



**BLOK ZİNCİRİ TABANLI KRİPTO
PARALARIN ÇALIŞMA YÖNTEMLERİNİN
ARAŞTIRILMASI**

Engin AKKOCA

Yüksek Lisans Tezi

**Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Doç. Dr. ERCAN BULUŞ
2021**

T.C.
TEKİRDAĞ NAMIK KEMAL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

**BLOK ZİNCİRİ TABANLI KRİPTO PARALARIN
ÇALIŞMA YÖNTEMLERİNİN ARAŞTIRILMASI**

Engin AKKOCA

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN: Doç. Dr. ERCAN BULUŞ

TEKİRDAĞ-2021

Her hakkı saklıdır.

ÖZET

Yüksek Lisans Tezi

BLOK ZİNCİRİ TABANLI KRİPTO PARALARIN ÇALIŞMA YÖNTEMLERİNİN ARAŞTIRILMASI

Engin AKKOCA

Tekirdağ Namık Kemal Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Ercan BULUŞ

Bir e-posta grubuna 2008 senesinde Satoshi Nakamoto tarafından gönderilen bir makale ile tanıtılan ve 2009 senesinde kullanımına başlanılan kripto para teknolojisi geçen zaman zarfında, dünya üzerinde gerek ekonomik gerekse teknolojik bir çok unsuru değiştirmiş ve bir devrim olarak kabul edilmiştir. Devletlerden, kişilerden ve sistemlerden bağımsız çalışan kripto paralar, merkezi olmadan, para transfer işlemlerinde aracı kurumları ortadan kaldırırken, büyük bir yatırım aracı haline dönüşmüş ve giderek kullanımı artmıştır. Özellikle son zamanlarda devletlerin gündemlerine aldıkları kripto paraların nasıl çalıştıkları, hangi algoritmaları kullandıkları, kullanılan bu algoritmaları nasıl bir senkronizasyon ile çalıştırdıkları gibi konularda kaynak bulmak zor olduğu gibi bir anlam karmaşası da yaşanmaktadır. Ayrıca kripto paraların nasıl olup da devletlerin dahi müdahale edemeyecekleri, dünyada elektrik ve internet olduğu sürece bu sistemin nasıl çalışmaya devam edeceği birçok kişi tarafından merak konusu olmuştur. Zaman içerisinde kripto paraların hem çalışma düzenleri hem de güvenlikleri açısından bir takım problemler ortaya çıkmıştır. Bu tez çalışmasında kripto paraların, Bitcoin özelinde, hangi algoritmaları nasıl bir senkronizasyon ile kullandıkları, blok zinciri teknolojisi ile kripto paraların nasıl bir ilişkisi olduğu gibi konular detaylı olarak anlatılmış ve geliştirilmeleri açısından bir takım önerilerde bulunulmuştur. Son olarak, bu tezde yer alan önerilerin uygulandığı bir kripto para simülatörü geliştirilmiş ve önerilerin uygulanabilirliği gösterilmiştir.

Anahtar kelimeler: Kriptoloji, Kripto Para, Blok Zinciri

2021, 84 sayfa

ABSTRACT

MSc. Thesis

INVESTIGATION OF THE WORKING METHODS OF BLOCKCHAIN BASED CRYPTO CURRENCIES

Engin AKKOCA

Tekirdağ Namık Kemal University

Graduate School of Natural and Applied Sciences

Department of Computer Engineering

Supervisor: Doç. Dr. Ercan BULUŞ

Cryptocurrency technology which has been introduced into a mailing list with a paper by Satoshi Nakamoto in 2008 and is being used since 2009 has changed many aspects in both finance and technology and is regarded as a revolution. Cryptocurrencies working independently of the governments, persons and systems without central bodies, has turned into a very popular means of investment and increased in demand while eliminating intermediary companies of money transfer. Governments has been closely monitoring cryptocurrencies particularly in the recent years and trying to find out how they work and utilise, and synchronise their algorithms especially in a scarsely sourced and confusing environment. In addition, cryptocurrencies are a topic of debate by many regarding how they will be able to operate even when the governmental bodies cannot intervene and the fact that the cryptocurrency system will be operational as long as it has the essential supplies such as power and internet. Cryptocurrencies have over time developed certain problems regarding security and how they work. In this dissertation, it has been detailed how the cryptocurrencies, specifically Bitcoin, utilise their synronisation with their algorithms and the type of their relationship with the block-chain technology along with a number of suggestions for further development. Lastly, in this study, a cryptocurrency simulator has been developed demonstrating the applicability of the suggested development strategies in practice.

Key words: Cryptology, Cryptocurrency, Block Chain

2021, 84 pages

İÇİNDEKİLER

ÖZET	i
ABSTRACT	ii
İÇİNDEKİLER.....	iii
ÇİZELGE DİZİNİ.....	v
ŞEKİL DİZİNİ.....	vi
SİMGELER ve KISALTMALAR.....	viii
TEŞEKKÜR.....	ix
1. GİRİŞ.....	1
2. KRİPTOLOJİ	3
2.1. Kriptografi	3
2.1.1. Simetrik Kriptografi Algoritmaları.....	4
2.1.1.1. Blok şifreleme	5
2.1.1.2. Akış şifreleme	8
2.1.2. Asimetrik Kriptografi Algoritmaları	10
2.1.2.1. RSA algoritması.....	12
2.1.2.2. Eliptik eğri şifreleme algoritması.....	13
2.1.3. Kriptografik Özet (Hash) Algoritmaları	15
2.1.3.1. MD5 kriptografik özet algoritması	16
2.1.3.2. SHA 256 kriptografik özet algoritması.....	19
2.1.3.3. SHA 512 kriptografik özet algoritması.....	24
2.2. Kriptanaliz	30
3. DİJİTAL İMZA	31
4. BLOK ZİNCİRİ (BLOCK CHAIN)	34
4.1. Blok Zinciri Çeşitleri	37
4.1.1. Açık Blok Zinciri (Public Block Chain).....	37
4.1.2. Özel Blok Zinciri (Private Blockchain).....	37
4.1.3. Konsorsiyum Blok Zinciri (Consortium Blockchain)	38
4.2. Blok.....	38
4.2.1. Zaman Damgası	39
4.2.2. Önceki Bloğun Özet Değeri	39
4.2.3. Nonce Değeri	39
5. BITCOIN	41

5.1. Bitcoin Blok Yapısı ve Madencilik (Mining) İşlemi	42
5.1.1. Blok Başlığı	44
5.1.1.1. Versiyon	45
5.1.1.2. Önceki bloğun özet değeri	45
5.1.1.3. Merkle kökünün özet değeri	46
5.1.1.4. Zaman damgası	48
5.1.1.5. Hedef	48
5.1.1.6. Nonce	50
5.1.1.7. Genişletme + Uzunluk	51
5.1.2. Blok Gövdesi	52
5.1.2.1. Transfer işlemlerinin yapısı	52
6. ÖNERİLER VE GELİŞTİRİLEN UYGULAMA	59
6.1. Bitcoin Sistemi İçin Öneriler	59
6.2. Geliştirilen Uygulama	60
6.2.1. Para Gönderme Algoritması	61
6.2.2. Bakiye Hesaplama Algoritması	62
6.2.3. Blok Özeti Bulma (Madencilik) Algoritması	64
6.2.4. Blok Zincirine Blok Eklenmesi	65
6.2.5. Blok Zincirinin Doğrulanması	66
6.2.6. Simülatör Programı Ağ Yapısı	67
6.3. Yapılan Testler ve Sonuçları	67
6.3.1. Ritmik Artan Nonce Değeri İle Yapılan Test ve Sonuçları	67
6.3.2. Rasgele Değişen Nonce Değeri İle Yapılan Test ve Sonuçları	69
6.4. Sonuç	69
KAYNAKLAR	71
ÖZGEÇMİŞ	Hata! Yer işareti tanımlanmamış.

ÇİZELGE DİZİNİ

Çizelge 2.1. Genişleme işleminde kullanılan tablo	7
Çizelge 2.2. MD5 algoritmasının başlangıç değerleri	17
Çizelge 2.3. SHA 256 Algoritmasının başlangıç değerleri	21
Çizelge 2.4. İlk 64 asal sayının küp köklerinin ikilik (binary) sayı sistemindeki karşılıklarının ondalık kısımlarının ilk 32 bitinden elde edilen değerlerin onaltılık (hexadecimal) sayı sistemindeki karşılıkları.....	23
Çizelge 2.5. SHA 512 Algoritmasının başlangıç kelime değerleri.....	26
Çizelge 4.1. Genel, Özel ve Konsorsiyum blok zincirlerinin karşılaştırılması	38
Çizelge 5.1. Bitcoin blok başlığında yer alan alanlar	45
Çizelge 5.2. Madencinin bloğa eklediği işlemde yer alan bileşenler	51
Çizelge 5.3. Transfer işlemlerinin yapısı.....	53
Çizelge 5.4. Transfer işlemi çıktısı.....	55
Çizelge 5.5. Transfer işlemi girdisi	56
Çizelge 6.1. Bir bloğun 100 defa özet değerinin hesaplanma süreleri (Ritmik Nonce)	68
Çizelge 6.2. Bir bloğun 100 defa özet değerinin hesaplanma süreleri (Rasgele Nonce)	69

ŞEKİL DİZİNİ

Şekil 2.1. Kriptoloji bilimi alt bilim dalları	3
Şekil 2.2. DES algoritmasının genel bir gösterimi	6
Şekil 2.3. DES algoritmasındaki “f” fonksiyonunun çalışması.....	6
Şekil 2.4. DES anahtar üretme algoritması	7
Şekil 2.5. A 5/1 algoritmasının genel şeması	8
Şekil 2.6. A 5/1 algoritmasında kullanılan 3 LFSR.....	9
Şekil 2.7. Asimetrik şifreleme altyapısı	11
Şekil 2.8. Eliptik eğri örneği.....	13
Şekil 2.9. MD5 algoritmasının 64 defa tekrarlanan bölümü	17
Şekil 2.10. MD5 Algoritmasının genel mimarisi	18
Şekil 2.11. Merkle – Damgard Düzeni	19
Şekil 2.12. SHA 256 algoritmasının genel yapısı.....	20
Şekil 2.13. SHA 256 Mesaj Sıkıştırma Algoritmasının yapısı	22
Şekil 2.14. SHA-512 Mesaj Özeti	25
Şekil 2.15. SHA 512 içindeki dolgu ve alan uzunlukları	25
Şekil 2.16. SHA 512 Sıkıştırma fonksiyonu (compression function)	27
Şekil 2.17. SHA 512 sıkıştırma fonksiyonundaki her turun yapısı	28
Şekil 2.18. SHA 512 sıkıştırma algoritmasındaki A ve E kelimelerinin belirlemesi	28
Şekil 2.19. SHA 512 Kelime genişletme fonksiyonunun yapısı	30
Şekil 3.1. Elektronik imza çalışma algoritması	31
Şekil 4.1. Veri Tabanı modellerinin şematik gösterimi.....	34
Şekil 4.2. Blok zinciri yapısı	35
Şekil 5.1. Bitcoin blok başlığı ve madencilik işlemi için kullanılan özet algoritması	43
Şekil 5.2. Davis-Meyer Düzeni	44
Şekil 5.3. Bir Merkle ağacındaki düğümlerin hesaplanması	46
Şekil 5.4. Birçok veri ögesini özetleyen bir Merkle Ağacı	48
Şekil 6.1. Geliştirilen uygulamanın ekran görüntüsü	60
Şekil 6.2. Para gönderme algoritmasının akış diyagramı	61
Şekil 6.3. Bakiye kontrolü algoritmasının akış diyagramı	63
Şekil 6.4. Blok özeti bulma (madencilik) algoritmasının akış diyagramı	64
Şekil 6.5. Oluşturulan bir bloğun içeriği	65

Şekil 6.6. Blok Zinciri doğrulama algoritmasının akış diyagramı	66
Şekil 6.7. SHA256 ve SHA512 özet algoritmalarının aynı girdiyi 100 defa hesaplama süreleri grafiği (Ritmik Nonce)	68



SİMGELER VE KISALTMALAR

LFSR	: Linear Feedback Shift Registers
DES	: Data Encryption Standard
AES	: Advanced Encryption Standard
RSA	: Ron Rivest, Adi Shamir, Leonard Adleman
SHA	: Secure Hash Algorithm
NSA	: National Security Agency
BTC	: Bitcoin
TL	: Türk Lirası
UTXO	: Unspent Transaction Output

TEŞEKKÜR

Ekonomide paranın icadı büyük bir devrim olarak kabul edilmektedir. Paradan sonra ticari faaliyetler birçok yönü ile kolaylaşmış ve ticari işlemlerin büyümesinde para önemli bir rol oynamıştır. Bilgisayarların ve internetin icadı ile zaman içerisinde parasal ticari faaliyetler bilgisayar vasıtasıyla internet üzerinden gerçekleştirilmeye başlanmış ve bu durum mobil cihazlar ile hız kazanmıştır. Tüm bu gelişmeler beraberinde güvenlik sorunlarını doğurmuş, parasal işlemleri gerçekleştirmek için aracı finans kurumlarına ihtiyaç hasıl olmuştur ve kişiler bu kurumlara güvenmek zorunda kalmışlardır. Çünkü merkezi sistemlerde verilerin merkezi bir veritabanında saklanması gerekir ki veriler bu veritabanı kadar güvenlidir. Tam bu noktada para konusunda ikinci devrim niteliğinde bir sistem önerilmiştir. Kripto para adı verilen bu sistem parasal işlemlerde merkezi kuruluşları ortadan kaldırmış ve kişilerin güvenmek zorunda oldukları bir kurum veya kişi olmadan para transferi yapmalarına olanak sağlamıştır. Aynı zamanda kripto para teknolojisi, verimli şekilde kullanılmaya çalışılan blok zinciri sistemini kullandığı için bu sistemin yaygınlaşmasına yardımcı olmuştur. Kripto para teknolojisi yaklaşık 10 yıllık bir teknoloji olmasına rağmen bir çok kişi tarafından nasıl çalıştığı, kaynak eksikliği sebebi ile bilinmemektedir. Kripto paralar çıktığı günden beri geçen yaklaşık 10 yıllık zaman diliminde çok popüler hale gelmekle beraber bir takım teknik sıkıntılarla da karşı karşıya kalmışlardır.

Kripto paraların kullandıkları kriptoloji algoritmalarını, bu algoritmaları nasıl bir düzen içinde kullandıklarını ve kripto para sistemlerinin yaşamaları muhtemel bazı sıkıntılara çözüm önerileri getirdiğim bu tezde benden hiçbir yardımını esirgemeyen Namık Kemal Üniversitesi, Çorlu Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Bölüm Başkanı Doç. Dr. Ercan BULUŞ' a teşekkürü bir borç bilirim.

Haziran, 2021

Engin AKKOCA

Bilgisayar Sistemleri Öğretmeni
Bilgisayar Mühendisi

1. GİRİŞ

Para icat edildiği zamandan günümüze kadar yapısı ve kullanım şekilleri yönünden çok kez değişiklik göstermiş ve her zaman bulunduğu dönemin ekonomik ve teknolojik gelişmelerine bir şekilde uyum sağlamıştır. Ekonomide var olan ilk paralar, bakır, gümüş veya altın gibi bulunması ve işlenmesi zor, değerli olduğu toplum tarafından kabul gören madenlerden imal edilmiştir. Kağıt paralar ise ekonomi ve ticaretin daha kompleks bir hal alması ve gelişmesi ile kullanılmaya başlanmıştır (Fidan et al., 2019). Bilgisayarın ve internetin icadına para duyarsız kalmamış ve kâğıt üzerinde saklanan, gerçek ve tüzel kişilere ait para bilgileri bilgisayar ortamında saklanmaya başlamıştır. Bilgisayar kullanımının artarak insanlar arasında yaygınlaşması ve akabinde internetin icat edilip, bilgisayar kullanımıyla doğru orantılı olarak kullanımının yaygınlaşması ile para ile ilgili işlemler sadece kurumlar değil kişiler tarafından da internet üzerinden dijital olarak gerçekleştirilmeye başlanmış, özellikle akıllı mobil cihazların (akıllı cep telefonlarının) yoğun kullanımı ile banka şubeleri büyük oranda mobil cihazlara taşınmıştır. Bu durum para ilgili işlemlerde son derece önemli olan güvenlik meselesini bir kat daha arttırmış ve paranın güvenliği siber güvenlik¹ haline dönüşmüştür.

Güvenlik noktasında, zaten paranın yönetimini elinde bulunduran 3. parti kuruluşlara (bankalar, finans kuruluşları vb.) kişiler bağımlı hale gelmiş ve bu kuruluşlara güvenmek zorunda kalmışlardır. Tabii bu kuruluşlar gerçek veya tüzel kişilerin gerçekleştirmiş oldukları parasal işlemlerden ve güvenliğini sağladıkları paradan kendi paylarına düşeni fazlası ile almışlardır ve almaya devam etmektedirler. Tam bu noktada para transfer işlemlerini ve bu işlemlerin güvenliklerini 3. parti merkezi bir kuruluş yerine merkezi olmayan dağıtık bir sistemin sağlaması fikri gündeme gelmiş ve Satoshi Nakamoto takma ismi ile anılan kişi veya kişiler blok zinciri tabanlı dağıtık bir dijital para birimi önermişlerdir. 2008 yılında bir e-posta grubunda yayılan makalede önerilen sistemin bir merkezi ve sahibi olmadığı için makaleyi yayınlayan yazar veya yazarlar orijinal isimleri yerine bir takma ad kullanmışlardır. İlk olarak Satoshi Nakamoto' nun önerdiği, dijital para, sanal para, kripto para gibi değişik isimlerle anılan para birimlerinin Avrupa Merkez Bankası (2013) tarafından yapılan tanımı; “Düzenlemesi olmayan, genellikle onu geliştiren kişiler tarafından kontrol edilen ve belli bir sanal toplum tarafından kabul edilip, kullanılan dijital para.” olarak belirtilmiştir (Özbaş, 2019).

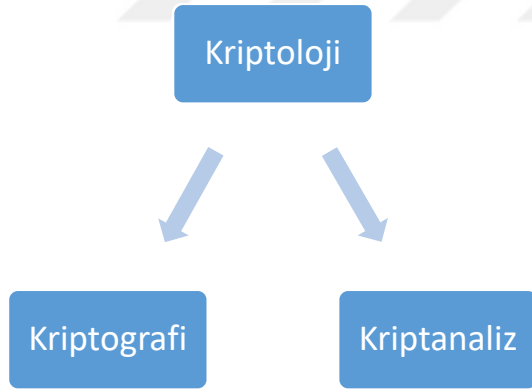
¹ Bilgisayarları, sunucuları, mobil cihazları, elektronik sistemleri, bilgisayar ağlarını ve verileri kötü amaçlı saldırılardan koruma teknikleri.

Söz konusu para olunca ekonomik ve teknik olmak üzere iki farklı inceleme alanı ortaya çıkmaktadır. Bu tezde Satoshi Nakamoto tarafından önerilen kripto para biriminin hangi kriptoloji algoritmalarını ve bu kriptoloji algoritmalarını nasıl bir senkronizasyon ile kullandığı incelenmiş ve ilgili kripto paranın kullanımı sırasında zaman içinde, gelişen teknoloji ile ortaya çıkan veya çıkması öngörülen bazı sorunlara çözümler önerilmiştir.



2. KRİPTOLOJİ

Bilginin gerek saklanması gerekse iletilmesi aşamalarında güvenliğinin ve gizliliğinin sağlanması gerekir. Bilgi güvenliğini sağlamak için kullanılan teknikler, içinde bulunulan zamana göre farklılıklar gösterse de genel olarak şifreleme yani kriptoloji teknikleridir. Kriptoloji, Yunanca saklı anlamına gelen krypto's ve kelime anlamına gelen lo'gos sözcüklerinden oluşturulmuştur ve haberleşmede gizlilik bilimi olarak bilinmektedir (Yerlikaya et al., 2006). Teknolojinin gelişmesi ile dijital ortamlarda saklanan ve işlenen veri miktarı artmış ve aynı zamanda dijital ortamlarda saklanan verilere illegal yollarla erişme yani saldırı teknikleri de geliştirilmiştir. Dijital ortamlarda metin ile birlikte resim, ses ve diğer bilgilerin de saklanması giderek artmaktadır. Günlük hayatımızda görüntü çok fazla kullanıldığı için güvenliğini sağlamak da oldukça önemli hale gelmiştir (Obaid et al., 2016). Hem bu saldırı tekniklerine karşı savunma yapabilmek için hem de teknolojiyle beraber işlemci gücünün artması sebebi ile kriptoloji algoritmaları çok daha karmaşık ve çözülmesi zor bir hal almıştır. Kriptoloji bilimi Şekil 2.1.'de görüldüğü gibi kriptografi ve kriptanaliz olmak üzere iki alt bilim dalına ayrılır. Kriptoloji, sayı bilimi, kriptografi ve kriptanalizin içeriğindeki denklem ve algoritmaların senkron olarak çalışmasıdır (Bhattacharya, 2019).



Şekil 2.1. Kriptoloji bilimi alt bilim dalları

2.1. Kriptografi

Kriptografi, bir verinin istenmeyen tarafların anlayamayacağı hale getirilmesi için verinin üzerine uygulanan tekniklerdir. Kriptografi gizlilik, bütünlük, kimlik denetimi, inkâr

edememe kavramlarını elde edebilmek için uygulanan hesaplama yöntemlerini içerir (Akleyek et al., 2011).

- **Gizlilik:** Verinin herhangi bir zamanda legal veya illegal yollarla istenmeyen kişi veya kuruluşlar tarafından ele geçirilmesi durumunda anlaşılabilmesidir. Yani verinin istenmeyen bir şekilde ele geçirilmesi durumuna karşı şifrelenip değiştirilmesi gerekir. Bu durumda ancak şifreyi bilen kişi veya kurumlar ele geçirilen bilgiyi anlamlı hale getirebilirler.

- **Bütünlük:** İstenmeyen kişi ve kuruluşlar tarafından ele geçirilen verinin tamamı veya bir bölümü değişik amaçlarla değiştirilebilir. Böyle durumlara karşı tarafa verinin hiç değiştirilmeden orijinal halinin korunduğunun doğrulanması gerekir.

- **Kimlik Denetimi:** Dijital dünyada yapılan saldırıların bir bölümü verinin iletimi esnasında gerçekleştirilir. Bu durumda gönderici, gönderdiği bilginin, göndermek istediği kişiye gittiğinden, alıcı ise eline geçen bilginin, beklediği kişiden geldiğinden emin olmak ister.

- **İnkâr Edememe:** Verinin iletimi için gerekli olan gizlilik, bütünlük, kimlik doğrulama şartları sağlanıp veri iletimi gerçekleştiikten sonra gönderici tarafın veriyi gönderdiğini, alıcı tarafın ise veriyi aldığını inkâr edememesi durumudur.

Kriptografi algoritmalarını genel olarak simetrik, asimetrik ve özet (hash) algoritmaları olarak üç bölümde inceleyebiliriz. Bu algoritmalarından simetrik şifreleme algoritmaları ise blok şifreleme algoritmaları ve akış şifreleme algoritmaları olarak iki grupta incelenebilir (Sakallı et al., 2007).

2.1.1. Simetrik Kriptografi Algoritmaları

Simetrik algoritmalarda veriyi şifrelerken kullanılan anahtar ile şifrelenmiş veriyi açarken kullanılan anahtar aynı anahtardır. Yani açık metni şifrelemek için kullanılan algoritmada bir anahtara ihtiyaç vardır ve bu anahtarı bilen herkes bu açık metni elde edebilir. Bu algoritmaların güvenlik düzeyi anahtarın gizlenme düzeyi ile doğru orantılıdır. Anahtar istenmeyen kişilerin eline geçtiği takdirde algoritmaların diğer güvenlik kriterlerinin bir anlamı kalmaz. Simetrik şifreleme algoritmaları blok şifreleme algoritmaları ve akış şifreleme algoritmaları olmak üzere iki başlık altında incelenebilir (Sakallı et al., 2007).

2.1.1.1. Blok şifreleme

Blok Şifreleme Algoritmaları veriyi bazen birbirinden bağımsız bazen de birbirine bağlı bloklar halinde şifrelemektedir veya şifresini çözmektedir (SAHİN, 2015). Blok şifreler, şifreli metin ile açık metin arasındaki ilişkiyi gizlemeye çalışırken, açık metine ait benzerlikleri şifreli metinde ortadan kaldırmayı amaçlar. Şifreli metin ile açık metin arasındaki ilişkiyi gizleme işlemine karıştırma, açık metnin benzerliklerini ortadan kaldırma işlemine ise yayılma adı verilir.

Blok şifreleme algoritmalarında anahtarın uzunluğu ya da bit sayısı en temel saldırı olan geniş anahtar arama saldırısına karşın güçlü olmalıdır. Örneğin DES algoritması 56-bit anahtar kullanır. AES algoritması DES algoritmasının açığı kapamak için 128, 192, 256 bit uzunluğunda anahtarlar kullanabilir. Anahtarların rasgele verilmesi de önemli bir unsurdur (SAHİN, 2015).

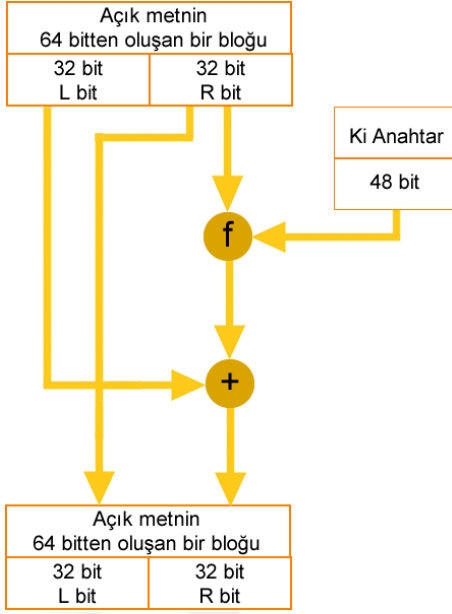
DES – Veri Şifreleme Standardı

DES, açık metni şifrelemek ve şifreli metni açmak için geliştirilmiş bir standarttır. Des şifrelenecek metni bloklara bölerek her bloğu ayrı ayrı şifreler. Şifrelenmiş bloklar arasında şifreleme aşamasında bir bağlantı yoktur. Şifrelenmiş metni açmak için ise her bloğu ayrı ayrı açar ve birleştirir. Her bloğun uzunluğu 64 bittir. Benzer biçimde 64 bit anahtar kabul eder fakat anahtarın 8 biti parity (eşlik)² olarak ayrıldığı için 56 biti tam olarak anahtar görevi yapar.

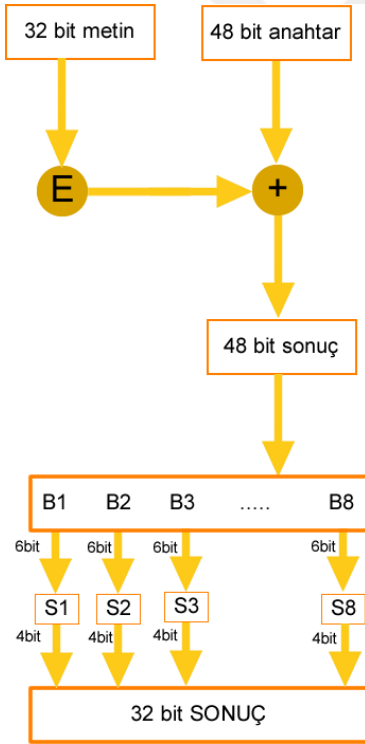
DES' in kullanım alanı oldukça geniştir ve güvenliği noktasında ise çok fazla tartışılan konu vardır (Yerlikaya et al., 2006).

Şekil 2.2.' de DES' in bir bölümü görsel olarak gösterilmiştir ve bu bölüm algoritma içerisinde 16 defa tekrar eder. Görselde ilk olarak 64 bitten oluşan açık metnin bir bloğu 32 bitten oluşan L bit ve R bit olarak adlandırılan iki parçaya bölünür. Parçalardan biri anahtar ile birlikte bir fonksiyona girdi olarak gönderilir. Görselde “f” olarak gösterilen fonksiyonun nasıl çalıştığı ise Şekil 2.3.' de gösterilmiştir. Bu fonksiyonun çıktısı ile fonksiyona girdi olarak gönderilmeyen diğer 32 bitlik blok parçasına özel veya işlemi uygulanır ve işlemin sonucu ile fonksiyona girdi olarak 32 bitlik blok parçası birleştirilerek yine 64 bitlik bir çıktı üretilir ve elde edilen çıktı üzerinde tekrar aynı işlemler uygulanır.

² İkili gösterimdeki bir sayıda 1'lerin toplamının her zaman çift ya da her zaman tek olmasını sağlayacak şekilde sözcüğe eklenen bit.



Şekil 2.2. DES algoritmasının genel bir gösterimi



Şekil 2.3. DES algoritmasındaki “f” fonksiyonunun çalışması

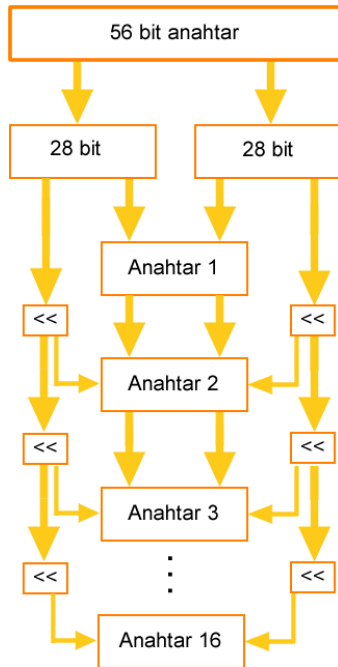
Şekil 2.3.’ de DES algoritmasının akışındaki “f” fonksiyonun içeriği görsel olarak gösterilmiştir. Fonksiyona açık metnin 64 bitlik bir bloğunun 32 bitlik iki parçasından biri ve 48 bitlik bir anahtar girdi olarak verilir ilk olarak bu girdilere özel veya uygulanır ancak

32 bitlik girdi görselde “E” ile gösterilen bir genişleme işlemi ile 48 bit yapılır. Genişleme işleminde Çizelge 2.1.’de verilen tablo kullanılır. 32 bit veri genişleme tablosuna yerleştirilir. Tablo dikkatle incelenirse bazı bitlerin birden fazla yazıldığı görülür. Bu sayede 32 bit veri 48 bit yapılır. Özel veya işlemi sonucunda elde edilen 48 bitlik çıktı 6 bitlik 8 bloğa bölünür ve her bloğa genişleme işleminin tersi uygulanarak blok uzunlukları 4 bit olarak ayarlanır. 4 bit uzunluğundaki 8 blok birleştirilerek 32 bit sonuç elde edilir.

Çizelge 2.1. Genişleme işleminde kullanılan tablo

32	1	3	1	2	4	3	6
4	5	5	7	28	30	7	9
9	7	10	8	22	11	11	15
29	12	14	17	15	16	12	18
13	23	19	21	15	17	14	18
31	20	21	20	24	25	26	27

DES’ in bir diğer önemli noktası da anahtar üretimidir. 64 bitlik anahtarın 8 biti eşlik olarak ayrıldığı için 56 bit anahtar kullanıldığına değinilmişti fakat hem Şekil 2.2.’de hem de Şekil 2.3.’ de kullanılan anahtarlar 48 bittir. 56 bitlik esas anahtardan 48 bitlik anahtarlar üretilir. Şekil 2.4.’ de DES’ in anahtar üretme algoritması verilmiştir.



Şekil 2.4. DES anahtar üretme algoritması

2.1.1.2. Akış şifreleme

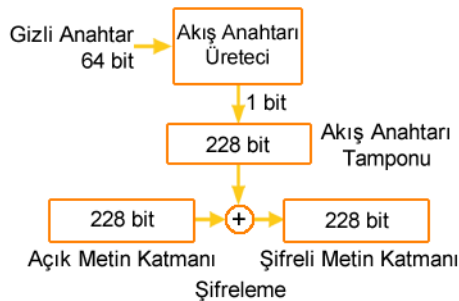
Akış şifreler daha önceden de bahsedildiği gibi açık metinde yer alan her karaktere tek seferde işlem zamanında değişen bir fonksiyon uygulayarak açık metinde bulunan karakterleri ayrı ayrı şifreler (Menezes et al., 1997). Akış şifreler temel olarak eşzamanlı ve eşzamansız olmak üzere iki başlık altında incelenebilir. Eşzamanlı akış şifrelerde anahtar dizisi, açık metin ve gizli anahtardan bağımsız olarak üretilir. Eşzamansız akış şifrelerde anahtar dizisi, bir önceki anahtarın ve şifreli metnin bir fonksiyonu ile oluşturulur (Sakallı et al., 2007).

Akış şifreler yazılımsal olarak uygulanabileceği gibi donanımsal olarak ta uygulanabilir. Özellikle bu algoritmaların donanımsal olarak tercih edilmesinin sebebi donanımsal uygulamalara uygun olmalarıdır. GSM haberleşmesinde akış şifreleme sıklıkla kullanılır.

En çok kullanılan akış şifreleme algoritmaları RC4 ve GSM haberleşmesinde kullanılan A5 algoritmalarıdır. Bu algoritmalarından RC4 rastlantısal karıştırma üzerine kurulmuştur. A5 algoritması ise doğrusal geri beslemeli saklayıcı (Linear Feedback Shift Registers- LFSR) üzerine kurulmuş fakat durum güncellemesi doğrusal olmayan algoritmalarındandır (Sakallı et al., 2007).

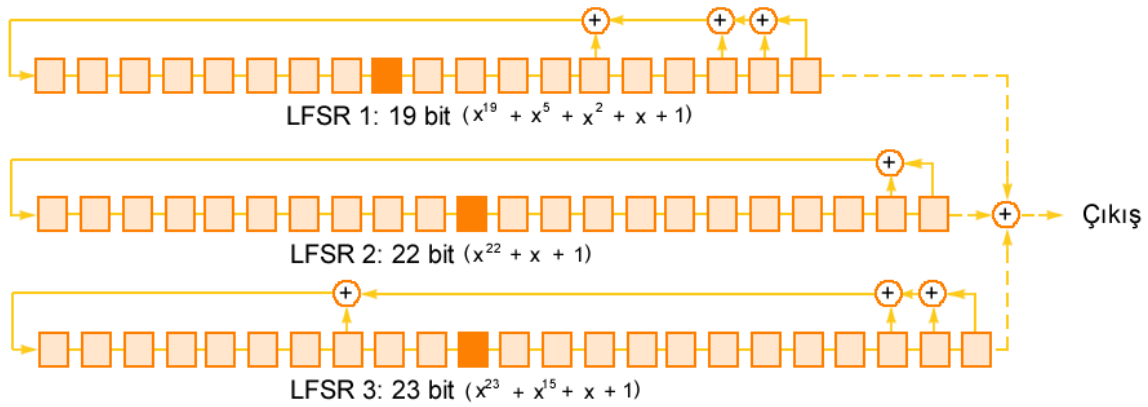
A 5/1 Akış Şifreleme Algoritması

A 5/1 algoritması A 5 şifreleme algoritmalarının bir türevidir ve GSM haberleşmesinde kullanılır. GSM üzerinden yapılan telefon haberleşmesi, her biri 4,6 milisaniye süren 228bit katmanın sıralanmasıyla yapılır. A5/1 64 bit anahtarın dışında bit akışı oluşturur. Bit akışı, 228 bit katmanla özel veya işlemine tabi tutulmak için 228 bit tamponda toplanır (B. A. Forouzan, 2007). Şekil 2.5.' de A 5/1 algoritmasının genel şeması görülmektedir.



Şekil 2.5. A 5/1 algoritmasının genel şeması

A 5/1 algoritması doğrusal geri beslemeli öteleyici saklayıcı (Linear Feedback Shift Register - LFSR) temelinde çalışan bir algoritmadır ve 19, 22 ve 23 bitlik 3 adet LFSR kullanır.



Şekil 2.6. A 5/1 algoritmasında kullanılan 3 LFSR

Şekil 2.6.' da A 5/1 algoritmasının kullandığı 3 LFSR görülmektedir. Üstteki LFSR 19, ortadaki 22, alttaki ise 23 bitten oluşmaktadır toplam 64 bit kullanılmaktadır. Şekil dikkatli incelendiğinde 8. ve 10. bitlerin koyu renkle işaretlendiği görülmektedir ve bu bitler algoritmanın ilerleyen aşamalarında çoğunluk fonksiyonuna girdi olarak gönderilecektir.

1. Algoritmanın ilk aşamasında bütün LFSR bitleri 0 yapılır ve elektronik aygıt 64 bit uzunluğunda bir oturum anahtarı üretir. Şekil 2.6.' da görüldüğü gibi LFSR dizilerinin en değersiz (en sağ) biti ve bir solundaki bit özel veya işlemine tabî tutulur. Bu işlemde üretilen değere ilk LFSR için 2. ve 5. bitler ile ve son LFSR için ise 2. ve 15. bitler ile sırası ile özel veya işlemi uygulanır. 3 LFSR' dan elde edilen özel veya işlemi sonuçları ayrı ayrı, elektronik aygıt tarafından üretilen 64 bit uzunluğundaki oturum anahtarının ilk biti ile özel veya işlemine tabî tutularak çıkan sonuç her LFSR' ın kendi dizisinin en değerli (en soldaki) bitine yazılır ve diğer bütün bitler sağa kaydırılır. Bütün bu işlemler 64 bit oturum anahtarının her biti için ayrı ayrı uygulanır.

2. Bu aşamada elektronik aygıt bu sefer 22 bit katman sayacı (frame counter) üretir ve 1. aşamadaki işlemleri aynısı 22 bit katman sayacı için uygulanır.

3. Önceki iki aşamada 64 bit oturum anahtarı ve 22 bit katman sayacı için toplam 86 tur çalışan LFSR dizileri bu aşamada oturum anahtarı veya katman sayacı gibi herhangi bir girdi almadan kendi içlerinde 100 tur önceki iki aşamadaki işlemi tekrar ederler. Fakat hepsi sürekli işlemi gerçekleştirmez. İlk LFSR için soldan 8. bit ve diğer LFSR' lar için 10. bit çoğunluk fonksiyonuna girdi olarak gönderilir. Bu bitlere tetikleme bitleri denir ve çoğunluk

fonksiyonunun sonucu ile tetikleme biti aynı olan LFSR işlemi gerçekleştirir. Örneğin tetikleme bitlerinin değerleri (1,0,1) olsun. Çoğunluk fonksiyonuna girdi olarak gönderilen (1,0,1) değerleri 1 sonucunu üretir. Çünkü (1,0,1) girdisinde 1 değerlerinin sayısı 0 değerlerinin sayısından fazladır. Bu durumda tetikleme biti çoğunluk fonksiyonunun çıktısı ile aynı olan yani 1 olan LFSR' lar çalışacaktır, 0 olanlar ise çalışmayacaktır.

4. LFSR dizileri 3. aşamada anlatıldığı gibi 100 defa karıştırıldıktan sonra 3. aşamadaki işlemler 228 tur tekrar edilir fakat her turda LFSR dizilerinin en değersiz (en sağdaki) bitlerine özel veya işlemi uygulanır ve elde edilen çıktılar ile 228 bit akış anahtarı elde edilir.

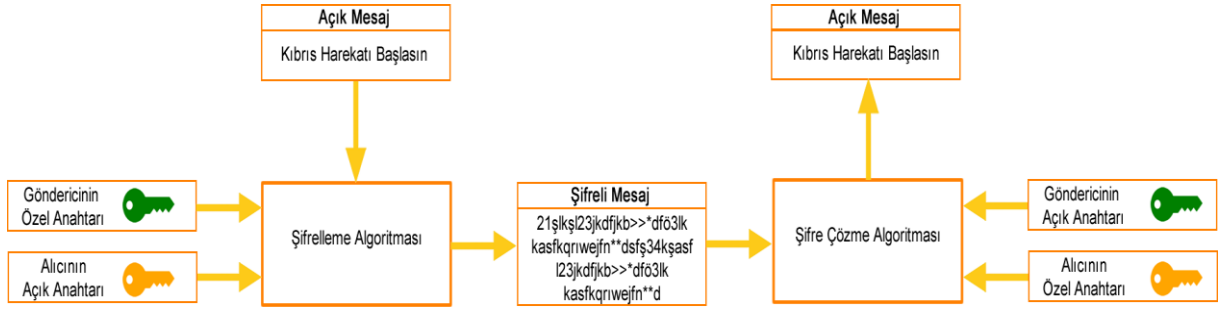
5. Son aşamada açık metin ile akış anahtarı bit bit özel veya işlemine tabi tutulur ve şifreli metin elde edilir.

Şifreli metni alan cihaz oturum numarasını ve katman sayacını bildiği takdirde aynı işlem basamaklarını uygulayarak akış anahtarını üretecek ve şifreli metni ürettiği akış anahtarı ile özel veya işlemine tabi tutunca açık metni elde edecektir.

Burada önemli olan nokta oturum anahtarı ve katman sayacının gizli tutulmasıdır. A 5/1 algoritması ilk kullanıldığı zamanlarda, oturum anahtarını ve katman sayacını ele geçiren saldırıların şifreleri metni çözmelerini engellemek için algoritma gizli tutulmuş fakat sonraları bir şekilde algoritma sızdırılmıştır. 2003 yılında Ekdahl ve Johannson, A5/1' i birkaç dakikada kıran bir saldırı yayınlamışlardır (B. A. Forouzan, 2007).

2.1.2. Asimetrik Kriptografi Algoritmaları

Simetrik şifreleme algoritmalarında tek anahtar vardır ve bu anahtar hem açık metni şifrelerken hem de şifreli metni çözerken kullanılır. Bu durum bazı güvenlik açıkları doğurmaktadır. Asimetrik şifreleme algoritmalarında ise kişiye ait iki farklı anahtar vardır. Bu anahtarlardan biri özel anahtardır (private key) ve bu anahtarı sadece kişi kendisi bilir. Diğer anahtar ise açık anahtardır (public key) ve bu anahtarı, anahtar sahibine mesaj göndermek isteyen herkes bilir. Algoritmada kullanılan matematiksel fonksiyonların özellikleri sebebiyle anahtar çiftleri her kullanıcı için farklı yani her anahtar çifti sadece bir kullanıcıya özeldir (Kodaz & Bostalı, 2010). Bir kullanıcının açık anahtarı ile şifrelenen mesaj ancak açık anahtarın sahibi olan kullanıcının özel anahtarı ile açılabilir. Bu şekilde mesajın gizliliği sağlanıp istenmeyen kişiler tarafından açılması engellenir. Benzer şekilde bir kullanıcı kendi özel anahtarı ile mesajı şifreleyip yayınladığı zaman sadece o kullanıcının açık anahtarı ile mesaj açılabilir ve böylelikle ilgili mesajın sahibi doğrulanmış olur.



Şekil 2.7. Asimetrik şifreleme altyapısı

Şekil 2.7.’ de asimetrik şifreleme algoritmalarının altyapısı görülmektedir. “Kıbrıs Harekatı Başlasın” mesajı bir yere iletilmek istenilmektedir ve bu mesaj ile ilgili güvenlik beklentileri ve asimetrik şifreleme algoritmalarının bu beklentilere getirdiği çözümler şu şekildedir:

1. Mesajı gönderen kişi mesajın bir saldırganın eline geçmesi durumunda saldırganın mesajı anlamaması için alıcının açık anahtarı ile şifreler ve şifreli mesaj sadece alıcının özel anahtarı ile açılabilir. Saldırgan mesajı ele geçirebilse bile alıcının özel anahtarını bilmediği için mesajı açamaz.

2. Mesajı gönderen kişi, alıcının, mesajı kendisinin gönderdiğinden emin olması için kendi özel anahtarı ile şifreler. Bu durumda mesaj sadece göndericinin açık anahtarı ile çözülebilir. Saldırgan mesajı ele geçirip, alıcıya gitmesini engelleyip kendi mesajını alıcıya gönderebilir. Bu durumda saldırgan göndericinin özel anahtarını bilmediği için mesajı kendi gizli anahtarı ile şifrelemek zorunda kalacaktır ve alıcı mesajı göndermesini beklediği kişinin açık anahtarı ile açamayacaktır ve mesajın bir başkasından geldiğini anlayacaktır.

Asimetrik şifreleme algoritmaları ile hem mesajın istenilen hedefe gönderilip bir saldırgan tarafından okunması engellenir hem de alıcı mesajı göndermesi gereken kişinin gönderdiğinden emin olur.

Asimetrik şifreleme algoritmaların açık anahtar ve gizli anahtar birbirlerine matematiksel olarak bağlıdır. Bu durum açık anahtarı bilen saldırganların her zaman özel anahtarı bulma ihtimalini gündeme getirir. Bu saldırı biçimine karşı iki anahtar arasındaki matematiksel bağ olabildiğince çözülmesi zor şekilde tasarlanmalıdır.

2.1.2.1. RSA algoritması

RSA algoritması Ron Rivest, Adi Shamir ve Leonard Adleman tarafından 1977 yılında geliştirilmiş bir asimetrik anahtarlı şifreleme algoritmasıdır ve anahtar oluşturma fonksiyonunda büyük asal sayıları kullanır. Algoritmanın güvenlik düzeyinin arttırmak için anahtar oluşturma aşamasında büyük asal kullanılmıştır (Yerlikaya et al., 2011).

Anahtar oluşturma algoritması şu şekildedir (Yerlikaya et al., 2011):

- P ve Q olarak iki asal sayı seçilir. Bu asal sayılar çok büyük olmalıdır.
- Bu iki asal sayının çarpımı 2.1'deki gibi ve bir eksiklerinin çarpımı 2.2'deki gibi hesaplanır.
- 2.3'deki gibi bir E sayısı seçilir ve seçilen E sayısı $\varphi(N)$ ile aralarında asal bir tamsayı olmalıdır.
- E tamsayısının $\text{mod } \varphi(N)$ 'de tersi alınır, D tamsayısı elde edilir.
- E ve N tamsayıları açık anahtarı, D ve N tamsayıları ise özel anahtarı oluşturur.

$$N = P \times Q \quad (2.1)$$

$$\varphi(N) = (P - 1)(Q - 1) \quad (2.2)$$

$$1 < E < \varphi(N) \quad (2.3)$$

Yukarıdaki işlem basamakları kuralına göre uygulandıktan sonra artık açık metin, açık anahtar olan E ve N tamsayıları ile şifrelenirse, sadece özel anahtar olan D ve N anahtarları ile açılabilir. Aynı şekilde, açık metin özel anahtar olan D ve N anahtarları ile şifrelenirse açık anahtar olan E ve N anahtarları ile açılabilir.

Açık metnin sayısal karşılığı (a) ve şifreli metnin sayısal karşılığı (c) ise, açık anahtar ile şifreleme işlemi 2.4'deki eşitlik ile yapılır. Şifreli metnin sayısal karşılığından gizli anahtar ile açık metin 2.5' eşitlik ile deki elde edilir.

$$a^E = c \text{ mod}(N) \quad (2.4)$$

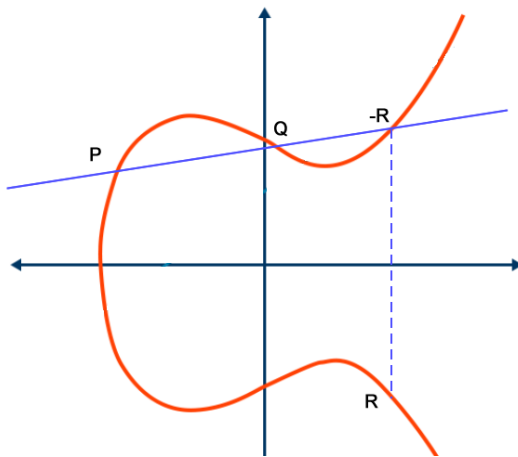
$$c^D = a \text{ mod}(N) \quad (2.5)$$

Bu algoritmada şifreleme işlemi asal olmayan tamsayılar kullanılarak da gerçekleştirilebilir. Asal sayılar kullanılarak saldırıların kullanabilecekleri matematiksel işlemler daha zorlu hale getirilip algoritmanın güvenliği artırılmıştır.

2.1.2.2. Eliptik eğri şifreleme algoritması

1985 yılında Neal Koblitz ve Victor S. Miller, kriptografide eliptik eğrilerin kullanımı ilk defa önermişlerdir (Pollard, 1978). Eliptik eğri şifreleme algoritması da RSA algoritması gibi asimetrik şifreleme algoritmasıdır. Eliptik eğrilerin matematiksel fonksiyonlarını kullanarak anahtar üretir. Eliptik eğrileri kullandığı için RSA algoritmasına göre daha fazla matematiksel fonksiyon içerir (Yerlikaya et al., 2005). Eliptik eğri şifreleme algoritmasının en önemli özelliği RSA algoritmasından daha düşük anahtar boyutlarıyla RSA algoritması kadar kriptografik güç sağlamasıdır. Genellikle dijital imza uygulamalarında kullanır.

Eliptik eğriler $y^2 = x^3 + ax + b$ gibi kübik denklemler ile elde edilir. Şekil 2.8'de eliptik eğri örneği görülmektedir. Şekil 2.8' de görülen P ve Q noktalarının toplamı R noktasını vermektedir ve R noktası eğri üzerinde yer almaktadır. Yani eliptik eğri üzerinde bulunan iki noktanın toplamı yine eğri üzerinde bulunan başka bir noktadır. Bu durum göz önünde bulundurularak toplamı alınan iki noktanın da aynı nokta olması durumunda noktanın kendisi ile toplanması sonucu noktanın iki katı alınmış olur ve elde edilen nokta da yine eğri üzerindedir. Yeni şekil 2.8' de görülen P noktasının iki ile çarpımı olan $2P$ noktası ve $2P$ noktasının iki ile çarpımı olan $4P$ noktası da yine eğri üzerindedir.



Şekil 2.8. Eliptik eğri örneği

Eğer eğri üzerinde bulunan $P(x, y)$ noktası sürekli olarak artarda kendine eklenirse $P + P + P + \dots + P = nP = Q$ şeklinde bir denklem elde edilir. Bu denklemde eğer P ve Q değerleri biliniyorsa n değerini bulmak hiç kolay olmayacaktır. Eliptik eğri şifreleme algoritması bu denklemin çözümünün zorluğu üzerine geliştirilmiştir ve güvenliği bu denklemin çözümünün zorluğu ile doğru orantılıdır.

Kripto paraların kullandığı eliptik eğri fonksiyonu $y^2 = x^3 + ax + b$ fonksiyonundaki a değişkeni 0 ve b değişkeni 7 alınarak elde edilen $y^2 = x^3 + 7$ fonksiyonudur ve SECP256k1 olarak adlandırılır. Eliptik eğri algoritmasında anahtar üretilirken aşağıdaki adımlar izlenir:

- SECP256k1 eğrisi üzerinden bir nokta (P) seçilir.
- $[1, n - 1]$ aralığında bir d tamsayısı seçilir.
- $Q = d \times P$ hesaplaması yapılır.

Bu denklemdeki Q kişinin açık anahtarı d ise kişinin gizli anahtarıdır. Q açık anahtarını ve P noktasını isteyen herkesin öğrenebilmesine rağmen d gizli anahtarını bulmak çok kolay olmayacaktır. Özellikle 256 bitlik değerler kullanıldığı zaman 2^{256} farklı ihtimal vardır ki hesaplaması oldukça uzun zaman alacaktır.

Mesajı şifrelemek isteyen kişi mesajı göndereceği kişinin Q açık anahtarını öğrenir ve aşağıdaki adımları sırası ile uygular:

- $[1, n - 1]$ aralığında bir k tamsayısı seçer.
- $(x_1, y_1) = k \times P$ eşitliğini hesaplar
- $(x_2, y_2) = k \times Q$ eşitliğini hesaplar. $x_2 = 0$ ise k değeri tekrar seçilir.
- Şifreli metin $c = m \times x_2$ denklemi ile elde edilir
- (x_1, y_1, c) değerleri alıcıya gönderilir.

Mesajı alan taraf ise aşağıdaki işlem basamakları ile kendi gizli anahtarını kullanarak şifreli mesajı açar.

- $(x_2, y_2) = d \times (x_1, y_1)$ eşitliği ile (x_2, y_2) noktası bulunur.
- $m = (c \times x_2)^{-1}$ eşitliği ile açık metin elde edilir.

2.1.3. Kriptografik Özet (Hash) Algoritmaları

Kriptografik özet işlevi, iletişim hassas olduğu alanlarda önemli bir yer teşkil eder. Parola gibi önemli bilgilerin depolanmasında ve haberleşme zamanında veri bütünlüğünün sağlanmasında önemli rol oynar ve güvenli iletişim sağlar (El Moumni et al., 2019). Kriptografik özet algoritmaları değişken uzunluktaki girdiyi sabit uzunlukta bir çıktıya dönüştüren yani girdinin bir nebi özetini veya başka bir deyişle parmak izini çıkaran algoritmalar ve aynı girdiye her zaman aynı çıktıyı üretirler. Özet algoritmalarının geri dönüşü yoktur. Yani açık metinden şifreli metin elde edilir fakat şifreli metinden açık metin elde edilemez.

Şifreleme biliminde kriptografik özet algoritmaları ilk olarak 1970'lerin sonlarında geliştirilmeye başlanmıştır; 1980'lerde ise daha fazla öneri sunulmuştur. 1990'larda, özet algoritmalarının sayısı oldukça artmış fakat önerilen algoritmaların birçoğunda güvenlik açıkları tespit edilmiştir (Preneel, 2010). Özet algoritmalarının en büyük güvenlik sıkıntısı çakışma problemidir. Özet algoritmalarının her farklı girdiye farklı çıktı üretmesi beklenir fakat algoritmalar istenmeyen bir şekilde birden fazla farklı girdiye aynı çıktıyı üretebilir. Bu ihtimal ne kadar düşükse algoritma o kadar güvenlidir. Dolayısıyla, özet fonksiyonları birebir fonksiyonlar olmasalar da, bir verinin özeti sadece o veriyle ilişkilendirilebilen bir tür parmak izi gibi davranmalıdır. Bu özellik “zayıf çakışmaya dayanıklılık” (weak collision resistance) olarak adlandırılır. Algoritmaları Zayıf çakışmaya karşı dayanıklı tasarlamak oldukça önemli bir dinamik bir o kadar da zor bir problemidir. Yapılan kriptanaliz çalışmaları zayıf çakışmaya karşı dayanıklı olduğu iddia edilen bir çok algoritmanın aslında böyle olmadığını yıllar içerisinde ortaya çıkarmıştır (Manap & Apohan, n.d.).

Kriptografik özet işlevi, dijital imza hesaplama, kimlik doğrulama, veri bütünlüğü denetimi, rastgele sayı oluşturma ve parola saklama gibi birçok uygulamanın sorumluluğunu alan ana protokoldür (El Moumni et al., 2019). Bir veritabanında saklanan parolalar açık bir şekilde saklandığı zaman veritabanına girme yetkisi olan kişiler kayıtlı herkesin parola bilgilerini öğrenebilir. Bu duruma karşı parolalar kriptografik özet işleminden geçirildikten sonra saklanırsa kötü niyetli kişiler veritabanına ulaşım sağladıkları zaman parolalar yerine parolaların özetlerini göreceklendir ve kriptografik özet algoritmalarının geri dönüşü olmadığı yani şifreli metinden açık metin elde edilemediği için parolaları öğrenemeyeceklerdir. Ya da iki kurum veya kişi arasında yapılan bir protokolü taraflardan birinin değiştirme ihtimaline karşı protokol metninin özeti alınarak bütün taraflara verilir ve bir değişiklik olması durumuna karşı

özet saklanır. Taraflardan biri protokolde bir nokta dahi deęiştirse bile kriptografik özet algoritmasının sonucu çok farklı olacağı için bu tarz usulsüz işlemlerin önüne geçilmiş olunur.

Kriptografik özet algoritmalarının bir dięer kullanım alanı son zamanlarda oldukça popüler olan blok zinciri uygulamalarıdır. Blok zincirinde bir önceki bloğun özeti bir sonraki blokta saklanarak bloklar birbirine bağlanır ve blokların deęiştirilmesinin önüne geçilmeye çalışılır. Blok zinciri tabanını kullanan kripto paralar da belirli bir özet çıktısını bulmak için çalışan ve madenci (miner) olarak adlandırılan bilgisayarların birbirine bağlandığı bir sistemde saklanır. Blok zinciri ve kripto paralar hakkında sonraki başlıklar altında daha detaylı bilgi verilecektir.

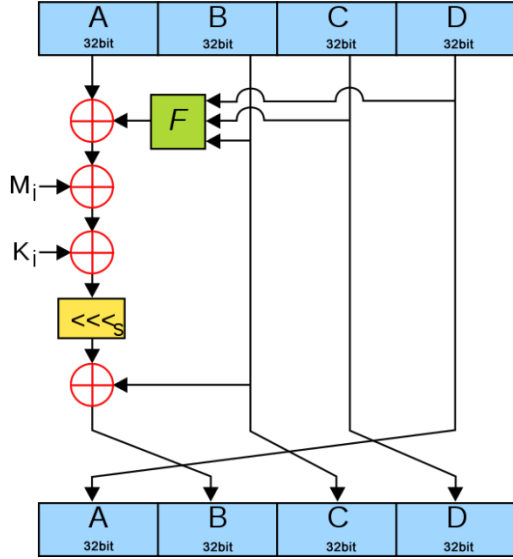
2.1.3.1. MD5 kriptografik özet algoritması

MD5 Kriptografik Özet Algoritması 1992 senesinde RSA algoritmasının yaratıcılarından biri olan Ron Rivest tarafından önerilmiştir. MD5 algoritması, gelişigüzel, sonlu bit uzunluklu bir giriş mesajından 128 bitlik mesaj özeti üretir. Algoritma, 32 bitlik kelimelere bölünmüş 512 bitlik blokları işler. 64 turdan oluşur. Biraz daha hızlı olan ve yalnızca 48 turdan oluşan, ancak aynı zamanda daha az güvenli olduğu kanıtlanmış olan MD4 algoritmasının bir uzantısıdır (Pamula & Ziebinski, 2009).

Dolgu alanı bütün kriptografik özet algoritmalarının ortak problemidir ve algoritma tasarılırken algoritmanın yapısına göre bu probleme farklı çözümler bulunmuştur. MD5 açık metni önce 512 bitlik bloklara böler ve bu blokları sonra 128 ve 32 bitlik ayrı bloklara böler. Açık metnin işlenebilmesi için metnin sonuna 2.6’ da gösterildiği kadar dolgu eklenir. Burada M mesajın bit uzunluğunu, P ise dolgu alanının bit uzunluğunu ifade eder. Mesaj uzunluğu bilindiğine göre dolgu alanı uzunluğu rahatlıkla hesaplanabilir. Dolgu alanının ilk biti “1” olarak, geriye kalan bütün bitler “0” olarak eklenir.

$$M + P = 448 \text{ mod } 512 \quad (2.6)$$

Şekil 2.8’ de MD5 algoritmasının 64 defa tekrarlanan bölümü görülmektedir. Şekil 2.8’ de görülen A, B, C ve D blokları başlangıçta Çizelge 2.2’ de verilen deęerleri alırlar. Her turun sonunda B bloęu C bloęuna, C bloęu D bloęuna ve D bloęu A bloęuna yerleřtirilmektedir. A bloęu ise bir takım işlemlerden sonra B bloęuna yerleřtirilmektedir.



Şekil 2.9. MD5 algoritmasının 64 defa tekrarlanan bölümü (Şeker, 2008)

Çizelge 2.2. MD5 algoritmasının başlangıç değerleri

<i>A</i>	01 23 45 67
<i>B</i>	89 <i>ab cd ef</i>
<i>C</i>	<i>fe dc ba</i> 98
<i>D</i>	76 54 32 10

A bloğu ilk olarak B, C ve D bloklarının girdi olarak gönderildiği bir fonksiyonun sonucu ile özel veya işleme tabi tutulur. Şekil 2.9' da *F* olarak ifade edilen bu fonksiyon ilk 16 turda 2.7'de verilen, 16 ve 32 arası turlarda 2.8' de verilen, 32 ve 48 arası turlarda 2.9'da verilen ve 48 ve 64 arası turlarda 2.10' da verilen denklemlere göre çıktı üretmektedir.

$$F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z), g := i \quad (2.7)$$

$$G(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \neg Z), g := (5 \times i + 1) \bmod 16 \quad (2.8)$$

$$H(X, Y, Z) = X \wedge Y \wedge Z, g := (3 \times i + 5) \bmod 16 \quad (2.9)$$

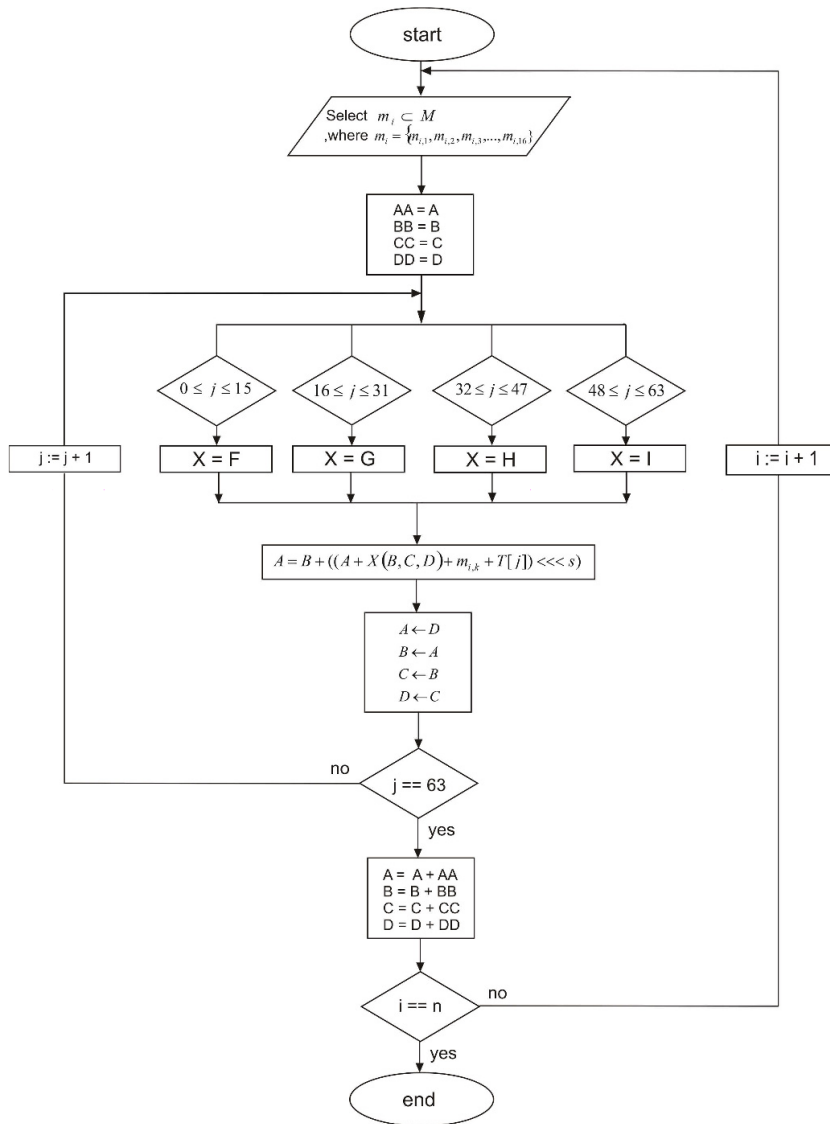
$$I(X, Y, Z) = Y \wedge (X \vee \neg Z), g := (7 \times i) \bmod 16 \quad (2.10)$$

Fonksiyonun çıktısı ve A bloğunun işlendiği özel veya işleminin sonucu ile mesajın ilk 32 bitlik bölümü tekrar özel veya işleme tabi tutulur. Bu işlemin sonucu ise Şekil 2.8' de K_i

olarak gösterilen girdi ile tekrar özel veya işleme sokulur. K_i girdisi ise 2.11' de verilen denklem ile hesaplanır. K_i değerini her turda hesaplamak zaman ve performans kaybına neden olacağı için algoritmanın başında hesaplanarak saklanır ve her seferinde saklandığı yerden çağrılarak kullanılır.

$$K_i = \text{abs}(\sin(i + 1)) \times 2^{32} \quad (2.11)$$

Son özel veya işleminin çıktısı bir defa dairesel sola kaydırıldıktan sonra B bloğu ile tekrar özel veya işleme tabi tutularak sonuç B bloğuna yazılır.



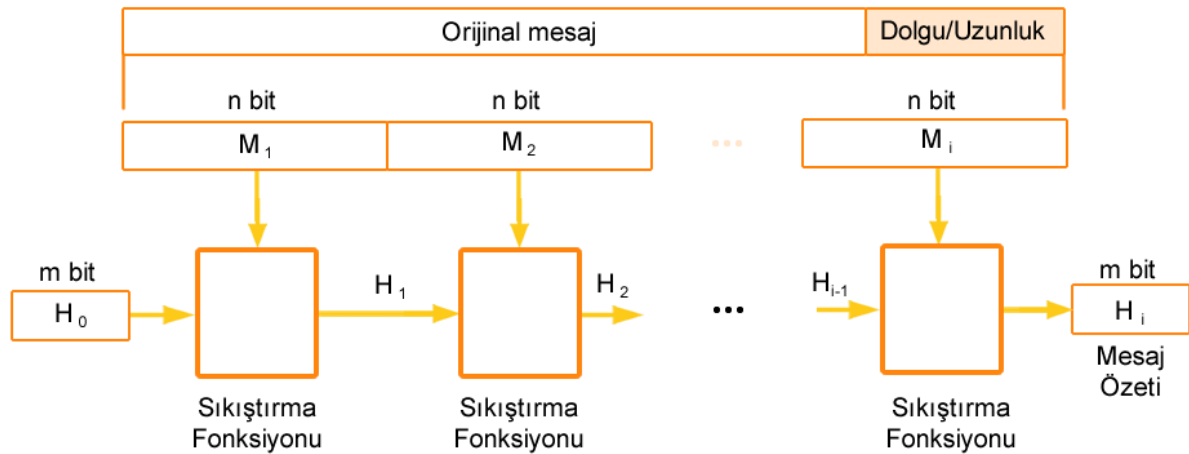
Şekil 2.10. MD5 Algoritmasının genel mimarisi (Pamula & Ziebinski, 2009)

Böylelikle MD5 algoritmasının ilk turu tamamlanmış olur ve elimizdeki yeni A, B, C ve D değerleri ile yeni tura başlanabilir. 16 turun sonunda mesajın 512 bitlik ilk bloğu şifrelenmiş

olur ve F fonksiyonu değiştirilerek ikinci 16 turluk döngü başlar. Böylelikle mesajın her 512 bitlik bloğu için ayrı bir fonksiyon kullanılmış olur. Her turda 32 bitlik 4 blok üzerinde işlem yapıldığı için mesaj uzunluğu ne olursa olsun şifreli metin 128 bit olacaktır. Şekil 2.10’ da MD5 algoritmasının akış şeması görülmektedir.

2.1.3.2. SHA 256 kriptografik özet algoritması

SHA 256 Kriptografik Özet Algoritması, ABD Ulusal Güvenlik Ajansı (NSA) tarafından tasarlanmış kriptografik özet (hash) fonksiyonları kümesi olan SHA2 ailesinin 256 bit özet değerine sahip olan üyesidir (Penard & Werkhoven, 2008). Diğer özet algoritmalarında olduğu gibi SHA 256 algoritması da değişken uzunluktaki girdiye sabit uzunlukta, 256 bit, çıktı üretir ve her farklı girdi için farklı çıktı üretmesi beklenir. SHA 256 algoritması Merkle-Damgard düzenini temel alır. Çizelge 2.’ de Merkle-Damgard düzeninin yapısı görülmektedir.



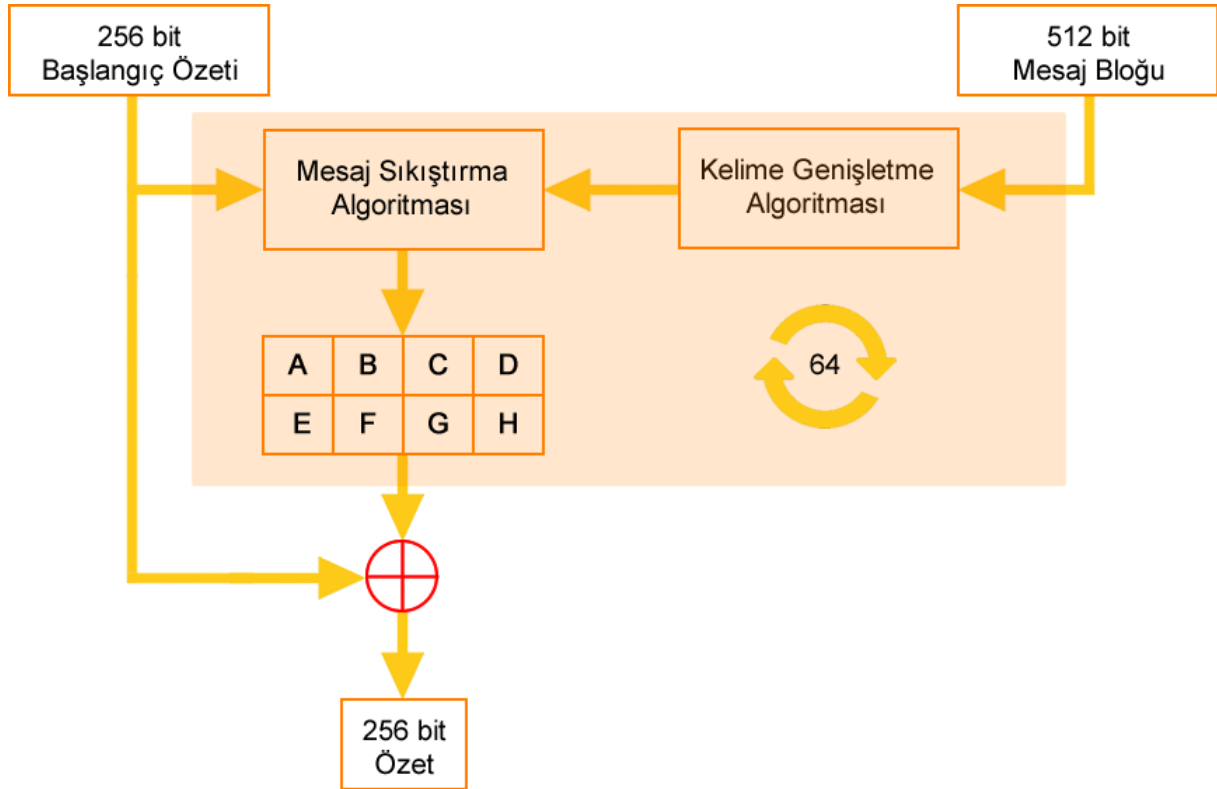
Şekil 2.11. Merkle – Damgard Düzeni

Merkle – Damgard düzeninde mesaj, algoritmanın yapısına göre belirli bit uzunluğunda bloklara bölünür ve son bloğun istenilen uzunlukta olması için eksik kalan bit sayısı kadar dolgu yapılır. SHA 256 algoritmasında blok uzunluğu 512 bittir ve dolgu alanı 2.12’ de verilen eşitlik ile hesaplanır. Bu eşitlikte “M” mesaj artan blok uzunluğunu, “P” ise dolgu alanının uzunluğunu ifade eder. SHA 256 algoritmasında mesajın sonuna 64 bit uzunluğunda, mesajın orijinal uzunluğunu belirten bir alan eklenir. Mesaj uzunluğunu belirten bu alan ile mesajın arasına 2.12’de hesaplanan dolgu miktarı ilk biti 1 gerisi 0 olacak şekilde eklenir. İlk blok bir başlangıç değeri ile algoritmanın yapısına göre bir sıkıştırma fonksiyonunu (Compression function) besler ve bu fonksiyonun çıktısı ikinci blok ile aynı sıkıştırma fonksiyonunu besler. Bu şekilde

bütün bloklar sırası ile sıkıştırma fonksiyonuna girdi olarak gönderilir ve son blok ve son dan bir önceki fonksiyonun çıktısının beslediği sıkıştırma fonksiyonun çıktısı istenilen özet değerini verir. SHA 256 algoritmasında başlangıç değeri 256 bit uzunluğundadır ve nasıl belirlendiği sonraki başlıklar altında detaylı olarak açıklanacaktır.

$$M + P = 448 \text{ mod } 512 \quad (2.12)$$

Şekil 2.11’ de Compression function olarak gösterilen sıkıştırma fonksiyonunun içeriği Merkle-Damgard düzenini kullanan algoritmalarda, algoritmanın yapısına göre farklılık göstermektedir. Şekil 2.12’ de SHA 256 algoritmasının kullandığı sıkıştırma fonksiyonunun genel yapısı görülmektedir ve bu yapı Bitcoin sisteminin omurgasını oluşturmaktadır.



Şekil 2.12. SHA 256 algoritmasının genel yapısı

Şekil 2.12’ de de görüleceği üzere mesajın 512 bit uzunluğundaki bir bloğu ilk olarak Kelime Genişletme Algoritmasına gönderilir ve bu işlemin çıktısı 256 bit uzunluğundaki başlangıç değeri ile Mesaj Sıkıştırma Algoritması besler. Kelime Genişletme Algoritması ve Mesaj Sıkıştırma Algoritması aşamaları 64 tur tekrar edilir ve her tur sonunda A, B, C, D, E, F, G ve H değişkenlerinin içerikleri tekrar tekrar hesaplanır. Bu değişkenlerin uzunlukları 32 bittir

ve 64 tur sonunda başlangıç değeri ile özel veya işlemine tabi tutularak ilk bloğun 256 bit uzunluğunda mesaj özetini oluştururlar. SHA 256 algoritması Merkle-Damgard düzenini temel aldığı için ilk bloktan elde edilen özet bir sonraki blok için başlangıç değeri niteliğinde sıkıştırma fonksiyonunu besler. Mesajdaki blok sayısı kadar bu işlem tekrar edildikten sonra son sıkıştırma algoritmasının çıktısı mesajın SHA 256 özetini verir.

SHA 256 Başlangıç Değerlerinin Ayarlanması

SHA 256 algoritmasının başlangıç özetinin belirlenmesi için asal sayılar kullanılır. İlk 8 asal sayının (2, 3, 5, 7, 11, 13, 17 ve 19) karekökü alınarak, karekök değerlerinin kesirli kısımlarının ilk 32 biti alınarak belirlenir. Örneğin 8. Asal sayı 19 dur ve karekökü $\sqrt{19} = 4,35889894354$. Bulunan sayının kesirli kısmının sadece 32 bitlik bölümü binary olarak dönüştürüldüğünde 2.13' deki gibi H_7 sabiti bulunur.

$$(100,0101\ 1011\ 1110\ 0000\ 1100\ 1101\ 0001\ 1001)_2 = (4,5BE0CD19)_{16} \quad (2.13)$$

Çizelge 2.3. SHA 256 Algoritmasının başlangıç değerleri

H_0	H_1	H_2	H_3	H_4	H_5	H_6	H_7
6a09e667	bb67ae85	3c6ef672	a54ff53a	510e527f	9b05688c	1f83d9ab	5be0cd19

Çizelge 2.3' de SHA 256 algoritmasının 256 bit başlangıç değerlerinin hexadecimal karşılıkları görülmektedir. Bu hexadecimal sayılar binary sayı sistemi çevrildiği zaman Şekil 2.12' de görülen 256 bit Başlangıç Özetini oluştururlar.

SHA 256 Kelime Genişletme Algoritması

Kelime genişletme algoritması, girdi olarak aldığı 512 bit uzunluğundaki mesaj bloğunu 32 bitten oluşan 64 kelimeye genişletir. $W_0, W_1 \dots W_{63}$ olarak etiketlenmiş olan bu kelimeler Mesaj Sıkıştırma Algoritması için her turda girdi olarak kullanılır. Kelime Genişletme Algoritması kelime değerlerini 2.14 ve 2.15' deki gibi hesaplar.

$$0 \leq t \leq 15 \Rightarrow W_t = M_t \quad (2.14)$$

$$16 \leq t \leq 63 \Rightarrow \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-5}) + W_{t-16} \quad (2.15)$$

2.14 ve 2.15' deki denklemlerden de anlaşıldığı üzere Kelime Genişletme Algoritması ilk 16 turda, 512 bit uzunluğundaki mesaj bloğunu 32 bitlik kelimeler halinde mesaj sıkıştırma

algoritmasına girdi olarak gönderir. Sonraki 48 turda ise 2.15' deki denklemi kullanarak mesajı genişletir. 2.15' deki denklemde yer alan σ_0 ve σ_1 işlemleri SHA 256' nın Kelime Genişletme Algoritmasına özgü iki mantıksal işlemdir. 2.16' da σ_0 mantıksal işlemi 2.17' de σ_1 mantıksal işlemi verilmiştir.

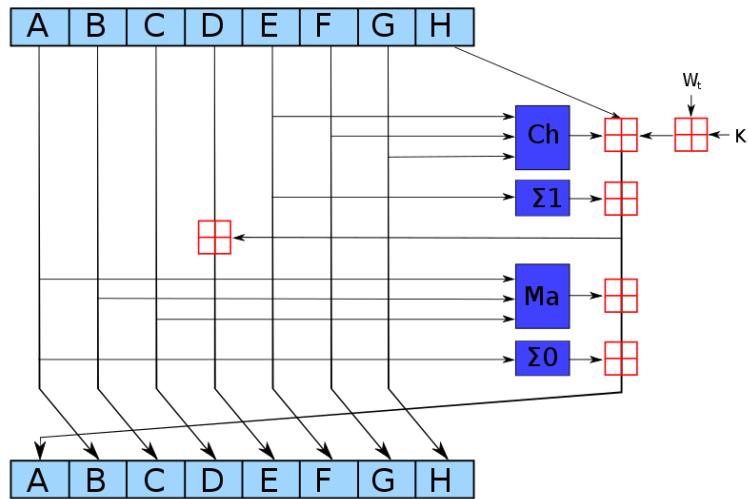
$$\sigma_0 = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x) \quad (2.16)$$

$$\sigma_1 = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \quad (2.17)$$

2.16' da ve 2.17' de görülen $ROTR^y(x)$ işlemi mantıksal olarak dairesel sağa kaydırma ve $SHR^y(x)$ işlemi mantıksal olarak doğrusal sağa kaydırma işlemidir ve her iki işlemde y kadar tekrar edilir. Örneğin $ROTR^2(11010010)$ işleminin sonucu (10110100) , $SHR^2(11010010)$ işleminin sonucu ise (00110100) olacaktır. İki denklemde de yer alan \oplus işlemi ise mantıksal özel veya (XOR) işlemidir.

SHA 256 Mesaj Sıkıştırma Algoritması

Mesaj Sıkıştırma Algoritması SHA 256' dan beklenen işlemi gerçekleştiren kısımdır. Başlangıç vektörü, Kelime Genişletme Algoritması ve bazı asal sayıların küp köklerinden aldığı değerleri karıştırır ve 64 tur sonunda başlangıç vektörünü bir kez daha kullanarak 256 bit özeti oluşturur.



Şekil 2.13. SHA 256 Mesaj Sıkıştırma Algoritmasının yapısı (SHA-2, 2020)

Şekil 2.13' de gösterilen Mesaj Sıkıştırma Algoritması' nın yapısında yer alan A, B, C, D, E, F, G ve H değerleri ilk mesaj bloğu için başlangıç değerlerinden, sonraki bloklar için bir

önceki bloğun SHA 256 özetinden gelir. W_t değeri Kelime Genişletme Algoritması' nın 512bit mesaj bloğunu kullanarak ürettiği 32 bit uzunluğunda 64 adet kelimedir ve algoritmanın her turunda bu kelimeler sırası ile kullanılır. K_t değeri ise ilk 64 asal sayının küp köklerinin binary karşılıklarının ondalık kısımları ile üretilir. Ondalık kısımlarının ilk 32 biti alınarak elde edilir.

Çizelge 2.4. İlk 64 asal sayının küp köklerinin ikilik (binary) sayı sistemindeki karşılıklarının ondalık kısımlarının ilk 32 bitinden elde edilen değerlerin onaltılık (hexadecimal) sayı sistemindeki karşılıkları.

t	K_t	t	K_t	t	K_t	t	K_t
0	428a2f98	16	e49b69c1	32	27b70a85	48	19a4c116
1	71374491	17	efbe4786	33	2e1b2138	49	1e376c08
2	b5c0fbcf	18	0fc19dc6	34	4d2c6dfc	50	2748774c
3	e9b5dba5	19	240ca1cc	35	53380d13	51	34b0bcb5
4	3956c25b	20	2de92c6f	36	650a7354	52	391c0cb3
5	3956c25b	21	4a7484aa	37	766a0abb	53	4ed8aa4a
6	923f82a4	22	5cb0a9dc	38	81c2c92e	54	5b9cca4f
7	ab1c5ed5	23	76f988da	39	92722c85	55	5b9cca4f
8	d807aa98	24	983e5152	40	a2bfe8a1	56	748f82ee
9	12835b01	25	a831c66d	41	a81a664b	57	78a5636f
10	243185be	26	b00327c8	42	c24b8b70	58	84c87814
11	550c7dc3	27	bf597fc7	43	c76c51a3	59	8cc70208
12	72be5d74	28	c6e00bf3	44	d192e819	60	90befffa
13	80deb1fe	29	d5a79147	45	d6990624	61	a4506ceb
14	9bdc06a7	30	06ca6351	46	f40e3585	62	bef9a3f7
15	c19bf174	31	14292967	47	106aa070	63	c67178f2

Çizelge 2.4' de ilk 64 asal sayının küp köklerinin ikilik (binary) sayı sistemindeki karşılıklarının ondalık kısımlarının ilk 32 bitinden elde edilen değerlerin onaltılık

(hexadecimal) sayı sistemindeki karşılıkları görülmektedir. Bu değerler Mesaj Sıkıştırma Algoritmasının her turunda sırası ile kullanılır.

Şekil 2.12' de görülen Ch fonksiyonu 2.18'de, Ma fonksiyonu 2.19' da, $\Sigma_0(A)$ fonksiyonu 2.20' de ve $\Sigma_1(E)$ fonksiyonu 2.21' de verilmiştir.

$$Ch(E, F, G) = (E \wedge F) \oplus (\neg E \wedge G) \quad (2.18)$$

$$Ma(A, B, C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C) \quad (2.19)$$

$$\Sigma_0(A) = ROTR^2(A) \oplus ROTR^{13}(A) \oplus ROTR^{22}(A) \quad (2.20)$$

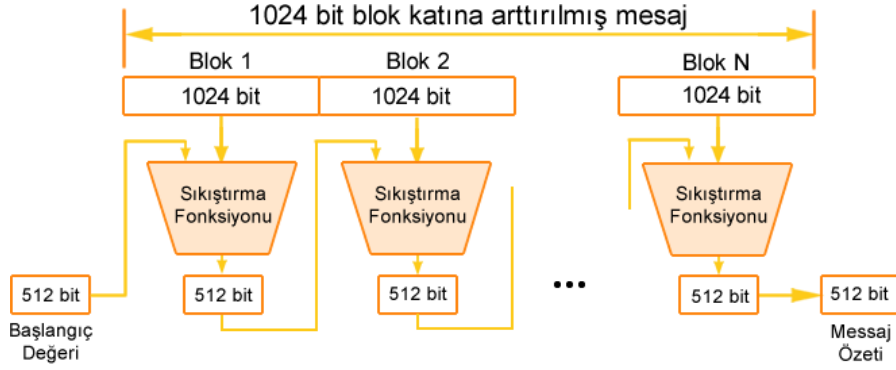
$$\Sigma_1(E) = ROTR^6(E) \oplus ROTR^{11}(E) \oplus ROTR^{25}(E) \quad (2.21)$$

2.20' da ve 2.21'de görülen $ROTR^y(x)$ işlemi mantıksal olarak dairesel sağa kaydırma işlemidir y kadar tekrar edilir. Örneğin $ROTR^2(11010010)$ işleminin sonucu (10110100) olacaktır. Denklemden yer alan \oplus işlemi ise mantıksal özel veya (XOR) işlemidir.

Mesaj Sıkıştırma Algoritması, Kelime Genişletme Algoritması ile senkron olarak 64 tur tekrar edildikten sonra elde edilen A, B, C, D, E, F, G ve H değerleri 256 bit başlangıç özeti ile özel veya işlemine tabi tutularak 512 bitlik ilk mesaj bloğunun özeti alınmış olur ve bu özet bir sonraki 512 bit blok için kullanılmak üzere sonraki sıkıştırma fonksiyonuna gönderilir. Mesajın bütün blokları için algoritmanın tüm adımları düzenli olarak uygulandığı takdirde son olarak elde edilen A, B, C, D, E, F, G ve H değerleri mesajımızın SHA 256 özet çıktısı olacaktır.

2.1.3.3. SHA 512 kriptografik özet algoritması

SHA ailesinin 512 bitlik özet algoritması olan SHA512 genel yapı olarak diğer SHA ailesi algoritmalarına, özellikle de SHA 256 özet algoritmasına oldukça benzer. SHA 512 özet algoritması SHA 256 gibi Merkle-Damgard düzenini temel alır ve değişken uzunluktaki mesaj girdilerine karşı 512 bit uzunluğunda özet oluşturur. Şekil 2.14' de SHA 512 algoritmasının genel yapısı görülmektedir. Şekil incelendiğinde SHA 256 algoritmasında olduğu gibi mesaj bloklara bölünmüştür ve ilk blok başlangıç değeri ile birlikte bir sıkıştırma fonksiyonunu beslemektedir. Sonraki bloklar ise bir önceki sıkıştırma fonksiyonunun çıktısı ile birlikte sıkıştırma fonksiyonunu beslemektedir. Mesaj bloklara bölünürken mesajın sonunda kalan bloğu uygun uzunluğa ayarlamak için dolgu alanı kullanılır.



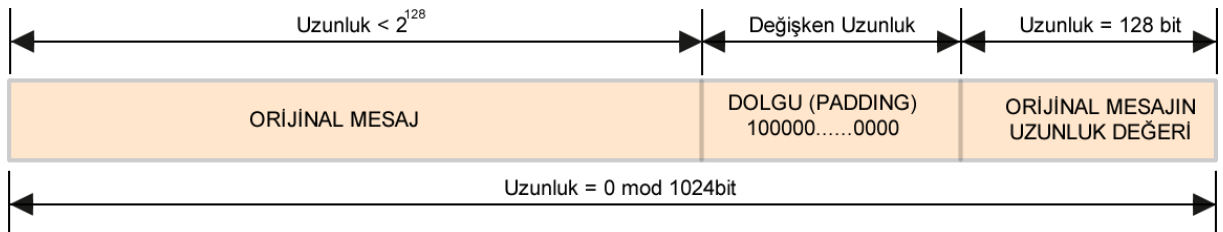
Şekil 2.14. SHA-512 Mesaj Özeti

SHA 512, orijinal mesaj boyutunun 2^{128} bitten az olması gerekir. Bu durumda mesaj boyutu 2^{128} bite eşit veya daha büyük olursa SHA 512 çalışmayacaktır (B. A. Forouzan, 2007). Bu durum başlangıçta kısıtlayıcı bir durum gibi görünse de 2^{128} bitin hesaplandığı zaman normal bir depolama aygıtının depolayamayacağı kadar büyük bir veri olduğu görülecektir.

SHA 512 Blok Uzunluğu ve Dolgu Alanı Boyutu

Şekil 2.14' de görüldüğü gibi SHA 512 algoritmasının blok uzunluğu 1024 bittir. Mesaj 1024 bit uzunluğunda bloklara bölündükten sonra mesaj sonuna 128 bit uzunluğunda orijinal mesajın boyutunu belirten bir ekleme yapılır ve sonrasında dolgu alanı eklenir. Dolgu alanı uzunluğu 2.22'de verilen denklem ile belirlenir. Denklemde $|M|$ orijinal mesaj uzunluğunu, $|P|$ dolgu alanının uzunluğunu temsil etmektedir. Mesaja bütün eklentiler yapıldıktan sonra mesaj uzunluğu 1024 bit veya katı olmak zorundadır. Şekil 2.15 incelendiğinde dolgu alanının orijinal mesaj ile 128 bit mesaj uzunluğu eklentisinin arasına eklendiği ve ilk bitinin 1 diğer bitlerinin 0 olduğu görülmektedir.

$$(|M| + |P| + 128) = 0 \text{ mod } 1024 \rightarrow |P| = (-|M| - 128) \text{ mod } 1024 \quad (2.22)$$



Şekil 2.15. SHA 512 içindeki dolgu ve alan uzunlukları

SHA 512 Başlangıç Değerinin Ayarlanması

Şekil 2.14 incelendiği zaman mesajın ilk bloğu (Block1) bir başlangıç değeri (Initial Value) ile sıkıştırma fonksiyonunu beslemektedir. Birçok özet algoritmasında bir başlangıç değeri vardır ve bu değer algoritmanın yapısına göre farklı biçimlerde belirlenir. Belirlenen başlangıç değeri algoritmanın kelime uzunluğu ile uyumlu olmalıdır. Örneğin bir önceki başlık altında anlatılan SHA 256 algoritması mesajı bloklara böldükten sonra her bloğu 32 bitlik kelimelere bölerek işlemiştir ve bu duruma bağlı olarak başlangıç değeri 32 bit 8 kelimedenden oluşturularak 256 bit olarak belirlenmiştir. SHA 512 algoritmasının kelime uzunluğu 64 bittir ve başlangıç değeri 8 kelimedenden oluşur yani 512 bit başlangıç değerine sahiptir. Başlangıç değerindeki kelimeler $A_0, B_0, C_0, D_0, E_0, F_0, G_0$ ve H_0 olarak adlandırılır ve her kelime sırası ile ilk 8 asal sayının (2, 3, 5, 7, 11, 13, 17 ve 19) karekök değerlerinin kesirli kısımlarının ilk 64 biti alınarak belirlenir. Örneğin 8. Asal sayı 19 dur ve karekökü $\sqrt{19} = 4,35889894354$. Bulunan rakamın kesirli kısmının sadece 64 bitlik bölümü binary olarak dönüştürüldüğünde 2.23’ deki gibi H_0 sabiti bulunur.

$$(100,0101\ 1011\ 1110\ 0000\ 1100 \dots 1001)_2 = (4,5BE0CD19137E2179)_{16} \quad (2.23)$$

Gerekli hesaplamalar yapıldığı zaman SHA 512 algoritmasının başlangıç değerleri çizelge 2.5’ deki gibi bulunur.

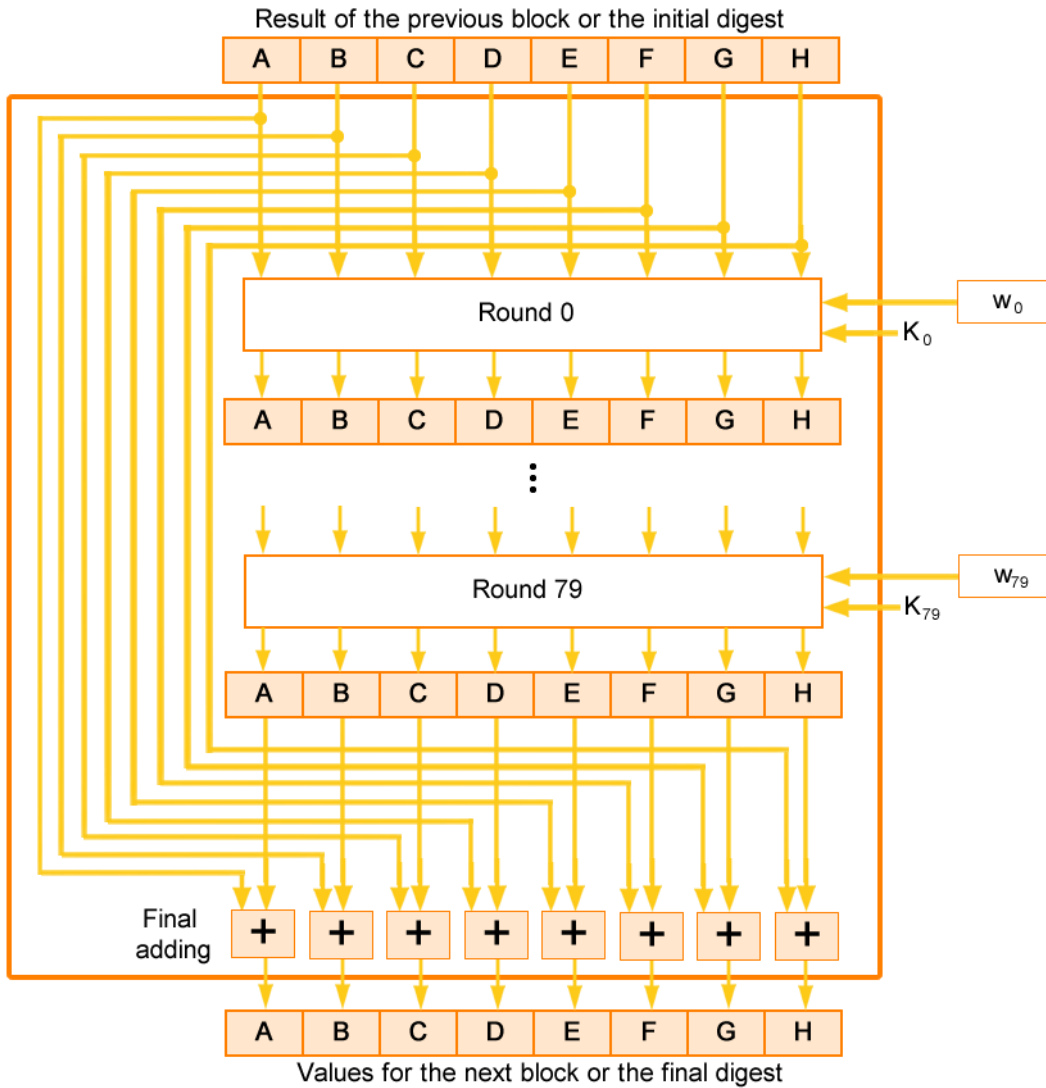
Çizelge 2.5. SHA 512 Algoritmasının başlangıç kelime değerleri

A_0	6A09E667F3BCC908	E_0	510E527FADE682D1
B_0	BB67AE8584CAA73B	F_0	9B05688C2B3E6C1F
C_0	3C6EF372EF94F828	G_0	1F83D9ABFB41BD6B
D_0	A54FE53A5F1D36F1	H_0	5BE0CD19137E2179

SHA 512 Mesaj Sıkıştırma Fonksiyonu

Şekil 2.14’ de görülmekte olan sıkıştırma fonksiyonu (compression function) SHA 512 algoritmasının omurgasını oluşturmaktadır. İlk sıkıştırma fonksiyonu 1024 bit uzunluğundaki mesaj bloğunu ve 512 bit uzunluğundaki başlangıç değerini girdi olarak alır ve 512 bit uzunluğunda ilk mesaj bloğunun özetini çıktı olarak verir. İlk sıkıştırma fonksiyonunun çıktısı, ikinci mesaj bloğu ile birlikte ikinci sıkıştırma fonksiyonunu beslemektedir. Mesajın N adet bloğa bölüldüğü düşünülürse $(N - 1)$. sıkıştırma fonksiyonun çıktısı ve N . blok, N . sıkıştırma

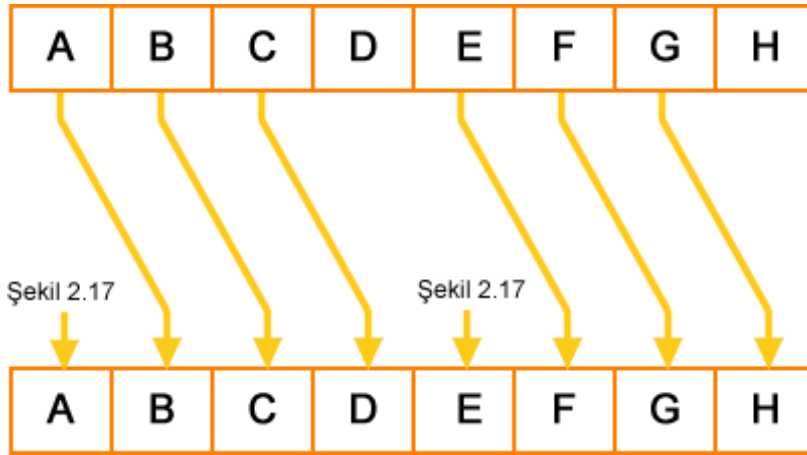
fonksiyonuna girdi olarak gönderilir ve bu fonksiyonun çıktısı mesajın SHA 512 özeti olur. Şekil 2.16’ da SHA 512 sıkıştırma fonksiyonunun yapısı görülmektedir.



Şekil 2.16. SHA 512 Sıkıştırma fonksiyonu (compression function) (B. A. Forouzan, 2007)

SHA 512 sıkıştırma fonksiyonu başlangıç değeri ile başlar ve 80 turdan oluşur. Başlangıç kelime değerleri Çizelge 2.4’ de verilmiştir ve nasıl elde edildikleri aynı başlık altında anlatılmıştır. Her tur (Round) bir önceki turun A, B, C, D, E, F, G ve H olarak ifade edilen ve 64 bit uzunluğunda kelimelerden oluşan toplam 512 bit özetini, $W_0, W_1, W_2 \dots W_{79}$ olarak gösterilen kelime genişletme fonksiyonunun çıktısını ve $K_0, K_1, K_2 \dots K_{79}$ ile gösterilen ve asal sayılardan elde edilen değerleri girdi olarak alır ve yeni A, B, C, D, E, F, G ve H değerlerini üretir. 80.turun sonunda üretilen 8 adet kelime değeri başlangıç kelime değerleri ile toplanarak sıkıştırma fonksiyonun 512 bit uzunluğundaki çıktısı elde edilir ve bir sonraki sıkıştırma

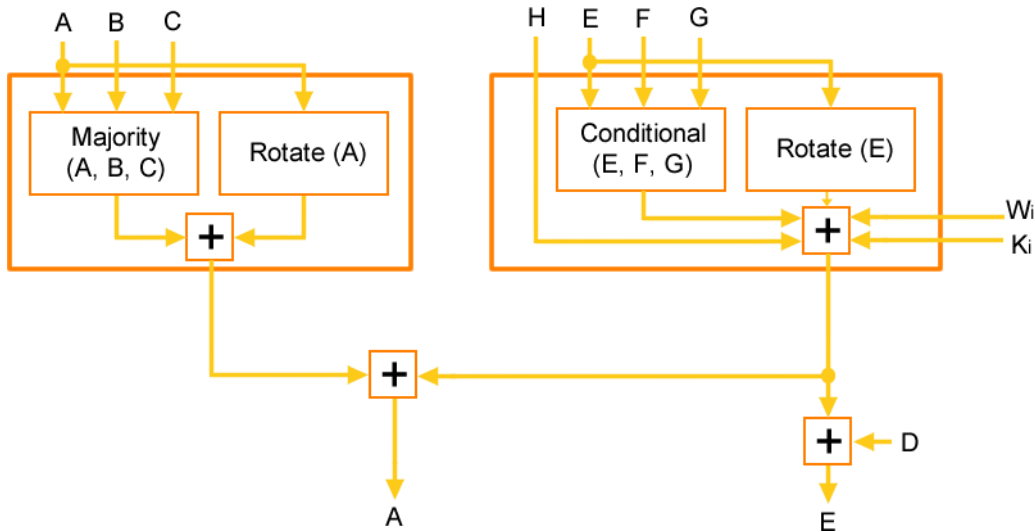
fonksiyonuna girdi olarak gönderilir. Şekil 2.16’ da Round olarak ifade edilen turların iç yapısı Şekil 2.17’ de verilmiştir.



Şekil 2.17. SHA 512 sıkıştırma fonksiyonundaki her turun yapısı

Şekil 2.17’ de görüldüğü üzere her turun girişindeki 8 kelimedenden 6 tanesi 2.24 verildiği gibi ilgili turun çıkışındaki 6 kelimeye direkt olarak aktarılır. Çıkış kelimelerinden *A* ve *E* ise bir takım matematiksel işlem sonucunda belirlenir. Şekil 2.18’ de *A* ve *E* değerlerinin nasıl belirlendiği görülmektedir.

$$A \rightarrow B, B \rightarrow C, C \rightarrow D, E \rightarrow F, F \rightarrow G, G \rightarrow H \quad (2.24)$$



Şekil 2.18. SHA 512 sıkıştırma algoritmasındaki *A* ve *E* kelimelerinin belirlenmesi

Şekil 2.18' de görülen Majority (A, B, C) fonksiyonu 2.25'de, Rotate (A) ve Rotate (E) fonksiyonları 2.26' da ve Conditional (E, F, G) fonksiyonu 2.27' de verilmiştir.

$$\text{Majority}(A, B, C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C) \quad (2.25)$$

$$\text{Rotate}(x) = \text{RotR}_{28}(x) \oplus \text{RotR}_{34}(x) \oplus \text{RotR}_{39}(x) \quad (2.26)$$

$$\text{Conditional}(E, F, G) = (E \wedge F) \oplus (\neg E \wedge \neg G) \quad (2.27)$$

2.26' da verilen $\text{RotR}_i(x)$ işlemleri binary x değerini i bit kadar dairesel sağa kaydırma anlamına gelmektedir. Denklemde x değeri 28, 34 ve 39 bit dairesel olarak sağa kaydırıldıktan sonra elde edilen üç değere özel veya işlemi uygulanır. Şekil 2.18' de yer alan toplama işlemleri bit düzeyinde $\text{mod}2^{64}$ toplama işlemidir. Sıkıştırma fonksiyonunda yer alan $W_0, W_1, W_2 \dots W_{79}$ değerleri kelime genişletme fonksiyonunun çıktısından ve $K_0, K_1, K_2 \dots K_{79}$ değerleri ise ilk 80 asal sayıdan elde edilmektedir.

$K_0, K_1, K_2 \dots K_{79}$ değerleri belirlenirken ilk 80 asal sayı kullanılır. Her değer, karşılık gelen asal sayının küp kökünün ikilik biçime dönüştürüldükten sonra kesirli kısmının yalnızca ilk 64 biti alınarak elde edilir. Örneğin 80. Asal sayı olan 409' a bağlı olarak K_{79} değerinin nasıl hesaplandığı 2.28' de verilmiştir.

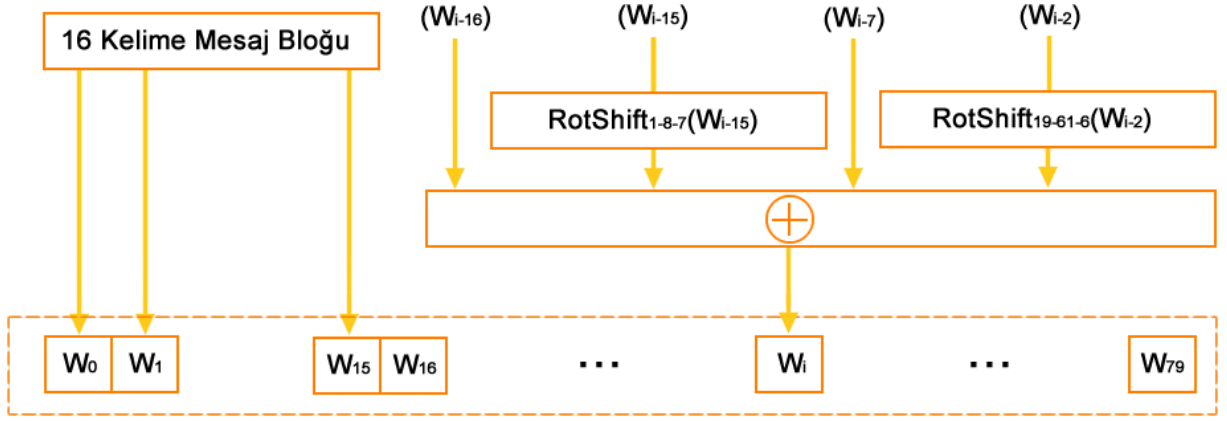
$$\sqrt[3]{409} = 7,42291412044$$

$$(7,42291412044)_{10} = (111,0110 \ 1100 \ 0100 \ 0100 \ \dots \ 0111)_2$$

$$(111,0110 \ 1100 \ 0100 \ 0100 \ \dots \ 0111)_2 = (7,6C44198C4A475817)_{16} \quad (2.28)$$

SHA 512 Kelime Genişletme Fonksiyonu

SHA 512 Algoritmasının 1024 bit uzunluğundaki mesaj bloğu 64 bit uzunluğundaki 16 kelimedenden oluşur. Bu kelimeler sıkıştırma fonksiyonunun her turunda kullanılmak üzere genişletilerek 64 bit uzunluğundaki 80 kelimeye genişletilir. Şekil 2.19' da SHA 512 Kelime genişletme fonksiyonunun yapısı verilmiştir. SHA 512 kelime genişletme fonksiyonunda mesaj bloğunun ilk 16 kelimesi $W_0, W_1, W_2 \dots W_{15}$ kelimelerine aynen aktarılır. 16. kelimedenden sonra ise Şekil 2.19' da gösterildiği önceden oluşturulan kelimeler bir takım mantıksal işlemlerle 64 kelime daha üreterek mesaj sıkıştırma fonksiyonunu besler.



Şekil 2.19. SHA 512 Kelime genişletme fonksiyonunun yapısı

Şekil 2.19' da yer alan *RotShift* fonksiyonu 2.29' da verilmiştir ve fonksiyon içeriğinde yer alan $RotR_i(x)$ işlemleri binary x değerini i bit kadar dairesel olarak sağa kaydırma ve $ShL_i(x)$ işlemi ise x değerini i bit doğrusal sola kaydırma anlamına gelmektedir. Bu fonksiyon sonucunda elde edilen 80 adet kelime sıkıştırma fonksiyonunun her turunda A ve E kelimeleri belirlenirken kullanılırlar.

$$RotShift_{l-m-n}(x) = RotR_l(x) \oplus RotR_m(x) \oplus ShL_n(x) \quad (2.29)$$

2.2. Kriptanaliz

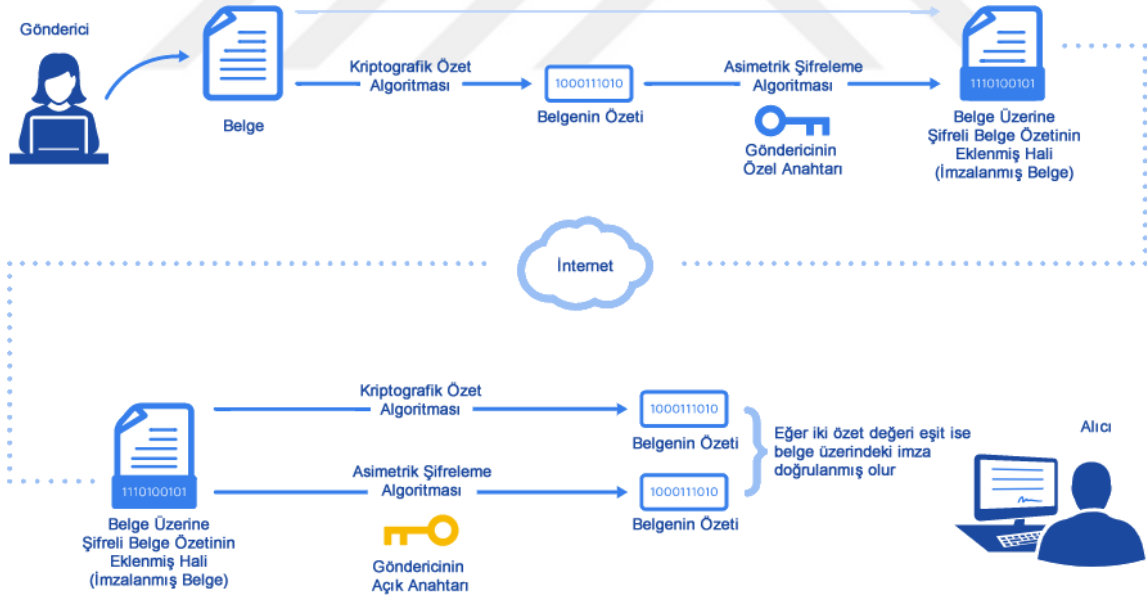
Kriptanaliz, şifreli metinden açık metni elde etme yöntemlerinin tümüdür (Akleylek et al., 2011). Diğer bir deyişle kriptografik algoritmalar kullanılarak şifrelenmiş verilerin şifrelerinin kırılmasında kullanılan tekniklerin çalışması, yeni kısaca şifre kırma bilimidir (Sakallı et al., 2007). Bir açık metin şifrelenirken elbette tekrar açık metin elde edilmesi gerektiği göz önünde bulundurularak şifreleme algoritması ile beraber şifreyi çözme, yani şifreli metinden açık metni elde etme algoritması da geliştirilir. Bu algoritma kullanılarak şifreli metinden açık metin elde edilir. Bu açık metni elde etme işlemi istenmeyen 3. şahıslar tarafından saldırı şeklinde de olabilir. Yani kriptanaliz kripto sistemlerin şifrelerinin kırılmasını amaçlamaktadır (Bhattacharya, 2019).

3. DİJİTAL İMZA

Günlük hayatta kağıt üzerinde oluşturulan bir belge, belgeyi oluşturan kişi tarafından mürekkepli kalemle imzalanır ki buna ıslak imza denir. Türk Dil Kurumu'nda ıslak imza: "kişinin kağıt üzerine kalemle attığı imza" olarak tanımlanmaktadır. Kişilere ait genelde bir adet ıslak imza vardır ve bütün belgelere aynı imza atılır. Bir belgenin ıslak imza ile imzalanmasındaki amaçlar şöyle sıralanabilir.

1. İnkâr Edememe: belgeyi imzalayan kişinin ilgili belgeyi oluşturduğunu inkar edememesi.
2. Bütünlük: oluşturulan belgenin başka kişiler tarafından değiştirilmesinin engellenmesi.
3. Kimlik Denetimi: kişi adına başka kişiler tarafından belge oluşturulmasının önüne geçilmesidir.

Islak imza ile imzalanmış bir belgede yukarıda sıralanan amaçlar ile ilgili bir sorun çıktığı zaman ıslak imzanın yetkili kurum veya kuruluşlar tarafından incelenmesi gerekir.



Şekil 3.1. Elektronik imza çalışma algoritması

Bilgisayarların hayatımızın vazgeçilmez bir unsuru olması ile birlikte imzalanması gereken evraklar da dijitalleşmeye başlamış ve ıslak imzanın bir evrak üzerinde sağladığı inkar edememe, bütünlük ve kimlik denetimi gibi unsurları sağlayacak dijital imza algoritmaları

kullanılmaya başlanmıştır. Bir taraf, bir belge için dijital imza oluşturursa, yalnızca belge ile imzalayan arasında doğrulanan bir ilişki kurar. Bir belgenin dijital imza ile imzalanması belgenin değiştirilmediğini, bütünlüğünü kanıtlar (Pierro, 2017). Dijital imza, ıslak imza ile aynı görevi yerine getirmektedir fakat ıslak imza kullanılırken aynı kişi tarafından bütün belgelere aynı imza atılmasına rağmen dijital imza kullanılırken ise aynı kişi tarafından atılan imza her belgede farklılık göstermektedir. Bu farklılığın sebebi dijital imzanın çalışma algoritması incelendikten sonra daha iyi anlaşılacaktır. Şekil 3.1’ de dijital imzanın çalışması şematik olarak verilmiştir.

Dijital imza uygulamalarında kriptografik özet algoritmaları ve asimetrik şifreleme algoritmaları senkron olarak kullanılır. Kriptografik özet algoritması olarak genelde SHA 256 gibi SHA ailesi algoritmaları, asimetrik şifreleme algoritması olarak da genelde RSA algoritması tercih edilir. Şekil 3.1’ de de görüldüğü gibi belgeyi oluşturan kişi bir kriptografik özet algoritması ile belgenin özetini alır ve bu özeti bir asimetrik şifreleme algoritması ile kendi özel anahtarını kullanarak şifreler. Şifreleme işleminden sonra şifrelenmiş özet değerini belgenin üzerine ekler ve göndermek istediği kişi veya kuruluşa paket halinde gönderir. Her belgenin özet değeri birbirinden farklı olacağı için her belge üzerinde atılan imzada farklı olacaktır. Belgeyi alan kişi belgeyi gönderici ile aynı kriptografik özet algoritmasına girdi olarak göndererek eline ulaşan belgenin özetini alır. Yine belgeyi alan kişi belge üzerinde yer alan ve göndericinin eklemiş olduğu şifreli belge özetini, göndericinin açık anahtarını ve gönderici ile aynı şifreleme algoritmasını kullanarak açar. Eğer alıcının belgeden elde ettiği özet değeri ile şifrelenmiş özet değerini açarak elde ettiği özet değeri birbiri ile aynı ise belge üzerindeki imza doğrulanmış olur. Belge üzerinde imzalayan kişi dışında bir değişiklik yapıldı ise ya da belge üzerine oluşturan kişi dışında başkasının imzası atılmaya çalışıldı ise alıcının belge ile elde ettiği özet değeri, şifreli özet değerinden elde ettiği özetten farklı olacağı için belgenin imzalayan kişi tarafından oluşturulmadığı veya başka kişi veya kişiler tarafından değiştirildiği anlaşılacaktır. Böylelikle doğrulama işlemi gerçekleşmemiş olacaktır.

Islak imza ve elektronik imza karşılaştırıldığında şu farklılık öne çıkar:

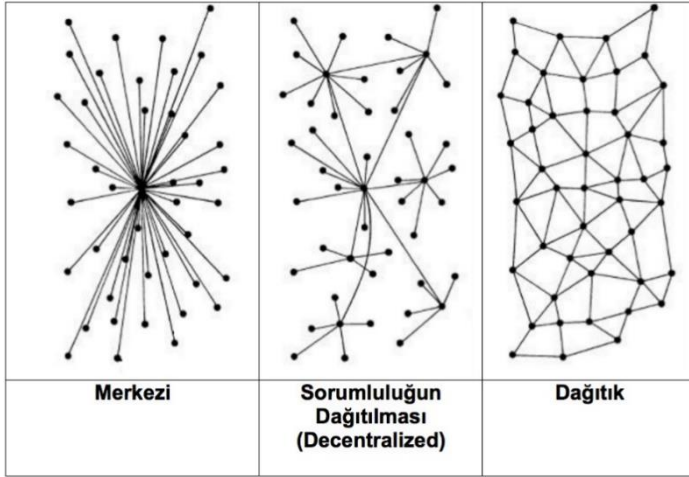
- Islak imza bütün belgelerde aynı iken elektronik imza her belgede farklılık gösterir.
- Islak imzanın doğrulanması için kriminal inceleme gerekirken elektronik imza için doğrulama işlemi çok daha kolaydır.
- Islak imza gözle fark edilemeyecek şekilde taklit edilebilir ama elektronik imzanın özel anahtarının güvenliği sağlandığı takdirde taklit edilmesi imkansızdır.

Elektronik imza sadece resmi belge imzalamak için kullanılmamaktadır. Dijital ortamda yer alan herhangi bir dosya, program, tasarım gibi çalışmaların imzalanmasında kullanılabileceđi gibi tezin konusu olan Bitcoin gibi kripto para birimlerinin işleyişı içinde illegal mükerrer para transferlerini engellemek içinde kullanılır. Dijital imzanın kripto paraların işleyişinde nasıl kullanıldığı konusuna ilerleyen başlıklarda daha detaylı olarak değinilecektir.



4. BLOK ZİNCİRİ (BLOCK CHAIN)

Yazılım ve internet teknolojisinin gelişmesiyle beraber eşten eşe iletişim kurabilmek ve merkezi olmayan sistemler oluşturabilmek mümkün olmuştur ve bu çerçevede belirli bir merkeze bağlı kalmadan eşler arasında iletişim kurabilen ve verileri dağıtık biçimde saklayan yeni sistemler gündeme gelmiştir. Benzer uygulamalar daha denenmiş olsa da blok zinciri mantığı ile bu olay farklı bir boyut kazanmıştır (Aşan & Avunduk, 2018). Blok zinciri, günlük hayatta kullanılan, halka açık muhasebe defteri gibi işlem kayıtlarının bir listesini tutan bir blok dizisidir (Chuen, 2015). Genel bir tanım olarak "Blok Zincirinin özünün, merkezi olmayan ve güvenilir yöntemlerle topluca tutulan güvenilir bir veri tabanının teknik bir planı" olduğunu belirtilmektedir (Tian, 2016). Yani Blok zincir teknolojisi veritabanı teknolojisinin dağıtık olarak uygulanmasıdır (Aşan & Avunduk, 2018).



Şekil 4.1. Veri Tabanı modellerinin şematik gösterimi (Tübitak, n.d.)

Literatürde Blok Zincirine dair değişik tanımlamalar bulunmaktadır. Bu tanımlamaların ortak noktası Blok Zincirinin özünde bir veri tabanı modeli olduğudur. Gelişen teknoloji beraberinde farklı ihtiyaçlar doğurmuş ve bu ihtiyaçlara cevap verebilmek için klasik veri tabanı modellerinin değişmesi ihtiyacı hâsıl olmuştur. Klasik veri tabanı modelinde veriler merkezi bir veritabanında saklanır ve bu veriler ile çalışmak isteyen tüm istemciler bu veri tabanından istekte bulunarak kayıtlı verilere ekleme yapar, verileri çeker veya veriler üzerinde güncelleme yapar. Tek merkezden yönetilen veri tabanlarında güvenlik ve hız gibi bir takım sorunlarında çıkması üzerinde sorumluluğun dağıtıldığı modeller uygulanmıştır. Bu modellerde veri tabanı belirli merkezlerden yönetiliyor olmasından ötürü veri güvenliği her zaman sorun olmuştur. Verilerin saklandığı sistemlere kaçak yollarla ulaşabilen bir saldırgan veri tabanını istediği gibi

değiştirebileceği gibi verilerin saklandığı makinelerde çıkacak teknik bir sorun bütün istemcileri olumsuz etkileyecektir. Bu probleme çözüm olarak ise verilerin tüm istemcilerde saklandığı dağıtık veri tabanı yapısı önerilmiştir. Şekil 4.1’ de Veri Tabanı modellerinin şematik gösterimi görülmektedir.

Dağıtık yapılarda ağa bağlı bütün bilgisayarlara düğüm denir ve veriler bu düğümlerde saklanır. Veriler üzerinde işlem yapılmak istenildiği zaman ağdaki düğümlerden onay alınması gerekir ve çoğunluğun onayladığı işlemler gerçekleştirilir. Dağıtık yapıların en büyük avantajlarından biri, bir merkezi olmadığı için bir düğümde sorun çıkması sistemin genelinde bir soruna neden olmayıp, sistemin çalışmaya devam etmesidir. Bu durum ağ katılımcılarının ağ duydukları güveni artırır, çünkü katılımcılar bir şahsa veya merkezi bir bilgisayara güvenmek zorunda değildirler (Nofer et al., 2017). Ayrıca işlemler çoğunluğun onayı ile gerçekleştiği için sisteme kaçak yollarla müdahale etmek isteyen bir saldırgan ağdaki düğümlerin %50’ sinden fazlasına saldırmalı ki dağıtık veritabanında ki veriler üzerinde istediği değişikliği yapabilsin. Bir saldırganın, büyük sistemlerde ağdaki düğümlerin çoğunluğuna saldırması ve ele geçirmesi çok mümkün olmamaktadır.



Şekil 4.2. Blok zinciri yapısı

Dağıtık yapılarda veri güvenliğinin sağlanmak için Blok Zinciri (Block Chain) kullanılır. Blok zinciri sisteminde veriler gruplanarak bloklar oluşturulur ve bu bloklar birbirine bağlanarak zincir oluşturulur. Blok zinciri oluşturulurken zincire özgü belli kurallar çerçevesinde bloklar sisteme yazılır (Uluyol & Ünal, 2020). Şekil 4.2’ de blok zincirinin genel yapısı görülmektedir. Her blok günlük hayatta kullanılan kayıt defterlerinin bir sayfası gibi düşünülebilir. Haliyle blok zinciri genel yapı olarak bir kayıt defterine benzetilebilir. Bu kayıt defteri yani blok zinciri ağdaki bütün düğümlerde belirli kurallar çerçevesinde saklanır. Kuralına uygun biçimde yeni bir blok oluştuğu zaman bu blok oluşturan makine tarafından bütün ağa ilan edilir ve diğer düğümlerde yeni bloğun kurallara uyumluluğunu kontrol ettikten

sonra kendi defterlerine yani blok zincirlerine eklerler. Böylelikle blok zinciri ağdaki bütün düğümlerde saklanır ve bir düğümün ağdan çıkması veya kendi blok zincirini bozması sistemin işleyişini engellemez. Blokların içerisinde saklanmak istenilen verilerin dışında ilgili bloğu tanımlamak için bir takım bilgiler bulunur ki bu bilgiler blok zinciri kullanan farklı sistemlerde farklılık göstermektedir. Genel yapı olarak Kriptografik Özet Algoritmaları başlığı altında anlatılan algoritmalarından biri veya benzerleri kullanılarak her bloğun özeti alınır. Şekil 4.2’ de görüldüğü gibi yeni oluşturulan bir bloğa, bloğu eşsiz yapabilmek için zaman damgası, belirli bir özet değerini bulabilmek için nonce değeri ve bloğu blok zincirine bağlayabilmek için zincirdeki en son bloğun özeti eklenir. Böylelikle bütün bloklar kendisinden bir önce oluşturulan bloğun özetini içinde barındırdığı için bu özet değerleri takip edilerek bütün zincir içinde gezilebilir ki kripto paralarda işlem yapmak isteyen istemcinin yeterli bakiyeye sahip olup olmadığı bu şekilde anlaşılır. Blok zincirinde bulunan herhangi bir blok üzerinde bir saldırgan değişiklik yapmak istediğinde ilgili bloğun özeti değişeceği için o bloktan sonra gelen bütün blokların tekrardan yapılandırılması gerekir. Zincirde bulunan bütün bloklar birbirine bağlı olduklarından bir blokta var olan herhangi bir işlemi değiştirmek, değişiklik yapılan bloktan sonra gelen bütün blokları değiştirmeyi gerektirir (Nakamoto, 2008). Bir düğümdeki bloklarda yapılan bu değişiklik ağdaki düğümlerin çoğunluğunda değiştirilmesi gerekir ki bu da büyük sistemlerde imkânsıza yakındır. Bu şekilde veri güvenliği sağlanmış olur. Özetle blok zincirinin özellikleri şu şekilde sıralanabilir:

- **Ademi Merkeziyetçilik:** Blok zincirini diğer sistemlerden farklı kılan özelliği verilerin tek bir merkezde saklanmaması, dağıtık olarak kaydedilmesi, güncellenmesi ve depolanmasıdır (Uluyol & Ünal, 2020). Klasik merkezi işlem sistemlerinde tüm katılımcılar işlemleri yöneten merkeze güvenmek zorundadır. Bu durum ekstra maliyet ve performans kaybına neden olmaktadır. Blok zinciri sistemlerinde sistemin bir merkezi olmadığı için merkezi sistemlerden kaynaklanan maliyet ve performans sorunları ortadan kalkmıştır.

- **Kalıcılık:** Ağ üzerinde yapılan bir işlemi ağdaki bütün düğümlerin onaylaması gerektiğinden ve ilgili işlemin ağdaki bütün düğümlerde saklanmasından ötürü verileri değiştirmek neredeyse imkansızdır (Zheng et al., 2018).

- **Anonimlik:** İsteyen her kullanıcı blok zinciri ağına istediği kadar hesap oluşturup dahil olabilir yani kimliğini gizleyebilir fakat blok zinciri merkezi bir sunucuda saklanmak yerine ağ üzerindeki düğümlerde açık bir şekilde saklandığı için verilerin tam anlamı ile gizliliği garanti edilemez.

- Denetlenebilirlik: Blok zincirindeki bloklar birbirine bağılı oldukları için ağdaki herhangi bir düğüme erişen bir kullanıcı önceki kayıtları doğrulayabilir ve izleyebilir. Bitcoin blok zincirinde, bloklar takip edilerek genesis bloğa kadar ulaşılabilir (Zheng et al., 2018).

4.1. Blok Zinciri Çeşitleri

Blok zinciri izin mekanizmasına göre üç çeşitten oluşmaktadır: Açık, özel ve konsorsiyum blok zinciri (Uluyol & Ünal, 2020).

4.1.1. Açık Blok Zinciri (Public Block Chain)

Açık Blok Zincirini kullanmak isteyen herkes ağa katılabilir ve blok ekleyebilir. Ethereum akıllı sözleşmelerin kullanımına izin verdiği ve kullanıcıların ağ üzerinde değişik uygulamaları yayınlamalarına olanak sağladığı için bu yapıya örnek olarak gösterilebilir (Mukhopadhyay et al., 2016). Açık blok zinciri sistemlerde katılımcıları belirlemek için bir sınırlama yoktur. Ancak Bitcoin gibi sistemlerde istenilen kurallara uygun blok değeri üretebilen katılımcılar ödüllendirilirler. Bu ödül, Bitcoin için, maddi olarak oldukça değerli olduğundan katılımcılar diğer katılımcılardan önce istenilen kurallara uygun blok değeri üretebilmek için daha fazla işlemci gücüne ihtiyaç duymaktadırlar. Bu durum ağa herkesin katılabilmesine rağmen bazı işlemleri sadece yüksek işlemci gücüne sahip düğümlerin gerçekleştirebileceği anlamına gelir. Bu durum Bitcoin başlığı altında daha detaylı anlatılacaktır.

4.1.2. Özel Blok Zinciri (Private Blockchain)

Özel blok zinciri uygulamalarında sadece sistem tarafından izin verilen katılımcılar ağa katılabilmektedir. Özel blok zinciri sisteminde sistem üzerinde işlem yapan katılımcılar sisteme katılım sağladıktan sonra izin gerekmeden işlem yapabiliyorlarsa bu sistemlere kısmen izin gerektiren sistem adı verilmektedir (Uluyol & Ünal, 2020). Açık blok zincirinde ağ yapısı ile ilgili bir kural değişikliği yapılmak istenildiği zaman yeni kuralı katılımcıların çoğunluğunun kabul etmesi gerekir. Aksi takdirde eski kurallar kullanılmaya devam eder. Özel blok zinciri sistemlerde ise merkezi otorite yeni kurallar belirleyip sistemin işleyişini değiştirebilir ve katılımcılar bu yeni kurallara uymak zorundadır. Aksi takdirde merkezi otoritenin inisiyatifine göre sistem dışı bırakılmak gibi bir takım cezalara çarptırılabilirler. Özel blok zinciri sistemler blok zinciri teknolojisinin maliyet, verimlilik, güvenlik gibi bir takım avantajlarından

yaralanmak için kurulurlar ve kurucular kontrolün kendi ellerinde olmasını isterler. Eris Industries isimli blok zinciri bu sisteme örnek olarak gösterilebilir (Uluyol & Ünal, 2020).

4.1.3. Konsorsiyum Blok Zinciri (Consortium Blockchain)

Açık ve özel blok zinciri sistemlerin birlikte kullanıldığı sistemler konsorsiyum blok zinciri sistemler olarak adlandırılırlar. Konsorsiyum blok zinciri, sistem içerisindeki bazı işlemlerin önceden belirlenen bazı düğümler tarafından gerçekleştirildiği, izinli ve kısmen özel blok zinciri olarak tarif edilebilir. Ağa kimin katılım sağlayacağını ve ağ katılımcılarından hangilerinin madencilik yapacağına bu düğümler karar verir. Ağa katılım kriterlerini, katılımcıların okuma, yazma gibi yetkilerini bir konsorsiyum karar verir ve uygular (Tanrıverdi et al., 2019). Bu sistemlerde konsorsiyum katılımcıları önceden belirler ve katılımcıların yetkileri de bazı düğümler veya konsorsiyum tarafından belirlenir. IBM firması tarafından geliştirilen Hyperledger zinciri bu sistemin en büyük örneğidir (Uluyol & Ünal, 2020). Çizelge 4.1’ de Açık, Özel ve Konsorsiyum blok zinciri yapılarının karşılaştırılması görülmektedir.

Çizelge 4.1. Genel, Özel ve Konsorsiyum blok zincirlerinin karşılaştırılması (Zheng et al., 2017)

	Açık Blok Zinciri	Özel Blok Zinciri	Konsorsiyum Blok Zinciri
Uzlaşma sağlayıcılar	Bütün düğümler	Seçilmiş düğümler	Bir organizasyon
Okuma izinleri	Açık	Açık veya İzinli	Açık veya İzinli
Verimlilik	Düşük	Yüksek	Yüksek
Merkeziyetçilik	Hayır	Kısmen	Evet
Uzlaşma işlemlerine katılım	İzinsiz	İzinli	İzinli

4.2. Blok

Blok zincirini kriptografik şekilde birbirine bağlı bloklar oluşturur. Bir blok zincirinin oluşabilmesi için blok zincirini oluşturan blokların sahip olması gereken bazı temel bileşenler vardır. Blokların içerikleri blok zincirinin tasarımına göre farklılık gösterebilir. Blokların asıl varoluş amaçları bir takım işlem bilgilerini saklamak olduğu için bir bloğun içinde kesinlikle saklamakla yükümlü olduğu işlemler vardır. İşlemlere ek olarak, her blok bir zaman damgası,

önceki bloğun özet (hash) değerini ve özeti doğrulamak için rastgele bir sayı olan bir nonce değerini içerir. Bu konsept, tüm blok zincirinin bütünlüğünü ilk bloğa kadar sağlar (Nofer et al., 2017). Blok zincirinin tasarımına göre blok içerisinde bu sayılan değerlere ek olarak başka değerler olabileceği gibi bu değerlerin boyutları da farklılık gösterebilir. Şekil 4.2’ de genel olarak blok zinciri yapısı görülmektedir. Bitcoin sisteminin kullandığı blok yapısı sonraki başlıklar altında daha detaylı incelenecektir.

4.2.1. Zaman Damgası

Blok başlığında yer alan zaman damgası bloğun benzersiz olmasına katkıda bulunduğu gibi katılımcıların önceki kayıtlara erişerek izlemesine ve kolayca doğrulamasına olanak sağlar. Zaman damgası farklı blok zinciri sistemlerinde farklı olabilir. Örneğin Bitcoin sistemindeki bloklarda zaman damgası 01/01/1970 tarihi saat 00:00’ dan itibaren bloğun oluştuğu zaman kadar geçen sürenin saniye cinsinden 32 bit uzunluğundaki değeridir.

4.2.2. Önceki Bloğun Özet Değeri

Blok zincirini oluşturan her bloğun Kriptografik Özet Algoritmaları başlığı altında anlatılan algoritmalarından biri veya benzerleri kullanılarak özet alınır bu özet değeri bir sonraki bloğun içine eklenir. Böylelikle bloklar birbirine bağlanmış ve zincir oluşturulmuş olur. Ayrıca zincirde bulunan bir bloğun içeriği herhangi bir sebeple değiştirildiği zaman o bloktan sonra gelen bütün blokların içerikleri ve haliyle özet değerleri değişecektir ve sonraki blokların hepsinin tekrar yapılandırılması gerekecektir. Bu işlem belki bir düğümde yapılabilir fakat blok zinciri ağdaki bütün düğümlerde tutulduğu için ağdaki düğümlerin çoğunluğunda bu işlemin gerçekleştirilmesi gerekir. Bu durum Bitcoin gibi sistemlerde para transferinin güvenliği amacıyla kullanılmaktadır. Bu konuya Bitcoin başlığı altında detaylı olarak değinilecektir.

4.2.3. Nonce Değeri

Bitcoin gibi blok zincirlerinde blokların belirli zaman aralıkları ile oluşturulması istenir. Bu zaman aralığını sağlayabilmek için sistem düğümlerden, blok özeti alınırken belirli değerlerden küçük özet değeri bulmalarını ister. Blok içeriği sabit iken özet değerinin belirli değer altında olması için blok içerisinde değiştirilebilir bir değere ihtiyaç vardır. Nonce değeri 0’ dan başlayarak 32 bit uzunluğunda tek tek artırılır ve her seferinde bloğun özeti alınarak özet değerinin istenilen değerin altında olup olmadığı kontrol edilir. İstenilen kritere uygun özet

değerini veren nonce değeri ağa ilan edilir ve diğer düğümler tarafından kontrol edilerek eğer blok uygun ise blok zincirine eklenir. Bu işlen kripto para sistemlerinde madencilik (mining) olarak adlandırılır.



5. BITCOIN

İnternet üzerinden ticaret neredeyse tamamen, elektronik ödemeleri işlemek için güvenilir üçüncü şahıslar olarak hizmet veren finansal kurumlara dayanmaya başladı. Sistem çoğu işlem için yeterince iyi çalışsa da, yine de güvene dayalı modelin içsel zayıflıklarından mustarıptır. Finansal kurumlar anlaşmazlıklarda arabuluculuk yapmaktan kaçınmadığı için, tamamen geri döndürülemez işlemler gerçekten mümkün değildir. Arabuluculuğun maliyeti işlem maliyetlerini artırır, minimum pratik işlem boyutunu sınırlar ve küçük geçici işlemler olasılığını ortadan kaldırır ve geri döndürülemez hizmetler için geri döndürülemez ödemeler yapma kabiliyetinin kaybında daha geniş bir maliyet vardır. Tersine dönme olasılığı ile güven ihtiyacı yayılır. Tüccarlar, müşterilerine karşı dikkatli olmalı ve aksi takdirde ihtiyaç duyacaklarından daha fazla bilgi için onları zorlamalıdır. Dolandırıcılığın belirli bir yüzdesi kaçınılmaz olarak kabul edilir. Bu maliyetler ve ödeme belirsizlikleri, fiziksel para birimi kullanılarak şahsen önlenebilir, ancak güvenilir bir taraf olmadan bir iletişim kanalı üzerinden ödeme yapmak için hiçbir mekanizma yoktur. (Nakamoto, 2008).

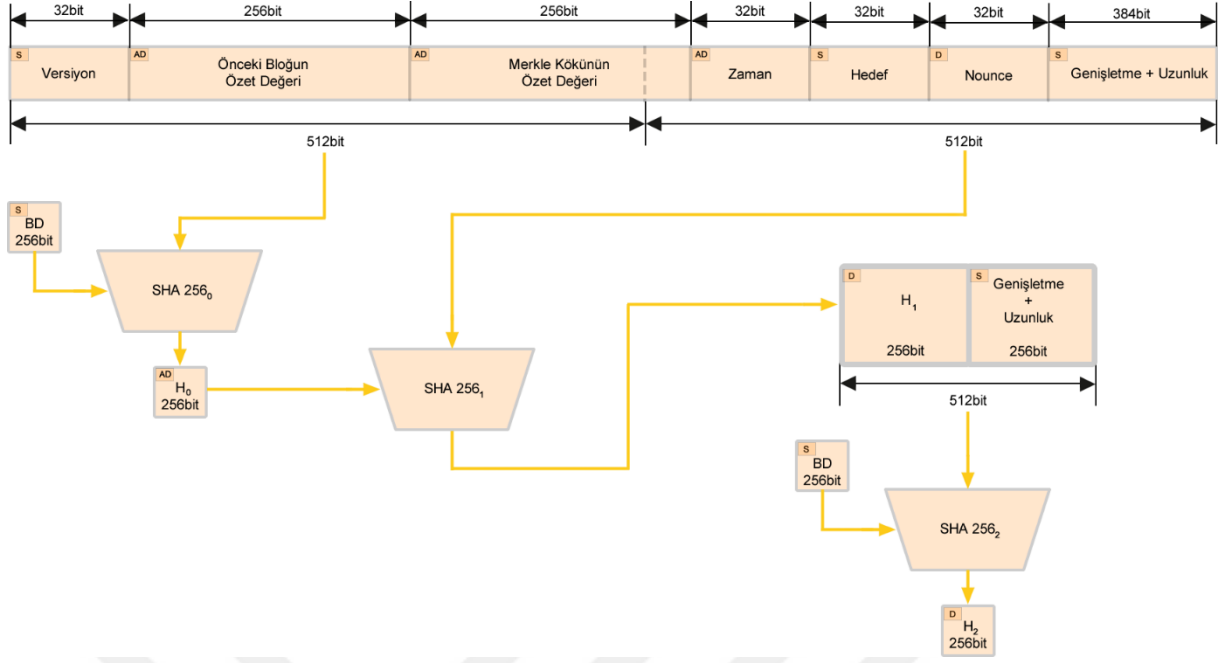
2008 yılında, bir eposta grubunda, kimliği tam olarak bilinmeyen ve Satoshi Nakamoto takma adını kullanan bir kişi veya grup tarafından finans kuruluşlarına ihtiyaç olmadan internet üzerinden para transferine imkân tanıyan, blok zinciri tabanlı sistemin önerildiği bir makale yayınlanmıştır. Satoshi Nakamoto bu makaleye dayalı olarak 2009 yılında geliştirdiği blok zincirine dayalı sistemde “Genesis Blok” adını verdiği ilk bloğu oluşturmuş ve blok oluşturduğu için kendine yazmış olduğu 50BTC ile Bitcoin sistemini başlatmıştır. Bitcoin sisteminde her blok oluşturulduğunda, blok oluşturan düğüme belirli bir miktar Bitcoin yazılarak, madenci diye tabir edilen blok oluşturucuları ödüllendirilir. Bu ödül Satoshi Nakamoto tarafından ilk oluşturulan Genesis blokta 50 Bitcoin olarak belirlenmiş ve her 210.000 blokta bir bu ödül yarı yarıya düşecek şekilde ayarlanmıştır. 28/11/2012 tarihinde ilk 210.000 blok sayısına ulaşıldığı için ödül 25BTC, 06/07/2016 tarihinde ikinci 210.000 blok sayısına ulaşıldığı için ödül 12,5 BTC ve 11/05/2020 tarihinde üçüncü 210.000 blok sayısına ulaşıldığı için ödül 6,25 BTC olarak ayarlanmıştır. Günümüzde blok oluşturulduğunda bloğu oluşturan madenci 6,25 BTC ile sistem tarafından ödüllendirilmektedir. Bitcoin Blok Zinciri 840.000 blok sayısına ulaştığı zaman blok oluşturan madencinin kazanacağı ödül 3,125 BTC olarak sistem tarafından güncellenecektir. Bu başlık yazıldığı sırada Bitcoin zincirinde son olarak 669.532. blok oluşturulmuş durumdadır ve Bitcoin zincirinde sistem yaklaşık 10

dakikada bir blok oluşturacak şekilde ayarlanmıştır. Bu ödüllendirme yöntemi ile aynı zamanda sistem üzerinde para üretimi sağlanmaktadır.

5.1. Bitcoin Blok Yapısı ve Madencilik (Mining) İşlemi

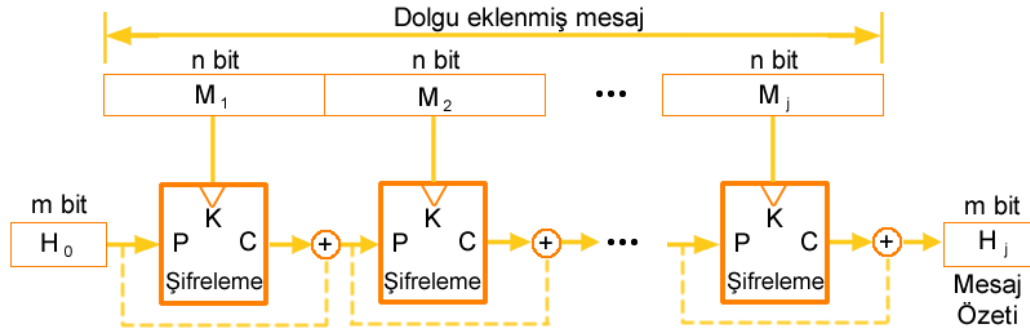
Blok zinciri altyapısını kullanan birçok sistem gibi Bitcoin sisteminin de kendine özgü blok yapısı vardır. Diğer sistemlere benzer şekilde Bitcoin blok yapısında para transfer işlemlerinin (transaction) yer aldığı blok gövdesi ve bloğa ait bir takım bilgilerin yer aldığı blok başlığı bulunur. Blok başlığı iki defa SHA256 algoritmasına girdi olarak gönderilerek üretilen özet değerinin belirlenen bir değerin altında olması beklenir. Örneğin SHA256 algoritmasının üretmiş olduğu 256 bit uzunluğundaki özet değerinin ilk 30 bitinin 0 (sıfır) olması beklenir. Eğer üretilen özet değeri bu kurala uymuyorsa blok başlığında bulunan Nonce değeri bir artırılarak blok başlığı tekrar iki defa SHA256 algoritmasına girdi olarak gönderilir ve bu şekilde uygun değer bulunmaya çalışılır. Bitcoin sistemi yaklaşık her 10 dakikada bir blok oluşturulacak şekilde tasarlanmıştır. Eğer blok oluşma süresi 10 dakikanın altına düşer ise özet değerinin başında bulunması istenilen 0 (sıfır) değeri artırılarak işlem zorlaştırılır ve blok oluşum süresi 10 dakikaya yaklaştırılır. Bu işlemin tersi de geçerlidir. Yani blok oluşma süresi 10 dakikanın üzerine çıkar ise özet değerinin başında bulunması istenilen 0 (sıfır) sayısı azaltılarak işlem kolaylaştırılır ve yine blok oluşum süresi 10 dakikaya yaklaştırılır. Bitcoin madenciliği yapan düğümlerin bulmaya çalıştıkları değer bu özet değeridir ve bu özet değerini arama işlemine madencilik (mining) adı verilir.

Şekil 5.1’ de Bitcoin blok başlığında yer alan bilgiler ve madencilik işleminde kullanılan algoritma şematik olarak gösterilmiştir. Blok başlığında yer alan bazı değerler madencilik işlemi boyunca sabit kalır, bazıları ise değişim gösterir. Şekil 5.1’de görülen blok başlığındaki değerlerden madencilik işlemi sürecinde sabit kalanlar sol üst köşelerindeki “S” harfi, duruma göre zaman zaman değişim gösteren değerler “az değişen” manasında “AD” ve sürekli değişen değerler ise “D” harfi ile gösterilmiştir. Örneğin nonce değerinin sol üst köşesinde “D” harfi olması bu değerlin sürekli değişim gösterdiği, versiyon değerinin sol üst köşesinde “S” harfi olması bu değerlin daima sabit kaldığı anlamına gelir.



Şekil 5.1. Bitcoin blok başlığı ve madencilik işlemi için kullanılan özet algoritması

Bitcoin blok başlığı 640 bit uzunluğundadır. Başlık boyutu dolgu alanı ile 1024 bit uzunluğa genişletilir ve bu başlık SHA256₀ ve SHA256₁ fonksiyonlarını besler. Aslında bu iki fonksiyon tek bir SHA256 algoritmasının parçalarıdır ve blok başlık boyutunun dolgu alanı ile 1024 bit uzunluğu çıkarılması SHA 256 algoritmasının standart işlemidir. Yani bu yapılan işlemler Bitcoin blok yapısına özgü işlemler değildir. SHA256 algoritması girdi olarak kullanılacak mesajı 512 bit uzunluğunda bloklara böler ve bu blokları bir önceki fonksiyonun çıktısı ile sırayla kullanır. SHA256 algoritmasının çalışması Kriptografik Özet (Hash) Algoritmaları başlığı altında detaylı olarak anlatılmıştır. Bitcoin sisteminde standart SHA256 algoritması sonucunda Şekil 5.1’de görülen H₁ özet değeri üretilmiştir ve Bitcoin sistemine özgü olarak H₁ özet değeri bir kez daha SHA256₂ algoritmasını besler ve istenilen H₂ özet değeri elde edilir. SHA256 algoritması Merkle-Damgard düzenini temel alırken Bitcoin bu yönü ile Davis-Meyer düzenine benzemektedir. Şekil 5.2’ de Davis-Meyer düzeni görülmektedir.



Şekil 5.2. Davis-Meyer Düzeni

Belirlenen değerin altında olması gereken değer H_2 özet değeridir. Eğer belirlenen büyüklükte değer elde edilemedi ise nonce değeri bir artırılarak H_2 özet değeri tekrar hesaplanır. İstenilen büyüklükte H_2 özet değerini bulan düğüm doğru nonce değeri ile birlikte bloğu Bitcoin ağındaki diğer düğümlere yayımlar ve H_2 özet değerinin doğruluğunu kontrol eden düğümler bu bloğu kendi zincirlerine eklerler. H_2 özet değerini bulan düğüm günümüzde 6,25BTC Bitcoin ile ödüllendirilir.

Bitcoin madenciliğini temel olarak 5.1’ de yer alan H_2 özet değerinin yine şekilde görülen nonce değerini değiştirerek belirli bir değeri altında olmasını sağlamak olarak açıklayabiliriz (Courtois et al., 2014). SHA256 algoritması Kriptografik Özet Fonksiyonları başlığı altında detaylı olarak anlatıldığı gibi 64 döngüden oluşur. Satoshi Nakamoto 5.1’de görüldüğü gibi iki SHA256 algoritmasını arka arkaya kullanarak döngü sayısını 128 yapmış ve böylelikle gelişen işlemci gücü ile SHA256 algoritmasına karşı geliştirilebilecek saldırılar için önlem almıştır.

$$H_2 = SHA256(SHA256(blok\ başlığı)) \quad (5.1)$$

5.1.1. Blok Başlığı

Bitcoin blok başlığı versiyon, önceki bloğun özet değeri, merkle kökünün özet değeri, zaman, hedef ve nonce değerlerinden oluşur. Bitcoin blok başlığında yer alan alanlar, uzunlukları ve kısa tanımları Çizelge 5.1’de gösterilmiştir. Şekil 5.1’ de görülen “Genişletme + Uzunluk” alanı normalde blok başlığında yoktur, bu alanı SHA256 algoritması mesaj uzunluğunu 512bit katlarına ayarlamak için ekler. “Genişletme + Uzunluk” alanının detayları Kriptografik Özet Algoritmaları başlığı altında detaylı olarak ele alınmıştır.

Para transfer işlemleri (transactions) blok gövdesinde yer alır ve transfer işlemlerinin sayısı blok başlığının uzunluğunu etkilemez.

Çizelge 5.1. Bitcoin blok başlığında yer alan alanlar

Alan	Boyut	Tanım
Versiyon	32 bit	Bitcoin yazılımının yeni blok oluşturmak için geçerli olan kuralların versiyonu.
Önceki Bloğun Özet Değeri	256 bit	Bitcoin blok zincirine en son eklenen bloğun özet değeri.
Merkle Kökünün Özet Değeri	256 bit	Para transfer işlemlerinin (transactions) oluşturduğu Merkle Ağacının özet değeri.
Zaman Damgası	32 bit	1970-01-01 T00:00 UTC' den bloğun oluştuğu zamana kadar geçen saniye cinsinden süre.
Hedef	32 bit	Mevcut hedef değeri.
Nonce	32bit	Hedefte belirtilen özet değerinin altında özet değeri bulmak için değiştirilecek alan.
Genişletme + Uzunluk	384 bit	SHA256 algoritması tarafından eklenen alan.

5.1.1.1. Versiyon

Bitcoin yazılımının yeni blok oluşturmak için geçerli olan kuralların versiyonun belirtildiği bu alan 32 bit uzunluğundadır. Blok oluşturma kurallarında yeni bir değişiklik yapılanaya kadar sabit kalan bir değerdir. Sabit bir alan olduğu için Şekil 5.1'de "S" harfi ile işaretlenmiştir.

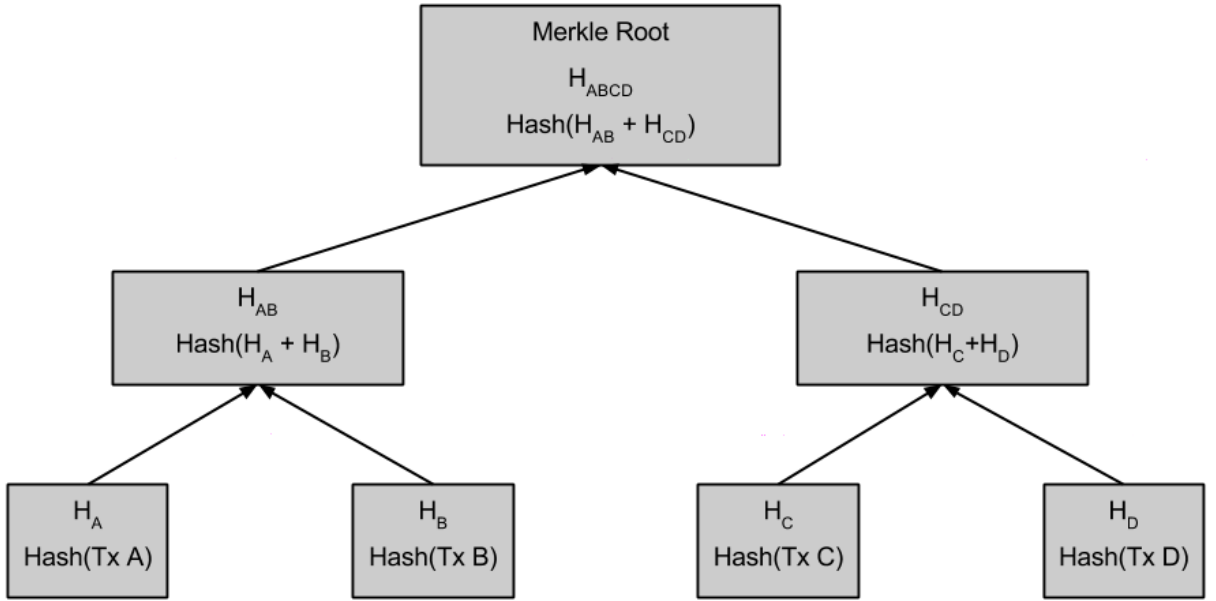
5.1.1.2. Önceki bloğun özet değeri

Bitcoin blok zincirine en son eklenen bloğun özet değeridir. Bitcoin blok zincirine yeni bir blok eklendiği zaman madenci bu bloğun özet değerini yeni oluşturacağı bloğun başlığına ekler ve nonce değerini değiştirerek belirlenen hedef doğrultusunda yeni özet değeri oluşturmak için çalışmalara başlar. Bitcoin sistemi yaklaşık her 10 dakikada bir yeni blok oluşturacak şekilde tasarlandığı için, yeni oluşturulacak bloğun başlığında yer alan önceki blok özet değeri, yaklaşık 10 dakikada bir değişir. Bu sebeple Şekil 5.1' de "AD" olarak işaretlenmiştir.

5.1.1.3. Merkle kökünün özet değeri

Merkle Ağacı (Merkle Tree), kriptografi ve bilgisayar bilimlerinde kullanılan, bir alanda saklanan verilerin ne olduğunu açığa çıkarmadan bu verileri doğrulamak ve özetlemek için kullanılan bir veri yapısıdır. Merkle Ağaçları kriptografik özet fonksiyonlarını kullanırlar ve büyük veri yapılarının verimli ve güvenli bir şekilde doğrulanmasını sağlarlar. Merkle ağacı tepe kısmında kök ve taban kısmında yapraklardan oluşmaktadır.

Merkle Ağacı, Bitcoin blok zincirindeki bir blokta yer alan tüm transfer işlemlerinin (transactions) özetini almak için kullanılır ve bir işlemin bir bloğa dahil edilip edilmediği doğrulamak için çok önemli bir rol oynar. Bitcoin bloklarında kullanılan Merkle Ağacı kriptografik özet algoritması olarak SHA256 algoritmasını kullanır ve iki defa kullanmasından ötürü çift SHA256 olarak adlandırılır. Bitcoin Merkle Ağacı' nın özet alma işleminde kullandığı denklem 5.2'de verilmiştir.



Şekil 5.3. Bir Merkle ağacındaki düğümlerin hesaplanması (A. M. Antonopoulos, 2016)

Merkle Ağacı aşağıdan yukarıya doğru yapılandırılır. Şekil 5.3'de görülen Merkle Ağacında A, B, C ve D işlemlerinin özeti alınmaktadır. Özet alma işleminde 5.2'de verilen denklem kullanılır. Merkle Ağacının tabanında yer alan bu işlemlerin ilk olarak özetleri alınır ve H_A, H_B, H_C ve H_D değerleri elde edilir.

$$H_A = SHA256(SHA256(Transaction A)) \quad (5.2)$$

H_A, H_B, H_C ve H_D deęerleri ikili ikili gruplanarak tekrar zet deęerleri hesaplanır. H_A ve H_B birleřtirilir ve zeti deęeri hesaplanarak H_{AB} ve H_C ve H_D birleřtirilir ve zet deęeri hesaplanarak H_{CD} zet deęerleri elde edilir. Bu hesaplama iřleminde 5.3’de verilen denklem kullanılır.

$$H_{AB} = SHA256(SHA256(H_A + H_B)) \quad (5.3)$$

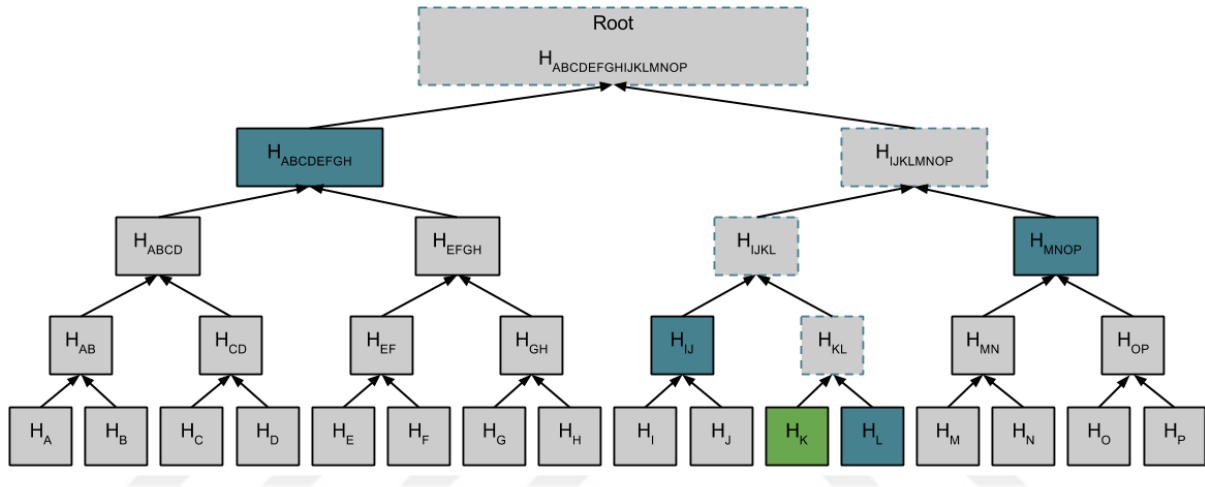
İřlem, Merkle Aęacında yer alan btn dęmlerin ikili gruplar halinde zetleri alınarak bir tek zet deęeri elde edilene kadar devam eder ve bu son bulunan deęere Merkle kk denir. Merkle Kk 32Byte uzunluęundadır ve blok bařlıęına kaydedilerek blokta bulunan btn iřlemleri zetler (A. M. Antonopoulos, 2016).

Merkle Aęacı ikili aęaç yapısında olduęu iin, aęaç yapraklarının sayısı ift olmak zorundadır. Eęer aęaçta yer alan iřlemler tek sayıda ise, son iřlem oęaltılarak farklı bir iřlemmiř gibi aęaca dahil edilerek aęaç dengeli hale getirilir (A. M. Antonopoulos, 2016). Őekil 5.3’deki D iřleminin olmadıęını varsayarsak Merkle Aęacı’ nın dengesiz bir aęaç olur. Bu durumda Merkle Aęacı’ nı dengeli aęaç yapabilmek iin C iřlemi iki defa dęm olarak eklenir.

Merkle Kk, Merkle Aęacı’ nda bulunan yaprakların sayısında baęımsız olarak sabit bir uzunluęa sahiptir. Bitcoin bloęu ierisinde yzlerce iřlem olabilir. İřlem sayısı ne olursa olsun Bitcoin bařlıęında bulunan Merkle Kk 256 bit uzunluęundadır ve bir iřlemin blok ierisinde olduęunu doęrulamak iin kullanılır. Ayrıca blok ierisinde bulunan bir iřlemde usulsz Őekilde deęiřiklik yapıldıęı zaman blok bařlıęında yer alan Merkle Kk deęiřir ve buna baęlı olarak blok bařlıęının zet deęeri deęiřir. Blok bařlıęının zet deęerinin deęiřmesi blok zincirinde bulunan sonraki bloęun blok bařlıęında yer alan “nceki Blok zet Deęeri” alanının deęiřmesine sebep olur ve sonraki bloęunda blok bařlıęının zet deęeri deęiřir. Bylelikle deęiřiklik yapılan bloktan sonraki btn blokların zet deęerlerinin tekrar hesaplanması gerekir ve bu deęiřiklik iřleminin aęda bulunan btn dęmlerde de uygulanması gerekir. Bitcoin sisteminde bir bloęun oluřturulması yaklařık 10 dakika srdę dřnlrse, oluřturulmuř bir blokta deęiřiklik yapmak neredeyse imkansızdır. Bylelikle Bitcoin sisteminde gerekleřtirilen bir iřlemin usulsz bir Őekilde deęiřtirilmesi engellenmiř olur.

Bitcoin sisteminde gerekleřtirilen bir iřlemin yer aldıęı bloęun iinde var olup olmadıęı anlamak iin $\log_2(N)$ adet 256 bitlik zet deęeri gereklidir. Bu durum iřlem sayısı arttıka doęrulama iřlemini olduka kolaylařtırır. nk bir sayının artıřına baęlı olarak o

sayının iki tabanlı logaritması yavaş artar. Bitcoin blok zincirindeki bir blok 1MB büyüklüğündedir ve binden fazla işlem içerir. Binden fazla işlem içerisinde bir işlemin kanıtını sağlamak için 10 veya 12 özet değerine sahip kimlik doğrulama yolu yeterli olacaktır. Böylelikle bir işlemin blok içerisindeki varlığını doğrulamak için bütün blok içeriğini kontrol etmeye gerek kalmaz. Şekil 5.4’de görülen, birçok veri ögesini özetleyen bir Merkle Ağacı’nda yer alan “K” işleminin blok içerisindeki varlığını doğrulamak için 4 adet 256 bit özet değeri yeterlidir. H_L , H_{IJ} , H_{MNOP} ve $H_{ABCDEFGH}$ özet değerlerinin bilinmesi durumunda Merkle Köküne ulaşılabilirse “K” işleminin varlığı doğrulanmış olur.



Şekil 5.4. Birçok veri ögesini özetleyen bir Merkle Ağacı (A. M. Antonopoulos, 2016)

5.1.1.4. Zaman damgası

Bitcoin sistemindeki bloklarda zaman damgası 01/01/1970 tarihi saat 00:00’ dan itibaren bloğun oluştuğu zaman kadar geçen sürenin saniye cinsinden 32 bit uzunluğundaki değeridir. 1 Saniye içerisinde blok başlığının hedeflenen özet değeri üretilemezse zaman damgası değişeceği için istenilen özet değerinin üretilmesini mümkün kılar.

5.1.1.5. Hedef

Hedef değeri üretilen bloğun blok başlığının özet değerinin maksimum olacağı değeri belirler. Daha öncede bahsedildiği üzere Bitcoin sistemi her 10 dakikada bir blok oluşturulacak şekilde ayarlanmıştır. Bu 10 dakikalık süre hedef değeri değiştirilerek ayarlanır. Merkezi sistemlerde bu tarz ayarlamalar merkezdeki makine tarafından gerçekleştirilir. Bitcoin gibi merkezi olmayan sistemlerde ise ayarlama işlemini her düğüm kendi içinde gerçekleştirir.

Hedef değeri 2016 blokta bir tekrar ayarlanır. Her bloğun 10 dakikada bir oluşturulduğu göz önüne alınırsa 2016 blok için geçmesi gereken zaman 20.160 dakikadır. 20.160 dakika iki haftalık bir zamana tekamül eder. Yani iki haftada bir blok başlığında yer alan hedef değeri güncellenir. Bütün düğümler her 2016 blokta bir geçen süreyi kontrol ederler. Eğer geçen süre 20.160 dakikadan kısa ise hedef değeri değiştirilerek işlemlerin zorluk derecesi artırılır, eğer 20.160 dakikadan uzun ise hedef değeri değiştirilerek işlemlerin zorluk derecesi azaltılır.

Hedef değeri 256 bit uzunluğunda olması gerekirken blok başlığında 32 bit alan ayrılmıştır. 32 bit uzunluğundaki hedef değerinden matematiksel bir işlem ile 256 bit hedef değeri elde edilir. Sistem tarafından belirtilen 32 bitlik hedef değerinden karşılaştırma için kullanılacak 256 bit şu şekilde elde edilir: Örneğin, Bitcoin sistemi tarafından 32 bitlik hedef değeri $0 \times 1903a30c$ olarak verilsin. Bu değerın son altı onaltılık basamağı katsayı, ilk iki basamağı ise üs olarak adlandırılmaktadır. Yani, bu bloğun doğrulanması için verilmiş hedef değerinin üs kısmı 0×19 , katsayı kısmı ise $0 \times 03a30c$ olarak bulunmaktadır. Hedef değerinin 256 bitlik gösterimi 5.4'deki denkleme göre hesaplanmaktadır (A. M. Antonopoulos, 2016).

$$Hedef = Katsayı \times 2^{(8 \times (\text{üs} - 3))} \quad (5.4)$$

Bu denklem kullanılarak 32 bit uzunluğundaki $0 \times 1903a30c$ değerinden 256 bit uzunluğundaki hedef değeri 5.5' deki gibi hesaplanır.

$$Hedef = 0x03a30c \times 2^{(0x08 \times (0x19 - 0x03))}$$

$$Hedef = 0x03a30c \times 2^{0x08 \times 0x16}$$

$$Hedef = 0x03a30c \times 2^{0xB0} \quad (5.5)$$

5.5'de elde edilen değerın onluk sayı sistemindeki karşılığı 5.6'daki gibi bulunur.

$$Hedef = 238.348 \times 2^{176} \quad (5.6)$$

5.6'daki işlem yapıp elde edilen değer ikilik sayı sistemine dönüştürüldüğünde 196bit uzunluğunda bir değer elde edilir. Bu değeri 256bite tamamlamak için başına 60 adet 0 (sıfır) değeri konulmalıdır ki bu durum blok başlığından üretilen özet değerinin ilk 60 bitinin 0 (sıfır) olması anlamına gelir. Bu şartlara uyan bir blok üreten madenci ürettiği bloğu ağa yayınlamaya ve

ağdaki diğer düğümler üretilen bloğun özet değerinin hedef değerinden küçük olduğunu kontrol eder ve bloğu zincire ekleyerek iş kanıtı problemini onaylamış olur.

Bitcoin sistemi son 2016 bloğun oluşma zamanını kontrol ederek 5.7’de verilen denklem ile yeni hedef değerini hesaplar.

$$Yeni Hedef Değeri = Eski Hedef Değeri \times \left(\frac{Son\ 2016\ Blok\ İçin\ Geçen\ Süre}{20160\ Dakika} \right) \quad (5.7)$$

5.1.1.6. Nonce

32 bit uzunluğundaki Nonce değeri, blok başlığının her SHA256 özet değeri hesaplandığında, özet değerinin hedef alanında belirlenen değere uyup uymadığı kontrol edilip, eğer uyumsuz bir özet değeri üretildi ise lineer olarak tek tek arttırılan alandır. Nonce değeri değiştirildikten sonra blok başlığının tekrar SHA256 özeti alınır ve hedef değeri ile tekrar karşılaştırılması yapılır. Bu döngü hedef alanına uygun özet değeri bulunana kadar devam eder. Nonce değeri 32 bit uzunluğunda olduğu için 2^{32} farklı değer alabilir.

Bitcoin sistemi 2009 yılında kullanılmaya başlandığı zaman hem Bitcoin ağındaki katılımcı azlığı hem de Bitcoin madenciliği için kullanılan işlemcilerin günümüze göre yavaş olmasından dolayı bütün Nonce değerlerini denemek bir saniyeden uzun zaman alıyordu. Haliyle mevcut blok için bütün Nonce değerleri istenilen hedef değerine uygun özet üretmediği durumlarda zaman damgası değişeceğinden ötürü özet değeri tekrar hesaplanarak uygun özet değeri bulunabiliyordu. Ancak günümüzde katılımcı sayısının ve işlemci gücünün oldukça yüksek seviyelere ulaşması sebebi ile bütün Nonce değerlerini denemek bir saniyeden daha az zaman almaktadır ve eğer mevcut blok için istenilen özet değeri üretilmedi ise düğümler zaman damgasının değişmesini beklemek zorunda kalacaklardır. Bu problemi çözmek için madenciler blokta yer alan işlemlerin başına kendilerine özgü olarak ekledikleri işlemin içinde yer alan ekstra nonce değerini değiştirirler. Ekstra nonce değeri Çizelge 5.2’de yer alan değişken alanın içerisinde yer almaktadır. Her bir madenci seçmiş olduğu işlemlerden bir blok oluştururken ilk işlem olarak kendi adres bilgisini içeren ve bazı sabit alanlar bulunan değeri bloğa ekleyerek iş kanıtı problemini çözdüğünde bu değerde yer alan adres bilgisine göre sistem tarafından kendisine ödeme yapılmaktadır. Bu ilk işlem Bitcoin sisteminde üretim işlemi olarak adlandırılmaktadır. Üretim işleminde madencinin kendine özgü eklediği işlemde yer alan bileşenler Çizelge 5.2’de verilmiştir (A. M. Antonopoulos, 2016).

Çizelge 5.2. Madencinin bloğa eklediği işlemde yer alan bileşenler

Boyut	Bileşen	Açıklama
32 bayt	İşlem özet değeri	Tüm bitlerin değeri 0'dır.
4 bayt	Çıktı indeksi	Tüm bitlerin değeri 1'dir.
1-9 bayt	Madenci tarafından yeni bir blok doğrulandığında alınacak ödül ve işlem ücretlerinin yer aldığı bilginin boyutu	2 ile 100 bayt arasında değişen bilgi kısmıdır.
Değişken	Madenci tarafından yeni bir blok doğrulandığında alınacak ödül ve işlem ücretlerinin yer aldığı bilgi	Keyfi bir değerdir ve madencilik için gerekli olan ekstra Nonce değeri bu keyfi değerinde yer almaktadır.
4 bayt	Sıra numarası	Tüm bitlerin değeri 1'dir.

5.1.1.7. Genişletme + Uzunluk

Genişletme + Uzunluk alanı SHA256 algoritmasının eklediği bir alandır. SHA256 algoritması özet alma işlemine başlamadan önce özeti alınacak mesajı 512 bit uzunluğunda bloklara böler ve blokların en sonuna 128 bit uzunluğunda, mesajın uzunluk bilgisini ekler. Mesaj ve uzunluk bilgisinin arasında ise son mesaj bloğunu 512bit uzunluğa tamamlayacak şekilde dolgu alanı ekler. Kriptografik Özet Algoritmaları başlığı altında bu alanın nasıl hesaplandığı detaylı olarak anlatılmıştır. Şekil 5.1'de de görüldüğü gibi Bitcoin blok başlığı 640bit uzunluğundadır ve Bitcoin sistemi çift SHA256 algoritması kullanır. SHA256₀ ve SHA256₁ algoritmaları aslen tek bir SHA256 algoritmasının parçalarıdır ve blok başlığının sonuna 128 bit uzunluğunda 640 sayısını ikilik olarak ekler ve mesaj uzunluğunu 512bit katı yapabilmek adına blok başlığı ile uzunluk bilgisinin arasına ilk biti 1 diğer bitleri 0 olacak şekilde 256 bit dolgu alanı ekler. SHA256₂ algoritması ise girdi olarak SHA256₁ algoritmasının ürettiği özet değerini kullandığı için mesaj uzunluğu 256 bittir. 256bit uzunluğundaki mesajın sonuna 256 sayısı 128bit ile ikili olarak eklenir. Mesaj uzunluğunu 512bit katı yapmak için mesaj ile uzunluk bilgisinin arasında ilk biti 1 (bir) sonraki bitler 0 (sıfır) olacak şekilde 128 bit dolgu alanı eklenir.

Bitcoin sisteminin detayları incelendiğinde akıllara bütün düğümlerin aynı özet değerini üreteceği gelebilir. Oysa her düğümün blok oluşturmak için seçeceği işlemler ve bu işlemlerin başına eklenen madenciye ait işlemler farklılık göstereceği için her bloktaki Merkle Kökü değeri değişecektir ve dolayısı ile her düğümün ürettiği blok ve özet değeri farklı olacaktır.

5.1.2. Blok Gövdesi

Bitcoin sisteminin asıl var oluş amacı merkezi olmayan bir sistem üzerinden güvenli şekilde para transferi işlemlerini gerçekleştirmektir ve sistem üzerindeki diğer her şey güvenli para transferi işlemleri için tasarlanmıştır. Bu amacı gerçekleştirmek için dağıtık veritabanı yapısını ve blok zinciri teknolojisini kullandığından daha önce bahsedilmişti. Bitcoin sisteminin asıl işlevi olan para transfer işlemleri blok zincirini oluşturan blokların gövdesinde yer alır. Her blok yaklaşık olarak 500 adet transfer bilgisi içerir.

Transfer işlemi oluşturulurken kullanılacak paralar, transfer edilecek cüzdan bilgisi, transfer işleminin yer alacağı bloğu blok zincirine ekleyen madenciye ödenecek işlem ücreti ve transfer işleminde yer alan paraları harcama yetkisine sahip olan kişinin imzası gibi bir takım bilgiler transfer işleminin içerisine eklenir. Bir işlemin boyutu yaklaşık 300 – 400 byte uzunluğundadır. Transfer işlemi oluşturulduktan sonra Bitcoin ağında yayınlanır ve ağdaki bütün düğümlerin transfer işlemi doğrulaması sağlanır. Transfer işlemi Bitcoin ağına yayınlanırken ilk olarak bir düğüme gönderilir ve bu düğüm transfer işlemi içerisinde yer alan doğrulama scriptini çalıştırarak işlemin uygun olup olmadığını kontrol eder. Eğer transfer işlemi uygun ise düğüm tarafından gönderene onay mesajı gönderilirken transfer işlemi düğümün bağlı olduğu 3 veya 4 düğüme gönderilir. İşlemi alan diğer düğümler tarafından tekrar aynı doğrulama işlemi yapıldıktan sonra bu düğümlerinde bağlı olduğu yine 3 veya 4 düğüme gönderilir. Bu şekilde transfer işlemi ağa bağlı tüm düğümler tarafından kabul edilene kadar katlanarak genişleyen bir dalgalanma şeklinde dağılır. Transfer işlemi doğrulayan düğümler işlemi transfer işlemleri havuzuna ekler ve transfer işlemi burada bir bloğa dahil edilmeyi bekler. Eğer işlem doğrulanmazsa transfer işlemleri havuzuna eklenmeyeceği gibi gönderene ret mesajı gönderilir. Böylelikle transfer işleminin ağdaki düğümlere hızlı bir şekilde yayılması sağlanırken transfer işlemi teslim alan bütün düğümlerin işlemi bağımsız şekilde doğrulaması sağlanır. Bu şekilde hatalı oluşturulan bir işlem bir veya birkaç düğüm tarafından doğrulanabilir fakat ağdaki düğümlerin birçoğu tarafından reddedilir ve hatalı işlemlerin gerçekleştirilmesinin önüne geçilmiş olur.

5.1.2.1. Transfer işlemlerinin yapısı

Bitcoin sistemindeki standart bir transfer işlemi girdiler ve çıktılar olmak üzere iki ana bloktan oluşur. Ancak madencilik sonucunda elde edilen bazı transfer işlemlerinde girdiler yoktur, sadece madencinin cüzdanına kazandığı ödül miktarını belirten bir çıktı vardır. Bu özel

bir durumdur ve blok gövdesinde yer alan transfer işlemlerinin ilk sırasında yer alır. Standart olarak oluşturulan bir transfer işleminde girdiler ve çıktılar arasında uyumsuzluk varsa veya işlemi oluşturanın girdilerden bir kısmını harcama yetkisi yoksa işlem doğrulanmaz yani gerçekleştirilemez. Bitcoin transfer işleminin yapısı Çizelge 5.3’ de gösterilen alaları içerir.

Çizelge 5.3. Transfer işlemlerinin yapısı

Boyut	Alan	Açıklama
4 Byte	Versiyon	İşlemin uyduğu kurallar
1-9 Byte	Girdi sayacı	İşlemin içerdiği girdi sayısı
Değişken	Girdiler	Bir veya daha fazla işlem girdisi
1-9 Byte	Çıktı sayacı	İşlemin içerdiği çıktı sayısı
Değişken	Çıktılar	Bir veya daha fazla işlem çıktısı
4 Byte	Kilitleme Zamanı	Zaman değeri veya blok numarası

Bitcoin sisteminde harcama yapılırken kullanılan cüzdanlar gerçek hayatta kullanılan cüzdanlara benzetilerek oluşturulmuştur. Örneğin gerçek bir cüzdanda 1 adet 50TL’lik, 2 adet 20TL’lik ve 1 adet 10TL’lik banknot olmak üzere toplam 100TL para olsun. Eğer bu cüzdandan 75TL değerinde bir ödeme yapmak istersek 50TL’lik, 20TL’lik ve 10TL’lik banknotları cüzdandan çıkararak toplam 80TL ödeme yaparız ve karşılığında para üstü olarak aldığımız 5TL’lik banknotu cüzdana geri koyarız. Bitcoin sisteminde de cüzdan işlemlerinin temelini gerçek cüzdanlarda kullanılan banknotlara benzetilebilecek harcanmamış işlem çıktıları (unspent transaction output, UTXO) oluşturur. Nasıl ki cüzdanımıza bulunan bir banknot bize aitse ve o banknotu harcama yetkisi bizdeyse benzer şekilde bir Bitcoin cüzdanında bulunan UTXO’ nun harcama yetkisi o cüzdan sahibine aittir. Ödeme sırasında UTXO’ların kullanım şekli de banknotlara benzer. Nasıl ki bir banknot bölünemiyorsa UTXO’larda bölünemez. Örneğin bir Bitcoin cüzdanında 2, 3 ve 7 BTC değerinde UTXO’lar olsun ve cüzdan sahibi 4BTC değerinde bir ödeme yapmak istesin. Bu ödeme iki farklı şekilde yapılabilir. Ya transfer işlemine 2 ve 3 BTC değerindeki UTXO’lar girdi olarak, 4BTC değerindeki UTXO çıktı olarak kaydedilir ve aynı transfer işlemine 1BTC değerinde UTXO para üstü olarak işlenir ya da 7BTC değerindeki UTXO transfer işlemine girdi olarak, 4BTC değerindeki UTXO çıktı olarak kaydedilir ve 3BTC değerindeki UTXO para üstü olarak işlenir. Ödemenin hangi şekilde

yapılacağı bitcoin cüzdan uygulaması tarafından otomatik olarak belirlenir. Burada dikkat edilirse her işlemde de daha önce var olmayan 1 ve 3BTC değerindeki UTXO para üstü olarak, 4BTC değerindeki UTXO ise çıktı olarak oluşturuldu. Bitcoin sisteminde ödeme yaparken transfer işlemine girdi olarak kaydedilecek UTXO lar daha önce oluşturulup bu cüzdana gönderilmiş UTXO lar olmak zorundadır ancak transfer işleminde oluşan çıktı ve para üstü daha önce oluşturulan miktarda UTXO olmak zorunda değildir. Yani 3BTC değerinde para üstü almak için üç adet 1BTC değerindeki UTXO lar kullanılmak zorunda değildir. Böylelikle Bitcoin sisteminde istenilen her miktarda UTXO oluşturulabilir fakat oluşturulan UTXO lar bölünemez, az önceki örnekte olduğu gibi bütün olarak harcanmalıdırlar. Türk Lirası kuruş olarak iki ondalık basamağa bölünebildiği gibi Bitcoin’de sistemin en küçük birimi olan satoshi olarak sekiz ondalık basamağa bölünebilir.

Bitcoin sisteminde bir cüzdanın depolanmış bakiyesi diye olgu yoktur, o cüzdana kilitlenmiş UTXO’lar vardır. Cüzdan bakiyesini öğrenmek istediğimizde sistem blok zincirine kayıtlı bütün blokları tek tek dolaşarak o cüzdana kilitlenmiş UTXO’ları bulur ve bunları toplayarak cüzdan bakiyesini hesaplar.

Transfer işlemleri, UTXO’yu mevcut sahibinin imzasıyla kilidini açarak tüketir ve UTXO’yu yeni sahibinin bitcoin adresine kilitler. Çıktı ve girdi zincirinin istisnası, her bloktaki ilk işlem olan, coinbase işlemi adı verilen özel bir işlem türüdür. Bu işlem transfer işlemlerini bir bloğa eklenerek blok zincirine dahil edilmesini sağlayan madenci tarafından yapılır ve madenciliğin ödülü olarak o madenciye ödenecek yepyeni bir bitcoin yaratır. Bu şekilde sistemde yeni bitcoinler üretilmiş yeni sistemde para basılmış olur.

Transfer işlemi çıktısı

Bitcoin sisteminde gerçekleştirilen her transfer işlemi, sadece ödeme yapılan cüzdan sahibinin kullanabilmesi için kilitlenmiş bir UTXO oluşturur. Oluşturulan bu UTXO, oluşturulduğu transfer işleminin eklendiği bloğun blok zincirine eklenmesi ile bitcoin ağı üzerindeki bütün düğümler tarafından tanınır ve sadece cüzdan sahibinin özel anahtarı ile harcanabilir. İşlem çıktıları iki bölümden oluşur:

- Satoshi cinsinden belirtilen bir miktar bitcoin.
- İşlem çıktısı sonucunda oluşturulan UTXO’nun harcanabilmesi için gerekli koşulları içeren kitleme komut dosyası.

Çizelge 5.4’ de bir işlem çıktısının içerdiği alanların boyutları ve açıklamaları görülmektedir.

Çizelge 5.4. Transfer işlemi çıktısı

Boyut	Alan	Açıklama
4 Byte	Miktar	Satoshi cinsinden Bitcoin miktarı
1-9 Byte	Kilitleme komut dosyası boyutu	Kilitleme komut dosyasının Byte olarak boyutu
Değişken	Kilitleme komut dosyası	UTXO’ yu harcamak için gerekli koşulları içeren kilitleme komut dosyası

Kilitleme ve Kilit açma komut dosyaları FORTH benzeri bir programlama dilinde yazılmıştır. Forth, orijinal olarak Chuck Moore tarafından tasarlanan zorunlu yığın tabanlı bir bilgisayar programlama dilidir. FORTH hem komutları etkileşimli yürütür yani resmi işletim sistemi olmayan sistemler için uygun bir kabuk haline getirir hem de daha sonra çalıştırılmak üzere komut dizilerini derler.

Bir UTXO’ ya ait kilitleme komut dosyası, ilgili UTXO’nun kayıtlı olduğu cüzdan adresi, üretildiği transfer işleminin özet değeri ve indeks numarası, satoshi cinsinden miktarı gibi bilgileri içerir. Bir işlem doğrulandığında, harcama koşulunu karşılayıp karşılamadığını görmek için her girdideki kilit açma komut dosyası, karşılık gelen kilitleme komut dosyasıyla birlikte çalıştırılır. Bitcoin sistemindeki transfer işlemlerinde yer alan çıktılarını kilitlemek ve girdilerin kilidini açmak için bir komut dosyası kullanılmasının amacı kilitleme ve kilit açma işlemleri için sonsuz sayıda koşul tanımlama imkanı sunmasıdır. Böylelikle zaman içerisinde değişen koşullara uygun olarak UTXO’ların harcama koşulları istenildiği gibi her 210.000 blokta bir yarıya düşürülür. İlk blokta bu ödül 50BTC iken günümüzde 6,125BTC ye düşmüştür.

Transfer işlemi girdileri

Bir transfer işleminin birden fazla girdisi olabilir. Örneğin cüzdanında 2 ve 3BTC değerinde UTXO’lar olan bir cüzdan sahibi 4BTC değerinde bir ödeme yapmak istediği takdirde, UTXO’ların bölünme özelliği olmadığı için, transfer işlemine 2 ve 3BTC değerindeki iki UTXO’yu da girdi olarak eklemek zorundadır. Aksi takdirde 4BTC değerinde bir ödeme gerçekleştiremez. Bu işlem sonucunda 4BTC karşı tarafa gönderilmek için, 1BTC para üstü

olarak aynı cüzdana geri gönderilmek için iki farklı çıktı üretilir. Bu işlemde yer alan iki girdi de ayrı olarak kendi harcanabilme koşullarını içeren kilit açma komut dosyalarını içerirler. Çizelge 5.5’ de bir işlem girdisinin içerdiği alanların boyutları ve açıklamaları görülmektedir.

Çizelge 5.5. Transfer işlemi girdisi

Boyut	Alan	Açıklama
32 Byte	Transfer işleminin özet değeri	UTXO’ nun üretildiği transfer işleminin özet değeri
4 Byte	Çıkış indeks değeri	Harcanacak UTXO’ nun indeks değeri
1-9 Byte	Kilit açma komut dosyası uzunluğu	Kilit açma komut dosyasının byte cinsinden uzunluğu
Değişken	Kilit açma komut dosyası	Kilitleme komut dosyasında belirtilen koşulları karşılayan kilit açma komut dosyası
4 Byte	Sıra numarası	Transfer işleminin sıra numarası

Transfer işlemi ücretleri

Bitcoin sisteminde gerçekleştirilen transfer işlemlerinde, işlemi blok zincirine ekleyen madenciye ücret ödemek transfer işlemi gerçekleştiren cüzdan sahibinin inisiyatifine bırakılmıştır. Ancak ücret ödemesi yapılmayan işlemleri madenciler bir bloğa ekleme konusunda çok istekli davranmadıkları için işlem ücreti içermeyen transfer işlemlerinin blok zincirine eklenmesi, işlem ücreti içeren transfer işlemlerinden daha uzun sürmektedir. Yani işlem ücretleri teşvik olarak hizmet eder. Dışardan herhangi bir müdahale olmadığı sürece cüzdanlar işlem ücretlerini otomatik olarak hesaplar ve transfer işlemine dahil eder. İşlem ücretleri cüzdan tarafından otomatik olarak hesaplanırken transfer işleminin miktarına göre değil byte cinsinden boyutuna göre hesaplanır. Mevcut durumda byte başına işlem ücreti 2satoshi olarak belirlenmiştir.

Transfer işlemlerinin yapısında işlem ücreti ile ilgili herhangi bir alan yoktur, sadece girdiler ve çıktılar vardır. İşlem ücreti ise girdiler ve çıktılar arasındaki farka göre hesaplanır. Örneğin cüzdanında 2 ve 3BTC değerinde UTXO’lar olan bir cüzdan sahibi 4BTC değerinde bir ödeme yapmak istediği takdirde, UTXO’ların bölünme özelliği olmadığı için, transfer işlemine 2 ve 3BTC değerindeki iki UTXO’yu da girdi olarak eklemek zorundadır. Yani

transfer işlemine toplam 5BTC değerinde girdi ve 4BTC değerinde çıktı ekleyecektir. Transfer işlemine ek olarak geri alınması gereken para üstü miktarını da eklemesi gerekir. Cüzdan sahibi para üstü olarak 0,75BTC eklemiş olsun. Bu durumda girdiler ve çıktılar arasındaki 0,25BTC fark, transfer işlemi blok zincirine ekleyen madenciye işlem ücreti olarak aktarılacaktır. Örnekten de anlaşılacağı üzere işlem ücretleri 5.8'deki gibi hesaplanır.

$$\text{Ücretler} = \text{Toplam (Girdiler)} - \text{Toplam (Çıktılar)} \quad (5.8)$$

Madenciler transfer işlemlerini bloğa eklerken transfer işlemindeki ücreti ve transfer işleminin yaşını dikkate alırlar. Eğer blokta yeteri kadar yer varsa işlem ücreti içermeyen transfer işlemleri de bloğa eklenir. Her blokta 50 kilobyte boyutunda bir alan yüksek işlem önceliği olan işlemler için ayrılır. İşlem öncelik değeri 57.600.000' den büyük olan işlemler yüksek öncelikli işlem olarak kabul edilir ve öncelik değerine göre sırayla aday bloğa dahil edilirler. Bu alana işlemler alınırken işlem ücretine bakılmaz. Bu şekilde işlem ücreti olmayan transfer işlemlerinin de gerçekleştirilmesi sağlanır fakat havuzda bekleyen, işlem ücreti içermeyen transfer işlemi sayısı fazla ise işlem ücreti olmayan işlemlerin bir bloğa dahil edilip onaylanması uzun zaman alabilir. 50 kilobyte boyutundaki yüksek işlem önceliği olan işlemlere ayrılan alan dolduktan soran diğer işlemler işlem ücretine göre seçilerek blok doldurulur. İşlem önceliği, bir günde üretilen blok miktarının (144), işlem boyutuna (byte) bölümünün, 100.000.000 satoshi ile çarpılarak hesaplanır. 5.9' da işlem önceliği hesaplama denklemi görülmektedir.

$$\text{İşlem Önceliği} = \frac{144}{\text{işlem boyutu (byte)}} \times 100.000.000 \text{ satoshi} \quad (5.9)$$

Transfer işlemlerinin doğrulanması

Bitcoin ağındaki herhangi bir düğüme ulaşan bir transfer işlemi düğüm tarafından doğrulandıktan sonra gönderene onay mesajı gönderirken transfer işlemi, bağlı olduğu 3 ila 4 düğüme gönderir ve aynı doğrulama işlemi bu düğümler tarafından da gerçekleştirilir. Bu şekilde transfer işlemi geçerli ise kısa sürede katlanarak ağdaki bütün düğümlere yayılır, geçerli değilse yani işlemde bir hile söz konusu ise düğümlerin çoğunluğu tarafından reddedileceği için işlem gerçekleştirilmeyecektir. Her düğüm her işlemi şu kriter listesine göre kontrol eder:

- İşlemlerin girdileri ve çıktıları mevcut mu?
- Her çıktı değeri ve çıktı değerlerinin toplamları izin verilen aralıkta mı? (0BTC -21.000.000BTC aralığında mı?)
- İşlem boyutu 100 Byte' dan büyük mü?

- İşlemin söz dizimi (syntax) ve veri yapısı doğru mu?
- Her girdi için referans verilen bir çıktı var mı? Bu çıktı harcanabilir durumda mı?
- Girdi değerlerinin toplamı çıktı değerlerinin toplamından büyük mü veya Girdi değerlerinin toplamı çıktı değerlerinin toplamına eşit mi?
- İşlem boyutu maksimum blok boyutunda küçük mü?
- İşlemden yer alan girdilerin hepsinin özet değeri var mı?
- İşlemden yer alan herhangi bir girdinin referans verdiği çıktı sadece bu işlemde mi kullanılıyor?
- İşlem ücreti çok mu?
- Her giriş için var olan kilit açma komut dosyası karşılık gelen kilitleme komut dosyası ile doğrulanıyor mu?

Bitcoin sisteminde oluşturulan bir transfer işlemi bu soruların hepsine olumlu cevap veriyorsa doğrulanır ve bir bloğa dahil edilip blok zincirine eklendikten sonra işlem gerçekleştirilmiş olur.

6. ÖNERİLER VE GELİŞTİRİLEN UYGULAMA

2009 senesinde Bitcoin sistemi ile başlayan blok zinciri tabanlı kripto para teknolojisi geçen yıllar içerisinde, bilgisayarların donanımsal ve yazılımsal olarak gelişmeleri neticesinde farklı bir boyut kazanmış, bir blok zinciri üzerinde birden fazla kripto para uygulaması çalıştırılabildiği gibi bununla birlikte akıllı kontratlar gibi farklı uygulamalar da barındırabilecek bir hal almıştır. Kripto paralar ilk çıktıkları zaman bir eğlence aracı gibi görünmesine karşın, zaman içerisinde yasadışı ödemelerde kullanılmış ve akabinde özellikle büyük firmaların kripto paralar ile ödeme almaya başlamaları ile birlikte, günümüzde devletlerin yasalar ile kontrol etmeye çalıştıkları sistemler, yatırımcılar için önemli bir yatırım aracı haline gelmişlerdir. Yaşanan bu gelişmeler gösteriyor ki kripto paranın icadı, ekonomide paranın icadı kadar önemli bir gelişmedir. Kripto paraların ekonomik boyutunda bunlar yaşanırken teknik boyutunda ise bir takım sıkıntılar ortaya çıkmaktadır. Bu sıkıntıların bazıları önceden tahmin edilebilirken bazıları gelişen teknoloji ile zamanla ortaya çıkmıştır.

6.1. Bitcoin Sistemi İçin Öneriler

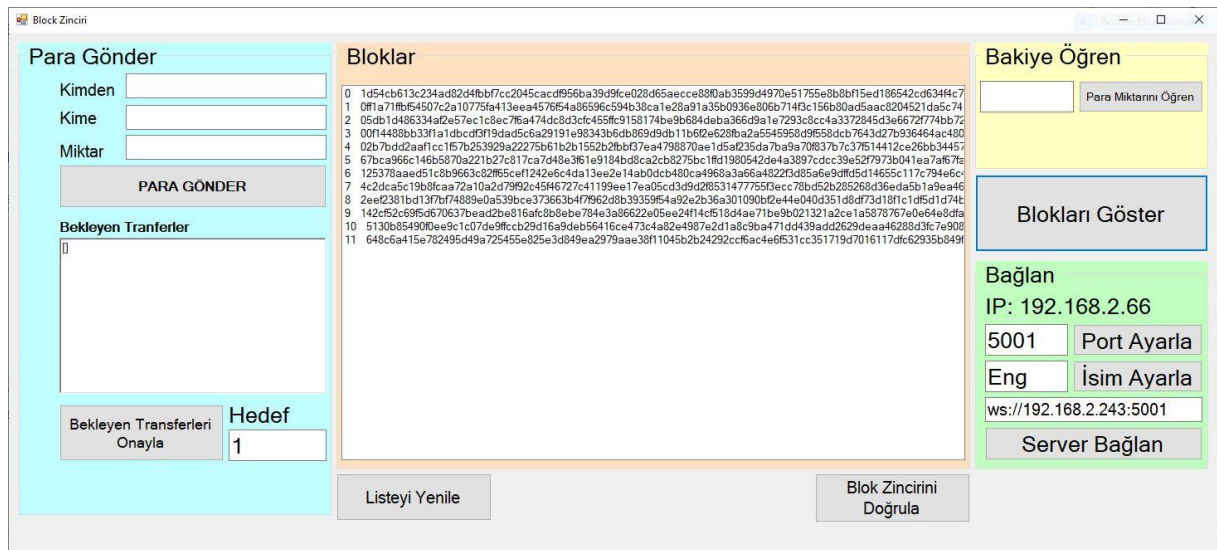
Bitcoin sisteminin çalışmaya başladığı 2009 senesinden buyana bilgisayarların işlemci güçleri oldukça artmıştır. Sistemin çalışmaya başladığı ilk zamanlarda oluşturulan bir bloğun özet değeri hesaplamak beklenen zaman dilimlerinde sürerken, gelişen teknoloji ile bu zaman dilimi beklenenin çok altına inmiş hatta bütün nonce değerlerini denemek için geçen süre 1 saniyenin altına düşmüştür. Bu durumda madenciler saniyede bir değişen zaman damgasının beklemek zorunda kalmışlardır ve bu problemi çözmek için blokta yer alan işlemlerin başına kendilerine özgü olarak ekledikleri işlemin içindeki ekstra nonce değerini değiştirme yoluna gitmişlerdir. Zaman içerisinde bilgisayarların işlem gücündeki artıştan dolayı ekstra nonce değerini değiştirmek bu problemi çözmeyebilir. Böyle bir durumda ekstra nonce değerini değiştirmeye ek olarak, mevcut durumda kullanılan SHA256 kriptografik özet algoritmasının yerine, daha karmaşık olan ve hesaplaması daha fazla zaman alan SHA512 özet algoritması kullanılabilir. Geliştirilen uygulamada SHA512 özet algoritması kullanılmış ve sistemin çalışabilirliği gösterilmiştir. Şekil 6.1' de geliştirilen uygulamanın arayüzü görülmektedir.

Mevcut Bitcoin sisteminde nonce değeri her blok için 1 değerinden başlar ve 32 bit değer tek tek denenir. Sistemde bulunan işlem gücü yüksek düğümler çok hızlı hesaplama yapabildikleri için düşük güçlü düğümlerin yeni blok üretme ihtimallerini ortadan kaldırmışlardır. Bu durum güçlü makinelerin bir kesimin elinde bulunmasına sebep olabilir ve

blok zinciri sistemlerin en büyük tehditlerinden biri olan yüzde elli bir saldırılarını mümkün kılar. Bu problemin çözümü ağdaki düğümlerin belli bir kesimin elinde bulundurulması yerine sistemin olabildiği kadar dağıtılmasıdır. Eğer düşük güçlü düğümlerinde zincire blok ekleme ihtimalleri söz konusu olabilirse ağdaki düğümler genele yayılacaktır. Eğer nonce değerini tek tek artırarak hesaplama yapmak yerine, düğümler rasgele nonce değeri üreterek hesaplama yaparlarsa düşük güçlü düğümlerinde blok üretme şansları olacaktır ve sistem daha çok yayılacaktır. Ancak blok zinciri sistemlerde bir güncelleme yapmak için mevcut düğümlerin yarısından fazlasının güncellemeyi kabul etmesi gerekir ki bu durumu kabul ettirmek biraz zor olacaktır.

6.2. Geliştirilen Uygulama

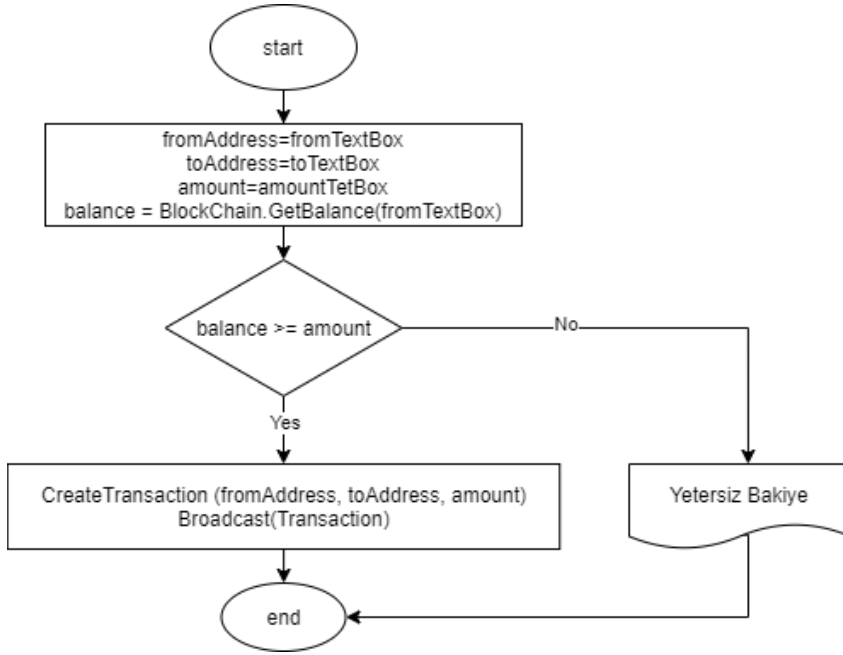
Geliştirilen program blok zinciri tabanlı bir kripto para simülatörüdür. Şekil 6.1’ de önerilen güncellemelerin gerçekleştirildiği simülatör programının arayüzü görülmektedir. Simülatör programı tezde bulunulan önerilerin gerçekleştirilebildiğini göstermek ve önerileri test edebilmek için tasarlandığından önerilen alanların geliştirilmesine ağırlık verilmiştir. Program Visual Studio 2017 çatısı altında C# programlama dili ile geliştirilmiştir ve blok zinciri JSON formatında kaydedilip okunmuştur. Yazılımın geliştirilme ve test aşamalarında i7-7500U 2.7Ghz işlemci, 8GB DDR4 RAM bellek ve 2GB ekran kartı donanımlarına sahip Win10 işletim sistemi kurulu bilgisayar kullanılmıştır.



Şekil 6.1. Geliştirilen uygulamanın arayüzü

6.2.1. Para Gönderme Algoritması

Programın para gönder bölümünde bulunan “Kimden” alanına, programın test aşaması bittiği zaman, otomatik olarak kullanıcının kendi cüzdan adresi gelecektir. “Kime” alanına ise ödeme yapılmak istenilen cüzdan adresi girilecektir ve miktar alanına gönderilmek istenilen para miktarı girildikten sonra ödeme işlemi yapılabilir.



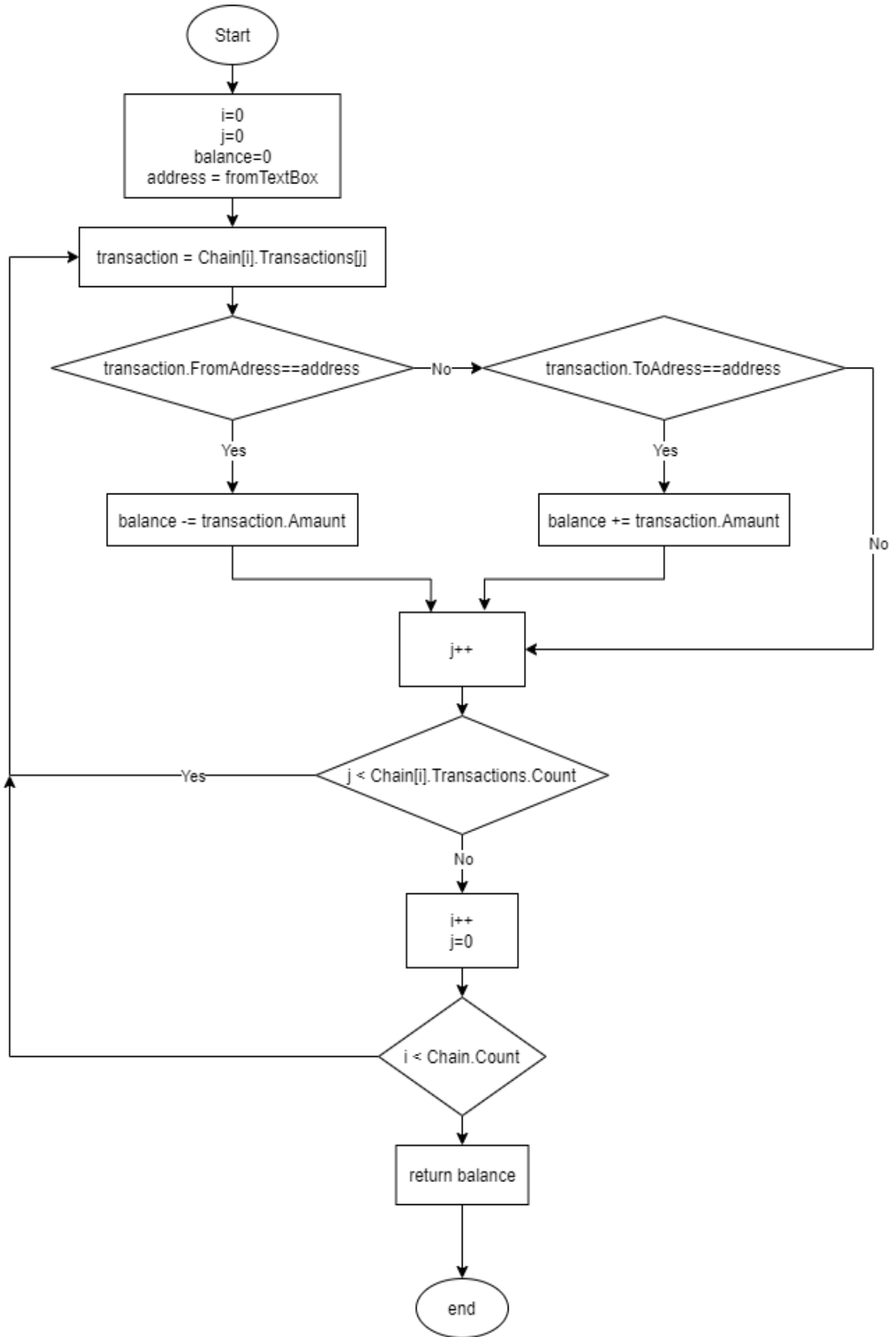
Şekil 6.2. Para gönderme algoritmasının akış diyagramı

“Para Gönder” butonuna basıldığı zaman program tüm blok zincirinde gönderici adrese ait olan bütün transfer işlemlerini tarayarak gönderici adresin yeterli bakiyeye sahip olup olmadığını hesaplar ve eğer daha önce gönderici cüzdan adresine yapılan ödemelerin toplamı transfer edilmek istenilen miktardan az ise program yeterli bakiye olmadığına dair bir mesaj verir işlemi gerçekleştirmez. Eğer yeterli bakiye var ise yani gönderici cüzdan adresine daha önce yapılan transferlerin toplamı ödeme miktarına eşit veya çok ise transfer işlemi bekleyen transferlerin tutulduğu bir listeye eklenirken düğüme bağlı diğer düğümlere gönderilir. Şekil 6.2’ de Para gönderme algoritmasının akış diyagramı verilmiştir. Şekil 6.2’ de verilen Akış diyagramındaki $balance=Blockchain.GetBalance(fromTextBox)$ satırında yer alan $Blockchain.GetBalance(fromTextBox)$ metodunun algoritmasının akış diyagramı Şekil 6.3’ de verilmiştir. Transfer işlemini alan düğümler kendi blok zincirlerinde Şekil 6.3’ de verilen akış diyagramına göre bakiye kontrolü yaparlar ve akabinde Şekil 6.2’ de verilen akış diyagramına göre yeterli bakiye varsa transfer işlemini bekleyen transferler arasına eklerler. Böylelikle hileli

bir transfer işlemi yapılma durumuna karşın önlem alınmış olur. Yani yeni transfer işlemi alan bütün düğümler bakiye kontrolü yapacağı için, uygunsuz bir ödeme işlemi yapılmak istenirse bu işlem ağdaki düğümlerin çoğunluğu tarafından reddedileceği için gerçekleşmeyecektir.

6.2.2. Bakiye Hesaplama Algoritması

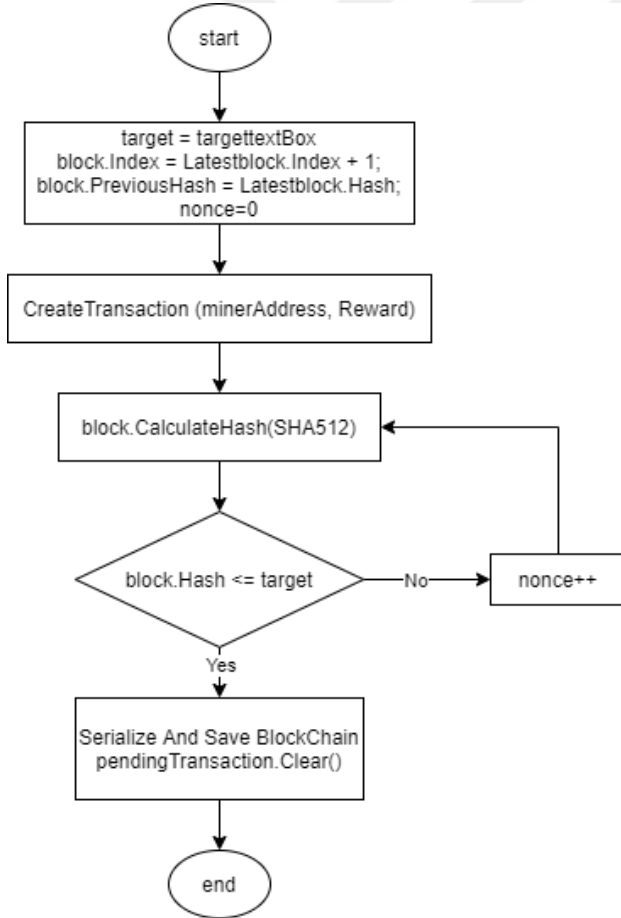
Bütün para transferi ve kripto para uygulamalarında olduğu gibi geliştirilen uygulamada da mükerrer ödeme işlemlerinin önüne geçmek için ödeme işlemi gerçekleştirilmeden önce para transferi yapmak isteyen adresin yeterli bakiyeye sahip olup olmadığı kontrol edilir. Bazı blok zinciri tabanlı kripto para uygulamalarında bu işlem için blok zincirindeki bütün bloklar ve bloklar içinde yer alan bütün transfer işlemleri tek tek kontrol edilerek para transferi yapmak isteyen adrese gelen ve giden paralar toplanır ve ilgili adresin bakiyesi hesaplanır. Bitcoin sistemi bu işlem için, daha önce “Transfer işlemlerinin yapısı” başlığı altında detaylı olarak anlatılan UTXO’ ları kullanır. Geliştirilen simülatör programında bakiye hesaplamak için Şekil 6.3’ de akış diyagramı görülen algoritma kullanılır. Bakiye Hesaplama Algoritmasına göre geliştirilen program ilk bloktan başlanarak son bloğa kadar bütün bloklar teker teker kontrol edilir. Her bloğun içinde bulunan bütün transfer işlemlerinin (transactions) içinde bakiye hesaplaması yapılacak adrese ait olan işlemler aranır. İlgili adrese ait işlemler tespit edildikten sonra eğer bu adrese para transferi yapıldı ise bakiye miktarı transfer yapılan para miktarı kadar arttırılır. Eğer ilgili adresten para transferi yapıldı ise bakiye miktarı transfer yapılan para miktarı kadar azaltılır. Son blokta yer alan son transfer işlemi kontrol edildikten sonra bakiye hesaplama işlemi biter.



Şekil 6.3. Bakiye kontrolü algoritmasının akış diyagramı

6.2.3. Blok Özeti Bulma (Madencilik) Algoritması

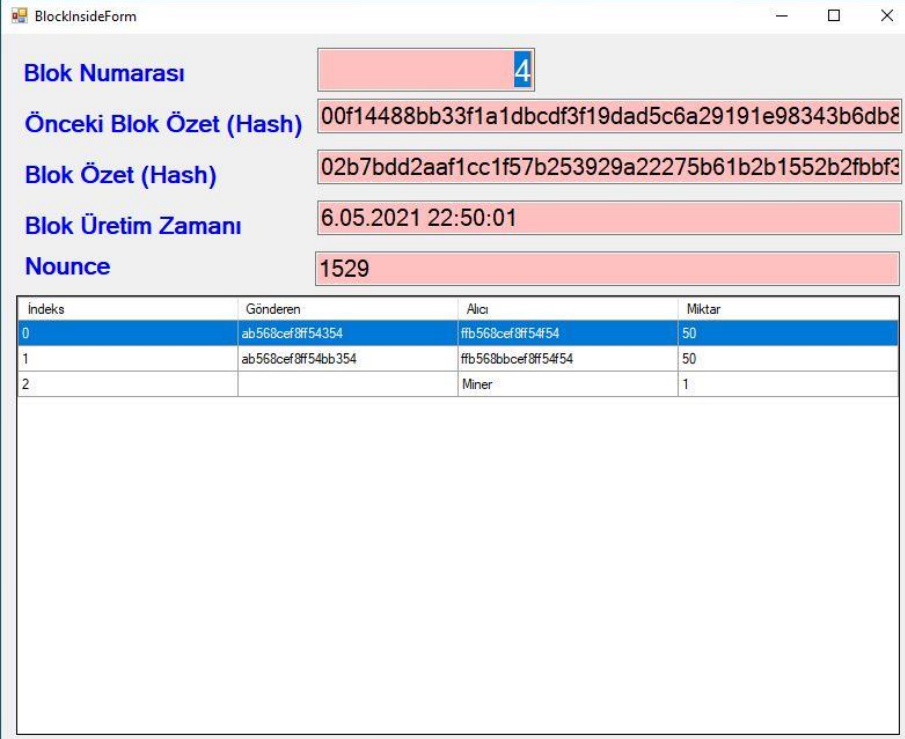
Transfer işlemi, bekleyen transferler arasına eklendikten sonra bekleyen transferlerden bir blok oluşturularak uygun SHA512 özet değeri bulunmaya çalışılır. 512 bit özet değerinin ilk kaç bitinin 0 (sıfır) olacağı hedef alanında belirlenir. Hedef alanında belirlenen değer aslında 512 bit uzunluğundaki hedef değerinin değersiz kaç bitinin 1 (bir) olacağını belirler. Blok içerisine beklemede olan transfer işlemleri ile birlikte blok zincirine son eklenen bloğun indeks numarasının bir fazlası indeks numarası olarak, yine son eklenen bloğun özet değeri önceki bloğun özet değeri (previoushash) olarak ve sistem saati zaman damgası olarak eklenir. Bu şekilde blok oluşturulur ve belirlenen kriptografik özet algoritması kullanılarak özet alma işlemi gerçekleştirilir. Özet alma işlemi hedef değeri ile belirlenenden daha küçük bir değer bulunana kadar nonce değeri artırılarak işleme devam edilir. Uygun özet değeri bulununca, özet değerini bulmak için kullanılan nonce değeri bloğa eklenir. Şekil 6.4’ de Blok özeti bulma (madencilik) algoritmasının akış diyagramı görülmektedir.



Şekil 6.4. Blok özeti bulma (madencilik) algoritmasının akış diyagramı

6.2.4. Blok Zincirine Blok Eklenmesi

Uygun nonce değeri bulunduktan sonra ilgili blok, bloğu bulan düğümün bağlı olduğu bilgisayarlara gönderilir. Geliştirilen uygulama bazı önerilerin gerçekleştirilebilir olduğunu göstermek için geliştirilen bir simülatör olduğu için blok bulan düğüm sadece bloğu değil tüm blok zincirini bağlı olduğu düğümlere gönderir. Blok zincirini yakalayan düğümler kendi blok zincirleri ile gelen blok zincirini karşılaştırırlar ve yeni blok eklenerek gelen zincir uygunsuz kabul edilir ve zincirdeki blokların indeks numaraları ve SHA512 özet değerleri program arayüzünde bulunan “Bloklar” alanında listelenir. Listelenen özet değerlerinin üzerine çift tıkladığı zaman blok içerisindeki işlemleri ve bloğun diğer değerlerini gösteren bir pencere açılır. Şekil 6.5’ de örnek bir bloğun içerisindeki işlemleri gösteren pencere görülmektedir. Şekil 6.5 incelendiğinde blok içerisinde bloğun indeks numarası, önceki bloğu SHA512 özet değeri, mevcut bloğun SHA512 özet değeri, bloğun oluşturulma zamanı, uygun SHA512 özet değerini yakalamak için bulunan nonce değeri ve transfer işlemlerinin yer aldığı görülmektedir. Transfer işlemlerini içerisinde yer alan, gönderen bilgisi olmayan ve alıcı kısmında “Miner” yazan transfer işlemi bloğu üreten madencinin kazandığı ödül miktarını göstermektedir ve bu şekilde para üretimi gerçekleştirilmektedir.



The screenshot shows a window titled "BlockInsideForm" with the following fields:

- Blok Numarası**: 4
- Önceki Blok Özet (Hash)**: 00f14488bb33f1a1dbcdf3f19dad5c6a29191e98343b6db8
- Blok Özet (Hash)**: 02b7bdd2aaf1cc1f57b253929a22275b61b2b1552b2fbbf3
- Blok Üretim Zamanı**: 6.05.2021 22:50:01
- Nounce**: 1529

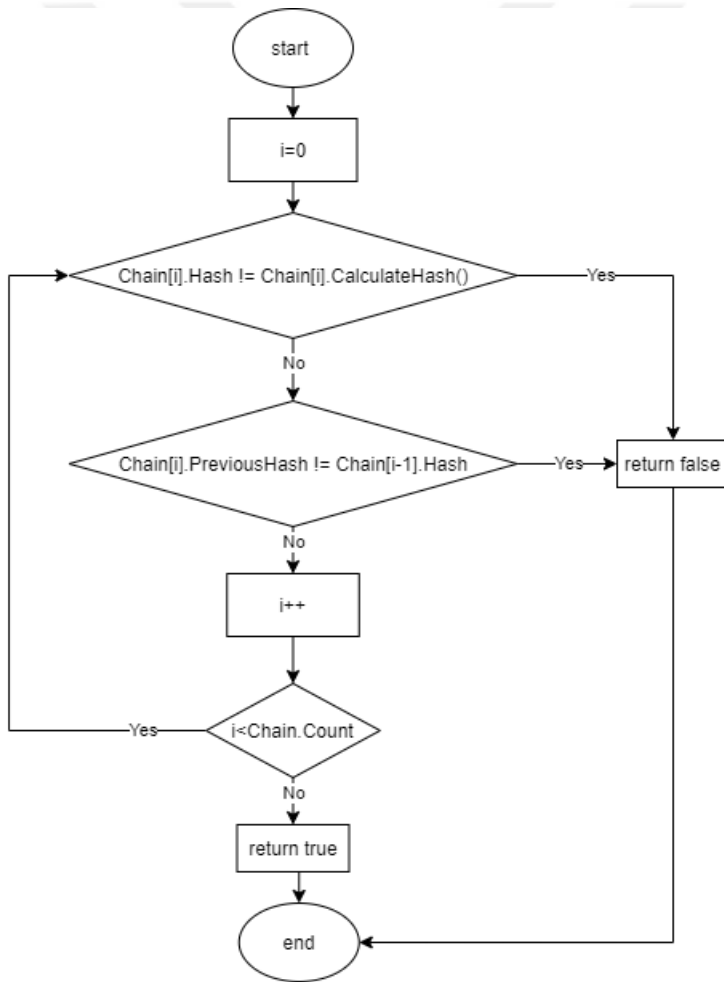
Below the fields is a table with the following data:

İndeks	Gönderen	Alıcı	Miktar
0	ab568cef8f54354	fb568cef8f54f54	50
1	ab568cef8f54bb354	fb568bbcef8f54f54	50
2		Miner	1

Şekil 6.5. Oluşturulan bir bloğun içeriği

6.2.5. Blok Zincirinin Doğrulanması

Her yeni blok üretildiği zaman blok zincirinin doğruluğu kontrol edildiği gibi test amaçlı olarak program arayüzünde bulunan “Blok Zincirini Doğrula” butonu yardımı ile de kayıtlı blok zincirinin doğruluğu kontrol edilebilir. Şekil 6.6’ da blok zinciri doğrulama algoritmasının akış diyagramı görülmektedir. Doğruluk kontrolü yapılırken bütün blokların tek tek özet değerleri tekrar alınır ve blok içerisinde var olan özet değeri ile karşılaştırılarak iki özet değerinin aynı olması beklenir. Böylelikle bloğun içeriğinde herhangi bir değişiklik yapılmadığı tespit edilmiş olur ve doğrulama işleminin ilk aşaması bitmiş olur. Blok zincirinde var olan mevcut blokların hepsinin nonce değerleri daha önceden bulunduğu için tekrar SHA512 özet değerini hesaplamak çok zaman almayacaktır.



Şekil 6.6. Blok Zinciri doğrulama algoritmasının akış diyagramı

Doğrulama işleminin ikinci aşamasında ise mevcut blokta yer alan önceki bloğun özet değeri ile önceki bloğun hesaplanan özet değerinin aynı olması beklenir. Bu aşamada ise bloğun uygun yerde bulunduğu kontrolü yapılır. Bu aşamadan sonra blok doğrulanmış olur bir

sonraki bloğun kontrolüne geçilir. Bu doğrulama işlemi zincirde yer alan bütün bloklar için gerçekleştirilir. Eğer herhangi bir blok içeriğinde usulsüz bir şekilde değişiklik yapıldı ise, örneğin bir saldırgan kendine para transferi yaptı ise, doğrulama işlemi esnasında usulsüz işlem yapılan blokta var olan özet değerleri ile hesaplanan özet değerleri birbiri ile uyuşmayacağı için program hatalı bloğu indeks numarasını belirterek sorunlu bloğu belirten bir uyarı mesajı verecektir. Eğer doğrulama işlemi yeni blok oluşturulduğu zaman otomatik olarak yapılıyorsa ve yeni gelen blok zinciri doğrulanmasa blok zincire eklenmez. Bu şekilde ağdaki düğümlerin yarısı yeni oluşturulan bloğu doğrulayamazsa blokta yer alan transfer işlemleri gerçekleştirilemez.

6.2.6. Simülasyon Programı Ağ Yapısı

Simülasyon test aşamasında yerel ağdaki düğümler arasında çalışmaktadır ve düğümler arası bağlantı el ile yapılmaktadır. Bağlanılmak istenilen düğümün IP (Internet Protocol) adresi ve port numarası girildikten ilgili düğüme bağlantı sağlanabilir. Bağlantı sağlandıktan sonra iki düğümden birbirlerine kendi blok zincirlerini gönderirler ve her düğüm gelen blok zincirini doğrular. Doğrulama işlemi gerçekleştiğinden sonra düğümler gelen blok zincirini kendi blok zincirleri ile karşılaştırırlar ve hangi blok zinciri daha büyük ise, yani hangisinde daha fazla blok var ise o blok zinciri üzerinden işlemlere devam edilir.

6.3. Yapılan Testler ve Sonuçları

Geliştirilen simülasyon uygulamasında SHA256 kriptografik özet algoritmasından daha karmaşık olan ve özet değerini bulması daha fazla zaman alan SHA512 özet algoritması kullanılmıştır. Test işlemleri aşamasında hem SHA256 kriptografik özet algoritması ile hem de SHA512 kriptografik özet algoritması ayrı ayrı ayarlanarak aynı transfer işlemine sahip blokların 5 farklı hedef değeri değiştirilmek sureti ile özet değerini bulma süreleri hesaplanmıştır. Test işlemleri iki aşamada yapılmıştır. Birinci aşamada nonce değeri ritmik olarak artırılmış ikinci aşamada ise nonce değerleri rasgele seçilmiştir.

6.3.1. Ritmik Artan Nonce Değeri İle Yapılan Test ve Sonuçları

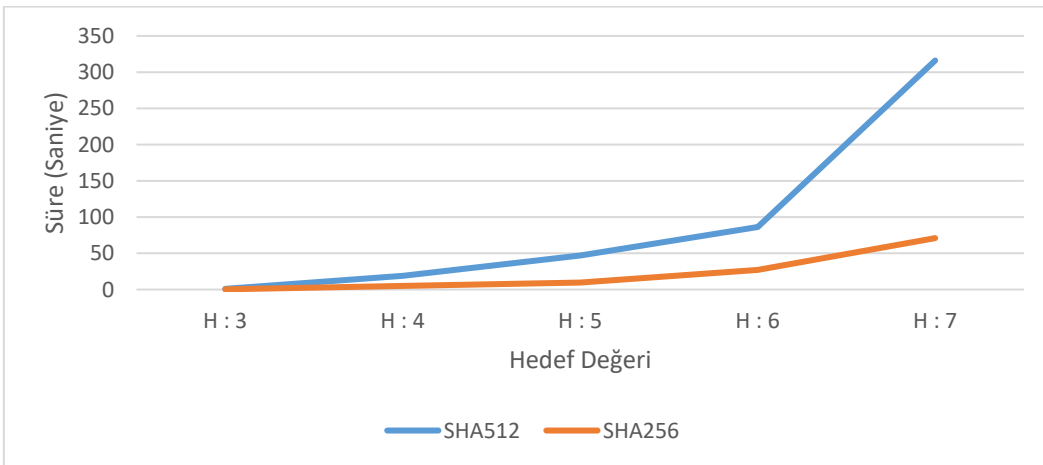
Nonce değeri ritmik olarak artırılarak yapılan testlerde ilk olarak iki algoritma için de 5 farklı hedef değeri için birer hesaplama yapılmış fakat zaman zaman algoritmaların tesadüfi olarak uygun nonce değerini erken bulmalarından ötürü net bir ölçüm yapılamamıştır. Bu

durum üzerine farklı 5 hedef değeri için aynı transfer işlemine sahip bloğun özet değeri için iki algoritma ile ayrı ayrı 100 defa hesaplama yapılarak daha net sonuçlar elde edilmiştir. Çizelge 6.1’ de farklı 5 hedef değeri için aynı transfer işlemine sahip bloğun özet değerinin iki algoritma ile ayrı ayrı 100 defa hesaplanması için geçen süreler saniye cinsinden verilmiştir.

Çizelge 6.1. Bir bloğun 100 defa özet değerinin hesaplanma süreleri (Ritmik Nonce)

Özet Algoritması	Hedef Değerleri				
	3	4	5	6	7
SHA512	0,98sn	18,98sn	46,92sn	86,21sn	316,11sn
SHA256	0,32sn	5,12sn	9,75sn	27,09sn	70,97sn

Çizelge 6.1 incelendiğinde SHA512 kriptografik özet algoritması ile gerçekleştirilen özet alma işlemlerinin, bütün hedef değerlerinde SHA256 kriptografik özet algoritmasıyla gerçekleştirilen özet alma işlemlerinden daha uzun sürdüğü görülmektedir. Şekil 6.7’ deki grafikte ise Çizelge 6.1’ deki değerler grafik olarak verilmiştir. Grafik incelendiğinde ise hedef değeri arttıkça SHA512 kriptografik özet algoritmasının işlem süresinin SHA256 özet algoritmasına göre daha çok arttığı görülmektedir. Bu durum gelişen teknoloji ile ortaya çıkabilecek olan erken blok oluşturma sorununa SHA512 kriptografik özet algoritmasını kullanmanın geçici de olsa bir çözüm olabileceğini göstermektedir.



Şekil 6.7. SHA256 ve SHA512 özet algoritmalarının aynı girdiyi 100 defa hesaplama süreleri grafiği (Ritmik Nonce)

6.3.2. Rasgele Değişen Nonce Değeri İle Yapılan Test ve Sonuçları

Nonce değeri rasgele seçilerek yapılan testlerde, aynı blok kullanılmasına rağmen, ritmik artan nonce değeri ile yapılan testlere kıyasla zaman zaman daha kısa zaman da zaman zaman daha uzun zaman sonuca ulaşılmıştır. Bu test çalışmasında da yine iki algoritma ile ayrı ayrı 100 defa hesaplama yapılarak daha net sonuçlar elde edilmeye çalışılmıştır. Çizelge 6.2’ de rasgele nonce değeri kullanılarak bir bloğun 100 defa özet değerinin her iki algoritma ile de hesaplanma süreleri görülmektedir. Rasgele, ve ritmik artan nonce değerleri ile elde edilen sonuçların birbirine yakın olması rasgele nonce değeri seçmenin düşük güçlü makinelerle madencilik imkanı tanımayacağı gibi görünse de ritmik artan nonce değeri ile yapılan hesaplamada sıfır olan ihtimal rasgele nonce değeri ile yapılan hesaplamada daha yüksektir. Ritmik artan nonce değeri seçildiği zaman güçlü ve zayıf makineler aynı değerden başlayarak işlem yapmaktadırlar. Dolayısı ile gerekli olan nonce değeri kaç olursa olsun işlem gücü yüksek makineler bu değere daha çabuk ulaşacaklardır. Rasgele nonce değeri seçilerek yapılan işlemde ise düşük güçlü makinenin de uygun nonce ve özet değerini bulma ihtimali vardır. Bu durum ağdaki katılımcı sayısını arttırarak sistemi daha güvenli hale getirecektir. Ancak Bitcoin ağında yapılmak istenilen bir güncellemeyi katılımcıların çoğunun kabul etmesi gerekir. Yüksek güçlü düğümler tarafından böyle bir güncelleme kabul edilmeyeceği için hayata geçmesi pek mümkün görünmemektedir.

Çizelge 6.2. Bir bloğun 100 defa özet değerinin hesaplanma süreleri (Rasgele Nonce)

Özet Algoritması	Hedef Değerleri				
	3	4	5	6	7
SHA512	1,15sn	18,15sn	43,15sn	94,44sn	325,13
SHA256	0,29sn	5.11sn	11.28sn	28,78	81.09sn

6.4. Sonuç

Geliştirilen simülasyon programı ile Bitcoin sisteminde SHA256 kriptografik özet algoritması yerine SHA512 kriptografik özet algoritmasının kullanılabilmesi ve nonce değerinin ritmik artmak yerine rasgele seçilerek işlem yapılabileceği gösterilmiştir. Aynı transfer işlemlerine sahip blok üzerinde yapılan test çalışmalarının sonuçlarına göre SHA512 kriptografik özet algoritması SHA256 kriptografik özet algoritmasına göre daha uzun zamanda

sonuç üretmektedir ve istenilen özet değerinin üst sınırı olan hedef değeri küçüldükçe aynı işlemi yapmak için geçen süreler arasındaki makas açılmaktadır. Bu durum teknoloji ile birlikte artan işlemci gücüne karşı Bitcoin sisteminde önlem olarak SHA512 kriptografik özet algoritmasının kullanılabilceğini göstermektedir.

İş kanıtı temeline dayalı çalışan bütün blok zinciri uygulamaları için gelecekte öngörülen en büyük tehlike, mevcut sistemlerden kat ve kat hızlı çalışan bilgisayarların icat edilmesidir. Quantum Bilgisayarı olarak bilinen ve çok hızlı çalışabilecek bu bilgisayarlar üzerine çalışma yapan bazı firmalar zaman zaman duyulsa da konu hakkında net bir bilgi yoktur. Örneğin Bitcoin ağına böylesine hızlı çalışan bir düğüm eklenirse muhtemelen bütün blokları o düğüm oluşturacak ve ağdaki diğer katılımcıların ağda kalmalarının bir anlamı kalmayacaktır. Benzer tehlike saldırı durumları içinde geçerlidir. Ayrıca Bitcoin sisteminin her 10dk'da bir blok oluşturacak şekilde programlandığı gibi bütün blok zinciri sistemler kendilerine bir blok oluşturma zamanı belirlemişlerdir. Yüksek hızlı düğümlerin sistemlere dahil olması durumunda bu süreler de istenmeyen seviyelere düşecektir bu teknolojiye uygun yazılımlar geliştirme ihtiyacı hasıl olacaktır.

KAYNAKLAR

- A. M. Antonopoulos. (2016). Mastering Bitcoin. In *Journal of World Trade* (Vol. 50, Issue 4).
<https://www.bitcoinbook.info/>
- Akleyek, S., Yıldırım, H. M., & Tok, Z. Y. (2011). Kriptoloji ve Uygulama Alanları: Açık Anahtar Altyapısı ve Kayıtlı Elektronik Posta. *Akademik Bilişim 2011*, 713–718.
- Aşan, H., & Avunduk, H. (2018). Blok Zinciri (Blockchain) Teknolojisi ve İşletme Uygulamaları: Genel Bir Değerlendirme. *Dokuz Eylül Üniversitesi İktisadi ve İdari Bilimler Dergisi*, 33(1). <https://doi.org/10.24988/deuiibf.2018331746>
- B. A. Forouzan. (2007). Cryptography and Network Security. In *Cryptography and Network Security*.
- Bhattacharya, S. (2019). Cryptology and Information Security-Past, Present, and Future Role in Society. *International Journal on Cryptography and Information Security (IJCIS)*, 9(1), 13–36. <https://doi.org/10.5121/ijcis.2019.9202>
- Chuen, D. L. K. (2015). Handbook of Digital Currency: Bitcoin, Innovation, Financial Instruments, and Big Data. In *Handbook of Digital Currency: Bitcoin, Innovation, Financial Instruments, and Big Data*. <https://doi.org/10.1016/C2014-0-01905-3>
- Courtois, N. T., Grajek, M., & Naik, R. (2014). Optimizing SHA256 in Bitcoin Mining. *Communications in Computer and Information Science*, 448 CCIS, 131–144.
https://doi.org/10.1007/978-3-662-44893-9_12
- El Mounni, S., Fettach, M., & Tragha, A. (2019). High Throughput Implementation of SHA3 Hash Algorithm on Field Programmable Gate Array (FPGA). *Microelectronics Journal*, 93(October 2018). <https://doi.org/10.1016/j.mejo.2019.104615>
- Fidan, M., Dilek, S., & Esev, A. (2019). Dünden Bugüne Paranın Tarihi ve Türkiye’de Kağıt Para Kullanımı. *Sosyal Bilimler Dergisi*, 9(18), 141–162.
<https://doi.org/10.31834/kilissbd.613107>
- Kodaz, H., & Bostalı, F. M. (2010). Simetrik ve Asimetrik Şifreleme Algoritmalarının Karşılaştırılması. *Selçuk-Teknik Dergisi*, 9(1), 10–23.
- Manap, C., & Apohan, A. M. (n.d.). Özet Fonksiyonlarındaki Zayıflıklar ve Elektronik İmzalara Etkisi. *Kamu Sertifikasyon Merkezi*, 1–3.
http://www.kamusm.gov.tr/dosyalar/makaleler/Ozet_Fonksiyonlarindaki_Zayifliklar_Ve

- Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1997). Handbook of Applied Cryptography. *Choice Reviews Online*, 34(08), 34-4512-34-4512.
<https://doi.org/10.5860/CHOICE.34-4512>
- Mukhopadhyay, U., Skjellum, A., Hambolu, O., Oakley, J., Yu, L., & Brooks, R. (2016). A Brief Survey of Cryptocurrency Systems. *2016 14th Annual Conference on Privacy, Security and Trust, PST 2016*. <https://doi.org/10.1109/PST.2016.7906988>
- Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*.
- Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. *Business and Information Systems Engineering*, 59(3), 183–187. <https://doi.org/10.1007/s12599-017-0467-3>
- Obaid, Z., Sabonchi, A., & Akay, B. (2016). Klasik Kriptoloji Yöntemlerinin Karşılaştırılması. *Engineering Sciences*, 7231(June), 1–61.
<https://doi.org/10.12739/NWSA.2016.11.4.1A03>
- Özbaş, M. Y. (2019). Elektronik Para Ve Sanal Para: Bitcoin Geleceğin Pa Birimi Olabilir Mi? *İşletme Ekonomi ve Yönetim Araştırmaları Dergisi*, 2(1), 85–104.
<https://doi.org/10.33416/baybem.434712>
- Pamula, D., & Ziebinski, A. (2009). Hardware implementation of the MD5 algorithm. In *IFAC Proceedings Volumes (Vol. 42, Issue 1)*. IFAC. <https://doi.org/10.3182/20090210-3-cz-4002.00012>
- Penard, W., & Werkhoven, T. van. (2008). On the secure hash algorithm family. *Cryptography in Context*, 1–18.
<https://www.staff.science.uu.nl/~tel00101/liter/Books/CrypCont.pdf>
- Pierro, M. D. (2017). What Is the Blockchain? *Computing in Science and Engineering*, 19(5), 92–95. <https://doi.org/10.1109/MCSE.2017.3421554>
- Pollard, J. M. (1978). Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32(143), 918. <https://doi.org/10.2307/2006496>
- Preneel, B. (2010). The First 30 Years of Cryptographic Hash Functions And The NIST SHA-3 Competition. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5985 LNCS, 1–14.

https://doi.org/10.1007/978-3-642-11925-5_1

- SAHIN, F. (2015). Modern Blok Sifreleme Algoritmaları. *Istanbul Aydın Üniversitesi Dergisi*, 7(26), 23–40. <https://doi.org/10.17932/iau.iaud.m.13091352.2015.7/26.23-40>
- Sakallı, M. T., Buluş, E., Şahin, A., & Büyüksaraçoğlu, F. (2007). Akış Şifrelerinde Tasarım Teknikleri ve Güç İncelemesi. *Akademik Bilişim Konferansı Bildirileri*, 635–642.
- Şeker, S. E. (2008). *MD5 (Message Digest , Mesaj Özet)*.
<http://bilgisayarkavramlari.com/2008/04/30/md5-message-digest-mesaj-ozet/>
- SHA-2. (2020). Wikipedia. <https://tr.wikipedia.org/wiki/SHA-2>
- Tanrıverdi, M., Uysal, M., & Üstündağ, M. T. (2019). Blokzinciri Teknolojisi Nedir ? Ne Değildir ? : Alanyazın İncelemesi. *Bilişim Teknolojileri Dergisi*, 203–217.
<https://doi.org/10.17671/gazibtd.547122>
- Tian, F. (2016). An Agri-Food Supply Chain Traceability System For China Based on RFID & Blockchain Technology. *2016 13th International Conference on Service Systems and Service Management, ICSSSM 2016*. <https://doi.org/10.1109/ICSSSM.2016.7538424>
- Tübitak. (n.d.). *Blokzincir Teknolojileri*. <https://blokzincir.bilgem.tubitak.gov.tr/blok-zincir.html>
- Uluyol, Ç., & Ünal, G. (2020). Blok zinciri teknolojisi. *Bilişim Teknolojileri Dergisi*, 167–175. <https://doi.org/10.17671/gazibtd.516990>
- Yerlikaya, T., Buluş, E., & Arda, D. (2005). Eliptik Eğri Şifreleme Algoritması Kullanan Dijital İmza Uygulaması. *Ağ ve Bilgi Güvenliği Ulusal Sempozyumu-ABG2005*, 124–128.
- Yerlikaya, T., Buluş, E., & Buluş, N. (2006). Kripto Algoritmalarının Gelişimi Ve Önemi. *Akademik Bilişim Konferansları*. <http://ab.org.tr/ab06/bildiri/132.pdf>
- Yerlikaya, T., Gençoğlu, H., Emir, M. K., Çankaya, M., & Buluş, E. (2011). RSA Şifreleme Algoritması ve Aritmetik Modül Uygulaması. *Istanbul Aydın Üniversitesi Dergisi*, 3, 96–104.
- Zheng, Z., Dai, H., Chen, X., & Wang, H. (2018). Blockchain Challenges And Opportunities. *IEEE International Symposium on High Performance Distributed Computing, Proceedings*, 14(4), 352–375.
- Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of Blockchain

Technology: Architecture, Consensus, and Future Trends. *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017.*

<https://doi.org/10.1109/BigDataCongress.2017.85>

