CRYPTTECH
CYBER SECURITY INTELLIGENCE

CRYPTTECH
KRİPTOLOJİ LABORATUVARI

BİZANS GENERALLERİ

Bir ordu kaç haine rağmen ayakta kalabilir?
Bizans tarihi, generallerin iktidarı ele geçirmek için hükümdarlara kurdukları komplolarla doludur.
Hükümdarlara komplo kuran generallerin birbirlerine komplo kurmaları, kurmaya çalışmaları da çok doğal karşılanmalıdır.
Bizans'ın bu tarihi ünü, çok önemli bir ağ problemini temsil eden bir metafora dönüşmüş ve bu ağ problemi bu isimle ünlenmiştir.
Bizans Generalleri Problemi, kripto para teorisinde önemli bir yer tutan blok zincirinin güvenlik kriterindeki tartışma nedeniyle tekrar merak uyandırdı.
Bir güvenlik tartışmasına girmeden önce, temel problemin detaylarını anlamış olmak şarttır...

MART — 2018
https://www.crypttech.com/

BİZANS GENERALLERİ
GÜVENİLİR SİSTEMLERDE HATA TOLERANSI

Bir şeyi mümkün olan en basit şekilde anlatın; ama asla daha basit değil...

Otomasyon sistemlerinde kontrol mekanizmasının doğru, verimli ve kararlı çalışmasını garanti altına almayı hedefleyen bir algoritma problemidir Bizans Generalleri Problemi. Bir kısmı vatanına sadık bir kısmı vatan haini generallerden kurulu bir ordunun gerçekleştirdiği kuşatmayı bozguna uğramadan sonlandırabilmesinin koşullarını arayan bir metafor üzerinden aslında bir ağ probleminin çözümü tartışılmaktadır. Bir ordu kaç vatan hainine rağmen bozguna uğramadan ayakta durabilir? Bu temel sorunun cevabı, aynı zamanda bir sistemin kaç tane sorunlu birime karşın çökmeden çalışabileceği sorusunun da yanıtını oluşturacaktır. Problemin genel çözümü halen araştırılmakta olup var olan cevaplar çeşitli kısıtlamalar barındırmaktadır. Aşağıdaki yazıda sadece Lamport, Shostak ve Pease tarafından önerilmiş olan algoritma tartışılacaktır.

Otomasyon kavramının ve uygulamalarının hayatın içine gittikçe daha fazla girmesiyle beraber, sistem güvenilirliği artan bir önem kazanmaktadır. Otomasyonun, baştan sona bir bütün olarak, tek bir birimin kontrolüne verilmesi, bu birimde çıkabilecek olan anlık veya kalıcı hatalara karşı sistem güvenilirliğini tehlikeye atmak anlamına gelir. Hele de bahsedilen sistem, örneğin, bir nükleer reaktörün reaksiyon kontrollerinde kullanılmakta veya bir uzay mekiği fırlatma rampasında ateşleme senkronizasyon biriminde yer almaktaysa tek birimli kontrol söz konusu bile olamaz. Bu tür sistemlerde kontrol, birbirinden bağımsız olarak çalışan çok sayıda eş birim tarafından ortaklaşa gerçekleştirilir. Bu eş birimlerin ortaklaşa karar alma mekanizmasına sistemin **konsensüs kuralı** veya **konsensüs yöntemi** adı verilir. En sık rastlanan yöntemler; çoğunluk tespiti, oy birliği ve ortalama değer analizi olsa da daha birçok farklı yöntem konsensüs yöntemi olarak kullanılabilir.

Teknolojinin ilerlemesiyle beraber işlemci ve dolayısıyla, kontrol birim güvenilirliği de ilerlemiştir. Birimlerin hatalı çalışmaları, özellikle üst seviye ürünler kullanıldığında gittikçe az rastlanan bir durum olma eğilimindedir. Fakat teknolojinin yaygınlaşması, ağ kullanımının genişlemesi, sistemlerde sorun yaratmak isteyen saldırgan tarafın da mevzi kazanmasına, mesafe kaydetmesine neden olmaktadır. Dolayısıyla, güvenilir bir sistem yaratmada karşılaşılan tek sorun birim güvenilirliği değil, aynı zamanda birim güvenliğidir. Birim güvenliklerini sağlamak amacıyla kullanılan çeşitli yöntemler bulunmaktadır ve bu yöntemler bu yazının konusunu oluşturmaz. Sistem güvenilirliğinin en kötü senaryoya göre düzenlenmesi gerekir. Bu düzenlemelerin hazırlık aşamasında, sistemin, saldırı altında olsa ve bazı birimler ele geçirilse dahi nereye kadar hatasız çalışabileceği araştırılmalıdır. Diğer bir deşiyle, bir sistem kaç tane sorunlu birime rağmen sorunsuzca çalışabilir? Doğal olarak bu sorunun cevabı bir mutlak sayıdan değil, bir orandan oluşa-

caktır. Belli oranda sorunlu birime rağmen sorunsuzca çalışabilen bir sistemin **hata toleransına sahip** veya **hata toleranslı bir sistem** olduğu söylenir. Bir bakışla, bu adlandırma kendi içinde çelişik bir yapıdadır: En küçük bir hatayı bile tolere edemeyecek kadar kritik bir noktada yer alan sistem, hata riskinin minimize edilmesi için kurulan düzenekten dolayı hata toleranslı olarak adlandırılmaktadır!

Bilgisayar araştırmalarında, algoritma sorunlarını günlük hayat veya tarih üzerinden kurgulanan bir metafor aracılığıyla anlatmak ve genel problemi bu metafor üzerinden adlandırmak sık rastlanan bir şeydir. Yukarıda ortaya koymuş olduğumuz sorun da **Bizans Generalleri Problemi** olarak bilinir. Aşağıda bu metafordan bahsedeceğiz. Ancak anlatıma bu metafordan başlamanın, konunun anlaşılmasını basitleştirmekten çok, lafı uzatmaya neden olacağını ve konuyu bilmeyenlerde ciddi bir kafa karışıklığı yaratacağını düşünmekteyiz. Bu nedenle, bu yazıda sorun ilk önce teknik bir yaklaşımla incelenecektir.

Sorunun bilgisayar araştırmalarındaki adı Bizans Generalleri Problemi olduğu için, böyle bir sistemde yer alabilecek olan birim hataları da Bizans hataları olarak adlandırılır. Bu tür hatalardan muzdarip birimlere de **Bizans birimleri** adı verilir. Bizans hataları bazı açılardan genel işlemci hatalarından ayrılırlar. Aynı algoritmayı çalıştıran ve biri genel hatalardan, diğeri Bizans hatalarından muzdarip iki birim arasındaki farklar şu şekilde sıralanabilir.

1. Genel işlemci hatalarında bir birim verilen girdiden ya doğru sonucu üretir ya da hiçbir sonuç üretmez. Bizans birimleri ise yanlış sonuçlar üretebilir.
2. Genel işlemci hatalarında sorunlu birim her zaman tespit edilebilir. Bizans birimlerini tespit etmek ise oldukça zordur.
3. Genel işlemci hatalarında birimler tekil olarak hatalıdır. Bizans birimleri, sistemi çökertmek amacıyla ortaklaşa çalışabilir.

Bir sistemin hata toleransı, tüm birimlerin bir Bizans birimi olma potansiyeline sahip olduğu düşünülerek tasarlanır.

1. Sorunun tanımlanması

Bir otomasyon sisteminde kontrolün çok sayıda eş birim tarafından bir konsensüs doğrultusunda gerçekleştirilmesi fikrini şu şekilde somuta indirgeyebiliriz. Sistem, işleyen bir algoritmanın belli adımları sonunda, algoritmadan elde edilecek olan sonuca göre bir eylem gerçekleştirecektir. Bu sonucun elde edilmesi işi tek bir işlemciye emanet edilemez; çünkü işlemci sorunlu olabilir veya, daha da kötüsü, bir Bizans birimi olabilir. Buna göre yapılması gereken, çok sayıda eş birime aynı algoritmayı aynı girdi değeri ile işlettirmek ve nihai sonuca, her birimin elde ettiği değeri göz önüne alarak ulaşmaktır. Her birimin elde ettiği değeri göz önüne alarak nihai sonuca ulaşmanın yöntemi, sistemin konsensüs kuralıdır ve biz ilk önce çoğunluk yöntemini kabul edeceğiz; fakat farklı bir yöntemin kabul edilmesi anlatacaklarımızın genelliğini ortadan kaldırmaz. Bu yöntemin işlemesinin nedeni, tüm sorunsuz birimlerin aynı girdiden aynı sonucu üretecek olmalarıdır. Diğer bir deyişle, sistemdeki sorunsuz birimlerin hepsi aynı sonucu üretecekler ve eğer çoğunlukta yer alıyorlarsa varılacak nihai sonuç, girdinin ürettiği sonuç; yani, *doğru sonuç* olacaktır. Aksi olarak, eğer sistemde sorunlu birimler, sorunsuz birimlerden fazlaysa ve hepsi, ortaklaşa çalıştıkları için veya tesadüfen, aynı sonucu vererek bir çoğunluk oluşturabiliyorlarsa nihai sonuç, girdiden üretilmeyen bir sonuç, yanlış sonuç olacak ve sistem girdinin üretmediği bir eyleme yönelecektir. Buna göre, sistem sorunsuz birimler çoğunlukta olduğu sürece işlevini yerine getirecektir. Fakat bunun aksi her zaman doğru olmaz; sorunlu birimler ortaklaşa bir çalışma içinde değillerse nihai sonuca etki edecek bir çoğunluk oluşturamayabilirler. Sorunsuz birimlerin çoğunlukta olmalarını gerektiren bu mekanizmanın çalışma prensibine **%51 kuralı** adı verilir. Öyleyse böyle bir sistemin %49'luk bir hata toleransından bahsedebiliriz. Bu değer oldukça yüksek bir değerdir ve aşağıda göreceğimiz üzere, sadece özel koşullar altında geçerli olup diğer faktörler göz önüne alındığında gerçeği yansıtmaz.

Dikkat edilirse yukarıda bir aksiyom ortaya attık ve tüm teorimizi bu aksiyom üzerine kurduk: *Tüm sorunsuz birimler, aynı girdi ile aynı algoritmadan aynı sonucu üretirler.* Bu aksiyomun derinlemesine tartışması bu yazının kapsam ve sınırlarını aşar; ancak sezgisel olarak aksiyomun doğruluğu açıktır ve biz bununla yetineceğiz. Buradan sonra adım adım gidersek: Sorunsuz birimlerin aynı girdiden aynı sonucu ürettikleri bilinmektedir. Öyleyse hepsinin aynı sonucu üretmeleri aynı girdi ile beslenmeleri gerekir. Gerçekleşecek eyleme, birimlerin oy çokluğu ile karar verileceğine göre, bir sistemin güvenilirliği için sorunsuz birimlerin çoğunlukta olmaları gereklidir. Bu durumda çözüm iki şarta indirgenmektedir:

1. Sorunsuz birimlerin aynı girdi ile beslenmesi;
2. Sorunsuz birimlerin çoğunlukta olması.

Sorunsuz birimlerin aynı girdi ile beslenmeleri meselesi veya beslenememeleri problemi ilk bakışta düşünüldüğü kadar basit bir problem değildir. İlk önce şunu tespit etmek gerekir ki, bir işlemcinin kendisine ulaşan girdinin doğru veya yanlış olduğunu anlayabilmesi kategorik olarak imkânsızdır. Eğer girdiler bir muhtemel değerler kümesinde yer almaktaysa; yani, girdi değeri birkaç farklı değerden biri olacaksa işlemci sadece bu küme dışında kalan anlamsız girdileri eleyebilir. Eğer gelen değer, gelmesi muhtemel değerlerden biriyse bu değerin doğru veya yanlış olarak değerlendirilebilme ihtimali yoktur. Ayrıca, eğer gelmesi muhtemel olan tek bir değer varsa da besleme birimine ihtiyaç olmayacaktır.

Birimlerin aynı girdi değeri ile beslenememesinin en basit nedeni, besleme biriminin sorunlu olmasıdır. Ama bu tek neden değildir. Örneğin, girdi değerlerinin bir sürekli değişken üzerinden birimler tarafından okunması, değişkenin değişim hızının çok yüksek olduğu durumlarda zorlaşabilir. Böyle bir durumda birimler tarafından okunacak olan değerler birbirine belli derecede komşu olsalar da tam anlamıyla eşit olmayabilirler. Bu ve benzer nedenler birimlerin eşit olmayan girdilerle beslenmelerine yol açabilir. Bu durumda sistem, bir algoritma ile birimlerin aynı girdi ile beslenip beslenmediklerini kontrol etmelidir.

2. Sorunun incelenmesi

Yukarıda söylediklerimizin ışığında görülmektedir ki, sorun iki ayaklı olabilir. Birincisi, besleme ünitesi her birime farklı bir girdi değeri gönderebilir, veya aynı anlama gelmek üzere, her birim besleme ünitesindeki değeri farklı okuyabilir. İkincisi de sistem içindeki işlem birimlerinden bazıları sorunlu veya ele geçirilmiş olabilir. Üstelik, bu iki olumsuzluk aynı anda da gerçekleşebilir.

Diğer taraftan, sistemde iki farklı sorun yaşanabilir. Birincisi, sistem karar verme ve bir eylem gerçekleştirme noktasında yetersiz kalır ve bir sonuç üretemez. Bu durum da iki farklı biçimde ortaya çıkabilir. Birinci halde birimler kullanacakları girdi değerinden emin olamazlar ve işlevsiz kalırlar. İkinci halde ise birimlerin ürettiği sonuçlar arasında bir çoğunluk oluşturulamaz ve sistem kararsız kalır. Sistemin yaşayacağı ikinci tür sorun ise ele geçirilmiş birimlerin sistemi istedikleri yönde hareket etmesini sağlamalarıdır. Bu durum ilkinden daha kötü sonuçlar doğurabilir.

Besleme ünitesinin tüm birimleri aynı değerle beslediğini; yani, sorunsuz olduğunu düşünürsek problem, sorunsuz birimlerin çoğunlukta olup olmadıkları sorusuna indirgenir ve çözümü için sorunsuz birimlerin %51 çoğunluğu sağlamaları gerekir. Ancak asıl zorluk çıkaran besleme ünitesinin sorunlu olduğu durumdur.

Besleme ünitesinin sorunlu olduğu genel bir halde, ünite her birimi farklı bir değerle besleyebilir. Ancak biz buradaki incelemede kolaylık olması için (ve aslında sık rastlanan durum bu olduğu için de) besleme ünitesinin iki farklı değerle besleme yaptığını düşüneceğiz ve bu değerleri D ve Y ile göstereceğiz; D doğru değeri, Y yanlış değeri temsil ede-

cek. Notasyon kolaylığı amacıyla, işlemcilerin işlettikleri algoritmanın birim olduğunu; yani, D girdisinden D sonucunu, Y girdisinden de Y sonucunu ürettiklerini farz ediyoruz. Diğer taraftan, sorunlu birimler doğru girdiden yanlış, yanlış girdiden doğru sonuç üretebildiklerine göre; sorunsuz işlemciler için girdi ve çıktı arasında bir *fark*, sorunlu birimlerdeyse girdi ile sonuç arasında bir *bağ* bulunmamaktadır.

Ayrıca, az önce tekil birimler için söylediğimiz şeyi tüm sistem için tekrar edelim: Ünitenin, birimleri muhtemel girdilerden hangisi ile besleyeceği bilinemez; daha doğrusu, bunu bilme otoritesi ünitenin kendisidir. Dolayısıyla, D değerinin doğru, Y değerinin yanlış olması meta–sistem seviyesinde bir değerlendirmedir: Bir şekilde, bir nedenle, öyle veya böyle... ünite, tüm birimleri D girdisi ile beslemesi gerekirken Y girdisi ile beslerse sistem açısından herhangi bir sorun görülemez. Çünkü sistem aldığı girdiden sonuç üretmekle mükelleftir. Bu bakış açısına göre, sistem seviyesinden bakıldığında besleme ünitesinde ortaya çıkabilecek olan tek sorun, kararsızlık; yani, birimlerin yarısını D girdisi ile, diğer yarısını da Y girdisi ile beslemesidir. Birimlerin çoğunluğu D girdisi yerine, herhangi bir nedenle, meta–sistem seviyesinde yanlış olduğu düşünülen Y girdisiyle besleniyorsa sistem seviyesinde doğru girdi Y değeridir [1].

Bu arada bir ara not olarak belirtelim ki, yukarıda bahsettiğimiz kararsızlık durumunun önüne geçmek için birim sayısının bir tek sayı olması gerektiği sonucuna varmak gerçekçi değildir. Bu çözüm, öncelikle, sadece ve sadece iki değerli besleme üniteleri için geçerli olur; ayrıca, iki değerli bir besleme ünitesi, bazı birimleri anlamsız değerlerle besleyerek tek sayıda birim barındıran bir sistemde dahi anlamlı besleme değerlerinin kararsızlık sonucu doğurmasına neden olabilir.

Yukarıda konuşulanlardan yola çıkarak şu üç maddeyi yazabiliriz. N birimden oluşan bir sistemde:

1. Doğru olarak kabul edilecek olan besleme değeri, en az $\lfloor N/2 \rfloor + 1$ birimin beslendiği değerdir.
2. Hem sorunsuz hem de sorunlu birimler doğru olduğu kabul edilen değerle beslenmiş olabilir.
3. Sistemin doğru girdiden doğru sonucu çıkarabilmesi için birimlerin çoğunluğunun doğru sonucu üretmesi gerekir.

Şimdi, bu yazdığımız maddeleri matematik diliyle ifade edeceğiz. Sorunsuz birimleri $\mathcal{S}$, sorunlu birimleri $\mathcal{B}$ ile gösterelim. Birinci ve ikinci maddelere göre,

$$\sum \mathcal{S}_D + \sum \mathcal{B}_D \geq \lfloor N/2 \rfloor + 1$$

yazabiliriz. Alt simgeler birimlerin beslenme değerlerini göstermektedir. Eşitsizlikte, *doğru girdi* ile beslenen sorunlu ve sorunsuz birimler toplamının, toplam birim sayısının yarısından fazla olması gerektiği ifade edilmektedir—aksi halde, D değeri *sistem açısından doğru girdi* olarak değerlendirilemez. Üçüncü madde ise

$$\sum \mathcal{S}_D + \sum \mathcal{B}^D \geq \sum \mathcal{S}_Y + \sum \mathcal{B}^Y + 1$$

şeklinde yazılabilir. Bu eşitsizlik de *doğru sonuç* üreten sorunlu ve sorunsuz birimler toplamının, yanlış sonuçlar üreten birimler toplamından fazla olması gerektiğini ifade etmektedir. Bizans birimlerinde girdi ile sonuç arasında bir bağ bulunmadığı için sonuçlar üst simge ile gösterilmiştir—sorunsuz birimlerde girdi ile sonuç aynıdır; dolayısıyla, alt simge yeterlidir.

Sistemin D girdisinden D sonucunu (ve aslında Y girdisinden de Y sonucunu) üretmesini temsil eden bu eşitsizlik sisteminin sayısal olarak çözülebilme imkânı yoktur. Ancak sistemin özellikleri düşünülerek bazı imkânsız durumlar ortaya konabilir.

1: Sistem, meta–sistem ayrışmasına dair en güzel örnek Russell Paradoksu olarak bilinir. Konu hakkında güzel bir açıklama https://plato.stanford.edu/entries/russell-paradox/ linkinde bulunabilir

3. Bizans Generalleri metaforu

Güvenilir sistemlerin özelliklerini mecaz bir yaklaşımla ortaya koyan Bizans Generalleri metaforu, ilk kez, 1982 yılında Leslie Lamport, Robert Shostak ve Marshall Pease tarafından yazılmış **The Byzantine Generals Problem** adlı makalede şematize edilmiştir. Bu şemada besleme ünitesi ve işlem birimleri arasında bir fark gözetilmez; besleme ünitesi de sistemin bir birimi olarak düşünülür.

Her birinin komutasında bir bölük asker bulunan belli bir sayıdaki Bizans generali, bir şehri kuşatmıştır ve saldırmakla geri çekilme arasında bir karar verme noktasındadır. Her general kendine göre bir durum değerlendirmesi yapacak ve bu değerlendirmenin sonucunda saldırma veya geri çekilme kararı verecektir. Bunun ardından, diğer tüm generallere kendi kararını bildirecek, nihai karar ise oy çokluğu ile verilecektir; yani, generallerin çoğunluğu saldırma yönünde karar vermişse bütün bölükler saldıracak veya, aksi durumda, bütün bölükler geri çekilecektir. Bölüklerden bir kısmının saldırıp bir kısmının geri çekilmesi metaforun felaket senaryosudur.

Sorun, bazı generallerin hain olmalarıdır. Hain generaller, diğer generallerden bazılarına saldırma yönünde görüş bildirirken bazılarına da geri çekilmeyi önermektedirler. Bu durumda, generallerin ellerindeki görüş listeleri birbirinden farklı olduğu için bazı bölükler saldırıya geçerken bazı bölükler geri çekilmeye başlayacak ve kuşatma bir bozgunla sonuçlanacaktır. Metaforun bu kısmı, Bizans biriminin varlığının, birimlerin kendi aralarında bir karara varmalarını engellediğini göstermektedir.

Metaforun ikinci kısmında, generallerden biri tüm komutayı ele almıştır ve saldırı veya geri çekilme kararını bu komutan vermektedir. *Kolayca görüleceği üzere, komutan besleme ünitesini temsil etmektedir.* Ancak generaller ya komutanın emrini doğru anlayıp anlamadıklarından ya da komutanın bir hain olup olmadığından emin olamadıkları için birbirlerine komutanın verdiği emri iletmektedirler. Böylece her generalin elinde yine bir liste oluşmakta ve saldırı veya çekilme kararını bu listeye göre vermektedir. Hem komutanın hem de generallerden bazılarının hain olmaları mümkündür. Ordunun bozguna uğramaması için her generalin elinde aynı sonucun çıktığı bir listenin olması gerektiği açıktır.

4. Çözümsüzlük hali

Besleme ünitesinin de bir birim olarak görüldüğü durumda bir sistemin en az üç birimden oluşması gerektiği görülür. İki birimden oluşamaz; çünkü bu, sistemde karar verme işinin tek bir işlemciye bırakıldığı anlamına gelir. Aşağıda göstereceğimiz üzere, üç birimli bir sistemde tek bir Bizans birimi dahi doğru sonucun üretilebilmesine engeldir. Bunu göstermek için Bizans Generalleri metaforundan yararlanalım.

Birinci durumda, komutan hain olsun ve iki generalden birine saldırı, diğerine geri çekilme emri versin. Generaller orduya sadık olduklarından ötürü, birbirlerine komutanın verdiği emri doğru bir şekilde ileteceklerdir. Bu durumda her iki generalin listesi de bir saldırı ve bir de geri çekilme emrinden oluşacak ve her ikisi de kararsız kalacaklardır. İkinci durumda ise generallerden biri hain olsun. Komutan her iki generale de saldırı emri vermiştir. Sadık olan general diğerine saldırı yönünde emir geldiğini söyleyecektir; fakat hain general sadık generale ters yönde bir iletimde bulunur. Bu durumda, sadık generalin listesi kararsızlık doğurmaktadır; hain generalin listesi kararlı bir şekilde saldırı emrini üretmekteyse de general hain olduğu için bunu yapmayacaktır; komutan ise saldırıya geçecektir. Bozgun kaçınılmazdır.

Yukarıdaki analizde dikkat edilmelidir ki; üçlü sistemde bir karara varılamamasının nedeni, hain generalin kurduğu becerikli bir plan de-

ğil, her iki senaryoda da sadık generallerden birinin kararsız kalmasıdır. Buradan yola çıkarak, sistemdeki hain sayısı m olmak üzere $3m$ birimden oluşan bir sistemde de karara varmanın imkânsız olduğunu gösterebiliriz. Bunun için çelişki yöntemini kullanacağız. Farz edelim ki, $3m$ sayıdaki Arnavut generali için ortak karara varma imkânı vardır. Bütün ordu, sadık generallerden oluşan iki tane m generalli parça ile m tane hain generalden oluşan bir parçaya bölünsün. Sadık generallerden oluşan parçalardan her biri bir general ile temsil edilebilir; çünkü sadık generaller her zaman aynı şekilde davranmaktadır. Hain generallerin davranış şekillerinin bir önemi olmadığı için (ilk cümlede belirttiğimiz üzere) tüm hain generaller de bir hain general ile temsil edilebilir. Bu durumda, sistem üç generalli bir sisteme dönüşmüştür. Ancak üç Bizans generalinden biliyoruz ki, üç generalli bir sistemin ortak karar alma imkânı bulunmamaktadır. Bu bir çelişkidir ve buna göre, $3m$ sayıdaki Arnavut generali için de çözüm imkânı yoktur. Çözümün olabilmesi için ağırlığın bir tarafa kaymasını sağlayacak bir sadık generale daha ihtiyaç vardır. Öyleyse çözümün var olabilmesi için sistemdeki birim sayısı en az $(3m + 1)$ olmalıdır.

Yukarıdaki ispatın baştan sona informal olması üzerinde biraz düşünülmesini gerektirir. Bu konu üzerine, "İki tane uygulamalı örnek" başlıklı bölümde konuşacağız.

5. Lamport, Shostak ve Pease algoritması

İlk iki bölümde söylediklerimizi Bizans Generalleri metaforunu kullanarak çok kısaca tekrar edelim. Komutanın hain olduğu ve (komutan hariç) çift sayıda generalin bulunduğu bir orduda tüm generaller sadık olsalar bile bir çoğunluk sağlamak imkânı bulunmayabilir. Eğer generaller içinde de hainler varsa ordunun, komutanın emrinin aksi bir eyleme kalkışması bile olasıdır.

Aşağıda, Bizans Generalleri probleminin çözümü için **Lamport, Shostak ve Pease** tarafından önerilen algoritmayı inceleyeceğiz. Algoritmanın amacı sistem içindeki Bizans birimlerini tespit etmektir. Eğer bu gerçekleştirilebilirse Bizans birimleri karar alma mekanizmasının dışına çıkarılacak ve karar alma noktasındaki konsensüs de oy çokluğundan oy birliğine kendiliğinden dönecektir. Çünkü algoritma, aynı zamanda, tüm birimlerin aynı girdi ile beslenmelerini de sağlayacaktır. Aynı girdi ile beslenen tüm sorunsuz birimler aynı sonucu üreteceğine ve sadece sorunsuz birimlerin ürettiği sonuç göz önüne alınacağına göre, verilen karar oy birliği ile oluşacaktır. Bu mantık tersine yürütülürse oy birliğinin sağlanamadığı her durum da—sistem karar almada bir sorun yaşamasa bile—sistemde en az bir Bizans biriminin varlığının ispatı olarak görülecektir.

Lamport, Shostak ve Pease algoritmasının işleyebilmesi için, sistemin üç şartı gerçeklediğini kabul etmekteyiz:

1. Sistemdeki tüm birimler birbirleri ile haberleşebilir.
2. Bir birim aldığı mesajın kimden geldiğini bilir.
3. Bir birimin mesaj göndermediği anlaşılabilir.

İlk adım sistemdeki tüm birimlerin aynı girdi ile beslenip beslenmediğinin kontrol edilmesidir. Bunun için her bir birim, diğer tüm birimlere besleme ünitesinden gelen girdi değerini yollar. Bekleneceği üzere, Bizans birimleri her bir birime farklı bir değer yollayabilirler. Her birim, besleme ünitesinden kendine gelen girdiyi ve diğer $(N-1)$ birimden gelen girdi değerlerini listeler ve çoğunluğu oluşturan girdiyi gerçek girdi olarak kullanır. Örneğin; bir birim, kendisi A girdisi ile beslenmesine karşın, diğer tüm birimlerden girdi değeri B olarak gelirse kendisi de algoritmayı B değeri ile işletir. Girdi değerlerinin paylaşımının ardından her bir birim doğru olduğunu kabul ettiği girdi değerine göre bir sonuç üretir. Eğer tüm birimlerin ürettiği sonuç aynı değilse; yani, sonuçlarda bir oy birliği oluşmuyorsa birimlerden en az biri bir Bizans birimidir. Aksi halde oy birliğinin oluşmaması mümkün değildir; çünkü, teorik

olarak her birim girdi değerini aynı listeye bakarak bulmaktadır. Eğer oy birliği oluşmuyorsa ya birimlerin elindeki liste aynı değildir ve dolayısıyla, en az bir Bizans birimi herkese ayrı bir değer yollamaktadır; ya da birimlerin kendileri Bizans birimi olarak girdi ile sonuç arasındaki bağı bozuyorlardır. Her iki durum da en az bir Bizans biriminin varlığını gösterir. Eğer oy birliği sağlandıysa sistem nihai sonuca ulaşmış olduğundan algoritma burada sona erecektir.

Oy birliği sağlanamıyorsa her birim diğer birimlere elindeki listeyi gönderir. İki listenin farklı olması demek, en az bir değerde birbirlerinden ayrılıyor olmaları demektir. Buna göre, bir birim eline geçen listeleri birbirleri ile karşılaştırarak en az bir birimin, farklı birimlere farklı değerler gönderdiğini tespit eder. Bu birim bir Bizans birimidir ve hem gönderdiği girdiler dikkate alınmaz hem de oy birliğinin oluşmasında hesaba katılmaz. Bu birim ayıklandıktan sonra oy birliğinin oluşması mümkündür; ama yine oluşmuyorsa bir Bizans birimi daha var demektir. Algoritma bir tur daha döndürülerek diğer Bizans birimi de ayıklanır. Bu yönteme göre, algoritma en fazla m tur işletilerek tüm Bizans birimleri ayıklanmış olacaktır.

6. İki tane uygulamalı örnek

Bu bölümde Lamport—Shostak—Pease algoritmasını, bir tane $3m$, bir tane de $(3m + 1)$ birimli sisteme çeşitli seçenekleri göz önüne alarak uygulayacağız.

İlk örneğimiz 4 birimli bir sistem. **Lamport, Shostak ve Pease** algoritmasına göre,

$$3m + 1 = 4 \quad \Rightarrow \quad m = 1$$

olacağından dolayı bu sistem tek bir Bizans birimi ile baş edebilir. Bu örneği özellikle basit tutmamızın nedeni, durumu birkaç şekilde irdeleyecek olmamız. İlk önce durumu komutansız Bizans Generalleri metaforuna göre inceleyelim. Bu kurguya göre, dört general kendilerine göre bir durum değerlendirmesi ve harekât planı yaparlar. Sadık olan üç generalden ikisi saldırı, biri ise geri çekilme yönünde bir değerlendirme yapmıştır. Dördüncü general bir hain olduğu için ne karar verdiğinin bir önemi yoktur; çünkü kararını, orduyu bozguna uğratmak amacıyla her an değiştirebilir. Algoritmanın çalışma performansını, doğal olarak, en kötü senaryo üzerinden kontrol etmemiz gerekir. Aşağıdaki tabloda, generallerin kendi yaptıkları değerlendirme sonucunda vardıkları kararlar görülmektedir.

	1	2	3	4
Alınan kararlar	S	Ç	S	

Bu noktada algoritmanın ilk adımı işletilir ve her general diğerlerine kendi kararını iletir. Sadık generallerin iletileri alt alta yazılırsa

	1	2	3	4
Alınan kararlar	S	Ç	S	
1	×	S	S	S
2	Ç	×	Ç	Ç
3	S	S	×	S
4	X	Y	Z	×

tablosu elde edilir. Bu tabloya göre, dördüncü general, 1, 2 ve 3 numaralı generallere sırasıyla X, Y ve Z kararlarını iletmiştir. Birinci generalin elindeki liste; yani, tablodaki birinci kolon (S,Ç,S,X) şeklindedir. Eğer X=Ç olursa; yani, dördüncü general birinci generale kararının çekilme yönünde olduğunu bildirirse birinci general ne yapacağı konusunda kararsız kalacaktır. Benzer şekilde Y=Z=Ç olursa diğer iki general de kararsız duruma düşeceklerdir. Ancak bu durumda, dördüncü general bir Bizans birimi gibi davranmamış olur: Herkese aynı kararı yolladığına göre, kendi kararı çekilme yönündedir.

Dördüncü generalin bir Bizans birimi olarak değerlendirilebilmesi için, kendinin karasız kalmaması ve saldırıya geçmesi veya çekilmesi gerekir. Bu durumda bir Bizans birimi olduğu anlaşılacaktır; fakat (biz en kötü durum senaryosunu işlemekte olduğumuz için) kendini saklamak isteyen general böyle bir şeyi asla yapmayacaktır. Dolayısıyla, bu generalin bir Bizans birimi olarak değerlendirilebilmesi için farklı generallere farklı mesajlar göndermesi gerekir. (Sistemin dört sadık generalle kararsız kalması ayrı bir sorundur.) Bunun için X=Y=Ç, Z=S olduğunu düşünelim. Bu durumda üç sadık generalin ellerindeki listeler, sırasıyla, (S,Ç,S,Ç), (Ç,S,S,Ç), (S,S,Ç,S) şeklinde olacaktır. Bu durumda, üçüncü general saldırı kararına varır ki, dört generalin oy birliğinde olmamaları bir Bizans biriminin varlığını gösterir. Bu noktada bütün generaller ellerindeki listeleri birbirlerine gönderirler. Hain general kendi gönderdiği listede değişiklik yaparak durumu manipüle etmeye çalışabilir; fakat sadık generallerin listeleri karşılaştırıldığında kimin Bizans birimi olduğu anlaşılır. Dördüncü general devre dışı bırakılırsa üç sadık general saldırı yönünde oy birliğiyle anlaşacaklardır.

Şimdi aynı örneği farklı bir sunumla inceleyelim. Generallerden bir tanesi komutandır ve hain olan general de odur. Bu durumda, komutanın emirlerinden oluşan tablo aşağıdaki gibidir.

	1	2	3
Komutanın emri	S	Ç	S

Generaller aldıkları emri birbirleriyle paylaşırlarsa generallerin ellerindeki listeler

	1	2	3
Komutanın emri	S	Ç	S
1	×	S	S
2	Ç	×	Ç
3	S	S	×

şeklinde oluşacaktır. Bu listelere göre, generaller bir oy birliğine varabilirler. Eğer komutan sadık ama generallerden biri (üç numaralı general) hain olursa listeler şu şekilde oluşur:

	1	2	3
Komutanın emri	S	S	S
1	×	S	S
2	S	×	S
3	X	Y	×

Bu listelere göre de, X ve Y değerleri ne olursa olsun sadık generaller oy birliğine varabilir. Üçüncü general saldırıya katılmazsa Bizans birimi olduğu ortaya çıkacaktır. Eğer algoritma bir adım öteye taşınırsa 3 numaralı generalin Bizans birimi olduğu gene anlaşılacaktır.

Görüldüğü gibi, algoritmanın yardımıyla, dört birimli bir sistem tek sorunlu birimi tolere edebilmektedir.

İkinci örneğimiz, **Lamport, Shostak ve Pease** algoritmasına göre iki haini tolere edemeyecek olan 6 birimli bir sistem. Yukarıda anlatılanları, 5 ve 6 numaralı generallerin hain olduklarını kabul ederek hızlıca uygularsak

	1	2	3	4	5	6
Alınan kararlar	S	Ç	S	Ç		
1	×	S	S	S	S	S
2	Ç	×	Ç	Ç	Ç	Ç
3	S	S	×	S	S	S
4	Ç	Ç	Ç	×	Ç	Ç
5	A	B	C	D	×	E
6	U	V	Y	Z	X	×

tablosunu elde ederiz. Eğer, A, B, C, D değerleri Ç; U, V, Y, Z değerleri S şeklinde olursa (veya tam tersi) tüm generaller kararsız kalacaktır. Ancak bu durumda ya hain generallerin kimlikleri hemen tespit edilir (kendileri kararsızlık dışında bir tavır takınırsa) ya da hain generaller sadık bir tutum içinde olurlar (kendileri de kararsız kalırlarsa).

Buna göre, bu sistemdeki kötü senaryo; yani, hainlerin sistemi tıkaması ve kimliklerinin tespit edilememesi her generalin ayrı bir davranışa yönlendirilmesi şeklindedir. Bunun için A, C, U ve Y değerinin S; C, D, V ve Z değerinin de Ç olması yeterlidir (veya tam tersi). Bu durumda 1 ve 3 numaralı generaller saldırıya geçerken 2 ve 4 numaralı generaller çekilmeyi seçeceklerdir. Algoritma işletildiğinde Bizans birimlerinin kimlikleri tespit edilebilir; fakat kararsızlık hali aşılamaz.

İlginçtir ki, generallerden biri komutan olduğunda kurgu dramatik bir şekilde değişir. Eğer komutanın sadık, 4 ve 5 numaralı generallerin hain olduğunu düşünürsek tablomuz

	1	2	3	4	5
Komutanın emri	S	S	S	S	S
1	×	S	S	S	S
2	S	×	S	S	S
3	S	S	×	S	S
4	A	B	C	×	D
5	U	V	Y	Z	×

şeklinde olacaktır. Tablodan görüldüğü üzere, hain generallerin sistemi manipüle etme ihtimalleri yoktur. Sadık olan 1, 2 ve 3 numaralı generaller, algoritmayı işleterek Bizans birimlerini tespit ettikten sonra oy birliği ile saldırı kararı alıp saldırıya geçeceklerdir. Bu noktada, **Lamport, Shostak ve Pease** algoritmasının $3m + 1$ şeklinde özetlediği sonuç yanlışlanmış gibi durmaktadır: 6 birimli sistemdeki iki sorunlu birim sonuca etki edememektedir; sistem iki sorunlu birimle başa çıkabilmektedir. Ancak Lamport, Shostak ve Pease adı geçen belgede bu durumu şu şekilde işaret etmektedirler:

> **Bizans Generalleri Problemi:** *Bir komutan emrindeki (n-1) generale emir yolladığında;*
> *1. Tüm sadık generaller aynı emri uygular,*
> *2. Eğer komutan sadıksa tüm sadık generaller aynı emri uygular.*

Görüldüğü gibi, problemin yapısı gereği, eğer komutan sadıksa tüm sadık generaller aynı emri uygulayacaktır. Dolayısıyla sorun komutanın sadık olmadığı durumlarda çıkmaktadır—sorun, besleme ünitesinden aynı değeri okuma sorunudur. Dikkat edilirse yukarıdaki örnek, aynı girdi değeri ile beslenen bir sistem örneğine dönüşmüştür ve 5 birimli bir sistemde, çoğunlukta olan (3 tane) sorunsuz birimler, sistemin kararlı işlemesi için yeterlidir.

Komutan sadık değilse her generale farklı bir emir göndereceği için liste tablosu

	1	2	3	4	5
Komutanın emri	S	Ç	S	Ç	S
1	×	S	S	S	S
2	Ç	×	Ç	Ç	Ç
3	S	S	×	S	S
4	Ç	Ç	Ç	×	Ç
5	U	V	Y	Z	×

şeklini alır. 5 numaralı general haindir. Algoritmanın işletilmesi Bizans biriminin kimliğini ortaya çıkarır; fakat sistemdeki kararsızlık yine ortadan kaldırılamaz.

Verdiğimiz iki örnekten görülmektedir ki, sistemdeki sorunlu birim

sayısı m olmak ve besleme ünitesi de bir birim sayılmak üzere $3m + 1$ ve $3m + 2$ birimli sistemlerde, Lamport, Shostak ve Pease algoritması uygulanarak sistemin kararlı ve sorunsuz çalışması sağlanabilmektedir. Ancak birim sayısı $3m$ olduğu takdirde durum değişmektedir. Eğer besleme ünitesi sorunlu değilse sistem bazı durumlarda çökmeden çalışmaya devam edebilir. Fakat besleme ünitesi sorunluysa bu asla mümkün olmaz ve sistem kararsızlık halinde kalır.

Yukarıda gerçekleştirdiğimiz anlatımda birimler arasındaki iletişimin dijital imza gibi kriptolojik uygulamalar kullanılmadan yapıldığını farz ettik. Dijital imza kullanımı, sorunlu birimlerin, diğer birimlerden gelen mesajları başka birimlere gönderirken değiştirmelerini engelleyecektir. Bu da algoritmanın daha az çalışarak sonuca varmasını sağlar.

Bizans Generalleri Problemi'nin çözümüne dair verilmiş olan tek algoritma Lamport, Shostak ve Pease algoritması değildir. Daha pek çok algoritma sorunun çözümüne dair yöntemler sunmaktadır. Ancak bu algoritmalardan büyük bir kısmı, problemin genel çözümüne dair kayda değer bir ilerleme sunmadıkları gibi, protokol hızı açısından da elverişlilikleri tartışmalıdır. Diğer birçok ağ problemi gibi, Bizans Generalleri Problemi'nin de temeli çizge teorisine (graph theory) dayanır ve matematiksel incelemesi genellikle bu konuya değinilerek yapılır.

Bizans Generalleri Problemi ile ilgili çalışmalar hâlen sürmekte olup aslında çok geniş bir alana yayılmış olan bu problemin tüm detaylarına, bu hacimdeki bir yazıda değinmek olanaksızdır. Burada bahsetmediğimiz detayları yazının sonunda verilen kaynaklarda bulmak mümkündür.

Yararlanılan Kaynaklar

1. **The Byzantine Generals Problem**
   Leslie Lamport, Robert Shostak, and Marshall Pease
   ACM Transactions on Programming Languages and Systems
   Volume 4 Issue 3, July 1982 — Pages 382-401
   https://people.eecs.berkeley.edu/ luca/cs174/byzantine.pdf

2. **Practical Byzantine Fault Tolerance**
   Miguel Castro and Barbara Liskov
   Proceedings of the Third Symposium on Operating Systems Design
   and Implementation, New Orleans, USA, February 1999 — Pages 173-186
   http://pmg.csail.mit.edu/papers/osdi99.pdf

3. **Easy Impossibility Proofs for Distributed Consensus Problems**
   Michael J. Fischer, Nancy A. Lynch, and Michael Merritt
   Distributed Computing
   Volume 1 Issue 1, January 1986 — Pages 26-39
   https://groups.csail.mit.edu/tds/papers/Lynch/FischerLynchMerritt-dc.pdf

4. **The Consensus Problem in Unreliable Distributed Systems (A Brief Survey)**
   Michael J. Fischer
   Proceedings of the 1983 International FCT-Conference on Fundamentals
   of Computation Theory, August 1983 — Pages 127-140
   https://pdfs.semanticscholar.org/38e1/f9b4d1dd4dc153b7205b43c3e65585aff9cd.pdf