BSM 471-AĞ GÜVENLİĞİ

SSL/TLS

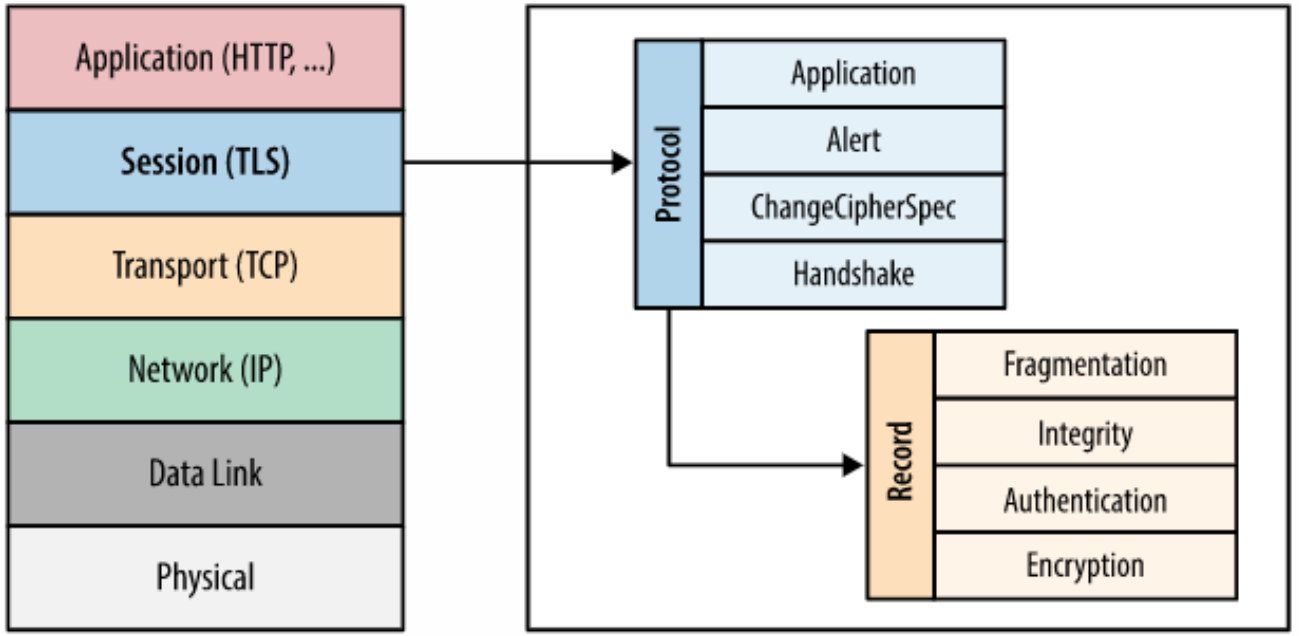
Dr. Öğr. Üyesi Musa BALTA
Bilgisayar Mühendisliği Bölümü
Bilgisayar ve Bilişim Bilimleri Fakültesi

SSL (Secure Socket Layer) Protokolü

- Güvenli Yuva Katmanı (SSL), 1994 yılında ilk olarak **Netscape'te**, müşterilerin kişisel verilerini korumak için şifrelemenin yanı sıra güvenli bir işlem sağlamak için kimlik doğrulama ve bütünlük garantileri gerektiren Web'de e-ticaret işlem güvenliğini sağlamak için geliştirilmiştir.
- SSL doğru kullanıldığında, üçüncü taraf bir gözlemci yalnızca bağlantı uç noktalarını, şifreleme türünü ve ayrıca gönderilen verilerin sıklığını ve yaklaşık miktarını anlayabilir, ancak ***gerçek verileri okuyamaz veya değiştiremez***.
- IETF, SSL'yi TLS olarak yeniden adlandırmış ve Ocak 1999'da ilk spesifikasyon olan sürüm 1.0'ı yayınlamıştır.
- TLS 1.0, SSL'nin en son sürümü olan sürüm 3.0'ın muadilidir.
- TLS 1.1 Nisan 2006'da, ***TLS 1.2 Ağustos 2008'de ve TLS 1.3 Ağustos 2018'de*** piyasaya sürüldü.
- TLS 1.3, TLS protokolünün büyük bir revizyonudur ve önceki sürümlere göre önemli güvenlik ve performans iyileştirmeleri sağlar.

TLS (Transport Layer Security) Protokolü

- TLS, İnternet iletişimleri için kullanılan standart TCP/IP soket protokolüne güvenli bir geliştirme sağlar.
- TLS ile en sık kullanılan uygulamalar;
  - HTTP
  - NNTP,
  - Telnet,
  - LDAP,
  - IMAP
  - FTP



TLS Çalışma Yapısı

- TLS, TCP gibi güvenilir bir aktarım protokolünün üzerinde çalışacak şekilde tasarlanmıştır. Bununla birlikte, UDP gibi datagram protokolleri üzerinde çalışacak şekilde de uyarlanmıştır.
- RFC 6347'de tanımlanan Datagram Aktarım Katmanı Güvenliği (**DTLS**) protokolü, TLS protokolünü temel alır ve datagram teslim modelini korurken benzer güvenlik garantileri sağlayabilir.
- TLS'nin etkili olmasının nedenlerinden biri, birkaç farklı şifreleme işlemi kullanmasıdır.
- TLS, kimlik doğrulama sağlamak için ***açık anahtarlı şifrelemeyi*** ve gizlilik ve veri bütünlüğü sağlamak için karma işlevlere sahip ***gizli anahtarlı şifrelemeyi*** kullanır.

Kriptografik İşlemler

- Kriptografinin birincil amacı, yetkisiz bir üçüncü tarafın iki taraf arasındaki özel iletişime erişmesini ve bunları anlamasını zorlaştırmaktır.
- Şifreleme, orijinal mesajı (açık metin) şifreli bir mesaja (şifreli metin) dönüştürmek için karmaşık algoritmalar kullanır. Bir ağ üzerinden aktarılan verileri şifrelemek ve şifresini çözmek için kullanılan algoritmalar genellikle iki kategoriye ayrılır: ***gizli anahtarlı kriptografi*** ve ***açık anahtarlı kriptografi***.
- Hem gizli anahtarlı kriptografi hem de açık anahtarlı kriptografi, üzerinde anlaşmaya varılan bir kriptografik anahtarın veya anahtar çiftinin kullanımına bağlıdır. Anahtar, verilerin şifrelenmesi ve şifresinin çözülmesi işlemi sırasında kriptografik algoritma veya algoritmalar tarafından kullanılan bir bit dizisidir. Bir kriptografik anahtar, bir kilidin anahtarı gibidir; sadece doğru anahtarla kilidi açabilirsiniz.
- İletişim kuran iki taraf arasında bir anahtarı güvenli bir şekilde iletmek önemlidir. Açık anahtar sertifikası, alıcıya açık anahtarın orijinalliği konusunda güvence sağlarken, bir tarafın kendi açık anahtarını güvenli bir şekilde iletmesini sağlar.

Gizli Anahtarlı Kriptografi

- Gizli anahtarlı şifrelemede, iletişim kuran her iki taraf, Alice ve Bob, mesajları şifrelemek ve şifrelerini çözmek için **aynı anahtarı** kullanır. Herhangi bir şifrelenmiş verinin ağ üzerinden gönderilebilmesi için hem Alice hem de Bob'un anahtara sahip olması ve şifreleme ve şifre çözme için kullanacakları ***kriptografik algoritma*** üzerinde anlaşmaları gerekir.
- Gizli anahtarlı kriptografiyle ilgili **en büyük sorunlardan** biri, bir saldırganın erişimine izin vermeden anahtarın bir taraftan diğerine nasıl aktarılacağına ilişkin lojistik sorundur. Alice ve Bob, verilerini gizli anahtar şifrelemesiyle güvence altına alıyorsa ve Charlie anahtarlarına erişim kazanırsa, Charlie, Alice ile Bob arasında yakaladığı tüm gizli mesajları anlayabilir. Charlie, Alice'in ve Bob'un mesajlarının şifresini çözmekle kalmaz, aynı zamanda Alice'miş gibi davranabilir ve Bob'a şifrelenmiş veriler gönderebilir. Bob, mesajın Alice'ten değil Charlie'den geldiğini bilmeyecektir.

Gizli Anahtarlı Kriptografi-devam

- Gizli anahtar dağıtımı sorunu çözüldükten sonra, gizli anahtarlı kriptografi değerli bir araç olabilir. Algoritmalar mükemmel güvenlik sağlar ve verileri nispeten hızlı bir şekilde şifreler. Bir **TLS oturumunda** gönderilen hassas verilerin çoğu, gizli anahtarlı şifreleme kullanılarak gönderilir.
- Gizli anahtarlı kriptografiye **simetrik kriptografi** de denir, çünkü aynı anahtar verileri hem şifrelemek hem de şifresini çözmek için kullanılır. İyi bilinen gizli anahtar şifreleme algoritmaları arasında Gelişmiş Şifreleme Standardı (**AES**), Üçlü Veri Şifreleme Standardı (**3DES**) ve Rivest Cipher 4 (**RC4**) bulunur.

Açık Anahtarlı Kriptografi

- Açık anahtarlı kriptografi, hem **genel anahtarı** hem de **özel anahtarı** kullanarak anahtar dağıtımının lojistik sorununu çözer. Genel anahtar, ağ üzerinden açık bir şekilde gönderilebilirken, özel anahtar, iletişim kuran taraflardan biri tarafından özel olarak tutulur. ***Genel ve özel anahtarlar birbirinin kriptografik tersidir; bir anahtarın şifrelediğini, diğer anahtar şifresini çözecektir.***

- Bob'un açık anahtar şifreleme kullanarak Alice'e gizli bir mesaj göndermek istediğini varsayalım. Alice'in hem genel anahtarı hem de özel anahtarı vardır, bu nedenle özel anahtarını güvenli bir yerde tutar ve genel anahtarını Bob'a gönderir. Bob, Alice'in genel anahtarını kullanarak Alice'e gönderilen gizli mesajı şifreler. Alice daha sonra özel anahtarıyla mesajın şifresini çözebilir. Alice, özel anahtarını kullanarak bir mesajı şifreler ve şifreli mesajı Bob'a gönderirse, Bob aldığı verilerin Alice'ten geldiğinden emin olabilir; Bob, Alice'in genel anahtarıyla verilerin şifresini çözebilirse, mesajın Alice tarafından kendi özel anahtarıyla şifrelenmiş olması gerekir ve Alice'in özel anahtarına yalnızca Alice sahiptir. Sorun şu ki, Alice'in genel anahtarı herkese açık olduğu için başka biri de mesajı okuyabiliyor.

Açık Anahtarlı Kriptografi-devam

- Bu senaryo, güvenli veri iletişimine izin vermese de **dijital imzalar** için temel sağlar. Dijital imza, ortak anahtar sertifikasının bileşenlerinden biridir ve TLS'de bir istemcinin veya sunucunun kimliğini doğrulamak için kullanılır.
- Açık anahtarlı kriptografiye **asimetrik kriptografi** de denir çünkü verileri şifrelemek ve şifresini çözmek için farklı anahtarlar kullanılır. TLS ile sıklıkla kullanılan iyi bilinen bir açık anahtar şifreleme algoritması, Rivest Shamir Adleman (**RSA**) algoritmasıdır. Özel olarak gizli anahtar değişimi için tasarlanmış TLS ile kullanılan bir diğer açık anahtar algoritması, Diffie-Hellman (**DH**) algoritmasıdır.
- Açık anahtarlı kriptografi, kapsamlı hesaplamalar gerektirir ve bu da onu çok yavaş hale getirir. Bu nedenle, genellikle şifrelenmiş veri iletişimlerinin büyük bir kısmı yerine, yalnızca gizli anahtarlar gibi *küçük veri parçalarını şifrelemek* için kullanılır.

Gizli Anahtar vs Açık Anahtar

- Hem gizli anahtarlı kriptografinin hem de açık anahtarlı kriptografinin güçlü ve zayıf yönleri vardır:
- Gizli anahtarlı kriptografi ile veriler hızlı bir şekilde şifrelenebilir ve şifresi çözülebilir, ancak iletişim kuran her iki tarafın da aynı gizli anahtar bilgisini paylaşması gerektiğinden, anahtar alışverişinin lojistiği sorun olabilir.
- Açık anahtarlı kriptografide, anahtar değişimi bir sorun değildir çünkü açık anahtarın gizli tutulması gerekmez, ancak verileri şifrelemek ve şifresini çözmek için kullanılan algoritmalar kapsamlı hesaplamalar gerektirir ve bu nedenle çok yavaştır.

Açık Anahtar Sertifikaları

- Bir açık anahtar sertifikası, bir varlığın asimetrik kriptografide kullanılmak üzere kendi genel anahtarını iletmesi için güvenli bir yol sağlar. Açık anahtar sertifikası aşağıdaki durumu önler:
  - Charlie kendi genel anahtarını ve özel anahtarını oluştururursa, kendisinin Alice olduğunu iddia edebilir ve genel anahtarını Bob'a gönderebilir. Bob, Charlie ile iletişim kurabilecektir, ancak Bob, verilerini Alice'e gönderdiğini düşünecektir.
- Açık anahtar sertifikası, **pasaportun dijital eşdeğeri** olarak düşünülebilir. Güvenilir bir kuruluş tarafından verilir ve hamiline kimlik sağlar.
- Ortak anahtar sertifikaları veren güvenilir bir kuruluş, **Sertifika Yetkilisi (CA)** olarak bilinir. CA notere benzetilebilir.
- Bir CA'dan sertifika almak için, kişinin kimlik kanıtı sağlaması gerekir. CA, başvuranın temsil ettiğini söylediği kuruluşu temsil ettiğinden emin olduktan sonra, CA, sertifikada yer alan bilgilerin geçerliliğini onaylayan sertifikayı imzalar.

Açık Anahtar Sertifikaları-devam

- *Bir açık anahtar sertifikası aşağıdaki alanları içerir:*
  - **Issuer**; Sertifikayı veren CA. Kullanıcı, sertifikayı veren CA'ya güveniyorsa ve sertifika geçerliyse, kullanıcı sertifikaya güvenebilir.
  - **Period of validity**; Sertifikanın bir son kullanma tarihi vardır. Bir sertifikanın geçerliliği doğrulanırken bu tarih kontrol edilmelidir.
  - **Subject**; Sertifikanın temsil ettiği varlık hakkında bilgi içerir.
  - **Subject's public key**; Sertifikanın sağladığı birincil bilgi, öznenin genel anahtarıdır. Diğer tüm alanlar, bu anahtarın geçerliliğini sağlamak için sağlanmıştır.
  - **Signature**; Sertifika, sertifikayı veren CA tarafından dijital olarak imzalanır. İmza, CA'nın özel anahtarı kullanılarak oluşturulur ve sertifikanın geçerliliğini sağlar. TLS işleminde gönderilen veriler değil sadece sertifika imzalandığı için, TLS inkar edilemezlik sağlamaz.

Açık Anahtar Sertifikaları-devam

- Bir sertifika zincirinde birden çok sertifika birbirine bağlanabilir. Bir sertifika zinciri kullanıldığında, ilk sertifika her zaman gönderene ait olur. Sonraki, gönderenin sertifikasını veren kuruluşun sertifikasıdır.
- Zincirde daha fazla sertifika varsa, her biri bir önceki sertifikayı veren yetkiliye aittir. Zincirdeki son sertifika, bir kök CA'nın sertifikasıdır. Bir kök CA, yaygın olarak güvenilen bir genel Sertifika Yetkilisidir.
- Birkaç kök CA'ya ilişkin bilgiler genellikle müşterinin İnternet tarayıcısında depolanır. Bu bilgiler, CA'nın genel anahtarını içerir. İyi bilinen CA'lar arasında ***DigiCert, Entrust*** ve ***GlobalSign*** bulunur.

Şifreleme Özet Fonksiyonları (Hash Function)

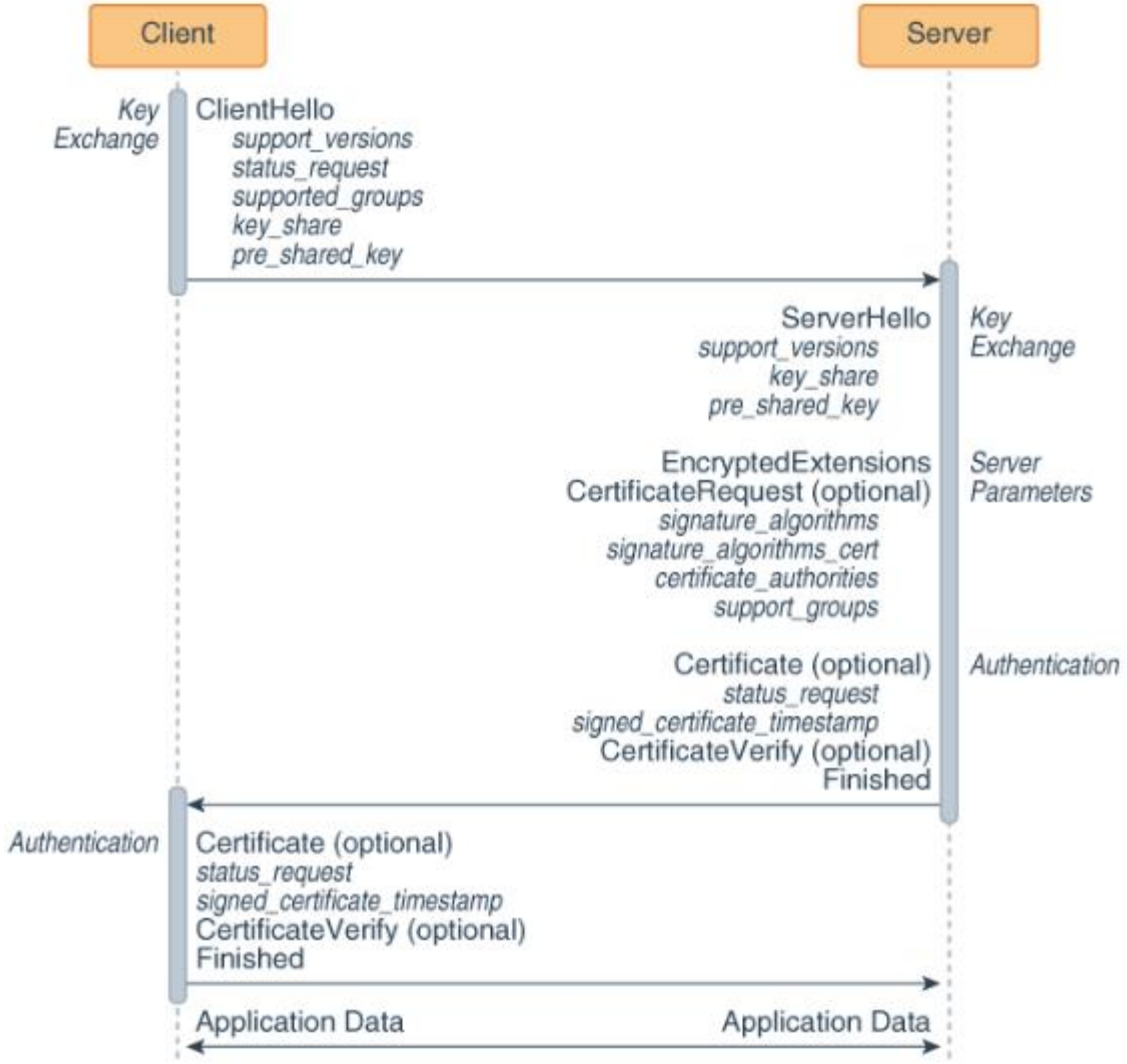
- TLS, şifrelenmiş verileri gönderirken, veri bütünlüğünü sağlamak için tipik olarak bir **kriptografik özet fonksiyonu işlevi** kullanır.
- Kriptografik hash işlevi, sağlama toplamına benzer. Temel fark, bir sağlama toplamı verilerdeki kazara yapılan değişiklikleri tespit etmek için tasarlanırken, bir kriptografik özet işlevinin kasıtlı değişiklikleri tespit etmek için tasarlanmış olmasıdır.
- Veriler bir kriptografik hash işlevi tarafından işlendiğinde, hash olarak bilinen küçük bir bit dizisi oluşturulur. Mesajdaki en ufak bir değişiklik, tipik olarak, elde edilen karmada büyük bir değişiklik yapar.
- Bir kriptografik hash işlevi, bir kriptografik anahtar gerektirmez. TLS ile sıklıkla kullanılan bir hash işlevi, Güvenli Karma Algoritmadır (**SHA**). SHA, ABD Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) tarafından önerilmiştir.

Mesaj Doğrulama Kodu

- Mesaj kimlik doğrulama kodu (MAC), gizli bir anahtara dayalı olması dışında kriptografik karmaya benzer. Bir kriptografik sağlama işlevi tarafından işlenen verilere gizli anahtar bilgisi eklendiğinde, elde edilen sağlama **HMAC** olarak bilinir.
- Bir mesaj için bir kriptografik hash oluşturulduğunda, hash gönderenin özel anahtarıyla şifrelenir. Bu şifrelenmiş karma ***dijital imza olarak*** adlandırılır.

TLS El Sıkışma (Handshake)

- İstemci ve sunucu, uygulama verilerini TLS üzerinden alıp vermeye başlamadan önce, şifreli tünel üzerinde anlaşmaya varılmalıdır:
  - istemci ve sunucu, TLS protokolünün sürümü üzerinde anlaşmalı,
  - şifre paketini seçmeli,
  - gerekirse sertifikaları doğrulamalıdır.



TLS El Sıkışma Adımlar

1. Key Exchange

- İstemci, sunucuya bir **ClientHello** mesajı gönderir.
- Sunucu, ClientHello mesajını işler ve bağlantı için uygun şifreleme parametrelerini belirler. Daha sonra anlaşmalı bağlantı parametrelerini gösteren kendi **ServerHello** mesajıyla yanıt verir. TLS 1.3 için, ServerHello mesajı yalnızca anahtar ve şifre seçeneklerini belirler. Diğer el sıkışma parametreleri daha sonra belirlenebilir.

2. Server Parameters: *Sunucu, sunucu parametrelerini oluşturmak için iki mesaj gönderir.*

- *EncryptedExtensions:* Bu mesaj, bağımsız sertifikalara özgü olanlar dışında, şifreleme parametrelerini belirlemek için gerekli olmayan ClientHello uzantılarına verilen yanıtları içerir.
- *Certificate Request* (isteğe bağlı): Sertifika tabanlı istemci kimlik doğrulaması isteniyorsa, sunucu o sertifika için istenen parametreleri içeren bu mesajı gönderir. İstemci kimlik doğrulaması istenmiyorsa bu mesaj atlanır.

TLS El Sıkışma Adımlar-devam

3. Authentication

- Sunucu 3 doğrulamala mesajı gönderir.
  - *Certificate* (isteğe bağlı): Bu mesaj, kimlik doğrulama sertifikasını ve sertifika zincirindeki diğer destekleyici sertifikaları içerir. Sunucu bir sertifikayla kimlik doğrulaması yapmıyorsa bu mesaj atlanır.
  - *CertificateVerify* (isteğe bağlı): Bu mesaj, Sertifika mesajındaki ortak anahtara karşılık gelen özel anahtarı kullanan tüm anlaşma üzerinde bir imza içerir. Sunucu bir sertifikayla kimlik doğrulaması yapmıyorsa bu mesaj atlanır.
  - *Finished*: Tüm anlaşma üzerinde bir MAC (Mesaj Kimlik Doğrulama Kodu).
- İstemci kendi Sertifika, Sertifika Doğrulama ve Tamamlandı mesajlarıyla yanıt verir. Sunucu bir Sertifika Talebi mesajı göndermediyse, Sertifika mesajı atlanır. İstemci bir sertifikayla kimlik doğrulaması yapmıyorsa, CertificateVerify mesajı atlanır.

Key Exchange-ClientHello

- Anahtar değişim mesajları, ClientHello ve ServerHello, istemcinin ve sunucunun güvenlik yeteneklerini belirler ve anlaşmanın geri kalanını ve uygulama verilerini korumak için kullanılan trafik anahtarları dahil olmak üzere paylaşılan sırlar oluşturur.
- **ClientHello;**
- TLS anlaşması, istemcinin sunucuya bir ClientHello mesajı göndermesiyle başlar. Bu mesaj aşağıdaki alanları içerir:
  - cipher_suites: Bu alan, istemci tarafından desteklenen simetrik şifreleme seçeneklerinin bir listesini, özellikle kayıt koruma algoritmasını (gizli anahtar uzunluğu dahil) ve Anahtarlı Karma Mesaj Kodu (HMAC) tabanlı Çıkar ve Genişlet ile kullanılacak bir karma içerir. Anahtar Türetme İşlevi (HKDF).
  - *Extensions:* Uzantılar, mevcut istemciler üzerinde minimum etkiyle TLS protokolüne yeni özelliklerin eklenmesini kolaylaştırır.

ClientHello-devam

- Extensions bu alanlardan oluşur:
  - *supported versions*: Bu uzantı, istemcinin hangi TLS sürümlerini desteklediğini gösterir. ClientHello mesajı bu mesajı içermelidir.
  - *status request*: Bu uzantı, istemcinin bir sertifika durum protokolü kullanmak istediğini belirtir; sunucu kullanmayı kabul etmeyebilir.
  - *supported groups*: Bu uzantı, istemcinin anahtar değişimi için desteklediği adlandırılmış grupları gösterir. Bu adlandırılmış gruplar, eliptik eğri gruplarını (ECDHE) ve sonlu alan gruplarını (DHE) içerir. ClientHello mesajı, ECDHE veya DHE anahtar değişimi kullanıyorsa bu mesajı içermelidir.
  - *key_share*: Bu uzantı, anahtar değişimi için şifreleme parametrelerinin bir listesini içerir. Bu listeyi içeren client_shares adlı bir alan içerir. Bu listedeki her öğe aşağıdaki alanları içerir
    - Group; Anahtar değişimi şifreleme yönteminin dayandığı grubun adı.
    - key_Exchange; Grup alanının değerine göre belirlenen anahtar değişim bilgileri.

ClientHello-devam

- *pre_shared_key;* Önceden paylaşılan bir anahtar (PSK), daha önce iki taraf arasında kullanılması gerekmeden önce bazı güvenli kanallar kullanılarak paylaşılan paylaşılan bir sırdır. PSK'lar önceki bir bağlantıda kurulabilir ve daha sonra yeni bir bağlantı kurmak için kullanılabilir. Bir el sıkışma tamamlandığında, sunucu istemciye ilk el sıkışmadan türetilen benzersiz bir anahtara karşılık gelen bir PSK kimliği gönderebilir.
- *Cookie;* Bir sunucu bir HelloRetryRequest mesajı gönderdiğinde, istemciye bu uzantıyı dahil edebilir. (Sunucu, kabul edilebilir bir parametre kümesi bulabilirse, bir ClientHello mesajına yanıt olarak bir HelloRetryRequest mesajı gönderir, ancak ClientHello mesajı, anlaşmaya devam etmek için yeterli bilgiye sahip değildir.) Bu uzantının bir amacı, sunucuyu etkinleştirmektir. istemciyi görünen ağ adresinde erişilebilirlik göstermeye zorlamak için (bu, bazı hizmet reddi saldırısı (DoS) koruması sağlar. İstemci yeni bir ClientHello mesajı gönderdiğinde, HelloRetryRequest'te alınan içerikleri bir tanımlama bilgisi uzantısına kopyalamalıdır.
- *server_name;* TLS 1.3, bir istemcinin bir sunucuya iletişim kurduğu sunucunun adını söylemesi için bir mekanizma sağlamaz. İstemciler, tek bir ağ adresinde birden çok sanal sunucu barındıran sunuculara bağlantıları kolaylaştırmak için bu bilgileri sağlamak üzere bu uzantıyı kullanabilir. Bazı sunucuların istemcilerin bu uzantıyı göndermesini gerektirebilir.

Key Exchange-ServerHello

- Sunucu, kabul edilebilir bir anlaşma parametreleri kümesi üzerinde anlaşabiliyorsa, istemcinin ClientHello mesajına bir ServerHello mesajıyla yanıt verir. Bu mesaj aşağıdaki alanları içerir:
  - ***cipher_suites***: Bu alan, sunucu tarafından ClientHello.cipher_suites alanındaki listeden seçilen tek şifre paketini içerir.
  - ***Extensions:*** Bu alan, şifreleme bağlamını oluşturmak ve protokol sürümünü müzakere etmek için gereken uzantıları içerir. SeverHello'nun içerebileceği uzantılar şunları içerir:
    - *supported_versions:* Hangi TLS sürümünü kullandığını gösterir. ServerHello mesajı bu uzantıyı içermelidir.
    - *key_share:* Bu uzantı, anahtar değişimi için şifreleme parametrelerinin bir listesini içerir.
    - *pre_shared_key:* Bu uzantı, sunucunun kullanmayı kabul ettiği önceden paylaşılan anahtarı içerir.
- Sunucu, EncryptedExtensions mesajında diğer uzantıları ayrı olarak gönderir.

Server Parameters

- Sunucu, istemciye bir ServerHello mesajı gönderdikten sonra, sunucu parametrelerini oluşturmak için iki mesaj gönderir: EncryptedExtensions ve CertificateRequest:
  - ***EncryptedExtensions***: Bu mesaj, bağımsız sertifikalara özgü olanlar dışında şifreleme parametrelerini belirlemek için gerekli olmayan ClientHello uzantılarına verilen yanıtları içerir.
  - ***CertificateRequest***: Sertifika tabanlı istemci kimlik doğrulaması isteniyorsa, bu mesaj gönderilir. İstemciden istenen bir sertifika için parametreler içerir. Aşağıdaki alanları içerir:
    - certificate_request_context; Bu alan, sertifika talebini tanımlayan bir tanımlayıcı içerir.
    - Extensions; Bu alan, istenen sertifikanın parametrelerini açıklayan uzantıları içerir.

Server Parameters-devam

- Extensions; Bu alan, istenen sertifikanın parametrelerini açıklayan uzantıları içerir.
  - *signature_algorithms*; Bu uzantı, CertificateVerify mesajlarında hangi imza algoritmalarının kullanılabileceğini gösterir. ServerHello mesajı bu uzantıyı içermelidir.
  - *signature_algorithms_cert*; Bu uzantı, dijital imzalarda hangi imza algoritmalarının kullanılabileceğini gösterir. Eğer bu mesaj gönderilmezse,signature_algorithms uzantısında belirtilen değerleri kullanır.
  - *certificate_authorities*; Bu uzantı, sunucunun hangi sertifika yetkililerini kabul ettiğini gösterir.
  - *supported_groups*; Bu mesaj, sunucunun tercih ettiği adlandırılmış grupları içerir. İstemci, sonraki bağlantılarda key_share uzantısında hangi grupları kullandığını değiştirmek için bu bilgileri kullanabilir.

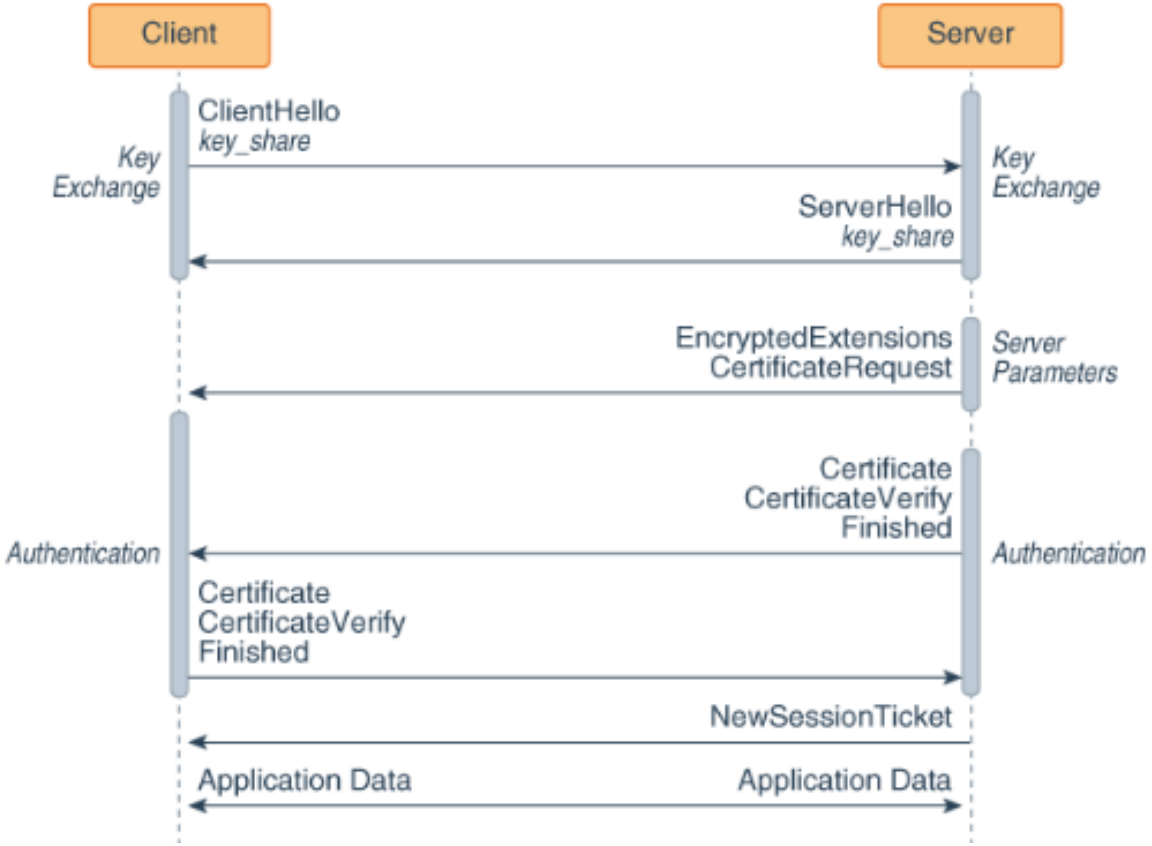
Authentication

- Sunucu ve istemcinin bir TLS anlaşmasında birbirlerine gönderdikleri son üç mesaj;
- **Certificate**; Bu mesaj, kimlik doğrulama sertifikasını ve sertifika zincirindeki diğer destekleyici sertifikaları içerir. Anahtar değişim yöntemi kimlik doğrulama için sertifika kullanıyorsa, sunucunun bu mesajı göndermesi gerekir. İstemci bunu ancak ve ancak sunucu bir Sertifika İsteği mesajı yoluyla istemci kimlik doğrulaması talep ettiyse göndermelidir. Sertifika mesajı aşağıdaki alanları içerir:
  - *Certificate_list*; her biri tek bir sertifika ve bir dizi uzantı içeren bir Sertifika Girişi yapısı dizisi içerir.
  - *Extensions*== Status report+signed_certificate_timestamp
- **CertificateVerify**; Bu mesaj, Sertifika mesajındaki ortak anahtara karşılık gelen özel anahtarı kullanan tüm anlaşma üzerinde bir imza içerir. İstemcinin veya sunucunun sertifikasına karşılık gelen özel anahtara sahip olduğunun kanıtını sağlar. Bu mesaj aşağıdaki alanları içerir:
  - *Algorithm*; kullanılan imza algoritmasını içerir.
  - *Signature*; algoritmayı kullanan dijital imzayı içerir.
- **Finished**; Bu mesaj, el sıkışmanın tamamında bir Mesaj Kimlik Doğrulama Kodu (MAC) içerir. İstemci ve sunucu, eşlerinden aldıkları Bitti mesajlarını doğruladıktan sonra, her iki taraf da bağlantı üzerinden uygulama verileri gönderip alabilir.

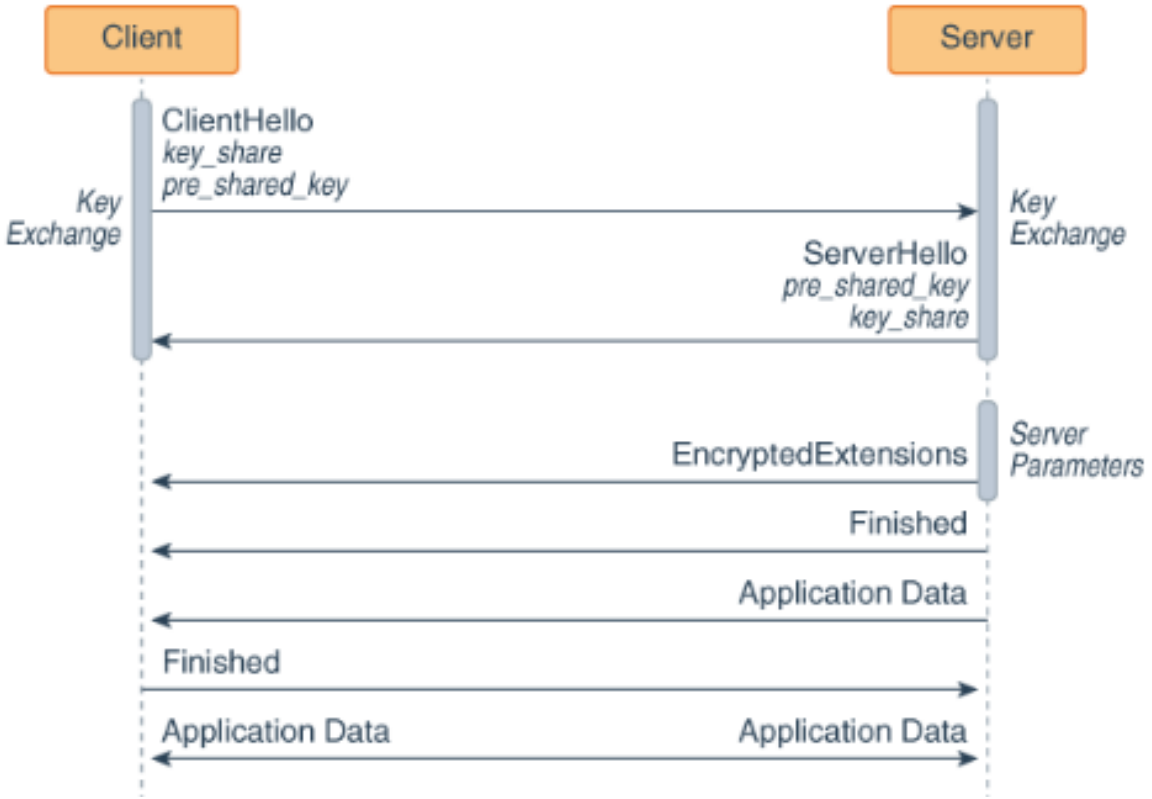
Önceden Paylaşılan Anahtarla Oturum Sürdürme (Session Resumption with a Pre-Shared Key)

- Önceden paylaşılan bir anahtar (PSK), daha önce iki taraf arasında kullanılması gerekmeden önce bazı güvenli kanallar kullanılarak paylaşılan paylaşılan bir sırdır.
- Bir TLS anlaşması sırasında bir PSK oluşturabilir ve ardından bunu başka bir anlaşmada yeni bir bağlantı kurmak için kullanabilirsiniz; buna PSK ile oturumun yeniden başlatılması denir. PSK, ilk el sıkışmadan elde edilen benzersiz bir anahtara karşılık gelir.
- Sunucu yeni bir bağlantı kurarken PSK'yı kabul ederse, bu bağlantının güvenlik içeriği orijinal bağlantıya kriptografik olarak bağlanır ve ilk anlaşmadan türetilen anahtar, tam TLS anlaşması yerine kriptografik durumu önyüklemek için kullanılır.

TLS 1.3 PSK Kurulumu



TLS 1.3 PSK Kullanımı



TLS 1.3 PSK Adımları

1. İstemci, sunucuya key_share uzantılı bir ClientHello mesajı gönderir. Bu uzantı, istemcinin desteklediği anahtar değiş tokuşu şifreleme yöntemlerini listeler.
2. Sunucu, key_share uzantılı bir ServerHello mesajıyla yanıt verir. Bu uzantı, anahtar değişimi için kullanmak istediği şifreleme yöntemini içerir.
3. Sunucu, sunucu parametrelerini istemciye gönderir.
4. Hem sunucu hem de istemci kimlik doğrulama mesajlarını değiştirir.
5. Sunucu, istemciye, istemcinin daha sonra ClientHello iletisindeki pre_shared_key uzantısına dahil ederek gelecekteki el sıkışmaları için kullanabileceği bir PSK içeren bir NewSessionTicket iletisi gönderir.
6. İstemci ve sunucu artık şifrelenmiş uygulama verilerini değiş tokuş edebilir.
7. Gelecekteki bir anlaşmada, istemci sunucuya key_share ve pre_shared_key uzantılarıyla bir ClientHello mesajı gönderir. pre_shared_key uzantısı, NewTicketSession mesajında gönderilen bir PSK içerir.
8. Sunucu, pre_shared_key ve key_share uzantılarına sahip bir ServerHello mesajıyla yanıt verir. pre_shared_key uzantısı, sunucunun kullanmayı kabul ettiği şekilde PSK'yı içerir.
9. Sunucu, parametrelerini istemciye gönderir.
10. Sunucu ve istemci birbirlerine Bitti mesajları gönderir. Bu bağlantının güvenlik içeriği orijinal bağlantıya kriptografik olarak bağlı olduğundan, kimlik doğrulama aşamasını gerçekleştirmezler.
11. İstemci ve sunucu artık şifrelenmiş uygulama verilerini değiş tokuş edebilir.

TLS Record Protocol

Byte	+0	+1	+2	+3
0	Content type			
1..4	Version		Length	
5..n	*Payload*			
n..m	MAC			
m..p	Padding (block ciphers only)			