



# TLS/SSL & Cipher Suites

Merkblatt 10



IT-Sicherheitsbeauftragter UZH

## Versionskontrolle

Version	Änderung	Autor	Datum
1.0	Erstellung	Kevin Willimann	02.07.2025
1.1	Anpassung Grammatik	Kevin Willimann	10.07.2025

# 1. Liste der Cypher Suites

Die aktuellen Empfehlungen können via folgender URL geprüft werden:

<https://ciphersuite.info/cs/>

## Recommended (Empfohlen)

- Alle „empfohlenen“ Chiffren sind per Definition „sichere“ Chiffren.
- Empfohlen bedeutet, dass diese Chiffren auch PFS (Perfect Forward Secrecy) unterstützen und Ihre erste Wahl sein sollten, wenn Sie das höchste Mass an Sicherheit wünschen.
- Es kann jedoch zu Kompatibilitätsproblemen mit älteren Clients kommen, die PFS-Chiffren nicht unterstützen.

## Secure (Sicher)

- Sichere Verschlüsselungen gelten als Stand der Technik, und wenn Sie Ihren Webserver absichern wollen, sollten Sie auf jeden Fall aus dieser Gruppe wählen.
- Nur sehr alte Betriebssysteme, Browser oder Anwendungen sind nicht in der Lage, sie zu verarbeiten.

## Weak (Schwach)

- Diese Chiffren sind veraltet und sollten deaktiviert werden, wenn Sie zum Beispiel einen neuen Server einrichten.
- Aktivieren Sie sie nur, wenn Sie einen speziellen Anwendungsfall haben, bei dem die Unterstützung für ältere Betriebssysteme, Browser oder Anwendungen erforderlich ist.

## Insecure (Unsicher)

- Diese Chiffren sind sehr alt und sollten unter keinen Umständen verwendet werden. Ihr Schutz kann heutzutage mit minimalem Aufwand gebrochen werden.

## 2. Was ist der Unterschied zwischen TLS und SSL?

TLS entstand aus einem früheren Verschlüsselungsprotokoll namens Secure Socket Layer (SSL), das von Netscape entwickelt wurde.

TLS Version 1.0 begann eigentlich mit der Entwicklung als SSL-Version 3.1, aber der Name des Protokolls wurde vor der Veröffentlichung geändert, um anzuzeigen, dass es nicht mehr mit Netscape assoziiert war.

Aufgrund dieser Vorgeschichte werden die Begriffe TLS und SSL manchmal als synonym verwendet.

## 3. Was ist der Unterschied zwischen TLS und HTTPS?

HTTPS ist eine Implementierung der TLS-Verschlüsselung zusätzlich zum HTTP-Protokoll, das von allen Webseiten sowie einigen anderen Webdiensten verwendet wird.

Jede Webseite, die HTTPS verwendet, verwendet daher auch die TLS-Verschlüsselung.

## Aktuelle und Veraltete SSL/TLS Versionen

Version	Erscheinungsjahr	Bemerkungen
<b>Veraltet</b>		
SSL 1.0	1994	
SSL 2.0	1995	seit März 2011 überholt
SSL 3.0	1996	seit Juni 2015 überholt
TLS 1.0	1999	seit März 2021 überholt
TLS 1.1	2006	seit März 2021 überholt
<b>Aktuell</b>		
TLS 1.2	2008	
TLS 1.3	2018	RFC 8446, enthält auch neue Anforderungen für TLS 1.2.

## Was gibt es sonst noch für Protokolle?

Protokoll	Status	Verwendung	Besonderheit
<b>SSL</b>	<b>Veraltet</b>	<b>Keine sichere Nutzung mehr</b>	<b>Nicht mehr empfohlen</b>
TLS	Aktuell	Webseiten, E-Mails, VPNs	Standard für sichere Kommunikation
DTLS	Aktuell	VoIP, Videokonferenzen, VPNs	Speziell für UDP
QUIC-TLS	Aktuell	HTTP/3, Streaming, Gaming	Schnellere Verbindungen

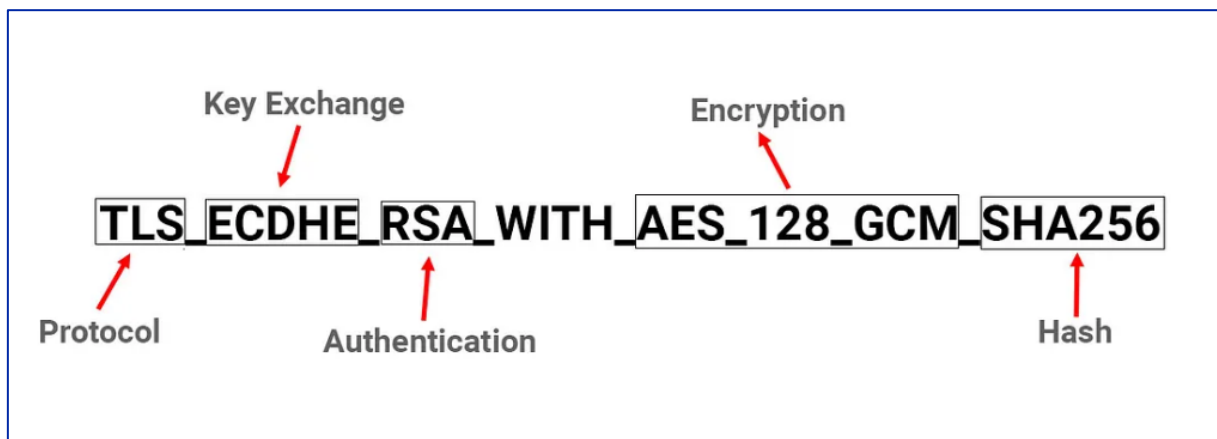
## 4. Was sind die Cipher Suites

Cipher Suites sind Sammlungen von kryptographischen Verfahren.

Diese bilden einen wichtigen Grundstein, wie Daten bei der Verwendung von HTTPS, FTPS, SMTP und anderen Netzwerkprotokollen sicher übertragen werden können. Über Cipher Suites wird festgelegt, welche Algorithmen ein Server für den Verkehr akzeptiert.

### Wie setzt sich die Suite zusammen?

Die Bezeichnung der Suite besteht aus folgenden Teilen:



### Protocol (Sicherheitsprotokoll)

Die erste Komponente einer Cipher Suite gibt das zugrunde liegende Sicherheitsprotokoll an.

#### 1. SSL (Secure Sockets Layer) [VERALTET]

- Vorgänger von TLS, aber unsicher (SSL 2.0 und 3.0 sind veraltet).
- Sollte nicht mehr verwendet werden.
- Beispiel: `SSL_RSA_WITH_3DES_EDE_CBC_SHA`

#### 2. TLS (Transport Layer Security)

- Aktuelles Standard-Protokoll für sichere Kommunikation im Internet
- Versionen: TLS 1.2, TLS 1.3 (ältere Versionen wie 1.0 und 1.1 sind unsicher)
- Beispiel: `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`

#### 3. DTLS (Datagram Transport Layer Security)

- Eine Variante von TLS für UDP-Verbindungen (z. B. VoIP, VPNs)
- Unterstützt ähnliche Cipher Suites wie TLS
- Beispiel: `DTLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`

#### 4. QUIC-TLS

- TLS-basierte Verschlüsselung für das QUIC-Protokoll (z. B. Google & HTTP/3)
- Beispiel: `TLS_AES_128_GCM_SHA256` (direkt in QUIC integriert, keine extra Bezeichnung)

# Key Exchange (Schlüsselaustausch)

Der Schlüsselaustausch bestimmt, wie die beiden Kommunikationspartner (Client & Server) geheime Schlüssel sicher austauschen.

## 1. RSA (Rivest-Shamir-Adleman) [VERALTET]

- Früher weit verbreitet, aber unsicher gegen Quantencomputer und Forward-Secrecy-Probleme.
- Beispiel: TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- Nachteil: Kein Perfect Forward Secrecy (PFS), wodurch vergangene Sitzungen entschlüsselt werden könnten.

## 2. DH (Diffie-Hellman) [VERALTET]

- Frühe Implementierung des sicheren Schlüsselaustauschs
- Beispiel: TLS\_DH\_RSA\_WITH\_AES\_256\_CBC\_SHA
- Nachteil: Feste Parameter, wodurch Sicherheitsrisiken entstehen.

## 3. DHE (Diffie-Hellman Ephemeral)

- Verbessert DH durch die Verwendung temporärer Schlüssel (Ephemeral-Modus).
- Beispiel: TLS\_DHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384
- Vorteil: Perfect Forward Secrecy (PFS)

## 4. ECDH (Elliptic Curve Diffie-Hellman)

- Bietet mehr Sicherheit bei kürzeren Schlüssellängen als klassisches DH.
- Beispiel: TLS\_ECDH\_RSA\_WITH\_AES\_128\_CBC\_SHA
- Nachteil: Kein Ephemeral-Modus → Kein PFS

## 5. ECDHE (Elliptic Curve Diffie-Hellman Ephemeral)

- Aktueller Standard für sichere TLS-Verbindungen
- Beispiel: TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384
- Vorteil: Bietet Perfect Forward Secrecy (PFS)
- Vorteil: Kürzere Schlüssel als DHE bei gleicher Sicherheit → Schnellere Berechnungen

## 6. PQ (Post-Quantum Cryptography) [ZUKUNFT]

- Entwickelt, um gegen Angriffe von Quantencomputern resistent zu sein.
- Beispiel: TLS\_HYBRID\_KYBER\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 (Hybrid-Ansätze in Entwicklung)

# Authentication (Authentifizierung)

## 1 RSA (Rivest-Shamir-Adleman) [VERALTET]

- Früher weit verbreitet, aber unsicher gegen Quantencomputer und Forward-Secrecy-Probleme
- Beispiel: TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- Nachteil: Kein Perfect Forward Secrecy (PFS), wodurch vergangene Sitzungen entschlüsselt werden könnten. Schwächen bei kleineren Schlüsselgrößen
- Empfohlene Verwendung: Nur mit Schlüssellängen von 2048 Bit oder mehr

## 2 DSA (Digital Signature Algorithm) [VERALTET]

- Verwendet für digitale Signaturen, aber unsicher bei kurzen Schlüssellängen
- Beispiel: TLS\_DSA\_WITH\_AES\_256\_CBC\_SHA
- Nachteil: Anfällig für Angriffe bei Schlüssellängen unter 2048 Bit. Zudem ineffizienter als modernere Alternativen wie ECDSA.
- Empfohlene Verwendung: Vermeiden, ausser in alten Systemen mit speziellen Anforderungen.

## 3 ECDSA (Elliptic Curve Digital Signature Algorithm)

- Verwendet elliptische Kurven, bietet hohe Sicherheit bei kurzen Schlüssellängen und schnellere Berechnungen.
- Beispiel: TLS\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256
- Vorteil: Höhere Sicherheit bei kürzeren Schlüssellängen als RSA
- Nachteil: Wird zunehmend von EdDSA abgelöst, aber immer noch sicher und weit verbreitet.

## 4 EdDSA (Edwards-curve Digital Signature Algorithm)

- Moderne Alternative zu ECDSA, bietet höhere Sicherheit und Performance bei kleinerem Schlüssel
- Beispiel: TLS\_ED25519\_WITH\_AES\_256\_GCM\_SHA384
- Vorteil: Robuster gegen Angriffe, schnelle Berechnungen, und bietet bessere Sicherheit
- Empfohlene Verwendung: Der bevorzugte Standard für digitale Signaturen in modernen Systemen

## 5 PSK (Pre-Shared Key) [EINGESCHRÄNKT VERWENDBAR]

- Verwendet einen vorab geteilten geheimen Schlüssel für Authentifizierung.
- Beispiel: TLS\_PSK\_WITH\_AES\_256\_GCM\_SHA384
- Nachteil: Weniger sicher als asymmetrische Verfahren, da der Schlüssel zwischen den Parteien geteilt werden muss.
- Empfohlene Verwendung: Nur für einfache Netzwerke oder Geräte mit begrenzten Ressourcen, nicht für sicherheitskritische Anwendungen.

## 6 GOST (russischer Standard) [EINGESCHRÄNKT VERWENDBAR]

- Russischer Standard für digitale Signaturen und Verschlüsselung
- Beispiel: TLS\_GOST\_WITH\_AES\_256\_CBC\_SHA
- Nachteil: Wird international nicht weit verbreitet und hat in westlichen Systemen keine breite Unterstützung.
- Empfohlene Verwendung: Nur in russischen oder spezialisierten Netzwerken, in denen GOST erforderlich ist.

# Encryption (Verschlüsselung)

## 1 3DES (Triple Data Encryption Standard) [VERALTET]

- Eine erweiterte Version des DES-Algorithmus, bei dem die Daten dreimal verschlüsselt werden
- Beispiel: TLS\_3DES\_EDE\_CBC\_SHA
- Nachteil: 3DES ist heutzutage als unsicher und ineffizient anerkannt, da es anfällig für Kollisionen und Angriffe ist (z. B. "Sweet32"-Angriff bei längeren Verbindungen)
- Empfohlene Verwendung: Sollte dringend vermieden werden, da es nicht mehr den modernen Sicherheitsanforderungen entspricht.

## 2 RC4 (Rivest Cipher 4) [VERALTET]

- Ein Stromverschlüsselungsverfahren, das früher in SSL/TLS verwendet wurde.
- Beispiel: TLS\_RC4\_WITH\_AES\_128\_CBC\_SHA
- Nachteil: RC4 hat mehrere bekannte Sicherheitslücken (z. B. Angriffe auf den Stromgenerator), die es in modernen Systemen ungeeignet machen.
- Empfohlene Verwendung: Nicht mehr verwenden. RC4 gilt als unsicher und sollte aus allen modernen Cipher Suites entfernt werden.

## 3 AES (Advanced Encryption Standard)

- Der Standard für symmetrische Verschlüsselung und weit verbreitet. AES bietet eine hohe Sicherheit und Effizienz bei der Verschlüsselung
- Beispiel: TLS\_AES\_128\_GCM\_SHA256
- Vorteil: Sehr sicher, schnell und flexibel in der Schlüssellänge (128, 192 und 256 Bit)
- Empfohlene Verwendung: AES-128 oder AES-256 wird in modernen TLS-Versionen bevorzugt.
- Nachteil: Bei Verwendung von AES-128 könnte es theoretisch eine geringere Sicherheit bieten als AES-256, jedoch ist AES-128 in der Praxis sehr sicher und effizient.

## 4 ChaCha20

- Ein symmetrisches Verschlüsselungsverfahren, das als Alternative zu AES entwickelt wurde. Es bietet hohe Sicherheit und Performance, insbesondere in mobilen Geräten und bei Hardware mit geringer Rechenleistung.
- Beispiel: TLS\_CHACHA20\_POLY1305\_SHA256
- Vorteil: Sehr sicher, schneller und robuster als AES in Umgebungen mit begrenzter Rechenleistung
- Empfohlene Verwendung: Wird in TLS 1.2 und 1.3 als Ersatz für AES in einigen Cipher Suites empfohlen.
- Nachteil: Geringere Verbreitung und weniger unterstützte Hardware im Vergleich zu AES, aber dafür weniger anfällig für bestimmte Angriffe.

## 5 IDEA (International Data Encryption Algorithm) [EINGESCHRÄNKT VERWENDBAR]

- Ein symmetrisches Verschlüsselungsverfahren, das in der Vergangenheit für seine Sicherheit und Effizienz bekannt war.
- Beispiel: TLS\_IDEA\_CBC\_SHA (wird in der Regel nur in alten Implementierungen gefunden)
- Nachteil: Weniger effizient als AES und wird in modernen Systemen kaum noch verwendet.
- Empfohlene Verwendung: Vermeiden, da AES und ChaCha20 sicherer und effizienter sind.

# Hash (Prüfziffer)

## 1 SHA-1 (Secure Hash Algorithm 1) [VERALTET]

- SHA-1 erzeugt einen 160-Bit-Hashwert und war früher weit verbreitet. Aufgrund von Sicherheitslücken, die durch Kollisionen ausgenutzt werden können, gilt SHA-1 heute als unsicher.
- Beispiel: TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA1
- Nachteil: Es sind Kollisionen (zwei unterschiedliche Daten mit demselben Hashwert) nachgewiesen, was den Algorithmus angreifbar macht.
- Empfohlene Verwendung: SHA-1 sollte nicht mehr verwendet werden. Es wird zunehmend aus modernen Systemen entfernt und durch sicherere Algorithmen wie SHA-256 ersetzt.

## 2 MD5 (Message Digest Algorithm 5) [VERALTET]

- MD5 war früher ein gängiger Hash-Algorithmus, der 128-Bit-Hashwerte erzeugt. Wie SHA-1 ist MD5 jedoch aufgrund von Sicherheitslücken nicht mehr sicher.
- Beispiel: TLS\_RSA\_WITH\_AES\_128\_CBC\_MD5
- Nachteil: Bekannt für Kollisionsangriffe, wodurch der Algorithmus für kryptografische Anwendungen nicht mehr empfohlen wird.
- Empfohlene Verwendung: MD5 sollte niemals mehr verwendet werden. Es ist aufgrund seiner Sicherheitslücken in allen modernen Systemen obsolet.

## 3 SHA-256 (Secure Hash Algorithm 256-bit)

- SHA-256 ist der Standard-Hash-Algorithmus in vielen modernen kryptografischen Anwendungen. Es erzeugt einen 256-Bit-Hashwert und ist resistent gegen Kollisionen (d. h. zwei unterschiedliche Eingaben, die denselben Hashwert erzeugen).
- Beispiel: TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- Vorteil: Sehr sicher und weit verbreitet, vor allem für Integritätsprüfungen und digitale Signaturen
- Empfohlene Verwendung: SHA-256 ist der bevorzugte Hash-Algorithmus für moderne Systeme, da er eine gute Balance aus Sicherheit und Performance bietet.

## 4 SHA-384 (Secure Hash Algorithm 384-bit)

- SHA-384 ist eine Erweiterung von SHA-256 und erzeugt einen längeren Hashwert von 384 Bit. Es bietet eine höhere Sicherheit und wird häufig in sicherheitskritischen Anwendungen verwendet.
- Beispiel: TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384
- Vorteil: Höhere Sicherheit als SHA-256, aber mit einem leichten Performanceverlust
- Empfohlene Verwendung: SHA-384 wird bevorzugt, wenn eine höhere Sicherheit erforderlich ist, z. B. bei besonders sensiblen Daten.