

# BSM 471-AĞ GÜVENLİĞİ

## SSL/TLS

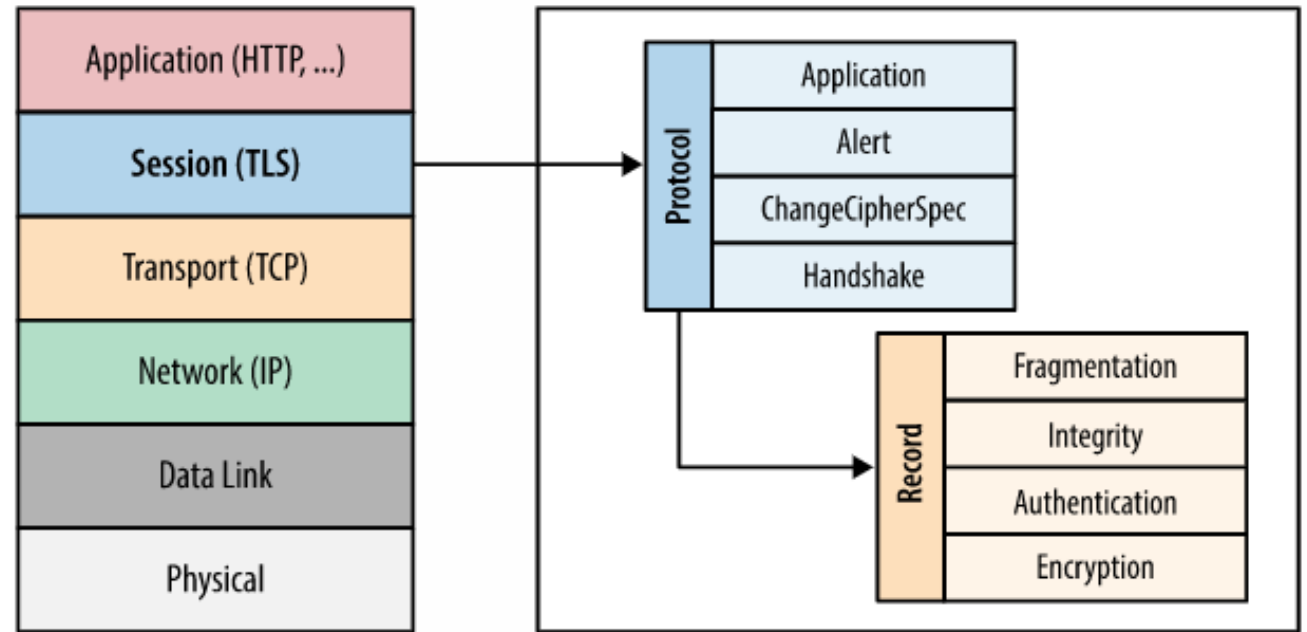
Dr. Öğr. Üyesi Musa BALTA  
Bilgisayar Mühendisliği Bölümü  
Bilgisayar ve Bilişim Bilimleri Fakültesi

# 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

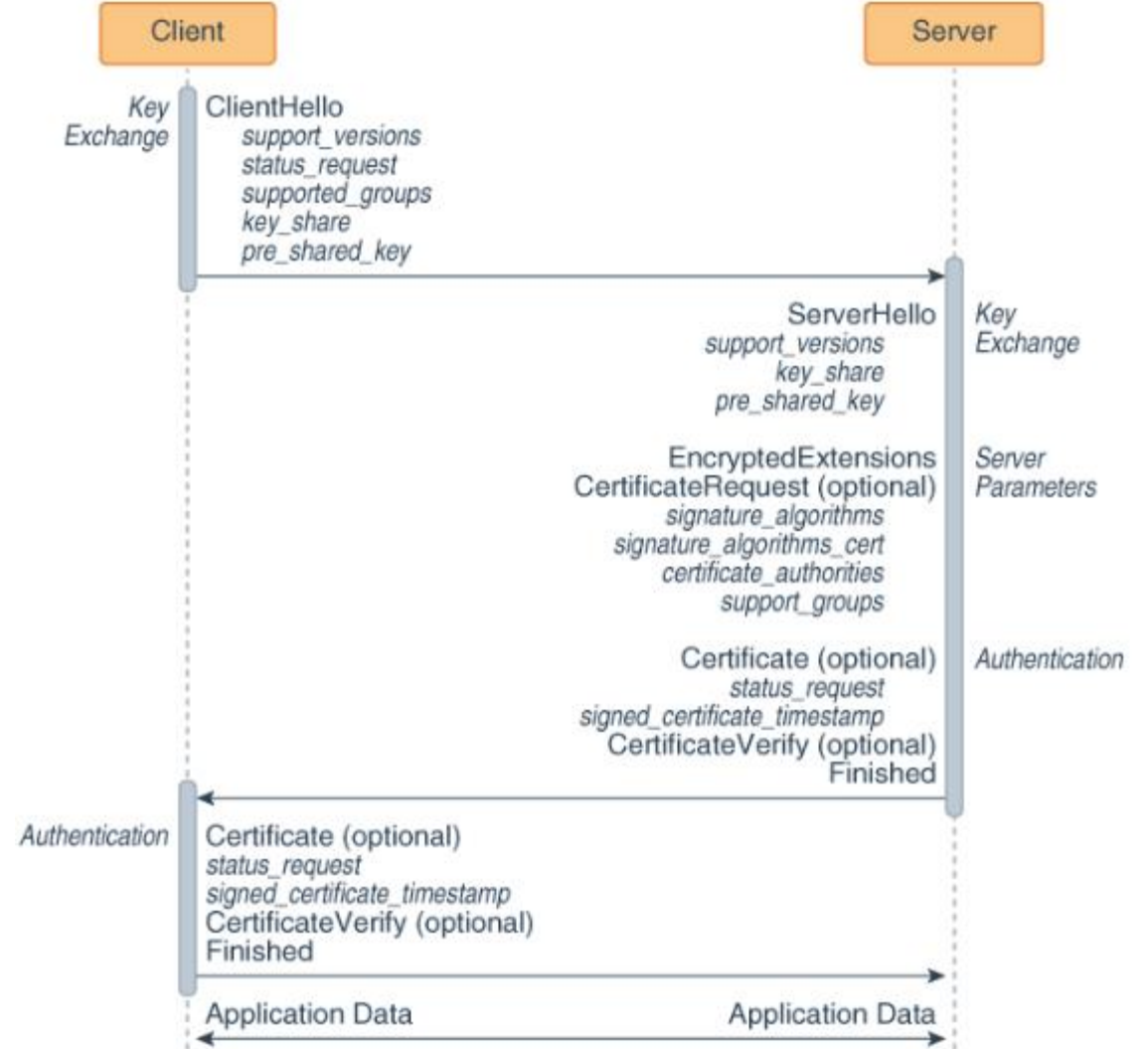


# TLS Çalışma Yapısı

- 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 işlemlere sahip **gizli anahtarlı şifrelemeyi** kullanır.

# 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 öge 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. ServerHello'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.

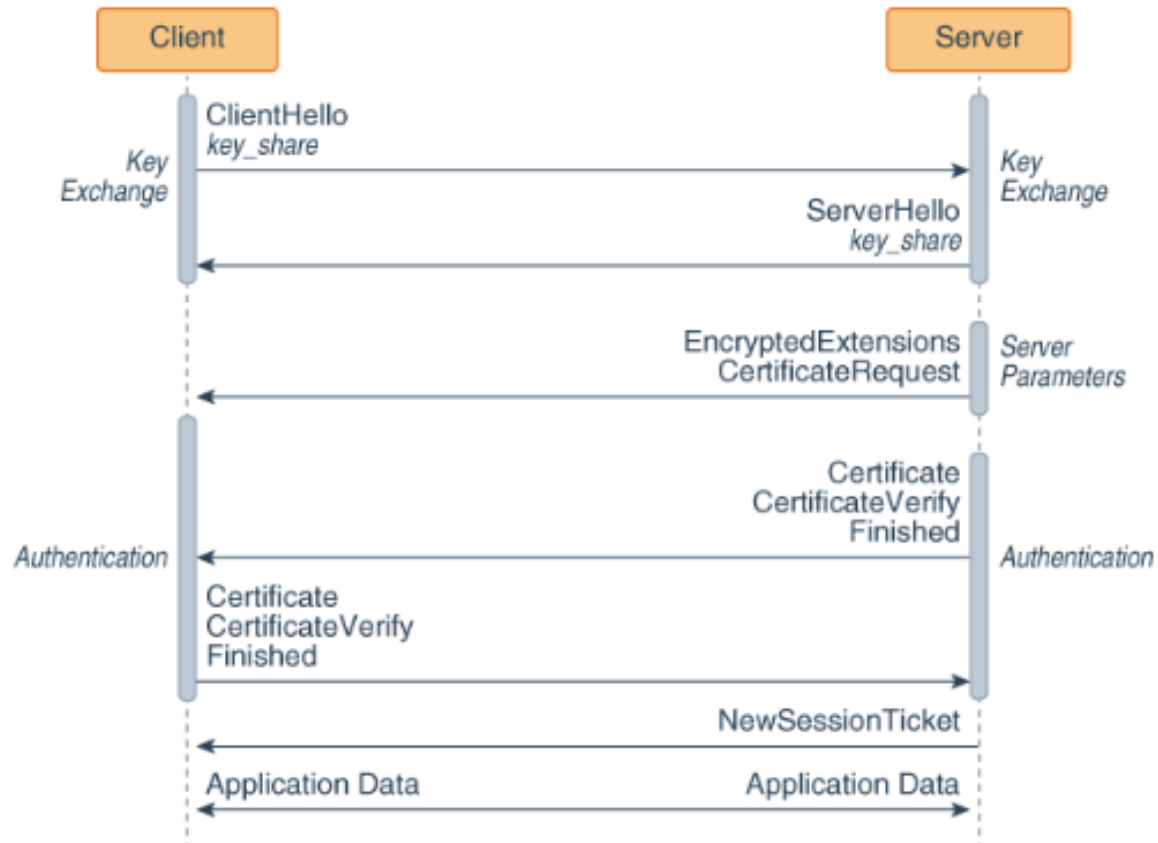
# Authentication

- 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.

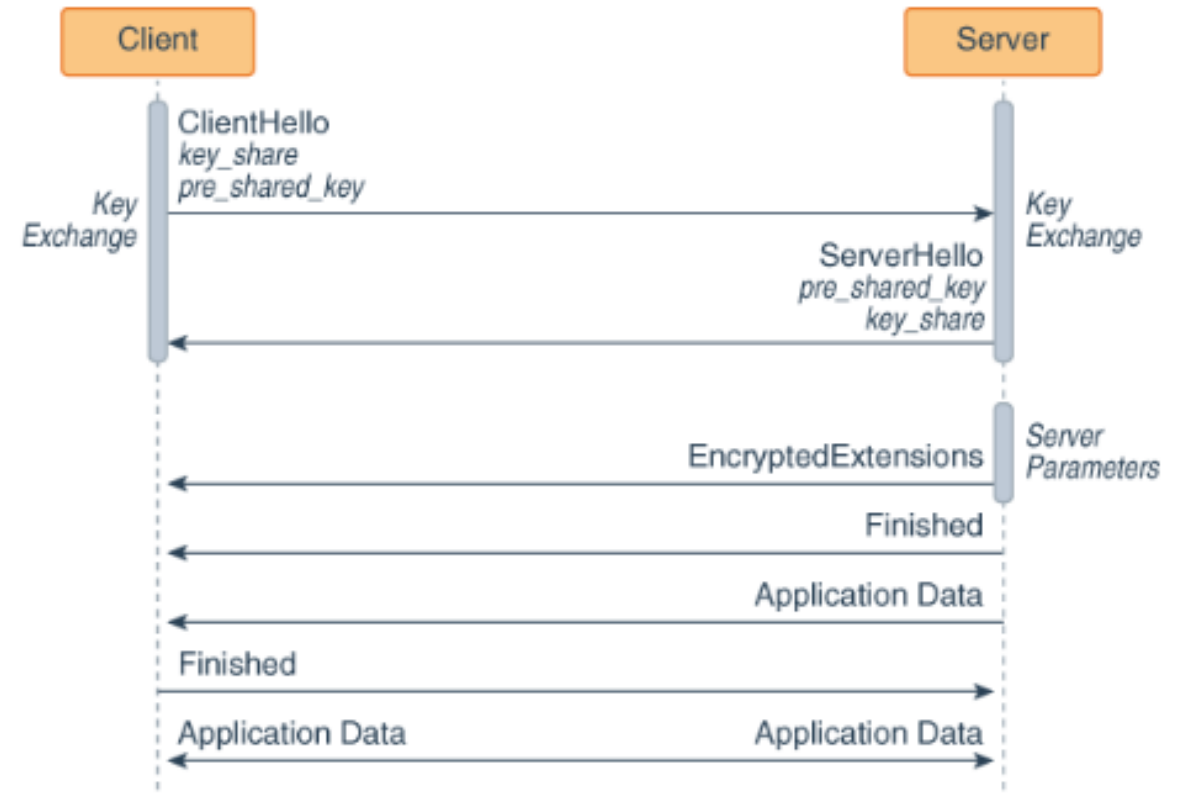
# Önceden Paylaşılan Anahtarla Oturum Sürdürme (Session Resumption with a Pre-Shared Key)

- 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 tokuş 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) |    |        |    |