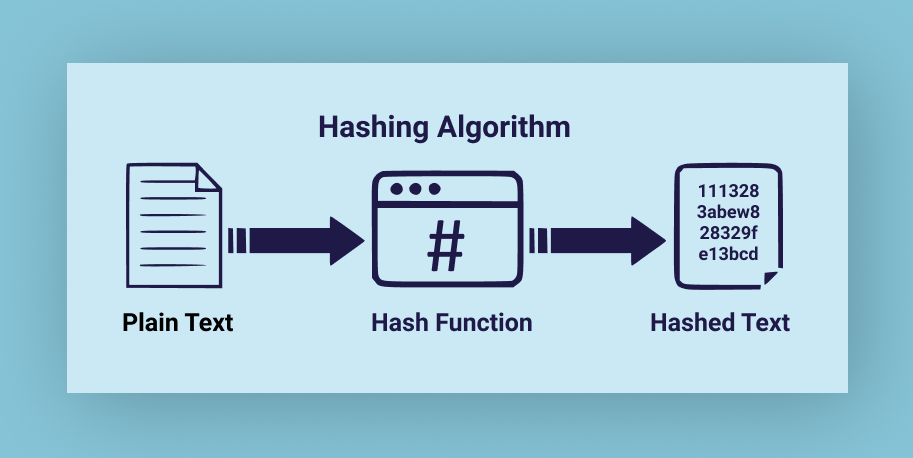
**KARMA (HASH) FONKSİYONLARI**

Hash fonksiyonu verileri sabit boyutlara sıkıştıran matematiksel fonksiyondur. Günümüzde verilerin doğruluğunu kanıtlamak, benzersiz bir kimlik oluşturmak ve veri bütünlüğü gibi önemli görevleri bulunan fonksiyonlardır. Özellikle bilgi güvenliği için önemli temeller oluştururken pek çok alanda kullanıldığını görebiliriz.

**Hash Nedir? Özellikleri Nelerdir?**

Hash fonksiyonunu oluşturan hash, belirli bir veri parçasının benzersiz bir şekilde şifreli değerlere dönüştüren sabit uzunlukta matematiksel bir işlem türüdür. Elde edilen sonuç hash değeri olarak bilgi güvenliğinin pek çok alanında kullanılmaktadır. Verilen girdi için hesaplanan hash değerleri verinin büyüklüğünden bağımsız hep aynı uzunlukta olacaktır. Bu sayede verilerin daha kolay işlenmesi ve hash değerlerinin daha kolay karşılaştırılması sağlanır. Aynı veri her zaman aynı hash değerini verir ve girdi biraz değişse bile hash değerinin değiştiği görülebilir.



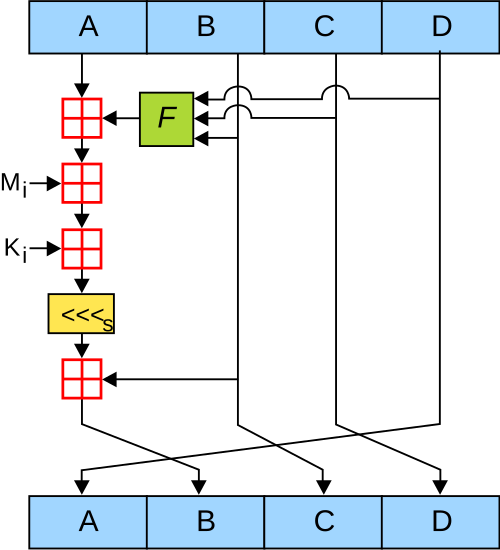
* Tek yönlülük: tek yönlü fonksiyon olması sebebiyle bir veri bir hash ile işaret edilebilirken, bir hash üzerinden gerçek veriye ulaşılamaz.
* Deterministik olma: Aynı veri her zaman aynı hash değerinin üretir. Özellikle parola doğrulamalarından örnek vermek gerekirse, parolanın geçerli olup olmadığını karşılaştırmak için hash değerinin deterministik özelliği devreye girer. Aynı hash değerini gören sistem parolanın doğru olduğuna karar verecektir.
* Çarpışma direnci: İki farklı veririn aynı hash sonucunu vermemesi sağlanmalıdır. Aksi halde verilerin doğruluğu ve bütünlüğü kontrol edilemez. Aynı hash değerinin oluşmasına “çakışma (collision)” denir. İyi bir hash fonksiyonunun bunu net bir şekilde önlemesi gerekir.
* Ön görülemezlik: Hash fonksiyonları tek yönlü olması sebebiyle bir hash değerinden veriyi ortaya çıkarmak imkansız olmalıdır. Veri güvenliği için önemlidir.
* İkinci görüntü direnci: Bir veriye karşılık gelen hash değerinin, aynı şekilde başka bir veriye de karşılık gelmesi önlenmelidir.
* Şifre saklama, veri doğrulama, blockchain ve kripto teknolojileri, dijital imza ve bilgi güvenliğinin söz konusu olduğu her yerde hash kullanımını görmek mümkündür.

Günümüze gelene kadar farklı hash fonksiyonları kurulmuş ve hash fonksiyonlarının temel özelliklerini sağlayabilmek için farklı algoritmalar kullanılmıştır. Bu fonksiyonlar:

**MD5 (Message Diggest 5)**

1991 yılında Ron Rivest tarafından geliştirilen bu algoritma özetleme algoritmalarından biridir. Verileri sabit boyutlu karakter dizisine dönüştürerek dosya ve veri bütünlüğünün doğrulanması amacıyla kullanılır. MD5 verilen girdileri 128 bit uzunluğunda bir özete dönüştürür. Bu özet benzersiz olması sebebiyle küçük değişiklikler bile bu özeti değiştireceğinden veri doğrulaması yapmayı kolaylaştırmıştır. Bu algoritmanın çalışma mantığı:

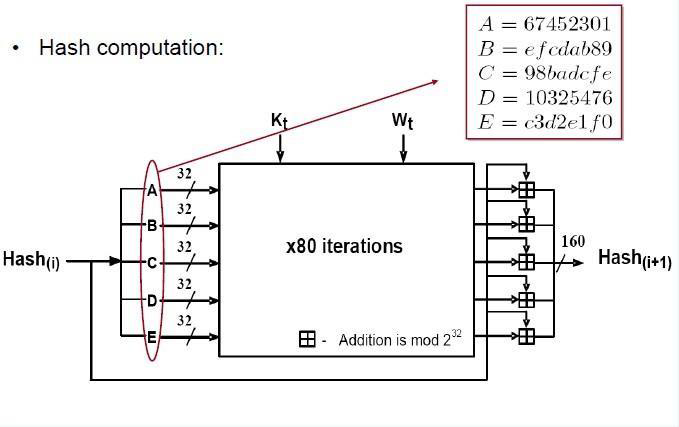
* Girdi verisi alınır ve bu veri 512 bitlik bloklara bölünür.
* Blok veri bitleri eksik kalan kısımlar için padding uygulanır ve blok uzunlukları eşlenir.
* 4 adet başlangıç için 32 bitlik A,B,C,D sabitleri ile işlemler başlar.
* 512 bitlik bloklar içinde her blok 16 parçaya ayrılarak 32 bitlik bloklar oluşturulur.
* 4 ana fonksiyon ile mantıksal bit işlemleri gerçekletirilir ve 64 kez dönerek A,B,C,D değerleri güncellenir.
* Her dönme esnasında veri parçaları belirli işlemlerden geçer ve çıkan sonuç A,B,C,D ile birleştirilir.
* En son A,B,C,D değerleri de birbiriyle birleşerek 128 bitlik bir hash değeri oluştururlar.
* Örn: MD5("Merhaba") = 49c00ca7b7cd504af634d4b0e7d70a3b



Uzun bir süre kullanılsa da ciddi zafiyetler içermesi sebebiyle güvenlik uygulamaları için pek de tercih edilmemeye başlanmıştır. Özellikle çarpışma(collision) saldırılarına karşı zayıf kalması bu algoritmaya duyulan güveni kırmıştır çünkü bu saldırılar farklı girdilerin aynı MD5 çıktısını ürettiği zafiyetler olması sebebiyle kimlik doğrulama sistemlerini yanıltabilir ve veri bütünlüğünü bozabilir. Bu da bir hash fonksiyonu için istenmeyecek bir durumdur.

**SHA-1 (Secure Hash Algorithm)**

Ron Rivest tarafından geliştirilen algoritma bir kriptografik özet fonksiyonudur. 160 bitlik bir özet değer üretir ve MD5’e benzer bir yapısı olsa da, MD5’e göre daha sadedir. Oluşturulan özet değer 40 basamaklı onaltılık sayı olarak işlenir. Bir döneme kadar dijital imza, sertifika, dosya doğrulama işlemlerinde kullanılıyordu. 2005’ten sonra bazı güvenlik açıklarına sahip olması sebebiyle güvenilirliğini yitirmiştir. Çalışma adımlarına değinecek olursak:



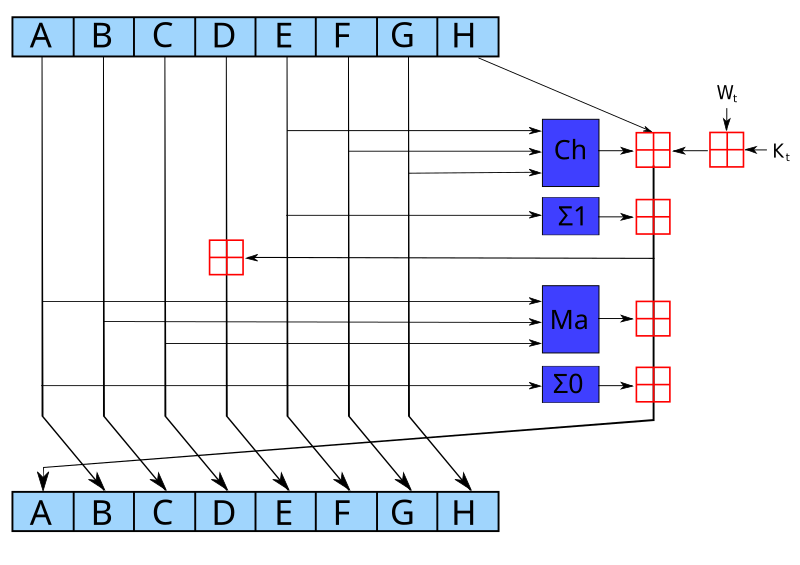
* “MMM” gibi bir giriş mesajıyla işlemler başlatılır.
* Daha sonraki işlemlerde 512 bitlik eşit parçalara bölüneceği için doldurmalar yapılır.
* Bölünme işleminden sonra tekrar 32 bitlik 16 kelimeye bölünür.
* Bu bölünmeler sonucu bazı genişletmeler ile 80 adet 32 bitlik blok elde edilir.
* İşleme girecek başlangıç değerleri ayarlanır (A,B,C,D,E)
* Ana döngü, mesaj kelimeleri ve 4 farklı tur sabitiyle 4 farklı gruba ayrılmış aşamalar; 80 turluk bir hesaplama yapılır.
* Her tur sonucu elde edilen çıktılar başlangıç değerlerine eklenerek son mesaj özeti oluşturularak aşamalar tamamlanır.

SHA-1 algoritmasındaki güvenlik açıklarının başında çakışma (collison) oluşturulabilme üretmesi gelmektedir. Bu açık yüzünden iki farklı veri aynı hash değerini üretebiliyor. Bu da verinin doğruluğunun kanıtlanamamasına sebep olur. Bu açık 2017’de SHAttered çakışma saldırısında kullanılmış ve başarılı olunmuştur ve iki farklı pdf için aynı hash üretilmiştir.

**SHA-2 Algoritması**

ABD tarafından Ulusal Güvenlik Ajansı (NSA)’nın tasarladığı 2001 yılında ortaya çıkan kriptografik özet fonksiyonları kümesidir. Çeşitli uzunluklarda olabilmesiyle birlikte en çok kullanılanı 256 bitlik olanıdır. Bu nedenle SHA-2 fonksiyonu “SHA-256”, “SHA-384”, “SHA-512” gibi kullanılan bit sayısında göre isimler alabilir. SHA-2 özellikle SHA-1’den sonra çakışma ve ön görüntü saldırıları gibi tehdit durumları için çok daha sağlam olacak şekilde tasarlanmıştır. Bu yüzden pek çok kurum SHA-2 ‘ye geçişi zorunlu kılmıştır. Bu sebeple güvenliğin sağlanması, sektörle uyum sağlamak ve kullanıcılara güven verebilmek için SHA-2 ‘ ye geçiş yapmak oldukça önemli bir hale gelmiştir. SHA-2 algoritması adımları:

* Giriş verisiyle başlanır. Örneğin “algoritma”
* Daha sonraki eşit bölme işlemleri için gerekli doldurma işlemleri yapılarak uzunluklar eşitlenir.
* Giriş verisi 512 bitlik bloklara bölünür.
* Her blok ayrı ayrı işlenecek şekilde algoritma 32 bit sayı ile başlatılır.
* Giriş verisinden oluşturulan 32 bitlik 64 tane kelime 64 tur içinde kullanılacak şekilde her bloğa:
* A,B,C,D,E,F,G,H isimli 8 değişken sabit K ve W[i] değerleri ile SHA-2 ye özgü bazı matematiksel işlemler her tur uygulanır. (Mor kutucuklarda gösterilen işlemler)

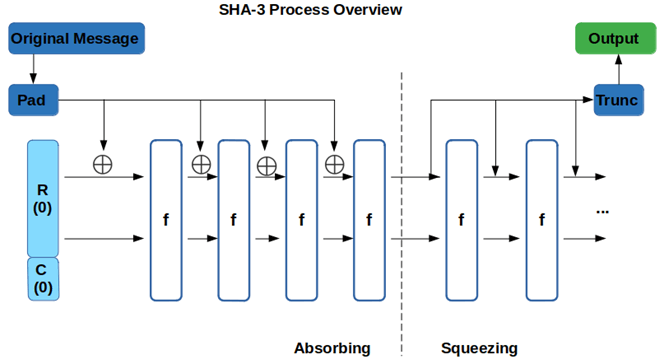


* Her tur sonunda sonuç değerleri A’ dan H’ ye kadar değişkenlere tanımlanır ve bir sonraki bloğun giriş verisi olarak kullanılır.
* En son A-H değişkenlere tanımlanan bütün değerler birleştirilerek 256 bit uzunluğunda bir hash değeri oluşturulur.

Bu şekilde işleyen bir algoritmada çakışma üretmenin zor olması küçük değişikliklerin özet için büyük fark yaratması ve bu özetten açık veriye dönüşün imkansız olması algoritmayı tercih edilebilir hale getirmiştir. Günümüzde de yaygın bir kullanıma sahiptir.

**SHA-3 Algoritması**

NIST tarafından yayımlanan Güvenli Karma Algoritması 3, 2015 yılında piyasaya sürülen özet fonksiyonu algoritmasıdır. SHA-2 yerine gerektiği durumlarda kullanılabilecek bir alternatif olarak ortaya çıkmıştır. Çalışma mantığında ikili veri dizilerinin belli bir boyutta mesaj özetine rastgele eşleşme sağlayarak güvenliği sağlamak yer alır. SHA-1 ve SHA-2 den tamamen farklı bir yapı kullanmasıyla, Keccak algoritması bu yapı için kullanılan algoritmadır. Sünger yapısı denen yapı üzerine kurulan algoritma, giriş verilerinin esnek biçimde işlenmesi ile sabit uzunlukta çıktı alınabilmesini sağlar.



* Resimde görüldüğü gibi önce giriş verileri padding işlemleri ile ufak doldurmalar yapılarak 1600 bitin katı olacak şekilde ayarlanır.
* R bitlik bloklara bölünerek ilk R biti XOR işlemine tabi tutulur ve Keccak-f permütasyonu uygulanarak absorbing işlemi yapılır. Keccak-f permütasyonu bitlerin yayılma, kaydırma konumlarının karıştırılması, doğrusal olmayan işlemler ve tur sabiti eklemeleri gibi bir dizi işlemin birleşimidir.
* Giriş verileri işlenmesi bittikten sonra Sıkma (squeeze) işlemi başlar. Bu işlemde oluşan durumun ilk r bitinden bir hash çıktısı üretilerek oluşan ondalıklı sayıların tam kısımlarının alınmasıyla (trunc) işlemler tamamlanır.

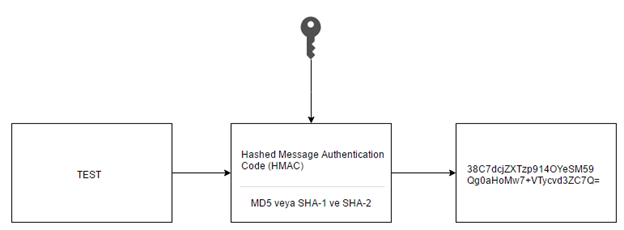
Diğer algoritmalardan farklı olarak giriş verisinin boyutunun sınırsız olabilmesi esnekliği ve aynı zamanda sabit uzunlukta çıktılar alınabilmesi yüksek güvenlikli bir yapı sunar.

* Çakışmalara dirençlidir
* Hash’den veriye ulaşmak nerdeyse imkansızdır.
* Alternatif bir güvenlik katmanına sahiptir.
* Kriptografik uygulamalarda kullanmak için oldukça uygundur.

**HMAC (Hash-based Message Authentiaction Code)**

Hash tabanlı mesaj kimlik doğrulama algoritması olarak ortaya çıkan HMAC, kriptografik özet fonksiyonu barındıran ve gizli bir anahtar kullanılarak yapılan bir mesaj doğrulama kodu türüdür. Diğer MAC türleri gibi mesaj içeriğini onaylamak ve veri bütünlüğü kontrol etmek için kullanılabilmektedir. Özet fonksiyonu kullanılarak HMAC hesaplaması yapılabilir bu da HMAC algoritmasının dayanıklılığını belirleyen faktörün kullandığı özet fonksiyonuna bağlı olmasını sağlar. Bunun yanında elde edilen özetin boyutu ve kullanılan anahtarın boyutu ve kalitesi de bu dayanaklılık için önemli faktörlerdir.

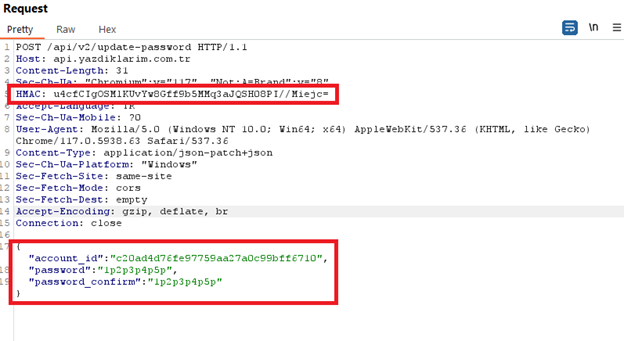
Özellikle oturum yönetimlerinde sıkça kullanıldığını görebilirsiniz.



* Gizli anahtar belirli uzunluklara kısaltılır veya uzatılır.
* Anahtar ve açık metin birlikte bir özet fonksiyonu tarafından hash’lenir.
* Alıcı da aynı gizli anahtar ile kendi HMAC hesaplamasını yaparak ve elde ettiği HMAC değerini karşılaştırarak aynı olma durumuyla verinin güvenli olduğunu çıkarım yapabilir.

Yani burada hash fonksiyonundan farkı, veriyle birlikte gizli anahtarı hashlemesi ve kimlik doğrulama imkanı sunmasıdır. Bu kimlik doğrulamayı gizli anahtar sayesinde yaptığı işlemlerle HMAC değerini bulması ve bu değeri karşılaştırması sonucu yapar. Çünkü sadece gönderici ve alıcı gizli anahtara sahip olduğu için mesajın farklı olması ya da anahtarın farklı olması HMAC değerlerinin tutarsız olmasına sebep olur.

* E-posta imzalama, JSON Web Token ( JWT) güvenliğinde, Uygulamalarda istemci-sunucu haberleşmelerinde, API doğrulama sistemlerinde, Webhook güvenliğinde HMAC kullanıldığı görülebilir.
* Aşağıdaki resimde görüldüğü gibi bir http isteğinin HMAC değeri, kullanıcının account id ‘si ile işleme girerek oluşmuş bir değerdir. Burada account id’nin değişmesi HMAC değerini de değiştireceğinden HMAC değerlerinin eşleşmemesi yüzünden http istekleri artık kabul edilmeyecektir. Eğer encryption key’e sahip olunursa yeni HMAC değeri üretilir ve http isteği değiştirilebilir. Bu yüzden anahtarın başkalarının eline geçmemesi gerekir.



**Zayıflıklar ve Kriptoanaliz**

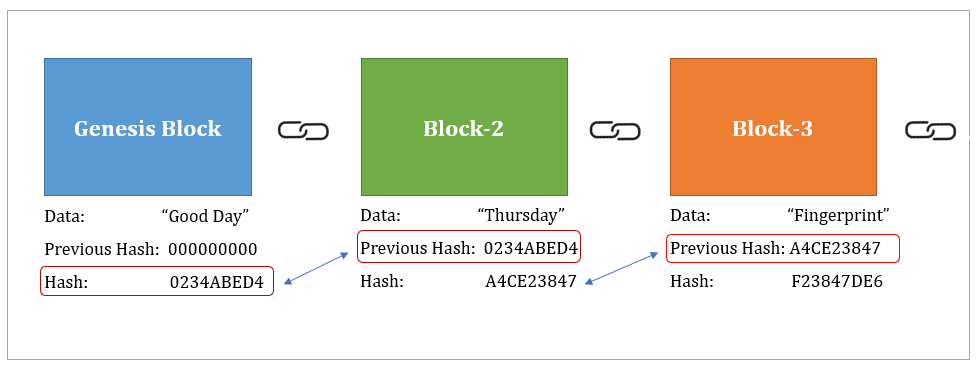
Hash fonksiyonları ilk kullanılan fonksiyondan bu yana güncellenerek günümüze gelmiş fonksiyonlardır. Bu güncellemelerin en büyük sebebi fonksiyonun içerdiği zafiyetlerdi. Bu zafiyetler:

* ***Çakışma (collision):***Aynı hash çıktısını veren iki farklı veri olması veri güvenliğini tehlikeye sokmasıyla birlikte fonksiyonun değiştirilmesine sebep olmuştur. Özellikle MD5 ve SHA-1 gibi algoritmalarda görülen bir zayıflıktır.
* ***Ön görüntü saldırısı:*** Hash değerine bakılarak bu değerinin hangi girdiden üretildiğini anlamaya çalışmak. Hash fonksiyonu zayıfsa mümkün olabilecek bir saldırıdır.
* ***Brute Force Saldırıları:***  Hash değerine karşılık gelen veri deneme yanılma yoluyla bulunabilir. Tahmin edilebilir verilerin hashlenmesi, kısa parolalar buna sebep olabilir.
* ***Rainbow Table Saldırıları:*** Belirli açık metinlerin önceden hesaplanan hash değerleriyle eşleşmesini bulunduran tablolarla gerçekleştirilebilir. Özellikle parola hashlerinin kırılmasını kolaylaştırdığı için hash fonksiyonları bu durumlara karşı savunmasızdırlar.
* ***Karma işlevi zayıflıkları:*** Bazı hash fonkiyonları algoritmanın yetersizliği yüzünden bazı saldırılara karşı bariz olarak savunmasızdır. MD5 buna bir örnektir. Bu yüzden güvenli kabul edilemeyen bir algoritmadır.
* ***Performans Sorunları:*** Kriptografik hash fonkisyonları yüksek hesaplama gücüne karşın bilgisayar performansının aynı güce sahip olmadığı durumlarda bu, işlemin yavaş çalışmasına sebep olabilmektedir. Özellikle büyük veri setlerinde fazlaca yaşanabilmektedir.

Bunun gibi zayıflıklar kriptoanaliz sürecinde farklı yollardan hash verilerinden gerçek verilerin çıkarımını yapabilecek sonuçlar üretir. Bu analiz süreci kötü amaçlı kişiler tarafından uygulanırsa veri güvenliği suistimal edilebilir. Bu sebeple hash fonksiyonlarının güncellenmesi, zayıflıkların giderilmesi veya yeni algoritmaların yerine konulması gibi önlemler bu süreçte önemli olacaktır.

***HASH FONKSİYONLARI ve BLOKZİNCİR İLİŞKİSİ***

Blockchain yani blok zincir teknolojisi, birbirine bağlı olan veri bloklarının oluşturduğu dağınık veri tabanı yapısıdır. Bu yapılardaki her blok içerisinde belli bir veri setini, bir önceki bloğun hash değerini ve kendi hash bilgilerini içermektedir. Bu nedenle zincirdeki ufak bir değişiklik bütün yapıyı etkileyeceğinden veri sahteciliği yapılmanın önüne net bir şekilde geçilebilmektedir. Verilerin güvenliği, şeffaflığı ve merkeziyetsiz bir şekilde saklanmasını bu yapı sayesinde sağlarız. Sadece kripto paralar değil daha pek çok alanda bu teknoloji kullanılmaktadır.



Hash yapısı ile blok zincir arasındaki bağ, blokzincirin oldukça güvenilir ve şeffaf yapıda olmasını sağlayan bir bağdır. Öyle ki zincir yapısındaki bloklarda saklanan veriler, hash değerlerinin sürekli kontrol edilmesiyle ve tam olarak değiştirilmediğinden emin olunmasıyla bu güvenilirliği sağlamaktadır.

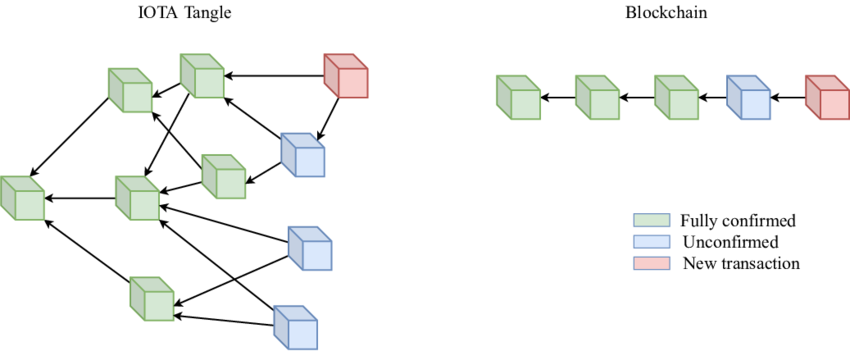
Blockchain teknolojisinde en çok kullanılan hash algoritması SHA-256’dır fakat blok zincirde aslında hash algoritmalarının bir kombinasyonunu görürürüz. Çünkü farklı hash fonskiyonları blokzincir yapısında yapıları gereği farklı işlevler için kullanılır. Örn:

* Blok verilerinin değiştirilip değiştirilmediğinin kontrolü için🡪 SHA-256, SHA-3
* Dosya veya veri kümesinin bütünlüğünü onaylamak için 🡪 MD5, RIDEMP-160, CRC32
* Yeni nesil olması ve daha güvenilir bir algoritma olarak SHA-256 yerine 🡪 Blake2,3

Gibi daha pek çok hash fonskiyonu farklı görevler için kullanılmaktadır.

**TANGLE (DAG)**

IOTA vakfı tarafından geliştirilen Tangle, DAG (Directed Acyclic Graph) yapısına dayanan yönlü ve döngü içermeyen grafik olarak tanımlanabilir. Blokzincir yapısındaki bazı sınırlamaları çözümlemek için verimli bir yoldur. Blok zincirdeki zincir yapısı gibi dümdüz ilerlememekle birlikte döngüsüz ve yönlendirmeler yapılmış bir grafik görürüz.



Blokzincir yapısındaki bazı dezavantajlar için iyi bir alternatif sağlamaktadır. Örneğin:

* Yüksek bir güvenlik sağlamak için blok zincir teknolojisinde her işlemin madenciler tarafından onaylanması ve doğrulanması gerekiyor. Bu işlem hızını düşüren bir durumdur. Tangle da ise insan müdahalesi gerekmeksizin birbiriyle iletişim kuran cihazlar kendi kendini yöneterek bu duruma bir çözüm getirmiştir. Tangle’da kullanılan DAG yapısı ile işlem yapılmak istendiğinde önceki iki işlemi doğrulanarak başlanır ve her yeni işlemde önceki iki işlem direkt olarak doğrulanıp onaylandığı için herhangi biri tarafından onaylanma ihtiyacını da ortadan kaldırılmış olur.
* Tangle’da veriler küçük birimler halinde taşındığı için son derece akıcı ve ve ölçeklenebilir bir kaydetme mekanizması vardır. Blokzincir ise ağ büyüdükçe daha az üretken ve yavaş çalışmaktadır. Ağın büyüklüğü söz konusu olduğunda ağa katılan katılımcı sayısının artması Tangle için daha çok verimlilik oluşturur. Bu açıdan yüksek işlem hacmi ve ağın büyüklüğünün arttığı durumlar için Tangle güzel bir alternatiftir. Blokzincir bu durumlarda çok daha fazla enerji ve işlemci gücü kullanır.
* Blok zincir yapısı daha yavaş çalışsa da Tangle’ a göre çok daha güvenliklidir. Tangle ağını çok daha az karmaşık saldırılarla ele geçirmek mümkündür.
* Tangle yapısı özellikle IOT için tasarlandığından küçük veri paketlerini hızlı ve ücretsiz bir şekilde bu cihazlara aktarabilir.
* Paralel işlem doğrulama yeteneği bulunan Tangle için aynı anda farklı kollarda işlemler yürütülebilir fakat blok zincirde bloklar sıra sıra zincire eklendiği için kuyruk oluşturma ve yavaş çalışmaya sebep olabilmektedir.
* Blok zincir yapısı merkeziyetsiz veri kaydetme özelliğine sahip olmasından dolayı veriler dünyanın dört bir yanından birbirine bağlı şekilde dağınıktır. Tangle yapısı da merkeziyetsiz bir yapı olarak görülse de IOTA ağı merkezi bir düğümle ara ara kilit taşı işlemler yayınlayarak bu kilit taşlarıyla ağdaki işlemlerinin geçerliliğini sağlar.

**Tangle Yapısı ve Hash Fonksiyonu Arasındaki Bağımlılıklar**

Her Tangle işleminde benzersiz bir kimlik ile kendini tanımlar. Bu kimlik de verilerin hash fonksiyonu tarafından özetlemesiyle elde edilen bir ID oluşturur. Blok zincir mimarisinde olduğu gibi işlemlerin ve verilerin değiştirilemez, benzersiz olmasının denetlenebilmesini bu hash fonksiyonu sağlar. Örneğin IOTA, SHA-3 hash fonksiyonu kullanır.

Tangle yapısında bir işlemin oluşturulabilmesi için iki önceki işlemin onaylanması gereklidir. Bu onaylama işleminin yapılabilmesi için önceki iki işlemde yer alan hash verileri referans alınarak bu onaylama işlemi yapılır. Bu sayede işlemler birbirinin hem kendi hash’i hem de başka hash’lere sahip olmasıyla blok zincirden farklı yapıda bir zincir bağı oluşur bu da güvenliğin sağlanması için oldukça işe yarar bir çözümdür.