AÇIK ANAHTARLI KRİPTOGRAFİ

(PUBLIC KEY CRYPTOGRAPHY)

Bu belgede, açık anahtarlı kriptografinin ne olduğundan ve açık anahtarlı kriptografi uygulamalarından bahsedilmektedir. Belgede verilen bilgileri tam olarak anlayabilmek için -şart olmamakla beraber-, geleneksel şifrelemenin mantığı ve geleneksel (simetrik) şifreleme algoritmaları ile ilgili bilgi sahibi olunması faydalı olacaktır. Belgenin son hali, http://wolf.comu.edu.tr/~evreniz/belgeler/pkc/pkc.html adresinden temin edilebilir.
Bu belge, A. Murat EREN, Faruk ESKİCİOĞLU ve S. Serdar YÜKSEL'in 2002 yılındaki çalışmalarının bir kısmından A. Murat EREN tarafından derlenmiştir; belgeyi bu açıklama satırlarını silmemek kaydı ile istediğiniz gibi kullanabilirsiniz.
Belge içerisindeki bilgiler Cryptography and Network Security (William Stallings) isimli eser başta olmak üzere Applied Cryptography (Alfred J. Menezes, Paul C. van Oorschot, A. Vanstone) ve konu ile ilgili çeşitli makalelerden derlenmiştir.
Belge hakkındaki eleştirileriniz, tavsiyelerinizi ve kriptografi hakkındaki sorularınız için meren@comu.edu.tr mail adresini kullanabilirsiniz.

v.0.2 (05/2003) ***A. Murat EREN****, meren@comu.edu.tr, http://wolf.comu.edu.tr/~evreniz/*

İÇİNDEKİLER

AÇIK ANAHTARLI KRİPTOGRAFİ

Açık anahtarlı kriptografinin gelişmesi, bütün kriptografi tarihindeki en büyük devrimdir. Başlangıcından günümüze kadar, bütün kriptografik sistemler, süpstütüsyon ve permütasyon işlemlerinin temel alınmasıyla oluşturuldular. Sadece elle hesaplanabilen algoritmalarla çalışabilme döneminden sonra, şifreleme/deşifreleme yapan rotor makinelerinin ortaya çıkması sonucunda, geleneksel kriptografide büyük bir gelişme kaydedildi. Elektro mekanik rotor, çok fazla inceliklere sahip ve karmaşık kriptografik sistemlerin geliştirilebilmesini sağladı. Mevcut bilgisayarlarla daha karmaşık sistemler tasarlandı ve en tanınanlarından olan -IBM´in- Lucifer girişimi gelişerek DES´i oluşturdu ve DES`i dünyadaki kriptografi teknikleri arasında en yüksek seviyeye getirdi. Rotor makineleri ve DES (Data Encryption Standart), önemli avantajlar sunmalarına rağmen, halen, süpstütüsyon ve permütasyon işlemlerine bağımlıdırlar.

Açık anahtarlı kriptografi, daha önceki gelişmelerden radikal bir kopuştur. Açık anahtarlı kriptografik sistemlerin en önemli noktaları, süpstütüsyon ve permütasyondan çok matematiksel işlevler üzerine temellenmiş olmalarıdır. Daha da önemlisi, açık anahtarlı kriptografi, tek anahtar kullanan simetrik geleneksel şifreleme algoritmalarının tersine, iki ayrı anahtarın asimetrik kullanımını öngörür. Birazdan göreceğimiz gibi, anahtar dağıtımı ve kimlik denetimi gibi gizlilik ve güven gerektiren durumlarda, iki anahtar kullanımı etkili sonuçlar ortaya koymuştur.

İlerlemeden önce, açık anahtarlı şifreleme ile ilgili bazı yaygın, yanlış bilgilerden bahsetmeliyiz. Bu yanlış düşüncelerden birisi, açık anahtarlı şifrelemenin, kriptoanalize karşı geleneksel şifreleme yöntemlerinden daha güvenli olduğudur. Örneğin böyle bir iddia, Gardner´ın meşhur *Scientific America* adlı 1977 yılında yayınladığı makalesinde yapıldı . Aslında, şifrelemenin güvenliği, anahtarın uzunluğuna ve, kırılan şifreli metnin içerdiği hesapsal işlemlerin

karmaşıklığına dayanır. İster geleneksel ister açık anahtarlı şifreleme olsun, kriptonaliz bakış açısına göre birini direğinden üstün tutmak yanlış olur.

Bir ikinci yanlış düşünce de, genel amaçlı kullanım için geliştirilmiş bir teknik olan açık anahtarlı şifrelemenin, geleneksel şifrelemeyi modası geçmiş kıldığıdır. Tam tersine, geleneksel şifrelemeden vazgeçileceği sanısı, açık anahtarlı şifreleme yöntemlerinin, matematiksel fonksiyonlarından dolayı, ihtimal dışı gözüküyor.

Son olarak, açık anahtarlı şifreleme kullanılırken, geleneksel şifrelemenin daha hantal anahtar dağıtım merkezleri ile karşılaştırıldığında, açık anahtarlı sistemlerin anahtar dağıtımının üzerinde kafa yorulması gerekmeyen, sıradan ve basit bir iş olduğuna dair yanlış bir anlayış vardır. Aslında, protokolün bazı biçimleri gereklidir fakat, geleneksel şifreleme yöntemlerinin ihtiyaç duyduğu merkez temsilciler ve prosedürler, açık anahtarlı şifrelemenin ihtiyaç duyduklarından daha basit daha karmaşık ya da daha etkili değildir.

Bu bölüm, açık anahtarlı şifrelemeye genel bir giriş mahiyetinde olacaktır. İlk önce, işin kavramsal çerçevesine bakmaya çalışacağız. Bu noktada açık anahtarlı kriptografi ile ilgili enteresan bir anektodu es geçmek olmaz: açık anahtarlı kriptografinin, pratik olarak uyarlanışı gösterilmeden, tekniğin mimarisi geliştirildi ve doğru kabul edilerek yayınlandı. Kimse pratiğini görmeden teorisi kabul gördü. Daha sonra, açık anahtarlı şifreleme yöntemi için, uygulanabilir olarak gösterilen en önemli şifreleme/deşifreleme algoritması olan, RSA algoritmasını inceleyeceğiz. Daha sonrada, açık anahtarlı sistemler için, anahtar dağıtımını ve anahtar dağıtım yönetimlerini inceleyeceğiz.

Açık anahtarlı kripto sistemlerinin çoğunluğu, sayılar teorisini temel almıştır. Bu bölümde verilen sonuçları algılamak için, sayılar teorisini anlamanıza yada biliyor olmanıza çok da gerek yoktur. Bununla birlikte, açık anahtarlı şifreleme algoritmaları hakkında kesin bir yargıya varmak için, sayılar teorisinin bazı kısımlarını bilmek gerekmektedir.

Açık Anahtarlı Kripto Sistemlerin Prensipleri

Açık anahtarlı şifrelemenin genel amacı, gerçekleştireceği devrim ile geleneksel şifrelemenin en büyük iki problemine çözüm sağlamaktı. Bu problemlerden ilki gizli anahtarların dağıtımıdır. Gizli anahtar derken, geleneksel kriptografi uygulamalarının (DES, IDEA, Blowfish, CAST128, RC5, ...) kullandığı anahtarları kastediyoruz.

Geleneksel şifrelemeden yararlanarak birbirlerine şifrelenmiş metinler gönderecek olan taraflar, şifreleme ve de şifreleme işlemleri için, ya bir şekilde kendilerine ulaştırılmış olan anahtarı kullanacaklar, ya da, bir anahtar dağıtım merkezinden faydalanacaklardır. Açık anahtarlı kriptografinin mucitlerinden birisi olan Whitfield Diffie (diğeri de Stanford Üniversitesinden Martin Hellman`dır), kriptografinin özü olan, iletişimde %100 güvenlik esasını hiçe sayan bir anahtar dağıtım merkezi kullanma gerekliliğini ortadan kaldırdı. Tarafların kullanacakları gizli anahtarları bir anahtar dağıtım yetkilisinden almaları, istediği taktirde üçüncü parti bir kişinin iletişimi anlaşılır kılabileceği tehlikesini barındırmakta idi.

Diffie, üzerinde düşündüğü ikinci problem olan "dijital imza" konusunun, önceki ile ilgisi olmayan başka bir konu olduğunu gördü. Eğer kriptografinin kullanımı, sadece askeri konularda değil, özel ve kâr amaçlı uygulamalarda da kullanılacak kadar yaygın olsaydı, bu durumlar için kullanılacak elektronik belge ve dokümanlarda da, kağıt dokümanlarda kullanılan kişisel imzalara gerek duyulurdu. Ve dijital imzalar sayesinde, bir mesajı kimin gönderdiği kesinlikle bilinmiş olur ve bu da herkesi memnun eden bir metot olurdu.

Diffie ve Hellman, 1976`da, her iki probleme de, daha önceki bütün kriptografik gelişimlerden ve buluşlardan farklı, radikal bir çözüm getiren hayret verici bir buluş gerçekleştirmeyi başardılar.

Az sonra, açık anahtarlı kriptografinin iskeletine göz atacağız. Daha sonra da, bu yöntemin kalbi olan şifreleme/de şifreleme algoritmalarının ihtiyaçlarını göreceğiz.

Açık Anahtarlı Kripto Sistemlerin Karakteristikleri

Açık anahtarlı şifreleme/deşifreleme algoritmaları, şifreleme için bir anahtara, de şifreleme içinse bu anahtarla ilişkisi olan ama bu anahtar olmayan ikinci bir anahtara ihtiyaç duyarlar. Bu durumda bir güvenlik sağlamış olur. Bu algoritmalar şu önemli karakteristiğe sahiptirler:

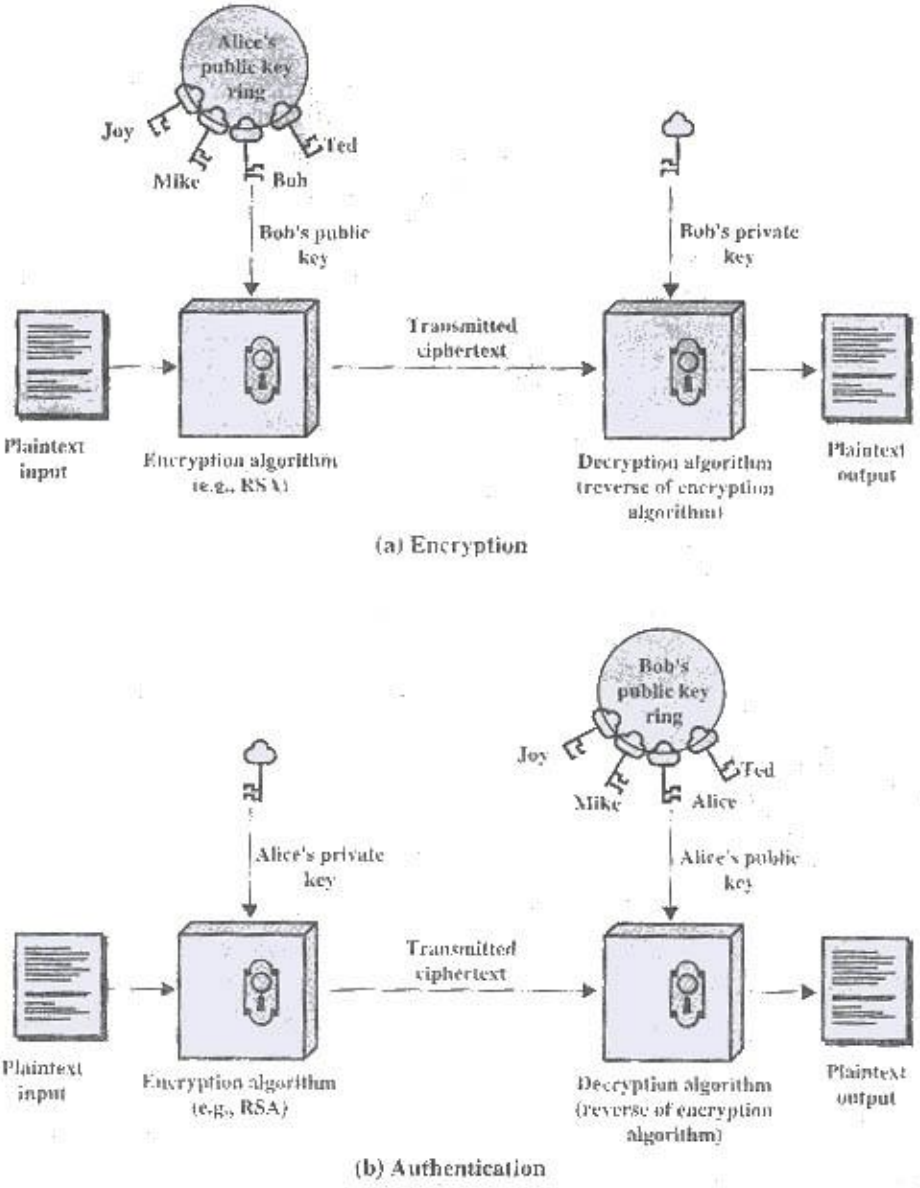
- Sadece kriptografik algoritma ve de şifreleme anahtarı verilmişken, bir takım hesaplamalar yolu ile şifreleme anahtarını bulmak mümkün değildir.

Bununla beraber RSA gibi bazı algoritmalar şu karakteristikleri de gösterirler:

- Her iki benzer anahtar da şifreleme ve de şifreleme için kullanılabilir. Bununla beraber, bir anahtar şifreleme için kullanılmışsa, de şifreleme için diğer anahtar kullanılmalıdır.

Şekil 1(a), açık anahtarlı şifreleme yöntemi gösterilmiştir. Başlıca adımlar şunlardır:

1. Her ağdaki her son sistem, mesaj alındığında şifreleme ve de şifreleme için kullanacak olduğu anahtar parçasını yaratır.
2. Her sistem, şifreleme anahtarını herkesçe erişilebilecek bir dosya yada yazmaç içerisine kaydederek paylaştırır. Bu anahtarın, açık olan kısmıdır (public key). Özel anahtar saklı tutulur.
3. Eğer, A, B`ye bir mesaj yollamak isterse, mesajı B`nin açık anahtarını kullanarak şifreler.
4. B, mesajı aldığında, bu mesajı kendi özel anahtarını kullanarak de şifre eder. Diğer hiçbir alıcı mesajı de şifreleyemez, çünkü mesajı de şife edecek olan özel anahtarı sadece B bilir.



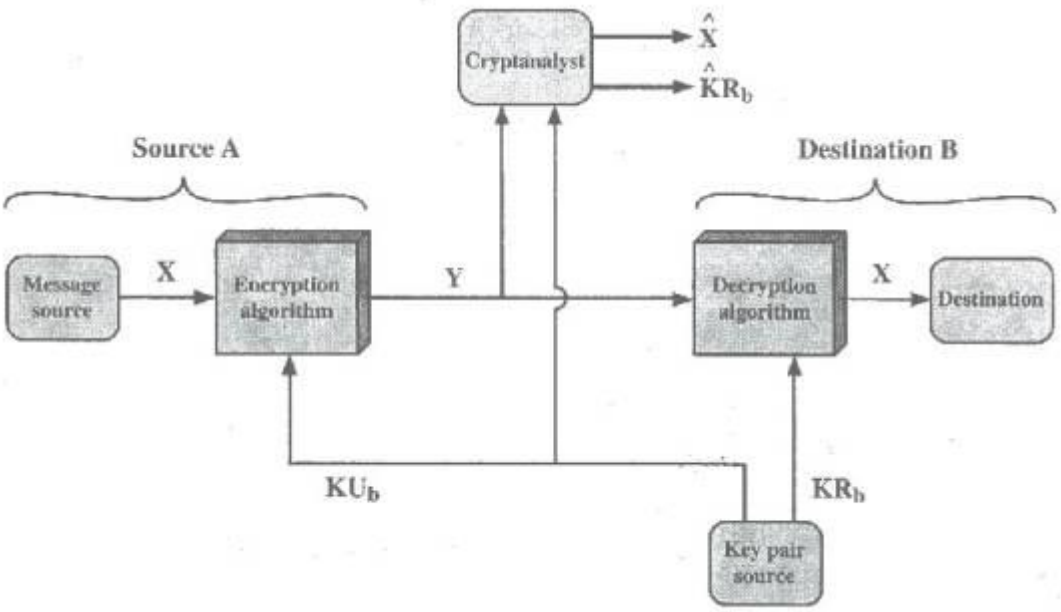
Şekil 1

Bu şekilden de anlaşıldığı üzere her katılımcı, diğerlerinin açık anahtarlarına erişim hakkına sahiptir. Ve katılımcılar özel anahtarlarını lokal olarak yaratırlar. Bu yüzden, özel anahtarların paylaşılmasına gerek yoktur. Herhangi bir sebepten ötürü özel anahtarlar sahipleri tarafından değiştirilmek istenebilirler, bu durumda değişmiş olan yeni açık anahtar ilgili yerlere yeniden gönderilerek eskisi ile yer değiştirilir.

Tablo 1, geleneksel ve açık anahtarlı şifrelemenin farklarını açıkça göstermektedir. Geleneksel şifrelemede kullanılan anahtarı, açık anahtarlı şifrelemede kullanılan anahtarlardan ayırmak için onu **gizli anahtar** olarak anacağız. Açık anahtarlı şifrelemede kullanılan iki anahtarı da, **genel anahtar** ve **özel anahtar** olarak anacağız. Özel anahtar, her zaman gizli tutulacak olan anahtardır, fakat, geleneksel şifrelemedeki gizli anahtarla karışmaması için ona gizli anahtar yerine özel anahtar diyoruz.

Geleneksel şifrelemede:	**Açık anahtarlı şifrelemede:**
Çalışması için:	
1. Şifreleme ve de-şifreleme için aynı algoritma aynı anahtarla birlikte kullanılır.	1. Şifreleme ve de-şifreleme için bir algoritma ve anahtarlardan birisi kullanılır. Şifreleme için kullanılan anahtar, de-şifreleme için kullanılamaz.
2. Gönderen ve alan, algoritmayı ve anahtarı paylaşmalıdır.	2. Gönderen ve alan, ilişkili anahtarlardan birine sahip olmalıdırlar (aynı olanı değil).
Güvenlik için:	
1. Anahtar gizli tutulmalıdır.	1. Anahtarlardan biri gizli tutulmalıdır.
2. Diğer bilgiler saklandığında, mesajı deşifre etmek imkansız olmalıdır.	2. Diğer bilgiler saklandığında, mesajı deşifre etmek imkansız olmalıdır.
3. Algoritma ve şifreli metin örnekleri bilmek, anahtarı çözmek için yetersiz olmalıdır.	3. Algoritma, şifreli metin örnekleri bilmek ve anahtarlardan birine sahip olmak, diğer anahtarı bulmak için yetersiz olmalıdır..

Tablo 1 Geleneksel ve açık anahtarlı şifreleme



Şekil 2

Şimdi, Şekil 2 yardımıyla, açık anahtarlı şifreleme yapısını oluşturan başlıca elementlere bakalım. A, mesaj gönderecek bir kaynak olsun. Ve göndermeyi düşündüğü $X = [X_1, X_2, \cdots, X_M]$, genel bir alfabe kullanarak oluşturduğu, M sonlu sayısında kelimeden oluşan bir mesaj olsun. Bu X mesajı, B alıcısı için tasarlanır. B, birbiri ile ilişkili olan bir anahtar çifti yaratır: bir genel anahtar; KU_b ve bir özel anahtar KR_b. KR_b, yalnız B tarafından bilinir. KU_b ise, A tarafından erişilebilecek olan B`nin açık anahtarı olacaktır.

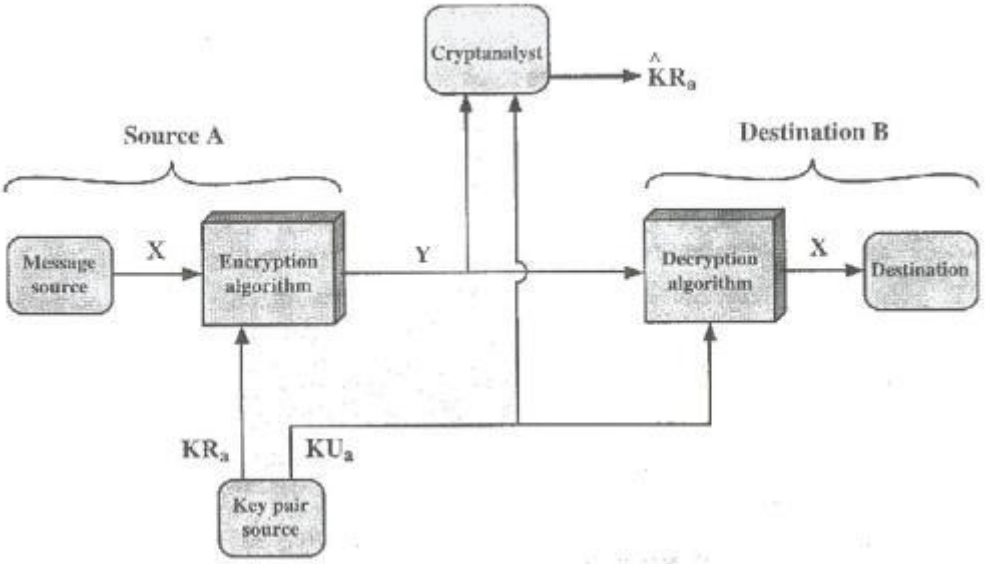
X düz metnini ve KU_b anahtarını girdi olarak alan A, mesajı şifreleyerek, $Y = [Y_1, Y_2, \cdots, Y_N]$ metnine dönüştürür.

$$Y = E_{KUb}(X)$$

Alıcı olan B, özel anahtarın sahibi olarak, şifreli metni düz yazıya aşağıdaki fonksiyon ile çözümleyebilir.

$$X = D_{KRb}(Y)$$

Bir rakip, iletişimi izleyerek şifreli metin Y`yi ele geçirirse, ve $KU_b$`ye sahipse, aynı zamanda, KR_bya da, X için bir erişim hakkına sahip değilse, X ya da KR_b`yi elde etme girişiminde bulunacaktır. Rakibin, şifreleme ve deşifreleme algoritmalarını bildiği varsayılır. Eğer, rakip sadece bu mesaj ile ilgileniyorsa, $\hat{X}$ şifreli metni üzerinde yapacağı hesaplamalarla, bir X düz metni oluşturmaya çabalayacaktır. Bununla beraber, rakibin teşebbüsü genellikle gelecek mesajları da okuyabilmek üzere, tahmini $\hat{KR}_b$üstünde hesaplamalar yaparak KR_b`yi elde etmek olur.



Şekil 3

Her iki benzer anahtardan, yani, şifreleme için kullanılacak olan anahtar ve diğeri de deşifreleme için kullanılmak anahtardan genel olarak bahsetmiş olduk. Umarız, oldukça farklı olan bu kriptografik yapıyı açıklayabilmek için yeterli olmuştur. Şekil 2`de, güvenliğin sağlanması gösterilirken, 2 ve 3`te, açık anahtarlı kriptografinin kimlik doğrulamayı nasıl sağladığı gösterilmiştir:

$$Y = E_{KRa}(X)$$

$$X = D_{KUa}(Y)$$

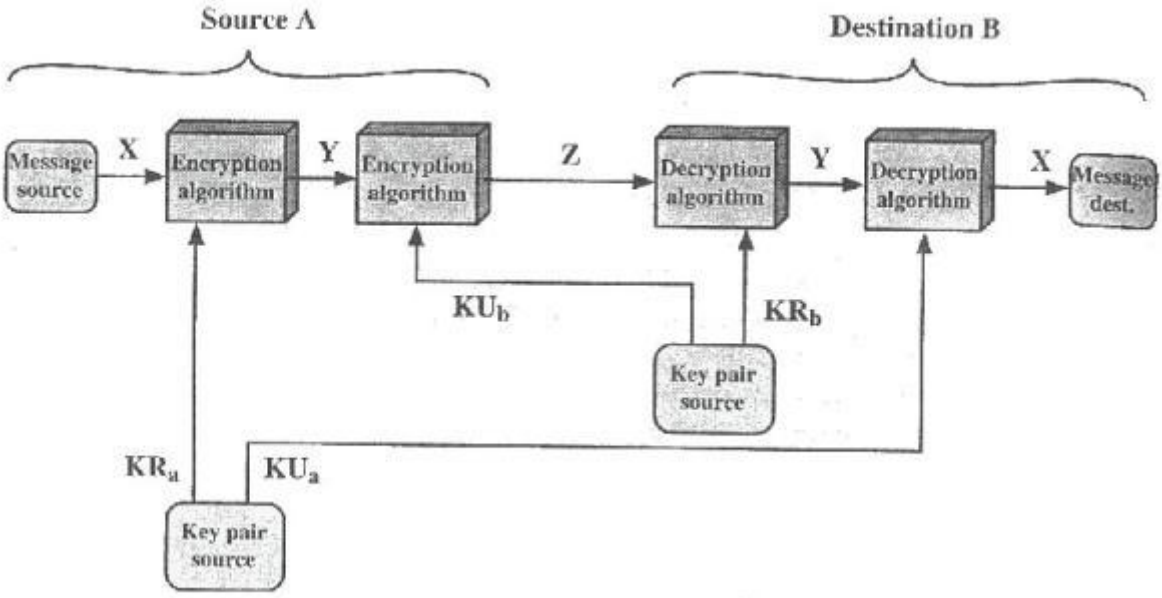
Bu durumda, A, B`ye göndermek üzere bir mesaj hazırlıyor ve göndermeden önce mesajı kendi özel anahtarıyla şifreliyor. B, bu mesajı, -sadece- A`nın genel anahtarını kullanarak deşifre edebilir. Çünkü, A mesajı kendi özel anahtarı ile şifrelemiştir dolayısıyla sadece, A bu mesajı hazırlayabilir. Bu yüzden özel anahtar ile şifrelenmiş tüm mesajlar dijital imza olarak düşünülebilir. Bununla birlikte, A`nın kendi özel anahtarı ile şifrelediği mesajın, A`nın özel anahtarına sahip olmayan bir kişi tarafından değiştirilmesi imkansızdır, dolayısıyla, bu şekilde, hem bütünlük hem de kaynak doğrulama ihtiyaçları karşılanmış olur.

Önceki yapıda, tüm mesaj şifrelenmiş ve gönderici ve mesajın içeriğinin güvenliği sağlanmıştı. Fakat bu, mesajın saklanması ve pratik kullanımı esnasında sorunlara neden olacaktır. Her doküman, pratik kullanım için düz metin halinde saklanmalıdır. Fakat orijinalliğinin ispat edilmesi gereken durumlarda kullanılmak üzere, mesajın şifrelenmiş hali de ayrıca saklanmalıdır. Bu işi başarmanın daha verimli bir yolu olarak, mesajın en önemli olan bitlerinin şifrelenmesi düşünülebilir. Örneğin öyle bir bit grubu olsun ki, bu kısım mesajın tanımlayıcısı olsun ve bu belirleyici bilinmeden/değiştirilmeden dokümanda bir değişiklik yapılması mümkün olmasın. Eğer bu belirleyici, göndericinin özel anahtarı ile şifrelenmişse, bu kısım, mesajın orijinalliğini, ardışıklığını ve içeriğini güvenlik altında tutan bir imza gibi düşünülebilir. Bu belgede dijital imzanın ayrıntılarına derinlemesine girilmeyecektir. Belki ilerleyen zamanlarda, talebe göre bu konuda bilgi içeren bir kısım da belgeye eklenebilir.

Şunu vurgulamak önemlidir ki, şifreleme olayı açıklandığı kadarki hali ile gizliliği sağlamaz. Yani, mesajın değiştirilmesi engellenilmiş olsa da, bu mesajın gizlice dinleyenlerce ele geçirilmesini engelleyemez. Ortadadır ki, mesajın bir parçasının imza olacak şekilde şifrelenmiştir ve mesajın geriye kalan kısmı şifrelenmemiş şekilde gönderilmiştir. Hatta mesajın tamamının şifrelenmiş olması halinde bile, Şekil 3`te gösterildiği gibi, gizlilik mümkün değildir. Çünkü, herhangi bir izleyici, mesajı göndericinin genel anahtarı yardımıyla deşifre edebilir.

Bununla birlikte, hem kimlik doğrulama, hem de, gizlilik iki açık anahtar kullanılması ile sağlanabilir (Şekil 4):

$$Z = E_{KUb}[E_{KRa}(X)]$$
$$X = D_{KUa}[D_{KRb}(Z)]$$



Şekil 4

Bu durumda, bir mesajı şifrelemeye başlamadan önce, onu özel anahtarımız ile şifreleriz. Bu adım kimlik doğrulamayı sağlar. Daha sonra, bu yeni şifreli mesajı, alıcının genel anahtarı ile yeniden şifreleriz. Bu da gizliliği sağlar. Bu metodun dezavantajı iki kez şifrelenmiş olan metinin iki kez deşifrelenerek açılması esnalarında kaybedilecek fazladan zaman olarak düşünülebilir. Fakat mesaja sağladığı gizlilik ve kimlik doğrulama vazgeçilemez bir özelliktir.

Açık Anahtarlı Kripto Sistem Uygulamaları

Açıklamaya başlamadan önce, açık anahtarlı kripto sistemlerin bir yönünü açıklamalıyız yoksa, karışıklığa yol açmış oluruz. Açık anahtarlı sistemler karakteristik olarak, birisi gizli tutulan, diğeri ise genel kullanım için açılmış olan iki anahtarla çalışan kriptografik algoritmalar kullanırlar. Uygulamaya bağımlı olarak, gönderici, ya kendisinin özel anahtarını, ya alıcının genel anahtarını ya da ikisini birden, kimi kriptografik fonksiyonları gerçeklemek için kullanır. Geniş bir bakış açısı ile, açık anahtarlı kripto sistemlerin kullanımını üç kategoride inceleyebiliriz:

- **Şifreleme/De-Şifreleme:** Gönderici, bir mesajı alıcının genel anahtarı ile şifreler.
- **Dijital İmza:** Gönderen, mesajı kendi özel anahtarı ile imzalar. Bu imzalama, mesajın tamamını yada önemli görülen belirleyici bir kısmını şifrelemek ile yapılır.
- **Anahtar Değişimi:** İki taraf ortaklaşa bir oturum anahtarını değiş tokuş ederler. Bir çok farklı yöntem mümkündür. Anahtar değişimi senaryoları ilerde ayrıntıları ile ele alınacaktır.

Kimi algoritmalar, bu özelliklerden sadece bir yada iki tanesini gerçekleştirebilirken, bazıları bunların tümünü gerçekleştirebilir. Tablo 2, kimi açık anahtarlı algoritmaların bu özelliklerden hangilerini desteklediğini göstermektedir.

Algoritma	Şifreleme/De-şifreleme	Dijital İmza	Anahtar Değişimi
RSA	Evet	Evet	Evet
Diffie-Hellman	Hayır	Hayır	Evet
DSS	Hayır	Evet	Hayır

Tablo 2 Açık anahtarlı kripto sistemler için uygulamalar

Açık Anahtarlı Kriptografi için Gereklilikler

Kripto sistem, Tablo 2`de, iki benzer anahtarı temel alan kriptografik algoritmaya bağlıkları yoluyla ifade edilmiştir. Diffie ve Hellman, bu algoritmaların varlığını göstermeksizin bu sistemi varsaymışlardır. Bununla beraber, bu algoritmaların yerine getirmeleri gereken durumları şöyle ifade etmişlerdir:

1. Bir B için, anahtar parçalarını (genel anahtar ve özel anahtar) yaratmak, hesapsal olarak kolay olmalıdır.
2. Gönderenin (A olsun), mesajı göndereceği kişinin (B olsun) genel anahtarını bildiği ve şifrelenecek olan mesajı (M olsun) bildiği durumda, uygun şifreli metni yaratmak hesapsal olarak kolay olmalıdır.

$$C = E_{KU_b}(M)$$

3. Alıcı B`nin, özel anahtarını kullanarak, şifrelenmiş mesajı orijinal haline getirmesi hesapsal olarak kolay olmalıdır.

$$M = D_{KRb}(C) = D_{KRb}[E_{KUb}(M)]$$

4. Herhangi bir rakip için, genel anahtarı bilerek, özel anahtarı bulması hesapsal olarak imkansız olmalıdır.
5. Herhangi bir rakip için, genel anahtarı, şifreli metini (C`yi) bilerek orijinal mesajı (M`yi) elde etmesi hesapsal olarak imkansız olmalıdır.

Bunlara ek olarak, yararlı olmasına rağmen gerekli olmayan altıncı bir madde ekleyebiliriz:

6. Şifreleme ve de-şifreleme fonksiyonları her iki sıra ile de uygulanabilir olmalıdır.

$$M = E_{KUb}[D_{KRb}(M)]$$

Bunlar gerçekleştirilmesi çok zor gerekliliklerdir bu yüzden, açık anahtarlı kriptografi fikrinin ileri sürüldüğünden bu yana geçen yıllar süresince sadece tek bir algoritma geniş bir kitle tarafından kabul edilmiştir.

Bu kadar zor gerekliliklerin istenmesinin sebeplerini açıklamadan önce en önemli noktayı, tek yönlü fonksiyonu (one-way function) açıklayalım. Söz konusu olan tek yönlü fonksiyon şöyledir: fonksiyonun bire-bir olduğu bir aralıkta, tersini hesaplamak imkansız iken, fonksiyonun kendisinin hesaplanması kolaydır.

$Y = f(X)$ çok kolay,

$X = f^{-1}(Y)$ imkansız...

Genellikle, "kolay" dan kasıt, fonksiyonun girdi uzunluğuna bağlı olarak polinomal bir zaman süresi içerisinde çözülebilir olmasıdır. Şöyle ki, eğer girdi uzunluğu n bit kadarsa, fonksiyonun hesaplanması için gereken süre a bir sabit sayı iken, n^a gibi bir fonksiyonla orantılı olmalıdır. Çoğu algoritmanın, **P** sınıfı

algoritma olduğu söylenir. "imkansız" ise, oldukça bulanık bir durumu ifade etmek için kullanılır. Bir problemin çözümünün olanaksız olduğundan, giriş büyüklüğüne bağlı olarak çözüm için harcanan çabanın, polinomal zamandan daha hızlı arttığı durumda bahsedebiliriz. Örneğin, girdi n bit ile gösterilirken, fonksiyonun çözülme zamanı 2^n gibi bir fonksiyona bağlı olarak artıyorsa, bu fonksiyonun çözümünün imkansız olduğunu düşünebiliriz. Ne yazık ki, eğer bir algoritma parçası bu kompleksliği barındırıyorsa, bu karışıklığı belirlemek zordur. Ayrıca , hesapsal kompleksliğin geleneksel fikirleri bir algoritmanın kompleksliğini en kötü duruma yada ortalama bir duruma odaklar. Bu oranlar kriptografi için değersizdir. Çünkü kriptografide bir fonksiyonu tüm girilenler için tersine çevirmek nerdeyse olanaksızdır, bu genelleme, en kötü durum yada ortalama durum için geçerli değildir.

Şimdi, bir taraftan hesaplanması kolay, diğer bir taraftan ise belirli ek bilgiler bilinmedikçe hesaplanması olanaksız olan tuzak kapılı tek yönlü fonksiyonun (trap-door one-way function) açıklanmasına bakalım. Polinominal zamanda fonksiyonun tersi ek bilgiyle hesaplanabilir. Adım-adım özetleyebiliriz: Tuzak kapılı tek yönlü fonksiyon, tersine çevrilebilir fonksiyonların (f_k) bir ailesidir. Şöyle ki;

$Y = f_k(X)$	k ve X biliniyorsa, **kolay**
$X = f_k^{-1}(Y)$	k ve Y biliniyorsa, **kolay**
$X = f_k^{-1}(Y)$	Y biliniyor fakat k bilinmiyorsa **çözülemez**

Böylece, uygulamalı genel anahtarın yapısının gelişimi uygun bir tuzak kapılı tek yönlü fonksiyon bulunuşuna bağlıdır.

Açık Anahtarlı Kripto Analiz

Brute-force saldırısına karşı (Brute-force saldırısı, anahtar uzayının tüm elemanlarının sırası ile algoritma içerisinde teker teker denenmesi yolu ile doğru anahtarı elde etmeye çalışmaktır), bir genel anahtarlı şifreleme yapısı geleneksel şifreleme yapıları ile aynı derecede savunmasızdır. Çözüm aynıdır: Geniş anahtar uzayı kullanmak. Bununla birlikte, hesaba katılan bir trade-off vardır. Açık anahtarlı kripto sistemler, bazı kısa, tersine dönüştürülebilir matematiksel fonksiyonların kullanımına bağlıdır. Bu fonksiyonların hesabının karışıklığı, anahtardaki bitlerin sayısıyla doğrusal olarak ölçülemeyebilir; ancak karışıklık daha hızlı artar. Anahtarın büyüklüğü, brute-force saldırısını makul olanaklılık derecesinin dışına çıkarmak için yeterince büyük olmalıdır. Aynı zamanda da pratik şifreleme ve de-şifreleme için yeterince küçük olmalıdır. Pratikte, önerilmiş olan anahtar büyüklükleri brute-force saldırısını olanaksız kılar (Elbette bu hesaplama teknolojilerindeki hızlı gelişime bağlı olarak göreceli bir durumdur). Ancak, sonuçta şifreleme ve de şifreleme hızları genel amaç kullanımı için çok yavaş olur. Gizli anahtarlı simetrik şifrelemenin açık anahtarlı şifrelemeye nazaran çok daha hızlı olmasından ötürü, açık anahtarlı şifreleme yaygın olarak anahtar yönetimi ve imza kullanımlarında sınırlandırılır.

Saldırının diğer bir şekli; özel anahtarı hesaplamanın bir yolunu bulmak için, verilen genel anahtarı kullanmaktır. Şu ana dek, bir genel anahtarlı algoritma parçacığı için bu tip bir saldırının başarılı olmasının mümkün olmadığı matematiksel olarak **ispatlanmamıştır**. Böylece, verilen herhangi bir algoritma (geniş çapta kullanılan RSA algoritmalarını içeren) şüphelidir. Kriptonalizin tarihi, bütünüyle farklı bir yönden bakıldığında çözümü bulunabilecek, bir yönden de çözümsüz gibi gözüken bir problemin varlığını gösterir.

Son olarak, genel anahtarlı sistemler için tuhaf bir saldırı şekli daha vardır. Özünde bu saldırı bir olası-mesaj saldırısıdır. Varsayalım ki; 56 bitlik DES anahtarıyla gönderilmek üzere oluşturulmuş bir mesaj olsun. Bir saldırgan, genel anahtarı kullanarak olası tüm anahtarları şifreleyebilir ve herhangi bir mesajı gönderilmiş şifreli metinle karşılaştırarak deşifre edebilirdi. Böylece genel anahtarlı yapının anahtar büyüklüğünün önemi kalmaz, saldırı 56 bitlik bir

anahtara yapılan brute-force saldırısına dönüştürülürdü. Bu saldırı; bu gibi basit mesajlara rasgele bazı bitler eklenerek, önlenebilir.

RSA ALGORİTMASI

Diffie ve Hellman tarafından hazırlanmış öncü bir makale 1976 yılında, kriptografi için yeni bir yöntem tanıttı, ve sonuçta, genel anahtarlı sistemlerin gerekliliklerini yerine getiren bir kriptografik algoritmada görüş birliğine varan kriptolojistlere karşı meydan okudu. Bu meydan okumaya karşı yanıtlardan ilki; 1977'de MIT'de Ron Rivest, Adi Shamir, ve Len Adleman (RSA) tarafından ortaya atıldı, ve ilk olarak 1978'de (A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, February 1978) basıldı. Rivest-Shamir-Adleman (RSA) yapısı; üstünlüğü kabul gördüğünden itibaren, geniş çapta tek olarak kabul edildi ve genel anahtarlı şifreleme yönteminin genel amacını yerine getirdi.

RSA yapısı, birtakım n`ler için $0$ve $n-1$arasındaki tamsayılardan oluşan şifreli ve düz metinin içinde yer alan bir blok şifrelemedir. Bu bölümde RSA`yı bazı detaylarla birlikte, algoritmanın açıklamasıyla başlayarak inceleyeceğiz. Daha sonra RSA`nın hesapsal ve kripto analitik anlamlarını incelemeye çalışacağız.

Algoritmanın Tanımı

Rivest, Shamir, ve Adleman tarafından bulunan yapı, destekleyici görüşlerle birlikte ifadenin kullanımını meydana getirir. Düz metin, blokların içinde şifrelenir. Her blok, birtakım $n$sayısından daha az bir ikili değere sahiptir. Bloğun büyüklüğü, $\log_2(n)$`e eşit yada ondan daha az olmalıdır; pratikte, blok büyüklüğü 2^kbittir, $2^k < n \leq 2^{k+1}$aralığında. Şifreleme ve de şifreleme bazı düz metin bloğu Mve şifreli metin bloğu Ciçin şu şekildedir:

$$C = M^e \bmod n$$
$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Hem gönderen hem de alıcı n`in değerini bilmelidir. Gönderen $e$`nin değerini bilir, ve sadece alıcı d`nin değerini bilir. Böylece; $KU = \{e,n\}$ bir genel anahtar, ve $KR = \{d,n\}$ bir özel anahtar olur ve bu bir genel anahtarlı şifreleme algoritmasıdır. Genel anahtarlı şifreleme için tatmin edici olması için, bu algoritma için aşağıdaki gereklilikler yerine getirilmelidir:

1. $M < n$ olduğu koşulda, $M^{ed} = M \bmod n$ iken, e,d,n değerlerini bulmak mümkün olmalıdır.

2. $M < n$ koşulunu sağlayan tüm M değerleri için, M^e ve C^d hesaplanması nisbeten kolay olmalıdır.

3. Yalnız e ve n verildiğinde, d`nin hesaplanması imkansız olmalıdır.

Şimdi, ilk sorun üzerinde odaklanalım ve diğerlerine sonra geçelim. Aşağıdaki form için bir ilişki bulmamız gerekiyor:

$$M^{ed} = M \bmod n$$

Euler`in teoremine göre, verilen iki asal sayı p ve q, ve iki tamsayı n ve m olmak üzere, $n = pq$ ve $0 < m < n$ olduğu durumda, keyfi seçilmiş bir k tamsayısı seçilmiş sayılar ile şöyle bir ilişki oluşturur:

$$m^{k\Phi(n)+1} = m^{k(p-1)(q-1)+1} \equiv m \bmod n$$

Buradaki $\Phi(n)$ fonksiyonunun döndürdüğü değer, n`den küçük olan ve n ile aralarında asal olan tam sayıların sayısıdır. p ve q asal sayı olmak üzere

$\Phi(pq) = (p-1)(q-1)$ olur (Bunun ispatı konumuzun dışındadır, bu yüzden burada yapılmayacaktır). Böylece aşağıdaki eşitlik sağlanıyorsa istenilen ilişkiye ulaşabiliriz:

$$ed = k\Phi(n) + 1$$

Bu durumda aşağıdaki denkliklerden söz edilebilir:

$$ed \equiv 1 \bmod \Phi(n)$$
$$d \equiv e^{-1} \bmod \Phi(n)$$

e ve d , $\bmod \Phi(n)$ `in çarpmaya göre tersidir. Dikkat edecek olursak, modüler aritmetiğin kurallarına göre, bu denkliğin doğru olması yalnız $d$ `nin (ve sonucunda e `nin) $\Phi(n)$ ile aralarında asal olması durumunda mümkündür. Bu durumda, gcd($\Phi(n)$, d) = 1 demektir.

Şimdi, RSA yapısını açıklamaya hazırız. Yapının bileşenleri sırasıyla şöyledir (parantez içlerinde, sayıların nasıl elde edildiklerini ve genel mi yoksa gizlimi olduklarını belirtilmektedir):

p, q ; iki asal sayı (gizli, seçilmiş)
$n = pq$;(genel, hesaplanmış)
e ; gcd($\Phi(n)$, e) = 1; $1 < e < \Phi(n)$ olacak şekilde (genel, seçilmiş)
$d \equiv e^{-1} \bmod \Phi(n)$; (gizli, hesaplanmış)

$\{d, n\}$ çifti özel anahtarı, $\{e, n\}$ çifti ise genel anahtarı oluşturur. Varsayalım ki, bir B kullanıcısı, A kullanıcısına, A kullanıcısının genel anahtarını kullanarak bir M mesajını göndermek istiyor. Bu durumda B , $C = M^e \pmod n$ formülü yardımı ile C şifreli mesajını elde edecek ve bunu A `ya gönderecektir. A kullanıcısı bu mesajı aldığında, $M = C^d \pmod n$ formülü ile mesajı de şifre edecektir.

Bu kıymetli algoritmanın doğrulunu özetlemek için biraz sıkıntıya girmeye değer :) Biz, e ve d `yi aşağıdaki denkliği sağlayacak şekilde seçmiştik:

$$d \equiv e^{-1} \bmod \Phi(n)$$

Bunun sonucunda,

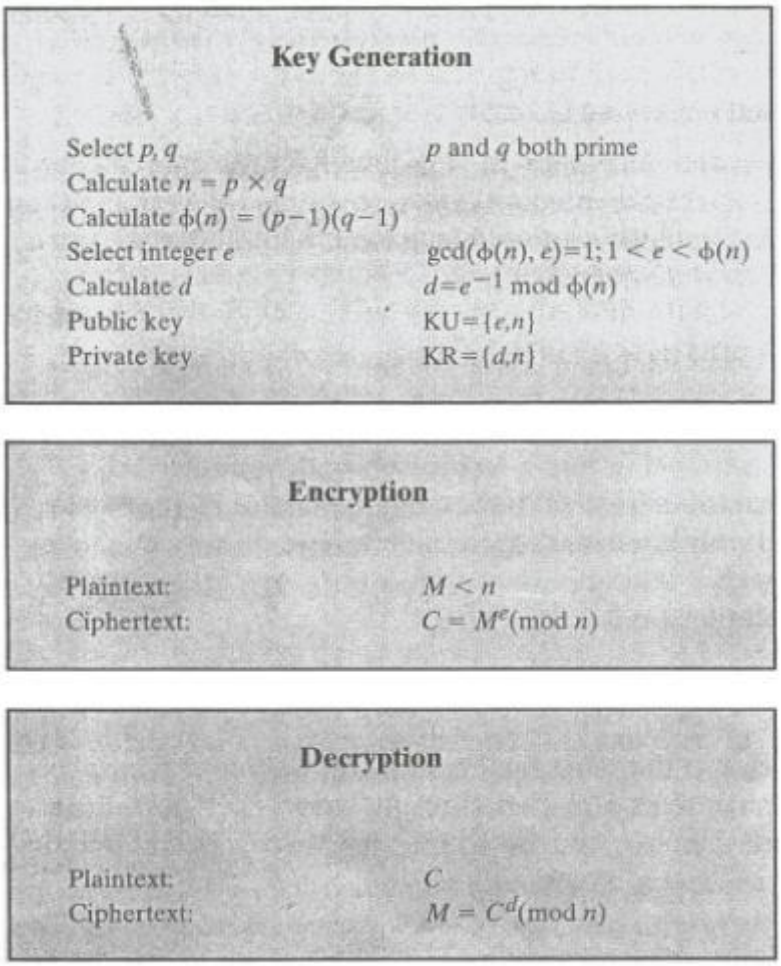
$$ed = 1 \bmod \Phi(n)$$

Bu yüzden ed, $k\Phi(n)+1$`in bir formudur. Bu denkliğin Euler teoreminin bir sonucu olduğu, iki asal sayı olan p ve q, birer tamsayı olan n ($n = pq$) ve M ($0 < M < n$) alınarak kolaylıkla ispatlanabilir.

$$M^{k\Phi(n)+1} = M^{k(p-1)(q-1)+1} \equiv M \bmod n$$

Dolayısıyla görüyoruz ki, $M^{ed} = \bmod(n)$;

$$C = M^e \bmod n$$
$$M = C^d \bmod(n) \equiv (M^e)^d \bmod(n) \equiv M^{ed} \bmod(n) \equiv M \bmod n$$

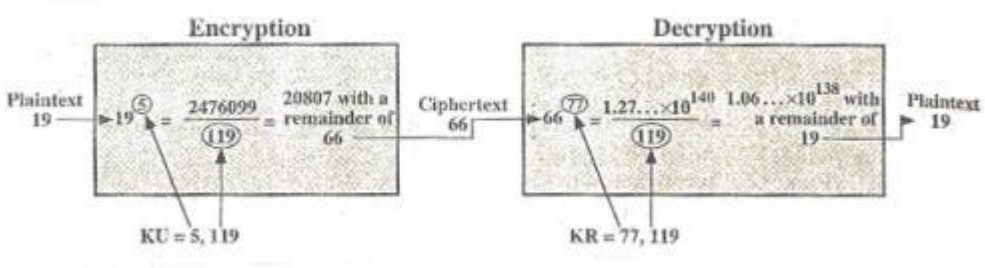


Şekil 5

Şekil 5`te, RSA algoritması özetlenmiştir. Ve Şekil 6` da da bir örnek vierlmiştir ve bu örnekteki anahtarlar şu şekilde yaratılmıştır (siz de kağıt üstünde kendi seçtiğiniz sayılar ile deneyiniz):

1. İki adet asal sayı seçelim, $p = 7$ ve $q = 17.$
2. n değerini hesaplayalım, $n = pq = 7x17 = 119.$
3. $\Phi(n)$ değerini hesaplayalım, $\Phi(n) = (p-1)(q-1) = 96.$

4. Şimdi de, $\Phi(n)$`den küçük olacak ve $\Phi(n)$ile aralarında asal olacak şekilde bir $e$sayısı seçelim, burada $e = 5$olsun…
5. Son olarak, öyle bir $d$belirleyelim ki, $de = 1 \bmod 96$ve $d < 96$olsun. Doğru değer, $d = 77$olacaktır. Çünkü, $77x5 = 385 = 4x96 + 1$`dir.



Şekil 6

Bu işlemler sonucunda elde ettiğimiz değerlere göre genel anahtar $KU = \{5,119\}$, özel anahtar da $KR = \{77,119\}$oldular. Bu örnek bize, bu anahtarların $M = 19$düzmetni için kullanımını gösteriyor. Şifreleme için, 19un beşinci kuvveti alınıyor ve 2476099sayısı elde ediliyor. Daha sonra bu sayının 119`a bölümünden kalan bulunuyor ve ortaya şifreli metin olan 66çıkıyor. Deşifreleme işleminde de, $66^{77} \equiv 19 \bmod 119$işlemi sayesinde de şifrelenmiş metin olarak 19 elde ediliyor.

Hesaplama Yöntemleri

Şimdi, RSA`nın kullanımı için gerekli hesabın karışıklığıyla ilgili önemli noktaya geri dönüyoruz. Aslında düşünülmesi gereken iki önemli nokta vardır: anahtar üretimi, şifreleme/deşifreleme. Önce şifreleme/deşifreleme işlemlerine bakmaya çalışacak ve daha sonra anahtar üretimi konusuna döneceğiz.

Şifreleme ve Deşifreleme

RSA`da hem şifreleme hem de deşifreleme, tamsayıların tamsayı kuvvetlerini almayı ve mod alma işlemlerini gerektirir. Eğer ilk önce tamsayıların üslerini alıp, daha sonra $\bmod n$ ile indirgersek, ara değerler devasa büyüklükte sayılar olurlar. Neyse ki bu sorunu bir nebze azaltmak için modüler aritmetiğin şu özelliğinden yararlanabiliriz:

$$[(a \bmod n) x (b \bmod n)] \bmod n = (axb) \bmod n$$

Bu sayede, ara değerleri modül n`e göre indirgeyebiliriz. Bu da hesaplamayı pratik hale getirir.

Diğer bir husus, üssün verimidir. Çünkü, RSA ile, potansiyel olarak büyük üsler ile işlem yaparız. Verimin nekadar artabiliceğini görmek için, x^{16}`yı hesaplamak istediğimizi varsayalım. Dürüst bir yöntem 15 çarpım gerektirir:

$$x^{16} = x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x$$

Bununla birlikte, aynı sonuca, her bir kısmi sonucun karesini alarak x^2, x^4, x^8, x^{16} olacak şekilde dört adımda da ulaşabiliriz.

Daha genel olarak varsayalım ki biz a^m değerini hesaplamak istiyoruz ve biliyoruz ki a ve m birer pozitif tam sayı. Eğer biz m`i $b_k b_{k-1} \cdots b_0$ gibi ikilik sayı gibi ifade edecek olursak:

$$m = \sum_{b_i \neq 0} 2^i$$ olur.

Böylece,

$$a^m = a^{\left(\sum_{b_i \neq 0} 2^i\right)} = \prod_{b_i \neq 0} a^{(2^i)}$$

$$a^m \bmod n = \left[\prod_{b_i \neq 0} a^{(2^i)}\right] \bmod n = \prod_{b_i \neq 0} \left[a^{(2^i)} \bmod n\right]$$ olur.

Bu sonuç sayesinde, $a^b \bmod n$ işlemini hesaplamak üzere aşağıdaki algoritmayı geliştirebiliriz. Ve algoritmanın altındaki tablo da algoritmanın çalışmasını örneklemektedir. c değerinin aslında gerekli olmadığını düşünebilirsiniz; gerçekten de algoritma içerisinde direk bir fonksiyonu yoktur. Fakat son değeri üssün değerine eşit olacağından dolayı açıklayıcı bir niteliktedir.

```
c←0; d←1
for i←k downto 0
    do c←2 x c
       d←(d x d)mod n
       if bi=1
          then c←c+1
               d←(d x a)mod n
return d
```

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

Anahtar Üretimi

Açık anahtarlı kriptosistemin uygulamasından önce, her iki katılımcı anahtar parçalarını üretmelidir. Bu üretim işlemi aşağıdaki vazifeleri ihtiva eder:

- İki asal sayı hesaplanması, p ve q,
- e ya da d `nin seçilip diğerinin hesaplanması.

Öncelikle, p ve q nun seçimini düşünelim. Çünkü, herhangi bir potansiyel saldırgan $n = pq$ `nun değerini biliyor olacaktır, ayrıntılı methodlarla p ve q nun bulunmasını engellemek için, p ve q sayıları yeterince büyük bir seriden seçilmiş olmalıdırlar (p ve q büyük sayılar olmalıdır). Diğer bir yandan, büyük asal sayıları bulmak için kullanılan yöntem yeterince verimli olmalıdır.

Günümüzde, verimli ve büyük asal sayılar üreten kullanışlı bir teknik yoktur. Genel olarak kullanılan yöntem şöyle çalışmaktadır: istenen büyüklük aralığında rastgele bir tek sayı seçilir ve bunun asal olup olmadığını kontrol edilir. Eğer asal değilse, bu işlem asal bir sayı buluncaya kadar sürdürülür.

Asallığı kontrol eden bir çok test geliştirilmiştir. Testlerin nerdeyse tümü yaklaşıklıktan bahseder yani, test, verilen (yeterince büyük) bir tam sayının muhtemelen asal olup olmadığını belirleyecektir. Bu kesinlik eksikliğine karşın, testler, sayının asal olma olasılığının 1,00`a çok yakın olduğu durumlarda çalışabilirler. Örnek olarak, en verimli ve popüler test algoritmalarından birisi Miller-Rabin algoritmasıdır. Bu algoritmada ve diğer bir çok algoritmada, n sayısının asallığını kontrol etmek için, n ile bir takım işlemlere sokulmak üzere n den küçük bir rastgele a sayısı seçilir. Eğer n testi geçemezse bunun anlamı n sayısının asal olmadığıdır. Eğer n testi geçerse bu durumda sayı asal olabilir, olmayadabilir. Eğer n sayısı bu gibi çok fazla sayıda (milyonlarca..) testten başarılı olursa, n için muhtemelen asal denilir.

Özet olarak, asal sayı kontrol prosedürü aşağıdaki adımları izler:

1. Rastgele bir tek tam sayı "n" seçilir.
2. $a < n$ koşulunu sağlayan bir rastgele bir a sayısı seçilir.
3. n için asallığı yaklaşık olarak test eden Miller-Rabin gibi bir test gerçekleştirilir, eğer n testi geçemezse birinci adıma geri dönülür.
4. Eğer n bir çok sayıda testi başarı ile geçmişse n kabul edilir yada ikinci adıma geri dönlülür.

Bu, oldukça sıkıcı bir işlemdir fakat biliyoruz ki, bu işlem sadece yeni bir genel ve özel anahtar (KU, KR) gerektiğinde uygulanır.

Asal bir sayı bulunana dek geri çevrilen sayıların sayısı önemli bir ayrıntıdır. Sayılar teoreminin bir sonucuna, asal sayılar teorisine göre bir N asal sayısına yakın olan ilk asal sayı, bu sayıya ortalama $\ln N$ adet tam sayı kadar uzaklıktadır. Bu durumda, bir asal sayı bulunduğunda, bir sonraki asal sayı yaklaşık olarak $\ln N$ tamsayı ötede olacaktır. Ve bu aralıkta bulunan çift sayıların asal olmadığı kesin olduğundan, bir asal sayıya ulaşması için en fazla $\ln N/2$ deneme yapması gerekecektir. Bunun anlamı şudur: örneğin 2^{200} sayısını merkez alarak asal sayı aramaya başlayan bir kişinin bir asal sayıya ulaşmak için sayı ekseninde bir yöne doğru yaklaşık $\ln(2^{200})/2=70$ sayı denemesi gerekmektedir.

p ve q değerlerine sahip olunmasından sonra bir e değeri seçilmesi ve d değerinin hesaplanması ya da alternatif olarak d değeri seçilerek e değerinin hesaplanmasının ardından anahtar üretme işlemi tamamlanmış olacaktır. e sayısını seçerken, bu sayının gcd($\Phi(n), e$)=1 eşitliğini sağlaması gerektiğini biliyoruz. Daha sonra da, $d = e^{-1} \bmod \Phi(n)$ eşitliği sonucunda d değerini elde etmeliyiz. Neyseki, aynı anda iki tam sayının en büyük ortak bölenini bulan, ve bu bölen 1 ise, bu sayılardan birisinin tersini, diğerinin modülüne göre hesplayan tek bir algoritma var... Bu algoritma uzatılmış Euclid algoritması olarak anılmaktadır. Prosedür, yarattığı bir seri halindeki rastgele sayıların her birisini, $\Phi(n)$ ile arasında asal olan bir sayı buluncaya dek teker teker dener. Tekrar şu soruyu sorabiliriz: $\Phi(n)$ ile aralarında asal olacak şekilde kullanışlı bir sayı bulmak için kaç rastgele sayıyı test etmeliyiz? Şunu kolayca gösterebiliriz ki, rastgele seçilmiş iki sayının aralarında asal olma olasılığı 0.6`dır. Bu da demek oluyor ki, aradığımız tam sayıya ulaşmak için bir kaç test yapmamız yeterli olacak.

RSA`nın Güvenliği

Rsa algoritmasına saldırmak için kullanılabilecek üç metod aşağıdaki gibidir:

- **Brute-force:** Bütün özel anahtarların denenmesi ile gerçekleştirilir.
- **Matematik Ataklar:** Birkaç yöntem vardır, hepsinini amacı çarpımı oluşturan iki asal sayıyı bulmaktır.
- **Zaman Atakları:** Deşifreleme algoritmasının çalışması esnasında geçen zamana bağlıdır.

Brute-force yöntemine karşı RSA`nın savunması diğer diğer kripto sistemlerin çözümünden farklı değildir, çözüm geniş anahtar uzayı kullanmaktır. $e$ ve $d$ `nin bit sayıları ne kadar büyük olursa o kadar iyidir. Bununla beraber, hem anahtar üretimi hem de şifreleme/de şifreleme işlemleri için algoritmanın içerdiği hesaplamalar kompleksleşecek, büyük anahtarlarla çalışan sistemin de çalışma zamanından kaybı olacaktır.

Biz bu alt kısımda Brute-force`dan ziyade matematiksel ve zaman ataklar ile ilgileneceğiz.

Çarpan Problemi

RSA`ya gerçekleştirilebilecek matematik atak yöntemlerini üç alanda inceleyebiliriz:

- n `in çarpanlarından iki tanesi, kendisini oluşturan asal sayıların bulunması. Bu sayıların bulunmasıyla $\Phi(n) = (p-1)(q-1)$ hesaplanabilir ve dolayısıyla $d = e^{-1} (\mathrm{mod}\, \Phi(n))$ hesaplanabilir.
- $\Phi(n)$ `in p ve q bulunmadan, doğrudan hesaplanması. Yine buradan $d = e^{-1} (\mathrm{mod}\, \Phi(n))$ hesaplanabilir.
- d `nin, $\Phi(n)$ hesaplanmadan hesaplanması.

RSA`ya yönelik en çok tartışılan kriptoanaliz yöntemi, $n$ sayısını kendisini oluşturan iki asal çarpanına ayırmaktır. Verilmiş bir $n$ için $\Phi(n)$ hesaplanması, $n$`in çarpanlarına ayrılması demektir. Günümüzde, verilen n ve e için d değerini hesaplayan algoritmalar üs alma problemi gibi zaman ile de ilgili bir problem yaratmaktadırlar. Bu nedenle çarpanlara ayırma performansını, RSA`nın güvenliği için bir tehdit olarak düşünebiliriz.

Çok büyük asal sayılardan oluşturulmuş çok büyük bir n`i çarpanlarına ayırmak çok zor bir işlem olacaktır; fakat bu zorluk $n$`in kullanımından daha büyük bir zorluk değildir. 1977 yılında, RSA`nın üç yaratıcısı, Scientific American okuyucularına meydan okuyarak, derginin Martin Gardner tarafından hazırlanan "Matematik Oyunları" kısmına, deşifre edilmesi için RSA ile şifrelenmiş bir metin koydular. Bu metni deşifre ederek getirecek olan kişiye de $100 ödül vereceklerini, fakat bu işlemin 40 milyar yıldan uzun süreceğini söylediler. 1994 Nisanında, internet üzerinden çalışan bir grup, sadece 8 aylık bir çalışma ile ödülü almaya hak kazandılar. Bu meydan okumada kullanılan genel anahtar 129 rakam yani yaklaşık 428 bitti. RSA Laboratuvarları, 100, 110, 120 ve bu sekilde artan anahtar boyutları ile şifrelenmiş metinler için meydan okumaları sürdürdü. En son sonuçlanan idda da 130 rakamdan oluşan anahtar ile şifrelenmiş metnin çözülmesi oldu. 3 numaralı tabloda, tarihlerine göre alınmış sonuçları listelemekte. Sarfedilen efor kolonu MIPS birimi ile ifade ediliyor ve 1 MIPS-yılı demek, saniyede 1 milyon işlem yapan bir işlemcinin bir yıl süre ile çalışması anlamına geliyor. Bu durumda, yaklaşık $3x10^{13}$ adet işlem gerçekleştirilmiş oluyor. 200 Mhz.`lik bir Pentium işlemcili bir makine yaklaşık 50-MIPS` lık bir makinedir.

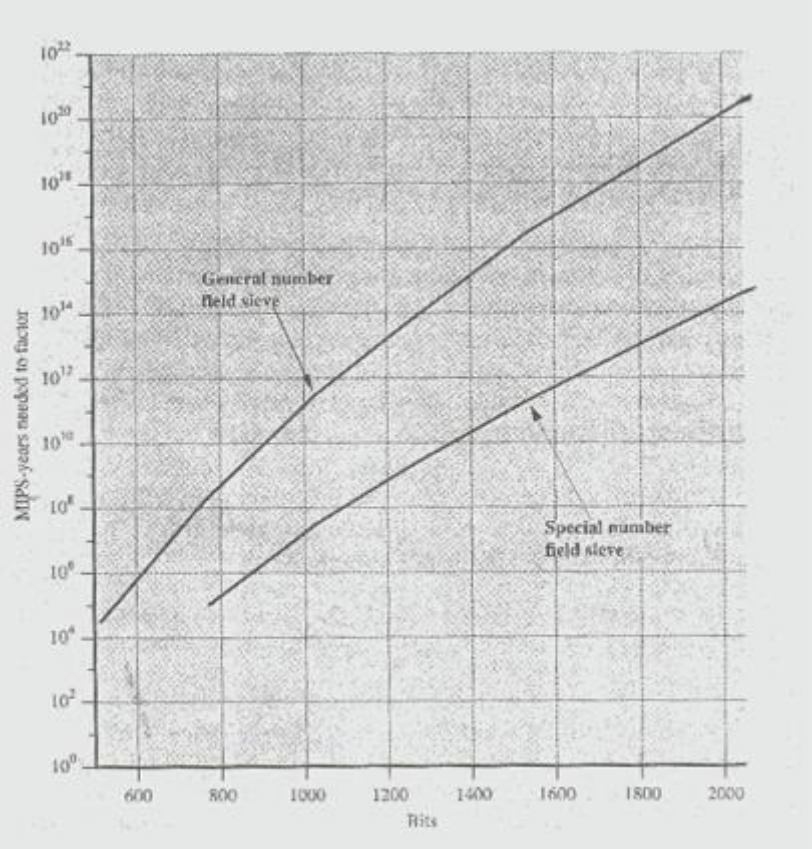
Anahtarın Rakam Sayısı	Yaklaşık Bit Sayısı	Verinin Elde Edilmesi	MIPS-yılı	Algoritma
100	332	Nisan 1991	7	**Quadratik eleme**
110	365	Nisan 1992	75	**Quadratik eleme**
120	398	Haziran 1993	830	**Quadratik eleme**
129	428	Nisan 1994	5000	**Quadratik eleme**
130	431	Nisan 1996	500	**Genelleştirilmiş sayı alanı elemesi**

Tablo 3

Son zamanlara kadar, çarpanlara ayırma atakları quadratik eleme adı verilen bir yöntem kullanılarak yapılıyordu. RSA-130 için yapılan atakta, daha yeni bir algoritma kullanıldı: "Genelleştirilmiş sayı alanı elemesi (GNFS: Generalized Number Field Sieve)". Bu sayede, RSA-129` dan daha büyük bir sayının çarpanlarına ayrılması 10%`luk bir çaba ile gerçekleştirilebildi.

Bilgisayar teknolojisinin hızla gelişmesi ve çarpanlara ayırma algoritmalarının zamanla rafine edilerek daha kaliteli hale gelmeleri neticesinde, büyük anahtar boyutları kullanmanın verdiği gözdağı yavaş yavaş kriptoanalistler için önemsizleşiyor. Tablodan da, bir algoritma değişikliğinin ne kadar muazzam bir hız yükselişi sağladığını görebilirsiniz. Üstelik, GNFS`nin üzernde yapılacak bir rafinasyon işlemi sonucunda çok daha etkin bir algoritma ortaya çıkarılması çok mümkündür. Aslında, "Özel Sayı Alanı Elemesi (SNFS: Special Number Field Sieve)" adında bir algoritma, özelleştirilmiş şekli ile sayıların çarpanlarını GNFS`ye göre çok daha hızlı bulmaktadır. 9 numaralı şekilde, bu iki algoritmanın performans karşılaştırmasını görebilirsiniz. Şunun beklenmesi gereklidir ki, ani bir atakla SNFS kadar yada ondan daha hızlı çalışabilecek bir genel çarpanlara

ayırma algoritması geliştirilebilir. Bu yüzden RSA`da kullanacağımız anahtar boyutunu seçerken çok dikkatli olmalıyız. Yakın bir gelecekte, 1024 yerine 2048 bitlik anahtarlar kullanılması çok da imkansız görünmüyor...



Şekil 9

Ek olarak, algoritma geliştiricileri ve araştırmacılar, asal çarpanlarına kolayca ayrılabilen n sayılarının üretilmemesi için, p ve q sayıları seçilirken bazı kısıtlamaların göz önünde bulundurulması gerektiğini belirtiyor ve şunları tavsiye ediyorlar:

1. p ve q sayılarının uzunlukları birbirinden sadece birkaç rakam farklı olmalı. Bu yüzden, hem p hem de q sayısı 10^{75} ile 10^{100} aralığından seçilmeli.

2. Hem (p -1) hem de (q -1) büyük bir asal çarpan içermeli.
3. gcd(p-1,q-1) küçük olmalı.

Ayrıca, [WIEN90 (Referansı kaybetmişim en kısazamanda ekleyeceğim)]`da da ispatlandığı üzere, eğer $e < n$ ve $d < n^{1/4}$ ise, d kolaylıkla hesaplanabilirdir.

Zaman Atakları

Eğer halen, bir kriptografik algoritmanın güvenliği konusunda emin olmanın ne kadar zor olduğu konusunda ders almamış birisi varsa, zaman atakları onun için çok çekici ve çarpıcı bir örnek olacaktır. Bir kriptografi uzmanı olan Paul Kocher, bir bilgisayarın şifreli bir metni çözerken harcadığı zaman dilimlerinden yararlanarak, o metnin oluşturulması esnasında kullanılan özel anahtarı hesaplayabileceğini kanıtladı (1996, Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems). Zaman atakları sadece RSA için değil, diğer açık anahtarlı kriptografik sistemler içinde uygulanabilir bir saldırı metodudur. Bu atak şu iki konu sebebiyle çok dikkat çekici: bu atak tamamiyle beklenmedik bir yönden geliyor ve, ikinciside bu sadece chipertext-only atağı (Yani sadece şifreli metnin üstünde, metin hakkında herhangi bir ek bilgi olmaksızın gerçekleştirilen atak; bu atak yöntemi normal koşullarda başarıya ulaşması en az umulan ataklardan birisidir).

Zaman atağı, bir hırsızın, soymak istediği bir kasanın parola kombinasyonunu tahmin etmek için, o kasayı açan bir kişinin çevirdiği her bir numaranın ne kadar süre aldığını dikkatle izlemesine oldukça benzemektedir. Bu atak yöntemini, modüler üs alma algoritmasını kullanarak açıklayabiliriz fakat, atak yöntemi belirlenmiş zamanlarda çalışmayan herhangi bir uygulamaya da adapte edilebilir. Bu algoritmada, her adımda, modüler üs alma işlemi bit-bit ve bir modüler çarpım işlemi gerçekleştiriliyor. Ayrıca, ekstra bir modüler çarpım işlemide değeri 1 olan her bit için yapılıyor.

Kocher`in de makalesinde belirttiği gibi, bu atağı anlamak son derece kolay. Farzedin ki, hedef sistem çoğu koşulda çok hızlı, sadece bazı girdiler için

normalden biraz daha yavaş çalışan bir modüler çarpım algoritması kullanıyor olsun. Atak, en soldaki bitten, b_k`dan başlayarak bit bit ilerlemeye başlasın. Farzedelim ki, ilk j bit biliniyor. Verilmiş bir şifreli metin için atağı gerçekleştiren kişi, ilk j adımı **for** döngüsü ile bitirebilir. İşlemin sonradan gelen adımları bilinmeyen üs bitine bağlıdır. Eğer bit set edilmişse, d$\leftarrow$(d x a) mod n işlemi çalışacaktır. a ve d`nin bazı değerleri için modüler çarpım son derece yavaş olacaktır ve atağı gerçekleştiren kişi bu bitlerin değerinin ne olduğunu anlayacaktır. Bu yüzden, eğer deşifreleme algoritmasının çalışma zamanı dikkatle gözlenirse, bu algoritma parçası her 1 bit için çalışma zamanını yavaşlatacaktır ve o an üzerinde çalıştığı bitin değerinin 1 olduğu tahmin edilebilecektir. Aynı şekilde algoritmanın anlık çalışma zamanı hızlıysa o anda üstünde çalışılan bitin değerinin 0 olduğu anlaşılacaktır.

Pratikte, modüler üs alma tanımlamaları tüm algoritmanın çalışma zamanı üzerinde bu kadar büyük değişiklikler yaratmamaktadır, yinede algoritmalar içerisinde, bu atağı pratik hale getirecek yeterince materyal bulunmaktadır. Detaylar için Paul Kocher`in yukarda ismi verilmiş olan makalesini inceleyebilirsiniz.

Bununla beraber zaman atakları önemli bir tehlikedir. Atağı önlemek için çok basit önlemler kullanılabilir:

- **Sabit üs alma zamanı:** Tüm üs alma fonksiyonlarının çalışma zamanı eşit hale getirilebilir. Fonksiyonun çalışması erken bitse dahi, geriye bir değer döndürmesi bir miktar bekletilerek hepsinin çalışma zamanı belli bir değerde sabitlenebilir. Bu çok basit bir önlem olur fakat, performansı kötü yönde çok fazla etkileyecektir...
- **Rastgele gecikme:** Daha iyi bir performans, üs alma algoritmasına rastgele bekleme süresi eklenerek zaman atağını yanıltılmasıyla elde edilebilir. Kocher, bu rastgele değerlerin dikkatsizce seçilmesi durumunda, zaman atağı gerçekleştiren kişinin biraz daha fazla gayret göstererek yine başarıya ulaşmasının mümkün olacağına dikkat çekiyor.
- **Körleştirmek:** Üs alma işleminin gerçekleştirilmesinden önce, şifreli metin parçasını rastgele bir sayı ile çoğaltarak zaman atağını

gerçekleştiren kişinin sağlıklı bir veri elde etmesi engellenebilir. RSA Data Security kuruluşunun geliştirdiği bir koruma yöntemi mevcuttur. RSA Data Security, bu atak yönteminin toplam performansa 2% ila 10%`luk bir kayıp getirdiğininde altını çizmekte...

ANAHTAR YÖNETİMİ

Açık anahtarlı kriptografinin en önemli rollerinden birisi, anahtar dağıtımı problemine getirdiği yeniliktir. Açık anahtarlı şifreleme kullanmak için iki çok önemli neden vardır:

- Açık anahtarların dağıtımı.
- Gizli anahtarların dağıtımı için açık anahtarlı şifreleme kullanımı.

Sırayla bu iki konuyuda inceleyeceğiz.

Açık Anahtarların Dağıtımı

Açık anahtarları dağıtmak için kullanılan birkaç teknik vardır. Nerdeyse tüm önerilmiş yöntemleri açağıdaki gibi gruplayabiliriz:

- Genel duyuru,
- Herkez tarafından erişilebilir adres rehberi,
- Açık anahtar yetkilisi,
- Açık anahtar sertifikası.

Açık Anahtarların Duyurulması

Açık anahtarlı şifrelemenin en önemli özelliği, açık anahtarın açık olmasıdır. Bu yüzden, eğer birisi RSA gibi bir açık anahtarlı şifreleme algoritmasının kullanımını tamamiyle kabul etmişse, bu kişi açık anahtarını bir başkasına gönderebilir ya da,

bütün iletişim ağına duyurabilir. Örneğin, RSA algoritmasını kullanan PGP`nin (Pretty Good Privacy) popülaritesinin artması sonucu bir çok PGP kullanıcısı, herkese açık forumlara, USENET haber gruplarına ya da mail listelerine attıkları mesajların sonuna açık anahtarlarını eklemeyi benimsediler.

Bu yöntemin çok kolay ve kullanışlı olmasına karşın, büyük bir yetersizliği vardır: Herhangi birisi siz olduğunu idda ederek açık anahtarını kişilere duyurabilir. Yani bir kullanıcı kendisini olmadığı halde A kullanıcısı gibi tanıtabilir ve kendi açık anahtarını A kullanıcısının açık anahtarıymış gibi kişilere ilan edebilir. Gerçek A kullanıcısı sahtekarlığın farkına varana kadar ya da bir başka kullanıcı onu uyarana dek, sahte A kullanıcısı gerçek A kullanıcısına gönderilmek üzere şifrelenmiş bütün mesajları okuyabilir.

Herkes Tarafından Erişilebilir Adres Rehberi

Herkezce erişilebilir dinamik bir açık anahtar adres rehberinin iyi bir şekilde korunması ve organize edilmesiyle çok yüksek derecede güvenlik sağlanabilir. Adres rehberlerindeki anahtarların dağıtımı ve korunması güvenilir kimi kişi veya kuruluşların sorumluluğunda olmalıdır. Böyle bir hizmet tasarlanırken –en azından- aşağıdaki koşullar sağlanmış olunmalıdır:

1. Adres rehberinn her elemanı için bulunacak {ad, açık anahtar} bölümleri iyi korunmalıdır.

2. Her kullanıcı rehber yetkilileri ile bir açık anahtar kaydetmeli ve bu işlemi yüz yüze ya da kimlik kontrolü ile güvenli hale getirilmiş bir iletişim metodu ile yapmalıdırlar.

3. Herhangi bir üye adres rehberindeki genel anahtarını, bu anahtarla çok fazla verinin şifrelenmiş olmasından dolayı ya da bu anahtarla ilişkili

olan özel anahtarın herhangi bir sebepten ötürü güvenilirliğini yitirmesinden ötürü istediği bir zaman yenisi ile değiştirebilmelidir.

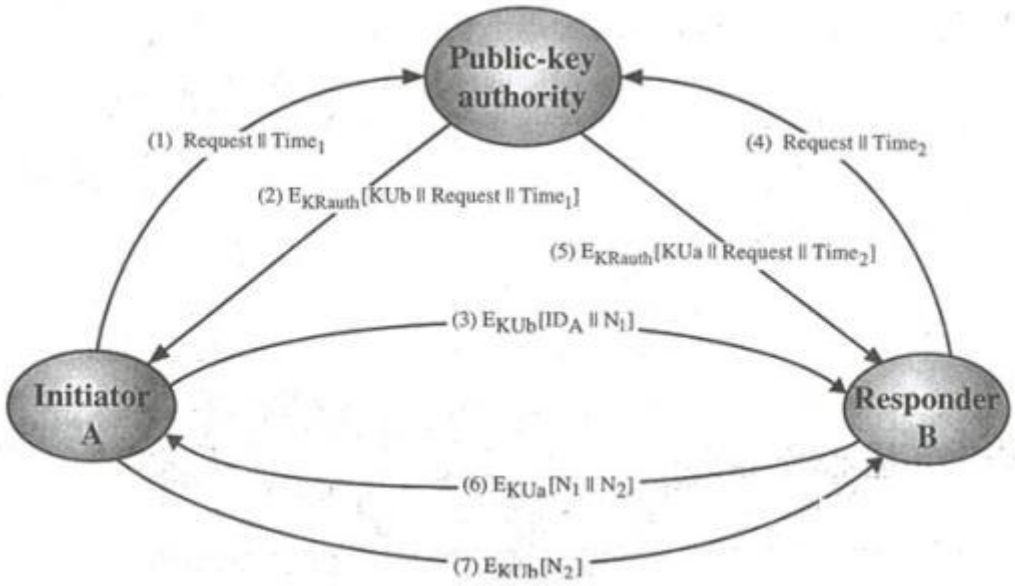
4. Bu sistemin yönetimi, periyodik olarak adres rehberini güncellemeli, bir telefon rehberinde olduğu gibi isimleri ve genel anahtarlarının kopyasını saklamalı ve güncellemeleri, geniş bir kitleye ulaşabilen yayın organı (gazete gibi) yoluyla diğer kullanıcılara haber vermelidir.

5. Üyeler aynı zamanda adres rehberine elektronik yolla da ulaşabilmelidirler, bunun için de adres rehberinin yönetimi, gizlilik ve kimlik denetimi gerektiren güvenli bir iletişim metodu kullanılması zorunluluğunu yerine getirilmiş olmalıdır.

Bu yapının bireysel genel duyuru metodundan daha güvenli olduğu oldukça açıktır fakat, halen kimi savunmasız noktalar bulunmaktadır. Eğer bir saldırgan, adres rehberi yöneticisinin özel anahtarını bir şekilde ele geçirir ya da onu hesapsal metodlarla bulmayı başarırsa, kolay bir şekilde güvenlik engellerini aşarak kişilerin açık anahtarlarını istediği açık anahtarlar ile değiştirip, onlara gelecek olan mesajları okuyabilir, ve istediği kişiyi taklit ederek diğer üyelere onun adına mesaj atabilir.

Açık Anahtar Yetkilisi

Genel anahtar yönetimi için daha güçlü bir güvenlik, herkez tarafından erişilebilir adres defteri üzerinde daha sıkı bir kontrol uygulanması ile elde edilebilir. G. L. Popek ve C. S. Kline'ın 1979 yılında yayımladıkları "Encryption and Secure Computer Networks" isimli makaleleri baz alınarak oluşturulmuş tipik senaryo Şekil 12`de gösterilmiştir. Bu yapıda da, önceki yapıda olduğu gibi, bütün üyelerin açık anahtarlarının saklandığı ve dağıtımının yapıldığı dinamik bir merkezi rehberin varlığı söz konusu. Buna ek olarak, her üye yöneticiye ait olduğunu bildikleri bir açık anahtara sahipler. Şekil 12`de de numaralanmış olan aşağıdaki adımlar ile anahtar dağıtımı gerçekleşir:



Şekil 12

1. A kullanıcısı, B kullanıcısının şu anki açık anahtarını öğrenmek istediğini ifade eden ve zaman ile imzalanmış (timestamped) bir mesajı genel anahtar yöneticisine gönderir.

2. Yönetici, kendi özel anahtarı olan KR_{auth}ile şifrelenmiş bir mesajı A kullanıcısına cevap olarak gönderir. Bu sayede, A kullanıcısı bu mesajı yöneticinin açık genel anahtarı ile deşifre eder ve bu mesajın yöneticiden geldiğine emin olur. Mesaj aşağıdakileri ihtiva eder:

    - A kullanıcısının B kullanıcısı için mesaj yollayabilmesi için B kullanıcısının genel anahtarı,
    - Orijinal istek; A kullanıcısının, yöneticiye gönderdiğini düşündüğü istek ile, yöneticinin aldığını bildirdiği isteğin aynı olup olmadığını karşılaştırması için,
    - Orijinal zaman imzası; bu sayede A kullanıcısınınnın gelen mesajın eski olmadığı ve dolayısıyla, gönderilmiş genel anahtarın B kullanıcısı yerine bir başkasının genel anahtarı olmadığından emin olması için.

3. A kullanıcısı B`nin genel anahtarını kullanarak, içierisinde $A$`nın tanımlayıcısı (ID_A) ve bu iletişimin tanımlayıcısı olan bir kelime (N_1) içeren mesajı şifreler ve B kullanıcısına gönderir.

4. 4. ve 5. adımlarda, B kullanıcısı A kullanıcısının genel anahtarını öğrenmek üzere yöneticiye 1. adımda A`nın yaptığı gibi bir mesaj gönderir ve 1. ve 2. adımlardaki iletişimin aynısı, yönetici ile B arasında gerçekleşir.

Bu noktaya gelindiğinde, genel anahtarlar A ve B`ye güvenli şekilde iletilmiştir. Ve artık bu kullanıcılar kendi aralarında güvenli iletişime başlayabilirler. Yine de, ekstra iki adım daha eklenebilir:

5. B kullanıcısı A`nın genel anahtarını kullanarak, $A$`nın tanımlayıcı kelimesini (N_1) ve kendi yarattığı yeni bir tanımlayıcı kelimeyi (N_2) şifreler ve gönderir. Çünkü (3) mesajını sadece B deşifre edebilir, ve B`nin $N_1$`i (6) mesajı ile döndürmesi, A`nın konuştuğu kişinin umduğu B olduğundan emin olmasını sağlayacaktır.

6. A kullanıcısı da B`nin genel anahtarı ile, içerisinde $N_2$`nin bulunduğu mesajı şifreleyip B`ye gönderir ve böylece, B konuştuğu kişinin umduğu A olduğundan emin olur.

Görüldüğü gibi, toplam 7 mesaj gerekmektedir. Bununla beraber, ilk dört adım her mesajlaşma istendiğinde gerçekleştirilmeyebilir, çünkü lullanıcılar birbirlerinin genel anahtarlarını daha sonraki kullanımlar için saklayabilirler. Bir kullanıcı periyodik olarak, sürekli konuştuğu kişilerin genel anahtarlarının halen geçerli olduğundan emin olmak için genel anahtar isteğinde bulunmalıdır.

Açık Anahtar Sertifikası

Şekil 12`de verilen yöntem çok çekici olmasına karşın hala bazı açık noktaları vardır. Genel anahtar yöneticisi, sistemin önemli bir dar boğazını teşkil etmektedir. Bir kullanıcı temasa geçmek istediği her kullanıcı için genel anahtarlarını istemek üzere yöneticiye başvurmalıdır. Yine öncekinde olduğu gibi, isim rehberi ve anahtarları, çalınmaya ya da müdahalelere karşı yönetici tarafından korunmaya çalışılmaktadır.

Alternatif bir yöntem, ilk kez L. M. Kohnfelder tarafından, 1978 yılında yayımlanan "A Method for Certification" isimli makalede sunulmuştur. Bu yönteme göre kullanıcılar sertifika kullanarak, bir genel anahtar yöneticisine başvurmadan güvenli bir yolla anahtarlarını değiştirebilmekte ve güvenli iletişime başlayabileceklerdir. Bir sertifika yöneticisi/yetkilisi tarafından oluşturulmuş sertifika, bir genel anahtarı ve ek bilgiyi içerir. Ve bu sertifika üzerindeki genel anahtara uygun özel anahtar ile beraber kullanıcıya verilir. Bir kullanıcı diğer bir kullanıcıya açık anahtarını, sertifikasını göndererek gösterir. Diğer kullanıcı da, bu sertifikanın bir otorite tarafından yaratılmış olduğunu doğrulayabilir. Bu yöntemin gerekliliklerini şu şekilde sıralayabiliriz:

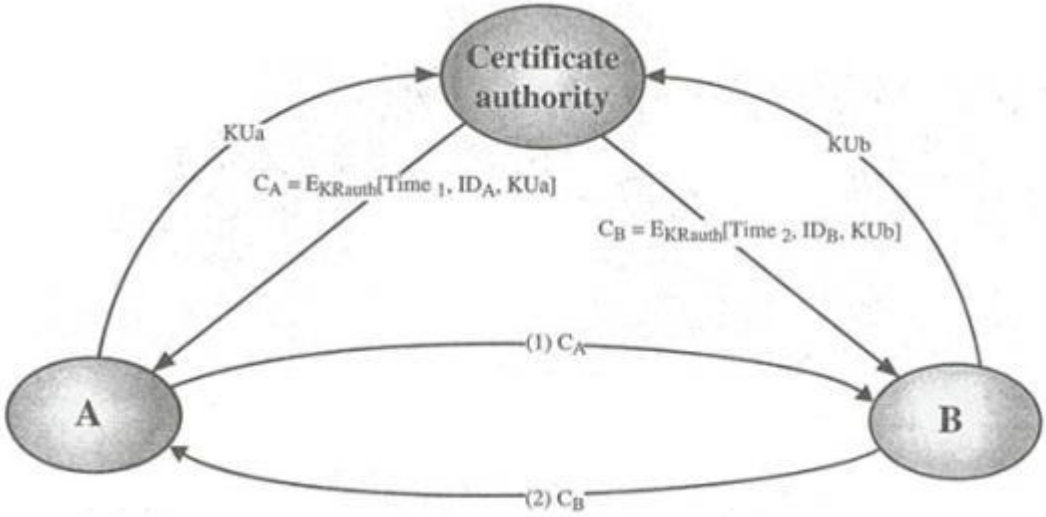
1. Herhangi bir kullanıcı bir sertifikanın kullanıcısının adını ve açık anahtarını öğrenmek üzere okuyabilir.
2. Herhangi bir kullanıcı bu sertifikanın, sertifika yöneticileri tarafından oluşturuluş orijinal bir sertifika olduğunu kanıtlayabilir.
3. Sertifikaları sadece sertifika yöneticisi/otoritesi değiştirebilir ve yaratabilir.

Bu gereklilikler, Kohnfelder'in makalesinde geçen gereklilikler. Bu gerekliliklere 1983 yılında D. E. Denning tarafından 1983 yılındaki "Cryptography and Data Security" isimli makalesi ile şu gereklilik de eklenmiştir:

4. Herhangi bir kullanıcı bir sertifikanın geçerliliğini kanıtlayabilir.

Bir sertifika şeması Şekil 13`te gösterilmiştir. Her kullanıcı bir genel anahtara ve bir sertifikaya sahip olmak üzere sertifika yöneticisine başvurmalıdır.

Başvurular yüz yüze, ya da kimlik denetimi ve gizlilik sağlanması ile güvenli hale getirilmiş bir iletişim yöntemi ile gerçekleştirilmelidir.



Şekil 13

Sertifika yöneticisi/otoritesi, bir A kullanıcısı için sertifikayı şu şekilde oluşturur:

$$C_A = E_{KR_{AUTH}}[T, ID_A, KU_A]$$

burada KR_{auth}, sertifika otoritesi tarafından kullanılmış olan özel anahtardır. Daha sonra A kullanıcısının sertifikasını gönderdiği bir kullanıcı, sertifikayı şu şekilde okur ve doğrular:

$$D_{KU_{AUTH}}[C_A] = D_{KU_{AUTH}}[E_{KR_{AUTH}}[T, ID_A, KU_A]] = (T, ID_A, KU_A)$$

kullanıcı, sertifikayı deşifre etmek için, otoritenin açık anahtarını kullanır. Çünkü sertifika sadece otoritenin açık anahtarı kullanıldığı taktirde okunabilir hale gelebilir; bu da sertifikanın geldiği yerin gerçekten sertifika otoritesi tarafından

oluşturulduğunu kanıtlar. ID_A ve KU_A, elemanları, kullanıcıya, A kullanıcısının tanımlayıcı bilgilerini ve genel anahtarını sunar. Son olarak T zaman pulu da, sertifikanın son geçerli olabileceği tarihi gösterir.

Bu koşullar altında, bir özel anahtarın tehlikeye düşmesi ile bir kredi kartının kaybedilmesi birbirine benzetilebilir. Kredi kartı kaybedildiğinde, sağlayıcısından işlemlerinin durdurulması istenir, fakat bunda geç kalındığı taktirde sorunlar yaşanabilir.

Gizli Anahtarların Açık Anahtarlı Dağıtımı

Açık anahtarlar dağıtıldığında ya da erişilebilir hale geldiğinde, iki kişi arasındaki iletişim güvenlik altına alınmış olacaktır. Buna rağmen, bazı kullanıcılar, kimi sebeplerden ötürü, gizli veri transferi için geleneksel şifreleme yöntemlerini kullanmak istiyor olabilirler. Bu durumda, geleneksel şifrelemenin gizli anahtarlarının dağıtımı için açık anahtarlı şifreleme yöntemleri çok iyi bir araç olacaklardır.

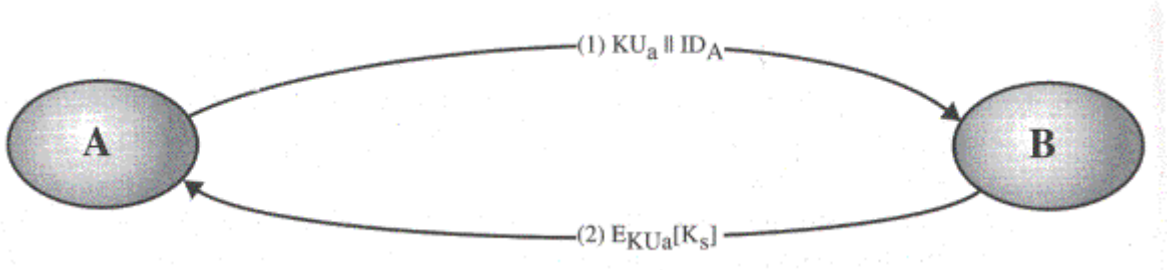
Basit Gizli Anahtar Dağıtımı

Bunun için Şekil 14`te de gösterilmiş son derece basit bir örnek, 1979 yılında R. C. Merkle tarafından ifade edildi. Eğer A, B ile bir iletişim içerisine girmek istiyorsa, aşağıdaki prosedür kullanılabilir:

1. A bir açık/özel anahtar çifti olan (KU_a, KR_a)`yı yaratır ve $B$ kullanıcısına $KU_a$ ve $A$`nın tanımlayıcısı olan ID_A`yı içeren bir mesaj gönderir.
2. B bir gizli anahtar üretir (K_s), ve bunu, A kullanıcısına $E_{KU_a}[K_s]$ şeklinde şifreledikten sonra gönderir.

3. A, $D_{KR_a}[E_{KU_a}[K_s]]$`yi hesaplar ve gizli anahtarı elde eder. Yanlız Amesajı deşifre edebilecek ve $K_s$gizli anahtarı sadece Ave Btarafından bilinecektir.

4. A, KU_ave KR_a`yı, $B$`de KU_a`yı görevleri bittiğinden dolayı yok eder.

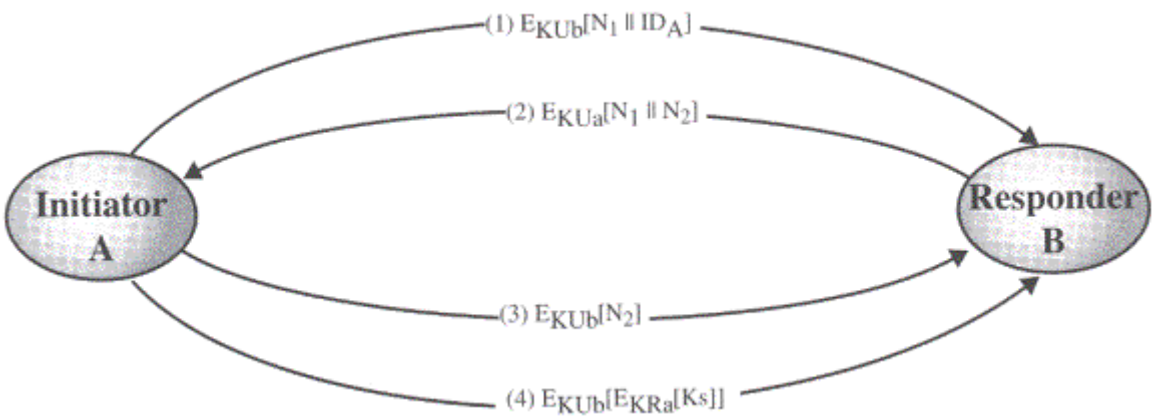


Şekil 14

Artık Ave Bgeleneksel şifreleme ve oturum anahtarı $K_s$ile güvenli şekilde iletişim kurabilirler. İletişimin sona ermesi ile beraber iki kullanıcıda oturum anahtarını yok ederler. Bu yöntem, basitliğine karşın çok hoş bir protokoldür. İletişimin başlamasından önce hiçbir anahtarın olmadığı gibi, iletişimin sonunda da geriye hiçbir anahtar kalmaz. Bu sayede, anahtarların güvenliğini tehlikeye düşürecek riskler minimuma indirgenmiş olur. Aynı zamanda iletişim, pasif ataklara karşı da güvenlik altında gerçekleşmiş olur.

Fakat, bu yöntem, aktif ataklara karşı savunmasız kalmaktadır. Eğer bir rakip (E), iletişim kanalının akışına müdahale etme şansına sahipse, bu kişi, tespit edilmesi için herhangi bir iz bırakmadan aşağıdaki şekilde iletişimin güvenliğine gölge düşürebilir:

1. A bir açık/özel anahtar çifti olan (KU_a, KR_a)`yı yaratır ve $B$ kullanıcısına $KU_a$ ve $A$`nın tanımlayıcısı olan ID_A`yı içeren bir mesaj gönderir.
2. E, iletişimi durdurup, kendi özel/açık anahtar cifti olan (KU_e, KR_e)`yi oluşturur ve $KU_e \| ID_a$ ikilisini B kullanıcısına gönderir.
3. B bir gizli anahtar üretir (K_S), ve bunu, A kullanıcısına $E_{KUe}[K_S]$ şeklinde şifreledikten sonra gönderir.
4. E, iletişimi durdurur ve $D_{KRe}[E_{KUe}[K_S]]$ hesaplaması ile gizli anahtarı öğrenir.
5. E kullanıcısı, $E_{KUa}[K_S]$ mesajını A kullanıcısına gönderir.



Şekil 15

Sonuçta şu gerçekleşmiş olur, hem A kullanıcısı, hem de B kullanıcısı gizli anahtarı bilirler fakat bu anahtardan E`nin de haberdar olduğunu bilmezler. Dolayısıyla, $A$ ve $B$, $K_S$ gizli anahtarını kullanarak mesajlaşmaya başlarlar. $E$`nin artık kanala aktif şekilde müdahale etmesine gerek yoktur; basit şekilde

mesaj trafiğini dinleyip, elindeki gizli anahtar yardımıyla mesajları deşifreleyecek, A ve B kullanıcısının bu problemden haberi olmayacaktır. Böylece, bu basit iletişim sadece, tek tehlikenin ağın pasif olarak dinlenilmesi olduğu ortamlarda kullanılabilir.

Güvenli Gizli Anahtar Dağıtımı ve Kimlik Doğrulama

Şekil 15, N. M. Reedham ve M. D. Shroede'in 1978 yılında yayınladıkları "Using Encryption for Authentication in Large Networks of Computers" isimli çalışmalarında işaret edilen, hem aktif hem de pasif ataklara karşı güvenlik sağlayan yaklaşımı ifade etmektedir. Başlamadan önce, A ve B`nin önceki kısımda açıkladığımız herhangi bir yöntem kullanarak açık anahtarlarını birbirlerine ulaştırdıklarını varsayıyoruz. Bunun sonrasında şu adımlar gerçekleşir:

1. A kullanıcısı, B`nin açık anahtarını kullanarak, içerisinde $A$`nın bir tanımlayıcısı olan ID_A ve bu iletişimi özel olarak ifade etmek üzere bir nonce (N_1) içeren mesajı şifreler ve B`ye gönderir.
2. B kullanıcısı, A`nın açık anahtarını kullanarak, içerisinde $A$`nın gönderdiği nonce (N_1) ve B`nin $A$ gibi hazırladığı yeni bir nonce`u (N_2) içeren mesajı şifreler ve A`ya gönderir. Bu şekilde $A$, $B$ kullanıcısının kimliğinden emin olur çünkü birinci mesajı deşifreleyebilecek tek kişi $B$`dir.
3. A, B`nin de emin olması için $N_2$`yi B`nin açık anahtarı ile şifreleyip kendisine gönderir.
4. A bir gizli anahtar belirler (K_S) ve $M = E_{KUb}[E_{KRa}[K_S]]$ şeklinde oluşturduğu mesajı B`ye gönderir. Mesajın, $B$`nin açık anahtarı ile şifrelenmesi, bu mesajı sadece B`nin okuyabileceğini, $A$`nın özel anahtarı ile şifrelenmiş olması da bu mesajın sadece A tarafından gönderilebileceğini garantiler.

Dikkat etiyseniz, bu yöntemin ilk üç adımı, Şekil 12`de gösterilen yöntemin son üç adımıyla aynıdır. Sonuç olarak, bu yöntem gizli anahtarların dağıtımı esnasında hem güvenliği hemde kimlik doğrulamayı sağlar.

Melez (Hibrit) Bir Yöntem

Gizli anahtarların açık anahtarlı şifreleme yapılarından yararlanılarak dağıtılmasının bir diğer yolu da, halen IBM mainframe`leri üzerinde kullanılan hibrit yapısıdır. Bu yöntem, tüm kullanıcı oturumlarında kullanılacak gizli oturum anahtarlarını bir asıl anahtar ile şifreleyerek dağıtacak olan bir anahtar dağıtım merkezi (KDC: Key Distribution Center) kullanır ve bu merkez gizli asıl anahtarları tüm kullanıcılara paylaştırır. Asıl anahtarların dağıtımı için de açık anahtarlı bir yöntem kullanılır. Aşağıdaki açıklama, bu üç seviyeli yöntemin kullanımını açıklamaktadır.

- **Performans:** Sık sıkanahtar anahtar değiştiren mesajlaşmadan ziyade özellikle işlem yapma amaçlı bir çok uygulama vardur. Oturum anahtarlarının açık anahtarlı şifreleme yöntemi ile dağıtılması, açık anahtarlı şifreleme ve deşifreleme esnasında hesaplamaların oldukça karmaşık olmasından dolayı sistemin büyük ölçüde performans kaybetmesine neden olmaktadır. Bir üç seviye hiyerarşisi ile, açık anahtarlı şifreleme sadece, aradasırada güncellenmesi gereken asıl anahtar (master key)`ın kullanıcılara dağıtılması esnasında gerçekleşir.
- **Geriye doğru uyumluluk:** Hibrit yöntemi kolay bir şekilde mevcut olan KDC (anahtar dağıtım merkezi) üzerine inşa edilebilir.

Bir açık anahtar katmanının eklenmesi, güvenlik ve asıl anahtarların dağıtımı esnasında yüksek verim sağlar. Bu yöntem, bir KDC`nin bir çok kullanıcıya anahtar dağıttığı durum için büyük bir avantaj getirecektir.

DIFFIE-HELLMAN ANAHTAR DEĞİŞİMİ

İnsanlığa ilk duyurulan açık anahtar algoritması, Diffie ve Hellman`ın açık anahtarlı kriptografi olarak tanımladıkları 1976 yılında yayımlanmış "New Directions in Cryptography" isimli makalelerinde yer aldı ve bu algoritma kriptografik sisteme örnek olarak Diffie-Hellman Key Exchange idi. Bir çok ticari uygulama bu anahtar değişimini kullandı.

Algoritmanın amacı, iki kullanıcının bir anahtarı güvenli şekilde birbirlerine iletmeleri ve daha sonrasında da bu anahtar yardımı ile şifreli mesajları birbirlerine gönderebilmelerini sağlamaktı. Algoritma anahtar değişimi ile sınırlıdır.

Diffie-Hellman algoritması efektifliğifliğini, ayrık logaritmik ifadelerin hesaplanmasının zorluğundan almaktadır. Kısaca, ayrık logaritmayı şu şekilde tanımlayabiliriz: Önce, bir asal sayı olan p`nin öyle bir pirimitiv kökünü seçeriz ki, bu sayının kuvvetleri 1`den $p-1$`e kadar olan tüm tam sayıları yaratabilir.Eğer $a$, asal sayı olan $p$`nin pirimitiv kökü ise, bu durumda sayılar

$$a \bmod p, a^2 \bmod p, \ldots, a^{p-1} \bmod p$$

sayıları birbirinden farklı, ve 1`den $p-1$`e kadar olan tam sayıları bir permütasyonla oluştururlar.

Herhangi bir tamsayı b ve asal sayı olan p`nin bir pirimitiv kökü olan a için, aşağıdaki eşitliği sağlayacak bir tane ve sadece bir tane olan i üssü bulunabilir:

$b = a^i \bmod p \quad 0 \le i \le (p-1)$ koşulunu sağlayan.

i üssü, $a \bmod p$ merkezindeki b`nin, ayrık logaritması ya da indexi olarak adlandırılır ve **$\mathbf{ind_{a.b}(b)}$** notasyonu ile ifade edilir.

Global Public Elements

q	Prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Generation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Generation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

Şekil 16

Bu altyapıyı da anladıktan sonra, Şekil 16`da ifade edilmiş olan Diffie-Hellman anahtar değişimini açıklayabiliriz. Bu yöntemde, herkezce bilinen iki sayı vardır: bir asal sayı olan q ve bu sayının pirimitiv kökü olan α. Diyelim ki bir anahtarı değiştirmek isteyen A ve B adında iki kullanıcı var. A kullanıcısı $X_A < q$ olacak şekilde rastgele bir X_A tamsayısı seçer ve, $Y_A = \alpha^{X_A} \bmod q$ değerini hesaplar. Benzer şekilde B kullanıcısı diğer kullanıcan bağımsız şekilde $X_B < q$ olacak şekilde bir X_B tamsayısı seçer ve o da, $Y_B = \alpha^{X_B} \bmod q$ değerini hesaplar. Her iki kullanıcıda X değerini özel olarak saklar ve Y değerini diğer taraf ile paylaşır. A kullanıcısı anahtarı $K = (Y_B)^{X_A} \bmod q$, B kullanıcısı da, $K = (Y_A)^{X_B} \bmod q$ şeklinde hesaplarlar. Bu iki hesaplamanın sonucunun birbiri ile aşağıdaki modüler aritmetik kuralları ile ispatlanabilir:

$$
\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}
$$

Bu sayede, iki tarafta bir gizli anahtarı değişmiş olur. Ayrıca, X_A ve X_B`nin gizli olmasından dolayı, herhangi bir saldırgan çalışmak için sadece şu değerleri bilir: q, a, Y_A, Y_B. Örneğin bir saldırganın B kullanıcısına ait olan gizli anahtarı bulmak için aşağıdaki hesaplamayı yapması gerekir:

$$X_B = \text{ind}_{\alpha,q}(Y_B)$$

Bu hesaplamanın başarısı sonrasında saldırgan gizli anahtarı B kullanıcısının hesapladığı gibi hesaplayabilecektir.

Burada, [SEBE89 (Referansı kaybetmişim, en kısa zamanda ekleyeceğim)] çalışmasından alınmış bir örnek var. Anahtar değişimi için asal sayı olan $q = 97$ ve pirimitiv kökü olarakta $\alpha = 5$ alınmış. A ve B kullanıcısı da gizli anahtarlarını $X_A = 36$ ve $X_B = 58$ olarak seçmişler. İkisi de açık anahtarlarını şu şekilde hesaplarlar:

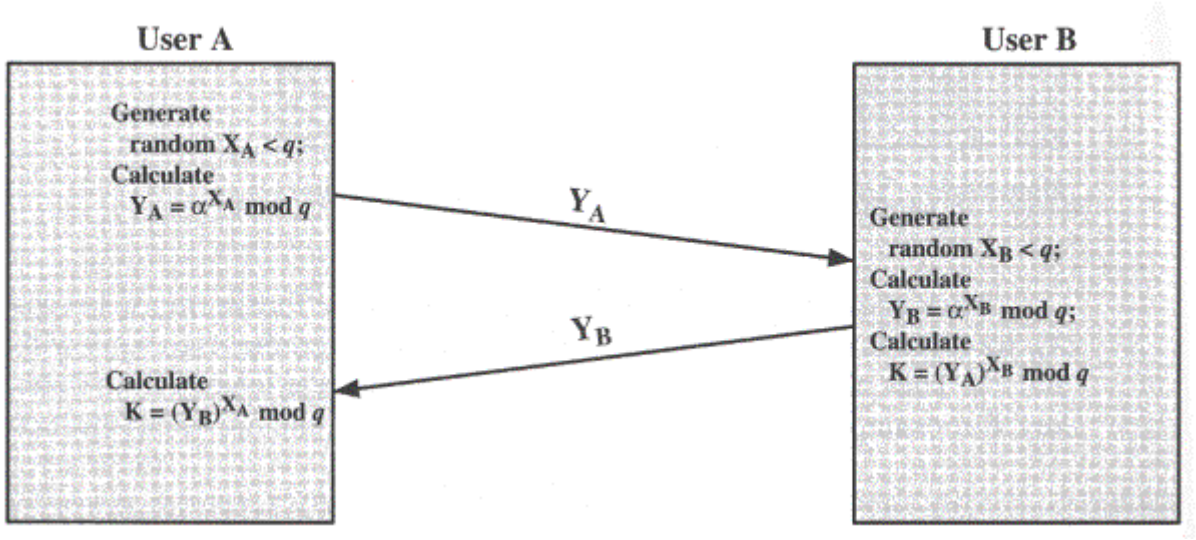
$$Y_A = 5^{36} = 50 \bmod 97$$
$$Y_B = 5^{58} = 44 \bmod 97$$

Açık anahtarlarını değiştikten sonra, gizli anahtarlarını da şu şekilde hesaplarlar:

$$K = (Y_B)^{X_A} \bmod 97 = 44^{36} = 75 \bmod 97$$
$$K = (Y_A)^{X_B} \bmod 97 = 44^{36} = 75 \bmod 97$$

{50,44} bilgisine sahip olan bir saldırgan, 75 değerini kolayca hesaplayamayacaktır. Ve bu hesapsal zorluk seçilen sayıların büyüklüğü ile orantılı olarak artacaktır.

Şekil 17, Diffie-Hellman hesaplamasını basit şekilde ifade etmektedir. Diffie-Hellman algoritmasının LAN (yerel network) kullanıcıları arasında kullanımı için bir diğer örnekte şu şekilde verilebilir: Diyelim ki bu ağ altında çalışan her kullanıcı dayanıklı ve uzun birer X_A ve buna bağlı genel bir Y_A hesaplamış olsunlar. Kişilerin açık anahtarları ve herkezce bilinen q ve α değerleri herkezin erişebileceği merkezi bir rehberde tutulduğu taktirde, herhangi bir anda bir B kullanıcısı mesajlaşmak istediği bir A kullanıcısının açık değerine ulaşabilecek ve onun için şifrelediği mesajı kendisine gönderebilecektir. Eğer merkezi rehber güvenilir ise, bu iletişim gizliliği ve kimlik denetimini sağlamış olacaktır. Tüm bunlara rağmen bu teknik, aktif tekrarlama gibi ataklara karşı korumasız kalmaktadır.



Şekil 17

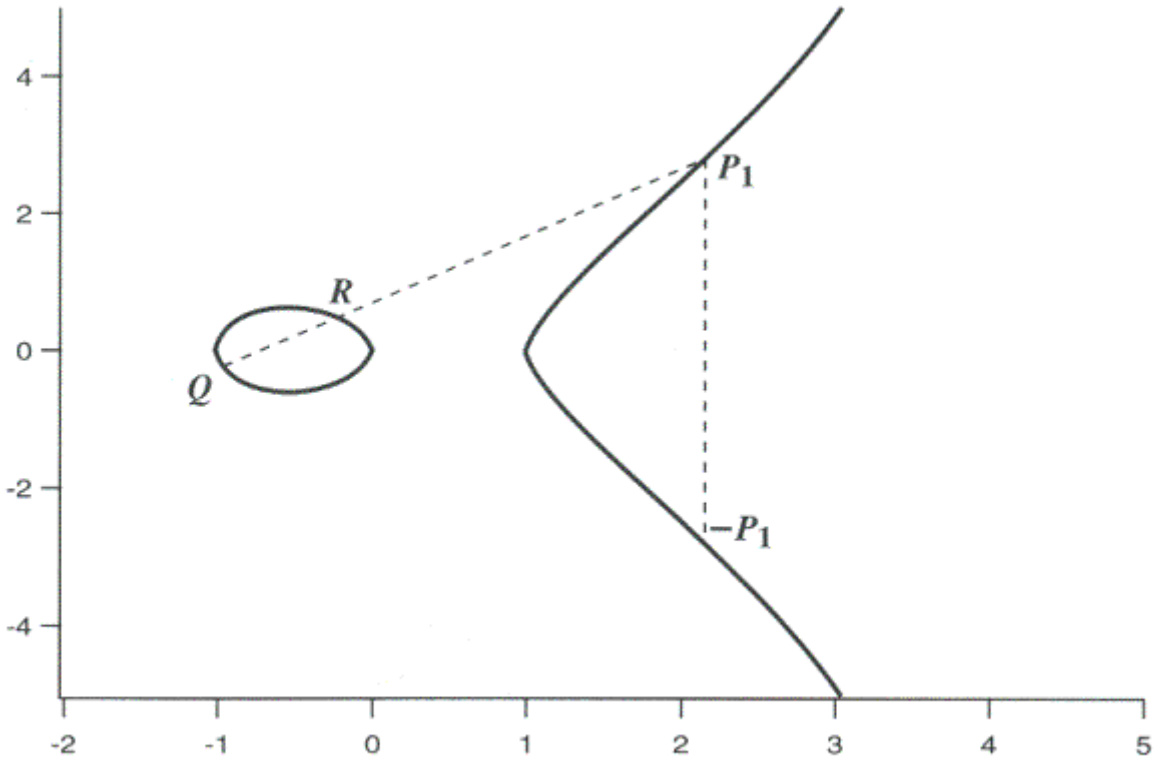
ELİPTİK EĞRİ KRİPTOGRAFİSİ

Açık anahtarlı kriptografi kullanan standartların ve ürünlerin hemen hemen hepsi, şifreleme ve dijital imza için RSA kullanmaktadır. Daha önceki bölümlerde de gördüğümüz gibi RSA`nın güvenli kullanımı için, çalışılan bit uzunlukları zaman içerisinde büyümüş ve dolayısıyla RSA kullanan uygulamalar üzerine büyük bir hesapsal ağırlık getirmiştir. Özellikle büyük sayıların transferini güvenlik içinde gerçekleştirmesi gereken ticari siteler bundan çok fazla etkilenmişlerdir. Son zamanlarda, RSA`ya karşı rakip olarak geliştirilmiş bir sistem ortaya atıldı: Eliptik Eğri Kriptografisi (ECC). ECC, şimdiden açık anahtarlı kriptografi için öngörülmüş IEEE P1363 standartlarını yerine getiriyor.

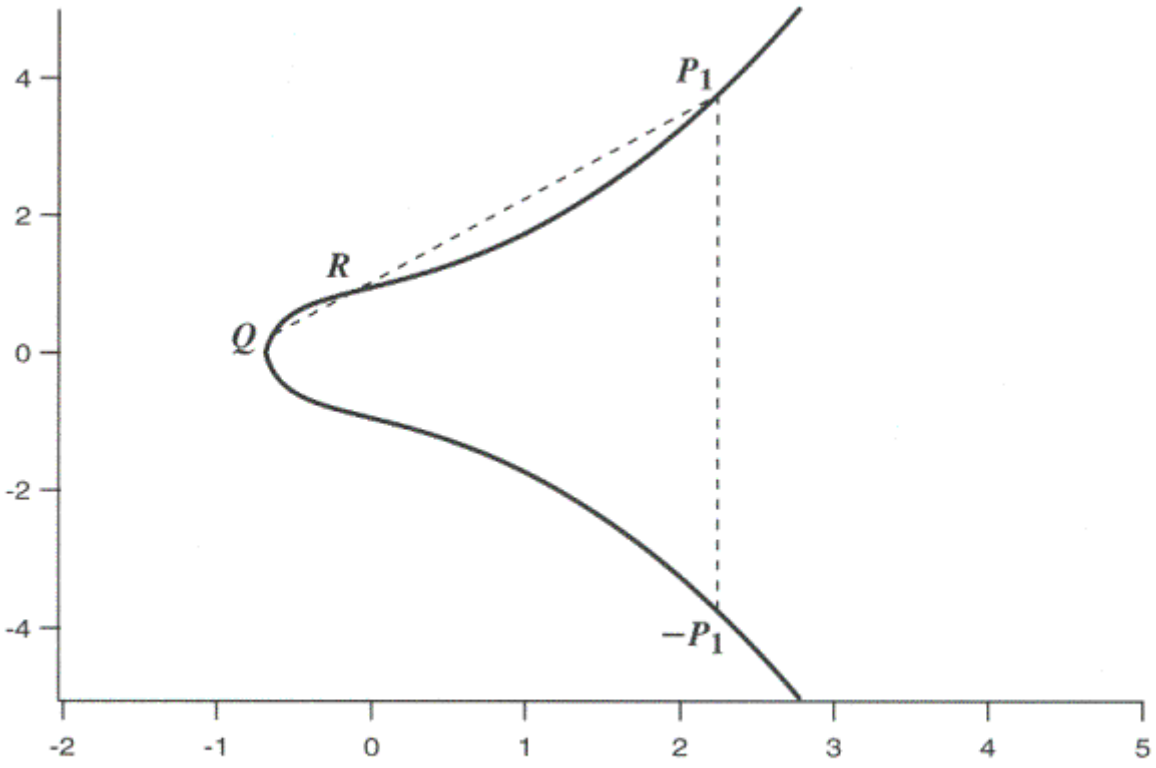
ECC`nin RSA`ya karşı en büyük avantajı, daha küçük bitler kullanılarak yapılan işlemlerin RSA gibi yüksek güvenlik sağlayabilmesi, bunun sayesinde kullanıcıların şifreleme ve deşifreleme esnasında hesaplamalar için harcadıkları

efor azalıyor. Diğer bir taraftan da, hernekadar ECC`nin teorisi kısa bir süre önce ortaya atıldıysa da, ECC kullanan ürünler kısa süre içerisinde kendisini göstermeye başladı ve bu ürünler üzerinde yapılan güvenlik testleri ECC`nin henüz RSA`nın sağladığı kadar yüksek bir güvenlik sağlayamadığını gösterdi.

ECC`nin matematiksel teorisinin açıklanması RSA ya da Diffie-Hellman teorilerinin açıklanmasından daha zordur ve, tam matematiksel açıklaması bu yazıda verilmeyecektir; bu kısım ECC ve eliptik kriptografi üstüne küçükte olsa bir altyapı oluşturmak üzere hazırlanmıştır.



(a) $y^2 = x^3 - x$



(b) $y^2 = x^3 + x + 1$

Şekil 18

Eliptik Eğriler

Eliptik eğriler elips değildirler [:)]. Bu şekilde adlandırılmalarının sebebi, bir elipsin çemberinin hesaplanması için kullanınan kübik denkliklere benzer ifadeler ile gösterilmeleridir. Genel olarak, eliptik eğriler için kübik denklemler aşağıdaki formdadır:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

bu denklemdeki a,b,c,d ve e sayıları reel sayılardır ve bazı basit koşulları sağlarlar. Ayrıca, eliptik eğrinin tanımlamasında, daha sonra daha ayrıntılı şekilde inceleyeceğimiz, sonsuzluk yada sıfır nokta adı verilen bir O notasyonu vardır. En büyük dereceli üs 3 olduğundan dolayı bu tip denklemler kübik olarak adlandırılırlar. Şekil 18 iki eliptik eğri örneği göstermektedir. Gördüğünüz gibi formüller bazen garip görünüşlü eliğptik eğriler doğurmaktadırları...

Eğer bir eliptik eğrinin 3 noktası düz bir çizgi üzerinde bulunuyorsa, bunlar O olarak özetlenir. Bu açıklamadan yola çıkarak, bir eliptik eğri için şu kuralları tanımlayabiliriz:

1. Elptik eğri üzerindeki herhangi bir P noktası için, $P + O = P$ olur.
2. Bir dikey çizgi, aynı x değeri için eliptik eğriyi $P_1 = (x, y)$ ve $P_2 = (x, -y)$ gibi iki noktasında kesiyorsa, Bu çizgi aynı zamanda eliptik eğriyi sonsuzluk noktasında da kesiyordur. Bu yüzden, $P_1 + P_2 + O = O$ ve $P_1 = -P_2$ olur. Böylece bir noktanın negatifi, x ekseni üzerinde aynı değeri alacak şekilde bir noktadır ve bu noktanın y ekseni üzerindeki değeri ilk noktanın negatiflisidir. Bunu Şekil 18`de de görebilirsiniz.
3. x koodrinatı farklı olan Q ve R noktası seçip bu iki noktadan geçen düz bir çizgi çizdiğimizde kesişimin üçüncü noktası olan P_1`i buluruz. Ve cok kolay bir şekilde görülebilir ki, P_1 noktası sadece bir tanedir (eğer

çizdiğimiz doğru, Q veya R noktalarından birisinden teğet geçiyorsa bu durumda $P_1 = Q$ veya $P_2 = R$ alırız). Bu durumda $Q + R + P_1 = O$ ve dolayısıyla $Q + R = -P_1$ olacaktır. 18 numaralı şekli inceleyiniz.

4. Bir Q noktasını çift katlı yapmak için, bir teğet çizgisi çizip eğriyi kestiği diğer noktayı buluruz. Eğer bu noktaya S diyecek olursak $Q + Q = 2Q = -S$ eşitliği sağlanır.

Sonlu Alanlardaki Eliptik Eğriler

ECC için, eliptik eğrilerin, "sonlu alanlardaki eliptik eğriler" olarak tanımlanan bir formu ile ilgileneceğiz. Bu şu şekilde gösterilir: p bir asal sayı olsun ve a ve b, p `den küçük, negatif olmayan iki tam sayı olsun:

$$4a^3 + 27b^2 \pmod{p} \neq 0$$

Bu durumda, $E_p(a,b)$, (x,y) `nin $p$ `den küçük negatif olmayan tam sayılar olduğu durum için, O sonsuz noktası ile beraber şu eşitliği ifade eder:

$$y^2 \equiv x^3 + ax + b \pmod{p} \quad (1)$$

Örneğin, $p = 23$ ve eliptik eğrimiz de $y^2 = x^3 + x + 1$ olsun. Bu durumda, $a = b = 1$ olur. Bu durumda, 4 x 1^3 + 27 x 1^2 (mod 23) = 8 ≠ 0, bizim eliptik grubumuzun mod 23`e göre durumunu gösterir.

Önceki paragrafta gösterilen eşitlik, 6 şeklindeki ikinci grafik ile aynıdır. Eliptik grup için sadece (mod p`den dolayı), (0, 0)-(p, p) aralığında olan pozitif tamsayılar ile denklem oluştururuz. Tablo 4`te, $E_{23}(1, 1)$ için O dışındaki noktaları listenmiştir. Genel olarak liste aşağıdaki yolla oluşturulmuştur:

1. $0 \le x < p$ koşulunu sağlayan her x değeri için $x^3 + ax + b (\mathrm{mod}\ p)$ denklemi hesaplanmıştır.
2. Önceki adımın her sonucu için, sonucun $\mathrm{mod}\ p$ `ye göre çift katlı kökü olup olmadığına bakılır, eğer yoksa bu $x$ değeri için, $E_p(a,b)$ `nin bir değeri yoktur. Aksi taktirde, çift katlı kök koşulunu sağlayan iki adet y vardır (y `nin 0 olduğu durum haricinde). Bu (x,y) değerleri $E_p(a,b)$ `nin noktalarıdır.

Tablo 4 - $E_{23}(1, 1)$ Eliptik eğrisi için noktalar.

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

$E_p(a,b)$ için bahsedilmiş kurallar, Şekil 18`deki geometrik şekil ile uyuşmaktadır. Kurallar, her $P,Q \in E_p(a,b)$ olacak şekilde alınan noktalar için aşağıdaki gibi gösterilebilir:

1. $P + O = P$.
2. Eğer $P = (x,y)$ ise, $P + (x,-y) = O$ olur. $(x,-y)$ noktası, P `nin negatifidir ve $-P$ olarak gösterilir. Diyelim ki, $(x,-y)$ Şekil 18b şeklinde gösterilen

$E_p(a,b)$ eliptik eğrisi üzerinde bir nokta. Örneğin, $E_{23}(1, 1)$` de $P=(13,7)$ alalım. Bu durumda $-P=(13,-7)$ olacaktır. Fakat –7 mod 23 e göre 16 ettiğinden dolayı, bizim $-P$ noktamız aslında, yine $E_{23}(1, 1)$`de yer alan (13,16) noktasıdır.

3. Eğer $P=(x_1,y_1)$ ve $Q=(x_2,y_2)$ ise ve $P\neq -Q$ ise, bu durumda, $P+Q=(x_3,y_3)$ şu kuralla hesaplanır:

$$x_3 \equiv \lambda^2 - x_1 - x_2 (\mathrm{mod}\ p)$$
$$y_3 = \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\ p)$$

λ için koşul,

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & eger \quad P \neq Q \quad ise \\ \dfrac{3x_1^2 + a}{2y_1} & eger \quad P = Q \quad ise \end{cases}$$

Eliptik Eğriler İle Kriptografi

ECC`deki toplama işlemi ile RSA`daki modüler çarpım işlemi ve çoklu toplama işlemi ile de, RSA`daki modüler üs alma işlemi birbirlerine çok benzemektedir. Eliptik eğriler kullanan bir kriptografik sistem oluşturabilmek için, bir sayıyı iki asal çarpanına ayırmak ya da ayrık logaritma almak gibi zor bir problem bulmamız gerekmektedir.

Diyelim ki, $P,Q \in E_p(a,b)$ ve $k<p$ iken, $Q=kP$ olsun. k ve P verildiğinde Q değerini hesaplamak nispeten kolay olduğu halde, Q ve P verildiğinde k değerini hesaplamak gerçekten çok zordur.

Bu kısımda, ECC`ye lezzetini veren iki yöntemi inceleyeceğiz.

Diffie-Hellman Anahtar Değişimi Örneği

Eliptik eğriler kullanılarak anahtar değişimi aşağıdaki şekilde yapılabilir. Önce $p \approx 2^{180}$ olacak şekilde bir p asal sayısı ve 1 denklemindeki eliptik eğri parametreleri olan a ve b seçilsin. Bu, eliptik noktalar grubu olan $E_p(a,b)$`yi oluşturdu. Sonrasında $E_p(a,b)$ içerisinden, başlangıç noktası (generator point) olacak olan $G=(x_1,y_1)$ seçilsin. $G$`nin serçilmesindeki en önemli kriter, $nG=O$ eşitliğini sağlayan en küçük n değerinin çok bir büyük bir asal sayı olması gerekliliğidir. Artık $E_p(a,b)$ ve G, kriptosistemin tüm katılımcılarca bilinecek parametreleri oldular.

Bir A ve B kullanıcısı arasındaki anahtar değişimi aşşağıdaki gibi gerçekleşir:

1. A, n`den küçük bir $n_A$ tamsayısı seçer. Bu $A$`nın özel anahtarıdır. Daha sonra A, $P_A=n_AG$ hesabıyla $E_p(a,b)$`nin bir noktası olan kendi açık anahtarını oluşturur.
2. B`de aynı metodla kendi açık anahtarı $P_B$`yi oluşturur.
3. A gizli anahtarı $K=n_AP_B$ ile, B`de gizli anahtarı $K=n_BP_A$ ile elde eder.

Üçüncü aşamadaki iki hesaplamanın sonucuda aynıdır. Çünkü,

$$n_AP_B=n_A(n_BG)=n_B(n_AG)=n_BP_A$$ eşitliği mevcuttur.

Bu yönteme bir atak gerçekleştirmek isteyen saldırgan, verilmiş G ve kG değerlerinden yola çıkara k değerini hesaplamak isteyecektir ve bu çok zordur.

Bu konuda bir örnek verelim: $p=211$ olarak alalım $E_p(0,-4)$, eliptik eğri $y^2=x^3-4$ ve $G=(2,2)$ olsun. Hesapladığımız taktirde görürüz ki,

$241G = O$ olacaktır. A`nın özel anahtarı $n_A = 121$ ve bu durumda bu kullanıcının genel anahtarı $P_A = 121(2,2) = (115,48)$. $B$`nin özel anahtarı $n_B = 203$ ve bu durumda bu kullanıcının genel anahtarı $P_B = 203(2,2) = (130,203)$. Bu koşullar altında paylaşılmış gizli anahtar $121(130,203) = 203(115,48) = (161,169)$ olur.

Görüldüğü gibi, anahtar iki parçadan oluşuyor. Eğer bu anahtar geleneksel şifreleme için bir oturum anahtarı olarak kullanılacaksa, sadece bir sayının yaratılması gerekir. Basit olarak sadece x koordinatını ya da y koordinatını anahtar olarak kullanabiliriz.

Eliptik Eğri Şifrelemesi/Deşifrelemesi

Literatürde, eliptik eğriler yardımıyla şifreleme/deşifreleme yapan bir çok yöntem bulunmaktadır. Biz burada elbette en basitini inceleyeceğiz. Bu sistem içerisindeki ilk görev, plaintext mesaj olan m`yi, bir x-y koordını ile belirlenmiş P_m noktası şeklinde göndermek üzere encode etmektir. Bu P_m noktası bir chipertext gibi şifrelenecek daha sonrasında da deşifre edilecektir. Bir mesajı x ya da sadece y koordinat noktası olarak basitçe encode edemeyiz çünkü, tüm olası noktalar $E_p(a,b)$ içinde bulunmayabilir. Elbette bu encode işlemi içinde bir çok yöntem mevcut fakat burada bu yöntemlerden bahsetmeyeceğiz.

Anahtar değişimi sisteminde olduğu gibi şifreleme / deşifreleme sistemi de parametre olarak bir G noktası ve bir $E_p(a,b)$ eliptik grubuna ihtiyaç duyar. Her bir kullanıcı, bir n_A özel anahtarı seçer ve $P_A = n_A G$ ile bir açık anahtar üretir.

P_m gibi bir mesajı şifrelemek ve bir B kullanıcısına göndermek isteyen bir A kullanıcısı, rastgele pozitif bir k tam sayısı seçer ve C_m chipertextinin noktalarını aşağıdaki şekilde elde eder:

$$C_m = \{kG, P_m + kP_b\}$$

A kullanıcısının şifreleme esnasında B`nin açık anahtarı olan $P_b$`den yararlandığına dikkat ediniz. B aşağıdaki şekilde mesajı deşifre eder:

$$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$$

A, P_m mesajını, ona kP_B ekleyerek maskeledi. k değerini bilmeyen kimse A`nın uyguladığı maskeyi kaldıramaz. Bunun yanında A, n_B özel anahtarını bilen bir kişinin maskeyi kaldırabilmesi için bir ipucu bırakmaktadır.

Eliptik Eğri Kriptografisinin Güvenliği

ECC`nin güvenliği, $kP$ ve $P$ verildiğinde $k$ değerini elde etmenin zorluğuna bağlıdır. Bu durum, eliptik eğri logaritması problemi olarak adlandırılır. Eliptik eğri logaritması alan bilinen en hızlı teknik Pollard rho (rho=eşkenar dörtgen) yöntemidir. Tablo 5`te, eliptik eğri metodu ile RSA`daki tamsayının iki asal çarpanına generalized number field sieve yöntemi ile ayrılması esnasında gereken hesapsal efor karşılaştırılmıştır. Tablodan da anlaşılacağı üzere, RSA`nın sağladığı direnci ECC, çok daha düşük anahtar boyutları ile sağlamakadır. Bu yüzden ECC, düşük anahtar boyutu ile sağladığı yüksek güvenlik sayesinde RSA`ya karşı büyük bir hesapsal üstünlük sağlamaktadır.

Key Size	MIPS-Years
150	3.8×10^{10}
205	7.1×10^{18}
234	1.6×10^{28}

(a) Elliptic Curve Logarithms Using the Pollard rho Method

Key Size	MIPS-Years
512	3×10^{4}
768	2×10^{8}
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

(b) Integer Factorization Using the General Number Field Sieve

Tablo 5