BÖLÜM 11

KRİPTOGRAFİK PROTOKOLLER

KRİPTOGRAFİK PROTOKOL NEDİR?

Protokol, basitçe iki veya daha fazla kişi arasındaki önceden belirlenmiş belli bir amaca yönelik haberleşme metodu olarak tanımlanabilir. Kriptografik protokol ise protokol olarak kriptografik bir algoritma içeren bir protokol kastedilir. Ancak genelde amaç temel gizliliğin ötesindedir. Haberleşen taraflar düşman ya da dost olabilirler.

KRİPTOGRAFİK PROTOKOLLERİN ÖZELLİKLERİ

Kriptografik algoritmalarda olduğu gibi bir protokolün de güvensiz olduğunu ispatlamak güvenliliği ispatlamaya göre çok daha kolaydır. Bir protokolü incelerken yine tıpkı algoritmalardaki gibi protokolü nasıl bir cihazda hayata geçireceğimizden çok temel çalışma prensipleri ile ilgilenilir.

HAKEM (arbitrator) ve DÜZENLEYİCİ (adjudicator)

Kimi zaman protokollerde haberleşmenin düzgün bir şekilde işleyebilmesi için hakem olarak adlandırdığımız güvenilir 3. kişilere ihtiyaç duyulur. Hakemlerin en önemli özelliği tüm protokolün onların gözetiminde yürütülüyor olmasıdır. Hakemi çıkardığınız zaman protokol işlemez. Düzenleyici ise hakemin aksine sadece bir anlaşmazlık durumunda başvurulan güvenilir 3. kişilere denir.

DÜZENLEYİCİLİ (adjudicated) PROTOKOLLER

Bu protokoller temelde taraflarin dürüstlüğüne dayanır. Bir anlaşmazlık ya da birisinin hile yapması durumunda düzenleyici kişi bunu farkeder. Denilebilir ki iyi bir düzenleyicili protokolde düzenleyici aynı zamanda hile yapan tarafın kim olduğunu da farkeder. Dolayısı ili bu kişinin varlığı, hile önünde engel teşkil eder.

KENDİ İŞLEYİŞİNİ ZORLAYAN (Self-Enforcing) PROTOKOLLER

Protokol herhangi bir hileye yer bırakmaz. Protokolün sorunsuz işlemesi otomatik olarak herhangi bir hile yapılmadığı anlamına gelir. Hile halinde protokol durur. Devam edilemez bir hal alır. Ve bunu herhangi bir hakem ya da düzenleyiçiye ihtiyaç olmadan sağlayacak şekilde tasarlanmıştır. Bir protokol için her zaman istenen bir özelliktir. Ancak her durumda kendi işleyişini zorlayan bir protokol bulmak mümkün olamayabilir.

AKTİF ve PASİF SALDIRILAR (attacks)

Bir protokol işlerken her zaman saldırı olması olasıdır. Pasif saldırıca kötü amaçlı kişi sadece arada gelip giden trafiği dinleyerek protokol tasarlanırken kendisinin ulaşması umulmayan bir bilgiye erişmeye çalışır. Aktif saldırıda ise kötü niyetli kişi sadece dinlemekle kalmaz. Mesajları ya da kayıtları okumanın ötesinde kesebilir, bozabilir ya değiştirebilir. Aktif saldırgan tamamen dışardan birisi olmak zorunda değildir. Sistemdeki başka legal bir kullanıcı ve hatta sistem yöneticisi olabilir. Saldırganın protokolü uygulayan taraflardan birisi olması durumunda ise daha çok hile ve saldırıyı uygulayan kişi

içinse hilekar tabirleri kullanılır. Saldırganlarda olduğu gibi hilekarları da pasif hilekar ve aktif hilekar olarak ikiye ayırmak mümkündür.

TİPİK SİMETRİK ALGORİTMA HABERLEŞME PROTOKOLÜ

Simetrik kriptografi kullanılarak yapılan tipik bir haberleşmede A ve B kişileri bir kriptosistem ve bir anahtar üzerinde anlaşırlar ve A kişisi bu anahtarı kullanarak şifrelediği mesajını B kişisine gönderir. B kişisi de yine aynı anahtarı kullanarak mesajı deşifre eder ve haberleşme tamamlanır.

Bu sistemin önemli dezavantajları vardır. Öncelikle A ve B kişiler anahtar değiştirmek için biraraya gelmelidirler (anahtar değişimi için başka bir algoritma kullanılmadığını varsayarsak). Araya giren kötü niyetli Z kişisi mesajlara ulaştığı taktirde mesajın diğer tarafa gitmesini engelleyebilir ve hatta anahtarı bilmesi durumunda bu mesajı kendisininkilerle değiştirerek iki haberleşme tarafını da farkında olmaksızın bambaşka mesajlar gönderebilir.

TİPİK AÇIK ANAHTAR KRİPTOSİSTEMİ İLE HABERLEŞME

A ve B kişileri bir açık anahtar kriptosistemi üzerinde anlaşırlar. Simetrik anahtarlı sistemde olduğu gibi burdada kriptosistem üzerinde anlaşma gizli yapılmak zorunda değildir. B kişisi A kişisine kendi açık anahtarını gönderir. A kişisi mesajını B'nin açık anahtarı ile şifreler ve gönderir. B kişisi ise kendi gizli anahtarı ile mesajı çözer ve mesaja ulaşır.

Açık anahtarlı tipik haberleşme sistemlerinin en önemli dezavantajı ortalama olarak simetrik bir algoritmadan 1000 kat yavaş olmalarıdır. Uzun mesajları bu yolla göndermek

pratik değildir.

TİPİK KARMA (hybrid) KRİPTOSİSTEMLER

B, A'ya açık anahtarını gönderirir. A bir oturumda kullanılmak üzere rastgele bir oturum anahtarı üretir ve B'nin açık anahtarı ile şifreleyerek B'ye gönderir. B kişisi ise gizli anahtarı aracılığıyla edindiği oturum anahtarını açar ve daha sonra bu anahtarı belirledikleri simetrik kriptosistemde kullanarak haberleşmelerini sağlarlar.

Tahmin edileceği üzere bu sistemde taraflar, simetrik kriptosistem anahtarı belirlemek için biraraya gelmek zorunda kalmamaktadırlar.

HASH FONKSİYONLARI

Hash fonksiyonlarının kriptografide birazdan bir miktar inceleyeceğimiz üzere çok geniş bir kullanım alanı vardır. Hash fonksiyonları, öncelikle tek-yönlü (one-way) fonksiyonlardır. Yani bir verinin fonksiyon altında görüntüsünü hesaplamak kolaydır ama görüntüden fonksiyonun tersi aracılığıyla ana veriyi elde etmek hesaplama gücü anlamında zordur. Hash fonksiyonları ile elde ettiğimiz değer, yani hash değeri, fonksiyon girdisinin değişken boyuta sahip olmasına karşın sabit boyuta sahiptir ve genelde hash değeri, girdiye göre çok daha ufak boyuttadır. Kullanılan Hash fonksiyonlarından birbirine çok benzer girdi değerlerini için dahi çok farklı çıktılar üretmesi beklenir. Ve yine önemli bir özellik olarak hash fonksiyonlarından çakışmasız (collision-free) olmaları umulur. Yani hash fonksiyonumuz altında aynı hash değerini veren iki girdinin bulunması hesaplama gücü göze alındığında çok zor olmalıdır.

Yukarda bahsedilen özellikler ışığında hash değerleri, verilerin parmak izi olarak düşünülebilir. Dolayısıyla birisinde olan bir dökümanın sizde de olduğunu dökümanı o kişiye göndermeden ispat etmek isterseniz hash değerini ona söyleme yolunu kullanabilirsiniz. Nitekim dokümanın elinizde gerçekten olmaması durumunda o dökümana ait hash değerini üretmeniz çok zordur. En çok kullanılan hash fonksiyonları arasında SHA ve MD5 algoritmaları sayılabilir.

GÜNLÜK HAYATTA İMZA

Günlük hayatta sıkça kullandığımız imzalarda aradığımız bizim için çok önemli özellikleri gözden geçirelim;

1) İmza otentiktir(authentic), imzalayanın kimliğini gösterir

2) İmza çoğaltılamaz.

3) İmza tekrar kullanılamaz.

4) İmza değiştirilemez. Üzerinde oynama yapılamaz.

5) İmza atan kişi attıktan sonra imzasını inkar edemez.

AÇIK ANAHTAR KRİPTOSİSTEMLERLE DİJİTAL İMZA

Tipik bir açık anahtar kriptosisteminde A kişisi kendi gizli anahtarı ile imzalamak istediği dökümanı şifreler. B kişisi ise A'nın açık anahtarı ile dökümanı açar. Eğer açamazsa imza geçerli değildir, imza birisi tarafından ya da doğal etkenlerle bozulmuştur.

Bu basit imza protokolüne kötü taraftan bakacak olursak imzalı dökümanı alan kişi dijital ortamda bu imzalı dökümanı dilediği kadar çoğaltabilir. İmzalı döküman bir anlaşma

metni ise bu elbette ciddi bir sorun teşkil etmeyebilir ancak dökümanın para kaşılığı olan bir çek olduğunu kabul edersek durum farklı olabilir. B kişisinin farklı zamanlarda farklı çeklermiş gibi aynı imzalı çeki bozdurmasını istemeyiz. Bu gibi tekrar kullanımları önlemek için dijital dökümanlar çoğu zaman hangi tarihe ait olduklarını gösteren bir zamanpulu (timestamp) ile beraber kullanılır. Mesaj algoritma ile imzalanıp kapatılmadan önce içine tarih bilgileri de eklenir.

AÇIK ANAHTAR KRİPTOSİSTEM VE HASH FONKSİYONUYLA İMZA

A kişisi bu defa dökümanın kendisini (zamanpulu ile veya zamanpulsuz) imzalamak yerine dökümanın bir hash fonksiyonu sonucu üretilmiş hash değerini imzalar. Ve A kişisi imzalamak üzere kullandığı dökümanla beraber imzalanmış hash değerini B'ye gönderir. Alıcı olan B ise imzayı A'nın açık anahtarı ile açtıktan sonra dökümanın hashini kendisi hesaplar ve bu değerin gönderilen hash değeri ile uyup uymadığına bakar. Eğer uyuyorsa imza geçerlidir. Aksi halde geçersizdir.

Bu protokol dökümanın kendisi yerine sadece hash değerini imzaladığımız için dökümanın boyutuna bağlı olarak bir öncekine göre inanılmaz ölçüde hızlı olabilir. İki farklı dökümanın hash değerlerinin örneğin 160-bit'lik bir çıktı üreten bir hash fonksiyonunda $1/2^{160}$ olduğu düşünülürse hash değerini imzalamakla dökümanın kendisini imzalamanın rahatlıkla eşleştirilebileceği ortaya çıkar. Ancak hash fonksiyonunun temel özellikleri sağlaması gerektiği unutulmamalıdır. Şayet tek-yönlü olmayan bir hash fonksiyonu kullanılsaydı, bir döküman imzalandığı zaman onunla aynı hash değerini veren tüm dökümanlar da beraberinde aynı kişi tarafından imzalanmış olacaktı.

Bu metodla A kişisi bir dökümanı açığa çıkarmadan dökümana sahip olduğunu ispatlaya-

bilir. Örneğin 'copyright' hakkına sahip olmak isteyeceğimiz bir dökümanı dökümanın kendisini açığa çıkamadan imzalayabiliriz.

DİJİTAL İMZALARDA İNKAR EDEMEMEZLİK (nonrepudiation)

Yukardaki algoritmanın eksik bir yanı, belgeyi imzalayan kişiye her zaman hile olanağı vermesidir. Şöyle ki, eğer dökümanda bir zamanpulu mevcut değilse A kişisi dökümanı imzaladığını inkar etmek istediği bir durumda her zaman için gizli anahtarını herhangi bir dijital ortamda bırakarak daha sonra da gizli anahtarının çalındığını ve dökümanın kendisi değil de imzasını çalan kişi tarafından imzalandığını öne sürebilir. Bu gibi durumlarda çoğu zaman 3. güvenilir kişi olan hakemlerin kontrolünde olan bir protokole ihtiyaç duyulabilir.

İNKAR EDİLEMEYEN DİJİTAL İMZALAR

Protokolümüz bu defa bir parça daha karışık ve de 3. güvenilir T kişisine dayanıyor. Şöyle ki; A kişisi mesajı imzalar ve mesaja kendisini tanımlayan bir başlık ekler. Elde ettiği bu yeni birleştirilmiş mesajı tekrar imzalar ve T kişisine gönderir. 3. kişide herkesin açık ve gizli anahtarı bulunmaktadır dolayısıyla dıştaki mesajı A kişisinin açık anahtarı ile açarak A kişisinin kendisini tanımladığı eklentiye erişir. T daha sonra mesajı gerçekten a kişisinin gönderdiğini B'ye onaylamak için mesajın ne zamana ait olduğu bilgisini de ekleyerek imzalar ve hem A'ya, hem de B'ye gönderir. B kişisi T'nin imzasını açar ve A'nın kimlik bilgilerine ulaşarak imzayı onaylar. İmzanın A'ya da gönderiliyor olmasının sebebi bir anlamda A'ya "Senin adına şöyle bir mesaj gönderildi, şayet bunu gerçekten

gönderen sen değil de (örneğin anahtarını çalan) bir başkası ise acele konuş, daha sonra bunu inkar edemezsin" denmesidir.

Bahsedilebilecek bir nokta da şudur ki bu protokolde zaman bilgileri açısından T'ye güvenilmektedir. Dolayısı ile T ile bir şekilde anlaşan birisi dökümanının tarihi hakkında hileye girişebilir. Protokolde B'nin A'da mesajı aldıktan sonra gerçekten ona dair olup olmadığını onaylaması için T'ye sorması gibi değişiklikler yapılması suretiyle alternatifleri de türetilebilir.

ŞİFRELEME ve DİJİTAL İMZA

A ve B kişilerinin her ikisinin de açık ve gizli anahtarlarının bulunduğu bir ortamda bir mesajı hem şifrelemek ve hem de imzalamak istediğimiz bir durumda A kişisi, mesajı, basitçe söylemek gerekirse önce kendi gizli anahtarı ile imzalar ve daha sonra da imzalanmış mesajı B'nin açık anahtarı ile şifreleyerek B'ye gönderir. B de mesajı kendi kapalı anahtarı ile deşifre eder ve daha sonra da A'nın açık anahtarı ile imzaıyı onaylar. Şayet mesajı sorunsuz bir şekilde aldığını A'ya iletmek isterse A'nın yaptığı işin tersini yaparak mesajı tekrar A'ya gönderebilir. Açıkça ifade etmek gerekirse, B kişisi mesajı kendi gizli anahtarı ile mesajı imzalayıp daha sonra da A'nın açık anahtarı ile imzayı şifreleyerek mesajı A'ya gönderebilir. A'nınsa geri ters operasyonlarla gönderdiği orjinal mesajı elde etmesi durumunda işlemlerin başarıyla gerçekleştiği kabul edilir.

Bu operasyonlarda mesajın şifrelenmeden önce imzalanması doğal gözükmektedir. Tersi bir sıra uygulanması durumunda tıpkı kağıt yerine zarfı imzalayan birisinin, bir başkasına zarfın içinden kağıdı çıkararak başka bir kağıt koyup, koyduğu kağıdı imzalanmış göstermesi gibi bir imkan verdiği açıktır.

Burada dikkat edilmesi gereken bir nokta da şudur ki bu protokol de bir takım ataklara maruz kalabilir. Örneğin başkalarından gelen rasgele mesajların imzalanmaması gerekir. Aynı şekilde başkalarından gelen rasgele mesajların deşifre edilip sonucun başkalarına verilmesi de direkt olarak saldırıya meydan verebilmektedir. Ancak genel olarak şifreleme ve imzalama için farklı algoritmalar kullanıldığı taktirde ya da aynı algoritma kullanılsa bile bu iki operasyon için farklı anahtarlar kullanıldığı zaman bu gibi basit bir saldıyıa meydan verilmediği söylenebilir. Benzer şekilde zamanpulu eklentisi de mesajları farklılaştırdığı için hash fonksiyonlarını burada kullanmak saldırılara karşı iyi bir çözüm olabilmektedir. Protokol, değişik amaçlar ve ortamlar doğrultusunda şekillendirilebilir.

ANAHTAR DEĞİŞİMİ (Key Exchange)

Kriptografide en temel uygulamalardan birisi anahtar değişimidir. Standart şifreli haberleşme için, karşılıklı anahtar değişimi başarı ile gerçekleşmişse, çoğu zaman herhangi bir güvenilir 128-bit (yada daha geniş) blok şifre kullanılarak iletişim sağlanabilir. Ancak değinildiği gibi öncelikli olarak anahtar değişiminin güvenilir bir şekilde gerçekleşmesi gereklidir. Sadece belli bir oturum (yada haberleşme seansı) için kullanılan anahtara *oturum anahtarı* denir.

Anahtar değişimi için simetrik algoritmalar kullanılırsa, güvenilir üçüncü kişinin de yardımıyla anahtar değişimini gerçekleştirmek mümkündür. Ancak burada, ilk aşamada güvenilir 3. bir kişiye ihtiyaç bırakmayan, açık anahtar metoduyla gerçekleştirilen ve çok daha işlevsel olan bir yöntem ele alınacaktır.

AÇIK ANAHTAR METODUYLA ANAHTAR DEĞİŞİMİ

Bu uygulamada A kişisi önce B'nin açık anahtarını temin eder, rastgele oluşturduğu oturum anahtarını B'nin açık anahtarı ile şifreleyerek B'ye gönderir. B kişisi ise kendi gizli anahtarı ile A'nın mesajını açar ve haberleşmelerini gönderilen oturum anahtarını kullanarak, simetrik bir algoritma aracılığı ile yürütürler. Burada açık ve kapalı anahtarlar bir açık anahtar algoritmasına, oturum anahtarı ise bir simetrik şifreleme algoritmasına (DES, SAFER, Vigenere vb) aittir.

Burada pasif bir saldırgan için, yani herhangi bir müdahalede bulunmayan kötü niyetli kişi E için kullanılan açık anahtar algoritmasını kırmaya çalışmaktan başka bir yol görünmemektedir. Ancak aktif saldırgan için aynısı söz konusu değildir. Aktif saldırgan, eğer iki kişi arasındaki mesajlara erişme ve hatta değiştirme imkanına sahipse, B kişisi A'ya açık anahtarını gönderirken araya girip ona kendi açık anahtarını gönderebilir. A kişisi de oturum anahtarını, B'nin anahtarı sandığı bu anahtarla şifreleyerek geri gönderdiğinde ise, kötü niyetli kişi zaten kendi açık anahtarı ile şifrelenmiş bu mesajı rahatlıkla açıp okuyabilir. Dahası E, mesajı B'nin gerçek açık anahtarı ile tekrar şifreleyerek B'ye göndererek onların hiçbirşey olmamışcasına anahtarı kendisinde olan bir simetrik algoritmayla haberleşmelerini sağlayabilir. Aktif saldırgan, hissedilir bir yavaşlamaya sebep olmadığı sürece bu şekilde A ve B kişilerinin haberi bile olmadan onların şifreleyerek gönderdikleri tüm mesajları okuyabilir. Ortadaki 3. kişi tarafından gerçekleştirilen bu saldırıya "ortadaki-adam saldırısı" (man-in-the-middle attack) adı verilir.

Burada bir nokta ise protokolün adım adım işlemesi yerine A kişisi tek bir gönderimle, B'nin açık anahtarı ile şifrelediği oturum anahtarını, o oturum anahtarı ile şifrelediği mesajla birlikte B'ye gönderebilir. B de önce kendi gizli anahtarı ile oturum anahtarını elde eder, sonra da bu anahtarı kullanarak simetrik algoritma ile şifrelenmiş mesajı açarak orjinal mesaja ulaşır.

ARAKİLİT PROTOKOLÜ (Interlock Protocol)

Yukarıda değinilen ortadaki-adam saldırısı, A ve B kişilerinin gerçekten birbirleri ile konuştuklarını onaylama şansları yokken kullanılabilir. Arakilit protokolünde A ve B kişileri birbirlerine karşılıklı olarak açık anahtarlarını gönderirler. Burada aktif saldırgan hala araya girip tarafların açık anahtarlarını kendi belirlediği bir açık anahtar ile değiştirebilir) Ancak sonrasında protokol farklı işler. A kişisi oturum anahtarını (veya herhangi bir mesajı da olabilir) yine B'nin açık anahtarı ile şifreler ancak bu defa şifrelenmiş mesajın kendisi yerine *yarısını* B'ye gönderir. Aynı şekilde B kişisi de kendi şifrelediği mesajın ilk yarısını A'ya gönderir. Ve daha sonra, yine önce A kişisi ve sonra da B kişisi olmak üzere her iki taraf da şifrelediği mesajın kalan yarısını birbirine gönderirler ve iki yarımı birleştirerek kendi gizli anahtarlarıyla mesajın hepsini deşifrelerler.

Burada mesajın ikinci yarısı olmadan ilk yarısı kullanışsızdır. Taraflar da ortadaki adam gibi mesajın ikinci yarısı gelmeden şifreli mesajı deşifre etme şansına sahip değildirler. Burada ortadaki-adamı safdışı bırakan etken nedir? Aktif saldırganın daha önce açık anahtarları kendisininki ile değiştirerek daha sonra şifreli metinlere nasıl eriştiğini hatırlayınız.

Ancak burada açık anahtar ile giden mesaj, mesajın tamamı değil. Dolayısıyla saldırgan, mesajı kendi gizli anahtarı ile açıp B kişisinin açık anahtarı ile tekrar şifreleme şansına sahip değil. Çünkü şifreli mesaj *yarım* olduğu için mesajı herhangi bir şekilde açma/ deşifre etme şansına sahip değil. Peki bu özelliği sağlayan yarım mesaj nedir? Mesajı iki 'yarım'a bölmek için nasıl bir mekanizma kullanılabilir?

Burada kullandiğımız mesajı iki parçaya bölüp gönderme metodundaki parçalardan ilki şifreli mesajın bir hash fonksiyonu ile elde edilmiş bir hash değeri olurken ikincisi ise şifreli mesajın kendisi olabilir örneğin. Ya da gerçekten bir 64-bit blok şifre algoritması ile şifrelenmiş mesajın 32'şer bit uzunluğunda iki yarısı olabilir. Her iki durumda da ikinci mesaj gelmeden deşifreleme yapılamayacaktır. Ve yine her iki durumda da aktif saldırganın 2. şifreli mesaj yarısını da bilmeden mesajı değiştirmesine olanak yoktur. İkinci şifreli mesaj, taraflardan birisi tarafından gönderildiğinde ise çoktan mesajın onay için kullanılacak olan ilk yarısı diğer tarafın eline ulaşmıştır. Dolayısıyla ortadaki adam için saldırı vakti çoktan geçmiştir. Nitekim protokolün işleyişi bunu gerektirmektedir. Arakilit protokolü Rivest ve Shamir tarafından geliştirilmiştir.

DİJİTAL İMZALARLA ANAHTAR DEĞİŞİMİ

Şu ana kadar mesajımızın kendisini ya da mesajımızı iletmede kullanacağımız simetrik algoritmaya ait anahtarımızı göndermekte kullandığımız metodlarda açık anahtarımızı gönderirken aktif bir saldırganın her zaman için gerçek dünya uygulamaları ile son derece örtüşen bir ihtimal olan araya girip tarafların açık anahtarlarını değiştirmek suretiyle

çeşitli saldırılarda bulunabileceğini kabul ettik. Bu sebeple açık anahtarları dağıtmanın nasıl daha güvenli bir şekilde olabileceği sorusuna cevap olarak 3. güvenilir T kişisinin imzasını da protokole dahil edelim. Şöyle ki; basit bir ifade ile, A ve B kişileri anahtarlarını direkt olarak birbirlerine göndermeleri yerine, 3. güvenilir T kişisi onlar için anahtar belirleyip bu anahtarı imzalayabilir. Ve bu şekilde A ve kişileri sadece T kişisinden imzalı anahtarlar kullanmak üzere karşılıklı anlaşmışlardır. Ve T kişisinin güvenli olduğunu, yani onun anahtar veri tabanına kimse tarafından erişilemediğini kabul edersek (ki bu belli ve tek bir anahtar dağıtıcısının güvenliği sağlamak ayrı ayrı pek çok kullanıcının güvenliğini sağlamaktan her zaman için daha kolaydır) aktif saldırgan araya girip açık anahtarları kendi anahtarı ile değiştiremez.

Bu arada her zaman için protokol detayları üzerinde düzenlemeler yaparak protokolde faydalı değişiklikler yapabilmemizin olası olduğunu unutmayalım. Örneğin yukarda açık ve gizli anahtarlarımızı ilk aşamada bizim için T kişisinin değil de kendimizin ürettiğini ve T kişisinin sadece bizim ürettiğimiz anahtarların imzası için protokolde yer aldığını düşünürsek, anahtarı göndereceğimiz B kişisi anahtarımızda her zaman T'nin imzasını arayacağı için ve aktif saldırgan da dijital imza metodlarının güvenilirliği ölçüsünde T kişisinin imzasını değiştiremeyeği için açık anahtar gönderimlerimizi çok daha güvenilir bir şekilde yürütebiliriz. Üstelik bu durumda aktif saldırgan T'nin dijital anahtarını ele geçirse bile bu anahtarı A ve B taraflarının mesajlarını okumakta kullanamaz, sadece yeni açık anahtarlar imzalayabilir. Çünkü T'de sadece bizim açık anahtarlarımızı imzalamada kullandığı bir dijital imza anahtarı bulunmaktadır.

OTENTİKASYON(kimlik doğrulama)

Bir bilgisayara kendimizi tanıtmak için kullandığımız yegane metod parola kullanmaktır. Ancak ilk kez Needham ve Guy ikilisinin ortaya koyduğu şekliyle otentikasyon yapan bilgisayarda parolaların kendilerinin bulunmasına gerek yoktur. Dolayısıyla sistem yöneticisinin (ya da sizin parolalarınıza direkt erişme şansına sahip birisinin) parolanızı farklı amaçlar için kullanmasını (örneğin aynı parolayı kullandığınız başka bir sisteme sizin adınıza erişmesini) önemli ölçüde zorlaştırmaya yol açan gerçek şudur ki, bilgisayar için gerekli olan içinde tüm kullanıcı parolalarının bulunması değil, sadece yanlış girilen parolaları diğerlerinden ayırt edebilmesidir. Bu da hash fonksiyonları sayesinde gerçekleşmektedir, şöyle ki; kullanıcı otentikasyon yapılacak bilgisayara parolasını girer, bilgisayar bu parolanın hash değerini hesaplar ve kendi veri tabanındaki aynı kullanıcı için tutulan hash değeri ile uyup uymadığını kontrol eder.

Burada otentikasyon yapılan bilgisayara erişimi olan bir saldırgan hala şunu yapabilir; parola olarak kullanılabilecek çok miktarda sözcük üreterek bunların hash değerlerini hesaplar ve veri tabanındakilerle karşılaştırır. Bu metodla yeterli miktarda süreye sahip olduğu taktirde tüm olası kelimeleri deneyerek parolalara ulaşabilir. Bu operasyonu pratik olarak zorlaştırmak için *tuzlama* (salting) işlemi uygulanır. Tuzlama, parolalara hash fonksiyonundan geçirmeden önce rastgele metin eklemektir. Örneğin UNIX sistemlerde 12-bit tuzlama uygulanmaktadır. Tuzlama metoduyla otentikasyon işlemini önemli oranda yavaşlatmaksızın bir sözlük saldırısını çok ciddi biçimde zorlaştırmak mümkündür. Ancak her zaman için kolay tahmin edilebilir parolalar böyle bir saldırıda önce kırılanlar

olacaktır ve tuzlama onları daha güçlü hale getirmemektedir.

AÇIK ANAHTAR KRİPTOGRAFİ İLE OTENTİKASYON

Otentikasyonun çok uzaktaki bir makina tarafından yapıldığı bir ortam düşünelim. Örneğin bizden birkaç ülke uzaklıkta olan bir internet sitesine kendimizi tanıtmak için kullandığımız parolamızın açık bir şekilde onca yolu gitmesi ne kadar sağlıklı olabilir? Bunun için yukardaki basit otentikaston protokolümüzü açık anahtar sistemli otentikaston kullanacak nasıl değiştiririz?

Açık anahtar kullanan otentikaston metodunda Biz parola yerine kendi gizli anahtarımızı bilmekteyiz. Makinada ise bizim açık anahtarımız bulunur. Otentikasyon şöyle işler; makina bize rastgele bir metin gönderir. Biz de gizli anahtarımızla bu metni şifreler ve makinaya geri göndeririz. Makina, bizim açık anahtarımız ile gönderdiğimiz mesajı açtığında kendi gönderdiği rastgele metni bulursa bize sistem erişimi verir. Bu metodun kullanıldığı ortamda, ne otentikasyon makinasının (gizli anahtarımız sadece bizde bulunuyor, makina veritabanında mevcut değil), ne de iletişim yolunun (açık anahtar kriptosisteminin temel özelliğinden dolayı) güvenli olması gerekmemektedir.

SIR PARÇALAMA (secret splitting)

Bir bilgiyi iki kişi arasında paylaştırmak istediğimizi düşünelim. Öyle ki; her ikisi de biraraya gelmeden bilgiye ulaşamasınlar. Paylaştırılacak olan M mesaji için aynı bit uzunluğunda R gibi bir rastgele metin oluşturduğumuzu düşünürsek $M \oplus R = S$ şeklinde

bir xorlama operasyonu ile S metini elde ettiğimizi düşüenlim. Bu durumda bir kişiye R'ı ve de diğer kişiye de S'i verdiğimizi düşünürsek her ikisi de biraraya gelmeden M mesajını elde edemezler. Biraraya geldiklerinde ise $R \oplus S = M$ ile mesajı geri dönüştürebilirler.

Bu basit metodu ikiden fazla kişide uygulamak içinse bir sırrı 4 kişi arasında paylaştırmak istediğimizi düşünelim. Bu defa sır metni ile yine aynı uzunlukta R,S ve T rastgele metinlerine ihtiyacımız olacak. $M \oplus R \oplus S \oplus T = U$ şeklinde bir operasyonla, R,S,T ve U metinlerini farklı 4 kişiye dağıtarak ancak biraraya geldiklerinde tüm sahip oldukları 4 metni xorlayarak M'i elde etmelerine izin verebiliriz. Dahası bu 4 kişiden birisini safdışı bırakmak istediğimizde onun metinini diğerlerine dağıtmamız yeterli olacaktır. Bu sayede o olmadan da bu xor operasyonunu yaparak ana metne ulaşabilirler.

SAKLI İLETİŞİM YOLU (subliminal channel)

Birisiyle haberleşmek istediğiniz, ancak bunun 3. bir başka kişi aracılığıyla gerçekleşmesi dışında mümkün olmadığını bir senaryo düşünelim. Ve bu 3. kişi de mesajın şifreli olmasına izin vermiyor, kendisi mesajınızı iletmeyi ancak o içeriğini okuyabildiği zaman kabul ediyor. İşte böylesi bir durumda bir saklı iletişim yolu metodu kullanımı çözüm olabilir. Yani ilettiğimiz mesajda gizli bir algoritmayla saklanmış, görünenin ardındaki başka bir mesajla hem diğer tarafa mesajımızı ulaştırıp hem de 3. aracı kişinin yalnızca görünürdeki mesajı gönderdiğimizi sanmasına yol açarak gerçekte ne gönderdiğimiz hakkında hiç bir fikri olmamasını sağlayabiliriz.

Çok basit bir saklı iletişim yolu metodu olarak, kurduğumuz cümlelerde tek sayıda kelime olması durumunda 1-bitini, çift sayıda kelime içeren cümleler ile de 0-bitini gönderdiğimizi düşünebiliriz. Saklı iletişim yoluyla, tıpkı steganografide olduğu gibi herhangi bir anahtar bulunmamakta ve mesajın gizliliği tamamen algoritmanın gizliliğine dayanmaktadır. Saklı iletişim yollarına çok çeşitli uygulama alanları bulmak olasıdır. Örneğin dijital imzalama sırasında saklı iletişim yolu metodları ile mesaji bir taraftan imzalarken diğer taraftan imzanın içine "ben bu mesajı baskı altında imzalıyorum" gibi bir ifade saklamak mümkündür.

BİT VADETME (bit commitment)

Şöyle bir senaryo düşünelim; Bir yatırımcıya hisse senedi önereceğiz. Yükselecek olan hisseleri önerebilirsek, karşılığında ondan bir iş alacağız. Ancak önerdiğimiz hisse senetlerini hemen bilmesini istemiyoruz, çünkü o zaman bize işi vermeden o senetlere kendisi para yatırıp bizi yüzüstü bırakabilir. Ancak diğer taraftan, yatırımcı da bizim samimiyetimize inanmak istiyor. Eğer hisse senedi sonuçları belli olduktan sonra ona açıklarsak, sonucu baştan tahmin ettiğimize dair onu ikna etme şansımız kalmaz. Özetle durum şu ki; A kişisi, B kişisine, belli bir tarihte belli bir bilgiyi bildiğini aradan bir süre geçtikten sonra ona ispatlamak istiyor, ve kendisi izin verene kadar bilginin ne olduğuna dair B'ye ipucu vermek istemiyor.

B kişisi rastgele bir R metni üretir ve bunu A'ya gönderir. A'nın belli bir grup bit içerisinden seçimini gösteren bite 'b' dersek A kişisi, simetrik bir algoritma kullanarak

kendi belirlediği bir anahtar ile (R,b) ikisisini şifreler ve geri B'ye gönderir. Ve seçtiği biti açığa çıkarmak istediği gün geldiğinde ise şifrelemede kullandığı anahtarı B'ye vererek onun paketi açmasını sağlar. B kişisi ise kendi ilk başta gönderdiği rastgele R metninin pakette seçim biti ile beraber bulunup bulunmadığını kontrol eder. Eğer protokolde böyle bir rastgele R metni olmasaydı, sözkonusu olan yalnızca bir bit olabileceği için A kişisi son aşamada gönderdiği paketi rahatlıkla istediği bite (veya hatta bit grubuna) deşifre edebilen bir anahtar gönderebilirdi. Ancak paketin içinde fazladan bir rastgele metnin farklı anahtarlı deşifre etmeler için bozulacak olması A kişisini böylesi bir hileden alıkoyar.

SIFIR-BİLGİ İSPAT METODU (zero-knowledge proof)

Sıfır bilgi ispat, kısaca belli bir bilgiye sahip olduğumuzu, o bilginin ne olduğunu açığa çıkarmadan ispatlamamızı sağlayan metodtur. Şöyle ki, ispatlayacağımız problemi onunla eşyapıda (isomorphic) başka bir probleme dönüştürerek problemin kendisi yerine yeni problemi ispatlıyoruz. Ancak dönüştürdüğümüz problemi, öyle bir şekilde seçiyoruz ki bu yeni problemin çözümü orjinal problemin çözümü hakkında bilgi vermiyor. Burada orjinal problem hakkında bilgi vermeyecek sözü ile ifade edilen şu ki; hesapsal olarak karşı tarafın yeni çözümü orjinal problemin çözümüne dönüştürmesi, en az orjinal problemin kendisini çözmesi kadar zor olmalıdır. Ancak diğer taraftan, bizim aslında çözümüne sahip olmadığımız bir probleme eşyapıdaki bir problem için çözüm sahibiymişiz gibi davranmamız olasılığına karşı diğer taraf her seferinde bize şu soruyu soruyor. "Ya bu eşyapıdaki problemin çözümünü bana ver, ya da onun asıl probleme eş yapıda olduğunu göster". Ve eğer iddiamızda haklı isek bu iki isteği de gerçekleştirebiliriz. Ancak bu iki önermeden birini

gerçekten sağlayamıyo olmamız, yani hile yapmamız durumuna yer bırakmamak içinse bu işlemler n kez (karşı taraf ikna oluncaya kadar) tekrarlanır. Öyle ki, gerçekten orjinal problemin cevabını bilmediğimiz bir durumda her defasında sorulan soruya doğru cevabı verme olasılığımız $1/2$ dir. n kez tekrarlanma durumunda ise bu şekilde hile ihtimalimiz $1/n$ e düşer ki yeterli tekrardan sonra ihmal edilebilecek bir olasılıktır. Sıfır-bilgi ispat'ın dayandığı nokta ise bu olasılığın düşüklüğüdür. Dolayısıyla sıfır-bilgi ispatta yeterli miktarda tekrar için mutlak manada bir ispattan çok karşı tarafın iknası söz konusudur.

Sıfır-bilgi ispat metodları ne yazık ki her matematiksel probleme aynı verimde uygulanamamaktadır. Kimi zaman ispat için çözümün bir kısmını açığa çıkarmak zorunda olduğumuz problemler sözkonusudur. Bu yüzden sıfır bilgi ispat metodları, karşı tarafın sorusuna ne kadar sıklıkta doğru cevap verebileceğimiz ölçüsüne göre kusursuz, istatistiksel, hesapsal ya da kullanışsız gibi kategorilere ayrılmıştır.