Bir Çarpışmanın Anatomisi: Güvercin Yuvası İlkesi, Doğum Günü Paradoksu ve Dijital Doppelgängerlar

ESRA KORKMAZ

Düzce Üniversitesi Gölyaka Meslek Yüksekokulu
esrakorkmaz@duzce.edu.tr

Hiç size tıpatıp benzeyen ancak aranızda hiçbir kan bağı olmayan birine, yani bir *doppelgängere* rastladınız mı? Avrupa mitolojisinde bu karşılaşma, yaklaşan bir felaketin habercisi ve bir uğursuzluk olarak kabul edilirdi. Çünkü özünde, birey olarak benzersizliğimize duyduğumuz inancın temelden sarsılması demekti.

Peki ya bu benzersizlik yanılgısı, yalnızca insanlara özgü değilse? Ya içinde yaşadığımız devasa veri evreninde, size özel bir fotoğrafın, kritik bir belgenin veya gizli bir formülün, tamamen alakasız başka bir veriyle tıpatıp aynı dijital parmak izine sahip olması mümkünse? İşte bu, bir efsane ya da bir tesadüf değil, matematiğin en temel prensiplerinden biri olan Güvercin Yuvası İlkesi'nin doğrudan bir sonucudur.

Bu yazıda, matematiğin bu zarif ilkesinden yola çıkarak, olasılığın şaşırtıcı dünyasına adım atacağız. Ardından, bu fikirlerin dijital çağın görünmez koruyucuları olan hash fonksiyonlarıyla yollarının nasıl kesiştiğini keşfedeceğiz. Böylece, basit bir mantık oyunundan doğan matematiksel bir kavramın, modern dünyanın dijital güvenliğini ayakta tutan temel taşlarından biri hâline nasıl geldiğini görmüş olacağız.

■ Güvercin Yuvası İlkesi

Güvercin Yuvası İlkesi, kombinatoriğin en temel ilkelerinden biridir ve son derece basit bir gerçeği ortaya koyar:

n > m olmak üzere, n adet nesne m farklı kutuya yerleştirilirse, en az bir kutuda birden fazla nesne bulunmak zorundadır.

Ya da özel olarak, elimizdeki güvercin sayısı yuva sayısından fazlaysa, en az bir yuvayı birden fazla güvercinin paylaşması kaçınılmazdır.

Bu fikir ilk olarak 17. yüzyılda Fransız matematikçi Jean Leurechon tarafından ortaya atılmıştır. Ancak, bilim dünyasında sıkça rastlanan ve Stigler Yasası (N1) olarak bilinen adlandırma eğilimine de uygun olarak, Alman matematikçi Dirichlet'ye atıfla *Dirichlet'in Çekmece İlkesi* olarak bilinir.

Şimdi, bu ilkenin ne kadar güçlü ve derin çıkarımlar barındırdığını anlamak için aşağıdaki örneği birlikte inceleyelim.

Örnek: Türkiye'de yaşayan her insanın farklı sayıda saç teline sahip olması matematiksel olarak imkansızdır. Yani, bir yerlerde sizinle tam olarak aynı sayıda saç teli olan birileri mutlaka vardır.

Bu iddiayı ispatlamak için önce yuvalarımızı ve güvercinlerimizi belirleyelim.

Bilimsel araştırmalar bir insanın başında yaklaşık 100.000 ila 150.000 saç teli olduğunu söyler. O halde, olası saç teli sayısı seçeneklerimiz şunlardır: 0 saç teli, 1 saç teli, 2 saç teli, …, 150.000 saç teli. Bu bize

toplam 150.001 adet farklı yuva verir. Bu yuvalara yerleştireceğimiz güvercinler ise, Türkiye'de yaşayan yaklaşık 85,7 milyon kişidir.

Şimdi elimizde 85,7 milyon güvercin ve sadece 150.001 yuva var. Güvercin sayımız yuva sayımızdan katbekat fazla. Demek ki Güvercin Yuvası İlkesi'ne göre, ülkenin bir yerlerinde sizinle tam olarak aynı sayıda saç teline sahip en az bir kişi olmak zorundadır.

■ Doğum Günü Paradoksu

Güvercin Yuvası İlkesi, nesne sayısı kutu sayısını aştığında bazı nesnelerin aynı kutuya düşmek zorunda olduğunu söyleyerek, çakışmanın kaçınılmazlığını garanti eder. Ancak bu ilkenin ötesinde, çok daha şaşırtıcı bir durum vardır. Bazen bir çakışma olasılığı, kaçınılmazlık için gereken sayıya ulaşmadan çok daha önce dikkate değer bir ölçüde yükselir. Bu olguyu gözler önüne seren en güzel örnek Doğum Günü Paradoksudur.

Güvercin Yuvası İlkesi'ni doğrudan uygular ve 365 günü yuvalar, insanları ise güvercinler olarak kabul edersek, 366 kişilik bir grupta en az iki kişinin aynı doğum gününü paylaşmasının kaçınılmaz olduğunu söyleyebiliriz. Doğum günü paradoksunun ortaya koyduğu şaşırtıcı gerçek ise, çakışma olasılığının %50'yi aşması için yalnızca 23 kişinin yeterli olduğudur.

Peki, sezgilerimize aykırı gelen bu durum nasıl açıklanabilir? Buradaki kilit nokta, paradoksun "Gruba yeni katılan biri benimle aynı doğum gününe sahip mi?" sorusunu değil, "Grupta herhangi iki kişinin aynı doğum gününü paylaşma olasılığı nedir?" sorusunu sormasıdır (N2).

Bu ikinci sorunun anahtarı, potansiyel çift sayısının kişi sayısından çok daha hızlı artmasıdır. Yani grupta:

- 2 kişi varsa kontrol edilecek sadece 1 çift,
- 3 kişi varsa, 3 çift (A, B ve C kişileri için, (A,B), (A,C), (B,C)),
- 23 kişi varsa, $\frac{23\times 22}{2} = 253$ farklı çift vardır. (Burada, n kişilik bir gruptan 2 kişilik kaç farklı çift oluşturulabileceğini hesaplamak için kullanılan $\binom{n}{2} = \frac{n(n-1)}{2}$ kombinasyon formülünü hatırlayalım.)

Her bir çift, bir doğum günü çakışması için yeni bir fırsattır. 253 farklı deneme yapmak, olasılığı hızla %50'nin üzerine taşır.

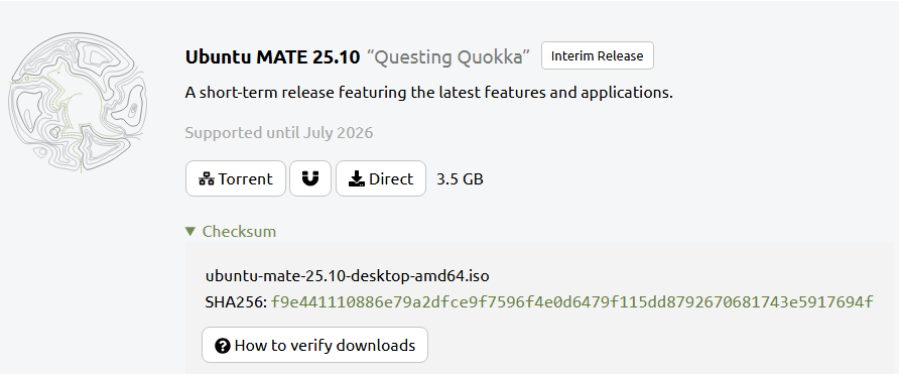
Genel bir kural olarak, olası sonuçların sayısı K ise, bir çakışmanın gerçekleşme olasılığının %50'yi aşması için yaklaşık olarak $\sqrt{K}$ deneme yapmak gerekir (N3).

Peki bu prensip dijital evrenimizin temel taşlarından biri olan kriptografi ile nasıl ilişkilidir? Bu sorunun cevabı, dijital parmak izi olarak da bilinen hash fonksiyonlarında saklıdır.

Hepimizin de bildiği gibi, günümüz dünyasında verilerimiz — fotoğraflarımız, yazışmalarımız, bankacılık işlemlerimiz— dijital formda saklanır ve paylaşılır. Bu devasa veri akışında, bir mesajın yolda değiştirilmediğinden, indirdiğimiz bir yazılımın orijinal olup olmadığından ya da bir dosyanın içeriğinin bozulmadığından nasıl emin olabiliriz? İşte bu noktada, verilerin parmak izi niteliğinde eşsiz bir kimlik taşıması hayati derecede önemlidir ve kriptografik hash fonksiyonları tam olarak bu ihtiyaca cevap verir.

Bu fonksiyonlar, değişken uzunluktaki herhangi bir veriyi alıp, onu sabit uzunlukta bir veriye (yani 0 ve 1'lerden oluşan sabit uzunlukta bir diziye) dönüştürür. Matematiksel olarak ifade etmek gerekirse, d sabit bir bit sayısı (örneğin 128, 160 veya 256 bit) olmak üzere, bir H hash fonksiyonu $H : \{0,1\}^* \to \{0,1\}^d$ biçiminde tanımlıdır. Burada $\{0,1\}^*$ keyfi uzunluktaki tüm olası girdilerin (metinler, fotoğraflar, dosyalar vb.), $\{0,1\}^d$ ise d-bit uzunluğundaki çıktıların kümesini gösterir.

Hash fonksiyonları parola güvenliğinden dosya bütünlüğüne, dijital imzalardan blokzincir teknolojisine kadar pek çok kritik uygulamanın temelini oluşturur. Örneğin, internetten bir program indirdiğimizde gördüğünüz o anlamsız karakter dizisi, dosyanın dijital parmak izidir. Bu sayede dosyanın orijinal ve değiştirilmemiş olduğunu saniyeler içinde doğrulayabiliriz.



Şekil 1. Ubuntu MATE İndirme Sayfası ve SHA256 Hash Fonksiyonu

Yüzlerce sayfadan oluşan bir sözleşmeye atılan dijital imzalar, belgenin tamamını şifrelemek yerine, çok daha küçük olan hash değerini şifreleyerek hem güvenliği hem de verimliliği garanti altına alır. Bitcoin ve benzeri kripto paraların temelini oluşturan blokzincir teknolojisi ise, her bir blokta önceki bloğun hash değerini saklayarak kırılması neredeyse imkânsız bir güven zinciri oluşturur.

Hash fonksiyonlarının en yaygın ve vazgeçilmez kullanım alanlarından biri de parola korumasıdır. Bir web sitesine üye olurken girilen şifreler açık metin hâlinde veri tabanına kaydedilmez; bunun yerine şifrenin hash değeri saklanır. Böylece kötü niyetli bir saldırgan veri tabanına erişse bile, şifrenin kendisini değil, geri döndürülemez (N4) matematiksel gölgesini ele geçirmiş olur. İşte bu tek yönlülük özelliği sebebiyle, unuttuğumuz bir parola bize yeniden gönderilerek değil de 'şifrenizi sıfırlayın' seçeneğiyle yenilenebilir. Sistem bile orijinal şifreyi bilmez, sadece girilen şifrenin hash değerini doğrular.

Günümüzde en yaygın kullanılan ve en güvenli hash algoritmalarından biri SHA-256 (Secure Hash Algorithm 256-bit)'dır. Bu algoritma herhangi bir veriyi bir dizi zincirleme matematiksel işlemden geçirerek 256 bitlik bir hash üretir.

Örneğin 'hacettepe' girdisinin SHA-256 değeri:

`d3d52d1f1e73d151637c610fe21f6910d8bd64fd3956eb3a7cea79e656d4cfd1`

biçimindedir (N5).

Hash fonksiyonları *Çığ Etkisi (Avalanche Effect)* olarak bilinen bir özelliğe sahiptir. Bu özelliğe göre, orijinal veride yapılan küçük bir değişiklik bile tamamen farklı bir hash değeri üretir. Örneğin sadece ilk harfi değiştirerek elde ettiğimiz 'Hacettepe' girdisinin SHA-256 değeri,

`72ea48626f122b8cf0ac2396b86acaf0e58076f3a1b41b39cae351f825fb91d2`

gibi bambaşka bir dizi olacaktır.

SHA-256 gibi hash fonksiyonları dijital parmak izi işlevi görseler de yapıları gereği bazı matematiksel sınırlılıkları bulunmaktadır. Bu fonksiyonlar her zaman sabit uzunlukta bir çıktı ürettikleri için, üretebilecekleri tüm olası çıktıların sayısı sınırlıdır. Örneğin SHA-256 için bu değer 2^{256}'dır (N6). Bu sayı yaklaşık 1.15×10^{77} gibi akıl almaz bir büyüklükte olsa da sonludur.

Öte yandan, olası tüm girdilerin kümesi teorik olarak sonsuzdur; çünkü mevcut bir veriye sadece bir harf, bir piksel veya bir bit ekleyerek yeni bir girdi yaratmak mümkündür. O halde, Güvercin Yuvası İlkesi'ne göre, sonsuz sayıda girdi (güvercin) ve sonlu sayıda çıktı (yuva) olduğundan, en az iki farklı girdinin aynı SHA-256 değerine sahip olması kaçınılmazdır (N7).

Ancak bu dijital *doppelgängerlar* teorik olarak var olsalar da, pratikte bulunmaları son derece zordur. Çünkü Doğum Günü Paradoksu'na göre, SHA-256'da %50 çarpışma olasılığına ulaşmak için yaklaşık $\sqrt{2^{256}} = 2^{128}$ deneme yapmak (N8) gereklidir ve bu sayı astronomik büyüklüktedir. Dünyadaki tüm bilgisayarlar saniyede milyarlarca hash üretse bile, bu kadar deneme için evrenin yaşını bile katbekat aşan bir süre gerekir.

Bu nedenle SHA-256 gibi modern hash fonksiyonlarının pratikte çarpışma direncine sahip olduğu kabul edilir. Matematiksel *doppelgängerlar* teorik olarak hep var olacak olsalar da, onları bulmamızı engelleyen o devasa sayı duvarı, dijital evrenimizin (en azından şu an için) en büyük güvencesidir.

■ Kaynaklar

[1] Doppelgänger Etkisi: Yakın Akraba Bile Olmayan "İkizler" Neden Birbirine Bu Kadar Benziyor? *Evrim Ağacı*. https://tinyurl.com/4hyk39hk (Erişim Tarihi: 17.11.2025).

[2] The Pigeonhole Principle. *Maths Careers*. https://www.mathscareers.org.uk/pigeonhole-principle/ (Erişim Tarihi: 17.11.2025).

[3] How Many Hairs on a Human Head? *Healthline*. https://www.healthline.com/health/how-many-hairs-on-a-human-head (Erişim Tarihi: 17.11.2025).

[4] Tesler, G. The Birthday Problem, *UC San Diego Mathematics*.

[5] Kak, A. Hashing for Message Authentication and Digital Signatures, Lecture Notes, *Purdue University - School of Electrical and Computer Engineering*.

[6] Emn178. Online SHA256 Hash Generator. *GitHub Pages*. https://emn178.github.io/online-tools/sha256.html (Erişim Tarihi: 17.11.2025).

■ NOTLAR

(N1) Stigler'in Eponim Yasası, buluşların ve keşiflerin genellikle orijinal kaşifinin adını almadığını öne sürer. Tıpkı bu fikrin aslında Robert K. Merton'a ait olup, Stigler Yasası olarak bilinmesi gibi.

(N2) İlk soruda karşılaştırma hedefi sabit ve tektir. Örneğin doğum günümüz 7 Temmuz ise, gruptaki diğer herkesi sadece bu tek tarihle karşılaştırırız. Odaya giren herhangi bir kişinin doğum gününün tam olarak 7 Temmuz olma olasılığı $\frac{1}{365}$'dir, yani oldukça düşüktür. İkinci soruda ise ortada belirli bir tarih yoktur. Herhangi bir eşleşmenin olması yeterlidir.

(N3) $\sqrt{K}$ kuralı, Doğum Günü Paradoksu'nun altında yatan karmaşık matematiksel formüllerden elde edilen bir yaklaşımdır. Bu yaklaşım, K sayısının çok büyük olduğu durumlarda oldukça isabetli sonuçlar verse de, $K = 365$ gibi görece küçük değerler için gerçek sonuçtan sapmalar gösterebilir.
Doğum Günü Paradoksu'ndaki 23 kişinin nasıl elde edildiğini görmek için problemi tersten ele alalım. Bunun için, n kişilik bir grupta kimsenin aynı doğum gününü paylaşmama olasılığını P(Çakışma Yok) ile gösterelim. Bu durumda aradığımız çakışma olasılığı P(Çakışma Var) $= 1 - P$(Çakışma Yok) olacaktır. Amacımız, P(Çakışma Var) > 0.50 eşitsizliğini sağlayan en küçük n tamsayısını bulmaktır. n kişilik bir grupta hiç çakışma olmaması olasılığını, odaya sırayla giren kişileri tek tek ele alarak hesaplayabiliriz. Grup içinde çakışma olmaması için, her yeni gelen kişi kendinden önceki kişilerden farklı bir günde doğmuş olmalıdır. P(Çakışma Yok) değeri, bu adımların her birinin gerçekleşme olasılıklarının çarpımıdır.

1. Odaya giren ilk kişi için, kimseyle çakışmayan bir doğum gününe sahip olma olasılığı $\frac{365}{365} = 1$'dir, çünkü zaten odada kimse yoktur.
2. İkinci kişinin birinci kişiden farklı bir günde doğmuş olma, yani doğum günlerinin çakışmaması olasılığı $\frac{364}{365}$'dir.
3. Üçüncü kişinin, ilk iki kişiden farklı bir günde doğmuş olma olasılığı $\frac{363}{365}$'dir.
4. n. kişinin, kendinden önceki $n-1$ kişinin tamamından farklı bir günde doğmuş olma olasılığı $\frac{365-(n-1)}{365} = \frac{365-n+1}{365}$'dir.

Dolayısıyla, hiç çakışma olmama olasılığı bu bağımsız olayların çarpımıdır:

$$P(\text{Çakışma Yok}) = \frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \times \cdots \times \frac{365-n+1}{365} = \frac{\frac{365!}{(365-n)!}}{365^n}$$

O halde, $n = 22$ kişi için $P(\text{Çakışma Var}) \approx 0.4757$ iken, $n = 23$ için $P(\text{Çakışma Var}) \approx 0.5073$ olur. Böylece $n = 23$ kişilik bir grupta, en az iki kişinin aynı doğum gününü paylaşma olasılığı %50'yi aşar.

(N4) Bu özellik *ön-görüntü direnci (pre-image resistance)* veya *tek yönlülük (one-way property)* olarak adlandırılır. H bir hash fonksiyonu ve h bir hash değeri olmak üzere, $H(m) = h$ olacak şekilde bir m girdisi bulmanın hesaplama açısından mümkün olmadığı *(computationally infeasible)* anlamına gelir.

(N5) Burada gördüğümüz 64 karakterlik hash değeri 16'lık sayı sisteminde yazılmıştır ve her bir karakter 4 bitlik bilgiyi temsil eder. Dolayısıyla, d $\to$ 1101, 3 $\to$ 0011, 5 $\to$ 0101, ...şeklinde ikilik sayı sistemine dönüştürülürse, $64 \times 4 = 256$ bitlik bir dizi elde edilir.

(N6) Her bir bitin 0 veya 1 olmak üzere iki farklı ihtimali olduğu göz önüne alındığında, 2^{256} ihtimal olduğu kolayca görülebilir.

(N7) İki farklı veri parçasının, aynı hash fonksiyonundan geçirilmesi sonucu tamamen aynı hash değerini üretmesine *çakışma (collision)* denir. Kriptografik hash fonksiyonları için bu durum, fonksiyonun ideal özelliklerinden biri olan çakışma direncinin kırıldığı anlamına gelir ve ciddi güvenlik zafiyetlerine yol açabilir.

(N8) Burada "deneme yapmak", genellikle *kaba kuvvet (brute-force)* saldırısı olarak bilinen bir yöntemi ifade eder. Bu süreç temelde şu adımlardan oluşur: Önce rastgele bir veri girdisi (örneğin bir metin, karakter dizisi veya dosya) seçilir. Bu girdinin hash değeri SHA-256 gibi bir hash fonksiyonu kullanılarak hesaplanır ve bulunan değer daha sonra karşılaştırmak üzere saklanır. Ardından tamamen farklı yeni bir girdi seçilir ve onun da hash değeri hesaplanır. Bu yeni hash, daha önce kaydedilen tüm hash değerleriyle karşılaştırılır. Eğer bir eşleşme —yani bir çarpışma— bulunursa işlem sonlandırılır.
Ancak genellikle eşleşme hemen bulunamaz. Bu durumda, birinci adıma geri dönülür ve tamamen farklı yeni bir girdi seçilerek aynı döngü (yani girdi seçme, hash hesaplama ve karşılaştırma işlemi) trilyonlarca, hatta katrilyonlarca kez tekrarlanır. Amaç, astronomik sayıdaki olası girdinin içinden, tesadüfen aynı hash değerini üreten iki tanesini bulmaktır. Hash fonksiyonu yeterince güvenliyse, bu yöntemle pratik bir çarpışma bulmak hesaplama açısından imkânsıza yakın süreler gerektirir. Çarpışmaların pratikte bulunamaması kritiktir. Çünkü bir saldırgan, sizin imzaladığınız meşru bir dijital belgeyle aynı hash değerini taşıyan sahte bir belge üretebilseydi, dijital imzanız bu sahte belge için de geçerli olurdu. Bu durum ise, dijital güvenliğin temel mekanizmalarının tamamen çökmesi anlamına gelirdi.