Bitcoin Whitepaper (Türkçe Çeviri)

Bitcoin: Eşler Arası Elektronik Nakit Sistemi

Satoshi Nakamoto

satoshin@gmx.com

www.bitcoin.org

Orijinal Whitepaper:

Bitcoin: A Peer-to-Peer Electronic Cash System

Çeviren: Onur İlyas Tokay

contact@onurilyastokay.com.tr

www.onurilyastokay.com.tr

Özet. Çevrimiçi ödemelerin bir finansal kurumdan geçmeden doğrudan bir taraftan diğerine gönderilmesini sağlayan, elektronik paranın tamamen eşler arası (*peer-topeer*) versiyonu. Dijital imzalar, çözümün bir kısmını sağlar, ancak çift harcamayı önlemek için hâlâ güvenilir bir üçüncü taraf gerekiyorsa faydalar kaybolur. Çift harcama sorununa çözüm olarak eşler arası ağ kullanmayı önermekteyiz. Ağ, işlemleri sürekli bir hash tabanlı iş ispatı (*proof-of-work*) zincirine hash ederek zaman damgası vurur ve iş ispatı yeniden yapılmadan değiştirilemeyen bir kayıt oluşturur. En uzun zincir, yalnızca tanık olunan olayların sırasının kanıtı olarak hizmet etmez, aynı zamanda kendisinin en büyük işlemci gücüne sahip havuzdan geldiğinin de kanıtı olmaktadır. CPU gücünün çoğunluğu ağa saldırmak için işbirliği yapmayan düğümler (*node*) tarafından kontrol edildiği sürece, en uzun zinciri oluşturacak ve saldırganları geride bırakacaklardır. Ağın kendisi minimal bir yapıya ihtiyaç duyar. Mesajlar en iyi çaba temelinde yayınlanır, ve düğümler istedikleri zaman ağdan ayrılabilir ve yeniden katılabilir, en uzun iş kanıtı zincirini ayrıldıkları sürede neler olduğunun kanıtı olarak kabul eder.

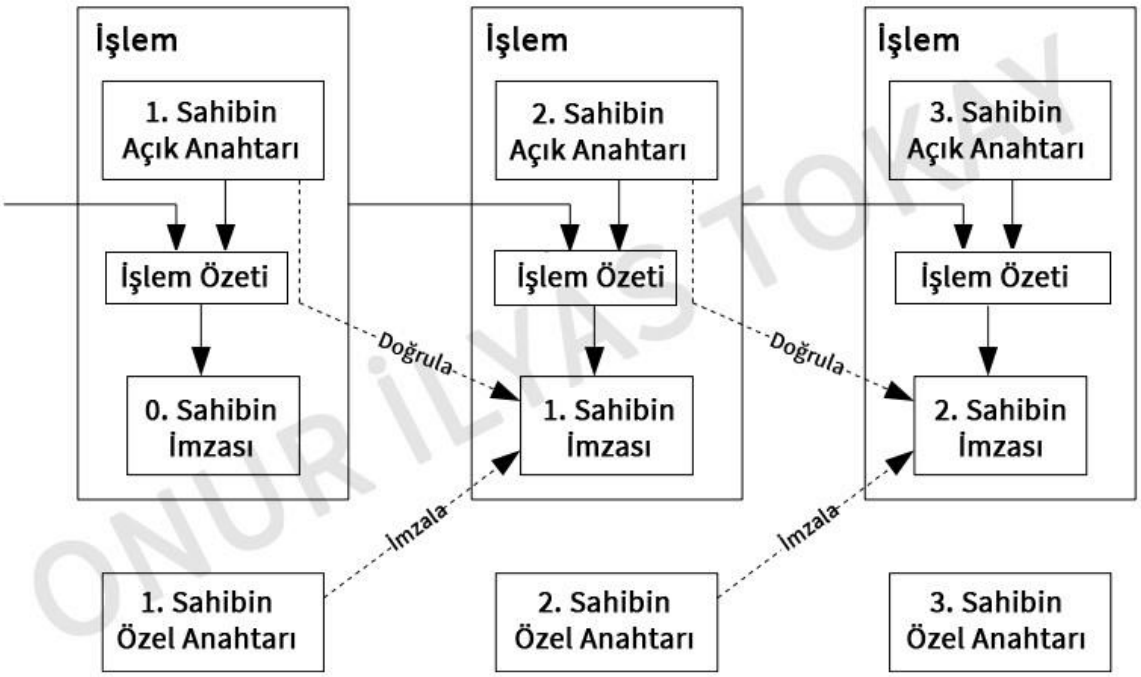
Giriş

İnternet üzerinden ticaret, elektronik ödemeleri işlemek için güvenilir üçüncü taraflar olarak hizmet veren finansal kurumlara neredeyse tamamen bağımlı hale geldi. Sistem çoğu işlem için yeterince iyi çalışsa da, yine de güven tabanlı modelin doğuştan gelen zayıflıklarından muzdariptir. Finansal kurumlar anlaşmazlıklara arabuluculuk etmekten kaçınamayacağı için geri dönüşü olmayan işlemler gerçekten mümkün değildir. Arabuluculuğun maliyeti işlem maliyetlerini artırır, asgari pratik işlem büyüklüğünü sınırlar ve küçük gündelik işlem olasılığını ortadan kaldırır; ayrıca geri dönüşü olmayan hizmetler için geri dönüşü olmayan ödemeler yapma kabiliyetinin kaybı daha geniş bir maliyete yol açar. Geri dönüş ihtimaliyle birlikte güven ihtiyacı da artar. Satıcılar müşterilerine karşı dikkatli olmalı, onlardan normalde ihtiyaç duyacakları bilgiden daha fazlası için 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 mevcut değildir.

İhtiyaç duyulan şey, güven yerine kriptografik kanıta dayalı, iki istekli tarafın, güvenilir bir üçüncü tarafa ihtiyaç duymadan doğrudan birbirleriyle işlem yapabilmesine olanak tanıyan bir elektronik ödeme sistemidir. Geri alınması hesaplama açısından pratik olmayan işlemler, satıcıları dolandırıcılıktan koruyacaktır ve alıcıları korumak için rutin emanet mekanizmaları (*routine escrow mechanisms*) kolayca uygulanabilecektir. Bu makalede, işlemlerin kronolojik sırasının hesaplamalı kanıtını oluşturmak için eşler arası dağıtılmış bir zaman damgası sunucusu kullanarak çift harcama sorununa bir çözüm öneriyoruz. Dürüst düğümler, işbirliği yapan herhangi bir saldırgan düğüm grubundan toplu olarak daha fazla CPU gücünü kontrol ettiği sürece sistem güvenlidir.

İşlemler

Elektronik parayı, dijital imzaların bir zinciri olarak tanımlıyoruz. Her bir sahip, bir önceki işlemin özetini (*hash*) ve bir sonraki sahibin açık anahtarını (*public key*) dijital olarak imzalayarak bir sonrakine aktarır, ve bunları paranın sonuna ekler.
Alacaklı, mülkiyet zincirini doğrulamak için imzaları doğrulayabilir.

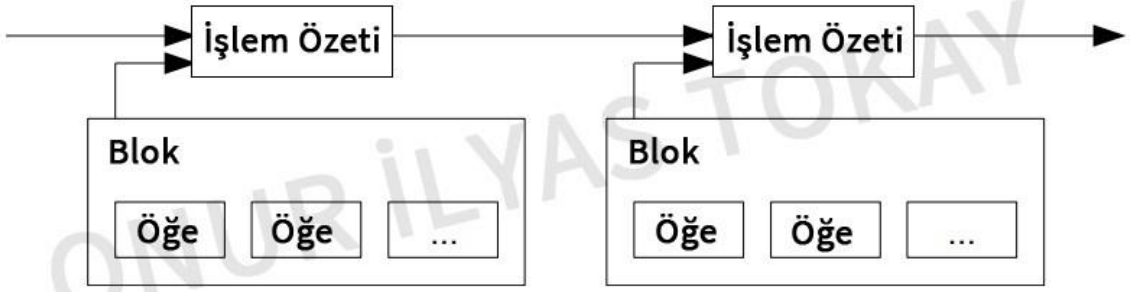


Elbette sorun, alacaklının, sahiplerden birinin parayı iki kez harcamadığını doğrulayamamasıdır. Yaygın çözüm, her işlemde çifte harcama olup olmadığını kontrol eden güvenilir bir merkezi otorite ya da darphanedir. Her işlemden sonra paranın yeni bir para basılması için darphaneye iade edilmesi gerekir ve yalnızca doğrudan darphaneden basılan madeni paraların iki kez harcanmadığına güvenilir. Bu çözümdeki problem, tüm para sisteminin kaderinin darphaneyi işleten şirkete bağlı olması ve her işlemin tıpkı bir banka gibi onlar üzerinden geçmesi zorunluluğudur.

Alıcının, önceki sahiplerinin daha önceki herhangi bir işlemi imzalamadığını bilmesini için bir yola ihtiyacımız var. Amaçlarımız doğrultusunda, en eski işlem geçerli olan işlemdir, bu nedenle daha sonraki çifte harcama girişimlerini önemsemiyoruz. Bir işlemin olmadığını doğrulamanın tek yolu, tüm işlemlerden haberdar olmaktır. Darphane tabanlı modelde, darphane tüm işlemlerden haberdardı ve hangisinin önce geleceğine karar veriyordu. Bunu güvenilir bir tarafa ihtiyaç duymadan başarmak için, işlemlerin kamuya duyurulması [1] ve katılımcıların işlemlerin alınma sırasına ilişkin tek bir geçmiş üzerinde anlaşmaları için bir sisteme ihtiyacımız var. Alıcının, her işlem sırasında düğümlerin çoğunluğunun ilk alınan para olduğunu kabul ettiğinin kanıtına ihtiyacı vardır.

Zaman Damgası (Timestamp) Sunucusu

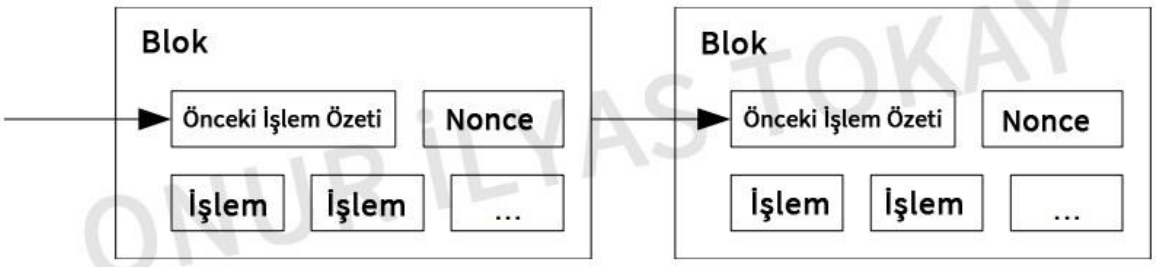
Önerdiğimiz çözüm bir zaman damgası sunucusu ile başlar. Bir zaman damgası sunucusu, zaman damgası vurulacak öğelerden oluşan bir bloğun özetini alarak ve bu özeti bir gazete ya da Usenet gönderisi [2-5] gibi geniş bir alanda yayınlayarak çalışır. Zaman damgası, verinin işlem özetine girebilmesi için o anda var olması gerektiğini kanıtlar. Her zaman damgası, kendisinden önceki zaman damgasını da içerecek şekilde bir zincir oluşturur ve eklenen her zaman damgası, kendinden öncekileri güçlendirir.



İş Kanıtı (Proof-of-Work)

Eşler arası dağıtılmış bir zaman damgası sunucusu uygulamak için, gazete veya Usenet gönderileri yerine Adam Back'in Hashcash'ine [6] benzer bir iş kanıtı sistemi kullanmamız gerekecektir. İş kanıtı, SHA-256'da olduğu gibi hash edildiğinde işlem özetinin bir dizi sıfır bit ile başladığı bir değerin taranmasını içerir. Ortalama olarak gereken iş, gereken sıfır bitlerinin sayısına göre eksponansiyeldir (üstel) ve tek bir hash yürütülerek doğrulanabilir.

Zaman damgası ağımız için, bloğun işlem özetine gerekli sıfır biti veren bir değer bulunana kadar bloktaki bir "nonce" değerini artırarak iş kanıtını uygularız. İş kanıtını yerine getirmek için CPU gücü harcandıktan sonra, blok işi yeniden yapmadan değiştirilemez. Sonraki bloklar kendisinden sonra zincirlendiği için, bloğu değiştirmek için yapılacak iş, ondan sonraki tüm blokların yeniden yapılmasını içerecektir.



İş kanıtı aynı zamanda çoğunluk karar alma sürecinde temsiliyetin belirlenmesi sorununu da çözmektedir. Eğer çoğunluk bir IP adresi bir oy esasına dayansaydı, çok sayıda IP tahsis edebilen herhangi biri tarafından altüst edilebilirdi. İş kanıtı esasen bir CPU bir oy demektir. Çoğunluk kararı, en fazla iş ispatı çabasının harcandığı en uzun zincir tarafından temsil edilir. CPU gücünün çoğunluğu dürüst düğümler tarafından kontrol ediliyorsa, dürüst zincir en hızlı şekilde büyüyecek ve rakip zincirleri geride bırakacaktır. Geçmiş bir bloğu değiştirmek için, saldırganın bloğun ve ondan sonraki tüm blokların iş kanıtını yeniden yapması ve ardından dürüst düğümlerin işini yakalaması ve aşması gerekir. daha yavaş bir saldırganın yetişme olasılığının, sonraki bloklar eklendikçe eksponansiyel olarak azaldığını göstereceğiz. Zaman içerisinde artan donanım hızını ve çalışan düğümlere yönelik değişen ilgiyi telafi etmek için iş kanıtı zorluğu, saat başına ortalama blok sayısını hedefleyen hareketli bir ortalama ile belirlenir. Çok hızlı oluşturulursa zorluk da artar.

Ağ

Ağ çalıştırma adımları şu şekildedir:

1. Yeni işlemler tüm düğümlere yayınlanır.
2. Her düğüm yeni işlemleri bir blokta toplar.
3. Her düğüm kendi bloğu için zor bir iş kanıtı bulmaya çalışır.
4. Bir düğüm iş kanıtını bulduğunda, bloğu tüm düğümlere yayınlar.
5. Düğümler, yalnızca geçerli işlemler içeriyorsa ve daha önce harcanmamışsa bloğu kabul eder.
6. Düğümler, zincirdeki bir sonraki bloğu oluşturmak için çalışarak bloğu kabul ettiklerini ifade ederler; bu sırada kabul edilen bloğun hash değerini bir önceki işlem özetine dönüştürürler.

Düğümler her zaman en uzun zinciri doğru zincir olarak kabul eder ve onu uzatmak için çalışmaya devam eder. Eğer iki düğüm bir sonraki bloğun farklı versiyonlarını aynı anda yayınlarsa, bazı düğümler birini veya diğerini önce alabilir. Bu durumda aldıkları ilk dal üzerinde çalışırlar ancak diğer dalı daha uzun olması ihtimaline karşı saklarlar. Bir sonraki iş kanıtı bulunduğunda ve bir dal daha uzun olduğunda eşitlik bozulacak; diğer dal üzerinde çalışan düğümler daha sonra daha uzun olana geçecektir.

Yeni işlem yayınlarının tüm düğümlere ulaşması gerekmez. Çok sayıda düğüme eriştikleri sürece, çok geçmeden bir bloğa gireceklerdir. Blok yayınları mesaj kayıplarına karşı toleranslıdır. Eğer bir düğüm bir bloğu almazsa, bir sonraki bloğu aldığında ve bir bloğu kaçırdığını fark ettiğinde bunu talep edecektir.

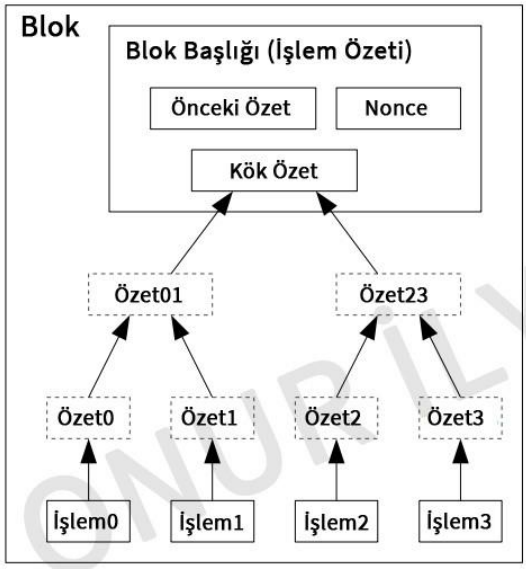
Teşvik

Kural olarak, bir bloktaki ilk işlem, bloğun yaratıcısına ait yeni bir paranın yaratıldığı özel bir işlemdir. Bu, düğümlerin ağı desteklemesi için bir teşvik sağlar ve merkezi bir otorite olmadığı için, başlangıçta paraları dolaşıma sokmanın bir yolunu sağlar. Sabit miktarda yeni paranın düzenli olarak piyasaya sürülmesi, altın madencilerinin dolaşıma altın eklemek için kaynak harcamasına benzer. Bizim durumumuzda, harcanan CPU zamanı ve elektriktir. Teşvik, işlem ücretleriyle de fonlanabilir. Bir işlemin çıktı değeri girdi değerinden azsa, aradaki fark, işlemi içeren bloğun teşvik değerine eklenen bir işlem ücretidir. Önceden belirlenmiş sayıda para dolaşıma girdiğinde, teşvik tamamen işlem ücretlerine dönüşebilir ve enflasyondan arındırılabilir.

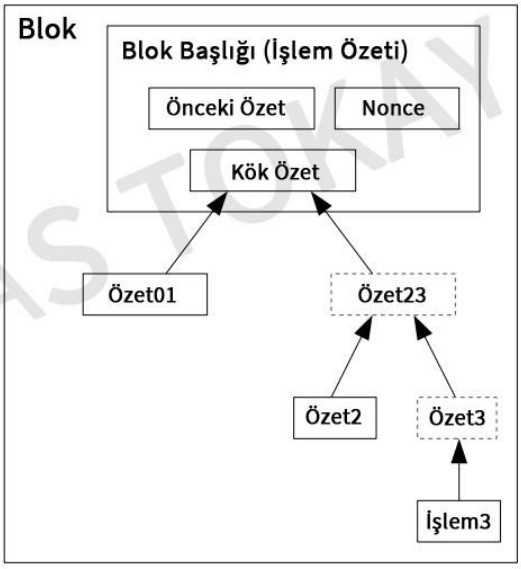
Teşvik, düğümleri dürüst kalmaya teşvik etmeye yardımcı olabilir. Eğer açgözlü bir saldırgan tüm dürüst düğümlerden daha fazla CPU gücü toplayabilirse, ödemelerini geri alarak insanları dolandırmak veya yeni paralar üretmek için kullanmak arasında seçim yapmak zorunda kalacaktır. Sistemi ve kendi servetinin geçerliliğini baltalamaktansa, herkesin toplamından daha fazla yeni para kazanmasını sağlayacak kurallara göre oynamayı daha kârlı bulmalıdır.

Disk Alanının Geri Kazanılması

Bir coin'deki en son işlem yeterince bloğun altına gömüldüğünde, disk alanından tasarruf etmek için ondan önce harcanan işlemler atılabilir. Bunu bloğun hash'ini bozmadan kolaylaştırmak için, işlemler bir Merkle Ağacında [7][2][5] hash edilir ve bloğun hash'ine yalnızca kök dahil edilir. Eski bloklar daha sonra ağacın dalları budanarak sıkıştırılabilir. İç özetlerin saklanmasına gerek yoktur.



Merkle Ağacında Özetlenmiş İşlemler

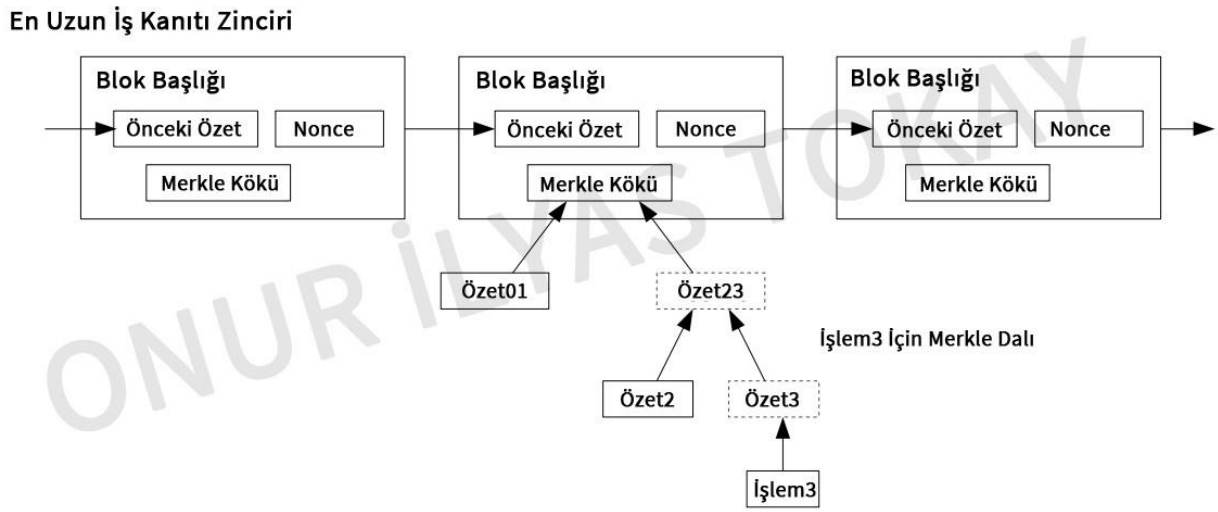


İşlem0-2 Bloktan Budandıktan Sonra

Hiçbir işlem içermeyen bir blok başlığı yaklaşık 80 bayt olacaktır. Her 10 dakikada bir blok üretildiğini varsayarsak, 80 bayt * 6 * 24 * 365 = yıllık 4.2MB yapar. Bilgisayar sistemlerinin 2008 yılı itibariyle genellikle 2GB RAM ile satıldığı ve Moore Yasası'nın yılda 1,2GB'lık bir büyüme öngördüğü düşünüldüğünde, blok başlıklarının bellekte tutulması gerekse bile depolama bir sorun teşkil etmeyecektir.

Basitleştirilmiş Ödeme Doğrulaması

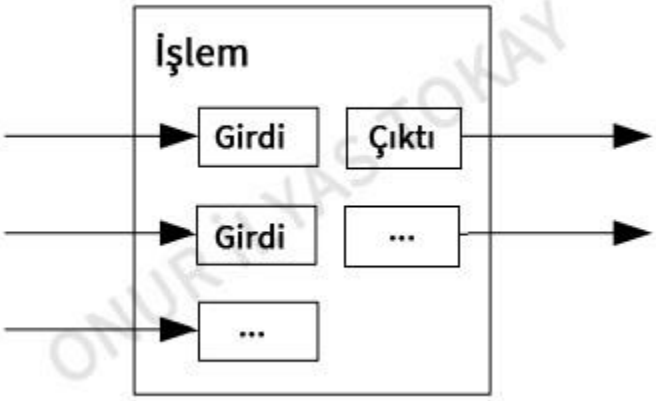
Tam bir ağ düğümü çalıştırmadan ödemeleri doğrulamak mümkündür. Bir kullanıcının yalnızca en uzun iş kanıtı zincirinin blok başlıklarının bir kopyasını saklaması gerekir; bunu en uzun zincire sahip olduğuna ikna olana kadar ağ düğümlerini sorgulayarak elde edebilir ve işlemi zaman damgalı bloğa bağlayan Merkle dalını elde edebilir. Kullanıcı, işlemi kendisi kontrol edemez, ancak bunu zincirdeki bir yere bağlayarak bir ağ düğümünün işlemi kabul ettiğini görebilir ve bundan sonra eklenen bloklar ağın işlemi kabul ettiğini doğrular.



Bu nedenle, dürüst düğümler ağı kontrol ettiği sürece doğrulama güvenilirdir, ama ağın bir saldırgan tarafından aşırı güçlendirilmesi durumunda daha savunmasızdır. Ağ düğümleri işlemleri kendileri doğrulayabilirken, saldırgan ağa hakim olmaya devam ettiği sürece, basitleştirilmiş yöntem, saldırganın uydurduğu işlemlerle kandırılabilir. Buna karşı korunmak için strateji, geçersiz bir blok tespit ettiklerinde ağ düğümlerinden gelen uyarıları kabul etmek ve tutarsızlığı doğrulamak için kullanıcının yazılımını tüm bloğu ve uyarılı işlemleri indirmeye yönlendirmek olabilir. Sıklıkla ödeme alan şirketler muhtemelen daha bağımsız güvenlik ve daha hızlı doğrulama için kendi düğümlerini çalıştırmak isteyeceklerdir.

Değerleri Birleştirme ve Bölme

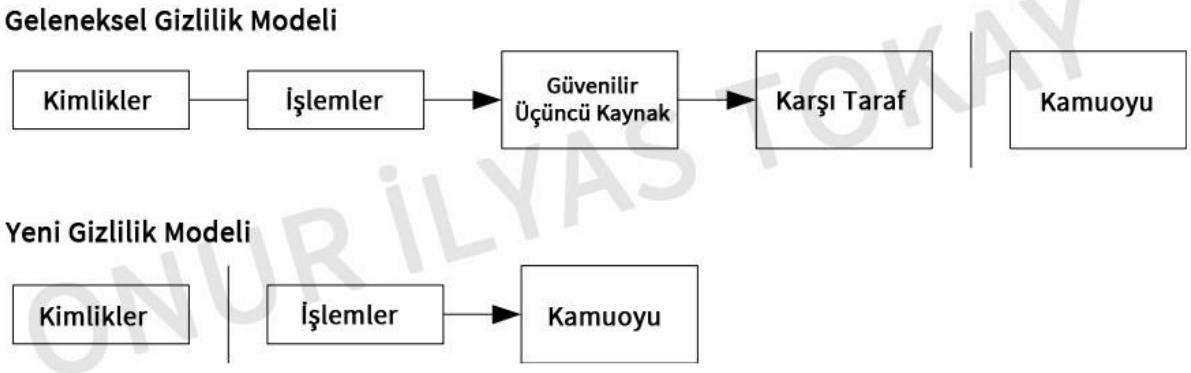
Paraları ayrı ayrı ele almak mümkün olsa da, bir transferdeki her sent için ayrı bir işlem yapmak hantal olacaktır. Değerin bölünmesine ve birleştirilmesine olanak vermek için işlemler birden fazla girdi ve çıktı içerir. Normalde ya daha büyük bir önceki işlemden tek bir girdi ya da daha küçük miktarları birleştiren birden fazla girdi ve en fazla iki çıktı olacaktır: biri ödeme için, diğeri ise -değişiklik varsa- para üstünü göndericiye geri döndürmek içindir.



Burada dikkat edilmesi gereken husus, bir işlemin birden fazla işleme bağlı olması ve bu işlemlerin de çok daha fazla işleme bağlı olması durumu olan fan-out'un burada problem değildir. Bir işlemin geçmişinin tam ve bağımsız bir kopyasını çıkarmaya hiçbir zaman gerek yoktur.

Gizlilik

Geleneksel bankacılık modeli, bilgiye erişimi ilgili taraflarla ve güvenilen üçüncü tarafla sınırlandırarak bir gizlilik düzeyine ulaşır. Tüm işlemlerin kamuya duyurulması gerekliliği bu yöntemi engeller, ancak başka bir yerdeki bilgi akışını keserek gizlilik korunabilir: genel anahtarları anonim tutarak. Kamuoyu, birinin başkasına bir miktar gönderdiğini görebilir, ama işlemi herhangi biriyle ilişkilendiren bir bilgi yoktur. Bu, münferit işlemlerin zamanının ve büyüklüğünün, yani "bandın" (*tape*) kamuya açıklandığı, ancak tarafların kim olduğunu söylemeden borsalar tarafından yayınlanan bilgi düzeyine benzer.



Ek bir güvenlik duvarı olarak, her bir işlemin ortak bir sahiple bağlantılı olmasını önlemek için yeni bir anahtar çifti kullanılmalıdır. Girdilerinin aynı mal sahibine ait olduğunu zorunlu olarak ortaya koyan çok girdili işlemlerde bazı bağlantıların kurulması hala kaçınılmazdır. Risk, bir anahtarın sahibi ortaya çıkarsa, bağlantının aynı sahibine ait diğer işlemleri de ortaya çıkarabilmesidir.

Hesaplamalar

Dürüst zincirden daha hızlı alternatif bir zincir oluşturmaya çalışan bir saldırganın senaryosunu ele alalım. Bu başarılsa bile, sistemi yoktan değer yaratmak veya saldırgana ait olmayan parayı almak gibi keyfi değişikliklere açık hale getirmez. Düğümler geçersiz bir işlemi ödeme olarak kabul etmeyecek ve dürüst düğümler asla bunları içeren bir bloğu kabul etmeyecektir. Bir saldırgan sadece yakın zamanda harcadığı parayı geri almak için kendi işlemlerinden birini değiştirmeyi deneyebilir.

Dürüst zincir ile saldırgan zincir arasındaki yarış "Binom Rastgele Yürüyüşü (Binomial Random Walk)" şekilde karakterize edilir. Başarılı olay, dürüst zincirin bir blok uzatılması ve farkının +1 artmasıdır; başarısız olay ise saldırganın zincirinin bir blok uzatılması ve farkın -1 azalmasıdır.

Bir saldırganın belirli bir açığı yakalama olasılığı, bir "Kumarbazın İflası (Gambler's Ruin)" sorununa benzer. Sınırsız kredisi olan bir kumarbazın borçla başladığını ve başabaş durumuna ulaşmak için potansiyel olarak sonsuz sayıda deneme oynadığını varsayalım. Başabaş olma olasılığını veya bir saldırganın dürüst zinciri yakalama olasılığını şu şekilde hesaplayabiliriz [8]:

p = dürüst bir düğümün bir sonraki bloğu bulma olasılığı

q = saldırganın bir sonraki bloğu bulma olasılığı

qz = saldırganın z blok geriden yetişebilme olasılığı

$$q_z = \begin{cases} 1 & \text{if } p \le q \\ (q/p)^z & \text{if } p > q \end{cases}$$

p > q varsayımımız göz önüne alındığında, saldırganın yetişmesi gereken blok sayısı arttıkça olasılık üstel olarak düşer. İhtimaller aleyhindeyken, erkenden şanslı bir hamle yapmazsa, daha da geride kaldıkça şansı yok denecek kadar azalır.

Şimdi, gönderenin işlemi değiştiremeyeceğinden yeterince emin olmadan önce yeni bir işlemin alıcısının ne kadar beklemesi gerektiğini ele alacağız. Göndericinin, alıcıyı bir süreliğine kendisine ödeme yaptığına inandırmak ve bir süre geçtikten sonra kendisine geri ödeme yapmak üzere değiştirmek isteyen bir saldırgan olduğunu varsayıyoruz. Bu gerçekleştiğinde alıcı uyarılır, ancak gönderici bunun için çok geç olmasını umar.

Alıcı yeni bir anahtar çifti oluşturur ve imzalamadan kısa bir süre önce gönderene ortak anahtarı verir. Bu, gönderenin yeterince ilerleyebilecek kadar şanslı olana kadar sürekli üzerinde çalışarak vaktinden önce bir blok zinciri hazırlamasını ve ardından işlemi o anda gerçekleştirmesini engeller. İşlem gönderildikten sonra, dürüst olmayan gönderen, işleminin alternatif bir sürümünü içeren paralel bir zincir üzerinde gizlice çalışmaya başlar.

Alıcı, işlem bir bloğa eklenene ve sonrasında z bloklar bağlanana kadar bekler. Saldırgan tam olarak ne kadar ilerleme kaydettiğini bilmiyor, ancak dürüst blokların blok başına beklenen ortalama süreyi aldığını varsayarsak, saldırganın potansiyel ilerlemesi beklenen değere sahip bir Poisson dağılımı olacaktır:

$$\lambda = z\frac{q}{p}$$

Saldırganın şu anda yetişebilme olasılığını bulmak için, her ilerleme miktarı için Poisson yoğunluğunu, o noktadan itibaren yetişebilme olasılığıyla çarpıyoruz:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} (q/p)^{(z-k)} & \text{if } k \le z \\ 1 & \text{if } k > z \end{cases}$$

Dağılımın sonsuz kuyruğunu toplamaktan kaçınmak için yeniden düzenliyoruz...

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

C koduna dönüştürüyoruz...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
 double p = 1.0 - q;
 double lambda = z * (q / p);
 double sum = 1.0;
 int i, k;
 for (k = 0; k <= z; k++)
 {
 double poisson = exp(-lambda);
 for (i = 1; i <= k; i++)
 poisson *= lambda / i;
 sum -= poisson * (1 - pow(q / p, z - k));
 }
 return sum;
}
```

Bazı sonuçları çalıştırdığımızda, olasılığın z ile üstel olarak düştüğünü görebiliriz.

```
q=0.1
z=0 P=1.0000000
z=1 P=0.2045873
z=2 P=0.0509779
z=3 P=0.0131722
z=4 P=0.0034552
z=5 P=0.0009137
z=6 P=0.0002428
z=7 P=0.0000647
```

```
z=8 P=0.0000173
z=9 P=0.0000046
z=10 P=0.0000012
q=0.3
z=0 P=1.0000000
z=5 P=0.1773523
z=10 P=0.0416605
z=15 P=0.0101008
z=20 P=0.0024804
z=25 P=0.0006132
z=30 P=0.0001522
z=35 P=0.0000379
z=40 P=0.0000095
z=45 P=0.0000024
z=50 P=0.0000006
```

P'nin %0,1'den küçük olduğu durumlarda çözüm...

```
P < 0.001
q=0.10 z=5
q=0.15 z=8
q=0.20 z=11
q=0.25 z=15
q=0.30 z=24
q=0.35 z=41
q=0.40 z=89
q=0.45 z=340
```

Sonuç

Güvene dayanmayan elektronik işlemler için bir sistem önerdik. Mülkiyet üzerinde güçlü bir kontrol sağlayan, ancak çifte harcamayı önlemenin bir yolu olmadan eksik kalan dijital imzalardan yapılmış paraların olağan çerçevesiyle başladık. Bunu çözmek için, herkese açık bir işlem geçmişini kaydetmek için iş kanıtını kullanan eşler arası bir ağ önerdik. CPU gücünün çoğunluğunu dürüst düğümler kontrol ettiği takdirde, bir saldırganın değiştirmesi bu hızla hesaplama açısından pratik olmayan hâle gelir. Ağ, yapılandırılmamış basitliğiyle sağlamdır. Düğümler aynı anda çok az koordinasyonla çalışır. Mesajların belirli bir yere yönlendirilmediğinden ve yalnızca en iyi çaba temelinde iletilmesi gerektiğinden, bunların tanımlanması gerekmez. Düğümler, iş kanıtı zincirini gittikleri süre boyunca olan bitenin kanıtı olarak kabul ederek ağdan istedikleri zaman ayrılabilir ve yeniden katılabilir. CPU gücüyle oy kullanırlar, geçerli blokları genişletme üzerinde çalışarak kabul ettiklerini ve geçersiz blokları üzerinde çalışmayı reddederek reddettiklerini ifade ederler. Bu mutabakat mekanizması ile ihtiyaç duyulan tüm kurallar ve teşvikler uygulanabilir.

Referanslar

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In 20th Symposium on Information Theory in the Benelux, May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In Journal of Cryptology, vol 3, no 2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In Sequences II: Methods in Communication, Security and Computer Science, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In Proceedings of the 4th ACM Conference on Computer and Communications Security, pages 28-35, April 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocols for public key cryptosystems," In Proc. 1980 Symposium on Security and Privacy, IEEE Computer Society, pages 122-133, April 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.