TERS SSL: SSL SUNUCULARINDAKİ PERFORMANS PROBLEMİNE YENİ BİR ÇÖZÜM YÖNTEMİ

Kemal Bıçakcı
TOBB Ekonomi ve Teknoloji Üniversitesi
bicakci@etu.edu.tr

ABSTRACT

Today, the most widely used security protocol is undoubtedly the SSL protocol. SSL is used not only for the security of the web transactions but also it becomes the de facto standard to provide the security for the communication between the client and server in almost all networking applications. SSL is composed of two subprotocols. With "SSL Handshake", using public-key cryptography two parties agree on a common symmetric key. Afterwards, using the common secret key the secure communication is realized.

It is well known that the source of performance problem in SSL is SSL handshake which is based on public-key cryptography that is much slower than symmetric cryptography. In this paper we show a novel method that improves the performance of SSL handshake using online/offline signatures. In other words, by increasing the number of clients that can be served in unit time, we aim to clear away or minimize the requirement for specialized SSL accelerator hardware devices.

Key words: cryptography, security protocol, digital signature, secure socket layer

1. GİRİŞ

Günümüzde en yaygın olarak kullanılan güvenlik protokolü hiç kuşkusuz SSL protokolüdür. SSL sadece web işlemlerinin güvenliğinde değil neredeyse tüm ağ uygulamalarında sunucu ve istemci arasındaki haberleşmeyi güvenli hale getirmek için tercih edilir hale gelmiştir. SSL protokolü iki alt-protokolden oluşmaktadır. "SSL elsıkışması" ile açık-anahtar kriptografisi kullanılarak ortak bir simetrik anahtar üzerinde iki taraf arasında karşılıklı anlaşma sağlanmakta daha sonra ise ortak simetrik anahtar kullanılarak güvenli haberleşme gerçekleşmektedir.

Çip teknolojisindeki hızlı ilerleme yüksek performanslı işlemcileri beraberinde getirmiştir. Bu durum, SSL'in istemci tarafında sorunsuz çalışmasını garantilerken aynı iyimserlik sunucu tarafında ne yazık ki geçerli değildir. Çünkü sunucular doğası gereği birden fazla istemciye aynı anda hizmet verirler ve hizmet verdikleri ortalama istemci sayısı son yıllarda işlemci hızlarından daha hızlı artmaktadır. Bu gözlemin doğruluğunu özel tasarlanmış SSL hızlandırıcı donanımlarının pazar büyüklüğü teyit etmektedir. 2005 yılında dünyada toplam 1 milyar $ tutarında SSL hızlandırıcı satıldığı tahmin edilmektedir [1].

Bu çalışmada amacımız SSL sunucularının performansını pahalı donanımlara gereksinim duymadan arttırabilmektir. Başka bir ifadeyle SSL sunucularının birim zamanda hizmet verebildikleri istemci sayısını arttırarak SSL hızlandırıcı donanımlarına olan ihtiyacın ortadan kaldırılması veya minimize edilmesi hedeflenmektedir.

SSL'de performans probleminin kaynağının simetrik kriptografiye oranla çok daha yavaş olan açık anahtar kriptografisine dayanan SSL elsıkışması olduğu kesin olarak bilinmektedir. Literatür incelendiğinde konunun pek çok farklı açıdan incelenmiş olduğu görülmektedir. Bununla birlikte bu çalışmanın özgün tarafı SSL elsıkışmasında çevrim-içi/çevrim-dışı imza yapıları kullanılarak performans arttırımının çok daha üst boyutlara taşınmasının mümkün olduğunun gösterilmesi olacaktır.

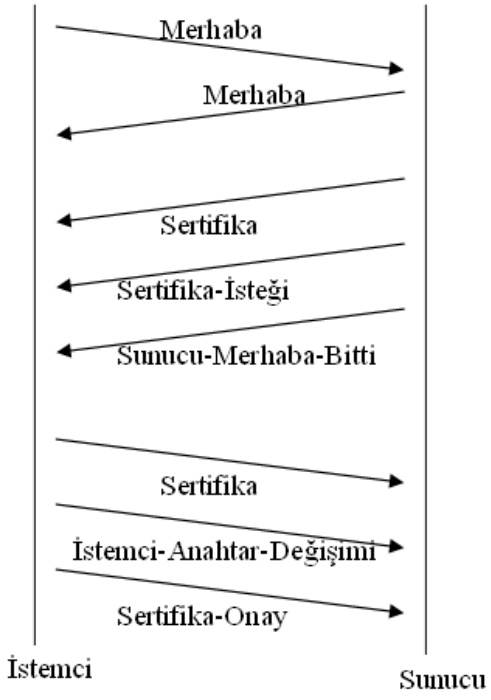
Makalenin geri kalan bölümleri aşağıdaki şekilde planlanmıştır: İkinci bölümde SSL elsıkışma protokolünün orijinal hali, üçüncü bölümde ise çevrim içi/çevrim dışı imzalar tanıtılacaktır. Dördüncü bölümde SSL protokolünde Ters SSL olarak isimlendirdiğimiz değişiklik anlatılacak, beşinci bölümde ise bu değişiklik ile elde edilen performans iyileştirmelerinin hangi seviyelerde olabileceği gösterilecektir. Altıncı bölümde konu ile ilgili önceki çalışmalar özetlenecek ve yedinci bölümde kısa bir özet ile makale sonlandırılacaktır.

2. SSL ELSIKIŞMA PROTOKOLÜ

SSL elsıkışma protokolü sunucu ve istemci arasındaki veri transferini şifrelemek için gerekli olacak gizli anahtarı oluştururken sunucunun ve isteğe bağlı olarak da istemcinin karşı tarafa tanıtılmasını sağlamaktadır. Şekil 1'de istemci tanımalı SSL elsıkışmasında değiş tokuş edilen mesajlar gösterilmektedir.

Elsıkışma protokolü SSL sürümü, kullanılacak şifreleme algoritmaları vb. parametrelerin belirlendiği merhabalaşma ile

başlamaktadır. Daha sonra sunucu istemciye kendi sertifikasını göndermekte ve istemciden sertifikasını istemektedir. İstemci sunucuya kendi sertifikasını gönderdikten sonra SSL elsıkışmasındaki en kritik iki mesajı oluşturup sunucuya iletmektedir. Bu mesajlardan ilki (*İstemci-Anahtar-Değişimi*) sunucunun sertifikasından doğrulanıp elde edilen sunucu açık anahtarı ile şifrelenen bir esas anahtarı (diğer gizli anahtarlar bu esas anahtardan üretilecektir) içermekte, diğeri ise (*Sertifika-Onay*) istemcinin kendi özel anahtarı ile şu ana kadarki tüm iletişimi imzalaması ile oluşturulmaktadır.



Şekil 1 Orjinal SSL Elsıkışması

Bu iki mesajı alınca sunucu kendi özel anahtarı ile esas anahtarın şifresini çözmektedir. Eğer başarılı bir şekilde çözebilirse ki çözemezse daha sonra istemci ile şifreli haberleşme yapması mümkün olmayacaktır bu işlem aynı zamanda sunucunun kimlik bilgisinin doğru olduğunu göstermektedir. Çünkü açık anahtar kriptografideki temel mantık sunucu sertifikasının içerdiği açık anahtar ile şifrelenen bir verinin sadece o açık anahtara karşılık gelen özel anahtara sahip sunucu tarafından açılabileceği üzerine kuruludur. Sunucu bu şekilde istemciye tanıtılırken istemci ürettiği sayısal imzanın istemci sertifikasından elde edilen açık anahtar ile onaylanması ile sunucuya tanıtılmaktadır. Hatırlanmalıdır ki açık anahtar kriptografisindeki ikinci temel kural açık anahtar ile onaylanabilen herhangi bir sayısal imzanın sadece o açık anahtara karşılık gelen özel anahtar ile üretilebilmiş olduğudur.

SSL elsıkışması Şekil 1'de gösterilmemiş olan "Bitti" mesajlarının değiş tokuşu ile bitmektedir. Daha sonraki karşılıklı haberleşmenin gizliliği ve bütünlüğü esas anahtardan üretilen oturum anahtarları ile sağlanmaktadır.

3. ÇEVRİM İÇİ/ÇEVRİM DIŞI İMZALAR

Çevrim-içi / çevrim-dışı imza (ÇÇİ) yapıları ilk defa Even ve arkadaşları tarafından 1990 yılında önerilmiştir [2]. Temel felsefesi sayısal imza üretme işlemini iki aşamada gerçekleştirmektir. Hesaplama olarak yavaş olan çevrim-dışı aşaması imzalanacak mesajdan bağımsız olarak yürütülebilmekte, çevrim-içi aşaması ise mesaj hazır hale geldikten sonra "çok hızlı" bir şekilde tamamlanabilmektedir. Bu bölümde Even ve arkadaşlarının önerdiği ÇÇİ'ler tanıtılmadan önce ÇÇİ'lerde kullanılan tek-zamanlı imzalar (TZİ) tanıtılacaktır. Bu bağlamda literatürde daha farklı yöntemlere dayanan ÇÇİ yapılarının da önerilmiş olduğunu hatırlatırız.

TZİ yeni bir yöntem değildir. 1979 yılında Lamport tarafından bulunmuştur [2]. TZİ sadece tek yönlü işlevleri kullandığı ve herhangi bir matematiksel problemin zorluğuna bağlı olmadığı için RSA, DSS gibi yöntemlere göre çok daha hızlıdır. TZİ'nin nasıl çalıştığını kısaca şu şekilde anlatabiliriz:

Bir mesaja imza hazırlamak için gönderici olarak öncelikle *r* olarak adlandırabileceğimiz rastlantısal sayı üretiriz. Bu sayı bizim özel değerimizdir ve *H* tek-yönlü işlevimizi bu sayıya uyguladığımız anda *H(r)* çıktısı açık değer olur. Daha sonra, *H(r)*'yi güvenli bir şekilde alıcılara göndeririz. İmza *r* değerinin kendisini dağıtmak suretiyle oluşturulur. Alıcılar mesajın sadece gönderici tarafından gönderilebileceğini *r* değerine *H* işlevini uygulayıp *H(r)* çıktısını elde ederek kanıtlarlar. Eğer bu değer imza sertifikasındaki *H(r)* değeriyle aynı ise, TZİ doğru bir imza olarak düşünülür çünkü sadece gönderici *r* değerini bilebilir. Bu yöntem bu haliyle sadece önceden bilinen 1-bit değeri imzalayabilir. Herhangi bir 1-bit değerini ise iki rastlantısal sayı *(r1,r2)* kullanılarak imzalayabiliriz. Bu durumda *H(r1)* ve *H(r2)* değerlerinin ikisini de önceden dağıtırız fakat *(r1,r2)*'den sadece birini imza olarak göndeririz.

Diğer imza yöntemlerinde olduğu gibi imzalanacak mesajın uzunluğu ilk önce mesajımıza mesaj özeti işlevi (örnek: SHA-1, MD5) uygulayarak kısaltılabilir. Bu durumda imzalamamız gereken uzunluk 160-bit (SHA-1 için) veya 128-bit (MD5 için) olacaktır. Yukarıda anlatılan yöntem ile her bir bit için iki rastlantısal sayı kullanıldığını varsayarsak, 160-bitlik bir mesaj için 320 tane rastlantısal sayıya gereksinim duyarız. Fakat bu sayı azaltılabildiği ve 165 sayının 160 bitlik bir mesajı imzalamak için yeterli olduğu

ispatlanmıştır [4]. Yeni yöntemde imza olarak ise bu 165 sayıdan 75'ini özel bir eşleme (mapping) algoritmasıyla seçmemiz gerekmektedir. Özetle, 165 rastlantısal sayı kullanılarak oluşturulan 75 sayılı kombinasyonlardan her biri 160 bitlik mesaj uzayından farklı bir mesajı imzalamak için kullanılmaktadır.

Even ve arkadaşlarının önerdiği ÇÇİ yapıları daha önce belirtildiği gibi TZİ'ye dayanmakta ve normal imzalar gibi üç algoritma içermektedir:

1. Anahtar Üretimi: Normal imzalar ile aynıdır. Seçilen imza algoritması türüne bağlı olarak bir özel anahtar ve ona karşılık gelen bir açık anahtar üretilmektedir. Açık anahtar sertifika yardımı ile alıcılara güvenli bir şekilde ulaştırılmaktadır.
2. İmzalama: İki aşamadan oluşmaktadır
   a. Çevrim-dışı aşaması: İmzalanacak mesaj belirlenmeden önceki aşamadır. TZİ için gereken rastlantısal sayıların üretimini, bu sayılara tek-yön işlevinin uygulanmasını ve tek-yön işlev çıktılarının seçilen imza algoritması kullanılarak özel anahtar ile imzalanmasını içerir.
   b. Çevrim-içi safhası: İmzalanacak mesaja bağlı olarak hangi rastlantısal sayıların TZİ'yi oluşturacağını belirlemekten (eşleme algoritması ile) ibarettir.
3. İmza Onaylama: İmza onaylama işlemi önce tek-yön işlev çıktıları üzerindeki imzanın onaylanmasını daha sonra ise TZİ'nin onaylanmasını içermektedir. TZİ onaylanırken imza olarak gönderilen rastlantısal sayıların imzalanan mesaja ait olduğu ters eşleme algoritması ile onaylandıktan sonra tek-yön işlev çıktılarının imzalanmış tek-yön işlev çıktıları ile aynı olup olmadığı kontrol edilir.

4. TERS SSL YÖNTEMİ

Bu bölümde ikinci bölümde anlatılan orijinal SSL yöntemine alternatif olarak ÇÇİ yapılarının kullanıldığı Ters SSL yöntemi anlatılacaktır. Ters SSL yönteminin amacının sunucu performansını arttırmak olduğunu hatırlayınız.

Günümüzde SSL uygulamalarının büyük çoğunluğu RSA algoritmasını kullanmaktadır. RSA algoritmasını diğer algoritmalardan ayıran bir özellik hem imzalama hem şifreleme için kullanabiliyor olmasıdır. Fakat RSA'i tercih edilir kılan faktör daha iyi çalışılmış, anlaşılmış ve standartlaşmış olmasıdır denilmektedir. Bu bölümde anlatılacak olan Ters SSL yöntemi de RSA algoritmasını kullanmakta, ilave olarak kullanılan tek-yönlü işlevler de çoğu güvenlik protokollerinde gerekli olan mesaj özeti işlevleri (örnek: SHA-1, MD5) kullanılarak elde edilebilmektedir.

Tablo 1'de temel RSA işlemlerinin hız karşılaştırmaları basit şekliyle verilmiştir. Şifreleme ve imza doğrulama işlemelerinin yüksek hızda gerçekleştirilebilmeleri kullanılan açık anahtarın küçük bir değer olarak seçilebilmesinden dolayıdır. Bir diğer ifadeyle $y = m^e \bmod n$ RSA işleminde "e" açık değerinin "3" gibi küçük bir sayı olabilmesi buradaki modüler aritmetik işlem sonucunun çok hızlı bir şekilde hesaplanabilmesini sağlamaktadır.

Tablo 1 RSA işlem hızları

RSA Şifre Çözme	Yavaş
RSA Şifreleme	Hızlı
RSA İmza Oluşturma	Yavaş
RSA İmza Doğrulama	Hızlı

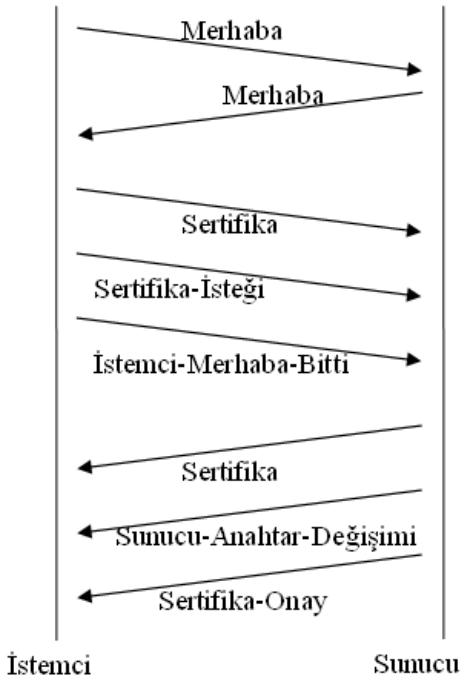
Bu gözlem ışığında Şekil 1'e tekrar baktığımızda sunucunun bir tane RSA şifre çözme bir tane de RSA imza doğrulama işlemi yaptığını görmekteyiz. Dolayısıyla SSL performansında hatırı sayılır bir iyileştirme yapmak için doğal hedef RSA şifre çözme işlemi olmalıdır. RSA şifre çözme işlemini sunucu yerine istemci gerçekleştirse ve sunucu çok daha hızlı RSA şifreleme işlemini yapmış olsa performans da arzuladığımız artış olacak fakat yeni bir problemi çözme zorunluluğu ortaya çıkacaktır. Bu problem sunucunun istemciye tanıtılma problemidir. Orjinal SSL'de sunucu kendi açık anahtarı ile şifrelenmiş bir mesajı doğru çözebilmesi ile istemciye tanıtılıyordu. Bu işlemi yapmayınca sunucunun tanınabilmesi için akla gelen ilk çözüm sayısal imza yönteminden yararlanmaktır. Yani sunucunun oluşturduğu imzanın doğrulanması ile tanımanın gerçekleştirilmesidir. Tahmin edilebileceği gibi imza oluşturma yavaş bir işlem olduğu için bu haliyle performans yönünden sunucunun iş yükü aşağı yukarı aynı kalmış olacaktır.

Orijinal SSL'deki bu değişikliği yani istemci ve sunucunun yapmış oldukları açık anahtar kripto işlemlerinin değiştirilmesini anlamlı hale getiren ikinci bir yeniliktir. Bu yenilik standart imza algoritmaları yerine ÇÇİ yapılarını kullanmak olarak özetlenebilir.

ÇÇİ kullanılarak sunucunun gerçek-zamanlı imza oluşturma performansında çok büyük oranlarda artış sağlanabileceği açıktır. Çünkü bölüm 3'de anlatıldığı gibi imza oluşturma işleminin çevrim-içi aşaması ÇÇİ kullanılınca sadece basit bir eşleştirme algoritmasının çalıştırılmasından ibarettir. ÇÇİ kullanımındaki temel hipotezimiz SSL sunucularının günün her saatinde aynı yoğunlukta olmadığı ve mesela sabahın erken saatlerinde sunucu yoğunluğu az iken ÇÇİ'lerin çevrim-dışı aşaması tamamlanarak bu hesaplama işlemlerinin SSL'in normal şartlarda kapasitesi üzerinde hizmet vermesi gereken saatlerdeki iş yükünü azaltmak için

kullanılmasıdır. Böylelikle SSL sunucusunun işlemci gücü çok daha verimli kullanılmış olacaktır.

Bu tartışmaların sonucunda Şekil 2'de gösterilen Ters SSL elsıkışmasının nasıl çalıştığını şu şekilde özetleyebiliriz.



Şekil 2 Ters SSL Elsıkışması

Ters SSL iki aşamadan oluşmaktadır:

1. Çevrim-dışı Aşama: Sunucunun ÇÇİ çevrim-dışı imzalama aşamasını yürüterek çevrim-içi kullanıma hazır ÇÇİ'leri hazırladığı aşamadır.
2. Çevrim-içi Aşama: Sunucu ve istemci arasındaki modifiye edilmiş el sıkışması gerçekleştirilir. Sunucu ve istemci merhabalaşırken istemci Ters SSL protokol uzantısını desteklediğini belirtir, sunucu da destekler durumda ise iki taraf Ters SSL kullanımına geçerler. Aksi halde taraflardan birinin Ters SSL desteği yoksa SSL protokolü eskisi gibi çalışır. Böylelikle geri-uyum (backward-compatibility) sağlanmış olur. Sonrasında istemci ve sunucunun rolleri değişir ve istemci sunucuya kendi sertifikasını gönderir ve sunucudan sertifikasını ister. Sunucu istemcinin sertifikasını onaylar ve üç mesaj gönderir: Kendi sertifikası, istemci açık anahtarı ile şifrelenmiş esas anahtar ve tüm haberleşmenin sayısal imzası. Sunucu imzayı üretirken haberleşmenin mesaj özetini üretir ve daha önce çevrim-dışı aşaması tamamlanmış ÇÇİ'nin çevrim içi aşamasını gerçekleştirir. İstemcinin ÇÇİ imza onaylama algoritması ile imzayı doğrulaması ve esas anahtarın şifresini çözmesi ile "Bitti" mesajları değiş-tokuş edilir ve bu şekilde Ters SSL tamamlanmış olur.

5. PERFORMANS DEĞERLENDİRME

Ters SSL'in performans iyileştirmesi hakkında bir ön değerlendirme yapabilmek için "openssl speed" komutunu [5] dört 1 GHz Sun UltraSPARC işlemci ve 8 GB bellek içeren bir sunucuda çalıştırarak ölçüm sonuçlarını inceledik. Mesaj özet işlevi 5 micro-saniyeden az bir sürede gerçekleştirildiği için bu işlemi Tablo 3'de yer alan karşılaştırmalarda ihmal edebildik.

Tablo 2 RSA özel ve açık anahtar işlem hızları

	Özel Anahtar	Açık Anahtar	Oran
RSA 512-bit	1.7 ms	0.2 ms	8.5
RSA 1024-bit	8.1 ms	0.5 ms	16.2
RSA 2048-bit	50.0 ms	1.5 ms	33.3
RSA 4096-bit	339.7 ms	5.3 ms	64.1

Tablo 3 Orjinal SSL ve Ters SSL'de çevrim-içi hesaplama gereksinimleri

	Orjinal SSL	Ters SSL	Hız Artışı
RSA 512-bit	1.9 ms	0.2 ms	9.5
RSA 1024-bit	8.6 ms	0.5 ms	17.2
RSA 2048-bit	51.5 ms	1.5 ms	34.3
RSA 4096-bit	345.0 ms	5.3 ms	65.1

Tablo 2 ve Tablo 3'den Ters SSL kullanıldığında 65 kata kadar çıkabilen oranlarda performans iyileştirmesinin mümkün olduğunu görmekteyiz. Fakat unutulmamalıdır ki, bu iyileştirme teorik limittir, Ters SSL kullanımında ÇÇİ yapılarının imza uzunluğunun büyüyeceği, ÇÇİ'nin çevrim-dışı aşamasındaki sonuçlara çevrim-içi aşamasında bellekten veya sabit diskten erişimin gerektiği gibi pek çok pratik sebepten dolayı elde edilecek hız artışının bir miktar düşeceği öngörülmektedir. Bu düşüsün ne seviyede olacağı kesin olarak ancak gerçek uygulamalar üzerinde ölçümler alarak mümkün olacaktır. Bu şekilde bir gerçek uygulama çalışması tarafımızca planlanmaktadır.

6. İLGİLİ ÇALIŞMALAR

SSL protokolu Netscape firması tarafından 1994 yılında geliştirilmiştir. Şu an yaygın olarak kullanılan üçüncü versiyon 1996 yılında yayınlanmış, 1999 yılında TLS 1.0 ismi ile RFC 2246 ile İnternet standardı haline getirilmiştir.

SSL sunucularının performans problemine akla ilk gelen iki çözüm aşağıdaki gibidir:

1. Özel tasarlanmış SSL hızlandırıcılar: Bu çözüm şu an endüstride tercih edilen etkin fakat pahalı bir çözümdür [6].

2. Önbellekleme (Caching): Sunucu ve istemci arasında daha evvelki oturumda üretilen esas anahtarı tekrar kullanma tekniğine dayanır [7]. Diğer performans teknikleriyle beraber kullanılabilir, getirisi istemcilerin hangi yoğunlukta sunucuya bağlandığına bağlı bir çözümdür. SSL elsıkışması işlemini hızlandırıcı bir özelliği yoktur.

Bu iki iyi bilinen yöntem dışında ilk defa 2001 yılında Shacham ve Boneh SSL elsıkışması performansını sunucu tarafında iyileştirme problemi ile ilgilenmişlerdir [8]. Probleme ilk önerilen çözüm yığınlama (batching) diye isimlendirilen birden fazla istemci ile ilgili açık anahtar kripto işlemini (RSA algoritması şifre çözme işlemi) beraber yapma tekniğine dayanmaktadır. Bu çözümün olumlu yönleri olmakla birlikte en büyük dezavantajı maksimum 2.5 kat oranında bir performans kazancı getirmesidir. 2005 yılında Qi ve arkadaşları bu tekniğin birden fazla sertifika kullanması problemine el atmışlar fakat performans konusunda fazla bir ilerleme kaydetmemişlerdir [9].

SSL elsıkışmasının hızını arttırmak için diğer bir yöntem RSA kripto algoritması yerine Eliptik Eğri Kriptografisini (EEK) kullanmaktır [10]. EEK kullanılarak sadece 2.4 katlık bir hız elde edilmesinin yanısıra EEK'nin patent sınırlamalarını beraberinde getirmesi bilhassa uygulama boyutu düşünüldüğünde büyük bir handikap olabilmektedir.

Castellucia ve arkadaşları şu ana kadarki en büyük hız artış oranını yakaladıklarını söylemektedirler [11]. Kullandıkları teknik kısaca sunucunun kripto (RSA şifre çözme) hesaplama yükünü büyük oranda istemci tarafından destek alarak yürütmesi olarak açıklanabilir. Bu çalışmada iddia edilen performans iyileştirmesi 11 ila 19 kat gibi çarpıcı oranlara çıkmaktadır. Fakat çalışmanın eksikliği bu iddiayı gerçek bir uygulama üzerinde test etmemiş olmalarıdır. Sadece kripto algoritma hesaplama kısmını ele almışlar ve performans karşılaştırmasını bu şekilde yapmışlar, mesela yeni tekniğin beraberinde getirdiği ekstra mesajlaşmanın etkisini ihmal etmişlerdir.

7. SONUÇ

Bu çalışmada SSL sunucularının performans problemini pahalı donanım gerektirmeden protokol bazında değişiklikler yardımı ile çözmeyi amaçlayan Ters SSL yöntemi tanıtılmıştır.

Ters SSL yönteminde sunucu istemciye açık anahtar şifre çözme yerine sayısal imza yardımı ile tanıtılmakta ve çevrim içi / çevrim dışı imza yapılarındaki çevrim-dışı aşamanın sunucu yoğunluğunun değişken olmasından hareketle yoğunluğun az olduğu saatlerde gerçekleştirilebileceği hipotezinden hareket etmektedir.

Konu ile ilgili daha ayrıntılı bilgi edinmek isteyenlere ve bilhassa yer darlığı sebebiyle burada bahsedilemeyen Ters SSL yönteminin servis engelleyici saldırılara (denial of service attacks) karşın çözüm olarak nasıl kullanılabileceğini merak edenlere daha önce yazmış olduğumuz teknik rapor önerilmektedir [12].

KAYNAKLAR

[1] http://www.crn.com/security/18822452

[2] [Shimon Even, Oded Goldreich, Silvio Micali, On-Line/Off-Line Digital Signature Schemes. Advances in Cryptology - CRYPTO '89 Proceedings. Lecture Notes in Computer Science 435 Springer 1990.

[3] L.Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, October 1979.

[4] Kemal Bıçakcı, Brian Tung, Gene Tsudik. On constructing optimal one-time signatures. Proceedings of the Fifteenth International Symposium on Computer and Information Sciences, ISCIS 2000, Ekim 2000, Istanbul.

[5] OpenSSL kütüphanesi, http://www.openssl.org.

[6]www.ncipher.com/cryptographic_hardware/ssl_acceleration/22/nfast.

[7] A. Goldberg, R. Buff, A. Schmitt, "Secure Web Server Performance Dramatically Improved By Caching SSL Session Keys," in Workshop on Internet Server Performance, June 1998.

[8] Hovav Shacham, Dan Boneh: Improving SSL Handshake Performance via Batching. CT-RSA 2001: 28-43.

[9] Fang Qi, Weijia Jia, Feng Bao, Yongdong Wu: Batching SSL/TLS Handshake Improved. ICICS 2005: 402-413.

[10] Vipul Gupta, Douglas Stebila, Stephen Fung, Sheueling Chang Shantz, Nils Gura, Hans Eberle, Speeding up Secure Web Transactions Using Elliptic Curve Cryptography. In proceedings of Network and Distributed System Security Symposium NDSS 2004, Internet Society 2004.

[11] Claude Castelluccia, Einar Mykletun, Gene Tsudik: Improving secure server performance by re-balancing SSL/TLS handshakes. ASIACCS 2006: 26-34.

[12] Kemal Bicakci, Bruno Crispo, Andrew S. Tanenbaum, “Reverse SSL: Improved Server Performance and DoS Resistance for SSL Handshakes”, Cryptology ePrint Archive, Report 2006/212.