Fikir birliği algoritmalarının ağdaki işlemleri doğrulama yaklaşımlarının o blok zincir ağın performansı için de çok önemli olduğu gösterilmiştir. Blok zincir teknolojisi, akıllı sözleşmeler ile insanlara herhangi bir otoriteye güvenerek ona bağımlı olmadan yepyeni merkeziyetsiz bir mimari ile verileri daha güvenli, son derece şeffaf ve dağıtık olarak saklama imkanını sağlamaktadır. Bu çalışmada Ethereum platformu için Solidity programlama dili ile örnek akıllı sözleşme uygulamaları da geliştirilmiştir. Projelerin kodları ve ekran görüntüleri çalışmanın ekler bölümünde paylaşılmıştır.

Sonraki çalışmalar için öneriler aşağıda verilmiştir. Bitcoin (PoW), Ethereum (PoW ve PoS), EOS (DPoS) ve Tangle IOTA (DAG) tabanlı blok zincirlerini kullanmaktadır. Avalanche, Tezos, Stellar, Openchain, Quorum, Corda R3, Tron, Hedera Hashgraph platformlarını da bulunmaktadır (Kumar, 2021). PoW, PoS, DPoS ve DAG bunların haricinde Practical Byzantine Fault Tolerance (PBFT), Proof of Burn (PoB), Proof of Authority (PoA), Hash Graph, Proof of Authority (PoA), Proof of Weight (PoWeight), Proof of personhood, Proof of Space, Proof of Elapsed time, PAXOS gibi başka doğrulama konsensüs mekanizmaları da mevcuttur (Aggarwal ve Kumar, 2021). Platformların altyapısında kullanılan fikir birliği mekanizmalarından ağa saldırı ve stres algoritmaları geliştirmek için de kumarbazın iflası (Akbarzadeh ve Tekin, 2016) ve sarhoşun yürüyüşü problemini de araştırıp incelenebilir (Ehrhardt, 2013).

KAYNAKLAR

Adewumi, T. P., Liwicki, M. 2020. Inner For-Loop for Speeding Up Blockchain Mining. Open Computer Science, 10(1), (pp. 42-47).

Aggarwal, S., Kumar, N., 2021. Cryptographic Consensus Mechanisms. Advances in Computers (pp. 211-226). Elsevier.

Akbarzadeh, N., Tekin, C., 2016. Gambler's Ruin Bandit Problem. 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton) (pp. 1236-1243). IEEE.

Alodat, Z., Ali, M., Khan, S. U., 2018. Mitigation and Improving SHA-1 Standard Using Collision Detection Approach. International Conference on Frontiers of Information Technology (FIT) (pp. 333-338). IEEE.

Aung, Y. N., Tantidham, T., 2019. Ethereum Based Emergency Service for Smart Home System, Smart Contract Implementation. 21st International Conference on Advanced Communication Technology (ICACT) (pp. 147-152). IEEE.

Benhamouda, F., Halevi, S., Halevi, T., 2019. Supporting Private Data on Hyperledger Fabric with Secure Multiparty Computation. IBM Journal of Research and Development, 63(2/3), 3-1.

Bhandary, M., Parmar, M., Ambawade, D., 2020. Securing Logs of a System an IoTA Tangle Use Case. International Conference on Electronics and Sustainable Communication Systems (ICESC) (pp. 697-702). IEEE.

Bitlo, 2018. Kripto paralar, borsa. BHC, Bitcoin Cash ile Bitcoin Arasındaki Fark. Erişim Tarihi: 19.12.2021. https://www.bitlo.com/rehber/bitcoin-cash-nedir

Blockchain, 2022. BTC. Blockchain.com, The Most Trusted Crypto Company. Erişim Tarihi: 02.02.2022. https://www.blockchain.com/btc/block/0

Bowden, R., Keeler, H. P., Krzesinski, A. E., Taylor, P. G., 2018. Block Arrivals in Bitcoin Blockchain. Arxiv Preprint, 1801.07447.

Brownworth, A., 2021. Distributed Blockchain. Erişim Tarihi: 20.11.2021. https://andersbrownworth.com/blockchain/distributed

Buterin, V., 2014. A Next Generation Smart Contract and Decentralized Application Platform. White Paper, 3(37).

Cao, B., Zhang, Z., Feng, D., Zhang, S., Zhang, L., Peng, M., Li, Y., 2020. Performance Analysis and Comparison of PoW, PoS and DAG Based Blockchains. Digital Communications and Networks, 6(4), (pp. 480-485).

Chen, Y., Liu, F., 2021. Improvement of DPoS Consensus Mechanism in Collaborative Governance of Network Public Opinion. 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE) (pp. 483-488). IEEE.

Dernayka, I., Chehab, A., 2021. Blockchain Development Platforms, Performance Comparison. 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS) (pp. 1-6). IEEE.

Doğa, Ö., 2020, Blokzincir Mimarisi ve Merkezi Olmayan Uygulamalar, 2.Baskı. Pusula.

Dong, S., Yang, H., Yuan, J., Jiao, L., Yu, A., Zhang, J., 2020. Blockchain Based Cross Domain Authentication Strategy for Trusted Access to Mobile Devices in the IoT. International Wireless Communications and Mobile Computing (IWCMC) (pp. 1610-1612). IEEE.

Dragon, N., 2021. Random Generator, Bitcoin Address, Private Key, Public Key. Erişim Tarihi: 28.10.2021. https://www.ntcfinans.com.tr/visual-btc-generator.asp

Edureka, 2021. Guide to Blockchain Technology. Erişim Tarihi: 03.09.2021. https://www.edureka.co/blog/blockchain-tutorial/

Ehrhardt, G., 2013. The Not-So-Random Drunkard's Walk. Journal of Statistics Education, 21(2).

Ehmke, C., Wessling, F., Friedrich, C. M. 2018. Proof-of-Property a Lightweight and Scalable Blockchain Protocol. In Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (pp. 48-51).

Ethereum, F.,2021. A Country's Worth of Power, No More! Erişim Tarihi: 28.08.2021 https://blog.ethereum.org/2021/05/18/country-power-no-more/

Feller, W. An Introduction to Probability Theory and Its Applications. 1957, Wiley.

Ganache, 2021. Truffle Suite, A Tool for Creating a Local Blockchain for Fast Ethereum Development. Erişim Tarihi: 01.10.2021. https://github.com/trufflesuite/ganache

Guo, F., Xiao, X., Hecker, A., Dustdar, S., 2020. Characterizing IOTA Tangle with Empirical Data. Globecom. IEEE Global Communications Conference (pp. 1-6). IEEE.

Harrison, W. L., Procter, A. M., Allwein, G., 2016. Model Driven Design Synthesis of SHA-256 Cryptographic Hash Function in Rewire. International Symposium on Rapid System Prototyping (RSP) (pp. 1-7). IEEE.

Imtiaz, M. A., Starobinski, D., Trachtenberg, A. 2020. Characterizing Orphan Transactions in the Bitcoin Network. IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (pp. 1-9). IEEE.

Jiang, Y., Lian, Z., 2019. High Performance and Scalable Byzantine Fault Tolerance. In 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) (pp. 1195-1202). IEEE.

Košťál, K., Krupa, T., Gembec, M., Vereš, I., Ries, M., Kotuliak, I., 2018. On transition between PoW and PoS. International Symposium ELMAR (pp. 207-210). IEEE.

Kumar, P., 2021. List of Blockchain Platforms. LeewayHertz, Software Development Company. Erişim Tarihi: 15.09.2021. https://www.leewayhertz.com/top-blockchain-platforms

Kuo, T. T., Zavaleta Rojas, H., Ohno-Machado, L. 2019. Comparison of Blockchain Platforms: a Systematic Review and Healthcare Examples. Journal of the American Medical Informatics Association, 26(5), (pp. 462-478).

Meynkhard, A., 2019. Fair Market Value of Bitcoin Halving Effect. Investment Management and Financial Innovations, 16(4), (pp. 72-85).

Mishra, R. A., Kalla, A., Singh, N. A., Liyanage, M., 2020. Implementation and Analysis of Blockchain Based Dapp for Secure Sharing of Students' Credentials. IEEE 17th Annual Consumer Communications Networking Conference (CCNC) (pp. 1-2). IEEE.

Moralis, A., 2022. Importance of Web3.js, An Easy Explanation. Erişim Tarihi: 03.01.2022. https://academy.moralis.io/blog/importance-of-web3-js

Morin, A., Vasek, M., Moore, T., 2021. Detecting Text Reuse in Cryptocurrency Whitepapers. IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (pp. 1-5). IEEE.

Myung, S., Lee, J. H., 2020. Ethereum Smart Contract Based Automated Power Trading Algorithm in a Microgrid Environment. The Journal of Supercomputing, 76(7) (pp. 4904-4914).

Nakamoto, S., 2008. Bitcoin, A Peer to Peer Electronic Cash System. Decentralized Business Review, 21260.

Nguyen, T. S. L., Jourjon, G., Potop-Butucaru, M., Thai, K. L., 2019. Impact of Network Delays on Hyperledger Fabric. Infocom IEEE Conference on Computer Communications Workshops (pp. 222-227). IEEE.

O3schools, 2021. Ultimate Guide to Segregated Witness, SegWit, Lightning Network. Erişim Tarihi: 14.10.2021. https://o3schools.com/guide-to-segregated-witness/

Patil, P., Sangeetha, M. 2021. Blockchain Based Double Spending Prevention for Invoice Financing. Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET) (pp. 41-43). IEEE.

Sagirlar, G., Carminati, B., Ferrari, E., Sheehan, J. D., Ragnoli, E., 2018. Hybrid Iot, Hybrid Blockchain Architecture for Internet of Things Pow Sub Blockchains.bIEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (Cpscom) and IEEE Smart Data (SmartData) (pp. 1007-1016). IEEE.

Sayeed, S., Maron G., H., 2019. Assessing Blockchain Consensus and Security Mechanisms Against the 51% attack. Applied Sciences, 9(9), 1788.

Seijas, P. L., Thompson, S., McAdams, D., 2016. Scripting Smart Contracts for Distributed Ledger Technology. Cryptology Eprint Archive.

Supreet, Y., Vasudev, P., Pavitra, H., Naravani, M., Narayan, D. G., 2020. Performance Evaluation of Consensus Algorithms in Private Blockchain Networks. International Conference on Advances in Computing, Communication Materials (ICACCM) (pp. 449-453). IEEE.

Tonev, I., 2020. Energy Trading Web Platform Based on the Ethereum Smart Contracts and Blockchain. 12th Electrical Engineering Faculty Conference (Bulef) (pp. 1-4). IEEE.

Truffle S., 2021. Sweet Tools for Smart Contracts, Truffle Suite. Erişim Tarihi: 01.10.2021. https://trufflesuite.com/

Usta A., Doğantekin S., 2017. Blockchain 101. Erişim Tarihi: 02.10.2021. https://bctr.org/dokumanlar/Blockchain101v2r2.pdf

Veness, C., 2017. Movable type scripts. Information Management. Erişim Tarihi: 10.12.2021. https://www.movable-type.co.uk/scripts/sha256.html

Vujičić, D., Jagodić, D., Ranđić, S., 2018. Blockchain Technology, Bitcoin and Ethereum, A Brief Overview. 17th International Symposium Infoteh Jahorina (Infoteh) (pp. 1-6). IEEE.

Waldman, J., 2018. Blockchain Fundamentals. Microsoft Docs. Developer Tools, Technical and Coding Documentation. Erişim Tarihi: 07.10.2021. https://docs.microsoft.com/en-us/archive/msdn-magazine/2018/march/blockchain-blockchain-fundamentals

Wang, T., Wang, Q., Shen, Z., Jia, Z., Shao, Z., 2020. Understanding Intrinsic Characteristics and System Implications of DAG Based Blockchain. IEEE International Conference on Embedded Software and Systems (ICESS) (pp. 1-6). IEEE.

Weimeng L., 2019. Understanding How Blockchain Works. NDC Blog. Erişim Tarihi: 11.11.2021. https://blog.ndcconferences.com/understanding-blockchain/

Wikipedia, 2002. Poisson Distribution. The Free Encyclopedia. Erişim Tarihi: 02.02.2022. https://en.wikipedia.org/wiki/Poisson_distribution

Worley, C., Skjellum, A., 2018. Blockchain Tradeoffs and Challenges for Current and Emerging Applications, Generalization, Fragmentation, Sidechains, and Scalability. IEEE International Conference on Internet of Things (Ithings) and IEEE Green Computing and Communications (Greencom) and IEEE Cyber, Physical and Social Computing (Cpscom) and IEEE Smart Data (SmartData) (pp. 1582-1587). IEEE.

Xu, G., Liu, Y., Khan, P. W., 2019. Improvement of the DPoS Consensus Mechanism in Blockchain Based on Vague Sets. IEEE Transactions on Industrial Informatics, 16(6), (pp. 4252-4259).

Yaga, D., Mell, P., Roby, N., Scarfone, K., 2019. Blockchain Technology Overview. Preprint Arxiv, 1906.11078.

Yang, X., Chen, Y., Chen, X., 2019. Effective Scheme Against 51% Attack on Proof of Work Blockchain with History Weighted Information. IEEE International Conference on Blockchain (pp. 261-265). IEEE.

Yermack, D., 2019. Blockchain Technology's Potential in the Financial System. In Proceedings of the 2019 Financial Market's Conference.

Zhao, L., Yu, J., 2019. Evaluating DAG Based Blockchains for IoT. 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications 13th IEEE International Conference on Big Data Science and Engineering (Trustcom, Bigdatase) (pp. 507-513). IEEE.

EKLER

EK A. Kodlar

EK A. Kodlar

Blok zinciri ağ güvenligi saldırı olasılıkları analizi

BlokZinciriAgGuvenligi. Java

```
import java.text.DecimalFormat;
import static java.lang.Math.*;

public class BlokZinciriAgGuvenligi {

   static double saldirininBasariOlasiligi(double q, int z) {
      double p = 1.0 - q;
      double lambda = z * (q / p);
      double sum = 1.0;

      for (int k = 0; k <= z; k++) {
         double poisson = exp(-lambda);
         for (int i = 1; i <= k; i++)
            poisson *= lambda / i;
         sum -= poisson * (1 - pow(q / p, z - k));
      }
      return sum;
   }

   public static void main(String[] args) {
      int num = 12;
      int z;
      double P;
      double q = 0.1;
      System.out.println("q= " + q);

      for (z = 0; z <= num; z++) {
         P = saldirininBasariOlasiligi(q, z);
         System.out.println("z= " + z + "\t\tp= " + new
DecimalFormat("#.#######").format(P));
      }

      System.out.println("--------------------------------------------------");

      int zTimes4;
      q = 0.3;
      System.out.println("q= " + q);

      for (z = 0; z <= num; z++) {
         zTimes4 = z * 4;
         P = saldirininBasariOlasiligi(q, zTimes4);
         System.out.println("z= " + zTimes4 + "\t\tP= " + new
DecimalFormat("#.#######").format(P));
```

```
    }
  }
}
```

Akıllı sözleşme terminalden çalışan uygulamanın kodları

Migrations.sol

```
pragma solidity>=0.4.22 <0.9.0;

// Sözleşmenin adı
contract Migrations {

// Adresin sahibi
 address public owner = msg.sender;
 uint public last_completed_migration;

modifier restricted() {
require(

//İletilmek istenen tüm verileri msg içermektedir.
msg.sender == owner,
" Bu işlev, sözleşmenin sahibiyle sınırlıdır."
  );
// Kontrol metotlarını çağırır.
  _;
 }

// Kontrol metodu
function setCompleted(uint completed) public restricted {
  last_completed_migration = completed;
 }
}
```

AkilliSozlesmeBlokZinciri.sol

```
pragma solidity >=0.4.22 <0.9.0;

contract AkilliSozlesmeBlokZinciri {
  string selamMesaji = "Blockchain Smart Contract, Blok Zinciri Teknolojisi";

  function selamla() public view returns (string memory) {
    return selamMesaji;
  }

  //  Mustafa Cem => 61
  //  Mimar Aslan => 35
  mapping(string => int256) numaralar;
```

```
    function numaraEkle(string memory adi, int256 numara) public {
        if (numaralar[adi] > 0) {
            revert();
        } else {
            numaralar[adi] = numara;
        }
    }

    function numaraYaz(string memory adi) public view returns (int256) {
        return numaralar[adi];
    }
}
```

1_initial_migration.js

```
const Migrations = artifacts.require("Migrations");
const AkilliSozlesmeBlokZinciri = artifacts.require("AkilliSozlesmeBlokZinciri");

module.exports = function (deployer) {
  deployer.deploy(Migrations);
  deployer.deploy(AkilliSozlesmeBlokZinciri);
};
```

2_deploy_contracts.js

```
const Migrations = artifacts.require("Migrations");
const AkilliSozlesmeBlokZinciri = artifacts.require("AkilliSozlesmeBlokZinciri");

module.exports = function(deployer) {
  deployer.deploy(Migrations);
  deployer.deploy(AkilliSozlesmeBlokZinciri);
};
```

truffle-config.js

```
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 8545,
      network_id: "*",
    },
  },
  mocha: {
},
  compilers: {
```

```
  solc: {
   version: "0.8.10",
  }
 },
};
```

Akıllı sözleşme ile web ara yüzünden çalışan uygulamanın kodları

package.json

```
{
 "name": "app",
 "version": "1.0.0",
 "description": "",
 "private": true,
 "scripts": {
  "build": "webpack",
  "dev": "webpack-dev-server"
 },
 "devDependencies": {
  "copy-webpack-plugin": "^5.0.5",
  "webpack": "^4.41.2",
  "webpack-cli": "^3.3.10",
  "webpack-dev-server": "^3.9.0"
 },
 "dependencies": {
  "web3": "^1.2.4"
 }
}
```

webpack.config.js

```
const path = require("path");
const CopyWebpackPlugin = require("copy-webpack-plugin");

module.exports = {
 mode: 'development',
 entry: "./src/index.js",
 output: {
  filename: "index.js",
  path: path.resolve(__dirname, "dist"),
 },
 plugins: [
  new CopyWebpackPlugin([{ from: "./src/index.html", to: "index.html" }]),
 ],
 devServer: { contentBase: path.join(__dirname, "dist"), compress: true },
};
```

Migrations.sol

```
pragma solidity >=0.4.22 <0.9.0;

contract Migrations {
  address public owner;
  uint public last_completed_migration;

  constructor() public {
    owner = msg.sender;
  }

  modifier restricted() {
    if (msg.sender == owner) _;
  }

  function setCompleted(uint completed) public restricted {
    last_completed_migration = completed;
  }
}
```

AslanCoin.sol

```
pragma solidity >=0.4.22 <0.9.0;

import "./ConvertLib.sol";

contract AslanCoin {
  mapping(address => uint256) balances;

  event Transfer(address indexed _from, address indexed _to, uint256 _value);

  constructor() public {
    balances[msg.sender] = 50000;
  }

  function sendCoin(address receiver, uint256 amount) public returns (bool
sufficient){
    if (balances[msg.sender] < amount) return false;
    balances[msg.sender] -= amount;
    balances[receiver] += amount;
    emit Transfer(msg.sender, receiver, amount);
    return true;
  }

  function getBalanceInEth(address addr) public view returns (uint256) {
    return ConvertLib.convert(getBalance(addr), 2);
  }
```

```
    function getBalance(address addr) public view returns (uint256) {
        return balances[addr];
    }
}
```

ConvertLib.sol

```
pragma solidity >=0.4.22 <0.9.0;

library ConvertLib {
    function convert(uint amount,uint conversionRate) public pure returns (uint
convertedAmount) {
        return amount * conversionRate;
    }
}
```

1_initial_migration.js

```
const Migrations = artifacts.require("Migrations");

module.exports = function(deployer) {
  deployer.deploy(Migrations);
};
```

2_deploy_contracts.js

```
const ConvertLib = artifacts.require("ConvertLib");
const AslanCoin = artifacts.require("AslanCoin");

module.exports = function(deployer) {
  deployer.deploy(ConvertLib);
  deployer.link(ConvertLib, AslanCoin);
  deployer.deploy(AslanCoin);
};
```

truffle-config.js

```
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 9545,
      network_id: "*",
    },
  },
  mocha: {
  },
  compilers: {
    solc: {
```

```
  }
 }
}
```

index.js

```
import Web3 from "web3";
import AslanCoinArtifact from "../../build/contracts/AslanCoin.json";

const App = {
 web3: null,
 account: null,
 meta: null,

 start: async function() {
  const { web3 } = this;

  try {
   const networkId = await web3.eth.net.getId();
   const deployedNetwork = AslanCoinArtifact.networks[networkId];
   this.meta = new web3.eth.Contract(
    AslanCoinArtifact.abi,
    deployedNetwork.address,
   );

   const accounts = await web3.eth.getAccounts();
   this.account = accounts[0];

   this.refreshBalance();
  } catch (error) {
   console.error("Sözleşmeye veya zincire bağlanılamadı.");
  }
 },

 refreshBalance: async function() {
  const { getBalance } = this.meta.methods;
  const balance = await getBalance(this.account).call();

  const balanceElement = document.getElementsByClassName("balance")[0];
  balanceElement.innerHTML = balance;
 },

 sendCoin: async function() {
  const amount = parseInt(document.getElementById("amount").value);
  const receiver = document.getElementById("receiver").value;

  this.setStatus("İşlem başlatılıyor. Lütfen bekleyiniz.");

  const { sendCoin } = this.meta.methods;
```

```
    await sendCoin(receiver, amount).send({ from: this.account });

    this.setStatus("İşlem başarılı olarak tamamlandı.");
    this.refreshBalance();
  },

  setStatus: function(message) {
    const status = document.getElementById("status");
    status.innerHTML = message;
  },
};

window.App = App;

window.addEventListener("load", function() {
  if (window.ethereum) {
    App.web3 = new Web3(window.ethereum);
    window.ethereum.enable();
  } else {
    console.warn(
      "Web3 algilanmadi.",
    );
    App.web3 = new Web3(
      new Web3.providers.HttpProvider("http://127.0.0.1:9545"),
    );
  }

  App.start();

});
```

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>ASLAN Coin | Blok Zinciri Web Demo Projesi</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.c
ss" rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW
3" crossorigin="anonymous">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.
min.js" integrity="sha384-
ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+
1p" crossorigin="anonymous"></script>
  </head>
```

```
 <body class="bg-light">
  <div class="container">
   <main>

    <div class="py-5 text-center">
     <img
src="https://upload.wikimedia.org/wikipedia/commons/4/44/%C4%B0stanbul_Ticaret_%C3%9Cniversitesi_Logo.png" width="220" class="rounded float-start" alt="ITUCU">
     <img src="https://www.linkpicture.com/q/AslanCoin.png" width="150" class="rounded float-end" alt="Aslan Coin">
     <h2>Blok Zinciri Akıllı Sözleşme ile <br>Kripto Para Transferi</h2>
     <p class="py-2 lead">ASLAN Coin - DApps Web3.0 Demo Projesi</p>
    </div>

    <div class="row g-12">
     <div class="col-md-4 col-lg-12 order-md-last">
      <h4 class="d-flex justify-content-between align-items-center mb-3">
       <span >Kripto Para Transfer</span>
       <p>Cüzdanınızdaki toplam ASLAN Coin :  <span class="alert alert-warning ">      <strong class="balance">yükleniyor...</strong>
        </p>
       </span>
      </h4>

       <h4 class="mb-3" id="status">Gönderme Platformu</h4>
       <div class="row gy-3">
        <div class="col-md-6">
         <label for="amount" class="form-label">Gönderilecek Tutar</label>
         <input type="text" class="form-control"
             id="amount" >
        </div>

        <div class="col-md-6">
         <label for="receiver" class="form-label">Gönderilecek Cüzdan Adresi</label>
          <input type="text" class="form-control" id="receiver" >
        </div>
       </div>

       <hr class="my-4">
       <button class="w-100 btn-lg btn btn-warning" type="submit"
           onclick="App.sendCoin()">Onayla</button>
      </form>
     </div>
    </div>
   </main>
  </body>
```

```
    <footer class="my-5 pt-4 text-muted text-center text-small">
      <p class="mb-1">© ASLAN Coin, Mimar ASLAN'ın yüksek lisans bitirme tezi
için hazırlanmış bir demo projesidir.</p>
      <ul class="list-inline">
        <li class="list-inline-item"><a
href="https://www.linkedin.com/in/mimaraslan/"
target="_blank">İletişim</a></li>
        <li class="list-inline-item"><a
href="https://www.linkedin.com/in/mimaraslan/"
target="_blank">Destek</a></li>
      </ul>
    </footer>

  </div>
  <script src="index.js"></script>
</html>
```

ÖZGEÇMİŞ

Adı Soyadı : Mimar ASLAN

Eğitim Durumu

Lise : Prof. Faik Sömer Lisesi, 2000

Lisans : Kırgızistan Türkiye Manas Üniversitesi, 2006
Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü

Yayınları

Aslan, M., 2010. Java Server Pages (JSP). Umuttepe Yayınları, 606, Kocaeli.

Aslan, M., 2011. Java Server Faces (JSF). Umuttepe Yayınları, 624, Kocaeli.

Aslan, M., 2012. Mobile Andorid Programlama. Umuttepe Yayınları, 679, Kocaeli.

Aslan, M., 2013. Mobile iOS Objective-C Programlama Dili. Umuttepe Yayınları, 344, Kocaeli.

Aslan, M., 2014. JPA ve Hibernate ile ORM Kitabı. Umuttepe Yayınları. 535, Kocaeli.

Aslan, M., 2015. Spring Framework Kitabı. Level Yayınları, 468, İstanbul.