
GAZİ ÜNİVERSİTESİ
Gazi
1926

ÖZEL BLOK ZİNCİRİ AĞLARDA ADLİ BİLİŞİM İNCELEMESİ VE KAYNAK KODA DÖNÜŞTÜREN ARAÇLARIN SEÇİMİ İÇİN DİNAMİK BİR UZMAN SİSTEM TASARIMI

Kevser AÇIKALIN

DOKTORA TEZİ
ADLİ BİLİŞİM ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

OCAK 2024

Kevser AÇIKALIN tarafından hazırlanan "ÖZEL BLOK ZİNCİRİ AĞLARDA ADLİ BİLİŞİM İNCELEMESİ VE KAYNAK KODA DÖNÜŞTÜREN ARAÇLARIN SEÇİMİ İÇİN DİNAMİK BİR UZMAN SİSTEM TASARIMI" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ / OY ÇOKLUĞU ile Gazi Üniversitesi Adli Bilişim Ana Bilim Dalında DOKTORA TEZİ olarak kabul edilmiştir.

Danışman: Prof. İsmail ŞAHİN

Endüstriyel Tasarım Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum/onaylamıyorum. ...……..

Başkan: Doç. Dr. M. Hanefi CALP

Yönetim Bilişim Sistemleri Ana Bilim Dalı, Hacı Bayram Veli Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum/onaylamıyorum. ……..…….

Üye: Doç. Dr. İ. Alper DOĞRU

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum/onaylamıyorum. ...……...

Üye: Doç. Dr. Ümit ATİLA

Bilgisayar Mühendisliği Ana Bilim Dalı, Gazi Üniversites

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum/onaylamıyorum. ...……

Üye: Doç. Dr. Utku KÖSE

Bilgisayar Bilimleri Ana Bilim Dalı, Süleyman Demirel Üniversitesi

Bu tezin, kapsam ve kalite olarak Doktora Tezi olduğunu onaylıyorum/onaylamıyorum. ...……

Tez Savunma Tarihi: 12/01/2024

Jüri tarafından kabul edilen bu tezin Doktora Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

……..…….

Prof. Dr. Aslıhan TÜFEKCİ

Bilişim Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Bilişim Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Kevser AÇIKALIN

12/01/2024

ÖZEL BLOK ZİNCİRİ AĞLARDA ADLİ BİLİŞİM İNCELEMESİ VE KAYNAK KODA DÖNÜŞTÜREN ARAÇLARIN SEÇİMİ İÇİN DİNAMİK BİR UZMAN SİSTEM TASARIMI

(Doktora Tezi)

Kevser AÇIKALIN

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

Ocak 2024

ÖZET

Blok zinciri alt yapısında gelişen akıllı sözleşmelerin çok farklı alanlarda karşımıza çıkması ile denetim mekanizmalarına da ihtiyaç olacağı açıktır. Ethereum açık ağında bulunan akıllı sözleşmeleri bayt koddan kaynak koda dönüştüren, hem akademik hem de ticari alanda araçlar bazı kısıtlamaları olmakla birlikte mevcuttur ancak özel ağlara özgü literatürde bir çalışmaya rastlanmamıştır. Bu çalışmada özel blok zincirlerde akıllı sözleşmelerin bayt koduna erişim ve bayt koddan kaynak koda çevirmek için izlenmesi gereken prosedürler çıkartılmıştır. Buna ek olarak, bir denetim sırasında açık ağlarda bayt koddan kaynak koda dönüştürmeye ihtiyaç olması halinde mevcut araçlardan hangisinin kullanılması gerektiğine karar verebilmek için gerekli testler uygulanmıştır. Daha önce karşılaştırmaları yapılmayan Elipmoc, Dsol ve Erays araçlarından; Elipmoc'un tespit edebildiği public fonksiyon oranı, hata vermeden dönüştürme yapabildiği sözleşme oranı ve tespit edebildiği evet oranı bakımından daha etkin olduğu görülmüştür. Bu çalışmada ayrıca bir kaynak koda dönüştürme aracı seçilmesine yardımcı dinamik bir uzman sistem tasarımı yapılmıştır. Özetle çalışmanın üç çıktısı bulunmaktadır; ilki akıllı sözleşmelerde bayt koddan kaynak koda dönüşüm yapan araçların seçimini yapacak dinamik uzman sistemin tasarlanmasıdır, ikincisi özel ağlarda yapılacak adli bilişim incelemelerinde izlemesi önerilen prosedür listesidir ve sonuncusu da özel ağlarda yapılacak incelemelerde kullanılabilecek ve zaman kazandıracak uygulamadır.

Bilim Kodu	:	92401
Anahtar Kelimeler	:	Akıllı sözleşmeler denetimi, özel blok zinciri, uzman sistemler
Sayfa Adedi	:	103
Danışman	:	Prof. Dr. İsmail ŞAHİN

A DYNAMIC EXPERT SYSTEM DESIGN FOR SELECTION OF DECOMPILATION TOOLS AND EXAMINE ON PRIVATE BLOCKCHAIN NETWORKS

(Ph. D. Thesis)

Kevser AÇIKALIN

GAZİ UNIVERSITY

INSTITUTE OF INFORMATICS

June 2023

ABSTRACT

The emergence of smart contracts in the blockchain infrastructure across various domains underscores the necessity for robust auditing mechanisms. While academic and commercial tools facilitating the decompilation of smart contracts from bytecode to source code on the Ethereum public network exist, they come with certain limitations. Notably, there is a gap in the literature regarding studies specifically tailored to private networks. This research outlines the recommended procedures for accessing the bytecode of smart contracts on private blockchains and translating them from bytecode to source code. Additionally, tests were conducted to ascertain the appropriate tool for bytecode-to-source transcoding in public networks during an audit. Among Elipmoc, Dsol, and Erays, which have not been compared previously, Elipmoc demonstrated superior effectiveness in detecting public functions, error-free contract decompilation, and event detection proportions. Furthermore, a dynamic expert system was designed in this study to assist in the selection of a decompiler tool. In summary, the research yields three primary outcomes: firstly, the design of a dynamic expert system for selecting tools for byte code to source code conversion in smart contracts; secondly, a recommended procedural list for digital forensics examinations conducted on private networks; and lastly, an application that can be employed in examinations on private networks, saving valuable time.

Science Code	:	92401
Key Words	:	Smart contract audit, private blockchain networks, expert systems
Page Number	:	103
Supervisor	:	Prof. Dr. İsmail ŞAHİN

TEŞEKKÜR

Tez çalışması boyunca desteğini esirgemeyen danışmanım Prof. Dr. İsmail ŞAHİN'e ayrıca tezin nihai halini almasında yönlendirmeleriyle yardımcı olan Doç. Dr. M. Hanefi CALP'e, Doç. Dr. İ. Alper DOĞRU'ya ve Doç. Dr. Utku KÖSE'ye saygılarımı ve minnetlerimi sunuyorum.

Doktora derslerime gelebilmem için Yiğit oğlumla ilgilenen eşime ve elini her zaman üzerimde hissettiğim, sonsuz sevgisiyle beni yüreklendiren kıymetli anneciğime çok teşekkür ediyorum.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklamalar
Abi	Uygulama İkili Arayüzü
API	Uygulama Programlama Arayüzü
AST	Soyut Sözdizimi Ağacı
BTC	Bitcoin
CLI	Komut Satırı Arayüzü
CFG	Kontrol Akışı Grafiği
DApp	Dağıtık uygulamalar
Defi	Dağıtık finans uygulamaları
dBFT	Yetkilendirilmiş Bizans Hata Toleransı
DSR	Tasarım Bilimi Araştırması
EVM	Ethereum Sanal Makinesi
IR	Ara Temsil
KB	Kilo Bayt
NIST	Ulusal Standartlar ve Teknoloji Enstitüsü
pBFT	Bizans Hata Toleransı
PoI	İtibar İspatı
PoS	Hisse İspatı
PoW	İşgücü İspatı
UTX-O	Harcanmamış İşlem Çıktısı

1. GİRİŞ

Kripto paraların popüler hale gelmesi ile dikkatleri üzerine çeken blok zincir altyapısı, çok farklı alanlarda şeffaflık, denetlenebilirlik ve gizliliğin artırılmasına katkıda bulunmasından dolayı kullanılmaktadır. Blok zincir uygulamaları tedarik zincirlerinden, merkeziyetsiz finans uygulamalarına, hak sahipliğinden evrak doğrulama sistemlerine kadar çeşitli alanlarda karşımıza çıkmaktadır [1,2,3]. Özellikle tedarik sistemleri için önerilerde ve iyileştirmelerde bulunan birçok model bulunmaktadır [4]. Kullanım alanlarının çeşitlenmesi, bu yapılarda yapılacak denetim mekanizmalarına ihtiyacın artmasını da beraberinde getirmektedir.

Blok zincirler, genellikle üç farklı izinlendirme türüne göre tasarlanabilir: açık (public), özel (private) ve konsorsiyum (consortium) [5,6]. Açık ağlar, her katılımcının ağdaki bilgilere erişebildiği, mutabakat ve onay işlemlerine katılabildiği yapılar olarak bilinir. Bu tür ağlarda, kurallara uyum sağlayan herkes, mutabakat yapısına dâhil olarak maddi kazanç sağlayabilir. İşlemler, tüm ağ açık olduğu için dışarıdan izlenebilir ve görüntülenebilir. Ancak, dağınık yapının mutabakat mekanizması ile güvenilirliği artarken işlem hızı yavaşlar. Her bir bloğun eklenmesi için gereken hesaplamalar, enerji tüketiminin artmasına neden olur. Bitcoin [7] bu ağlara örnektir, özel ağlara göre daha yavaştır. Açık ağları geliştiren kişilerin dahi zincir içindeki verileri değiştirmeye yetkisi yoktur.

Özel ağlar, blok eklemeye erişimin bir kurum veya kuruluş tarafından koordine edildiği ve ağ erişiminin herkese açık olmadığı veya sınırlı olduğu yapıları ifade eder. Aynı blok zincirini kullanmalarına rağmen, güvenilir bir tarafın mutabakatı sağlaması gerekebilir. Bu tür ağlar, özellikle tek bir kurum içindeki veri tabanı yönetimi ve denetimi gibi konularda kullanılabilir ve açık ağlara göre daha hızlıdır.

Konsorsiyum ağları, kısmen özel ağlar olarak düşünülebilir. Bu tür ağlar, açık ağlardaki gibi herkesin blok ekleyebileceği ya da özel ağlardaki gibi tek bir düğümün blok ekleyebileceği bir yapı yerine, önceden tanımlanmış birkaç noktanın eklemelerde bulunabileceği yapıları temsil eder.

Blok zincirleri bu derece önemli hale getiren en büyük unsurlardan biri akıllı sözleşmelerdir. Akıllı sözleşmeler, bir blok zincir üzerinde tutulan, tarafların önceden belirlediği koşullar meydana geldiğinde otomatik olarak sonucunu çalıştıran programlardır [8]. Blok zincir üzerinde çalıştığı için birbirine güvenmeyen taraflar arasında güvenilirliği sağlayan yapıyı oluşturur.

Ethereum, Turing-complete olan Evm (Ethereum Virtual Machine) sayesinde gelişmiş ve özelleştirilmiş akıllı sözleşmeleri destekleyen ilk açık ağdır [9]. Geliştiriciler, sözleşme taraflarınca mutabık kalınan kuralları Solidity, Vyper vb. diller ile koda dönüştürdükten sonra ilgili dilin derleyicisi ile kodu derler ve ağa yükler [10, 11]. Ağa yüklemek için yapılan işlemin[1] (transaction) giriş verisi (input data) sözleşmenin bayt kodudur. Ethereum'da bulunan akıllı sözleşmelerin kodunun ağa yüklenmesi zorunlu değildir ve geliştiricinin inisiyatifindedir. Ethereum için blokların, işlemlerin ve akıllı sözleşmelerin görüntülenebildiği, geliştiriciler için API'lar sunan gelişmiş özellikli blok zincir gezgini olarak tanımlanan siteye etherscan.io adresiyle ulaşılabilmektedir [12].

Kodu ağa yüklenmeyen akıllı sözleşmelerde inceleme yapılması gerekirse, ağ üzerinde yalnızca bayt koda erişilebildiğinden, kaynak koda dönüştürme araçlarının (decompile) kullanılması gerekmektedir. Kodu uzun ve karmaşık sözleşmeler için tam bir çevirme her zaman mümkün olmasa da gerek literatürde gerekse ticari alanda ihtiyaca yönelik uygulamalar bulunmaktadır. Gigahorse [13], onun gelişmiş versiyonu olan Elipmoc [14], Erays [15] ve kaynak kodu açık olmayan Ethervm [16] Solidity dili ile Ethereum ağına yüklenen bayt koddan kaynak koda dönüştürmek için kullanılan araçlardandır. Ayrıca Jeb [17] ve Trustlook [18] aynı işlevi gören ticari uygulamalardandır.

Bu çalışmada kolluk kuvvetlerinin ve denetim çalışanlarının ihtiyaç halinde kullanacakları aracı seçmelerine yardımcı olmak için, Ethereum açık ağında bayt koddan kaynak koda dönüştüren araçların detaylı analizi yapılmıştır. Analiz için 10.000 adet kodu bilinen akıllı sözleşmenin bayt koddan ilgili araçlar kullanarak dönüşümü yapılmıştır ve ağda var olan kaynak kod ile karşılaştırılmaları ile araçların performanslarını gösteren sonuçlar elde edilmiştir.

[1] Transaction ifadesinin Türkçe karşılığı olarak işlem kelimesi kullanılmıştır ancak transaction bir işlem ya da değer aktarımından daha kapsamlı bir paket olarak düşünülmelidir.

Çalışmanın literatüre yapacağı asıl katkı ise incelenecek ağın özel ağ olması halinde üzerinde bulunan akıllı sözleşmelerin incelenmesinin nasıl yapılması gerektiğini ortaya koyacak olmasıdır. İncelenecek ağın özel/konsorsiyum ağ olması halinde denetimlerde dikkat edilmesi gereken unsurları genel hatlarıyla belirtilen çalışmalar [19] vardır. Ancak farklı özel ağ oluşturma araçları kullanılarak oluşturulan özel ağlarda akıllı sözleşmelerin bayt kodunun elde edilerek kaynak koda dönüşümünü yapacak uygulama ve özel ağ oluşturma araçlarına özgü aramalar yapma konularında çalışma bulunmamaktadır.

Bu tez çalışmasının amacı özetle, özel blok zinciri üzerinde yapılan incelemelerde izlenecek prosedür listesinin çıkarılması, incelemenin etkinliğini artırmak için gerekli uygulamayı geliştirmek ve bayt koddan kaynak koda dönüştüren araç kullanımına gerek olması halinde hangi aracın kullanılması gerektiğinin tespitini yapmaktır.

Tezin sonucunda elde edilen önerilen prosedür listesi ve uygulamanın, kolluk kuvvetlerinin ve denetim birimlerinin yapacağı adli bilişim incelemelerinde zaman tasarrufu sağlayacağı değerlendirilmektedir. Ayrıca yapılacak adli bilişim incelemelerinde, akıllı sözleşmelerde bayt koddan kaynak koda dönüştürme ihtiyacı bulunması halinde var olan araçlardan hangisinin seçilmesi gerektiğini öneren bir uzman sistem tasarlanmıştır. Tez çalışması beş ana bölümden oluşmaktadır. İlk bölümde blok zincirinin temelleri ve bileşenleri; ikinci bölümde Ethereum ve akıllı sözleşmeler ile ilgili genel bilgiler bulunmaktadır. Üçüncü bölümde akıllı sözleşme bayt kodundan kaynak koda dönüştüren araçların analizine yer verilecektir. Dördüncü bölümde özel ağların nasıl oluşturulabileceği ve mevcut altyapıların karşılaştırılması, beşinci bölümde özel ağlarda yapılacak incelemelerde izlenmesi önerilen prosedür listesi, altıncı bölümde var olan çalışmalarla bu tez çalışmasının karşılaştırılmasına ve katkılarına yer verilmiştir. Son olarak uygulama bölümünde prosedür listesini takip ederken yardımcı olacak uygulama ile bayt koddan kaynak koda dönüştüren araçların seçiminde kullanılacak uzman sistem tasarımı bulunacaktır. Uygulama ile bir bilgisayardaki tüm adreslerin tespiti, özel ağ konfigürasyon dosyalarının bulunması, Remix IDE üzerinden oluşturulan akıllı sözleşmelerin listelenmesi ve özel ağlarda işlem bilgisi ile akıllı sözleşme bayt kodundan kaynak kodun elde edilmesi sağlanmaktadır. Tasarlanan uzman sistem ise akıllı sözleşmeler için bayt koddan kaynak koda dönüşüm yapan araçların önerilme sırasını oluşturmaktadır.

Tez çalışmasında bulunan araştırma soruları aşağıdaki gibidir.

Soru 1. Bayt koddan kaynak koda dönüştüren araçlar içinde hangisi daha efektiftir? (Bölüm 4.2.)

Soru 2. Özel blok zincir ağlarda yapılan incelemelerde takip edilmesi gereken adımlar nelerdir? (Bölüm 6.1.)

Soru 3. Özel blok zincir ağlarda yapılan incelemeyi kolaylaştıracak bir araç geliştirilebilir mi? (Bölüm 7.1.)

Soru 4. Bayt koddan kaynak koda dönüştüren araçların seçimi için uzman sistem geliştirilebilir mi? (Bölüm 7.5.)

Çalışmada tasarım bilimi araştırması (design science research (DSR)) yöntemi [20] kullanılmıştır ayrıca bayt koddan kaynak koda dönüştüren araçların seçiminde uzman sistem içinde ileri zincirleme yöntemi kullanılmıştır.

2. BLOK ZİNCİRİNİN BİLEŞENLERİ VE ÖZELLİKLERİ

Blok zinciri (blockchain), veri paketlerinin ve işlemlerin (blok) kriptografik olarak birbirine bağlı şekilde ardışık olarak tutulduğu, güvenilir bir merkezi otoriteye ihtiyaç duymayan, dağıtık ve açık bir veri depolama yöntemidir. Bu metod, dağıtık olarak tutulan verinin tutarlılığını ve şifrelenmiş olmasına rağmen dışarıdan erişilebilirliğini sağlayarak veri doğrulamaya imkân tanır [21].

Bitcoin ile tanınırlığı artan ve o günden beri çok dikkat çeken blok zincirin temelleri aslında 1990'lı yıllara dayanmaktadır [22]. Bitcoin (BTC), "Bitcoin: Uçtan Uca Elektronik Nakit Ödeme Sistemi" adlı yayınla, Satoshi Nakamoto takma adını kullanan bir grup veya akademisyen tarafından tanıtılmış kripto paradır [7]. Bu önerilen sistem, 2009 yılı başından beri açık bir ağ olarak kullanılmaktadır. Bitcoin ile herhangi bir üçüncü güvenilir taraf veya merkezi otorite olmadan güvenli bir şekilde kişiden kişiye para transferi yapmak mümkündür. Yapılan ödeme işleminin değiştirilmesi neredeyse imkânsızdır, bu da satıcıların korunmasını sağlamayı amaçlar. Ayrıca, alıcıların korunması için sistemde emanet mekanizmaları kolaylıkla uygulanabilmektedir.

Bitcoin ile ünlenen ve çok dikkat çeken blok zincirin uygulama alanları kripto paralarla sınırlı değildir. Bu yöntem; tedarik sistemleri, küresel ödeme sistemleri, telif hakkı kayıt sistemleri, bağış toplama ve yönetimi, seçim sistemleri ve dijital kimlik gibi çok farklı alanlarda kullanılabilmektedir [1].

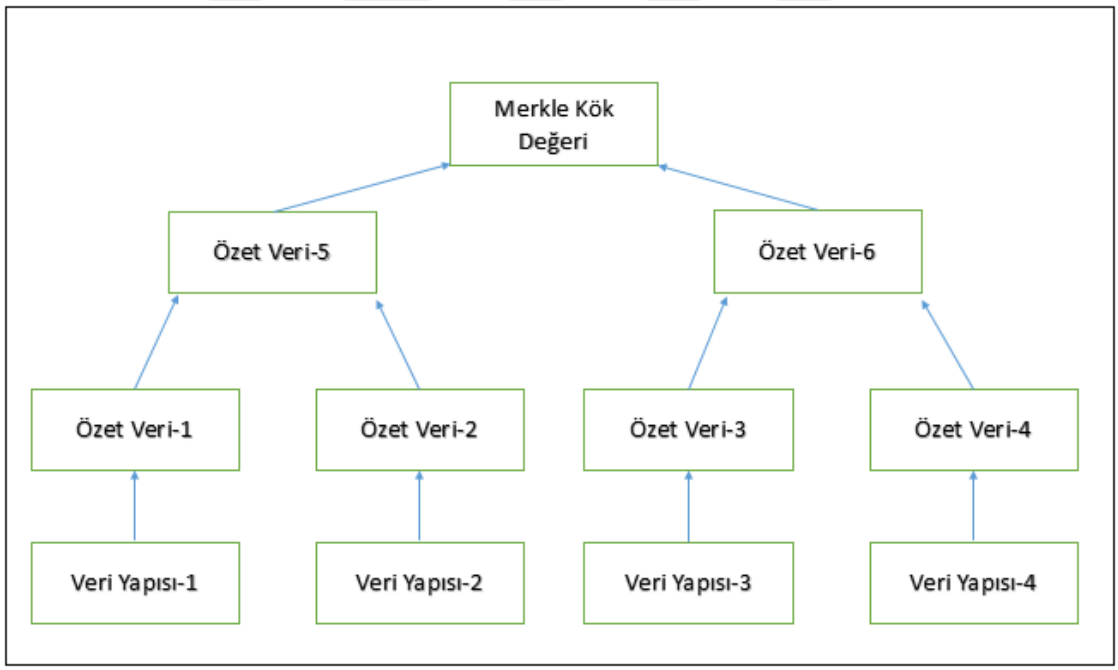
Blok zincirinin nasıl çalıştığının anlaşılabilmesi için özetleme algoritmaları, asimetrik şifreleme, elektronik imza ve eşler arası ağ (peer-to-peer network) kavramlarının bilinmesi gerekir.

2.1. Özetleme Algoritmaları

Özetleme işlemi genellikle bir mesajın veya dosyanın özeti alınarak sabit uzunlukta ve o dosyaya özel bit serisi elde edilmesi olarak tanımlanmaktadır. Mesajda en küçük bir değişiklik olduğunda bile özet değeri değişir. Tek yönlü olan özetleme algoritmalarında özetten mesajı geri getirmek imkânsızdır. Özet değerleri birbiriyle eşleşen iki farklı mesajın

bulunması oldukça zordur [23]. Keccak ve SHA256, blok zincir altyapılarında kullanılan özetleme algoritmalarından ikisidir.

Birden fazla girdinin doğrulanması için bütünü oluşturan elemanların ikili şekilde özeti alınarak tek bir özet değerin elde edildiği Merkle ağacı oluşturulmaktadır. Ralph Merkle tarafından 1979'da geliştirilen ve ismini verdiği bu yapıda, yığınlarca veri hızlı şekilde doğrulanabilir [24]. Veri yığınını oluşturan parçalar ikili bir ağaç yapısı oluşturularak ağacın en alt seviyesine yerleştirilir, daha sonra ikili gruplar halinde özet değerleri alınarak tüm ağacın verilerini ifade eden tek bir özet değeri elde edilir. Bu özet değeri ağacın kök değeridir, ihtiyaç olması halinde tüm ağaçtaki veriler yerine yalnızca bu kök değerin kontrolü ile ağacın değiştirilmediğine dair doğrulama yapılabilir [25]. Örnek Merkle ağacı Şekil 2.1.'de bulunmaktadır.



Şekil 2.1. Merkle Ağacı Yapısı

2.2. Asimetrik Şifreleme

Asimetrik anahtarlı bir algoritma, şifreleme ve şifreyi çözme işlemleri için açık ve özel anahtar çiftini kullanır. Genellikle imza oluşturma verisi özel anahtarla yapılırken imza doğrulama verisi açık anahtardadır. Kullanılan yönteme bağlı olarak özel anahtar ile açık anahtar arasında matematiksel bir ilişki kurularak birlikte oluşturulurlar. Anahtarlardan biri

kullanılarak oluşturulan şifre ancak diğer anahtarın kullanılmasıyla çözülebilir [26]. RSA (Rivest-Shamir-Adleman), DSA (Digital Signature Algorithm) ve Eliptik Eğri algoritmaları, asimetrik şifreleme algoritmaları arasındadır. Blok zincir altyapısında ise Eliptik Eğri algoritması kullanılmaktadır. Eliptik eğriler, RSA'nın sağladığı güvenlik düzeyini daha kısa anahtar uzunluğu ile sağladığı için tercih edilir. Şifreleme süresi de RSA'ya göre daha az olduğundan mobil cihazlar ve akıllı kartlar gibi bellek ve işlemci kaynaklarının sınırlı olduğu yerlerde tercih edilmektedirler [27].

2.3. Elektronik İmza

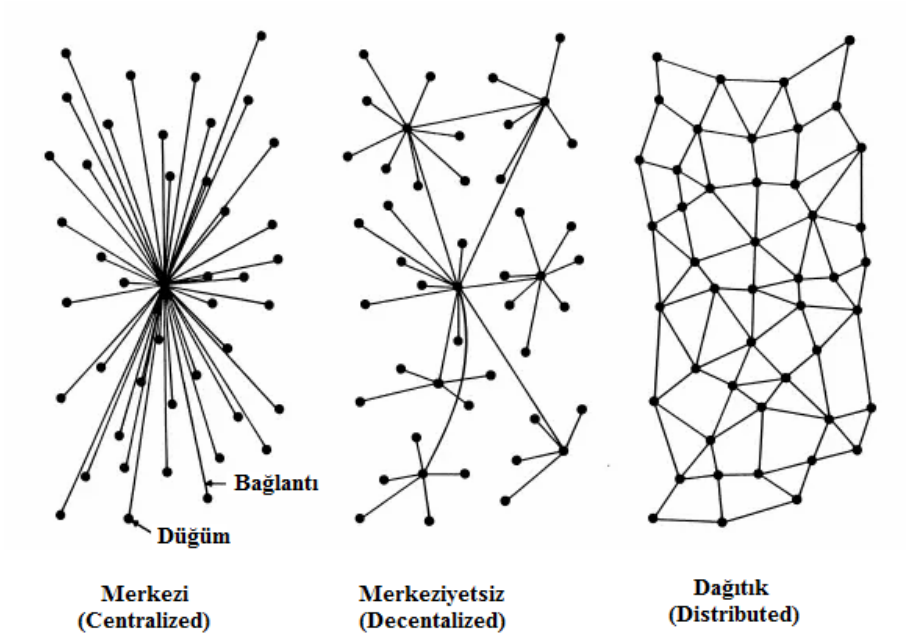
Bir elektronik dokümanı imzalama işlemi gönderici tarafında iki adımda gerçekleşmektedir. İlk adımda, imzalanacak elektronik verinin özet değeri oluşturulur. Ardından elde edilen özet bilgisi açık anahtarlı imza algoritmasıyla imzalanır. Artık oluşan elektronik imza ile şifresiz haldeki elektronik veri karşı tarafa güvenle gönderilmeye hazırdır. Alıcı tarafında imzalı olarak gelen elektronik verinin doğrulanması işlemi iki adımdan oluşur. Birincisi, şifresiz açık şekilde alınan elektronik verinin özet değerinin alıcı tarafında tekrar elde edilmesidir. Ardından teslim alınan elektronik imza, açık anahtar kullanılarak açık anahtarlı imzalama algoritmasıyla deşifre edilirek elektronik verinin özet değerine ulaşılır. İlk adımdaki özet değer ile ikinci adımdaki özet değer eşleşiyorsa teslim alınan elektronik verinin içeriği değiştirilmediği ve imzalanan kişi tarafından gönderildiği anlaşılır [28].

2.4. Eşler Arası Ağ (Peer-to-Peer Network)

Eşler arası (Peer-to-peer/P2P), birden çok bilgisayar arasında veri paylaşımını sağlayan ağ protokolüdür. Katılımcı her bir düğüm, bir veriyi indirmek istediği zaman elinde aradığı veri bulunan diğer katılımcıları bulur ve onlardan veriyi çeker. Karşılığında da elinde bulunan veriyi o veriyi talep eden başka katılımcıların bilgisayarına aktarılması için izin verir. Bu uygulamalara, BitTorrent ve eDonkey örnek verilebilir. Eşler arası ağda da veri blok zincire benzer şekilde dağıtık şekilde tutulur ancak içerik şifrelenmesi ve indirilen verinin istenilen dosya olduğundan emin olunamaması sebebiyle güvenli bir çözüm değildir [25].

Dağıtık tabanlı defter teknolojileri (DLT) bu güven problemine çözüm olarak ortaya çıkmıştır. Ağa veri eklenmesi istendiğinde çalışan mutabakat yapısı ile merkezi bir sisteme ihtiyaç duymadan düğümler güncellemeleri alır ve dağıtık bir şekilde güncel verileri tutmuş

olur [25]. Dağıtık ve merkeziyetsiz yapıların isimleri zaman zaman birbirinin yerine kullanılmaktadır, merkezi, merkeziyetsiz ve dağıtık ağlarda düğümlerin birbiriyle ilişkisi Şekil 2.2.'de bulunmaktadır [29].



Şekil 2.2. Merkezi, merkeziyetsiz ve dağıtık ağ yapıları [29]

2.5. Blok Zincirinin Özellikleri

Bitcoin'in tanıtıldığı yayında, blok zinciri öne çıkaran özellikler kalıcılık, dağıtık mimari, anonimlik ve denetlenebilirlik olarak sıralanmıştır [7]:

Kalıcılık (Persistency): Blok zincirinde bloğa eklenen işlemlerin onaylanması hızlıdır, doğrulanamayan işlemler güvenilir madenciler tarafından reddedilir. Blok zincire ekleme yapıldıktan sonra bir işlemi silmek veya geri almak neredeyse imkânsızdır. Böylece bir birim paranın tekrarlı harcanmasının (double spending) önüne geçilir ve dolandırıcılıklar önlenmiş olur.

Dağıtık mimari (Decentralization): Merkezi geleneksel transfer sistemlerinde, her bir işlemin onaylanması merkezi ve güvenilir bir yapıya ihtiyacı doğurur (banka, noter vs.). Bu sistemlerde maliyet ve performansta istenmeyen durumların oluşması kaçınılmazdır. Blok zincirde üçüncü bir tarafa ihtiyaç bulunmaz. Mutabakat yapıları ile dağıtık mimaride

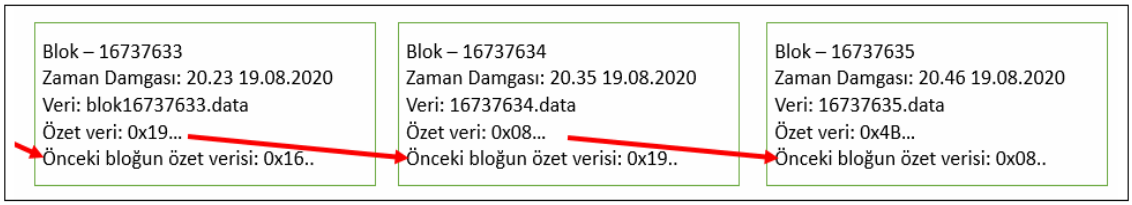
verilerin herkeste aynı veri olacak şekilde tutarlı halde bulunması sağlanır. Mutabakat yapılarına bölüm 2.7.'de yer verilmiştir. Üzerinde mutabık kalınan veriler tek bir merkezde tutulmak yerine dağıtık olarak birden fazla noktada bulunduğundan, tek bir merkezi noktanın kırılganlığına bağlı güvenlik açıklarından etkilenmeyecek niteliktedir.

Anonimlik (Anoniymity): Blok zincirde işlem yapacak kişi, işlemlerini gerçekleştirmek için bir cüzdan adresi oluşturabilir, bunun için kişisel bilgilerini paylaşmak zorunda değildir. Ancak, blok zincir tam anlamıyla gizlilik sağlamaz, çünkü cüzdan adresleri oluşturulurken kişisel bilgilere ihtiyaç duyulur ayrıca bir zincir baştan sona incelendiğinde bir hesabın tüm işlemlerine erişilebilir.

Denetlenebilirlik (Auditability): Bu özellik Bitcoin özelinde açıklanacak olursa, tüm kullanıcıların hesapları harcanmamış işlem çıktısı (Unspent Transaction Output, UTX-O) modeli ile tutulur. Bir işlem, harcanmamış başka bir işlem çıktısına referans göstermek zorundadır. Bloğa eklenen bir işlemin durumu, harcanmamış işlemden harcanan işlem olacak şekilde değiştirilir. Bu sayede işlemler kolaylıkla doğrulanıp takip edilebilir.

2.6. Blok Yapısı

Blok zincir ağında verilerin saklandığı yapılar blok olarak adlandırılır. İlk blok (genesis) oluşturulduktan sonra zaman sıralamasına göre birbiriyle ilişkili bir zincir oluşturulur. Birinci blokta önceki bloğun özet değeri 000..0 olarak tutulur. Bir blokta, kendinden önceki bloğun özet değeri de tutulur. Bu sayede aradaki bir bloğun dışardan müdahale ile bozularak değiştirilmesi durumunda, o bloktan sonraki blokların da değiştirilmesi gerekir. Güvenliği sağlayan mekanizma bu zincir yapısıdır, oluşturulduktan sonra değiştirilemez olması yapıyı güvenilir kılar. Örnek blok yapısı Şekil 2.3.'te bulunmaktadır.



Şekil 2.3. Blok yapısı

2.7. Mutabakat ve İşlem Süreci

Dağıtık bir yapıda, eklenen bloğun içeriğinin ağdaki tüm bilgisayarlar tarafından kabul edilmesi için mutabakat yapısına uygun olarak eklenmesi gerekir. Bu sayede hatalı işlemlerin veya geçerliliği olmayan işlemlerin ağa yazılması önlenir, böylece birbirini tanımayan insanlar o ağa eklenmiş verinin güvenilirliğinden emin olur.

Bitcoin'de de kullanılan İş-İspatı [Proof-of-Work (PoW)] en çok tercih edilen yöntemlerin başında gelir. Bu yöntemle, blok zincire bir bloğun eklenebilmesi için bir matematik problemi ortaya konulmuştur, bu problemi çözmek denemelere dayandığından zaman alır ancak çözümün doğruluğu bir değerin özetini almaya yetecek kadar kısa sürede olmaktadır. Bu matematik problemini çözme süreci madencilik olarak adlandırılır. Bu yapılara özel problem zorluk değerine göre belirlenen karakterde 0 ile başlayan özet değeri bulmaya dayalıdır. Zorluk değeri arttıkça çözümü bulmak için gereken deneme sayısı da giderek artacaktır. Zorluk değerine özgü olması istenen formatta özet değeri oluşturan ilk kişi, blok zincire yeni bloğu eklemeye hak kazanır ve bloğu ekleyen kişilere ödül verilir.

Bir blok eklendikten sonra bir bloğu değiştirmek isteyen kişi hem ilgili bloğu hem de o bloktan sonraki blokları değiştirmek zorundadır, blok zincir güvenliği bu şekilde korunmaktadır. Gelişen teknoloji ile hızlanan donanımların bu sürece etkilerini kontrol edebilmek için zorluk değeri değiştirilerek blok üretiminin belirli bir hızın üstüne çıkması önlenir [7].

Bitcoin özelinde işlem (trancation), ağa yayılan ve bloklar altında toplanan Bitcoin transferleridir. Diğer bir deyişle Bitcoin'in bir yerden (input) diğer bir yere (output) aktarılmasının kaydıdır. Her bir işlem, önceki işlem çıktılarını yeni işlemin girdisi olarak alır ve Bitcoin girdi değerlerinin tümünü yeni çıktılara dönüştürür. Şifrelenmedikleri için takip edilerek listelenmeleri ve bloklardaki tüm işlemlerin görüntülenmesi mümkündür. İşlemler için yeterli onay alındıktan sonra geri döndürülemezdir denilebilir. Standart bir işlemde çıktı olarak adresler gösterilir ve daha sonraki herhangi bir girdinin ödemesi ilgilinin onayını gerektirir.

Tüm işlemler blok zincirde görülebilir ve bir hex editörü ile dışardan bir kişi tarafından görüntülenebilir. Blok zincir görüntüleyici (scan sayfaları) ile işlemler teknik ayrıntıları ile

görüntülenip doğrulanabilir. Blok içindeki işlemlerin Merkle ağaç özeti tutularak değiştirilmemesi sağlanır [30].

Kullanılan diğer mutabakat yöntemleri; Bizans Hata Toleransı [Practical Byzantine Fault Tolerance (pBFT)], Yetkilendirilmiş Bizans Hata Toleransı [Delegated Byzantine Fault Tolerance (dBFT)], Hisse İspatı [Proof-of-Stake (PoS)] ve İtibar İspatı [Proof-of-Importance (PoI)] şeklinde sıralanabilir. Yöntemlerin açıklamalarına kısaca aşağıda yer verilmiştir.

2.7.1. Bizans Hata Toleransı [Practical Byzantine Fault Tolerance (pBFT)]

Bu yöntem ismini Bizans imparatorunun generallerle haberleşme yönteminden almaktadır. İmparatorlar, generallere haber ulaştırırken birden fazla ulak kullanmaktaydı ve generaller de kendi aralarında ulaklar göndererek haberin doğrulanmasına yardımcı olmaktaydı. Böylece ulakların çoğunluğu tarafından getirilen emirin doğru olduğundan emin olunmaktaydı. Dağıtık mimaride ise, ağdaki tüm katılımcılar için bir özel-genel anahtar ikilisi bulunmaktadır. Her katılımcı (makine) kendisine gelen her işlemi kontrol etmekte ve onaylaması durumunda imzaladıktan sonra ağ ile paylaşmaktadır. Bir işlem ağdaki katılımcıların yarısından fazla sayıda makine tarafından onaylandıysa işlem geçerli sayılmaktadır. Bu yöntemin öne çıkan özelliği, her katılımcının onaylama işlemine katkı sağlayabilmesidir. Ağdaki tüm işlemleri doğrulayan makinelerin birbiriyle iletişim halinde olması ve yeni doğrulayıcı makinelerin eklenebilmesi için merkezi bir onay yapısına ihtiyaç vardır. Bu da herkese açık, dağıtık blok zincir mantığıyla uyumlu değildir; bu nedenle açık (public) ağlardan ziyade özel (private) ağlar için uygundur. Hyperledger Fabric altyapısında kullanılabilen yöntemdir [25, s. 101] [31].

2.7.2. Yetkilendirilmiş Bizans Hata Toleransı [Delegated Byzantine Fault Tolerance (dBFT)]

Standart bir BFT'ye benzer olan dBFT, NEO [32] tanıtım yazısında da bulunmaktadır. Bu yöntemde makineler hesaplayıcı ve normal düğümler olarak ikiye ayrılmaktadır. Normal düğümler mutabakat aşamasında görev almaz, ancak hesaplayıcı makineleri seçerek belirlerler. Oylanarak seçilen temsili hesaplayıcı makineler, mutabakat işlemlerini yapar. Rastgele bir hesaplayıcı makine seçilerek tüm işlemleri ağa yayılır. Diğer hesaplayıcı makinelerin en azından %66'sının işlem verilerini doğrulaması durumunda işlemler

onaylanır ve kalıcı olarak blok zincire eklenir.

2.7.3. Hisse İspatı [Proof-of-Stake (PoS)]

Bu yöntemde blok ekleme ve mutabakat süreci, bloğu üreten makinenin ağdaki payı ile ilişkilidir, madenciler sahip olduğu kripto para miktarını ispatlamak durumundadır. Sistemde daha fazla payı olan kişinin ağa saldırma ihtimalinin düşük olduğuna inanılır. Ancak en fazla kişinin baskın olduğu bir ağ pek adil olmayabilecektir [21]. Hangi makinenin payının daha büyük olduğuna ve hangisinin bir sonraki bloğu işleyeceğine karar vermek için farklı yöntemler vardır. Rastgele seçim yönteminde, yeni bloğu ekleyecek makinenin pay oranı ile ilişkili rastgele seçim yapan bir algoritmayla seçilebilir, eğer seçilen makine belirlenen sürede bloğu eklemezse bir sonraki makineye geçilir. Yeni bloğu ekleyecek makine belirlenmez ancak sahip olduğu pay PoW'a benzer şekilde zorluk derecesini değiştirir. Pay sahipliği büyüdükçe zorluk azalır. Bir diğer yöntem, elde tutulan paranın ne kadar süredir tutulduğuyla orantılı olarak avantaj sağlayan seçime dayanır. Daha yüksek miktarda daha uzun süredir tutulan para miktarına sahip olan makinenin bloğu işleme olasılığı daha yüksektir.

Peercoin'de [33], pay sahipliğinin yanında ağdaki işlemlerde kullanılan bir diğer kıstas olarak yaş (age) değeri kullanılmıştır. Başlangıçta makinelerin yaş değeri sıfırdır, zamanla yaş değerleri artar ve pay sahipliğinin yanında yaş değeri de blok eklemede öncelik kazanmasını sağlar. Bu yaklaşımda madencilik (mining) yerini para basma (minting veya forgining) terimine bırakır. PoW ile karşılaştırıldığında PoS, enerji tasarrufu sağlar ve daha hızlıdır ancak madencilik maliyetinin az olması saldırıları arttırabilmektedir. Çoğu kripto para ilk aşamada PoW'u benimser daha sonra PoS'a geçer [21]. Ethereum da Aralık 2022'de Pos'a geçmiştir [34].

2.7.4. İtibar İspatı (Proof-of-Importance (PoI)

PoI, XEM kripto parasının altyapısı olan NEM ağında kullanılmıştır [35]. NEM ağındaki her hesabın XEM bakiyesi iki kısımdan oluşmaktadır: kazanılmış (vested) ve koşullu bakiyeler (unvested). XEM'de bir hesaba para gönderimi olduğunda öncelikle koşullu bakiye hesabında tutulmaktadır, her 1440 blokta bir koşullu bakiyenin onda biri kazanılmış bakiye kısmına aktarılır. Bir hesap XEM gönderdiğinde, XEM hem kazanılmış hem de

koşullu bakiye hesaplarından alınır böylece hesaplar arasındaki oran sürdürülür.

Bir hesabın itibarının hesabı (Importance Calculation) yapılabilmesi için en az 10.000 XEM kazanılmış bakiyeye sahip olması gerekmektedir. İtibar hesabı yapılırken, NEM ağı tarafından belirlenen sabitler olan, kazanılmış XEM bakiyesi ve hesabın lokasyonunu temel alan ağırlık faktörü kullanılmaktadır. Bu bölümde blok zinciri bileşenleri özellikleri ile ele alınmıştır, sonraki bölümü Ethereum ve akıllı sözleşmeler oluşturmaktadır.

Bunların dışında Faaliyet İspatı (Proof-of-Activitiy, PoAc), Yakım İspatı (Proof-of-Born, PoB), Geçen Sürenin İspatı (Proof-of-Elapsed-Time, PoET) ve Kapasitenin İspatı (Proof-of-Capacity, PoC) gibi çeşitli mütabakat yöntemleri vardır [36].

Bunlar sırayla şöyledir; PoAc, PoW ve PoS'un karışımı gibi düşünülebilir, başlangıçta madencilik sürecine PoW ile başlanır ve onun avantajları olan %51 saldırılarını engelleyebilmiş olur. Bir blok çıkarıldıktan sonra bloğun ağın geri kalanı ile paylaşılması için PoS mekanizması kullanılmaya başlar. Toplam bekleme süresine göre rastgele seçilen bir grup doğrulayıcı yeni bloğun eklenmesinden sorumlu olur. Bu şekilde ödüller madenciler ile doğrulayıcılar arasında paylaştırılır [37]. Daha az veri kullanması ve daha fazla güvenlik sağlaması nedeniyle yapay zeka uygulamaları için kullanışlıdır.

PoB, PoW'un çok fazla enerji tüketimine ihtiyaç duyması problemine yönelik ortaya çıkarılan bir çözümdür. Katılımcılar ellerinde bulunan ve o ağa özgü olan coinleri yakarak blok oluşturabilir. Bu aynı zamanda ağdaki coini azaltarak değerini yükseltir [36].

PoET, katılımcıları doğrulama sürecine dahil etmek yerine yeni bloklar oluşturabilecek bir lider düğüm tanımlar. Lider düğümü seçmeyi her düğümle rastgele eşleştirdiği zamanlayıcıları kullanarak yapar, sürekli olarak yeni lider düğümler seçilir. Bu düğüm yeni blokları oluşturur ve imzasını tüm düğümlere iletir.

PoC'ta, katılımcılar bilgisayarlarının hesaplama gücü yerine sahip oldukları depolama alanlarını kullanarak madencilik yaparlar [36, 38].

3. ETHEREUM ve AKILLI SÖZLEŞMELER

Ethereum, Turing-complete olan Ethereum Virtual Machine (EVM)sayesinde gelişmiş ve özelleştirilmiş akıllı sözleşmeleri destekleyen ilk açık ağdır [9]. Evm, akıllı sözleşmeler için bir çalışma ortamıdır (runtime environment), Solidity [10] veya Vyper [11] gibi birçok yüksek seviyeli programlama dilleri, Ethereum ağında akıllı sözleşmeleri yazmak için kullanılabilmektedir. Ethereum akıllı sözleşmeler için en çok kullanılan geliştirme platformudur [39] ve dijital sahiplik yönetimi, kitle fonlaması, şans oyunları gibi çeşitli merkezi olmayan uygulamaları (DApps) tasarlamak için kullanılabilmektedir. DApps, herhangi bir sunucuya ihtiyaç duymadan çalışabilen uygulamalardır. Ethereum'un dışında Algorand [40], Polkadot [41], Solana [42], Binance Smart Chain [43], Avalanche [44] ve EOSIO [45] gibi öne çıkmış farklı altyapılar mevcuttur ve alt katman (layer 1) olarak adlandırılmaktadır.

Ethereum üzerinde hem kripto para alışverişi hem de akıllı sözleşme işlemleri yapılabilmektedir. Akıllı sözleşme işlemlerinde var olan bir fonksiyon çağırabilir ya da yazılan bir akıllı sözleşme ağda bayt kod (bytecode) olarak çalıştırılabilmektedir (deploy). Ağda çalıştırılan bir akıllı sözleşmede görülen giriş verisi (input data), yazılan sözleşmeye ait bayt koddur. Gerek ağa aktarma işleminde gerekse var olan fonksiyonun çağrılması sırasında ihtiyaç duyulan kaynak (işlemci ve hafıza) için madencilere belirli bir işlem ücreti (Gas) Ether cinsinden ödenmektedir. İşlem ücreti, sistemin devamlılığı için madencileri teşvik etmesi bakımından ve ağı meşgul etmek amacıyla yapılan olası saldırılarda kodda bulunan döngülerin artırılması ve buna bağlı olarak Gas miktarının artmasıyla ağa saldırıları engellemesi bakımından önemlidir [46].

Ethereum'da işlem ücreti Eş. 3.1'de bulunan formül ile hesaplanmaktadır. İşlem ücreti ve Gas ücreti Ether cinsindendir. Gas ise ağa gönderilen işlemin madencilere ne kadara mal olacağını gösteren bir miktardır. Madenciler daha yüksek işlem ücreti olan işlemleri bloğa eklemeyi tercih etmektedirler. Bir akıllı sözleşmeyi ağda çalıştırmak isteyen kişinin, ödenen ortalama ücret üzerinden bir tahmin yaparak bu miktarı belirlemesi gerekmektedir. Bu değerler sürekli güncelleneceği [47] için bu tahmini ve hesaplamaları yapan API'lar kullanılmaktadır [48]. Gas limit ise, yazılan sözleşmede dinamik olarak büyüyen dizilerden veya döngü sayılarından dolayı işlem ücretinin belirli bir seviyenin üzerine çıkmasını

engellemek için belirlenmektedir, gas limitini aşan sözleşmeler çalıştırılmamaktadır.

$$İşlem\ ücreti = Gas\ x\ Gas\ ücreti \tag{3.1}$$

Bir akıllı sözleşmenin kodu yazılıp derlendikten sonra bir ağa aktarıldığında, bu sözleşmeye özgü bir Ethereum adresi alınır. Bu adresin karşılığında sözleşmenin bayt kodu bulunmaktadır. Önce bir düğüm bu sözleşmeyi bir işlem (transaction) olarak alır, ardından diğer düğümlere yayar. Madenci olan düğümlerden biri bu işlemi alarak elindeki n kadar işlem ile birleştirir ve bu işlemleri içeren bir blok oluşturduğunu ağa yayar. Madenci bu bloğu oluştururken işlem özetini oluşturmuş olur. Yayılan bloklar diğer düğümler tarafından doğrulanır ve kabul edilerek blok zincirine eklenmiş olur. Madenci bloğa eklediğinde akıllı sözleşmeyi çalıştırmış olur ve dönüşünü akıllı sözleşmeyi çalıştıran tarafa yapar. Bu dönüş bir değer (return değeri) ya da tetiklenen başka bir olay (event) olabilir. Daha sonra, başka bir akıllı sözleşme içinden bir akıllı sözleşme çağrılmak istendiğinde o sözleşmeye ait adres ile çağrı yapılabilmektedir.

Akıllı sözleşmelerin nasıl çalıştığını anlamak için bir sonraki bölümde Ethereum altyapısında kullanılan hesap (accounts), değişken durumları (state), işlem (transaction) yürütme mantığı ve Evm yapısına yer verilecektir.

3.1. Ethereum Altyapısı

Ethereum altyapısını anlayabilmek için bu bölümde hesap, değişken durumu (state), işlem (transaction) ve Evm yapısı ile ilgili önemli hususlara yer verilecektir [49]. Transaction ifadesinin Türkçe karşılığı olarak işlem kelimesi kullanılmıştır ancak transaction bir işlem ya da değer aktarımından daha kapsamlı bir paket olarak düşünülmelidir.

Ethereum hesabı (accounts), kullanıcılar tarafından kontrol edilebilir veya akıllı sözleşme olarak ağa aktarılabilir. Ethereum üzerinden işlem gönderebilir ve Ether bakiyesine sahiptir. Hesap türleri; sözleşme hesapları (contract account) ve para transferi amacıyla kullanılan harici hesaplar (externally owned account - EOA) olarak ikiye ayrılabilir. İki tür de token ve Ether alabilir, gönderebilir veya ağda aktif (deploy edilmiş) bir akıllı sözleşme ile etkileşimde bulunabilir. Ana farklılıkları ise şu şekilde sıralanabilir [49]:

- Bir harici hesap oluşturmanın hiçbir maliyeti yokken, bir sözleşme adresi oluşturmak için ağ kullanılacağından bir maliyeti vardır.

- Harici bir hesaptan bir işlem başlatılabilirken, sözleşme adresinden yalnızca kendisine gelen işlemlere cevap olarak yeni bir işlem gönderilebilir.

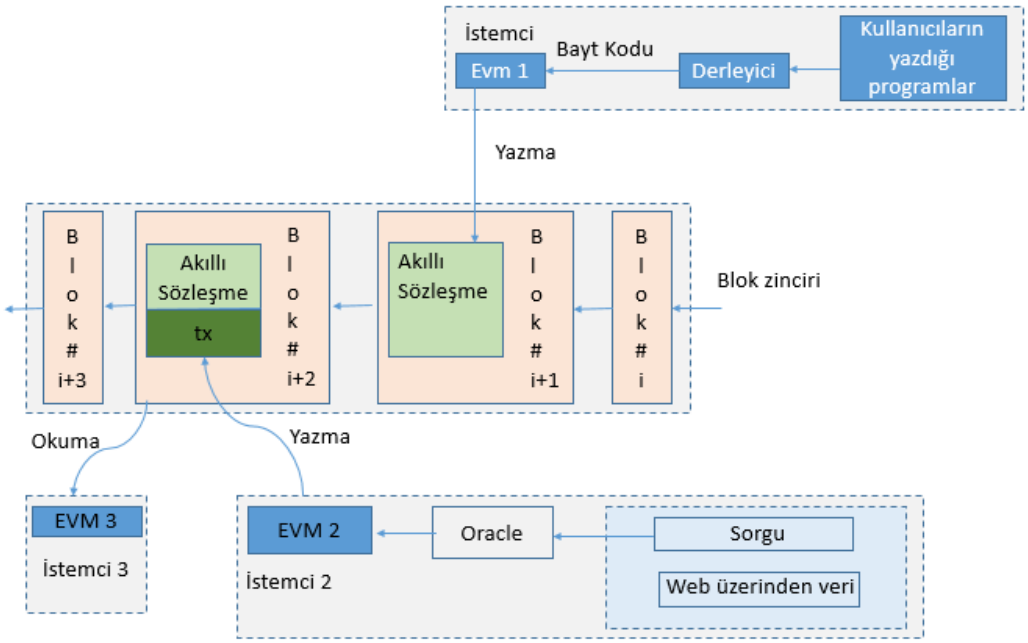
- Harici hesaplar arasında sadece Ether transferi yapılabilirken; harici hesaptan sözleşme hesabına gelen işlemler birçok farklı işlemi tetikleyebilir.

Bir Ethereum hesabının dört alanı vardır. Bunlar; sayaç (nonce), bakiye (balance), kod özeti (codeHash), hafıza özeti (storage root ya da storage hash) olarak sıralanabilir. Nonce, hesaptan gönderilen işlemlerin sayısını gösteren sayaçtır. Bir sözleşme hesabında sayaç (nonce) değeri, hesap tarafından oluşturulan sözleşmelerin sayısını ifade etmektedir. Bakiye (balance), adresin sahip olduğu Ether'in wei cinsinden miktarıdır (1 Ether= 10^18 wei). Kod özeti (codeHash), harici hesaplarda içeriği boş olan, değiştirilemeyen alandır. Evm üzerindeki hesabın kodunun özetidir. Hafıza özeti (storageRoot ya da storageHash), hesabın hafızada tuttuklarının özetidir. Bir hesap, genel ve özel olmak üzere kriptografik bir anahtar çiftinden oluşmaktadır. Özel anahtar ile işlem imzalanır; bu da hem imzalayan kişiye hesapla ilişkili paranın sahipliğini vermiş olur, hem de kötü niyetli kişilerin sahte işlemleri ağa yaymasını engellemiş olur.

İşlem, kriptografik olarak imzalanmış olarak hesaplardan alınan direktiflerdir. Bir hesap, Ethereum ağında değişken durumu (state) güncellemek için bir işlem başlatacaktır. En basit işlem, bir hesaptan diğerine Ether aktarmak olarak düşünülebilir. Evm üzerinde bir değer güncellenmesi için işlemin tüm ağa yayılması gerekmektedir. Ağdaki düğümlerden biri Evm'de çalıştırılacak bir işlem için bir istek yayınlar, ardından bir madenci işlemi çalıştırır ve değer değişikliğinin sonucunda güncel durumu ağa yayar. İşlemin çalışmasının bir maliyeti vardır ve geçerli olabilmeleri için madencilik sürecinin tamamlanmış olması gerekmektedir. Bir işlem; alıcının adresi, göndericinin imzası, miktar, veri, gas limiti ve gas fiyatı bilgilerini içermektedir. Alıcı adresi harici bir hesapsa alıcı alanında işlemin para göndereceği hesap bulunur, alıcı adresi sözleşme hesabıysa işlem akıllı sözleşmenin kodunu bu adreste yürütecektir. Miktar, gönderenden alıcıya gönderilen wei cinsinden Ether miktarıdır. Veri alanı zorunlu değil opsiyoneldir. Gas limiti, işlem tarafından tüketilebilecek maksimum gas miktarıdır, bu miktarın aşılması durumunda işlemde yapılanlar geri alınır.

Gas fiyatı, göndericinin gas birimi başına ödediği ücrettir. Gas, bir madencinin işlemi işlemesi için gerekenlere ilişkin yaptığı hesaplamalar ve katlandığı maliyetlerin ölçümü için bir referanstır. Gas, gwei cinsindendir; 1 Ether = 10^9 gwei'dir. Madencilerin teşvik edilmesi ve ağı saldırılara karşı korumak için gas yapısı kurulmuştur [46].

Evm (Ethereum Virtual Machine), Ethereum istemcisi çalıştıran ve birbirine bağlı binlerce bilgisayarın oluşturduğu yapıdır. Ethereum protokolü bu yapının sürekli, kesintisiz ve değiştirilemez olarak çalışmasını sağlamak için vardır. Özel bir "durum makinesi" olarak da ifade edilebilmektedir (state machine). Evm yığın tabanlı sanal makinedir ve bir yığın (stack) gibi çalışmaktadır. Derlenen akıllı sözleşmenin bayt kodu XOR, AND, ADD, SUB vb. standart yığın işlemlerini gerçekleştiren bir dizi Evm işlem kodu (opcode) olarak yürütülmektedir. Ayrıca blok zincirine özgü ADDRESS, BALANCE, KECCAK256 gibi yığın işlemleri de yürütülebilmektedir. Tüm Ethereum istemcileri bir Evm içermektedir. Ethereum altyapısının işleyişi Şekil 3.1.'de gösterilmiştir [50].



Şekil 3.1. Ethereum akıllı sözleşme işleyişi [50].

EVM bir yığın yapısında, big-endian 256 bit kelime (words) boyutuna sahiptir. Bu boyut, KECCAK256 ve Eliptik Eğri (Elliptic Curve) özetleme algoritmalarıyla uyumludur ancak küçük değerlerin işlemlerini yapmakta etkin değildir [51]. Yığının boyutu 1024'tür, işlem kodları (opcodes) yürütüldüğünde genellikle parametrelerini yığından alırlar, ayrıca Evm içinde verileri geçici olarak depolamak için bellek (memory) alanı bulunmaktadır. Bellek 32

baytlık kelimeler halinde düzenlenmiştir. Bellek ile iletişime geçen komutlara örnek olarak mstore, mload ve mstore8 verilebilir.

Evm'deki diğer bir depolama alanı kalıcı hafıza (storage) olarak isimlendirilir, her akıllı sözleşme kendi alanına erişebilir, veriler anahtar-değer (key-value) ilişkisi ile tutulur [52]. Evm'in ayrıca erişebileceği yığın (stack), bellek (memory) ve kalıcı hafıza (storage) dışında calldata alanı vardır, bu alan işlemlerin veri parametrelerini tutar. Örneğin calldataload komutu bu işlemin parametresindeki veriyi yığına kopyalar [52].

3.2. Akıllı Sözleşmeler

Akıllı sözleşmeler, önceden belirlenmiş koşullar meydana geldiğinde çalışan, o koşullar meydana geldiğinde yeni aksiyonlar tanımlanabilen ve bir blok zincirinde tutulan programlardır. Genel olarak bir sözleşmenin yürütülmesi sürecini otomatik hale getirmek için kullanılmaktadırlar. Böylece, sözleşmenin tarafları herhangi bir güvenilir üçüncü tarafın onayına ihtiyaç duymadan ve zaman kaybı yaşamadan sonuçtan hemen haberdar olmaktadırlar. Ayrıca koşullar meydana geldiğinde, bir sonraki eylemi tetikleyerek bir iş akışını otonom hale getirebilmektedirler [53].

Hızlı, verimli ve doğru sonuçlar alınmasını sağlamaktadır; bir koşul meydana geldiğinde hemen o koşula bağlı eylemler yürütülmektedir. Şeffaftır ve güven ortamını kod sağlamaktadır; güvenilir üçüncü taraflara ihtiyaç olmadığından ve katılımcılar arasında işlemlerin şifrelenmiş kayıtları paylaşıldığından, bilgilerin kötü niyetle değiştirilmesi mümkün değildir [54]. Ancak koşulların kontrol edilebilmesi için kontrol ya da tetiklemeyi sağlayacak dijital kaynaklara ihtiyaç vardır, dijitalleştirilemeyecek süreçlerde insan kaynağı kullanması zorunlu olacaktır. Ayrıca Oracle'ın, yani akıllı sözleşmelere veri sağlayan kaynağın güvenilir olmadığı durumlarda dolaylı olarak akıllı sözleşmeler manipüle edilebilecektir [55].

Bir blok zincire dışarıdan müdahale etmesi zordur; merkezi sistemlere izinsiz erişim ve verilerin değiştirilmesi blok zincirde imkânsıza yakındır. Ağda bir değerin değiştirilmesi durumunda, o bloktan sonraki blok, önceki bloklardan oluşan özet değerini tuttuğundan, değiştirilen bloktan sonraki tüm blokların da yeni değerin özet verisi ile değiştirilmesi gerekmektedir. Değiştirmeye çalışılan bu zamanda yeni bloklar oluşmaya devam

edeceğinden verilerin manipüle edilmesi imkânsız denilebilmektedir [7]. Akıllı sözleşmeler ile üçüncü bir güvenilir tarafa ihtiyaç kalmadığından, buna bağlı gecikme süresi ve maliyeti de yok edilmiş olmaktadır. Tedarik zinciri, sigortacılık, finansal uygulamalar, kamusal işlemler ve sağlık gibi çok çeşitli alanlarda kullanılabilmektedir [56]. Yukarıdaki tanımlara benzer şekilde Amerika Birleşik Devletleri Ulusal Standartlar ve Teknoloji Enstitüsü (NIST); akıllı sözleşmeleri, "blok zinciri ağında kriptografik olarak imzalanmış işlemler kullanılarak dağıtılan bir kod ve veri (fonksiyonlar/işlevler ve durum olarak da ifade edilebilmektedir) kümesi" olarak tanımlamıştır [57].

3.2.1. Akıllı Sözleşme Geliştirme

Herkes bir sözleşme yazarak ağa aktarabilir, bunun için akıllı sözleşmenin nasıl yazılacağının öğrenilmesi ve Ethereum için yeterli Eth'in hesapta tutulması yeterlidir. Bir akıllı sözleşmeyi ağa aktarmak teknik olarak bir işlemdir (transaction), dolayısıyla basit bir Eth transferi için nasıl gas ödeniyorsa sözleşme aktarımında da gas harcanır. Sözleşmenin ağa aktarımında gas maliyeti daha yüksektir.

Ethereum üzerinde Solidity [10] ve Vyper [11] dilleriyle sözleşme geliştirilebilir, Yul ve Yul+ ise tecrübeli geliştiriciler için önerilir. EVM'in akıllı sözleşmeyi yorumlayabilmesi ve saklayabilmesi için sözleşme ağa aktarılmadan derlenmelidir. Solidity özellikle geliştirmeye yeni başlayan kullanıcılar için topluluk desteği ve kapsamlı dokümanları ile avantaj sağlar. Vyper ise akıllı sözleşme yazmak isteyen Python geliştiricileri için akıllı sözleşme geliştirmeye başlamak için iyi bir tercih olabilir. Vyper'ın daha az özelliği vardır, bu hızlı prototip oluşturmaya katkı sağlar. Vyper, denetimin kolay olmasını ve geniş kitlelerce kolay okunmayı hedefler. Yul ve Yul+ [58] ise daha düşük seviyeli bir dildir, böylece gas optimizasyonu yardımcı olur.

Açık ağlardaki akıllı sözleşmeler halka açık API'lar gibi düşünülebilir. Böylece bir akıllı sözleşme geliştirirken diğer sözleşmeler çağrılabilir. Hatta bir sözleşme başka bir sözleşmeyi ağa yükleme işlemini bile yapabilir [59].

3.2.2. Akıllı Sözleşmelerin Kısıtları ve Çoklu İmza Özelliği

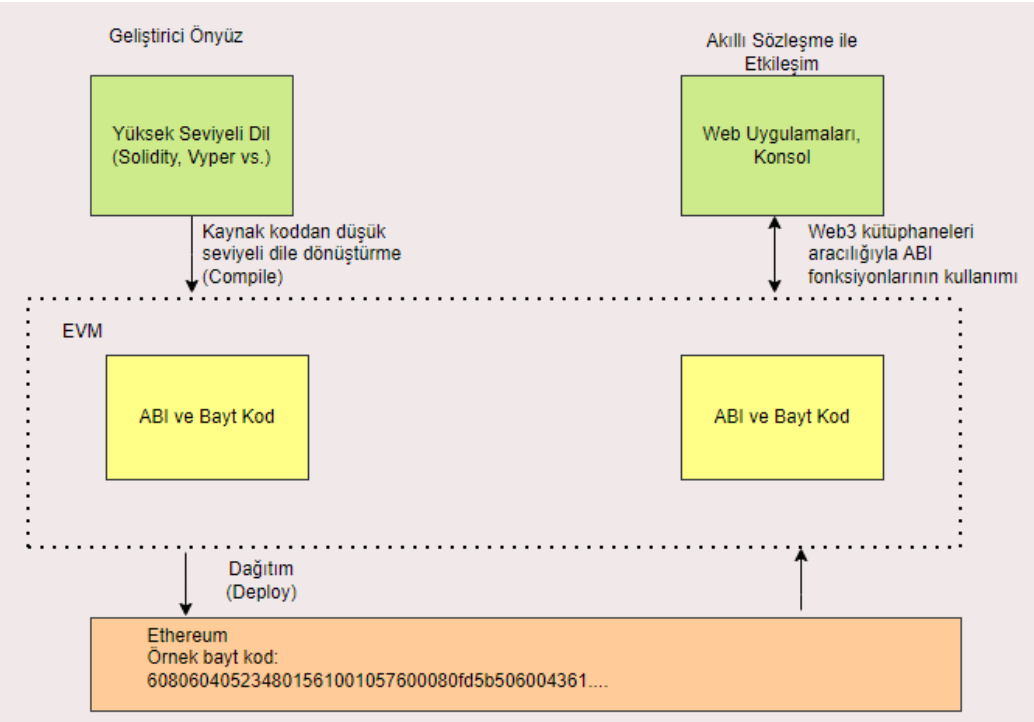
Akıllı sözleşmeler temel olarak güvenlik gerekçesi ile ağ dışındaki bir veri sağlayıcıdan veri

alamazlar ancak bu şekilde de gerçek dünyadaki olaylara karşılık veremezler. Çözüm olarak zincir dışında yer alan ve akıllı sözleşmelerin kullanımına sunulan “oracle” araçları kullanılır. Akıllı sözleşmenin bir diğer kısıtı ise maksimum sözleşme büyüklüğüdür. Bir akıllı sözleşmenin maksimum büyüklüğü 24 KB olabilir ancak bu sınır tasarım deseni (Diamond Pattern) ile dolaylı olarak aşılabilir.

Çoklu imza isteyen sözleşmeler ile tek bir kişinin veya noktanın kırılganlığı aşılabilmektedir. Bu özellik ile bir işlem için birden fazla kişinin imzası gerekir. Böylece sözleşmenin yürütülmesi ve özel anahtar yönetimi sorumluluğu taraflar arasında paylaşılır [59].

3.3. Akıllı Sözleşmelerin Ağa Yüklenmesi

Bu bölümde, bir akıllı sözleşmenin ağa yüklenmesinin nasıl olduğu açıklanacaktır. Ethereum üzerinde genellikle Solidity [10] ve Vyper [11] dilleriyle birlikte birçok farklı dil ile, EOSIO üzerinde ise C++ [60] dili ile geliştirilen akıllı sözleşmeler ilgili dilin derleyicisi tarafından derlendikten sonra platforma özgü işlemlerden geçirilerek abi ve bayt kod elde edilir. Derleme sonucundaki kodlar ağa aktarıldığında bu sözleşmeye özgü Ethereum adresi oluşur. Bir akıllı sözleşmenin yazıldıktan sonra ağa dağıtımı ve onunla etkileşimi Şekil 3.2.’de görülmektedir [61].



Şekil 3.2. Akıllı sözleşmenin derlenip ağa aktarılması ve etkileşim adımları [61]

Solidity dili için yazılan bir akıllı sözleşmenin, ağa aktarılmasından çağrılmasına geçen süreç şu şekildedir; Bir akıllı sözleşmenin kodu dilin derleyicisi tarafından derlendikten sonra bir ağa aktarıldığında, bu sözleşmeye özgü Ethereum adresi alınır. Bu adresin karşılığında sözleşmenin bayt kodu ve abi bulunmaktadır. İlk olarak bir düğüm bu sözleşmeyi To alanı boş olan bir işlem (transaction) olarak alır, ardından diğer düğümlere yayar. Madenci olan düğümlerden biri bu işlemi alarak n kadar diğer işlem ile birleştirir ve bir blok oluşturduğunu ağa yayar. Madenci bu bloğu oluştururken işlemlerin özetini de oluşturur. Yayılan blok diğer düğümler tarafından doğrulanır ve kabul edilerek blok zincire eklenmiş olur. Madenci akıllı sözleşmenin kodunu içeren işlemi bloğa eklediğinde akıllı sözleşmeyi çalıştırmış olur ve dönüşünü akıllı sözleşmeyi oluşturan tarafa yapar. Ağa yayılmış durumda olan akıllı sözleşme daha sonra başka bir akıllı sözleşme ya da kişi tarafından çağrılmak istendiğinde o sözleşmeye özgü Ethereum adresi ile çağrı yapılır [52].

3.4. Akıllı Sözleşmenin Derlenmesi

Önceki bölümde genel hatlarıyla anlatılan derleme ve ağa yükleme işleminin ardından bu bölümde bir akıllı sözleşme derleyicisi olan Solc'un aldığı parametrelere ve ürettiği dosyalara ayrıntılı şekilde yer verilecektir.

Bir akıllı sözleşme derlendiğinde bin ve runtime bin olmak üzere iki dosya oluşmaktadır. Bin dosyasında runtime bin'e ek olarak programın yapıcısı (constuctor) ve programın blok zincire yerleşmesi için adres başlama bitiş yerleri gibi gerekli bilgileri de içerir. Blok zincirde bulunan ve analiz edilecek ikilik sistemde tutulan dosya runtime bin dosyasıdır. Bu dosyadaki bayt halinde tutulan veriler, ilgili akıllı sözleşmenin Evm tarafından okunması, yorumlanması ve yürütülmesi için Evm'in kullanabileceği işlem kodlarının (opcode) onaltılık tabanda (hexadecimal) gösterilmiş halidir. Akıllı sözleşmenin bayt kodu olarak ifade edilecek bu dosya, Evm üzerinde sanal bir rom'da tutulur bu nedenle değiştirilemez şeklinde ifade edilir. Ancak gelişen ihtiyaçlar nedeniyle dolaylı şekilde farklı sözleşmeyi çağırması ve ağdan kaldırılması gibi yöntemler geliştirilmiştir, bu ayrıntı bu çalışmanın odak konusu dışındadır. Evm tarafından işlem kodlarına karşılık gelen bayt kodlar listelenmiştir [62].

Bir akıllı sözleşmeyi oluşturan bin dosyası yalnızca ilk defa oluşturulma aşamasında çalıştırılırken, runtime bin akıllı sözleşme her çağrıldığında çalıştırılır. Remix Ide üzerinde

yazılan ve Şekil 3.4.'te görünen kod Remix Ide dışında solc ile derlenmesi istenirse Şekil 3.3'teki gibi komut ile yapılabilir.

```
> solc gazi.sol --bin --abi -optimize -overwrite
```

Şekil 3.3. Solc ile akıllı sözleşme derlemesi

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;
contract gazi {
    uint256 public deger1;
    uint256 public deger2;

    constructor()  {
        deger2=5;
    }

    function set(uint256 _param) external {
        deger1=_param;
    }
}
```

Şekil 3.4. Akıllı sözleşme örneği

Remix Ide üzerinde derleme ve ağa aktarma işlemleri cüzdan uygulamalarının yardımıyla kolaylıkla yapılabilmektedir. Şekil 3.4.'te bulunan kod derlendikten sonra elde edilen bayt kodu Şekil 3.5.'te bulunmaktadır. Bayt kodunun işaretli ilk kısmı akıllı sözleşmeyi başlatan kısımdır bu nedenle blok zincire dağıtılmaz ve ağda tutulmaz.

"object":
"6080604052348015610010576000 80fd5b506005600181905550 61017c806100
2860003960 00f3fe6080604052348015610010576000 80fd5b506004361061004
15760003560e01c80630279f536146100465780632dc0c1311461006457806360
fe47b114610082575b600080fd5b61004e61009e565b60405161005b919061010
5565b60405180910390f35b61006c6100a4565b60405161007991906101055565b
60405180910390f35b61009c60048036038101906100979190610100c9565b6100a
a565b005b60015481565b60005481565b8060008190555050565b60008135905
06100c38161012f565b92915050565b60006020828403121561 00df576100de61
012a565b5b60006100ed848285016100b4565b91505092915050565b6100ff816
10120565b82525050565b60006020820190506101 1a60008301846100f6565b92
915050565b60008190509190505 65b600080fd5b61013881610120565b811461
014357600080fd5b5056fea2646970667358221220f6a539ed959e3d3adacca25ac
654d9922d44107be635b4d8dc26795e1794f0ab64736f6c63430008070033",

Şekil 3.5. Bayt kod örneği

Evm bayt kodları işlem koduna çevirerek yorumlayabileceği kodları elde etmiş olur. Bayt koddan işlem koduna nasıl çevrildiğini gösteren şekil aşağıdaki (Şekil 3.6.) gibidir [63]. Örneğin 6080 ile başlayan bayt kodunda 60 Push1 işlemini, 80 ise bu işlemin parametresinin 0x80 olacağını dolayısıyla 0x80 değerini yığına eklemeyi ifade etmektedir.

```
PS: 0X0, işlem kodu: PUSH1 0x80
PS: 0X2, işlem kodu:PUSH1 0x40
PS: 0X4, işlem kodu:MSTORE
PS: 0X5, işlem kodu:CALLVALUE
PS: 0X6, işlem kodu:DUP1
PS: 0X7, işlem kodu:ISZERO
PS: 0X8, işlem kodu:PUSH2 0x10
PS: 0Xb, işlem kodu:JUMPI
PS: 0Xc, işlem kodu:PUSH1 0x0
PS: 0Xe, işlem kodu:DUP1
PS: 0Xf, işlem kodu:REVERT
PS: 0X10, işlem kodu:JUMPDEST
...
```

Şekil 3.6. İşlem kodu çevrimi [63]

Şekil 3.5.'te işaretli kısımda bulunan ve akıllı sözleşmeyi başlatan bayt kodlar ile; hafızada kullanılmayan alanlar işaretlenir, işlem ile birlikte Wei gönderilmişse geri alma (revert) işleminin yapılmasını sağlar, dönüş değeri olarak akıllı sözleşmenin kodunu gösteren işaretçi ile kodun uzunluğunu gönderir [63]. İşaretli ilk kısım çalıştığında akıllı sözleşmenin çalışabilir halde bayt kodu elde edilmiş olur, blok zincir üzerinde tutulan akıllı sözleşmenin bayt kodu budur (runtime bytecode).

Kod geliştirici tarafından akıllı sözleşme içinde yazılan fonksiyonlara karşılık gelen bayt kodun bulunması için fonksiyon_adı(parametre_tipleri)'nin özetinin alınması ve on altılık formatta tutulan bu özet değerin soldan ilk dört baytının alınması yöntemi kullanılmaktadır. Örneğin kod geliştirilirken oluşturulan function hesapla(int256 deger) şeklinde ismi ve parametresi girilen fonksiyonun imzası hesapla(int256)'dır. Bu imzanın keccak256 ile özeti alındığında elde edilen on altılık bayt serisinin ilk dört baytı 0x38d9fdb9'dur ve fonksiyonun adı ile parametrelerini ifade eden bayt imzasıdır. Kaynak koddan bayt koda dönüştürülen akıllı sözleşmede ilgili fonksiyonun bulunduğu kısım bulunan bu bayt kodu ile başlamaktadır. Daha önce girilen fonksiyon ve parametreler için bayt imzası hesaplanan fonksiyonları bayt imzası ile aramak için mevcut veri tabanından faydalanılabilir [64]. Bu

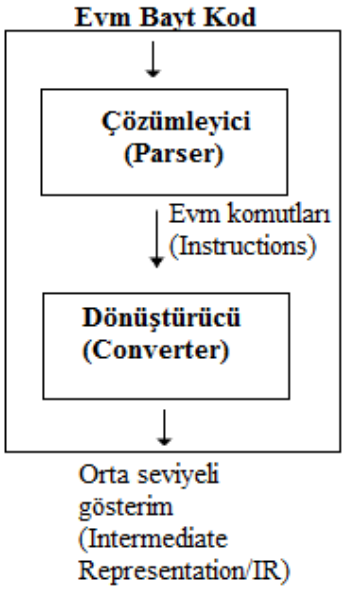
bölümde Ethereum altyapısı ve akıllı sözleşmeler kapsamlı bir şekilde ele alınmıştır, sonraki bölümde kodu ağa yüklenmeyen akıllı sözleşmelerde inceleme yapmak için gerekli bayt koddan kaynak koda dönüştürme işlemine yer verilecektir.

4. BAYT KODDAN KAYNAK KODA DÖNÜŞTÜRME SÜRECİ

Alt seviyeli programlama dilinden yüksek seviyeli dile çevrim kaynak koda dönüştürme (decompile) olarak isimlendirilmektedir [65]. Başarılı kaynak koda dönüştürme işlemi, zafiyet analizlerini kolaylaştıracaktır ve akıllı sözleşmelerin taraflarının kod geliştiricilere güvenlerini artıracaktır [51]. Bir akıllı sözleşme derlenirken derleyicisi tarafında yapılan optimizasyonlar ile fonksiyon ve event adları gibi alanlar tek yönlü dönüşümlerden geçmektedir bu sebeple bayt koddan kaynak koda yüzde yüz bir dönüşüm mümkün değildir. Ancak bayt koda göre anlaşılırlığı bir derece yüksek kodlar elde etmek mümkündür ayrıca fonksiyon isimleri ile parametrelerinin özetinin ilk dört baytından oluşan imza veri tabanlarında [64] arama yaparak daha önce tespit edilen fonksiyon adlarından çıkartım yapılabilmektedir. Evm bayt kodu kaynak koda dönüştüren araçların performanslarını gösteren kriterlere bölüm 4.2.'de yer verilecektir.

Geliştiriciler akıllı sözleşmeleri ağa yüklediklerinde yalnızca bayt kodu ağda bulunmaktadır. Kaynak kodun ağda doğrulanması zorunlu bir süreç değildir. Dune [66] üzerinden alınan raporlara göre 2022 yılının ilk çeyreğinde 7.811.630 adet akıllı sözleşme oluşturulmuştur, Etherscan [67] üzerinden alınan istatistiklerde 2022 yılının ilk çeyreğinde kaynak kodu ağa yüklenen akıllı sözleşme sayısı 28.161'dir. 2022 ilk çeyreği için elde edilen istatistiklere göre kaynak kodun ağa yüklenmesi oranı %0,36'dır. Bu oranın dışında kalan akıllı sözleşmelerin herhangi bir inceleme ya da soruşturmada incelenmesi gerektiğinde yalnızca bayt koduna erişilebilmektedir. Bu durumda inceleme için bayt koddan daha anlaşılır, yüksek seviyeli dile çevrime ihtiyaç bulunmaktadır.

Bu bölümde örnek bir aracın Evm bayt koddan Solidity diline çevrim işlemini nasıl yaptığı incelenmektedir. Örnek olarak seçilen araç olan Dsol'ün [51] kaynak koda dönüştürme işlemi üç adımdan oluşmaktadır, ilk adım (front-end) Şekil 4.1.'de bulunmaktadır, bu adımda çözümleme işlemi bayt kodu Evm komutlarını ifade edecek şekilde ayırır ve komutları bloklara böler. Dönüştürüme aşamasında Evm komutları orta seviyeli gösterim olan IR formuna dönüştürülür. IR, blokların CFG (control flow graph) olarak gösterimi ile ifade edilir. CFG'nin doğruluğu sonraki adımlarda artırılmaya çalışılır.

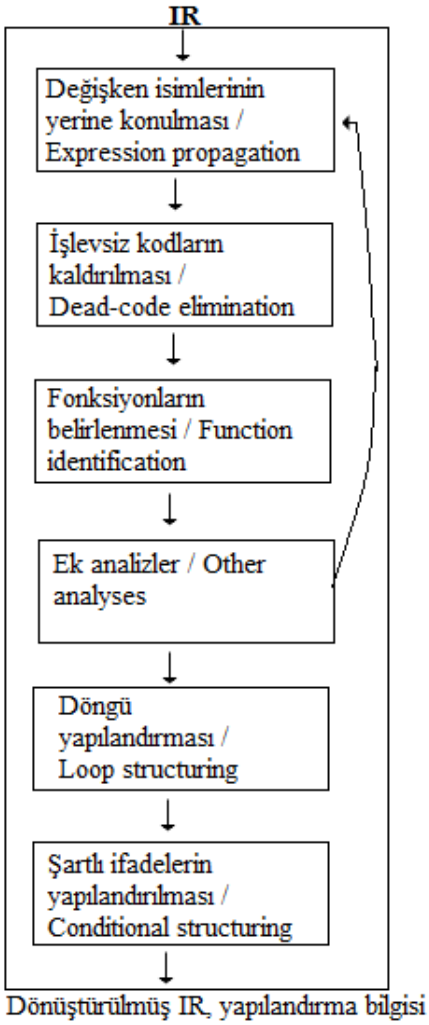


Şekil 4.1. Dsol ilk adım [51]

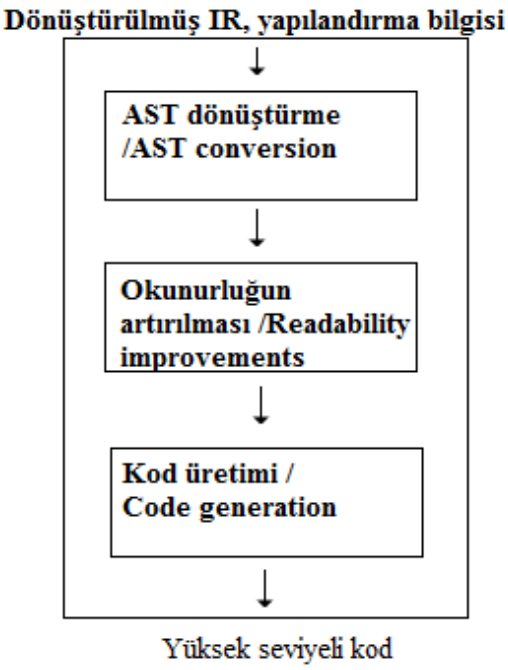
İkinci adım (middle-end) Şekil 4.2.'de bulunmaktadır. Bu adımın amacı koddan düşük seviyeli ayrıntıları kaldırmak ve onu yüksek seviyeli dile çevrime uygun hale getirmektir. Örneğin bu adımın ilk aşaması değişken isimlerinin yerine konulmasıdır, bunun ile bazı tanımlar işlevsiz hale gelebilir. Bu sebeple bir sonraki aşamada işlevsiz hale gelen kodlar kaldırılır. Fonksiyonların belirlenmesi aşamasında JUMP ifadelerinin fonksiyon çağrısına karşılık geldiği yerler baz alınarak fonksiyonlar ayrılır ve ayrı bir blok olarak analiz edilir.

Döngü yapılandırılmasında; döngüler ve içerdikleri bloklar tespit edilir. Şartlı ifadelerin yapılandırılmasında ise şartlı JUMP ifadelerinde dallanacak blokların tespiti yapılır.

Üçüncü adım (back-end) Şekil 4.3.'te bulunmaktadır. Bu adım AST, okunabilirliğini artırma ve kod üretimi aşamalarından oluşmaktadır. İlk aşamada her fonksiyon için oluşturulan IR diğer girdi parametresi olan yapılandırma bilgisi yardımıyla bir AST'ye (Abstract Systax Tree) dönüştürülür. İkinci aşamada AST üzerinde, okunurluğu artırmak amacıyla değişkenleri isimlendirme, dizi erişimlerini tanımlama ve benzeri küçük dönüşümler yapılır. Son aşamada AST'de bulunan fonksiyonlardaki düğümler gezilerek koda dönüştürülür.



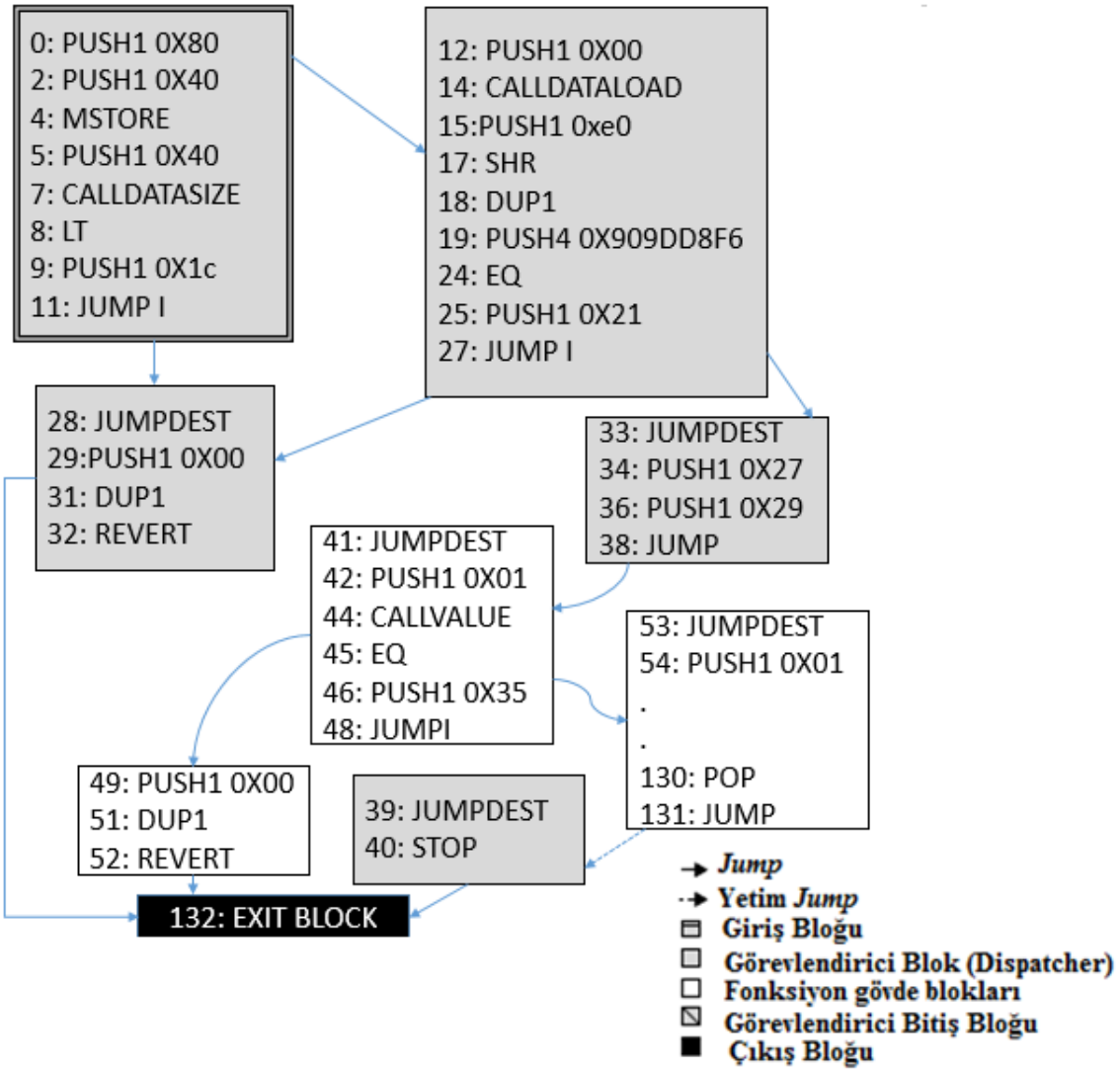
Şekil 4.2. Dsol ikinci adım [51]



Şekil 4.3. Dsol üçüncü adım [51]

Bazı durumlarda IR üzerinde tüm alt seviye detay bilgileri kaldırmak mümkün olmayabilir. Örneğin değişken isimlerinin yerine konması veya fonksiyonların belirlenmesinde hata alınması halinde, gidilecek yerin hafızadan alınması anlamına gelen bazı dolaylı jump (indirect jump) ifadeleri kalmaya devam edebilir. Bu durumda kod üretimi aşamasında aslında Solidity'de olmayan go to ifadesi eklenir, bu da kod üzerinde manuel değişiklik yapmadan derlenemeyeceği anlamına gelir. İdeal dönüşümlerde Solidity dilinde olmayan herhangi bir ifadenin elde edilen kaynak kodda olmaması beklenmektedir. Araçlar ile üretilen örnek bir CFG aşağıda (Şekil 4.4.) yer almaktadır [68].

Örnek CFG'de görüldüğü gibi bir blok JUMP, STOP veya REVERT gibi komutlarla bitmektedir. JUMPI komutu için bir hedef adres bulunmalıdır, hedef adrese göre ilgili bloğa dallanır.



Şekil 4.4. CFG örneği [68]

Bu bölümde örnek bir aracın nasıl çalıştığına yer verilmiştir, sonraki bölümde mevcut kaynak koda dönüştürme araçlarının özellikleri ve araçları karşılaştırabilmek için bulunan kriterler sunulacaktır.

4.1. Bayt Koddan Kaynak Koda Dönüştüren Araçlar

Literatürde bulunan Evm bayt koddan yüksek seviyeli kaynak koda dönüştürme işlemini (decompile) yapan araçlar şu şekilde sıralanabilir; Porosity [69], Vandal [70], Gigahorse [13], Elipmoc [14], Erays [15], Panoramix [71], Dsol [51]. Ayrıca Jeb [17], Trustlook [18] ticari, Ethervm [16] ise açık kaynak kodlu olmayan uygulamalardandır. Bayt koddan daha okunabilir assembly diline çevrim disassembly, daha yüksek seviyeli dillere çevrim ise

decompilation olarak tanımlanmaktadır [72].

Porosity, Evm bayt kodunu assembly diline çevirdikten sonra bir CFG oluşturur. C++ ile yazılmış ve komut satırından çalıştırılan bir araçtır. Akıllı sözleşmeler üzerinde zafiyet tespitini yapmak için bayt kodu kaynak koda dönüştürmektedir. Örnek çıktısı Şekil 4.5.'teki gibidir [69]. Bu araçta 2017 yılında beri geliştirme yapılmamakta ve geliştiricileri aracın yeni ihtiyaçlara cevap vermeyeceğini belirtilerek Jeb kullanılmasını önermiştir [73].

```
function getBalance(address) {
    return store[arg_4];
}

function addToBalance() {
    store[msg.sender] = store[msg.sender];
    return;
}

function withdrawBalance() {
    if (msg.sender.call.value(store[msg.sender])()) {
        store[msg.sender] = 0x0;
    }
}

**L12 (D8193): Potential reentrant vulnerability found.**
```

Şekil 4.5. Porosity çıktı örneği

Vandal, Evm bayt kodunu register tabanlı IR'a dönüştürür ve etkileşimli bir html sayfasında görüntülenebilen CFG oluşturur. Kaynak koda dönüştüren, blok zincir kazıyıcı (scraper) ve DataLog tanımlarıyla beslenen Souffle [74] ile geliştirilmiş zafiyet analizi yapan araçlardan oluşur. Python ile geliştirilmiştir ve komut satırından çalıştırılır. Github üzerinde [70] Bsd lisanslı şekilde kullanıma açıktır [75].

4.1.1. Gigahorse

Gigahorse, Evm bayt kodunu, 3-adresli kod gösterimiyle (ör: a=b operatör c) ifade edilen yüksek seviyeli dile çeviren araçtır. Aşağıdaki aşamalardan geçerek işlemi tamamlar;

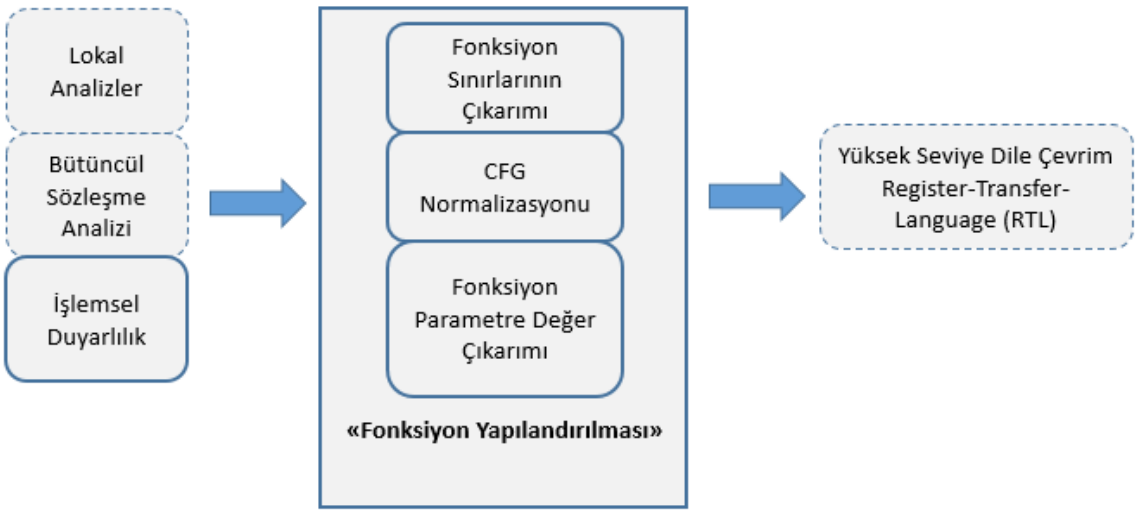
- İlk aşamada bayt kod blok sınırları belirlendikten sonra bloklara ayrılır.
- Her blokta yığının nasıl değiştiğini gösteren ifadeler çıkartılır (stack effects).
- Ardından CFG oluşturur ve global değişkenlerle birlikte 3 adres gösterimli IR oluşturur.

- Fonksiyon sınırlarını blok girişleri ve çıkışları ile fonksiyon çağrıları ile birlikte genel ve özel fonksiyonlar için belirler.
- Fonksiyonların aldığı parametreler ile dönüş parametreleri belirlenir. Bu parametreler için yeni değişkenler oluşturulur, fonksiyon içeriğindeki akışa göre bir analiz yapılır ve sonucunda fonksiyonlar için 3 adres gösterimli IR oluşturulur. Veriler fonksiyon parametreleri, fonksiyon dönüş parametreleri, hafıza (storage) veya bellekte (memory) aracılığıyla aktarılır.

Sonuçta elde edilen kaynak kod için mantık tabanlı kurallar (Datalog) kullanılır. Gigahorse yayınında, Gigahorse'un Vandal ve Porosity'e göre daha hızlı olduğu görülmüştür. Ayrıca kaynak koda dönüştürme işleminde hata alma ve zaman aşımı alma oranlarının Gigahorse'ta daha düşük olduğu tespit edilmiştir. Diğer taraftan kaynak koda dönüştürmede başarı kriterlerinden sayılabilecek oluşan kodda erişilemeyen blokların oranı Vandal'a göre daha düşüktür [13].

4.1.2. Elipmoc

Gigahorse 2.0 gibi düşünülen ve aynı ekip tarafından geliştirilen Elipmoc'un mimarisi aşağıdaki (Şekil 4.7.) gibidir. Mimaride görülen kesik çizgiler Gigahorse ile aynı olan kısımları ifade etmektedir. Kesikli olmayan çizgiler yeni geliştirilen modüllerdir.



Şekil 4.6. Elipmoc Mimarisi [14]

Elipmoc yayınında Elipmoc'un Panoramix ve Gigahorse ile karşılaştırılması yapılmıştır. Kaynak koda dönüştürme işleminde zaman aşımı alınmayanlarla yapılan hesaplamada ortalama dönüştürme süresi Elipmoc'un 1,23 saniye iken Gigahorse'un 3.94'tür. Bu da Elipmoc'un Gigahorse'a göre daha hızlı olduğunu gösterir. Ölçeklenebilirliğin göstergelerinden biri olabilecek zaman aşımına uğrama oranı Elipmoc için %4,94 iken Gigahorse için %18,74'tür.

Diğer taraftan Elipmoc'un Panoramix karşılaştırılmasından elde edilen sonuçlar şu şekildedir; dönüştürme işlemi esnasında zaman aşımı oranı Elipmoc'un %5 iken Panoramix'in %7,12'dir. Zaman aşımı almayanlar üzerinde yapılan değerlendirmede ortalama dönüştürme süresi Elipmoc'un 1,41 saniye iken Panoramix'in 14,12 saniyedir. Özetle Elipmoc Gigahorse ve Panoramix'e göre daha hızlıdır. Karşılaştırma yapılacak bir diğer araç Dsol'ün nasıl çalıştığına önceki bölümde yer verilmiştir.

4.1.3. Erays

Erays, Evm için tasarlanan bir tersine mühendislik aracıdır (reverse engineering). Bir akıllı sözleşmeyi hex formatında olan bayt kodunu alarak onu daha okunabilir ve anlaşılabilir bir forma dönüştürür. İlk aşamada hex kodu Evm komutlarına dönüştürülür ve bloklara ayrılır. Komutlar doğrusal bir tarama ile oluşturulmaktadır [76]. Evm'in yığınlar kullanarak çalışan (stack-based) komutları bir değişken/kayıtçı kullanarak çalışan (register-based) komutlara çevrilir. Komutların üzerinde çalıştığı değişkenleri üretebilmek için $stack_0'dan stack_1023'e kadar değişken üretilerek yığın verileri ile eşleştirilir. Ayrıca bellekten alınan veri için $m değişkeni ve değişken içeriklerinin yer değiştirilmesinde geçici olarak kullanabilmek için $t değişkeni kullanılır. Okunurluğu artırmak için Evm'de olmayan komutlar da eklenmiştir (Intcall, neq, geq vb.).

Erays, komut dönüşümlerini yaptıktan sonra kodu bloklara ayırır ardından Cfg dönüşümlerini yapar. Örnek kod bloğu ayrımı Şekil 4.7.'de bulunmaktadır.

```
b0:
   6000     PUSH1 0x60
   54       SLOAD
   600a     PUSH1 0xa
   6008     PUSH1 0x8
   56       JUMP
b1:
   5b       JUMPDEST
   56       JUMP
...
```

Şekil 4.7. Bloklara bölünmüş komut örneği [15]

Cfg'de her düğüm bir bloğu gösterir ve aralarındaki bağlantı bir dallanmayı ifade eder. Bir bloktan sonra gelecek olan bloğu belirlemek için ilgili bloğun son komutuna bakılır, son komut incelendiğinde aşağıdaki gibi üç farklı durumda olabilir.

1. Cfg akışını değiştirmeyen
2. Yürütmeyi durduran (STOP, REVERT, INVALID, RETURN, SELFDESTRUCT)
3. Dallara ayrılan (JUMP, JUMPI)

İlk durumda bloktan sonraki bloğa (ardıl blok) geçilir, ikinci durumda yürütme durduğu için ilgili bloktan sonraki blok olmayacaktır. Son durumda dallanmayı sağlayan komutun hedefinde olan adres ile başlayan blok sonraki bloktur. Hedef adres yığında en üstte bulunan adrestir. Şekil 4.8.'de görünen örnekte b1 bloğu ele alındığında nereye dallanması gerektiği yalnızca b1'e bakılarak anlaşılamamaktadır. Bu nedenle b1'den önceki blokların da incelenmesi ve yığının analiz edilmesi gerekmektedir. Önceki blokların analizinde yığına etkisi olan komutlar incelenir ve dallanan adreslerin tespiti yapılır [76].

Erays içerisinde kodların daha anlaşılır olmasını sağlayan optimizasyon ve sadeleştirme işlemleri de yapılır. Erays'ın karşılaştırılması ilgili yayında yalnızca Porosity ile yapılmıştır. Porosity'nin, 34.000 akıllı sözleşmenin olduğu veri setinde hata vermeden yalnızca %5,3'ünü yüksek seviyeli dile çevrilebildiği belirtilmiştir. Buna karşılık Erays'ın hata almadan çevirebildiği akıllı sözleşme oranı %97,7'dir.

Elipmoc'un önceki bölümde örnek araç olarak tanıtılan Dsol ve Erays karşılaştırılmasının şimdiye kadar incelenen yayınlarda yapılmadığı görülmüştür. O nedenle tez çalışmasının sonraki bölümünde Elipmoc, Erays ve Dsol'ün karşılaştırılması yapılarak, inceleme yapacak kurum ya da kişilere araç seçiminde yardımcı olmak hedeflenmiştir. Yapılacak çalışmada yalnızca kodu açık olan araçlar karşılaştırılmıştır. Ticari uygulamalar olan Jeb [13], Trustlook [14] ile kodu açık olmayan Ethervm.io [12] üzerindeki kaynak koda dönüştürme aracı çalışmaya dahil edilememiştir. Kaynak kodu açık olan uygulamalar Elipmoc, Dsol, Erays ve Panoramix'tir. Ethervm.io [12] da kendi sitesi üzerinden kullanılabilmektedir. Etherscan üzerinde bulunan ara yüzün arka planda kullandığı araç da Panoramix'tir [62].

4.1.4. Mevcut Yayınlarda Bulunan Karşılaştırmalar

Yapılan literatür çalışmasında Grech ve arkadaşlarının çalışması Gigahorse [13] ile Elipmoc [14] ve Zhou ve arkadaşlarının çalışması Erays [15] incelenerek hangi araçlarla karşılaştırmalarının yapıldığı değerlendirilmiştir. Bu yayınlarda sunulan araçların hangi araçlarla karşılaştırıldığı Çizelge 4.1'de bulunmaktadır.

Çizelge 4.1. Mevcut yayınlarda karşılaştırılması yapılan araçlar

Yayın / Araç	Gigahorse [13]	Erays [15]	Elipmoc [14]
Porosity [69]	✓	✓	
Vandal [70]	✓		
Gigahorse [13]			✓
Panoramix [71]			✓

Kaynak: Taranan yayınlardan derlenmiştir.

Mevcut yayınlarda karşılaştırılan kriterleri ve değerlerini içeren tablo Çizelge 4.2.'de bulunmaktadır. Burada okunurluğu artırmak için Gigahorse (G) [13], Porosity (Po) [69], Erays (Er) [15], Elipmoc (El) [14], Vandal (V) [70] ve Panoramix (Pa) [71] için belirtilen kısaltmalar kullanılmıştır. Tespit edilemeyen sonuçlar için T.E. kısaltması kullanılmıştır.

Karşılaştırması yapılan kriterlerden sözleşme başına fonksiyon sayısı ile değişken başına veri akışı sözleşmelerdeki karmaşıklığı ifade etmek için sunulan kriterlerdir. Veri seti büyüklüğü ilgili yayınlarda kullanılan setlerde tekrarlı sözleşmeler çıkarıldıktan sonra elde edilen setin büyüklüğünü gösterir.

Çizelge 4.2. Mevcut yayınlarda karşılaştırma sonuçları

Kriter	Gigahorse [9]	Erays [11]	Elipmoc [10]
Hata Vermeden Çalışma Oranı	G:% 99,98 V:% 88,13 Po:48,56	Er:%97,7 Po:%5,3	G: ~ %88 El: ~ %95
Ortalama Çalışma Süresi	G: 0,7 sn V: 5,7 sn Po:1,2 sn	-	G: 4,03 sn Pa:15,6 sn El: 2,74 sn
Hedefi Bilinmeyen Jump Oranı	G: %0 V: %0,81 Po: T.E.	-	-
Erişilemeyen Kod Bloğu Oranı	G: %8,7 V: %19,8 Po: T.E.	-	-
Sözleşme Başına Fonksiyon Sayısı	G: 21,0 V: 12,2 Po: 1,92	-	-
Çözümlenmemiş Terim (Unresolved Operand)	-	-	G:%37,2 Pa: T.E. El:%0,5
Veri Seti Büyüklüğü	91.8 bin tekil sözleşme	34.328 bin tekil sözleşme	5 bin tekil sözleşme
Polimorfik Jump Oranı	G: %2,58 V: % 2,72	-	G: %38,6 El: %11,4
Yapılandırılmamış Döngü Oranı	G:% 0,0 V: % 11,7	-	G: %47,1 El: %8,8
Çıkma Koşulu Olan Döngü Oranı	G: % 96,6 V: %91,7	-	-
Değişken Başına Veri Akışı	G: 3,82 V: 8,20	-	-

Kaynak: Taranan yayınlardan derlenmiştir, değerlerin hangi yayında bulunduğu başlık satırında belirtilmiştir.

Grech ve arkadaşları [10] tarafından yapılan çalışmada Elipmoc, Gigahorse ve Panoramix araçları ile karşılaştırılmıştır. Çalışmada elde edilen sonuçlara göre Elipmoc Gigahorse'a göre daha iyi bir kesinlik oranına sahiptir. Bu oranı bulmak için çözümlenememiş operatörler incelenmiştir. Gigahorse için %37.2 olan çözümlenemeyen operatörlerin oranı Elipmoc'ta %0.5'e düşmüştür. Diğer taraftan Panoramix ile karşılaştırıldığında yapılan çağrıların (CALL) %40 daha fazlası derlenen ifadeler ile eşleştirilir. Ayrıca Panoramix'e göre Elipmoc beş kat daha hızlıdır ve işlemin zaman aşımına uğrama oranı Panoramix'te %17.8 iken Elipmoc'ta %5'in altındadır bu değerler Elipmoc'un daha ölçeklenebilir olduğunu göstermektedir. Çizelge 4.2'de bulunan diğer kriterlerin açıklaması sırayla şu şekildedir [13, 14]:

- Hedefi bilinmeyen jump oranı: JUMP komutunun hedef adresinin belirlenememesi durumunda bu oran artar, artması kodun anlaşılırlığını zayıflatacağından fazla olması aracın daha işlevsiz olduğunu gösterir.

- Erişilemeyen kod bloğu oranı: Bloklara ayrılan temel bloklardan hiç erişilemeyen blokların oranlarını ifade eder, aracın değerlendirilmesinde düşük olması istenen parametrelerdendir.

- Sözleşme başına fonksiyon sayısı: Bu kriter sözleşmelerin büyüklüğünü ve karmaşıklığını değerlendirilirken kullanılır, sözleşmedeki fonksiyon sayısıdır.

- Çözümlenememiş terim sayısı (unresolved operand): Çıktıda hatalı ya da eksik işlenen terimleri ifade eder.

- Polimorfik jump oranı: Geçişsiz Cfg kenarları (edge) için birden fazla hedefe çözümlenen Jump oranlarını ifade eder. Düşük olması aracın daha etkin olduğunu gösterir.

- Yapılandırılamamış döngü oranı: Yapılandırılmamış döngü veya Cfg'ler, olağan dışı kontrol-akış yapısına sahip döngü veya koşul ifadelerini gösterir. Bu oranın yüksek olması akıştaki belirsizliği ifade ettiğinden istenmez, düşük olması aracın daha etkin olduğunu ifade eder.

- Çıkma koşulu olan döngü oranı: Çıkma koşulu olan döngülerin oranını ifade eder, tersi durum Cfg'de daha çok belirsizlik olduğunu gösterir. Yüksek olması aracın daha etkin olduğunu gösterir.

- Değişken başına veri akışı: Bu değer aracın kendi analizleri ve daha çok zafiyet analizi için kullanacak araçlar için elde edilmiştir. Değişken başına veri akışı sayısını gösterir. Tablo incelendiğinde Gigahorse'un Vandal ve Porosity'e göre, Elipmoc'un ise Gigahorse'a göre daha etkin bir araç olduğu görülmektedir.

Çizelge 4.3.'te mevcut araçların geliştirildiği programlama dili, geliştirildiği yıl ve araçların bayt koddan dönüşüm adımlarına yer verilmiştir. Sonraki bölümde performansları karşılaştırılacak Dsol, Erays ve Elipmoc için daha detaylı tabloya sonraki başlıkta yer verilecektir.

Çizelge 4.3. Literatürde mevcut araçların dönüştürme adımları

Araç	Dil	Dönüştürme Adımları
Porosity (2017)	C++	Bayt Kod > Assembly > CFG
Vandal (2018)	Python	Bayt Kod > Register tabanlı IR > CFG (html)
Dsol (2018)	Python	Bayt Kod > IR > Solidity Kodu
Erays (2018)	Python	Bayt Kod > Assembly > CFG > Stack based IR > Register based IR > Komutlara göre yığın güncellenmesi (lifting) > Optimizasyon > 3 adres formlu komutlar
Gigahorse (2019)	Python	Bayt Kod > Blok sınırlarının belirlenmesi > Her blok için yığına etkilerinin belirlenmesi > CFG > Global 3 adres formlu IR > Public ve private fonksiyonların sınırlarının belirlenmesi ile global CFG'den lokal CFG ve çağrı grafı çıkarılılr > Tüm fonksiyonlar için parametreler ve dönüş parametreleri çıkarılır > Fonksiyonel 3 adres formlu IR > Analiz > Solidity benzeri yüksek seviyeli dil
Panoramix (2020)	Python	Bayt Kod / Sözleşme adresi > Fonksiyonlar > Hafıza yapısının çıkarılması (tüm sözleşme için) > koşul ifadelerini basitleştirir > disassembly > sözleşmenin json formatı > Solidity benzeri yüksek seviyeli dil.
Elipmoc (2022)	Python	Gigahorse 2.0'dır, farklı olarak işlemsel duyarlılık (transactional sensitivity) ve özel fonksiyonlarının sınırlarının belirlenmesi için bağlama duyarlı yapılandırma süreci eklendi.

Kaynak: Taranan yayınlardan ve kodlardan derlenmiştir.

Tabloda araçlardan en yenisinin Elipmoc olduğu, araçların çoğunlukla Python ile geliştirildiği görülmektedir. En detaylı karşılaştırma Gigahorse 2.0 gibi düşünülen Elipmoc ile Gigahorse arasında bulunmaktadır. Görüldüğü gibi araçlar arasında çeşitlilik fazladır, ileride geliştirilecek araçları da göz önüne alarak daha genel, bayt koddan kaynak koda dönüştüren araçların seçiminde yardımcı uzman sistem uygulaması Bölüm 7.2.'de yer almaktadır.

4.2. Araçların Karşılaştırılması

Araçların karşılaştırılabilmesi için öncelikle veri seti hazırlanmıştır, Ethereum'a kodu yüklenen akıllı sözleşmelerden 10.000 tanesinin kodları, bayt kodları ve adresleri indirilmiştir[2]. Kaynak koda dönüştürme işlemini yapan araçların karşılaştırılmasında kullanılabilecek ölçütler aşağıda bulunmaktadır [13, 14].

[2] Veri seti ve kriterlerle ilgili ek bilgiler https://github.com/kacikalin/gazi adresinde mevcuttur.

Tamlık (completeness / coverage): Dönüştürülen kodun (decompile) orijinal kodu kapsama oranını ifade eder. Bu kriter için kullanılabilecek parametreler; bir akıllı sözleşme içinde dışarıdan çağrılabilecek tekil fonksiyonların imzaları ile olay (event) tanımlarıdır.

Kesinlik (precision): Dönüştürülen kodun (decompile) daha yüksek seviyeli semantik ile eşleşme oranını ifade eder (if/else/while, fonksiyonların ayrımı, kullanılan operatörlerle benzer sonuç çıkarımı vb.)

Ölçeklenebilirlik (scalibility): Hata vermeden başarılı şekilde dönüştürülen (decompile) akıllı sözleşme oranını ifade eder. Burada time out alan sözleşmeleri dışarda tutarak decompile için gerekli süre dikkate alınır. Time out olan sözleşme sayısı da bir diğer göstergedir. Araçların çalıştırıldığı ortam ve çıktılarının formatları Çizelge 4.4.'te bulunmaktadır.

Çizelge 4.4. İncelenen araçların çalıştırıldığı ortam ve çıktı formatları

Araç Adı	Ortam	Çıktı
Dsol	Windows	Metin
Erays	Windows	Pdf
Elipmoc	Linux	Metin/Excel

Kaynak: Taranan yayınlardan ve kodlardan derlenmiştir.

Bu çalışmada bayt koddan yüksek seviyeli dile dönüştürme işlemi yapılarak performansları karşılaştırılan Dsol, Erays ve Elipmoc'un kullandığı yöntem ve öne çıkan yönlerini içeren tablo Çizelge 4.5.'te yer almaktadır.

Çizelge 4.5. İncelenen araçların yöntemleri ve öne çıkan yönleri

	Dsol	Erays	Elipmoc
Yöntem	Geleneksel [77]	CFG üretimi için ürettiklerini kontrol etmek için Geth [78] [79]	Datalog [74]
Özel Fonksiyonlar Destekleniyor mu?	Hayır	Hayır	Evet
Son Güncelleme Tarihi	2018	2018	2023
Recompile Edilebiliyor mu?	Evet (go to hariç)	Hayır	Hayır
Tanıtıldığı yayında kullandığı veri seti büyüklüğü	27.165 sözleşme	10.387 solidity dosyasından 7.500 tekil bayt kodu	5.000 tekil sözleşme

Kaynak: Taranan yayınlardan ve kodlardan derlenmiştir.

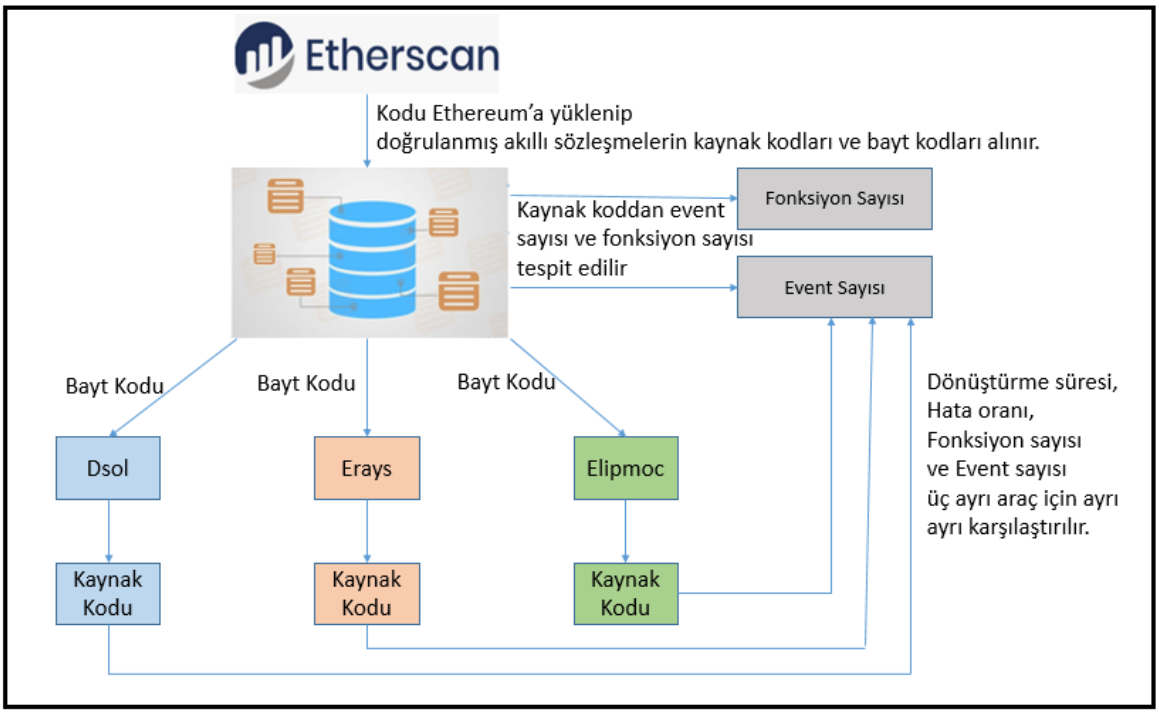
Bu tez çalışmasının öne çıkan özelliklerinden biri, daha önce karşılaştırılması yapılmamış Dsol, Erays ve Elipmoc araçlarının karşılaştırılmasını yaparak, araçlar arasında seçim yapacak uzman sistemi tasarlanmasıdır.

4.2.1. Araçların Karşılaştırılmasında Kullanılan Yöntem

Araçların karşılaştırılması için kullanılan yöntemin adımları şu şekildedir;

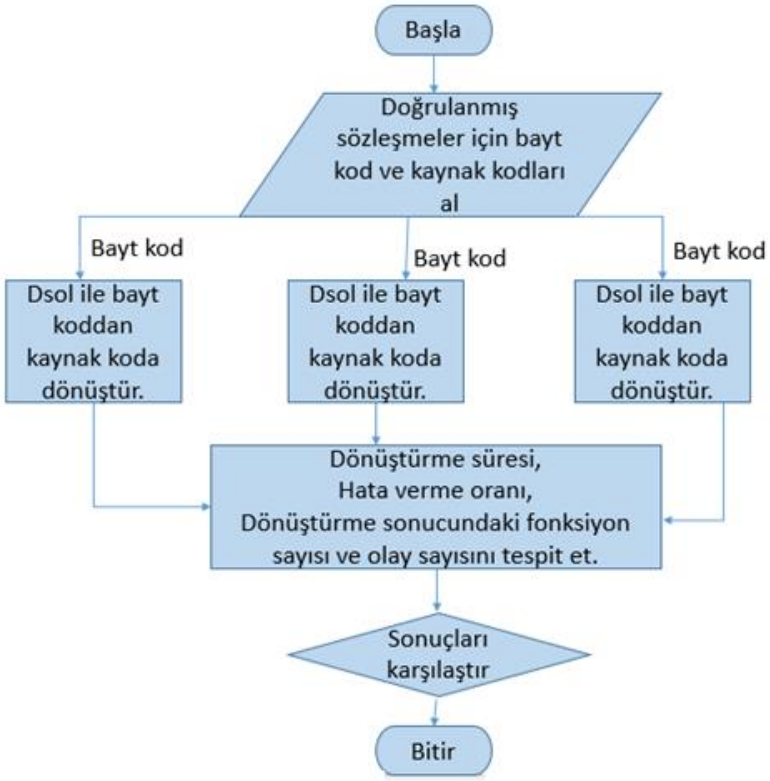
- Daha önce kodu Ethereum ağına yüklenen akıllı sözleşmelerin adresi ve kodları elde edilmiştir.
- Bu akıllı sözleşmelerin ağ üzerinde bulunan bayt kodları alınarak üç araçla kaynak koda dönüştürme işlemi (decompile) yapılmıştır.
- Akıllı sözleşmenin hem ağa yüklenen kodu hem de araçların dönüştürdükleri kaynak koda yakın kodlar elde edildiği için bu ikisi karşılaştırılarak tamlık, kesinlik ve ölçeklenebilirlik karşılaştırılması yapılabilmiştir.
- Tamlık için ağdaki kaynak kod ile dönüştürmeden sonra elde edilen kodun public fonksiyon sayıları ile olay (event) sayıları karşılaştırılır.
- Ölçeklenebilirlik için hata vermeden başarılı şekilde dönüştürme işlemi yapılan akıllı sözleşme oranı tespit edilir.
- Kesinlik için tüm araçlarda kullanılabilecek geçerli kriterler mevcuttur ancak hepsini kapsayacak bir ölçüm yapılamamıştır. Bu kriter Elipmoc'ta sonuçları iyi çıkmayan akıllı sözleşmelerin tespiti ve bu sözleşmelerin Ethervm.io üzerinde test edilebilmesi için kullanılmıştır. Gigahorse ve gelişmiş hali Elipmoc kendi içinde analiz araçlarını da bulundurmaktadır.

Araçların karşılaştırılması için geliştirilen yazılımın adımlarını gösteren şema Şekil 4.8.'de görülmektedir.



Şekil 4.8. Bayt koddan kaynak koda dönüştürme yapan araçların karşılaştırılması

Geliştirilen yazılımın akış şeması Şekil 4.9'da görülmektedir. Adımları ve akışı belirtilen yönteme göre elde edilen sonuçlar Çizelge 4.6.'da görülmektedir. Test edilen akıllı sözleşme sayısı 10.000'dir. Bunların içinde Solidity ile yazılanların sayısı 9.914 iken Solidity versiyonları ise 0.4.7 ile 0.8.6 arasındadır.



Şekil 4.9. Bayt koddan kaynak koda dönüştüren araçların karşılaştırmasının akış şeması

Çizelge 4.6. Araçların Karşılaştırılması

Araç Adı	Tespit Edilen Ortalama Public Fonksiyon Sayısı	Tespit Edilen Ortalama Olay Sayısı	Hata Vermeden Dönüştürme Yapılan Sözleşme Sayısı	Dönüştürme İçin Gereken Ortalama Süre (s)
	Var Olan:20.255	Var Olan: 3.913	Test edilen: 9914	
Dsol	3.637	Üretemiyor	5.348	0.185
Erays	14.317	Tespit edilemiyor	838	0.061
Elipmoc	19.370	3.418	9.705	7.575

Kaynak: Araçlar ayrı ayrı çalıştırılarak sonuçlar elde edilmiştir.

Çizelgede görüldüğü gibi en iyi dönüştürme oranına ve public fonksiyon sayısına en yakın değere sahip araç Elipmoc'tur. Elipmoc yayınında detaylarına yer verilen işlem içerik duyarlılığının (transactional context sensitivity) bir özelliği, analiz boyunca public fonksiyonu yani işlemin giriş noktalarını tutmasıdır. Bu aynı zamanda bir sözleşmedeki aynı public fonksiyonların çağrılmasının ayrı ayrı analiz edileceği anlamına gelir. Elipmoc, bağlama duyarlı private fonksiyonlarının yeniden yapılandırılması sürecini ve fonksiyon parametreleri ve dönüş değerlerinin tespitinde yola duyarlı ve yol birleştirme yöntemi ile ölçeklenebilir bir algoritma sağlamaktadır. Hata vermeden dönüştürme yapılan sözleşme oranında da en iyi sonuç Elipmoc için alınmıştır. Bayt koddan kaynak koda dönüştürmeye ihtiyaç olduğunda Elipmoc'un kullanılması, hata alınması durumunda Ethervm.io'nun web tarayıcısı üzerinden kullanılabilir.

5. ÖZEL BLOK ZİNCİR AĞLARI

Açık blok zincirinde inceleme yapmak blok arama, akıllı sözleşme tespiti ve işlem detaylarını görüntülemenin görece kolay olmasından dolayı mevcut araçların da yardımı ile inceleme yapmak mümkündür. Ancak özel ağlar için, Etherscan benzeri bir kullanıcı arayüzü mevcut değildir. Bu nedenle, Literatürde yeterli bilgiye rastlanmadığı tespit edilen ve önemli olduğu değerlendirilen aşağıdaki sorular bu bölümde incelenecektir:

1. Bir özel ağ nasıl oluşturulabilir?
2. Özel ağ üzerinde akıllı sözleşmeler nasıl çalıştırılacak?
3.Akıllı sözleşmelerin kaynak kodları, sözleşmeyi ağa yükleyen düğümler dışındaki düğümler tarafından nasıl tespit edilecek?

5.1. Özel Ağ Geliştirme Ortamları

Özel ağ geliştirmek için bulunan ortamlar ve özelliklerine bu bölümde yer verilmektedir.

5.1.1. Truffle Suite

Truffle suite, Evm kullanan blok zincirleri için geliştirme ve test ortamı sunar. Bu geliştiriciler için kolaylık sağlar. Akıllı sözleşmelerin derleyip ağa yüklenmesine, bayt yönetimine imkân tanır. Kesme noktaları (breakpoint), değişken analizi ve adım adım ilerletme işlevleri ile gelişmiş hata ayıklamaya imkân oluşturur. Harici komut dosyası (script) çalıştırmaya izin verir, akıllı sözleşme ile doğrudan iletişim sağlayabilmek için konsolu bulunur. Otomatik uyguladığı akıllı sözleşme testleri ile geliştirmelerin daha hızlı yapılabilmesini sağlar. ERC 190 standartlarını kullanarak NPM [80] ile paket yönetimini destekler. Açık ve özel ağlarda akıllı sözleşmelerin yönetimine yardımcı olur. Ağa yükleme ve dağıtım için komut dosyası yazılabilen bir çerçeve sunar [81].

Bir Truffle projesi oluşturulduktan sonra istenilen proje taslağı ve örneklerinin olduğu “box”lardan biri seçilerek kullanılır. Truffle kurulduğunda contracts klasöründe akıllı sözleşmeler, migrations klasöründe betik dilleriyle yüklenebilecek akıllı sözleşmeler, test klasöründe test dosyaları ve truffle.js dosyasında konfigürasyon ayarları bulunur.

Bir akıllı sözleşmeyi test etmek için “truffle test ./test/sozlesmeadi.sol” komutu ile sözleşmenin beklendiği gibi çalışıp çalışmadığı kontrol edilir. Sözleşme derlenmek istenirse “truffle compile” komutu kullanılır.

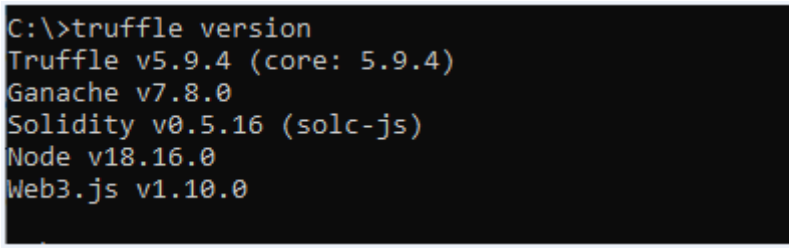
İncelenen bilgisayarda truffle-config.js dosyası varsa, o dosya içinde yazan host, port ve network numaralarına dikkat etmek gerekir. Çünkü bir akıllı sözleşme “truffle develop” komutuyla ağa yüklenmek istenirse, yükleneceği ağ bilgileri bu konfigürayon dosyasında tutulur.

Bir sözleşme geliştirilirken etkileşimleri görmek için hali hazırda aktif bir truffle komut penceresi varsa ikinci bir komut satırı açılır ve “truffle develop –log” komutu kullanılır. Bu komutla ağda meydana gelen RPC (uzaktan prosedür çağrısı) etkileşimi görüntülenir [81].

5.1.2. Ganache

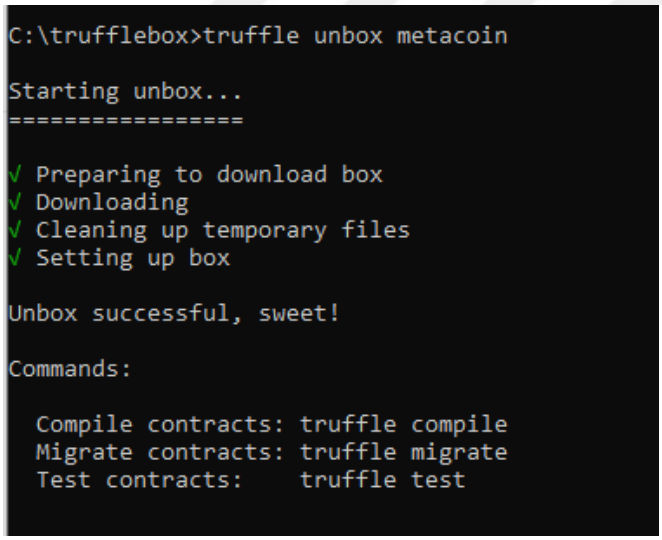
Ganache, Truffle ekosisteminin bir parçasıdır. Ethereum ve Filecoin [82] için dağıtık uygulama geliştirme için (dApp) kişisel/özel bir blok zincirdir. Ganache ile dağıtık uygulamalar güvenli ve kararlı bir ortamda geliştirilip dağıtılır ve testleri yapılabilir. Ara yüzü ve komut satırı ile kullanılabilir. Ethereum ve Filecoin teknolojisini destekleyen bir masa üstü uygulamasıdır. Truffle ve Ganache ile özel ağ oluşturma adımları şu şekildedir. İki ayrı bilgisayar veya sanal makinede kurulacağı için bu ayarlar aynı port konfigürasyonu ile yapılabilir ancak aynı bilgisayarda birden fazla düğüm oluşturulacağı durumda farklı port numarası ile oluşturulmalıdır.

Node.js ve Truffle paketlerinin kurulumundan sonra Şekil 5.1.’de görünen komut ile Truffle içindeki bileşenlerin versiyonları tespit edilebilir. Bu çalışmada kullanılan bileşenlerin versiyonları komutun çıktısı olarak görülmektedir.



Şekil 5.1. Truffle versiyonu

Kurulumun ardından bir proje oluşturmak için istenilen dizinde bir klasör oluşturulur, o klasöre komut satırında gidilerek hangi eklentiler ile proje oluşturulmak isteniyorsa o eklenti için unbox işlemi yapılır (Şekil 5.2.). Farklı özellikler için özelleştirilmiş çeşitli eklentiler mevcuttur. Burada akıllı sözleşme eklentilerini içeren metacoin seçilmiştir. Bu komut çalıştırıldığında belirlenen klasör altında build, contracts, migrations ve test klasörlerinin oluştuğu görülebilir. Bir akıllı sözleşme derlenmek istendiğinde contracts klasörü altına sol uzantısıyla kaydedilir. Derlemek için proje klasöründe komut satırından "truffle compile" yazılarak çalıştırılır. Derleme sonrasında komutun çıktısı Şekil 5.3.'te bulunmaktadır. Derlenen akıllı sözleşmeyi ağa yüklemek için "truffle migrate" komutu kullanılır. Yüklerken hangi ağ olduğunu belirtmek için –network parametresine truffle proje dosyasında ayarların olduğu js dosyasında belirtilen ağ numarası referans olarak verilebilir. Ganache'ın sağladığı arayüzde bilgisayarda bulunan ağdaki hesaplar, bloklar, işlemler, sözleşmeler, olaylar ve loglar görüntülenebilmektedir. Ancak aynı özel ağa bağlı diğer düğümlerle ilgili bilgiler için Ganache'ın komut satırı kullanılmalıdır, bu bilgilerin elde edilmesi için gerekli uygulamaya bu bölümün sonunda yer verilecektir.

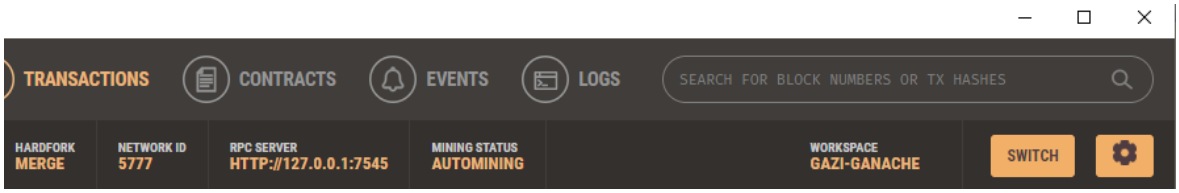


Şekil 5.2. Truffle unbox işlemi

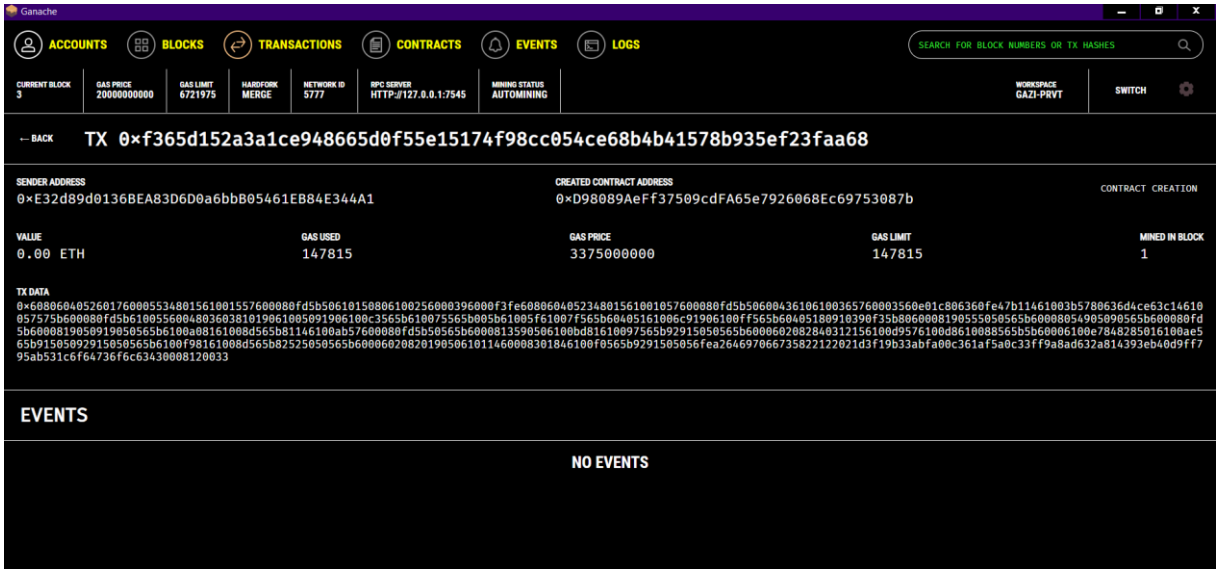
```
C:\trufflebox>truffle compile

Compiling your contracts...
===========================
√ Fetching solc version list from solc-bin. Attempt #1
√ Downloading compiler. Attempt #1.
> Compiling .\contracts\Getset.sol
> Artifacts written to C:\trufflebox\build\contracts
> Compiled successfully using:
   - solc: 0.8.13+commit.abaa5c0e.Emscripten.clang
```

Şekil 5.3. Truffle compile işlemi

Ganache arayüzü üzerinde bir çalışma ortamı (workspace) olarak Truffle'ın ayar dosyaları referans gösterilerek o ağdaki işlemler, akıllı sözleşmeler, hesaplar ve sunucu ile portları görüntülenebilmektedir. Ganache UI üzerinde hangi ağda olduğu ve lokalde hangi portu kullandığı Şekil 5.4.'te görünmektedir. Şekilde görünen Swithch butonu ile gelen ekranda farklı ayar dosyaları seçilerek o ayarların olduğu ağlar görüntülenebilir. Transaction sekmesinde Contract Creation olarak görünen işlemlerin input data alanında bulunan hex kodu bu ağa yüklenen akıllı sözleşmenin bayt kodudur (Şekil 5.5.) Bu bayt kodu dördüncü bölümde test edilen araçlarla kaynak koda dönüştürülebilir.



Şekil 5.4. Ganache UI üzerinden ağ numarası ve port tespiti



Şekil 5.5. Ganache UI üzerinden akıllı sözleşme bayt kodu görüntüleme

5.1.3. Go-Ethereum (Geth)

Go-Ethereum kısaca Geth, Go dili ile geliştirilmiş Ethereum istemcisidir. Geth kurulumu yapıldıktan sonra bir blok zincirin başlaması için gerekli olan ayarların bulunduğu genesis dosyası oluşturulur (genesis.json). Bu dosyada bağlanılacak ağa verilecek tekil numaraya dikkat edilmelidir. Ağların tekil numaraları listelenmektedir [83], özel bir ağ oluşturulacaksa kullanılmamış bir sayı seçilmelidir. Blok zinciri oluşturacak ilk düğümü başlatmak için gerekli komut "geth init –datadir dugumadi genesis.json" şeklindedir.

İlk düğüm oluşturulduktan sonra belirtilen ağda düğümün başlatılması için istenilen parametreler girilerek şu şekilde çalıştırılır; "geth –datadir dugumadi –networkid 230623 –http –allow-insecure-unlock –nodiscover". Bu komut ile konfigürasyon dosyasında ağ numarası 230623 ile belirtilen olan özel ağda düğüm başlatılmış olur.

Web uygulamalarının ve metamask gibi cüzdan uygulamalarının erişimi için http izinlendirilmesinin yapılması gerekmektedir. –allow-insecure-unlock ile hesap kilidinin http üzerinden gelen işlemlerle açılmasına izin verilir. Bu komutun çıktısında "IPC endpoint opened" ifadesinin karşısında geth.ipc dosyasının yolu bulunur. Bir sonraki adımda bu dosya, yolu ile birlikte parametre olarak verilir. Yeni bir terminal açılarak javascript konsoluna erişmek için "geth attach \\.\pipe\geth.ipc" komutu çalıştırılır. Gelen javascript konsolunda admin ve düğüm işlemleri için mevcut komutlar kullanılabilir [84]. Örneğin "admin.nodeInfo" komutu ile o düğümle ilgili bilgiler kapsamlı şekilde gelir, enode bilgisi de bu alanlardandır. Bir bilgisayardaki özel bir ağa başka bir bilgisayardaki düğüm eklenmesi gerektiğinde öncelikle bu iki bilgisayarın birbiri ile aynı ağda olmalıdır. Ip adreslerine ping atılabiliyorsa enode bilgisinde 127.0.0.1 yazan kısma ip adresi yazarak "admin.addPeers" komutu ile eklenir. Ağa bağlı düğüm bilgileri görülmek istendiğinde "admin.peers" komutu kullanılır. Geth'in özel ağı çalışırken örnek ekran görüntüsü Şekil 5.6.'daki gibidir.

Şekil 5.6. Geth çalışma durumu

5.1.4. Hardhat

Hardhat, Nomic Labs tarafından geliştirilmiş Ethereum geliştirme ortamıdır. Akıllı sözleşmeler için derleme, çalıştırma ve test imkânı sunar [85]. Önemli özellikleri; solidity dili için hata ayıklama yapması, başarısız işlemler için hata mesajı vermesi ve yığın izlemeyi sağlaması sayılabilir. İçinde geliştirmeye yönelik olarak yerel bir Ethereum ağının olması ile öne çıkmaktadır. Hardhat'in temel işlevleriyle esnek bir şekilde çalışan komut satırı arabirimi mevcuttur (CLI). Hardhat akıllı sözleşme geliştirme ve hata yönetiminde Truffle'a göre daha esnektir [86].

Hardhat ile proje oluşturma, sözleşme derleme, test etme ve ağda çalıştırma için izlenmesi gereken adımlar şu şekildedir;

İstenilen bir dizinde proje klasörü oluşturulduktan sonra, o dizindeyken "npx hardhat" komutu ile proje oluşturulur. Proje klasöründe contracts, scripts ve test klasörleri ile ayarlamaları yapmaya yardımcı konfigürasyon dosyaları bulunur. Sözleşmeleri derlemek için "npx hardhat compile" komutu kullanılır. Ardından testler için "npx hardhat test" komutu çalıştırılır. Derlenen kodun çalıştırılması için "npx hardhat run" komutu kullanılır, parametre olarak verilen dosya scripts klasörünün altında bulunacaktır. Lokalde özel hardhat ağında bir düğüm "npx hardhat node" komutu ile oluşur, bu komutta –network parametresi ile düğümün hangi ağda oluşacağı belirtilebilir.

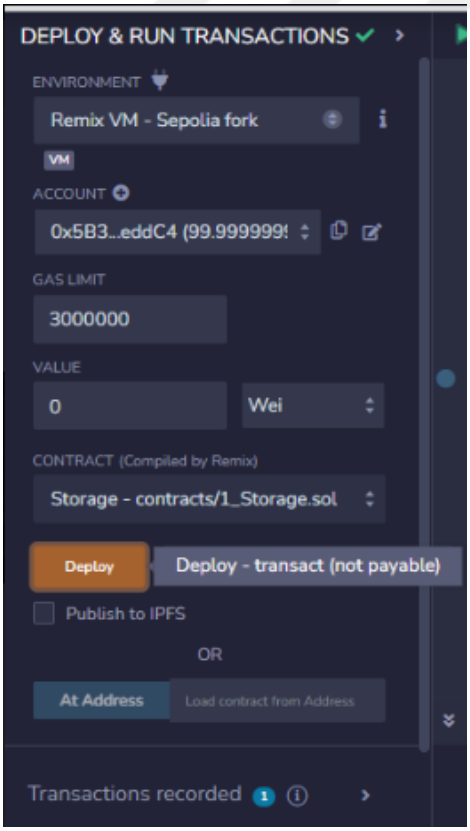
5.1.5. Remix IDE

Remix, soliditiy ile akıllı sözleşme yazma, derleme, ağa yükleme ve etkileşime imkan tanıyan web tarayıcısı tabanlı IDE'dir [87]. Örnek bir akıllı sözleşme solidity dili ile yazıldıktan sonra derlenir ardından istenilen ağa yüklenir (Şekil 5.7.). Ağ seçme kısmında lokal ağların nasıl ekleneceği belirtilmiştir, test ağları da kullanılabilir durumdadır.

Yüklenen akıllı sözleşmenin abi kodu ve yükleme yapacak bayt kodu bu aşamada elde edilmektedir. Etherscan üzerinde geliştirici tarafından kaynak kodu yükleme ve akıllı sözleşme doğrulama işlemi yapılacağı zaman bu abi kodu kullanılabilmektedir.

Yüklenen akıllı sözleşmenin fonksiyonları kullanılacağı zaman Remix IDE sayfasında tarayıcıda Metamask uzantısı aktifleşerek gerekli işlem ücreti cüzdan adresinden düşer.

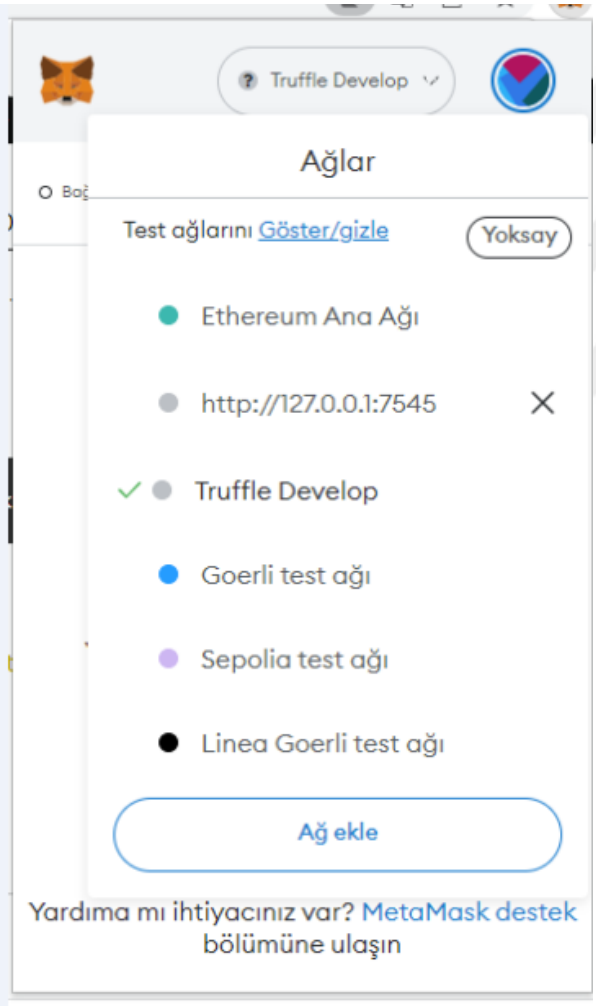
Özel bir ağa akıllı sözleşme Remix IDE üzerinden geliştirilmişse tarayıcı geçmişi silinmediği sürece sayfada görüntülenecektir. O nedenle yapılacak denetimlerde Remix IDE sayfasının incelenmesi faydalı olacaktır.



Şekil 5.7. Remix IDE üzerinde test ağına yükleme örneği

5.1.6. Metamask

Metamask, blok zincir uygulamalarını kullanmak için geliştirilen bir cüzdan uygulamasıdır. Tarayıcılar üzerinde uzantı olarak çalışması onu kullanışlı kılar. Blok zincir etkileşiminin olduğu sitelerde uzantı yardımıyla işlem onaylama (akıllı sözleşme ağa yükleme vb.), değer transferi, akıllı sözleşme etkileşimi gibi işlemler kolay ve güvenli şekilde yapılır [88]. Metamask üzerinde cüzdanlar bulunduğu ağa göre filtrelenir. Yerel ağlar da metamask üzerinde eklenerek görüntülenebilir (Şekil 5.8.)



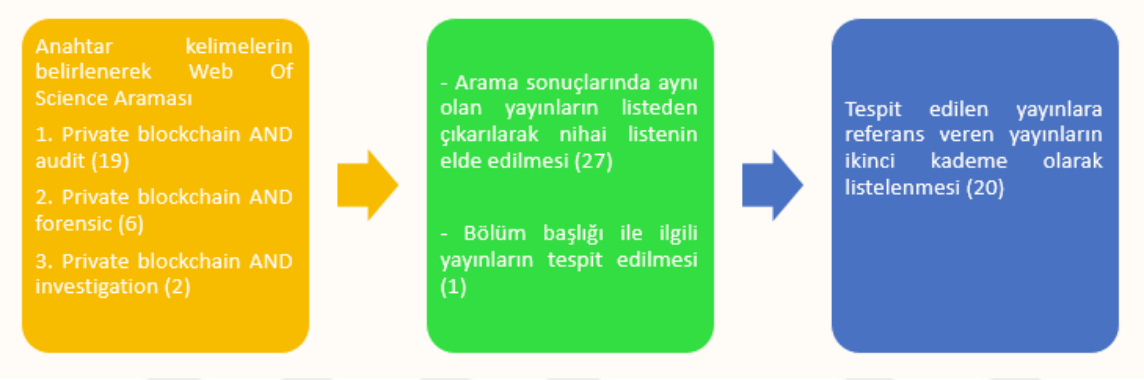
Şekil 5.8. Metamask ağları

Araçlardan öne çıkan Hardhat ve Truffle içinde, Hardhat akıllı sözleşme geliştirme ve hata yönetiminde Truffle'a göre daha esnektir [74]. Metamask gibi bir cüzdan eklentisi kullanmak gerekmektedir, Remix IDE ise tarayıcı üzerinden kolay geliştirme sağlaması

yönüyle avanajlıdır. Yapılacak incelemelerde karşılaşılabileceği için güncel araçların kullanımı ve kurulumlarının bilgisayar ve telefonlarda bıraktığı izlerin neler olduğu takip edilmelidir. Bir bilgisayarda yapılacak incelemelerde önerilen prosedür listesinde tarayıcıda RemixIDE'nin açılması ve onun üzerinden geliştirilmiş akıllı sözleşmeleri görüntüleme adımı da mevcuttur.

6. ÖZEL BLOK ZİNCİR AĞLARINDA İNCELEME

Bu bölümde, özel ağlarda inceleme yaparken dikkat edilmesi gerekenlerin bulunacağı önerilen prosedür listesi ile incelemeyi daha etkin hale getirecek uygulama bulunmaktadır. Web of Science [89] veri tabanı üzerinde yapılan "private blockchain AND audit", "private blockchain AND forensic", "private blockchain AND investigation" aramaları sonucunda sırasıyla 19, 8 ve 3 adet çalışma olduğu görülmüştür. Literatür taramasında kullanılan yöntemin akış diyagramı Şekil 6.1.'de mevcuttur, adımlarda elde edilen yayın sayısı parantez içinde belirtilmiştir. Bu kısımda toplamda 47 yayın incelenmiştir.



Şekil 6.1. Literatür taramasında kullanılan yöntemin gösterimi

Bunlardan aynı olan çalışmalar çıkarılarak elde edilen 27 adet yayının içeriği ile ilgili özet bilgiler aşağıdaki çizelgede (Çizelge 6.1.) bulunmaktadır.

Çizelge 6.1. Literatür Taraması

Kaynak	Çalışma Özeti
Audit-1 [19]	Bilgi Teknolojileri Genel Kontrol (ITGC) süreçlerinde özel ve izinli blok zincirlerde yapılacak denetimler için önerileri bulunuyor. Finans alanındaki incelemeler için hazırlanmıştır. Bayt kod dönüşümü ya da işlem takibi ile ilgili bir husus yok.
Audit-2 [90]	Geleneksel audit ile blok zincir üzerinde yapılacak denetimleri karşılaştırır ve blok zincir incelemeleri için bir akış önerir (Şekil 6.2.).
Audit-3 [91]	GDPR denetimleri için özel blok zincir önermiştir. Veri kontrolörü ve verinin sahibi için veri üzerindeki erişimlerin kontrolünü kolaylaştırdığını belirtiyor.
Audit-4 [92]	ENSAT (the European Network for the Study of Adrenal Tumors) kanserli hastalar hakkında klinik, fenotipik ve genetik bilgilerin bir üretim deposudur. Verilerin denetimi ve kayıtların erişimini daha güvenli hale getirmek için blok zincir tabanlı hale getirilmesi önerilmiştir.
Audit-5 [93]	Sanal makineler üzerinde farklı özelliklerde ethereum ağları kurularak Dos

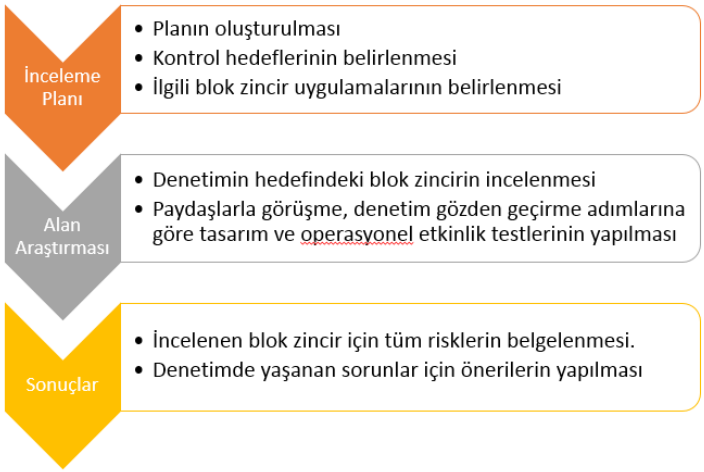
Çizelge 6.1. Literatür Taraması (Devam)

Kaynak	Çalışma Özeti
	karşısındaki dirençleri karşılaştırılmış, özel ağlarda varsayılan ayarlar kullanıldığında Dos'a karşı dirençli olamayabileceği gösterilmiştir.
Audit-6 [94]	Birden fazla kişi tarafından yürütülen denetimlerde veriler incelenirken gizliliğin sağlanması için konsorsiyum blok zinciri önerilmiştir.
Audit-7 [95]	GDPR'a yönelik olarak kişisel verilerin işlenmesinde şeffaflığı sağlamak için özel blok zincirlerini öneriyor.
Audit-8 [96]	Kendi elektriğini üreten kullanıcıların (microgrid trading) olduğu sistemde, düşük işlem verimliliği, yüksek piyasa bakım maliyeti, işlemlerde şeffaflığın sağlanamaması, kullanıcının gizliliğinin sağlanamaması gibi sorunlar olduğu belirtiliyor. Bu problemleri gidereceği düşünülen hibrit (özel ve konsorsiyum) bir blok zincir altyapısı öneriliyor. Özel blok zincirde kullanıcı tanımlamaları ve kullanıcının işlemleri; konsorsiyum blok zincirde ise işlem bilgilerinin tutulması öneriliyor.
Audit-9 [97]	Elektronik cihazların israfı oluşturan en önemli kalemlerden biri olan ömrü biten telefonların geri dönüştürülememesinin önündeki önemli engellerden birinin gizlilik endişesi olduğu belirtiliyor. Eşyaların dönüşümü ve yeniden kullanımını ifade eden RL (reverse logistics) için özel blok zincir ve akıllı sözleşme kullanımı önerilerek süreci izlemenin kolaylaşacağı ifade edilmektedir.
Audit-10 [98]	Varlıkların devri için özel blok zincir altyapısı ile çözüm önerir.
Audit-11 [99]	IoMT (Internet of Medical Things) medikal Iot cihazları ve medikal kayıtlar için blok zincir temelli yetkilendirme çerçevesi önerilmiştir.
Audit-12 [100]	Blok zincirin temellerinin anlatıldığı kitap bölümüdür.
Audit-13 [101]	MEC (mobil edge computing) ile ilgilidir, kaynak kıtlığı problemine çözüm olarak Iot cihazlarından kaynağa ihtiyacı olmayandan kaynağın alınıp ihtiyacı olana iletilmesi gibi çözümü mevcuttur (task offloading). Problem hangi MEC sağlayıcı kullanıcı verilerine erişebilir ve MEC servislerinin kısıtlı olmasından kaynaklanmaktadır. Çözüm olarak Iot cihazlarının verilerini drone ile blok zincire aktarmak önerilmiştir. Bu şekilde şeffaflık ve erişim denetimi sağlanabilir.
Audit-14 [102]	Çalışmada SC kısaltması Supply Chain'e, SmCs Smart contract'a karşılık geliyor. Tedarik zincirinde bulunan tüm malzemeler ve yarı mamüller için değiştirilebilir (mutable) belirteç (token) kullanarak blok zincir tabanlı ve akıllı sözleşme destekli tedarik sistemi izleme mekanizması için çerçeve oluşturulmuştur.
Audit-15 [103]	Talep taraflı yönetim (Demand side management) kullanıcıların enerji tüketimini düzenlemek için bir araç sunar. Ancak müşteri ile hizmet arasında bağ olması gerekliliği gizlilik ihlaline neden olabilir. Bu soruna çözüm olarak gürültü ekleme, şifreleme entegrasyonu veya ek donanımlara dayalı yöntemler önerilmişse de maliyeti artırdığı için ve verilerin bütünlüğünü etkilediği için etkin bir çözüm değildir. Müşterileri bilgisinin anonimliğini sağlayan blok zincir çözümleri olsa da hem anonimliği hem veri bağlantısını sunan çözüm henüz bulunmamıştır. Bu çalışmada özel blok zincir tabanlı yeni bir ağ stratejisi ile bunu sağlayabildiğini ve test parametrelerini içerir.
Audit-16 [104]	GDPR'ı sağlamaya yönelik blok zincir altyapısı önerir ve gereksinimlerini detaylı şekilde belirterek karşılar.
Audit-17 [105]	GDPR'ı sağlamaya yönelik, Multichain [106] ile özel blok zincir altyapısı önerir.

Çizelge 6.1. Literatür Taraması (Devam)

Kaynak	Çalışma Özeti
Audit-18 [107]	Geleneksel yapılardan tedarik zinciri, şirketler için kaynak planlama sistemlerinin vb. blok zincirin hangi özelliği ile karşılanabildiğine değinilmiştir. Blok zincirinin değişmezlik ve yalnızca ekleme özelliğinin, kamu idarelerin, denetleyiciler, regülatörler ve benzeri aktörler için önemli bir araç olduğunun altı çizilmiştir. Yayının katkısı blok zincirleri sahip olduğu farklı özelliklere göre sınıflandırmasıdır.
Audit-19 [108]	Akıllı evlerde anomali tespiti için yapay zeka destekli ve blok zincir tabanlı bir uygulama sunar.
Forensic-1 [109]	Adli tıp alanında toplanan verilerin Hyperledger özel ağında tutulabilmesi için önerilen modeli içerir.
Forensic-2 [110]	Bulut sistemlerde adli bilişim sürecinde kanıtların Hyperledger Fabric blok zincirinde tutulmasını öneren çalışmadır. Çalışmada kontainer kullanan bulut servislerinde adli bilişim delillerinin bulunduğu konum, veri merkezindeki diğer kiracılar, siyasi konular gibi nedenlerle elde edilmesinde ve değiştirilmeden tutulmasının zor olduğu belirtilmiştir. Bulut hizmet sağlayıcılar ile müşterileri arasındaki güveni kolaylaştıracağı düşünülen önerilen yapı ile verilerin hem aktarılması hem de depolaması sırasında dijital kanıtların bütünlüğünün güvence altına alınacağı ifade edilmiştir.
Forensic-3 [111]	Çalışmada mahkemelere sunulacak dijital delillerin özel blok zincirlerde tutulması önerilir.
Forensic-4 [112]	Big data ile çalışan sistemleri özel blok zincir ağına bağlar, sistemde oluşan tüm olayları (event) bu özel ağa kaydeder ve sonrasında makine öğrenmesi süreçleri ile bu verileri analiz eder.
Forensic-5 [113]	Sosyal mühendislik saldırılarından vishing saldırılarına karşı etkili blok zincir tabanlı korunma modeli geliştirilmiştir.
Forensic-6 [114]	Entegre devre sistemlerinde parçalarının hızlı ve esnek bir şekilde tanımlanabilmesi için blok zincir üzerinde çalışan uygulama ve akıllı sözleşmelerden oluşan bir model önerilir.
Investigation -1 [115]	GDPR uyumlu çalışacak, sağlık verilerinin gizliliğini ve bütünlüğünü sağlamak için geliştirilmiş özel blok zincir tabanlı bir uygulama önerilir.
Investigation -2 [116]	İnşaat güvenliği sağlama sürecinde farklı kanallardan toplanana çok çeşitli verinin toplanmasını, yönetilmesini ve dağıtılmasını blok zincir tabanlı bir yapı ile daha şeffaf, izlenebilir ve denetlenebilir kılacak model önerilir.

Audit-2 [90] yayınında önerilen bir blok zincir üzerindeki denetim adımları Şekil 6.2.'de bulunmaktadır. Görüldüğü gibi genel ifadelerle öneriler bulunmaktadır.



Şekil 6.2. Blok zincir denetimlerinde önerilen adımlar

Çizelge 6.1.'de listelenen yayınlarda özel blok zincir ağlarında denetim ve incelemelerin nasıl yapılması gerektiğine dair öneri içeren herhangi bir çalışma gözlemlenmemiştir. Listedeki yayınlardan bu bölümde odaklanan konuya en yakın çalışma olan Audit-2 [90] yayınını referans gösteren yayınlar da konuyla ilgili herhangi bir çalışmanın gözden kaçırılmaması için incelenmiştir. Audit-2'yi kaynaklarında gösteren 20 adet yayın bulunmaktadır.

Çizelge 6.2. Literatür Taraması İkinci Aşama

Yayın No	İçerik Özeti
Audit-2.1 [117]	2016-2020 yılları arasında yayınlanan makalelerin sayıları analiz edilerek temel alt konu başlıkları çıkarılmıştır. Bu alanda araştırmaların hangi yöne doğru gittiği sorusunun cevabı aranmıştır.
Audit-2.2 [118]	Akıllı sözleşmelerin dışarıdan gelen veriler ile beslenmesi ve bu verilerde oluşacak güvenlik ihlallerinin akıllı sözleşmenin mantığını devre dışı bırakacağı riski Oracle problemi olarak tanımlanır. Bu çalışmada veri kaynağı olan Oracle'ların AICPA ve PCAOB denetim standartları kapsamında hizmet kuruluşları olarak görülmesi gerektiği savunulur. AICPA ve PCAOB hakkında bilgi için bkz. [119]
Audit-2.3 [120]	Firmaların dijitalleşmesinin muhasebe süreçlerine yaptığı katkı üzerinedir.
Audit-2.4 [121]	Muhasebe şirketlerine denetim süreçlerinde veri analitiğinin ne kadar gerekli olduğuna dair bir çalışmadır, farklı rollerdeki denetçiler için denetim veri analitiğinin kullanımına ilişkin kılavuz sunar.
Audit-2.5 [122]	Değişmez olan blok zincirlerin akıllı sözleşmelerin kullandığı kaynaklar (oracle) aracılığı ile manipüle edilebilmesi paradoksunun etkilerini azaltmak amaçlanmıştır. Çalışma tasarım bilimi araştırması yöntemini kullanan (design science research), iş süreçleri yönetimi modeli önerir. Bu model ile üçüncü parti bir oracle kaynağı olan IoT ile akıllı sözleşme destekli tedarik zinciri içindir.

Çizelge 6.2. Literatür Taraması İkinci Aşama (Devam)

Yayın No	İçerik Özeti
Audit-2.6 [123]	Blok zincir ile muhasebe kayıtları tutmanın getireceği değişiklikleri inceleyen bir çalışmadır. İzlenebilir ve denetlenebilir blok zincirler üzerindeki verilerini kullanacak yapay zeka araçları denetimin etkinliğini artırabilir.
Audit-2.7 [124]	Bu yayında blok zincirin özellikleri harici finansal raporlamadan sorumlu kişiler ve denetçilerin bakış açısıyla değerlendirilmiştir. Mutabakat yapısındaki doğrulama işleminin mali tabloların ya da gerekli denetimin yerini tutamayacağının altı çiziliyor.
Audit-2.8 [125]	Dağıtık muhasebe sözleşmeleri için bir tasarım modeli geliştirir.
Audit-2.9 [126]	Bu çalışma, Bitcoin ve blok zincir için bir vaka çalışması örneğidir. Denetim ve sigortacılık derslerini alan öğrencilerden, e-ticaret sitesi üzerinden satış yapan perakendecinin denetlenmesi istenir.
Audit-2.10 [127]	Büyük veri (BD) ve yapay zeka (AI) alanlarında yapılan, muhasebe denetimi konusundaki çalışmaların mevcut durumunu ve gelecekte evrileceği alanın tespiti için biblio metrik analiz kullanılmıştır. Analiz için VOSviewer yazılımı kullanılmıştır.
Audit-2.11 [128]	2012-2020 yılları arasında akademik ve endüstri literatürünün tarandığı çalışmada yazarlar; vergilendirme, kripto varlıkların/yükümlülüklerin muhasebe işlemleri ve ayrıntılı denetim prosedürleri gibi yeterince çalışılmamış alanlardaki mevcut düzenlemeleri, muhasebe standartlarını, yönergeleri ve olası değişiklikleri tanımlar.
Audit-2.12 [129]	Üçüncü JISC (Journal of Information Systems Conference) 2018 konferansının özel bölümü ve raporlarını içerir, muhasebe bilgi sistemleri alanında bulut bilişim araştırmalarının geliştirilmesine katkı sunmak hedeflenmiştir.
Audit-2.13 [130]	Kripto varlığıyla ilgili denetimlerde rehberlik sağlamak için bu makalenin amaçları; kripto varlık ekosistemindeki çeşitli katılımcıları tespit etmek ve bunların denetlenen kuruluşla olan ilişkilerini göstermek, mali tablo için yeni zorlukları ve riskleri belirlemek ve detaylandırmaktır. Compound [131] ve Brave [132] gibi popüler DeFi platformları için akıllı sözleşmeler üzerinde güvenlik denetimlerini yapan OpenZeppelin [133] gibi şirketlerin oluşturdukları raporları açık olarak paylaştıkları belirtilmiştir.
Audit-2.14 [134]	Akademik taraftaki çalışmaların denetim standartları belirleyen kişilere doğru ve tam olarak aktarılmasının önemini sunan çalışmada tasarım bilimi (design science) yaklaşımı kullanılmıştır.
Audit-2.15 [135]	Pathways Komisyonu'nun mevcut ve gelişmekte olan teknolojileri daha iyi yansıtmak için öğrenme deneyimlerini geliştirmeye odaklanan ek araştırma çağrısına cevap vermektedir. Deneyimsel bir öğrenme yaklaşımı (experiental learning approach) kullanarak muhasebe öğrencilerine blok zincir teknolojisi ve mekanizmalarının yanı sıra bunun bitcoin dışındaki etkileri hakkında kavramsal bir anlayış kazandırmayı amaçlayan interaktif bir öğrenme etkinliğidir.

Çizelge 6.2. Literatür Taraması İkinci Aşama (Devam)

Yayın No	İçerik Özeti
Audit-2.16 [136]	Denetim aşamalarında blok zincirin zaman damgası ve çifte harcamayı önleme yönteminin önemli iki özellik olduğu belirtilmiştir. Bu çalışmada blok zincirin kabul edilmesinin yaygınlaşabilmesi için denetçilerin bilgi ve farkındalıklarının güçlendirilmesi gerektiği sonucunun da çıktığı anket çalışmasıdır.
Audit-2.17 [137]	Bu çalışma, (1) bir blok zincirinin tasarımı ve yönetişimi, (2) varlık oluşturma, (3) varlık transferi ve (4) varlık kullanımdan kaldırma dahil olmak üzere dört aşamada maddi varlıkların sahipliğini ve menşeini izlemek için blok zincirin kullanımını inceler. Bu çalışma, bir konsorsiyumun maddi varlıkların mülkiyetini ve menşeini izlemek için izin verilen blok zincirinin güvenilirliğini değerlendirmek için bir kılavuz olarak kullanılabilir.
Audit-2.18 [138]	Muhasebe işlemlerini yapan birimler için, kripto paralarla ilgili ele alınması gereken araştırma soruları belirlenmiştir. Çalışmada bu araştırma soruları; teknolojinin tasarımının, benimsenmesinin ve uygulamasının geliştirilmesi için kripto para birimlerinde muhasebe araştırmalarını teşvik etmek ve genişletmek amacıyla Rogers'ın yeniliğin yayılması (Rogers' theory of diffusion of innovation) teorisi [139] kullanılarak kategorize edilmiştir.
Audit-2.19 [140]	Bu çalışma, kar amacı gütmeyen kuruluşların denetiminde ortaya çıkan veri madenciliği tekniklerinin, özellikle kümelemenin (clustering) kullanımına ilişkin literatürü araştırması içerir. Kümeleme, denetçilere nereye dikkat etmelerini söyleyebilir. Düzenli yapıyı ve anormallikleri görmek için maddi doğrulama prosedürlerinde işlem düzeyindeki verilere kümeleme de uygulanabilir. Bu tür veriler, modeller elde etmek ve aykırı değerleri tespit etmek için diğer yıllara göre kümelenebilir. Bu çalışmada k-means ve hiyerarşik (hierarchical) iki farklı kümeleme yöntemi kullanılmıştır.
Audit-2.20 [141]	Bankaların onay süreçlerinin değerlendirilmesi ile ilgili bir çalışmadır, blok zincir elektronik kayıtların yetkilendirilmesinde blok zincir kullanımının ele alınabileceği belirtiliyor.

Yapılan literatür taramasında özel blok zincirlerde denetim ve incelemeler için önerilen prosedür listesine rastlanmamıştır. Bu tez çalışması bu alandaki boşluğu gidermek için yapılmıştır. Bu bölümde önerilen prosedür listesi ile manuel incelemeyi daha etkin hale getirebilecek aracın tasarımı ve uygulaması bulunacaktır. Anahtar kelimeler sütununda "private blockchain" ile birlikte aranan anahtar kelime bulunmaktadır. Literatür taramasında incelenen çalışmaların konulara göre dağılımı Çizelge 6.3.'te bulunmaktadır. Gerçek problemlere uyarlama başlığında blok zincir altyapısının farklı sektörlerde kullanımına örnek çalışmalar yer almaktadır. Güvenlik başlığı Dos gibi siber güvenlik konularını içermektedir.

Çizelge 6.3. İncelenen yayınların konuları ve anahtar kelimeleri

Yayın Referansları	Konusu	Anahtar kelimeler
[100], [117], [128], [138]	Derleme	Audit
[19], [90], [94], [118], [126], [130], [134], [136], [140]	Denetim Süreçleri	Audit
[91], [95], [104], [115]	GDPR	Audit, Investigation
[97], [103], [109], [110], [115]	Veri Gizliliği	Audit, Forensic, Investigation
[92], [94], [101], [110]	Veri Erişimi Denetimi	Audit, Forensic
[94], [96], [98], [101], [102], [103], [123]	Veri Şeffaflığı	Audit
[99], [101], [108], [122]	IoT	Audit
[120], [121], [123], [124], [125], [127], [129], [130], [135]	Mali Tablo İncelemesi	Audit
[109], [111]	Delil Zinciri	Forensic
[96], [97], [98], [102], [103], [107], [114], [116], [120], [137], [141]	Gerçek Problemlere Uyarlama	Audit, Forensic, Investigation
[93], [113], [129]	Güvenlik	Audit, Forensic
[92], [99]	Sağlık	Audit

Çizelgede de görüldüğü gibi, taranan yayınlar içinde en çok blok zincir uygulamalarının gerçek problemlere uyarlanması, mali tablo incelemeleri ve denetim süreçleri için öneriler içeren çalışmalar bulunmaktadır. Yayınlarda özel blok zincirler üzerinde adli bilişim incelemesine yönelik önerilen prosedür listesine rastlanmamıştır.

6.1. İncelemelerde İzlenmesi Önerilen Prosedür Listesi

İnceleme yapacak kurum ya da kişiler için oluşturulacak kontrol listesi bu tez çalışmasının önemli çıktılarından biridir. Aşağıda özel ağlarda yapılacak incelemelerde izlenmesi önerilen adımlar bulunmaktadır.

- Bilgisayar üzerinde dizin araması Truffle, Geth ve Hardhat'in kullanılması halinde oluşacak klasörler göz önünde tutularak yapılmalıdır. (Contracts klasörünün, konfigürasyon dosyalarının aranması vb.)

- İncelenen bilgisayarda tarayıcıda bulunan cüzdan eklentileri incelenerek mevcut ağlar kontrol edilmelidir. Bilgisayardaki özel ağ ile iletişim kurması için manuel eklenen ağlar bu şekilde tespit edilebilir.

- Örnek senaryo: özel ağ tespiti yapıldı, üzerindeki tüm akıllı sözleşmelerin tespit edilmesi gerekiyor. Tespit edilen ağ Ganache üzerinden görüntülenmek üzere yeni proje oluşturulabilir. Akıllı sözleşmeyi oluşturan işlemler tespit edildiğinde o işlemin "input data" alanında bulunan bayt kodu alınarak kaynak koda dönüştürme araçlarıyla kaynak koda dönüştürülebilir.

- Remix IDE üzerinde görünen akıllı sözleşmeler incelenir.

- Bir dizinde bulunan tüm akıllı sözleşme adreslerini tespit edebilmek iki adımdan oluşur. İlk adımda dosya adında sözleşme adresi arayan komuttur. Bu komut Power Shell üzerinde çalıştırıldığında dosya adında 40 karakterli onaltılık karakter bulunan tüm dosyalar gelmektedir.

```
Get-ChildItem –Path D:\ -Recurse –File | Where-Object {$_.Name –match '[a-fA-F0-9]{40}'}
```

Şekil 6.3. Dosya adında akıllı sözleşme adresi arama

- İkinci adımda dosya içindeki karakterlerde adres olması muhtemel karakterler aranır.

```
Get-ChildItem *.txt -Recurse -File | Where-Object { (Get-Content $_.FullName) -match &quot;[a-fA-F0-9]{40}&quot; -or (Get-Content $_.FullName) -match &quot;^0x[a-fA-F0-9]{40}&quot; }
```

Şekil 6.4. Dosya içeriğinde akıllı sözleşme adresi arama

İnceleme ve denetimler sırasında akıllı sözleşmeler incelenirken dikkat edilmesi gereken alt başlıklara bu bölümden sonra yer verilecektir. Açık ağları da kapsayarak akıllı sözleşmelerin ağdan kaldırılması, güncellenmesi ve akıllı sözleşme değişken değerlerinin ağdan elde edilmesi ile solidity diline özgü fonksiyonların bilinmesi incelemeler sırasında önemli olacaktır.

6.2. Akıllı Sözleşmelerin Kendini İmha Etmesi Durumu

Akıllı sözleşmelerde bir hata olması durumunda geri döndürülemez maliyetlerde zarara sebep olmasının üzerine ağda bulunan bir akıllı sözleşmedeki bakiyeyi sahibinin hesabına

aktaracak ve sözleşmenin ağdan silinmesini sağlayacak self-destruct fonksiyonu sisteme alınmıştır [142]. Ropsten test ağının açık olduğu ve desteklendiği dönemde aşağıda adresleri bulunan sözleşmeler aktarılmıştır. Mevcut durumda erişimi olmasa da Ethereum ağında ve diğer test ağlarında yapılan benzer işlemlerle aynı sonuçlar takip edilebilecektir. Test ağına aktarılan bu sözleşmelerden ilkinde kaynak kod ağa aktarıldıktan sonra self-destruct fonksiyonu çağrılmıştır, ikincisinde ise ağa kodu aktarılmadan self-destruct fonksiyonu çağrılmıştır.

Adresi: 0x19719dc88aA8b947d44d24eC8fAAE9b08dE1BD88 Adı:HelloScSD2, SelfDestructed - Verified
Adresi: 0xe04C90C1154d36FbE6224eE46eD917015f8AB13f Adı:HelloSCNotVerified, SelfDestructed - Not Verified

Ardından Python web3 api ile bayt kodları çekilmeye çalışıldığında bayt kodlarının boş geldiği görülmektedir. Bir akıllı sözleşmeye ait bayt kodu getiren kod bloğu Şekil 6.5'te yer almaktadır.

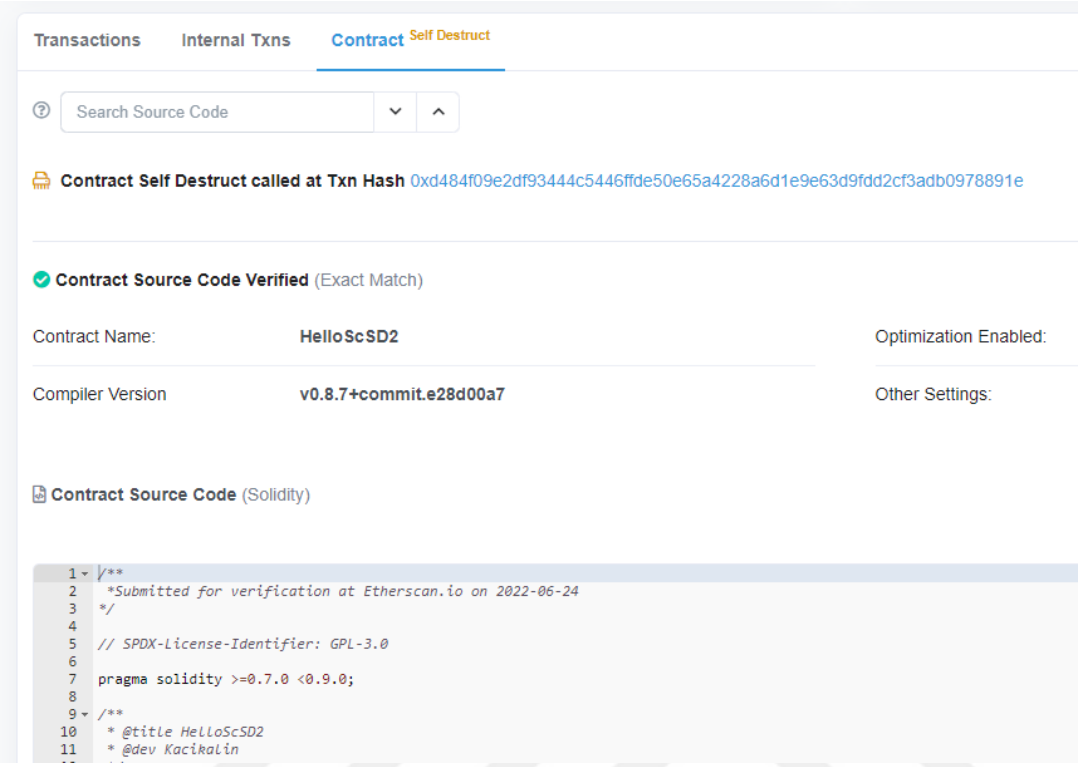
```
web3=Web3(Web3.HTTPProvider(url))
print("Ropstene baglandi mi:",web3.isConnected())
adddress='0x19719dc88aA8b947d44d24eC8fAAE9b08dE1BD88'
...
print("GetCode:")
byteCode=web3.eth.get_code(web3.toChecksumAddress(adddress))
print(byteCode)
print(byteCode.hex())
```

```
RopstenSD (34) × RopstenSD (35) × RopstenS
Ropstene baglandi mi: True
GetCode:
b''
0x
```

Şekil 6.5. Akıllı sözleşme ağından adres ile bayt kod elde etme

Bayt kodu gelmeyen bu akıllı sözleşme adresi ile etherscan adresi üzerinde arandığında geliştiricisi olarak ağa kaynak kodu aktardığımız için kod incelemesi hala yapılabilir haldedir. Şekil 6.6.'da görüldüğü gibi Contract sekmesinde Self Destruct yapıldığına dair

uyarı bulunduğu halde kod incelemesi yapılabilecektir.



Şekil 6.6. Self-destruct sözleşmenin ağda görüntülenmesi

Ancak kaynak kod daha önce aktarılmadıysa ve sözleşmede bulunan self-destruct fonksiyonu çağrıldıysa bayt kodu ya da abi kodu web3 api ile çekilememektedir, bu nedenle akıllı sözleşmenin koduna ait herhangi bir tespit yapılamayacaktır. Sözleşme için henüz self-destruct fonksiyonu çağrılmadıysa ve sözleşme kodu ağda bulunmuyorsa sözleşmenin bayt koduna hem siteden (etherscan.io) hem de web3 api aracılığı ile erişilebilmektedir.

6.3. Akıllı Sözleşme Değişkenlerinin Değerinin Elde Edilmesi

Bir akıllı sözleşmede tutulan değişkenlerin içeriğine ihtiyaç duyulması halinde, Python gibi bir dil yardımıyla ve web3 kütüphanesinin sağladığı getStorageAt fonksiyonu kullanılarak erişilebilecektir. Bu durumda Şekil 6.8.'deki gibi bir program yazılarak değerlere erişilebilecektir. Kullanılan fonksiyona gönderilen 0 parametresi, değişkenin sırasını ifade etmektedir. Bu kısım daha önce geliştirildiği için Ropsten üzerinden gösterilmiştir ancak diğer test ağları ya da Ethereum ağı için de mantık aynıdır. Python web3 bağlantısı için gerekli Infura [143] gibi Ethereum ile etkileşime giren ve web api hizmeti sunacak araca

ihtiyaç vardır. Programda değişkenlerine erişilen sözleşmenin adresi görüldüğü gibi "0xf1f5896ace3a78c347eb7eab503450bc93bd0c3b" şeklindedir. Bu adres Ropsten ağında arandığında değişkenlere atanan değerler görülebilmektedir. Değişken değerini alan getStorageAt fonksiyonunda ikinci değişken olan 0, ilk sıradaki değişkenin görüntüleneceğini ifade etmektedir. Sözleşmenin kodunda storeduint1 değişkeninin, Şekil 6.7.'de çekilen değer ile aynı şekilde 15 (Şekil 6.8.) olduğu görülmektedir.

Şekil 6.7. Bir ağdan akıllı sözleşmedeki değişken değerinin çekilmesi

```
5  pragma solidity ^0.4.0;
6
7  contract testStorage {
8
9      uint storeduint1 = 15;
10     uint constant constuint = 16;
11     uint128 investmentsLimit = 17055;
12     uint32 investmentsDeadlineTimeStamp = uint32(now);
13
14     bytes16 string1 = 'test1';
15     bytes32 string2 = 'test1236';
16     string string3 = 'lets string something';
17
18     mapping (address => uint) uints1;
19     mapping (address => DeviceData) structs1;
20
```

Şekil 6.8. Akıllı sözleşmede tutulan ilk değişkenin değeri

6.4. Akıllı Sözleşmelerin Değiştirilmesi

Bu bölümde bir ağda inceleme yapmadan önce göz önünde bulundurulması gereken ihtimallerden biri olan akıllı sözleşmelerin güncellenmesiyle ilgili bilgiler bulunmaktadır. Ethereum ilk tasarlandığında akıllı sözleşme yazıldıktan sonra üzerinde herhangi bir

değişikliğe izin vermezken, bu kısıt saldırılar sonrası ciddi maddi kayba neden olduğundan sözleşmenin iş mantığında değişikliğe (business logic) izin veren yükseltme/iyileştirme (upgrading) özelliği getirilmiştir. Sözleşmenin durumu (state) korunur ancak kullanıcılar sözleşmeyle etkileşime geçtiğinde aşağıdaki beş farklı yöntemle iş mantığında değişiklik yapılabilir [144].

- Sözleşmenin yeni bir adrese taşınması ve bakiyesinin aktarılması yöntemi kullanılabilir. Görece basit ve güvenli bir yöntemdir ancak manuel olarak taşımak zaman alabilir ve yüksek gas maliyetine neden olabilir.

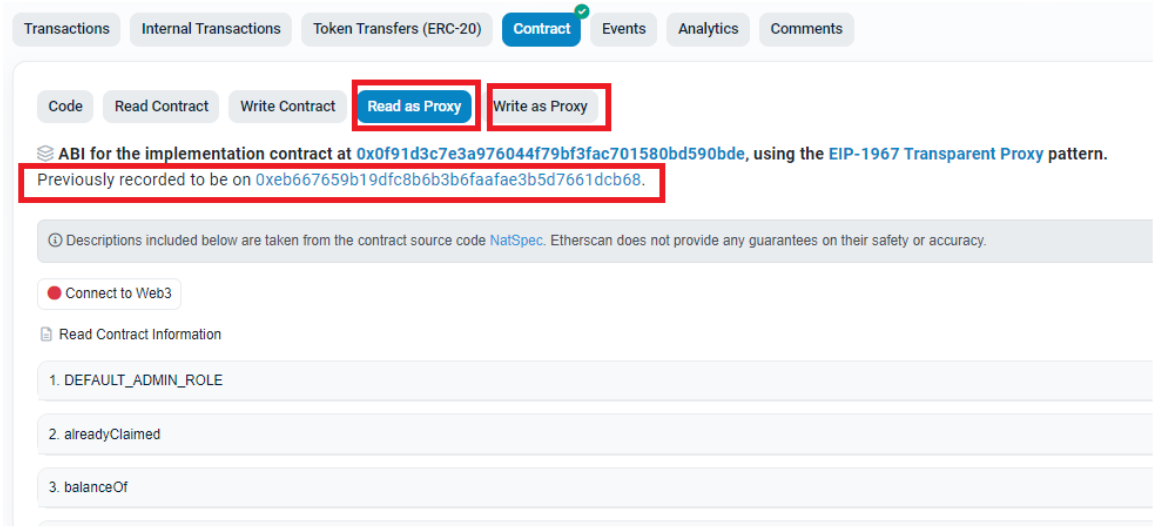
- Veri ayırma yönteminde, iş mantığı ile verilerin depolanması farklı sözleşmelere ayrılır. İş mantığının yürütüldüğü sözleşmede kullanıcılarla etkileşimler devam ederken kullanıcı bakiyeleri ve adresleri gibi durum verileri ayrı bir sözleşmede tutulur. İş mantığının yürütüldüğü sözleşme verilerin tutulduğu sözleşmenin sahibidir ve verileri almak ve düzenlemek için o sözleşme ile etkileşime girer. Verileri depolayan sözleşme varsayılan olarak değiştirilemezdir, ancak iş mantığı sözleşmesi değiştirilebilir bu şekilde durumlar ve bakiyeler korunurken Evm'de çalışan kod değiştirilebilir.

- Vekil modeli (Proxy Pattern) ile yine iş mantığı ve verilerin ayrı tutulması yöntemi izlenir, farklı olarak vekil (Proxy) sözleşme verileri tutan sözleşmedir (storage contract). Bu modelde, kullanıcılar verileri depolayan ancak iş mantığını barındırmayan vekil sözleşme ile etkileşime girer. Bu çağrı iş mantığını yürüten sözleşmeye iletilir ve dönüşü de aynı şekilde kullanıcıya geri gönderilir. Bu modeli kullanmak için delegatecall fonksiyonunun iyi anlaşılması gerekir, sonraki bölümde fonksiyon çağrıları ile bilgi bulunmaktadır. Temel olarak delegatecall, bir sözleşmenin başka bir sözleşmeyi çağırmasına işlem veren bir işlem kodudur. Kod yürütme çağıran sözleşme bağlamında gerçekleşir. Vekil modelinde delegatecall kullanılarak, vekil sözleşmenin kendi deposunu okuyup yazması ve iş mantığı sözleşmesinde tutulan mantık katmanını sanki dahili bir işlevi çağırıyormuş gibi yürütmesi mümkün olur. Vekil sözleşme yeni bir iş mantığı sözleşmesine yönlendirildiğinde, kullanıcılar vekil sözleşmenin fonksiyonlarını çağırdığında çalışacak kod değişmiş olur. Böylece kullanıcıdan yeni bir sözleşme ile etkileşimde bulunmalarını istenmeden sözleşmeyi dolaylı olarak değiştirilip iyileştirilebilir.

- Strateji modeli (Strategy Pattern) ile diğer sözleşmelerden fonksiyonları çağıran yeni bir akıllı sözleşme oluşturulur. Vekil modeli ile benzer görünür ancak strateji modeli, kullanıcıların etkileşimde bulunduğu ana sözleşme iş mantığı katmanını içerir. Strateji modeli ile temel alt yapıyı etkilemeden sınırlı da olsa değişiklik imkanı doğar. Küçük iyileştirme ve yükseltmeler için kullanılabilir ancak güvenlik zafiyeti durumunda bu yöntem yeterli olmaz.

- Elmas modeli (Diamond Pattern), vekil modelinin gelişmişi gibi düşünülebilir. Elmas vekil sözleşmeleri, delegate fonksiyonları birden fazla iş mantığı sözleşmelerinden çağırabilir bu yönüyle vekil modelden farklıdır. Elmas modeldeki iş mantık sözleşmelerine yönler/taraflar (facets) denir. Bu modelin çalışması için bu yönler ile hangi adreslerin eşleştiğinin belirtilmesi gerekir. Vekil sözleşme modelinde sözleşmedeki küçük değişiklikler için bile yeni iş mantığı sözleşmesinin oluşturulması gerekiyorken, elmas modelinde buna gerek yoktur. Elmas modelde kodun tamamı yerine yalnızca belirli yerlerin iyileştirilmesi/yükseltilmesi de mümkündür.

Bu bölümde akıllı sözleşmelerin dolaylı olarak nasıl değiştirilebileceğine değinilmiştir, yapılacak incelemelerde Ethereum üzerinde akıllı sözleşmeler ile değerlendirilirken Ethernscan üzerinde aşağıdaki işlemlerin yapılabildiğine dikkat edilmelidir. Bu işlemlerin yapılabilir olması akıllı sözleşmenin güncellendiğini göstermektedir. Şekil 6.9’da işaretlenen kısımda görüldüğü gibi daha önce kaydedilen adreste de bulunmaktadır o nedenle incelemenin o adreste de yapılması gerekir.



Şekil 6.9 Proxy model örnekli akıllı sözleşme

6.5. Solidity Diline Özgü Fonksiyonlar ve Değişken Tipleri

Akıllı sözleşmelerde kaynak kod incelemesi yapılırken Solidity diline terimlere, fonksiyon ve tetikleme (event) tanımlarının nasıl yapıldığının bilinmesi önemli olacağından bu bölümde yer verilecektir [145].

Solidity ile yazılmış bir akıllı sözleşme "pragma solidity versiyonNumarası" ile başlamaktadır. Versiyon numarasının tespiti o kod ile yapılabileceklerin değerlendirilmesi aşamasında önemli olabilecektir.

```
// SPDX-License-Identifier: GPL-3.0    //lisanslama için kullanılan ifade

pragma solidity >=0.7.0 <0.9.0;  //Solidity versiyon aralığını belirtir


contract sozlesmeOrnegi {   //Akıllı sözleşmenin adı contract ifadesi ile belirtilir

    uint256 deger;    //Tutulan değerin tipi ve adı


    function store(uint256 yeniDeger) public {   //Fonksiyon tanımı, parametresi ve erişim seviyesi
        deger = yeniDeger;
    }


    function retrieve() public view returns (uint256){  //Fonksiyon tanımı, dönüş tipi, view değerin değiştirilmediğini gösterir
        return deger;
    }
}
```

Şekil 6.10. Basit bir sözleşme kod örneği

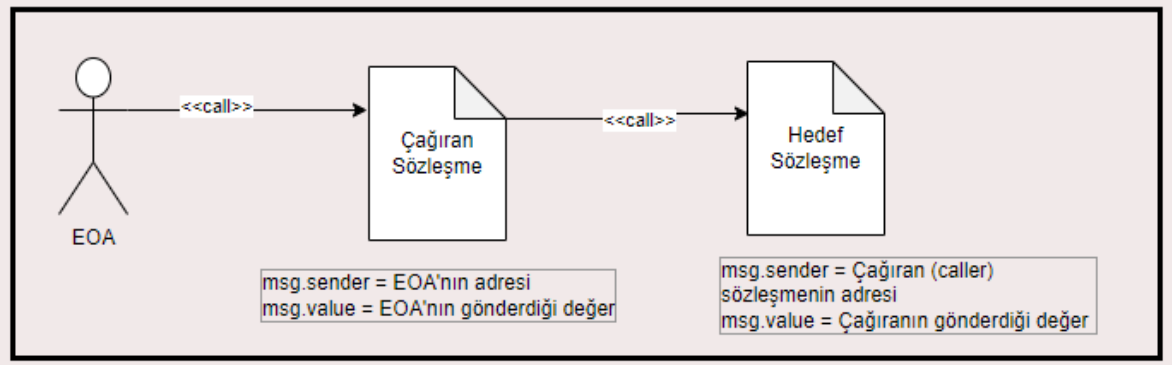
Şekil 6.10'daki kod örneğinde de görüldüğü üzere, "contract" sözcüğü ile sözleşme oluşturulmaktadır. Ardından, tutulan değerlerin tipleri ve varsa içeriklerinin atandığı kısım gelmektedir. Solidty'de üç tip değişken vardır bunlar; durum (state) değişkenler, lokal

değişkenler, global değişkenlerdir. Durum (state) değişkenleri, kalıcı olarak sözleşmede tutulan değişkenlerdir, Şekil 6.10'da bulunan "deger" adlı değişken durum değişkenine örnektir. Lokal değişkenler, sözleşmedeki fonksiyonların içinde tanımlanan o kapsamda kullanılan değişkenlerdir. Global değişkenler ise bu dile özel, ağdaki bloklar ve transaction ile ilgili bilgi alınan değişkenlerdir. Global değişkenlerin listesinden en çok kullanılanlara aşağıda yer verilmiştir.

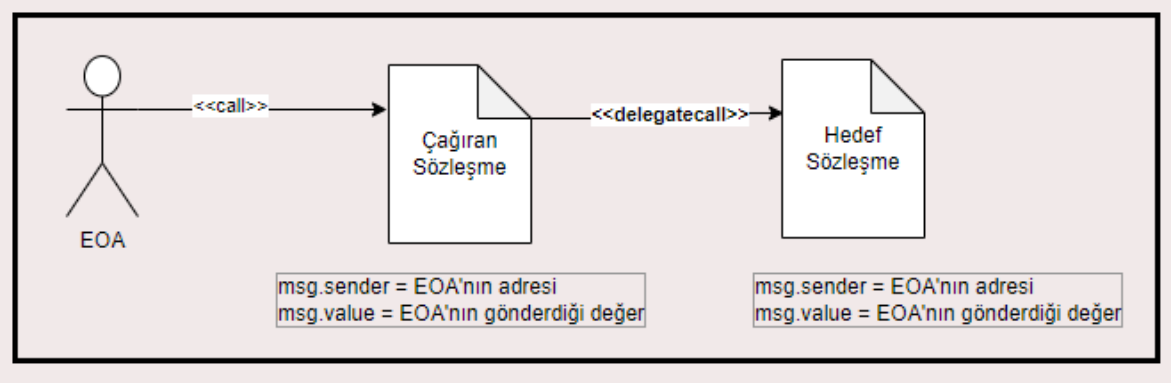
Blockhash (uintblokSayısı): referans olarak verilen sayıdaki bloğun özetini, block.difficulty (uint): referans olarak verilen sayıdaki bloğun zorluk derecesini, block.gaslimit (uint): geçerli bloğun gasLimit değerini, block.number (uint): geçerli bloğun sayısını, block.timestamp (uint): bloğun zaman damgasını Unix zamanı cinsinde, gasLeft(): uint256 tipinde kalan gas miktarını, msg.Sender: geçerli çağrıda mesajı gönderen kişinin adresini, msg.Value: geçerli çağrıda mesajla gönderilen değeri wei cinsinden döndürür [146].

Solidity dilindeki değişken tipleri; boolean (true/false tutar), integer (tam sayı), virgülden sonra kaç karakter olacağı belirli olan sabit noktalı sayılar, adres, bayt dizileri ve enum olarak sıralanmaktadır.

Adres tipinin üyeleri; balance, send, transfer, call, delegatecall ve staticcall'dur. Bir adresin bakiyesini balance göstermektedir, send ve transfer ile o adresten para göndermeyi sağlamaktadır. Send ve transfer fonksiyonlarının farkı, dönüş değerlerinde ortaya çıkmaktadır. Transfer ile gönderim sırasında hata olduğunda, sözleşme istisna (exception) ile durmaktadır. Send ile gönderim sırasında hata olduğunda ise sözleşme devam etmekte, ancak hata durumunda send fonksiyonunun dönüş değeri false olmaktadır. Call ile diğer bir sözleşme, adresi aracılığıyla çağrılmakta; başarılı olup olmadığına göre true ya da false değeri dönmektedir. Delegatecall da Call gibi, diğer bir sözleşmeyi adresi aracılığıyla çağırmaya yardımcı olmaktadır; farkları ise ilk çağıran harici hesabın bilgilerinin en son çağrılan hedef sözleşmede görünmesidir. İki çağrı arasındaki fark Şekil 6.11 ve Şekil 6.12'de gösterilmiştir [147]. Burada EOA (externally owned account) sözleşme adresi haricinde sahip olunan bir cüzdan adresine sahip kullanıcının adresidir. Staticcall ise call ile benzer işlev görmektedir, ancak çağrılan fonksiyon herhangi bir durum değişkeninde güncelleme olması durumunda, geri döndürülür bu yönüyle farklıdır. Bu üç çağrı türü de, güvenlik açıklarına neden olabileceğinden dikkatli kullanılması önerilmektedir.



Şekil 6.11. Call ile sözleşme çağrımı [147]



Şekil 6.12. Delegatecall ile sözleşme çağrımı [147]

Değişken tiplerinden bayt dizileri ve numaralandırılmış tipler (enum), diğer dillerle benzer şekilde tasarlanmıştır. Fonksiyon tipleri ise sözleşmedeki fonksiyonların türlerini ifade etmektedir. Temelde dâhili (internal) ve harici (external) olarak ayrılabilir; dâhili fonksiyonlar, sadece mevcut sözleşmenin içinden erişilebilen fonksiyonlardır. Varsayılan olarak dâhili fonksiyon belirlenmiştir. Harici fonksiyonlar ise bir transaction ile herhangi bir hesaptan veya sözleşmeden çağrılabilmektedir. Bu fonksiyonlar bir adres ve bir fonksiyon imzasından oluşmaktadırlar. Fonksiyon tanımının notasyonu aşağıda gösterilmiştir.

function (<parameter types>) {internal|external} [pure|view|payable] [returns (<return types>)]

Burada pure ifadesi olması fonksiyonda, durum değişkenlerini (state) okuma ya da yazma yapılmadığını, yalnızca hesaplamalar vb. işlemlerin yapılacağını göstermektedir. Fonksiyon tanımında view ifadesinin olması, durum değişkenlerinin (state) üzerinde değişiklik yapılmadığını ifade etmektedir. Payable ifadesi ise, msg.value değişkeni ile Ether işlemleri

yapılabileceğini göstermektedir. Bir fonksiyonda payable ifadesi olmadan Ether gönderme işlemi yapılmaya çalışılırsa hata alınacaktır.

Son olarak, Solidity'de olaylar olarak tanımlanabilen Event ile log kaydı yönetimi yapılmaktadır. Event öncelikle tanımlanır, daha sonra gerekli yerlerde tetiklenir (emit). Bu log kayıtları, blok zincirinde tutulur. Akıllı sözleşme ağda durduğu sürece, sözleşme adresi kullanılarak log kayıtlarına erişilebilir. Remix IDE'de [87] contracts dizininde bulunan Owner.sol örneğinde de görülebileceği gibi; event OwnerSet(address oldOwner, address newOwner) ile Event tanımlaması yapılabilmektedir. Daha sonra tanımlanan Event'ın kullanılması emit ifadesi ile olmaktadır. Örnekteki bu sözleşmenin yapıcı sınıfında (constructor) oluşturulan Event Şekil 6.13'te ifade edildiği gibi tetiklenebilmektedir.

```
constructor() {
    owner = msg.sender; //  mevcut çağrıyu gönderen: 'msg.sender'
    emit OwnerSet(address(0), owner);
}
```

Şekil 6.13. Event tetikleme örneği

Özel bir blok zincirde bütüncül olarak inceleme yapmak için bilinmesi gerekenler ile inceleme sırasında uyulması önerilen prosedür listesinin verildiği bu bölümün ardından sonraki tartışma bölümünde bu tez çalışmasının literatürdeki diğer çalışmalarla farkı ve katkısı bulunacaktır. Prosedür listesine yardımcı uygulama ile kaynak kodu bilinmeyen akıllı sözleşmelerde bayt koddan kaynak koda dönüştürme ihtiyacı olması durumunda değerlendirilen araçların hangisinin kullanımının daha uygun olduğuna karar vermeye yardımcı uzman sistem tasarımı uygulama bölümünde bulunacaktır.

7. UYGULAMA

Bu bölümde tezin ana amaçlarını yerine getiren iki çıktı bulunmaktadır. İlki blok zincirler üzerinde yapılacak incelemelerde yardımcı olacak python uygulaması ikincisi ise bayt koddan kaynak koda dönüştüren araçların seçiminde yardımcı olacak uzman sistem tasarımıdır.

7.1. İncelemelerde Yardımcı Olacak Uygulama

Geliştirilen Python uygulaması ile elde edilen bilgiler şu şekildedir;

- İncelenen bilgisayarda mevcut özel ağların tespitine yönelik arama sonuçları listelenir. Anahtar kelime listesi: geth, truffle-config.js, truffle, contracts, clef, hardhat, genesis.json, node-modules olarak oluşturulmuştur. Bu anahtar listesinin güncel kalması için yeni teknolojiler takip edilerek kurulumlarında oluşan dizinler ve dosyalar incelenmelidir. Konfigürasyon dosyalarının uzantıyla verilmesi ile aramalar daha etkin yapılabilecektir.

- Akıllı sözleşme veya cüzdan adresi olabilecek özelliklerde (40 karakter uzunlukta onaltılık gösterimde sayı) karakter dizileri getirilir.

- Remix IDE üzerinde bir akıllı sözleşme geliştirilmişse onun tespitin yapılması.

- Lokal bilgisayarda bulunan özel ağlarda geliştirilen akıllı sözleşmelerin bayt kodunun tespiti ile işlemlerin takibinin yapılması.

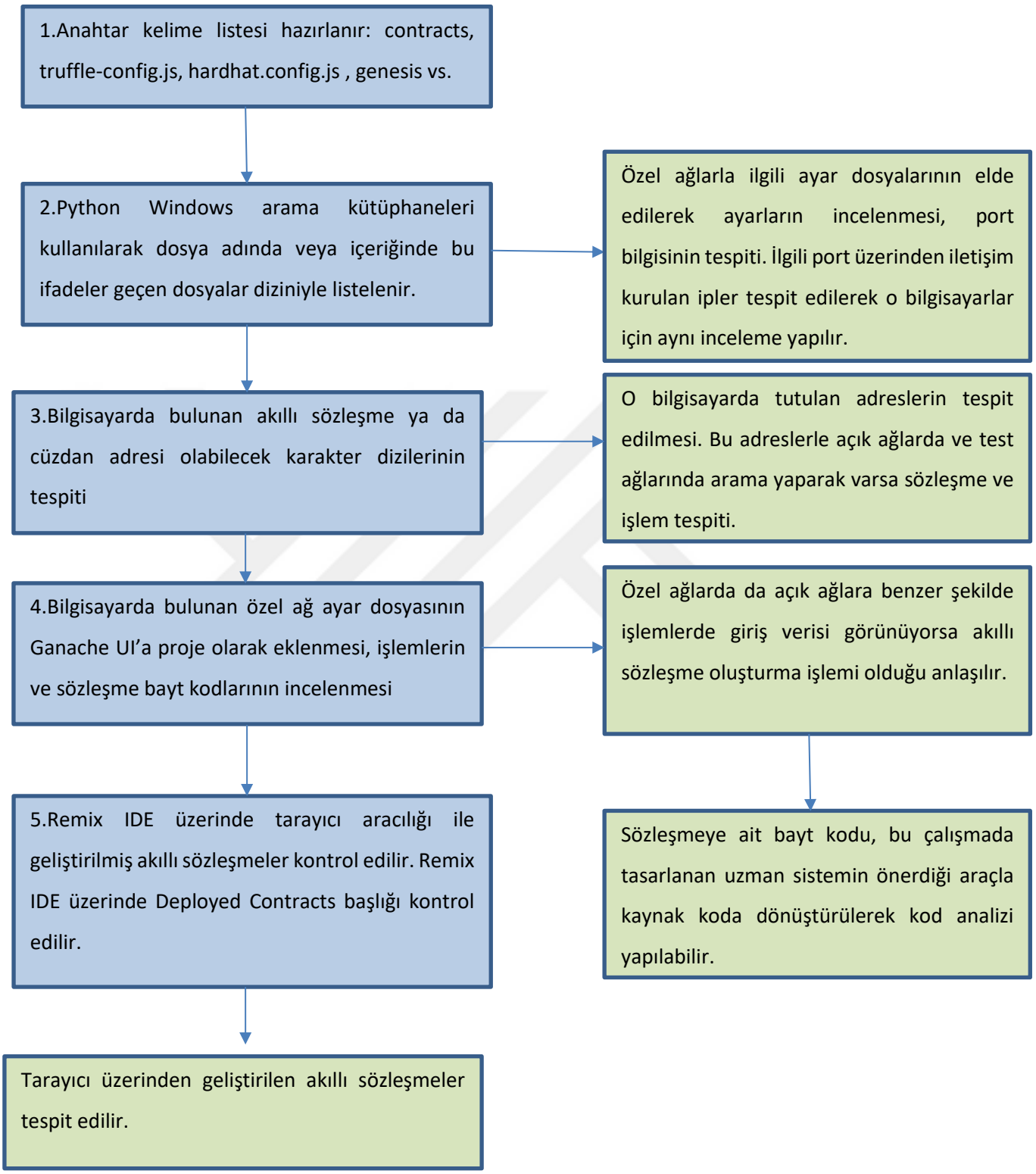
- O bilgisayarla iletişimdeki diğer bilgisayarlar listelenir.

- Uygulamaya belirtilen adres[3] üzerinden erişilebilir.

Geliştirilen uygulama ile incelenen özel ağlarda yapılacak incelemelerde elde edilen delillerin daha hızlı ve etkin şekilde tespiti sağlanacaktır. Özel ağlarda inceleme için uygulamanın içinde bulunan adımlar (mavi renkle) ve çıktıları (yeşil renkle) Şekil 7.1.'de

[3] Uygulamanın kaynak koduna erişim için bkz. https://github.com/kacikalin/privatenetworkaudit

bulunmaktadır.



Şekil 7.1. İncelemelerde yardımcı olacak uygulamanın adımları ve çıktıları.

İncelemelerde yardımcı olacak uygulamanın adımları ve çıktılarının açıklamaları aşağıda verilmiştir;

Adım 1. Anahtar kelime listesinin çıkarılması: Öncelikle inceleme yapılacak bilgisayarlarda veya mobil cihazlarda aranacak anahtar kelime listesi çıkarılır. Anahtar kelime listesi, özel blok zincir ağlarının incelendiği bölümde kurulumlarda herbir araç için hangi klasör ve dosyaların oluştuğu gözetilerek çıkarılmıştır. Özel blok zincirler için yeni alt yapılar oluştukça anahtar kelime listesi güncellenmesi gerekir.

Adım 2. Adım 1'de oluşturulan anahtar kelimeleri incelenen cihazda aranır. Bulunan konfigürasyon belgelerinden ilgili portlar, izin verilen Ip adresleri vs tespit edilir. İlgili port ile iletişime giren ip adresleri network üzerinde yapılan inceleme ile tespit edilir. Ip adresi tespit edildikten sonra o bilgisayarda bu bölümde önerilen adımlar tekrarlanır.

Adım 3. İncelenen cihazlarda daha önce etkileşime geçilmiş cüzdan veya sözleşme adresleri dosya içinde veya adında tutulabileceğinden, verilen komutlarla arama yapılarak adresler tespit edilir. Bu adresler açık ağlarda ve test ağlarında aranarak incelenen konuyla ilgili olup olmadığı incelenir. Aynı kullanıcının başka ağda aynı

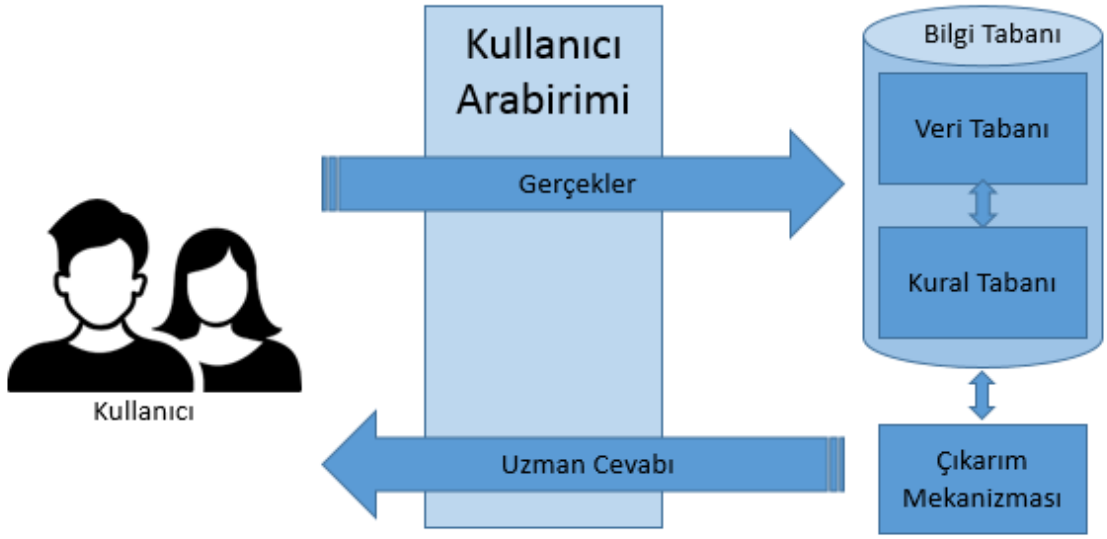
Adım 4. İncelenen makinelerde Ganache UI gibi uygulamalar varsa üzerinde özel ağlar görüntülenir, işlemlerinde Input verisi görünüyorsa akıllı sözleşme oluşturulduğu anlaşılır. Input verisi olan bayt koddan kaynak koda dönüştürme için bu çalışmada bulunan uzman sistemin önerdiği araç ile dönüştürme yapılır. Elde edilen kaynak kod incelenir.

Yapılan anahtar kelime aramasıyla özel bir ağa ait iz bulunması halinde, daha önce etkileşime geçilmiş cihazların bilgilerinin tespiti önemli olacaktır. Hali hazırda bağlıysa geth için admin.peers komutu kullanılabilir. Elde edilen konfigürasyon dosyalarında bulunan portlar üzerinden daha önce hangi cihazlarla iletişime geçildiğini görmek için ağ cihazlarında adli bilişim incelemesi yapılır.

Adım 5. Remix IDE'ye incelenen makine üzerinden erişilir, daha önce geliştirdiği akıllı sözleşmeler Deployed Contracts dizininde kaynak kodları açık şekilde bulunur. Bulunan kaynak kod incelenir.

7.2. Uzman Sistemler

Uzman sistemler, uzmanın sahip olduğu bilgi ve bilgilerle elde ettiği çıkarımları bu alanda uzman olmayan kişilerce kullanılmasına imkan tanıyan bilgisayar programlarıdır [148]. Genellikle kullanıcı arabirimi, bilgi tabanı ve çıkarım mekanizmasına sahip olan bu programlar bir uzmanın o problem karşısındaki davranışlarını taklit eder [149]. Bir uzman sistemin genel yapısı aşağıda bulunmaktadır [150].



Şekil 7.2. Uzman sistem genel yapısı

Şekil 7.2.'de genel yapısı bulunan uzman sistemde görüldüğü gibi temel olarak kullanıcı arabirimi, bilgi tabanı ve çıkarım mekanizması bulunmaktadır. Kullanıcı arabirimi uzman sistem ile sistemi kullanan kullanıcılar arasında etkileşimi sağlayan birimdir. Bilgi tabanında iki kısım bulunmaktadır; veri tabanı bilgilerin saklanmasını sağlarken kural tabanında uzmanın belirleyeceği kurallar tutulmaktadır. Son olarak çıkarım mekanizması kullanıcıdan alınan verileri kural tabanında bulunan şartlar ile karşılaştırır ve bir sonuca varır. Bu şartlar kontrol edilirken kullanılan yöntem geriye zincirleme ve ileri zincirleme olarak iki farklı şekilde yapılabilir. İleri zincirlemede, uzman sistem elindeki bilgi ile kontrole başlar, kendisine uyan kural bulursa bu kuralın mevcut koşullarını sağlıyor mu diye kontrol eder. Koşullar sağlanıyorsa o koşullar için uzman sistemin sunduğu sonuç elde edilmiş olur. Geriye zincirlemede ise tümdengelim ilkesine göre, sonuçta bulunan hipotezin doğruluğunu sağlayacak koşullar sağlanıyor mu diye kontrol edilir. Bu tez çalışmasında modellenen

uzman sistemde ileri zincirleme metodu kullanılmıştır. Sonraki bölümde uzman sistem bileşenleri ve önerilen model bulunmaktadır.

7.2.1. Uzman Sistem Bileşenleri

Uzman sistemler bu bölümün girişinde özetlendiği gibi kullanıcı arabirimi, bilgi tabanı ve çıkarım mekanizması bileşenlerinden oluşur. Kullanıcı arabirimi, bu sistemin kullanıcı olan, uzman olmayan kullanıcılarla sistem arasında iletişim sağlar. Bu arabirimde kullanıcılar arayüzde bulunan soruları yanıtlar veya alanları doldurur [151].

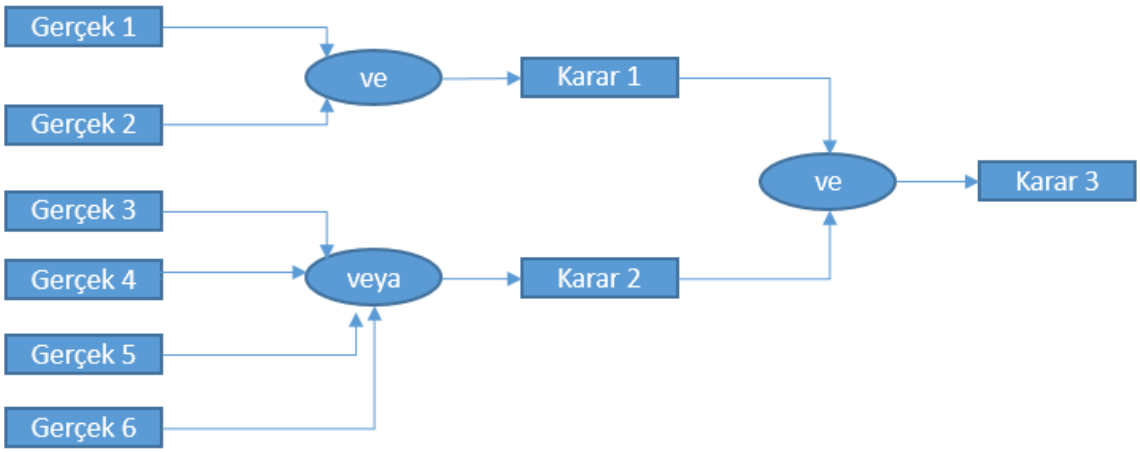
Bilgi Tabanı

Kural tabanı ve veri tabanı alt bileşenlerinden oluşan bilgi tabanı, uzmanlar tarafından sağlanan verileri tutar. Sistemin amacına yönelik durumlar ve olaylar hakkında bilgileri ve aralarındaki ilişkileri kapsar [152]. Bilgi tabanında bulunan kural tabanı, olayların sahip olduğu özellikler ile elde edilen sonuçları sebep-sonuç ilişkisi içinde gösteren kuralları saklar. Bilgi tabanının bir diğer alt bileşeni olan very tabanında ise zamanla değişebilen dinamik kısım tutulur. Bu kısımda mevcut gerçeklere kurallar uygulanarak yeni gerçekler elde edilir. Uzman sistemin başarını doğrudan etkileyen kısım bilgi tabanıdır, bilgiler yeterli kapsamda ve doğrulukta olmalıdır.

Çıkarım Mekanizması

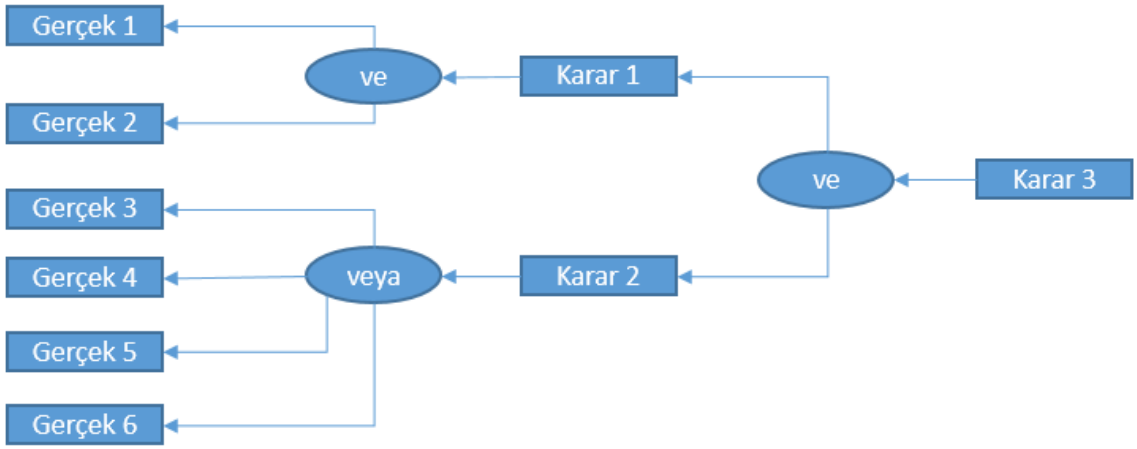
Bilgi tabanının değerlendirilip yorumlandığı bu kısımda kurallar uygulanarak kullanıcının girdiği bilgilere göre sonuçlar elde edilir. Kullanıcının girdiği yeni bilgilere göre çözümün bulunduğu ve sunulduğu bu mekanizma iki farklı çıkarım stratejisi kullanabilir. Bu stratejiler ileri zincirleme ve geri zincirlemedir.

İleri zincirleme, veri odaklı çıkarım olarak da isimlendirilebilir. Bilinen gerçeklerle başlar, kuralları kullanır ve bilgi tabanında depolaacak yeni gerçekler üretir [153]. Bu yöntemde tümevarım mantığı kullanılır. Çıkarım mekanizması artık yeni gerçekler çıkarılamayana kadar veya önceden belirlenmiş belli bir seviyeye ulaşana kadar işlemi tekrarlar. İleri zincirleme için mantıksal süreç örneği Şekil 7.3.'te bulunur.



Şekil 7.3. İleri zincirleme mantıksal süreç örneği [151]

Geri zincirleme, hedefe yönelik çıkarım olarak isimlendirilebilir. Bir hedef veya belirli bir gerçekle başlar, hangi kuralların bu amacı karşılayacağını belirlemeyi hedefler [153]. Bu yöntemde tümden gelim mantığı kullanılır. Öncelikle elde edilmek istenen karara odaklanarak o karar ile sonuçlanan gerçeklerin var olup olmadığı sorgulanır. Geri zincirleme için mantıksal süreç örneği Şekil 7.4.'te bulunmaktadır.

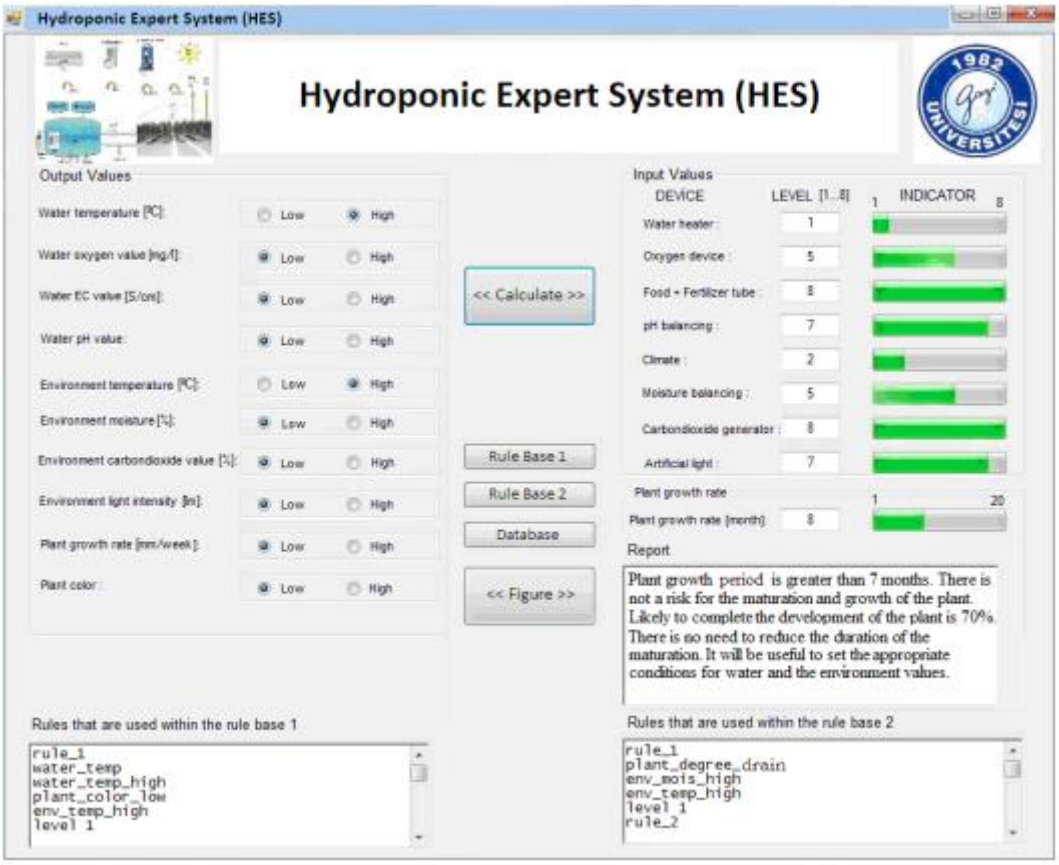


Şekil 7.4. Geri zincirleme mantıksal süreç örneği [151]

Kullanıcı Arabirimi

Sistemin kullanıları ile uzman sistem arasında iletişimi sağlayan birim kullanıcı arabirimidir. Sistem bu arabirim aracılığıyla uzman sisteme verileri alır ve çıkarım mekanizmasına aktarır. Çıkarım mekanizmasının elde ettiği sonucu kullanıcıya geri döner. Ayrıca sisteme

bilgi ve kural eklemek için de kullanılır [151]. Örnek bir arayüz Şekil 7.5.'te gösterilmiştir.

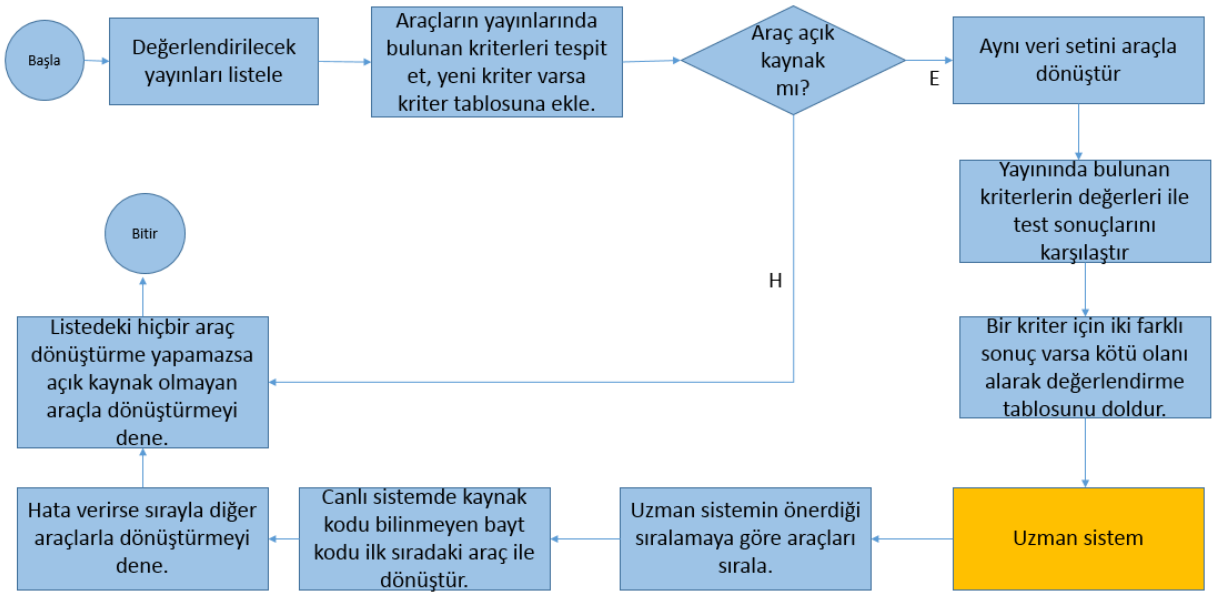


Şekil 7.5. Kullanıcı Arabirimi Örneği [154]

7.3. Araçların Seçimi İçin Uzman Sistem Tasarımı

Bu bölümde mevcut araçlar göz önünde tutularak bayt koddan kaynak koda dönüştüren araçların arasından seçim yapmak gerektiğinde yardımcı olacak uzman sistem uygulaması yer alacaktır. Gelişitirilen Uzman Sistem yardımıyla ileride yayınlanacak çalışmalarda bulunan, geliştirilecek olan araçların da dahil edilmesiyle küme büyüse bile inceleme yapacak kişilerin karar vermesi kolaylaşacaktır. Bunun için öncelikle araçların karşılaştırma kriterleri genelleştirilerek nasıl elde edileceği tespit edilmiştir. İncelenen araçlar için bu kriterler tespit edilmiştir. Ardından bu kriterlerin daha sonra değerlendirilecek araçların seçimine ne kadar etki edeceği önceliklendirerek etki katsayıları belirlenmiştir. Tasarlanan uzman sistem ile sonraki yıllarda incelenen araçların mevcut araçlar arasında tercih edilmesi önerilen öncelik sırasının ne olduğu tespit edilir.

En iyi kaynak koda dönüştürme aracını seçmek için geliştirilen model Şekil 7.6.'da bulunmaktadır.



Şekil 7.6. En iyi kaynak koda dönüştürme aracını seçmek için sunulan model

En ideal araç seçimi ve kaynak koda dönüştürme süresi için önerilen bu modelde, öncelikle değerlendirilecek yayınlar ve araçlar listelenir. Listelenen araçlar için yayınlarında veya dokümanlarında sunulan kriterler ve değerler çıkarılır. Test etmek için kaynak kodu bilinen bayt kodlar doğrulanmış akıllı sözleşmelerden elde edilir. Aynı veri seti ile listelenen araçlarla bayt koddan kaynak koda dönüştürme yapılır. Dönüştürme sonuçlarında kriterler için aldığı değerler tespit edilir. Eğer yayınlarında sunulan ile uygulamada elde edilen değerler arasında farklılık varsa daha kötü olan sonuç alınır. Örneğin ortalama dönüştürme süresi ilgili aracın yayınında a iken, yapılan uygulamada b ise ve b>a ise dönüştürme süresi olarak b alınarak her araç için kriterlerin karşılığında sahip olduğu değerler çıkarılır.

Tasarlanan uzman sistem bu değerleri alarak incelenen araçlardan hangisinin daha iyi olduğunu bir sıralama yaparak önerir. Kaynak kodu bilinmeyen bayt kodlar için, uzman sistemin önerdiği ilk sıradaki araç ile dönüştürme yapılır. Hata verip herhangi bir kaynak kod üretemeyen olmaması durumunda sırayla diğer araçlar denenir, en son tüm araçlar hata verir ve herhangi bir sonuç elde edilemezse kaynak kodu açık olmayan ve tarayıcıdan erişilebilen araçlar ile dönüştürme yapılması denenir. Modelde bulunan uzman sistem araçların kriterlerini değerlendirerek önerilen araç sıralaması oluşturmaktadır. Araçların değerlendirilmesinde kullanılan kriterler aşağıdaki tabloda yer almaktadır. Bu kriterlerden

daha önce incelenen yayınlarda bulunanlar belirtilmiştir, herhangi bir referans verilmeyen kriterler bu çalışma sırasında çıkarılmıştır.

Çizelge 7.1. Kaynak koda dönüştüren araçların değerlendirilmesinde kullanılan kriterler

Kriter Adı	Ölçme Yöntemi	Aldığı Değerlerin Yorumlanması
Hata verme oranı	N bayt kod için ilgili araçla dönüştürme yapılır x kadarından hata alınırsa x/N hata verme oranıdır.	Hata verme oranının düşük olması istenir [13], [14].
Dönüştürme Süresi	N bayt kod için ilgili araçla dönüştürme için gereken süre t ise t/N ortalama dönüştürme süresidir.	Dönüştürme süresinin düşük olması istenir [13], [14]. Ancak öncelikli bir ister olmayabilir.
Kaynak kodla eşleşen public fonksiyon oranı	Kaynak kodu bilinen N sözleşmede ortalama p1 kadar açık fonksiyon vardır, dönüştürme sonrasında elde edilen kaynak kodlarda ortalama p2 kadar açık fonksiyon vardır. Fonksiyon eşleşme oranı p2/p1 ile hesaplanır.	Fonksiyon eşleşme oranının yüksek olması istenir [13], [14].
Kaynak kodla eşleşen event oranı	Kaynak kodu bilinen N sözleşmede ortalama e1 kadar event vardır, dönüştürme sonrasında elde edilen kaynak kodlarda ortalama e2 kadar event vardır. Event eşleşme oranı e2/e1 ile hesaplanır.	Event eşleşme oranının yüksek olması istenir [14].
Özel fonksiyonları çıkartıyor mu?	Araçların tanıtım yayınlarında veya kodlarıyla ilgili verilen bilgilerde özel fonksiyonların tespit edilip edilmediği belirtilir. Yapılan dönüştürmelerde de özel fonksiyonların tespit edilip edilmediği tespit edilir.	Özel fonksiyonların da dönüştürülebiliyor olmasıyla bir sözleşme daha tam ve doğru şekilde incelenebilir [14].
Araç ya da yayınının tanıtım yılı	Github repolarına veya ilgili yayına bakılarak tespit edilir.	Daha yeni araçların daha gelişmiş olduğu varsayılır. Güncel tarih ile tanıtım yılı arasındaki fark kriter değerlendirmesinde kat sayı olarak verilir.
Aracın son güncelleme tarihi	Kaynak kodu açık olanlarda aracın kaynak kodunun en son ne zaman güncellendiği tespit edilir.	Çok yeni olması hataların henüz tespit edilememiş olduğuna işaret edebilir bununla birlikte çok eski olması güncel gelişmelerin yansıtılmadığını gösterebilir. Güncel tarih ile aracın son güncelleme tarihi arasındaki fark kriter değerlendirmesine kat sayı olarak verilir.

Çizelge 7.1. Kaynak koda dönüştüren araçların değerlendirilmesinde kullanılan kriterler (Devam)

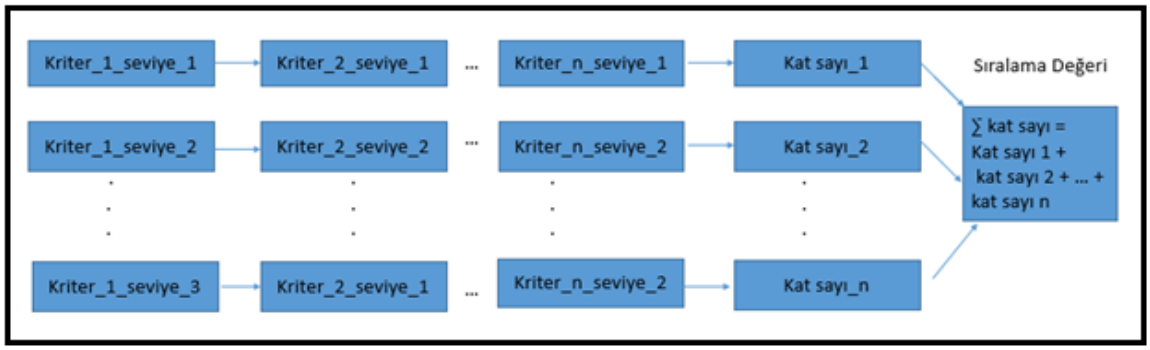
Dönüştürülen kod tekrar derlenebiliyor mu?	Dönüştürme sonrasında elde edilen kodlar incelenerek tespit edilir. Dönüştürme sonrasında ilgili dile benzer bir kaynak kod elde edilmesi uzun kodlar için incelemeyi zorlaştırabilir. Tekrar derlenebilir nitelikte bir kod çıkarılması kodların satır sayısının fazla olması gibi durumlarda avantaj sağlayabilir.	Aracın tanıtım yayınında veya kod açıklamasında çıktı olarak elde edilen kod Solidity'e benzer bir çıktı olarak belirtiliyorsa yeniden derlenebilir değildir olarak işaretlenir. Hesaplama içeren uzun kodlara sahip sözleşmelerin üreteceği çıktılar takip edilmesi gereken durumlarda yeniden derlenebilir çıktı üreten araçlar tercih edilebilir.
Test edilen veri setinin sözleşme başına fonksiyon sayısı	Kaynak kodu bilinen girdiler için kaynak kodun kompleksliğini belirtmek için toplam açık fonksiyon sayısı sözleşme sayısına bölünerek tespit edilir.	Bu değerin büyük olması test edilen sözleşmenin kompleks olduğunu gösterir. Aynı veri seti ile ölçme imkanı yoksa veri setlerinin niteliği için kullanılabilir bir kriterdir [13].
Dönüştürme aşamasında ayrılan bloklardan ulaşılamayan kodların oranı (unreachable code)	Açık kaynak ve dönüştürme aşamaları izlenebilir araçlar için dönüştürmenin tam ve kapsamlı olduğunu gösteren kriterlerden biridir. Bloklara bölünen kod blokları birbiriyle ilişkilendirildiğinde hiçbir şekilde gidilmeyecek kod bloklarının oranıdır.	Ulaşılamayan kod blokları ne kadar azsa o kadar iyi bir dönüştürme yapılarak bayt kodları o kadar kapsamlı şekilde dönüştürülebilmiştir diye yorumlanır [13], [14].
Tespit edilemeyen gidilecek jump adresi oranı (unknown jump target)	Açık kaynak ve dönüştürme aşamaları izlenebilir araçlar için jump ifadelerinde hedef adreslerden tespit edilemeyenlerin oranını gösterir.	Hedef jump adresinin bilinmeme oranı ne kadar azsa o kadar eksiksiz bir dönüştürme yapıldığı kabul edilir. Sözleşme başına bu oranın düşük olması dönüştürmenin daha tam olduğunu gösterir [13], [14] .

Tabloda belirtilen kriterlerin öncelik sıralamasını ve hangi durumda hangi kriterin daha öncelikli olacağını uzman sistem belirleyerek önerilen sıralama oluşturacaktır. Bir sonraki bölümde uzman sistemle ilgili temel bilgiler, çıkarım süreci ve bazı çıkarım kurallarından örneklere yer verilecektir.

7.4. Önerilen Uzman Sistem

Uzman sistem değerlendirmesinden önce karşılaştırması yapılacak araçların kaynak koda

dönüştürme işlemi sonucunda, Çizelge 7.1.'de bulunan kriterler için hangi değerlere sahip olduğu çıkarılır. Araçların sonuçları, diğer araçların sonucuna göre sınıflandırılır. Örneğin hata verme oranı en fazla olan yüksek, en düşük olan düşük diğerleri orta olarak sınıflandırılabilir veya her kritere özgü aralıklar belirlenerek her sonuç seviye_1, seviye_2 gibi etiketlenebilir. Test edilecek araç sayısının çok fazla olmayacak olması beklendiğinden standart sapmaya göre sınıflandırma teknikleri önerilmemiştir. Ancak sayının fazla olması ve sonuçlar arasında farkların çok fazla olması durumunda en düşük ve en yüksek çıkarıldıktan sonra standart sapma değerine göre yorumlanarak var olan değerler sınıflandırılabilir. İleri zincirleme metoduyla kaynak koda dönüştürme araçlarının önerilme sırasını veren sürecin kuralları ile çıkarım zincirini gösteren şekil Şekil 7.7'de bulunmaktadır.



Şekil 7.7. Araçların önerilme sırasını çıkaran kurallar ve çıkarım zinciri

Şekil 7.7.'de de görüldüğü gibi her kriter için seviye kontrolü yapıldıktan sonra uyan zincirin sonucunda elde edilen kat sayılar toplanarak bir sıralama değeri elde edilir. Bu sıralama değeri küçükten büyüğe sıralanır ve en büyük değere sahip olan en öncelikli önerilen olarak yorumlanır.

Bu kısımda yeni yayın ve araçlarla kriterlerin güncellenebiliyor oluşu önerilen sistemi dinamik bir uzman sistem haline getirmektedir. Kriterlerin kuralları ile sıralamasına etki edecek kat sayının belirlenmesi için yine uzman görüşü gerekecektir. Mevcut kriterler ile sıralama değerinin hesaplamasında kullanılan bazı örnek kurallar Şekil 7.8.'de verilmiştir.

Şekil 7.8. Sıralama değerinin hesaplamasında kullanılan örnek kurallar

Şekilde görülen çıkarımlardan koşulları sağlanaların katsayıları Şekil 7.7.'de görüldüğü gibi toplanarak her araç için bir sıralama değeri elde edilir. Bu kurallarda önerilen kat sayılar kriterlerin toplama ne kadar etki edeceğini belirler. Her aracın özelliklerine göre sahip olduğu kat sayılar toplanarak tek bir sıralama değeri çıkarılır. Aracın sahip olduğu kat sayılardan eksi değerde olanlar aracın sıralamasını geriye götürürken artı değerler sıralamasnı yükseltecektir. Bu şekilde tavsiye edilen araçların sıralaması oluşturulmuş olur. Yeni bir dava ya da incelemede kaynak koda dönüştürme aracının kullanılması gerektiğinde araçların bu sıra ile kullanılması önerilir. Önerilen sıralamada ilk sırada olan araç ile dönüştürme denenir, hata alma durumunda ikinci araca geçilerek hata alınmayıncaya kadar devam edilir. Tüm araçlarda hata alınması durumunda kaynak kodu açık olmayan araçlarla dönüştürme denenir ve kaynak koda ulaşmak hedeflenir.

8. TARTIŞMA

Blok zincirler üzerinde yapılan akıllı sözleşme incelemelerinde, akıllı sözleşmenin kaynak kodu ağa yüklenmemişse içeriği ile ilgili bilgi yalnızca ağda bulunan bayt koddan alınabilmektedir. Akıllı sözleşmenin kodunun incelenebilmesi ancak bayt koddan kaynak koda dönüştüren araçlar yardımı ile yapılabilmektedir. Literatür incelendiğinde Gigahorse aracının Porosity ve Vandal ile; Erays aracının Porosity ile; Elipmoc aracının ise Gigahorse ve Panoramix ile karşılaştırıldığı görülmüştür. Bu tez çalışmasında daha önce karşılaştırılması yapılmayan Dsol, Erays ve Elipmoc'un karşılaştırılması yapılarak bayt koddan kaynak koda dönüştüren araçlar incelenmiştir. 4. Bölümde elde edilen bulgular değerlendirildiğinde; Elipmoc'un diğer iki araca göre her ne kadar daha yavaş olsa da tespit edebildiği fonksiyon sayısı, event sayısı ve hata vermeden dönüştürme yapılan sözleşme oranı yönüyle daha tam ve doğru bir dönüştürme yaptığı görülmüştür.

Bayt koddan kaynak koda dönüştüren araçların sayısındaki artış ile seçiminin gelecekte zorlaşabileceği düşünülerek bu araçların seçiminde kullanılabilecek uzman sistem uygulama bölümünde tasarlanmıştır.

Bu tez çalışmasında ayrıca özel blok zincirlerde yapılacak adli bilişim incelemeleri için prosedür listesi çıkarılarak inceleme yaparken zaman kazandıracak program geliştirilmiştir. 6. Bölümde detayları bulunan literatür taraması ile elde edilen sonuçlardan özel blok zincirlerde yapılacak incelemelere yönelik bir çalışmanın ve önerilen prosedür listesinin olmadığı tespit edilmiştir. Literatür taraması private blockchain AND audit, private blockchain AND forensic ve private blockchain AND investigation anahtar kelimeleri ile ayrı ayrı ve kapsamlı şekilde yapılmıştır. Geliştirilen program ile incelemede yardımcı araç kullanıma sunulmuştur.

Özetle bu tez çalışmasının katkıları; özel blok zincirlerde yapılacak adli bilişim incelemeleri için prosedür listesi önermesi, bu prosedürlere yardımcı uygulamanın geliştirilmesi, incelemeler esnasında karşılaşılabilecek kaynak kodu bilinmeyen akıllı sözleşmeler için bayt koddan kaynak koda dönüştüren araçlardan daha önce karşılaştırılması yapılmayan araçların karşılaştırmasını yapması ve bu araçların seçimi için bir uzman sistem tasarlanmasıdır.

Bu çalışmanın çıktılarından olan dinamik uzman sistem Ethereum dışında diğer platform ve diller için de kullanılabileceğinden tezin güçlü yanlarından sayılabilir. Benzer şekilde önerilen prosedür listesinde sunulan anahtar kelime listesi Ethereum dışındaki ağları da kapsayacak şekilde geniş tutulmuştur. Bu çalışmada sınırlılık olarak belirtilebilecek tek kısım, özel blok zinciri ağlarda, akıllı sözleşme yüklemesini yapmayan bilgisayarda tespit edilen konfigürasyon dosyasını kullanarak bayt kod elde edilmesi aşamasında, önerilen Ganache UI'a yükleme önersinin yalnızca Ethereum özel ağları için yapılacak olmasıdır. Diğer öneriler ve uygulamalar tüm platform ve ağlar için kullanılabilir niteliktedir.

Gelecek çalışmalarda özel blok zincirlerde adli bilişim incelemelerinde kullanılacak uygulamalar geliştirilerek daha fonksiyonel hale getirilebilir. Araçların sayısı arttıkça bu çalışmada önerilen uzman sistemin önerilerini doğrulayacak uygulama geliştirilebilir. Bu alanda çalışan farklı kişilerce anlamlı veri seti mümkün olması halinde makine öğrenmesi ile hangi aracın daha iyi olduğuna karar veren uygulamalar geliştirmek mümkün olabilir.

9. SONUÇ VE ÖNERİLER

Akıllı sözleşmeler Defi ve DApps için temel öge durumundadır, kötüye kullanımının tespitinin denetçiler tarafından yapılabilmesi için Ethereum alt yapısının, ağ mantığının, işlem ve sözleşme yüklemenin tüm detaylarıyla bilinmesi gerekir. Denetim yapılacak ortamın çok çeşitli olabileceği bu çalışmayı zorlaştıran unsurlardandır. Bir özel ağ üzerinde akıllı sözleşme geliştirme ve çalıştırma Hardhat, Truffle, Geth ve Remix IDE yardımıyla yapılabileceğinden bu araçların kullanımına ve sundukları işlevlere hakim olunmalıdır.

Blok zincirler üzerinde yapılan akıllı sözleşme incelemelerinde, akıllı sözleşmenin kaynak kodu ağa yüklenmemişse içeriği ile ilgili bilgi yalnızca ağda bulunan bayt koddan alınabilmektedir. Akıllı sözleşmenin kaynak kodunun Ethereum ağına yüklenme oranının oldukça düşük olduğu bilinmektedir, bu oranın 2022 yılının ilk çeyreği için yüklenme oranının %0,36 olduğu görülmüştür. Denetleyici kurumlar veya kolluk kuvvetlerince bir akıllı sözleşmenin kodu incelenmesi gerektiğinde inceleme ancak bayt koddan kaynak koda dönüştüren araçlar yardımı ile yapılabilecektir. Literatür incelendiğinde Gigahorse aracının Porosity ve Vandal ile; Erays aracının Porosity ile; Elipmoc aracının ise Gigahorse ve Panoramix ile karşılaştırıldığı görülmüştür. Bu tez çalışmasında daha önce karşılaştırılması yapılmayan Dsol, Erays ve Elipmoc'un karşılaştırılması yapılarak bayt koddan kaynak koda dönüştüren araçlar incelenmiştir. 4. Bölümde geliştirilen uygulama ile elde edilen bulgular değerlendirildiğinde; Elipmoc diğer iki araca göre her ne kadar daha yavaş olsa da tespit edebildiği fonksiyon sayısı, event sayısı ve hata vermeden dönüştürme yapılan sözleşme oranı yönüyle daha tam ve doğru bir dönüştürme yapabilmektedir.

Bayt koddan kaynak koda dönüştüren araçların sayısındaki artış ile araç seçiminin gelecekte zorlaşabileceği düşünülerek bu araçların seçiminde kullanılabilecek dinamik bir uzman sistem tasarlanmıştır. Uzman sistem, değerlendirilen araçların her biri için bu çalışmada katkı sunularak genişletilen kriterlerin etki ettiği bir sıralama değeri oluşturur. Tüm araçlar için değerlendirme bittiğinde sıralama değeri büyükten küçüğe sıralanır ve önerilen araç kullanım sırası belirlenmiş olur. Önerilen bu sistem ile araçların sonuçlarının değerlendirilmesi için yeknesak bir sistem önerilmiştir.

Bu tez çalışmasında ayrıca özel blok zincirlerde yapılacak adli bilişim incelemeleri için prosedür listesi çıkarılarak inceleme yaparken zaman kazandıracak program geliştirilmiştir. 6. Bölümde detayları bulunan literatür taraması ile elde edilen sonuçlardan özel blok zincirlerde yapılacak incelemelerde kullanılması önerilen prosedür listesinin olmadığı tespit edilmiştir. Literatür taraması private blockchain AND audit, private blockchain AND forensic ve private blockchain AND investigation anahtar kelimeleri ile ayrı ayrı ve kapsamlı şekilde yapılmıştır. Geliştirilen program ile incelemede yardımcı araç kullanıma sunulmuştur.

Yapılacak incelemelerde hangi anahtar kelimeler ile arama yapılması gerektiği Ethereum'un dışında diğer özel ağlar da göz önünde tutularak listelenmiştir bu çalışmada. İncelenen bilgisayarlarda akıllı sözleşme veya cüzdan adresi olabilecek karakterleri içeren dosyaları listeleyen PowerShell komutu bu çalışma sırasında geliştirilmiştir.

Özel blok zincirlerde yapılacak incelemelerde açık ağlarda da inceleme yapılması ihtimaline karşı 6. Bölümde akıllı sözleşmelerin kendini imha etmesi durumu, akıllı sözleşmelerdeki değişkenlerin değerinin elde edilmesi, akıllı sözleşmelerin değiştirilmesi ve Solidity diline özgü fonksiyon ve değişken tipleri ile ilgili detaylı bilgilere yer verilerek inceleme yapacak kişilerin kullanımına sunulmuştur.

Bu çalışmanın gerek bayt koddan kaynak koda dönüştüren araçların değerlendirilmesi gerekse özel blok zincirlerde ve özellikle akıllı sözleşmelerde inceleme yapılması adımlarına yardımcı bir kılavuz niteliğinde olacağı değerlendirilmektedir.

Tartışma bölümünde de belirtildiği gibi gelecek çalışmalarda özel ağlarda incelemede yardımcı olması için geliştirilen uygulama daha fonksiyonel hale getirilebilir. Konuyla ilgili çalışan ve anlamlı çok sayıda veri üretebilecek farklı kişiler olması ile kaynak koda dönüştüren araçların seçimini makine öğrenmesi yöntemiyle yapan uygulamalar geliştirilebilir.

KAYNAKLAR

[1] P. Tasatanattakool ve C. Techapanupreeda. (2018). Blockchain: Challenges and applications. *2018 International Conference on Information Networking (ICOIN)*, Oca. 2018, ss. 473-475. doi: 10.1109/ICOIN.2018.8343163.

[2] A. S. Rajasekaran, M. Azees, ve F. Al-Turjman. (2022). A comprehensive survey on blockchain technology. *Sustainable Energy Technologies and Assessments*, c. 52, s. 102039, Ağu. 2022, doi: 10.1016/j.seta.2022.102039.

[3] T. Ahram, A. Sargolzaei, S. Sargolzaei, J. Daniels, ve B. Amaba. (2017). Blockchain technology innovations, içinde. *2017 IEEE Technology & Engineering Management Conference (TEMSCON)*, Haz. 2017, ss. 137-141. doi: 10.1109/TEMSCON.2017.7998367.

[4] K. Govindan, P. Jain, R. Kr. Singh, ve R. Mishra. (2024). Blockchain technology as a strategic weapon to bring procurement 4.0 truly alive: Literature review and future research agenda, *Transportation Research Part E: Logistics and Transportation Review*, c. 181, s. 103352, Oca. 2024, doi: 10.1016/j.tre.2023.103352.

[5] W. Viriyasitavat ve D. Hoonsopon. (2019). Blockchain characteristics and consensus in modern business processes. *Journal of Industrial Information Integration*, c. 13, ss. 32-39, Mar. 2019, doi: 10.1016/j.jii.2018.07.004.

[6] H. T. M. Gamage, H. D. Weerasinghe, ve N. G. J. Dias, "A Survey on Blockchain Technology Concepts, Applications, and Issues", *SN COMPUT. SCI.*, c. 1, sy 2, s. 114, Nis. 2020, doi: 10.1007/s42979-020-00123-0.

[7] İnternet: S. Nakamoto. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. URL: https://bitcoin.org/bitcoin.pdf. Son Erişim Tarihi: 01.06.2022

[8] İnternet: V. Buterin. (2014). Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. URL: https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf. Son Erişim Tarihi: 01.06.2022

[9] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, ve F. Wang, "Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends", *IEEE Transactions on*

Systems, Man, and Cybernetics: Systems, c. 49, sy 11, Art. sy 11, Kas. 2019, doi: 10.1109/TSMC.2019.2895123.

[10] İnternet: Solidity — Solidity 0.8.15 documentation. URL: https://docs.soliditylang.org/en/v0.8.15/index.html, Son Erişim Tarihi: 13.07. 2022.

[11] İnternet: Vyper — Vyper documentation. URL: https://vyper.readthedocs.io/en/stable, Son Erişim Tarihi: 13.07. 2022

[12] İnternet: etherscan.io, Ethereum (ETH) Blockchain Explorer, *Ethereum (ETH) Blockchain Explorer*. URL:http://etherscan.io/ Son Erişim Tarihi:28.06.2022.

[13] N. Grech, L. Brent, B. Scholz, ve Y. Smaragdakis. (2019). Gigahorse: Thorough, Declarative Decompilation of Smart Contracts, *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, May. 2019, ss. 1176-1186. doi: 10.1109/ICSE.2019.00120.

[14] N. Grech, S. Lagouvardos, I. Tsatiris, ve Y. Smaragdakis. (2022). Elipmoc: advanced decompilation of Ethereum smart contracts, *Proc. ACM Program. Lang.*, c. 6, sy OOPSLA1, s. 77:1-77:27, Nis. 2022, doi: 10.1145/3527321.

[15] Y. Zhou, D. Kumar, S. Bakshi, J. Mason, A. Miller, ve M. Bailey. (2018). Erays: Reverse Engineering Ethereum's Opaque Smart Contracts, *27th USENIX Security Symposium (USENIX Security 18)*, 2018, ss. 1371-1385. Erişim: 24 Şubat 2023. [Çevrimiçi]. Erişim adresi: https://www.usenix.org/conference/usenixsecurity18/presentation/zhou

[16] İnternet: Online Solidity Decompiler, *Online Solidity Decompiler*. URL:https://ethervm.io/decompile. Son Erişim Tarihi:24.02.2023.

[17] İnternet: JEB Decompiler by PNF Software. URL:https://www.pnfsoftware.com/ Son Erişim Tarihi:24.02.2023.

[18] İnternet: Smart Contract Audit | Trustlook. URL:https://www.trustlook.com/smart-contract-audit, Son Erişim Tarihi:07.03.2023.

[19] M. D. Sheldon. (2019). A Primer for Information Technology General Control Considerations on a Private and Permissioned Blockchain Audit, CURR. ISS. AUDIT., c. 13, sy 1, ss. A15-A29, SPR 2019, doi: 10.2308/ciia-52356.

[20] J. vom Brocke, A. Hevner, ve A. Maedche, “Introduction to Design Science Research”, 2020, ss. 1-13. doi: 10.1007/978-3-030-46781-4_1.

[21] Z. Zheng, S. Xie, H. Dai, X. Chen, ve H. Wang, “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends”, içinde *2017 IEEE International Congress on Big Data (BigData Congress)*, Haz. 2017, ss. 557-564. doi: 10.1109/BigDataCongress.2017.85.

[22] N. Szabo. (1994). Smart Contracts. URL: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html Son Erişim Tarihi: 2.3.2021.

[23] T. Bartkewitz ve R.-U. Bochum, “Building Hash Functions from Block Ciphers, Their Security and Implementation Properties”, s. 22, 2009.

[24] R. C. Merkle, “A Digital Signature Based on a Conventional Encryption Function”, içinde *Advances in Cryptology — CRYPTO ’87*, C. Pomerance, Ed., içinde Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1988, ss. 369-378.

[25] A. USTA ve S. DOĞANTEKİN, “BlockChain 101 v2”, BKM, 2017.

[26] W. Diffie ve M. Hellman, “New directions in cryptography”, *IEEE Transactions on Information Theory*, c. 22, sy 6, ss. 644-654, Kas. 1976, doi: 10.1109/TIT.1976.1055638.

[27] İnternet: S. E. Seker, Eliptik Eğri Şifrelemesi. URL: http://www.academia.edu/2428678/Eliptik_E%C4%9Fri_%C5%9Eifrelemesi Erişim Tarihi: 26.10.2022

[28] R. L. Rivest, A. Shamir, ve L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, *Commun. ACM*, c. 26, sy 1, ss. 96-99, Oca. 1983, doi: 10.1145/357980.358017.

[29] P. Baran, “On Distributed Communications Networks”, *IEEE Transactions on Communications Systems*, c. 12, sy 1, Art. sy 1, Mar. 1964, doi: 10.1109/TCOM.1964.1088883.

[30] İnternet: Transaction - Bitcoin Wiki. URL: https://en.bitcoin.it/wiki/Transaction Son Erişim Tarihi: 27.12.2018.

[31] İnternet: practical Byzantine Fault Tolerance(pBFT), *GeeksforGeeks*, 11 Ocak 2019. URL: https://www.geeksforgeeks.org/practical-byzantine-fault-tolerancepbft/ Son Erişim Tarihi: 21.06.2023

[32] İnternet: Neo Project, Neo Documentation, 2020. URL:https://docs.neo.org/docs/en-us/basic/whitepaper.html. Son Erişim Tarihi: 03.03.2021

[33] W. Zhao, S. Yang, X. Luo, ve J. Zhou. (2021). On PeerCoin Proof of Stake for Blockchain Consensus, *2021 The 3rd International Conference on Blockchain Technology*, ICBCT '21. New York, NY, USA: Association for Computing Machinery, Tem. 2021, ss. 129-134. doi: 10.1145/3460537.3460547.

[34] İnternet: Proof-of-stake (PoS) URL: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/ Son Erişim Tarihi: 21.06.2023.

[35] İnternet: NEM, General Information | NEM Documentation, 2018. URL:https://docs.nem.io/en Son Erişim Tarihi:03.03.2021

[36] F. Gündüz, S. Birogul, ve U. Kose. (2023). Proof of Optimum (PoO): Consensus Model Based on Fairness and Efficiency in Blockchain, *Applied Sciences*, c. 13, sy 18, Art. sy 18, Oca. 2023, doi: 10.3390/app131810149.

[37] D. Mourtzis, J. Angelopoulos, ve N. Panopoulos. (2023). Blockchain Integration in the Era of Industrial Metaverse, *Applied Sciences*, c. 13, sy 3, Art. sy 3, Oca. 2023, doi: 10.3390/app13031353.

[38] G. Mathy. (2023). Eliminating Environmental Costs to Proof-of-Work-Based Cryptocurrencies: A Proposal, *Eastern Econ J*, c. 49, sy 2, ss. 206-220, Nis. 2023, doi: 10.1057/s41302-023-00235-4.

[39] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, ve A. Bani-Hani. (2021). Blockchain smart contracts: Applications, challenges, and future trends, *Peer-to-Peer Netw. Appl.*, c. 14, sy 5, ss. 2901-2925, Eyl. 2021, doi: 10.1007/s12083-021-01127-0.

[40] İnternet: Algorand White Papers. URL: https://www.algorand.com/technology/white-papers Son Erişim Tarihi: 13.07.2022.

[41] İnternet: D. G. Wood, POLKADOT: VISION FOR A HETEROGENEOUS MULTI-CHAIN FRAMEWORK, URL: https://assets.polkadot.network/Polkadot-whitepaper.pdf Son Erişim Tarihi: 13.07.2022

[42] İnternet: A. Yakovenko, Solana: A new architecture for a high performance blockchain, URL: https://solana.com/solana-whitepaper.pdf. Son Erişim Tarihi: 13.07.2022

[43] İnternet: Binance Smart Chain White Paper. URL:https://github.com/bnb-chain/whitepaper Son Erişim Tarihi: 13.07.2022

[44] İnternet: Avalance Whitepapers.URL: https://www.avalabs.org/whitepapers Son Erişim Tarihi: 13.07.2022.

[45] İnternet: Home – EOSIO Blockchain Software & Services. URL: https://eos.io/ Son Erişim Tarihi:28.12.2022.

[46] L. Marchesi, M. Marchesi, G. Destefanis, G. Barabino, ve D. Tigano, "Design Patterns for Gas Optimization in Ethereum", içinde 2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), 2020, ss. 9-15. doi: 10.1109/IWBOSE50093.2020.9050163.

[47] İnternet: Ethereum Average Gas Price Chart | Etherscan URL: http://etherscan.io/chart/gasprice. Son Erişim Tarihi: 13.07.2022

[48] İnternet:Eth Gas Station. URL: https://ethgasstation.info/ Son Erişim Tarihi:13.07.2022

[49] İnternet: Introduction to smart contracts. URL: https://ethereum.org/en/developers/docs/smart-contracts/ Son Erişim Tarihi:13.08.2022

[50] Hu, Yining & Liyanage, Madhusanka & Manzoor, Ahsan & Thilakarathna, Kanchana & Jourjon, Guillaume & Seneviratne, Aruna. (2019). Blockchain-based Smart Contracts - Applications and Challenges. Erişim: 15 Haziran 2023. [Çevrimiçi]. Erişim adresi: https://www.researchgate.net/publication/328230865_Blockchain-based_Smart_Contracts_-_Applications_and_Challenges

[51] T. E. Hybel, (2018). Decompilation of Ethereum smart contracts, Master Tezi, Aarhus University, Danimarka.

[52] İnternet: Understanding the Yellow Paper's EVM Specifications, URL: https://ethereum.org/en/developers/tutorials/yellow-paper-evm/ Son Erişim Tarihi: 21.01.2023

[53] K. Christidis ve M. Devetsikiotis. (2016). Blockchains and Smart Contracts for the Internet of Things, *IEEE Access*, c. 4, ss. 2292-2303, 2016, doi: 10.1109/ACCESS.2016.2566339.

[54] İnternet: BCTR Rapor: Akıllı Sözleşme. (2021). *Blockchain Türkiye Platformu*, 28 Ağustos 2021. URL:https://bctr.org/bctr-rapor-akilli-sozlesme-23349/ Son Erişim Tarihi:07.01.2022

[55] A. Beniiche. (2021). A Study of Blockchain Oracles. Journal of Computer and Communications, Vol.9 No.3, s. 9.

[56] L. Zhang, Y. Wang, F. Li, Y. Hu, ve M. H. Au. (2019). A game-theoretic method based on Q-learning to invalidate criminal smart contracts, *Information Sciences*, c. 498, ss. 144-153, Eyl. 2019, doi: 10.1016/j.ins.2019.05.061.

[57] D. Yaga, P. Mell, N. Roby, ve K. Scarfone. (2018). Blockchain Technology Overview, National Institute of Standards and Technology, *NIST Internal or Interagency Report (NISTIR) 8202*, Eki. 2018. doi: https://doi.org/10.6028/NIST.IR.8202.

[58] İnternet: "Yul — Solidity 0.8.24 documentation", 10 Kasım 2023. URL: https://docs.soliditylang.org/en/latest/yul.html Son Erişim Tarihi: 10.11.2023

[59] İnternet: "Ethereum Whitepaper", V. Buterin, 04 Mart 2021. URL: https://ethereum.org. Son Erişim Tarihi:04.03.2021

[60] İnternet: Technical Features. URL: https://developers.eos.io/welcome/v2.0/introduction-to-eosio/technical_features Son Erişim Tarihi: 23.01.2023

[61] İnternet: Ethernaut Lvl 0 Walkthrough: ABIs, Web3, and how to abuse them, URL: https://medium.com/hackernoon/ethernaut-lvl-0-walkthrough-abis-web3-and-how-to-abuse-them-d92a8842d71b. Son Erişim Tarihi:03.02.2023

[62] İnternet: Opcodes for the EVM. URL: https://ethereum.org/tr/developers/docs/evm/opcodes/. Son Erişim Tarihi: 05.02.023.

[63] İnternet: EVM: From Solidity to byte code, memory and storage. Çevrim içi video. URL: https://www.youtube.com/watch?v=RxL_1AfV7N4. Son Erişim Tarihi: 26.01.2023

[64] İnternet: Ethereum Signature Database. URL: https://www.4byte.directory/ Son Erişim Tarihi:10.02.2023.

[65] C. Cifuentes. (1994). *Reverse compilation techniques*, Doktora tezi, Queensland University of Technology. Erişim adresi: https://eprints.qut.edu.au/36820/.

[66] İnternet: Smart Contracts Created (granular)", URL: https://dune.com/queries/688911/1278066. Son Erişim Tarihi: 20.02.2023.

[67] İnternet: Ethereum Daily Verified Contracts Chart | Etherscan, URL: http://etherscan.io/chart/verified-contracts. Son Erişim Tarihi: 20.02.2023.

[68] F. Contro, M. Crosara, M. Ceccato, ve M. D. Preda. (2021). EtherSolve: Computing an Accurate Control-Flow Graph from Ethereum Bytecode, içinde *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, May. 2021, ss. 127-137. doi: 10.1109/ICPC52881.2021.00021.

[69] M. Suiche, "Porosity: A Decompiler For Blockchain-Based Smart Contracts Bytecode", 2017. Çevrimiçi erişim adresi: https://infocon.org/cons/DEF%20CON/DEF%20CON%2025/DEF%20CON%2025%20presentations/DEF%20CON%2025%20-%20Matt-Suiche-Porosity-Decompiling-Ethereum-Smart-Contracts-WP.pdf

[70] L. Brent *vd.*,(2022). Vandal: A Scalable Security Analysis Framework for Smart Contracts, *arXiv:1809.03981 [cs]*, Eyl. 2018, Erişim: 22 Mart 2022. [Çevrimiçi]. Erişim adresi: http://arxiv.org/abs/1809.03981

[71] İnternet: Palkeo, Panoramix. URL: https://github.com/palkeo/panoramix Son Erişim Tarihi: 30.12.2022.

[72] J. Xu, F. Dang, X. Ding, ve M. Zhou, "A Survey on Vulnerability Detection Tools of Smart Contract Bytecode", içinde *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, Eyl. 2020, ss. 94-98. doi: 10.1109/ICISCAE51034.2020.9236931.

[73] İnternet: msuiche/porosity: *UNMAINTAINED* Decompiler and Security Analysis tool for Blockchain-based Ethereum Smart-Contracts. URL:https://github.com/msuiche/porosity. Son Erişim Tarihi: 24.02.2023.

[74] H. Jordan, B. Scholz, ve P. Subotić, "Soufflé: On Synthesis of Program Analyzers", içinde Computer Aided Verification, S. Chaudhuri ve A. Farzan, Ed., içinde Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, ss. 422-430. doi: 10.1007/978-3-319-41540-6_23.

[75] M. Angelo ve G. Salzer. (2022). A Survey of Tools for Analyzing Ethereum Smart Contracts. Erişim: 05 Ekim 2022. Çevrimiçi erişim adresi: https://www.researchgate.net/publication/334786201_A_Survey_of_Tools_for_Analyzing_Ethereum_Smart_Contracts

[76] B. Schwarz, S. Debray, ve G. Andrews. (2022). Disassembly of executable code revisited, *Ninth Working Conference on Reverse Engineering, 2002. Proceedings.*, Kas. 2002, ss. 45-54. doi: 10.1109/WCRE.2002.1173063.

[77] B. C. Housel, "A Study of Decompiling Machine Language into High-Level Machine Independent Languages". Purdue Üniversitesi, 1973.

[78] Y. Shoshitaishvili *vd.*.(2016). SOK: (State of) The Art of War: Offensive Techniques in Binary Analysis, *2016 IEEE Symposium on Security and Privacy (SP)*, May. 2016, ss. 138-157. doi: 10.1109/SP.2016.17.

[79] İnternet: Go Ethereum. URL: https://geth.ethereum.org/docs/rpc/ns-admin Son Erişim Tarihi:13.07.2022.

[80] F. R. Cogo, G. A. Oliva, ve A. E. Hassan. (2022). Deprecation of Packages and Releases in Software Ecosystems: A Case Study on NPM, *IEEE Transactions on Software Engineering*, c. 48, sy 7, ss. 2208-2223, Tem. 2022, doi: 10.1109/TSE.2021.3055123.

[81] İnternet: Truffle Documentation - Truffle Suite. URL: https://trufflesuite.com/docs/ Son Erişim Tarihi: 01.01.2023

[82] P. Labs. (2017). Filecoin: A Decentralized Storage Network. Erişim Adresi: https://filecoin.io/filecoin.pdf. Son Erişim Tarihi: 01.01.2019.

[83] İnternet: Chainlink. URL: https://chain.link/ Son Erişim Tarihi:19.12.2019.

[84] İnternet: Go Ethereum. URL: https://geth.ethereum.org/docs/rpc/ns-admin Son Erişim Tarihi:13.07.2022.

[85] İnternet: Hardhat | Ethereum development environment for professionals by Nomic Foundation. URL: https://hardhat.org. Son Erişim Tarihi:17.06.2023.

[86] İnternet: J. Howell, Hardhat Vs Truffle - Key Differences, URL: https://101blockchains.com/hardhat-vs-truffle/ Son Erişim Tarihi:17.06.2023.

[87] İnternet: Remix - Ethereum IDE. URL:https://remix.ethereum.org Son Erişim Tarihi:13.07.2022.

[88] İnternet: Metamask, The crypto wallet for Defi, Web3 Dapps and NFTs. URL: https://metamask.io/. Son Erişim Tarihi:17.06.2023

[89] İnternet: "Document search - Web of Science Core Collection". URL: https://www.webofscience.com/wos/woscc/basic-search. Son Erişim Tarihi: 18.06.2023

[90] D. Appelbaum ve R. A. Nehmer. (2020). Auditing Cloud-Based Blockchain Accounting Systems, *J. Inf. Syst.*, c. 34, sy 2, ss. 5-21, SUM 2020, doi: 10.2308/isys-52660.

[91] S.-S. Jung, S.-J. Lee, ve I.-C. Euom. (2021). Delegation-Based Personal Data Processing Request Notarization Framework for GDPR Based on Private Blockchain, *Appl. Sci.-Basel*, c. 11, sy 22, s. 10574, Kas. 2021, doi: 10.3390/app112210574.

[92] A. Stell, V. Chauhan, ve R. Sinnott. (2020). Secure Audit in Support of an Adrenal Cancer Registry, *Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies, Vol 5: Healthinf*, F. Cabitza, A. Fred, ve H. Gamboa, Ed., Setubal: Scitepress, 2020, ss. 237-248. doi: 10.5220/0009160402370248.

[93] J. H. F. Battisti, G. P. Koslovski, M. A. Pillon, C. C. Miers, ve N. M. Gonzalez. (2022). Analysis of an Ethereum Private Blockchain Network Hosted by Virtual Machines Against Internal DoS Attacks, *Advanced Information Networking and Applications, Aina-2022, Vol 1*, L. Barolli, F. Hussain, ve T. Enokido, Ed., Cham: Springer International Publishing Ag, 2022, ss. 479-490. doi: 10.1007/978-3-030-99584-3_42.

[94] X. Hu *vd.* (2021). Towards Efficient Co-audit of Privacy-Preserving Data on Consortium Blockchain via Group Key Agreement, *2021 17th International Conference on Mobility, Sensing and Networking (msn 2021)*, Los Alamitos: Ieee Computer Soc, 2021, ss. 494-501. doi: 10.1109/MSN53354.2021.00079.

[95] C. Schaefer ve C. Edman. (2023). Transparent Logging with Hyperledger Fabric, *2019 Ieee International Conference on Blockchain and Cryptocurrency (icbc)*, New York: Ieee, 2019, ss. 65-69. Erişim: 18 Haziran 2023. [Çevrimiçi]. Erişim adresi: https://www.webofscience.com/wos/woscc/full-record/WOS:000491257000020

[96] Z. Song, X. Zhang, ve M. Liang. (2021). Reliable Reputation Review and Secure Energy Transaction of Microgrid Community Based on Hybrid Blockchain, *Wirel. Commun. Mob. Comput.*, c. 2021, s. 9916735, Haz. 2021, doi: 10.1155/2021/9916735.

[97] T. K. Dasaklis, F. Casino, ve C. Patsakis. (2020). A traceability and auditing framework for electronic equipment reverse logistics based on blockchain: the case of mobile phones *2020 11th International Conference on Information, Intelligence, Systems and Applications (iisa 2020)*, New York: Ieee, 2020, ss. 295-301. Erişim: 18 Haziran 2023. [Çevrimiçi]. Erişim adresi: https://www.webofscience.com/wos/woscc/full-record/WOS:000659941900043

[98] C. Munoz-Ausecha, J. E. G. Gomez, J. Ruiz-Rosero, ve G. Ramirez-Gonzalez. (2023). Asset Ownership Transfer and Inventory Using RFID UHF TAGS and Ethereum Blockchain NFTs, *Electronics*, c. 12, sy 6, s. 1497, Mar. 2023, doi: 10.3390/electronics12061497.

[99] V. Malamas, T. K. Dasaklis, P. Kotzanikolaou, M. Burmester, ve S. Katsikas. (2019) A forensics-by-design management framework for medical devices based on blockchain, *2019 Ieee World Congress on Services (ieee Services 2019)*, C. K. Chang, P. Chen, M. Goul, K. Oyama, S. ReiffMarganiec, Y. Sun, S. Wang, ve Z. Wang, Ed., Los Alamitos: Ieee Computer Soc, 2019, ss. 35-40. doi: 10.1109/SERVICES.2019.00021.

[100] S. Aggarwal ve N. Kumar. (2021). Core components of blockchain, *Blockchain Technology for Secure and Smart Applications Across Industry Verticals*, c. 121, S.

Aggarwal, N. Kumar, ve P. Raj, Ed., San Diego: Elsevier Academic Press Inc, 2021, ss. 193-209. doi: 10.1016/bs.adcom.2020.08.010.

[101] S. Luo *vd.*, (2021). Blockchain-Based Task Offloading in Drone-Aided Mobile Edge Computing, *IEEE Netw.*, c. 35, sy 1, ss. 124-129, Şub. 2021, doi: 10.1109/MNET.011.2000222.

[102] T. K. Dasaklis, F. Casino, C. Patsakis, ve C. Douligeris. (2019). A Framework for Supply Chain Traceability Based on Blockchain Tokens, *Business Process Management Workshops (bpm 2019)*, C. DiFrancescomarino, R. Dijkman, ve U. Zdun, Ed., Cham: Springer International Publishing Ag, 2019, ss. 704-716. doi: 10.1007/978-3-030-37453-2_56.

[103] M. Abouyoussef ve M. Ismail. (2021). Blockchain-based Networking Strategy for Privacy-Preserving Demand Side Management, *Ieee International Conference on Communications (icc 2021)*, New York: Ieee, 2021. doi: 10.1109/ICC42927.2021.9500789.

[104] M. Zichichi, S. Ferretti, G. D'Angelo, ve V. Rodriguez-Doncel. (2022). Data governance through a multi-DLT architecture in view of the GDPR, *Cluster Comput.*, Ağu. 2022, doi: 10.1007/s10586-022-03691-3.

[105] M. M. H. Onik, C.-S. Kim, N.-Y. Lee, ve J. Yang. (2019). Privacy-aware blockchain for personal data sharing and tracking, *Open Comput. Sci.*, c. 9, sy 1, ss. 80-91, Nis. 2019, doi: 10.1515/comp-2019-0005.

[106] İnternet: MultiChain | Open source blockchain platform. URL: https://www.multichain.com/. Son Erişim Tarihi: 29.12.2023.

[107] D. E. O'Leary. (2017). Configuring blockchain architectures for transaction information in blockchain consortiums: The case of accounting and supply chain systems, *Intelligent Systems in Accounting, Finance and Management*, c. 24, sy 4, ss. 138-147, 2017, doi: 10.1002/isaf.1417.

[108] A. Raza, L. Hardy, E. Roehrer, S. Yeom, ve B. H. Kang. (2021). GPSPiChain-Blockchain and AI based Self-Contained Anomaly Detection Family Security System in Smart Home, *J. Syst. Sci. Syst. Eng.*, c. 30, sy 4, ss. 433-449, Ağu. 2021, doi: 10.1007/s11518-021-5496-2.

[109] M. Ahmed, S. Reno, N. Akter, ve F. Haque. (2020). Securing Medical Forensic System Using Hyperledger Based Private Blockchain, *2020 23rd International*

Conference on Computer and Information Technology (iccit 2020), New York: Ieee, 2020, s. 215. doi: 10.1109/ICCIT51783.2020.9392686.

[110] K. Awuson-David, T. Al-Hadhrami, O. Funminiyi, ve A. Lotfi. (2020). Using Hyperledger Fabric Blockchain to Maintain the Integrity of Digital Evidence in a Containerised Cloud Ecosystem, *Emerging Trends in Intelligent Computing and Informatics: Data Science, Intelligent Information Systems and Smart Computing*, F. Saeed, F. Mohammed, ve N. Gazem, Ed., Cham: Springer International Publishing Ag, 2020, ss. 839-848. doi: 10.1007/978-3-030-33582-3_79.

[111] F. E. Alruwaili. (2021). CustodyBlock: A Distributed Chain of Custody Evidence Framework, *Information*, c. 12, sy 2, s. 88, Şub. 2021, doi: 10.3390/info12020088.

[112] J. Moreno, M. A. Serrano, E. B. Fernandez, ve E. Fernandez-Medina. (2020). Improving Incident Response in Big Data Ecosystems by Using Blockchain Technologies, *Appl. Sci.-Basel*, c. 10, sy 2, s. 724, Oca. 2020, doi: 10.3390/app10020724.

[113] A. Fakieh ve A. Akremi, "An Effective Blockchain-Based Defense Model for Organizations against Vishing Attacks", *Appl. Sci.-Basel*, c. 12, sy 24, s. 13020, Ara. 2022, doi: 10.3390/app122413020.

[114] H.-C. Chen *vd.*, "An Implementation of Trust Chain Framework with Hierarchical Content Identifier Mechanism by Using Blockchain Technology", *Sensors*, c. 22, sy 13, s. 4831, Tem. 2022, doi: 10.3390/s22134831.

[115] A. Jesus Diaz-Honrubia *vd.*, "An overview of the CUREX platform", içinde *2019 Ieee 32nd International Symposium on Computer-Based Medical Systems (cbms)*, New York: Ieee, 2019, ss. 162-167. doi: 10.1109/CBMS.2019.00042.

[116] A. Morteza, M. Ilbeigi, ve J. Schwed, "A Blockchain Information Management Framework for Construction Safety", içinde *Computing in Civil Engineering 2021*, R. R. A. Issa, Ed., New York: Amer Soc Civil Engineers, 2022, ss. 342-349. Erişim: 18 Haziran 2023. [Çevrimiçi]. Erişim adresi: https://www.webofscience.com/wos/woscc/full-record/WOS:000948933100043

[117] E. Abad-Segura, A. Infante-Moro, M.-D. Gonzalez-Zamar, ve E. Lopez-Meneses, "Blockchain Technology for Secure Accounting Management: Research Trends Analysis", *Mathematics*, c. 9, sy 14, s. 1631, Tem. 2021, doi: 10.3390/math9141631.

[118] M. D. Sheldon, "Auditing the Blockchain Oracle Problem", *J. Inf. Syst.*, c. 35, sy 1, ss. 121-133, SPR 2021, doi: 10.2308/ISYS-19-049.

[119] N. Yalçın, "KPMG Vak'ası: PCAOB Verilerinin Yasadışı Kullanımı ve Eğitim Sınavlarında Hile Yapılması", *MED*, c. 0, sy 63, Haz. 2020, doi: 10.26650/MED.740011.

[120] M. T. Fulop, D. I. Topor, C. A. Ionescu, S. Capusneanu, T. O. Breaz, ve S. G. Stanescu, "Fintech Accounting and Industry 4.0: Future-Proofing or Threats to the Accounting Profession?", *J. Bus. Econ. Manag.*, c. 23, sy 5, ss. 997-1015, 2022, doi: 10.3846/jbem.2022.17695.

[121] D. Appelbaum, D. S. Showalter, T. Sun, ve M. A. Vasarhelyi, "A Framework for Auditor Data Literacy: A Normative Position", *Account. Horiz.*, c. 35, sy 2, ss. 5-25, Haz. 2021, doi: 10.2308/HORIZONS-19-127.

[122] A. Albizri ve D. Appelbaum, "Trust but Verify: The Oracle Paradox of Blockchain Smart Contracts", *J. Inf. Syst.*, c. 35, sy 2, ss. 1-16, SUM 2021, doi: 10.2308/ISYS-19-024.

[123] H. Han, R. K. Shiwakoti, R. Jarvis, C. Mordi, ve D. Botchie, "Accounting and auditing with blockchain technology and artificial Intelligence: A literature review", *International Journal of Accounting Information Systems*, c. 48, s. 100598, Mar. 2023, doi: 10.1016/j.accinf.2022.100598.

[124] C. S. Sargent, "Replacing Financial Audits with Blockchain: The Verification Issue", *J. Comput. Inf. Syst.*, c. 62, sy 6, ss. 1145-1153, Kas. 2022, doi: 10.1080/08874417.2021.1992805.

[125] C.-C. Chou, N.-C. R. Hwang, G. P. Schneider, T. Wang, C.-W. Li, ve W. Wei, "Using Smart Contracts to Establish Decentralized Accounting Contracts: An Example of Revenue Recognition", *J. Inf. Syst.*, c. 35, sy 3, ss. 17-52, FAL 2021, doi: 10.2308/ISYS-19-009.

[126] R. T. Dunn, J. G. Jenkins, ve M. D. Sheldon, "Bitcoin and Blockchain: Audit Implications of the Killer Bs", *Iss. Account. Educ.*, c. 36, sy 1, ss. 43-56, Şub. 2021, doi: 10.2308/ISSUES-19-049.

[127] M. A. Agusti ve M. Orta-Perez, "Big data and artificial intelligence in the fields of accounting and auditing: a bibliometric analysis", *Span. J. Financ. Account.*, Tem. 2022, doi: 10.1080/02102412.2022.2099675.

[128] D. D. Jayasuriya ve A. Sims, “From the abacus to enterprise resource planning: is blockchain the next big accounting tool?”, *Account. Audit Account.*, c. 36, sy 1, ss. 24-62, Oca. 2023, doi: 10.1108/AAAJ-08-2020-4718.

[129] P. J. Schmidt ve G. Gal, “INTRODUCTION Editors’ Introduction to the Special Section on Cloud Computing and Accounting Information Systems (JISC 2018)”, *J. Inf. Syst.*, c. 34, sy 2, ss. 1-4, SUM 2020, doi: 10.2308/isys-10726.

[130] S.-F. Hsieh ve G. Brennan, “Issues, risks, and challenges for auditing crypto asset transactions”, *Int. J. Account. Inf. Syst.*, c. 46, s. 100569, Eyl. 2022, doi: 10.1016/j.accinf.2022.100569.

[131] İnternet: Compound Audit - OpenZeppelin blog. URL: http://blog.openzeppelin.com/compound-audit. Son Erişim Tarihi: 21.06.2023.

[132] İnternet: Basic Attention Token (BAT) Audit - OpenZeppelin blog. URL: http://blog.openzeppelin.com/basic-attention-token-bat-audit Son Erişim Tarihi : 21.06.2023.

[133] İnternet: OpenZeppelin. URL: https://www.openzeppelin.com. Son Erişim Tarihi: 21.06.2023.

[134] K. Hoang, Y. Luo, ve S. E. Salterio, “Evidence-Informed Audit Standard Setting: Exploring Evidence Use and Knowledge Transfer*”, *Contemp. Account. Res.*, c. 39, sy 4, ss. 2243-2283, Eki. 2022, doi: 10.1111/1911-3846.12795.

[135] L. S. Lee, D. Appelbaum, ve R. D. Mautz III, “Blockchains: An Experiential Accounting Learning Activity”, *J. Emerg. Technol. Account.*, c. 19, sy 1, ss. 181-197, 2022, doi: 10.2308/JETA-2020-009.

[136] Y. Li ve A. H. Juma’h, “The Effect of Technological and Task Considerations on Auditors’ Acceptance of Blockchain Technology”, *J. Inf. Syst.*, c. 36, sy 3, ss. 129-151, 2022, doi: 10.2308/ISYS-2020-022.

[137] M. D. Sheldon, “Tracking Tangible Asset Ownership and Provenance with Blockchain”, *J. Inf. Syst.*, c. 36, sy 3, ss. 153-175, 2022, doi: 10.2308/ISYS-2020-042.

[138] N. E. Vincent ve S. A. Davenport, "Accounting Research Opportunities for Cryptocurrencies", *J. Emerg. Technol. Account.*, c. 19, sy 1, ss. 79-93, 2022, doi: 10.2308/JETA-19-11-14-46.

[139] İnternet: Diffusion of Innovation Theory. URL: https://sphweb.bumc.bu.edu/otlt/mph-modules/sb/behavioralchangetheories/behavioralchangetheories4.html. Son Erişim Tarihi: 10.06.2023

[140] Z. S. Alzamil, D. Appelbaum, W. Glasgall, ve M. A. Vasarhelyi, "Applications of Data Analytics: Cluster Analysis of Not-forProfit Data", *J. Inf. Syst.*, c. 35, sy 3, ss. 199-221, FAL 2021, doi: 10.2308/ISYS-2020-025.

[141] P. Caster, R. J. Elder, ve D. J. Janvrin, "An Exploration of Bank Confirmation Process Automation: A Longitudinal Study", *J. Inf. Syst.*, c. 35, sy 3, ss. 1-16, FAL 2021, doi: 10.2308/ISYS-19-038.

[142] J. Chen, X. Xia, D. Lo, ve J. Grundy, "Why Do Smart Contracts Self-Destruct? Investigating the Selfdestruct Function on Ethereum", *ACM Trans. Softw. Eng. Methodol.*, c. 31, sy 2, ss. 1-37, Nis. 2022, doi: 10.1145/3488245.

[143] İnternet: Ethereum API | IPFS API & Gateway | ETH Nodes as a Service. URL: https://www.infura.io. Son Erişim Tarihi: 22.06.2023.

[144] İnternet: Upgrading smart contracts, URL: ethereum.org. Son Erişim Tarihi: 18.10.2023

[145] İnternet: Solidity — Solidity 0.8.21 documentation". URL: https://docs.soliditylang.org/en/v0.8.21/index.html, Son Erişim Tarihi: 19.10.2023

[146] İnternet: Solidity Tutorial - A Detailed Introduction, D. Geroni, 101 Blockchains. Çevrimiçi erişim adresi: https://101blockchains.com/solidity-tutorial/, Son Erişim Tarihi: 19.10.2023

[147] İnternet: DelegateCall: Calling Another Contract Function in Solidity, zeroFurit, Coinmonks. Çevrimiçi erişim adresi: https://medium.com/coinmonks/delegatecall-calling-another-contract-function-in-solidity-b579f804178c, Son Erişim Tarihi: 19.10.2023

[148] K. P. Tripathi. (2011). A Review on Knowledge-based Expert System: Concept and Architecture, *Artificial Intelligence Techniques*, 2011.

[149] H. Başak, İ. Şahin, ve M. Gülen, "İnsansız hava aracı kazalarının önlenmesi için uzman sistemedayalı risk yönetim modeli", *Teknoloji*, c. 11, sy 3, 2008, Erişim: 31 Ekim 2023. [Çevrimiçi]. Erişim adresi: https://avesis.gazi.edu.tr/yayin/9b2e89ce-e20e-4daf-8d91-d2d47063fbc4/insansiz-hava-araci-kazalarinin-onlenmesi-icin-uzman-sistemedayali-risk-yonetim-modeli

[150] İ. Şahin ve H. R. Bölüklü, "2B GÖRÜNÜSLERDEN OTOMATİK KATI MODELLER OLUSTURMADA UZMAN BİR YAKLASIM", *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, c. 14, sy 2, Art. sy 2, Şub. 2008.

[151] N. Top ve İ. Şahin, "Uzman Sistemler", *Yapay Zeka - Kuramdan Uygulamaya*, Nobel Akademik Yayıncılık, ss. 69-93. Çevrimiçi erişim adresi: https://www.nobelyayin.com/kitap_19599.html

[152] N. Allahverdi, *Uzman Sistemler: Bir Yapay Zeka Uygulaması*. Atlas Yayın Dağıtım, 2002. Erişim: 09 Kasım 2023. Çevrimiç erişim adresi: https://www.nobelyayin.com/uzman-sistemler-bir-yapay-zeka-uygulamasi-2127.html

[153] M. J. Adams, CHEMOMETRICS AND STATISTICS | Expert Systems, *Encyclopedia of Analytical Science (Second Edition)*, P. Worsfold, A. Townshend, ve C. Poole, Ed., Oxford: Elsevier, 2005, ss. 33-40. doi: 10.1016/B0-12-369397-7/00081-9.

[154] İ. ŞAHİN, M. CALP, and A. ÖZKAN, "An Expert System Design and Application for Hydroponics Greenhouse Systems", Gazi University Journal of Science, vol. 27, no. 2, pp. 809–822, 2014.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : AÇIKALIN, Kevser
Uyruğu : T.C.
Doğum tarihi ve yeri :
Medeni hali :
Telefon :
e-mail :

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Doktora	Gazi Üniversitesi / Adli Bilişim	Devam ediyor
Yüksek lisans	Fırat Üniversitesi / Bilgisayar Mühendisliği	2012
Lisans	Fırat Üniversitesi / Bilgisayar Mühendisliği	2008
Lise	Elazığ Anadolu Lisesi	2004

İş Deneyimi

Yıl	Yer	Görev
2016-Halen	Rekabet Kurumu	Rekabet Uzmanı
2011-2016	Hazine Müsteşarlığı	Programcı
2009-2011	Sağlık Bakanlığı	Programcı
2008-2009	YD Yazılım	Bilgisayar Mühendisi

Yabancı Dil

İngilizce

Yayınlar

1. K. Açıkalın And İ. ŞAHİN, (2023). Blok Zincir Altyapısı ile Devlet Desteklerinde Mükerrerliğin Önlenmesi için Bir Model Önerisi, **Politeknik Dergisi/Journal of Polytechnic**. 1-1.

GAZİLİ OLMAK AYRICALIKTIR.